Al lettore

La realizzazione di un libro comporta costi variabili (carta, stampa, legatura) e costi fissi, cioè indipendenti dal numero di copie stampate (traduzione, preparazione degli originali, redazione, composizione, impaginazione). I fotocopiatori possono contenere il prezzo perché, oltre a non pagare i diritti d'autore, non hanno costi fissi.
Ogni fotocopia, d'altra parte, riducendo il numero di copie vendute dall'editore, aumenta l'incidenza dei costi fissi a copia e costringe l'editore ad aumentare il prezzo; questo naturalmente, fornisce un ulteriore incentivo a fotocopiare. Se questo circolo vizioso non verrà spezzato, arriveremo al punto in cui gli editori non avranno più convenienza economica a realizzare libri di testo per l'università.

In quel momento non ci saranno più neppure fotocopie.

L'editore

William Stallings

Crittografia e sicurezza delle reti

Seconda edizione

Edizione italiana a cura di
Luca Salgarelli

McGraw-Hill

Milano • New York • San Francisco • Washington D.C. • Auckland
Bogotá • Lisboa • London • Madrid • Mexico City • Montreal
New Delhi • San Juan • Singapore • Sydney • Tokyo • Toronto

McGraw-Hill
A Division of The McGraw-Hill Companies

Editor: Paolo Roncoroni
Consulenza scientifica: Achille Pattavina
Produzione: Donatella Giuliani
Revisione tecnica: Renato Cortinovis
Traduzione: Paolo Poli, Renato Cortinovis
Composizione: PMT sas, Monza
Grafica di copertina: G&G
Stampa: Gruppo Imprenta, Colturano (MI)

ISBN 88-386-6377-7
Printed in Italy
1234567890GIMLIL210987

Indice

Prefazione

"The tie, if I might suggest it, sir, a shade more tightly knotted. One aims at the perfect butterfly effect. If you will permit me—"
"What does it matter, Jeeves, at a time like this? Do you realize that Mr. Little's domestic happiness is hanging in the scale?"
"There is no time, sir, at which ties do not matter."
—Very Good, Jeeves! P. G. Wodehouse

Nell'era della connettività elettronica, dei virus e degli hacker, delle intercettazioni e delle frodi elettroniche, non esiste praticamente alcun ambito in cui la sicurezza non sia fondamentale. Vi sono due tendenze che rendono questo volume di fondamentale interesse. Innanzitutto la diffusione esponenziale dei computer e della loro interconnessione via rete hanno aumentato la dipendenza sia delle aziende sia degli individui dalle informazioni contenute e comunicate utilizzando questi sistemi. Questo, a sua volta, ha condotto a una più elevata consapevolezza della necessità di proteggere i dati e le risorse da qualsiasi violazione, di garantire l'autenticità dei dati e dei messaggi e di proteggere i sistemi dagli attacchi provenienti dalla rete. In secondo luogo le discipline della crittografia e della sicurezza delle reti sono maturate portando allo sviluppo di applicazioni pratiche e di agevole uso in grado di garantire la sicurezza della rete.

Obiettivi

Questo volume tratta i principi e le tecniche pratiche di crittografia e di sicurezza delle reti. Nelle prime due parti del volume vengono esplorati i principali problemi da risolvere nel campo della sicurezza delle reti, fornendo una guida alle tecnologie disponibili nel campo della crittografia e della sicurezza delle reti. La parte finale del volume tratta gli aspetti pratici della sicurezza delle reti: le applicazioni che sono state sviluppate e che sono attualmente utilizzate per garantire la sicurezza delle reti.

L'argomento, e pertanto questo intero volume, si basa su varie discipline. In particolare, è impossibile comprendere alcune delle tecniche trattate in questo volume senza una conoscenza di base della teoria dei numeri e alcuni risultati della teoria delle probabilità. Ciononostante, si è cercato di fare in modo che il volume fosse il più possibile completo. Il testo non presenta solo i risultati matematici di base ma fornisce al lettore una conoscenza intuitiva di questi risultati. Gli argomenti tecnici di supporto vengono introdotti nel momento in cui divengono necessari, consentendo di motivare le ragioni di tale introduzione. L'autore considera questo approccio preferibile rispetto alla scelta di presentare tutto il materiale matematico in un blocco teorico all'inizio del volume.

A chi si rivolge questo volume

Il volume si rivolge sia a un'utenza accademica che professionale. Come libro di testo è indicato per un corso universitario sulla crittografia e la sicurezza delle reti. Il volume è adatto anche come manuale di riferimento e risorsa di autoapprendimento.

Organizzazione del volume

Il volume è suddiviso in quattro parti.

- **Parte prima – Cifrature simmetriche:** presenta un esame dettagliato degli algoritmi di crittografia convenzionali e dei principi progettuali, comprendente una discussione sull'uso della crittografia convenzionale per la segretezza.
- **Parte seconda – Crittografia a chiave pubblica e funzioni hash:** presenta un esame dettagliato degli algoritmi di crittografia a chiave pubblica e dei relativi principi progettuali. Questa parte tratta anche l'utilizzo dei codici di autenticazione dei messaggi e delle funzioni hash, della firma digitale e dei certificati a chiave pubblica.
- **Parte terza – Applicazioni per la sicurezza delle reti:** descrive alcuni importanti strumenti e applicazioni per la sicurezza delle reti fra cui Kerberos, i certificati X.509v3, PGP, S/MIME, IPSec, SSL/TLS e SET.
- **Parte quarta – Sicurezza del sistema:** tratta le problematiche di sicurezza a livello di sistema, tra cui le minacce e le contromisure relative agli attacchi degli hacker e dei virus, l'uso dei firewall e i sistemi fidati.

Ciascuna parte inizia con un riepilogo dettagliato dei capitoli contenuti. Ciascun capitolo comprende svariati problemi da risolvere, domande di ripasso, un elenco di parole chiave, suggerimenti per letture di approfondimento e un elenco di siti Web consigliati.

Servizi Internet per i docenti e gli studenti

Questo volume è affiancato da siti Web che forniscono il supporto a studenti, docenti e professionisti. Il sito in inglese che si trova all'indirizzo `WilliamStallings.com/Crypto/Crypto4e.html` comprende oltre alle Appendici C, D, E, F, G, H citate nel corso del volume, utili link, una copia delle figure e delle tabelle in formato PDF (Adobe Acrobat) e presentazioni in PowerPoint. Il sito Computer Science Student Resource all'indirizzo `WilliamStallings.com/StudentSupport.html` fornisce ulteriori documenti, informazioni e link utili a studenti e professionisti.

All'indirizzo Web `www.ateneonline.it/stallings` è inoltre possibile accedere al sito Web in italiano dedicato al libro, contenente oltre alle Appendici A e B citate nel testo, appendici tecnico-didattiche, un ampio glossario, un elenco di sigle e la bibliografia.

Progetti per l'insegnamento della crittografia e della sicurezza delle reti

Per molti docenti, in un corso sulla crittografia e la sicurezza delle reti è importante assegnare agli studenti uno o più progetti che consentano loro di migliorare la comprensione dei concetti trattati nel testo applicandoli concretamente. Questo volume fornisce una grande quantità di spunti per progetti. L'appendice Progetti didattici sulla crittografia e la sicurezza delle reti, disponibile all'indirizzo `www.ateneonline.it/stallings`, include una guida sul modo in cui assegnare e strutturare i progetti sugli argomenti trattati nel testo.

- **Progetti di ricerca:** una serie di proposte di ricerca nelle quali si chiede agli studenti di ricercare informazioni su un particolare argomento e poi redigere una relazione.
- **Progetti di programmazione:** una serie di progetti che trattano un'ampia varietà di argomenti e che sono implementabili in qualsiasi linguaggio appropriato su qualsiasi piattaforma.
- **Esercizi di laboratorio:** una serie di progetti che coinvolgono la programmazione e la sperimentazione con i concetti esposti nel volume.
- **Esercizi di redazione:** suggerimenti per esercizi di redazione, organizzati per capitolo.
- **Letture e relazioni:** un elenco di documenti per ciascun capitolo che potranno essere consultati dallo studente che al termine dovrà redigere una relazione.

Le novità della quarta edizione americana

Nei tre anni trascorsi dalla terza edizione del volume, questo campo ha subito continui miglioramenti. In questa nuova edizione si è tentato di incorporare questi cambiamenti mantenendo nel contempo un'ampia copertura dell'intero argomento. Per iniziare questo processo di revisione, la terza edizione è stata ampiamente riveduta da vari docenti. Inoltre i singoli capitoli sono stati riveduti e corretti da numerosi professionisti del settore. In questo modo il testo è stato reso più chiaro e rigoroso e molte illustrazioni sono state migliorate. Inoltre sono stati aggiunti numerosi problemi "verificati sul campo".
Oltre a questi miglioramenti che avevano lo scopo di aumentare l'utilità didattica e la facilità d'uso, sono stati apportati sostanziali cambiamenti a tutto il volume. Ecco un breve elenco.

- **AES semplificato:** versione educativa semplificata di AES (Advanced Encryption Standard) che favorisce la comprensione dei concetti fondamentali di AES da parte degli studenti.

- **Whirlpool:** importante nuovo algoritmo hash basato sulla cifratura a blocchi simmetrica.
- **CMAC:** nuova modalità operativa per la cifratura a blocchi. CMAC (Cipher-based Message Authentication Code) fornisce l'autenticazione dei messaggi tramite l'impiego di una cifratura a blocchi simmetrica.
- **Infrastruttura a chiave pubblica (PKI):** argomento fondamentale che viene trattato in questa nuova edizione.
- **Attacchi DDoS (DoS Distribuiti):** gli attacchi DDoS sono divenuti sempre più significativi negli ultimi anni.
- **Criteri Comuni per la valutazione della sicurezza delle tecnologie dell'informazione (CC):** i cosiddetti Common Criteria sono divenuti il riferimento internazionale per esprimere i requisiti di sicurezza e per valutare prodotti e implementazioni.
- **Appendici online:** sei appendici disponibili sul sito Web `www.WilliamStallings.com` di supporto arricchiscono questo volume.

Molta altra documentazione contenuta nel libro è stata aggiornata e migliorata.

Ringraziamenti

Questa nuova edizione è stata riveduta e corretta da varie persone che hanno offerto generosamente il proprio tempo e la propria esperienza. Le seguenti persone hanno riveduto completamente o in parte il manoscritto: Danny Krizanc (Wesleyan University), Breno de Medeiros (Florida State University), Roger H. Brown (Rensselaer at Hartford), Cristina Nita-Rotarul (Purdue University) e Jimmy McGibney (Waterford Institute of Technology).
Vorrei ringraziare anche le numerose persone che hanno effettuato la revisione tecnica dettagliata dei singoli capitoli: Richard Outerbridge, Jorge Nakahara, Jeroen van de Graaf, Philip Moseley, Andre Correa, Brian Bowling, James Muir, Andrei Holt, Décio Luiz Gazzoni Filho, Lucas Ferriera, Dr. Kemal Bicakci, Routo Terada, Anton Stiglic, Valere Pryamikov e Yongge Wang.
Joan Daemen ha gentilmente revisionato il capitolo su AES. Vincent Rijmen ha revisionato il materiale relativo a Whirlpool. Edward F. Schaefer ha revisionato il materiale relativo a S/DES.
I problemi per la nuova versione sono stati forniti da: Joshua Brandon Golden (Rose-Hulman Institute of Technology), Kris Gaj (Gorge Mason University) e James Muir (University of Waterloo).
Sanjay Rao e Ruben Torres della Purdue University hanno sviluppato gli esercizi di laboratorio contenuti nel manuale per il docente.
Le seguenti persone hanno fornito i progetti contenuti nel manuale per il docente: Henning Schulzrinne (Columbia University), Cetin Kaya Koc (Oregon State University) e David Balenson (Trusted Information Systems e George Washington University).
Infine vorrei ringraziare le numerose persone responsabili della pubblicazione di questo volume, che hanno svolto come al solito il loro eccellente lavoro. Fra queste il personale di Prentice Hall, in particolare il responsabile della produzione Rose Kernan, il mio coresponsabile Sarah Parser e il mio nuovo editor Tracy Dunkelberger.

Capitolo 0

Guida alla lettura

Questo volume e il sito web che lo accompagna sviluppano numerosi argomenti. Questo capitolo ne presenta l'organizzazione generale.

0.1 Contenuti generali del volume

Dopo un capitolo introduttivo (Capitolo 1) il volume è organizzato in quattro parti.

Parte prima Cifrature simmetriche. Tratta la crittografia simmetrica, tra cui gli algoritmi più classici e quelli più moderni. Si pone l'accento sui due algoritmi più importanti: DES (Data Encryption Standard) e AES (Advanced Encryption Standard). Questa parte esamina anche l'autenticazione dei messaggi e la gestione della chiave.

Parte seconda Crittografia a chiave pubblica e funzioni hash. Tratta gli algoritmi a chiave pubblica, fra cui l'algoritmo RSA (Rivest-Shamir-Adelman) e quello a curva ellittica. Tratta inoltre le applicazioni a chiave pubblica fra cui le firme digitali e lo scambio di chiavi.

Parte terza Sicurezza di rete. Esamina l'uso degli algoritmi di crittografia e dei protocolli di sicurezza per garantire la sicurezza nelle reti e in Internet. Fra gli argomenti trattati vi sono l'autenticazione degli utenti, la posta elettronica, la sicurezza IP e la sicurezza nel Web.

Parte quarta Sicurezza di sistema. Riguarda i sistemi di sicurezza sviluppati per proteggere un computer da qualsiasi minaccia, fra cui gli hacker, i virus e i worm. Questa parte si occupa anche della tecnologia dei firewall.

Molti degli algoritmi crittografici e dei protocolli e delle applicazioni di sicurezza della rete descritti in questo volume sono stati specificati come standard. I più importanti di questi sono gli standard Internet, definiti in vari documenti RFC (Request for Comments) e FIPS (Federal Information Processing Standards) emessi dal NIST (National Institute of Standards and Technology). L'Appendice A disponibile online sul sito `www.ateneonline.it/`

stallings illustra il processo di definizione degli standard ed elenca gli standard citati in questo volume.

0.2 Struttura generale

Contenuti

I contenuti sono organizzati in tre ampie categorie.

- **Crittologia:** lo studio delle tecniche per garantire la segretezza e/o autenticità dell'informazione. I due settori principali della crittologia sono la crittografia, ovvero lo studio della progettazione di tali tecniche, e l'analisi crittografica, ovvero lo studio delle tecniche per contrastare i tentativi di ottenere informazioni segrete o di creare informazioni false che siano considerate autentiche.
- **Sicurezza di rete:** quest'area riguarda l'uso degli algoritmi di crittografia nei protocolli e nelle applicazioni di rete.
- **Sicurezza dei computer:** in questo volume si utilizza tale termine per fare riferimento alla sicurezza dei computer contro intrusi (per esempio hacker) e software doloso (per esempio virus). Solitamente il computer da rendere sicuro è connesso in rete, dalla quale proviene la maggior parte delle minacce.

Le prime due parti del volume si occupano di due approcci alla crittografia distinti: gli algoritmi di crittografia simmetrici e gli algoritmi di crittografia a chiave pubblica, o asimmetrici. Gli algoritmi simmetrici utilizzano un'unica chiave condivisa dalle due entità in comunicazione. Gli algoritmi a chiave pubblica utilizzano invece due chiavi: una chiave privata, conosciuta esclusivamente da un'entità, e una chiave pubblica disponibile alle altre entità.

Sequenza degli argomenti

Questo volume contiene molto materiale, tuttavia vi sono varie possibilità di utilizzo selettivo.

Per comprendere dettagliatamente gli argomenti trattati nelle prime due parti del volume, è necessario leggere i capitoli in sequenza. Nessun argomento trattato nella Parte Prima, ad esclusione dell'Advanced Encryption Standard (AES), richiede conoscenze matematiche approfondite. Per comprendere AES è necessaria invece una conoscenza elementare dei campi finiti, che a sua volta richiede conoscenze di base dei numeri primi e dell'aritmetica modulare. Il Capitolo 4 tratta queste conoscenze matematiche di base prima del loro impiego nel Capitolo 5 su AES. Se non si affronta il Capitolo 5, si può quindi tralasciare anche il Capitolo 4.

Il Capitolo 2 introduce alcuni concetti utili nei capitoli successivi della Parte Prima. Tuttavia il lettore esclusivamente interessato alla crittografia contemporanea può scorrere rapidamente questo capitolo. I due algoritmi di crittografia simmetrici più importanti sono DES e AES trattati, rispettivamente, nei Capitoli 3 e 5. Il Capitolo 6 presenta due altri

algoritmi interessanti, entrambi utilizzati commercialmente. Questo capitolo può essere tralasciato nel caso in cui questi algoritmi non dovessero interessare.

Per poter affrontare lo studio della Parte Seconda si richiedono conoscenze matematiche aggiuntive nella teoria dei numeri, trattata nel Capitolo 8. Il lettore che avesse trascurato i Capitoli 4 e 5 dovrebbe prima rivedere i paragrafi da 4.1 a 4.3.

I due algoritmi a chiave pubblica più diffusi e di impiego generale sono RSA e le curve ellittiche, con RSA che gode di una diffusione molto superiore. Il lettore potrebbe, per lo meno a una prima lettura, ignorare il Capitolo 10 relativo alla crittografia con curve ellittiche. Whirlpool e CMAC, nel Capitolo 12, sono di minore importanza.

Le Parti Terza e Quarta sono relativamente indipendenti tra di loro: possono essere lette in qualsiasi ordine. Entrambe richiedono una comprensione di base del materiale trattato nelle Parti Prima e Seconda.

0.3 Risorse su Internet e sul Web

In Internet sono disponibili varie risorse che supportano questo volume e che aiutano a mantenersi aggiornati in questo settore.

Siti Web per questo volume

Questo volume è affiancato da un sito Web in inglese, disponibile all'indirizzo:

WilliamStallings.com/Crypto/Crypto4e.html

Il sito comprende:

- **Useful Web sites:** collegamenti ipertestuali che rimandano ad altri interessanti siti Web, organizzati capitolo per capitolo; includono tutti i siti menzionati in questo volume.
- **Errata sheet:** un elenco aggiornato degli eventuali errori contenuti nel volume.
- **Figures:** tutte le figure di questo volume in formato PDF (Adobe Acrobat).
- **Tables:** tutte le tabelle di questo volume in formato PDF.
- **Slides:** un insieme di slide di PowerPoint organizzate capitolo per capitolo.
- **Cryptography and network security courses:** collegamenti ipertestuali a home page riguardanti corsi che si basano su questo testo; queste pagine possono fornire utili informazioni ai docenti per strutturare il proprio corso.

L'autore gestisce inoltre il sito Computer Science Student Resource all'indirizzo:

WilliamStallings.com/StudentSupport.html

Lo scopo di questo sito è quello di fornire documenti, informazioni e collegamenti ipertestuali utili a studenti e professionisti nel campo informatico. I collegamenti e i documenti sono organizzati in quattro categorie.

- **Math:** include un corso di ripasso sulla matematica di base, un'introduzione all'analisi delle code, un'introduzione ai sistemi di numerazione e collegamenti a numerosi siti che si occupano di matematica.
- **How-to:** consigli sulla soluzione dei problemi, sulla scrittura di report tecnici e sulla preparazione di presentazioni tecniche.
- **Research resources:** collegamenti ipertestuali che rimandano a importanti raccolte di articoli, rapporti tecnici e altri riferimenti bibliografici.
- **Miscellaneous:** una varietà di documenti e collegamenti ipertestuali utili.

All'indirizzo `www.ateneonline.it/stallings` è possibile accedere al sito Web in italiano dedicato al libro, contenente le Appendici A e B citate nel testo, due appendici tecnico-didattiche, un glossario, un elenco di sigle e la bibliografia relativa ai riferimenti riportati fra parentesi quadre nei capitoli; tale bibliografia integra quella presentata alla fine di ogni capitolo e ricalca quella riportata alla fine del volume.

Altri siti Web

Esistono molti siti Web che forniscono informazioni relative agli argomenti più rilevanti trattati in questo volume. Nei paragrafi "Letture e siti Web consigliati" di ciascun capitolo si trovano vari puntatori ai siti Web più specifici. Poiché l'indirizzo dei siti Web tende a cambiare con una certa frequenza, si è cercato di evitare di introdurli nel volume. Il collegamento appropriato a qualsiasi sito Web elencato in questo testo è disponibile nel sito Web associato al volume. Altri collegamenti ipertestuali non menzionati in questo volume verranno aggiunti nel corso del tempo direttamente al sito Web.

I gruppi di discussione USENET

Esistono vari gruppi di discussione dedicati a specifici aspetti della sicurezza o della crittografia. Come per tutti i gruppi USENET, vi è un elevato livello di "rumore" nei messaggi ma vale la pena di consultarli per trovare informazioni utili. I più importanti sono i seguenti.

- **sci.crypt.research:** il miglior gruppo sulla crittografia. Si tratta di un gruppo di discussione moderato che tratta argomenti di ricerca; i messaggi devono avere una stretta relazione con gli aspetti tecnici della crittografia.
- **sci.crypt:** discussioni generali sulla crittografia e argomenti correlati.
- **sci.crypt.random-numbers:** discussioni sull'importanza dei numeri casuali nella crittografia.
- **alt.security:** discussioni generali su argomenti della sicurezza.
- **comp.security.misc:** discussioni su argomenti relativi alla sicurezza dei computer.
- **comp.security.firewalls:** discussioni sui prodotti e le tecnologie firewall.
- **comp.security.announce:** informazioni e annunci da parte del CERT.
- **comp.risks:** discussioni sui rischi per il pubblico derivanti dai computer e dagli altri utenti.
- **comp.virus:** discussioni moderate riguardanti i virus.

Capitolo 1

Introduzione

Concetti essenziali

- L'architettura di sicurezza OSI (Open Systems Interconnection) fornisce il contesto sistematico per definire attacchi, meccanismi e servizi di sicurezza.
- Gli **attacchi alla sicurezza** sono classificati come passivi, tra i quali la lettura non autorizzata di messaggi, di file e l'analisi del traffico, e attivi, tra i quali la modifica di messaggi, file e attacchi DoS (Denial of Service).
- Per **meccanismo di sicurezza** si intende qualsiasi processo (o dispositivo che incorpora tale processo) progettato per prevenire, rilevare o riparare i danni provocati da un attacco alla sicurezza. Esempi di tali meccanismi sono gli algoritmi di crittografia, le firme digitali e i protocolli di autenticazione.
- Fra i **servizi di sicurezza** si citano l'autenticazione, il controllo degli accessi, la segretezza dei dati, la non-ripudiazione e la disponibilità.

I requisiti di **sicurezza delle informazioni** di un'organizzazione hanno subito due importanti cambiamenti nei decenni passati. Prima della diffusione dei sistemi di elaborazione delle informazioni, la sicurezza delle informazioni veniva garantita principalmente con mezzi fisici e amministrativi. Un esempio dei primi: i documenti più riservati venivano riposti in robusti armadi metallici con chiusura a combinazione. Un esempio dei secondi sono le procedure di assunzione del personale.

Con l'introduzione dei computer, divenne evidente la necessità di utilizzare strumenti automatizzati per la protezione dei file e di altre informazioni memorizzate. Questa esigenza è particolarmente sentita per i sistemi condivisi, come i sistemi time-sharing, e ancora di più per i sistemi accessibili tramite la rete telefonica pubblica, una rete dati o Internet. Gli strumenti destinati alla protezione dei dati e alla difesa dagli hacker venivano genericamente indicati con il termine **sicurezza dei computer**.

Il secondo grande cambiamento che ha coinvolto la sicurezza è stato l'introduzione dei sistemi distribuiti, l'uso delle reti e di apparecchiature di comunicazione per il trasferimen-

to dei dati fra terminali utente e computer centrale o fra computer. Diviene necessario adottare misure di sicurezza della rete per proteggere i dati durante la trasmissione. In realtà il termine **sicurezza di rete** è in qualche modo fuorviante, in quanto i sistemi di elaborazione dati di qualsiasi organizzazione commerciale, governativa o accademica sono collegati tramite un insieme di reti interconnesse chiamato internet[1], e quindi viene utilizzato il termine **sicurezza della internet**.

Non esistono confini esatti fra queste due forme di sicurezza. Per esempio, uno degli attacchi più noti ai sistemi informatici è rappresentato dai virus. Un virus può essere introdotto in un sistema fisicamente, per esempio tramite un dischetto, o anche via rete. In entrambi i casi, una volta che il virus si troverà in un computer, sarà necessario impiegare degli strumenti di sicurezza in grado di rilevare il virus, rimuoverlo e riparare i danni da esso provocati.

Questo volume si occupa principalmente di sicurezza di una rete internet, ovvero delle misure per scoraggiare, prevenire, rilevare e correggere le violazioni alla sicurezza che riguardano la trasmissione delle informazioni. Si tratta però di un'affermazione estremamente generica che tratta un'ampia gamma di possibilità. Per dare un'idea dei campi trattati in questo volume, si considerino i seguenti esempi di violazione della sicurezza.

1. L'utente A trasmette un file all'utente B. Il file contiene informazioni riservate (per esempio le registrazioni degli stipendi) che non devono essere assolutamente divulgate. L'utente C, non autorizzato a leggere tale file, può monitorare la trasmissione e catturare una copia del file.
2. Un amministratore della rete, D, trasmette un messaggio a un computer, E, posto sotto il suo controllo. Il messaggio indica al computer E di aggiornare un file di autorizzazioni per includere l'identità di alcuni nuovi utenti cui è stato assegnato l'accesso a tale computer. L'utente F intercetta il messaggio, altera il suo contenuto aggiungendo o cancellando delle voci e poi inoltra il messaggio a E che accetta il messaggio come se provenisse dall'amministratore D e aggiorna quindi il file delle autorizzazioni.
3. Invece di intercettare un messaggio, l'utente F costruisce un proprio messaggio specificando le voci desiderate e lo trasmette al computer E fingendosi l'amministratore D. Il computer E accetta il messaggio come se provenisse da D e aggiorna il proprio file delle autorizzazioni.
4. Un dipendente viene licenziato senza preavviso. Il responsabile del personale invia un messaggio a un server per eliminare il suo account. Quando l'annullamento viene eseguito, il server dovrà inviare invia una nota sul file dei dipendenti per confermare l'operazione. Ma il dipendente intercetta il messaggio di annullamento e lo ritarda il tempo sufficiente per prelevare dal server varie informazioni riservate. Alla fine il messaggio viene inoltrato, il comando viene eseguito e la conferma inviata. Questa operazione potrebbe passare inosservata per un lungo periodo di tempo.

[1] In questo volume si usa la parola "internet" (con la lettera "i" minuscola) per far riferimento a qualsiasi insieme di reti interconnesse. Una intranet aziendale è un esempio di una internet. La rete Internet (con la lettera "I" maiuscola) può essere uno degli elementi utilizzati da un'organizzazione per costruire la propria internet.

5. Un cliente invia al suo agente di borsa un messaggio chiedendogli di eseguire determinate transazioni. Successivamente gli investimenti richiesti perdono valore e il cliente nega di avere inviato il messaggio.

Questo breve elenco non esaurisce certamente tutte le possibili violazioni alla sicurezza ma illustra il tipo di problemi che si devono affrontare nel campo della sicurezza delle reti. La sicurezza dell'interconnessione fra reti è un argomento affascinante e complesso. Ecco alcuni dei motivi.

1. La sicurezza delle comunicazioni nelle reti non è un argomento semplice come potrebbe sembrare a prima vista. I requisiti sembrano essere piuttosto semplici, infatti è possibile indicare i principali con una singola parola: segretezza, autenticazione, non ripudiabilità, integrità. Ma i meccanismi utilizzati per rispondere a questi requisiti possono essere piuttosto complessi e la loro comprensione può richiedere ragionamenti molto sottili.
2. Nello sviluppo di un determinato meccanismo o algoritmo di sicurezza, occorre anche considerare i potenziali attacchi contro questi elementi. In molti casi un attacco sferrato con successo è dovuto al fatto che si è osservato il problema da un punto di vista completamente diverso e si è sfruttato un punto debole non considerato nel meccanismo di sicurezza.
3. Per quanto si è detto nel punto 2, le procedure utilizzate per fornire determinati servizi sono spesso tutt'altro che intuitive: la necessità di certe soluzioni particolarmente elaborate non risulta affatto ovvia dalla specifica di un particolare requisito. Le misure adottate appaiono sensate solo quando vengono considerate le varie contromisure necessarie.
4. Dopo aver progettato i vari meccanismi di sicurezza, è necessario decidere dove utilizzarli sia in termini di posizionamento fisico (il punto della rete in cui implementare determinati meccanismi di sicurezza) sia in senso logico (il livello o i livelli di un'architettura come TCP/IP – Transmission Control Protocol/Internet Protocol - in cui devono essere situati tali meccanismi).
5. Normalmente i meccanismi di sicurezza richiedono più del semplice impiego di un determinato algoritmo o protocollo. Essi richiedono che i partecipanti siano in possesso di certe informazioni segrete (per esempio una chiave di crittografia) e questo solleva ulteriori problemi relativi alla creazione, distribuzione e protezione di tali informazioni segrete. Inoltre si devono utilizzare protocolli di comunicazione il cui comportamento può complicare lo sviluppo dei meccanismi di sicurezza. Per esempio, se il corretto funzionamento dei meccanismi di sicurezza richiedesse la definizione di limiti temporali al tempo di transito del messaggio dal mittente al destinatario, qualsiasi protocollo o rete che introducesse dei ritardi variabili e imprevedibili potrebbe vanificare l'impiego di questi limiti temporali.

È necessario quindi considerare molti aspetti. Questo capitolo fornisce una panoramica generale degli argomenti che determinano la struttura della parte rimanente del volume. Si inizia con una discussione generale dei servizi e dei meccanismi di sicurezza di rete e dei tipi di attacchi per cui sono stati realizzati. Successivamente si sviluppa un modello globale generale nel quale poter inquadrare i vari servizi e meccanismi.

1.1 Tendenze nella sicurezza

Nel 1994 l'Internet Architecture Board (IAB) ha pubblicato un documento intitolato "Security in the Internet Architecture" (RFC 1636). Questo documento esprime il consenso generale sul fatto che Internet richiede migliori e ulteriori funzioni di sicurezza e identifica alcune aree chiave per i meccanismi di sicurezza. Fra queste vi è la necessità di proteggere l'infrastruttura di rete da spionaggio e da interventi non autorizzati, e di garantire la sicurezza del traffico utente tramite meccanismi di autenticazione e cittografia.

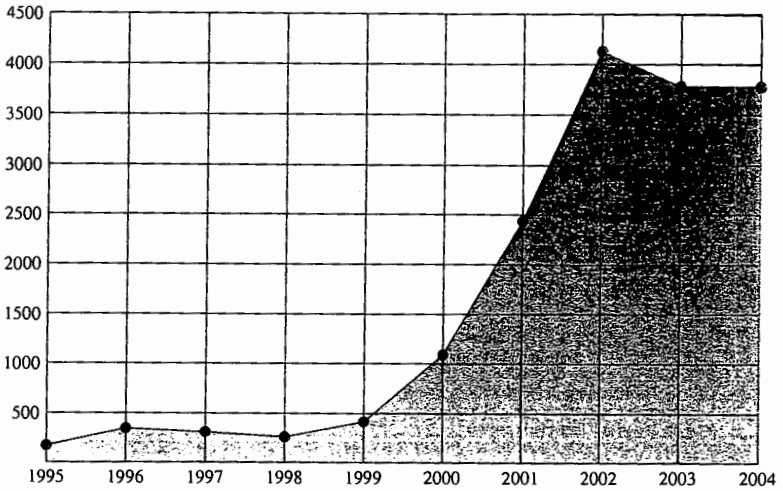
Queste preoccupazioni sono ben fondate. Come conferma, si considerino i dati riportati dal Centro di Coordinamento del Computer Emergency Response Team (CERT). La Figura 1.1a illustra la tendenza nel numero di lacune di sicurezza legate a Internet comunicate al CERT in un periodo decennale. Queste comprendono lacune di sicurezza nei sistemi operativi dei computer (per esempio Windows, Linux), nei router Internet e in altri dispositivi di rete. La Figura 1.1b illustra il numero di incidenti legati a problemi di sicurezza riportati al CERT: essi comprendono attacchi Denial of Service (DoS), falsificazioni di indirizzi IP (meccanismo tramite il quale gli intrusi creano pacchetti con indirizzi IP falsi per sfruttare le applicazioni che utilizzano l'autenticazione basata sugli indirizzi IP) e varie forme di spionaggio con le quali gli hacker vengono a conoscenza delle informazioni trasmesse in rete tra le quali informazioni di accesso ai sistemi e contenuti di database.

Con il passare del tempo gli attacchi a Internet e ai sistemi a essa connessi sono divenuti sempre più sofisticati mentre le competenze e conoscenze necessarie per sferrare tali attacchi sono diminuite (Figura 1.2). Gli attacchi sono divenuti sempre più automatizzati e possono quindi creare danni superiori.

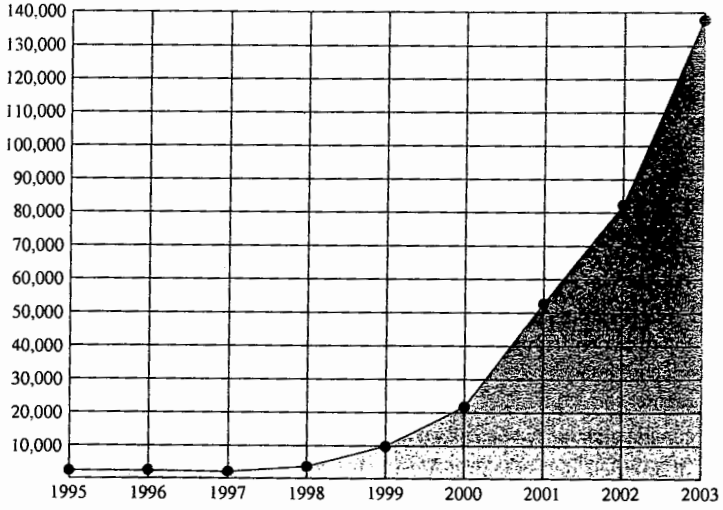
Questa espansione nel numero e nella complessità degli attacchi coincide con l'aumento dell'utilizzazione di Internet e della complessità dei protocolli, delle applicazioni e di Internet stessa. Le infrastrutture critiche impostano sempre più la propria operatività su Internet. Gli utenti privati si affidano sempre più alla sicurezza di Internet, della posta elettronica, del Web e di applicazioni basate sul Web. Per questi motivi si rende necessaria un'ampia gamma di tecnologie per contrastare questa crescente minaccia. Al livello base, gli algoritmi di crittografia per la segretezza e l'autenticazione acquisiscono un'importanza crescente. I progettisti devono inoltre focalizzarsi sui protocolli Internet e sulle lacune di sicurezza dei sistemi operativi e delle applicazioni Internet. Questo volume affronta tutti questi argomenti.

1.2 L'architettura di sicurezza del modello OSI

Per valutare in modo efficace le esigenze di sicurezza di un'organizzazione e per poter scegliere fra i vari prodotti e le varie strategie di sicurezza, il responsabile della sicurezza deve poter impiegare un metodo sistematico per definire i requisiti di sicurezza e per caratterizzare gli approcci che soddisfano tali requisiti. Si tratta di un compito già difficile in un ambiente centralizzato di elaborazione dati, che si complica ulteriormente con l'adozione di reti locali e geografiche.

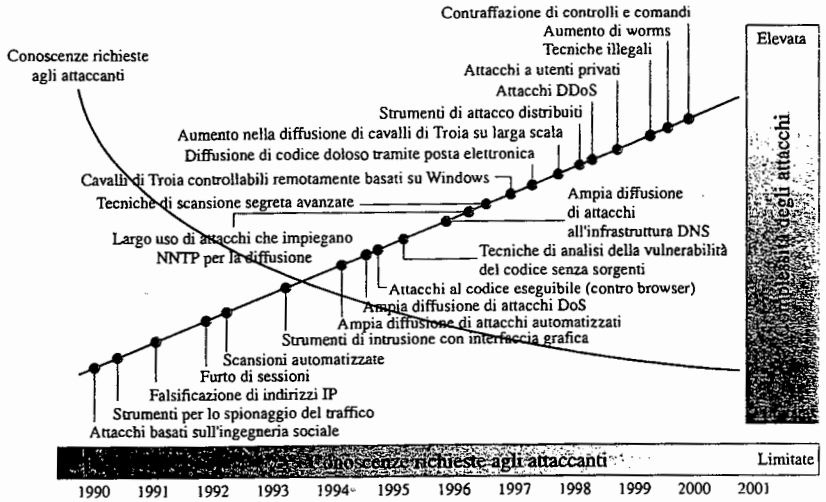


(a) Lacune di sicurezza riportate.



(b) Incidenti riportati.

Figura 1.1 Statistiche del CERT.



Fonte: CERT

Figura 1.2 Tendenza nella complessità degli attacchi e nelle conoscenze richieste agli attaccanti.

La "raccomandazione" X.800 dell'ITU-T[2], *Security Architecture for OSI*, definisce questo approccio sistematico[3]. L'architettura di sicurezza OSI è utile per organizzare le operazioni necessarie a garantire la sicurezza. Inoltre, poiché questa architettura è uno standard internazionale, i produttori di computer e di sistemi di comunicazione hanno sviluppato degli elementi di sicurezza per i propri prodotti e servizi facendo riferimento a questa definizione strutturata dei servizi e dei meccanismi.

L'architettura di sicurezza OSI fornisce un'utile panoramica, anche se astratta, di molti concetti trattati in questo volume. L'architettura di sicurezza OSI si concentra sui servizi, i meccanismi e gli attacchi alla sicurezza. Questi possono essere sinteticamente definiti nel modo seguente:

- **Attacco alla sicurezza:** qualsiasi azione che compromette la sicurezza delle informazioni di proprietà dell'organizzazione.
- **Meccanismo di sicurezza:** processo (o dispositivo che incorpora tale processo) progettato per rilevare, prevenire o riparare i danni prodotti da un attacco alla sicurezza.

[2] Il settore Telecommunication Standardization Sector (ITU-T) dell'International Telecommunication Union (ITU) agenzia specializzata delle Nazioni Unite, sviluppa standard (chiamati "raccomandazioni") relativi alle telecomunicazioni e all'interconnessione di sistemi aperti.

[3] L'architettura di sicurezza OSI è stata sviluppata nel contesto del modello OSI, descritto nell'Appendice H. Tuttavia, per quanto riguarda questo capitolo, la conoscenza del modello OSI non è richiesta.

- **Servizio di sicurezza:** servizio che migliora la sicurezza dei sistemi di elaborazione e trasmissione delle informazioni di un'organizzazione. I servizi proteggeono dagli attacchi alla sicurezza e sfruttano uno o più meccanismi di sicurezza.

Si deve notare che, nella letteratura scientifica, i termini *minaccia* e *attacco* sono frequentemente utilizzati sostanzialmente come sinonimi. La Tabella 1.1 fornisce le definizioni tratte dal documento RFC 2828, *Internet Security Glossary*.

Tabella 1.1 Minacce e attacchi (RFC 2828).

Minaccia Potenziale violazione alla sicurezza dettota da una circostanzo, una copacità, un'azione o un evento che potrebbe violare la sicurezza e provocare danni. Pertanto una minaccio è un pericolo potenziale che potrebbe sfruttare un punto debole.
Attacco Assalto alla sicurezzo di un sistema ovvero tentativo deliberato (specialmente nel senso del metodo o della tecnica) di eludere i servizi di sicurezza e di violare la politica di sicurezza di un sistema.

1.3 Attacchi alla sicurezza

Un modo utile per classificare gli attacchi alla sicurezza, utilizzato sia in X.800 che nel documento RFC 2828, è quello di catalogarli in *attacchi passivi* o in *attacchi attivi*. Un attacco passivo tenta di rilevare o di utilizzare le informazioni del sistema ma non agisce sulle sue risorse. Un attacco attivo tenta di alterare le risorse del sistema o comunque di alterarne il funzionamento.

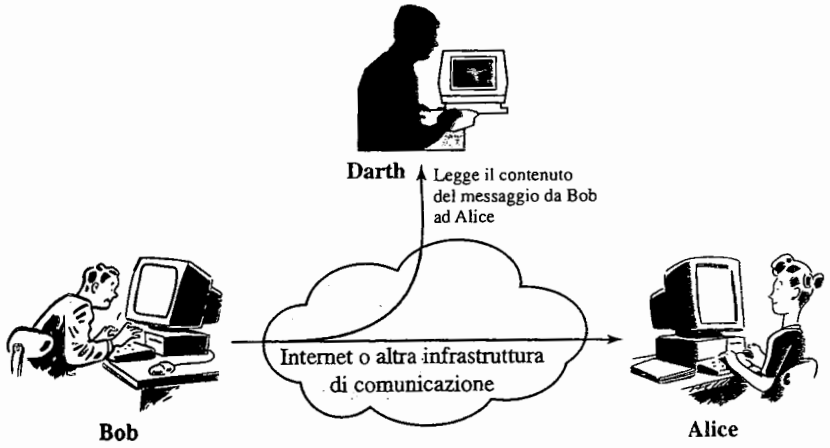
Attacchi passivi

Gli attacchi passivi si preoccupano di intercettare o monitorare le trasmissioni. L'obiettivo è quello di carpire le informazioni che vengono trasmesse. Due dei tipi di attacchi passivi sono l'intercettazione del contenuto dei messaggi e l'analisi del traffico.

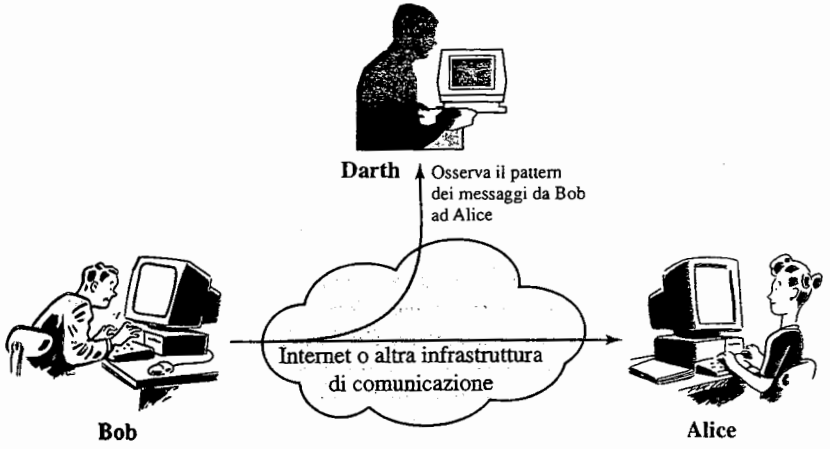
È facile comprendere l'**intercettazione del contenuto dei messaggi** (Figura 1.3a). Una conversazione telefonica, un messaggio di posta elettronica o un file possono contenere informazioni delicate o confidenziali. L'obiettivo è evitare che estranei possano carpire il contenuto di queste trasmissioni.

Il secondo tipo di attacco passivo, l'**analisi del traffico**, è più subdolo (Figura 1.3b). Si supponga di poter mascherare il contenuto dei messaggi o di un altro tipo di traffico in modo che anche se il messaggio stesso venisse intercettato, non fosse comunque possibile estrarre le informazioni in esso contenute. La tecnica solitamente utilizzata per mascherare i contenuti è la crittografia. Ma anche se si utilizzasse la crittografia, un estraneo potrebbe comunque individuare certe informazioni quali la posizione e l'identità degli host che comunicano e la frequenza e la lunghezza dei messaggi scambiati. Queste informazioni possono comunque essere utili per scoprire la natura delle comunicazioni che si stanno svolgendo.

Gli attacchi passivi sono molto difficili da rilevare poiché non comportano alcuna alterazione dei dati. In genere i messaggi vengono inviati e ricevuti in modo apparentemente normale, senza che il mittente e il destinatario possano rendersi conto che un estraneo ha



(a) Spionaggio del contenuto dei messaggi



(b) Analisi del traffico

Figura 1.3 Attacchi passivi.

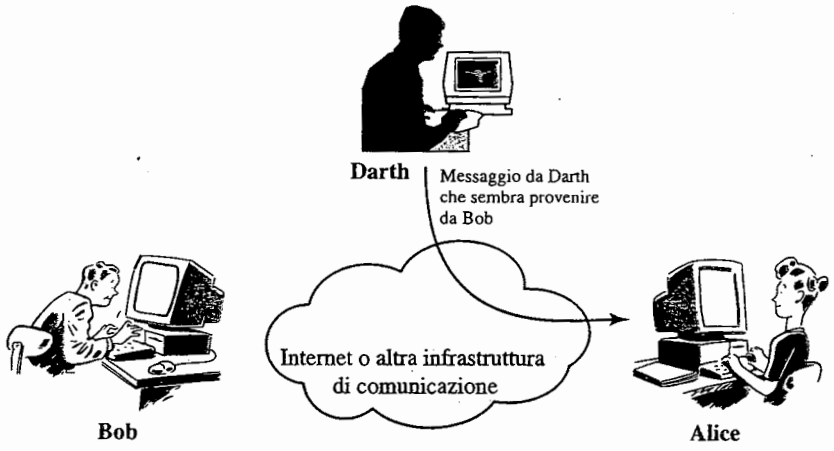
letto i messaggi o ha osservato il pattern del traffico. Tuttavia è possibile impedire il successo di questi attacchi, solitamente utilizzando la crittografia. Pertanto la difesa contro gli attacchi passivi riguarda più la prevenzione che la rilevazione.

Attacchi attivi

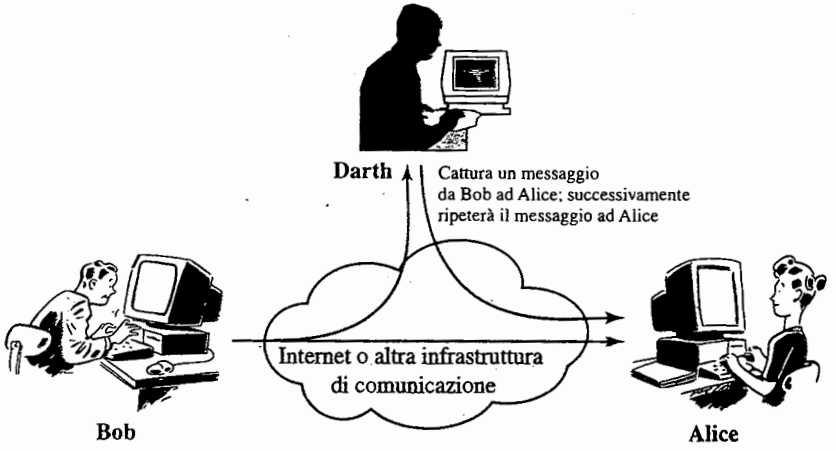
Gli attacchi attivi prevedono una modifica del flusso dei dati o la creazione di un falso flusso e possono essere suddivisi in quattro categorie: mascheramento, ripetizione, modifica dei messaggi e denial-of-service.

Si ha un **mascheramento** quando una determinata entità finge di essere un'altra entità (Figura 1.4a). Un attacco a mascheramento comprende in genere una delle altre forme di attacco attivo. Per esempio, le sequenze di autenticazione possono essere intercettate e riprodotte dopo che si è completata una sequenza valida di autenticazione, consentendo quindi a un'entità autorizzata ma con pochi privilegi di ottenere dei privilegi aggiuntivi, mascherandosi da entità dotata di tali privilegi.

La **ripetizione** prevede la cattura passiva di un'unità dati e la sua successiva ritrasmissione per produrre un effetto non autorizzato (Figura 1.4b).



(a) Mascheramento

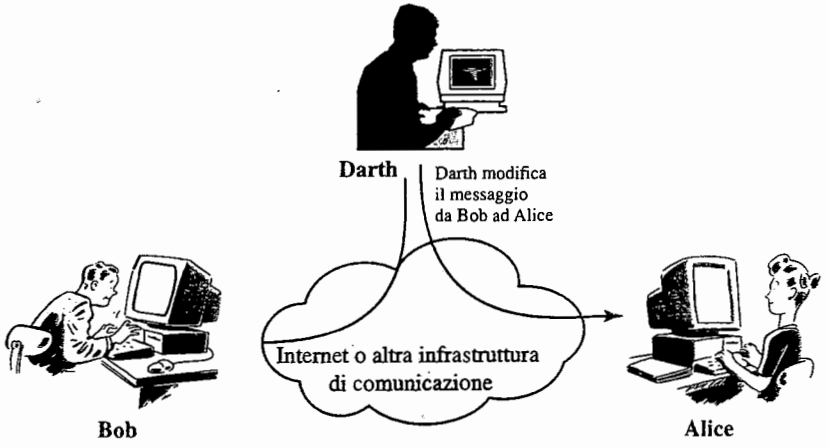


(b) Replay

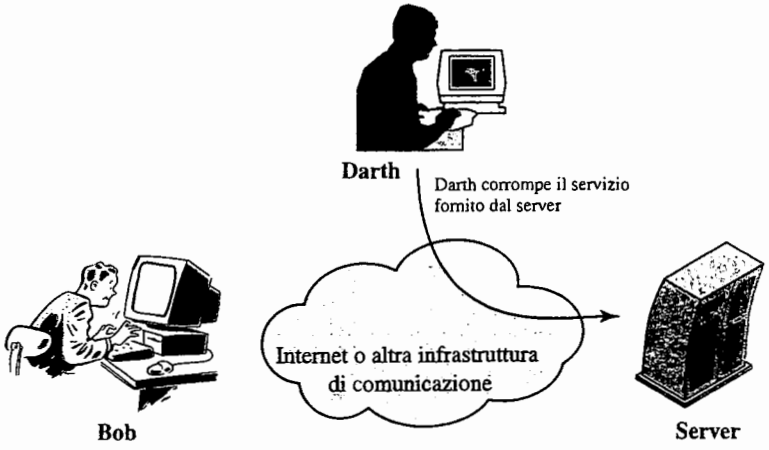
Figura 1.4 Attacchi attivi (parte 1 di 2).

Con **modifica dei messaggi** si intende semplicemente l'alterazione di un messaggio legittimo o il ritardo o il riordino di messaggi per produrre effetti non autorizzati (Figura 1.4c). Per esempio, un messaggio con significato "consentire a Mario Rossi di leggere il file segreto *conti*" può essere modificato come "consentire a Bruno Bianchi di leggere il file segreto *conti*".

Un attacco **denial-of-service** impedisce il normale utilizzo o la gestione di un sistema di comunicazione (Figura 1.4d). Questo attacco può avere un determinato bersaglio; per



(c) Modifica dei messaggi



(d) Denial of service

Figura 1.4 Attacchi attivi (parte 2 di 2).

esempio un'entità potrebbe sopprimere tutti i messaggi diretti a una determinata destinazione (come il servizio di auditing della sicurezza). Un'altra forma di attacco denial-of-service può riguardare un'intera rete, che potrebbe essere disattivata o sovraccaricata di messaggi degradandone le prestazioni.

Gli attacchi attivi hanno caratteristiche opposte rispetto agli attacchi passivi. Gli attacchi passivi sono difficili da rilevare, ma esistono misure per prevenire il loro successo. Al contrario è piuttosto difficile prevenire in modo assoluto gli attacchi attivi, per l'enorme varietà di potenziali lacune di sicurezza a livello fisico, logico e di rete. In questo caso l'obiettivo è dunque quello di rilevarli e recuperare qualsiasi alterazione o ritardo da essi provocato. Poiché il rilevamento ha un effetto deterrente, può contribuire anche alla prevenzione.

1.4 Servizi di sicurezza

X.800 definisce un servizio di sicurezza come un servizio, fornito da un particolare livello dei protocolli dei sistemi di comunicazione aperti, che garantisce la sicurezza dei sistemi o del trasferimento dati. Una definizione più chiara si trova forse nel documento RFC 2828: un servizio di elaborazione o comunicazione fornito da un sistema per offrire un determinato livello di protezione delle risorse del sistema; i servizi di sicurezza implementano le strategie di sicurezza e sono implementate dai meccanismi di sicurezza.

X.800 suddivide questi servizi in cinque categorie e 14 servizi specifici (vedere la Tabella 1.2). Segue la descrizione di ciascuno di essi.[4]

Tabella 1.2 Servizi di sicurezza (X.800).

AUTENTICAZIONE La garanzia che l'entità comunicante è chi sostiene di essere.
Autenticazione dell'entità peer Garantisce l'identità delle entità connesse in una connessione logica.
Autenticazione dell'origine dei dati Garantisce la provenienza dei dati ricevuti in un trasferimento in modalità non connessa.
CONTROLLO DEGLI ACCESSI Impedisce ogni utilizzo non autorizzato di una risorsa (in protica questo servizio controlla chi può avere accesso a una risorsa, in quali condizioni può avvenire l'accesso e che cosa sono autorizzati a fare coloro che accedono alla risorsa).
SEGRETEZZA DEI DATI La protezione dei dati da qualsiasi accesso non autorizzato.

(segue)

[4] Non vi è un accordo generale su molti dei termini utilizzati nei testi sulla sicurezza. Per esempio, il termine *integrità* viene talvolta utilizzato per far riferimento a tutti gli aspetti della sicurezza delle informazioni. Il termine *autenticazione* viene talvolta utilizzato per far riferimento sia alla verifica dell'identità che alle varie funzioni elencate sotto la voce integrità nel seguente elenco. Questo volume utilizza la terminologia prevista nei documenti X.800 e RFC 2828.

Tabella 1.2 Servizi di sicurezza (X.800). *(continua)*

Segretezza in modalità connessa La protezione di tutti i trasferiti in una connessione.
Segretezza in modalità non connessa La protezione di tutti i dati contenuti in un singolo blocco di dati (pacchetto).
Segretezza selettiva a campi La segretezza di campi selezionati all'interno dei dati utente su una connessione o in un singolo blocco dati.
Segretezza del traffico La protezione delle informazioni che potrebbero essere ottenute dall'analisi del flusso del traffico.
INTEGRITÀ DEI DATI La garanzia che i dati ricevuti siano esattamente quelli inviati dall'entità autorizzata, senza essere stati modificati in alcun modo (modifica, inserimento, cancellazione e ripetizione).
Integrità in modalità connessa con ripristino Garantisce l'integrità di tutti i dati utente in una connessione e rileva ogni modifica, inserimento, cancellazione o ripetizione di qualsiasi dato all'interno di una sequenza dati; se necessaria effettua tentativi di ripristino.
Integrità in modalità connessa senza ripristino Come sopra ma con il solo rilevamento, senza ripristino.
Integrità selettiva a campi in modalità conessa Garantisce l'integrità di determinati campi dei dati utente di un blocco dati trasferito su una connessione consentendo di rilevare se sono stati modificati, inseriti, cancellati o riprodotti.
Integrità in modalità non connessa Garantisce l'integrità di un singolo blocco dati in modalità non connessa rilevando eventuali modifiche ai dati. Inoltre può fornire una forma limitata di rilevamento delle ripetizioni.
Integrità selettiva a campi in modalità non connessa Garantisce l'integrità di determinati campi di un blocco dati in modalità non connessa rilevando se i campi sono stati modificati.
NON RIPUDIABILITÀ Protegge dall'eventualità che una delle entità coinvolte in una comunicazione neghi di aver partecipato, in toto o in parte, a detta comunicazione.
Non ripudiabilità, origine Prova che il messaggio è stato inviato dal mittente specificato.
Non ripudiabilità, destinazione Prova che il messaggio è stato ricevuto dal destinatario indicato.

Autenticazione

Il servizio di autenticazione cerca di garantire l'autenticità di una comunicazione. Nel caso di un unico messaggio, per esempio un segnale di allarme, la funzione del servizio di autenticazione è quella di assicurare al destinatario che il messaggio proviene realmente dalla sorgente indicata. Nel caso di un'interazione più prolungata, per esempio la connessione di un terminale a un host, vengono coinvolti due aspetti. Innanzitutto, al momento dell'attivazione della connessione, il servizio garantisce che le due entità siano autentiche, ovvero che ciascuna delle due entità sia effettivamente chi sostiene di essere. In secondo luogo il servizio deve garantire che la connessione non abbia interferenze evitando che un terzo possa fingersi una delle due parti legittime per ricevere o trasmettere le comunicazioni senza autorizzazione.

Lo standard X.800 definisce due specifici servizi di autenticazione.

- **Autenticazione dell'entità peer:** garantisce l'identità di un'entità peer nel contesto di un'associazione. Viene impiegata all'attivazione di una connessione o, talvolta, durante la fase di trasferimento. Tenta di garantire che un'entità non tenti di fingersi un'altra o non ripeta in modo illegittimo una connessione precedente.
- **Autenticazione dell'origine dei dati:** garantisce l'origine di un'unità di dati. Non protegge però contro la duplicazione o la modifica delle unità di dati. Questo servizio supporta applicazioni quali la posta elettronica, dove le interazioni dirette fra le entità in comunicazione sono scorrelate l'una dall'altra nel tempo.

Controllo degli accessi

Nel contesto della sicurezza di rete, il controllo degli accessi è la capacità di limitare e controllare l'accesso agli host e alle applicazioni via rete. Per ottenere ciò, le entità che tentano di ottenere l'accesso devono essere identificate (o autenticate) in modo da consentire la personalizzazione dei diritti di accesso.

Segretezza dei dati

Per segretezza dei dati si intendono la segretezza e la protezione dei dati trasmessi dagli attacchi passivi. Con riferimento al contenuto di una trasmissione dati, si possono identificare vari livelli di protezione. Il servizio più ampio protegge tutti i dati trasmessi fra due utenti in un determinato periodo di tempo. Per esempio, quando viene attivata una connessione TCP fra due sistemi, questo tipo di protezione impedisce l'intercettazione di dati utente trasmessi durante tale connessione. Si possono definire anche forme più rigide di questo servizio, tra cui la protezione di un singolo messaggio o di singoli campi di un messaggio. Questi livelli così specifici sono meno utili rispetto all'approccio più ampio e possono anche essere più complessi e costosi da implementare.

L'altro aspetto della segretezza è la protezione del flusso di traffico dall'analisi. In pratica un estraneo non deve essere in grado di rilevare l'origine, la destinazione, la frequenza, la lunghezza o altre caratteristiche del traffico in un sistema di comunicazione.

Integrità dei dati

Come nel caso della segretezza, anche l'integrità può essere applicata a un flusso di messaggi, a un singolo messaggio o a determinati campi di un messaggio. Anche in questo caso, l'approccio più utile e diretto è quello di proteggere l'intero flusso di dati.

Un servizio di integrità orientato alla connessione, avendo a che fare con un flusso di messaggi, garantisce che i messaggi vengano ricevuti così come sono stati inviati, senza duplicazioni, inserimenti, modifiche, variazioni d'ordine o ripetizioni. Questo servizio pro-

tegge anche dalla distruzione dei dati. Pertanto il servizio di integrità dei dati orientato alla connessione riguarda sia le modifiche nel flusso dei messaggi che gli attacchi Denial-of-Service. Al contrario un servizio di integrità in modalità non connessa, ovvero che tratta individualmente ciascun messaggio senza riferimento a un contesto più ampio, fornisce in genere solamente la protezione da eventuali modifiche al messaggio.

È possibile distinguere fra un servizio con o senza ripristino. Poiché il servizio di integrità fa riferimento agli attacchi attivi, si è più interessati al rilevamento che alla prevenzione. Se viene rilevata una violazione dell'integrità, il servizio può semplicemente evidenziare tale violazione; per ripristinare la situazione si dovrà ricorrere a qualche altro elemento software o a un intervento umano. Alternativamente esistono meccanismi in grado di ripristinare la perdita di integrità dei dati come si vedrà più avanti. L'incorporazione di meccanismi di ripristino automatico è in generale la soluzione più interessante.

Non ripudiabilità
La non ripudiabilità impedisce che il mittente o il destinatario possano negare di aver trasmesso o ricevuto un messaggio. Pertanto, quando viene inviato un messaggio, colui che lo riceve potrà dimostrare che il messaggio è stato in effetti inviato dal mittente indicato. Analogamente, quando il messaggio viene ricevuto, il mittente potrà dimostrare che il messaggio è stato in effetti ricevuto dal destinatario previsto.

Servizio di disponibilità
Sia X.800 che RFC 2828 definiscono la disponibilità come la proprietà di un sistema o di una sua risorsa, di essere accessibile e utilizzabile su richiesta da parte di un'entità autorizzata in base alle specifiche prestazionali del sistema (ovvero un sistema è disponibile se fornisce i servizi previsti in base alle specifiche quando richiesti dagli utenti). Vari attacchi provocano la perdita o la riduzione della disponibilità. Alcuni di questi attacchi possono essere affrontati con contromisure automatizzate, come l'autenticazione e la crittografia, mentre altri richiedono un intervento fisico per prevenire o ripristinare la situazione in seguito alla perdita di disponibilità degli elementi di un sistema distribuito.

X.800 tratta la disponibilità come una proprietà dei vari servizi di sicurezza. Tuttavia ha senso parlare in particolare di un "servizio di disponibilità". Un servizio di disponibilità protegge un sistema garantendone la disponibilità. Questo servizio riguarda i problemi di sicurezza sollevati dagli attacchi Denial-of-Service. Esso dipende dalla gestione e dal controllo appropriati delle risorse di sistema e quindi dal servizio di controllo degli accessi e da altri servizi di sicurezza.

1.5 Meccanismi di sicurezza

La Tabella 1.3 elenca i meccanismi di sicurezza definiti in X.800. Come si può vedere, i meccanismi si suddividono fra quelli implementati in un determinato livello di protocollo e quelli che non sono specifici di un determinato livello di protocollo o servizio di sicurezza. Questi meccanismi verranno trattati successivamente nelle sezioni appropriate di questo volume ma si vuole commentare già a questo livello la definizione di "cifratura". X.800

Tabella 1.3 Meccanismi di sicurezza (X.800).

MECCANISMI DI SICUREZZA SPECIFICI Possono essere incorporati nei livelli di protocollo appropriati, in modo da fornire alcuni dei servizi di sicurezza OSI.
Crittografia L'uso di algoritmi matematici per trasformare i dati in una forma illeggibile. La trasformazione e il successivo ripristino dei dati dipende da un algoritmo e da zero o più chiavi di crittografia.
Firma digitale Dati oggiunti o trasformazioni crittografiche dei dati che consentono al destinatario di dimostrare l'origine e l'integrità dei dati e di proteggersi da ogni alterazione delle informazioni (anche da parte del destinatario stesso).
Controllo degli accessi Una varietà di meccanismi che garantiscono il rispetto dei diritti di accesso alle risorse.
Integrità dei dati Una varietà di meccanismi utilizzati per garantire l'integrità di un'unità dati o di un flusso di dati.
Scambio di autenticazione Un meccanismo che ha lo scopo di garantire l'identità di un'entità tramite lo scambio di informazioni.
Tecniche di riempimento del traffico L'inserimento di bit all'interno del flusso dati per complicare ulteriormente il tentativo di analisi del traffico.
Controllo dell'instradamento Consente di scegliere percorsi fisici sicuri per determinati dati e di cambiare percorso ai dati, specialmente quando si sospetta una violazione alla sicurezza.
Autenticazione L'uso di una terza parte fidata per garantire determinate proprietà di uno scambio di dati.
MECCANISMI DI SICUREZZA PERVASIVI Meccanismi non specifici di un determinato servizio di sicurezza o livello di protocollo OSI.
Funzionalità fidata Ovvero percepita come corretta rispetto a determinati criteri (per esempio in base a una politica di sicurezza).
Etichetta di sicurezza Il contrassegno di una risorsa (che può essere un'unità dati) che nomina o designa i suoi attributi di sicurezza.
Rilevamento di eventi Rilevamento degli eventi importanti per la sicurezza.
Audit trail Dati raccolti e potenzialmente utilizzati per facilitare gli audit della sicurezza, ovvero un esame indipendente delle registrazioni e delle attività del sistema.
Ripristino della sicurezza Gestisce le richieste da meccanismi quali la gestione degli eventi e le funzioni di gestione, svolgendo azioni di ripristino.

distingue fra meccanismi di cifratura reversibili e irreversibili. Un meccanismo di cifratura reversibile è un algoritmo di crittografia che consente di crittografare e successivamente di decrittografare i dati. I meccanismi di cifratura irreversibili comprendono gli algoritmi hash e i codici di autenticazione dei messaggi che vengono utilizzati nelle applicazioni di firma digitale e di autenticazione dei messaggi.

La Tabella 1.4, basata su una delle tabelle di X.800, indica la relazione esistente fra i servizi di sicurezza e i meccanismi di sicurezza.

1.6 Un modello per la sicurezza di rete

La Figura 1.5 mostra un modello che cattura, in termini molto generali i concetti trattati in questo volume. Un messaggio deve essere trasferito dal mittente al destinatario attraversando una determinata rete internet. I due protagonisti di questa transazione devono cooperare per poter eseguire il trasferimento. Viene stabilito un canale logico per il trasferimento delle informazioni definendo un percorso attraverso l'internet dall'origine alla destinazione e tramite l'uso cooperativo dei protocolli di comunicazione (come TCP/IP) da parte dei due protagonisti.

Gli aspetti di sicurezza entrano in gioco quando è necessario o desiderabile proteggere la trasmissione delle informazioni da chiunque possa rappresentare una minaccia alla segretezza, all'autenticità e così via. Tutte le tecniche utilizzate per garantire la sicurezza hanno due componenti.

- Una trasformazione di sicurezza delle informazioni trasmesse. Fra le varie possibilità vi sono la crittografia del messaggio, che codifica il messaggio in modo che risulti illeggibile da parte di un estraneo e l'aggiunta di codice basato sul contenuto del messaggio, che può poi essere utilizzato per verificare l'identità del mittente.
- Un'informazione segreta condivisa dai due protagonisti e, si spera, sconosciuta agli estranei. Un esempio è la chiave di crittografia utilizzata per codificare il messaggio prima della trasmissione e per decodificarlo dopo la ricezione.[5]

Per garantire la sicurezza della trasmissione può essere necessario ricorrere a una terza parte fidata. Per esempio, questa terza parte può essere responsabile della distribuzione riservata delle informazioni segrete ai due protagonisti. La "terza parte" potrebbe anche essere necessaria per arbitrare le dispute fra i due protagonisti riguardanti l'autenticità della trasmissione di un messaggio.

Questo modello mostra che nella progettazione di un servizio di sicurezza vi sono quattro compiti fondamentali.

1. Progettazione di un algoritmo per l'esecuzione della trasformazione di sicurezza. L'algoritmo deve essere tale da impedire a un estraneo di conoscere il contenuto del messaggio.
2. Generazione delle informazioni segrete da utilizzare con l'algoritmo.
3. Sviluppo dei metodi per la distribuzione e la condivisione dell'informazione segreta.

[5] La Parte seconda di questo volume tratta una forma di crittografia nota come crittografia a chiave pubblica, in cui solo una delle due parti deve essere in possesso dell'informazione segreta.

Tabella 1.4 Relazioni fra i servizi e i meccanismi di sicurezza.

	Meccanismo							
Servizio	Cifratura	Firma digitale	Controllo degli accessi	Integrità dei dati	Scambio di dati di autenticazione	Tecniche di riempimento del traffico	Controllo dell'instradamento	Autenticazione
Autenticazione dell'entità peer	S	S			S			
Autenticazione dell'origine dei dati	S	S						
Controllo degli accessi			S					
Segretezza	S					-	S	
Segretezza del flusso del traffico	S					S	S	
Integrità dei dati	S	S		S				
Non ripudiabilità		S		S				S
Disponibilità				S	S			

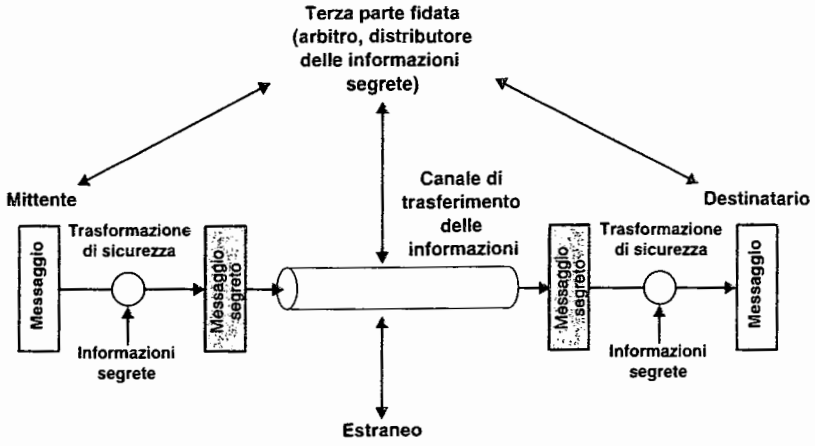


Figura 1.5 Modello della sicurezza di rete.

4. Specifica del protocollo impiegato dai due protagonisti che utilizzano l'algoritmo di sicurezza e dell'informazione segreta per ottenere un determinato servizio di sicurezza.

Le parti prima, seconda e terza di questo volume sono dedicate ai meccanismi e ai servizi di sicurezza che rientrano nel modello rappresentato nella Figura 1.5. Il volume affronta anche altre situazioni riguardanti la sicurezza che non rientrano esattamente in questo modello. Un modello generale che comprende anche queste aggiunte è rappresentato nella Figura 1.6 che riflette la preoccupazione di proteggere un sistema informativo da qualsiasi accesso indesiderato. La maggior parte dei lettori è probabilmente a conoscenza dell'esistenza di hacker che tentano di violare i sistemi accessibili attraverso le reti. L'hacker può essere una persona che, senza alcuna intenzione malevola, trae semplicemente soddisfazione dalla violazione di un computer. Ma l'intruso può anche essere un dipendente insoddisfatto che vuole creare danni alla sua azienda o un criminale che cerca di sfruttare le

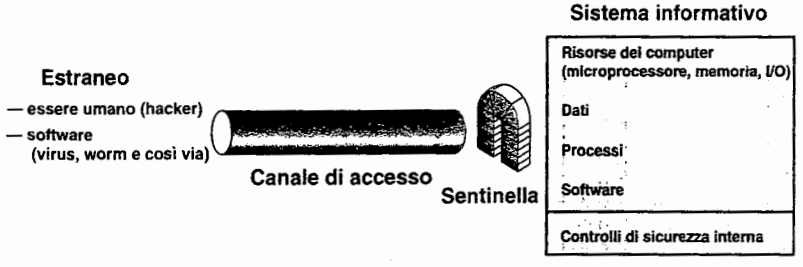


Figura 1.6 Modello di sicurezza degli accessi via rete.

risorse per acquisire vantaggi finanziari (per esempio per ottenere numeri di carte di credito o per effettuare trasferimenti illegali di denaro).

Un altro tipo di accesso indesiderato è l'inserimento in un computer di programmi che sfruttano i suoi punti deboli e possono pregiudicare l'impiego di applicazioni quali per esempio gli editor e i compilatori. Questi programmi possono rappresentare due minacce.

- Le **minacce di accesso alle informazioni** prevedono l'intercettazione o la modifica dei dati per conto di utenti che non dovrebbero avervi accesso.
- Le **minacce ai servizi** sfruttano eventuali lacune di sicurezza dei servizi presenti nei computer per impedirne l'uso agli utenti legittimi.

I virus e i worm sono due esempi di attacchi software. Tali software possono essere introdotti in un sistema tramite un disco che contiene questo codice doloso celato in altro software utile. Ma possono essere introdotti in un sistema anche utilizzando una rete: quest'ultimo caso concerne quindi più direttamente la sicurezza di rete.

I meccanismi di sicurezza necessari per impedire gli accessi indesiderati rientrano in due vaste categorie (vedere la Figura 1.6). La prima categoria è una sorta di sentinella che prevede l'uso di procedure di login a password progettate per impedire l'accesso agli utenti non autorizzati e di programmi di filtraggio in grado di rilevare e respingere i worm, i virus e altri attacchi simili. Una volta che un utente o un software indesiderato ha ottenuto l'accesso, la seconda linea di difesa è costituita da vari controlli interni che monitorizzano le attività e analizzano le informazioni memorizzate nel tentativo di rilevare la presenza di intrusi. Questi aspetti sono trattati nella parte quarta.

1.7 Letture e siti Web consigliati

[PFLE02] rappresenta una buona introduzione alla sicurezza dei computer e delle reti. [PIEP03] e [BISH05] sono altre due rassegne eccellenti. [BISH03] tratta sostanzialmente gli stessi argomenti di [BISH05] ma in modo matematicamente più approfondito e rigoroso. [SCHN00] è una lettura preziosa per tutti coloro che si occupano di sicurezza dei computer o delle reti. Discute i limiti delle tecnologie e in particolare della crittografia nel garantire la sicurezza, e la necessità di considerare l'hardware, l'implementazione software, le reti e le persone coinvolte nell'attacco e nella difesa della sicurezza.

BISH03 M. Bishop. *Computer Security: Art and Science*. Boston: Addison-Wesley, 2003.
BISH05 M. Bishop. *Introduction to Computer Security*. Boston: Addison-Wesley, 2005.
PFLE02 C. Pfleeger. *Security in Computing*. Upper Saddle River, NJ: Prentice Hall, 2002.
PIEP03 J. Pieprzyk, T. Hardjono e J. Seberry. *Fundamentals of Computer Security*. New York: Springer-Verlag, 2003.
SCHN00 B. Schneier. *Secrets and Lies: Digital Security in a Networked World*. New York: Wiley 2000.

I seguenti siti Web[6] sono particolarmente interessanti per la crittografia e la sicurezza delle reti.

- **COAST:** un ampio insieme di collegamenti ipertestuali relativi alla crittografia e alla sicurezza delle reti.
- **IETF Security Area:** materiale legato agli sforzi di standardizzazione nel settore della sicurezza in Internet.
- **Computer and Network Security Reference Index:** un buon elenco di prodotti commerciali, FAQ, archivi di newsgroup, articoli e altri siti Web.
- **The Cryptography FAQ:** Documento FAQ esteso e importante che riguarda tutti gli aspetti della crittografia.
- **Tom Dunigan's Security Page:** un eccellente elenco di puntatori a siti Web che si occupano di crittografia e sicurezza delle reti.
- **IEEE Technical Committe on Security and Privacy:** copie dei loro bollettini e informazioni sulle attività legate alla IEEE.
- **Computer Security Resource Center:** gestito dal NIST (National Institute of Standards and Technology), contiene un'ampia gamma di informazioni sui problemi, le tecnologie e gli standard di sicurezza.
- **Helgar Lipma's Cryptology Pointers:** un altro eccellente elenco di collegamenti a siti Web sulla crittografia e la sicurezza delle reti.
- **Security Focus:** una grande varietà di informazioni relative alla sicurezza, con enfasi su prodotti commerciali e preoccupazioni degli utenti finali.
- **SANS Institute:** simile al Security Focus. Estesa collezione di documenti di riferimento.

1.8 Termini chiave, domande di ripasso e problemi

Termini chiave

Analisi del traffico
Attacchi alla sicurezza
Attacco attivo
Attacco passivo
Autenticazione
Autenticità
Controllo degli accessi
Crittografia
Denial of Service (DoS)
Disponibilità
Integrità
Mascheramento
Meccanismi di sicurezza
Modello OSI
Non ripudiabilità
Ripetizione
Segretezza
Segretezza dei dati
Servizi di sicurezza

[6] Dato che le URL possono cambiare con una certa frequenza, non vengono riportate direttamente. I collegamenti appropriati a tutti i siti Web elencati in questo e nei successivi capitoli sono disponibili nel sito Web associato al volume.

Domande di riepilogo

1.1 Cos'è l'architettura di sicurezza OSI?
1.2 Qual è la differenza fra attacchi alla sicurezza attivi e passivi?
1.3 Elencare e descrivere brevemente le categorie di attacchi alla sicurezza attivi e passivi.
1.4 Elencare e descrivere brevemente le categorie dei servizi di sicurezza.
1.5 Elencare e descrivere brevemente le categorie dei meccanismi di sicurezza.

Problemi

1.1 Sviluppare una matrice simile alla Tabella 1.4 che illustri la relazione fra i servizi di sicurezza e gli attacchi.
1.2 Sviluppare una matrice simile alla Tabella 1.4 che illustri la relazione fra i meccanismi di sicurezza e gli attacchi.

Parte prima

Cifratura simmetrica

La crittografia è certamente la tecnica più importante per la sicurezza delle reti e delle comunicazioni. Vengono utilizzate due forme di crittografia: la crittografia convenzionale (o simmetrica) e la crittografia a chiave pubblica (o asimmetrica). La prima parte di questo volume introduce i principi, presenta gli algoritmi più utilizzati e discute le applicazioni della crittografia simmetrica.

Introduzione alla Parte prima

Capitolo 2: Tecniche di crittografia classiche

Il Capitolo 2 descrive le tecniche classiche di crittografia simmetrica. Costituisce un'introduzione interessante e graduale alla crittografia e all'analisi crittografica, che ne evidenzia i concetti più importanti.

Capitolo 3: Cifratura a blocchi e algorimo DES

Il Capitolo 3 introduce i principi della moderna crittografia simmetrica, ponendo l'accento sulla tecnica di crittografia più diffusa oggi, DES (Data Encryption Standard). Il capitolo comprende una discussione delle considerazioni progettuali e dell'analisi crittografica e introduce la cifratura Feistel, che sta alla base della maggior parte degli schemi di crittografia simmetrica moderni.

Capitolo 4: I campi finiti

I campi finiti sono sempre più importanti in crittografia. Vari algoritmi crittografici si basano sulle proprietà dei campi finiti, in particolare la crittografia AES (Advanced Encryption Standard) e la crittografia a curva ellittica. Questo capitolo introduce i concetti su cui si

necessari alla comprensione della tecnica AES, ovvero gli elementi di base necessari per comprendere l'aritmetica dei campi finiti della forma GF(2^n).

Capitolo 5: Lo standard AES (Advanced Encryption Standard)

Lo sviluppo più importante nella crittografia degli ultimi anni è l'adozione di un nuovo standard di cifratura simmetrica, AES. Il Capitolo 5 presenta una discussione approfondita di questa cifratura.

Capitolo 6: Approfondimenti relativi alla cifratura simmetrica

Il Capitolo 6 esplora altri argomenti relativi alla cifratura simmetrica. Questo capitolo si apre esaminando la crittografia multipla e, in particolare, triple DES. Successivamente analizza la modalità di cifratura a blocchi, che riguarda le modalità di cifratura di testo in chiaro più esteso di un singolo blocco. Introduce infine la cifratura a flussi e descrive RC4.

Capitolo 7: Segretezza e crittografia simmetrica

Oltre alle questioni che riguardano l'effettiva costruzione di un algoritmo di crittografia simmetrica, vi sono vari fattori progettuali relativi all'uso della crittografia simmetrica per garantire la segretezza. Il Capitolo 7 esplora i problemi più importanti in questo campo. Il capitolo confronta la crittografia end-to-end e a collegamento, e discute le tecniche per ottenere la segretezza del traffico e le tecniche di distribuzione della chiave. Viene trattato anche un altro argomento molto importante: la generazione di numeri casuali.

Capitolo 2

Tecniche di crittografia classiche

Concetti essenziali

- La **crittografia simmetrica** è un tipo di sistema crittografico nel quale sia la crittografia sia la decrittografia sono eseguite utilizzando la medesima chiave. Viene chiamata anche crittografia convenzionale.
- La crittografia simmetrica trasforma il testo in chiaro in testo cifrato utilizzando un algoritmo di crittografia con una chiave segreta. Il testo in chiaro viene riottenuto dal testo cifrato utilizzando *la medesima chiave* con un algoritmo di decrittografia.
- I due tipi di attacco a un algoritmo di crittografia sono l'**analisi crittografica**, basata sulle caratteristiche dell'algoritmo di cifratura, e l'attacco **a forza bruta**, che consiste nel tentare tutte le possibili chiavi.
- Le cifrature simmetriche convenzionali (sviluppate prima dell'avvento dei computer) utilizzano tecniche a **sostituzione** e/o **trasposizione**. Le tecniche a sostituzione mappano elementi del testo in chiaro (caratteri, bit) in elementi del testo cifrato. Le tecniche a trasposizione traspongono sistematicamente le posizioni degli elementi del testo in chiaro.
- Le **macchine a rotazione** sono sofisticati dispositivi hardware utilizzati prima dell'avvento dei computer, che utilizzano tecniche a sostituzione.
- La **steganografia** è una tecnica per celare un messaggio segreto all'interno di un altro messaggio in modo tale che altri non possano identificare la presenza o i contenuti del messaggio nascosto.

La crittografia simmetrica, chiamata anche crittografia convenzionale o crittografia a chiave unica, è stata l'unica forma di crittografia utilizzata prima dello sviluppo della crittografia a chiave pubblica negli anni Settanta e rimane tuttora la più utilizzata. La prima parte di questo volume esamina varie forme di cifratura simmetrica. In questo capitolo si introduce un modello generale del processo di crittografia simmetrica, che consentirà di comprendere il contesto all'interno del quale vengono utilizzati gli algoritmi. Poi si esamineranno vari

algoritmi utilizzati prima dell'avvento dei computer. Infine si accennerà brevemente a un altro approccio chiamato steganografia. Il Capitolo 3 esaminerà poi il sistema di cifratura più utilizzato: DES.

Ma prima di iniziare occorre definire alcuni termini. Il messaggio originale è chiamato **testo in chiaro** mentre il messaggio codificato viene chiamato **testo cifrato**. Il processo di conversione dal testo in chiaro al testo cifrato è chiamato **cifratura o crittografia**; l'estrazione del testo in chiaro dal testo cifrato è chiamato **decifratura o decrittografia**. Gli schemi utilizzati per la cifratura costituiscono un'area di studio chiamata anch'essa **crittografia**. Tali schemi sono chiamati sistemi **crittografici** o **cifrature**. Le tecniche utilizzate per decifrare un messaggio senza conoscere i dettagli della loro cifratura rientrano nell'area chiamata **analisi crittografica**. L'analisi crittografica consente pertanto di "violare il codice". Le aree della crittografia e dell'analisi crittografica formano insieme ciò che viene chiamata **criptologia**.

2.1 Il modello di cifratura simmetrico

Uno schema di cifratura simmetrico prevede cinque elementi (Figura 2.1).

- **Testo in chiaro:** il messaggio originale o i dati che costituiscono l'input all'algoritmo.
- **Algoritmo di crittografia:** l'algoritmo di crittografia esegue varie sostituzioni e trasformazioni sul testo in chiaro.
- **Chiave segreta:** la chiave segreta è anch'essa un input dell'algoritmo di crittografia. La chiave è un valore indipendente dal testo in chiaro e dall'algoritmo. L'algoritmo produrrà un output differente a seconda della particolare chiave utilizzata. Le sostituzioni e le trasformazioni eseguite dall'algoritmo dipendono dalla chiave.
- **Testo cifrato:** il messaggio codificato prodotto come output. Tale messaggio dipende dal testo in chiaro e dalla chiave segreta. Per un determinato messaggio in chiaro, chiavi differenti producono testi cifrati differenti. Il testo cifrato è un flusso apparentemente casuale di dati e, come si può immaginare, non è comprensibile.

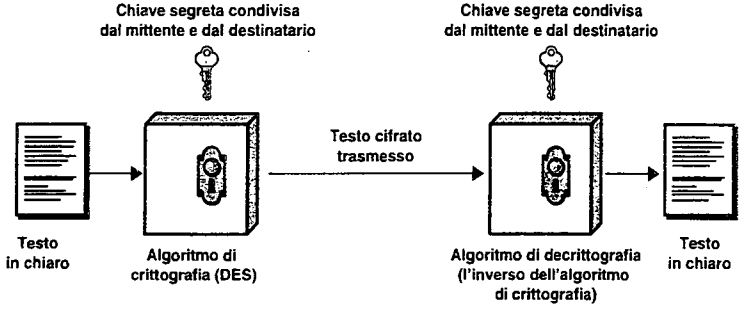


Figura 2.1 Modello semplificato di crittografia convenzionale.

- **Algoritmo di decrittografia:** si tratta fondamentalmente dell'algoritmo di crittografia eseguito al contrario. Accetta come input il testo cifrato e la chiave segreta e produce in output il testo in chiaro.

Vi sono due requisiti per utilizzare in modo sicuro la crittografia convenzionale.

1. Occorre un algoritmo di crittografia "forte". Come minimo l'algoritmo deve essere tale che un estraneo che conosca l'algoritmo e abbia accesso a uno o più testi cifrati non sia in grado di decifrare il testo cifrato o di scoprire la chiave. Questo requisito è normalmente formulato in modo più vincolante: un estraneo non deve essere in grado di decifrare il testo cifrato o di scoprire la chiave anche se fosse in possesso di più testi cifrati e dei relativi testi in chiaro.
2. Il mittente e il destinatario devono avere ottenuto copia della chiave segreta in un modo sicuro e devono mantenere sicura la chiave. Se qualcuno dovesse scoprire la chiave e conoscesse l'algoritmo, tutte le comunicazioni che utilizzano tale chiave risulterebbero leggibili.

Si suppone che non sia praticamente possibile decrittografare un messaggio sulla base del testo cifrato e della conoscenza dell'algoritmo di crittografia/decrittografia. In altre parole, non è necessario mantenere segreto l'algoritmo: è sufficiente mantenere segreta la chiave. Questa caratteristica della crittografia simmetrica è ciò che ne rende possibile l'utilizzo su ampia scala. Il fatto che l'algoritmo non debba essere mantenuto segreto, significa che i produttori possono sviluppare delle implementazioni hardware (chip) a basso costo degli algoritmi di crittografia. Questi chip sono ampiamente disponibili e incorporati in vari prodotti. Con l'uso della crittografia simmetrica, il principale problema di sicurezza è quello di mantenere segreta la chiave.

Ora si analizzeranno più da vicino gli elementi fondamentali di uno schema di crittografia simmetrico, utilizzando la Figura 2.2. Una sorgente produce un messaggio in chiaro $X = [X_1, X_2, \ldots, X_M]$. Gli M elementi di X sono lettere di un alfabeto finito. Tradizionalmente, l'alfabeto è costituito dalle 26 lettere maiuscole. Ma attualmente, viene tipicamente utilizzato l'alfabeto binario {0, 1}. Per la crittografia, viene generata una chiave nella forma $K = [K_1, K_2, \ldots, K_J]$. Se la chiave viene generata dalla sorgente del messaggio, deve poter essere fornita al destinatario tramite un canale sicuro. Alternativamente una terza parte potrebbe generare la chiave e fornirla in modo sicuro sia al mittente che al destinatario.

Usando come input il messaggio X e la chiave K, l'algoritmo di crittografia genera il testo cifrato $Y = [Y_1, Y_2, \ldots, Y_N]$. Si può rappresentare l'operazione come:

$$Y = E(K, X)$$

Questa notazione indica che Y è prodotto applicando l'algoritmo di crittografia E al testo in chiaro X con la specifica funzione determinata dal valore della chiave K.
Il destinatario, in possesso della chiave, sarà in grado di invertire la trasformazione:

$$X = D(K, Y)$$

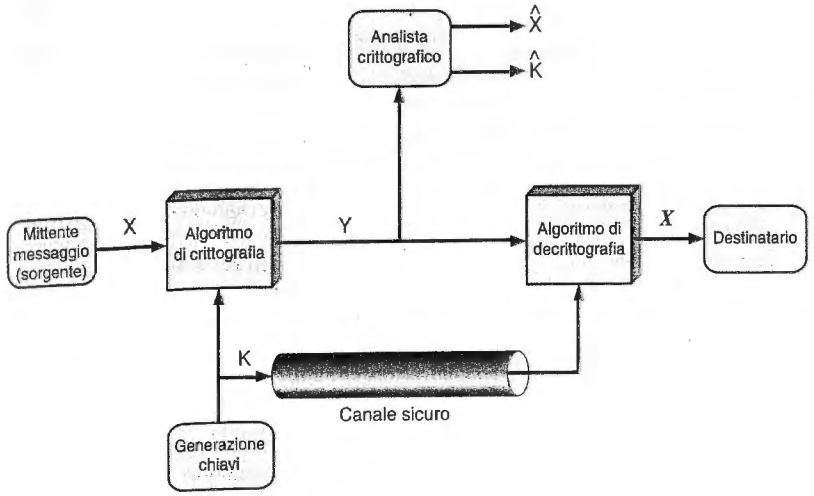


Figura 2.2 Modello di sistema crittografico convenzionale.

Un estraneo, osservando Y ma non avendo accesso a K o a X, può tentare di ottenere X, K o entrambi. Si suppone che l'estraneo conosca gli algoritmi di crittografia E e di decrittografia D. Se l'estraneo è interessato solo a questo particolare messaggio, si preoccuperà soprattutto di ripristinare X generando una stima del testo in chiaro, $\hat{X}$. Spesso tuttavia l'estraneo è interessato anche a leggere i futuri messaggi, nel qual caso tenterà quindi di individuare anche K generando una stima $\hat{K}$.

Crittografia

I sistemi crittografici sono caratterizzati da tre dimensioni indipendenti.

1. **Le operazioni utilizzate per trasformare il testo in chiaro in testo cifrato.** Tutti gli algoritmi di crittografia si basano su due principi generali: la *sostituzione*, in cui ciascun elemento del testo in chiaro (bit, lettera, gruppo di bit o di lettere) viene mappato su un altro elemento, e la *trasposizione*, in cui gli elementi del testo in chiaro vengono cambiati di posizione. Il requisito fondamentale è che non vengano perse informazioni (ovvero che tutte le operazioni siano reversibili). La maggior parte dei sistemi, chiamati sistemi prodotto, prevede più fasi di sostituzione e trasposizione.
2. **Il numero di chiavi utilizzate.** Se il mittente e il destinatario usano la stessa chiave, il sistema è detto simmetrico, a chiave unica, a chiave segreta o convenzionale. Se il mittente e il destinatario usano chiavi differenti, il sistema è chiamato asimmetrico, a due chiavi o a chiave pubblica.

3. **Il modo in cui viene elaborato il testo in chiaro.** La *cifratura a blocchi* elabora l'input un blocco di elementi per volta, producendo un blocco di output per ciascun blocco di input. La *cifratura a flussi* elabora gli elementi di input in modo continuativo, producendo l'output un elemento alla volta, a mano a mano che questo si presenta in input.

Analisi crittografica

In genere, l'obiettivo dell'attacco a un sistema di crittografia consiste nell'individuare la chiave utilizzata, piuttosto che semplicemente ottenere il testo in chiaro corrispondente a un singolo testo cifrato. Vi sono due approcci generali per attaccare uno schema di crittografia convenzionale.

- **Analisi crittografica:** un attacco ad analisi crittografica si basa sulla natura dell'algoritmo e sfrutta qualche conoscenza delle caratteristiche generali del testo in chiaro o eventualmente qualche esempio di coppia testo in chiaro/testo cifrato. Questo tipo di attacco sfrutta le caratteristiche dell'algoritmo per tentare di individuare il testo in chiaro o la chiave utilizzata.
- **Attacco a forza bruta:** si tenta ogni possibile chiave su un frammento di testo in chiaro finché non si riesce a ottenere una traduzione corretta. In media, per avere successo, occorre provare la metà delle chiavi possibili.

Se uno dei due tipi di attacchi riuscisse a dedurre la chiave, l'effetto sarebbe catastrofico: tutti i messaggi futuri e passati crittografati con tale chiave risulterebbero compromessi. Verrà considerata innanzitutto l'analisi crittografica per affrontare successivamente l'argomento degli attacchi a forza bruta.

La Tabella 2.1 riepiloga i vari tipi di **attacchi ad analisi crittografica**, sulla base delle informazioni note. Il problema più difficile si presenta quando si ha a disposizione solo il testo cifrato. In alcuni casi non è noto neppure l'algoritmo di crittografia utilizzato ma in generale si può supporre che l'estraneo lo conosca. In queste condizioni, una delle possibilità di attacco è l'approccio a forza bruta, utilizzando tutte le chiavi possibili. Questa soluzione non è tuttavia applicabile quando lo spazio delle chiavi è molto esteso, pertanto l'attacco deve basarsi sull'analisi del testo cifrato, in generale applicando vari test statistici. Per utilizzare questo approccio, occorre conoscere qualche informazione sul tipo di testo codificato, per esempio se si tratta di un testo in lingua inglese, italiana, di un file eseguibile, di un listato di codice sorgente Java, di un file di contabilità e così via.

L'attacco *Solo testo cifrato* è quello contro il quale è più facile difendersi, poiché l'estraneo ha pochissime informazioni su cui basarsi. In molti casi però l'estraneo ha a disposizione molte altre informazioni. In particolare potrebbe essere in grado di intercettare uno o più messaggi in chiaro con la rispettiva versione crittografata oppure potrebbe sapere che in un messaggio compaiono determinate configurazioni di testo in chiaro. Per esempio, un file in formato PostScript inizia sempre con la stessa configurazione o determinati messaggi potrebbero avere una certa intestazione standard nota e così via. Tutti questi esempi stanno a indicare che nella maggior parte dei casi chi svolge l'analisi crittografica ha determinate *conoscenze sul testo in chiaro ("noto")*. Sfruttando queste conoscenze, l'analista potrebbe individuare la chiave sulla base del modo in cui il testo in chiaro viene trasformato.

In stretta relazione con l'attacco a *Testo in chiaro noto* è quello che può essere chiamato *attacco a parole probabili*. Se l'estraneo stesse analizzando la versione crittografata di un messaggio di prosa generale, potrebbe avere poche conoscenze del contenuto del messaggio. Se però fosse alla ricerca di qualche informazione molto specifica, parte del messaggio potrebbe essere nota. Per esempio, se venisse trasmesso un file di contabilità, l'estraneo potrebbe conoscere la posizione di determinate parole chiave contenute nell'intestazione del file. Come esempio ulteriore, il codice sorgente di un programma sviluppato da una determinata società potrebbe contenere delle informazioni di copyright in una posizione ben precisa.

Se colui che svolge l'analisi fosse in grado, in qualche modo, di fare sì che il sistema di origine criptasse un messaggio da lui scelto, potrà attuare un attacco a *Testo in chiaro scelto*. Un esempio di questa strategia è l'analisi crittografica differenziale, trattata nel Capitolo 3. In generale, se chi svolge l'analisi è in grado di scegliere i messaggi da crittografare, potrà scegliere delle sequenze che consentiranno di individuare la struttura della chiave.

La Tabella 2.1 elenca altri due tipi di attacchi: *Testo cifrato scelto* e *Testo scelto*. Si tratta di tecniche meno utilizzate ma comunque possibili.

Solo gli algoritmi più deboli soccombono agli attacchi a *Solo testo cifrato*. In generale, un algoritmo di crittografia è progettato in modo da sopportare anche un attacco a *Testo in chiaro noto*.

Tabella 2.1 I vari tipi di attacco ai messaggi crittografati.

Tipo di attacco	Elementi noti per l'analisi crittografica
Solo testo cifrato	• Algoritmo di crittografia • Testo cifrato do decodificare
Testo in chioro noto	• Algoritmo di crittografia • Testo cifrato da decodificare • Una o più coppie testo in chiaro/testo cifrato create con la chiave segreta.
Testo in chiaro scelto	• Algoritmo di crittografia • Testo cifrato da decodificore • Messaggio in chiaro scelto da chi esegue l'analisi crittografica, insieme al corrispondente testo cifrato generato con la chiave segreta.
Testo cifroto scelto	• Algoritmo di crittografia • Testo cifrata da decodificare • Testo cifrato scelto da chi svolge l'analisi crittografica, insieme al corrispondente testo in chiaro decrittografato generato con la chiave segreta.
Testo scelto	• Algoritma di crittografia • Testo cifrato da decodificare • Messaggio in chiara scelto da chi svolge l'analisi crittografica insieme al corrispondente testo cifrato generato con la chiave segreta • Testo cifrato presunto scelto da chi svolge l'analisi crittografica insieme al corrispondente testo in chiaro decrittografato generato con la chiave segreta.

Altre due definizioni degne di nota. Uno schema di crittografia è **incondizionatamente sicuro** se il testo cifrato generato non contiene informazioni sufficienti per determinare in modo univoco il relativo testo in chiaro, indipendentemente dalla quantità di testo cifrato disponibile. Ovvero l'estraneo, indipendentemente dal tempo a sua disposizione, non può decrittografare il testo cifrato, semplicemente perché non contiene le informazioni ricercate. Con l'eccezione di uno schema chiamato One-Time Pad (di cui si parlerà più avanti nel capitolo), non vi è alcun algoritmo di crittografia incondizionamente sicuro. Pertanto gli algoritmi di crittografia possono solo soddisfare uno o entrambi i criteri seguenti.

- Il costo della violazione del testo cifrato supera il valore delle informazioni crittografate.
- Il tempo richiesto per violare il testo cifrato è superiore alla vita utile delle informazioni.

Uno schema di crittografia è detto **computazionalmente sicuro** se soddisfa a uno di questi due criteri. Il problema è che è molto difficile stimare l'impegno necessario per violare un testo cifrato.

Tutte le forme di analisi crittografica per gli schemi di crittografia simmetrici sono progettate per sfruttare il fatto che nel testo cifrato possono rimanere delle tracce distinguibili della struttura o degli schemi presenti nel testo in chiaro. Questo concetto diverrà più chiaro proseguendo nello studio dei vari schemi di crittografia descritti in questo capitolo. Nella Parte seconda si vedrà invece che l'analisi crittografica degli schemi a chiave pubblica deriva da una premessa radicalmente differente, ovvero che sia possibile derivare una chiave dall'altra sfruttando le loro proprietà matematiche.

L'attacco a forza bruta prevede il tentativo di ogni chiave possibile fino a ottenere una traduzione intelligibile del testo cifrato in testo in chiaro. In media, per avere successo occorre provare la metà di tutte le chiavi possibili. La Tabella 2.2 mostra il tempo necessario per differenti dimensioni dello spazio delle chiavi. I risultati vengono indicati per quattro diverse dimensioni delle chiavi binarie. Le chiavi a 56 bit vengono utilizzate dall'algoritmo DES (Data Encryption Standard) e le chiavi a 168 bit vengono utilizzate da triple-DES. Le dimensioni minime della chiave per AES (Advanced Encryption Standard) sono di 128 bit. Vengono anche indicati i risultati per i codici di sostituzione che utilizzano una chiave di 26 caratteri (se ne parlerà più avanti) in cui vengono utilizzate come chiavi tutte le possibili permutazioni dei 26 caratteri. Per ciascuna dimensione, vengono indicati i risultati supponendo che per svolgere una singola decrittografia sia necessario 1 μs, un ordine di grandezza ragionevole per le macchine di oggi. Con l'uso di architetture massivamente parallele è possibile ottenere velocità di elaborazione di molti ordini di grandezza superiori. L'ultima colonna della Tabella 2.2 considera i risultati per un sistema in grado di elaborare un milione di chiavi al microsecondo. Come si può vedere, a questo livello di prestazioni l'algoritmo DES non può più essere considerato computazionalmente sicuro.

Tabella 2.2 Tempi medi per una ricerca esaustiva della chiave.

Dimensioni della chiave (in bit)	Numero di chiavi alternative	Tempo necessario a 1 crittografia/ microsecondo	Tempo necessario a 10^6 crittografie/ microsecondo
32	$2^{32} = 4,3 \times 10^9$	2^{31} μs = 35,8 minuti	2,15 millisecondi

(segue)

Tabella 2.2 Tempi medi per una ricerca esaustiva della chiave. *(continua)*

Dimensioni della chiave (in bit)	Numero di chiavi alternative	Tempo necessario a 1 crittografia/ microsecondo	Tempo necessario a 10^6 crittografie/ microsecondo
56	$2^{56} = 7{,}2 \times 10^{16}$	2^{55} µs = 1142 anni	10,01 ore
128	$2^{128} = 3{,}4 \times 10^{38}$	2^{127} µs = $5{,}4 \times 10^{24}$ anni	$5{,}4 \times 10^{18}$ anni
168	$2^{168} = 3{,}7 \times 10^{50}$	2^{167} µs = $5{,}9 \times 10^{36}$ anni	$5{,}9 \times 10^{30}$ anni
26 caratteri (permutazione)	$26! = 4 \times 10^{26}$	2×10^{26} µs = $6{,}4 \times 10^{12}$ anni	$6{,}4 \times 10^{6}$ anni

2.2 Tecniche di sostituzione

In questa parte del capitolo e nella prossima parte verranno esaminati alcuni esempi di tecniche di crittografia classiche. Lo studio di queste tecniche consente di illustrare gli approcci di base alla crittografia simmetrica attualmente utilizzata e i tipi di attacchi ad analisi crittografica che si devono prevedere.

I due elementi fondamentali di tutte le tecniche di crittografia sono la sostituzione e la trasposizione, che verranno trattate nei due paragrafi seguenti. Infine si descriverà un sistema che combina la sostituzione con la trasposizione.

Una tecnica di sostituzione consente di sostituire le lettere del testo in chiaro con altre lettere o numeri o simboli.[1] Se si considera il testo in chiaro come una sequenza di bit, la sostituzione comporta la sostituzione delle sequenze di bit del testo in chiaro con le sequenze di bit del testo cifrato.

Cifratura di Cesare

La cifratura più semplice è quella utilizzata da Giulio Cesare che prevede la sostituzione di ciascuna lettera con la lettera che si trova a tre posizioni di distanza nell'alfabeto. Per esempio:

```
chiaro:  meet me after the toga party
cifrato: PHHW PH DIWHU WKH WRJD SDUWB
```

Si noti che l'alfabeto è ciclico e dunque la prima lettera dopo la Z è la A. Si può definire la trasformazione elencando tutte le possibilità come nel seguente esempio:

[1] Quando si utilizzano lettere, in questo volume vengono adottate le seguenti convenzioni: il testo in chiaro viene riportato in lettere minuscole, il testo cifrato viene riportato in lettere maiuscole, i valori della chiave vengono indicati in lettere minuscole corsive.


```
chiaro:  a b c d e f g h i j k l m n o p q r s t u v w x y z
cifrato: D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
```

Si assegni ora un equivalente numerico a ciascuna lettera:

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12

n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

L'algoritmo può essere quindi espresso nel seguente modo. Per ciascuna lettera p del testo in chiaro, sostituire la lettera C del testo cifrato:[2]

$$C = E(3, p) = (p + 3) \bmod 26$$

Lo scorrimento può in realtà essere di qualsiasi entità e dunque in generale l'algoritmo di Cesare può essere definito nel seguente modo:

$$C = E(k, p) = (p + k) \bmod 26$$

dove k assume un valore compreso fra 1 e 26. L'algoritmo di decrittografia è semplicemente:

$$p = D(k, C) = (C - k) \bmod 26$$

Se si sa che un determinato testo è stato cifrato con l'algoritmo di Cesare, è facile individuare un algoritmo di analisi crittografica a forza bruta. È sufficiente provare tutte le 25 possibili chiavi. La Figura 2.3 mostra i risultati dell'applicazione di questa strategia al testo cifrato dell'esempio. In questo caso il testo in chiaro viene individuato dalla terza riga.

Sono tre le caratteristiche significative di questo problema che consentono di utilizzare un'analisi crittografica a forza bruta.

1. Sono noti gli algoritmi di crittografia e decrittografia.
2. Si devono provare solo 25 chiavi.
3. Il linguaggio utilizzato per il testo in chiaro è noto e facilmente riconoscibile.

Nella maggior parte delle situazioni di rete si può supporre che gli algoritmi siano noti. Ciò che in generale rende poco pratica l'analisi crittografica a forza bruta è l'uso di algoritmi che impiegano un gran numero di chiavi. Per esempio, l'algoritmo triple DES esaminato

[2] Si definisce a mod n il resto della divisione fra a e n. Per esempio, 11 mod 7 = 4. Per ulteriori informazioni sull'aritmetica modulare, consultare il Capitolo 4.

KEY	PHHW PH DIWHU WKH WRJD SDUWB
1	oggv og chvgt vjg vqic rctva
2	nffu nf bgufs uif uphb qbsuz
3	meet me after the toga party
4	ldds ld zesdq sgd snfz ozqsx
5	kccr kc ydrcp rfc rmey nyprw
6	jbbq jb xcqbo qeb qldx mxoqv
7	iaap ia wbpan pda pkcw lwnpu
8	hzzo hz vaozm ocz ojbv kvmot
9	gyyn gy uznyl nby niau julns
10	fxxm fx tymxk max mhzt itkmr
11	ewwl ew sxlwj lzw lgys hsjlq
12	dvvk dv rwkvi kyv kfxr grikp
13	cuuj cu qvjuh jxu jewq fqhjo
14	btti bt puitg iwt idvp epgin
15	assh as othsf hvs hcuo dofhm
16	zrrg zr nsgre gur gbtn cnegl
17	yqqf yq mrfqd ftq fasm bmdfk
18	xppe xp lqepc esp ezrl alcej
19	wood wo kpdob dro dyqk zkbdi
20	vnnc vn jocna cqn cxpj yjach
21	ummb um inbmz bpm bwoi xizbg
22	tlla tl hmaly aol avnh whyaf
23	skkz sk glzkx znk zumg vgxze
24	rjjy rj fkyjw ymj ytlf ufwyd
25	qiix qi ejxiv xli xske tevxc

Figura 2.3 Analisi crittografica a forza bruta della cifratura di Cesare.

nel Capitolo 6, usa una chiave di 168 bit, quindi il numero di chiavi è pari a 2^{168}, ovvero, esistono più di $3{,}7 \times 10^{50}$ possibili chiavi.

È significativa anche la terza caratteristica. Se il linguaggio del testo in chiaro fosse sconosciuto, potrebbe essere difficile da riconoscere. Inoltre l'input potrebbe essere abbreviato o compresso in qualche modo, complicando ulteriormente il riconoscimento. Per esempio la Figura 2.4 mostra una porzione di file di testo compressa utilizzando un algoritmo chiamato ZIP. Se questo file venisse poi crittografato con una semplice cifratura a sostituzione (estesa per includere più dei 26 caratteri alfabetici), il testo in chiaro potrebbe non essere riconosciuto, quando venisse "scoperto" da un'analisi crittografica a forza bruta.


```
˜+Wμ"— Ω-O)≤4{∞‡¸ ë~Ω%ràu·¯Í ◊¯Z-
Ú≠2Ò#Åæð œ«q7¸Ωn·®3NÛÚ Œz'Y-ƒ∞Í[±Û_ èΩ,<NO¬±«ˇxã  Åäfèü3Å
x}ö§k°Â
_yÍ ^ΔÉ] ¸¤ J/˚iTê&ı 'c<uΩ-
 ÄD(G WÄC~y_ïõÄW PÔı«ÎÜ†ç],¤¡ˇÌ^üÑπ˙≈ˇL˘9OgflO˜&Œ≤ ¬≤ ØÔ§˝:
 ˇŒ!SGqèvo^ ú\¸S>h<-*6ø‡%x´˝|fiÓ#≈˜my‰ˇ≥ñP<¸ƒi Áj  Å◊¿˝Zù-
Ω¨Õ¯6Œÿ{% „ΩÊó ¸i π÷ÁÎ˚úO2çSŷ´O-
2ÄflßI /@^”∏Kº³PŒπ¸úé^´3∑˜ö˘ÔZÌ"Y¬ŸΩœY› Ω+eô/ ˙<K£¿*÷˜"≤û˜
B ZøK˜Qßÿüƒ¸!ÒflÎzsS/]›ÈQ ü
```

Figura 2.4 Esempio di testo compresso.

Cifratura monoalfabetica

Con solo 25 possibili chiavi, la cifratura di Cesare è tutt'altro che sicura. Un notevole incremento nello spazio delle chiavi può essere ottenuto consentendo una sostituzione arbitraria. Si ricordi l'assegnamento della cifratura di Cesare:

```
chiaro:    a b c d e f g h i j k l m n o p q r s t u v w x y z
cifrato:   D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
```

Se invece la riga di testo cifrato potesse essere una permutazione dei 26 caratteri alfabetici, vi sarebbero 26! ovvero più di 4×10^{26} possibili chiavi. Si tratta di un ordine di grandezza 10 volte superiore allo spazio delle chiavi utilizzato da DES, cosa che sembrerebbe eliminare la possibilità di utilizzare un attacco a forza bruta. Questo approccio è chiamato **cifratura a sostituzione monoalfabetica**, poiché viene utilizzato un unico alfabeto di cifratura (mapping da un alfabeto in chiaro a un alfabeto cifrato) per messaggio.

Vi è però un altro metodo di attacco. Se l'analista crittografico conosce la natura del testo in chiaro (per esempio sa che si tratta di testo inglese non compresso), potrà sfruttare le regolarità presenti nel linguaggio. Per vedere come applicare questo tipo di analisi crittografica, si utilizzerà un esempio adattato da [SINK66]. Ecco il testo cifrato da risolvere:

```
UZOSOVUOHXMDPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ
VUEPHZHMDZSHZOWSFPAPPDTSVPQUZWYMXUZUHSX
EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ
```

Innanzitutto si può determinare la frequenza relativa delle lettere e confrontarla con la distribuzione standard delle frequenze per la lingua inglese, come indicato nella Figura 2.5 (basata su [LEWA00]). Se il messaggio fosse sufficientemente lungo, questa tecnica potrebbe essere sufficiente, ma poiché si tratta di un messaggio relativamente breve, non ci si può attendere una corrispondenza esatta. In ogni caso, la frequenza relativa delle lettere nel testo cifrato (in percentuali) è la seguente:

P	13,33	H	5,83	F	3,33	B	1,67	C	0,00
Z	11,67	D	5,00	W	3,33	G	1,67	K	0,00
S	8,33	E	5,00	Q	2,50	Y	1,67	L	0,00
U	8,33	V	4,17	T	2,50	U	0,83	N	0,00
O	7,50	X	4,17	A	1,67	J	0,83	R	0,00
M	6,67								

Confrontando questo conteggio con la Figura 2.5, sembra plausibile che le lettere P e Z siano equivalenti alle lettere in chiaro "e" e "t" ma non è certo se a "P" corrisponda "e" piuttosto che "t". Le lettere S, U, O, M e H hanno anch'esse una frequenza relativamente elevata e probabilmente corrispondono alle lettere in chiaro dell'insieme {a, h, i, n, o, r, s}. Le lettere con le frequenze più basse (ovvero A, B, G, Y, I, J) sono probabilmente incluse nell'insieme {b, j, k, q, v, x, z}.

A questo punto vi sono vari modi per procedere. Si può tentare qualche assegnamento e iniziare a compilare il testo in chiaro per vedere se si ottiene una struttura ragionevole di un messaggio. Un approccio più sistematico prevede la ricerca di altre regolarità. Per esempio, si potrebbe sapere che il testo contiene determinate parole o si potrebbero ricercare sequenze ripetute di lettere per cercare di dedurre il loro equivalente in chiaro.

Uno strumento potente consiste nel ricercare la frequenza delle combinazioni di due lettere. Si può tracciare una tabella simile a quella della Figura 2.5 che mostra la frequenza relativa dei digrammi. Nella lingua inglese il più comune di questi digrammi è th. Nel testo cifrato il digramma più comune è ZW che si presenta per tre volte. Si può quindi provare la corrispondenza della lettera Z con la lettera t e della lettera W con la lettera h. In base alle ipotesi precedenti, si possono far corrispondere la lettera P e la lettera e. Ora si noti che nel testo cifrato compare la sequenza ZWP e si può dunque immaginare che rappresenti la sequenza "the". Questo è il trigramma (combinazione di tre lettere) più frequente in inglese, cosa che sembra indicare che si sta percorrendo la strada giusta.

Quindi si noti la sequenza ZWSZ nella prima riga. Non si può sapere se queste quattro lettere formino una parola completa, ma se così fosse, sarebbe nella forma th_t. Pertanto la lettera S corrisponderebbe alla lettera a.

Finora si ha quindi:

```
UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ
 t a       e  e te  a that e e a       a
VUEPHZHMDZSHZOWSFPAPPDTSVPQUZWYMXUZUHSX
  e t   ta t ha e ee  a e  th    t  a
EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ
 r  e e tat  e   the t
```

Sono state identificate solo quattro lettere ma si ha già una parte rilevante del messaggio. Ripetendo l'analisi delle frequenze con vari tentativi si dovrebbe giungere con facilità alla soluzione. Il testo in chiaro completo, con l'aggiunta degli spazi fra le parole è:

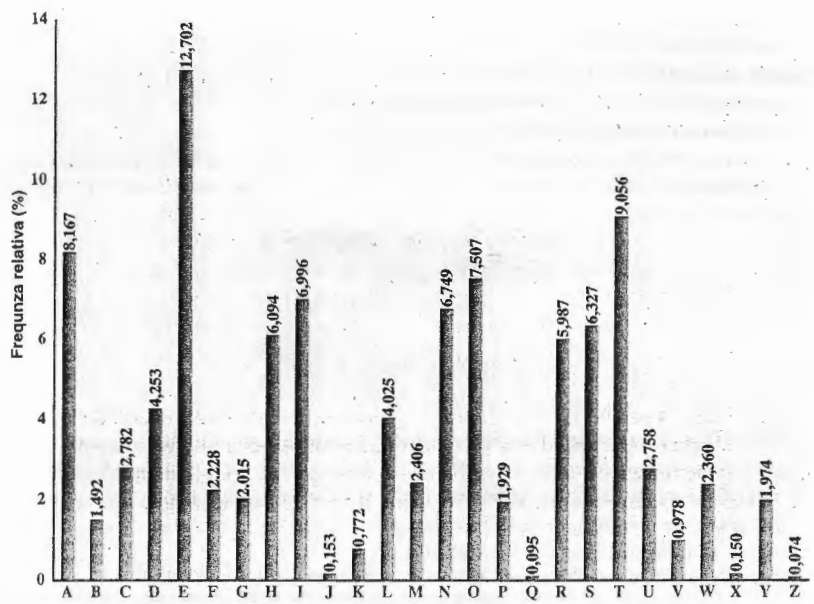


Figura 2.5 Frequenza relativa delle lettere nella lingua inglese scritta.

```
it was disclosed yesterday that several informal but
direct contacts have been made with political
representatives of the viet cong in moscow
```

Le cifrature monoalfabetiche sono facili da violare poiché conservano le informazioni di frequenza dell'alfabeto originario. Una contromisura è quella di utilizzare più sostituti (omofoni) per la medesima lettera. Per esempio, alla lettera "e" si potrebbero assegnare vari simboli cifrati (come 16, 74, 35 e 21) assegnati a rotazione o casualmente. Se il numero di simboli assegnati a ciascuna lettera è proporzionale alla sua frequenza relativa, le informazioni relative alla frequenza della singola lettera vengono completamente eliminate. Il grande matematico Carl Friedrich Gauss riteneva di avere individuato una cifratura inviolabile utilizzando gli omofoni ma anche in questo caso, ciascun elemento del testo in chiaro viene assegnato a un solo elemento del testo cifrato e le informazioni relative alle sequenze di più lettere (le frequenze dei digrammi per esempio) rimangono ancora nel testo cifrato, rendendo l'analisi crittografica relativamente semplice.

Per ridurre questa sopravvivenza della struttura del testo in chiaro nel testo cifrato si impiegano principalmente due metodi: un approccio consiste nel crittografare più lettere del testo in chiaro e l'altro consiste nell'utilizzare più alfabeti cifrati.

Di seguito verranno esaminati entrambi questi approcci.

Cifratura Playfair

La cifratura multi-lettera più nota è chiamata Playfair e tratta i digrammi del testo in chiaro come singole unità traducendoli in digrammi cifrati.[3]

L'algoritmo Playfair si basa sull'uso di una matrice 5 × 5 costruita utilizzando una parola chiave. Ecco un esempio risolto da Lord Peter Wimsey in *Have His Carcase* di Dorothy Sayers:[4]

M	O	N	A	R
C	H	Y	B	D
E	F	G	I/J	K
L	P	Q	S	T
U	V	W	X	Z

In questo caso la parola chiave è *monarchy*. La matrice è costruita introducendo le lettere della parola chiave (esclusi i duplicati) da sinistra a destra e dall'alto al basso e poi completando la parte rimanente della matrice con le altre lettere specificate in ordine alfabetico. Le lettere I e J contano come una sola lettera. Il testo in chiaro viene crittografato due lettere alla volta, in base alle seguenti regole.

1. Le lettere di testo in chiaro ripetute che cadrebbero nella stessa coppia devono essere separate da una lettera di riempimento, per esempio x; in questo modo la parola balloon verrebbe trattata come ba lx lo on.
2. Due lettere di testo in chiaro che rientrano nella stessa riga della matrice vengono tutte sostituite dalla lettera che si trova a destra (il primo elemento della riga segue l'ultimo in modo circolare). Per esempio, ar verrà crittografato come RM.
3. Due lettere di testo in chiaro che rientrano nella stessa colonna vengono sostituite dalla lettera sottostante (l'elemento superiore della colonna segue l'ultimo). Per esempio mu verrà crittografato come CM.
4. In ogni altro caso, ciascuna lettera di testo in chiaro di una coppia verrà sostituita dalla lettera che si trova nella stessa riga e nella colonna occupata dall'altra lettera di testo in chiaro. Pertanto hs diventerà BP e ea diventerà IM (o JM come si desidera).

La cifratura Playfair rappresenta già un notevole miglioramento rispetto alla semplice cifratura monoalfabetica. Per esempio, mentre vi sono solo 26 lettere, vi sono 26 × 26 = 676 digrammi e dunque l'identificazione dei singoli digrammi è più difficoltosa. Inoltre le frequenze relative delle singole lettere hanno un intervallo molto più ampio di quello dei digrammi, complicando ulteriormente l'analisi della frequenza. Per questi motivi, la cifratura Playfair è stata considerata per lungo tempo inviolabile. In particolare è stata utilizzata come sistema di comunicazione sul campo dell'esercito inglese nella prima guerra mondiale ed era ancora diffusamente utilizzata dall'esercito americano e dalle forze alleate durante la seconda guerra mondiale.

[3] Questa cifratura è stata in realtà inventata dallo scienziato inglese Sir Charles Wheatstone nel 1854 ma porta il nome del suo amico Playfair Barone di St. Andrews, che ha depositato la cifratura al Foreign Office britannico.
[4] Il testo fornisce un esempio interessantissimo di un attacco a probabilità delle parole.

Nonostante questo livello di fiducia nella sua sicurezza, la cifratura Playfair risulta comunque relativamente facile da violare poiché conserva ancora molta della struttura del linguaggio in chiaro. In generale bastano poche centinaia di lettere di testo cifrato per poterla violare.

Un modo per rivelare l'efficacia dell'algoritmo Playfair e delle altre cifratura è rappresentato nella Figura 2.6 che si basa su [SIMM93]. La linea etichettata *testo in chiaro* rappresenta la distribuzione delle frequenze degli oltre 70 000 caratteri alfabetici presenti nell'articolo dell'*enciclopedia britannica* sulla criptologia.[5] Questa è anche la distribuzione di frequenza di qualsiasi cifratura a sostituzione monoalfabetica. Il tracciato è stato realizzato nel modo seguente: il numero di occorrenze di ciascuna lettera nel testo è stata contata e divisa per il numero di occorrenze della lettera e (la lettera più utilizzata). Pertanto la lettera e ha una frequenza relativa di 1, t di circa 0,76 e così via. I punti sull'asse orizzontale corrispondono alle lettere in ordine decrescente di frequenza.

La Figura 2.6 mostra anche la distribuzione di frequenza risultante quando il testo viene crittografato utilizzando la cifratura Playfair. Per normalizzare il tracciato, il numero di occorrenze di ciascuna lettera nel testo cifrato è stato nuovamente diviso per il numero di presenze della lettera "e" nel testo in chiaro. Il tracciato risultante mostra pertanto quanto viene mascherata in cui la distribuzione di frequenza delle lettere (che rende banale la risoluzione della cifratura a sostituzione) dalla crittografia. Se le informazioni sulla distribuzione della frequenza venissero completamente celate dal processo di crittografia, il tracciato delle frequenze del testo cifrato sarebbe piatto e l'analisi crittografica utilizzando solamente il testo crittografato risulterebbe impossibile. Come si può vedere dalla figura, la cifratura Playfair ha una distribuzione più piatta rispetto al testo in chiaro ma ciononostante rivela molta struttura, sfruttabile da un'analisi crittografica.

Cifratura Hill[6]

Un'altra interessante cifratura multi-lettera è quella sviluppata dal matematico Lester Hill nel 1929. L'algoritmo di crittografia prende una sequenza di m lettere in chiaro e le sostituisce con m lettere di testo cifrato. La sostituzione è determinata da m equazioni lineari in cui a ciascun carattere viene assegnato un valore numerico (a = 0, b = 1, . . . z = 25). Per m = 3, il sistema può essere descritto nel seguente modo:

$$c_1 = (k_{11}p_1 + k_{12}p_2 + k_{13}p_3) \bmod 26$$
$$c_2 = (k_{21}p_1 + k_{22}p_2 + k_{23}p_3) \bmod 26$$
$$c_3 = (k_{31}p_1 + k_{32}p_2 + k_{33}p_3) \bmod 26$$

Questa formula può essere espressa in termini di vettori colonna e matrici:

[5] Gustavus Simmons ha fornito i tracciati e descritto il loro metodo di costruzione.
[6] Questa cifratura è un po' più complessa da comprendere rispetto alle altre presentate nel capitolo ma illustra un elemento importante dell'analisi crittografica che risulterà utile più avanti. Questo paragrafo può anche essere ignorato in una prima lettura.

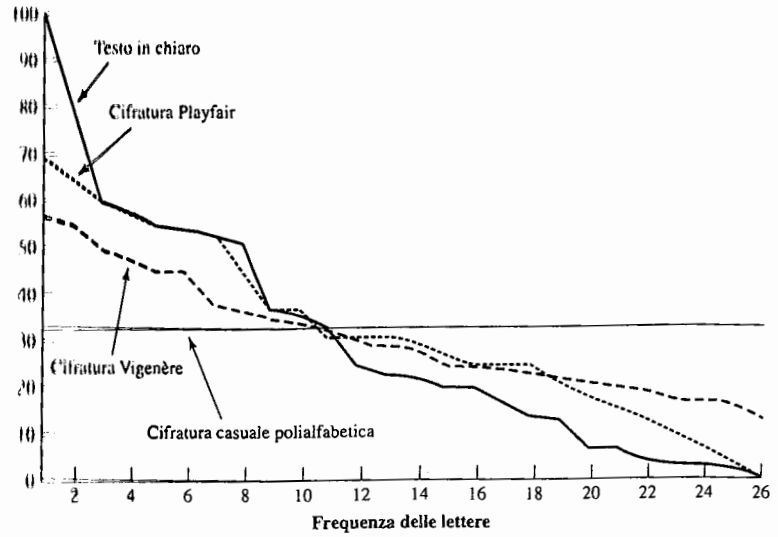


Figura 2.6 Frequenza relativa delle occorrenze delle lettere.

$$\begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} \bmod 26$$

oppure

$$\mathbf{C} = \mathbf{KP} \bmod 26$$

dove **C** e **P** sono vettori colonna di lunghezza 3 che rappresentano rispettivamente il testo in chiaro e il testo cifrato, e **K** è una matrice 3×3 che rappresenta la chiave di crittografia. Le operazioni vengono svolte in modulo 26.

Per esempio, si consideri il testo in chiaro "paymoremoney" e si utilizzi la chiave di crittografia:

$$\mathbf{K} = \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}$$

Le prime tre lettere del testo in chiaro sono rappresentate dal vettore $\begin{pmatrix} 15 \\ 0 \\ 24 \end{pmatrix}$. Quindi $\mathbf{K}\begin{pmatrix} 15 \\ 0 \\ 24 \end{pmatrix} = \begin{pmatrix} 375 \\ 819 \\ 486 \end{pmatrix} \bmod 26 = \begin{pmatrix} 11 \\ 13 \\ 18 \end{pmatrix} \equiv$ LNS. Continuando in questo modo, il testo cifrato dell'intero testo in chiaro è LNSHDLEWMTRW.

Per la decrittografia si utilizza l'inversa della matrice **K**. L'inversa $\mathbf{K}^{-1}$ di una matrice **K** è definita dall'equazione $\mathbf{K}\mathbf{K}^{-1} = \mathbf{K}^{-1}\mathbf{K} = \mathbf{I}$, dove **I** è la matrice che contiene tutti valori 0 (zero) ad eccezione degli 1 (uno) nella diagonale principale dall'angolo superiore sinistro a quello inferiore destro. L'inversa di una matrice non esiste sempre ma quando esiste soddisfa l'equazione precedente. In questo caso l'inversa è:

$$\mathbf{K}^{-1} = \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix}$$

Ecco la dimostrazione:

$$\begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix} \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix} = \begin{pmatrix} 443 & 442 & 442 \\ 858 & 495 & 780 \\ 494 & 52 & 365 \end{pmatrix} \bmod 26 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Si può vedere con facilità che se la matrice $\mathbf{K}^{-1}$ viene applicata al testo cifrato, si riottiene il testo in chiaro. Per spiegare il modo in cui viene determinata l'inversa di una matrice, si introducono alcune nozioni di algebra lineare[7]. Per ogni matrice quadrata ($m \times m$), il *determinante* è uguale alla somma di tutti i prodotti che possono essere calcolati prendendo esattamente un elemento da ciascuna riga ed esattamente un elemento da ciascuna colonna con alcuni termini del prodotto precedute dal segno meno. Per una matrice 2×2:

$$\begin{pmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{pmatrix}$$

il determinante è $k_{11}k_{22} - k_{12}k_{21}$. Per una matrice 3×3, il valore del determinante è $k_{11}k_{22}k_{33} + k_{21}k_{32}k_{13} + k_{31}k_{12}k_{23} - k_{31}k_{22}k_{13} - k_{21}k_{12}k_{33} - k_{11}k_{32}k_{23}$. Se una matrice quadrata ha un determinante diverso da 0 (zero), allora l'inversa della matrice viene calcolata come $[A^{-1}]_{ij} = (-1)^{i+j}(D_{ji})/\det(A)$, dove (D_{ji}) è il sottodeterminante creato cancellando la j-esima riga e la i-esima colonna di **A** e det(**A**) è il determinante di **A**. Per gli scopi di questo volume, tutta l'aritmetica viene eseguita in modulo 26.

In termini generali, il sistema di Hill può essere espresso nel seguente modo:

$$\mathbf{C} = E(\mathbf{K}, \mathbf{P}) = \mathbf{KP} \bmod 26$$

$$\mathbf{P} = D(\mathbf{K}, \mathbf{C}) = \mathbf{K}^{-1}\mathbf{C} \bmod 26 = \mathbf{K}^{-1}\mathbf{KP} = \mathbf{P}$$

Come con Playfair, la potenza della cifratura di Hill deriva dal fatto che essa nasconde completamente le frequenze mono-lettera. In effetti, la cifratura di Hill, usando una matrice di maggiori dimensioni nasconde ulteriori informazioni sulla frequenza. La cifratura di Hill 3×3 nasconde non solo le informazioni sulla frequenza delle singole lettere ma anche quelle di due lettere.

[7] I concetti di base dell'algebra lineare sono riassunti nel documento Math Refresher disponibile sul sito **http://WilliamStallings.com/StudentsSupport.html**. Il lettore interessato potrà consultare un qualsiasi testo su questo argomento.

Anche se la cifratura di Hill è resistente a un attacco che considera solo il testo cifrato, può essere violata con facilità con un attacco a testo in chiaro noto. Per una cifratura di Hill $m \times m$, si supponga di avere m coppie di testo in chiaro/testo cifrato ognuna di lunghezza m. Si possono etichettare le coppie con $\mathbf{P}_j = \begin{pmatrix} p_{1,j} \\ p_{2,j} \\ \vdots \\ p_{m,j} \end{pmatrix}$ e $\mathbf{C}_j = \begin{pmatrix} c_{1,j} \\ c_{2,j} \\ \vdots \\ c_{m,j} \end{pmatrix}$ in modo che $\mathbf{C}_j = \mathbf{K}\mathbf{P}_j$ per $1 \le j \le m$ e per una certa matrice chiave sconosciuta $\mathbf{K}$. Si definiscano ora due matrici $m \times m$ $\mathbf{X} = (p_{ij})$ e $\mathbf{Y} = (c_{ij})$: così si può creare l'equazione di matrici $\mathbf{Y} = \mathbf{K}\mathbf{X}$. Se $\mathbf{X}$ ha un'inversa, si può determinare $\mathbf{K} = \mathbf{Y}\mathbf{X}^{-1}$. Se X non e invertibile, allora si può creare una nuova versione di $\mathbf{X}$ con ulteriori coppie testo in chiaro/testo cifrato fino a ottenere una $\mathbf{X}$ invertibile.

Si utilizzerà un esempio basato su [STIN02]. Si supponga che la parola in chiaro "friday" venga crittografata utilizzando una cifratura di Hill 2×2 per fornire il testo cifrato PQCFKU.

Pertanto si sa che $\mathbf{K}\begin{pmatrix} 5 \\ 17 \end{pmatrix} \bmod 26 = \begin{pmatrix} 15 \\ 16 \end{pmatrix}$; $\mathbf{K}\begin{pmatrix} 8 \\ 3 \end{pmatrix} \bmod 26 = \begin{pmatrix} 2 \\ 5 \end{pmatrix}$ e $\mathbf{K}\begin{pmatrix} 0 \\ 24 \end{pmatrix} \bmod 26 = \begin{pmatrix} 10 \\ 20 \end{pmatrix}$.

Utilizzando le prime due coppie testo in chiaro/testo cifrato si avrà:

$$\begin{pmatrix} 15 & 2 \\ 16 & 5 \end{pmatrix} = \mathbf{K}\begin{pmatrix} 5 & 8 \\ 17 & 3 \end{pmatrix} \bmod 26$$

Si può calcolare l'inversa di X:

$$\begin{pmatrix} 5 & 8 \\ 17 & 3 \end{pmatrix}^{-1} = \begin{pmatrix} 9 & 2 \\ 1 & 15 \end{pmatrix}$$

e pertanto:

$$\mathbf{K} = \begin{pmatrix} 15 & 2 \\ 16 & 5 \end{pmatrix}\begin{pmatrix} 9 & 2 \\ 1 & 15 \end{pmatrix} = \begin{pmatrix} 137 & 60 \\ 149 & 107 \end{pmatrix} \bmod 26 = \begin{pmatrix} 7 & 8 \\ 19 & 3 \end{pmatrix}$$

Questo risultato viene verificato con le rimanenti coppie di testo in chiaro/testo cifrato.

Cifratura polialfabetica

Un altro modo per migliorare la tecnica monoalfabetica semplice è quello di utilizzare delle sostituzioni monoalfabetiche differenti a mano a mano che si procede nel messaggio in chiaro. Questo approccio viene chiamato **cifratura a sostituzione polialfabetica**. Tutte queste tecniche hanno in comune le seguenti caratteristiche.

1. Un insieme di regole di sostituzione monoalfabetica.
2. Una chiave che determina la regola scelta per una determinata trasformazione.

La tecnica più nota (e una delle più semplici) è la cifratura di Vigenère. In questo schema, l'insieme di regole di sostituzione monoalfabetica è costituito da 26 cifrature di Cesare con scorrimenti da 0 a 25. Ogni cifratura è denotata da una lettera chiave che è la lettera cifrata

che sostituisce la lettera in chiaro "a". Pertanto una cifratura di Cesare con uno scorrimento pari a 3 sarà indicata dal valore della chiave *d*.

Per comprendere e utilizzare lo schema si può costruire una matrice chiamata tabella di Vigenère (vedere la Tabella 2.3). Le 26 cifrature sono disposte orizzontalmente e sulla sinistra viene indicata la lettera chiave di ciascuna cifratura. Nella parte superiore è indicato l'alfabeto in chiaro. Il processo di crittografia è semplice: data la chiave *x* e la lettera in chiaro y, la lettera cifrata si trova all'intersezione della riga *x* e della colonna y; in questo caso il risultato è V.

Per crittografare un messaggio, la chiave deve essere lunga quanto il messaggio stesso. Normalmente ciò si ottiene con una parola chiave ripetuta. Per esempio, se la parola chiave è *deceptive*, il messaggio "we are discovered save yourself" verrà crittografato nel seguente modo:

```
chiave:     deceptivedeceptivedeceptive
chiaro:     wearediscoveredsaveyourself
cifrato:    ZICVTWQNGRZGVTWAVZHCQYGLMGJ
```

La decrittografia è altrettanto semplice. La lettera chiave identifica ancora una volta la riga. La posizione della lettera cifrata all'interno di tale riga determina la colonna e la lettera in chiaro è indicata in cima alla colonna. La potenza di questo sistema di cifratura consiste nel fatto che vi sono più lettere cifrate per ciascuna lettera in chiaro, una per ciascuna lettera univoca della parola chiave. Pertanto le informazioni sulla frequenza delle lettere vengono ridotte sebbene non completamente eliminate. Per esempio, la Figura 2.6 mostra la distribuzione di frequenza per una cifratura di Vigenère con parola chiave di lunghezza pari a 9. Si ottiene un certo miglioramento rispetto alla cifratura Playfair ma rimangono comunque considerevoli informazioni relative alla frequenza.

È istruttivo descrivere un metodo per la violazione di questa cifratura, che illustra alcuni dei principi matematici impiegati nell'analisi crittografica. Innanzitutto si supponga che l'estraneo ritenga che il testo cifrato sia stato crittografato utilizzando una sostituzione monoalfabetica o una cifratura Vigenère: è sufficiente un semplice test per determinarlo. Se è stata utilizzata la sostituzione monoalfabetica, le proprietà statistiche del testo cifrato dovranno essere le stesse del linguaggio in chiaro. Pertanto, facendo riferimento alla Figura 2.5, vi sarà una lettera cifrata con una frequenza relativa di circa il 12,7%, una con circa il 9,006% e così via. Se per l'analisi è disponibile un solo messaggio, non si può immaginare di avere una corrispondenza esatta fra questo piccolo campione e il profilo statistico del testo in chiaro. Ciononostante, se la corrispondenza è ragionevole, si può supporre che sia stata utilizzata una sostituzione monoalfabetica.

Se invece si sospetta l'impiego di una cifratura di Vigenère, come si vedrà i progressi dipendono dall'individuazione della lunghezza della parola chiave. Ci si concentri inizialmente sul modo per determinare la lunghezza della chiave. La considerazione che porta alla soluzione di questo problema è il seguente: se due sequenze identiche delle lettere in chiaro si presentano a una distanza che è un multiplo intero della lunghezza della parola chiave, genereranno sequenze cifrate identiche. Nell'esempio precedente, vi sono due istanze della sequenza "red" separate da 9 caratteri. Di conseguenza, in entrambi i casi, la lettera r

Tabella 2.3 La tabella di Vigenère.

Testo in chiaro	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z

Chiave

viene crittografata utilizzando la lettera della chiave *e*, la lettera e viene crittografata utilizzando la lettera chiave *p* e la lettera d viene crittografata utilizzando la lettera chiave *t*. Pertanto in entrambi i casi, la sequenza cifrata sarà **VTW**.

Un'analista che analizzasse il solo testo cifrato rileverebbe la sequenza ripetuta VTW a una distanza di nove posizioni e ipotizzerebbe che la parola chiave ha una lunghezza di tre o nove lettere. La doppia sequenza VTW potrebbe anche comparire casualmente e non riflettere la presenza di lettere uguali nel testo in chiaro codificate con le stesse lettere nella chiave. Tuttavia, se il messaggio è sufficientemente lungo, vi saranno varie ripetizioni di sequenze di testo. Osservando i fattori comuni nello sfalsamento delle varie sequenze, si può riuscire a indovinare la lunghezza della chiave.

La soluzione della cifratura dipende ora da un'importante considerazione. Se la lunghezza della parola chiave è *N*, la cifratura sarà in realtà costituita da *N* cifrature a sostituzione monoalfabetica. Per esempio, con la parola chiave DECEPTIVE, le lettere nelle posizioni 1, 10, 19 e così via, saranno tutte crittografate con la stessa cifratura monoalfabetica. Pertanto si possono usare le caratteristiche di frequenza del linguaggio in chiaro per attaccare separatamente le varie cifrature monoalfabetiche.

La natura periodica della parola chiave può essere eliminata utilizzando una parola chiave che non si ripete, lunga quanto il messaggio stesso. Vigenère ha proposto l'impiego di un sistema cosidetto **autokey** in cui la parola chiave viene concatenata con il testo in chiaro in modo da fornire una chiave continuamente variabile. Nel nostro caso:

```
chiave:     deceptivewearediscoveredsav
chiaro:     wearediscoveredsaveyourself
cifrato:    ZICVTWQNGKZEIIGASXSTSLVVWLA
```

Ma anche questo schema è vulnerabile all'analisi crittografica. Poiché le chiavi e il testo in chiaro condividono la stessa distribuzione di frequenza, si può applicare una tecnica statistica. Per esempio, la lettera e crittografata da *e* nella Figura 2.5 si presenterà con una frequenza pari a $(0{,}127)^2 \approx 0{,}016$ mentre la lettera t crittografata da *t* avrà una frequenza pari a circa la metà. Queste regolarità possono essere sfruttate per eseguire l'analisi crittografica.[8]

L'ultima difesa contro l'analisi crittografica è quella di scegliere una parola chiave lunga quanto il testo in chiaro e senza alcuna relazione statistica con esso. Tale sistema è stato introdotto nel 1918 da Gilbert Vernam, ingegnere della AT&T. Il suo sistema si basa però su dati binari anzichè sulle lettere. Il sistema può essere espresso in breve nel seguente modo:

$$c_i = p_i \oplus k_i$$

dove

p_i = i-esima cifra binaria del testo in chiaro
k_i = i-esima cifra binaria della chiave

[8] Sebbene le tecniche utilizzate per violare una cifratura di Vigenère non siano affatto complesse, un articolo del 1917 di Scientific American indicò questo sistema come "impossibile da violare". Una lezione da tenere sempre in considerazione quando si sostengono simili affermazioni per gli algoritmi moderni.

c_i = i-esima cifra binaria del testo cifrato
$\oplus$ = operatore di OR esclusivo (XOR)

Il testo cifrato viene quindi generato eseguendo l'operazione di XOR bit-a-bit fra il testo in chiaro e la chiave. Date le proprietà dell'operatore XOR, la decrittografia comporta la stessa operazione bit-a-bit:

$$p_i = c_i \oplus k_i$$

Questa tecnica si basa quindi sul metodo di costruzione della chiave. Vernam propose l'impiego di un nastro che alla fine ripete comunque la chiave: in questo modo il sistema funziona con una parola chiave molto lunga ma ripetuta. Sebbene uno schema di questo tipo, con una chiave molto lunga, presenti una difficoltà di analisi crittografica formidabile, è tuttavia violabile avendo a disposizione una quantità sufficiente di testo cifrato o utilizzando sequenze di testo in chiaro note o probabili.

One-Time Pad

Un ufficiale dell'esercito americano, Joseph Mauborgne, propose un miglioramento della cifratura di Vernam che garantisce la massima sicurezza. Mauborgne suggerì l'impiego di una chiave casuale che fosse lunga quanto il messaggio, in modo da non dover ripetere la chiave. La chiave, inoltre, deve essere utilizzata per crittografare e decrittografare un solo messaggio, ed essere poi scartata. Ogni nuovo messaggio richiede una nuova chiave lunga quanto il messaggio. Tale schema, chiamato **One-Time Pad**, è inviolabile in quanto produce un output casuale che non ha più alcuna relazione statistica con il testo in chiaro. Poiché il testo cifrato non contiene alcuna informazione sul testo in chiaro, non vi è alcun modo per violare questo codice. Ecco un esempio illustrativo. Si supponga di utilizzare uno schema di Vigenère con 27 caratteri in cui il 27-esimo carattere è lo spazio, ma con una chiave mono-uso lunga quanto il messaggio. La Tabella 2.3 deve essere quindi espansa a 27×27. Si consideri il testo cifrato:

```
ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS
```

Ecco ora due diverse decrittografie ottenute da due chiavi differenti:

```
cifrato:    ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS
chiave:     pxlmvmsydoftyrvzwc tnlebnecvgdupahfzzlmnyih
chiaro:     mr mustard with the candlestick in the hall

cifrato:    ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS
chiave:     mfugpmiydgaxgoufhkllmhsqdqogtewbqfgyovuhwt
chiaro:     miss scarlet with the knife in the library
```


Si supponga che un analista crittografico sia riuscito a trovare queste due chiavi, che generano due testi in chiaro ugualmente plausibili. In quale modo si può decidere qual è la decrittografia corretta (e dunque qual è la chiave corretta)? Se la chiave è stata prodotta in modo veramente casuale, l'analista crittografico non potrà stabilire quale di queste due chiavi è più probabile dell'altra. Pertanto non vi è modo per decidere quale chiave è corretta e pertanto quale testo in chiaro è corretto.

Infatti, dato qualsiasi testo in chiaro di lunghezza pari al testo cifrato, vi è una chiave che produce tale testo in chiaro. Pertanto, se si facesse una ricerca esaustiva di tutte le chiavi possibili, si otterrebbero più testi in chiaro leggibili senza alcun modo per capire quale era il testo in chiaro originario. Il codice è quindi inviolabile.

La sicurezza della tecnica One-Time Pad è dovuta interamente alla casualità della chiave. Se il flusso di caratteri che costituisce la chiave è veramente casuale, il flusso di caratteri che costituisce il testo cifrato sarà veramente casuale. Pertanto non vi saranno schemi o regolarità utilizzabili da un analista crittografico per attaccare il testo cifrato.

In teoria non si dovrebbero ricercare altri metodi di cifratura in quanto la tecnica One-Time Pad offre una sicurezza completa, ma in pratica presenta due difficoltà fondamentali.

1. Vi è il problema pratico di creare grandi quantità di chiavi casuali. Un sistema molto utilizzato potrebbe richiedere regolarmente milioni di caratteri casuali. La generazione di caratteri veramente casuali a questi volumi è un compito oneroso.
2. Ancora più grave è il problema della distribuzione e protezione della chiave. Per ogni messaggio inviato, occorre una chiave di pari lunghezza sia per il mittente che per il destinatario. Pertanto esiste un enorme problema di distribuzione delle chiavi.

Per questi motivi, la tecnica One-Time Pad ha un impiego limitato ed è utile principalmente per i canali a bassa ampiezza di banda che richiedono una sicurezza molto elevata.

2.3 Tecniche di trasposizione

Tutte le tecniche esaminate finora comportano la sostituzione di un simbolo di testo cifrato con un simbolo di testo in chiaro. Si può ottenere un mapping molto differente eseguendo una permutazione delle lettere del testo in chiaro. Questa tecnica è chiamata cifratura a trasposizione.

La tecnica più semplice è chiamata Rail Fence (letteralmente "staccionata") in cui il testo in chiaro viene scritto come una sequenza di diagonali e poi letta come una sequenza di righe. Per esempio, per crittografare il messaggio "meet me after the toga party" con una profondità pari a 2, si scrive:

```
m e m a t r h t g p r y
 e t e f e t e o a a t
```

Il messaggio crittografato sarà:

```
MEMATRHTGPRYETEFETEOAAT
```

Questo tipo di tecnica è estremamente facile da analizzare. Uno schema più complesso consiste nello scrivere il messaggio in un rettangolo, riga per riga e poi leggerlo colonna per colonna permutando l'ordine delle colonne. L'ordine delle colonne diviene quindi la chiave dell'algoritmo. Ecco un esempio:

```
Chiave:     4 3 1 2 5 6 7
Chiaro:     a t t a c k p
            o s t p o n e
            d u n t i l t
            w o a m x y z

Cifrato:    TTNAAPTMTSUOAODWCOIXKNLYPETZ
```

Una cifratura a trasposizione pura è facile da riconoscere poiché presenta la stessa frequenza delle lettere del testo in chiaro originario. Per il tipo di trasposizione a colonne appena illustrato, l'analisi crittografica è piuttosto semplice e prevede la disposizione del testo cifrato in una matrice, giocando poi sulla posizione delle colonne. Può essere utile anche impiegare tabelle di frequenze di digrammi e trigrammi.

La cifratura a trasposizione può essere resa significativamente più sicura eseguendo la trasposizione in più fasi. Il risultato è una permutazione più complessa che risulta difficile da ricostruire. Pertanto, se il messaggio precedente viene ri-crittografato utilizzando lo stesso algoritmo, si avrà:

```
Chiave:     4 3 1 2 5 6 7
Input:      t t n a a p t
            m t s u o a o
            d w c o i x k
            n l y p e t z

Output:     NSCYAUOPTTWLTMDNAOIEPAXTTOKZ
```

Per visualizzare il risultato di questa doppia trasposizione, si designano le lettere del messaggio in chiaro originario con i numeri che indicano la loro posizione. Pertanto, con un messaggio di 28 lettere, la sequenza di lettere originaria sarà:

```
01 02 03 04 05 06 07 08 09 10 11 12 13 14
15 16 17 18 19 20 21 22 23 24 25 26 27 28
```

Dopo la prima trasposizione si avrà:

```
03 10 17 24 04 11 18 25 02 09 16 23 01 08
15 22 05 12 19 26 06 13 20 27 07 14 21 28
```


che presenta una struttura ancora troppo regolare. Ma dopo la seconda trasposizione si avrà:

```
17 09 05 27 24 16 12 07 10 02 22 20 03 25
15 13 04 23 19 14 11 01 26 21 18 08 06 28
```

Questa è una permutazione molto meno strutturata e quindi più difficile da analizzare.

2.4 Macchine a rotazione

L'esempio appena fornito suggerisce il fatto che aumentando le fasi della crittografia si produce un algoritmo molto più difficile da analizzare. Questo vale sia per le cifrature a sostituzione che per le cifrature a trasposizione. Prima dell'introduzione di DES, l'applicazione più importante del principio delle fasi multiple di crittografia era una classe di sistemi nota con il nome di macchine a rotazione.[9]

Il principio di queste macchine è illustrato nella Figura 2.7. La macchina consiste in un insieme di cilindri rotanti indipendenti attraverso i quali possono passare degli impulsi elettrici. Ogni cilindro ha 26 terminali di ingresso e 26 terminali di uscita con cablaggi interni che connettono ciascun terminale di ingresso a un solo terminale di uscita. Per semplicità, sono illustrate solo tre delle connessioni interne di ciascun cilindro.

Se si associa a ciascun terminale di ingresso e di uscita una lettera dell'alfabeto, allora un cilindro definisce una sostituzione monoalfabetica. Per esempio, nella Figura 2.7, se un operatore preme il tasto della lettera A, viene applicato un segnale elettrico al primo ingresso del primo cilindro che produce un uscita al venticinquesimo terminale di uscita.

Si consideri una macchina con un unico cilindro. Dopo la pressione di ciascun tasto di input, il cilindro ruota di una posizione e dunque le connessioni interne vengono ruotate di conseguenza. Pertanto verrà impiegata un'altra sostituzione cifrata monoalfabetica. Dopo 26 lettere di testo in chiaro, il cilindro torna alla posizione iniziale: si ha dunque un algoritmo di sostituzione polialfabetica con periodo pari a 26.

Un sistema con un unico cilindro è molto semplice e non rappresenta un problema per l'analisi crittografica. La potenza della macchina a rotazione è costituita dalla presenza di più cilindri dove le uscite di un cilindro sono connesse agli ingressi del cilindro successivo. La Figura 2.7 mostra un sistema a tre cilindri. La prima metà della figura mostra una posizione in cui l'input dell'operatore sul primo ingresso (la lettera in chiaro A) passa attraverso i tre cilindri fuoriuscendo dalla seconda uscita del terzo cilindro (lettera cifrata B).
Quando si usano più cilindri, quello più vicino all'input dell'operatore ruota di una posizione a ogni input. La metà destra della Figura 2.7 mostra la configurazione del sistema dopo un singolo input. Per ogni rotazione completa del cilindro più interno, il cilindro intermedio ruota di una posizione. Infine, per ogni rotazione completa del cilindro centrale, il cilindro più esterno ruota di una posizione. In pratica questo sistema si comporta

[9] Macchine basate sul principio del rotore sono state utilizzate sia in Germania (Enigma) sia in Giappone (Purple) durante la Seconda Guerra Mondiale. La violazione di questi due codici da parte degli alleati ha contribuito significativamente al risultato della guerra.

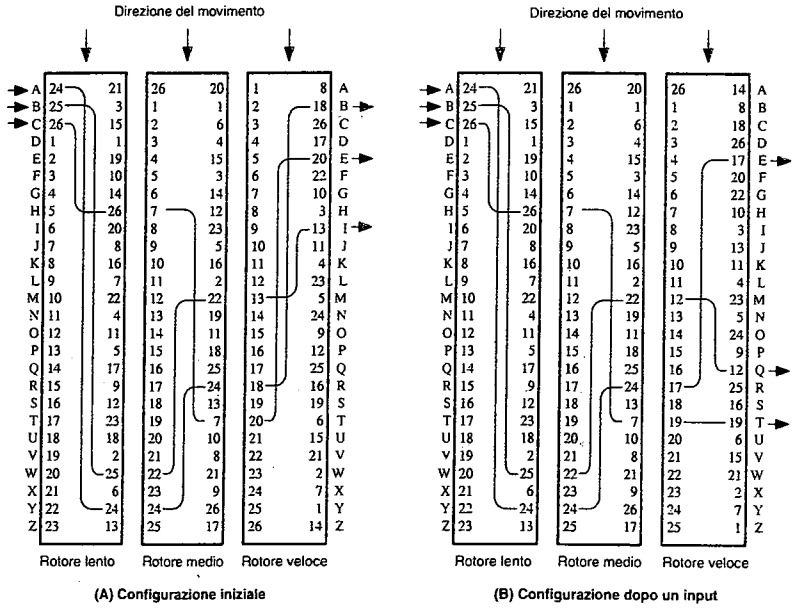


Figura 2.7 Una macchina a tre rotori dove i cablaggi sono rappresentati da contatti numerati.

come il contachilometri di un'automobile. Il risultato è che vi sono 26 × 26 × 26 = 17 576 diversi alfabeti di sostituzione prima che il sistema si ripeta. L'aggiunta di un quarto e di un quinto rotore porta rispettivamente a una ripetizione con periodo di 456 976 e 11 881 376. David Kahn ha riferito in modo eloquente a proposito della macchina a rotazione con 5 rotori [KAHN96, pagina 413]:

"Un periodo di questa lunghezza impedisce qualsiasi possibilità pratica di soluzione sulla base della frequenza delle lettere. Per ottenere una soluzione generale sarebbero necessarie circa 50 lettere per alfabeto cifrato, ovvero i cinque rotori dovrebbero svolgere il loro ciclo per 50 volte. Il testo cifrato dovrebbe essere lungo quanto tutti i discorsi tenuti al Senato e alla Casa dei Rappresentanti in tre sessioni successive del Congresso. Nessun analista crittografico sarebbe in grado di catturare un tale tesoro in tutta la sua vita; perfino i diplomatici, che possono essere prolissi quanto i politici, raramente raggiungono questi livelli di loquacità". La macchina a rotazione è importante in quanto apre la strada alla cifratura attualmente più utilizzata: l'algoritmo DES (Data Encryption Standard) che verrà descritto nel Capitolo 3.

2.5 Steganografia

Si concluderà il capitolo con una discussione di una tecnica che in realtà non è una vera e propria crittografia: la steganografia.

Un messaggio in chiaro può essere nascosto in due modi. I metodi della steganografia nascondono l'esistenza stessa del messaggio mentre i metodi di crittografia rendono il messaggio illeggibile da parte di estranei grazie a varie trasformazioni applicate al testo.[10]

Una forma di steganografia semplice ma laboriosa prevede la disposizione delle parole o delle lettere all'interno di un testo apparentemente innocuo. Per esempio, il messaggio nascosto può essere definito dalla sequenza delle prime lettere di ciascuna parola di un messaggio. La Figura 2.8 mostra un esempio in cui il messaggio nascosto è costituito da un sottoinsieme delle parole del messaggio.

3rd March

Dear George,

Greetings to all at Oxford. Many thanks for your letter and for the Summer examination package. All Entry Forms and Fees Forms should be ready for final despatch to the Syndicate by Friday 20th or at the very latest, I'm told, by the 21st. Admin has improved here, though there's room for improvement still; just give us all two or three more years and we'll really show you! Please don't let these wretched 16+ proposals destroy your basic O and A pattern. Certainly this sort of change, if implemented immediately, would bring chaos.

Sincerely yours,

Figura 2.8 Un quesito per l'ispettore Morse. FONTE: *The Silent World of Nicholas Quinn*, di Colin Dexter.

[10] "Steganografia" era una parola obsoleta rispolverata da David Kahn, che le ha assegnato l'attuale significato [KAHN96].

Storicamente sono state utilizzate varie altre tecniche fra cui le seguenti [MYER91].

- **Contrassegno dei caratteri:** alcune lettere del testo scritto o stampato vengono sovrascritte a matita. I segni non sono normalmente visibili a meno che la carta non venga mantenuta a una determinata angolazione rispetto ad una intensa sorgente luminosa.
- **Inchiostro invisibile:** varie sostanze consentono di scrivere senza lasciare alcuna traccia visibile se non applicando calore o una sostanza chimica.
- **Piccoli fori:** piccoli fori sulle lettere prescelte che non risultano visibili se non ponendo il foglio di carta davanti a una sorgente luminosa.
- **Nastro correttore per macchine per scrivere:** usato per scrivere il messaggio segreto fra le righe scritte con inchiostro in nero, lo rende visibile solo con una forte sorgente luminosa.

Queste tecniche, sebbene possano sembrare arcaiche, hanno degli equivalenti contemporanei. [WAYN93] propone di nascondere un messaggio nei bit meno significativi dei frame di un CD. Per esempio, la massima risoluzione del formato Kodak Photo-CD è di 2048 per 3072 pixel, ed ogni pixel contiene 24 bit di informazioni per i colori RGB. Il bit meno significativo di ciascun pixel di 24 bit può essere modificato senza alterare significativamente la qualità dell'immagine. In questo modo è possibile nascondere 2,3 MB di messaggi in un'unica fotografia digitale. Vi sono attualmente vari pacchetti software in grado di sfruttare questa tecnica steganografica.

La steganografia presenta numerosi difetti rispetto alla crittografia. Richiede un grosso impegno anche per nascondere un numero relativamente limitato di bit di informazioni, sebbene uno schema come quello proposto nel paragrafo precedente possa rendere più efficace questa soluzione. Inoltre il sistema, una volta scoperto, diviene praticamente inutile. Anche questo problema può però essere risolto se il metodo di inserimento dipende da un certo tipo di chiave (per un esempio vedere il Problema 2.11). Alternativamente un messaggio può essere prima crittografato e poi nascosto utilizzando la steganografia.

Il vantaggio della steganografia è che può essere impiegata da parti che non vogliono far sapere che sono in comunicazione. La crittografia invece segnala essa stessa che il traffico è importante o segreto o comunque che il mittente e il destinatario hanno qualcosa da nascondere.

2.6 Letture e siti Web consigliati

Tutti coloro che sono interessati alla storia della creazione e della violazione di codici, possono consultare [KAHN96]. Anche se riguarda più l'impatto della criptologia che il suo sviluppo tecnico, rappresenta comunque un'eccellente introduzione e una lettura interessante. Un altro eccellente resoconto storico è [SING99].

Una breve trattazione delle tecniche descritte in questo capitolo e di altre tecniche è [GARD72]. Vi sono vari volumi che descrivono la crittografia classica in modo più tecnico; uno dei migliori è [SINK66]. [KORN96] è un libro piacevole da leggere e contiene una lunga sezione dedicata alle tecniche classiche. Due volumi di crittografia che contengono una discreta quantità di materiale tecnico sulle tecniche classiche sono [GARR01] e [NICH99]. Per il lettore veramente interessato, il testo [NICH96] (in due volumi) descrive in dettaglio numerose cifrature classiche e fornisce vari testi cifrati da analizzare con le relative soluzioni.

Un'eccellente descrizione delle macchine a rotazione, compresa una discussione della loro analisi crittografica è [KUMA97].

[KATZ00] fornisce una trattazione approfondita della steganografia. Un'altra buona fonte è [WAYN96].

GARD72 M. Gardner. *Codes, Ciphers, and Secret Writing*. New York: Dover, 1972.
GARR01 P. Garrett. *Making, Breaking Codes: An Introduction to Cryptology*. Upper Saddle River, NJ: Prentice Hall, 2001.
KAHN96 D. Kahn. *The Codebreakers: The Story of Secret Writing*. New York: Scribner, 1996.
KATZ00 S. Katzenbeisser, ed. *Information Hiding Techniques for Steganography and Digital Watermarking*. Boston: Artech House, 2000.
KORN96 T. Korner. *The Pleasures of Counting*. Cambridge, England: Cambridge University Press, 1996.
KUMA97 I. Kumar. *Cryptology*. Laguna Hills, CA: Aegean Park Press, 1997.
NICH96 R. Nichols. *Classical Cryptography Course*. Laguna Hills, CA: Aegean Park Press, 1996.
NICH99 R. Nichols. ed. *ICSA Guide to Cryptography*. New York: McGraw-Hill, 1999.
SING99 S. Singh. *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*. New York: Anchor Books, 1999.
SINK66 A. Sinkov. *Elementary Cryptanalysis: A Mathematical Approach*. Washington, DC: The Mathematical Association of America, 1966.
WAYN96 P. Wayner. *Disappearing Cryptography*. Boston: AP Professional Books, 1996.

Siti Web consigliati

- **American Cryptogram Association:** un'associazione di crittografi amatoriali. Il sito Web comprende informazioni e link a siti riguardanti la crittografia classica.
- **Cripto Corner:** sito Web di Simon Singh. Notevole quantità di ottime informazioni e strumenti interattivi per comprendere la crittografia.
- **Steganografy:** Ottimo elenco di collegamenti e documenti.

2.7 Termini chiave, domande di ripasso e problemi

Termini chiave

Analisi crittografica
Attacco a forza bruta
Cifratura
Cifratura a trasposizione
Cifratura di Cesare
Cifratura di flusso
Cifratura di Hill
Cifratura di Vigenère

Cifratura monoalfabetica
Cifratura mono-chiave
Cifratura Playfair
Cifratura polialfabetica
Cifratura Rail Fence
Cifratura simmetrica
Computazionalmente sicuro
Criptologia
Crittografia
Crittografia a blocchi
Crittografia convenzionale
Decifratura
Decrittografia
Incondizionatamente sicuro
One-Time Pad
Sistema crittografico
Steganografia
Testo cifrato
Testo in chiaro

Domande di riepilogo

2.1 Quali sono gli elementi essenziali della cifratura simmetrica?
2.2 Quali sono le due funzioni di base utilizzate negli algoritmi di crittografia?
2.3 Quante chiavi sono necessarie perché due persone possano comunicare in modo cifrato?
2.4 Qual è la differenza fra la cifratura a blocchi e la cifratura a flussi?
2.5. Quali sono i due approcci generali per attaccare una cifratura?
2.6 Elencare e definire brevemente i vari tipi di attacchi ad analisi crittografica sulla base delle informazioni note.
2.7 Qual è la differenza fra una cifratura incondizionatamente sicura e una cifratura computazionalmente sicura?
2.8 Definire brevemente la cifratura di Cesare.
2.9 Definire brevemente la cifratura monoalfabetica.
2.10 Definire brevemente la cifratura Playfair.
2.11 Qual è la differenza fra cifratura monoalfabetica e polialfabetica?
2.12 Quali sono i due problemi della tecnica One-Time Pad?
2.13 Che cos'è la cifratura a trasposizione?
2.14 Che cos'è la steganografia?

Problemi

2.1 Una generalizzazione della cifratura di Cesare, nota come cifratura di Cesare affine, ha la formulazione seguente: per ogni lettera p del testo in chiaro, sostituire la lettera del testo cifrato C così calcolata:

$$C = \mathrm{E}([a, b], p) = (ap + b) \bmod 26$$

Un requisito di base di qualsiasi algoritmo di cifratura è che sia di tipo uno-a-uno. Ovvero, se $p \neq q$, allora $\mathrm{E}(k, p) \neq \mathrm{E}(k, q)$. In caso contrario la decifratura risulterebbe impossibile, perché più di un carattere del testo in chiaro verrebbe associato al medesimo carattere del testo cifrato. La cifratura di Cesare affine non è uno-a-uno per tutti i valori di a. Per esempio, per $a = 2$ e $b = 3$ si ha $\mathrm{E}([a, b], 0) = \mathrm{E}([a, b], 13) = 3$.

A. Esistono limitazioni nel valore di b? Spiegare il motivo.
B. Determinare i valori di a che non sono ammessi.
C. Fornire una descrizione generale di quali valori di a sono ammessi e quali non ammessi. Giustificare.

2.2 Quante cifrature di Cesare affini uno-a-uno esistono?

2.3 È stato generato un testo cifrato utilizzando una cifratura affine. La lettera più frequente nel testo cifrato è la "B", mentre la seconda lettera più frequente nel testo cifrato è la "U". Violare il codice.

2.4 Decrittografare il seguente testo cifrato, che è stato generato utilizzando un semplice algoritmo a sostituzione:

```
53‡‡†305))6*;4826)4‡.)4‡);806*;48†8¶60))85;;)8*;:‡*8†83
(88)5*†;46(;88*96*?;8)*‡(;485);5*†2:*‡(;4956*2(5*-4)8¶8*
;4069285);)6†8)4‡‡;1(‡9;48081;8:8‡1;48†85;4)485†528806*81
(‡9;48;(88;4(‡?34;48)4‡;161;:188;‡?;
```

Suggerimenti

1. Come è noto, la lettera più frequente in inglese è la "e". Pertanto la prima, la seconda (o forse la terza) lettera più comune nel messaggio è probabilmente questa lettera. Inoltre in molte parole inglesi la lettera "e" si presenta a coppie (per esempio meet, fleet, speed, seen, been, agree). Si tenti dunque di trovare nel testo cifrato il carattere corrispondente alla lettera e.
2. La parola più comune in inglese è "the". Utilizzare questo fatto per identificare i caratteri che rappresentano le lettere "t" e "h".
3. Decifrare la parte rimanente del messaggio cercando di indovinare altre parole.

Attenzione: il messaggio originale è in inglese ma potrebbe non avere molto senso a prima vista.

2.5 Un modo per risolvere il problema della distribuzione della chiave è quello di utilizzare una riga di un libro in possesso del mittente e del destinatario. Solitamente, quanto meno nei racconti di spionaggio, viene utilizzata come chiave la prima frase di un libro. Lo schema discusso in questo problema è tratto da uno dei migliori racconti riguardante i codici segreti, *Talking to Strange Men* di Ruth Rendell. Si consiglia di risolvere il problema senza consultare il volume.
Si consideri il seguente messaggio:

```
SIOKHKDM AF HCRKIABIE SHIMC KD LFEAILA
```

Questo testo cifrato è stato prodotto utilizzando la prima frase di *The Other Side of Silence* (un libro sulla spia Kim Philby):

```
The snow lay thick on the steps and the snowflakes driven by the wind
looked black in the headlights of the cars.
```

È stata utilizzata una semplice cifratura a sostituzione.
A. Quale algoritmo di crittografia è stato utilizzato?
B. Quanto è sicuro?

C. Per semplificare il problema della distribuzione della chiave, entrambe le parti si accordano sull'uso della prima o dell'ultima frase di un libro come chiave. Per cambiare la chiave, basta accordarsi su un nuovo libro. È preferibile utilizzare la prima frase rispetto all'ultima. Perché?

2.6 In uno dei suoi casi, Sherlock Holmes ha trovato il seguente messaggio:

534 C2 13 127 36 31 4 17 21 41
DOUGLAS 109 293 5 37 BIRLSTONE
26 BIRLSTONE 9 127 171

Watson sembrava confuso, mentre Holmes è riuscito immediatamente a indovinare il tipo di cifratura. Si è in grado di farlo?

2.7 Questo problema utilizza un esempio tratto da un vecchio manuale (di pubblico dominio) delle Forze Speciali statunitensi. Una copia è disponibile a: **ftp://shell.shore.net/members/w/s/ws/Support/Crypto/FM-31-4.pdf**.

A. Utilizzando le due chiavi (parole da ritenere a memoria) *cryptographic* e *network security*, crittografare il seguente messaggio: "Trovati alla terza colonna da sinistra all'esterno del teatro del liceo questa sera alle sette. Se non ti fidi vieni con due amici."
Fare ipotesi ragionevoli, da esplicitare, su come trattare le lettere ridondanti e le lettere in eccesso nelle parole chiave, e su come trattare gli spazi e i segni di punteggiatura. Nota: il messaggio è tratto dal racconto di Sherlock Holmes "*The Sign of Four*".

B. Decrittografare il messaggio, indicando i passaggi.

C. Indicare i casi in cui la tecnica risulta appropriata elencandone i vantaggi.

2.8 Uno svantaggio della cifratura monoalfabetica generale è il fatto che il mittente e il destinatario devono memorizzare la sequenza di cifre da permutare. Una soluzione comunemente adottata è quella di utilizzare una parola chiave dalla quale generare la sequenza di cifratura. Per esempio, utilizzando la parola chiave *CIPHER*, scrivere la parola chiave seguita dalle lettere non utilizzate in ordine normale e confrontarla con le lettere in chiaro:

chiaro: a b c d e f g h i j k l m n o p q r s t u v w x y z
cifrato: C I P H E R A B D F G J K L M N O Q S T U V W X Y Z

Se si pensa che questa operazione non produca una miscelazione sufficiente, scrivere le lettere rimanenti su righe successive e generare la frase leggendo le colonne:

C I P H E R
A B D F G J
K L M N O Q
S T U V W X
Y Z

Verrà prodotta la frase:

C A K S Y I B L T Z P D M U H F N V E G O W R J O X

Tale sistema viene utilizzato nell'esempio presentato nel paragrafo 2.2 (che inizia con "it was disclosed yesterday"). Determinare la parola chiave.

● 2.9 Quando la vedetta americana PT-109, sotto il comando del tenente John F. Kennedy, venne affondata da un cacciatorpediniere giapponese, una stazione australiana ricevette il seguente codice Playfair:

KXJEY UREBE ZWEHE WRYTU HEYFS
KREHE GOYFI WTTTU OLKSY CAJPO
BOTEI ZONTX BYBNT GONEY CUZWR
GDSON SXBOU YWRHE BAAHY USEDO

La chiave utilizzata era *royal new zealand navy*. Decrittografare il messaggio (tradurre TT con tt).

2.10 A. Costruire una matrice Playfair con la chiave *largest*.

B. Costruire una matrice Palyfair con la chiave *occurrence*. Fare ipotesi ragionevoli su come trattare le lettere ridondanti nella chiave.

●2.11 A. Utilizzando la seguente matrice Playfair, crittografare il messaggio: "Must see you over Cadogan West. Coming at once." Nota: Il messaggio è tratto dal racconto di Sherlock Holmes "*The Adventure of the Bruce-Partington Plans*".

M	F	H	I/J	K
U	N	O	P	Q
Z	V	W	X	Y
E	L	A	R	G
D	S	T	B	C

B. Ripetere la parte (A) utilizzando la matrice Playfair del Problema 2.10 A.

C. Giustificare i risultati di questo problema.

2.12 A. Quante possibili chiavi ha l'algoritmo di cifratura Playfair? Ignorare il fatto che alcune chiavi potrebbero produrre lo stesso testo cifrato. Esprimere la risposta come una potenza approssimata di 2.

B. Tenere ora in considerazione il fatto che alcune chiavi Playfair potrebbero produrre lo stesso testo cifrato. Quante possibili chiavi univoche ha l'algoritmo di cifratura Playfair?

2.13 Che sistema di sostituzione si ottiene quando si utilizza una matrice Playfair 25 × 1?

●2.14 A. Decifrare il messaggio YITJP GWJOW FAQTQ XCSMA ETSQU SQAPU SQGKC PQTYJ utilizzando la cifratura di Hill con chiave inversa $\begin{pmatrix} 5 & 1 \\ 2 & 7 \end{pmatrix}$. Illustrare i calcoli e il risultato.

B. Decifrare il messaggio MWALO LIAIW WTGBH JNTAK QZJKA ADAWS SKQKU AYARN CSODN IIAES OQKJY B utilizzando la cifratura di Hill con chiave inversa $\begin{pmatrix} 2 & 23 \\ 21 & 7 \end{pmatrix}$. Illustrare i calcoli e il risultato.

2.15 A. Crittografare il messaggio "meet me at the usual place at ten rather than eight oclock" utilizzando la cifratura di Hill con la chiave $\begin{pmatrix} 9 & 4 \\ 5 & 7 \end{pmatrix}$. Mostrare i calcoli e il risultato.

B. Mostrare i calcoli per decrittografare il testo cifrato e recuperare il testo in chiaro.

2.16 Si è visto che la cifratura di Hill soccombe a un attacco con testo in chiaro noto nel caso in cui si abbiano a disposizione un numero sufficiente di coppie testo in chiaro/testo cifrato. È ancora più facile risolvere la cifratura di Hill se può essere organizzato un attacco a testo in chiaro scelto. Descrivere questo tipo di attacco.

2.17 Si può dimostrare che la cifratura di Hill con la matrice $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ richiede che $(ad - bc)$ sia primo relativo di 26, ovvero che il solo fattore comune positivo di $(ad - bc)$ e 26 sia 1. Quindi la matrice non sarebbe ammessa se $(ad - bc)$ fosse uguale a 13 o fosse pari. Determinare il numero di possibili chiavi valide per la cifratura di Hill 2×2 senza enumerarle, procedendo come segue:

A. Trovare il numero di matrici il cui determinante è pari perché una o entrambe le righe sono pari (una riga si dice "pari" quando entrambi i suoi elementi sono pari).

B. Trovare il numero di matrici il cui determinante è pari perché una o entrambe le colonne sono pari (una colonna si dice "pari" quando entrambi i suoi elementi sono pari).

C. Trovare il numero di matrici il cui determinante è pari perché tutti i suoi elementi sono dispari.

D. Prendendo in considerazione le sovrapposizioni, trovare il numero totale di matrici il cui determinante è pari.

E. Trovare il numero di matrici il cui determinante è un multiplo di 13 perché la prima colonna è un multiplo di 13.

F. Trovare il numero di matrici il cui determinante è un multiplo di 13, nelle quali la prima colonna non è un multiplo di 13 ma la seconda è un multiplo della prima modulo 13.

G. Trovare il numero totale di matrici il cui determinante è un multiplo di 13.

H. Trovare il numero di matrici il cui determinante è un multiplo di 26 perché soddisfano i casi A ed E; B ed E; C ed E; A ed F ecc.

I. Trovare il numero totale di matrici il cui determinante non è né un multiplo di 2 né un multiplo di 13.

2.18 Utilizzando la cifratura di Vigenère, crittografare la parola "explanation" utilizzando la chiave *leg*.

2.19 Questo problema esplora l'impiego di una versione One-Time Pad della cifratura di Vigenère. In questo schema, la chiave è un flusso di numeri casuali compresi fra 0 e 26. Per esempio, se la chiave è 3 19 5 ..., la prima lettera del testo in chiaro viene crittografata con uno scorrimento di 3 lettere, la seconda con uno scorrimento di 19 lettere, la terza con uno scorrimento di 5 lettere e così via.

A. Crittografare il testo in chiaro sendmoremoney ("invia altro denaro") con il flusso di chiavi 9 0 1 7 23 15 21 14 11 11 2 8 9.

B. Utilizzare il testo cifrato prodotto nella parte A, per trovare una chiave in modo che il testo cifrato venga decrittografato nel testo in chiaro cashnotneeded ("denaro non necessario").

2.20 Qual è il messaggio nascosto nella Figura 2.8?

2.21 In uno dei misteri di Dorothy Sayers, Lord Peter trova il messaggio riportato nella Figura 2.9. Scopre inoltre la chiave del messaggio che è la sequenza di interi:

78765654343211234345656787887876565 4
34321123434565678788787656544332112 34

I thought to see the fairies in the fields, but I saw only the evil elephants with their black backs. Woe! how that sight awed me! The elves danced all around and about while I heard voices calling clearly. Ah! how I tried to see—throw off the ugly cloud—but no blind eye of a mortal was permitted to spy them. So then came minstrels, having gold trumpets, harps and drums. These played very loudly beside me, breaking that spell. So the dream vanished, whereat I thanked Heaven. I shed many tears before the thin moon rose up, frail and faint as a sickle of straw. Now though the Enchanter gnash his teeth vainly, yet shall he return as the Spring returns. Oh, wretched man! Hell gapes, Erebus now lies open. The mouths of Death wait on thy end.

Figura 2.9 L'enigma di Lord Peter.

A. Decrittografare il messaggio. *Suggerimento:* qual è il più grande valore intero?

B. Se è noto l'algoritmo ma non la chiave, quanto è sicuro lo schema?

C. Se è nota la chiave ma non l'algoritmo, quanto è sicuro lo schema?

Problemi di programmazione

2.22 Scrivere un programma di crittografia/decrittografia che utilizza la cifratura generale di Cesare, nota anche come cifratura additiva.

2.23 Scrivere un programma di crittografia/decrittografia che utilizza la cifratura affine descritta nel Problema 2.1.

2.24 Scrivere un programma capace di lanciare un attacco a frequenza di caratteri su una cifratura additiva senza intervento umano. Il software dovrebbe produrre i possibili

testi in chiaro in ordine di plausibilità. L'interfaccia dovrebbe poter consentire all'utente di specificare: "Determina i 10 testi in chiaro più probabili".

2.25 Scrivere un programma capace di lanciare un attacco a frequenza di caratteri su una cifratura a sostituzione monoalfabetica senza intervento umano. Il software dovrebbe produrre i possibili testi in chiaro in ordine di plausibilità. L'interfaccia dovrebbe poter consentire all'utente di specificare: "Determina i 10 testi in chiaro più probabili".

2.26 Scrivere un programma di crittografia/decrittografia che utilizza la cifratura di Hill 2×2.

2.27 Scrivere un programma capace di lanciare un attacco a testo in chiaro conosciuto su una cifratura di Hill, data la dimensione m. Qual è la velocità di esecuzione dell'algoritmo, come funzione di m?

Capitolo 3

Cifratura a blocchi e algoritmo DES

Concetti essenziali

- La **cifratura a blocchi** è uno schema di crittografia/decrittografia nel quale ciascun blocco di testo in chiaro viene elaborato in modo indipendente per produrre un blocco di testo cifrato di identiche dimensioni.
- Molte cifrature a blocchi hanno una struttura cosiddetta di **Feistel**. Tale struttura consiste in una serie di fasi di elaborazione identiche. In ciascuna fase viene effettuata una sostituzione su una metà dei dati da elaborare; questa viene seguita da una permutazione che scambia le due metà. La chiave originale viene espansa in modo da utilizzare chiavi differenti in ogni fase.
- Il **DES** (Data Encryption Standard) è stato l'algoritmo di crittografia più utilizzato fino a poco tempo fa. Possiede la classica struttura di Feistel, utilizza blocchi di 64 bit e una chiave di 56 bit.
- Due metodi di analisi crittografica fondamentali sono l'analisi crittografica **differenziale** e la **lineare**. DES è stato dimostrato essere particolarmente resistente a questi due tipi di attacco.

L'obiettivo di questo capitolo è quello di illustrare i principi delle moderne tecniche di cifratura simmetrica, trattando in particolare la cifratura simmetrica più utilizzata: l'algoritmo DES (Data Encryption Standard). DES rimane il più importante di questi algoritmi sebbene siano state sviluppate altre cifrature simmetriche successivamente alla sua introduzione e sebbene sia destinato a essere sostituito da AES (Advanced Encryption Standard). Inoltre lo studio dettagliato di DES consente di comprendere i principi utilizzati in altre cifrature simmetriche. Alcune di queste, fra cui AES, saranno trattate nei Capitoli 5 e 6.

Questo capitolo inizia con la discussione dei principi generali delle cifrature simmetriche a blocchi, ovvero delle tecniche di cifratura simmetrica studiate in questo volume (ad eccezione della cifratura a flussi RC4 trattata nel Capitolo 6). Quindi si affronterà lo studio di

DES. Dopo aver esaminato questo algoritmo specifico, si tornerà a una discussione più generale sulla progettazione delle cifrature a blocchi.

Rispetto alle cifrature a chiave pubblica come RSA, la struttura di DES e della maggior parte delle cifrature simmetriche è molto complessa e non può essere descritta con facilità come RSA e altri algoritmi simili. Il lettore potrà iniziare con una versione semplificata di DES, descritta nell'Appendice C. Questa consentirà al lettore di svolgere manualmente le operazioni di crittografia e decrittografia e di apprendere i dettagli del funzionamento di questi algoritmi. Per esperienza, lo studio di questa versione semplificata migliora la conoscenza di DES.[1]

3.1 Principi della cifratura a blocchi

La maggior parte degli algoritmi di crittografia simmetrici a blocchi attualmente utilizzati si basa su una struttura chiamata cifratura a blocchi di Feistel [FEIS73]. Per questo motivo è importante esaminare i principi progettuali di questa cifratura. Si inizierà con un confronto fra cifratura a flussi e cifratura a blocchi. Poi si motiverà la struttura della cifratura a blocchi di Feistel. Infine si discuteranno alcune delle sue implicazioni.

Cifratura a flussi e cifratura a blocchi

La **cifratura a flussi** crittografa un flusso digitale di dati un bit o un byte alla volta. Fra gli esempi classici di cifratura a flussi vi sono la cifratura Vigenère e la cifratura Vernam. Una **cifratura a blocchi** prevede invece che un blocco di testo in chiaro venga trattato come un'entità e utilizzato per produrre un blocco di testo cifrato di uguale lunghezza. In genere vengono utilizzati blocchi di 64 o 128 bit. Utilizzando alcune delle modalità operative descritte nel Capitolo 6, si può utilizzare una cifratura a blocchi anche per ottenere lo stesso effetto di una cifratura a flussi.

L'analisi della cifratura a blocchi è molto più sviluppata. In generale, questa tecnica sembra applicabile a una gamma di applicazioni molto più ampia rispetto alla cifratura a flussi. La maggior parte delle applicazioni crittografiche simmetriche per reti utilizza la cifratura a blocchi. Di conseguenza, in questo capitolo e in generale nel volume ogni volta che si parla di crittografia simmetrica, ci si concentrerà sulla cifratura a blocchi.

Le motivazioni della struttura di Feistel

Una cifratura a blocchi opera su un blocco di testo in chiaro di n bit per produrre un blocco di testo cifrato di n bit. Esistono 2^n possibili blocchi di testo in chiaro distinti e pertanto, affinché la crittografia sia reversibile (ovvero perché sia possibile eseguire la decrittografia),

[1] È anche possibile ignorare, quanto meno a una prima lettura, l'Appendice C. Se successivamente i dettagli di funzionamento di DES dovessero risultare troppo complicati, si potrà studiare il funzionamento di questa versione semplificata.

ogni blocco di testo in chiaro deve produrre un blocco di testo cifrato univoco. In questo caso si parla di trasformazione reversibile o non singolare. I seguenti esempi illustrano una trasformazione singolare e una non singolare per $n = 2$.

Mapping reversibile		Mapping irreversibile	
Testo in chiaro	Testo cifrato	Testo in chiaro	Testo cifrato
00	11	00	11
01	10	01	10
10	00	10	01
11	01	11	01

Nell'ultimo caso, il testo cifrato 01 potrebbe essere stato prodotto da due diversi blocchi di testo in chiaro. Quindi se ci si limita al mapping irreversibile, il numero di trasformazioni è pari a 2^n!

La Figura 3.1 illustra la logica di una cifratura generale a sostituzione per $n = 4$. Un input di 4 bit produce uno fra 16 possibili stati di input che può essere mappato univocamente in uno fra 16 possibili stati di output, ognuno dei quali è rappresentato da 4 bit di testo cifrato. Le associazioni fra crittografia e decrittografia possono essere definite da una tabella, come quella rappresentata nella Tabella 3.1. Questa è la forma più generale di cifratura a blocchi e può essere utilizzata per definire qualsiasi mapping reversibile fra testo in chia-

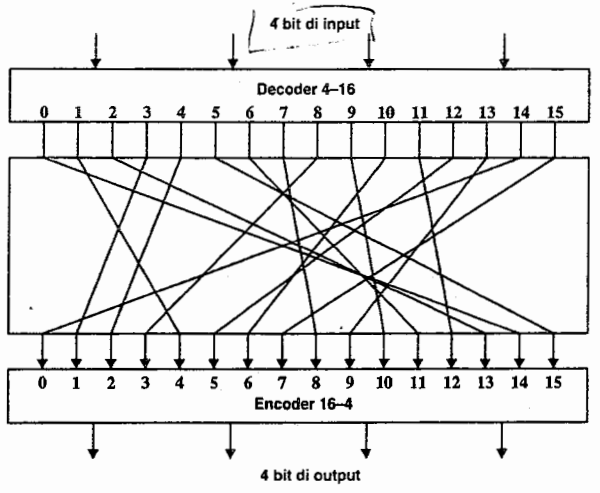


Figura 3.1 Sostituzione generale a blocchi tra *n* bit e *n* bit (rappresentata con $n = 4$).

ro e testo cifrato. Feistel chiama questa tecnica cifratura ideale a blocchi, perché consente il numero massimo di possibili mapping crittografici dal blocco di testo in chiaro [FEIS75].

La cifratura ideale a blocchi presenta però un problema pratico. Se viene utilizzato un blocco di piccole dimensioni, come $n = 4$, il sistema è equivalente a una classica cifratura a sostituzione. Tali sistemi, come si è già visto, sono vulnerabili all'analisi statistica del testo in chiaro. Questo punto debole non è dovuto all'uso della cifratura a sostituzione ma piuttosto all'uso di blocchi di dimensioni troppo piccole. Se n è sufficientemente grande ed è possibile utilizzare una sostituzione reversibile e arbitraria fra il testo in chiaro e il testo cifrato, le caratteristiche statistiche del testo in chiaro di origine verranno mascherate, al punto da rendere inapplicabile questo tipo di analisi crittografica.

Una cifratura a sostituzione reversibile arbitraria (la cifratura ideale a blocchi) per blocchi di grandi dimensioni non è però conveniente dal punto di vista dell'implementazione e delle prestazioni. Per una trasformazione di questo tipo, il mapping stesso costituisce la chiave. Si consideri ancora la Tabella 3.1 che definisce un particolare mapping reversibile

Tabella 3.1 Tabelle di crittografia e decrittografia per la cifratura a sostituzione rappresentata nella Figura 3.1.

Testo in chiaro	Testo cifrato	Testo in chiaro	Testo cifrato
0000	1110	0000	1110
0001	0100	0001	0011
0010	1101	0010	0100
0011	0001	0011	1000
0100	0010	0100	0001
0101	1111	0101	1100
0110	1011	0110	1010
0111	1000	0111	1111
1000	0011	1000	0111
1001	1010	1001	1101
1010	0110	1010	1001
1011	1100	1011	0110
1100	0101	1100	1011
1101	1001	1101	0010
1110	0000	1110	0000
1111	0111	1111	0101

dal testo in chiaro al testo cifrato per $n = 4$. Il mapping può essere definito dalle voci nella seconda colonna che mostrano il valore del testo cifrato per ciascun blocco di testo in chiaro. Questa, in pratica, è la chiave che determina lo specifico mapping fra tutti i possibili. In questo caso, utilizzando questo metodo diretto per definire la chiave, la lunghezza totale della chiave richiesta è (4 bit) × (16 righe) = 64 bit. In generale, per una cifratura ideale a blocchi di n bit, la chiave ha dimensioni pari a $n \times 2^n$. Per un blocco di 64 bit, ovvero di dimensioni opportune per impedire attacchi statistici, la chiave richiesta avrà dimensioni pari a $64 \times 2^{64} = 2^{70} \approx 10^{21}$ bit.

Considerando queste difficoltà, Feistel sostiene che è necessaria un'approssimazione a questo sistema ideale di cifratura a blocchi per n di elevate dimensioni, costituita da componenti facilmente realizzabili [FEIS75]. Ma prima di discutere l'approccio di Feistel, occorre fare un'altra osservazione. Si potrebbe utilizzare la cifratura generale per sostituzione a blocchi ma, per rendere possibile la sua implementazione, limitarsi a un sottoinsieme dei $2^n!$ possibili mapping reversibili. Per esempio, si supponga di definire il mapping in termini di un insieme di equazioni lineari. Nel caso di $n = 4$ si avrà:

$$
\begin{aligned}
y_1 &= k_{11}x_1 + k_{12}x_2 + k_{13}x_3 + k_{14}x_4 \\
y_2 &= k_{21}x_1 + k_{22}x_2 + k_{23}x_3 + k_{24}x_4 \\
y_3 &= k_{31}x_1 + k_{32}x_2 + k_{33}x_3 + k_{34}x_4 \\
y_4 &= k_{41}x_1 + k_{42}x_2 + k_{43}x_3 + k_{44}x_4
\end{aligned}
$$

dove le x_i sono le 4 cifre binarie del blocco di testo in chiaro, le y_i sono le 4 cifre binarie del blocco di testo cifrato, i k_{ij} sono i coefficienti binari e l'aritmetica utilizzata è modulo 2. Le dimensioni della chiave sono solo n^2, in questo caso 16 bit. Il pericolo con questo tipo di formulazione è che può essere vulnerabile all'analisi crittografica, se si conosce la struttura dell'algoritmo. In questo esempio, si ha sostanzialmente la cifratura Hill trattata nel Capitolo 2, applicata a dati binari invece che a caratteri. Come si è visto nel capitolo precedente, un semplice sistema lineare come questo è piuttosto vulnerabile.

La cifratura di Feistel

Feistel ha suggerito [FEIS73] di approssimare la semplice cifratura a sostituzione utilizzando il concetto di cifratura prodotto. Questa consiste nell'esecuzione di due o più cifrature di base in sequenza, in modo che il risultato finale, o prodotto, sia crittograficamente più resistente di qualsiasi cifratura componente. Il metodo consiste sostanzialmente nello sviluppare una cifratura a blocchi con lunghezza della chiave di k bit e lunghezza di blocco di n bit, consentendo un totale di 2^k possibili trasformazioni, invece delle $2^n!$ possibili con la cifratura ideale a blocchi. In particolare, Feistel ha proposto l'uso di una cifratura che alterni sostituzioni e permutazioni. In realtà si tratta di un'applicazione pratica di una proposta di Claude Shannon riguardante una cifratura che alterna funzioni di *confusione* e *diffusione* [SHAN49]. Si tratteranno ora questi due concetti per poi presentare la cifratura di Feistel. Vale la pena innanzitutto notare che: la struttura della cifratura di Feistel, che risale a circa un quarto di secolo fa e che, a sua volta, si basa su una proposta di Shannon

del 1945, è utilizzata dalle più importanti cifrature simmetriche a blocchi attualmente impiegate.

Diffusione e confusione

I termini *diffusione* e *confusione* sono stati introdotti da Claude Shannon per catturare i due concetti fondamentali alla base di qualsiasi sistema crittografico [SHAN49].[2] Lo scopo di Shannon era quello di ostacolare l'analisi crittografica basata su analisi statistiche. Il ragionamento è il seguente. Si supponga che un estraneo conosca alcune caratteristiche statistiche del testo in chiaro. Per esempio, in un messaggio leggibile in una certa lingua, potrebbe essere nota la distribuzione della frequenza delle varie lettere o potrebbero esservi determinate frasi o parole con elevata probabilità di apparire nel messaggio. Se queste informazioni statistiche venissero in qualche modo riflesse nel testo cifrato, l'analista crittografico potrebbe essere in grado di individuare la chiave di crittografia, una parte di questa chiave o quanto meno individuare un insieme di chiavi che potrebbero contenere la chiave esatta. In ciò che Shannon chiama una cifratura ideale, tutti i valori statistici del testo cifrato sono indipendenti dalla chiave utilizzata. La cifratura a sostituzione arbitraria di cui si è parlato in precedenza (Figura 3.1) è una cifratura di questo tipo ma, come si è visto, è poco pratica.

Se si esclude il ricorso a sistemi ideali, Shannon suggerisce due metodi per complicare l'analisi crittografica statistica: la diffusione e la confusione. Nella **diffusione**, la struttura statistica del testo in chiaro viene espansa in un ampio intervallo statistico del testo cifrato. Questo viene ottenuto facendo in modo che ogni cifra del testo in chiaro influenzi il valore di cifre del testo cifrato; questo è generalmente equivalente a fare in modo che ogni cifra del testo cifrato sia influenzata da più cifre del testo in chiaro. Un esempio di diffusione consiste nella crittografia di un messaggio $M = m_1, m_2, m_3, \ldots$ di caratteri con un'operazione di media:

$$y_n = \left(\sum_{i=1}^{k} m_{n+i} \right) \bmod 26$$

aggiungendo k lettere successive per ottenere una lettera y_n di testo cifrato. Si può dimostrare che in questo modo si annulla la struttura statistica del testo in chiaro. Pertanto le frequenze delle lettere nel testo cifrato saranno più simili tra loro di quelle del testo in chiaro; analogamente la frequenza dei digrammi risulterà più uniforme e così via. In una cifratura a blocchi binari, la diffusione può essere ottenuta eseguendo ripetutamente una permutazione sui dati, seguita dall'applicazione di una funzione; l'effetto è che più bit che si trovano in posizioni differenti del testo in chiaro contribuiscono alla definizione di uno stesso bit del testo cifrato.[3]

[2] L'articolo di Shannon del 1949 è apparso inizialmente come rapporto segreto nel 1945. Shannon gode di una posizione unica nella storia dei computer e delle scienze dell'informazione. Non solo ha sviluppato le idee di base della crittografia moderna ma ha anche inventato la teoria delle informazioni. Inoltre ha fondato un'altra disciplina, (con la sua tesi di master) l'applicazione dell'algebra booleana allo studio dei circuiti digitali.

[3] Alcuni volumi sulla crittografia considerano equivalenti la permutazione e la diffusione; si tratta di un errore. La permutazione, così com'è, non cambia le caratteristiche statistiche del testo in chiaro a livello delle singole lettere o dei blocchi permutati. Per esempio, in DES, la permutazione scambia due blocchi di 32 bit in modo che le statistiche delle stringhe di 32 bit o minori vengano mantenute.

Qualsiasi cifratura a blocchi comporta la trasformazione di un blocco di testo in chiaro in un blocco di testo cifrato, dove la trasformazione dipende dalla chiave. Il meccanismo di diffusione cerca di complicare il più possibile ogni relazione statistica fra il testo in chiaro e il testo cifrato in modo da complicare l'individuazione della chiave. La **confusione** cerca anche di complicare il più possibile la relazione fra le statistiche del testo cifrato e il valore della chiave di crittografia sempre nel tentativo di complicare l'individuazione della chiave. Pertanto, anche se un estraneo dovesse avere qualche informazione statistica sul testo cifrato, il modo in cui la chiave è stata utilizzata per produrre tale testo cifrato è talmente complesso da complicarne l'individuazione. Ciò si ottiene utilizzando un algoritmo di sostituzione complesso. Al contrario una semplice funzione di sostituzione lineare aggiungerebbe poca confusione.

Come evidenziato da [ROBS95b], i concetti di diffusione e di confusione hanno avuto così successo nel catturare l'essenza degli attributi richiesti a una cifratura a blocchi che sono diventati gli elementi di base della progettazione delle moderne cifrature a blocchi.

La struttura della cifratura di Feistel

La Figura 3.2 rappresenta la struttura proposta da Feistel. Gli input dell'algoritmo di crittografia sono costituiti da un blocco di testo in chiaro di lunghezza pari a $2w$ bit e da una chiave K. Il blocco di testo in chiaro è diviso in due metà, L_0 e R_0. Le due metà attraversano n fasi di elaborazione e poi si combinano per produrre il blocco di testo cifrato. Ciascuna fase i ha come input L_{i-1} e R_{i-1} che derivano dalla fase precedente e una sotto chiave K_i che deriva dalla chiave generale K. In generale le sottochiavi K_i sono differenti dalla chiave K e fra di esse.

Tutte le fasi hanno la stessa struttura. Viene eseguita una **sostituzione** sulla metà sinistra dei dati. A tale scopo viene applicata la funzione F (funzione di fase) alla metà destra dei dati e poi si svolge l'operazione di OR esclusivo dell'output di tale funzione con la metà sinistra dei dati. La funzione F ha la stessa struttura generale in ogni fase ma è parametrizzata dalla sottochiave K_i. Dopo questa sostituzione viene eseguita una **permutazione** che consiste nello scambio delle due metà dei dati.[4] Questa struttura è una forma particolare della rete a sostituzione e permutazione (SPN – Substitution-Permutation Network) proposta da Shannon.

L'effettiva realizzazione di una rete di Feistel dipende dalla scelta dei seguenti parametri e caratteristiche progettuali.

- **Dimensioni del blocco:** blocchi di grandi dimensioni migliorano la sicurezza (mantenendo uguali tutti gli altri fattori) ma riducono la velocità di crittografia e decrittografia. Nel passato un compromesso ragionevole era rappresentato da blocchi di 64 bit, quasi universale nella progettazione di sistemi di cifratura a blocchi. Tuttavia il nuovo algoritmo AES utilizza blocchi di 128 bit.

[4] L'ultima fase è seguita da uno scambio che annulla lo scambio che viene svolto nella fase finale. Si potrebbe escludere dallo schema entrambi gli scambi ma ciò pregiudicherebbe l'uniformità della presentazione. In ogni caso, la mancanza di uno scambio nella fase finale ha lo scopo di semplificare l'implementazione del processo di decrittografia, come si vedrà più avanti.

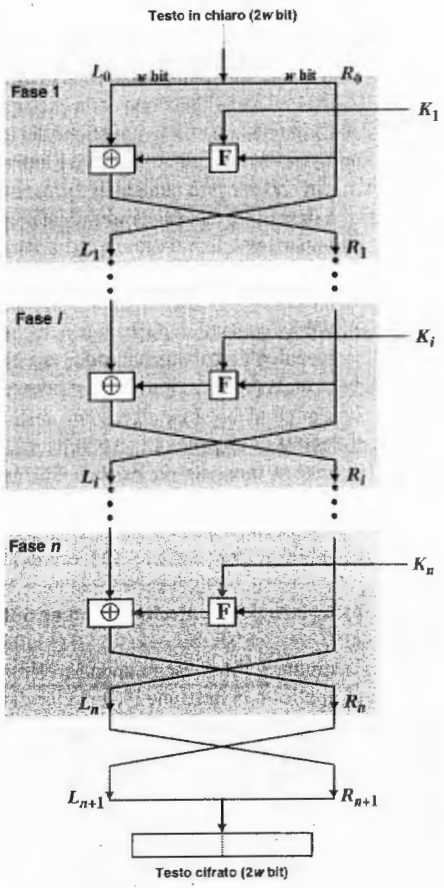


Figura 3.2 Rete classica di Feistel.

- **Dimensioni della chiave:** aumentando le dimensioni della chiave si aumenta la sicurezza ma si può ridurre la velocità di crittografia/decrittografia. La sicurezza superiore viene ottenuta da una maggiore resistenza agli attacchi a forza bruta e da una maggiore confusione. Le chiavi di dimensioni inferiori o uguali a 64 bit sono al giorno d'oggi considerate inadeguate e la dimensione più comune è 128 bit.
- **Numero di fasi:** in pratica nella cifratura di Feistel, un'unica fase offre una sicurezza inadeguata ma aumentando il numero delle fasi si ottiene una sicurezza via via superiore. Normalmente vengono utilizzate 16 fasi.
- **Algoritmo di generazione delle sottochiavi:** aumentando la complessità di questo algoritmo si dovrebbe complicare l'analisi crittografica.

- **Funzione di fase:** anche in questo caso, una maggiore complessità genera una maggiore resistenza all'analisi crittografica.

Vi sono altre due considerazioni nella progettazione di una cifratura di Feistel.

- **Velocità della crittografia/decrittografia software:** in molti casi la crittografia è incorporata nelle applicazioni o nelle funzioni di servizio, in modo da impedire un'implementazione hardware. Di conseguenza la velocità di esecuzione dell'algoritmo diviene un fattore importante.
- **Facilità di analisi:** sebbene sia preferibile che l'algoritmo sia il più possibile difficile da violare tramite analisi crittografica, vi è anche un grande vantaggio nel rendere l'algoritmo facile da analizzare. Se l'algoritmo può essere descritto concisamente e con chiarezza, infatti sarà più facile individuarne i punti deboli a un'analisi crittografica e pertanto aumentare il livello di confidenza dell'algoritmo. Per esempio, l'algoritmo DES non è facilmente analizzabile.

Algoritmo di decrittografia di Feistel

Il processo di decrittografia di una cifratura di Feistel è fondamentalmente uguale al processo di crittografia. Ecco la regola da utilizzare: come input dell'algoritmo si usa il testo cifrato ma si applicano le sottochiavi K_i in ordine inverso. Pertanto si usa la chiave K_n nella prima fase, K_{n-1} nella seconda fase e così via fino a usare K_1 all'ultima fase. Si tratta di una caratteristica interessante poiché evita di dover implementare due diversi algoritmi, uno per la crittografia e uno per la decrittografia.

Per dimostrare che lo stesso algoritmo a chiavi invertite produce il risultato corretto, si consideri la Figura 3.3 che mostra il processo di crittografia dall'alto verso il basso sul lato sinistro della figura e il processo di decrittografia dal basso verso l'alto sul lato destro della figura per un algoritmo a 16 fasi (i risultati sarebbero gli stessi per qualsiasi numero di fasi). Per chiarezza si usa la notazione LE_i e RE_i i dati che attraversano l'algoritmo di crittografia e LD_i e RD_i per i dati che attraversano l'algoritmo di decrittografia. Lo schema indica che, in ciascuna fase, il valore intermedio del processo di decrittografia è uguale al valore corrispondente del processo di crittografia con le due metà del valore scambiate. In altre parole, si indichi l'output della i-esima fase di crittografia con $LE_i \| RE_i$ (L_i concatenata con R_i). L'input corrispondente della fase (16-i)esima di decrittografia è quindi $RE_i \| LE_i$, equivalente a $RD_{16-i} \| LD_{16-i}$.

Si proverà a seguire la Figura 3.3 per dimostrare la validità delle affermazioni precedenti.[5] Dopo l'ultima iterazione del processo di crittografia, le due metà dell'output sono scambiate e dunque il testo cifrato è $RE_{16} \| LE_{16}$. L'output di tale fase è il testo cifrato. Ora si prende tale testo cifrato e lo si usa come input dello stesso algoritmo. L'input della prima fase è $RE_{16} \| LE_{16}$ che è uguale allo scambio a 32 bit dell'output della sedicesima fase del processo di crittografia.

[5] Per semplificare lo schema, non viene mostrato lo scambio che si verifica alla fine di ciascuna interazione ma si prega di notare che il risultato intermedio alla fine della fase i-esima del processo di crittografia è la quantità di $2w$ bit costituita concatenando LE_i e RE_i e il risultato intermedio alla fine della i-esima fase del processo di decrittografia è la quantità di $2w$ bit creata concatenando LD_i e RD_i.

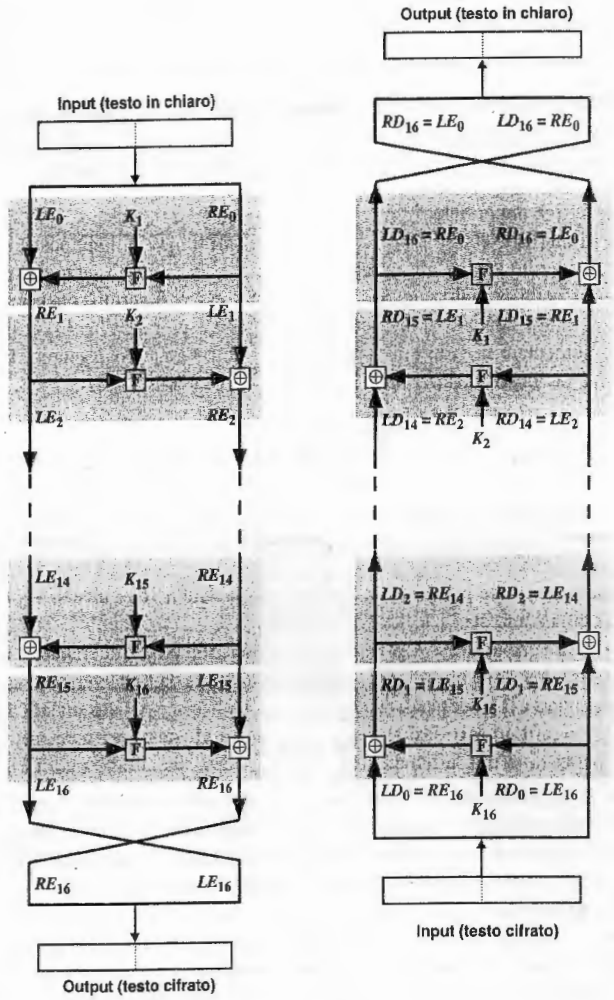


Figura 3.3 Crittografia e decrittografia di Feistel.

Ora si vuole dimostrare che l'output della prima fase del processo di decrittografia è uguale allo scambio a 32 bit dell'input della sedicesima fase del processo di crittografia. Innanzitutto si consideri il processo di crittografia. Si può vedere che:

$$LE_{16} = RE_{15}$$
$$RE_{16} = LE_{15} \oplus \mathrm{F}(RE_{15}, K_{16})$$

Sul lato della decrittografia,

$$\begin{aligned} LD_1 &= RD_0 = LE_{16} = RE_{15} \\ RD_1 &= LD_0 \oplus \mathrm{F}(RD_0, K_{16}) \\ &= RE_{16} \oplus \mathrm{F}(\mathrm{RE}_{15}, K_{16}) \\ &= [LE_{15} \oplus \mathrm{F}(RE_{15}, K_{16})] \oplus \mathrm{F}(\mathrm{RE}_{15}, K_{16}) \end{aligned}$$

L'operatore XOR ha le seguenti proprietà:

$$[A \oplus B] \oplus C = A \oplus [B \oplus C]$$
$$D \oplus D = 0$$
$$E \oplus 0 = E$$

Pertanto si avrà $LD_1 = RE_{15}$ e $RD_1 = LE_{15}$. Quindi l'output della prima fase del processo di decrittografia è $LE_{15} \| RE_{15}$, che è lo scambio a 32 bit dell'input della sedicesima fase della crittografia. Questa corrispondenza vale per tutte le 16 iterazioni. Si può riscrivere questo processo in termini generali. Per la i-esima interazione dell'algoritmo di crittografia:

$$LE_i = RE_{i-1}$$
$$RE_i = LE_{i-1} \oplus \mathrm{F}(\mathrm{RE}_{i-1}, K_i)$$

Ridisponendo i termini,

$$RE_{i-1} = LE_i$$
$$LE_{i-1} = RE_i \oplus \mathrm{F}(RE_{i-1}, K_i) = RE_i \oplus \mathrm{F}(LE_i, K_i)$$

Pertanto sono stati descritti gli input della i-esima iterazione come funzione degli output e queste equazioni confermano gli assegnamenti indicati sul lato destro della Figura 3.3.

Infine si può vedere che l'output dell'ultima fase del processo di decrittografia è $RE_0 \| LE_0$. Uno scambio a 32 bit ripristina il testo in chiaro originale, dimostrando la validità del processo di decrittografia di Feistel. Si noti che la derivazione non richiede che F sia una funzione reversibile. Per vedere ciò si può considerare il caso limite in cui F produce un output costante (per esempio tutti valori 1) indipendentemente dal valore dei suoi due argomenti. Le equazioni rimangono comunque valide.

3.2 L'algoritmo DES (Data Encryption Standard)

Lo schema di crittografia più utilizzato si basa sullo standard DES (Data Encryption Standard) adottato nel 1977 dal National Bureau of Standards, oggi NIST (National Institute of Standards and Technology), come FIPS PUB 46 (Federal Information Processing Standard 46). L'algoritmo è però chiamato DEA (Data Encryption Algorithm).[6] In DES i dati vengo-

[6] La terminologia è un po' confusa. Fino a poco tempo fa, i termini *DES* e *DEA* potevano essere considerati intercambiabili. Ma l'edizione più recente del documento DES include una specifica dell'algoritmo DEA (qui descritto) e dell'algoritmo TDEA (triple DEA) di cui si parla nel Capitolo 6. Sia DEA che TDEA fanno parte dello standard DES. Inoltre, fino alla recente adozione del termine ufficiale *TDEA*, l'algoritmo TDEA veniva normalmente chiamato *triple* DES (o 3DES). Per comodità, si utilizzerà l'indicazione 3DES.

no crittografati in blocchi di 64 bit utilizzando una chiave di 56 bit. L'algoritmo trasforma in una serie di passi l'input di 64 bit in un output di 64 bit. Gli stessi passi, con la stessa chiave consentono invece di invertire la crittografia.

L'algoritmo DES è ampiamente utilizzato ed è stato il soggetto di molte controversie riguardanti la sua sicurezza. Per comprendere la natura di queste controversie, è opportuno introdurre brevemente la storia di DES.

Alla fine degli anni '60, l'IBM diede origine a un progetto di ricerca nel campo della crittografia condotto da Horst Feistel. Il progetto portò nel 1971 allo sviluppo di un algoritmo chiamato LUCIFER [FEIS73], che venne venduto ai Lloyd di Londra per l'utilizzo nei sistemi di prelievo di contante, anch'essi sviluppati da IBM. LUCIFER è una cifratura a blocchi di Feistel che opera su blocchi di 64 bit utilizzando chiavi di 128 bit. Dati i risultati promettenti prodotti dal progetto LUCIFER, IBM pensò di sviluppare un prodotto di crittografia commerciale che potesse essere implementato su un chip. Questa operazione venne diretta da Walter Tuchman e Carl Meyer e coinvolse non solo i ricercatori IBM ma anche consulenti esterni e il personale tecnico della NSA. Il risultato di questo tentativo fu una versione raffinata di LUCIFER che era più resistente all'analisi crittografica ma utilizzava chiavi di 56 bit per poter rientrare in un chip.

Nel 1973, l'NBS (National Bureau of Standards) emise un bando di gara per proporre uno standard di cifratura a livello nazionale. IBM inviò i risultati del progetto Tuchman-Meyer. Questo era di gran lunga il migliore algoritmo proposto e nel 1977 fu adottato con il nome Data Encryption Standard.

Prima della sua adozione come standard, la proposta di DES fu soggetta ad ampie critiche che non si sono ancora spente. Due elementi in particolare fecero sorgere molte critiche. Innanzitutto la lunghezza della chiave nell'algoritmo LUCIFER originale di IBM era di 128 bit mentre quella del sistema proposto era di soli 56 bit con una riduzione enorme, pari a 72 bit. Le critiche sostenevano che questa chiave era troppo breve per sopportare un attacco a forza bruta. La seconda area problematica era il fatto che i criteri progettuali della struttura interna di DES, ovvero le S-box, erano segreti. Pertanto gli utenti non potevano sapere se la struttura interna di DES fosse esente da punti deboli nascosti che potessero consentire alla NSA di decifrare i messaggi senza utilizzare la chiave. I successivi eventi, in particolare i recenti lavori di analisi crittografica differenziale, sembrano indicare che DES ha una struttura interna molto resistente. Inoltre, secondo IBM, le sole modifiche apportate alla proposta riguardavano le S-box; tali modifiche, suggerite dalla NSA, hanno eliminato i punti deboli identificati nel corso del processo di valutazione.

In ogni caso, DES si è molto sviluppato ed è ampiamente utilizzato specialmente nelle applicazioni finanziarie. Nel 1994, il NIST confermò DES per l'utilizzo a livello Federale per altri cinque anni sebbene non per applicazioni relative alla protezione di informazioni militari segrete. Nel 1999, NIST emise una nuova versione dello standard (FIPS PUB 46-3) che indicava che DES doveva essere utilizzato solo per i sistemi preesistenti e che venisse invece adottata la versione triple DES (che in pratica prevede la ripetizione dell'algoritmo DES per tre volte sul testo in chiaro utilizzando due o tre chiavi differenti). Si tratterà l'algoritmo triple DES nel Capitolo 6. Poiché gli algoritmi di crittografia e decrittografia sono gli stessi sia in DES che in triple DES, è fondamentale conoscere il funzionamento della cifratura DES.

La cifratura DES

La Figura 3.4 illustra lo schema globale della crittografia DES. Come per ogni altro schema di crittografia, la funzione di crittografia prevede due input: il testo in chiaro da crittografare e la chiave. In questo caso, il testo in chiaro deve avere una lunghezza di 64 bit e la chiave una lunghezza di 56 bit.[7]

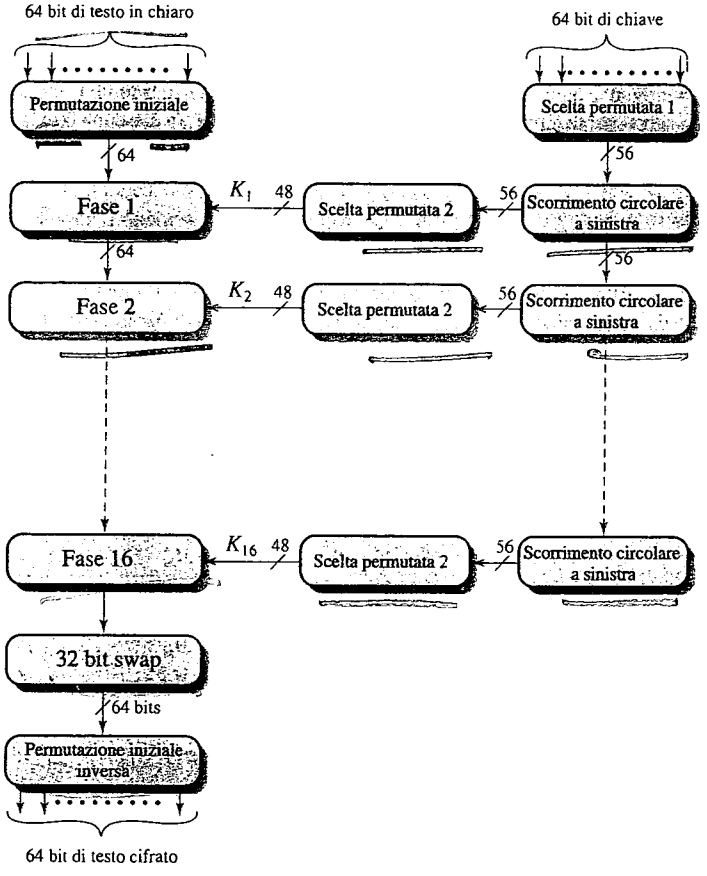


Figura 3.4 La presentazione generale dell'algoritmo di crittografia DES.

[7] In realtà la funzione si attende come input una chiave di 64 bit. Tuttavia vengono utilizzati solo 56 di questi bit; gli altri 8 bit possono essere impiegati come bit di parità o semplicemente impostati in modo arbitrario.

Osservando il lato sinistro della figura, si può vedere che l'elaborazione del testo in chiaro viene eseguita in tre fasi. Innanzitutto il testo in chiaro di 64 bit viene sottoposto a una permutazione iniziale (IP) che dispone i bit per produrre un *input permutato*. Questa operazione è seguita da una fase costituita da 16 ripetizioni della stessa funzione di permutazione e sostituzione. L'output dell'ultima fase (la sedicesima) è costituito da 64 bit che dipendono dal testo in chiaro e dalla chiave. Le metà di sinistra e di destra vengono scambiate per produrre un **preoutput** che attraversa poi la permutazione inversa della permutazione iniziale (IP^{-1}) per produrre il testo cifrato a 64 bit. Come si può vedere nella Figura 3.2, con l'eccezione delle permutazioni iniziale e finale, DES ha esattamente la stessa struttura della cifratura di Feistel. Il lato destro della Figura 3.4 mostra il modo in cui viene utilizzata la chiave di 56 bit. Inizialmente la chiave attraversa una funzione di permutazione, poi, per ognuna delle 16 fasi, viene prodotta una *sottochiave* K_i tramite la combinazione di uno scorrimento circolare a sinistra e di una permutazione. In ciascuna fase la funzione di permutazione è la stessa ma viene prodotta una sottochiave differente a causa degli scorrimenti ripetuti dei bit della chiave.

Permutazione iniziale

La permutazione iniziale e la sua inversa sono definite tramite tabelle come le Tabelle 3.2A e 3.2B. Le tabelle devono essere interpretate nel seguente modo. L'input di una tabella è costituito da 64 bit numerati da 1 a 64. Le 64 voci della tabella di permutazione contengono una permutazione dei numeri da 1 a 64. Ciascuna voce della tabella delle permutazioni indica la posizione di un bit di input nell'output, anch'esso costituito da 64 bit.

Per verificare che queste due funzioni di permutazione sono una l'inversa dell'altra, si consideri il seguente input M di 64 bit:

$$\begin{array}{cccccccc} M_1 & M_2 & M_3 & M_4 & M_5 & M_6 & M_7 & M_8 \\ M_9 & M_{10} & M_{11} & M_{12} & M_{13} & M_{14} & M_{15} & M_{16} \\ M_{17} & M_{18} & M_{19} & M_{20} & M_{21} & M_{22} & M_{23} & M_{24} \\ M_{25} & M_{26} & M_{27} & M_{28} & M_{29} & M_{30} & M_{31} & M_{32} \\ M_{33} & M_{34} & M_{35} & M_{36} & M_{37} & M_{38} & M_{39} & M_{40} \\ M_{41} & M_{42} & M_{43} & M_{44} & M_{45} & M_{46} & M_{47} & M_{48} \\ M_{49} & M_{50} & M_{51} & M_{52} & M_{53} & M_{54} & M_{55} & M_{56} \\ M_{57} & M_{58} & M_{59} & M_{60} & M_{61} & M_{62} & M_{63} & M_{64} \end{array}$$

dove M_i è una cifra binaria. La permutazione risultante $X = IP(M)$ è la seguente:

$$\begin{array}{cccccccc} M_{58} & M_{50} & M_{42} & M_{34} & M_{26} & M_{18} & M_{10} & M_2 \\ M_{60} & M_{52} & M_{44} & M_{36} & M_{28} & M_{20} & M_{12} & M_4 \\ M_{62} & M_{54} & M_{46} & M_{38} & M_{30} & M_{22} & M_{14} & M_6 \\ M_{64} & M_{56} & M_{48} & M_{40} & M_{32} & M_{24} & M_{16} & M_8 \\ M_{57} & M_{49} & M_{41} & M_{33} & M_{25} & M_{17} & M_9 & M_1 \\ M_{59} & M_{51} & M_{43} & M_{35} & M_{27} & M_{19} & M_{11} & M_3 \\ M_{61} & M_{53} & M_{45} & M_{37} & M_{29} & M_{21} & M_{13} & M_5 \\ M_{63} & M_{55} & M_{47} & M_{39} & M_{31} & M_{23} & M_{15} & M_7 \end{array}$$

Se si prende ora la permutazione inversa $Y = IP^{-1}(X) = IP^{-1}(IP(M))$, si può vedere che viene ripristinato l'ordinamento originale dei bit.

Dettagli di una fase

La Figura 3.5 mostra la struttura interna di una fase. Anche in questo caso è opportuno concentrarsi sul lato sinistro dello schema. Le metà sinistra e destra di ciascun valore intermedio di 64 bit vengono trattate come due entità distinte da 32 bit, indicate come L (left) e R (right). Come nella cifratura classica di Feistel, l'elaborazione generale in ciascuna fase può essere riassunta dalle seguenti formule:

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

La chiave della fase K_i è di 48 bit. L'input R è di 32 bit. Questo input R viene innanzitutto espanso a 48 bit utilizzando una tabella che definisce una permutazione più un'espansione che prevede la duplicazione di 16 dei bit di R (Tabella 3.2C). Ai risultanti 48 bit viene applicata, tramite l'operatore XOR, la chiave K_i. Questo risultato di 48 bit attraversa una funzione di sostituzione che produce un output di 32 bit che viene permutato secondo quanto definito nella Tabella 3.2D.

Tabella 3.2 Tabelle di permutazione per DES.

A. Permutazione iniziale (IP)

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

B. Permutazione iniziale inversa (IP^{-1})

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29

(segue)

Tabella 3.2 Tabelle di permutazione per DES. *(continua)*

B. Permutazione iniziale inversa (IP^{-1})

36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

C. Permutazione di espansione (E)

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

D. Funzione di permutazione (P)

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

La Figura 3.6 illustra il ruolo delle S-box nella funzione F. La sostituzione è costituita da un insieme di 8 S-box, ognuna delle quali accetta in input 6 bit e produce come output 4 bit. Queste trasformazioni sono definite nella Tabella 3.3 che può essere interpretata nel seguente modo: il primo e l'ultimo bit dell'input della box S_i formano un numero binario di 2 bit che seleziona una delle quattro sostituzioni definite dalle quattro righe della tabella per S_i. I 4 bit centrali selezionano una delle sedici colonne. Il valore decimale nella cella identificata dalla riga e dalla colonna viene poi convertito nella sua rappresentazione a 4 bit per produrre l'output. Per esempio, in S_1, per l'input 011001, la riga è 01 (riga 1) e la colonna è 1100 (colonna 12). Il valore nella riga 1, colonna 12 è 9 e dunque l'output è 1001.

Ogni riga di una S-box definisce una sostituzione generale reversibile. La Figura 3.1 può essere utile per comprendere le associazioni eseguite. La figura mostra la sostituzione per la riga 0 della box S_1.

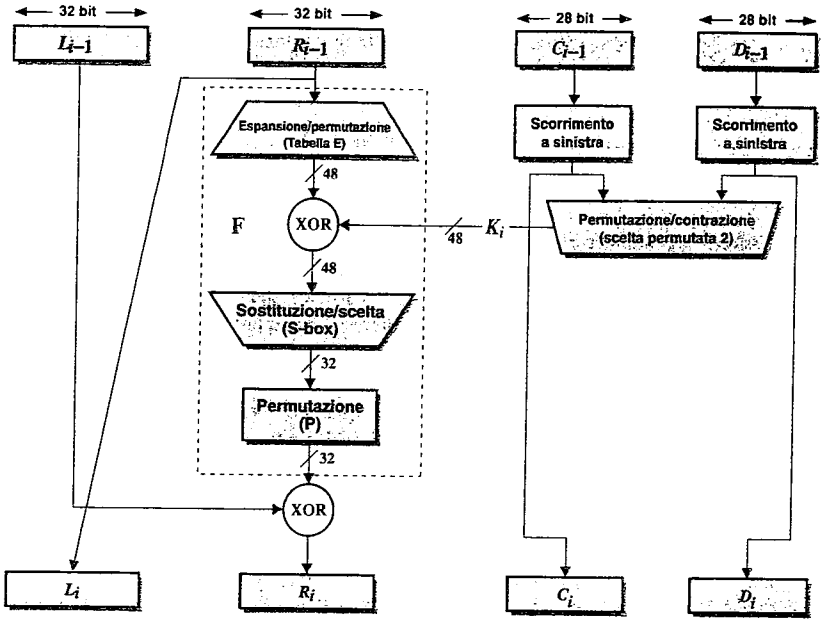


Figura 3.5 Una fase dell'algoritmo DES.

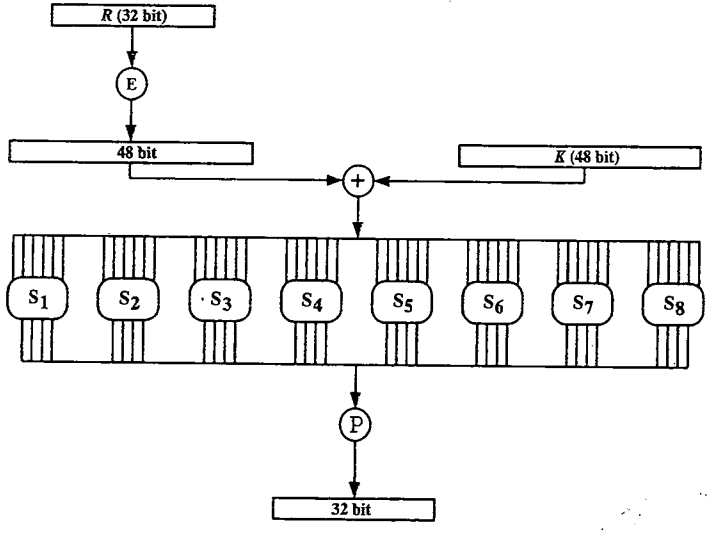


Figura 3.6 Calcolo di F(R, K).

Il funzionamento delle S-box merita ulteriori commenti. Si ignori per il momento il contributo della chiave K_i. Se si esamina la tabella di espansione, si vede che i 32 bit di input vengono suddivisi in gruppi di 4 bit, che divengono gruppi di 6 bit prendendo i bit esterni dai due gruppi adiacenti. Per esempio, se parte della word di input è:

. . . efgh ijkl mnop . . .

questa diverrà:

. . . defghi hijklm lmnopq . . .

I 2 bit esterni di ciascun gruppo selezionano una delle quattro possibili sostituzioni (una riga di una S-box). Quindi un valore di output di 4 bit viene sostituito al particolare input di 4 bit (i 4 bit di input intermedi). L'output di 32 bit delle 8 S-box viene poi permutato in modo che alla fase successiva l'output di ciascuna S-box interessi quanti più valori possibili.

Generazione della chiave

Tornando alle Figure 3.4 e 3.5, si noti che come input dell'algoritmo viene utilizzata una chiave di 64 bit. I bit della chiave sono numerati da 1 a 64; un bit ogni 8 viene ignorato come indicato dalla colonna in grassetto nella Tabella 3.4A. La chiave viene innanzitutto permutata secondo la tabella Permuted Choice 1 (Tabella 3.4B). La chiave risultante, di 56 bit, viene poi trattata come 2 quantità di 28 bit, chiamate C_0 e D_0.
In ciascuna fase, C_{i-1} e D_{i-1} sono separatamente soggette a uno scorrimento circolare o rotazione a sinistra di 1 o 2 bit come indicato dalla Tabella 3.4D. Questi valori fungeranno da input per la fase successiva. Inoltre vengono utilizzati come input per la tabella Permuted Choice 2 (Tabella 3.4C) che produce un output di 48 bit che funge da input alla funzione $F(R_{i-1}, K_i)$.

Tabella 3.3 Definizione delle S-Box DES.

S_1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

(segue)

Tabella 3.3 Definizione delle S-Box DES. *(continua)*

S_5	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Tabella 3.4 Calcolo della chiave in DES.

A.	**Chiave di input**							
1	2	3	4	5	6	7	**8**
9	10	11	12	13	14	15	**16**
17	18	19	20	21	22	23	**24**
25	26	27	28	29	30	31	**32**
33	34	35	36	37	38	39	**40**
41	42	43	44	45	46	47	**48**
49	50	51	52	53	54	55	**56**
57	58	59	60	61	62	63	**64**

B.	**Permuted Choice 1 (PC-1)**						
57	49	41	33	25	17	9	
1	58	50	42	34	26	18	
10	2	59	51	43	35	27	
19	11	3	60	52	44	36	
63	55	47	39	31	23	15	

(segue)

Tabella 3.4 Calcolo della chiave in DES. *(continua)*

B. Permuted Choice 1 (PC-1)						
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

C. Permuted Choice 2 (PC-2)							
14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

D. Scorrimenti a sinistra

Numero fase	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bit ruotati	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Decrittografia DES

Come nel caso della cifratura Feistel, la decifratura utilizza lo stesso algoritmo della cifratura, applicando le sottochiavi in ordine inverso.

Effetto valanga

Una proprietà desiderabile in un algoritmo di crittografia è che una piccola variazione nel testo in chiaro o nella chiave dovrebbe produrre una variazione significativa nel testo cifrato. In particolare, la modifica di un bit del testo in chiaro o della chiave dovrebbe provocare un cambiamento in più bit del testo cifrato. Se il cambiamento fosse invece limitato potrebbe essere possibile ridurre le dimensioni dello spazio delle chiavi o del testo in chiaro in cui eseguire la ricerca.

DES esibisce un forte effetto valanga. La Tabella 3.5 mostra alcuni risultati tratti da [KONH81]. Nella Tabella 3.5A sono stati usati due testi in chiaro che differiscono per un solo bit:

```
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```


Tabella 3.5 L'effetto valanga in DES.

A. Variazione nel testo in chiaro		B. Variazione nella chiave	
Fase	Numero di bit differenti	Fase	Numero di bit differenti
0	1	0	0
1	6	1	2
2	21	2	14
3	35	3	28
4	39	4	32
5	34	5	30
6	32	6	32
7	31	7	35
8	29	8	34
9	42	9	40
10	44	10	38
11	32	11	31
12	30	12	33
13	30	13	28
14	26	14	26
15	29	15	34
16	34	16	35

con la chiave:

```
0000001 1001011 0100100 1100010 0011100 0011000 0011100 0110010
```

La tabella 3.5A mostra che dopo tre sole fasi, fra i due blocchi vi sono 21 bit differenti. Al termine, i testi cifrati differiscono per 34 bit.

La Tabella 3.5B mostra un test simile che utilizza come input il valore:

```
01101000 10000101 00101111 01111010 00010011 01110110 11101011 10100100
```

utilizzando due chiavi che differiscono per un solo bit:

```
1110010 1111011 1101111 0011000 0011101 0000100 0110001 11011100
0110010 1111011 1101111 0011000 0011101 0000100 0110001 11011100
```

Ancora una volta i risultati mostrano che vi è una differenza in circa la metà dei bit del testo cifrato e che l'effetto valanga è pronunciato anche dopo poche fasi.

3.3 La potenza di DES

Fin dalla sua adozione come standard federale, sono stati espressi vari dubbi relativi al livello di sicurezza fornito da DES. Questi dubbi hanno due cause: le dimensioni della chiave e la natura dell'algoritmo.

L'uso di chiavi a 56 bit

Con chiavi di 56 bit esistono 2^{56} possibili chiavi, ovvero circa $7{,}2 \times 10^{16}$ chiavi. Pertanto, date queste dimensioni, un attacco a forza bruta sembra poco pratico. Supponendo che, in media, debba essere ricercata metà dello spazio delle chiavi, un'unica macchina che svolga una crittografia DES al microsecondo impiegherebbe più di un migliaio di anni (vedere la Tabella 2.2) per violare il codice.

Tuttavia la stima del calcolo di una crittografia al microsecondo è estremamente conservativa. Già nel 1977, Diffie e Hellman sostenevano che esisteva la tecnologia in grado di consentire lo sviluppo di una macchina parallela con un milione di dispositivi di crittografia, ognuno dei quali poteva svolgere una crittografia al microsecondo [DIFF77]. Questo avrebbe ridotto il tempo medio di ricerca a circa 10 ore. Gli autori stimarono che il costo sarebbe stato di circa 20 milioni di dollari del 1977.

L'algoritmo DES si è infine dimostrato non sicuro nel luglio del 1998, quando la EFF (Electronic Frontier Foundation) annunciò di aver violato una crittografia DES utilizzando un'apposita macchina "DES cracker" costruita con meno di 250 000 dollari. L'attacco richiese meno di tre giorni. La EFF pubblicò una descrizione dettagliata della macchina che consentiva a chiunque di costruire il proprio sistema di violazione [EFF98]. Naturalmente il prezzo dell'hardware continua a calare mentre la velocità aumenta sempre più, ciò che rende DES praticamente inutile.

È importante notare che è necessario qualcosa di più che un semplice attacco che prova tutte le chiavi possibili. L'analista deve essere in grado di riconoscere il testo in chiaro come tale a meno che non venga fornito. Se il messaggio fosse un semplice testo per esempio in inglese, allora il risultato corretto verrebbe individuato con facilità, anche se è necessario automatizzare l'operazione di riconoscimento. Se però il messaggio di testo fosse stato compresso prima della crittografia, il riconoscimento risulterebbe più difficile. Se poi il messaggio fosse costituito da dati di tipo più generico, per esempio un file numerico, anch'esso compresso, il problema diverrebbe ancora più difficile da automatizzare. Pertanto per affiancare l'approccio a forza bruta occorre avere qualche informazione sul testo in chiaro che ci si attende e avere un modo per distinguere automaticamente il testo in chiaro dai risultati prodotti da chiavi errate. L'approccio EFF considera anche questi problemi e introduce alcune tecniche automatizzate efficaci anche in molti altri contesti.

Fortunatamente vi sono varie alternative a DES, le più importanti delle quali sono AES e triple DES, entrambe trattate rispettivamente nei Capitoli 5 e 6.

La natura dell'algoritmo DES

Un altro potenziale problema è la possibilità di svolgere un'analisi crittografica sfruttando le caratteristiche dell'algoritmo DES. Il cuore del problema è costituito dalle otto tabelle di sostituzione, o S-box, utilizzate in ciascuna iterazione. Dato che i criteri progettuali di queste box e dell'intero algoritmo non sono stati resi pubblici, vi è il sospetto che le S-box siano state costruite in modo da consentire un'analisi crittografica da parte di chi conosca i punti deboli delle S-box. Questa affermazione è molto provocante e nel corso degli anni sono state scoperte varie regolarità e comportamenti inaspettati delle S-box. Nonostante questo, nessuno è riuscito finora a individuare i supposti punti deboli.[8]

Attacchi temporizzati

Si parlerà degli attacchi temporizzati più in dettaglio nella Parte seconda in quanto fanno riferimento agli algoritmi a chiave pubblica. Tuttavia l'argomento può essere rilevante anche per le cifrature simmetriche. Sostanzialmente un attacco temporizzato ottiene informazioni sulla chiave o sul testo in chiaro sulla base del tempo impiegato da una data implementazione a svolgere le decrittografia di vari testi cifrati. Un attacco temporizzato sfrutta il fatto che l'algoritmo di crittografia o decrittografia impiega tempi leggermente differenti a seconda dell'input. [HEVI99] ha prodotto un rapporto su un approccio che fornisce i pesi di Hamming (numero di bit uguali a 1) della chiave segreta. Si è ancora lontani dal conoscere la chiave, ma si tratta comunque di un passo interessante. Gli autori concludono sostenendo che DES risulta piuttosto resistente a un attacco temporizzato ma suggeriscono alcune vie da esplorare. Anche se questa è una linea di attacco interessante, sembra improbabile che questa tecnica possa avere successo contro DES o le cifrature simmetriche più potenti come triple DES e AES.

3.4 Analisi crittografiche differenziale e lineare

Per la maggior parte della sua vita, la preoccupazione più importante rispetto a DES è stata la sua vulnerabilità agli attacchi a forza bruta, data la relativa brevità della sua chiave (56 bit). Tuttavia vi è stato anche un certo interesse nella ricerca di attacchi basati su analisi crittografica. Data la crescente popolarità delle cifrature a blocchi con chiavi di lunghezza superiore, fra cui triple DES, gli attacchi a forza bruta sono diventati via via sempre meno

[8] Quanto meno, nessuno ha riconosciuto pubblicamente tale scoperta.

pratici. Pertanto vi è stata un'enfasi crescente sugli attacchi basati su analisi crittografica sia a DES che ad altre cifrature simmetriche a blocchi. In questa parte del capitolo verrà offerta una breve panoramica dei due approcci più potenti e promettenti: l'analisi crittografica differenziale e l'analisi crittografica lineare.

Analisi crittografica differenziale

Uno dei miglioramenti recenti più significativi nell'analisi crittografica è l'analisi crittografica differenziale. In questa parte del capitolo si tratteranno questa tecnica e la sua applicabilità all'algoritmo DES.

Cronologia

L'analisi crittografica differenziale venne presentata nella letteratura disponibile al pubblico solo nel 1990. Il primo materiale pubblicato di Murphy [MURP90] faceva riferimento all'analisi crittografica di una cifratura a blocchi chiamata FEAL. Questo articolo fu seguito da vari altri articoli di Biham e Shamir che dimostravano questa forma di attacco su vari algoritmi di crittografia e funzioni hash; i loro risultati sono riepilogati in [BIHA93].

I risultati più pubblicizzati di questo approccio sono stati quelli applicabili a DES. L'analisi crittografica differenziale è il primo attacco pubblicato in grado di violare l'algoritmo DES con una complessità minore di 2^{55}. Lo schema, riportato in [BIHA93], consente l'analisi crittografica di DES con complessità dell'ordine di 2^{47}operazioni crittografiche, richiedendo 2^{47} testi in chiaro scelti. Sebbene 2^{47} sia certamente molto inferiore a 2^{55}, la necessità per l'avversario di trovare 2^{47} testi in chiaro scelti rende l'interesse per questo tipo di attacco esclusivamente teorico.

Sebbene l'analisi crittografica differenziale sia uno strumento potente, non si comporta molto bene contro DES. Il motivo, secondo un membro del team IBM che ha sviluppato DES [COPP94], è il fatto che l'analisi crittografica differenziale era nota al team di sviluppo di IBM fin dal 1974. La necessità di rafforzare l'algoritmo DES contro gli attacchi a analisi crittografica differenziale ha pertanto dettato in larga misura la progettazione delle S-box e della funzione di permutazione P. Come prova dell'impatto di queste modifiche, si considerino i risultati comparativi riportati in [BIHA93]. L'analisi crittografica differenziale di un algoritmo LUCIFER a otto fasi richiede solo 256 testi in chiaro scelti mentre un attacco contro una versione a 8 fasi di DES richiede 2^{14} testi in chiaro scelti.

Attacchi ad analisi crittografica differenziale

Un attacco ad analisi crittografica differenziale è complesso; [BIHA93] ne fornisce una descrizione completa. L'idea alla base dell'analisi crittografica differenziale consiste nel considerare il comportamento di coppie di blocchi di testo nelle varie fasi della cifratura, invece di osservare l'evoluzione di un singolo blocco. Qui si fornirà una breve panoramica per dare un'idea dell'attacco.

Si inizierà cambiando leggermente la notazione per DES. Si consideri che il blocco di testo in chiaro originale m sia costituito da due metà m_0 e m_1. Ogni fase di DES mappa l'input di destra nell'output di sinistra e assegna all'output di destra una funzione dell'input di sinistra e della sottochiave di quella fase. Pertanto, in ciascuna fase, viene creato solo un nuovo blocco di 32 bit. Se si indica ciascun nuovo blocco con m_i ($2 \le i \le 17$), le metà intermedie del messaggio soddisfano la seguente relazione:

$$m_{i+1} = m_{i-1} \oplus f(m_i, K_i) \qquad i = 1,2,\ldots,16$$

Nell'analisi crittografica differenziale, si parte con due messaggi, m e m' con una differenza XOR nota $\Delta m = m \oplus m'$ e si considera la differenza fra le metà intermedie del messaggio: $\Delta m_i = m_i \oplus m'_i$. Allora si avrà:

$$\begin{aligned}\Delta m_{i+1} &= m_{i+1} \oplus m'_{i+1} \\ &= [m_{i-1} \oplus f(m_i, K_i)] \oplus [m'_{i-1} \oplus f(m'_i, K_i)] \\ &= \Delta m_{i-1} \oplus [f(m_i, K_i) \oplus f(m'_i, K_i)]\end{aligned}$$

Ora si supponga che più coppie di input di f con la stessa differenza generino la stessa differenza di output nel caso in cui venga utilizzata la stessa sottochiave. Più precisamente, si può dire che *X può provocare Y con probabilità p* se per una frazione p delle coppie in cui lo XOR di input è X, lo XOR di output è uguale a Y. Si vuole supporre che vi sia un certo numero di valori di X che hanno un'elevata probabilità di provocare una determinata differenza nell'output. Pertanto, se si conoscono Δm_{i-1} e Δm_i con elevata probabilità, allora si conosce Δm_{i+1} con elevata probabilità. Inoltre, se si determinano molte di queste differenze, sarà possibile determinare la sottochiave utilizzata nella funzione f.

La strategia generale dell'analisi crittografica differenziale si basa su queste considerazioni per ogni singola fase. La procedura consiste nel cominciare con due messaggi di testo in chiaro m e m' con una determinata differenza e individuare gli schemi di differenza dopo ciascuna fase in modo da ottenere una probabile differenza nel testo cifrato. In realtà vi sono due differenze probabili per le due metà di 32 bit: ($\Delta m_{17} \| \Delta m_{16}$). Poi si applica la crittografia a m e m' per determinare l'effettiva differenza con la chiave sconosciuta e si confrontano i risultati con la differenza probabile. Se vi è una corrispondenza,

$$E(K, m) \oplus E(K, m') = (\Delta m_{17} \| \Delta m_{16})$$

si deve sospettare che tutti gli schemi probabili in tutte le fasi intermedie siano corretti. Date queste premesse si possono fare alcune supposizioni sui bit della chiave. Questa procedura deve essere ripetuta più volte per determinare tutti i bit della chiave.

La Figura 3.7, tratta da [BIHA93], illustra la propagazione delle differenze attraverso tre fasi di DES. Le probabilità mostrate sulla destra fanno riferimento alla probabilità che un determinato insieme di differenze intermedie appaia in funzione delle differenze nell'input. In generale, la probabilità dopo tre fasi che la differenza nell'output sia quella indicata è uguale a $0,25 \times 1 \times 0,25 = 0,0625$.

Analisi crittografica lineare

Uno sviluppo più recente è costituito dall'analisi crittografica lineare, descritto in [MATS93]. Questo attacco si basa sull'individuazione delle approssimazioni lineari per descrivere le trasformazioni eseguite in DES. Questo metodo può trovare una chiave DES da 2^{43} testi in chiaro noti, rispetto ai 2^{47} testi in chiaro scelti dell'analisi crittografica differenziale. Seb-

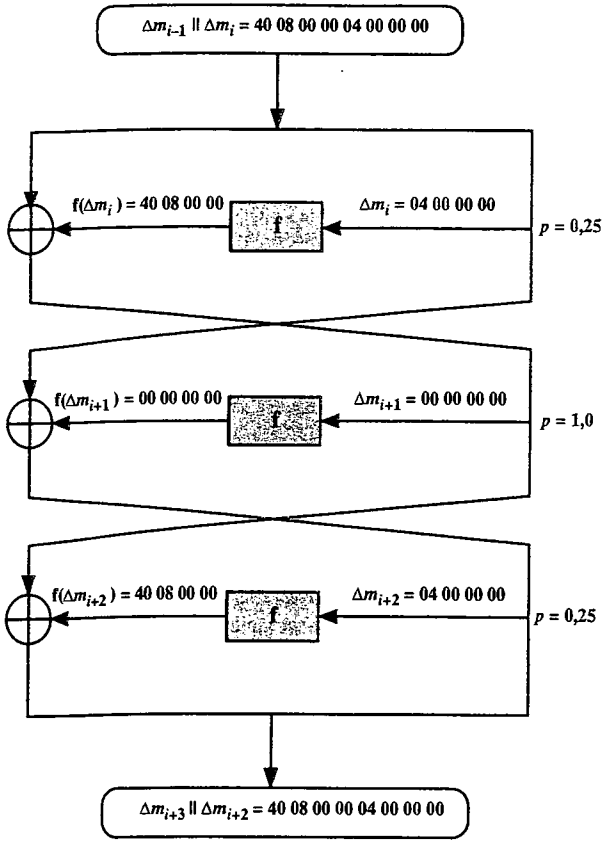


Figura 3.7 Propagazione differenziale attraverso tre fasi dell'algoritmo DES (numeri esadecimali).

bene vi sia un certo miglioramento, dato che è più facile ottenere il testo in chiaro noto rispetto al testo in chiaro scelto, l'analisi crittografica risulta comunque improponibile come attacco DES. Finora il lavoro svolto è insufficiente per poter convalidare l'approccio ad analisi crittografica lineare.

Si fornisce ora una breve riassunto del principio su cui si basa l'analisi crittografica lineare. Per una cifratura con testo in chiaro di n bit, blocchi di testo cifrato di n bit e una chiave di m bit, se il blocco di testo in chiaro è P[1], . . .P[n], il blocco di testo cifrato è C[1], . . . C[n] e la chiave è K[1], . . . , K[m], si definisce:

$$A[i,j,\ldots,k] = A[i] \oplus A[j] \oplus \ldots \oplus A[k]$$

L'obiettivo dell'analisi crittografica lineare è quello di trovare un'equazione lineare efficace nella forma:

$$P[\alpha_1,\alpha_2,\ldots,\alpha_a] \oplus C[\beta_1,\beta_2,\ldots,\beta_b] = K[\gamma_1,\gamma_2,\ldots,\gamma_c]$$

(dove $x = 0$ o 1, $1 \le a, b \le n$, $1 \le c \le m$ e dove i termini α, β e γ, rappresentano posizioni fisse e univoche dei bit) con la probabilità $p \neq 0{,}5$. Più lontano è p da 0,5, più efficace sarà l'equazione. Una volta determinata la relazione proposta, la procedura consiste nel calcolare i risultati del lato sinistro dell'equazione precedente per un numero esteso di coppie di testo in chiaro / testo cifrato. Se il risultato è 0 per più di metà delle volte, si suppone che $K[\gamma_1,\gamma_2, \ldots ,\gamma_c] = 0$. Se è 1 per la maggior parte delle volte, si suppone $K[\gamma_1,\gamma_2, \ldots ,\gamma_c] = 1$. Questo fornisce un'equazione lineare sui bit della chiave. Si cerca di ottenere quante più relazioni possibili in modo che sia possibile risolvere i bit della chiave. Poiché si tratta di equazioni lineari, il problema può essere affrontato una fase per volta e poi si possono combinare i risultati.

3.5 Principi di progettazione della cifratura a blocchi

Sebbene siano stati fatti molti progressi nella progettazione di cifrature a blocchi resistenti contro gli attacchi crittografici, i principi di base non sono cambiati poi molto dal lavoro di Feistel e del team di progettazione di DES svolto all'inizio degli anni '70. È utile iniziare questa discussione ricordando i criteri progettuali di DES che sono stati pubblicati. Si analizzeranno poi tre aspetti critici della progettazione delle cifrature a blocchi: il numero di fasi, la funzione F e la chiave.

Criteri progettuali di DES

I criteri utilizzati per la progettazione di DES, come indicato in [COPP94], si concentrano sulla progettazione delle S-box e della funzione P che usa l'output delle S-box (vedere la Figura 3.6). Ecco i criteri di progettazione delle S-box.

1. Nessun bit di output di nessuna S-box deve essere troppo simile a una funzione lineare dei bit di input. In particolare, se si selezionano un qualiasi bit di output e un qualiasi sottoinsieme dei 6 bit di input, la frazione degli input per i quali questo bit di output è uguale allo XOR di questi bit di input non deve essere troppo vicina a 0 o 1 ma il più possibile vicina a 1/2.
2. Ogni riga di una S-box (determinata da un valore fisso dei bit di input più a sinistra e più a destra) dovrebbe includere tutte le 16 possibili combinazioni dei bit di output.
3. Se due input di una S-box differiscono esattamente di 1 bit, gli output devono differire per almeno 2 bit.
4. Se due input di una S-box differiscono esattamente per i 2 bit centrali, gli output devono differire per almeno 2 bit.
5. Se due input di una S-box differiscono per i primi 2 bit e sono identici per gli ultimi 2 bit, i 2 output devono essere differenti.

6. Per ogni differenza diversa da zero dei 6 bit di input, non più di 8 delle 32 coppie di input che esibiscono tale differenza devono generare la stessa differenza di output.
7. Questo è un criterio simile al precedente ma per il caso di tre S-box.

Coppersmith sostiene che il primo criterio dell'elenco precedente è necessario poiché le S-box sono l'unica parte non lineare dell'algoritmo DES. Se le S-box fossero lineari (ovvero se ogni bit di output fosse una combinazione lineare dei bit di input), l'intero algoritmo risulterebbe lineare e dunque facilmente violabile. Si è già visto questo fenomeno con la cifratura di Hill che ha una natura lineare. I criteri rimanenti hanno principalmente lo scopo di complicare l'analisi crittografica differenziale e introdurre buone proprietà di confusione.

I criteri per la permutazione P sono i seguenti.

1. I 4 bit di ouptut di ciascuna S-box nella fase i sono distribuiti in modo che 2 di essi influenzino (ovvero rappresentino l'input per) i "bit centrali" della fase $i + 1$ e gli altri 2 influenzino i bit agli estremi. I 2 bit centrali dell'input di una S-box non sono condivisi con le S-box adiacenti. I bit finali sono i 2 bit a sinistra e i 2 bit a destra, che sono condivisi con le S-box adiacenti.
2. I 4 bit di output di ciascuna S-box influenzano 6 diverse S-box nella fase successiva mentre non ne esistono 2 che influenzano la stessa S-box.
3. Per 2 S-box j e k, se un bit di output di S_j influenza un bit centrale di S_k nella fase successiva, allora un bit di output di S_k non può influenzare un bit centrale di S_j. Questo implica che per $j = k$, un bit di output di S_j non deve influenzare il bit centrale di S_j.

Questi criteri hanno lo scopo di aumentare la diffusione dell'algoritmo.

Numero di fasi

La resistenza crittografica di una cifratura di Feistel deriva da tre aspetti progettuali: il numero di fasi, la funzione F e l'algoritmo di programmazione della chiave. Si partirà parlando della scelta del numero di fasi.

Maggiore è il numero di fasi e più difficile sarà svolgere l'analisi crittografica, anche per una funzione F relativamente debole. In generale, il numero di fasi viene scelto in modo che i tentativi di analisi crittografica richiedano più impegno di un semplice attacco a forza bruta per la ricerca della chiave. Questo criterio è stato certamente utilizzato nella progettazione di DES. Schneier [SCHN96] osserva che per un algoritmo DES a 16 fasi, un attacco ad analisi crittografica differenziale è leggermente meno efficiente di un attacco a forza bruta: l'attacco ad analisi crittografica differenziale richiede $2^{55.1}$ operazioni[9], mentre l'attacco a forza bruta ne richiede 2^{55}. Se DES avesse 15 o meno fasi, l'analisi crittografica differenziale richiederebbe meno impegno di una ricerca della chiave a forza bruta.

Questo criterio è interessante poiché facilita la determinazione della resistenza di un algoritmo e il confronto con algoritmi differenti. In assenza di nuove scoperte significative nell'analisi crittografica, la resistenza di un algoritmo che soddisfa questo criterio può essere giudicata esclusivamente in base alla lunghezza della chiave.

[9] Come si ricorderà, l'analisi crittografica differenziale di DES richiede 2^{47} testi in chiaro scelti a priori. Se il testo in chiaro è tutto quello che si ha a disposizione, si dovrà ricercare fra un enorme quantità di coppie testo in chiaro/testo cifrato alla ricerca delle coppie utili. Questo porta a un livello di complessità di $2^{55.1}$.

Progettazione della funzione F

Il cuore della cifratura a blocchi di Feistel consiste nella funzione F. Come si è visto, in DES questa funzione si basa sull'uso di S-box. Questo è anche il caso della maggior parte delle altre cifrature simmetriche a blocchi, come si vedrà nel Capitolo 4. Tuttavia, si possono fare alcuni commenti generali sui criteri per la progettazione della funzione F. Quindi si tratterà in modo più specifico la progettazione di una S-box.

Criteri progettuali della funzione F

La funzione F introduce l'elemento di confusione in una cifratura di Feistel. Pertanto deve essere difficile "decodificare" la sostituzione svolta da F. Un criterio ovvio è che F debba essere non lineare, come si è detto in precedenza. Meno lineare è F e più difficile sarà qualsiasi tipo di analisi crittografica. Esistono varie misure della non linearità che non rientrano però negli scopi di questo volume. In termini generali, più difficile è approssimare F con un insieme di equazioni lineari, meno sarà lineare F.

Nella progettazione della funzione F occorre considerare diversi altri criteri. Si desidera che l'algoritmo abbia un buon effetto "valanga". Questo, come si è visto, significa in generale che una variazione in un bit dell'input dovrebbe produrre una variazione di più bit dell'output. Una versione più restrittiva è il criterio **SAC (Strict Avalanche Criterion)** [WEBS86], che stabilisce che ogni bit di output j di una S-box dovrebbe cambiare con probabilità 1/2 quando viene invertito un singolo bit di input i, e questo per ogni i, j. Sebbene il criterio SAC sia espresso in termini di S-box, si può applicare un criterio simile all'intera funzione F. Questo è importante quando si considerano progetti che non includono S-box.

Un altro criterio proposto in [WEBS86] è il criterio **BIC (Bit Independence Criterion)**, che stabilisce che i bit di output j e k debbano cambiare in modo indipendente quando viene invertito un singolo bit di input i, per ogni i, j e k. I criteri SAC e BIC sembrano rafforzare l'efficacia della funzione di confusione.

Progettazione di una S-Box

Una delle aree di ricerca più intensa nel campo della cifratura simmetrica a blocchi è la progettazione delle S-box. I documenti disponibili sono troppo numerosi per poterli elencare[10]. Qui verranno introdotti solo alcuni principi generali. In pratica si vuol fare in modo che ogni modifica al vettore di input di una S-box produca una variazione apparentemente casuale nell'output. La relazione dovrebbe essere non lineare e difficile da approssimare tramite funzioni lineari.

Una caratteristica ovvia della S-box è data dalle sue dimensioni. Una S-box $n \times m$ ha n bit di input e m bit di output. DES usa S-box 6×4. L'algoritmo Blowfish, descritto nel

[10] Un ottimo riassunto degli studi sulla progettazione delle S-box che risale al 1996 si trova in [SCHN96].

Capitolo 6, usa S-box 8 × 32. Maggiori sono le S-box e più resistente sarà l'algoritmo alle analisi crittografica differenziale e lineare [SCHN96]. Ma d'altra parte, maggiori sono le dimensioni di n e maggiori (esponenzialmente) saranno le dimensioni della tabella di ricerca. Pertanto, per motivi pratici, normalmente viene imposto a n un limite compreso fra 8 e 10. Un'altra considerazione pratica è il fatto che maggiori sono le dimensioni della S-box, più difficile sarà progettarla correttamente. Le S-box sono normalmente organizzate in modo diverso rispetto a DES. Una S-box $n \times m$ è tipicamente costituita da 2^n righe di m bit ciascuna. Gli n bit di input selezionano una delle righe della S-box e gli m bit della riga rappresentano l'output. Per esempio, in una S-box 8 × 32, se l'input è 00001001, l'output è costituito dai 32 bit della riga 9 (la prima riga è la riga 0).

Mister e Adams [MIST96] propongono vari criteri per la progettazione di una S-box; tra questi suggeriscono che la S-box dovrebbe soddisfare i criteri SAC e BIC. Suggeriscono anche che tutte le combinazioni lineari delle colonne di S-box dovrebbero essere di tipo *bent*. Le funzioni bent sono una classe speciale delle funzioni booleane che prevede un'elevata non linearità in base a determinati criteri matematici [ADAM90]. Vi è un interesse crescente nella progettazione e analisi di S-box che utilizzano funzioni bent.

Un criterio correlato per le S-box viene proposto e analizzato in [HEYS95]. L'autore definisce il criterio **GA (Guaranteed Avalanche)** nel modo seguente: una S-box soddisfa il criterio GA di ordine γ se, per una variazione nell'input di un bit cambiano almeno γ bit nell'output. L'autore conclude che un GA di ordine compreso fra 2 e 5 offre forti caratteristiche di diffusione per l'algoritmo generale di crittografia.

Per S-box di maggiori dimensioni, per esempio 8 × 32, sorge la questione su quale sia il miglior metodo per selezionare le voci della S-box per poter soddisfare i criteri di cui sopra. Nyberg, che ha scritto molto sulla teoria e la pratica della progettazione delle S-box, suggerisce i seguenti approcci (tratti da [ROBS95b]).

- **Casuale:** per generare le voci delle S-box, si utilizza una generazione di numeri pseudocasuali o una tabella di cifre casuali. Questo può comportare che le box di piccole dimensioni (per esempio 6 × 4) manifestino caratteristiche indesiderabili, ma la situazione dovrebbe essere accettabile per S-box di grandi dimensioni (per esempio 8 × 32).
- **Casuale con test:** si scelgono le voci della S-box in modo casuale, se ne verificano i risultati rispetto a vari criteri, e si eliminano quelle che non passano i test.
- **Generazione manuale:** questo è un approccio più o meno manuale con il solo supporto di semplice matematica. Si tratta apparentemente della tecnica utilizzata nella progettazione di DES. Questo approccio è difficile da utilizzare con S-box di grandi dimensioni.
- **Generazione matematica:** si generano le S-box secondo certi principi matematici. In questo modo possono essere realizzate delle S-box che offrono una sicurezza dimostrabile contro l'analisi crittografica lineare e differenziale e che nel contempo manifestano buone caratteristiche di diffusione.

Una variante della prima tecnica consiste nell'impiegare S-box casuali ma anche dipendenti dalla chiave. Un esempio di questo approccio è Blowfish, descritto nel Capitolo 6, che inizia con S-box costruite con cifre pseudocasuali che vengono poi alterate utilizzando la chiave. Un grande vantaggio delle S-box dipendenti dalla chiave è che, non essendo fisse, è impossibile analizzarle a priori per individuarne i punti deboli.

Algoritmo di programmazione della chiave

L'ultima area della progettazione della cifratura a blocchi (che ha avuto un'attenzione inferiore rispetto alla progettazione delle S-box) è l'algoritmo di programmazione della chiave. Con la cifratura a blocchi di Feistel, la chiave viene utilizzata per generare una sottochiave per ogni fase. In generale, si vorrebbero selezionare delle sottochiavi in modo da massimizzare la difficoltà dell'individuazione delle singole sottochiavi e di risalire alla chiave principale. In questo campo non sono ancora stati promulgati dei principi generali.

Hall [ADAM94] suggerisce che, come minimo, la programmazione della chiave debba garantire i criteri SAC e BIC per la chiave e per il testo cifrato.

3.6 Letture e siti Web consigliati

Vi è un'ampia varietà di informazioni sulla crittografia simmetrica. In questa parte del capitolo verranno indicate solo le opere più importanti. Un'opera di riferimento fondamentale è [SCHN96]. Questo importante lavoro contiene praticamente la descrizione di qualsiasi algoritmo crittografico e protocollo pubblicato fino a oggi. L'autore raccoglie i risultati da riviste, atti di conferenze, pubblicazioni governative e standard e li organizza in una vasta e accessibile rassegna. Un altro lavoro degno di nota e molto dettagliato è [MENE97]. Una trattazione rigorosamente matematica si trova in [STIN02].

I riferimenti menzionati descrivono la crittografia a chiave pubblica oltre che simmetrica. La descrizione forse più dettagliata di DES è [SIMO95]; questo volume contiene anche un'estesa discussione sull'analisi crittografica differenziale e lineare di DES. [BARK91] è un'analisi di facile lettura e interessante sulla struttura di DES e sui potenziali approcci criptoanalitici a DES. [EFF98] indica gli attacchi a forza bruta più efficaci contro DES. [COPP94] si occupa della resistenza intrinseca di DES e della sua capacità di resistere all'analisi crittografica.

BARK91 W. Barker. *Introduction to the Analysis of the Data Encryption Standard (DES)*. Laguna Hills, CA: Aegean Park Press, 1991.

COPP94 D. Coppersmith. "The Data Encryption Standard (DES) and Its Strength Against Attacks". *IBM Journal of Research and Development*, Maggio 1994.

EFF98 Electronic Frontier Foundation. *Cracking DES: Secrets of Encryption Research, Wiretap Politics, and Chip Design*. Sebastopol, CA: O'Reilly, 1998

MENE97 A. Menezes, P. Oorshcot e S. Vanstone. *Handbook of Applied Cryptography*. Boca Raton, FL: CRC Press, 1997.

SCHN96 B. Schneier. *Applied Cryptography*. New York: Wiley, 1996.

SIMO95 M. Simovits. *The DES: An Extensive Documentation and Evaluation*. Laguna Hills, CA: Aegean Park Press, 1995.

STIN02 D. Stinson. *Cryptography: Theory and Practice*. Boca Raton, FL: CRC Press, 2002.

3.7 Termini chiave, domande di ripasso e problemi

Termini chiave

Analisi crittografica differenziale
Analisi crittografica lineare
Chiave
Cifratura a blocchi
Cifratura di Feistel
Cifratura prodotto
Confusione
DES (Data Encryption Standard)
Diffusione
Effetto valanga
Fase
Funzione F
Mapping irreversibile
Mapping reversibile
Permutazione
Sostituzione
Sottochiave

Domande di ripasso

3.1 Perché è importante studiare la cifratura di Feistel?
3.2 Qual è la differenza fra una cifratura a blocchi e una cifratura a flussi?
3.3 Perché non è pratico utilizzare una cifratura a sostituzione irreversibile del tipo rappresentato nella Tabella 3.1?
3.4 Che cos'è una cifratura a prodotto?
3.5 Qual è la differenza fra diffusione e confusione?
3.6 Quali parametri e scelte progettuali determinano l'algoritmo della cifratura di Feistel?
3.7 Qual è lo scopo delle S-box in DES?
3.8 Descrivere l'effetto valanga.
3.9 Qual è la differenza fra analisi crittografica differenziale e lineare?

Problemi

3.1 A. Nel capoverso del Paragrafo 3.1 relativo alla motivazione della struttura della cifratura di Fiestel, è stato detto che, per un blocco di n bit, il numero di mapping reversibili differenti per la cifratura a blocchi ideale è $2^n!$ Giustificare l'affermazione.

B. Nella medesima discussione, è stato detto che nella cifratura a blocchi ideale, che consente tutti i possibili mapping inversi, la dimensione della chiave è $n \times 2^n$ bit. Ma se vi sono $2^n!$ possibili mapping, dovrebbero essere necessari $\log_2 2^n!$ bit per discriminare fra i differenti mapping: la lunghezza della chiave dovrebbe dunque essere $\log_2 2^n!$ Tuttavia, $\log_2 2^n! < n \times 2^n$. Spiegare la discrepanza.

3.2 Considerare una cifratura di Fiestel composta da 16 fasi con lunghezza del blocco di 128 bit e lunghezza della chiave di 128 bit. Supporre che, per un dato k, l'algoritmo di generazione delle sottochiavi determini i valori delle prime 8 chiavi di fase $k_1, k_2 \ldots k_8$ e poi calcoli:

$$k_9 = k_8, k_{10} = k_7, k_{11} = k_6 \ldots k_{16} = k_1.$$

Supporre di avere un testo cifrato c. Spiegare come, potendo avere accesso a un oracolo crittografico, si possa decrittografare c e determinare m utilizzando una sola richiesta all'oracolo. Questo dimostra che tale cifratura è vulnerabile agli attacchi a testo in chiaro scelto. Nota: un oracolo crittografico può essere pensato come un dispositivo che, a fronte di un dato testo in chiaro, restituisce il testo cifrato corrispondente. Il funzionamento interno del dispositivo non è noto e non può essere conosciuto. Dall'oracolo si possono solamente ottenere informazioni sottoponendo richieste e osservando le risposte.

3.3 Considerare un algoritmo di crittografia a blocchi che utilizza blocchi di lunghezza n e sia $N = 2^n$. Si supponga di avere t coppie di testo in chiaro/testo cifrato $P_i, C_i = E(K, P_i)$, dove si presuppone che la chiave K selezioni uno degli N! mapping possibili. Si immagini di voler trovare K tramite una ricerca esaustiva. Si potrebbe generare la chiave K' e verificare che $C_i = E(K', P_i)$ per $1 \le i \le t$. Se K' consente di crittografare ciascun P_i nel C_i corretto, si ha una ragionevole certezza che $K = K'$. Tuttavia può capitare che i mapping $E(K, \bullet)$ e $E(K', \bullet)$ siano perfettamente concordi sulle t coppie di testo in chiaro/testo cifrato P_i, C_i ma su nessun'altra coppia.

A. Qual è la probabilità che $E(K, \bullet)$ e $E(K', \bullet)$ siano in realtà mapping distinti?

B. Qual è la probabilità che $E(K, \bullet)$ e $E(K', \bullet)$ concordino su altre t' coppie testo in chiaro/testo cifrato dove $0 \le t' \le N - t$?

3.4 Sia π una permutazione degli interi 0, 1, 2, ... (2^n-1) tale che $\pi(m)$ dia il valore permutato di m, con $0 \le m < 2^n$. In altre parole, π mappa l'insieme di interi di n bit su se stessa in modo che non possano esistere due interi che vengono mappati nello stesso intero. DES è una permutazione di questo tipo per interi di 64 bit. Si dice che π ha un punto fisso m se $\pi(m) = m$. In pratica, se π è un mapping di crittografia, un punto fisso corrisponde a un messaggio che è uguale alla sua versione crittografata. Si è interessati alla probabilità che π non abbia punti fissi. Dimostrare il risultato inaspettato che oltre il 60% dei mapping ha almeno un punto fisso.

3.5 Considerare la sostituzione definita dalla riga 1 della S-box S_1 nella Tabella 3.3. Mostrare un diagramma a blocchi simile a quello della Figura 3.1 che corrisponda a questa sostituzione.

3.6 Calcolare i bit numero 1, 16, 33 e 48 all'uscita della prima fase della decrittografia DES, ipotizzando che il blocco di testo cifrato e la chiave esterna siano costituiti esclusivamente da cifre uno.

3.7 Supporre che la funzione DES mappi ogni 32 bit dell'input R, indipendentemente dal valore dell'input K, a:

A. stringhe di 32 bit composte esclusivamente da cifre uno,

B. complemento bit a bit di R.

Quale funzione calcolerebbe DES?

Come apparirebbe la decrittografia?

Suggerimento: utilizzare le seguenti proprietà dell'operazione XOR:

$$(A \oplus B) \oplus C = A \oplus (B \oplus C)$$
$$A \oplus A = 0$$
$$A \oplus 0 = A$$

dove A, B e C sono stringhe di n bit,
0 è una stringa di n bit nulli,
1 è una stringa di n bit a 1.

3.8 Questo problema fornisce un esempio numerico di crittografia utilizzando una versione mono-fase di DES. Si inizia con lo stesso schema di bit per la chiave e per il testo in chiaro, ovvero:

In notazione esadecimale: 0 1 2 3 4 5 6 7 8 9 A B C D E F
In notazione binaria: 0000 0001 0010 0011 0100 0101 0110 0111
1000 1001 1010 1011 1100 1101 1110 1111

A. Derivare K_1, la sottochiave della prima fase.
B. Derivare L_0, R_0.
C. Espandere R_0 per ottenere $E[R_0]$, dove E[•] è la funzione espansione di Figura 3.8.
D. Calcolare $A = E[R_0] \oplus K_1$.
E. Raggruppare i risultati a 48 bit di D in insiemi di 6 bit e valutare le corrispondenti sostituzioni nelle S-box.
F. Concatenare i risultati di E per ottenere un risultato di 32 bit, B.
G. Applicare la permutazione per ottenere P(B).
H. Calcolare $R_1 = P(B) \oplus L_0$.
I. Scrivere il testo cifrato.

3.9 Dimostrare che la decrittografia DES è l'inverso della crittografia DES.

3.10 Lo scambio a 32 bit dopo la sedicesima iterazione dell'algoritmo DES è necessario per rendere invertibile il processo di crittografia, applicando semplicemente l'algorimo al testo cifrato invertendo l'ordine delle chiavi. Questo è stato dimostrato nel Problema 3.7. Tuttavia può non essere ancora perfettamente chiaro perché sia necessario eseguire lo scambio a 32 bit. Per dimostrarlo, risolvere i seguenti esercizi. Innanzitutto alcune notazioni:

$A \| B$ = la concatenazione dei bit delle stringhe A e B.
$T_i(R \| L)$ = la trasformazione definita dalla i-esima iterazione dell'algoritmo di crittografia, per $1 \le i \le 16$.
$TD_i(R \| L)$ = la trasformazione definita dalla i-esima iterazione dell'algoritmo di decrittografia per $1 \le i \le 16$.
$T_{17}(R \| L)$ = L‖R. Trasformazione che viene eseguita dopo la sedicesima iterazione dell'algoritmo di crittografia.

A. Mostrare che la composizione $TD_1(IP(IP^{-1}(T_{17}(T_{16}(L_{15} \| R_{15})))))$ è equivalente alla trasformazione che scambia le due metà di 32 bit L_{15} e R_{15}. Ovvero dimostrare che:

$$TD_1(IP(IP^{-1}(T_{17}(T_{16}(L_{15} \| R_{15}))))) = R_{15} \| L_{15}$$

B. Ora supporre che venga eliminato l'ultimo scambio a 32 bit nell'algoritmo di crittografia. Si vorrebbe che la seguente uguaglianza rimanesse vera:

$$TD_1(IP(IP^{-1}(T_{16}(L_{15} \| R_{15})))) = L_{15} \| R_{15}$$

È così?

3.11 Confrontare la tabella di permutazione iniziale (Tabella 3.2A) con la Tabella 3.4B. Le strutture sono simili? In caso affermativo descrivere le analogie. Quali conclusioni si possono trarre da questa analisi?

3.12 Quando si usa l'algoritmo DES per la decrittografia, le 16 chiavi (K_1, K_2, . . . , K_{16}) vengono utilizzate in ordine inverso. Pertanto il lato destro della Figura 3.5 non è più valido. Progettare uno schema di generazione della chiave con la programmazione dello scorrimento appropriato (analogamente alla Tabella 3.4D) per il processo di decrittografia.

3.13. A. Se M' è il complemento bit-a-bit di M, dimostrare che se si prende il complemento del blocco di testo in chiaro e il complemento della chiave di crittografia, il risultato della crittografia con questi valori è il complemento del testo cifrato originario. Ovvero:
Se Y = E(K, X)
Allora Y' = E(K', X')
Suggerimento: iniziare dimostrando che per ogni coppia di stringhe di bit di uguale lunghezza, A e B, (A ⊕ B)' = A' ⊕ B' .

B. Si è detto che un attacco a forza bruta su DES richiede la ricerca in uno spazio di 2^{56} chiavi. I risultati della parte A di questo problema cambiano questa affermazione?

3.14 Mostrare che in DES i primi 24 bit di ciascuna sottochiave provengono dallo stesso insieme di 28 bit della chiave iniziale e che i secondi 24 bit di ciascuna sottochiave provengono da un sottoinsieme disgiunto dei 28 bit della chiave iniziale.

3.15 La proprietà di non linearità di qualsiasi cifratura a blocchi è fondamentale per la sua sicurezza. Infatti, si supponga di avere una cifratura lineare a blocchi EL che crittografa blocchi di 128 bit di testo in chiaro in 128 bit di testo cifrato. Sia EL(k, m) il risultato della crittografia del messaggio di 128 bit m con chiave k (la lunghezza di k è irrilevante). Dunque, per tutte le configurazioni di 128 bit m_1 e m_2 vale:

$$EL(k,[m_1 \oplus m_2]) = EL(k,m_1) \oplus EL(k,m_2)$$

Descrivere ora come un avversario che abbia 128 *testi cifrati scelti* possa decrittografare qualsiasi testo cifrato senza conoscere la chiave segreta k. Nota: "testo cifrato scelto" significa che l'avversario ha la possibilità di scegliere un testo cifrato e ottenerne la decifratura. In questo caso si hanno 128 coppie di testo in chiaro/testo cifrato e si ha la possibilità di scegliere i testi cifrati.

3.16 Nota: il problema seguente fa riferimento al DES semplificato descritto nell'Appendice C. Far riferimento alla Figura C.2 che rappresenta la generazione delle chiavi per S-DES.

A. Quanto è importante la funzione di permutazione P10 iniziale?
B. Quanto sono importanti le due funzioni di scorrimento LS-1?

3.17 Le equazioni per le variabili q e r per S-DES sono definite nella sezione sull'analisi di S-DES. Fornire le equazioni per s e t.

3.18 Utilizzando S-DES, decrittografare a mano la stringa 10100010 utilizzando la chiave 0111111101. Mostrare i risultati intermedi dopo ciascuna funzione (IP, F_K, SW, F_K, IP^{-1}). Decodificare i primi 4 bit della stringa di testo in chiaro in una lettera e i secondi 4 bit in un'altra lettera posto che le lettere da A a P sono codificate in base 2 (A = 0000, B = 0001, . . . , P = 1111). *Suggerimento:* come controllo intermedio, dopo l'applicazione di SW, la stringa dovrebbe essere 00010011.

Problemi di programmazione

3.19 Scrivere un programma di crittografia/decrittografia a blocchi che utilizzi la cifratura generale a sostituzione.

3.20 Scrivere un programma di crittografia/decrittografia a blocchi che utilizzi S-DES. Come verifica del corretto funzionamento, utilizzare il testo in chiaro, il testo cifrato e la chiave del Problema 3.15.

Capitolo 4

I campi finiti

Concetti essenziali

- Un **campo** è un insieme di elementi tra i quali sono definite due operazioni (somma e moltiplicazione) che godono delle proprietà dell'aritmetica ordinaria, quali le proprietà di chiusura, associativa, commutativa e distributiva, e che sono caratterizzati inoltre dall'esistenza degli inversi additivi e moltiplicativi.
- L'**aritmetica modulare** è un tipo di aritmetica fra interi che riduce tutti i numeri a un insieme fisso $\{0 . . n - 1\}$ per un certo intero n. Qualsiasi intero al di fuori di questo insieme viene ridotto a un elemento dell'insieme prendendo il resto della divisione intera per n.
- Il **massimo comun divisore** fra due interi è il più grande intero positivo che li divide esattamente (senza dare resto).
- I campi finiti sono di grande interesse in vari ambiti della crittografia. Un **campo finito** è semplicemente un campo con un numero finito di elementi. Si può dimostrare che l'ordine di un campo finito (il numero di elementi del campo) deve essere una potenza di un numero primo p^n, dove n è un intero positivo.
- I campi finiti di ordine p possono essere definiti utilizzando l'aritmetica **modulo** p.
- I campi finiti di ordine p^n, per $n > 1$, possono essere definiti utilizzando l'aritmetica **polinomiale**.

I campi finiti sono molto importanti in crittografia in quanto esistono vari algoritmi crittografici, in particolare l'algoritmo AES (Advanced Encryption Standard) e la crittografia a curva ellittica, che si basano proprio sulle proprietà dei campi finiti.

Il capitolo si apre con una breve panoramica dei concetti di gruppo, anello e campo. Questa parte è piuttosto astratta: il lettore la potrà scorrere rapidamente in una prima lettura. Affronta successivamente alcuni aspetti elementari dell'aritmetica modulare e dell'algoritmo di Euclide. A questo punto si avranno tutte le basi teoriche necessarie per parlare dei campi finiti nella forma GF(p) dove p è un numero primo. Quindi sarà necessario fornire

altre informazioni di base, questa volta sull'aritmetica polinomiale. Il capitolo si conclude con una discussione dei campi finiti della forma GF(2^n) dove n è un intero positivo.

I concetti e le tecniche della teoria dei numeri sono piuttosto astratti e spesso è difficile acquisirli intuitivamente senza l'ausilio di un esempio [RUBI97]. Pertanto questo capitolo e il Capitolo 8 comprendono una grande quantità di esempi.

4.1 Gruppi, anelli e campi

I gruppi, gli anelli e i campi sono elementi fondamentali dell'algebra moderna astratta. Nell'algebra astratta si è interessati agli insiemi sui cui elementi si può operare in modo algebrico; ovvero si possono combinare due elementi di un insieme, anche in più modi, per ottenere un terzo elemento dell'insieme. Queste operazioni sono soggette a specifiche regole che definiscono la natura dell'insieme. Per convenzione, la notazione per le due principali operazioni sugli elementi dell'insieme coincide solitamente con la notazione della somma e della moltiplicazione sui numeri. Tuttavia è importante notare che, nell'algebra astratta, non esistono solo le normali operazioni aritmetiche. Tutto risulterà più chiaro procedendo nella discussione.

Gruppi

Un **gruppo** G, indicato anche come $\{G, \bullet\}$, è un insieme di elementi affiancato da un'operazione binaria, rappresentata dal simbolo $\bullet$, che associa a ciascuna coppia (a, b) di elementi di G un elemento $(a \bullet b)$ sempre di G in modo tale che valgono i seguenti assiomi[1]:

(A1) Chiusura: se a e b appartengono a G allora anche $a \bullet b$ appartiene a G.
(A2) Associatività: $a \bullet (b \bullet c) = (a \bullet b) \bullet c$ per ogni a, b, c in G.
(A3) Elemento identità: vi è un elemento in G tale che $a \bullet e = e \bullet a = a$ per ogni a in G.
(A4) Elemento inverso: per ciascun a in G vi è un elemento a' sempre in G tale che $a \bullet a' = a' \bullet a = e$.

Sia N_n un insieme di n simboli distinti che, per comodità, indichiamo come $\{1, 2, \ldots, n\}$. Una permutazione di n simboli distinti è un mapping uno-a-uno da N_n a N_n. Si definisca S_n come l'insieme di tutte le permutazioni di n simboli distinti. Ciascun elemento di S_n è rappresentato da una permutazione degli interi contenuti in $\{1, 2, \ldots, n\}$. È facile dimostrare che S_n è un gruppo.

A1 Se $\pi, \rho \in S$ allora il mapping composito $\pi \bullet \rho$ si forma permutando gli elementi di ρ in base alla permutazione π. Per esempio $\{3, 2, 1\} \bullet \{1, 3, 2\} = \{2, 3, 1\}$. Chiaramente, $\pi \bullet \rho \in S_n$.
A2 La composizione dei mapping è anch'essa evidentemente associativa.
A3 Il mapping identità è la permutazione che non altera l'ordine degli n elementi. Per S_3, l'elemento identità è $\{1, 2, \ldots, n\}$.

[1] $\bullet$ è un operatore generico che può essere la somma, il prodotto o qualche altra operazione matematica.

A4 Per ogni $\pi \in S_n$, il mapping che annulla la permutazione definita da π è l'elemento inverso di π. Vi sarà sempre questo inverso. Per esempio, $\{2, 3, 1\} \bullet \{3, 1, 2\} = \{1, 2, 3\}$.

Se un gruppo contiene un numero di elementi finito, si chiama **gruppo finito** e l'**ordine** del gruppo è uguale al numero degli elementi del gruppo. Altrimenti il gruppo è chiamato **gruppo infinito**.

Un gruppo è chiamato **abeliano** se soddisfa la seguente condizione aggiuntiva:

(A5) Commutatività $a \bullet b = b \bullet a$ per ogni a, b in G.

L'insieme degli interi (positivi, negativi e 0) con la somma è un gruppo abeliano. L'insieme dei numeri reali diversi da zero con la moltiplicazione è un gruppo abeliano. L'insieme S_n dell'esempio precedente è un gruppo ma non abeliano per $n > 2$.

Quando l'operazione del gruppo è la somma, l'elemento identità è 0; l'elemento inverso di a è $-a$ e la sottrazione è definita dalla seguente regola $a - b = a + (-b)$.

Gruppo ciclico

Si definisce l'elevamento a potenza all'interno di un gruppo come l'applicazione ripetuta dell'operatore del gruppo, così che $a^3 = a \bullet a \bullet a$. Inoltre si definiscono $a^0 =$ e (l'elemento identità) e $a^{-n} = (a')^n$. Un gruppo G è detto **ciclico** se ogni elemento di G è una potenza a^k (k intero) di un elemento fisso $a \in G$. Si dice che l'elemento a **genera** il gruppo G o è un **generatore** di G. Un gruppo ciclico è sempre abeliano e può essere finito o infinito.

Il gruppo additivo degli interi è un gruppo ciclico infinito generato dall'elemento 1. In questo caso le potenze sono interpretate in modo additivo in modo che n è la n-esima potenza di 1.

Anelli

Un **anello** R, rappresentato anche come $\{R, +, \times\}$, è un insieme di elementi con due operazioni binarie, la *somma* e la *moltiplicazione*,[2] tali che per ogni a, b, c di R valgono i seguenti assiomi.

(A1-A5) R è un gruppo abeliano rispetto alla somma, ovvero R soddisfa gli assiomi da A1 a A5. Nel caso di gruppo additivo, si rappresenta l'elemento identità come 0 e l'inverso di a come $-a$.
(M1) Chiusura rispetto alla moltiplicazione: se a e b appartengono a R allora anche ab appartiene a R.
(M2) Associatività della moltiplicazione: $a(bc) = (ab)c$ per ogni a, b, c in R.
(M3) Leggi distributive: $a(b + c) = ab + ac$ per ogni a, b, c in R.
$(a + b)c = ac + bc$ per ogni a, b, c in R.

[2] In generale per la moltiplicazione non si utilizzerà il simbolo $\times$ ma semplicemente si concatenano i due elementi.

In pratica un anello è un insieme in cui si possono eseguire la somma, la sottrazione [a – b = a + (–b)] e la moltiplicazione senza uscire dall'insieme stesso.

Rispetto alla somma e alla moltiplicazione, l'insieme di tutte le matrici quadrate n di numeri reali è un anello R.

Un anello si dice **commutativo** se soddisfa la seguente condizione aggiuntiva.

(M4) Commutatività della moltiplicazione: $ab = ba$ per ogni a, b in R.

Sia S l'insieme degli interi pari (positivi, negativi e 0) con le normali operazioni di somma e moltiplicazione. S è un anello commutativo. L'insieme di tutte le matrici quadrate n definite nell'esempio precedente non è un anello commutativo.

Un **dominio integrale** è un anello commutativo che obbedisce ai seguenti assiomi.

(M5) Identità moltiplicativa: vi è un elemento 1 in R tale che $a1 = 1a = a$ per ogni a in R.

(M6) Annullamento del prodotto: se a, b in R e $ab = 0$, allora o $a = 0$ o $b = 0$.

Sia S l'insieme degli interi positivi, negativi e 0, con le normali operazioni di somma e moltiplicazione. S è un dominio integrale.

Campi

Un **campo** F, rappresentato anche come $\{F, +, \times\}$, è un insieme di elementi con due operazioni binarie, chiamate *somma* e *moltiplicazione*, tali che per ogni a, b, c di F, valgano i seguenti assiomi.

(A1-M6) F è un dominio integrale, ovvero F soddisfa gli assiomi da A1 a A5 e da M1 a M6.

(M7) Inverso moltiplicativo: per ogni a in F, eccetto 0, vi è un elemento a^{-1} in F tale che $aa^{-1} = (a^{-1})a = 1$.

In pratica un campo è un insieme in cui si possono eseguire somme, sottrazioni, moltiplicazioni e divisioni senza uscire dall'insieme. La divisione è definita dalla seguente regola: $a/b = a(b^{-1})$.

Fra gli esempi più familiari dei campi vi sono i numeri razionali, i numeri reali e i numeri complessi. Si noti che l'insieme di tutti gli interi non è un campo poiché non tutti gli elementi dell'insieme hanno un inverso moltiplicativo; infatti solo gli elementi 1 e –1 hanno un inverso moltiplicativo intero.

La Figura 4.1 riepiloga gli assiomi che definiscono i gruppi, gli anelli e i campi.

4.2 Aritmetica modulare

Dato un intero positivo n e un intero a non negativo, se si divide a per n, si ottiene un quoziente intero q e un resto intero r che soddisfano alla seguente relazione:

$$a = qn + r \quad 0 \le r < n; \quad q = \lfloor a/n \rfloor \tag{4.1}$$

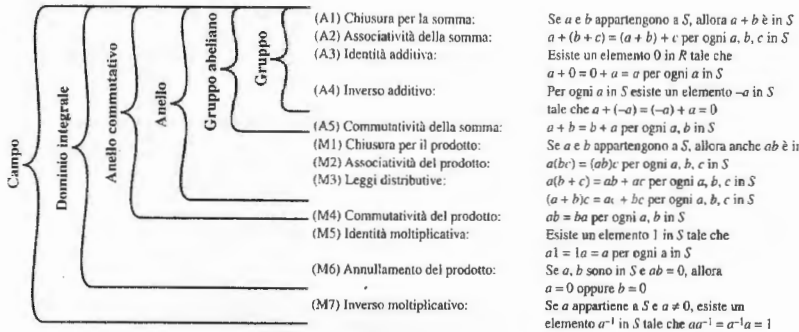


Figura 4.1 Gruppi, anelli e campi.

dove $\lfloor x \rfloor$ è il più grande intero minore o uguale a x.

La Figura 4.2 dimostra che, dato a e un n positivo, è sempre possibile trovare q e r che soddisfano la relazione precedente. Si rappresentino gli interi su una retta; a si troverà in qualche punto di questa retta (in questo caso viene impiegato un a positivo ma si può fare una dimostrazione analoga per a negativo). Partendo da 0, si proceda a n, $2n$ fino a qn tale che $qn \le a$ e $(q + 1)n > a$. La distanza fra qn e a è r e sono stati quindi trovati i valori univoci di q e r.

$a = 11;$ $n = 7;$ $11 = 1 \times 7 + 4;$ $r = 4$ $q = 1$
$a = -11;$ $n = 7;$ $-11 = (-2) \times 7 + 3;$ $r = 3$ $q = -2$

Se a è un intero e n è un intero positivo, si definisce a mod n come il resto della divisione di a per n. Pertanto, per ogni intero a si può sempre scrivere:

$$a = \lfloor a/n \rfloor \times n + (a \bmod n)$$

$$11 \bmod 7 = 4; \qquad -11 \bmod 7 = 3$$

Due interi a e b sono detti **congruenti modulo** n se $(a \bmod n) = (b \bmod n)$. La relazione può essere scritta come $a \equiv b \pmod n$[3].

$73 \equiv 4 \pmod{23}$; $21 \equiv -9 \pmod{10}$

Divisori

Si dice che un valore b diverso da 0 divide a se $a = mb$ per qualche m, dove a, b e m sono numeri interi. In pratica b divide a se la loro divisione non dà resto. Per indicare che b divide a viene utilizzata la notazione $b|a$. Inoltre se $b|a$, si dice che b è un divisore di a.

[3] Si è utilizzato l'operatore mod con due finalità: prima, come operatore binario che produce un resto, come nell'espressione a mod b; seconda come relazione di congruenza che indica l'equivalenza di due interi, come nell'espressione $a = b \pmod n$. Per distinguere le due modalità, il termine *mod* viene racchiuso tra parentesi quando si tratta di una relazione di congruenza – come è comune, sebbene non universale, nella letteratura scientifica. Consultare l'Appendice D per ulteriori informazioni.

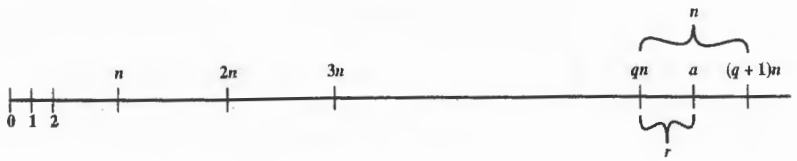


Figura 4.2 La relazione $a = qn + r, 0 \le r < n$.

I divisori positivi di 24 sono 1, 2, 3, 4, 6, 8, 12 e 24.

Valgono le seguenti relazioni.

- Se $a|1$, allora $a = \pm 1$.
- Se $a|b$ e $b|a$, allora $a = \pm b$.
- Qualsiasi $b \neq 0$ divide 0.
- Se $b|g$ e $b|h$, allora $b|(mg + nh)$ per m e n interi arbitrari.

Per provare questo ultimo punto, si noti che:

Se $b|g$, allora g ha la forma $g = b \times g_1$ per un certo intero g_1.
Se $b|h$, allora h ha la forma $h = b \times h_1$ per un certo intero h_1.

Quindi:

$$mg + nh = mbg_1 + nbh_1 = b \times (mg_1 + nh_1)$$

e pertanto b divide $mg + nh$.

$b = 7; g = 14; h = 63; m = 3; n = 2$.
7|14 e 7|63. Per mostrare $7|(3 \times 14 + 2 \times 63)$
Si ottiene $(3 \times 14 + 2 \times 63) = 7(3 \times 2 + 2 \times 9)$
E ovviamente si ha $7|7(3 \times 2 + 2 \times 9)$

Si noti che se $a \equiv 0 \pmod{n}$ allora $n|a$.

Proprietà delle congruenze

Le congruenze godono delle seguenti proprietà.

1. $a \equiv b \pmod{n}$ se $n|(a - b)$
2. $a \equiv b \pmod{n}$ implica $b \equiv a \pmod{n}$
3. $a \equiv b \pmod{n}$ e $b \equiv c \pmod{n}$ implicano $a \equiv c \pmod{n}$.

Per dimostrare il primo punto, se $n|(a - b)$ allora esiste k tale che $(a - b) = kn$, quindi si può scrivere $a = b + kn$. Pertanto, ($a \bmod n$) = (resto quando $b + kn$ è diviso per n) = (resto quando b è diviso per n) = ($b \bmod n$).

$23 \equiv 8 \pmod{5}$	poiché	$23 - 8 = 15 = 5 \times 3$
$-11 \equiv 5 \pmod{8}$	poiché	$-11 - 5 = -16 = 8 \times (-2)$
$81 \equiv 0 \pmod{27}$	poiché	$81 - 0 = 81 = 27 \times 3$

I punti rimanenti sono facilmente dimostrabili.

Operazioni dell'aritmetica modulare

Si noti che, per definizione (Figura 4.2), l'operatore (mod n) mappa tutti gli interi in un insieme di interi $\{0, 1, \ldots (n - 1)\}$. Questo suggerisce la seguente domanda: è possibile svolgere operazioni aritmetiche nei confini di questo insieme? In effetti è possibile, e questa tecnica è chiamata **aritmetica modulare**.

L'aritmetica modulare esibisce le seguenti proprietà.

1. $[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$
2. $[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$
3. $[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$

Verrà dimostrata la prima proprietà. Si definisca $(a \bmod n) = r_a$ e $(b \bmod n) = r_b$. Allora si può scrivere $a = r_a + jn$ per un intero j e scrivere $b = r_b + kn$ per un intero k. Quindi:

$$\begin{aligned}(a + b) \bmod n &= (r_a + jn + r_b + kn) \bmod n \\ &= (r_a + r_b + (k + j)\, n) \bmod n \\ &= (r_a + r_b) \bmod n \\ &= [(a \bmod n) + (b \bmod n)] \bmod n\end{aligned}$$

Le altre proprietà sono facili da dimostrare. Ecco alcuni esempi di queste tre proprietà:

$11 \bmod 8 = 3; \quad 15 \bmod 8 = 7$
$[(11 \bmod 8) + (15 \bmod 8)] \bmod 8 = 10 \bmod 8 = 2$
$(11 + 15) \bmod 8 = 26 \bmod 8 = 2$
$[(11 \bmod 8) - (15 \bmod 8)] \bmod 8 = -4 \bmod 8 = 4$
$(11 - 15) \bmod 8 = -4 \bmod 8 = 4$
$[(11 \bmod 8) \times (15 \bmod 8)] \bmod 8 = 21 \bmod 8 = 5$
$11 \times 15 \bmod 8 = 165 \bmod 8 = 5$

L'elevamento a potenza viene ottenuto ripetendo la moltiplicazione come nella normale aritmetica. Questo argomento verrà trattato in dettaglio nel Capitolo 8.

Per trovare $11^7 \bmod 13$ si può procedere nel seguente modo:

$11^2 = 121 \equiv 4 \pmod{13}$
$11^4 = (11^2)^2 \equiv 4^2 \equiv 3 \pmod{13}$
$11^7 = 11 \times 4 \times 3 = 132 \equiv 2 \pmod{13}$

Nell'aritmetica modulare valgono quindi le stesse regole per la somma, la sottrazione e la moltiplicazione che vengono impiegate nell'aritmetica ordinaria.

La Tabella 4.1 illustra la somma e la moltiplicazione in modulo 8. Osservando la somma, i risultati sono immediati e vi è uno schema regolare nella matrice. Entrambe le matrici sono simmetriche rispetto alla diagonale principale, in accordo alla proprietà commutativa dell'addizione e della moltiplicazione. Inoltre, come nella comune somma, vi è un inverso additivo o negativo per ciascun intero nell'aritmetica modulare. In questo caso, il negativo di un intero x è l'intero y tale che $(x + y) \bmod 8 = 0$. Per trovare l'inverso

additivo di un intero nella colonna di sinistra, si deve scorrere la riga corrispondente della matrice per trovare il valore 0; l'inverso additivo è l'intero che si trova nella prima riga della colonna; pertanto (2 + 6) mod 8 = 0. Analogamente le voci relative alla moltiplicazione sono molto semplici. Nell'aritmetica ordinaria, vi è un inverso moltiplicativo (reciproco) per ciascun intero. Nell'aritmetica modulare modulo 8, l'inverso moltiplicativo di x è l'intero y tale che $(x \times y) \bmod 8 = 1 \bmod 8$. Ora per trovare l'inverso moltiplicativo di un intero dalla tabella delle moltiplicazioni, basta scorrere la matrice nella riga corrispondente a tale intero per trovare il valore 1; l'inverso moltiplicativo è l'intero che si trova nella prima riga di tale colonna; pertanto $(3 \times 3) \bmod 8 = 1$. Si noti che non tutti gli interi modulo 8 hanno un inverso moltiplicativo; l'argomento verrà ripreso più avanti.

Proprietà dell'aritmetica modulare

Si definisca l'insieme Z_n come l'insieme degli interi non negativi minori di n:

$$Z_n = \{0, 1, \ldots, (n-1)\}$$

Questo viene chiamato **insieme dei resti** o delle **classi dei resti** modulo n. Per essere più precisi, ciascun intero di Z_n rappresenta una classe dei resti. Si possono etichettare le classi dei resti modulo n come [0], [1], [2], . . . , [n – 1], dove:

$$[r] = \{a: a \text{ è un intero, } a \equiv r \pmod{n}\}$$

Le classi dei resti modulo 4 sono:

$$[0] = \{\ldots, -16, -12, -8, -4, 0, 4, 8, 12, 16, \ldots\}$$
$$[1] = \{\ldots, -15, -11, -7, -3, 1, 5, 9, 13, 17, \ldots\}$$
$$[2] = \{\ldots, -14, -10, -6, -2, 2, 6, 10, 14, 18, \ldots\}$$
$$[3] = \{\ldots, -13, -9, -5, -1, 3, 7, 11, 15, 19, \ldots\}$$

Di tutti gli interi di una classe dei resti, il più piccolo intero non negativo è quello normalmente utilizzato per rappresentare la classe. La ricerca del più piccolo intero non negativo al quale k è congruente modulo n si dice **riduzione di k modulo n**.

L'aritmetica modulare in Z_n, soddisfa le proprietà riportate nella Tabella 4.2. Pertanto Z_n è un anello commutativo con un elemento di identità moltiplicativa (Figura 4.1).

Vi è una peculiarità dell'aritmetica modulare che la distingue dalla normale aritmetica. Innanzitutto si osservi che, come nell'aritmetica ordinaria, si può scrivere la seguente relazione:

$$\textbf{se } (a+b) \equiv (a+c) \pmod{n} \quad \textbf{allora } b \equiv c \pmod{n} \qquad (4.2)$$

$(5 + 23) \equiv (5 + 7) \pmod 8$; $23 \equiv 7 \pmod 8$

Tabella 4.1 Aritmetica modulo 8.

A. Somma modulo 8

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	0
2	2	3	4	5	6	7	0	1
3	3	4	5	6	7	0	1	2
4	4	5	6	7	0	1	2	3
5	5	6	7	0	1	2	3	4
6	6	7	0	1	2	3	4	5
7	7	0	1	2	3	4	5	6

B. Moltiplicazione modulo 8

X	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	0	2	4	6
3	0	3	6	1	4	7	2	5
4	0	4	0	4	0	4	0	4
5	0	5	2	7	4	1	6	3
6	0	6	4	2	0	6	4	2
7	0	7	6	5	4	3	2	1

C. Inversi additivi e moltiplicativi modulo 8

w	$-w$	w^{-1}
0	0	–
1	7	1
2	6	–
3	5	3
4	4	–
5	3	5
6	2	–
7	1	7

L'Equazione 4.2 è coerente con l'esistenza dell'inverso additivo. Sommando l'inverso additivo di a ai due lati dell'Equazione 4.2 si avrà:

$$((-a) + a + b) \equiv ((-a) + a + c) \pmod{n}$$
$$b \equiv c \pmod{n}$$

Tuttavia la seguente affermazione è vera solo con la condizione indicata:

$$\textbf{se } (a \times b) \equiv (a \times c) \pmod{n} \textbf{ allora } b \equiv c \pmod{n} \qquad \textbf{se } a \text{ e } n \text{ sono primi fra loro} \qquad (4.3)$$

Tabella 4.2. Proprietà dell'aritmetica modulare per gli interi in Z_n.

Proprietà	Espressione
Proprietà commutative	$(w + x) \bmod n = (x + w) \bmod n$ $(w \times x) \bmod n = (x \times w) \bmod n$
Proprietà associative	$[(w + x) + y] \bmod n = [w + (x + y)] \bmod n$ $[(w \times x) \times y] \bmod n = [w \times (x \times y)] \bmod n$
Proprietà distributive	$[w \times (x + y)] \bmod n = [(w \times x) + (w \times y)] \bmod n$ $[w + (x \times y)] \bmod n = [(w + x) \times (w + y)] \bmod n$
Identità	$(0 + w) \bmod n = w \bmod n$ $(1 \times w) \bmod n = w \bmod n$
Inverso additivo (-w)	Per ogni $w \in Z_n$, esiste z tale che $w + z \equiv 0 \pmod{n}$

Si dice che due interi sono **primi relativi** se l'unico fattore intero positivo che hanno in comune è 1. Analogamente al caso dell'Equazione 4.2, si può dire che l'Equazione 4.3 è coerente con l'esistenza di un inverso moltiplicativo. Applicando l'inverso moltiplicativo di a a entrambi i lati dell'Equazione 4.2 si ha:

$$((a^{-1})ab) \equiv ((a^{-1})ac) (\text{mod } n)$$
$$b \equiv c (\text{mod } n)$$

Per dimostrare ciò si consideri un esempio in cui non vale la condizione indicata nell'Equazione 4.3. Gli interi 6 e 8 non sono primi relativi in quanto hanno in comune il fattore 2. Si ha dunque:

$$6 \times 3 = 18 \equiv 2 \text{ (mod 8)}$$
$$6 \times 7 = 42 \equiv 2 \text{ (mod 8)}$$

Ma $3 \neq 7$ (mod 8)

Il motivo di questo strano risultato è che per ogni modulo generale n, un moltiplicatore a applicato agli interi da 0 a $n - 1$ non produrrà un insieme completo di resti se a e n hanno dei fattori in comune.

Con $a = 6$ e $n = 8$

Z_8	0	1	2	3	4	5	6	7
Multipli di 6	0	6	12	18	24	30	36	42
Resti	0	6	4	2	0	6	4	2

Dato che quando si moltiplica per 6 non si ha un insieme di resti completo, si avrà che lo stesso resto sarà prodotto da più di un intero di Z_n. In particolare, $6 \times 0 \bmod 8 = 6 \times 4 \bmod 8$; $6 \times 1 \bmod 8 = 6 \times 5 \bmod 8$ e così via. Poiché questo è un mapping molti-a-1, non esiste un inverso univoco per l'operazione di moltiplicazione.

Tuttavia, se si prende $a = 5$ e $n = 8$, che hanno come unico fattore in comune 1:

Z_8	0	1	2	3	4	5	6	7
Multipli di 5	0	5	10	15	20	25	30	35
Resti	0	5	2	7	4	1	6	3

La riga dei resti contiene tutti gli interi di Z_8, con ordine differente.

In generale, un intero ha un inverso moltiplicativo in Z_n se tale intero è primo relativo di n. La Tabella 4.1C mostra che gli interi 1, 3, 5 e 7 hanno un inverso moltiplicativo in Z_8 mentre 2, 4 e 6 non l'hanno.

4.3 L'algoritmo di Euclide

L'algoritmo di Euclide è una delle tecniche di base della teoria dei numeri ed è una semplice procedura per determinare il massimo comune divisore fra due interi positivi.

Il massimo comune divisore

Si ricordi che dati due interi a e b, b diverso da zero, si dice che b è un divisore di a se esiste un intero m per cui vale: $a = mb$. Per indicare il **massimo comune divisore** (greatest common divisor) di a e b si utilizzerà la notazione $\gcd(a, b)$. L'intero positivo c si dice massimo comune divisore di a e b se valgono le seguenti condizioni.

1. c è un divisore di a e di b.
2. Qualsiasi divisore di a e di b è divisore di c.

Ecco una definizione equivalente:

$$\gcd(a, b) = \max[k, \text{ tale che } k|a \text{ e } k|b].$$

Poiché si richiede che il massimo comune divisore sia positivo, $\gcd(a, b) = \gcd(a, -b) = \gcd(-a, b) = \gcd(-a, -b)$. In generale $\gcd(a, b) = \gcd(|a|, |b|)$.

$$\gcd(60, 24) = \gcd(60, -24) = 12$$

Inoltre, poiché tutti gli interi non nulli sono divisori di 0, si ha che $\gcd(a, 0) = |a|$.

Si è affermato che due interi a e b sono primi relativi se l'unico fattore intero positivo comune è 1. Questo equivale dire che a e b sono primi relativi se $\gcd(a, b) = 1$.

I numeri 8 e 15 sono primi relativi poiché i divisori positivi di 8 sono 1, 2, 4 e 8 e i divisori positivi di 15 sono 1, 3, 5 e 15 e dunque 1 è l'unico intero che compare in entrambe le liste.

Ricerca del massimo comune divisore

L'algoritmo di Euclide si basa sul seguente teorema: per ogni intero non negativo a e ogni intero positivo b:

$$\gcd(a, b) = \gcd(b, a \bmod b) \tag{4.4}$$

$$\gcd(55, 22) = \gcd(22, 55 \bmod 22) = \gcd(22, 11) = 11$$

Per dimostrare che l'Equazione 4.4 funziona, si supponga che $d = \gcd(a, b)$. Si avrà, per definizione di gcd, che $d|a$ e $d|b$. Per ogni intero positivo b, a può essere espresso nella seguente forma:

$$a = kb + r \equiv r (\text{mod } b)$$
$$a \bmod b = r$$

con k e r interi. Pertanto, $(a \bmod b) = a - kb$ per qualche intero k. Ma poiché $d|b$, d divide anche kb. Si ha anche $d|a$. Pertanto, $d|(a \bmod b)$. Questo dimostra che d è un divisore comune di b e $(a \bmod b)$. Analogamente, se d è divisore comune di b e $(a \bmod b)$, allora $d|kb$ e pertanto $d|[kb + (a \bmod b)]$, che è equivalente a $d|a$. Quindi l'insieme dei divisori comuni di a e b è uguale all'insieme dei divisori comuni di b e $(a \bmod b)$. Pertanto il massimo comune divisore di una coppia è uguale al massimo comune divisore dell'altra coppia, dimostrando il teorema.

L'Equazione 4.4 può essere utilizzata iterativamente per determinare il massimo comune divisore.

$$\gcd(18, 12) = \gcd(12, 6) = \gcd(6, 0) = 6$$
$$\gcd(11, 10) = \gcd(10, 1) = \gcd(1, 0) = 1$$

L'algoritmo di Euclide utilizza iterativamente l'Equazione 4.4 per determinare il massimo comune divisore nel seguente modo. L'algoritmo suppone che $a > b > 0$. È accettabile limitare l'algoritmo agli interi positivi in quanto $\gcd(a, b) = \gcd(|a|, |b|)$.

EUCLIDE(a,b)

1. $A \leftarrow a$; $B \leftarrow b$
2. **If** $B = 0$ **return** $A = \gcd(a,b)$
3. $R = A \bmod B$
4. $A \leftarrow B$
5. $B \leftarrow R$
6. **goto** 2

L'algoritmo ha il seguente sviluppo:

$$A_1 = B_1 \times Q_1 + R_1$$
$$A_2 = B_2 \times Q_2 + R_2$$
$$A_3 = B_3 \times Q_3 + R_3$$
$$A_4 = B_4 \times Q_4 + R_4$$

Per trovare il valore gcd(1970, 1066)

$1970 = 1 \times 1066 + 904$	gcd(1066, 904)
$1066 = 1 \times 904 + 162$	gcd(904, 162)
$904 = 5 \times 162 + 94$	gcd(162, 94)
$162 = 1 \times 94 + 68$	gcd(94, 68)
$94 = 1 \times 68 + 26$	gcd(68, 26)
$68 = 2 \times 26 + 16$	gcd(26, 16)
$26 = 1 \times 16 + 10$	gcd(16, 10)
$16 = 1 \times 10 + 6$	gcd(10, 6)
$10 = 1 \times 6 + 4$	gcd(6, 4)
$6 = 1 \times 4 + 2$	gcd(4, 2)
$4 = 2 \times 2 + 0$	gcd(2, 0)

Pertanto gcd(1970, 1066) = 2.

Qualcuno potrebbe chiedersi se questa operazione termina sempre. Ovvero, come si può essere sicuri che a un certo punto B divide A? Perchè in caso contrario si avrebbe una sequenza infinita di interi positivi ognuno dei quali strettamente inferiore di quello precedente: ciò è chiaramente impossibile.

4.4 Campi finiti nella forma GF(*p*)

Nel Paragrafo 4.1 si è definito un campo come un insieme che soddisfa a tutti gli assiomi elencati nella Figura 4.1 e sono stati dati alcuni esempi di campi infiniti. I campi infiniti non sono particolarmente interessanti nel contesto della crittografia mentre al contrario i campi finiti giocano un ruolo fondamentale in molti algoritmi crittografici. Si può dimostrare che l'ordine (il numero di elementi) di un campo finito deve essere una potenza di un numero primo p^n, dove n è un intero positivo. Si parlerà in dettaglio dei numeri primi nel Capitolo 8. Qui basti dire che un numero primo è un intero i cui unici fattori interi positivi sono se stesso e 1. Ovvero i soli interi positivi divisori di p sono p e 1.

Un campo finito di ordine p^n viene generalmente scritto come GF(p^n), da "Galois field" (campo di Galois), il matematico che ha studiato per primo i campi finiti. Vi sono due casi particolarmente interessanti per la crittografia. Per $n = 1$, si ha il campo finito GF(p); questo campo finito ha una struttura differente rispetto a quella dei campi finiti con $n > 1$ e verrà studiato in questa parte del capitolo. Nel Paragrafo 4.6 si parlerà invece dei campi finiti della forma GF(2^n).

Campi finiti di ordine *p*

Per un determinato numero primo, p, il campo finito di ordine p, GF(p), è definito come l'insieme Z_p degli interi $\{0, 1, \ldots, p-1\}$ insieme alle operazioni aritmetiche modulo p.

Nel Paragrafo 4.2 si è detto che l'insieme Z_n degli interi $\{0, 1, \ldots, n-1\}$, più le operazioni aritmetiche in modulo n formano un anello commutativo (Tabella 4.2). Inoltre si è osservato che ogni intero di Z_n ha un inverso moltiplicativo se e solo se tale intero è primo relativo di n (vedere l'Equazione 4.3[4]). Se n è primo, allora tutti gli interi diversi da 0 di Z_n sono primi rispetto a n e pertanto esiste un inverso moltiplicativo per tutti gli interi diversi da 0 di Z_n. Si può quindi aggiungere la seguente proprietà a quelle elencate nella Tabella 4.2 per Z_p.

Inverso moltiplicativo (w^{-1})	Per ogni $w \in Z_p$, $w \neq 0$, esiste $z \in Z_p$ tale che $w \times z \equiv 1 \pmod p$.

Dato che w è primo relativo rispetto a p, se si moltiplicano tutti gli elementi di Z_p per w, i resti risultanti sono tutti gli elementi di Z_p permutati. Pertanto esattamente uno dei resti avrà il valore 1. Quindi esiste un intero di Z_p che, quando moltiplicato per w, fornisce il resto 1. Tale intero è l'inverso moltiplicativo di w, rappresentato con w^{-1}. Pertanto Z_p è un campo finito. Inoltre, l'Equazione 4.3 è coerente con l'esistenza di un inverso moltiplicativo e può essere riscritta senza la condizione:

$$\textbf{se } (a \times b) \equiv (a \times c) \pmod p \textbf{ allora } b \equiv c \pmod p \qquad (4.5)$$

Moltiplicando entrambi i lati dell'Equazione 4.5 per l'inverso moltiplicativo di a si avrà:

[4] Come indicato nella discussione relativa all'Equazione 4.3, due interi sono primi relativi se il loro unico fattore comune positivo intero è 1.

$$((a^{-1})\times a\times b) \equiv ((a^{-1})\times a\times c) (\text{mod } p)$$
$$b \equiv c (\text{mod } p)$$

Il campo finito più semplice è GF(2). Le sue operazioni aritmetiche possono essere riepilogate con facilità nel seguente modo:

+	0	1
0	0	1
1	1	0

Somma

×	0	1
0	0	0
1	0	1

Moltiplicazione

w	$-w$	w^{-1}
0	0	–
1	1	1

Inversione

In questo caso, la somma è equivalente a un'operazione di OR esclusivo (XOR) e la moltiplicazione è equivalente a un'operazione di AND logico.

La Tabella 4.3 rappresenta GF(7). Si tratta di un campo di ordine 7 che utilizza l'aritmetica modulare con modulo 7; come si può notare, soddisfa tutte le proprietà di un campo (Figura 4.1). Si confronti questa tabella con la Tabella 4.1: in quest'ultimo caso l'insieme Z_8, costruito con l'aritmetica modulare con modulo 8, non è un campo. Più avanti in questo capitolo si dimostrerà come definire le operazioni di somma e moltiplicazione su Z_8 in modo tale che formi un campo finito.

Tabella 4.3 Aritmetica in GF(7).

A. Somma modulo 7.

+	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

B. Moltiplicazione modulo 7.

×	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

C. Inversi additivi e moltiplicativi modulo 7.

w	$-w$	w^{-1}
0	0	–
1	6	1
2	5	4
3	4	5
4	3	2
5	2	3
6	1	6

Ricerca dell'inverso moltiplicativo in GF(*p*).

È facile trovare l'inverso moltiplicativo di un elemento di GF(p) per valori piccoli di p. È sufficiente costruire una tabella di moltiplicazione, come quella rappresentata nella Tabella 4.3B e il risultato desiderato può essere letto direttamente. Tuttavia, per valori grandi di p, questo approccio è poco pratico.

Se gcd(m, b) = 1, allora b ha un inverso moltiplicativo modulo m. Ovvero per l'intero positivo $b < m$, esiste un $b^{-1} < m$ tale che $bb^{-1} = 1 \bmod m$. L'algoritmo di Euclide può essere esteso in modo che, oltre a trovare gcd(m, b), quando questo vale 1, l'algoritmo restituisca l'inverso moltiplicativo di b.

ALGORITMO DI EUCLIDE ESTESO (m,b)

1. (A1, A2, A3) ← (1,0,m); (B1,B2,B3) ← (0,1,b)
2. **if** B3 = 0 return A3 = gcd(m,b); nessun inverso
3. **if** B3 = 1 return B3 = gcd(m,b); B2 = $b^{-1} \bmod m$
4. $Q = \left\lfloor \frac{A3}{B3} \right\rfloor$
5. (T1, T2, T3) ← (A1 – QB1, A2 – QB2, A3 – QB3)
6. (A1, A2, A3) ← (B1, B2, B3)
7. (B1, B2, B3) ← (T1, T2, T3)
8. **goto** 2

In qualsiasi fase dell'elaborazione, valgono le seguenti relazioni:

$$m\text{T1} + b\text{T2} = \text{T3} \qquad m\text{A1} + b\text{A2} = \text{A3} \qquad m\text{B1} + b\text{B2} = \text{B3}$$

Per verificare che questo algoritmo restituisce correttamente il valore gcd(m, b), si noti che assumendo che A e B nell'algoritmo di Euclide siano uguali a A3 e B3 nell'algoritmo di Euclide esteso, il trattamento delle due variabili è identico. A ciascuna iterazione dell'algoritmo di Euclide, A è uguale al valore precedente di B e B è uguale al valore precedente di A mod B. Analogamente, in ciascun passo dell'algoritmo di Euclide, A3 è uguale al valore precedente di B3 e B3 è uguale al valore precedente di A3 meno il quoziente intero di A3 moltiplicato per B3. Quest'ultimo valore è semplicemente il resto di A3 diviso per B3, ovvero A3 mod B3.

Si noti inoltre che se gcd(m, b) = 1, allora al passo finale si ha B3 = 0 e A3 = 1. Pertanto, al passo precedente, B3 = 1. Ma se B3 = 1, valgono le seguenti uguaglianze:

$$m\text{B1} + b\text{B2} = \text{B3}$$
$$m\text{B1} + b\text{B2} = 1$$
$$b\text{B2} = 1 - m\text{B1}$$
$$b\text{B2} \equiv 1 (\text{mod } m)$$

Quindi B2 è l'inverso moltiplicativo di b modulo m.

La Tabella 4.4 è un esempio di esecuzione dell'algoritmo.
Mostra che gcd(1759, 550) = 1 e che l'inverso moltiplicativo di 550 è 355, ovvero 550 × 355 ≡ 1 (mod 1759).

Per una dimostrazione più dettagliata di questo algoritmo, vedere [KNUT97].

Riepilogo

In questo paragrafo si è descritto come costruire un campo finito di ordine p, con p primo. In particolare, si è definito GF(p) con le seguenti proprietà:

1. GF(p) contiene p elementi.

2. Nell'insieme sono definite le operazioni binarie + e ×. È possibile utilizzare le operazioni di somma, sottrazione, moltiplicazione e divisione senza lasciare l'insieme. Ogni elemento non nullo dell'insieme ha un inverso moltiplicativo.

Si è dimostrato che gli elementi di GF(p) sono gli interi {0, 1, 2,..., p–1} e che le operazioni aritmetiche sono la somma e la moltiplicazione mod p.

4.5 Aritmetica polinomiale

Prima di proseguire nella discussione sui campi finiti, è necessario introdurre l'argomento dell'aritmetica polinomiale. Si è particolarmente interessati ai polinomi in un'unica variabile, x; si possono distinguere tre classi di aritmetica polinomiale.

Tabella 4.4 Ricerca dell'inverso moltiplicativo di 550 in GF(1759).

Q	A1	A2	A3	B1	B2	B3
-	1	0	1759	0	1	550
3	0	1	550	1	-3	109
5	1	-3	109	-5	16	5
21	-5	16	5	106	-339	4
1	106	-339	4	-111	355	1

- Aritmetica polinomiale ordinaria, che utilizza le regole di base dell'algebra.
- Aritmetica polinomiale nella quale l'aritmetica sui coefficienti viene eseguita in modulo p; i coefficienti sono dunque in GF(p).
- Aritmetica polinomiale nella quale i coefficienti sono in GF(p) e i polinomi sono definiti in modulo a un polinomio $m(x)$ la cui potenza più elevata è un intero n.

Questa parte del capitolo esamina le prime due classi mentre la prossima parte tratta l'ultima classe.

Aritmetica polinomiale ordinaria

Un polinomio di grado n (intero $n \geq 0$) è un'espressione nella forma:

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 = \sum_{i=0}^{n} a_i x^i$$

dove i coefficienti a_i sono elementi di un determinato insieme di numeri S chiamato insieme dei coefficienti e $a_n \neq 0$. Si dice che tali polinomi sono definiti sull'insieme dei coefficienti S. Un polinomio di grado 0 è detto **polinomio costante** ed è semplicemente un elemento dell'insieme dei coefficienti. Un polinomio di grado n è detto **polinomio monico** se $a_n = 1$.

Nel contesto dell'algebra astratta, normalmente non si è interessati alla valutazione di un polinomio per un determinato valore di x (per esempio $f(7)$). Per evidenziare meglio questo fatto, la variabile x viene talvolta chiamata **indeterminata**.

L'aritmetica polinomiale include le operazioni di somma, sottrazione e moltiplicazione. Queste operazioni sono definite in modo naturale come se la variabile x fosse un elemento di S. La divisione è definita in modo analogo ma richiede che S sia un campo. Fra gli esempi di campi vi sono i numeri reali, i numeri razionali e Z_p per p primo. Si noti che l'insieme di tutti gli interi non è un campo e non supporta la divisione polinomiale.

La somma e la sottrazione vengono eseguite sommando o sottraendo i coefficienti corrispondenti. Pertanto se:

$$f(x) = \sum_{i=0}^{n} a_i x^i; \qquad g(x) = \sum_{i=0}^{m} b_i x^i; \qquad n \geq m$$

la somma è definita come:

$$f(x) + g(x) = \sum_{i=0}^{m} (a_i + b_i) x^i + \sum_{i=m+1}^{n} a_i x^i$$

e la moltiplicazione è definita come:

$$f(x) \times g(x) = \sum_{i=0}^{n+m} c_i x^i$$

dove:

$$c_k = a_0 b_k + a_1 b_{k-1} + \ldots + a_{k-1} b_1 + a_k b_0$$

Nell'ultima formula, a_i vale 0 per $i > n$ e b_i vale 0 per $i > m$. Si noti che il grado del prodotto è uguale alla somma dei gradi dei due polinomi.

Per esempio, si supponga che $f(x) = x^3 + x^2 + 2$ e $g(x) = x^2 - x + 1$. Allora:

$$f(x) + g(x) = x^3 + 2x^2 - x + 3$$
$$f(x) - g(x) = x^3 + x + 1$$
$$f(x) \times g(x) = x^5 + 3x^2 - 2x + 2$$

Le Figure da 4.3A a 4.3C riportano i calcoli manuali. La divisione verrà affrontata successivamente.

Aritmetica polinomiale con coefficienti in Z_p

Si considerino ora i polinomi i cui coefficienti siano elementi di un campo F, chiamati polinomi sul campo F. È facile dimostrare che l'insieme di tali polinomi è un anello, chiamato **anello polinomiale**. In pratica, se si considera ciascun polinomio distinto come un elemento dell'insieme, allora l'insieme è un anello.[5]

[5] In realtà, l'insieme di polinomi i cui coefficienti sono elementi di un anello commutativo forma un anello polinomiale, sebbene questo non sia di alcun interesse nel contesto considerato.

Quando si applica l'aritmetica polinomiale ai polinomi di un campo, diviene possibile eseguire la divisione. Si noti che questo non significa che sia possibile una *divisione esatta*. Forse è opportuno chiarire questa distinzione. Il quoziente *a/b* fra due elementi *a* e *b* è anch'esso un elemento del campo. Al contrario, in un anello *R* che non sia un campo, in generale la divisione produrrà un quoziente e un resto, ovvero la divisione non sarà esatta.

Si consideri la divisione 5/3 in un insieme *S*. Se *S* è l'insieme dei numeri razionali, che è un campo, allora il risultato è semplicemente espresso come 5/3 ed è un elemento di *S*. Ora si supponga che *S* sia il campo Z_7. In questo caso, si calcola (utilizzando la Tabella 4.3C):

$$5/3 = (5 \times 3^{-1}) \bmod 7 = (5 \times 5) \bmod 7 = 4$$

che è una soluzione esatta. Infine si supponga che *S* sia l'insieme degli interi, che è un anello ma non un campo. In questo caso 5/3 produrrà il quoziente 1 e il resto 2:

$$5/3 = 1 + 2/3$$
$$5 = 1 \times 3 + 2$$

Pertanto la divisione non è esatta nell'insieme degli interi.

Analogamente, se si tenta di eseguire la divisione polinomiale in un insieme di coefficienti che non è un campo, si troverà che la divisione non è sempre definita.

Se l'insieme dei coefficienti è quello degli interi, allora $(5x^2)/(3x)$ non ha soluzione poiché richiederebbe un coefficiente con il valore 5/3 che non appartiene all'insieme dei coefficienti. Si supponga di eseguire la stessa divisione polinomiale su Z_7. Si avrà $(5x^2)/(3x) = 4x$ che è un polinomio valido in Z_7.

Tuttavia, come dimostreremo, anche se l'insieme dei coefficienti è un campo, la divisione polinomiale non è necessariamente esatta. In generale la divisione produrrà un quoziente e un resto:

$$\frac{f(x)}{g(x)} = q(x) + \frac{r(x)}{g(x)}$$
$$f(x) = q(x)g(x) + r(x) \qquad (4.6)$$

Se il grado di $f(x)$ è n e il grado di $g(x)$ è m, $(m \geq n)$, allora il grado del quoziente $q(x)$ è $m - n$ e il grado del resto è al più $m - 1$. Dato che il resto è consentito, si può dire che la divisione fra polinomi è possibile se l'insieme dei coefficienti è un campo.

In un'analogia con l'aritmetica degli interi, si può scrivere $f(x)$ mod $g(x)$ per indicare il resto $r(x)$ nell'Equazione 4.6. In pratica $r(x) = f(x)$ mod $g(x)$. Se non vi è alcun resto (ovvero se $r(x) = 0$), si dice che $g(x)$ **divide** $f(x)$, e si scrive: $g(x)|f(x)$; si può anche dire che $g(x)$ è un **fattore** di $f(x)$ o che $g(x)$ è un **divisore** di $f(x)$.

Nell'esempio precedente [$f(x) = x^3 + x^2 + 2$ e $g(x) = x^2 - x + 1$], $f(x)/g(x)$ produce un quoziente $q(x) = x + 2$ e un resto $r(x) = x$ come indicato nella Figura 4.3D. Ciò è facilmente dimostrabile notando che:

$$q(x)g(x) + r(x) = (x+2)(x^2 - x + 1) + x = (x^3 + x^2 - x + 2) + x = x^3 + x^2 + 2 = f(x)$$

Per gli argomenti trattati in questo volume, sono particolarmente interessanti i polinomi su GF(2). Si ricordi dal Paragrafo 4.4 che in GF(2) la somma è equivalente all'operazione XOR

$$\begin{array}{r} x^3 + x^3 \quad + 2 \\ +\ (x^2 - x + 1) \\ \hline x^3 + 2x^2 - x + 3 \end{array}$$

(A) Addizione

$$\begin{array}{r} x^3 + x^2 \quad + 2 \\ -\ (x^2 - x + 1) \\ \hline x^3 \quad + x + 1 \end{array}$$

(B) Sottrazione

$$\begin{array}{r} x^3 + x^2 \quad + 2 \\ +\ (x^2 - x + 1) \\ \hline x^3 + x^2 \quad + 2 \\ -x^4 - x^3 \quad - 2x \\ \hline x^5 + x^4 \quad + 2x^2 \\ \hline x^5 \quad + 3x^2 - 2x + 2 \end{array}$$

(C) Moltiplicazione

$$\begin{array}{r|l} & x + 2 \\ \hline x^2 - x + 1 & x^3 + x^2 \quad + 2 \\ & x^3 - x^2 + x \\ \hline & 2x^2 - x + 2 \\ & 2x^2 - 2x + 2 \\ \hline & x \end{array}$$

(D) Divisione

Figura 4.3 Esempi di aritmetica polinomiale.

e la moltiplicazione è equivalente all'operazione logica AND. Inoltre la somma e la sottrazione sono equivalenti modulo 2: $1 + 1 = 1 - 1 = 0$; $1 + 0 = 1 - 0 = 1$; $0 + 1 = 0 - 1 = 1$.

La Figura 4.4 mostra un esempio di aritmetica polinomiale su GF(2). Per $f(x) = (x^7 + x^5 + x^4 + x^3 + x + 1)$ e $g(x) = (x^3 + x + 1)$, la figura riporta $f(x) + g(x)$; $f(x) - g(x)$; $f(x) \times g(x)$; e $f(x)/g(x)$. Si noti che $g(x)|f(x)$.

Un polinomio $f(x)$ su un campo F è detto **irriducibile** se e solo se $f(x)$ non può essere espresso come prodotto di due polinomi entrambi su F ed entrambi di grado inferiore a f(x). Per analogia con gli interi, un polinomio irriducibile è detto anche **polinomio primo**. Il polinomio[6] $f(x) = x^4 + 1$ su GF(2) è riducibile poiché $x^4 + 1 = (x + 1)(x^3 + x^2 + x + 1)$.

Si consideri il polinomio $f(x) = x^3 + x + 1$. È chiaro che x non è un fattore di $f(x)$. Si può agevolmente dimostrare che anche $x + 1$ non è un fattore di $f(x)$:

$$\begin{array}{r|l} & x^2 + x \\ \hline x + 1 & x^3 \quad + x + 1 \\ & x^3 + x^2 \\ \hline & x^2 + x \\ & x^2 + x \\ \hline & 1 \end{array}$$

[6] Nella parte rimanente di questo capitolo, se non indicato diversamente, tutti gli esempi sono polinomi su GF(2).

$$\begin{array}{rrrrrrrr} x^7 & +x^5 & +x^4 & +x^3 & & +x & +1 \\ & & & +(x^3 & & +x & +1) \\ \hline x^7 & +x^5 & +x^4 & & & & \end{array}$$

(A) Addizione

$$\begin{array}{rrrrrrrr} x^7 & +x^5 & +x^4 & +x^3 & & +x & +1 \\ & & & -(x^3 & & +x & +1) \\ \hline x^7 & +x^5 & +x^4 & & & & \end{array}$$

(B) Sottrazione

$$\begin{array}{rrrrrrrrr} & & x^7 & +x^5 & +x^4 & +x^3 & & +x & +1 \\ & & & & & \times(x^3 & & +x & +1) \\ \hline & & x^7 & +x^5 & +x^4 & +x^3 & & +x & +1 \\ & x^8 & & +x^6 & +x^5 & +x^4 & +x^2 & +x & \\ x^{10} & & +x^8 & +x^7 & +x^6 & & +x^4 & +x^3 & \\ \hline x^{10} & & & & & & +x^4 & +x^2 & +1 \end{array}$$

(C) Moltiplicazione

$$\begin{array}{r|l} & x^4+1 \\ \hline x^3+x+1 & x^7 \quad +x^5+x^4+x^3 \quad +x+1 \\ & x^7 \quad +x^5+x^4 \\ \hline & \qquad\qquad\quad x^3 \quad +x+1 \\ & \qquad\qquad\quad x^3 \quad +x+1 \\ \hline & \end{array}$$

(D) Divisione

Figura 4.4 Esempi di aritmetica polinomiale su GF(2).

Pertanto $f(x)$ non ha fattori di grado 1. Ma è chiaro che se $f(x)$ è riducibile, deve avere un fattore di grado 2 e un fattore di grado 1. Pertanto $f(x)$ è irriducibile.

Ricerca del massimo comune divisore

Si può estendere l'analogia fra l'aritmetica polinomiale su un campo e l'aritmetica degli interi definendo il massimo comune divisore nel seguente modo. Il polinomio $c(x)$ è detto massimo comune divisore di $a(x)$ e $b(x)$ se si verificano le seguenti condizioni.

1. $c(x)$ divide sia $a(x)$ che $b(x)$.

2. Ogni divisore di $a(x)$ e $b(x)$ è divisore anche di $c(x)$.

Ecco una definizione equivalente: gcd[$a(x)$, $b(x)$] è il polinomio di massimo grado che divide sia $a(x)$ che $b(x)$.

Si può adattare l'algoritmo di Euclide per calcolare il massimo comune divisore di due polinomi. L'uguaglianza dell'Equazione 4.4 può essere riscritta nel seguente teorema:

$$\gcd[a(x), b(x)] = \gcd[b(x), a(x) \bmod b(x)] \tag{4.7}$$

L'algoritmo di Euclide per i polinomi può essere riformulato nel seguente modo. L'algoritmo presuppone che il grado di $a(x)$ sia maggiore del grado di $b(x)$ e calcola quindi gcd[$a(x)$, $b(x)$].

ALGORITMO DI EUCLIDE[$a(x), b(x)$]

1. $A(x) \leftarrow a(x); B(x) \leftarrow b(x)$
2. **if** $B(x) = 0$ **return** $A(x) = \gcd[a(x), b(x)]$
3. $R(x) = A(x) \bmod B(x)$
4. $A(x) \leftarrow B(x)$
5. $B(x) \leftarrow R(x)$
6. **goto** 2

Trovare gcd[$a(x)$, $b(x)$] per $a(x) = x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$ e $b(x) = x^4 + x^2 + x + 1$.

$A(x) = a(x); B(x) = b(x)$

$$\begin{array}{r|l} & x^2+x \\ \hline x^4+x^2+x+1 & x^6+x^5+x^4+x^3+x^2+x+1 \\ & x^6 \quad +x^4+x^3+x^2 \\ \hline & \qquad x^5 \qquad\qquad +x+1 \\ & \qquad x^5 \quad +x^3+x^2+x \\ \hline & \qquad\qquad\quad x^3+x^2 \qquad +1 \end{array}$$

$R(x) = A(x) \bmod B(x) = x^3 + x^2 + 1$

$A(x) = x^4 + x^2 + x + 1; B(x) = x^3 + x^2 + 1$

$$\begin{array}{r|l} & x+1 \\ \hline x^3+x^2+1 & x^4 \quad +x^2+x+1 \\ & x^4+x^3 \quad +x \\ \hline & \qquad x^3+x^2 \quad +1 \\ & \qquad x^3+x^2 \quad +1 \\ \hline & \end{array}$$

$R(x) = A(x) \bmod B(x) = 0$

$\gcd[a(x), b(x)] = A(x) = x^3 + x^2 + 1$

Riepilogo

Questo paragrafo si è aperto con un'introduzione all'aritmetica dei polinomi ordinari. In questa aritmetica la variabile non viene valutata, ovvero non viene associato alcun valore alla variabile del polinomio. Le operazioni aritmetiche (somma, sottrazione, moltiplicazione e divisione) sono invece eseguite sui polinomi utilizzando le regole ordinarie dell'algebra. La divisione polinomiale non è possibile a meno che i coefficienti siano elementi di un campo. Successivamente si è affrontata l'aritmetica polinomiale nella quale i coefficienti sono elementi di GF(p). In questo caso sono possibili l'addizione, la sottrazione, la moltiplicazione e la divisione polinomiali. Tuttavia, la divisione non è esatta, ovvero, in generale, può generare un quoziente e un resto.

Infine si è dimostrato che l'algoritmo di Euclide può essere esteso per trovare il massimo comun divisore di due polinomi i cui coefficienti sono elementi di un campo.

Tutto il materiale di questo paragrafo costituisce le fondamenta per il paragrafo successivo nel quale si utilizzano i polinomi per definire dei campi finiti di ordine p^n.

4.6 Campi finiti della forma GF(2^n)

In precedenza, in questo stesso capitolo, si è detto che l'ordine di un campo finito deve essere della forma p^n dove p è un numero primo e n è un intero positivo. Nel Paragrafo 4.4 si è visto il caso particolare dei campi finiti di ordine p. Si è trovato che, utilizzando l'aritmetica modulare in Z_p, sono soddisfatti tutti gli assiomi relativi ai campi (Figura 4.1). Nel caso di polinomi su p^n, con n > 1, le operazioni in modulo p^n non producono un campo. In questa parte del capitolo si vedrà quale struttura soddisfa gli assiomi per un campo in un insieme con p^n elementi e ci si concentrerà su GF(2^n).

Motivazione

Praticamente tutti gli algoritmi di crittografia, sia quelli simmetrici che quelli a chiave pubblica, prevedono operazioni aritmetiche sugli interi. Se una delle operazioni utilizzate nell'algoritmo è la divisione, risulta necessario lavorare in un'aritmetica definita su un campo. Per comodità e per motivi di efficienza dell'implementazione si dovrebbe anche lavorare con interi rappresentabili esattamente con un determinato numero di bit senza sprecare pattern binari. In pratica si vuole lavorare con interi compresi nell'intervallo da 0 a $2^n - 1$ rappresentabili con una parola di n bit.

Si supponga di voler definire un algoritmo di crittografia convenzionale che operi su dati 8 bit alla volta e che si voglia eseguire la divisione. Con 8 bit si possono rappresentare gli interi nell'intervallo compreso fra 0 e 255. Tuttavia 256 non è un numero primo e dunque se l'aritmetica viene eseguita in Z_{256} (aritmetica modulo 256), questo insieme di interi non rappresenta un campo. Il numero primo più vicino a 256 e minore di 256 è 251. Pertanto l'insieme Z_{251} che utilizza l'aritmetica modulo 251 è un campo. In questo caso, tuttavia, il mancato utilizzo degli interi compresi fra 251 e 255 produrrebbe un utilizzo inefficiente dello spazio di memoria.

Come evidenziato dall'esempio precedente, se devono essere utilizzate tutte le operazioni aritmetiche e si vuole rappresentare in n bit un intervallo completo di interi, l'aritmetica in modulo 2^n non funzionerà. In modo equivalente, l'insieme di interi modulo 2^n per $n > 1$ non è un campo. Inoltre, anche se l'algoritmo di crittografia utilizzasse solo la somma e la moltiplicazione, ma non la divisione, l'uso dell'insieme Z_{2^n} sarebbe criticabile come illustrato dal seguente esempio.

Si supponga di voler usare nell'algoritmo di crittografia blocchi di tre bit e di utilizzare solo le operazioni di somma e moltiplicazione. Allora l'aritmetica modulo 8 è ben definita, come indicato nella Tabella 4.1. Tuttavia si noti che nella tabella delle moltiplicazioni gli interi diversi da 0 non compaiono un numero uniforme di volte. Per esempio, il numero 3 compare quattro volte mentre il numero 4 compare dodici volte. D'altro lato, come si è detto, vi sono campi finiti della forma GF(2^n) e dunque vi è un determinato campo finito di ordine $2^3 = 8$. L'aritmetica per questo campo è rappresentata nella Tabella 4.5. In questo caso, il numero di occorrenze degli interi diversi da 0 è uniforme per la moltiplicazione. Per riepilogare,

Interi	1	2	3	4	5	6	7
Occorrenze in Z_8	4	8	4	12	4	8	4
Occorrenze in GF(2^3)	7	7	7	7	7	7	7

Senza considerare per il momento come sia stata creata la matrice della Tabella 4.5, si osservi che:

1. Le tabelle della somma e della moltiplicazione sono simmetriche rispetto alla diagonale principale, in accordo alla proprietà commutativa delle due operazioni. Questa caratteristica è comune anche alla Tabella 4.1, che utilizza l'aritmetica modulo otto.
2. Tutti gli elementi non nulli definiti dalla Tabella 4.5 hanno un inverso moltiplicativo, a differenza di quanto avviene nella Tabella 4.1.
3. Lo schema definito dalla Tabella 4.5 soddisfa tutti i requisiti di un campo finito. È quindi possibile indicare questo schema con GF(2^3).

Per comodità, si indica anche la codifica di ciascun elemento di GF(2^3) con i 3 bit corrispondenti.

Intuitivamente, sembrerebbe che un algoritmo che mappa gli interi in modo non uniforme su se stessi sia crittograficamente più debole rispetto a un algoritmo che fornisce una distribuzione uniforme. Pertanto i campi finiti nella forma GF(2^n) sono i più interessanti per gli algoritmi crittografici.

Riepilogando, si è alla ricerca di un insieme di 2^n elementi che, con la definizione delle operazioni di somma e moltiplicazione, costituisca un campo. È possibile associare univocamente un intero, di valore compreso fra 0 e $2^n - 1$, ad ogni elemento dell'insieme. Si esclude l'impiego dell'aritmetica modulare dato che, come si è visto, non produce un campo. Si dimostrerà invece che l'aritmetica polinomiale consente di costruire il campo desiderato.

Tabella 4.5 Aritmetica in GF(2^3).

A. Somma

+	000 0	001 1	010 2	011 3	100 4	101 5	110 6	111 7
000 0	0	1	2	3	4	5	6	7
001 1	1	0	3	2	5	4	7	6
010 2	2	3	0	1	6	7	4	5
011 3	3	2	1	0	7	6	5	4
100 4	4	5	6	7	0	1	2	3
101 5	5	4	7	6	1	0	3	2
110 6	6	7	4	5	2	3	0	1
111 7	7	6	5	4	3	2	1	0

B. Moltiplicazione

x	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	3	1	7	5
3	0	3	6	5	7	4	1	2
4	0	4	3	7	6	2	5	1
5	0	5	1	4	2	7	3	6
6	0	6	7	1	5	3	2	4
7	0	7	5	2	1	6	4	3

C. Inverso additivo e moltiplicativo

w	$-w$	w^{-1}
0	0	–
1	1	1
2	2	5
3	3	6
4	4	7
5	5	2
6	6	3
7	7	4

Aritmetica polinomiale modulare

Si consideri l'insieme S di tutti i polinomi di grado $n-1$ o inferiore nel campo Z_p. Pertanto ciascun polinomio avrà la forma:

$$f(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots + a_1x + a_0 = \sum_{i=0}^{n-1} a_i x^i$$

dove ciascun a_i assume un valore nell'insieme $\{0, 1, \ldots, p-1\}$. In S vi è un totale di p^n polinomi differenti.

Per $p = 3$ e $n = 2$, i $3^2 = 9$ polinomi dell'insieme sono:

0	x	$2x$
1	$x+1$	$2x+1$
2	$x+2$	$2x+2$

Per $p = 2$ e $n = 3$, i $2^3 = 8$ polinomi dell'insieme sono:

0	$x+1$	x^2+x
1	x^2	x^2+x+1
	x^2+1	

Con l'appropriata definizione delle operazioni aritmetiche, ognuno di questi insiemi S è un campo finito. La definizione è costituita dai seguenti elementi.

1. L'aritmetica segue le regole ordinarie dell'aritmetica dei polinomi utilizzando le regole base dell'algebra con i due raffinamenti seguenti.
2. L'aritmetica sui coefficienti viene eseguita in modulo p. Si usano quindi le regole dell'aritmetica per il campo finito Z_p.
3. Se la moltiplicazione produce un polinomio di grado maggiore di $n-1$, allora il polinomio viene ridotto modulo $m(x)$ dove $m(x)$ è un polinomio irriducibile di grado n.

In pratica, si divide per $m(x)$ e si tiene il resto. Per un polinomio $f(x)$, il resto è espresso come $r(x) = f(x) \bmod m(x)$.

La crittografia AES (Advanced Encryption Standard) usa l'aritmetica nel campo finito GF(2^8) con il polinomio irriducibile $m(x) = x^8 + x^4 + x^3 + x + 1$. Si considerino i due polinomi $f(x) = x^6 + x^4 + x^2 + x + 1$ e $g(x) = x^7 + x + 1$. Allora:

$$\begin{aligned} f(x) + g(x) &= x^6 + x^4 + x^2 + x + 1 + x^7 + x + 1 \\ &= x^7 + x^6 + x^4 + x^2 \end{aligned}$$

$$\begin{aligned} f(x) \times g(x) &= x^{13} + x^{11} + x^9 + x^8 + x^7 + \\ &\quad x^7 + x^5 + x^3 + x^2 + x + \\ &\quad x^6 + x^4 + x^2 + x + 1 \\ &= x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 \end{aligned}$$

$$\begin{array}{r|l} & x^5 + x^3 \\ \hline x^8 + x^4 + x^3 + x + 1 & x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 \\ & x^{13} \quad + x^9 + x^8 \quad + x^6 + x^5 \\ \hline & + x^{11} \quad + x^4 + x^3 \\ & + x^{11} \quad + x^7 + x^6 \quad + x^4 + x^3 \\ \hline & + x^7 + x^6 + 1 \end{array}$$

pertanto, $f(x) \times g(x) \bmod m(x) = x^7 + x^6 + 1$.

Come per la normale aritmetica modulare, nell'aritmetica polinomiale modulare si ha la nozione di insieme dei resti. L'insieme dei resti modulo $m(x)$, un polinomio di grado n, è costituito da p^n elementi. Ognuno di questi elementi è rappresentato da uno dei p^n polinomi di grado $m < n$.

La classe dei resti $[x+1]$ modulo $m(x)$ è costituita da tutti i polinomi $a(x)$ tali che $a(x) \equiv (x+1) \pmod{m(x)}$. Analogamente, la classe dei resti $[x+1]$ è costituita da tutti i polinoni $a(x)$ che soddisfano l'uguaglianza $a(x) \bmod m(x) = x+1$.

Si può dimostrare che l'insieme di tuti i polinomi modulo $m(x)$, dove $m(x)$ è un polinomio irriducibile di grado n, soddisfa gli assiomi indicati nella Figura 4.1 e pertanto forma un campo finito. Inoltre, tutti i campi finiti di un determinato ordine sono isomorfi; ovvero due qualsiasi campi finiti di un determinato ordine hanno la stessa struttura, sebbene la rappresentazione o le etichette degli elementi possano essere differenti.

Per costruire il campo finito GF(2^3) occorre scegliere un polinomio irriducibile di grado 3. Vi sono solo due polinomi di questo tipo: $(x^3 + x^2 + 1)$ e $(x^3 + x + 1)$. Utilizzando l'ultimo, la Tabella 4.6 mostra le tabelle di somma e moltiplicazione per GF(2^3). Si noti che

questo insieme di tabelle ha una struttura identica a quelle nella Tabella 4.5. Ciò significa che si è identificata la tecnica per costruire un campo di ordine 2^3.

Tabella 4.6 Aritmetica polinomiale modulo $(x^3 + x + 1)$.

+	000 0	001 1	010 x	011 $x+1$	100 x^2	101 x^2+1	110 x^2+x	111 x^2+x+1
000 0	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
001 1	1	0	$x+1$	x	x^2+1	x^2	x^2+x+1	x^2+x
010 x	x	$x+1$	0	1	x^2+x	x^2+x+1	x^2	x^2+1
011 $x+1$	$x+1$	x	1	0	x^2+x+1	x^2+x	x^2+1	x^2
100 x^2	x^2	x^2+1	x^2+x	x^2+x+1	0	1	x	$x+1$
101 x^2+1	x^2+1	x^2	x^2+x+1	x^2+x	1	0	$x+1$	x
110 x^2+x	x^2+x	x^2+x+1	x^2	x^2+1	x	$x+1$	0	1
111 x^2+x+1	x^2+x+1	x^2+x	x^2+1	x^2	$x+1$	x	1	0

(a) Somma

×	000 0	001 1	010 x	011 $x+1$	100 x^2	101 x^2+1	110 x^2+x	111 x^2+x+1
000 0	0	0	0	0	0	0	0	0
001 1	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
010 x	0	x	x^2	x^2+x	$x+1$	1	x^2+x+1	x^2+1
011 $x+1$	0	$x+1$	x^2+x	x^2+1	x^2+x+1	x^2	1	x
100 x^2	0	x^2	$x+1$	x^2+x+1	x^2+x	x	x^2+1	1
101 x^2+1	0	x^2+1	1	x^2	x	x^2+x+1	$x+1$	x^2+x
110 x^2+x	0	x^2+x	x^2+x+1	1	x^2+1	$x+1$	x	x^2
111 x^2+x+1	0	x^2+x+1	x^2+1	x	1	x^2+x	x^2	$x+1$

(b) Moltiplicazione

Ricerca dell'inverso moltiplicativo

Così come l'algoritmo di Euclide può essere esteso per trovare il massimo comun divisore di due polinomi, l'algoritmo esteso di Euclide può essere esteso per trovare l'inverso moltiplicativo di un polinomio. In particolare, l'algoritmo troverà l'inverso moltiplicativo di $b(x)$ modulo $m(x)$ se il grado di $b(x)$ è minore del grado di $m(x)$ e $\gcd[m(x), b(x)] = 1$. Se $m(x)$ è un polinomio irriducibile, allora non vi sarà alcun fattore tale che $\gcd[m(x), b(x)] = 1$, tranne se stesso e 1. L'algoritmo è il seguente.

ALGORITMO DI EUCLIDE ESTESO $[m(x), b(x)]$

1. $[A1(x), A2(x), A3(x)] \leftarrow [1, 0, m(x)]; [B1(x), B2(x), B3(x)] \leftarrow [0, 1, b(x)]$
2. **if** $B3(x) = 0$ **return** $A3(x) = \gcd[m(x), b(x)]$; nessun inverso
3. **if** $B3(x) = 1$ **return** $B3(x) = \gcd[m(x), b(x)]$; $B2(x) = b(x)^{-1} \bmod m(x)$
4. $Q(x) =$ quoziente di $A3(x)/B3(x)$
5. $[T1(x), T2(x), T3(x)] \leftarrow [A1(x) - Q(x)B1(x), A2(x) - Q(x)B2(x), A3(x) - Q(x)B3(x)]$
6. $[A1(x), A2(x), A3(x)] \leftarrow [B1(x), B2(x), B3(x)]$
7. $[B1(x), B2(x), B3(x)] \leftarrow [T1(x), T2(x), T3(x)]$
8. **goto** 2

La Tabella 4.7 mostra il calcolo dell'inverso moltiplicativo di $(x^7 + x + 1) \bmod (x^8 + x^4 + x^3 + x + 1)$. Il risultato è che $(x^7 + x + 1)^{-1} = (x^7)$. Ciò significa che $(x^7 + x + 1)(x^7) \equiv 1 \pmod{(x^8 + x^4 + x^3 + x + 1)}$.

Considerazioni computazionali

Un polinomio $f(x)$ in GF(2^n):

$$f(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots + a_1x + a_0 = \sum_{i=0}^{n-1} a_i x^i$$

può essere rappresentato univocamente tramite i suoi n coefficienti binari $(a_{n-1}a_{n-2} \ldots a_0)$. Pertanto ogni polinomio contenuto in GF(2^n) può essere rappresentato come un numero di n bit.

Le Tabelle 4.5 e 4.6 mostrano le tabelle di somma e moltiplicazione per GF(2^3) modulo $m(x) = (x^3 + x + 1)$. La Tabella 4.5 usa la rappresentazione binaria e la Tabella 4.6 la rappresentazione polinomiale.

Tabella 4.7 Algoritmo esteso di Euclide $((x^8 + x^4 + x^3 + x + 1), (x^7 + x + 1))$.

Inizializzazione	$A1(x) = 1; A2(x) = 0; A3(x) = x^8 + x^4 + x^3 + x + 1$
	$B1(x) = 0; B2(x) = 1; B3(x) = x^7 + x + 1$
Iterazione 1	$Q(x) = x$
	$A1(x) = 0; A2(x) = 1; A3(x) = x^7 + x + 1$
	$B1(x) = 1; B2(x) = x; B3(x) = x^4 + x^3 + x^2 + 1$
Iterazione 2	$Q(x) = x^3 + x^2 + 1$
	$A1(x) = 1; A2(x) = x; A3(x) = x^4 + x^3 + x^2 + 1$
	$B1(x) = x^3 + x^2 + 1; B2(x) = x^4 + x^3 + x + 1; B3(x) = x$
Iterazione 3	$Q(x) = x^3 + x^2 + x$
	$A1(x) = x^3 + x^2 + 1; A2(x) = x^4 + x^3 + x + 1; A3(x) = x$
	$B1(x) = x^6 + x^2 + x + 1; B2(x) = x^7; B3(x) = 1$
Iterazione 4	$B3(x) = \gcd[x^7 + x + 1), (x^8 + x^4 + x^3 + x + 1)] = 1$
	$B2(x) = (x^7 + x + 1)^{-1} \bmod (x^8 + x^4 + x^3 + x + 1) = x^7$

Somma

Si è visto che la somma di polinomi viene eseguita sommando i coefficienti corrispondenti e che, nel caso di polinomi su Z_2, la somma è semplicemente l'operazione di XOR. Dunque la somma di due polinomi in GF(2^n) corrisponde a una operazione di XOR bit-a-bit.

Si considerino i due polinomi in GF(2^8) tratti dall'esempio precedente: $f(x) = x^6 + x^4 + x^2 + x + 1$ e $g(x) = x^7 + x + 1$.

$$(x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) = x^7 + x^6 + x^4 + x^2 \quad \text{notazione polinomiale}$$
$$(01010111) \oplus (10000011) = (11010100) \quad \text{notazione binaria}$$
$$\{57\} \oplus \{83\} = \{D4\} \quad \text{notazione esadecimale}^7$$

Moltiplicazione

Non vi è una semplice operazione analoga alla XOR in grado di eseguire la moltiplicazione in GF(2^n). Tuttavia è disponibile una tecnica relativamente semplice e facile da implementare. Si parlerà di questa tecnica con riferimento a GF(2^8) utilizzando $m(x) = x^8 + x^4 + x^3 + x + 1$, che è il campo finito utilizzato in AES. La tecnica è facilmente generalizzabile in GF(2^n).

La tecnica si basa sull'osservazione che:

$$x^8 \bmod m(x) = [m(x) - x^8] = (x^4 + x^3 + x + 1) \tag{4.8}$$

È facile dimostrare che l'Equazione 4.8 è vera e, in generale, in GF(2^n), con un polinomio $p(x)$ di grado n si ha $x^n \bmod p(x) = [p(x) - x^n]$.

Ora si consideri un polinomio in GF(2^8) che ha la forma $f(x) = b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$. Se si moltiplica per x si avrà:

$$x \times f(x) = (b_7x^8 + b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x) \bmod m(x) \tag{4.9}$$

Se $b_7 = 0$, allora il risultato è un polinomio di grado minore di 8 che è già in forma ridotta e non è più necessario alcun calcolo. Se $b_7 = 1$, allora la riduzione in modulo $m(x)$ viene ottenuta utilizzando l'Equazione 4.8:

$$x \times f(x) = (b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x) + (x^4 + x^3 + x + 1)$$

Ne consegue che la moltiplicazione per x (per esempio, 00000010) può essere implementata come uno scorrimento a sinistra di un bit seguito da un'operazione di XOR bit-a-bit condizionale con (00011011) che rappresenta $(x^4 + x^3 + x + 1)$. Per riepilogare:

$$x \times f(x) = \begin{cases} (b_6b_5b_4b_3b_2b_1b_0) & \text{se } b_7 = 0 \\ (b_6b_5b_4b_3b_2b_1b_0\,0) \oplus (00011011) & \text{se } b_7 = 1 \end{cases} \tag{4.10}$$

[5] Un riepilogo sui sistemi di numerazione (decimale, binario, esadecimale) si trova all'indirizzo WilliamStallings.com/StudentSupport.html. Ognuno dei due gruppi di 4 bit di un byte è rappresentato da un carattere esadecimale e i due caratteri vengono affiancati e racchiusi fra parentesi graffe.

La moltiplicazione per una potenza più elevata di x può essere ottenuta ripetendo l'applicazione dell'Equazione 4.10. Aggiungendo i risultati intermedi, si può ottenere la moltiplicazione per una qualsiasi costante di GF(2^8).

In uno degli esempi precedenti, si è dimostrato che per $f(x) = x^6 + x^4 + x^2 + x + 1$, $g(x) = x^7 + x + 1$ e $m(x) = x^8 + x^4 + x^2 + x + 1$, $f(x) \times g(x) \bmod m(x) = x^7 + x^6 + 1$. In aritmetica binaria occorre calcolare (01010111) × (10000011). Innanzitutto si determinano i risultati della moltiplicazione per le potenze di x:

(01010111)×(00000010)=(10101110)
(01010111)×(00000100)=(01011100)⊕(00011011)=(01000111)
(01010111)×(00001000)=(10001110)
(01010111)×(00010000)=(00011100)⊕(00011011)=(00000111)
(01010111)×(00100000)=(00001110)
(01010111)×(01000000)=(00011100)
(01010111)×(10000000)=(00111000)

Pertanto, (01010111) × (10000011) = (01010111) × [(00000001) ⊕ (00000010) ⊕ (10000000)] = (01010111) ⊕ (10101110) ⊕ (00111000) = (11000001) che è equivalente a $x^7 + x^6 + 1$.

L'impiego di un generatore

A volte risulta più conveniente utilizzare una tecnica equivalente per definire un campo finito della forma GF(2^n) utilizzando gli stessi polinomi irriducibili. Per cominciare è necessario introdurre due definizioni. Si dice **generatore** g di un campo finito F di ordine q (ovvero che contiene q elementi) un elemento le cui prime $q - 1$ potenze generano tutti gli elementi non nulli di F. Ovvero, gli elementi di F sono 0, g^0, g^1,..., g^{q-2}. Si consideri ora un campo F definito da un polinomio $f(x)$. La **radice** di un polinomio irriducibile è un elemento b contenuto in F tale che $f(b) = 0$. Infine, è possibile dimostrare che la radice g di un polinomio irriducibile è un generatore del campo finito definito su tale polinomio.

Si consideri il campo finito GF(2^3) definito sul polinomio irriducibile $x^3 + x + 1$ discusso in precedenza. Il generatore g deve soddisfare $f(g) = g^3 + g + 1 = 0$. Si tenga presente, come già visto, che non è necessario trovare una soluzione numerica a questa uguaglianza. Piuttosto, è necessario operare con l'aritmetica polinomiale nella quale l'aritmetica sui coefficienti viene calcolata modulo 2. Quindi, la soluzione dell'uguaglianza precedente è $g^3 = -g - 1 = g + 1$. Si dimostra ora che g genera infatti tutti i polinomi di grado inferiore a 3. Si ha:

$$g^4 = g(g^3) = g(g+1) = g^2 + g$$
$$g^5 = g(g^4) = g(g^2 + g) = g^3 + g^2 = g^2 + g + 1$$
$$g^6 = g(g^5) = g(g^2 + g + 1) = g^3 + g^2 + g = g^2 + g + g + 1 = g^2 + 1$$
$$g^7 = g(g^6) = g(g^2 + 1) = g^3 + g = g + g + 1 = 1 = g^0$$

Si può vedere che le potenze di g generano tutti i polinomi non nulli in GF(2^3). Dovrebbe essere chiaro anche che $g^k = g^{k \bmod 7}$ per qualsiasi intero k. La Tabella 4.8 illustra le rappresentazioni per potenze, polinomiale e binaria.

La rappresentazione per potenze facilita la moltiplicazione. Per eseguire un prodotto con questa notazione, si sommano gli esponenti modulo 7. Per esempio $g^4 \times g^6 = g^{(10 \bmod 7)} = g^3 = g + 1$. Il medesimo risultato si ottiene utilizzando l'aritmetica polinomiale nel modo seguente. Si ha $g^4 \times g^2 + g$ e $g^6 = g^2 + 1$. Quindi $(g^2 + g) \times (g^2 + 1) = g^4 + g^3 + g^2 + 1$. Si deve poi determinare $(g^4 + g^3 + g^2 + 1) \bmod (g^3 + g + 1)$ con la divisione:

$$
\begin{array}{r|l}
 & g + 1 \\
\hline
g^3 + g^2 + 1 & g^4 + g^3 + g^2 + g \\
 & g^4 + \quad g^2 + g \\
\hline
 & g^3 \\
 & g^3 + \quad g + 1 \\
\hline
 & g + 1
\end{array}
$$

Si ottiene il risultato $g + 1$, che coincide con il risultato ottenuto utilizzando la rappresentazione con le potenze.

La Tabella 4.9 riporta le tabelle di addizione e moltiplicazione per GF(2^3) utilizzando la rappresentazione delle potenze. Si noti che questo produce risultati identici alla rappresentazione polinomiale (Tabella 4.6) con alcune righe e colonne scambiate.

Tabella 4.8 Il generatore di GF(2^3) utilizzando il polinomio $x^3 + x + 1$.

Rappresentazione con potenze	Rappresentazione polinomiale	Rappresentazione binaria	Rappresentazione decimale (o esadecimale)
0	0	000	0
g^0 (=g^7)	1	001	1
g^1	g	010	2
g^2	g^2	100	4
g^3	$g+1$	011	5
g^4	$g^2 + g$	110	6
g^5	$g^2 + g + 1$	111	7
g^6	$g^2 + 1$	101	5

Tabella 4.9 Aritmetica GF(2^3) utilizzando il generatore del polinomio $x^3 + x + 1$.

+	000 (0)	001 (1)	010 (g)	100 (g^2)	011 (g^3)	110 (g^4)	111 (g^5)	101 (g^6)
000 (0)	0	1	g	g^2	$g+1$	g^2+g	g^2+g+1	g^2+1
001 (1)	1	0	$g+1$	g^2+1	g	g^2+g+1	g^2+g	g^2
010 (g)	g	$g+1$	0	g^2+g 1		g^2	g^2+1	g^2+g+1
100 (g^2)	g^2	g^2+1	g^2+g	0	g^2+g+1	g	$g+1$	1
011 (g^3)	$g+1$	g	1	g^2+g+10		g^2+1	g^2	g^2+g
110 (g^4)	g^2+g	g^2+g+1	g^2	g	g^2+1	0	1	$g+1$
111 (g^5)	g^2+g+1	g^2+g	g^2+1	$g+1$	g^2	1	0	g
101 (g^6)	g^2+1	g^2	g^2+g+1	1	g^2+g	$g+1$	g	0

(a) Addizione

×	000 (0)	001 (1)	010 (g)	100 (g^2)	011 (g^3)	110 (g^4)	111 (g^5)	101 (g^6)
000 (0)	0	0	0	0	0	0	0	0
001 (1)	0	1	g	g^2	$g+1$	g^2+g	g^2+g+1	g^2+1
010 (g)	0	g	g^2	$g+1$	g^2+g	g^2+g+1	g^2+1	1
100 (g^2)	0	g^2	$g+1$	g^2+g	g^2+g+1	g^2+1	1	g
011 (g^3)	0	$g+1$	g^2+g	g^2+g+1	g^2+1	1	G	g^2
110 (g^4)	0	g^2+g	g^2+g+1	g^2+1	1	g	g^2	$g+1$
111 (g^5)	0	g^2+g+1	g^2+1	1	g	g^2	$g+1$	g^2+g
101 (g^6)	0	g^2+1	1	g	g^2	$g+1$	g^2+g	g^2+g+1

(b) Multiplicazione

In generale, per il campo GF(2^n) con polinomio irriducibile $f(x)$, determinare $g^n = f(g) - g^n$. Quindi calcolare tutte le potenze di g da g^{n+1} a g^{2^n-2}. Gli elementi del campo corrispondono alle potenze di g da g^0 fino a g^{2^n-2}, oltre al valore nullo. Per la moltiplicazione di due elementi nel campo, utilizzare l'uguaglianza $g^k = g^{k \bmod (2^n-1)}$ valida per qualunque intero k.

Riepilogo

In questo paragrafo si è illustrato come costruire un campo finito di ordine 2^n. In particolare si è definito GF(2^n) con le seguenti proprietà:

1. GF(2^n) consiste di 2^n elementi.
2. Nell'insieme sono definite le operazioni binarie somma e moltiplicazione. Le operazioni di addizione, sottrazione, moltiplicazione e divisione possono essere effettuate senza lasciare l'insieme. Ogni elemento dell'insieme, a esclusione dell'elemento nullo, ha un inverso moltiplicativo.

Si è visto che gli elementi di GF(2^n) possono essere definiti come l'insieme di tutti i polinomi di grado $n - 1$ o inferiore con coefficienti binari. Ogni polinomio di questo tipo può essere rappresentato con un unico valore di n bit. L'aritmetica viene definita come aritmetica modulo un polinomio irriducibile di grado n. Si è anche visto che una definizione equivalente di un campo finito GF(2^n) utilizza un generatore e che l'aritmetica viene definita utilizzando le potenze del generatore.

4.7 Letture e siti Web consigliati

[HERS75], ancora in stampa, è una trattazione classica dell'algebra astratta; è leggibile e, nel contempo, rigorosa; [DESK92] è un'altra buona risorsa. [KNUT98] fornisce una buona copertura dell'argomento dell'aritmetica polinomiale.

Uno dei migliori trattati sugli argomenti descritti in questo capitolo è [BERL84], ancora in stampa. [GARR01] offre anch'esso una ampia descrizione. Un'ampia e rigorosa trattazione dei campi finiti si trova in [LIDL94]. [HORO71] è un'ottima rassegna dei temi trattati in questo capitolo.

BERL84 Berlekamp, E. *Algebraic Coding Theory*. Laguna Hills, CA: Aegean Park Press, 1984.

DESK92 Deskins, W. *Abstract Algebra*. New York: Dover, 1992.

GARR01 Garrett, P. Making, *Breaking Codes: An Introduction to Cryptology*. Upper Saddle River, NJ: Prentice Hall, 2001.

HERS75 Herstein, I. *Topics in Algebra*. New York: Wiley, 1975.

HORO71 E. Horowitz. "Modular Arithmetic and Finite Field Theory: A Tutorial". *Proceedings of the Second ACM Symposium on Symbolic and Algebraic Manipulation*, Marzo 1971.

KNUT98 Knuth, D. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*.

LIDL94 R. Lidl e H. Niederreiter. *Introduction to Finite Fields and Their Applications*. Cambridge: Cambridge University Press, 1994.

Sito Web consigliato

- **PascGalois Project:** contiene una grande quantità di esempi e progetti che aiutano gli studenti a comprendere in modo visuale i concetti chiave dell'algebra astratta.

4.8 Termini chiave, domande di ripasso e problemi

Termini chiave

Algoritmo di Euclide
Anello
Anello commutativo
Anello finito
Anello infinito
Anello polinomiale
Aritmetica modulare
Aritmetica polinomiale
Aritmetica polinomiale modulare
Campo
Campo finito
Campo infinito
Divisore
Dominio integrale
Elemento identità
Elemento inverso
Generatore
Gruppo
Gruppo abeliano
Gruppo ciclico
Gruppo finito
Gruppo infinito
Massimo comune divisore
Numero primo
Operatore modulo
Ordine
Polinomio
Polinomio irriducibile
Polinomio monico
Polinomio primo
Primo relativo
Proprietà associativa
Proprietà commutativa
Resto

Domande di ripasso

4.1 Definire brevemente un gruppo.
4.2 Definire brevemente un anello
4.3 Definire brevemente un campo.
4.4 Che cosa significa dire che b è divisore di a?
4.5 Qual è la differenza fra aritmetica modulare e aritmetica ordinaria?
4.6 Elencare tre classi di aritmetica polinomiale.

Problemi

4.1 Per il gruppo S_n di tutte le permutazioni di n simboli distinti.
   A. Qual è il numero di elementi di S_n?
   B. Dimostrare che S_n non è abeliano per $n > 2$.
4.2 L'insieme delle classi dei resti modulo 3 formano un gruppo...
   A. ... rispetto alla somma?
   B. ... rispetto alla moltiplicazione?
4.3 Si consideri l'insieme $S = \{a, b\}$ con la somma e la moltiplicazione definite nelle seguenti tabelle:

+	a	b
a	a	b
b	b	a

×	a	b
a	a	a
b	a	b

S è un anello? Giustificare la risposta.

4.4 Riformulare l'Equazione 4.1 eliminando la restrizione che a sia un intero non negativo. Ovvero, si consenta ad a di assumere qualsiasi valore intero.
4.5 Disegnare una figura simile alla Figura 4.2 per $a < 0$.
4.6 Trovare i numeri interi x tali che:
   A $5x \equiv 4 (\mathrm{mod}\, 3)$
   B $7x \equiv 6 (\mathrm{mod}\, 5)$
   C $9x \equiv 8 (\mathrm{mod}\, 8)$

4.7 In questo testo si è ipotizzato che il modulo sia un intero positivo. Tuttavia, la definizione dell'espressione a mod n è perfettamente legittima anche nel caso in cui n sia negativo. Calcolare:
A 5 mod 3
B 5 mod –3
C –5 mod 3
D –5 mod –3

4.8 Il modulo con il valore 0 non avrebbe significato nella definizione, ma viene stabilito per convenzione come segue: a mod 0 = a. Utilizzando questa convenzione, cosa significa l'espressione seguente: $a \equiv b (\text{mod } 0)$?

4.9. Nel Paragrafo 4.2 si è definita la relazione di congruenza dicendo che due interi a e b sono congruenti modulo n se (a mod n) = (b mod n). Quindi si è dimostrato che $a \equiv b \bmod n$ se $n|(a - b)$. Alcuni testi sulla teoria dei numeri usano quest'ultima relazione come definizione di congruenza: due interi a e b si dicono congruenti modulo n se $n|(a - b)$. Utilizzando quest'ultima definizione come punto di partenza, dimostrare che se (a mod n) = (b mod n), allora n divide $(a - b)$.

4.10 Qual è il più piccolo intero positivo che ha esattamente k divisori, per $1 \le k \le 6$?

4.11 Dimostrare le seguenti affermazioni.
A. $a \equiv b \pmod n$ implica $b \equiv a \pmod n$
B. $a \equiv b \pmod n$ e $b \equiv c \pmod n$ implica $a \equiv c \pmod n$

4.12 Dimostrare le seguenti affermazioni.
A. $[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$
B. $[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$

4.13 Trovare l'inverso moltiplicativo di ciascun elemento diverso da 0 contenuto in Z_5.

4.14 Dimostrare che un intero N è congruente modulo 9 con la somma delle sue cifre decimali. Per esempio $475 \equiv 4 + 7 + 5 \equiv 16 \equiv 1 + 6 \equiv 7 \pmod 9$. Questa è la base della procedura di "prova del nove" dei calcoli aritmetici.

4.15 A. Determinare il valore di gcd(24140, 16762).
B. Determinare il valore di gcd(4655, 12075).

4.16 Lo scopo di questo problema è quello di definire un limite superiore al numero di iterazioni dell'algoritmo di Euclide.
A. Supporre che $m = qn + r$ con $q > 0$ e $0 \le r < n$. Dimostrare che $m/2 > r$.
B. Se A_i è il valore di A nell'algoritmo di Euclide dopo la i-esima iterazione, dimostrare che:

$$A_{i+2} < \frac{A_i}{2}$$

C. Dimostrare che se m, n e N sono interi con $1 \le m, n \le 2^N$, allora l'algoritmo di Euclide richiede al più $2N$ passi per trovare gcd(m, n).

4.17 L'algoritmo di Euclide è noto da oltre duemila anni ma è sempre stato uno dei preferiti dai teorici dei numeri. Dopo tutti questi anni, vi è un potenziale concorrente inventato da J. Stein nel 1961. Ecco l'algoritmo. Determinare gcd(A, B) con $A, B \ge 1$.

FASE 1 Impostare $A_1 = A, B_1 = B, C_1 = 1$

FASE n (1) If $A_n = B_n$ stop. gcd$(A,B) = A_nC_n$
(2) If A_n e B_n sono entrambi pari, impostare $A_{n+1} = A_n/2, B_{n+1} = B_n/2, C_{n+1} = 2C_n$
(3) If A_n è pari e B_n è dispari, impostare $A_{n+1} = A_n/2, B_{n+1} = B_n, C_{n+1} = C_n$
(4) If A_n è dispari e B_n è pari, impostare $A_{n+1} = A_n, B_{n+1} = B_n/2, C_{n+1} = C_n$
(5) If A_n e B_n sono dispari, impostare $A_{n+1} = [A_n - B_n], B_{n+1} = \min(B_n, A_n), C_{n+1} = C_n$
Continuare al passo $n + 1$

A. Per provare a usare i due algoritmi, si provi a calcolare gcd(2152, 764) utilizzando sia l'algoritmo di Euclide che quello di Stein.
B. Qual è il vantaggio dell'algoritmo di Stein rispetto all'algoritmo di Euclide?

4.18 A. Dimostrare che se l'algoritmo di Stein non si ferma entro il passo n-esimo, allora:

$$C_{n+1} \times \gcd(A_{n+1}, B_{n+1}) = C_n \times (A_n, B_n)$$

B. Dimostrare che se l'algoritmo non si ferma entro il passo $n - 1$, allora:

$$A_{n+2}B_{n+2} \le \frac{A_nB_n}{2}$$

C. Dimostrare che se $1 \le A, B \le 2^N$, allora l'algoritmo di Stein richiede al massimo $4N$ passi per trovare gcd(m, n). Pertanto l'algoritmo di Stein opera più o meno nello stesso numero di passi dell'algoritmo di Euclide.
D. Dimostrare che l'algoritmo di Stein restituisce gcd(A, B).

4.19 Utilizzando l'algoritmo di Euclide esteso, trovare l'inverso moltiplicativo di:
A. 1234 mod 4321
B. 24140 mod 40902
C. 550 mod 1769

4.20 Sviluppare un insieme di tabelle simili alla Tabella 4.3 per GF(5).

4.21 Dimostrare che l'insieme dei polinomi i cui coefficienti formano un campo è un anello.

4.22 Dimostrare se ciascuna delle seguenti affermazioni è vera o falsa per i polinomi di un campo.
A. Il prodotto di polinomi monici è monico.
B. Il prodotto di polinomi di grado m e n ha grado $m + n$.
C. La somma di polinomi di grado m e n ha grado max[m, n].

4.23 Utilizzando l'aritmetica polinomiale con coefficienti in Z_{10}, svolgere i seguenti calcoli:
A. $(7x + 2) - (x^2 + 5)$
B. $(6x^2 + x + 3) \times (5x^2 + 2)$

4.24 Determinare quale dei seguenti polinomi è riducibile su GF(2):
A. $x^3 + 1$
B. $x^3 + x^2 + 1$
C. $x^4 + 1$ (fare attenzione)

4.25 Determinare il massimo comune divisore delle seguenti coppie di polinomi:
A. x^3+x+1 e x^2+x+1 su GF(2)
B. x^3-x+1 e x^2+1 su GF(3)
C. $x^5+x^4+x^3-x^2-x+1$ e x^3+x^2+x+1 su GF(3)
D. $x^5+88x^4+73x^3+83x^2+51x+67$ e $x^3+97x^2+40x+38$ su GF(101)

4.26 Sviluppare un insieme di tabelle simili alla Tabella 4.6 per GF(4), con $m(x)=x^2+x+1$.

4.27 Determinare l'inverso moltiplicativo di x^3+x+1 in GF(2^4) con $m(x)=x^4+x+1$.

4.28 Sviluppare un insieme di tabelle simili alla Tabella 4.8 per GF(2^4), con $m(x)=x^4+x+1$.

Problemi di programmazione

4.29 Scrivere una semplice calcolatrice a quattro funzioni su GF(2^4). Per il calcolo degli inversi moltiplicativi è possibile utilizzare una tabella di lookup.

4.30 Scrivere una semplice calcolatrice a quattro funzioni su GF(2^8). Si devono calcolare anche gli inversi moltiplicativi.

Capitolo 5

Lo standard AES (Advanced Encryption Standard)

Concetti essenziali

- **AES** è un algoritmo di cifratura a blocchi progettato per sostituire DES nelle applicazioni commerciali. Utilizza una dimensione di blocco di 128 bit e una chiave di lunghezza 128, 192 o 256 bit.
- AES non utilizza la struttura di Feistel. Ciascuna fase consiste in quattro funzioni separate: sostituzioni di byte, permutazioni, operazioni aritmetiche su un campo finito e XOR con una chiave.

Lo standard AES (Advanced Encryption Standard) è stato pubblicato dal NIST (National Institute of Standards and Technology) nel 2001. AES è una cifratura simmetrica a blocchi destinata a sostituire lo standard DES per un'ampia varietà di applicazioni. In questo capitolo si vedranno innanzitutto i criteri di valutazione impiegati dal NIST per selezionare un candidato per AES e poi si esaminerà il funzionamento di questa cifratura.

Paragonata alle cifrature a chiave pubblica come RSA, la struttura di AES (e della maggior parte delle cifrature simmetriche) è molto più complessa e non può essere spiegata così facilmente. Per questo motivo il lettore potrebbe iniziare con una versione semplificata di AES, descritta nell'Appendice 5.B. Questa versione consente di effettuare la cifratura e la decifratura manuale, permettendo così al lettore di comprendere il funzionamento dell'algoritmo nei suoi particolari. L'esperienza scolastica indica che lo studio di questa versione semplificata migliora la comprensione di AES[1].

[1] È tuttavia possibile ignorare l'Appendice 5.B alla prima lettura. Se si dovessero incontrare difficoltà nello studio di AES, si potrà ritornare alla sua versione semplificata.

5.1 Criteri di valutazione per lo standard AES

Le origini di AES

Nel Capitolo 3 si è detto che nel 1999 il NIST emise una nuova versione dello standard DES (FIPS PUB 46-3) che raccomandava di limitare l'utilizzo di DES ai soli sistemi preesistenti e di utilizzare invece il nuovo algoritmo, 3DES, di cui si parlerà nel Capitolo 6. 3DES ha due interessanti qualità che ne garantiranno l'ampio impiego nei prossimi anni. Innanzitutto, con la sua chiave di 168 bit risolve il problema della vulnerabilità di DES agli attacchi a forza bruta. In secondo luogo, l'algoritmo di crittografia di base di 3DES è lo stesso di DES. Questo algoritmo è stato attentamente esaminato più di qualsiasi altro algoritmo di crittografia e per un lungo periodo di tempo, ma non è stato trovato alcun attacco ad analisi crittografica in grado di sfruttare le caratteristiche intrinseche dell'algoritmo. Di conseguenza 3DES è ritenuto molto resistente all'analisi crittografica. Se l'unica considerazione fosse la sicurezza, 3DES sarebbe la scelta appropriata quale algoritmo di crittografia standard per i decenni a venire.

Il principale difetto di 3DES consiste nel fatto che l'algoritmo è relativamente lento nell'esecuzione software. L'algoritmo DES originario era stato progettato alla metà degli anni '70 per un'implementazione hardware e non si presta ad una implementazione software efficiente. 3DES, che ripete per tre volte l'algoritmo DES, è ovviamente ancora più lento. Un difetto secondario consiste nel fatto che sia 3DES che DES usano blocchi di 64 bit. Per motivi sia di efficienza che di sicurezza, è preferibile utilizzare blocchi di maggiori dimensioni. Dati questi difetti, 3DES non rappresenta il candidato ideale per un utilizzo a lungo termine. In sostituzione, il NIST emise nel 1997 una richiesta di proposte per un nuovo algoritmo chiamato AES (Advanced Encryption Standard) che presentasse doti di sicurezza uguali o migliori rispetto a 3DES e con un'efficienza decisamente superiore. Oltre a questi requisiti generali, il NIST specificò che AES fosse una cifratura simmetrica con blocchi di lunghezza di 128 bit e che supportasse chiavi di 128, 192 e 256 bit.

In una prima valutazione, vennero accettati 15 degli algoritmi proposti. Una seconda fase restrinse il campo a 5 algoritmi. Il NIST completò il processo di valutazione per pubblicare lo standard finale (FIPS PUB 197) nel novembre 2001. Il NIST selezionò l'algoritmo Rijndael per AES. I due ricercatori che hanno sviluppato e proposto Rijndael come algoritmo AES sono entrambi crittografi di origine belga: Dr. Joan Daemen e Dr. Vincent Rijmen. Alla fine, AES andrà a sostituire 3DES ma questa operazione richiederà vari anni. Anche il NIST ha confermato 3DES come uno degli algoritmi approvati per utilizzi governativi USA nel prossimo futuro.

Valutazione di AES

Vale la pena esaminare i criteri utilizzati dal NIST per valutare i potenziali candidati. Questi criteri riguardano i problemi di applicazione pratica delle moderne cifrature simmetriche a blocchi. In pratica si sono evoluti due insiemi di criteri. Quando il NIST ha emise la sua richiesta originaria di proposte di algoritmi nel 1997 [NIST97], la richiesta stabiliva che gli algoritmi candidati sarebbero stati confrontati sulla base dei fattori illustrati nella

Tabella 5.1 (in ordine di importanza relativa). I criteri erano suddivisi nelle tre categorie seguenti.

- **Sicurezza:** fa riferimento all'impegno necessario per analizzare crittograficamente un algoritmo. L'enfasi nella valutazione era sulle possibilità pratiche di un attacco. Poiché in AES la chiave ha dimensioni minime di 128 bit, gli attacchi a forza bruta con le tecnologie attuali e future erano considerati impraticabili. Per questo motivo si enfatizzava l'analisi crittografica rispetto agli attacchi a forza bruta.
- **Costo:** il NIST richiese che l'algoritmo AES potesse essere impiegato per un'ampia gamma di applicazioni. Di conseguenza l'algoritmo AES doveva avere un'elevata efficienza computazionale e doveva essere utilizzabile in applicazioni ad alta velocità, come per esempio i collegamenti a banda larga.
- **Caratteristiche dell'algoritmo e dell'implementazione:** questa categoria includeva vari aspetti fra cui la flessibilità, la possibilità di impiego in varie implementazioni hardware e software e la semplicità, per facilitare l'analisi della sicurezza.

Utilizzando questi criteri, i 21 algoritmi proposti vennero prima ridotti a 15 e poi a 5. Prima della valutazione finale, i criteri di valutazione (vedere [NECH00]) vennero aggiornati. Ecco i criteri utilizzati nella valutazione finale.

- **Sicurezza generale:** per valutare la sicurezza generale, il NIST si basò su un'analisi pubblica della sicurezza, condotta dalla comunità dei crittografi. Nel corso del processo di valutazione, durato tre anni, vari crittografi pubblicarono la propria analisi dei punti di forza e di debolezza dei vari candidati. Vi fu un'enfasi particolare sull'analisi della robustezza dei candidati agli attacchi noti, come per esempio le analisi crittografiche differenziale e lineare. Tuttavia, rispetto all'analisi di DES, il tempo e il numero di crittografi dedicati all'analisi dell'algoritmo Rijndael sono stati piuttosto limitati. Ora che è stata scelta un'unica cifratura AES, ci si può attendere un'analisi della sicurezza più ampia da parte della comunità dei crittografi.
- **Implementazioni software:** le principali richieste in questa categoria sono la velocità di esecuzione, le prestazioni in varie piattaforme e la variazione della velocità in base alle dimensioni della chiave.

Tabella 5.1 Criteri di valutazione del NIST relativi allo standard AES (12 settembre 1997).

Sicurezza
Effettiva sicurezza, da confrontare con gli altri algoritmi presentati (a parità di dimensioni della chiave e del blocco).
Casualità: il livello di indistinguibilità dell'output dell'algoritmo rispetto a una permutazione casuale del blocco di input.
Solidità delle basi matematiche per la sicurezza dell'algoritmo.
Altri fattori di sicurezza sollevati dal pubblico durante il processo di valutazione, fra cui eventuali attacchi che dimostrassero che l'effettiva sicurezza dell'algoritmo fosse inferiore a quella vantata dal proponente.

(segue)

Tabella 5.1 Criteri di valutazione del NIST relativi allo standard AES (12 settembre 1997). *(continua)*

Costo
Requisiti in termini di licenza: il NIST richiede che quando verrà adottato AES, gli algoritmi specificati siano disponibili globalmente, senza esclusioni e in modo gratuito, ovvero esenti da royalty.
Efficienza computazionale: la valutazione dell'efficienza computazionale sarà applicabile sia a implementazioni hardware che software. La Fase 1 di analisi da parte del NIST si concentrerà principalmente sulle implementazioni software e in particolare su una particolare combinazione di dimensioni della chiave e del blocco (128/128); si presterà un'attenzione più specifica alle implementazioni hardware e alle altre combinazioni chiave/blocco supportate nella Fase 2 dell'analisi. L'efficienza computazionale fa fondamentalmente riferimento alla velocità dell'algoritmo. Il NIST terrà in considerazione anche i commenti del pubblico sull'efficienza di ciascun algoritmo (in particolare per varie piattaforme e applicazioni).
Requisiti di memoria: il processo di valutazione valuterà anche la memoria necessaria per implementare l'algoritmo candidato (considerando implementazioni sia hardware che software). L'analisi di Fase 1 del NIST si concentrerà principalmente sulle implementazione software, mentre si darà più importanza alle implementazioni hardware nella Fase 2. I requisiti di memoria includono fattori quali il numero di porte per le implementazioni hardware nonché le dimensioni del codice e i requisiti di memoria RAM per le implementazioni software.
Caratteristiche dell'algoritmo e dell'implementazione
Flessibilità: fra i candidati, verranno preferiti, a parità di altre condizioni, gli algoritmi più flessibili che risponderanno alle esigenze di un maggior numero di utenti rispetto a quelli meno flessibili. Tuttavia certe funzionalità estreme (per esempio chiavi estremamente brevi) hanno scarsa applicabilità pratica; per questi casi non verrà attribuita alcuna preferenza. Ecco alcuni esempi di flessibilità. A. L'algoritmo può utilizzare chiavi e blocchi di varie dimensioni (per esempio blocchi di 64 bit), chiavi di dimensioni diverse rispetto a quelle specificate nel paragrafo "Requisiti minimi" (per esempio chiavi di dimensioni comprese fra 128 e 256 e che sia multipla di 32 bit e così via). B. L'algoritmo può essere implementato in modo sicuro ed efficiente in un'ampia varietà di piattaforme e applicazioni (per esempio microprocessori a 8 bit, reti ATM, canali, voce e dati via satellite, HDTV, B-ISDN e così via). C. L'algoritmo può essere implementato come cifratura di flusso, come generatore di MAC (Message Authentication Code), come generatore di numeri pseudocasuali, come algoritmo hash e così via.
Fattibilità hardware e software: un algoritmo candidato non dovrà essere restrittivo nel senso che può essere implementato esclusivamente in hardware. Se un algoritmo può essere implementato in modo efficiente in firmware, questo rappresenterà un vantaggio in termini di flessibilità.
Semplicità: un algoritmo candidato verrà giudicato in base alla semplicità di progettazione.

- **Ambienti con spazio limitato:** in alcune applicazioni, come per esempio le smart card, sono disponibili quantità relativamente ridotte di memoria RAM (Random Access Memory) e/o ROM (Read-Only Memory). È necessario quindi valutare la quantità di memoria richiesta per memorizzare il codice (solitamente in ROM), la rappresentazione di oggetti come le S-box (che possono essere memorizzate in ROM o in RAM, a seconda che venga utilizzata la pre-computazione o la rappresentazione booleana) e le sottochiavi (in RAM).
- **Implementazioni hardware:** come nel caso del software, le implementazioni hardware possono essere ottimizzate in termini di velocità o di dimensioni. Tuttavia, nel caso delle implementazioni hardware, più che in quelle software, le dimensioni si traducono direttamente in costi. In un computer di utilizzo generale, dotato di una grande quantità di memoria, raddoppiare le dimensioni di un programma di crittografia potrebbe costituire una differenza trascurabile mentre raddoppiare l'area utilizzata in un dispositivo hardware può più che raddoppiare il costo del dispositivo stesso.
- **Attacchi alle implementazioni:** il criterio di sicurezza generale, trattato al primo punto, riguarda gli attacchi ad analisi crittografica che sfruttano le proprietà matematiche degli algoritmi. Vi è un'altra classe di attacchi che utilizza le misurazioni fisiche condotte durante l'esecuzione dell'algoritmo per raccogliere informazioni su parametri come per esempio le chiavi. Tali attacchi sfruttano una combinazione di caratteristiche intrinseche dell'algoritmo e funzionalità dipendenti dall'implementazione. Fra gli esempi di questi attacchi vi sono gli attacchi a tempo e l'analisi della potenza. Gli attacchi a tempo sono stati descritti nel Capitolo 3. L'idea alla base dell'analisi dei consumi energetici [KOCH98, BIHA00] è l'osservazione che l'energia consumata da una smart card durante le operazioni di crittografia è correlata alle particolari istruzioni eseguite e ai dati elaborati. Per esempio, la moltiplicazione consuma più energia rispetto a una somma e la scrittura di valori "1" consuma più energia rispetto alla scrittura di valori "0".
- **Crittografia e decrittografia:** questo criterio riguarda vari problemi relativi a considerazioni sulla crittografia e la decrittografia. Se gli algoritmi di crittografia e decrittografia sono differenti, sarà necessario ulteriore spazio per la decrittografia. Inoltre vi possono essere differenze di temporizzazione fra la crittografia e la decrittografia indipendentemente dal fatto che utilizzino algoritmi identici o differenti.
- **Agilità della chiave:** l'agilità della chiave fa riferimento alla capacità di cambiare rapidamente le chiavi utilizzando la minima quantità possibile di risorse. Questo comprende sia il calcolo della sottochiave che la possibilità di commutare fra diverse configurazioni di sicurezza quando le chiavi siano già disponibili.
- **Altri elementi di versatilità e flessibilità:** [NECH00] indica due aree che rientrano in questa categoria. La flessibilità dei parametri include la facilità di supporto di chiavi e blocchi di altre dimensioni e la possibilità di aumentare il numero di fasi per rispondere ai nuovi attacchi eventualmente scoperti. La flessibilità dell'implementazione fa riferimento alla possibilità di ottimizzare gli elementi di cifratura per determinati ambienti.
- **Potenzialità di sfruttamento del parallelismo a livello delle istruzioni:** questo criterio fa riferimento alla capacità di sfruttare le funzionalità di esecuzione parallela nei processori attuali e futuri.

La Tabella 5.2 mostra la valutazione del NIST relativa all'algoritmo Rijndael in base a questi criteri.

Tabella 5.2 Valutazione finale del NIST relativa all'algoritmo Rijndael (2 ottobre 2000).

Sicurezza generale
Non esiste alcun attacco noto alla sicurezza di Rijndael. Rijndael utilizza delle S-box come componenti non lineari. Rijndael sembra avere un margine di sicurezza adeguato, ma ha ricevuto qualche critica che suggeriva che la sua struttura matematica potrebbe essere soggetta ad attacchi. D'altro canto, la semplicità della sua struttura dovrebbe aver facilitato l'analisi della sicurezza durante il processo di sviluppo dello standard AES.
Implementazioni software
Rijndael svolge molto bene le operazioni di crittografia e decrittagrafia in un'ampia varietà di piattaforme fra cui piattaforme a 8 e 64 bit e DSP. Tuttavia vi è una riduzione delle prestazioni quando aumentano le dimensioni delle chiovi dato il maggior numero di fasi che devono essere eseguite. L'elevato parallelismo intrinseco di Rijndoel facilita l'utilizzo efficiente delle risorse del microprocessore garantendo ottime prestazioni software anche quando l'implementazione non strutta l'interleaving. Il tempo di impostazione della chiave di Rijndael è rapido.
Ambienti con spazio limitato
In generale, Rijndael è molto adatto ad ambienti con spazio limitato dove vengono implementate la crittografia o la decrittografia (ma non entrambe). Ha ridottissimi requisiti in termini di RAM e ROM. Un difetto è il fatto che i requisiti di memoria ROM aumentano quando la crittografia e la decrittografia sono entrambe implementate anche se si rimane comunque all'interno dei limiti previsti per questo genere di ambienti. La programmazione della chiave per la decrittografia è distinta da quella della crittografia.
Implementazioni hardware
Rijndael è il più efficiente rispetto a tutti gli altri finalisti per le modalità can feedback e secondo per le modalità senza feedback. Per chiavi di 192 e 256 bit, l'efficienza è simile alle altre implementazioni dato il maggior numero di fasi. Le implementazioni con pipeline aumentano i requisiti in termini di area, ma ne lasciano inalterorata l'efficenza.
Attacchi alle implementazioni
Le operazioni utilizzate do Rijndael sono fra le più facili da difendere dagli attacchi a contrallo dell'energia e del tempo. L'impiego di tecniche di mascheramento fornisce a Rijndael una certa difesa contro questi attacchi senza provocare significativi degradi prestazionali rispetto agli altri finalisti e mantenendo ragionevoli requisiti in termini di memoria RAM. Rijndael sembra acquisire un importante vantaggio in termini di velocità rispetto ai suoi concorrenti quando vengono considerate tali protezioni.
Crittografia e decrittografia
Le funzioni di crittografio e decrittografia Rijndael sono differenti. Una studio della FPGA indica che l'implementazione combinata della crittografia e della decrittografia richiede circa il 60% di spazio in più rispetto all'implementazione della sola crittografia. La velocità di Rijndael non varia significativamente fra crittografio e decrittografia, sebbene i tempi di impostaziane della chiave siano più lenti per la decrittografia rispetto alla crittografia.
Agilità della chiave
Rijndael supporta il calcolo in tempo reole della sottochiave di crittografia. Rijndael richiede una sola esecuzione del processo di progrommazione delle chiavi per generare tutte le sottochiavi prima della prima decrittografia con uno data chiave. Questo introduce un leggero carico in termini di risorse sull'agilità della chiave di Rijndael.

(segue)

Tabella 5.2 Valutazione finale del NIST relativa all'algoritmo Rijndael (2 ottobre 2000). *(continua)*

Altri elementi di versatilità e flessibilità
Rijndael supporta blocchi e chiavi di 128, 192 e 256 bit in qualsiosi combinazione. In linea di principio, la struttura di Rijndael può impiegare blocchi e chiavi di qualsiasi dimensione (ma multipli di 32 bit) e anche variazioni nel numero delle fasi.
Potenzialità in termini di parallelismo delle istruzioni
Rijndael ha eccellenti potenzialità per sfruttare il parallelismo nella crittografia di un singolo blocco.

5.2 La cifratura AES[2]

La proposta Rijndael per AES definiva una cifratura in cui le dimensioni del blocco e della chiave potevano essere specificate in modo indipendente a 128, 192 o 256 bit. Le specifiche di AES prevedono per la chiave queste tre dimènsioni alternative, ma vincolano la lunghezza del blocco a 128 bit. Vi sono vari parametri di AES che dipendono dalla lunghezza della chiave (vedere la Tabella 5.3). In questo paragrafo si presuppone l'impiego di una chiave di 128 bit, che è probabilmente quella implementata più frequentemente.

L'algoritmo Rijndael è stato progettato per avere le seguenti caratteristiche.

- Resistenza contro tutti gli attacchi noti;
- Velocità e compattezza del codice su un'ampia gamma di piattaforme.
- Semplicità progettuale.

La Figura 5.1 mostra la struttura generale di AES. L'input degli algoritmi di crittografia e decrittografia è un singolo blocco di 128 bit. Nel documento FIPS PUB 197 questo blocco è rappresentato come una matrice quadrata di byte. Questo blocco viene copiato nell'array **State** che viene modificato in ciascuna fase della crittografia o decrittografia. Dopo la fase finale, **State** viene copiato in una matrice di output. Queste operazioni sono rappresentate nella Figura 5.2A. Analogamente, la chiave di 128 bit è rappresentata come una matrice quadrata di byte. Questa chiave viene poi espansa in un array di word per la programmazione della chiave; ciascuna word occupa 4 byte e la programmazione totale delle chiavi è di 44 word per una chiave di 128 bit (Figura 5.2B). Si noti che i byte nella matrice sono ordinati colonna per colonna. Dunque, per esempio, i primi 4 byte di un input di testo in chiaro di 128 bit della cifratura di crittografia occupano la prima colonna della matrice **in**, i secondi 4 byte occupano la seconda colonna e così via. Analogamente, i primi 4 byte della chiave espansa, che formano una word, occupano la prima colonna della matrice **w**.

Prima di entrare nei dettagli dell'algoritmo, si possono fare alcuni commenti relativi alla struttura generale di AES.

[2] La maggior parte del materiale presentato in questo paragrafo è tratta da [STAL02].

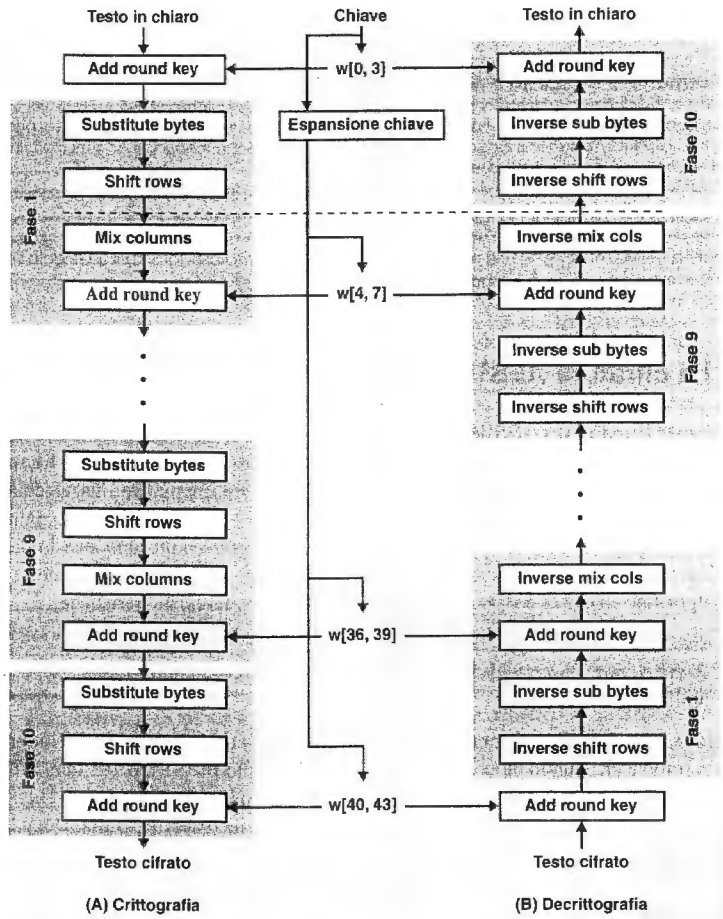


Figura 5.1 Crittografia e decrittografia AES.

Tabella 5.3 I parametri di AES.

Dimensioni della chiave (word/byte/bit)	4/16/128	6/24/192	8/32/256
Dimensioni del blocco di testo in chiaro (word/byte/bit)	4/16/128	4/16/128	4/16/128
Numero di fasi	10	12	14
Dimensioni della chiave di fase (word/byte/bit)	4/16/128	4/16/128	4/16/128
Dimensioni della chiave espansa (word/byte)	44/176	52/208	60/240

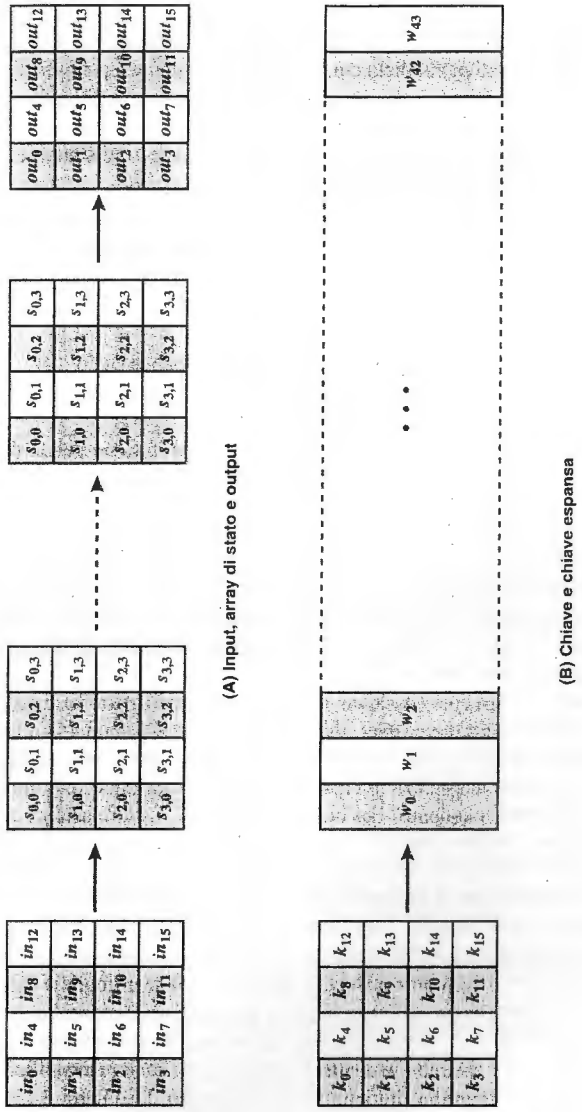


Figura 5.2 Strutture dati di AES.

1. Una caratteristica degna di nota è il fatto che non si tratta di una struttura di Feistel. Come si ricorderà, nella struttura classica di Feistel, una metà del blocco di dati viene

utilizzata per modificare l'altra metà e poi le due metà vengono scambiate. Due degli algoritmi finalisti di AES, fra cui Rijndael, non usano la struttura Feistel ma elaborano l'intero blocco di dati in parallelo durante ciascuna fase utilizzando sostituzioni e permutazioni.

2. La chiave fornita come input viene espansa in un array di 44 word a 32 bit, w[*i*]. Per ciascuna fase vengono utilizzate come chiavi quattro word distinte (128 bit); a tale proposito consultare la Figura 5.1.
3. Vengono utilizzati quattro diversi stadi, ciascuno con una trasformazione, una di permutazione e tre di sostituzione.
   - **Substitute bytes:** usa una S-box per svolgere una sostituzione del blocco byte per byte.
   - **ShiftRows:** una semplice permutazione.
   - **MixColumns:** una sostituzione che utilizza l'aritmetica su GF(2^8).
   - **AddRoundKey:** una semplice operazione di XOR bit-a-bit del blocco corrente con una porzione della chiave espansa.
4. La struttura è piuttosto semplice. Sia per la crittografia che per la decrittografia, l'algoritmo inizia con uno stadio chiamato AddRoundKey, seguito da nove fasi, ognuna delle quali comprende quattro stadi, seguita da una decima fase di tre stadi. La Figura 5.3 rappresenta la struttura di una fase completa della crittografia.
5. Solo la trasformazione AddRoundKey utilizza la chiave. Per questo motivo, la cifratura inizia e termina con una trasformazione AddRoundKey. Ogni altra trasformazione, applicata all'inizio o alla fine, è reversibile senza conoscere la chiave e pertanto non incrementerebbe il livello di sicurezza.
6. La trasformazione AddRoundKey è, in effetti, una forma di cifratura di Vernam e dunque non è particolarmente resistente in se stessa. Le altre tre trasformazioni introducono confusione, diffusione e non linearità ma, in se stessa non aggiungerebbe alcun livello di sicurezza in quanto non impiegano la chiave. Si può considerare la cifratura come una continua alternanza di crittografia XOR (AddRoundKey) di un blocco, seguita da un "rimescolamento" del blocco (le altre tre trasformazioni) seguito da una crittografia XOR e così via. Questo schema è efficiente ed estremamente sicuro.
7. Ogni trasformazione è facilmente reversibile. Per le trasformazioni Substitute Byte, ShiftRow e MixColumns, l'algoritmo di decrittografia utilizza una funzione inversa. Per la trasformazione AddRoundKey, l'inversa viene ottenuta eseguendo uno XOR fra la stessa chiave della fase e il blocco, sfruttando il fatto che $A \oplus A \oplus B = B$.
8. Come per molte altre cifrature a blocchi, l'algoritmo di decrittografia utilizza la chiave espansa in ordine inverso. Tuttavia, l'algoritmo di decrittografia non è identico all'algoritmo di crittografia. Questo è dovuto alla particolare struttura di AES.
9. Dopo aver stabilito che tutte le quattro trasformazioni sono reversibili, è facile dimostrare che la decrittografia riottiene il testo in chiaro. La Figura 5.1 rappresenta la crittografia e la decrittografia secondo direzioni opposte. In ciascun punto orizzontale (la linea tratteggiata in figura), lo **Stato** è lo stesso sia per la crittografia che per la decrittografia.

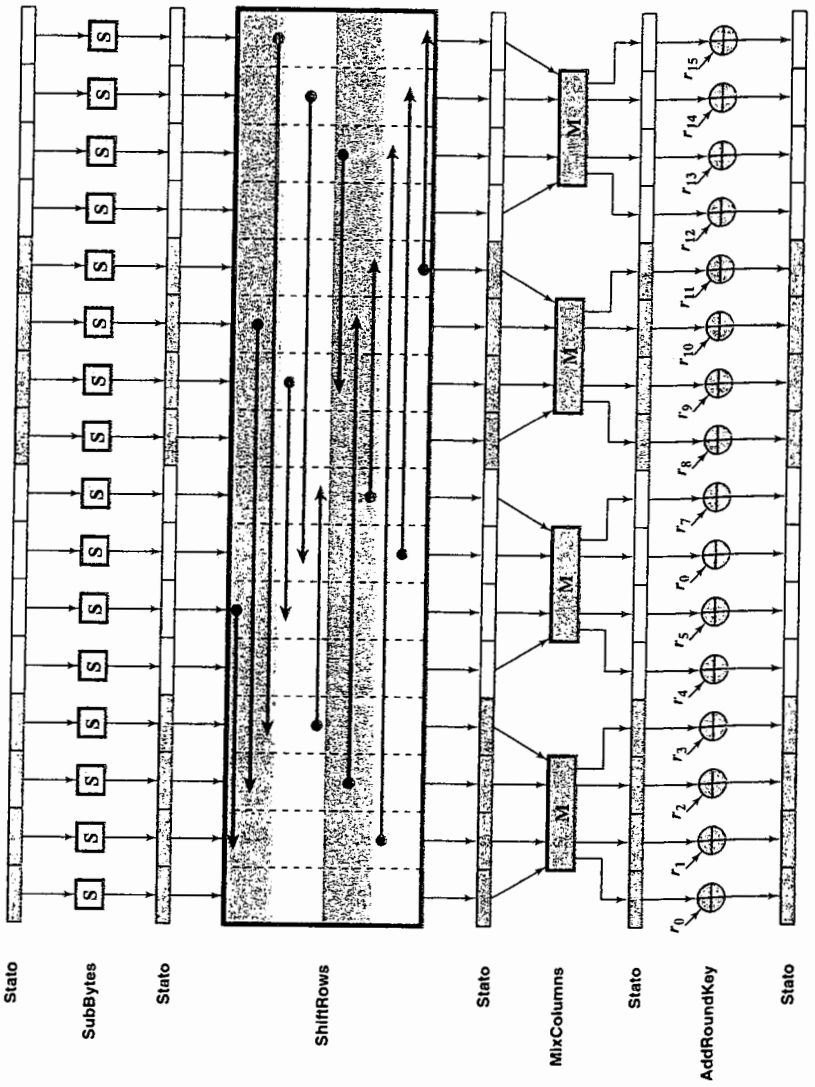


Figura 5.3 Una fase della crittografia di AES.

10. L'ultima fase della crittografia e della decrittografia è costituito unicamente da tre trasformazioni. Anche questo è dovuto alla particolare struttura di AES ed è necessario per rendere reversibile la cifratura.

Ora si tratterà di ciascuna delle quattro trasformazioni utilizzate in AES. Per ciascuna di esse viene presentato l'algoritmo di crittografia, l'algoritmo inverso di decrittografia e le motivazioni delle scelte effettuate. Successivamente verrà trattata l'espansione della chiave. Come si è detto nel Capitolo 4, AES utilizza l'aritmetica dei campi finiti GF(2^8), con il polinomio irriducibile[3] $m(x) = x^8 + x^4 + x^3 + x + 1$. Gli sviluppatori dell'algoritmo Rijndael giustificano la scelta di questo polinomio rispetto ai 30 polinomi irriducibili di grado 8 esistenti, con il fatto che è il primo elencato [LIDL94].

Trasformazione Substitute Bytes

Trasformazioni diretta e inversa

La **trasformazione Substitute Bytes diretta**, chiamata SubBytes, è una semplice ricerca su tabella (Figura 5.4A). AES definisce una matrice di 16 × 16 byte chiamata S-box (Tabella 5.4A) che contiene una permutazione di tutti i 256 valori a 8 bit. Ogni singolo byte di **State** viene mappato in un nuovo byte nel seguente modo: i primi quattro bit del byte indicano la riga e i secondi quattro bit indicano la colonna. Questi valori di riga e colonna fungono da indici della S-box per selezionare un valore univoco di output a 8 bit. Per esempio, il valore esadecimale[4] {95} fa riferimento alla riga 9 colonna 5 della S-box, che contiene il valore {2A}. Di conseguenza, il valore {95} viene mappato sul valore {2A}.

Ecco un esempio della trasformazione eseguita da SubBytes:

EA	04	65	85
83	45	5D	96
5C	33	98	B0
F0	2D	AD	C5

→

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

La S-box viene costruita nel seguente modo.

1. Inizializzare la S-box con il valore dei byte in una sequenza ascendente riga per riga. La prima riga contiene {00}, {01}, {02}, . . . , {0F}; la seconda riga contiene {10}, {11} e così via. Pertanto il valore del byte nella riga x colonna y sarà {xy}.
2. Associare a ciascun byte della S-box il suo inverso moltiplicativo nel campo finito GF(2^8); il valore {00} viene mappato su se stesso.
3. Considerare che ciascun byte della S-box è costituito da 8 bit indicati con (b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0). Applicare la seguente trasformazione a ciascun bit di ciascun byte della S-box:

$$b_i' = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i \qquad (5.1)$$

dove c_i è l'i-esimo bit del byte c con il valore {63}; cioè ($c_7c_6c_5c_4c_3c_2c_1c_0$) = (01100011). L'apice (') indica che la variabile deve essere aggiornata dal valore che si trova alla

[3] Nella parte rimanente di questa discussione, si fa riferimento a GF(2^8) come al campo finito definito da questo polinomio.

[4] In FIPS PUB 197, un numero esadecimale viene racchiuso fra parentesi graffe. In questo capitolo si utilizza la medesima convenzione.

destra. Lo standard AES rappresenta questa trasformazione in forma matriciale nel seguente modo:

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \qquad (5.2)$$

L'Equazione 5.2 deve essere interpretata attentamente. Nella normale moltiplicazione[5] fra matrici, ciascun elemento nella matrice prodotto è dato dalla somma dei prodotti degli elementi di una riga e di una colonna. In questo caso, ciascun elemento della matrice prodotto sarà lo XOR bit-a-bit dei prodotti degli elementi di una riga e di una colonna. Inoltre la somma finale rappresentata nell'Equazione 5.2 è un'operazione di XOR bit-a-bit.

Come esempio, si consideri il valore di input {95}. L'inverso moltiplicativo in GF(2^8) è {95}$^{-1}$ = {8A}, ovvero 10001010 in binario. Utilizzando l'Equazione 5.2,

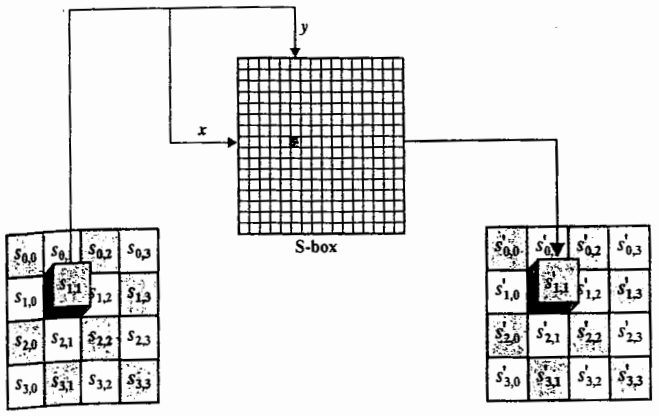
$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Il risultato è {2A} che dovrebbe comparire nella riga {09} colonna {05} della S-box. Si può verificare questo fatto controllando la Tabella 5.4A.

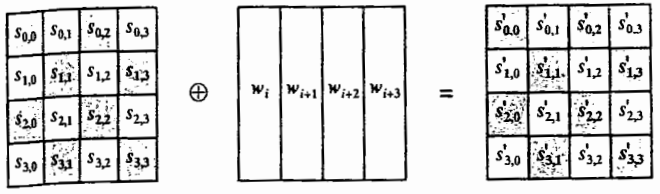
La **trasformazione Substitute Bytes inversa**, chiamata InvSubBytes, utilizza la S-box inversa rappresentata nella Tabella 5.4B. Si noti, per esempio, che l'input {2A} produce l'output {95} e che l'input {95} alla S-box produce {2A}. La S-box inversa viene costruita applicando l'inversa della trasformazione nell'Equazione 5.1 seguita dall'inversa moltiplicativa in GF(2^8). La trasformazione inversa è:

$$b_i' = b_{(i+2) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus d_i$$

[5] Per una breve rassegna delle regole relative alla moltiplicazione di vettori e matrici, vedere il documento MathRefresher disponibile al sito **www.williamstallings.com/StudentSupport.html**.



(A) Trasformazione Substitute Byte



(B) Tasformazione Add Round Key

Figura 5.4 Operazioni sui byte in AES.

dove il byte $d = \{05\}$ ovvero 00000101. Si può rappresentare questa trasformazione nel modo seguente:

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Tabella 5.4 Le S-Box di AES.

(a) S-box

x \ y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

(b) S-box inversa

x \ y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

Per verificare che InvSubBytes è l'inversa di SubBytes, è sufficiente etichettare le matrici di Sub-Bytes e InvSubBytes rispettivamente come **X** e **Y** e le versioni a vettori delle costanti c e d come, rispettivamente, **C** e **D**. Per un vettore a 8 bit **B**, l'Equazione 5.2 diviene $\mathbf{B' = XB \oplus C}$. Occorre dimostrare che $\mathbf{Y(XB \oplus C) \oplus D = B}$. Moltiplicando, si deve dimostrare che $\mathbf{YXB \oplus YC \oplus D = B}$.

Questo diviene:

$$\begin{bmatrix}0&0&1&0&0&1&0&1\\1&0&0&1&0&0&1&0\\0&1&0&0&1&0&0&1\\1&0&1&0&0&1&0&0\\0&1&0&1&0&0&1&0\\0&0&1&0&1&0&0&1\\1&0&0&1&0&1&0&0\\0&1&0&0&1&0&1&0\end{bmatrix}\begin{bmatrix}1&0&0&0&1&1&1&1\\1&1&0&0&0&1&1&1\\1&1&1&0&0&0&1&1\\1&1&1&1&0&0&0&1\\1&1&1&1&1&0&0&0\\0&1&1&1&1&1&0&0\\0&0&1&1&1&1&1&0\\0&0&0&1&1&1&1&1\end{bmatrix}\begin{bmatrix}b_0\\b_1\\b_2\\b_3\\b_4\\b_5\\b_6\\b_7\end{bmatrix}\oplus\begin{bmatrix}0&0&1&0&0&1&0&1\\1&0&0&1&0&0&1&0\\0&1&0&0&1&0&0&1\\1&0&1&0&0&1&0&0\\0&1&0&1&0&0&1&0\\0&0&1&0&1&0&0&1\\1&0&0&1&0&1&0&0\\0&1&0&0&1&0&1&0\end{bmatrix}\begin{bmatrix}1\\1\\0\\0\\0\\1\\1\\0\end{bmatrix}$$

$$\oplus\begin{bmatrix}1\\0\\1\\0\\0\\0\\0\\0\end{bmatrix}=\begin{bmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&1\end{bmatrix}\begin{bmatrix}b_0\\b_1\\b_2\\b_3\\b_4\\b_5\\b_6\\b_7\end{bmatrix}\oplus\begin{bmatrix}1\\0\\1\\0\\0\\0\\0\\0\end{bmatrix}\oplus\begin{bmatrix}1\\0\\1\\0\\0\\0\\0\\0\end{bmatrix}=\begin{bmatrix}b_0\\b_1\\b_2\\b_3\\b_4\\b_5\\b_6\\b_7\end{bmatrix}$$

Si è dimostrato che **YX** è uguale alla matrice identità e che **YC** = **D** e quindi **YC** ⊕ **D** è uguale al vettore nullo.

Motivazioni

La S-box è progettata per essere resistente agli attacchi ad analisi crittografica conosciuti. In particolare, gli sviluppatori dell'algoritmo Rijndael hanno cercato un progetto che avesse una bassa correlazione fra i bit di input e i bit di output e tale che l'output non potesse essere descritto come una semplice funzione matematica dell'input [DAEM01]. Inoltre la costante dell'Equazione 5.1 è stata scelta in modo che la S-box non avesse punti fissi [S-box(a) = a] e nessun "punto fisso opposto" [S-box(a) = $\bar{a}$], dove $\bar{a}$ è il complemento bit-a-bit di a.

Naturalmente la S-box deve essere invertibile, ovvero IS-box[S-box(a)] = a. Tuttavia la S-box non è auto-invertibile, nel senso che non è vero che S-box(a) = IS-box(a). Per esempio, S-box({95}) = {2A}, ma IS-box({95}) = {AD}.

La trasformazione Shift Rows

Trasformazioni diretta e inversa

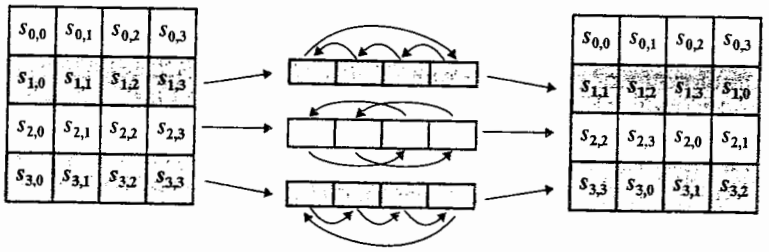
La **trasformazione Shift Rows diretta**, chiamata ShiftRows, è rappresentata nella Figura 5.5A. La prima riga di **State** non viene modificata. Per la seconda riga, viene eseguito uno scorrimento circolare a sinistra di un byte. Per la terza riga, viene eseguito uno scorrimento

circolare a sinistra di 2 byte. Per la quarta riga viene eseguito uno scorrimento circolare a sinistra di 3 byte. Quello che segue è un esempio di ShiftRows:

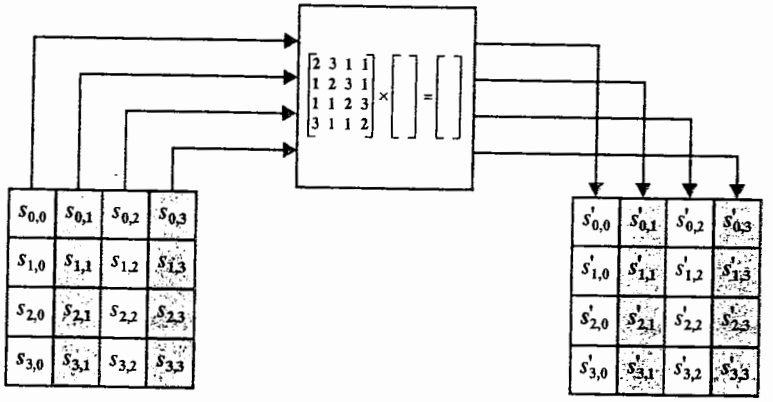
87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

→

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95



(A) Trasformazione Shift Rows



(B) Trasformazione Mix Columns

Figura 5.5 Operazioni sulle righe e sulle colonne in AES.

La **trasformazione Shift Rows inversa**, chiamata InvShiftRows, esegue gli scorrimenti circolari nella direzione opposta per ciascuna delle ultime tre righe, con uno scorrimento circolare a destra di un byte per la seconda riga e così via.

Motivazioni

La trasformazione Shift Rows è più importante di quanto possa sembrare a prima vista per il fatto che **State**, come la cifratura di input e output, viene trattato come un array di 4 colonne di 4 byte. Pertanto, nella crittografia, i primi 4 byte del testo in chiaro vengono copiati nella prima colonna di **State** e così via. Inoltre, come si vedrà, la chiave della fase viene applicata a **State** colonna per colonna. Pertanto uno scorrimento di riga sposta un singolo byte da una colonna a un'altra con una distanza lineare multipla di 4 byte. Inoltre si noti che la trasformazione garantisce che i 4 byte di una colonna vengano dispersi su 4 diverse colonne. La Figura 5.3 illustra questo effetto.

La trasformazione Mix Columns

Trasformazioni diretta e inversa

La **trasformazione Mix Columns diretta**, chiamata MixColumns, opera su ogni singola colonna. Ciascun byte di una colonna viene mappato in un nuovo valore che è una funzione dei quattro byte presenti nella colonna. La trasformazione può essere definita dalla seguente moltiplicazione di matrici su **State** (Figura 5.5B):

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix} \tag{5.3}$$

Ciascun elemento nella matrice prodotto è la somma dei prodotti degli elementi di una riga e una colonna. In questo caso, le singole somme e moltiplicazioni[6] vengono eseguite in GF(2^8). La trasformazione MixColumns su un'unica colonna j ($0 \le j \le 3$) di **State** può essere espressa come:

$$\begin{aligned} s'_{0,j} &= (2 \bullet s_{0,j}) \oplus (3 \bullet s_{1,j}) \oplus s_{2,j} \oplus s_{3,j} \\ s'_{1,j} &= s_{0,j} \oplus (2 \bullet s_{1,j}) \oplus (3 \bullet s_{2,j}) \oplus s_{3,j} \\ s'_{2,j} &= s_{0,j} \oplus s_{1,j} \oplus (2 \bullet s_{2,j}) \oplus (3 \bullet s_{3,j})] \\ s'_{3,j} &= (3 \bullet s_{0,j}) \oplus s_{1,j} \oplus s_{2,j} \oplus (2 \bullet s_{3,j}) \end{aligned} \tag{5.4}$$

Quello che segue è un esempio di trasformazione MixColumns.

[6] Si utilizza la convenzione usata in FIPS PUB 197 e quindi si impiega il simbolo • per indicare la moltiplicazione sul campo finito GF(2^8) e il simbolo ⊕ per indicare l'operazione di XOR bit-a-bit che corrisponde alla somma in GF(2^8).

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

→

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

Si può provare a verificare la prima colonna di questo esempio. Come si ricorderà dal Paragrafo 4.6, in GF(2^8) la somma è l'operazione di XOR bit-a-bit e la moltiplicazione può essere eseguita in base alla regola stabilita nell'Equazione 4.10. In particolare, la moltiplicazione di un valore per x (per esempio per {02}) può essere implementata come uno scorrimento a sinistra di un bit seguito da un'operazione di XOR bit-a-bit condizionale con (0001 1011) se il bit più a sinistra del valore originale (prima dello scorrimento) è 1. Pertanto, per verificare la trasformazione MixColumns sulla prima colonna, occorre dimostrare che:

$$\begin{aligned} (\{02\} \bullet \{87\}) &\oplus (\{03\} \bullet \{6E\}) \oplus \{46\} \oplus \{A6\} &&= \{47\} \\ \{87\} &\oplus (\{02\} \bullet \{6E\}) \oplus (\{03\} \bullet \{46\}) \oplus \{A6\} &&= \{37\} \\ \{87\} &\oplus \{6E\} \oplus (\{02\} \bullet \{46\}) \oplus (\{03\} \bullet \{A6\}) &&= \{94\} \\ (\{03\} \bullet \{87\}) &\oplus \{6E\} \oplus \{46\} \oplus (\{02\} \bullet \{A6\}) &&= \{ED\} \end{aligned}$$

Per la prima equazione si ha {02} • {87} = (0000 1110) ⊕ (0001 1011) = (0001 0101) e {03} • {6E } = {6E} ⊕ ({02} • {6E }) = (0110 1110) ⊕ (1101 1100) = (1011 0010).

Pertanto:

$$\begin{aligned} \{02\} \bullet \{87\} &= 0001\ 0101 \\ \{03\} \bullet \{6E\} &= 1011\ 0010 \\ \{46\} &= 0100\ 0110 \\ \{A6\} &= \underline{1010\ 0110} \\ & 0100\ 0111 = \{47\} \end{aligned}$$

Le altre equazioni possono essere verificate in modo analogo.

La **trasformazione Mix Columns inversa**, chiamata InvMixColumns, è definita dalla seguente moltiplicazione fra matrici:

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix} \tag{5.5}$$

Non è immediatamente chiaro che l'Equazione 5.5 è l'inversa dell'Equazione 5.3. Occorre dimostrare che:

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 02 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix}$$

che è equivalente a dire che:

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.6)$$

Ovvero la matrice di trasformazione inversa moltiplicata per la matrice di trasformazione diretta è uguale alla matrice identità. Per verificare la prima colonna dell'Equazione 5.6 occorre dimostrare che:

$$(\{0E\} \bullet \{02\}) \oplus \{0B\} \oplus \{0D\} \oplus (\{09\} \bullet \{03\}) = \{01\}$$
$$(\{09\} \bullet \{02\}) \oplus \{0E\} \oplus \{0B\} \oplus (\{0D\} \bullet \{03\}) = \{00\}$$
$$(\{0D\} \bullet \{02\}) \oplus \{09\} \oplus \{0E\} \oplus (\{0B\} \bullet \{03\}) = \{00\}$$
$$(\{0B\} \bullet \{02\}) \oplus \{0D\} \oplus \{09\} \oplus (\{0E\} \bullet \{03\}) = \{00\}$$

Per la prima equazione si ha $\{0E\} \bullet \{02\} = 00011100$ e $\{09\} \bullet \{03\} = \{09\} \oplus (\{09\} \bullet \{02\}) = 00001001 \oplus 00010010 = 00011011$. Pertanto:

$\{0E\} \bullet \{02\}$	=	00011100
$\{0B\}$	=	00001011
$\{0D\}$	=	00001101
$\{09\} \bullet \{03\}$	=	00011011
		00000001

Le altre equazioni possono essere verificate in modo analogo.

Il documento AES descrive un altro modo per caratterizzare la trasformazione MixColumns, in termini di aritmetica polinomiale. Nello standard, la trasformazione MixColumns viene definita considerando ciascuna colonna di **State** come un polinomio di quattro termini con i coefficienti in GF(2^8). Ciascuna colonna viene moltiplicata modulo $(x^4 + 1)$ per il polinomio fisso $a(x)$, dato da:

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} \quad (5.7)$$

L'Appendice 5A dimostra che la moltiplicazione di ciascuna colonna di **State** per $a(x)$ può essere scritta come la moltiplicazione di matrici dell'Equazione 5.3. Analogamente si può vedere che la trasformazione nell'Equazione 5.5 corrisponde a trattare ciascuna colonna come un polinomio di quattro termini e a moltiplicare ciascuna colonna per $b(x)$, dato da:

$$b(x) = \{0B\}x^3 + \{0D\}x^2 + \{09\}x + \{0E\} \quad (5.8)$$

È facile dimostrare che $b(x) = a^{-1}(x) \bmod (x^4 + 1)$.

Motivazioni

I coefficienti della matrice dell'Equazione 5.3 si basano su un codice lineare con una distanza massima fra le word di codice che garantisce una buona dispersione fra i byte di ciascuna colonna. La trasformazione Mix Columns, combinata con la trasformazione Shift Rows, garantisce che dopo alcune fasi, tutti i bit di output dipendano da tutti i bit di input. Per approfondimenti consultare [DAEM99].

Inoltre, la scelta dei coefficienti di MixColumns, che sono tutti {01}, {02} o {03} è stata influenzata da considerazioni implementative. Come si è detto, la moltiplicazione per questi coefficienti prevede al massimo uno scorrimento e uno XOR. I coefficienti di InvMixColumns sono più difficili da implementare. Tuttavia la crittografia è stata ritenuta più importante della decrittografia per due motivi.

1. Per le modalità di cifratura CFB e OFB (Figure 6.5 e 6.6) viene utilizzata solo la crittografia.
2. Come per ogni cifratura a blocchi, AES può essere utilizzato per costruire un codice di autenticazione dei messaggi (vedere la Parte Seconda) e per questo viene utilizzata solo la crittografia.

La trasformazione Add Round Key

Trasformazioni diretta e inversa

Nella **trasformazione Add Round Key diretta**, chiamata AddRoundKey, i 128 bit di **State** vengono sottoposti a uno XOR bit-a-bit con i 128 bit della chiave della fase. Come si può vedere nella Figura 5.4B, si tratta di un'operazione a colonne fra i 4 byte della colonna **State** e una word della chiave di fase; può anche essere considerata come una operazione a livello di byte. Quello che segue è un esempio di AddRoundKey:

47	40	A3	4C	AC	19	28	57	EB	59	8B	1B
37	D4	70	9F	77	FA	D1	5C	40	2E	A1	C3
94	E4	3A	42	66	DC	29	00	F2	38	13	42
ED	A5	A6	BC	F3	21	41	6A	1E	84	E7	D2

$\oplus$ $=$

La prima matrice è **State** e la seconda matrice è la chiave di fase.

La **trasformazione Add Round Key inversa** è identica a quella diretta in quanto l'operazione di XOR è invertibile.

Motivazioni

La trasformazione Add Round Key è semplicissima e interviene su tutti i bit di **State**. La sicurezza è garantita dalla complessità dell'espansione della chiave di fase e dalla complessità degli altri stadi di AES.

Espansione della chiave AES

Algoritmo di espansione della chiave

L'algoritmo di espansione della chiave di AES richiede come input una chiave di 4 word (16 byte) e produce un array lineare di 44 word (176 byte). Questo è sufficiente per fornire una chiave di fase di 4 word per la trasformazione Add Round Key iniziale e per ciascuna delle dieci fasi della cifratura. L'espansione è descritta dal seguente pseudocodice:

```
KeyExpansion (byte key[16], word w[44])
{
        word temp
        for (i = 0; i < 4; i++) w[i] = (key[4*i], key[4*i + 1], key[4*i + 2],
key[4*i + 3]);
        for (i = 4; i < 44; i++)
        {
                temp = w[i - 1];
                if (i mod 4 = 0) temp = SubWord (RotWord (temp)) ⊕ Rcon[i/4];
                w[i] = w[i-4] ⊕ temp
        }
}
```

La chiave viene copiata nelle prime quattro word della chiave espansa. Il resto della chiave espansa viene riempito quattro word alla volta. Ciascuna word aggiunta w[i] dipende dalla word immediatamente precedente, w[i – 1], e dalla word che si trova quattro posizioni indietro, w[i – 4]. In tre casi su quattro viene utilizzata una semplice operazione XOR. Per ogni word la cui posizione nell'array w è un multiplo di 4, viene utilizzata una funzione più complessa. La Figura 5.6 illustra la generazione delle prime 8 word della chiave espansa utilizzando il simbolo g per rappresentare tale funzione complessa. La funzione g è costituita dalle seguenti sottofunzioni.

1. RotWord svolge uno scorrimento a sinistra circolare di un byte su una word. Questo significa che una word di input [b0, b1, b2, b3] viene trasformata in [b1, b2, b3, b0].
2. SubWord svolge una sostituzione su ciascun byte della word di input, utilizzando la S-box (Tabella 5.4A).
3. Il risultato dei passi 1 e 2 è sottoposto a uno XOR con una costante di fase, Rcon[j].

La costante di fase è una word in cui i tre byte più a destra sono sempre 0. Pertanto l'effetto di un'operazione di XOR di una word con Rcon interessa solo il byte più a sinistra della word. La costante di fase è differente per ciascuna fase ed è definita come Rcon[j] = (RC[j], 0, 0, 0), con RC[1] = 1, RC[j] = 2 • RC[j – 1] e con la moltiplicazione definita sul campo GF(2^8). I valori di RC[j] in esadecimale sono:

j	1	2	3	4	5	6	7	8	9	10
RC[j]	01	02	04	08	10	20	40	80	1B	36

Per esempio, si supponga che la chiave per la fase 8 sia:

EA D2 73 21 B5 8D BA D2 31 2B F5 60 7F 8D 29 2F

I primi 4 byte (prima colonna) della chiave per la fase 9 vengono calcolati nel modo seguente:

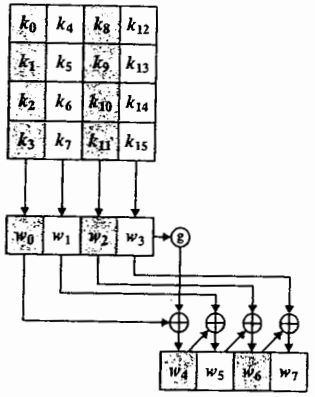


Figura 5.6 Espansione della chiave in AES.

i (dec.)	temp	Dopo RotWord	Dopo SubWord	Rcon (9)	Dopo XOR con Rcon	w(i – 4)	w(i) = temp ⊕ w(i – 4)
36	7F8D292F	8D292F7F	5DA515D2	1B000000	46A515D2	EAD27321	AC7766F3

Motivazioni

Gli sviluppatori di Rijndael hanno progettato l'algoritmo di espansione della chiave in modo che fosse resistente agli attacchi ad analisi crittografica noti. L'inclusione di una costante di fase elimina la simmetria o la similitudine fra le modalità con le quali vengono generate le chiavi nelle varie fasi. I criteri utilizzati [DAEM99] sono i seguenti.

- La conoscenza di una parte della chiave di cifratura o della chiave di fase non consente di calcolare molti altri bit della chiave di fase.
- La trasformazione è invertibile, ovvero la conoscenza di *Nk* word consecutive della chiave espansa consente la rigenerazione dell'intera chiave espansa (dove *Nk* = dimensioni della chiave in word).
- Alta velocità su un'ampia gamma di processori.
- Uso di costanti di fase per eliminare le simmetrie.
- Diffusione delle differenze della chiave di cifratura nelle chiavi di fase; ovvero ciascun bit della chiave influenza più bit della chiave di fase.
- Sufficiente non-linearità da impedire la determinazione completa delle differenze della chiave di fase utilizzando solo le differenze della chiave di cifratura.
- Facilità di descrizione.

Gli autori non quantificano il primo punto dell'elenco precedente ma l'idea è che se si conoscono meno di *Nk* word consecutive della chiave di cifratura o di una delle chiavi di fase, sarà difficile ricostruire i rimanenti bit sconosciuti. Meno bit si conoscono, più difficile è ricostruire o determinare gli altri bit della chiave espansa.

Cifratura inversa equivalente

Come si è detto, la decifratura di AES non è identica alla cifratura (vedere la Figura 5.1): la sequenza di trasformazioni per la decrittografia è diversa da quella per la crittografia anche se la forma della programmazione della chiave per la crittografia e la decrittografia coincidono. Questo rappresenta uno svantaggio, in quanto sono richiesti due diversi moduli software o firmware per le applicazioni che richiedono sia la crittografia che la decrittografia. Vi è però una versione equivalente dell'algoritmo di decrittografia che ha la stessa struttura dell'algoritmo di crittografia. La versione equivalente usa la stessa sequenza di trasformazioni dell'algoritmo di crittografia (utilizzando le trasformazioni inverse). Per ottenere questa equivalenza, occorre modificare la programmazione delle chiavi.

Sono necessarie due diverse modifiche per allineare la struttura di decrittografia con la struttura di crittografia. Una fase di crittografia ha la struttura SubBytes, ShiftRows, MixColumns, AddRoundKey. Una fase standard di decrittografia ha la struttura InvShiftRows, InvSubBytes, AddRoundKey, InvMixColumns. Pertanto si devono scambiare le prime due trasformazioni e le seconde due trasformazioni della fase di decrittografia.

Scambio fra InvShiftRows e InvSubBytes

InvShiftRows modifica la sequenza di byte di **State** ma non ne altera il contenuto e non dipende dal contenuto per svolgere la propria trasformazione. InvSubBytes modifica il contenuto dei byte di **State** ma non ne altera la sequenza e non dipende dalla sequenza per svolgere la propria trasformazione. Pertanto queste due operazioni possono essere scambiate fra loro. Per un determinato **State** S_i:

$$\text{InvShiftRows [InvSubBytes } (S_i)] = \text{InvSubBytes [InvShiftRows } (S_i)]$$

Scambio di AddRoundKey e InvMixColumns

Le trasformazioni AddRoundKey e InvMixColumns non alterano la sequenza di byte di **State**. Se si considera la chiave come una sequenza di word, allora sia AddRoundKey che InvMixColumns operano su **State** una colonna per volta. Queste due operazioni sono lineari rispetto all'input della colonna. Pertanto, per un determinato **State** S_i e una determinata chiave di fase w_j:

$$\text{InvMixColumns } (S_i \oplus w_j) = [\text{InvMixColumns } (S_i)] \oplus [\text{InvMixColumns } (w_j)]$$

Per dimostrare ciò, si supponga che la prima colonna di **State** S_i sia la sequenza (y_0, y_1, y_2, y_3) e che la prima colonna della chiave di fase w_j sia (k_0, k_1, k_2, k_3). Occorre dimostrare che:

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} y_0 \oplus k_0 \\ y_1 \oplus k_1 \\ y_2 \oplus k_2 \\ y_3 \oplus k_3 \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} \oplus \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} k_0 \\ k_1 \\ k_2 \\ k_3 \end{bmatrix}$$

Si proverà a eseguire la dimostrazione per la prima colonna. Occorre dimostrare che:

$$\begin{aligned}
&[\{0E\} \bullet (y_0 \oplus k_0)] \oplus [\{0B\} \bullet (y_1 \oplus k_1)] + \oplus [\{0D\} \bullet (y_2 \oplus k_2)] \oplus [\{09\} \bullet (y_3 \oplus k_3)] = \\
&[\{0E\} \bullet y_0] \oplus [\{0B\} \bullet y_1] \oplus [\{0D\} \bullet y_2] \oplus [\{09\} \bullet y_3] \oplus \\
&[\{0E\} \bullet k_0] \oplus [\{0B\} \bullet k_1] \oplus [\{0D\} \bullet k_2] \oplus [\{09\} \bullet k_3]
\end{aligned}$$

Questa equazione è evidentemente valida. Pertanto si possono scambiare AddRoundKey e InvMixColumns, sempre che si applichi innanzitutto InvMixColumns alla chiave di fase. Si noti che non occorre applicare InvMixColumns alla chiave di fase per l'input della prima trasformazione AddRoundKey (che precede la prima fase) né per l'ultima trasformazione AddRoundKey (nella fase 10). Ciò è dovuto al fatto che queste due trasformazioni AddRound-Key non vengono scambiate con InvMixColumns per produrre l'algoritmo di decrittografia equivalente.

La Figura 5.7 illustra l'algoritmo equivalente di decrittografia.

Aspetti implementativi

La proposta Rijndael [DAEM99] fornisce alcuni suggerimenti per un'implementazione efficiente su microprocessori a 8 bit tipicamente presenti sulle attuali smart card e su microprocessori a 32 bit, tipici dei PC.

Microprocessori a 8 bit

AES può essere implementato in modo molto efficiente su microprocessori a 8 bit. AddRoundkey è un'operazione di XOR byte per byte. ShiftRows è una semplice operazione di scorrimento dei byte. SubBytes opera a livello di byte e richiede solo una tabella di 256 byte.

La trasformazione MixColumns richiede la moltiplicazione matriciale nel campo GF(2^8): tutte le operazioni vengono quindi svolte su byte. MixColumns richiede solo una moltiplicazione per {02} e {03} che, come si è visto, comporta semplici scorrimenti, XOR condizionali e XOR. L'operazione può essere implementata in modo più efficiente eliminando gli scorrimenti e gli XOR condizionali. L'insieme di equazioni 5.4 mostra le equazioni per la trasformazione MixColumns su un'unica colonna. Utilizzando l'identità $\{03\} \bullet x = (\{02\} \bullet x) \oplus x$, si possono riscrivere le Equazioni 5.4 nel seguente modo:

$$\begin{aligned}
Tmp &= s_{0,j} \oplus s_{1,j} \oplus s_{2,j} \oplus s_{3,j} \\
s'_{0,j} &= s_{0,j} \oplus Tmp \oplus [2 \bullet (s_{0,j} \oplus s_{1,j})] \\
s'_{1,j} &= s_{1,j} \oplus Tmp \oplus [2 \bullet (s_{1,j} \oplus s_{2,j})] \\
s'_{2,j} &= s_{2,j} \oplus Tmp \oplus [2 \bullet (s_{2,j} \oplus s_{3,j})] \\
s'_{3,j} &= s_{3,j} \oplus Tmp \oplus [2 \bullet (s_{3,j} \oplus s_{0,j})]
\end{aligned} \tag{5.9}$$

Le Equazioni 5.9 possono essere verificate espandendo ed eliminando i termini.

La moltiplicazione per {02} comporta uno scorrimento e uno XOR condizionale. Tale implementazione può essere vulnerabile ad attacchi temporali del tipo descritto nel Paragrafo 3.4. Per difendersi da questi attacchi e per migliorare l'efficienza di elaborazione al

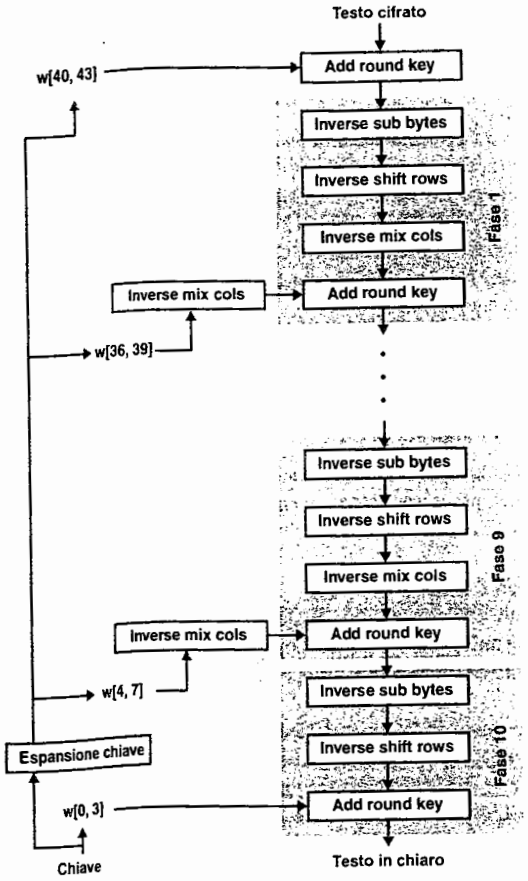


Figura 5.7 Cifratura equivalente inversa.

costo di un po' di spazio di memoria, la moltiplicazione può essere sostituita da una ricerca in tabella. Si può definire la tabella di 256 byte X2 tale che X2[i]={02} • i. Le Equazioni 5.9 possono essere quindi riscritte come:

$$Tmp = s_{0,j} \oplus s_{1,j} \oplus s_{2,j} \oplus s_{3,j}$$
$$s'_{0,j} = s_{0,j} \oplus Tmp \oplus X2[(s_{0,j} \oplus s_{1,j})]$$
$$s'_{1,j} = s_{1,j} \oplus Tmp \oplus X2[(s_{1,j} \oplus s_{2,j})]$$
$$s'_{2,j} = s_{2,j} \oplus Tmp \oplus X2[(s_{2,j} \oplus s_{3,j})]$$
$$s'_{3,j} = s_{3,j} \oplus Tmp \oplus X2[(s_{3,j} \oplus s_{0,j})]$$

Microprocessori a 32 bit

L'implementazione appena descritta utilizza solo operazioni a 8 bit. Per microprocessori a 32 bit è possibile ottenere un'implementazione più efficiente definendo le operazioni su word di 32 bit. Per dimostrare ciò occorre innanzitutto definire le quattro trasformazioni di una fase in forma algebrica. Si supponga di iniziare con una matrice **State** con elementi $a_{i,j}$ e una matrice di chiavi di fase con elementi $k_{i,j}$. Le trasformazioni possono essere espresse nel seguente modo:

SubBytes	$b_{ij} = S[a_{i,j}]$
ShiftRows	$\begin{bmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{bmatrix} = \begin{bmatrix} b_{0,j} \\ b_{1,j-1} \\ b_{2,j-2} \\ b_{3,j-3} \end{bmatrix}$
MixColumns	$\begin{bmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{bmatrix}$
AddRoundKey	$\begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} = \begin{bmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{bmatrix} \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$

Nell'equazione ShiftRows, gli indici della colonna vengono presi modulo 4. Si possono combinare tutte queste espressioni in un'unica equazione:

$$\begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S[a_{0,j}] \\ S[a_{1,j-1}] \\ S[a_{2,j-2}] \\ S[a_{3,j-3}] \end{bmatrix} \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix} =$$
$$\left(\begin{bmatrix} 02 \\ 01 \\ 01 \\ 03 \end{bmatrix} \bullet S[a_{0,j}] \right) \oplus \left(\begin{bmatrix} 03 \\ 02 \\ 01 \\ 01 \end{bmatrix} \bullet S[a_{0,j-1}] \right) \oplus \left(\begin{bmatrix} 01 \\ 03 \\ 02 \\ 01 \end{bmatrix} \bullet S[a_{0,j-2}] \right) \oplus \left(\begin{bmatrix} 01 \\ 01 \\ 03 \\ 02 \end{bmatrix} \bullet S[a_{0,j-3}] \right) \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$$

Nella seconda equazione si esprime la moltiplicazione di matrici come una combinazione lineare di vettori. Si definiscono quattro tabelle da 256 word (1024 byte) nel modo seguente:

$$T_0[x] = \begin{pmatrix} \begin{bmatrix} 02 \\ 01 \\ 01 \\ 03 \end{bmatrix} \bullet S[x] \end{pmatrix} \quad T_1[x] = \begin{pmatrix} \begin{bmatrix} 03 \\ 02 \\ 01 \\ 01 \end{bmatrix} \bullet S[x] \end{pmatrix} \quad T_2[x] = \begin{pmatrix} \begin{bmatrix} 01 \\ 03 \\ 02 \\ 01 \end{bmatrix} \bullet S[x] \end{pmatrix} \quad T_3[x] = \begin{pmatrix} \begin{bmatrix} 01 \\ 01 \\ 03 \\ 02 \end{bmatrix} \bullet S[x] \end{pmatrix}$$

Pertanto ciascuna tabella accetta come input un valore di un byte e produce un vettore colonna (una word di 32 bit) che è funzione della voce della S-box corrispondente a tale byte. Queste tabelle possono anche essere pre-calcolate.

Si può definire una funzione di fase che opera su una colonna nel modo seguente:

$$\begin{bmatrix} s_{0,j} \\ s_{1,j} \\ s_{2,j} \\ s_{3,j} \end{bmatrix} = T_0[S_{0,j}] \oplus T_1[S_{1,j-1}] \oplus T_2[S_{2,j-2}] \oplus T_3[S_{3,j-3}] \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$$

Come risultato, un'implementazione basata sull'equazione precedente richiede solo quattro ricerche su tabelle e quattro XOR per colonna per fase più 4 KB per memorizzare la tabella. Gli sviluppatori di Rijndael ritengono che questa implementazione così compatta ed efficiente sia stata uno dei fattori più importanti che ha fatto propendere per la scelta dell'algoritmo Rijndael per AES.

5.3 Letture e siti Web consigliati

La descrizione più completa di AES attualmente disponibile è il volume scritto dai suoi progettisti, [DAEM02]. Gli autori forniscono anche una descrizione più breve e le motivazioni progettuali in [DAEM01]. [LAND04] è una trattazione matematica rigorosa di AES e della relativa analisi crittografica.

DAEM01 J. Daemen e V. Rijmen. "Rijndael: The Advanced Encryption Standard". *Dr. Dobb's Journal*, Marzo 2001.

DAEM02 J. Daemen e V. Rijmen. *The Design of Rijndael: The Wide Trail Strategy Explained*. New York, Springer-Verlag, 2000.

LAND04 S. Landau. "Polynomials in the Nation's Service: Using Algebra to Design the Advanced Encryption Standard". *American Mathematical Monthly*, Febbraio 2004.

Siti Web consigliati

- **AES homepage:** la pagina del NIST su AES. Contiene lo standard e vari altri documenti.
- **The AES longue:** contiene una vasta bibliografia di documenti e articoli relativi ad AES, con accesso alle copie elettroniche.

5.4 Termini chiave, domande di ripasso e problemi

Termini chiave

AES (Advanced Encryption Standard)
Analisi della potenza
NIST (National Institute of Standards and Technology)
Rijndael
S-box

Domande di ripasso

5.1 Quali sono i criteri originariamente utilizzati dal NIST per valutare i candidati della cifratura AES?
5.2 Quali sono i criteri finali utilizzati dal NIST per valutare i candidati della cifratura AES?
5.3 Che cos'è l'analisi della potenza?
5.4 Qual è la differenza fra Rijndael e AES?
5.5 Qual è lo scopo dell'array **State**?
5.6 Come è costruita una S-box?
5.7 Descrivere brevemente la trasformazione SubBytes.
5.8 Descrivere brevemente la trasformazione ShiftRows.
5.9 Quanti byte di **State** sono coinvolti dalla trasformazione ShiftRows?
5.10 Descrivere brevemente la trasformazione MixColumns.
5.11 Descrivere brevemente la trasformazione AddRoundKey.
5.12 Descrivere brevemente l'algoritmo di espansione della chiave.
5.13 Qual è la differenza fra le trasformazioni SubBytes e SubWord?
5.14 Qual è la differenza fra le trasformazioni ShiftRows e RotWord?
5.15 Qual è la differenza fra l'algoritmo di decrittografia AES e la cifratura inversa equivalente?

Problemi

5.1 Nella discussione delle trasformazioni MixColumns e Inverse MixColumns si è detto che:

$$b(x) = a^{-1}(x) \bmod (x^4 + 1)$$

dove $a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$ e $b(x) = \{0B\}x^3 + \{0D\}x^2 + \{09\}x + \{0E\}$. Dimostrare la validità dell'affermazione.

5.2. A. Che cos'è $\{01\}^{-1}$ in GF(2^8)?
B. Verificare la voce per {01} nella S-box.

5.3 Mostrare le prime 8 word della chiave espansa per una chiave di 128 bit costituita unicamente da valori 0.

5.4 Dato il testo in chiaro {00010203040506070809 0A0B0C0D0E0F} e la chiave {0101010101010101010101010101010101}, svolgere le seguenti operazioni.
   A. Mostrare il contenuto originale di **State**, rappresentato come una matrice 4 × 4.
   B. Mostrare il valore di **State** dopo la trasformazione AddRoundKey iniziale.
   C. Mostrare il valore di **State** dopo la trasformazione SubBytes.
   D. Mostrare il valore di **State** dopo la trasformazione ShiftRows.
   E. Mostrare il valore di **State** dopo la trasformazione MixColumns.

5.5 Verificare l'Equazione 5.11, ovvero dimostrare che $x^i \bmod (x^4 + 1) = x^{i \bmod 4}$.

5.6 Confrontare AES con DES. Per ognuno dei seguenti elementi di DES indicare l'elemento corrispondente in AES o spiegare perché non è necessario in AES:
   A. XOR della sottochiave con l'input della funzione f.
   B. XOR dell'output della funzione f con la metà sinistra del blocco.
   C. La funzione f.
   D. La permutazione P.
   E. Lo scambio delle metà del blocco.

5.7 Nel paragrafo sugli aspetti implementativi, si dice che le tabelle aiutano a complicare gli attacchi a tempo. Suggerire una tecnica alternativa.

5.8 Nel paragrafo sugli aspetti implementativi, viene sviluppata una singola equazione algebrica che descrive i quattro stadi di una tipica fase dell'algoritmo di crittografia. Fornire l'equazione equivalente per la decima fase.

5.9 Calcolare il risultato della trasformazione MixColumns per la seguente sequenza di byte di input: "67 89 AB CD". Applicare la trasformazione InvMixColumns al risultato ottenuto per verificare la correttezza dei calcoli. Sostituire il primo byte di input con "77", calcolare nuovamente il risultato della trasformazione MixColumns e determinare quanti bit sono cambiati nel risultato. Nota: è possibile eseguire i calcoli manualmente o scrivere un opportuno programma. Se si decide di scrivere il programma, non utilizzare librerie o codici sorgente di pubblico dominio.

5.10 Utilizzare la chiave 1010 0111 0011 1011 per crittografare il testo in chiaro "ok" codificato in ASCII, ovvero 0110 1111 0110 1011. I progettisti di S-AES ottennero il testo cifrato 0000 0111 0011 1000. È corretto?

5.11 Dimostrare che la matrice seguente, i cui elementi sono in GF(2^4), è l'inversa della matrice utilizzata nel passo MixColumns di S-AES.

$$\begin{pmatrix} x^3+1 & x \\ x & x^3+1 \end{pmatrix}$$

5.12 Decifrare con attenzione il testo cifrato 0000 0111 0011 1000 utilizzando la chiave 1010 0111 0011 1011 e l'algoritmo S-AES. Si dovrebbe ottenere il testo in chiaro con il quale inizia il Problema 5.10. Si noti che l'inverso della S-box può essere calcolato con una tabella di lookup invertito. L'inverso del passo MixColumns è definito dalla matrice del problema precedente.

Esercizi di programmazione

5.13 Codificare un programma che utilizzi S-AES per la crittografia e la decrittografia. Per verificarne il corretto funzionamento utilizzare il testo in chiaro binario 0110 1111 0110 1011 da cifrare con chiave binaria 1010 0111 0011 1011, che dovrebbe produrre il testo cifrato binario 0000 0111 0011 1000). La decrittografia dovrebbe operare in modo corrispondente.

5.14 Implementare un attacco ad analisi crittografica sulla prima fase di S-AES.

Appendice 5.A Polinomi con coefficienti in GF(2^8)

Nel Paragrafo 4.5 si è parlato dell'aritmetica dei polinomi con coefficienti in Z_p e definiti modulo un polinomio $M(x)$ la cui potenza più elevata è un intero n. In questo caso, la somma e la moltiplicazione dei coefficienti rimane nel campo Z_p, ovvero la somma e la moltiplicazione vengono eseguiti modulo p.

Il documento AES definisce l'aritmetica per polinomi di grado 3 o inferiore con coefficienti GF(2^8). Si applicano le seguenti regole.

1. L'addizione viene eseguita sommando i coefficienti corrispondenti in GF(2^8). Come si è detto nel Paragrafo 4.5, se si trattano gli elementi di GF(2^8) come stringhe di 8 bit, la somma diviene equivalente all'operazione di XOR. Dunque si avrà:

$$a(x) = a_3x^3 + a_2x^2 + a_1x + a_0 \tag{5.8}$$

$$b(x) = b_3x^3 + b_2x^2 + b_1x + b_0 \tag{5.9}$$

   quindi:

$$a(x) + b(x) = (a_3 \oplus b_3)x^3 + (a_2 \oplus b_2)x^2 + (a_1 \oplus b_1)x + (a_0 \oplus b_0)$$

2. La moltiplicazione viene eseguita come una normale moltiplicazione di polinomi con due differenze.
   A. I coefficienti vengono moltiplicati in GF(2^8).
   B. Il polinomio risultante viene ridotto modulo ($x^4 + 1$).

Occorre chiarire quali polinomi vengono utilizzati. Si ricordi dal Paragrafo 4.6 che ciascun elemento di GF(2^8) è un polinomio di grado 7 o inferiore con coefficienti binari e che la moltiplicazione viene eseguita modulo un polinomio di grado 8. In modo equivalente, ciascun elemento di GF(2^8) può essere considerato come un byte di 8 bit in cui il valore dei bit corrisponde ai coefficienti binari del polinomio corrispondente. Per gli insiemi di cui si parla in questo paragrafo si sta definendo un anello di polinomi in cui ciascun elemento è un polinomio di grado 2 o inferiore con coefficienti in GF(2^8) e la moltiplicazione viene eseguita modulo un polinomio di grado 4. In modo equivalente, ciascun elemento di questo anello può essere considerato come una word di 4 byte i cui byte sono elementi di GF(2^8) che corrispondono ai coefficienti a 8 bit del polinomio corrispondente.

Si denota il prodotto modulare di $a(x)$ e $b(x)$ come $a(x) \otimes b(x)$. Per calcolare $d(x) = a(x) \otimes b(x)$, il primo passo è quello di eseguire una moltiplicazione senza l'operazione di modulo e raccogliere i coefficienti di pari potenza. Si può esprimere la cosa come $c(x) = a(x) \times b(x)$. Allora:

$$c(x) = c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0 \tag{5.10}$$

dove:

$$
\begin{aligned}
c_0 &= a_0 \bullet b_0 \\
c_1 &= (a_1 \bullet b_0) \oplus (a_0 \bullet b_1) \\
c_2 &= (a_2 \bullet b_0) \oplus (a_1 \bullet b_1) \oplus (a_0 \bullet b_2) \\
c_3 &= (a_3 \bullet b_0) \oplus (a_2 \bullet b_1) \oplus (a_1 \bullet b_2) \oplus (a_0 \bullet b_3) \\
c_4 &= (a_3 \bullet b_1) \oplus (a_2 \bullet b_2) \oplus (a_1 \bullet b_3) \\
c_5 &= (a_3 \bullet b_2) \oplus (a_2 \bullet b_3) \\
c_6 &= a_3 \bullet b_3
\end{aligned}
$$

Il passaggio finale consiste nell'esecuzione dell'operazione di modulo:

$$d(x) = c(x) \bmod (x^4 + 1)$$

Quindi $d(x)$ deve soddisfare l'equazione:

$$c(x) = [(x^4 + 1) \times q(x)] \oplus d(x)$$

tale che il grado di $d(x)$ sia 3 o inferiore.

Una tecnica pratica per eseguire la moltiplicazione su questo anello di polinomi si basa sull'osservazione che:

$$x^i \bmod (x^4 + 1) = x^{i \bmod 4} \tag{5.11}$$

Se si combinano le Equazioni 5.10 e 5.11 si ha che:

$$
\begin{aligned}
d(x) &= c(x) \bmod (x^4 + 1) = [c_6 x^6 + c_5 x^5 + c_4 x^4 + c_3 x^3 + c_2 x^2 + c_1 x + c_0] \bmod (x^4 + 1) \\
&= c_3 x^3 + (c_2 \oplus c_6) x^2 + (c_1 \oplus c_5) x + (c_0 \oplus c_4)
\end{aligned}
$$

Espandendo i coefficienti c_i si ottengono le seguenti equazioni per i coefficienti di $d(x)$:

$$
\begin{aligned}
d_0 &= (a_0 \bullet b_0) \oplus (a_3 \bullet b_1) \oplus (a_2 \bullet b_2) \oplus (a_1 \bullet b_3) \\
d_1 &= (a_1 \bullet b_0) \oplus (a_0 \bullet b_1) \oplus (a_3 \bullet b_2) \oplus (a_2 \bullet b_3) \\
d_2 &= (a_2 \bullet b_0) \oplus (a_1 \bullet b_1) \oplus (a_0 \bullet b_2) \oplus (a_3 \bullet b_3) \\
d_3 &= (a_3 \bullet b_0) \oplus (a_2 \bullet b_1) \oplus (a_1 \bullet b_2) \oplus (a_0 \bullet b_3)
\end{aligned}
$$

Ciò può essere scritto in forma matriciale nel seguente modo:

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \tag{5.12}$$

La trasformazione MixColumns

Nella discussione sulla trasformazione MixColumns si è affermato che vi sono due modi equivalenti per definire la trasformazione. Il primo è la moltiplicazione di matrici rappresentata nell'Equazione 5.3, ripetuta di seguito:

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

Il secondo metodo consiste nel trattare ciascuna colonna di **State** come un polinomio di quattro termini con i coefficienti in GF(2^8). Ciascuna colonna viene moltiplicata modulo $(x^4 + 1)$ per il polinomio fisso $a(x)$, dato da:

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$$

Dall'Equazione 5.8, si ha a_3 $\{03\}$; $a_2 = \{01\}$; $a_1 = \{01\}$; $a0 = \{02\}$. Per la j-esima colonna di **State** si ha il polinomio $\text{col}_j(x) = s_{3,j} x^3 + s_{2,j} x^2 + s_{1,j} x + s_{0,j}$. Sostituendo nell'Equazione 5.12 si può esprimere $d(x) = a(x) \times \text{col}_j(x)$ come:

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} s_{0,j} \\ s_{1,j} \\ s_{2,j} \\ s_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,j} \\ s_{1,j} \\ s_{2,j} \\ s_{3,j} \end{bmatrix}$$

che è equivalente all'Equazione 5.3.

Moltiplicazione per x

Si consideri la moltiplicazione di un polinomio nell'anello per x: $c(x) = x \otimes b(x)$. Si ha:

$$
\begin{aligned}
c(x) = x \otimes b(x) &= [x \times (b_3 x^3 + b_2 x^2 + b_1 x^1 + b_1 x + b_0)] \bmod (x^4 + 1) \\
&= (b_3 x^4 + b_2 x^3 + b_1 x^2 + b_0 x) \bmod (x^4 + 1) \\
&= b_2 x^3 + b_1 x^2 + b_0 x + b_3
\end{aligned}
$$

Pertanto la moltiplicazione per x corrisponde a uno scorrimento circolare a sinistra di un byte dei quattro byte della word che rappresenta il polinomio. Se si rappresenta il polinomio come un vettore colonna di 4 byte, si avrà:

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 00 & 00 & 00 & 01 \\ 01 & 00 & 00 & 00 \\ 00 & 01 & 00 & 00 \\ 00 & 00 & 01 & 00 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Appendice 5.B AES Semplificato

L'algoritmo AES Semplificato (S-AES) è stato sviluppato dal Professor Edward Schaefer dell'Università di Santa Clara con alcuni suoi studenti [MUSA03]. È un algoritmo a scopo educativo, piuttosto che un algoritmo di crittografia sicuro. Ha proprietà e struttura simili ad AES ma con parametri molto più piccoli. Il lettore potrà trovare utile sviluppare un esempio a mano, mentre segue la discussione in questa appendice. Una buona comprensione di S-AES faciliterà la comprensione della struttura e del funzionamento di AES.

Breve panoramica

La Figura 5.8 illustra la struttura generale di S-AES.

L'algoritmo di cifratura riceve come input un blocco di 16 bit di testo in chiaro e una chiave a 16 bit, per produrre un blocco di 16 bit di testo cifrato come output. L'algoritmo di decifratura S-AES riceve come input il blocco di 16 bit di testo cifrato e la medesima chiave a 16 bit utilizzata nella cifratura, per produrre il blocco originale di 16 bit di testo in chiaro.

L'algoritmo di cifratura comporta l'impiego di quattro funzioni differenti, o trasformazioni: *add key* (A_K), *nibble substitution* (NS), *shift row* (SR) e *mix columns* (MC), le cui operazioni saranno successivamente chiarite.

È possibile esprimere concisamente l'algoritmo di cifratura come composizione[7] di funzioni:

$$A_{K_2} \circ SR \circ NS \circ A_{K_1} \circ MC \circ SR \circ NS \circ A_{K_0}$$

in modo tale che A_{K_0} sia applicata per prima.

L'algoritmo di cifratura è organizzato in tre fasi. La fase 0 consiste semplicemente nella funzione add key, la fase 1 comprende tutte e quattro le funzioni e la fase 2 contiene solo tre funzioni. La funzione add key, che utilizza una chiave di 16 bit, è inclusa in ogni fase. La chiave iniziale di 16 bit viene espansa a 48 bit, in modo tale che ogni fase utilizza una chiave di fase di 16 bit distinta.

Ogni funzione opera su uno stato di 16 bit, trattato come matrice 2 x 2 di gruppi di 4 bit (*nibble*). Il valore iniziale di tale matrice di stato sono i 16 bit di testo in chiaro; la matrice di stato viene poi modificata da ciascuna funzione nel processo di cifratura, producendo, dopo l'applicazione dell'ultima funzione, i 16 bit di testo cifrato. Come illustrato nella Figura 5.9a, i nibble nella matrice sono ordinati per colonna. Per esempio, i primi otto bit dei 16 bit di testo in chiaro in input occupano la prima colonna della matrice, mentre i secondi otto bit occupano la seconda colonna. La chiave di 16 bit è organizzata in modo analogo, ma risulta più conveniente vedere la chiave come due byte anziché quattro nibble (Figura 5.9b). La chiave espansa di 48 bit viene considerata come tre chiavi di fase, i cui bit sono etichettati nel modo seguente: $K_0 = k_0 \ldots k_{15}$; $K_1 = k_{16} \ldots k_{31}$; $K_2 = k_{32} \ldots k_{47}$

La Figura 5.10 illustra gli elementi essenziali di una fase completa di S-AES.

[7] Definizione: date due funzioni f e g, la funzione F definita dall'equazione $y = F(x) = g(f(x))$ è chiamata funzione composta di f e g ed è indicata con $F = g \circ f$.

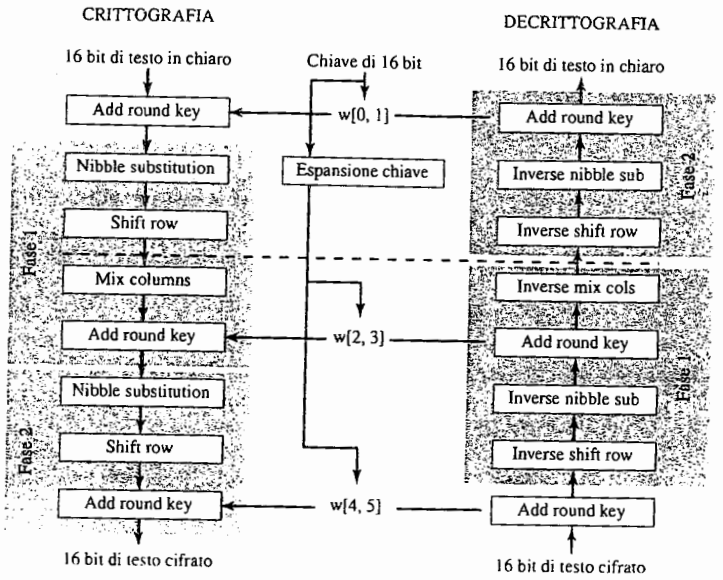


Figura 5.8 Crittografia e decrittografia con S-AES.

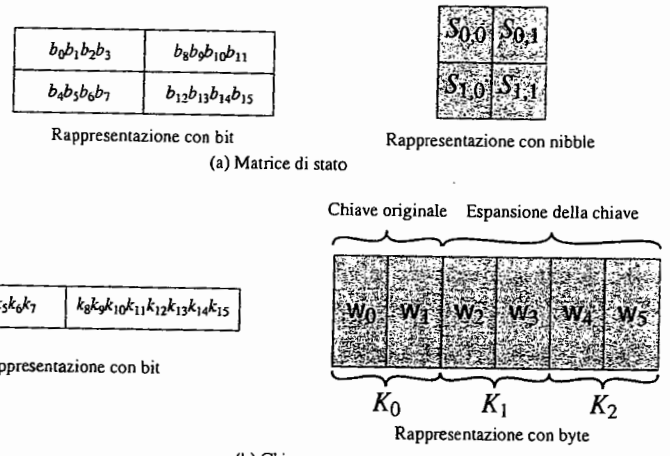


Figura 5.9 Strutture dati di S-AES.

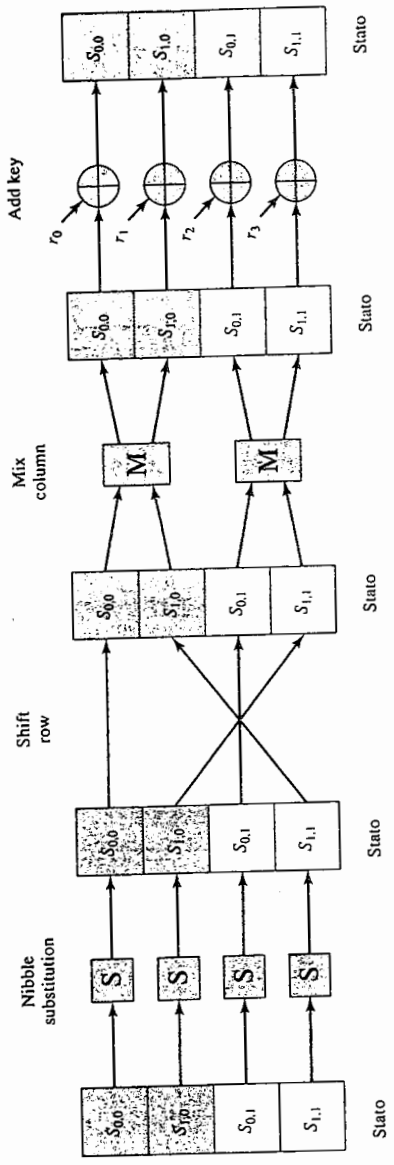


Figura 5.10 Fase di cifratura di S-AES.

La decifratura è illustrata nella Figura 5.8 ed è essenzialmente l'inverso della cifratura,

$$A_{K_0} \circ INS \circ ISR \circ IMC \circ A_{K_1} \circ INS \circ ISR \circ A_{K_2}$$

dove tre delle funzioni hanno una funzione corrispondente inversa: *nibble substitution inversa* (INS), *shift row inversa* (ISR) e *mix columns inversa* (IMC).

Cifratura e decifratura S-AES

Si esaminino ora le singole funzioni che fanno parte dell'algoritmo di cifratura.

AddKey

La funzione add key consiste nello XOR bit-a-bit della matrice di stato a 16 bit con la chiave di fase di 16 bit. La Figura 5.11 illustra il processo come operazione per colonne, ma può essere visto anche come operazione bit-a-bit o per nibble. Per esempio:

A	4	⊕	2	5	=	8	1
7	9		D	5		A	C

matrice di stato — chiave

L'inversa della funzione add key è identica alla funzione add key stessa, dato che l'operazione di XOR è l'inversa di se stessa.

Nibble substitution

La funzione nibble substitution è una semplice lookup in tabella (Figura 5.11). AES definisce una matrice 4 x 4 di valori nibble, chiamata S-box (Tabella 5.5a), che contiene una permutazione di tutti i possibili valori a 4 bit. I nibble della matrice di stato vengono univocamente mappati in nuovi nibble secondo lo schema seguente: i due bit più a sinistra del nibble rappresentano la riga, e i due bit più a destra la colonna. Questi due valori vengono utilizzati come indici nella S-box per selezionare un valore di quattro bit univoco. Per esempio, il valore esadecimale A fa riferimento alla seconda riga e alla seconda colonna della S-box, che contiene il valore zero.

Ecco un esempio della trasformazione nibble substitution:

8	1	→	6	4
A	C		0	C

La funzione nibble substitution inversa utilizza la S-box inversa riportata nella Tabella 5.5b. Si noti, per esempio, che l'input 0 produce l'output A, e che l'input A alla S-box produce 0.

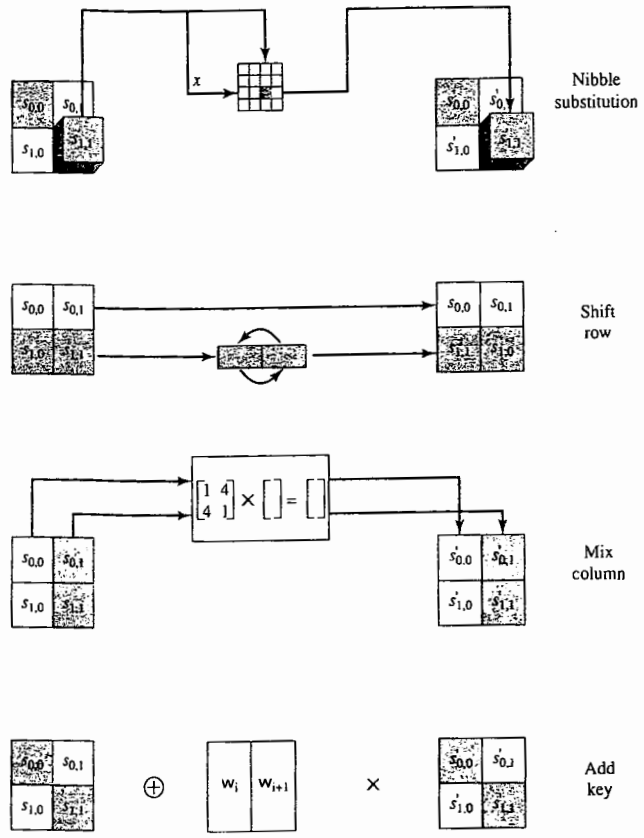
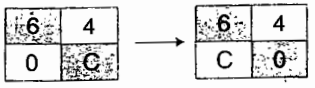


Figura 5.11 Le trasformazioni in S-AES.

Shift row

La funzione shift row effettua lo scorrimento circolare a sinistra di un nibble nella seconda riga della matrice di stato; la prima riga rimane inalterata (Figura 5.11). Ecco un esempio:



La funzione shift row inversa è identica alla funzione shift row stessa, dato che un nuovo scorrimento riporta la seconda riga al valore originale.

Mix Column

La funzione mix column opera su ciascuna colonna individualmente. Ciascun nibble di una colonna viene mappato in un nuovo valore che dipende da entrambi i nibble nella colonna. La trasformazione può essere definita dalla seguente moltiplicazione della matrice di stato (Figura 5.11):

Eseguendo la moltiplicazione si ottiene:

$$S'_{0,0} = S_{0,0} \oplus (4 \cdot S_{1,0})$$
$$S'_{1,0} = (4 \cdot S_{0,0}) \oplus S_{1,0}$$
$$S'_{0,1} = S_{0,1} \oplus (4 \cdot S_{1,1})$$
$$S'_{1,1} = (4 \cdot S_{0,1}) \oplus S_{1,1}$$

dove l'aritmetica è calcolata in GF (2^4) e il simbolo · indica la moltiplicazione in GF (2^4). L'Appendice E fornisce le tabelle di somma e moltiplicazione. Ecco un esempio:

$$\begin{bmatrix} 1 & 4 \\ 4 & 1 \end{bmatrix} \begin{bmatrix} 6 & 4 \\ C & 0 \end{bmatrix} = \begin{bmatrix} 3 & 4 \\ 7 & 3 \end{bmatrix}$$

La funzione mix columns inversa è definita nel modo seguente:

$$\begin{bmatrix} 9 & 2 \\ 2 & 9 \end{bmatrix} \begin{bmatrix} S_{0,0} & S_{0,1} \\ S_{1,0} & S_{1,1} \end{bmatrix} = \begin{bmatrix} S'_{0,0} & S'_{0,1} \\ S'_{1,0} & S'_{1,1} \end{bmatrix}$$

Si può facilmente dimostrare che tale definizione corrisponde all'inversa della trasformazione:

$$\begin{bmatrix} 9 & 2 \\ 2 & 9 \end{bmatrix} \begin{bmatrix} 1 & 4 \\ 4 & 1 \end{bmatrix} \begin{bmatrix} S_{0,0} & S_{0,1} \\ S_{1,0} & S_{1,1} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} S_{0,0} & S_{0,1} \\ S_{1,0} & S_{1,1} \end{bmatrix} = \begin{bmatrix} S_{0,0} & S_{0,1} \\ S_{1,0} & S_{1,1} \end{bmatrix}$$

La moltiplicazione precedente utilizza i seguenti risultati in GF(2^4): $9 + (2 \bullet 4) = 9 + 8 = 1$; $(9 \bullet 4) + 2 = 2 + 2 = 0$. Queste operazioni possono essere verificate utilizzando le tabelle aritmetiche nell'Appendice E o con l'aritmetica polinomiale.

La funzione mix columns è la più difficile da visualizzare. Per questo motivo si fornisce un approfondimento nell'Appendice E.

Espansione della chiave

Ai fini dell'espansione della chiave, i 16 bit della chiave iniziale vengono raggruppati in una riga di due parole di 8 bit. La Figura 5.12 illustra l'espansione in 6 parole, tramite il calcolo di 4 nuove parole a partire dalle due iniziali.

L'algoritmo è il seguente:

$$w_2 = w_0 \oplus g(w_1) = w_0 \oplus RCON(1) \oplus SubNib(RotNib(w_1))$$
$$w_3 = w_2 \oplus w_1$$
$$w_4 = w_2 \oplus g(w_3) = w_2 \oplus RCON(2) \oplus SubNib(RotNib(w_3))$$
$$w_5 = w_4 \oplus w_3$$

RCON è una costante di fase, definita come: $RC[i] = x^{i+2}$ in modo che $RC[1] = x^3 = 1000$ e $RC[2] = x^4 \bmod (x^4 + x + 1) = 0011$. $RC[i]$ costituisce il nibble più a sinistra di un byte, mentre il nibble più a destra è costituito da tutti 0. Di conseguenza si ha $RCON(1) = 10000000$ e $RCON(2) = 00110000$.

Si supponga, per esempio, che la chiave sia 2D55 = 0010 1101 0101 0101 = $w_0 w_1$. Quindi:

$w_2 = 00101101 \oplus 10000000 \oplus SubNib(01010101) = 00101101 \oplus 10000000 \oplus 00010001 = 10111100$
$w_3 = 10111100 \oplus 01010101 = 11101001$
$w_4 = 10111100 \oplus 00110000 \oplus SubNib(10011110) = 10111100 \oplus 00110000 \oplus 00101111 = 10100011$
$w_5 = 10100011 \oplus 11101001 = 01001010$

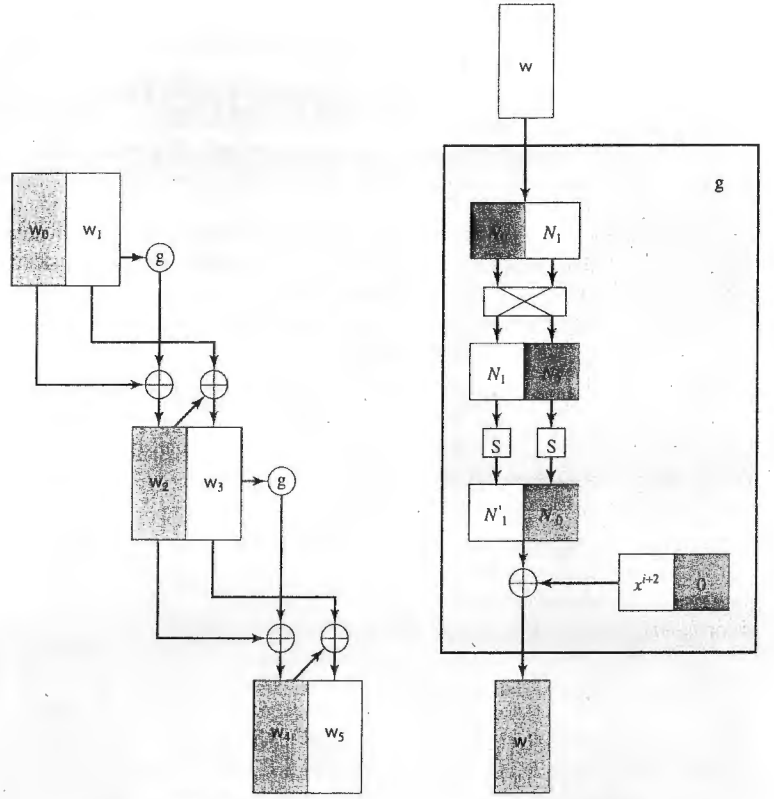
La S-box

La S-box viene costruita nel modo seguente:

1. Inizializzare la S-box con i valori dei nibble in ordine crescente riga per riga. La prima riga contiene i valori esadecimali 0, 1, 2, 3; la seconda riga contiene 4, 5, 6, 7 ecc. Il valore del nibble nella riga i, colonna j vale dunque $4i + j$.
2. Trattare ciascun nibble come un elemento del campo finito GF(2^4) modulo $x^4 + x + 1$. Ogni nibble $a_1a_2a_3a_4$ rappresenta un polinomio di grado 3.
3. Mappare ciascun byte della S-box al suo inverso moltiplicativo nel campo finito GF(2^4) modulo $x^4 + x + 1$; il valore 0 viene mappato su se stesso.
4. Considerare ogni byte nella S-box come una sequenza di 4 bit (b_0, b_1, b_2, b_3) Applicare la seguente trasformazione a ogni bit di ciascun byte della S-box; lo standard AES illustra questa trasformazione in forma matriciale:

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

L'apice indica che la variabile deve essere aggiornata con il valore sulla destra. Si ricordi che la somma e la moltiplicazione sono calcolate modulo 2. La Tabella 5.5a riporta la S-box risultante: una matrice non lineare invertibile. La S-box inversa è riportata nella Tabella 5.5b.



(a) Algoritmo generale (b) Funzione g

Figura 5.12 Espansione della chiave in S-AES.

La struttura di S-AES

È ora possibile esaminare alcuni aspetti interessanti relativi alla struttura di AES. Per prima cosa si osservi che gli algoritmi di cifratura e decifratura iniziano e terminano con la funzione add key. Qualsiasi altra funzione, all'inizio o al termine, è facilmente invertibile senza conoscere la chiave: non aumenterebbe dunque la sicurezza ma solo il carico computazionale. Per questo motivo vi è una fase 0 costituita esclusivamente dalla funzione add key.

Il secondo aspetto da osservare è che la fase 2 non comprende la funzione mix columns. Il motivo è legato a una terza osservazione: sebbene l'algoritmo di decifratura sia l'inverso dell'algoritmo di cifratura, come chiaramente illustrato nella Figura 5.8, esso non segue la medesima sequenza di funzioni. Infatti:

cifratura: $A_{K_2} \circ SR \circ NS \circ A_{K_1} \circ MC \circ SR \circ NS \circ A_{K_0}$

decifratura: $A_{K_0} \circ INS \circ ISR \circ IMC \circ A_{K_1} \circ INS \circ ISR \circ A_{K_2}$

Dal punto di vista dell'implementazione sarebbe preferibile che la sequenza di funzioni nella decifratura fosse identica alla sequenza di funzioni nella cifratura. Questo consentirebbe di implementare l'algoritmo di decifratura come l'algoritmo di cifratura, ottenendo un'efficienza superiore.

Si noti che se fosse possibile scambiare le funzioni seconda e terza, quarta e quinta, e sesta e settima nella sequenza di decifratura, si otterrebbe la medesima struttura dell'algoritmo di cifratura. È questo realmente possibile? Per prima cosa si consideri lo scambio di INS e ISR. Dato uno stato N costituito dai nibble (N_0, N_1, N_2, N_3), la trasformazione $INS(ISR(N))$ procede nel modo seguente:

$$\begin{pmatrix} N_0 & N_2 \\ N_1 & N_3 \end{pmatrix} \rightarrow \begin{pmatrix} N_0 & N_2 \\ N_3 & N_1 \end{pmatrix} \rightarrow \begin{pmatrix} IS[N_0] & IS[N_2] \\ IS[N_3] & IS[N_1] \end{pmatrix}$$

dove IS si riferisce alla S-box inversa. Invertendo le operazioni, la trasformazione $ISR(INS(N))$ procede nel modo seguente:

$$\begin{pmatrix} N_0 & N_2 \\ N_1 & N_3 \end{pmatrix} \rightarrow \begin{pmatrix} IS[N_0] & IS[N_2] \\ IS[N_1] & IS[N_3] \end{pmatrix} \rightarrow \begin{pmatrix} IS[N_0] & IS[N_2] \\ IS[N_3] & IS[N_1] \end{pmatrix}$$

I due risultati sono identici, dunque INS (ISR(N)) = ISR(INS(N)).

Si consideri ora l'operazione mix columns inversa seguita dalla add key: $IMC(A_{K_1}(N))$, dove la chiave di fase K_1 è costituita dai nibble $(k_{0,0}, k_{1,0}, k_{0,1}, k_{1,1})$. Si ottiene:

$$\begin{pmatrix} 9 & 2 \\ 2 & 9 \end{pmatrix}\left(\begin{pmatrix} k_{0,0} & k_{0,1} \\ k_{1,0} & k_{1,1} \end{pmatrix} \oplus \begin{pmatrix} N_0 & N_2 \\ N_1 & N_3 \end{pmatrix}\right) = \begin{pmatrix} 9 & 2 \\ 2 & 9 \end{pmatrix}\begin{pmatrix} k_{0,0} \oplus N_0 & k_{0,1} \oplus N_2 \\ k_{1,0} \oplus N1 & k_{1,1} \oplus N3 \end{pmatrix}$$

$$= \begin{pmatrix} 9(k_{0,0} \oplus N_0) \oplus 2(K_{1,0} \oplus N_1) & 9(k_{0,1} \oplus N_2) \oplus 2(K_{1,1} \oplus N_3) \\ 2(k_{0,0} \oplus N_0) \oplus 9(K_{1,0} \oplus N_1) & 2(k_{0,1} \oplus N_2) \oplus 9(K_{1,1} \oplus N_3) \end{pmatrix}$$

$$= \begin{pmatrix} (9k_{0,0} \oplus 2k_{1,0}) \oplus (9N_0 \oplus 2N_1) & (9k_{0,1} \oplus 2k_{1,1}) \oplus (9N_2 \oplus 2N_3) \\ (2k_{0,0} \oplus 9k_{1,0}) \oplus (2N_0 \oplus 9N_1) & (2k_{0,1} \oplus 9k_{1,1}) \oplus (2N_2 \oplus 9N_3) \end{pmatrix}$$

$$= \begin{pmatrix} (9k_{0,0} \oplus 2k_{1,0}) & (9k_{0,1} \oplus 2k_{1,1}) \\ (2k_{0,0} \oplus 9k_{1,0}) & (2k_{0,1} \oplus 9k_{1,1}) \end{pmatrix} \oplus \begin{pmatrix} (9N_0 \oplus 2N_1) & (9N_2 \oplus 2N_3) \\ (2N_0 \oplus 9N_1) & (2N_2 \oplus 9N_3) \end{pmatrix}$$

$$= \begin{pmatrix} 9 & 2 \\ 2 & 9 \end{pmatrix}\begin{pmatrix} k_{0,0} & k_{0,1} \\ k_{1,0} & k_{1,1} \end{pmatrix} \oplus \begin{pmatrix} 9 & 2 \\ 2 & 9 \end{pmatrix}\begin{pmatrix} N_0 & N_2 \\ N_1 & N_3 \end{pmatrix}$$

Tutti i passi precedenti sfruttano le proprietà dell'aritmetica dei campi finiti. Come risultato si ottiene che: $IMC(A_{K_1}(N)) = IMC(K_1) \oplus IMC(N)$. Sia ora $IMC(K_1)$ la funzione inversa round key della fase 1 e sia la trasformazione add key inversa IA_{K_1} lo XOR bit-a-bit della funzione inversa round key con il vettore di stato. In questo caso si ottiene: $IMC(A_{K_1}(N)) = IA_{K_1}(IMC(N))$. Si può quindi scrivere:

cifratura: $A_{K_2} \circ SR \circ NS \circ A_{K_1} \circ MC \circ SR \circ NS \circ A_{K_0}$

decifratura: $A_{K_0} \circ INS \circ ISR \circ IMC \circ A_{K_1} \circ INS \circ ISR \circ A_{K_2}$

decifratura: $A_{K_0} \circ ISR \circ INS \circ A_{IMC(K_1)} \circ IMC \circ ISR \circ INS \circ A_{K_2}$

Entrambe la cifratura e la decifratura seguono ora la medesima sequenza. Si noti che questa derivazione non sarebbe così efficace se la fase 2 dell'algoritmo di cifratura includesse la funzione MC. In quel caso si avrebbe:

cifratura: $A_{K_2} \circ MC \circ SR \circ NS \circ A_{K_1} \circ MC \circ SR \circ NS \circ A_{K_0}$

decifratura: $A_{K_0} \circ INS \circ ISR \circ IMC \circ A_{K_1} \circ INS \circ ISR \circ IMC \circ A_{K_2}$

Non sarebbe dunque possibile scambiare coppie di operazioni nell'algoritmo di decifratura per ottenere la medesima struttura dell'algoritmo di cifratura.

Capitolo 6

Approfondimenti sulla cifratura simmetrica

Concetti essenziali

- La **crittografia multipla** è una tecnica con la quale lo stesso algoritmo di crittografia viene applicato più volte. La prima volta, il testo in chiaro viene convertito in testo cifrato utilizzando l'algoritmo di crittografia. Il testo cifrato così ottenuto è quindi utilizzato come input per una nuova applicazione dell'algoritmo. Questo processo può essere ripetuto per il numero di volte desiderato.
- L'algoritmo triple DES utilizza tre volte il DES, con due o tre chiavi distinte.
- Per modalità operativa si intende una tecnica per ottimizzare l'efficacia di un algoritmo di crittografia o per adattarlo a una particolare situazione, quale, per esempio, l'applicazione di una cifratura a blocchi a una sequenza di blocchi di dati o a un flusso.
- Sono state standardizzate cinque modalità operative per l'utilizzo con cifrature a blocchi simmetriche quali DES e AES: electronic codebook, cipher block chaining, cipher feedback, output feedback e counter.
- Un **cifrario a flussi** è un algoritmo di crittografia simmetrico nel quale l'output di testo cifrato viene prodotto bit-per-bit o byte-per-byte da un flusso di testo in chiaro in ingresso. La cifratura più utilizzata di questo tipo è RC4.

Il capitolo prosegue l'analisi delle cifrature simmetriche. Il primo argomento trattato è la crittografia multipla, dove si esamina in particolare il Triple DES, lo schema più diffuso di cifratura multipla.

Si affrontano successivamente le modalità operative della cifratura a blocchi: esistono varie modalità alternative per applicare la cifratura a blocchi al testo in chiaro, ciascuna con i propri particolari vantaggi che la rendono adatta a particolari applicazioni.

Il capitolo si conclude esaminando la cifratura simmetrica a flussi, che differisce in modo significativo dalle cifrature simmetriche a blocchi. Si analizza anche RC4, la cifratura più importante di questa classe.

6.1 Crittografia multipla e Triple DES

Data la potenziale vulnerabilità di DES ad attacchi a forza bruta, vi è stato un grande interesse nella ricerca di alternative. Un approccio consiste nel progettare un algoritmo completamente nuovo: AES rappresenta un esempio fondamentale. Un'alternativa, che presenta il vantaggio di conservare l'investimento esistente in termini di software e dispositivi, consiste nell'utilizzare una crittografia DES multipla con più chiavi. Si partirà esaminando l'esempio più semplice di questa alternativa. Poi si vedrà l'approccio più diffuso rappresentato da Triple DES (3DES).

Double DES

La forma più semplice di crittografia multipla prevede due stadi di crittografia con due chiavi (Figura 6.1A). Dato un testo in chiaro P e due chiavi di crittografia K_1 e K_2, il testo cifrato viene generato nel seguente modo:

$$C = E(K_2, E(K_1, P))$$

La decrittografia richiede l'applicazione delle chiavi in ordine inverso:

$$P = D(K_1, D(K_2, C))$$

Per DES, questo schema prevede apparentemente una chiave della lunghezza di $56 \times 2 = 112$ bit con un notevole incremento della potenza crittografica. Ma occorre esaminare l'algoritmo in modo più specifico.

Riduzione a un'unica fase

Si supponga che in DES sia vero che, per tutte le chiavi a 56 bit, date due chiavi K_1 e K_2 sia possibile trovare una chiave K_3 tale che:

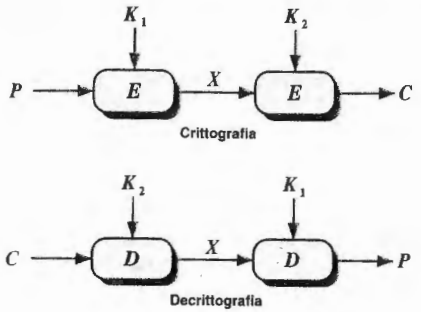
$$E(K_2, E(K_1, P)) = E(K_3, P) \qquad (6.1)$$

In questo caso, la doppia crittografia (e in realtà qualsiasi numero di stadi di crittografia multipla DES) sarebbe inutile poiché il risultato sarebbe equivalente a un'unica crittografia con una chiave da 56 bit.

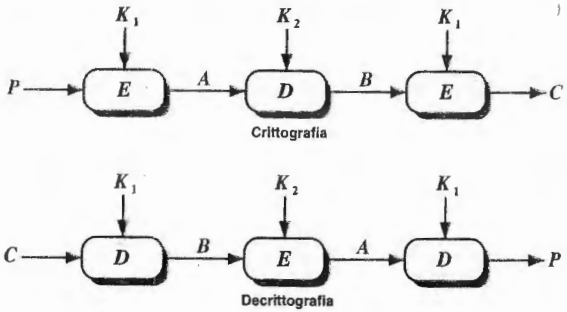
Non sembra che l'Equazione 6.1 sia valida. Si consideri che la crittografia con DES è un mapping di blocchi da 64 bit in blocchi da 64 bit. In effetti il mapping può essere considerato come una permutazione; ovvero se si considerano tutti i 2^{64} blocchi di input possibili, la crittografia DES con una determinata chiave mappa ciascun blocco in un blocco univoco da 64 bit. Se invece, per esempio, due blocchi di input venissero mappati sullo stesso blocco di output, sarebbe impossibile ottenere il testo in chiaro originario. Con 2^{64} input possibili, quanti mapping differenti esistono che generano una permutazione dei blocchi di input? Il valore è facile da calcolare:

$$(2^{64})! = 10^{3473800000000000000} > (10^{10^{20}})$$

D'altra parte, DES definisce un mapping per ciascuna chiave, per un numero totale di mapping pari a:



(A) Crittografia doppia



(B) Crittografia tripla

Figura 6.1 Crittografia multipla.

$$2^{56} < 10^{17}$$

Pertanto è ragionevole presupporre che se DES venisse applicato due volte con due chiavi diverse, produrrebbe uno dei tanti mapping non definiti da un'unica applicazione di DES. Sebbene questa affermazione abbia a supporto numerose prove, è stata dimostrata [CAMP92] solo nel 1992.

Attacco Meet-in-the-Middle

Pertanto l'uso di Double DES produce un mapping che non è equivalente a un'unica crittografia DES. Ma vi è un modo per attaccare questo schema, che non dipende da alcuna proprietà specifica di DES ma che può essere adottato contro qualsiasi cifratura a blocchi.

L'algoritmo, chiamato Meet-in-the-Middle, è stato descritto per la prima volta in [DIFF77]. Si basa sull'osservazione che, se:

$$C = E(K_2, E(K_1, P))$$

allora (vedere la Figura 6.1A):

$$X = E(K_1, P) = D(K_2, P)$$

Data una coppia nota, (P, C), l'attacco procede nel modo seguente. Innanzitutto si esegue la crittografia di P per tutti i 2^{56} valori possibili di K_1. Si memorizzano i risultati in una tabella e la si ordina per i valori di X. Si esegue poi la decrittografia di C utilizzando tutti i 2^{56} possibili valori di K_2. A mano a mano che vengono prodotti i risultati, si confrontano con la tabella, alla ricerca di una corrispondenza. Quando viene trovata una corrispondenza, si esegue il test delle due chiavi risultanti con una nuova coppia nota di testo in chiaro/testo cifrato. Se le due chiavi producono il testo cifrato corretto, si accettano le chiavi come corrette.

Per un determinato testo in chiaro P, esistono 2^{64} possibili valori cifrati che potrebbero essere prodotti da una doppia applicazione dell'algoritmo Double DES. L'algoritmo Double DES utilizza in pratica una chiave da 112 bit e dunque possono esistere 2^{112} possibili chiavi. Pertanto, in media, per un determinato testo in chiaro P, il numero di chiavi da 112 bit che produrranno un determinato testo cifrato C è $2^{112}/2^{64} = 2^{48}$. Pertanto la procedura descritta in precedenza produrrà circa 2^{48} falsi allarmi sulla prima coppia (P, C). Un argomento analogo indica che con altri 64 bit di testo in chiaro e con il corrispondente testo cifrato, i falsi allarmi si riducono a $2^{48-64} = 2^{-16}$. In altre parole, se l'attacco Meet-in-the-Middle viene eseguito su due blocchi di testo in chiaro/testo cifrato noti, la probabilità di determinare le chiavi corrette è $1 - 2^{-16}$. Il risultato è che l'attacco avrebbe successo contro Double DES con chiavi delle dimensioni di 112 bit, con un impegno dell'ordine di 2^{56}, non molto oltre 2^{55} necessario per l'algoritmo DES.

Triple DES con due chiavi

Una contromisura ovvia contro l'attacco Meet-in-the-Middle consiste nell'impiegare tre fasi di crittografia con tre diverse chiavi. Questo aumenta il costo dell'attacco con testo in chiaro noto a 2^{112}, decisamente oltre i limiti pratici prevedibili oggi e nel lontano futuro. Tuttavia si presenta il problema di dover usare chiavi della lunghezza di $56 \times 3 = 168$ bit.

Un'alternativa proposta da Tuchman [TUCH79] è una tripla crittografia con due sole chiavi. La funzione esegue una sequenza crittografia-decrittografia-crittografia (EDE – Encrypt-Decrypt-Encrypt) come indicato nella Figura 6.1B:

$$C = E(K_1, D(K_2, E(K_1, P)))$$

La decrittografia della seconda fase non è particolarmente significativa in termini di sicurezza: consente semplicemente agli utenti di 3DES di decrittografare i dati crittografati dagli utenti che hanno utilizzato la crittografia DES semplice:

$$C = E(K_1, D(K_1, E(K_1, P))) = E(K_1, P)$$

3DES con due chiavi rappresenta un'alternativa interessante a DES ed è stata adottata per gli standard di gestione della chiave ANS X9.17 e ISO 8732.[1]

[1] (ANS) American National Standard. Il nome dello standard, X9.17, (gestione delle chiavi per istituti finanziari) sembra un po' misterioso ma in realtà, come si vedrà nei prossimi capitoli di questo volume, varie tecniche specificate in questo standard sono state adottate nell'ambito di altri standard o applicazioni.

Attualmente non esiste un attacco ad analisi crittografica veramente efficace contro 3DES. Coppersmith [COPP94] nota che il costo di un attacco a forza bruta per la ricerca della chiave di 3DES è dell'ordine di $2^{112} \approx (5 \times 10^{33})$ e stima che il costo dell'analisi crittografica differenziale subisce una crescita esponenziale, rispetto a DES, superiore a 10^{52}.

Vale la pena di parlare di alcuni attacchi proposti contro 3DES che, sebbene non costituiscono in realtà una minaccia realistica, danno un'idea del tipo di attacchi che sono stati considerati e che potranno essere sviluppati nel futuro.

La prima proposta seria proviene da Merkle e Hellman [MERK81]. Il loro piano prevede la ricerca di valori di testo in chiaro che producono un primo valore intermedio di $A = 0$ (Figura 6.1B) e poi nell'utilizzo dell'attacco Meet-in-the-Middle per determinare le due chiavi. Lo sforzo richiesto è dell'ordine di 2^{56} ma la tecnica richiede 2^{56} coppie di testo in chiaro e testo cifrato, un volume che difficilmente verrà fornito dal possessore delle chiavi.

Un attacco a testo in chiaro noto viene invece presentato in [VANO90]. Questo metodo è un miglioramento rispetto all'approccio a testo in chiaro scelto ma richiede uno sforzo maggiore. L'attacco si basa sull'osservazione che se si conoscono A e C (Figura 6.1B), il problema si riduce a un attacco alla crittografia Double DES. Naturalmente, chi svolge l'attacco non conosce A anche se sono noti P e C, sempre che le due chiavi siano sconosciute. Tuttavia chi svolge l'attacco può scegliere un valore potenziale di A e quindi tentare di trovare una coppia nota (P, C) che produce A. L'attacco procede nel modo seguente.

1. Ottenere n coppie (P, C). Questo è il testo in chiaro noto. Inserire queste coppie in una tabella (Tabella 1) ordinata in base ai valori di P (Figura 6.2B).
2. Scegliere un valore arbitrario di A chiamato a e creare una seconda tabella (Figura 6.2C) le cui voci sono definite nel seguente modo. Per ognuna delle 2^{56} possibili chiavi $K_1 = i$, calcolare il valore del testo in chiaro P_i che produce a:

   $$P_i = D(i, a)$$

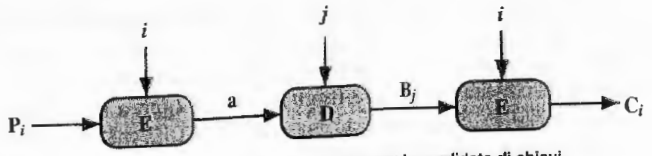
   Per ciascun P_i che corrisponde a una voce della Tabella 1, creare una voce nella Tabella 2 costituita dal valore di K_1 e dal valore di B prodotto per la coppia (P, C) della Tabella 1, supponendo che il valore di K_1 sia:

   $$B = D(i, C)$$

   Alla fine di questo passo, ordinare la Tabella 2 sulla base dei valori di B.
3. Ora nella Tabella 2 si ha una serie di valori candidati di K_1 e si ha la possibilità di ricercare un valore per K_2. Per ciascuna delle 2^{56} possibili chiavi $K_2 = j$, si deve calcolare il secondo valore intermedio per il valore scelto di a:

   $$B_j = D(j, a)$$

   In ciascuna fase, si deve ricercare B_j nella Tabella 2. Se si trova una corrispondenza, allora la chiave corrispondente i della Tabella 2 più questo valore di j sono valori candidati per le chiavi (K_1, K_2). Perché? Perché si è trovata una coppia di chiavi (i, j) che produce una coppia nota (P, C) (vedere la Figura 6.2A).
4. Verificare ogni coppia candidata di chiavi (i, j) su poche altre coppie di testo in chiaro/testo cifrato. Se una coppia di chiavi produce il testo cifrato desiderato, l'operazione è terminata, altrimenti si deve ripetere la procedura dal passo 1 con un nuovo valore di a.



(A) Tripla crittografia a due chiavi con una coppia candidata di chiavi

(B) Tabella delle *n* coppie note di testo in chiaro/testo cifrato, ordinate in base a P

(C) Tabella dei valori intermedi e delle chiavi candidate

Figura 6.2 Attacco a testo in chiaro noto contro Triple DES.

Per valori P e C noti, la probabilità di selezionare il valore univoco corretto è pari a $1/2^{64}$. Pertanto, date n coppie (P, C), la probabilità di successo per un singolo valore selezionato di a è $n/2^{64}$. Un risultato fondamentale della teoria delle probabilità è il fatto che il numero previsto di tentativi necessari per estrarre una pallina rossa da un'urna contenente n palline rosse e $N - n$ palline verdi è $(N+1)/(n+1)$. Dunque il numero previsto di tentativi per a è, per elevati valori di n:

$$\frac{2^{64}+1}{n+1} \approx \frac{2^{64}}{n}$$

Pertanto, il tempo di esecuzione previsto per l'attacco è dell'ordine di:

$$(2^{56})\frac{2^{64}}{n} = 2^{120-\log_2 n}$$

Triple DES con tre chiavi

Sebbene l'attacco appena descritto sembri poco realistico dal punto di vista pratico, chiunque utilizzasse 3DES con due chiavi potrebbe preoccuparsi della sicurezza della soluzione. Molti ricercatori ritengono ora preferibile l'alternativa rappresentata da 3DES a tre chiavi (ad esempio [KALI96a]). L'algoritmo 3DES a tre chiavi usa in pratica una chiave della lunghezza di 168 bit definita nel seguente modo:

$$C = E(K_3, D(K_2, E(K_1, P)))$$

La compatibilità con DES è garantita ponendo $K_3 = K_2$ o $K_1 = K_2$. Molte applicazioni per Internet, fra cui PGP e S/MIME di cui si parlerà nel Capitolo 15, hanno adottato l'algoritmo 3DES a tre chiavi.

6.2 Modalità di funzionamento della cifratura a blocchi

Un algoritmo di cifratura a blocchi è un elemento di base della sicurezza dei dati. NIST (FIPS 81) ha definito quattro "modalità di funzionamento" per applicarlo nelle varie situazioni. Una modalità di funzionamento, od operativa, è sostanzialmente una tecnica per ottimizzare l'efficacia di un algoritmo di crittografia o per adattarlo a una particolare situazione, quale, per esempio, l'applicazione di una cifratura a blocchi a una sequenza di blocchi di dati o a un flusso. Queste quattro modalità hanno lo scopo di considerare praticamente tutte le possibili applicazioni della crittografia per cui può essere utilizzato un algoritmo di cifratura a blocchi. Con la comparsa di nuove applicazioni e nuovi requisiti, il NIST ha espanso questo elenco delle modalità di funzionamento a cinque, nel documento Special Publication 800-38A. Queste modalità possono essere utilizzate con qualsiasi cifratura simmetrica a blocchi fra cui triple DES e AES. Queste modalità sono riepilogate nella Tabella 6.1 e descritte brevemente nella parte restante del capitolo.

Tabella 6.1 Modalità di funzionamento della cifratura a blocchi.

Modalità	Descrizione	Tipica applicazione
Electronic Codebook (ECB)	Ciascun blocco di testo in chiaro di 64 bit viene codificato in modo indipendente utilizzando la stessa chiave.	• Trasmissione sicura di singoli valori (per esempio una chiave di crittografia).
Cipher Block Chaining (CBC)	All'input dell'algoritmo di crittografia viene applicato uno XOR dei successivi 64 bit del testo in chiaro e dei precedenti 64 bit del testo cifrato.	• Trasmissione di carattere generale orientata ai blocchi. • Autenticazione
Cipher Feedback (CFB)	L'input viene elaborato *J* bit alla volta. Come input dell'algoritmo di crittografia viene utilizzato il testo cifrato precedente in modo da produrre un output pseudocasuale al quale viene applicato uno XOR con il testo in chiaro per produrre la successiva unità di testo cifrato.	• Trasmissione di carattere generale orientata al flusso di dati. • Autenticazione

(segue)

Tabella 6.1 Modalità di funzionamento della cifratura a blocchi. *(continua)*

Modalità	Descrizione	Tipica applicazione
Output Feedback (OFB)	Simile alla modalità CFB, tranne per il fatto che l'input dell'algoritmo di crittografia è l'output DES precedente.	• Trasmissione orientata al flusso di dati su canali rumorosi (per esempio comunicazioni via satellite).
Counter (CTR)	A ciascun blocco di testo in chiaro viene applicato uno XOR con un contatore crittografato. Il contatore viene incrementato per ogni blocco successivo.	• Trasmissione di carattere generale orientata ai blocchi. • Utile per requisiti di alta velocità.

La modalità di funzionamento Electronic CodeBook

Si tratta della modalità più semplice, nella quale il testo in chiaro viene gestito un blocco alla volta e ciascun blocco di testo in chiaro viene crittografato utilizzando la stessa chiave (vedere la Figura 6.3). Si usa il termine *codebook* poiché, per una determinata chiave, vi è un unico testo cifrato per ogni blocco di testo in chiaro di b bit. Pertanto si può immaginare un enorme elenco di codici in cui vi è una sola voce di testo cifrato per ogni possibile sequenza di testo in chiaro di b bit.

Per un messaggio più lungo di b bit, la procedura è semplicemente quella di suddividere il messaggio in blocchi di b bit utilizzando, se necessario, bit di riempimento per l'ultimo blocco. La decrittografia viene eseguita un blocco alla volta, sempre utilizzando la stessa chiave. Nella Figura 6.3 il testo in chiaro (con eventuali riempimenti) è costituito da una sequenza di blocchi di b bit, $P_1, P_2, \ldots, P_N$; la sequenza corrispondente di blocchi di testo cifrato è $C_1, C_2, \ldots, C_N$.

Il metodo ECB è ideale per volumi limitati di dati, come per esempio una chiave di crittografia. Pertanto, se si vuole trasmettere una chiave DES in modo sicuro, è preferibile impiegare la modalità ECB.

La caratteristica più significativa di ECB consiste nel fatto che se nel messaggio compare più volte lo stesso blocco di b bit di testo in chiaro, verrà prodotto sempre lo stesso testo cifrato.

Per messaggi più lunghi, la modalità ECB può non essere sicura. Se il messaggio fosse molto strutturato, l'analisi crittografica potrebbe sfruttarne le regolarità. Per esempio, se si supesse che il messaggio inizia sempre con determinati campi predefiniti, l'analista crittografico potrebbe disporre di numerose coppie di testo in chiaro/testo cifrato su cui lavorare. Se il messaggio contenesse elementi ripetitivi, con un periodo di ripetizione multiplo di b bit, tali elementi potrebbero essere facilmente identificati dall'analista. Questo potrebbe facilitare l'analisi od offrire lo spunto per sostituire o disporre in modo differente i blocchi.

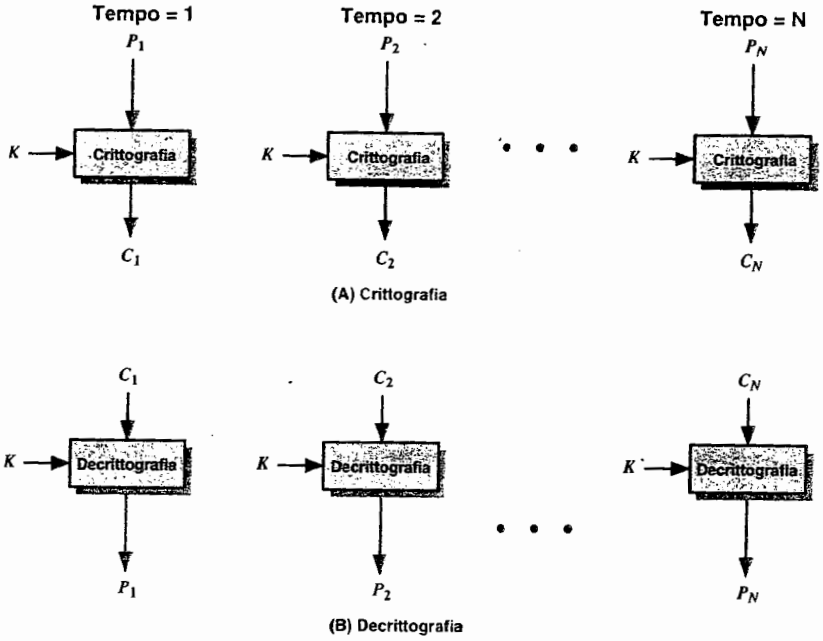


Figura 6.3 La modalità di funzionamento Electronic Codebook (ECB).

La modalità di funzionamento Cipher Block Chaining

Per superare i limiti di sicurezza di ECB, sarebbe auspicabile una tecnica in cui lo stesso blocco di testo in chiaro, se ripetuto, producesse blocchi di testo cifrato differenti. Un modo semplice per soddisfare questo requisito è rappresentato dalla modalità di funzionamento CBC (Cipher Block Chaining) rappresentata nella Figura 6.4. In questo schema, l'input dell'algoritmo di crittografia è il risultato dello XOR fra il blocco di testo in chiaro corrente e.il blocco di testo cifrato precedente; per ciascun blocco viene utilizzata la stessa chiave. In pratica si è concatenata l'elaborazione della sequenza di blocchi di testo in chiaro. L'input della funzione di crittografia per ciascun blocco di testo in chiaro non ha una relazione fissa con il blocco di testo in chiaro. Pertanto gli schemi ripetuti di b bit non vengono esposti nel testo cifrato.

Per quanto riguarda la decrittografia, ciascun blocco di testo cifrato passa attraverso l'algoritmo di decrittografia; il risultato subisce uno XOR con il blocco di testo cifrato precedente per produrre il blocco di testo in chiaro. Per dimostrare che questo meccanismo funziona correttamente, si può scrivere:

$$C_j = E(K[C_{j-1} \oplus P_j])$$

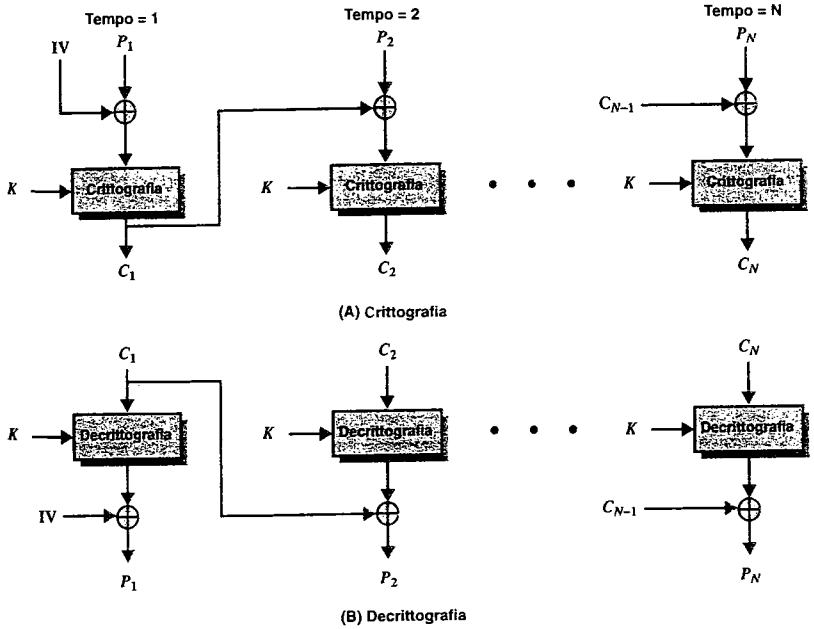


Figura 6.4 La modalità di funzionamento Cipher Block Chaining (CBC).

Quindi:

$$D(K, C_j) = D(K, E(K, [C_{j-1} \oplus P_j]))$$
$$D(K, C_j) = (C_{j-1} \oplus P_j)$$
$$C_{j-1} \oplus D(K, C_j) = C_{j-1} \oplus C_{j-1} \oplus P_j = P_j$$

Per produrre il primo blocco di testo cifrato, si calcola lo XOR di un vettore di inizializzazione IV con il primo blocco di testo in chiaro. Al momento della decrittografia, il vettore di inizializzazione subisce uno XOR con l'output dell'algoritmo di decrittografia in modo da ripristinare il primo blocco di testo in chiaro. IV ha le stesse dimensioni del blocco di testo cifrato.

Il vettore di inizializzazione IV deve essere noto sia al mittente che al destinatario ma non prevedibile da altri. Per ottenere la massima sicurezza, il vettore di inizializzazione dovrebbe essere protetto da modifiche non autorizzate. A tale scopo si può inviare il vettore di inizializzazione utilizzando la crittografia ECB. Un motivo per proteggere il vettore di inizializzazione è il seguente: se un estraneo dovesse riuscire a ingannare il destinatario e convincerlo a utilizzare un vettore di inizializzazione differente, sarebbe in grado di invertire determinati bit nel primo blocco di testo in chiaro. Per esempio si considerino le formule seguenti:

$$C_1 = E(K, [IV \oplus P_1])$$
$$P_1 = IV \oplus D(K, C_1)$$

Ora si può usare la seguente notazione: X[i] rappresenta l'i-esimo bit nella quantità X di b bit. Quindi:

$$P_1[i] = IV[i] \oplus D(K, C_1)[i]$$

Pertanto, utilizzando le proprietà dell'operatore XOR, si può stabilire che:

$$P_1[i]' = IV[i]' \oplus D(K, C_1)[i]$$

dove l'apice rappresenta l'operazione di complemento dei bit. Questo significa che se è possibile modificare i bit del vettore di inizializzazione IV, sarà anche possibile modificare i bit corrispondenti del valore ricevuto di P_1.

Per altri possibili attacchi basati sulla conoscenza del vettore di inizializzazione IV, consultare [VOYD83].

In conclusione la modalità di funzionamento CBC, dato il suo meccanismo di concatenamento, è la modalità più appropriata per la crittografia di messaggi con lunghezza maggiore di b bit.

Oltre che per ottenere la segretezza, la modalità di funzionamento CBC può essere utilizzata per l'autenticazione. Questo particolare impiego verrà descritto nella Parte seconda.

La modalità di funzionamento Cipher Feedback

Lo schema DES è fondamentalmente una tecnica di cifratura a blocchi con blocchi di b bit. Tuttavia è possibile convertire DES in una cifratura a flussi utilizzando le modalità di funzionamento cipher feedback (CFB) o output feedback. La cifratura a flussi elimina la necessità di eseguire riempimenti in un messaggio per generare un numero intero di blocchi. Inoltre può operare in tempo reale. Pertanto se si deve trasmettere un flusso di caratteri, ciascun carattere può essere crittografato e trasmesso immediatamente.

Una proprietà desiderabile in una cifratura a flussi è che il testo cifrato abbia la stessa lunghezza del testo in chiaro. Pertanto se venissero trasmessi caratteri di 8 bit, ciascun carattere dovrebbe essere crittografato utilizzando 8 bit. Se venissero utilizzati più di 8 bit, vi sarebbe uno spreco di capacità di trasmissione.

La Figura 6.5 rappresenta lo schema di funzionamento di CFB. In questa figura si presuppone che l'unità di trasmissione sia di s bit; normalmente viene utilizzato il valore $s = 8$. Come per la modalità CBC, le unità di testo in chiaro vengono concatenate in modo che il testo cifrato di qualsiasi unità di testo in chiaro sia funzione di tutto il testo in chiaro precedente. In questo caso, invece che in unità di b bit, il testo in chiaro viene suddiviso in *segmenti* di s bit.

Si consideri innanzitutto la crittografia. L'input della funzione di crittografia è un registro a scorrimento a b bit che viene inizialmente impostato con un vettore di inizializzazione IV. Gli s bit più a sinistra (i più significativi) dell'output della funzione di crittografia subiscono uno XOR con il primo segmento di testo in chiaro P_1 per produrre la prima unità di testo cifrato C_1 che viene quindi trasmessa. Il contenuto del registro di scorrimento viene fatto scorrere a sinistra di s bit e negli s bit più a destra (i meno significativi) del registro a

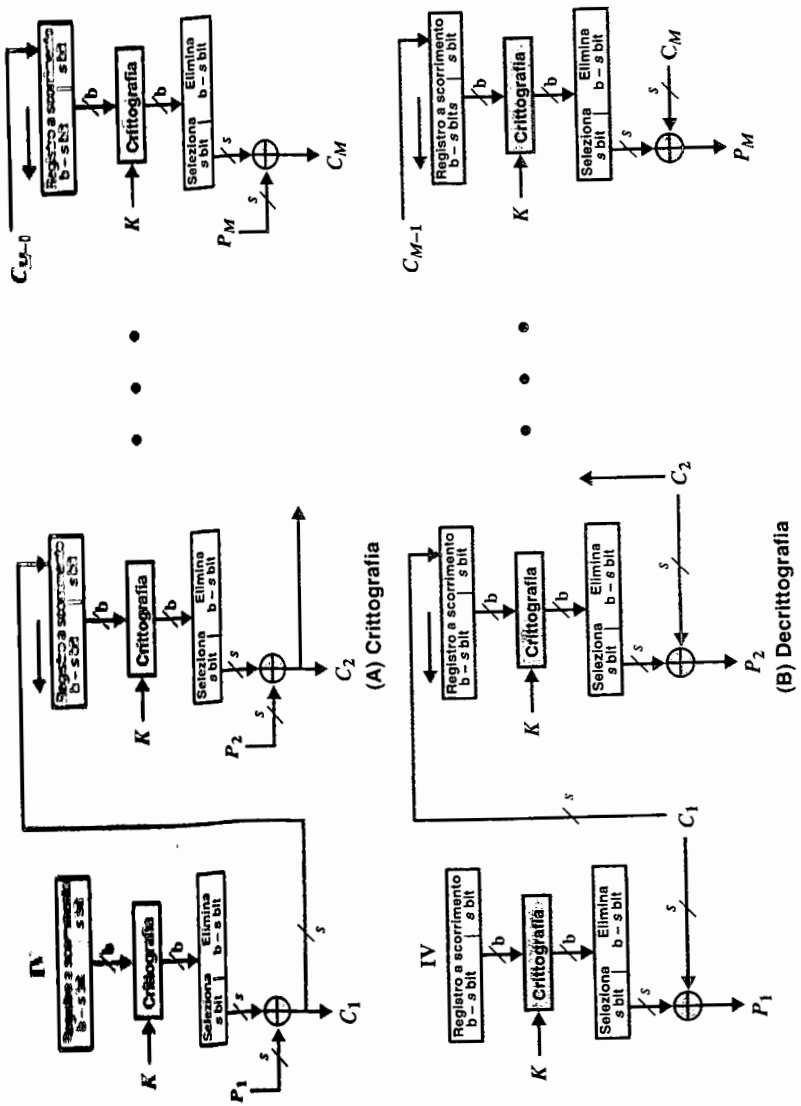


Figura 6.6 La modalità di funzionamento Cipher Feedback (CFB) a *s* bit.

scorrimento viene inserito C_1. Questo processo si ripete finché non sono state crittografate tutte le unità di testo in chiaro.

Per la decrittografia viene utilizzato lo stesso schema, tranne per il fatto che le unità di testo cifrato ricevute subiscono uno XOR con l'output della funzione di crittografia per produrre l'unità di testo in chiaro. Si noti che viene utilizzata la funzione di crittografia e non la funzione di decrittografia. Questo si può spiegare con facilità. Se $S_s(X)$ sono gli s bit più significativi di X, allora:

$$C_1 = P_1 \oplus S_s[E(K, \text{IV})]$$

Pertanto,

$$P_1 = C_1 \oplus S_s[E(K, \text{IV})]$$

Lo stesso ragionamento vale per i passi successivi dell'operazione.

La modalità di funzionamento Output Feedback

La modalità Output Feedback (OFB) ha una struttura simile a CFB ed è illustrata nella Figura 6.6. Come si può vedere, al registro di scorrimento viene inviato l'output della funzione di crittografia mentre in CFB è l'unità di testo cifrato ad essere inviata al registro a scorrimento.

Un vantaggio del metodo OFB è il fatto che non propaga gli errori di trasmissione dei bit. Per esempio, un errore nei bit di C_1, interesserebbe solo il valore recuperato di P_1, mentre le successive unità di testo in chiaro non verrebbero influenzate. Con CFB, C_1 funge anche da input del registro a scorrimento e pertanto provoca il propagarsi dell'errore.

Lo svantaggio di OFB è il fatto che è più vulnerabile a un attacco a modifica del flusso dei messaggi rispetto a CFB. Si consideri il fatto che complementando un bit nel testo cifrato si complementa il bit corrispondente nel testo in chiaro ottenuto. Pertanto, è possibile controllare dall'esterno il modo in cui viene ripristinato il testo in chiaro. Questo potrebbe consentire a un estraneo, eseguendo le modifiche opportune alla porzione di checksum del messaggio oltre che della porzione dati, di modificare il testo cifrato in modo che tale modifica non venga rilevata dal codice di correzione degli errori. Per ulteriori informazioni, consultare [VOYD83].

La modalità di funzionamento Counter

Anche se l'interesse nella modalità di funzionamento Counter (CTR) è cresciuto solo recentemente con la sua applicazione a ATM (Asynchronous Transfer Mode) e a IPSec (IP Security), questa modalità è stata proposta tempo addietro in [DIFF79].

La Figura 6.7 rappresenta questa modalità di funzionamento. Viene impiegato un contatore corrispondente alle dimensioni del blocco di testo in chiaro. L'unico requisito stabilito in SP 800-38A è che il valore del contatore debba essere differente per ciascun blocco di testo in chiaro crittografato. In genere il contatore viene inizializzato con un determinato valore e poi incrementato di un'unità per ogni blocco successivo (modulo 2^b dove b corri-

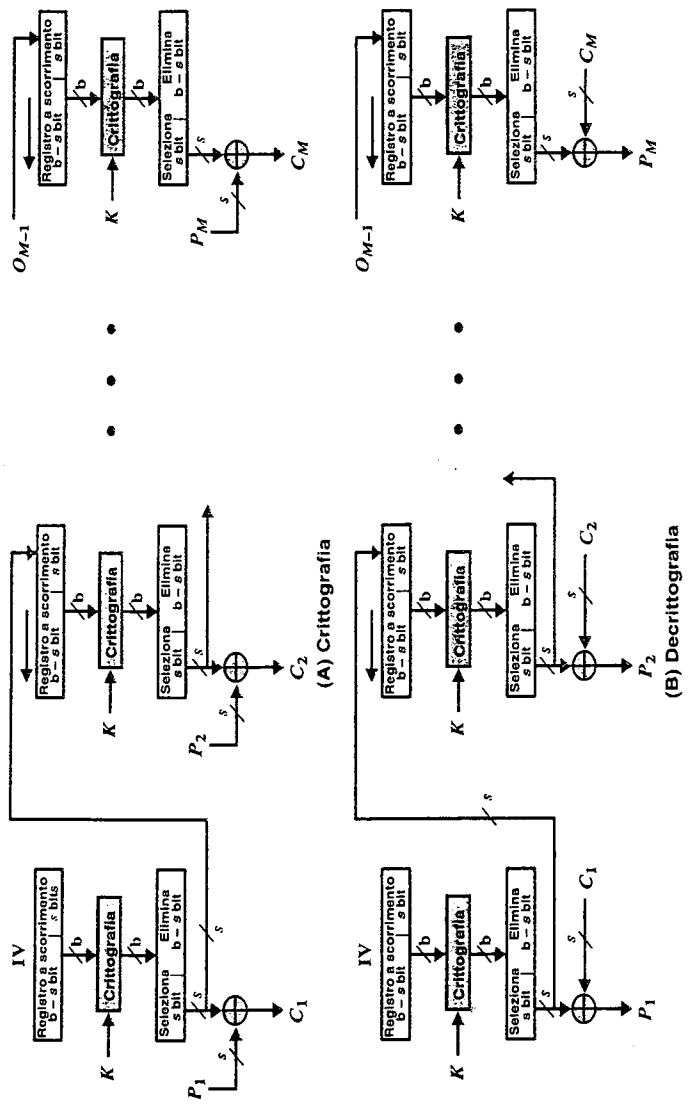


Figura 6.6 La modalità di funzionamento Output Feedback (OFB) a s bit.

sponde alle dimensioni del blocco). Per la crittografia, il contatore viene crittografato e poi viene applicato uno XOR con il blocco di testo in chiaro per produrre il blocco di testo cifrato; non vi è alcun concatenamento. Per la decrittografia viene utilizzata la stessa sequenza di valori del contatore e a ciascun contatore crittografato viene applicato lo XOR con un blocco di testo cifrato per ottenere il blocco di testo in chiaro corrispondente.

[LIPM00] elenca i seguenti vantaggi della modalità CTR.

- **Efficienza dell'hardware:** a differenza delle tre modalità a concatenamento, la crittografia (o decrittografia) in modalità CTR può essere eseguita in parallelo su più blocchi di testo in chiaro o di testo cifrato. Nelle modalità a concatenamento l'algoritmo deve completare il calcolo di un blocco prima di iniziare il blocco successivo. Questo limita il throughput massimo dell'algoritmo al reciproco del tempo di esecuzione della crittografia o decrittografia di un blocco. In modalità CTR il throughput è limitato solo dal livello di parallelismo implementato.
- **Efficienza del software:** analogamente, data la possibilità di esecuzione parallela della modalità CTR, possono essere utilizzati efficacemente microprocessori che supportano funzionalità di elaborazione concorrente come il pipelining aggressivo, l'invio di più istruzioni per ciclo di clock, una grande quantità di registri e l'impiego di architetture SIMD.
- **Pre-elaborazioni:** l'esecuzione dell'algoritmo di crittografia sottostante non dipende dall'input del testo in chiaro o del testo cifrato. Pertanto se è disponibile una quantità di memoria sufficiente e viene mantenuta la sicurezza necessaria, è possibile sfruttare la pre-elaborazione per preparare l'output delle box di crittografia che verranno inviate alle funzioni XOR nella Figura 6.7. Quando l'input di testo in chiaro o cifrato diviene disponibile, l'unico calcolo da eseguire sarà una serie di XOR. Questa strategia migliora enormemente il throughput.
- **Accesso diretto:** l'i-esimo blocco di testo in chiaro o testo cifrato può essere elaborato in modalità ad accesso diretto. Con le modalità a concatenamento, il blocco C_i non può essere calcolato se non dopo avere elaborato il blocco i - 1. Vi possono essere applicazioni in cui un testo cifrato viene memorizzato e si desidera decrittografare esclusivamente un blocco: per tali applicazioni risulta comodo impiegare un sistema ad accesso diretto.
- **Sicurezza dimostrabile:** si può dimostrare che la modalità CTR è sicura almeno quanto le altre modalità trattate in questa parte del capitolo.
- **Semplicità:** a differenza delle modalità di funzionamento ECB e CBC, la modalità CTR richiede solo l'implementazione dell'algoritmo di crittografia e non dell'algoritmo di decrittografia. Questo è ancora più importante quando l'algoritmo di decrittografia è sostanzialmente differente dall'algoritmo di crittografia, così come accade in AES. Inoltre non è necessario implementare la programmazione della chiave di decrittografia.

6.3 La cifratura a flussi e RC4

In questa parte del capitolo si parlerà della cifratura a flussi simmetrica forse più diffusa, RC4. Si partirà da una panoramica della struttura delle cifrature a flussi e quindi si esaminerà RC4.

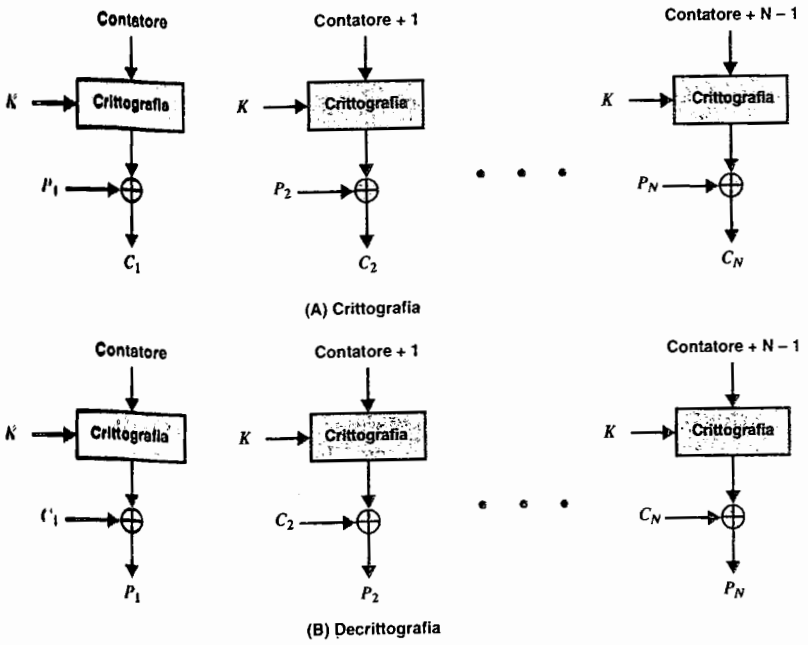


Figura 6.7 La modalità di funzionamento Counter (CTR).

La struttura delle cifrature a flussi

Una tipica cifratura a flussi esegue la crittografia del testo in chiaro un byte alla volta, anche se le cifrature a flussi possono essere progettate per operare un bit alla volta o anche su unità più grandi di un byte. La Figura 6.8 è un diagramma rappresentato dalla struttura della cifratura a flussi. Vi è una chiave che funge da input di un generatore di bit pseudocasuali che produce un flusso di numeri di 8 bit apparentemente casuali. Si parlerà dei generatori di numeri pseudocasuali nel Capitolo 7. Per il momento basti dire che un flusso pseudocasuale risulta imprevedibile senza conoscere la chiave di input. L'output del generatore, chiamato **keystream**, viene combinato un byte alla volta con il flusso di testo in chiaro utilizzando un OR esclusivo bit (XOR). Per esempio, se il byte successivo prodotto dal generatore è 01101100 e il byte successivo di testo in chiaro è 11001100, il testo cifrato risultante sarà:

```
  11001100 Testo in chiaro
⊕ 01101100 keystream
  10100000 Testo cifrato
```

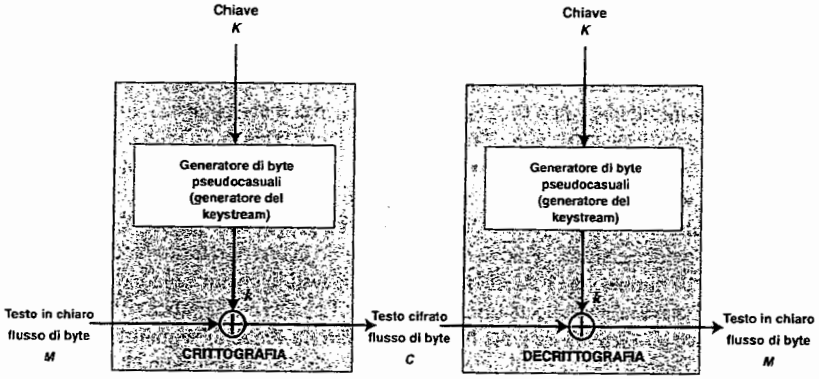



Figura 6.8 Diagramma della cifratura a flussi.

La decrittografia richiede l'uso della stessa sequenza pseudocasuale:

```
  10100000 Testo cifrato
⊕ 01101100 keystream
  11001100 Testo in chiaro
```

La cifratura a flussi è simile alla tecnica One-Time Pad trattata nel Capitolo 2. La differenza consiste nel fatto che One-Time Pad usa un flusso di numeri veramente casuali mentre una cifratura a flussi usa un flusso di numeri pseudocasuali.

[KUMA97] elenca le seguenti considerazioni, che sono particolarmente importanti per la progettazione di una cifratura a flussi.

1. La sequenza di crittografia deve avere un periodo molto ampio. Un generatore di numeri pseudocasuali usa una funzione che produce un flusso deterministico di bit che prima o poi si ripete. Più lungo è il periodo della ripetizione, più difficile sarà eseguire l'analisi crittografica. Si tratta fondamentalmente delle stesse considerazioni trattate per la cifratura di Vigenère: maggiore è la lunghezza della chiave e più difficoltosa è l'analisi crittografica.
2. Il keystream deve approssimare le proprietà di un vero flusso di numeri casuali. Per esempio vi si deve trovare un numero più o meno uguale di "1" e "0". Se il keystream viene trattato come un flusso di byte allora devono comparire approssimativamente nella stessa misura tutti i 256 possibili valori del byte. Più l'aspetto del keystream è casuale, più sarà casuale anche il testo cifrato e più sarà difficile eseguire un'analisi crittografica.
3. Dalla Figura 6.8 si noti che l'output del generatore di numeri pseudocasuali è condizionato dal valore della chiave di input. Per difendersi dagli attacchi a forza bruta, la chiave deve essere sufficientemente lunga. Valgono le stesse considerazioni già viste

per la cifratura a blocchi. Pertanto, con le tecnologie attualmente disponibili, è opportuno utilizzare una chiave di almeno 128 bit.

Con un generatore di numeri pseudocasuali progettato in modo accurato, la cifratura di un flusso può essere sicura quanto la cifratura di un blocco con una chiave di lunghezza analoga. Il vantaggio principale di una cifratura a flussi è in genere la velocità e la semplicità in termini di codice rispetto alla cifratura a blocchi. L'esempio presentato in questa parte del capitolo, RC4, può essere implementato con pochissime righe di codice. La Tabella 6.2, utilizzando dati tratti da [RESC01], confronta i tempi di esecuzione di RC4 con tre cifrature simmetriche a blocchi molto note. Il vantaggio di una cifratura a blocchi è il fatto che consente di riutilizzare le chiavi. Tuttavia, se due testi in chiaro vengono crittografati con la stessa chiave utilizzando una cifratura a flussi, l'analisi crittografica risulterà spesso molto semplice [DAWS96]. Se i due flussi di testo cifrato vengono combinati in uno solo con un'operazione XOR, il risultato è lo XOR dei testi in chiaro originali. Se i testi in chiaro sono stringhe di testo, numeri di carte di credito o altri flussi di byte con proprietà note, l'analisi crittografica può avere successo.

Tabella 6.2 Confronto fra velocità delle cifrature simmetriche su Pentium II.

Cifratura	Lunghezza della chiave	Velocità (Mbit/s)
DES	56	9
3DES	168	3
RC2	Variabile	0.9
RC4	Variabile	45

Per le applicazioni che richiedono la crittografia/decrittografia di un flusso di dati, per esempio un canale di comunicazione o un collegamento fra browser e server Web, la cifratura a flussi può essere la migliore alternativa. Per applicazioni che trattano blocchi di dati, come per esempio nel caso di trasferimento di file, messaggi di posta elettronica e informazioni memorizzate in database, può essere più appropriato utilizzare la cifratura a blocchi. Comunque, entrambe le forme di crittografia possono essere utilizzate in qualsiasi applicazione.

L'algoritmo RC4

RC4 è stata progettata nel 1987 da Ron Rivest per RSA Security. Si tratta di una cifratura a flussi con chiave di dimensione variabile e operazioni orientate ai byte. L'algoritmo si basa su una permutazione casuale. L'analisi mostra che il periodo della cifratura è in modo estremamente probabile maggiore di 10^{100} [ROBS95a]. Per ogni byte di output sono necessarie da 8 a 16 operazioni macchina e la cifratura può essere eseguita molto rapidamente anche in versione software. RC4 viene utilizzata negli standard SSL/TLS (Secure Sockets Layer/Transport Layer Security) che sono stati definiti per le comunicazioni fra i browser e i server Web. Viene utilizzata anche nel protocollo WEP (Wired Equivalent Privacy) e nel più recente protocollo WPA (WiFi Protected Access) che fanno parte dello standard per reti locali wireless IEEE 802.11. RC4 era mantenuto segreto commerciale da RSA Security; ma nel settembre 1994 l'algoritmo RC4 è stato diffuso in forma anonima via Internet dalla comunità degli hacker.

L'algoritmo RC4 è molto semplice e facile da descrivere. Una chiave di lunghezza variabile fra 1 e 256 byte (da 8 a 2048 bit) inizializza un vettore di stato S di 256 byte con gli elementi S[0], S[1], . . . , S[255]. Ogni volta, S contiene una permutazione di tutti i numeri a 8 bit compresi fra 0 e 255. Per la crittografia e la decrittografia viene generato un byte *k* (vedere la Figura 6.8) tratto da S selezionando una delle 255 voci. A ogni valore generato di *k*, le voci contenute in S vengono permutate.

Inizializzazione di S

Per iniziare, le voci di S vengono impostate utilizzando i valori compresi fra 0 e 255 ordinati in senso ascendente, ovvero S[0] = 0, S[1] = 1, ... , S[255] = 255. Viene creato anche un vettore temporaneo, T. Se la lunghezza della chiave K è di 256 byte, allora K viene trasferito in T. Altrimenti, per una chiave di lunghezza pari a *keylen* byte, i primi *keylen* elementi di T vengono copiati da K e poi K viene ripetuto il numero di volte necessario per riempire T. Queste operazioni preliminari possono essere riassunte nel seguente modo:

```
/* Inizializzazione */
for i = 0 to 255 do
      S[i] = i;
      T[i] = K[i mod keylen];
```

Successivamente si usa T per produrre la permutazione iniziale di S partendo da S[0] fino a S[255]; per ogni S[i] si scambia S[i] con un altro byte di S in base allo schema definito da T[i]:

```
/* Permutazione iniziale di S */
j = 0;
for i = 0 to 255 do
      j = (j + S[i] + T[i]) mod 256;
      Swap (S[i], S[j]);
```

Poiché l'unica operazione di S è uno scambio, l'unico effetto è una permutazione. S continua a contenere tutti i numeri compresi fra 0 e 255.

Generazione del flusso

Una volta inizializzato il vettore S, la chiave di input non viene più utilizzata. La generazione del flusso comporta un'iterazione su tutti gli elementi S[i]; per ogni S[i] si scambia

S[i] con un altro byte di S in base allo schema rappresentato dalla configurazione corrente di S. Una volta raggiunto S[255], il processo ricomincia da S[0]:

```
/* Generazione del flusso */
i, j = 0;
while (true)
      i = (i + 1) mod 256;
      j = (j + S[i]) mod 256;
      Swap (S[i], S[j]);
      t = (S[i] + S[j]) mod 256;
      k = S[t];
```

Per la crittografia, si esegue l'operazione di XOR del valore *k* con il byte successivo di testo in chiaro. Per la decrittografia si esegue l'operazione di XOR del valore *k* con il byte successivo di testo cifrato. La Figura 6.9 riepiloga la logica di funzionamento di RC4.

La robustezza di RC4

Sono stati pubblicati vari articoli che analizzano i metodi di attacco di RC4, fra cui [KNUD98], [MIST98], [FLUH00] e [MANT01]. Nessuno di questi approcci è realistico quando RC4 utilizza una chiave di lunghezza ragionevole, per esempio 128 bit. Un problema più grave è stato rilevato in [FLUH01]. Gli autori hanno dimostrato che il protocollo WEP, che aveva lo scopo di garantire la segretezza sulle reti locali wireless 802.11, è vulnerabile a un determinato tipo di attacco. Il problema non riguarda l'algoritmo RC4 ma il modo in cui vengono generate le chiavi utilizzate come input di RC4. Questo problema non sembra estendersi ad altre applicazioni che utilizzano RC4 e può essere risolto in WEP semplicemente modificando il modo in cui vengono generate le chiavi. Questo problema evidenzia la difficoltà di progettare un sistema veramente sicuro comprendente sia le funzioni crittografiche che i protocolli che le utilizzano.

6.4 Letture e siti Web consigliati

[SCHN96] fornisce dettagli su numerose cifrature simmetriche a blocchi e su alcune cifrature di flusso. [ROBS95b] esamina molti aspetti progettuali legati alle cifrature simmetriche a blocchi. [KUMA97] contiene un'ampia ed eccellente discussione sui principi di progettazione delle cifrature a blocchi. Un altro trattato molto interessante ma piuttosto tecnico è [RUEP92]. [ROBS95a] è un interessante esame dei problemi progettuali delle cifrature di flusso.

KUMA97 Kumar, I. *Cryptology*. Laguna Hills, CA: Aegean Park Press, 1997.

ROBS95a Robshaw, M. *Stream Ciphers*. RSA Laboratories Technical Report TR-701, Luglio 1995. http://www.rsasecurity.com/rsalabs/index.html.

ROBS95b Robshaw, M. *Block Ciphers*. RSA Laboratories Technical Report TR-601, Agosto 1995. http://www.rsasecurity.com/rsalabs/index.html

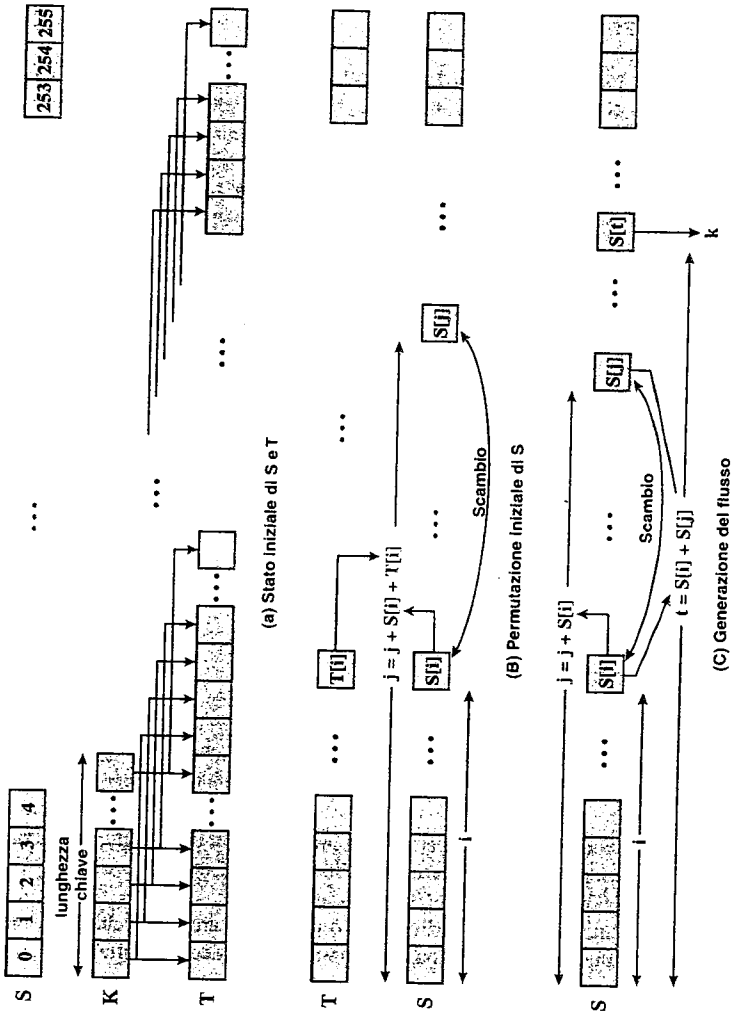


Figura 6.9 Il funzionamento dell'algoritmo RC4.

RUEP92 Rueppel, T. "Stream Ciphers". In [SIMM92].

SCHN96 Schneier, B. *Applied Cryptography*. New York: Wiley, 1996.

SIMM92 Simmons, G., ed. *Contemporary Cryptology: The Science of Information Integrity*. Piscataway, NJ: IEEE Press, 1992.

Sito Web consigliato

- **Block cipher modes of operation:** pagina NIST con la documentazione completa relativa alle modalità operative della cifratura a blocchi approvate da NIST.

6.5 Termini chiave, domande di ripasso e problemi

Termini chiave

Attacco Meet-in-the-Middle
Cifratura di flusso
RC4
Modalità CBC (Cipher Block Chaining)
Modalità CFB (Cipher Feedback)
Modalità CTR (Counter)
Modalità ECB (Electronic Codebook Mode)
Modalità OFB (Output Feedback)
Modalità operative della cifratura a blocchi
Triple DES (3DES)

Domande di ripasso

6.1 Che cosa si intende con tripla crittografia?
6.2 Che cos'è l'attacco Meet-in-the-Middle?
6.3 Quante chiavi vengono utilizzate nella crittografia tripla?
6.4 Perché la porzione centrale di un algoritmo 3DES è una decrittografia e non una crittografia?
6.5 Elencare le considerazioni progettuali più importanti per una cifratura a flussi.
6.6 Perché è preferibile non riutilizzare una chiave nella cifratura a flussi?
6.7 Quali operazioni primitive vengono utilizzate in RC4?
6.8 Perché alcune modalità di funzionamento della cifratura a blocchi usano solo la crittografia mentre altre usano sia la crittografia che la decrittografia?

Problemi

6.1 Si deve costruire un dispositivo hardware in grado di eseguire la crittografia a blocchi in modalità CBC (Cipher Block Chaining) utilizzando un algoritmo più forte di DES. Un buon candidato è rappresentato da 3DES. La Figura 6.10 mostra due possibilità che derivano dalla definizione di CBC. Quale di queste alternative si può scegliere:
   A. Per la sicurezza?
   B. Per le prestazioni?

6.2 È possibile suggerire un miglioramento alla sicurezza delle due opzioni rappresentate nella Figura 6.10, utilizzando solo 3 chip DES e un certo numero di funzioni XOR? Si supponga di continuare a utilizzare due sole chiavi.
6.3 L'attacco Merkle-Hellman a 3DES inizia supponendo che A = 0 (Figura 6.1B). Poi, per ciascuno dei 2^{56} valori possibili di K_1, viene determinato il testo in chiaro P che produce A = 0. Descrivere la parte rimanente dell'algoritmo.
6.4 Con la modalità ECB di DES, se vi è un errore in un blocco del testo cifrato trasmesso, verrà interessato solo il blocco di testo in chiaro corrispondente. Al contrario nella modalità CBC, questo errore si propaga. Per esempio, un errore nella trasmissione di C_1 (Figura 6.4) genera ovviamente errori anche in P_1 e P_2.
   A. I blocchi oltre P_2 vengono interessati dall'errore?
   B. Supponendo che vi sia un errore nei bit nella versione originale di P_1, in quanti blocchi di testo cifrato si propaga tale errore? Qual è l'effetto alla ricezione?
6.5 Se si verifica un errore nella trasmissione di un carattere in testo cifrato in modalità CFB a 8 bit, a quale distanza si propaga l'errore?

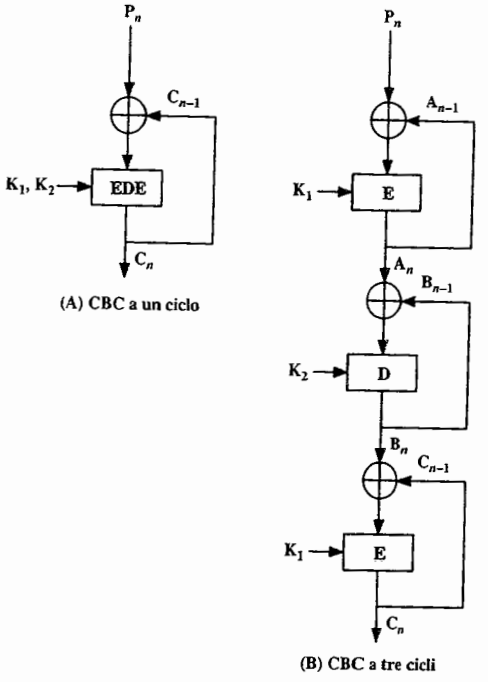


Figura 6.10 Uso di Triple DES in modalità CBC.

6.6 Terminare la compilazione della seguente tabella.

Modalità	Crittografia	Decrittografia
ECB	$C_j = E[K, P_j] \quad j = 1,\ldots,N$	$P_j = D[K, C_j] \quad j = 1,\ldots,N$
CBC	$C_j = E[K, P_j \oplus \text{IV}]$ $C_j = E[K, P_j \oplus C_{j-1}] \quad j = 2,\ldots,N$	$P_j = D[K, C_j] \oplus \text{IV}$ $P_j = D[K, C_j] \oplus C_{j-1} \quad j = 2,\ldots,N$
CFB		
OFB		
CTR		

6.7 CBC-Pad è una modalità operativa impiegata nella cifratura a blocchi RC5, ma potrebbe essere utilizzata con qualsiasi algoritmo di cifratura a blocchi. CBC-Pad gestisce testo in chiaro di qualsiasi lunghezza. Il testo cifrato risulta più lungo del testo in chiaro di, al massimo, la lunghezza di un singolo blocco. Viene utilizzata la tecnica di riempimento per fare in modo che la lunghezza del testo in chiaro in input sia un multiplo della lunghezza del blocco. Si suppone che la lunghezza in byte del testo in chiaro originale sia un numero intero. La lunghezza di questo testo viene portata a un numero multiplo della dimensione del blocco, riempiendo l'eventuale ultimo blocco parziale con dei byte di riempimento tutti identici, da 1 a *bb* (*bb* = dimensione del blocco) aventi valore uguale al numero di byte di riempimento. Per esempio, se ci fossero 8 byte di riempimento, ciascun byte avrebbe il valore 00001000. Per quale motivo non è possibile avere un numero nullo di byte di riempimento? Ovvero, se già la dimensione del testo in chiaro originale fosse un multiplo della dimensione del blocco, perché sarebbe necessario comunque effettuare il riempimento?

6.8 Il riempimento potrebbe non essere sempre conveniente. Per esempio si potrebbero voler memorizzare i dati cifrati nello stesso buffer che conteneva inizialmente il testo in chiaro: in questo caso il testo cifrato dovrebbe avere lunghezza pari a quella del testo in chiaro. Una modalità operativa che soddisfa questo vincolo è la modalità CTS (ciphertext stealing). La Figura 6.11a illustra un'implementazione di questa modalità.

A. Spiegarne il funzionamento.

B. Descrivere come decifrare C_{n-1} e C_n.

6.9 La Figura 6.11b illustra un'alternativa a CTS per produrre un testo cifrato di lunghezza uguale a quella del testo in chiaro, quando quest'ultima non è un multiplo intero della dimensione del blocco.

A. Spiegare l'algoritmo.

B. Spiegare il motivo per cui CTS è preferibile all'approccio illustrato nella Figura 6.11b.

6.10 Quale valore della chiave di RC4 lascia S invariato durante l'inizializzazione (ovvero, dopo la permutazione iniziale di S, le voci di S saranno uguali ai valori da 0 a 255 in ordine ascendente)?

6.11 RC4 ha uno stato interno segreto che consiste nella permutazione di tutti i possibili valori del vettore **S** e dei due indici *i* e *j*.

A. Utilizzando un metodo diretto per memorizzare lo stato interno, quanti bit sarebbero necessari?

B. Si vuole valutare il problema in termini di quantità di informazione rappresentata dallo stato. È necessario quindi determinare quanti sono i possibili stati differenti, per poi calcolarne il logaritmo in base 2 che determina il numero di bit di informazione rappresentati. Con questo metodo, quanti bit sarebbero necessari per rappresentare lo stato?

6.12 Alice e Bob si accordano per comunicare privatamente via posta elettronica con uno schema basato su RC4, ma vogliono evitare di utilizzare una nuova chiave segreta per ogni comunicazione. Definiscono quindi privatamente una chiave *k* di 128 bit. Per crittografare un messaggio m, che consiste di una stringa di bit, utilizzano il procedimento seguente:

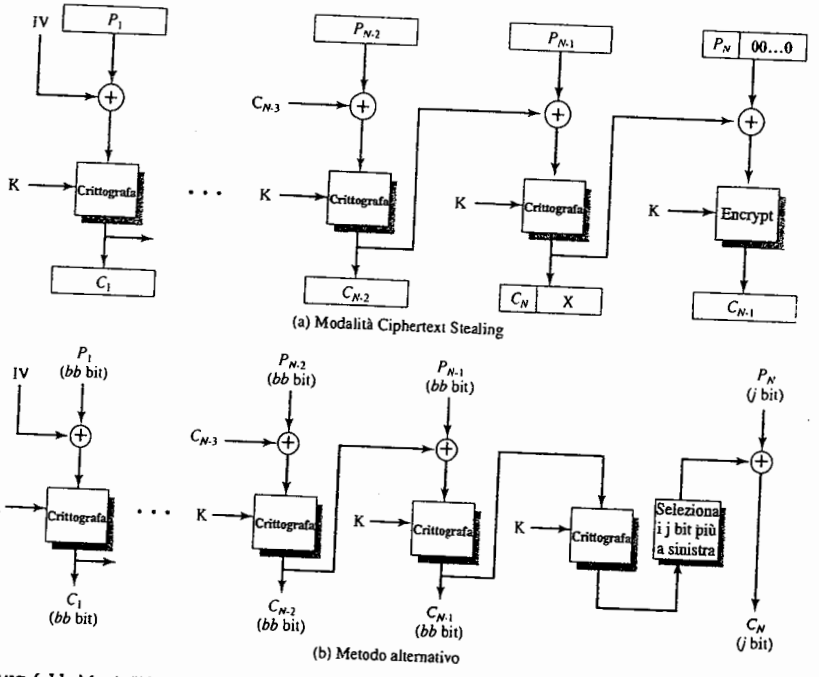


Figura 6.11 Modalità di cifratura a blocchi impiegabili quando la lunghezza del testo in chiaro non è multipla della dimensione del blocco.

1. Scegliere un valore v di 80 bit casuale.
2. Generare il testo cifrato $c = RC4(v \| k) \oplus m$.
3. Inviare la stringa di bit $(v \| c)$

A. Supporre che Alice utilizzi questa procedura per inviare un messaggio m a Bob. Descrivere come Bob possa ottenere il messaggio m da $(v \| c)$ utilizzando k.

B. Se un avversario intercettasse vari valori $(v_1 \| c_1), (v_2 \| c_2)$... trasmessi fra Alice e Bob, come potrebbe determinare quando è stata utilizzata la stessa chiave di flusso per crittografare due messaggi?

C. Quanti messaggi Alice, approssimativamente, può attendersi di inviare prima che la chiave di flusso venga riutilizzata? Utilizzare i risultati del paradosso del compleanno descritto nell'Appendice 11a (Equazione 11.7).

D. Quali sono le implicazioni relative al tempo di vita della chiave k (ovvero, qual è il numero dei messaggi che possono essere crittografati utilizzando k)?

Problemi di programmazione

6.13 Sviluppare un programma di crittografia/decrittografia in modalità CBC utilizzando uno degli algoritmi seguenti: affine modulo 256, Hill modulo 256, S-DES, DES. Dati di verifica per S-DES: utilizzando il vettore binario di inizializzazione 1010 1010, il testo in chiaro binario 0000 0001 0010 0011 e la chiave binaria di crittografia 01111 11101, si dovrebbe ottenere il testo cifrato 1111 0100 0000 1011. La decrittografia dovrebbe procedere in modo corrispondente.

6.14 Sviluppare un programma di crittografia/decrittografia in modalità Cipher Feedback a 4 bit, utilizzando uno dei seguenti algoritmi di crittografia: additivo modulo 256, affine modulo 256, S-DES;

oppure

in modalità Cipher Feedback a 8 bit utilizzando come algoritmo di crittografia Hill 2 × 2 modulo 256.
Dati di verifica per S-DES: utilizzando il vettore binario di inizializzazione 1010 1011, il testo in chiaro binario 0001 0010 0011 0100 e la chiave binaria di crittografia 01111 11101, si dovrebbe ottenere il testo cifrato 1110 1100 1111 1010. La decrittografia dovrebbe procedere in modo corrispondente.

6.15 Sviluppare un programma di crittografia/decrittografia in modalità Output Feedback a 4 bit, utilizzando uno dei seguenti algoritmi di crittografia: additivo modulo 256, affine modulo 256, S-DES;

oppure

in modalità Output Feedback a 8 bit utilizzando come algoritmo di crittografia Hill 2 x 2 modulo 256.

6.16 Sviluppare un programma di crittografia/decrittografia in modalità Counter, utilizzando uno dei seguenti algoritmi di crittografia: affine modulo 256, Hill modulo 256, S-DES.

Dati di verifica per S-DES: utilizzando il contatore inizializzato a 0000 0000, il testo in chiaro binario 0000 0001 0000 0010 0000 0100 e la chiave binaria di crittografia 01111 11101, si dovrebbe ottenere il testo cifrato 0011 1000 0100 1111 0011 0010. La decrittografia dovrebbe procedere in modo corrispondente.

6.17 Implementare un attacco ad analisi crittografica differenziale contro S-DES a tre fasi (3S-DES).

Capitolo 7

Segretezza e crittografia simmetrica

Concetti essenziali

- I dispositivi di crittografia, in un ambiente distribuito, possono essere utilizzati per supportare la crittografia di canale o la crittografia end-to-end. Nella **crittografia di canale** entrambi i capi di ciascun canale trasmissivo potenzialmente vulnerabile vengono dotati di un dispositivo di crittografia. Nella **crittografia end-to-end**, invece, il processo di crittografia viene attuato nei due sistemi finali.
- Anche nel caso in cui tutto il traffico fra gli utenti venisse crittografato, l'**analisi del traffico** potrebbe comunque fornire a un avversario informazioni utili. Una contromisura efficace consiste nel riempimento del traffico: questo comporta l'invio di bit casuali nei periodi in cui non è richiesta la trasmissione di alcun messaggio.
- La **distribuzione delle chiavi** è la funzione che consente di fornire le chiavi alle due parti che desiderano scambiare dati crittografati. Per consentire la distribuzione delle chiavi in modalità sicura sono necessari meccanismi e protocolli particolari.
- La **distribuzione delle chiavi** comporta spesso l'uso di chiavi master, utilizzate molto raramente ma valide per un arco di tempo molto lungo, e chiavi di sessione, generate e distribuite per l'impiego temporaneo da parte delle due parti.
- La generazione di **numeri casuali o pseudocasuali** è una funzionalità con applicazioni in numerosi ambiti della crittografia. Il requisito fondamentale per questa funzionalità è che il flusso di numeri generati non sia prevedibile.

Storicamente si è sempre stati interessati all'uso della crittografia simmetrica per garantire la segretezza delle informazioni. Solo ultimamente nella teoria e nella pratica della crittografia sono state considerate anche altre funzionalità, fra cui l'autenticazione, l'integrità dei dati, la firma digitale e la crittografia a chiave pubblica.

Prima di esaminare alcuni di questi argomenti più recenti, ci si concentrerà sull'uso della crittografia simmetrica per garantire la segretezza. Questo argomento rimane comunque il più importante. Inoltre, la conoscenza di queste problematiche aiuterà a motivare gli

sviluppi della crittografia a chiave pubblica e a chiarire i problemi legati ad altre applicazioni della crittografia, per esempio l'autenticazione.

Si inizierà con una serie di considerazioni sul punto in cui posizionare la logica crittografica; le scelte principali in questo campo sono fra crittografia di canale e crittografia end-to-end. Poi si parlerà dell'uso della crittografia per fronteggiare gli attacchi ad analisi del traffico. Quindi si affronterà il complesso problema della distribuzione della chiave. Infine si parlerà dei principi su cui si basa uno strumento importante per garantire la segretezza: la generazione di numeri casuali.

7.1 Posizionamento della funzionalità di crittografia

Se la crittografia deve essere utilizzata per fronteggiare gli attacchi alla segretezza, occorre decidere cosa crittografare e dove posizionare la funzionalità di crittografia. Questa parte del capitolo esamina i possibili punti d'attacco alla sicurezza e analizza successivamente i due principali approcci al posizionamento della crittografia: crittografia di canale e crittografia end-to-end.

Potenziali punti di attacco alla segretezza

Come esempio si consideri una workstation operante in una tipica organizzazione aziendale. La Figura 7.1 suggerisce i vari sistemi di comunicazione che possono essere impiegati da tale workstation e pertanto dà un'idea dei punti deboli esistenti.

In molte organizzazioni le workstation sono connesse a una rete locale (LAN). In genere l'utente può raggiungere altre workstation, host e server direttamente sulla propria rete locale o su altre reti locali dello stesso edificio e interconnesse tramite bridge e router. Ecco il primo punto debole: in questo caso, il principale problema è l'intercettazione da parte di un altro dipendente della stessa organizzazione. In genere una rete locale è di tipo broadcast; in altre parole le trasmissioni emesse da una stazione e dirette a un'altra stazione risultano visibili sul mezzo di trasmissione della rete locale a tutte le stazioni. I dati vengono trasmessi sotto forma di frame e ogni frame contiene l'indirizzo sorgente e di destinazione. È quindi possibile monitorare il traffico sulla rete locale e intercettarlo in base a tali indirizzi. Se la LAN è parzialmente o totalmente wireless, le possibilità di intercettazione sono ancora maggiori.

Inoltre l'intercettatore non deve necessariamente essere un dipendente nello stesso edificio. Se la rete locale, tramite un server di comunicazioni o uno degli host della rete offre la possibilità di connettersi dall'esterno, un intruso potrebbe avere accesso alla rete locale e monitorare il traffico.

L'accesso dalla rete locale al mondo esterno (Internet) è quasi sempre disponibile grazie a un router, a una batteria di modem o a qualche altro server di comunicazioni. Dal server vi è una linea che conduce a un armadio di cablaggio. L'armadio di cablaggio funge da punto di interconnessione per le linee interne, le linee telefoniche e le linee di comunicazione esterne.

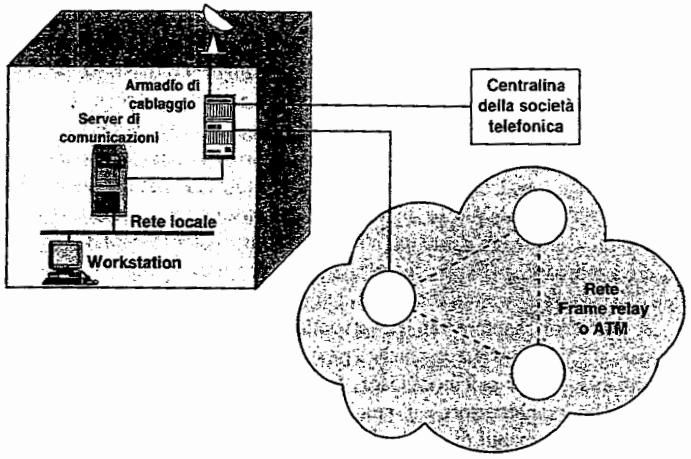


Figura 7.1 Punti deboli.

Lo stesso armadio di cablaggio è vulnerabile. Se un intruso dovesse riuscire a penetrare nell'edificio e raggiungere questo armadio, potrebbe mettersi in ascolto su ciascun cavo per determinare quali vengono utilizzati per la trasmissione dei dati. Dopo avere isolato una o più linee, l'intruso potrebbe connettervi un trasmettitore radio a bassa potenza. I segnali trasmessi potrebbero essere raccolti nelle immediate vicinanze (per esempio da un van parcheggiato nelle vicinanze o da un edificio adiacente).

Dall'armadio di cablaggio sono disponibili vari percorsi. Una configurazione standard prevede l'accesso al centralino più vicino della società telefonica. I cavi che raggiungono l'armadio vengono raccolti in un unico fascio di cavi che normalmente si unisce ad ulteriori cavi nel piano terreno dell'edificio. Da qui un grosso fascio di cavi raggiungerà il centralino locale.

L'armadio di cablaggio può offrire inoltre una connessione con un'antenna a microonde per un collegamento wireless via satellite o terrestre a microonde end-to-end. Il collegamento via antenna può far parte di una rete privata o può essere un collegamento locale che si aggancia a un collegamento operatore a larga distanza.

L'armadio di cablaggio può anche fornire un collegamento a un nodo di una rete a commutazione di pacchetto. Questo collegamento può essere costituito da una linea dedicata, da una linea privata diretta o da una connessione commutata tramite una rete pubblica di telecomunicazioni. All'interno della rete i dati attraversano vari nodi e collegamenti fino a giungere al nodo cui è connesso il sistema di destinazione.

Un attacco può aver luogo in uno qualsiasi di questi punti di collegamento. Per svolgere un attacco attivo, l'intruso deve acquisire il controllo fisico di una porzione di canale e inserirsi nelle trasmissioni per catturarle. Per svolgere un attacco passivo, l'intruso deve semplicemente essere in grado di osservare le trasmissioni. I collegamenti interessati possono essere via cavo (doppino telefonico, cavo coassiale o fibra ottica), collegamenti a microonde o canali via satellite. I doppini telefonici e i cavi coassiali possono essere attac-

cati utilizzando strumenti invasivi o dispositivi induttivi che monitorizzano le emissioni elettromagnetiche. I dispositivi invasivi consentono di svolgere sia attacchi attivi che passivi mentre i dispositivi induttivi sono utili solo per gli attacchi passivi. Nessuno di questi sistemi è però efficace contro le fibre ottiche e questo è uno di dei vantaggi di questo mezzo di trasmissione. La fibra ottica infatti non genera alcuna emissione elettromagnetica e pertanto non è vulnerabile ai dispositivi induttivi. La rottura fisica di un cavo a fibre ottiche produce un grave degrado della qualità del segnale e risulta pertanto facilmente rilevabile. Le trasmissioni a microonde e via satellite possono invece essere facilmente intercettate con rischi minimi. Questo vale particolarmente per le trasmissioni via satellite che coprono un'ampia area geografica. È anche possibile svolgere attacchi attivi ai collegamenti a microonde e via satellite, ma si tratta di operazioni tecnicamente più difficili e costose.

Oltre ai punti deboli rappresentati dai vari collegamenti, occorre anche considerare che possono essere soggetti a un attacco anche i processori posti lungo il percorso. Un attacco può consistere in un tentativo di modificare l'hardware o il software, di acquisire l'accesso alla memoria del microprocessore o di monitorare le emissioni elettromagnetiche. Questi attacchi sono meno probabili di quelli che riguardano i collegamenti ma rappresentano comunque potenziale fonte di rischio.

Un attacco può essere quindi sferrato in numerosi punti. Inoltre, nelle comunicazioni su scala geografica, molti di questi punti non sono sotto il controllo diretto degli utenti finali. Persino nel caso di reti locali, dove è possibile applicare misure di sicurezza fisica, occorre sempre considerare la minaccia rappresentata da un dipendente insoddisfatto.

Crittografia di canale o end-to-end

La crittografia rappresenta, come si è detto, l'approccio più potente e più comune per garantire la sicurezza dei punti deboli. Se si impiega la crittografia per fronteggiare questo tipo di attacchi, occorre decidere cosa crittografare e dove localizzare il sistema di crittografia. Come si può vedere nella Figura 7.2, esistono fondamentalmente due alternative: la crittografia di canale e la crittografia end-to-end.

Approcci di base

Con la crittografia di canale[1], ciascun collegamento vulnerabile viene dotato a entrambe le estremità di un dispositivo di crittografia. Questa tecnica rende sicuro tutto il traffico su tutti i collegamenti. Sebbene questo richieda l'impiego di una grande quantità di dispositivi di crittografia, il valore di questa soluzione è evidente. Uno degli svantaggi è che il messaggio deve essere decrittografato ogni volta che entra in uno switch (per esempio uno switch Frame Relay) in quanto, per poter indirizzare il frame, lo switch deve leggere l'indirizzo (numero di connessione logica) contenuto nell'intestazione del pacchetto. Pertanto il messaggio risulta vulnerabile in ogni switch. Se viene utilizzata la rete pubblica, occorre valutare il fatto che l'utente non ha alcun controllo sulla sicurezza dei nodi.

Occorre menzionare varie implicazioni della crittografia di canale. Perché questa strategia sia efficace, tutti i potenziali collegamenti del percorso che unisce la sorgente alla

[1] Detta anche "del collegamento", da "link encryption".

destinazione devono utilizzare la crittografia di canale. Ogni coppia di nodi che condivide un collegamento deve avere una chiave univoca condivisa e per ogni collegamento deve essere utilizzata una chiave differente. Pertanto occorre generare una grande quantità di chiavi.

Con la crittografia end-to-end, il processo di crittografia viene eseguito solo dai due sistemi terminali. L'host o il terminale di origine esegue la crittografia dei dati. I dati crittografati vengono poi trasmessi inalterati lungo la rete verso il terminale o l'host di destinazione. La destinazione condivide una chiave con la sorgente e dunque sarà in grado di decrittografare i dati. Questo piano sembra rendere sicure le trasmissioni contro attacchi ai collegamenti di rete o gli switch. Pertanto la crittografia end-to-end riduce le preoccupazioni degli utenti sul grado di sicurezza delle reti e dei collegamenti che supportano le comunicazioni. Rimane comunque un punto debole.

Si consideri la seguente situazione. Un host si connette a una rete Frame Relay o ATM, attiva una connessione logica con un altro host e si prepara a trasferirgli dei dati utilizzando la crittografia end-to-end. I dati vengono trasmessi in rete sotto forma di pacchetti costituiti da un'intestazione seguita dai dati utente. Quale parte di ciascun pacchetto deve essere crittografata dall'host? Si supponga che l'host esegua la crittografia dell'intero pacchetto, compresa l'intestazione. Il collegamento non potrebbe funzionare poiché, come si ricorderà, solo l'altro host potrebbe eseguire la decrittografia del pacchetto. Il nodo di commutazione Frame Relay o ATM riceverebbe i pacchetti crittografati: non sarebbe in grado di leggere l'intestazione e pertanto non potrebbe indirizzare il pacchetto. Ne consegue che l'host può crittografare solo la porzione dei dati utente del pacchetto e deve lasciare l'intestazione in chiaro.

Pertanto, con la crittografia end-to-end, i dati utente sono sicuri ma non il traffico in quanto l'intestazione dei pacchetti viene trasmessa in chiaro. D'altra parte, la crittografia end-to-end garantisce un certo livello di autenticazione. Se due sistemi terminali condividono una chiave di crittografia, il destinatario saprà con certezza che ogni messaggio che riceve proviene dal mittente indicato, poiché solo il mittente è in possesso della chiave di crittografia corrispondente. Tale autenticazione non è invece presente nello schema di crittografia di canale.

Per ottenere una maggiore sicurezza, è necessario adottare sia la crittografia di canale che la crittografia end-to-end, come indicato nella Figura 7.2. Quando vengono impiegate entrambe le forme di crittografia, l'host esegue la crittografia dei dati utente del pacchetto utilizzando la chiave di crittografia end-to-end. L'intero pacchetto viene poi crittografato utilizzando la chiave di crittografia di canale. Mentre il pacchetto attraversa la rete, ogni switch esegue la decrittografia del pacchetto utilizzando la chiave di crittografia di canale, legge l'intestazione e poi esegue nuovamente la crittografia dell'intero pacchetto per inviarlo al collegamento successivo. Ora l'intero pacchetto è sicuro, tranne quando il pacchetto si trova in uno switch, momento in cui l'intestazione del pacchetto è in chiaro.

La Tabella 7.1 riepiloga le principali caratteristiche delle due strategie di crittografia.

Posizionamento logico delle funzionalità di crittografia end-to-end

Con la crittografia di canale, la funzionalità di crittografia viene eseguita a un livello molto basso della gerarchia di comunicazione. In termini di livelli OSI (Open Systems Interconnection), la crittografia di canale avviene a livello fisico o di collegamento.

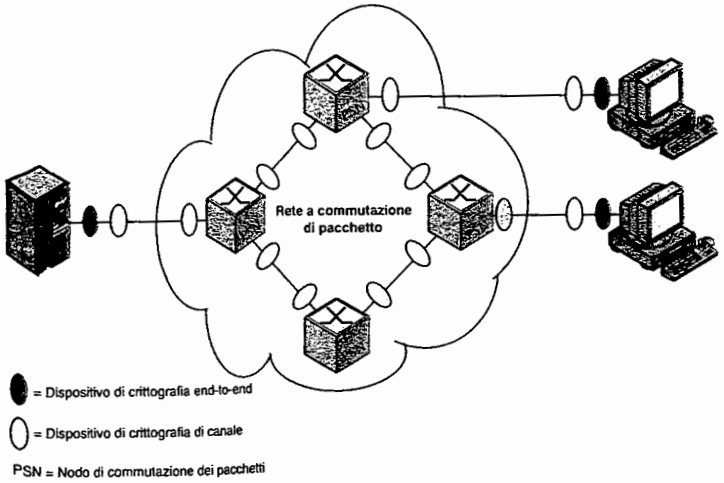


Figura 7.2 Crittografia in una rete o commutazione di pacchetto.

Tabella 7.1 Le caratteristiche della crittografia di canale e end-to-end (PFLE02).

Crittografia di canale	Crittografia end-to-end
Sicurezza nei sistemi terminali e nei sistemi intermedi	
Messaggio visibile nell'host mittente Messaggio visibile nei nodi intermedi	Messaggio crittografato nell'host mittente Messaggio crittografato nei nodi intermedi
Ruolo dell'utente	
Crittografia applicata dall'host trasmittente Trasparente all'utente Il sistema di crittografia è gestito dall'host Un sistema di crittografia per tutti gli utenti Può essere eseguita in hardware Vengono crittografati tutti i messaggi o nessuno di essi	Crittografia applicata dal processo trasmittente La crittografia è applicata dall'utente L'algoritmo è determinato dall'utente Lo schema di crittografia è selezionato dall'utente Implementazione software L'utente sceglie di crittografare o meno ciascun messaggio
Problemi implementativi	
Richiede una chiave per ogni coppia host/nodo intermedio e per ogni coppia nodo intermedio/nodo intermedio Garantisce l'autenticazione degli host	Richiede una chiave per ogni coppia di utenti Garantisce l'autenticazione dell'utente

Per il posizionamento logico della funzionalità di crittografia end-to-end sono invece disponibili varie scelte. Al livello più basso, la funzionalità di crittografia può essere eseguita al livello rete. Pertanto, per esempio, la crittografia può essere associata al protocollo Frame Relay o ATM in modo che venga crittografata la porzione dei dati utente di tutti i frame o delle celle ATM.

Con la crittografia a livello rete, il numero di entità identificabili e protette separatamente coincide con il numero di sistemi terminali della rete. Ciascun sistema terminale può attivare uno scambio crittografato con qualsiasi altro sistema terminale, posto che i due sistemi condividano una chiave segreta. Per raggiungere un determinato sistema terminale di destinazione, tutti i processi utente e le applicazioni di ciascun sistema terminale impiegano lo stesso schema di crittografia con la stessa chiave. In questo modo si possono demandare le funzionalità di crittografia a un processore FEP (Front-End Processor), in genere una scheda di comunicazione del sistema terminale.

La Figura 7.3 schematizza le funzionalità di crittografia del processore FEP. Sul lato dell'Host il FEP accetta i pacchetti. La porzione dei dati utente del pacchetto viene crittografata mentre l'intestazione del pacchetto scavalca il processo di crittografia.[2] Il pacchetto risultante viene consegnato alla rete. Nella direzione opposta, la porzione dei dati utente dei pacchetti in arrivo dalla rete viene decrittografata e viene consegnato all'host l'intero pacchetto. Se ill FEP gestisce le funzionalità di livello trasporto (per esempio TCP), anche l'intestazione del livello trasporto verrà lasciata in chiaro e verrà crittografata solo la porzione dei dati utente del protocollo di trasporto.

L'impiego dei servizi di crittografia nei protocolli end-to-end, come nel livello di rete di Frame Relay o TCP, garantisce la sicurezza end-to-end per il traffico all'interno di una rete perfettamente integrata. Tuttavia tale schema non può garantire tale sicurezza per il traffico che attraversa i confini della rete, per esempio la posta elettronica, lo scambio di dati elettronici e il trasferimento di file.

La Figura 7.4 illustra gli elementi coinvolti. In questo esempio un gateway di posta elettronica interconnette una rete che utilizza l'architettura OSI con una rete che utilizza l'architettura TCP/IP[3]. In una configurazione di questo tipo, non esiste alcun protocollo end-to-end sotto al livello applicazione. Le connessioni di trasporto e di rete da ciascuno dei sistemi terminali terminano al gateway di posta che configura nuove connessioni di trasporto e di rete per collegarsi all'altro sistema terminale.

Questa situazione non si limita al caso di un gateway posto fra due architetture differenti. Anche se entrambi i sistemi terminali usano TCP/IP o OSI, vi sono numerose configurazioni in cui i gateway di posta elettronica interconnettono reti che altrimenti sarebbero isolate. Pertanto, per applicazioni come la posta elettronica, che hanno una modalità di funzionamento store-and-forward, l'unico punto utile per impiegare la crittografia end-to-end è al livello applicazione.

Un difetto della crittografia a livello applicazione è il fatto che aumenta notevolmente il numero di entità da considerare. Una rete formata da centinaia di host può dover gestire migliaia di utenti e processi. Pertanto, sarà necessario generare e distribuire una enorme

[2] Vengono frequentemente utilizzati i termini *rosso* e *nero*. I dati *rossi* sono dati riservati o segreti in chiaro. I dati *neri* sono dati crittografati.

[3] L'Appendice H contiene una breve rassegna del modello OSI e dell'architettura TCP/IP.

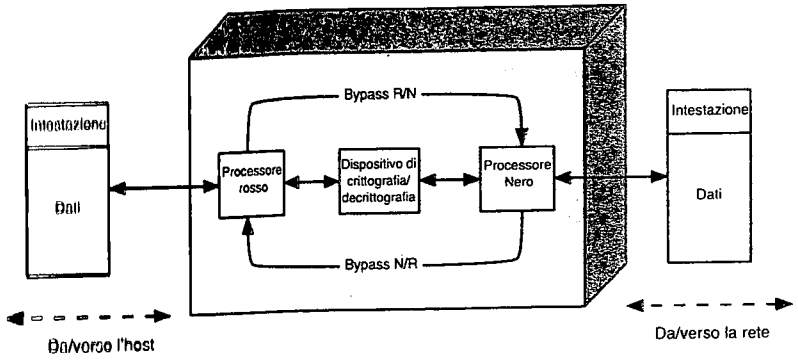


Figura 7.3 La funzione del processore FEP

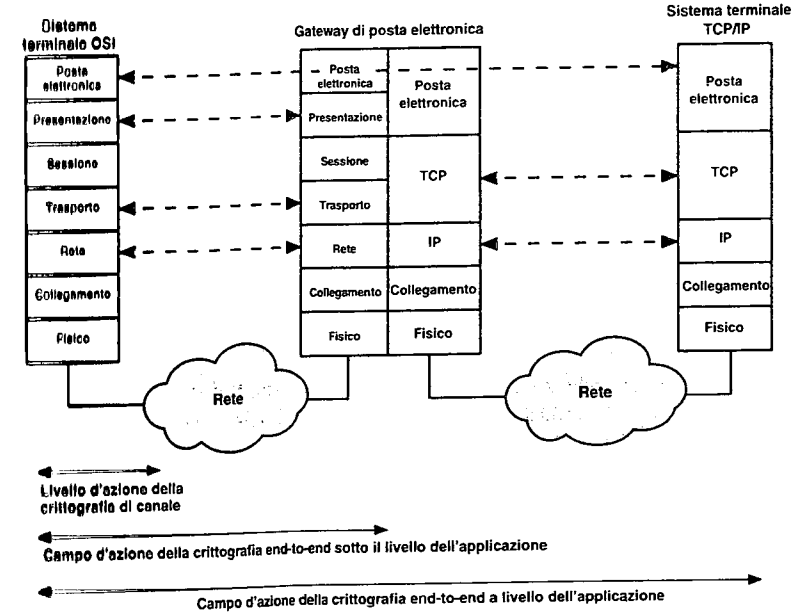


Figura 7.4 Implicazioni della crittografia per le comunicazioni a memorizzazione e inoltro.

quantità di chiavi segrete. Un modo interessante per visualizzare le alternative consiste nel notare che mentre ci si sposta verso l'alto nella gerarchia di comunicazione, vengono crittografate meno informazioni ma si ottiene un maggiore livello di sicurezza. La Figura 7.5 evidenzia questo fatto utilizzando come esempio l'architettura TCP/IP. Nella figura, il gateway a livello applicazione è un dispositivo store and forward operante a livello applicazione.[4] Con la crittografia a livello applicazione (Figura 7.5A), viene crittografata solo la porzione dei dati utente di un segmento TCP. Le intestazioni TCP, IP e a livello di collegamento dati e la coda a livello collegamento dati sono in chiaro. Al contrario, se la crittografia viene eseguita a livello TCP (Figura 7.5B), in una singola connessione end-to-end saranno crittografati sia i dati utente che l'intestazione TCP. L'intestazione IP rimane in chiaro poiché è necessaria per consentire ai router di indirizzare il datagram IP dall'origine alla destinazione. Si noti però che se un messaggio attraversa un gateway, la connessione TCP viene terminata e viene aperta una nuova connessione di trasporto per la tratta successiva.

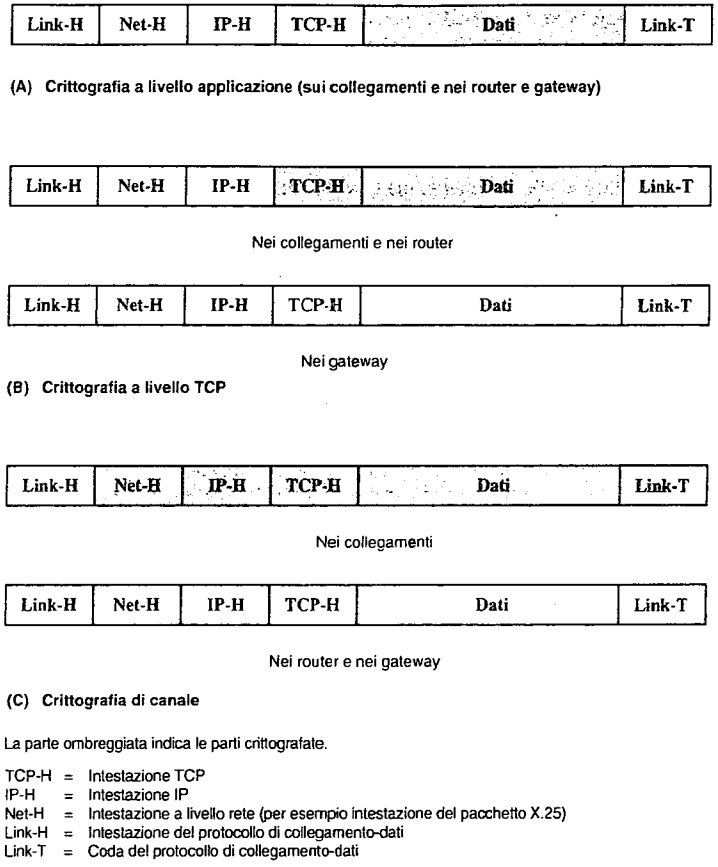


Figura 7.5 Relazione fra la crittografia e i livelli dei protocolli.

[4] Sfortunatamente, la maggior parte dei documenti TCP/IP usa il termine "gateway" per far riferimento a ciò che viene comunemente chiamato router.

Inoltre il gateway viene trattato come una destinazione dal protocollo IP sottostante. Pertanto le porzioni crittografate delle unità dati vengono decrittografate al gateway. Se la tratta successiva si trova su una rete TCP/IP, allora i dati utente e l'intestazione TCP vengono crittografati nuovamente prima della trasmissione. Tuttavia nel gateway stesso l'unità dati viene bufferizzata interamente in chiaro. Infine, per la crittografia di canale (Figura 7.5C), in ciascun collegamento viene crittografata l'intera unità dati ad eccezione dell'intestazione e della coda del pacchetto, ma l'intera unità dati si trova in chiaro sia nel router che nel gateway.[5]

7.2 Segretezza del traffico

Nel Capitolo 1 si è detto che, in alcuni casi, gli utenti sono preoccupati della sicurezza contro gli attacchi ad analisi del traffico. La conoscenza del numero e della lunghezza dei messaggi fra i nodi può consentire a un estraneo di determinare chi sta comunicando con chi. Questo può avere implicazioni ovvie in un conflitto militare. Anche nelle applicazioni commerciali, l'analisi del traffico può fornire informazioni che i soggetti interessati preferirebbero mantenere segrete. [MUFT89] elenca i seguenti tipi di informazioni che possono essere ottenuti da un'analisi del traffico.

- Identità dei partner.
- Frequenza di comunicazione dei partner.
- Schema di trasmissione dei messaggi, lunghezza o quantità di messaggi che suggeriscono lo scambio di informazioni importanti.
- Eventi correlati a particolari conversazioni fra determinati partner.

Altra preoccupazione legata al traffico è l'uso degli schemi di traffico per creare un **canale nascosto**. Un canale nascosto è un modo di comunicare non previsto dai progettisti del sistema di comunicazione; in genere viene utilizzato per trasferire informazioni che violano la politica di sicurezza. Per esempio, un dipendente potrebbe voler comunicare informazioni a un estraneo senza essere rilevato sfruttando semplicemente una intercettazione da parte del committente esterno. I due partecipanti potrebbero definire un codice in cui un messaggio apparentemente legittimo e minore di una certa lunghezza rappresenta uno "0" binario mentre un messaggio più lungo rappresenta un "1" binario. Naturalmente sono possibili molti altri schemi.

Approccio a crittografia di canale

Grazie all'uso della crittografia di canale, le intestazioni del livello rete (per esempio le intestazioni dei frame o delle celle) vengono crittografate riducendo le possibilità di analizzare il traffico. Tuttavia anche in queste circostanze è comunque possibile valutare il volume di traffico in rete osservando quanto traffico entra ed esce da ciascun sistema terminale. Un'efficace contromisura rispetto a questo attacco è la tecnica Traffic Padding, illustrata nella Figura 7.6.

[5] In realtà la figura mostra una sola alternativa. È anche possibile crittografare una parte o anche tutta l'intestazione e la coda del pacchetto tranne i flag di inizio e fine frame.

La tecnica Traffic Padding produce un output cifrato in modo continuo, anche in assenza di testo in chiaro. Dunque viene generato un flusso casuale continuo di dati. Quando è disponibile del testo in chiaro, questo viene crittografato e trasmesso. Ciò impedisce a un estraneo di distinguere il traffico vero e proprio dal traffico casuale di riempimento e dunque di determinare il volume del traffico.

Approccio a crittografia end-to-end

La tecnica Traffic Padding è fondamentalmente una funzionalità di crittografia di canale. Se viene impiegata solamente la crittografia end-to-end, le misure di difesa disponibili sono più limitate. Per esempio, se la crittografia viene implementata a livello applicazione, un estraneo potrà determinare quali entità di trasporto sono impegnate nel dialogo. Se le tecniche di crittografia sono ospitate a livello di trasporto, gli indirizzi del livello di rete e gli schemi di traffico rimarranno comunque accessibili.

Una tecnica che può dimostrarsi utile consiste nel riempire le unità di dati per ottenere una lunghezza uniforme a livello trasporto o di applicazione. Inoltre nel flusso di dati possono essere inseriti in modo casuale dei messaggi nulli. Queste tattiche impediscono a un estraneo di ottenere informazioni sul volume di dati scambiati fra gli utenti finali e pertanto celano lo schema di traffico reale.

7.3 Distribuzione delle chiavi

Per il funzionamento della crittografia simmetrica, le due parti devono condividere la stessa chiave che deve essere protetta da qualsiasi accesso non autorizzato. Inoltre è opportuno cambiare frequentemente le chiavi per limitare il volume dei dati che possono essere vio-

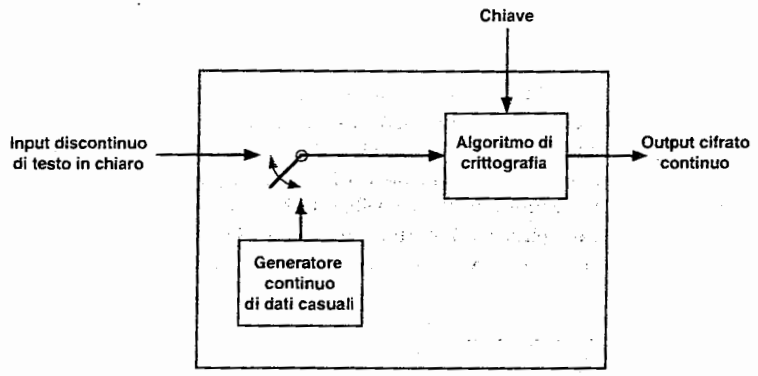


Figura 7.6 Un dispositivo di crittografia Traffic Padding.

lati nel caso in cui un estraneo dovesse scoprire la chiave. Pertanto la potenza di qualsiasi sistema crittografico dipende dalla *tecnica utilizzata per distribuire la chiave*, ovvero dalla tecnica di consegna di una chiave a due parti che vogliono scambiarsi dati evitando che possa essere intercettata. Per due parti A e B, la distribuzione della chiave può essere ottenuta in vari modi.

1. A può scegliere una chiave e consegnarla fisicamente a B.
2. Una parte "terza" può scegliere la chiave e consegnarla fisicamente ad A e B.
3. Se A e B hanno in precedenza e recentemente utilizzato una chiave, una parte può trasmettere all'altra parte la nuova chiave crittografandola con la vecchia chiave.
4. Se A e B hanno una connessione crittografata con una parte terza C, quest'ultima può consegnare una chiave sui collegamenti crittografati sia ad A che a B.

Le opzioni 1 e 2 richiedono la consegna manuale di una chiave. Per la crittografia di canale è un requisito ragionevole in quanto ciascun dispositivo di crittografia si troverà a scambiare dati solo con il partner che si trova all'altra estremità del canale di comunicazione. Al contrario, per la crittografia end-to-end, la consegna manuale può rappresentare un problema. In un sistema distribuito, qualsiasi host o terminale può dover scambiare dati con molti altri host e terminali. Pertanto ciascun dispositivo richiederà la fornitura dinamica di una grande quantità di chiavi. Il problema è particolarmente difficoltoso nel caso di sistemi distribuiti che occupano aree di grandi dimensioni.

La gravità del problema dipende dal numero di coppie in comunicazione che devono essere supportate. Se la crittografia end-to-end viene eseguita a livello rete o IP, sarà necessaria una chiave per ciascuna coppia di host della rete che ha bisogno di comunicare. Pertanto, se vi sono N host, il numero di chiavi richieste sarà $[N(N-1)]/2$. Se la crittografia viene eseguita a livello applicazione, sarà necessaria una chiave per ogni coppia di utenti o processi che richiedono l'attivazione di comunicazioni. Una rete può contare centinaia di host ma migliaia di utenti e processi. La Figura 7.7 illustra l'ordine di grandezza dell'operazione di distribuzione delle chiavi per la crittografia end-to-end.[6] Una rete che utilizzi la crittografia a livello dei nodi con 1000 nodi dovrà presumibilmente distribuire circa mezzo milione di chiavi.

Se la stessa rete supportasse 10 000 applicazioni sarebbero necessarie 50 milioni di chiavi per poter eseguire una crittografia a livello applicazione. Tornando alle possibilità di distribuzione delle chiavi elencate, l'opzione 3 può essere impiegata sia per la crittografia di canale che per quella end-to-end ma se un estraneo dovesse riuscire a ottenere l'accesso a una chiave, individuerebbe anche tutte le chiavi successive. Inoltre occorre comunque prevedere la distribuzione iniziale di, potenzialmente, milioni di chiavi.

Per la crittografia end-to-end sono ampiamente adottate alcune varianti dell'opzione 4. In questo caso, vi è un centro responsabile della distribuzione delle chiavi alle coppie di utenti (host, processi e applicazioni) in base alle richieste. Per garantire la distribuzione delle chiavi, ciascun utente deve condividere una chiave univoca con il centro di distribuzione delle chiavi.

[6] Si noti che questa figura usa una scala logaritmica e dunque una linea retta indica una crescita esponenziale. Le scale logaritmiche sono velocemente trattate nel documento math refresher disponibile all'indirizzo Web: www.williamStallings.com/StudentSupport.html.

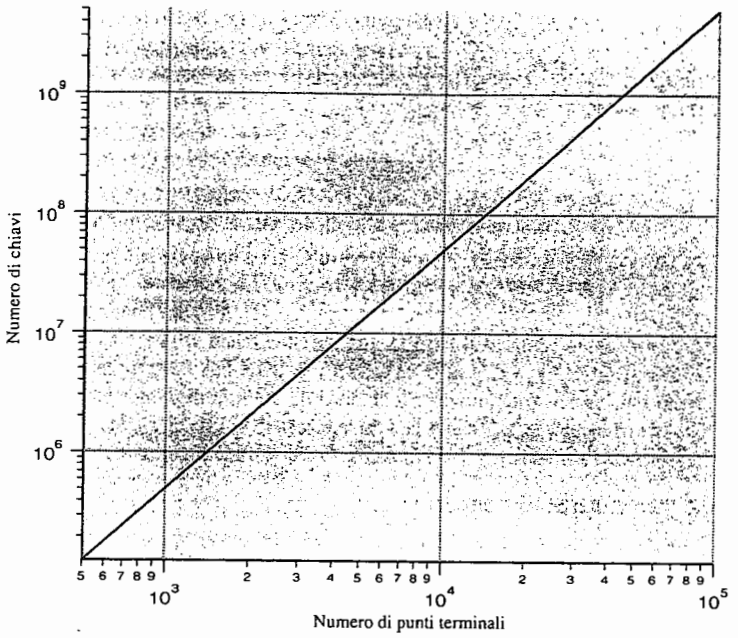


Figura 7.7 Numero di chiavi necessarie per supportare un numero di connessioni arbitrarie fra punti terminali.

L'adozione di un centro di distribuzione delle chiavi si basa sull'uso di una gerarchia di chiavi. Come minimo vengono utilizzati due livelli di chiavi (Figura 7.8). Le comunicazioni fra i sistemi terminali vengono crittografate utilizzando una chiave temporanea, spesso chiamata **chiave di sessione**. In genere, la chiave di sessione viene utilizzata per la sola durata di una connessione logica come per esempio una connessione Frame Relay o una connessione di livello, e quindi viene eliminata. Ciascuna chiave di sessione viene ottenuta dal centro di distribuzione delle chiavi utilizzando gli stessi sistemi di rete utilizzati per le comunicazioni fra utenti finali. Di conseguenza le chiavi di sessione vengono trasmesse in forma crittografata utilizzando una **chiave master** condivisa dal centro di distribuzione delle chiavi e dal sistema terminale o dall'utente.

Per ciascun sistema terminale o utente vi è un'unica chiave master condivisa con il centro di distribuzione delle chiavi. Naturalmente queste chiavi master devono essere distribuite in qualche modo. Tuttavia questa tecnica riduce enormemente l'entità del problema. Se vi sono N entità che vogliono comunicare fra loro a coppie, come si è detto, sarà necessario impiegare $[N(N-1)]/2$ chiavi di sessione. Al contrario sono necessarie solo N chiavi master, una per ogni entità. Pertanto le chiavi master possono essere distribuite in modo non crittografico, per esempio tramite consegna fisica.

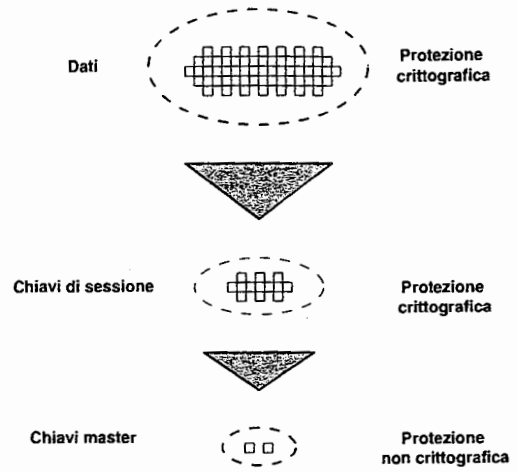


Figura 7.8 L'uso di una gerarchia di chiavi.

Un esempio di distribuzione delle chiavi

Il concetto di distribuzione delle chiavi può essere sviluppato in vari modi. La Figura 7.9 rappresenta un esempio che si basa su una figura contenuta in [POPE79]. Si suppone che ciascun utente condivida con il centro di distribuzione delle chiavi (KDC, Key Distribution Center) una chiave master univoca.

Si supponga che l'utente A voglia stabilire una connessione logica con l'utente B e richieda una chiave mono-sessione per proteggere i dati trasmessi lungo la connessione. A è in possesso di una chiave master, K_a, nota solo a lui e al KDC. Analogamente, B condivide la chiave master K_b con il KDC. Si svolgono le seguenti operazioni.

1. A richiede al KDC una chiave di sessione per proteggere la connessione logica con B. Il messaggio include l'identità di A e B e un identificatore univoco N_1 per questa transazione, chiamato **nonce**[7]. Il nonce può essere un'indicazione oraria, un contatore o un numero casuale. Il requisito minimo è che sia differente a ogni richiesta. Inoltre, per impedire il mascheramento, dovrà essere difficile da indovinare da parte di un estraneo. Pertanto anche un numero casuale è una buona scelta.
2. Il KDC risponde con un messaggio crittografato con K_a. Pertanto, A sarà l'unico utente in grado di leggere il messaggio e contemporaneamente A saprà che il messaggio ha avuto origine dal KDC. Il messaggio include due elementi per A.
   - La chiave mono-sessione K_s da utilizzare solo per questa sessione.

[7] Nella lingua inglese, una *nonce word* indica un termine coniato, inventato o comunque utilizzato una sola volta in una particolare occasione.

   - Il messaggio di richiesta originale, compreso il nonce, per consentire ad A di associare la risposta alla richiesta appropriata.

   Pertanto, A può verificare che la sua richiesta originaria non è stata modificata prima della ricezione da parte del KDC e, grazie alla presenza del nonce, che non si tratta di una risposta a una richiesta precedente.

   Inoltre il messaggio include due elementi per B.

   - La chiave mono-sessione K_s da utilizzare solo per questa sessione.
   - Un identificatore per A (per esempio il suo indirizzo di rete), ID_A.

   Questi ultimi due elementi sono crittografati con K_b (la chiave master che il KDC condivide con B). Questi valori devono essere inviati a B per stabilire la connessione e dimostrare l'identità di A.
3. A memorizza la chiave di sessione da utilizzare per la sessione seguente e inoltra a B le informazioni prodotte dal KDC per B, in particolare $E(K_b,[K_s \| ID_A])$. Poiché queste informazioni sono crittografate utilizzando K_b, saranno protette da ogni intercettazione. Ora B conosce la chiave di sessione (K_s), sa che all'altro capo si trova A (da ID_A) e sa che le informazioni hanno avuto origine dal KDC (in quanto sono crittografate con K_b).

   A questo punto è stata consegnata una chiave di sessione sicura sia ad A che a B e dunque le due parti possono iniziare il loro scambio di informazioni protetto. Tuttavia è opportuno svolgere anche altre due operazioni.
4. Utilizzando la nuova chiave di sessione per la crittografia, B invia ad A un nonce, N_2.
5. Sempre utilizzando K_s, A risponde con $f(N_2)$ dove f è una funzione che svolge una trasformazione su N_2 (per esempio somma una unità).

Queste operazioni garantiscono a B che il messaggio originario ricevuto (passo 3) non sia una replica.

Si noti che la distribuzione della chiave comporta solo i passi da 1 a 3 e che i passi 4, 5 e 3 svolgono una funzione di autenticazione.

Controllo gerarchico delle chiavi

Non è necessario limitare la funzione di distribuzione delle chiavi a un unico KDC. Infatti, per reti di grandi dimensioni può essere scomodo utilizzare questa tecnica. In alternativa, si può definire una gerarchia di centri di distribuzione delle chiavi. Per esempio vi possono essere dei centri locali di distribuzione delle chiavi, ognuno dei quali è responsabile solo di un piccolo dominio dell'intera rete, per esempio di una rete locale o di un edificio. Per comunicazioni fra entità dello stesso dominio locale, la distribuzione delle chiavi verrà effettuata dal KDC locale. Se invece devono comunicare due entità in domini differenti, i rispettivi KDC locali potranno comunicare con un KDC globale. In questo caso, la chiave potrà essere selezionata da uno qualsiasi dei tre KDC coinvolti. Il concetto gerarchico può essere esteso a tre o più livelli a seconda delle dimensioni della popolazione di utenti e dell'ampiezza geografica della rete.

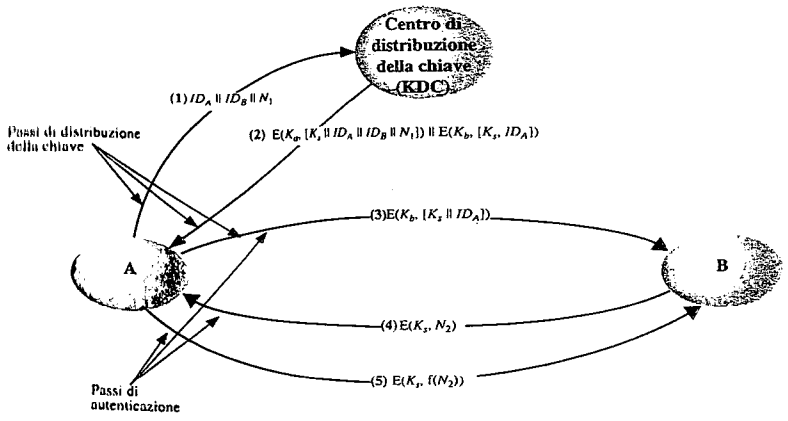


Figura 7.9 Distribuzione della chiove.

Uno schema gerarchico riduce l'impegno di distribuzione della chiave master in quanto la maggior parte delle chiavi master sarà condivisa da un KDC locale con le rispettive entità locali. Inoltre tale schema limita i danni provocati da un guasto o da un attacco al KDC alla sua sola area locale.

Durata di una chiave di sessione

Maggiore è la frequenza con cui si scambiano le chiavi di sessione e maggiore sarà la sicurezza delle chiavi, in quanto un eventuale intruso potrà lavorare su meno testo cifrato prodotto dalla stessa chiave di sessione. Ma d'altra parte, la distribuzione delle chiavi di sessione ritarda l'avvio di ogni operazione di scambio e rappresenta un carico per la capacità della rete. Il responsabile della sicurezza deve tentare di bilanciare queste considerazioni per determinare la durata ottimale di una chiave di sessione.

Per i protocolli orientati alla connessione, una scelta ovvia consiste nell'utilizzare una chiave di sessione solo per la durata della connessione e utilizzare una nuova chiave per ogni nuova sessione. Se una connessione logica ha una durata molto lunga, può essere prudente cambiare periodicamente la chiave di sessione, per esempio a ogni ciclo del numero di sequenza delle PDU (Protocol Data Unit).

Per un protocollo senza connessione, per esempio un protocollo orientato alla transazione, non esiste alcuna attivazione o terminazione esplicita della connessione. Pertanto può non essere facile stabilire quando cambiare la chiave di sessione. L'approccio più sicuro consiste nell'utilizzare una nuova chiave di sessione per ogni nuovo scambio. Tuttavia questo nega uno dei principali vantaggi dei protocolli senza connessione, ovvero un carico e un ritardo minimi per ciascuna transazione. Una strategia migliore consiste nel-

l'utilizzare una determinata chiave di sessione per un periodo di tempo fissato o per un determinato numero di transazioni.

Uno schema trasparente di controllo della chiave

L'approccio suggerito nella Figura 7.9 presenta numerose varianti, una delle quali è descritta in questa parte del capitolo. Lo schema rappresentato nella Figura 7.10 è utile per garantire una crittografia end-to-end a livello rete o trasporto in modo completamente trasparente agli utenti finali. L'approccio presuppone che le comunicazioni impieghino un protocollo end-to-end orientato alla connessione, come TCP. L'elemento più importante di questo approccio è il modulo di sicurezza della sessione (SSM o Session Security Module) che può essere costituito da funzionalità a un particolare livello dello stack di protocolli, che svolge la crittografia end-to-end e ottiene le chiavi di sessione per conto del proprio host o terminale. La Figura 7.10 mostra le operazioni richieste per stabilire una connessione. Quando un host desidera stabilire una connessione con un altro host, trasmette un pacchetto di richiesta della connessione (passo 1). L'SSM salva tale pacchetto e richiede al KDC il permesso di stabilire la connessione (passo 2). Le comunicazioni fra l'SSM e il KDC sono crittografate tramite una chiave master condivisa solo dall'SSM e dal KDC. Se il KDC approva la richiesta di connessione, genera una chiave di sessione e la consegna agli SSM appropriati utilizzando una chiave univoca e permanente per ciascuna interfaccia (passo 3). L'SSM richiedente può ora rilasciare il pacchetto di richiesta di connessione e quindi viene stabilita una connessione fra i due sistemi terminali (passo 4). Tutti i dati utente scambiati dai due sistemi terminali sono crittografati dai rispettivi SSM utilizzando la chiave monosessione.

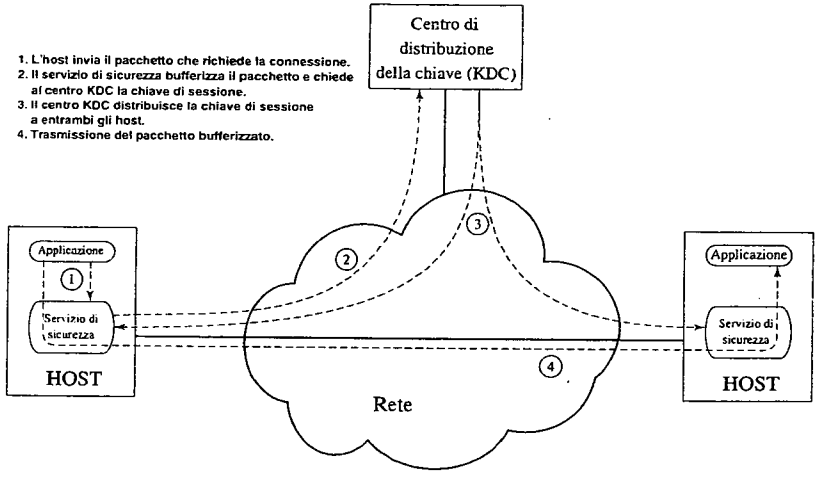


Figura 7.10 Distribuzione automatica della chiave per un protocollo orientato alla connessione.

La tecnica di distribuzione automatica della chiave offre le caratteristiche di flessibilità e dinamica necessarie per consentire a numerosi utenti terminali di accedere a numerosi host per l'interscambio dei dati.

Controllo della chiave decentralizzato

L'uso di un centro di distribuzione delle chiavi impone il requisito che il KDC sia fidato e protetto da ogni genere di attacchi. Questo requisito può essere evitato se la distribuzione delle chiavi è completamente decentralizzata. Sebbene una decentralizzazione completa non risulti pratica per reti di grandi dimensioni che utilizzano solo la crittografia simmetrica, può essere utile in un contesto locale.

L'approccio decentralizzato richiede che ciascun sistema terminale sia in grado di comunicare in modo sicuro con tutti i potenziali sistemi terminali con cui deve comunicare con lo scopo di distribuire in modo sicuro la chiave di sessione. Pertanto, per la configurazione di n sistemi terminali, vi possono essere fino a $[n(n-1)]/2$ chiavi master.

Una chiave di sessione può essere attivata utilizzando i seguenti passi (Figura 7.11).

1. A emette una richiesta a B di una chiave di sessione e include un nonce, N_1.
2. B risponde con un messaggio crittografato utilizzando la chiave master condivisa. La risposta include la chiave di sessione selezionata da B, un identificatore di B, il valore $f(N_1)$ e un altro nonce, N_2.
3. Utilizzando la nuova chiave di sessione, A restituisce a B il valore $f(N_2)$.

Pertanto, sebbene ciascun nodo debba mantenere un massimo di $n-1$ chiavi master, possono essere generate e utilizzate tutte le chiavi di sessione necessarie. Poiché i messaggi trasferiti utilizzando la chiave master sono brevi, l'analisi crittografica risulta difficoltosa. Come prima, per sicurezza le chiavi di sessione vengono utilizzate solo per un tempo limitato.

Controllo dell'utilizzo delle chiavi

Il concetto di gerarchia delle chiavi e l'uso di tecniche automatizzate di distribuzione delle chiavi riduce enormemente il numero di chiavi che devono essere gestite e distribuite manualmente. Può anche essere desiderabile imporre un certo controllo sul modo in cui vengono utilizzate le chiavi distribuite automaticamente. Per esempio, oltre a separare le chiavi master dalle chiavi di sessione, si potrebbero voler definire delle chiavi di sessione di tipo differente in base all'uso:

- chiavi di crittografia dei dati, per le comunicazioni generali in rete;
- chiavi di crittografia per i codici PIN (Personal Identification Numbers), utilizzabili nei trasferimenti di denaro e nelle applicazioni commerciali.
- chiavi per la crittografia dei file memorizzati in aree ad accesso pubblico.

Per illustrare il vantaggio della suddivisione delle chiavi in base al tipo, si consideri la possibilità che una chiave master venga importata in un dispositivo e utilizzata come chiave di crittografia dei dati. Normalmente la chiave master è fisicamente al sicuro nell'hardware del centro di distribuzione delle chiavi e dei sistemi terminali. Le chiavi di sessione crittografate con questa chiave master sono disponibili ai programmi applicativi, così come i dati crittografati con tali chiavi di sessione. Tuttavia, se una chiave master venisse trattata come una chiave di sessione, un'applicazione non autorizzata potrebbe ottenere (in chiaro) tutte le chiavi di sessione crittografate con questa chiave master.

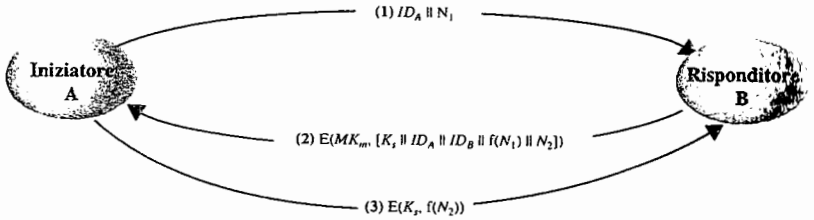


Figura 7.11 Distribuziane decentralizzata della chiave.

Pertanto può essere opportuno stabilire dei controlli nei sistemi in modo da limitare gli utilizzi delle chiavi sulla base delle caratteristiche delle chiavi stesse. Per esempio si può associare a ciascuna chiave un tag ([JONE82], vedere anche [DAVI89]). La tecnica proposta prevede l'utilizzo di DES e impiega gli 8 bit aggiuntivi di ciascuna chiave DES di 64 bit. In pratica, si tratta degli 8 bit normalmente riservati per il controllo di parità. Tali bit hanno il seguente significato:

- un bit indica se la chiave è una chiave di sessione o una chiave master;
- un bit indica se la chiave può essere utilizzata per la crittografia;
- un bit indica se la chiave può essere utilizzata per la decrittografia;
- i bit rimanenti sono riservati per utilizzi futuri.

Poiché il tag è incorporato nella chiave, viene crittografato insieme alla chiave nel momento in cui questa viene distribuita, offrendo quindi la protezione necessaria. I difetti di questo schema sono che (1) la lunghezza del tag è di soli 8 bit e ciò limita la sua flessibilità e funzionalità; (2) poiché il tag non viene trasmesso in chiaro, può essere utilizzato solo nel punto in cui i dati vengono de-crittografati e ciò limita i modi in cui è possibile controllare l'utilizzo della chiave.

Uno schema più flessibile, chiamato vettore di controllo, è descritto in [MATY91a e b]. In questo schema, a ciascuna chiave di sessione è associato un vettore di controllo costituito da campi che specificano gli utilizzi e i limiti di tale chiave di sessione. La lunghezza del vettore di controllo è variabile.

Il vettore di controllo è accoppiato crittograficamente alla chiave nel momento in cui la chiave stessa viene generata nel KDC. I processi di accoppiamento e disaccoppiamento sono illustrati nella Figura 7.12. Innanzitutto il vettore di controllo viene sottoposto a una funzione hash che produce un valore di lunghezza uguale alla chiave di crittografia. Le funzioni hash vengono discusse in dettaglio nel Capitolo 11: in pratica una funzione hash mappa i valori da un intervallo di maggiori dimensioni a un intervallo più piccolo, con una dispersione ragionevolmente uniforme. Pertanto, per esempio, se i numeri nell'intervallo

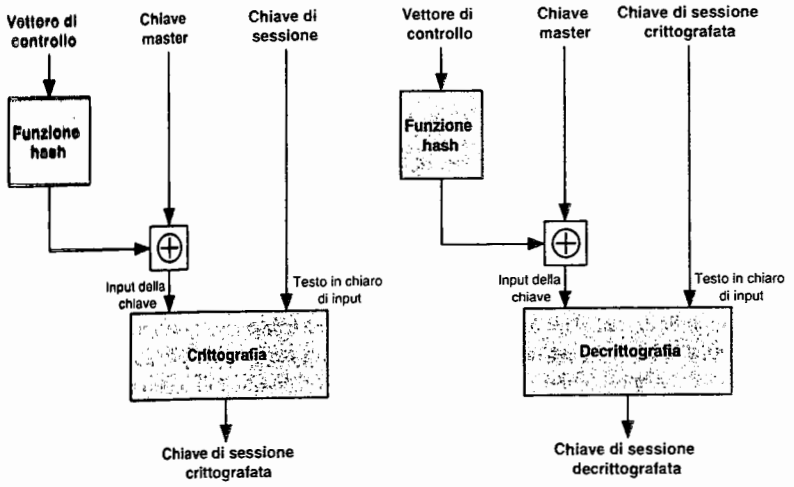


Figura 7.12 Crittografia e decrittografia a vettore di controllo.

da 1 a 100 dovessero essere mappati tramite una hash nell'intervallo da 1 a 10, a ciascuno dei valori di destinazione corrisponderà circa il 10% dei valori di origine.

Il valore hash subisce poi un'operazione di XOR con la chiave master per produrre un output che viene utilizzato come input per la crittografia della chiave di sessione. Pertanto:

$$\text{Valore hash} = H = \text{h}(CV)$$
$$\text{Input della chiave} = K_m \oplus H$$
$$\text{Testo cifrato} = E([K_m \oplus H], K_s)$$

dove K_m è la chiave master e K_s è la chiave di sessione. La chiave di sessione può essere ripristinata in chiaro dall'operazione inversa:

$$D([K_m \oplus H], E([K_m \oplus H], K_s))$$

Quando a un utente viene fornita una chiave di sessione da parte del centro KDC, tale chiave viene accompagnata dal vettore di controllo in chiaro. La chiave di sessione può essere ripristinata solo utilizzando sia la chiave master che l'utente condivide con il centro KDC che il vettore di controllo. Pertanto viene mantenuto il legame fra la chiave di sessione e il suo vettore di controllo.

L'uso del vettore di controllo presenta due vantaggi rispetto all'uso di un tag di 8 bit. Innanzitutto non vi sono limiti alla lunghezza del vettore di controllo: questo consente di imporre sull'utilizzo delle chiavi dei controlli arbitrariamente complessi. In secondo luogo il vettore di controllo è disponibile in chiaro in tutte le fasi: pertanto il controllo dell'utilizzo della chiave può essere esercitato in più punti.

7.4 Generazione di numeri casuali

I numeri casuali giocano un ruolo fondamentale nella crittografia. Questa parte del capitolo offre una breve panoramica sull'utilizzo dei numeri casuali nella sicurezza di rete e descrive alcuni approcci alla generazione di numeri casuali.

L'uso dei numeri casuali

I numeri casuali vengono utilizzati da numerosi algoritmi crittografici. Ecco alcuni esempi.

- Schemi di autenticazione reciproca, come quelli illustrati nelle Figure 7.9 e 7.11. In entrambe le modalità di distribuzione delle chiavi, per prevenire gli attacchi a ripetizione dei pacchetti vengono utilizzati dei numeri nonce. L'uso di numeri nonce casuali vanifica ogni tentativo di determinare o indovinare il numero.
- Generazione della chiave di sessione da parte di un centro di distribuzione delle chiavi o di uno dei sistemi di controllo.
- Generazione delle chiavi per l'algoritmo di crittografia RSA a chiave pubblica (descritto nel Capitolo 9).

Queste applicazioni danno origine a due requisiti distinti e non necessariamente compatibili fra loro per una sequenza di numeri casuali: la casualità e la imprevedibilità.

Casualità

Tradizionalmente, il problema della generazione di una sequenza di numeri dall'aspetto casuale consisteva nel fatto che la sequenza di numeri dovesse essere casuale in un senso statistico ben definito. Per verificare che una sequenza di numeri sia effettivamente casuale vengono utilizzati i due criteri seguenti.

- **Distribuzione uniforme:** la distribuzione dei numeri nella sequenza deve essere uniforme; in pratica ogni numero deve essere presente nella sequenza approssimativamente lo stesso numero di volte.
- **Indipendenza:** non deve essere possibile determinare un valore nella sequenza sulla base degli altri valori.

Mentre esistono test ben definiti per determinare che una sequenza di numeri rispetta una determinata distribuzione, per esempio una distribuzione uniforme, non esiste un test per dimostrare l'indipendenza. Piuttosto è possibile applicare un certo numero di test per dimostrare che una sequenza non manifesta indipendenza. La strategia generale consiste quindi nell'applicare un certo numero di tali test fino a raggiungere una relativa confidenza che esista indipendenza.

In questo contesto, nella progettazione degli algoritmi legati alla crittografia emerge spesso la necessità di usare una sequenza di numeri che sembra statisticamente casuale. Per esempio, un requisito fondamentale dello schema di crittografia a chiave pubblica RSA trattato nel Capitolo 9 è la capacità di generare numeri primi. In generale è difficile determinare se un determinato numero N di grandi dimensioni è primo. Un approccio a forza bruta prevede la divisione di N per ogni intero dispari minore di $\sqrt{N}$. Ma se N è, per

esempio, dell'ordine di grandezza di 10^{150}, una situazione piuttosto comune nella crittografia a chiave pubblica, l'approccio a forza bruta oltrepasserebbe i limiti degli analisti umani e dei loro computer. Tuttavia esistono vari algoritmi in grado di verificare se un numero è primo utilizzando una sequenza di interi casuali e calcoli relativamente semplici. Se la sequenza è sufficientemente lunga (ma molto, molto inferiore a $\sqrt{10^{150}}$) sarà possibile determinare se un numero è primo con una buona sicurezza. Questo tipo di approccio, chiamato casualizzazione, si presenta frequentemente nella progettazione degli algoritmi. In pratica, se un problema è troppo difficile o richiede troppo tempo per una soluzione esatta, si può usare un approccio più semplice basato sulla casualizzazione, in grado di fornire la risposta con il grado di confidenza desiderato.

Imprevedibilità

Nelle applicazioni quali l'autenticazione reciproca e la generazione di chiavi di sessione, il requisito non è tanto che la sequenza di numeri sia statisticamente casuale ma che i membri successivi della sequenza siano imprevedibili. Utilizzando sequenze "veramente" casuali, ciascun numero è statisticamente indipendente dagli altri numeri della sequenza e pertanto è imprevedibile. Tuttavia, come si vedrà fra breve, i numeri veramente casuali vengono utilizzati molto raramente; al loro posto si usano sequenze di numeri che sembrano casuali ma che in realtà sono generati da un algoritmo. In quest'ultimo caso occorre assicurarsi che non sia possibile prevedere gli elementi successivi della sequenza sulla base degli elementi precedenti.

Generatori di numeri pseudo casuali

Le applicazioni crittografiche fanno normalmente uso di tecniche algoritmiche per la generazione di numeri casuali. Questi algoritmi sono deterministici e pertanto producono sequenze di numeri che non sono statisticamente casuali. Tuttavia, se l'algoritmo è di buona qualità, le sequenze prodotte passeranno numerosi test di casualità ragionevoli. In questo caso si parla di **numeri pseudocasuali** e di relativi generatori (PRNG, Pseudo Random Numbers Generator).

Si potrebbe essere piuttosto scettici sulla possibilità di utilizzare numeri generati da un algoritmo deterministico come se si trattasse di numeri casuali. Nonostante le obiezioni di natura filosofica all'adozione di una tecnica di questo tipo, in realtà le cose generalmente funzionano. Ecco cosa ha scritto un esperto di teoria delle probabilità [HAMM91]:

Ai fini pratici, si è costretti ad accettare il concetto di "casualità relativa" intendendo con ciò il fatto che, per l'utilizzo previsto, la sequenza di numeri si comporta in modo casuale. Si tratta di un argomento molto soggettivo e poco accettabile per i puristi ma si tratta di ciò che gli statistici usano regolarmente quando prendono un "campione casuale": sperano che i risultati così ottenuti abbiano approssimativamente le stesse proprietà di un conteggio completo dell'intero spazio di campionamento.

Generatori a congruenza lineare

La tecnica di gran lunga più utilizzata per la generazione di numeri pseudocasuali (indicati in alcuni casi semplicemente come "numeri casuali") è un algoritmo proposto inizialmente da Lehmer [LEHM51] chiamato metodo di congruenza lineare. L'algoritmo utilizza quattro parametri numerici:

m	il modulo	$m > 0$
a	il moltiplicatore	$0 < a < m$
c	l'incremento	$0 \le c < m$
X_0	il valore iniziale o seme	$0 \le X_0 < m$

La sequenza di numeri casuali $\{X_n\}$ si ottiene tramite la seguente equazione iterativa:

$$X_{n+1} = (aX_n + c) \bmod m$$

Se m, a, c e X_0 sono interi, questa tecnica produrrà una sequenza di interi compresi fra $0 \le X_n < m$.

La scelta dei valori per a, c e m è fondamentale per sviluppare un buon generatore di numeri casuali. Per esempio, si consideri $a = c = 1$. La sequenza prodotta sarà ovviamente insoddisfacente. Ora si considerino i valori $a = 7$, $c = 0$, $m = 32$ e $X_0 = 1$. Questi parametri generano la sequenza {7, 17, 23, 1, 7 ecc.} anch'essa chiaramente insoddisfacente. Dei 32 valori possibili ne vengono usati solo 4; pertanto si dice che la sequenza ha un periodo uguale a 4. Se però si cambia il valore a in 5, allora si ottiene la sequenza {5, 25, 29, 17, 21, 9, 13, 1 ecc.}, che incrementa il periodo a 8.

Si vorrebbe che m fosse molto esteso in modo da produrre una lunga serie di numeri casuali distinti. Un criterio comunemente utilizzato è che m sia approssimativamente uguale al massimo intero non negativo rappresentabile in un computer. Pertanto viene normalmente scelto un valore di m prossimo o uguale a 2^{31}.

[PARK88] propone tre test per valutare un generatore di numeri casuali.

T_1: la funzione dovrebbe generare l'intero periodo. Ovvero, prima di ripetersi, la funzione dovrebbe generare tutti i numeri compresi fra 0 e m.

T_2: la sequenza generata dovrebbe sembrare casuale. Poiché viene generata in modo deterministico, la sequenza non è casuale. Vi sono vari test statistici che consentono di valutare il grado di casualità di una sequenza.

T_3: la funzione dovrebbe essere implementabile in modo efficiente nell'aritmetica a 32 bit.

Con valori appropriati di a, c e m, è possibile soddisfare questi test. Rispetto a T_1, si può mostrare che se m è primo e $c = 0$, allora per determinati valori di a il periodo della funzione di generazione è $m - 1$ con l'assenza del solo valore 0. Per l'aritmetica a 32 bit, un valore di m primo conveniente è $2^{31} - 1$. Pertanto la funzione di generazione diviene:

$$X_{n+1} = (aX_n) \bmod (2^{31} - 1)$$

Dei più di due miliardi di possibili scelte di a, solo pochi moltiplicatori passano i tre test appena elencati. Uno di questi valori è $a = 7^5 = 16807$ che è stato originariamente proget-

tato per l'utilizzo nella famiglia di computer IBM 360 [LEW169]. Questo generatore è ampiamente utilizzato ed è stato soggetto a test più approfonditi rispetto a qualsiasi altro generatore di numeri casuali. Viene frequentemente consigliato per operazioni statistiche e di simulazione (vedere [JAIN91] e [SAUE81]).

La potenza dell'algoritmo a congruenza lineare è tale che se il moltiplicatore e il modulo vengono scelti in modo appropriato, la sequenza di numeri prodotta sarà statisticamente indistinguibile da una sequenza veramente casuale (senza però riutilizzare i valori) nell'insieme 1, 2, ... , $m - 1$. Ma nell'algoritmo non vi è niente di casuale se non la scelta del valore iniziale X_0. Una volta scelto tale valore, i numeri seguono deterministicamente la sequenza prevista. Questo comporta delle conseguenze in termini di analisi crittografica.

Se un estraneo sapesse che viene utilizzato l'algoritmo a congruenza lineare e conoscesse i parametri (per esempio $a = 7^5$, $c = 0$, $m = 2^{31} - 1$), una volta scoperto un singolo numero conoscerebbe anche tutti i numeri successivi. Anche se l'estraneo sapesse solo che viene utilizzato un algoritmo a congruenza lineare, basterebbe conoscere una piccola parte della sequenza per determinare i parametri dell'algoritmo. Si supponga che l'estraneo possa determinare i valori di X_0, X_1, X_2 e X_3. Allora:

$$X_1 = (aX_0 + c) \bmod m$$
$$X_2 = (aX_1 + c) \bmod m$$
$$X_3 = (aX_2 + c) \bmod m$$

Queste equazioni possono essere risolte per a, c e m.

Pertanto, sebbene sia comodo utilizzare un buon generatore di numeri casuali, è desiderabile fare in modo che la sequenza utilizzata non sia riproducibile in modo che la conoscenza di una parte della sequenza non consenta di determinare elementi successivi. Questo obiettivo può essere ottenuto con varie strategie. Per esempio, [BRIG79] suggerisce di impiegare l'orologio di sistema interno per modificare il flusso di numeri pseudo casuali. Un modo per utilizzare l'orologio consiste nel riavviare la sequenza ogni N numeri utilizzando come seme il valore corrente dell'orologio (mod m). Un altro modo consiste semplicemente nell'aggiungere a ciascun numero pseudo casuale il valore corrente dell'orologio (mod m).

Numeri casuali generati in modo crittografico

Nelle applicazioni crittografiche, può essere conveniente produrre numeri casuali sfruttando la logica di crittografia disponibile. Sono stati utilizzati vari metodi e in questa parte del capitolo si vedranno tre esempi rappresentativi.

Crittografia ciclica

La Figura 7.13 illustra un approccio suggerito in [MEYE82]. In questo caso, la procedura viene utilizzata per generare chiavi di sessione a partire da una chiave master. L'input della logica di crittografia è fornito da un contatore con periodo N. Per esempio, se devono essere prodotte chiavi DES a 56 bit, si può utilizzare un contatore con periodo 2^{56}. Dopo le

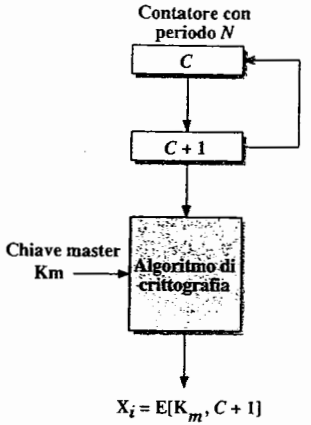


Figura 7.13 Generazione di numeri pseudocasuali tramite un contatore.

operazioni di ogni chiave, il contatore viene incrementato di una unità. Pertanto i numeri pseudocasuali prodotti da questo schema coprono l'intero periodo; ciascuno degli output $X_0, X_1, \ldots X_{N-1}$ si basa su un valore differente dal contatore, quindi $X_0 \neq X_1 \neq \ldots \neq X_{N-1}$. Poiché la chiave master è protetta, non è possibile dedurre le chiavi di sessione (numeri casuali) dalla conoscenza di una o più chiavi di sessione precedenti.

Per rafforzare ulteriormente l'algoritmo, l'input potrebbe essere prodotto da un generatore di numeri casuali dello stesso periodo invece che da un semplice contatore.

La modalità output feedback di DES

La modalità output feedback (OFB) di DES, illustrata nella Figura 6.6, consente di generare delle chiavi oltre che crittografare flussi. Si noti che l'output di ciascuna fase dell'operazione è un valore a 64 bit, di cui vengono utilizzati gli s bit più a sinistra come feedback dell'algoritmo. La successione di output di 64 bit costituisce una sequenza di numeri pseudocasuali con buone proprietà statistiche. Ancora una volta, come nell'approccio suggerito in precedenza, le chiavi di sessione generate sono protette tramite una chiave master.

Il generatore di numeri casuali ANSI X9.17

Uno dei generatori di numeri casuali più forti (dal punto di vista dell'analisi crittografica) è specificato nel documento ANSI X9.17. Vi sono varie applicazioni che adottano questa tecnica, fra cui applicazioni finanziarie e PGP (se ne parlerà nel Capitolo 15).

La Figura 7.14 illustra l'algoritmo, che impiega triple DES per la crittografia. Eccone gli elementi utilizzati.

- **Input:** il generatore usa come input due numeri pseudocasuali. Uno è una rappresentazione a 64 bit della data e dell'ora correnti. L'altro è il seme a 64 bit, inizializzato con un valore arbitrario e aggiornato durante il processo di generazione.

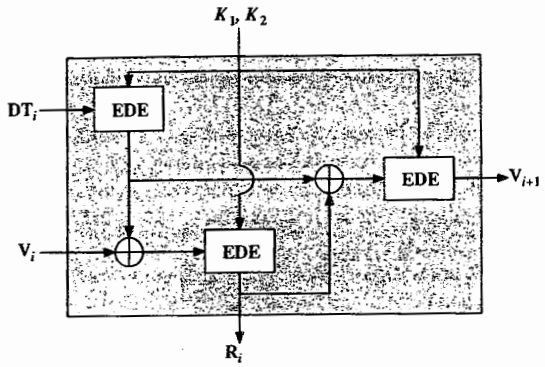


Figura 7.14 Generazione di numeri pseudocasuali ANSI X9.17.

- **Chiavi:** il generatore utilizza tre moduli di crittografia triple DES. Questi tre moduli utilizzano tutti la stessa coppia di chiavi a 56 bit che deve essere mantenuta segreta e utilizzata solo per la generazione di numeri pseudocasuali.
- **Output:** l'output è costituito da un numero pseudocasuale di 64 bit e un seme a 64 bit.

Si definiscano le seguenti quantità.

DT_i	Valore di data/ora all'inizio della *i*–esima fase di generazione.
V_i	Valore del seme all'inizio della *i*-esima fase di generazione.
R_i	Numero pseudocasuale prodotto dalla *i*-esima fase di generazione.
K_1, K_2	Chiavi DES utilizzate per ciascuna fase.

Allora:

$$R_i = EDE([K_1, K_2],[V_i \oplus EDE([K_1, K_2], DT_i)])$$
$$V_{i+1} = EDE([K_1, K_2],[R_i \oplus EDE([K_1, K_2], DT_i)])$$

dove EDE($[K_1, K_2]$, X) fa riferimento alla sequenza di crittografia-decrittografia-crittografia (Encrypt-Decrypt-Encrypt) con l'algoritmo triple DES a due chiavi, per crittografare X.

Vi sono vari fattori che contribuiscono alla potenza crittografica di questo metodo. La tecnica prevede l'impiego di una chiave a 112 bit e di tre crittografie EDE per un totale di nove crittografie DES. Lo schema è controllato da due input pseudocasuali, da un lato la data e l'ora e dall'altro un seme prodotto dal generatore e distinto dal numero pseudocasuale prodotto. Pertanto, per aver ragione dell'algoritmo, un estraneo dovrebbe riuscire a violare un'enorme quantità di materiale. Anche se un numero pseudocasuale R_i venisse violato, sarebbe impossibile determinare V_{i+1} da R_i, dato che per produrre V_{i+1} viene utilizzata un'ulteriore operazione EDE.

Il generatore Blum Blum Shub

BBS (Blum Blum Shub), che trae il proprio nome dai suoi sviluppatori [BLUM86], è un algoritmo molto noto per la generazione di numeri sicuri pseudocasuali. Questo algoritmo ha offerto la prova pubblica forse più forte di potenza crittografica. Viene utilizzata la seguente procedura. Innanzitutto si scelgono due numeri primi estesi, p e q, tali che, se divisi per 4, abbiano un resto 3, ovvero:

$$p \equiv q \equiv 3 (\text{mod} 4)$$

Questa notazione, descritta in modo esteso nel Capitolo 4, significa semplicemente che (p mod 4) = (q mod 4) = 3. Per esempio, i numeri primi 7 e 11 soddisfano la relazione $7 \equiv 11 \equiv 3 \pmod 4$. Si supponga che $n = p \times q$. Si sceglie un numero casuale s tale che s sia primo relativo rispetto a n; ciò equivale a dire che né p né q siano fattori di s. Quindi il generatore BBS produce una sequenza di bit B_i sulla base del seguente algoritmo:

$X_0 = s^2 \bmod n$
for $i = 1$ to ∞
  $X_i = (X_{i-1})^2 \bmod n$
  $B_i = X_i \bmod 2$

Pertanto a ciascuna iterazione viene preso il bit meno significativo. La Tabella 7.2 mostra un esempio di funzionamento dell'algoritmo BBS con n = 192 649 = 383 × 503 e seme s = 101 355.

L'algoritmo BBS è un **generatore di bit pseudocasuali sicuro** dal punto di vista crittografico; in pratica un algoritmo di questo tipo deve passare il cosiddetto test del prossimo bit che è definito nel seguente modo [MENE97]: "Un generatore di bit pseudocasuali passa il test del prossimo bit se non esiste un algoritmo di complessità polinomiale[8] tale che, utilizzando i primi quattro k bit di una sequenza di output possa prevedere il $(k + 1)^{esimo}$ bit con probabilità maggiore di ½". In altre parole, dati i primi k bit della sequenza, non esiste un algoritmo realistico in grado di stabilire che il prossimo bit sarà un 1 (o uno 0) con una probabilità maggiore di ½. Ai fini pratici, la sequenza risulta imprevedibile. La sicurezza di BBS si basa sulla difficoltà di fattorializzare n. Ovvero, dato n, occorre determinare i suoi due fattori primi p e q.

Tabella 7.2 Esempio di funzionamento del generatore BBS.

i	X_i	B_i	i	X_i	B_i
0	20749		2	177671	1
1	143135	1	3	97048	0

(segue)

[8] Un algoritmo di complessità computazionale polinomiale di ordine k è un algoritmo il cui tempo di esecuzione è limitato da un polinomio di ordine k.

Tabella 7.2 Esempio di funzionamento del generatore BBS. *(continua)*

i	X_i	B_i	i	X_i	B_i
4	89992	0	13	8630	0
5	174051	1	14	114386	0
6	80649	1	15	14863	1
7	45663	1	16	133015	1
8	69442	0	17	106065	1
9	186894	0	18	45870	0
10	177046	0	19	137171	1
11	137922	0	20	48060	0
12	123175	1			

I generatori di numeri realmente casuali

Un generatore di numeri realmente casuali (TRNG, o True Random Number Generator) utilizza una fonte non deterministica per produrre la casualità. La maggior parte funziona campionando processi naturali non prevedibili quali rilevatori di impulsi di radiazioni ionizzanti, scariche in tubi a gas e in condensatori. Intel ha sviluppato e commercializza un circuito integrato che campiona il rumore termico amplificando la tensione misurata ai capi di una resistenza [JUN99]. Un gruppo dei Bell Labs ha sviluppato una tecnica che utilizza le variazioni nel tempo di risposta delle richieste di lettura di un settore di un disco rigido [JAKO98]. LavaRnd è un progetto open source per creare numeri realmente casuali che utilizza economiche macchine fotografiche, codice open source e hardware economico. Questo sistema utilizza un CCD saturato in una lattina illuminata come sorgente casuale per produrre il seme. Il software trasforma il risultato in numeri realmente casuali in vari formati.

Vi sono problemi sia nella casualità che nella precisione di tali numeri [BRIG79], per non dire del fatto che sarebbe necessario connettere uno di questi dispositivi a ogni sistema della rete. Un'alternativa consiste nell'utilizzare una delle numerose sequenze di numeri casuali pubblicate (per esempio [RAND55], [TIPP27]). Tuttavia queste sequenze rappresentano una fonte di numeri molto limitata rispetto ai potenziali requisiti di un'applicazione per la sicurezza di una rete di dimensioni non banali. Inoltre, sebbene i numeri contenuti in questi volumi esibiscano caratteristiche di casualità statistica, essi sono prevedibili poiché un estraneo che sappia che viene impiegato un certo volume potrà ottenerne una copia.

Skew

Un TRNG può produrre risultati potenzialmente disomogenei, per esempio con più cifre uno che zero o viceversa. Per ridurre o eliminare questo problema sono stati sviluppati vari metodi, chiamati **algoritmi di deskew**. Uno di questi metodi consiste nell'elaborare il flusso di bit con una funzione hash quale MD5 o SHA-1 (descritti nella seconda parte di questo volume). La funzione hash produce un output di n bit da un input di lunghezza arbitraria. Per ottenere il risultato desiderato, si elaborano con la funzione hash blocchi di m bit, con $m \geq n$.

7.5 Letture e siti Web consigliati

[FUMY93] rappresenta una buona indagine sui principi di gestione della chiave.

La miglior trattazione dei generatori di numeri pseudocasuali si trova in [KNUT98]. Un'alternativa all'algoritmo standard a congruenza lineare, nota come algoritmo a ricorrenza lineare, è descritto in dettaglio in [BRIG79]. [ZENG91] valuta vari algoritmi di generazione di numeri pseudocasuali impiegabili nella generazione di chiavi di lunghezza variabile per cifrature di tipo di Vernam.

Un'eccellente rassegna dei PRNG, accompagnata da un'estesa bibliografia, è contenuta in [RITT91]. Anche [MENE97] contiene una buona analisi dei PRNG sicuri. Un altro buon testo, che pone l'enfasi su problemi pratici implementativi, si trova nel documento RFC 1750. Questa RFC descrive anche alcune tecniche di deskew. [KELS98] è una buona indagine sulle tecniche sicure di generazione di numeri pseudocasuali e sui rispettivi attacchi ad analisi crittografica.

BRIG79 H. Bright e R. Enison "Quasi-Random Number Sequences from Long-Period TLP Generator with Remarks on Application to Cryptography." *Computing Surveys*, Dicembre 1979.

FUMY93 S. Fumy e P. Landrock. "Principles of Key Management". *IEEE Journal on Selected Areas in Communications*, Giugno 1993.

KELS98 J. Kelsey, B. Schneier e C. Hall. "Cryptanalytic Attacks on Pseudorandom Number Generators." *Proceedings, Fast Software Encryption*, 1998. `http://www.schneier.com/paper-prngs.html`

KNUT98 Knuth, D. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Reading, MA: Addison-Wesley, 1998.

MENE97 Menezes, A; Oorshcot, P. e Vanstone, *S. Handbook of Applied Cryptography*. Boca Raton, FL: CRC Press, 1997.

RITT91 T.Ritter. "The Efficient Generation of Cryptographic Confusion Sequences). *Cryptologia*, vol. 15 n. 2, 1991. `http://www.cyphersbyritter.com/ARTS/CRNG2ART.HTM`.

ZENG91 Zeng. K., Yang, C., Wei, D. e Rao, T. "*Pseudorandom Bit Generators in Stream-Cipher Cryptography*". Computer, Febbraio 1991.

Siti Web consigliati

- **NIST Random Number Generation Technical Working Group:** contiene documenti e test sviluppati dal NIST e legati ai generatori di numeri casuali per applicazioni crittografiche. Offre anche un interessante elenco di link.
- **LavaRnd:** progetto open source che utilizza una sorgente casuale per generare numeri realmente casuali. Il sito contiene anche informazioni generali sui numeri casuali.
- **A Million Random Digits:** Elenco di numeri casuali compilato da RAND Corporation.

7.6 Termini chiave, domande di ripasso e problemi

Termini chiave

Armadio di cablaggio
Canale nascosto
Centro di distribuzione delle chiavi (KDC)
Chiave di sessione
Chiave master
Congruenza lineare
Crittografia di canale
Distribuzione delle chiavi
Generatore Blum, Blum, Shub
Generatore di numeri pseudocasuali (PRNG - Pseudorandom Number Generator)
Nonce
Riempimento del traffico
Generatore di numeri casuali
Skew

Domande di ripasso

7.1 Elencare i punti in cui possono verificarsi gli attacchi alla segretezza di una workstation utente operante in un tipico ambiente aziendale.
7.2 Qual è la differenza fra crittografia di canale e end-to-end?
7.3 Quali informazioni possono essere ottenute da un attacco ad analisi del traffico?
7.4 Cosa si intende con riempimento del traffico e qual è il suo scopo?
7.5 Elencare i modi in cui le chiavi segrete possono essere distribuite a due parti in comunicazione.
7.6 Qual è la differenza fra una chiave di sessione e una chiave master?
7.7 Che cos'è un nonce?
7.8 Che cos'è un centro di distribuzione delle chiavi?
7.9 Qual è la differenza fra casualità statistica e imprevedibilità?

Problemi

7.1 I sistemi di posta elettronica differiscono per il modo in cui vengono gestiti i destinatari multipli. In alcuni sistemi, il Mailer di origine crea tutte le copie necessarie e le invia in modo indipendente. Un approccio alternativo consiste nel determinare il percorso per ciascuna destinazione in anticipo. Sulla porzione comune del percorso verrà inviato un unico messaggio e poi verranno eseguite delle copie solo nei punti in cui i percorsi divergono; questa operazione è chiamata *mail bagging*.
   A. Tralasciando ogni considerazione sulla sicurezza, discutere i vantaggi e gli svantaggi dei due metodi.
   B. Discutere i requisiti e le implicazioni di sicurezza dei due metodi.

7.2 Il Paragrafo 7.2 descrive l'uso della lunghezza del messaggio per costruire un canale nascosto. Descrivere tre modalità alternative per l'utilizzo degli schemi di traffico per creare un canale nascosto.

7.3 Un rivenditore di reti locali fornisce un sistema di distribuzione delle chiavi come quello illustrato nella Figura 7.15.
   A. Descrivere lo schema.
   B. Confrontare questo schema con quello della Figura 7.9. Quali sono i pro e i contro?

7.4 "Siamo sotto pressione, Holmes". Il detective Lestrade sembra nervoso. "Abbiamo appreso che esistono delle copie di documenti segreti governativi nei computer di un'ambasciata estera qui a Londra. Normalmente questi documenti esistono in formato elettronico solo in alcuni computer governativi che soddisfano le più rigide norme di sicurezza. Tuttavia, talvolta devono essere inviati nelle reti che connettono i computer governativi. Ma tutti i messaggi di questa rete sono crittografati utilizzando un algoritmo di crittografia segretissimo certificato dai migliori esperti. Anche la

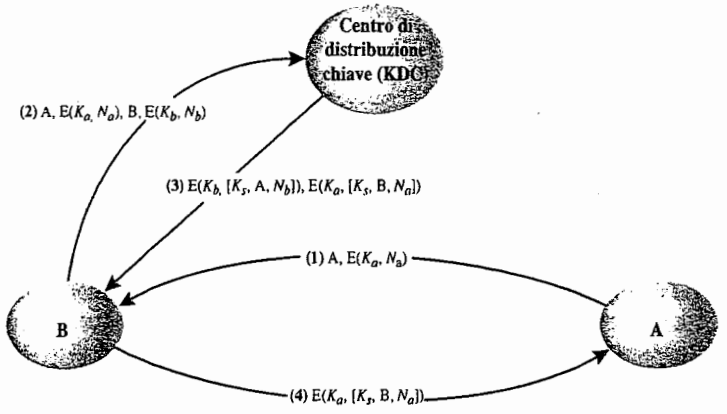


Figura 7.15 Figura per il Problema 7.3.

NSA e il KGB non sarebbero in grado di violarlo. Ora questi documenti si trovano nelle mani dei diplomatici di un piccolo e insignificante paese. Non si sa come ciò possa essere accaduto".

"Ma lei ha dei sospetti vero?" Chiese Holmes.

"Si, abbiamo svolto delle investigazioni di routine. Vi è un uomo che ha accesso a uno dei computer governativi e che ha contatti frequenti con i diplomatici di questa ambasciata. Ma il computer cui ha accesso non è di quelli fidati dove si trovano normalmente questi documenti. È lui il sospettato ma non abbiamo alcuna idea di come abbia potuto ottenere copia di questi documenti. Anche se riuscisse a ottenere una copia crittografata, non sarebbe in grado di interpretarla.

"Hmm, mi descriva il protocollo di comunicazione utilizzato nella rete". Holmes aprì gli occhi dimostrando così di aver seguito la conversazione di Lestrade con un'attenzione che contrastava con la sua aria sonnolenta.

"Il protocollo è il seguente. Ciascun nodo N della rete riceve una chiave segreta K_n. Questa chiave viene utilizzata per rendere sicure le comunicazioni fra il nodo e un server fidato. Tutte le chiavi sono memorizzate anche sul server. L'utente A, se vuole inviare un messaggio segreto M all'utente B, usa la seguente procedura.

1. A genera un numero casuale R e invia al server il proprio nome A, la destinazione B e $E(K_a, R)$.
2. Il server risponde inviando ad A $E(K_b, R)$.
3. A invia a B sia $E(R, M)$ che $E(K_b, R)$.
4. B conosce K_b e pertanto può decrittografare $E(K_b, R)$ per ottenere R e successivamente utilizzerà R per decrittografare $E(R, M)$ in modo da ottenere M.

Come si può vedere viene generata una chiave casuale ogni volta che deve essere inviato un messaggio. In effetti qualcuno potrebbe intercettare i messaggi scambiati dai nodi fidati e segreti ma non vi è alcun modo per decrittografare tali messaggi."

"Penso che abbiate trovato il vostro uomo, Lestrade. Il protocollo non è sicuro poiché il server non autentica gli utenti che gli inviano una richiesta. Apparentemente i progettisti del protocollo hanno ritenuto che l'invio di $E(K_X, R)$ autenticasse implicitamente l'utente X come mittente in quanto solo X (e il server) conosce K_x. Ma $E(K_X, R)$ può essere intercettato e riproposto successivamente. Una volta individuato il problema, sarà possibile controllare le attività di questo sospettato al computer cui ha accesso. Molto probabilmente le cose funzionano nel seguente modo: dopo aver intercettato $E(K_a, R)$ e $E(R, M)$ (vedere i passi 1 e 3 del protocollo), l'uomo, che chiameremo Z, sostiene di essere A e quindi ...". Terminare la frase di Holmes.

7.6 Se si prende l'algoritmo a congruenza lineare con un componente additivo 0:

$$X_{n+1} = (aX_n) \bmod m$$

si può dimostrare che se m è primo e se un determinato valore di a produce il massimo periodo $m - 1$, allora anche a^k produrrà il periodo massimo, sempre che k sia minore di m e che $m - 1$ non sia divisibile per k. Dimostrare ciò utilizzando $X_0 = 1$ e $m = 31$ e producendo le sequenze per $a = 3, 3^2, 3^3$ e 3^4.

7.7 A Quul è il massimo periodo ottenibile dal seguente generatore?

$$X_{n+1} = (aX_n) \bmod 2^4$$

B. Quale deve essere il valore di a?
C. Quali restrizioni è necessario imporre sul seme?

7.7 Ci si potrebbe chiedere perché nel metodo a congruenza lineare sia stato scelto il modulo $m = 2^{31} - 1$ invece di 2^{31}, dato che quest'ultimo numero può essere rappresentato senza bit aggiuntivi e l'operazione di modulo sarebbe più facile da svolgere. In generale è preferibile utilizzare il modulo $2^k - 1$ rispetto a 2^k. Perché?

7.8 Con l'algoritmo a congruenza lineare, una scelta di parametri che forniscono un periodo completo non fornisce necessariamente una buona casualizzazione. Per esempio, si considerino i due generatori seguenti:

$$X_{n+1} = (6X_n) \bmod 13$$
$$X_{n+1} = (7X_n) \bmod 13$$

Scrivere le due sequenze per dimostrare che entrambi generano periodi completi. Quale sequenza è più casuale?

7.9 In qualsiasi impiego dei numeri pseudocasuali, per crittografia, simulazione o progettazione statistica, è pericoloso fidarsi ciecamente del generatore di numeri casuali fornito con la libreria di sistema del computer. [PARK88] ha trovato che molti libri di testo contemporanei e ambienti di programmazione usano per la generazione di numeri casuali degli algoritmi contenenti difetti. Questo esercizio consentirà di eseguire un test del proprio sistema.

Il test si basa su un teorema attribuito a Ernesto Cesaro (per una dimostrazione vedere [KNUT98]) che dice che la probabilità che il massimo comune divisore di due interi scelti casualmente sia 1 è uguale a $6/\pi^2$. Usare questo teorema in un programma per determinare statisticamente il valore di π. Il programma principale dovrebbe richiamare tre sottoprogrammi: il generatore di numeri casuali delle librerie di sistema per generare gli interi casuali; un sottoprogramma che calcola il massimo comune divisore di due interi utilizzando l'algoritmo di Euclide e infine un sottoprogramma che calcola le radici quadrate. Se gli ultimi due programmi non sono disponibili, sarà necessario scriverli. Il programma principale dovrebbe utilizzare un ciclo con un gran numero di numeri casuali per dare una stima della probabilità di cui si sta parlando. Da qui è facile ottenere la stima di π.

Se il risultato è prossimo a 3,14, congratulazioni! In caso contrario, il risultato sarà probabilmente più basso, prevedibilmente un valore pari a circa 2,7. Perché si ottiene un risultato inferiore?

7.10 Si supponga di avere un generatore di numeri realmente casuali che generi un flusso di bit, nel quale ciascun bit abbia la stessa probabilità di essere 0 o 1 di qualunque altro bit nel flusso stesso e i cui bit non siano correlati. In altre parole, i bit hanno una distribuzione indipendente identica. Il flusso di bit, tuttavia, non è omogeneo: la probabilità di una cifra 1 è $0,5 + \varepsilon$ e la probabilità di una cifra 0 è $0,5 - \varepsilon$, con $0 < \varepsilon < 0,5$. Un semplice algoritmo di deskewing è il seguente: considerare il flusso di bit come una sequenza di coppie non sovrapposte; scartare tutte le coppie 00 e 11; sostituire ogni coppia 01 con 0 e ogni coppia 10 con 1.

A. Qual è la probabilità di ciascuna coppia nella sequenza originale?
B. Qual è la probabilità delle cifre 0 e 1 nella sequenza modificata?

C. Si supponga che l'algoritmo impieghi coppie di bit successivi sovrapposte, invece di coppie di bit successivi non sovrapposte. Ovvero, il primo bit in uscita dipenda dai bit 1 e 2, il secondo bit in uscita dipenda dai bit 2 e 3, e così via. Cosa si può affermare in relazione alle proprietà statistiche del flusso in uscita?

7.11 Un altro approccio di deskewing consiste nel considerare il flusso di bit come sequenza di gruppi non sovrapposti di n bit ciascuno, e considerare come uscita la parità di ciascun gruppo. Ovvero: se un gruppo contiene un numero dispari di uno l'uscita è uno, l'uscita è zero in caso contrario.

A. Esprimere l'operazione indicata come semplice funzione booleana.

B. Ipotizzare, come nel problema precedente, che la probabilità di una cifra 1 sia $0,5 + \varepsilon$. Se ciascun gruppo consistesse di due bit, quale sarebbe la probabilità di un'uscita 1?

C. Se ciascun gruppo consistesse di quattro bit, quale sarebbe la probabilità di un'uscita 1?

D. Generalizzare il risultato per trovare la probabilità di uscita di una cifra 1 per gruppi di ingresso di n bit.

7.12 Si supponga che qualcuno suggerisca il seguente modo per confermare che due persone sono in possesso della stessa chiave segreta. Si crea una stringa di bit casuali della stessa lunghezza della chiave, si esegue l'operazione di XOR con la chiave e si invia il risultato su un canale. All'altra estremità, la persona con cui si comunica esegue uno XOR dei dati ricevuti con la chiave (che dovrebbe essere uguale) e poi la rinvia. Si esegue la verifica e se si riceve di ritorno la stringa casuale originaria, si è verificato che l'altra persona usa la stessa chiave segreta anche se nessuno dei due ha trasmesso la chiave. Questo schema presenta difetti?

Parte seconda

Crittografia a chiave pubblica e funzioni hash

Dopo la crittografia simmetrica, l'altra importante forma di crittografia è la crittografia a chiave pubblica o asimmetrica, che ha rivoluzionato la sicurezza delle comunicazioni. Un campo correlato è quello delle funzioni hash crittografiche. Le funzioni hash vengono utilizzate nelle cifrature asimmetriche per la generazione delle firme digitali. Inoltre le funzioni hash vengono utilizzate per l'autenticazione dei messaggi. Le cifrature asimmetriche vengono utilizzate anche per la gestione delle chiavi. La Parte seconda del volume discute tutti questi argomenti.

Capitolo 8: Introduzione alla teoria dei numeri

La maggior parte degli schemi a chiave pubblica si basa sulla teoria dei numeri. Anche se il lettore può limitarsi ad accettare i risultati teorici, spesso è utile avere quanto meno una comprensione dei concetti della teoria dei numeri. Il Capitolo 8 fornisce una panoramica sull'argomento con numerosi esempi che aiutano a chiarire i concetti esposti.

Capitolo 9: Crittografia a chiave pubblica e RSA

Il Capitolo 9 introduce l'argomento della crittografia a chiave pubblica, concentrandosi sulla segretezza delle informazioni. Questo capitolo esamina anche la cifratura a chiave pubblica più ampiamente utilizzata, ovvero l'algoritmo RSA (Rivest-Shamir-Adleman).

Capitolo 10: La gestione delle chiavi; altri sistemi crittografici a chiave pubblica

Il Capitolo 10 ritorna sull'argomento della gestione delle chiavi nelle cifrature asimmetriche. Questo capitolo descrive anche la tecnica di scambio delle chiavi più utilizzata, la tecnica

Diffie-Hellman, ed esamina un approccio più recente alla tecnica a chiave pubblica che si basa sulle curve ellittiche.

Capitolo 11: Autenticazione dei messaggi e funzioni hash

Nel campo della sicurezza, l'autenticazione è importante quanto la segretezza. Come minimo l'autenticazione dei messaggi garantisce che un messaggio provenga dal mittente indicato. Inoltre l'autenticazione può includere la protezione contro qualsiasi modifica, ritardo, ripetizione e riordinamento. Il Capitolo 11 si apre con un'analisi dei requisiti di autenticazione e fornisce una presentazione sistematica degli approcci all'autenticazione. Un elemento chiave degli schemi di autenticazione è l'uso di un autenticatore, solitamente un codice MAC (Message Authentication Code) o una funzione hash. Verranno esaminate le considerazioni progettuali per entrambi questi tipi di algoritmi, analizzando numerosi esempi specifici.

Capitolo 12: Algoritmi hash e MAC

Il Capitolo 12 estende la discussione del capitolo precedente per trattare due delle funzioni hash più importanti (SHA e Whirlpool) e due degli algoritmi MAC più importanti (HMAC e CMAC).

Capitolo 13: Firma digitale e protocolli di autenticazione

Uno dei tipi di autenticazione più importanti è rappresentato dalle firme digitali. Il Capitolo 13 esamina le tecniche utilizzate per costruire le firme digitali e tratta lo standard DSS (Digital Signature Standard).

Le varie tecniche di autenticazione basate su firme digitali sono gli elementi costitutivi degli algoritmi di autenticazione. La progettazione di tali algoritmi deve prevedere l'analisi di attacchi subdoli, in grado di aggirare molti protocolli apparentemente sicuri. Questo argomento verrà trattato anche nel Capitolo 14.

Capitolo 8

Introduzione alla teoria dei numeri

Concetti essenziali

- Un **numero primo** è un intero che può essere diviso senza resto solo per 1, per se stesso e per il proprio opposto. I numeri primi giocano un ruolo critico sia nella teoria dei numeri che nella crittografia.
- Il teorema di **Fermat** e il teorema di **Eulero** rivestono un ruolo importante nella crittografia.
- Un requisito importante in numerosi algoritmi di crittografia è la possibilità di scegliere un numero primo molto esteso. Lo sviluppo di algoritmi efficienti per determinare se un dato numero intero molto esteso scelto a caso sia primo, è tuttora un'area di ricerca attiva.
- I **logaritmi discreti** sono fondamentali per numerosi algoritmi a chiave pubblica. I logaritmi discreti sono analoghi ai logaritmi ordinari, ma operano nell'aritmetica modulare.

Alcuni concetti della teoria dei numeri sono fondamentali nella progettazione degli algoritmi di crittografia a chiave pubblica. Questo capitolo offre una panoramica dei concetti trattati in altri capitoli. Chiunque ritenesse di conoscere questi concetti è naturalmente libero di saltare la lettura di questo capitolo. Questo capitolo, come il Capitolo 4, riporta numerosi esempi che sono evidenziati con uno sfondo grigio.

8.1 I numeri primi[1]

Un elemento fondamentale della teoria dei numeri è costituito dai numeri primi. Sono stati scritti interi libri su questo argomento (per esempio [CRAN01], [RIBE96]). Questa parte

[1] In questa parte del capitolo si parla in generale solo di interi non negativi. L'uso di interi negativi non introduce però differenze fondamentali.

del capitolo offre una panoramica sull'argomento, con riferimento in particolare agli argomenti trattati nel volume.

Un intero $p > 1$ è un numero primo se e solo se i suoi unici divisori[2] sono ± 1 e $\pm p$. I numeri primi giocano un ruolo fondamentale nella teoria dei numeri e nelle tecniche trattate in questo capitolo.

La Tabella 8.1 contiene tutti i numeri primi fino a 2000. Si noti come sono distribuiti i numeri primi, in particolare si osservi la quantità di numeri primi in ogni intervallo di 100 numeri.

Qualsiasi intero $a > 1$ può essere suddiviso in fattori in modo univoco come:

$$a = p_1^{a_1}, p_2^{a_2}, \ldots, p_t^{a_t} \tag{8.1}$$

dove $p_1 < p_2 < \ldots < p_t$ sono numeri primi e dove ogni a_i è un intero positivo. Questo è noto come teorema fondamentale dell'aritmetica; la dimostrazione è disponibile in qualsiasi testo sulla teoria dei numeri.

$$91 = 7 \times 13; \quad 3600 = 2^4 \times 3^2 \times 5^2 \quad 11011 = 7 \times 11^2 \times 13$$

È utile riformulare questo risultato in un altro modo. Se P è l'insieme di tutti i numeri primi, allora ogni intero positivo può essere scritto in modo univoco nella seguente forma:

$$a = \prod_{p \in P} p^{a_p} \qquad \text{dove ogni } a_p \geq 0$$

La parte destra è il prodotto di tutti i numeri primi di p; per un determinato valore di a, la maggior parte degli esponenti a_p sarà uguale a 0.

Il valore di un intero positivo può essere specificato semplicemente elencando tutti gli esponenti diversi da zero nella formulazione precedente.

L'intero 12 è rappresentato da $\{a_2 = 2, a_3 = 1\}$.
L'intero 18 è rappresentato da $\{a_2 = 1, a_3 = 2\}$.
L'intero 19 è rappresentato da $\{a_7 = 1, a_{13} = 1\}$.

La moltiplicazione di due numeri è equivalente alla somma degli esponenti corrispondenti:

$$k = mn \rightarrow k_p = m_p + n_p \quad \text{per tutti i } p \in P$$

Dato $a = \prod_{p \in P} p^{a_p}$ e $b = \prod_{p \in P} p^{b_p}$ si definisca $k = ab$. È noto che k può essere rappresentato come prodotto di potenze dei numeri primi: $k = \prod_{p \in P} p^{k_p}$. Ne segue che $k_p = a_p + b_p$ per qualsiasi $p \in P$.

$$k = 12 \times 18 = (2^2 \times 3) \times (2 \times 3^2) = 216$$
$$k_2 = 2 + 1 = 3; k_3 = 1 + 2 = 3$$
$$216 = 2^3 \times 3^3 = 8 \times 27$$

[2] Come si è detto nel Capitolo 4, si dice che a divide b quando la divisione fra i due numeri non ha resto.

Tabella 8.1 I numeri primi fino a 2000.

2	3	5	7	11	13	17	19	23	29	31	37	41	43	47	53	59	61	67	71	73	79	83	89	97
101	103	107	109	113	127	131	137	139	149	151	157	163	167	173	179	181	191	193	197	199				
211	223	227	229	233	239	241	251	257	263	269	271	277	281	283	293									
307	311	313	317	331	337	347	349	353	359	367	373	379	383	389	397									
401	409	419	421	431	433	439	443	449	457	461	463	467	479	487	491	499								
503	509	521	523	541	547	557	563	569	571	577	587	593	599											
601	607	613	617	619	631	641	643	647	653	659	661	673	677	683	691									
701	709	719	727	733	739	743	751	757	761	769	773	787	797											
809	811	821	823	827	829	839	853	857	859	863	877	881	883	887										
907	911	919	929	937	941	947	953	967	971	977	983	991	997											
1009	1013	1019	1021	1031	1033	1039	1049	1051	1061	1063	1069	1087	1091	1093	1097									
1103	1109	1117	1123	1129	1151	1153	1163	1171	1181	1187	1193													
1201	1213	1217	1223	1229	1231	1237	1249	1259	1277	1279	1283	1289	1291	1297										
1301	1303	1307	1319	1321	1327	1361	1367	1373	1381	1399														
1409	1423	1427	1429	1433	1439	1447	1451	1453	1459	1471	1481	1483	1487	1489	1493	1499								
1511	1523	1531	1543	1549	1553	1559	1567	1571	1579	1583	1597													
1601	1607	1609	1613	1619	1621	1627	1637	1657	1663	1667	1669	1693	1697	1699										
1709	1721	1723	1733	1741	1747	1753	1759	1777	1783	1787	1789													
1801	1811	1823	1831	1847	1861	1867	1871	1873	1877	1879	1889													
1901	1907	1913	1931	1933	1949	1951	1973	1979	1987	1993	1997	1999												

Cosa significa, in termini dei fattori primi di a e b, dire che a divide b ($a|b$)? Un intero nella forma p^k può essere diviso solo da un intero di una potenza minore o uguale dello stesso numero primo, p^j con $j \le k$. Pertanto si può dire:

$$\text{Dati } a = \prod_{p \in P} p^{a_p} \text{ e } b = \prod_{p \in P} p^{b_p}, \text{ se } a|b \text{ allora } a_p \le b_p \text{ per tutti i } p$$

$$a = 12; b = 36; 12|36;$$
$$12 = 2^2 \times 3; 36 = 2^2 \times 3^2$$
$$a_2 = 2 = b_2$$
$$a_3 = 1 \le 2 = b_3$$

Quindi la disuguaglianza $a_p \le b_p$ è valida per qualsiasi primo.

È facile determinare il massimo comune divisore[3] di due interi positivi se si esprime ciascun intero come un prodotto di numeri primi.

$$300 = 2^2 \times 3^1 \times 5^2$$
$$18 = 2^1 \times 3^2$$
$$\gcd(18,300) = 2^1 \times 3^1 \times 5^0 = 6$$

In generale la relazione seguente è sempre valida:

$$\text{Se } k = \gcd(a,b) \text{ allora } k_p = \min(a_p, b_p) \text{ per tutti i } p$$

L'individuazione dei fattori primi di un numero di grandi dimensioni non è un'impresa facile e dunque la precedente relazione non determina direttamente un metodo pratico di calcolo del massimo comune divisore.

8.2 I teoremi di Fermat e di Eulero

I teoremi di Fermat e di Eulero giocano un ruolo fondamentale nella crittografia a chiave pubblica.

Il teorema di Fermat[4]

Il teorema di Fermat stabilisce che se p è un numero primo e a è un intero positivo non divisibile per p, allora:

$$a^{p-1} \equiv 1 \bmod p \tag{8.2}$$

[3] Come si è detto nel Capitolo 4, il massimo comune divisore degli interi a e b, espresso come $\gcd(a,b)$, è un intero c che divide sia a che b senza resto e tale che ogni divisore di a e b è divisore di c.

[4] Talvolta questo teorema viene chiamato piccolo teorema di Fermat.

Dimostrazione: Si consideri l'insieme degli interi positivi minori di p: $\{1, \ldots, p-1\}$ e si moltiplichi ciascun elemento per a modulo p per ottenere l'insieme X: $\{a \bmod p, 2a \bmod p, \ldots, (p-1)\, a \bmod p\}$. Nessuno degli elementi di X è uguale a zero dato che p non divide a. Inoltre non esistono due numeri uguali in X. Per verificarlo, si supponga che $ja \equiv ka \pmod p$ con $1 \le j < k \le p-1$. Dato che a è primo[5] relativo di p, è possibile eliminare a da entrambi i termini dell'equazione (vedere l'Equazione 4.3) ottenendo il risultato $j \equiv k \pmod p$. Quest'ultima uguaglianza è impossibile dato che j e k sono entrambi interi positivi minori di p. Quindi si è dimostrato che i $p-1$ elementi di X sono tutti interi positivi differenti. È possibile concludere che X è costituito dall'insieme degli interi $\{1, 2, \ldots, (p-1)\}$ disposti in un qualche ordine. Moltiplicando i numeri di entrambi gli insiemi e prendendo il risultato modulo p, si avrà:

$$a \times 2a \times \ldots \times (p-1) \equiv [(1 \times 2 \times \ldots \times (p-1)] \pmod p$$
$$a^{p-1}(p-1)! \equiv (p-1)! \pmod p$$

Si può eliminare il termine $(p-1)!$ in quanto è primo relativo di p (vedere l'Equazione 4.3). Si ottiene pertanto l'Equazione 8.2.

$$a = 7, p = 19$$
$$7^2 = 49 \equiv 11 \pmod{19}$$
$$7^4 \equiv 121 \equiv 7 \pmod{19}$$
$$7^8 \equiv 49 \equiv 11 \pmod{19}$$
$$7^{16} \equiv 121 \equiv 7 \pmod{19}$$
$$a^{p-1} = 7^{18} = 7^{16} \times 7^2 \equiv 7 \times 11 \equiv 1 \pmod{19}$$

Ecco una forma alternativa del teorema di Fermat altrettanto utile: se p è primo e a è un intero positivo, allora:

$$a^p \equiv a \pmod p \tag{8.3}$$

Si noti che la prima forma del teorema (Equazione 8.2) richiede che a sia primo relativo di p, mentre questa forma non lo richiede.

$$p = 5, \quad a = 3, \quad a^p = 3^5 = 243 \equiv 3 \pmod 5 = a \pmod p$$
$$p = 5, \quad a = 10, \quad a^p = 10^5 = 100000 \equiv 10 \pmod 5 \equiv 0 \pmod 5 = a \pmod p$$

La funzione toziente di Eulero

Prima di presentare il teorema di Eulero, occorre introdurre una quantità importante nella teoria dei numeri, chiamata funzione toziente di Eulero, $\phi(n)$, definita come il numero di interi positivi minori di n e primi relativi di n. Per convenzione, $\phi(1) = 1$.

[5] Nel Capitolo 4 si è detto che due numeri sono primi fra di loro (o "primi relativi") se non hanno fattori primi in comune, ovvero se il loro unico divisore comune è 1. Ciò equivale a dire che due numeri sono primi fra loro se il loro massimo comun divisore è 1.

Determinare $\phi(37)$ e $\phi(35)$.

Poiché 37 è un numero primo, tutti gli interi positivi compresi fra 1 e 36 sono primi relativi di 37. Pertanto $\phi(37) = 36$.

Per determinare $\phi(35)$ si devono elencare gli interi positivi minori di 35 che siano primi relativi con questo numero:

1, 2, 3, 4, 6, 8, 9, 11, 12, 13, 16, 17, 18,

19, 22, 23, 24, 26, 27, 29, 31, 32, 33, 34.

L'elenco è costituito da 24 numeri e dunque $\phi(35) = 24$.

La Tabella 8.2 elenca i primi 30 valori di $\phi(n)$. Il valore $\phi(1)$ è senza significato ma per definizione è uguale a 1.

Dovrebbe essere chiaro che per un numero primo p:

$$\phi(p) = p - 1$$

Si supponga ora di avere due numeri primi p e q con $p \neq q$. Allora si può dimostrare che per $n = pq$:

$$\phi(n) = \phi(pq) = \phi(p) \times \phi(q) = (p-1) \times (q-1)$$

Per dimostrare che $\phi(n) = \phi(p) \times \phi(q)$, si consideri che l'insieme degli interi positivi minori di n è l'insieme $\{1, \ldots, (pq-1)\}$. Gli interi di questo insieme che non sono primi relativi di n sono l'insieme $\{p, 2p, \ldots, (q-1)p\}$ e l'insieme $\{q, 2q, \ldots, (p-1)q\}$. Di conseguenza:

$$\begin{aligned}\phi(n) &= (pq-1) - [(q-1) + (p-1)] \\ &= pq - (p+q) + 1 \\ &= (p-1) \times (q-1) \\ &= \phi(p) \times \phi(q)\end{aligned}$$

$\phi(21) = \phi(3) \times \phi(7) = (3-1) \times (7-1) = 2 \times 6 = 12$, dove i 12 interi sono $\{1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20\}$.

Il teorema di Eulero

Il teorema di Eulero stabilisce che per ogni a e n primi relativi vale la seguente relazione:

$$a^{\phi(n)} \equiv 1 \pmod{n} \tag{8.4}$$

$$a = 3; n = 10; \phi(10) = 4; a^{\phi(n)} = 3^4 = 81 \equiv 1 \pmod{10} = 1 \pmod{n}$$

$$a = 2; n = 11; \phi(11) = 10; a^{\phi(n)} = 2^{10} = 1024 \equiv 1 \pmod{11} = 1 \pmod{n}$$

Dimostrazione: l'Equazione 8.4 è vera se n è primo poiché in tal caso $\phi(n) = (n-1)$ e vale il teorema di Fermat. Tuttavia ciò vale anche per un qualsiasi intero n. Si ricordi che $\phi(n)$ è il numero di interi positivi minori di n e primi relativi di n. Si consideri l'insieme di tali interi, identificati nel seguente modo:

Tabella 8.2 Alcuni valori della funzione toziente di Eulero $\phi(n)$.

n	$\phi(n)$	n	$\phi(n)$	n	$\phi(n)$
1	1	11	10	21	12
2	1	12	4	22	10
3	2	13	12	23	22
4	2	14	6	24	8
5	4	15	8	25	20
6	2	16	8	26	12
7	6	17	16	27	18
8	4	18	6	28	12
9	6	19	18	29	28
10	4	20	8	30	8

$$R = \{x_1, x_2, \ldots, x_{\phi(n)}\}$$

Ovvero ogni elemento x_i di R è un intero positivo minore di n con $\gcd(x_i, n) = 1$.
Ora si moltiplichi ciascun elemento per a, modulo n:

$$S = \{(ax_1 \bmod n), (ax_2 \bmod n), \ldots, (ax_{\phi(n)} \bmod n)\}$$

L'insieme S è una permutazione di R in base al seguente ragionamento.

1. Poiché a è primo relativo di n e x_i è primo relativo di n, anche ax_i deve essere primo relativo di n. Pertanto tutti i membri di S sono interi minori di n e primi relativi di n.
2. In S non esistono duplicati. Si faccia riferimento all'Equazione 4.3. Se $ax_i \bmod n = ax_j \bmod n$, allora $x_i = x_j$.

Pertanto:

$$\prod_{i=1}^{\phi(n)} (ax_i \bmod n) = \prod_{i=1}^{\phi(n)} x_i$$

$$\prod_{i=1}^{\phi(n)} ax_i \equiv \prod_{i=1}^{\phi(n)} x_i \pmod{n}$$

$$a^{\phi(n)} \times \left[\prod_{i=1}^{\phi(n)} x_i\right] \equiv \prod_{i=1}^{\phi(n)} x_i \pmod{n}$$

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

Si tratta del medesimo schema di ragionamento utilizzato nella dimostrazione del teorema di Fermat. Analogamente al caso del teorema di Fermat, è utile anche esaminare una forma alternativa del teorema:

$$a^{\phi(n)+1} \equiv a (\text{mod } n) \tag{8.5}$$

Come nel caso del teorema di Fermat, la prima forma del teorema di Eulero (Equazione 8.4) richiede che a sia primo relativo di p, mentre questa forma non lo richiede.

8.3 Test di primalità

In vari algoritmi crittografici è necessario selezionare casualmente uno o più numeri primi di grandi dimensioni. Pertanto occorre poter determinare se un numero di grandi dimensioni è primo o meno. Non esiste un modo semplice ed efficiente per svolgere questa operazione.

In questa parte del capitolo si presenterà un algoritmo molto interessante. Ci si potrebbe sorprendere del fatto che questo algoritmo fornisce un numero non necessariamente primo ma che è quasi certamente primo. Di seguito ne verranno spiegate le ragioni. Si farà riferimento anche a un algoritmo deterministico per trovare numeri primi. Il paragrafo si conclude con una discussione che riguarda la distribuzione dei numeri primi.

Algoritmo di Miller-Rabin[6]

L'Algoritmo sviluppato da Miller e Rabin [Mill75, RABI80] è tipicamente utilizzato per determinare se un numero molto grande è primo. Prima di analizzare l'algoritmo sono necessarie alcune premesse. Prima, qualsiasi intero positivo dispari può essere espresso nella forma seguente:

$$n - 1 = 2^k q \qquad \text{con } k > 0, q \text{ dispari}$$

Per verificarlo, si noti che $(n - 1)$ è un intero pari. Quindi si divida ripetutamente $(n - 1)$ per 2 fino a ottenere un numero dispari q, eseguendo un totale di k divisioni. Se n è espresso come numero binario, tale risultato viene ottenuto scorrendo il numero verso destra fino a quando il bit più a destra vale uno, per un totale di k scorrimenti. Si sviluppano ora due proprietà dei numeri primi che saranno necessarie.

Due proprietà dei numeri primi

La **prima proprietà** dice che: se p è primo e a è un intero positivo minore di p, allora $a^2 \bmod p = 1$ se e solo se $a \bmod p = 1$ oppure $a \bmod p = -1 \bmod p = p - 1$. Secondo le regole dell'aritmetica modulare $(a \bmod p)(a \bmod p) = a^2 \bmod p$. Quindi se $a \bmod p = 1$ oppure $a \bmod p = -1$, allora $a^2 \bmod p = 1$. Viceversa se $a^2 \bmod p = 1$ allora $(a \bmod p)^2 = 1$, che è verificata solo se $a \bmod p = 1$ oppure $a \bmod p = -1$.

La **seconda proprietà** è formulata nel modo seguente. Sia p un numero primo maggiore di 2. È quindi possibile scrivere $p - 1 = 2^k q$ con $k > 0$, q dispari. Sia a un qualsiasi intero nell'intervallo $1 < a < p - 1$. Allora è valida una delle seguenti due condizioni:

1. a^q è congruente a 1 modulo p. Ovvero, $a^q \bmod p = 1$ oppure, in modo equivalente, $a^q \equiv 1 (\text{mod } p)$.
2. Uno dei numeri a^q, a^{2q}, a^{4q}, ..., $a^{2^{k-1}q}$ è congruente a -1 modulo p. Ovvero, vi è un numero j nell'intervallo $(1 \le j \le k)$ tale che $a^{2^{j-1}q} \bmod p = -\bmod p = p - 1$ o, in modo equivalente, $a^{2^{j-1}q} \equiv -1 (\text{mod } p)$.

Dimostrazione. Il teorema di Fermat (Equazione 8.2) afferma che $a^{n-1} \equiv 1 (\text{mod } n)$ se n è primo. Si ha $p - 1 = 2^k q$, quindi si sa che $a^{p-1} \bmod p = a^{2^k q} \bmod p = 1$. Quindi se si esamina la sequenza di numeri

$$a^q \bmod p, a^{2q} \bmod p, a^{4q} \bmod p, \ldots, a^{2^{k-1}q} \bmod p, a^{2^k q} \bmod p \tag{8.6}$$

si sa che l'ultimo numero della lista vale 1. Inoltre, ogni numero della lista è il quadrato del numero che lo precede. Quindi deve essere valida una delle due seguenti ipotesi:

1. Il primo numero della lista, e dunque tutti i numeri successivi, deve essere 1.
2. Alcuni numeri nella lista sono diversi da 1, ma il loro quadrato mod p è uguale a 1. Per la prima proprietà dei numeri primi definita in precedenza, il solo numero che soddisfa questa condizione è $p - 1$. Quindi, in questo caso, la lista contiene un elemento uguale a $p - 1$.

Questo completa la dimostrazione.

L'algoritmo in dettaglio

Queste considerazioni portano alla conclusione che se n è primo, o il primo elemento nell'elenco dei resti $(a^q, a^{2q}, \ldots, a^{2^{k-1}q}, a^{2^k q})$ modulo 1 è 1 o qualche elemento dell'elenco è $n - 1$; altrimenti n è composito (ovvero non è primo). Tuttavia se vale questa condizione, questo non significa necessariamente che n sia primo. Per esempio se $n = 2047 = 23 \times 89$ allora $n - 1 = 2 \times 1023$. Calcolando, $2^{1023} \bmod 2047 = 1$ così che 2047 soddisfa la condizione ma non è primo.

È possibile utilizzare la proprietà precedente per sviluppare un test di primalità. La procedura TEST prende come input un candidato intero n e restituisce il risultato "composito" se n è sicuramente non primo e il risultato "incerto" se n può essere primo.

TEST (n)
1. Trovare gli interi k, q con $k > 0$, q dispari, tali che $(n - 1 = 2^k q)$;
2. Scegliere un intero casuale a, $1 < a < n - 1$.
3. **if** $a^q \bmod n = 1$ **then** return("incerto");
4. **for** $j = 0$ **to** $k - 1$ **do**
5. **if** $a^{2^j q} \bmod n \equiv n - 1$ then return("incerto");
6. return("composito");

Come esempio si applichi il test della primalità al numero $n = 29$. Si ha $(n - 1) = 28 = 2^2(7) = 2^k q$. Innanzitutto si può provare con $a = 10$. Si calcola $10^7 \bmod 29 = 17$ che non è né 1 né

[6] Nella letteratura scientifica viene anche indicato come "algoritmo di Rabin-Miller", "test di Miller-Rabin" o "test di Rabin-Miller".

28 e dunque si procede con il test. Il calcolo successivo trova che $(10^7)^2 \bmod 29 = 28$, e il test restituisce "incerto" (ovvero 29 può essere primo). Si riprova con $a = 2$. Vengono eseguiti i seguenti calcoli: $2^7 \bmod 29 = 12$; $2^{14} \bmod 29 = 28$; e il test restituisce ancora "incerto". Se si esegue il test per tutti gli interi a compresi fra 1 e 28, si ottiene sempre il risultato "incerto" che è compatibile con il fatto che n è un numero primo.

Ora si proverà ad applicare il test al numero composito $n = 13 \times 17 = 221$. Allora $(n - 1) = 220 = 2^2(55) = 2^k q$. Si prova con $a = 5$. Si avrà $5^{55} \bmod 221 = 112$ che non è né 1 né 220; $(5^{55})^2 \bmod 221 = 168$. Poiché nella riga 4 dell'algoritmo TEST sono stati usati tutti i valori di j (ovvero $j = 0$ e $j = 1$), il test restituisce "composito" per indicare che 221 è effettivamente un numero composito. Ma si supponga di aver scelto $a = 21$; si avrà $21^{55} \bmod 221 = 200$; $(21^{55})^2 \bmod 221 = 220$; il test restituisce "incerto" indicando che 221 può essere primo. Infatti, dei 220 interi compresi fra 1 e 220, 6 restituiscono il risultato "incerto": 1, 21, 47, 174, 200 e 220.

Uso ripetuto dell'algoritmo di Miller-Rabin

Come si può usare l'algoritmo di Miller–Rabin per determinare con un elevato livello di confidenza se un determinato intero è primo? Si può dimostrare [KNUT98] che dato un numero dispari n non primo e un intero a scelto casualmente con $1 < a < n - 1$, la probabilità che TEST ritorni *incerto* (ovvero non riesca a determinare che n non è primo) è inferiore a 1/4. Quindi se si scelgono t valori differenti di a, la probabilità che tutti passino TEST (ritornino *incerto*) è inferiore a $(1/4)^t$. Per esempio, per $t = 10$ la probabilità che un numero non primo passi tutti e dieci i test è inferiore a 10^{-6}. Quindi, per un valore sufficientemente grande di t, si può ipotizzare con un elevato livello di confidenza che n è primo.

Questo fornisce le basi per determinare se un intero dispari n è primo con un livello di confidenza ragionevole. La procedura completa è la seguente: richiamare ripetutamente TEST (n) utilizzando valori casuali di a. Se a un certo punto TEST restituisce il risultato "composito", allora n è certamente non primo. Se TEST restituisce il risultato "incerto" per t volte di seguito, per un valore di t sufficientemente grande, si può essere ragionevolmente sicuri che n sia primo.

Algoritmo deterministico per il test di primalità

Prima del 2002 non vi era alcun metodo noto per provare in modo efficiente la primalità di numeri molto grandi. Tutti gli algoritmi utilizzati, fra cui l'algoritmo più diffuso (Miller-Rabin), producevano risultati probabilistici. Nel 2002 Agrawal, Kayal e Safena [AGRA02] hanno sviluppato un algoritmo deterministico relativamente semplice, capace di determinare in modo efficiente se un dato numero grande a piacere sia primo. L'algoritmo, noto come algoritmo AKS, non sembra comunque efficiente come quello di Miller-Rabin; allo stato attuale non ha sostituito la precedente tecnica probabilistica [BORN03].

Distribuzione dei numeri primi

Vale la pena di valutare quanti numeri possono essere rifiutati prima che possa essere trovato un numero primo utilizzando il test Miller-Rabin o un altro test della primalità. Un risultato della teoria dei numeri, chiamato teorema dei numeri primi, stabilisce che i numeri primi in prossimità di n sono spaziati in media uno ogni $\ln(n)$ interi. In pratica, in media, si dovrebbe eseguire il test su $\ln(n)$ interi prima di trovare un numero primo. Poiché tutti gli interi pari possono essere immediatamente rifiutati, il dato corretto è $0{,}5 \ln(n)$. Per esempio, se venisse ricercato un numero primo dell'ordine di grandezza di 2^{200}, per trovare un numero primo sarebbero necessari circa $0{,}5 \ln(2^{200}) = 69$ tentativi. Tuttavia questo dato è semplicemente una media. In alcune parti della retta dei numeri, i numeri primi sono posti a una distanza più ravvicinata mentre in altre parti vi sono lunghe successioni senza numeri primi.

I due interi dispari consecutivi 1 000 000 000 061 e 1 000 000 000 063 sono entrambi primi. Al contrario, 1001!+2, 1001!+3, . . . , 1001!+1000, 1001!+1001 è una sequenza di mille interi consecutivi compositi.

8.4 Il teorema cinese del resto

Il teorema cinese del resto (CRT) è uno dei risultati più utili della teoria dei numeri.[7] Sostanzialmente il teorema cinese del resto dice che è possibile ricostruire gli interi di un determinato intervallo partendo dai loro resti modulo una coppia di numeri primi relativi.

I dieci interi in Z_{10} (0, 1, . . . , 9) possono essere ricostruiti in base ai loro resti modulo 2 e 5 (i primi relativi di 10). Si supponga che i resti noti di una cifra decimale x siano $r_2 = 0$ e $r_5 = 3$; in pratica $x \bmod 2 = 0$ e $x \bmod 5 = 3$. Pertanto x è un intero pari in Z_{10} il cui resto, quando viene diviso per 5, è 3. La soluzione univoca è $x = 8$.

Il teorema CRT può essere formulato in vari modi. Qui si presenterà una formulazione particolarmente utile per gli argomenti trattati in questo testo. Una formulazione alternativa viene esplorata nel Problema 8.17. Sia:

$$M = \prod_{i=1}^{k} m_i$$

dove gli m_i sono coppie di numeri primi relativi; ovvero $\gcd(m_i, m_j) = 1$ per $1 \le i, j \le k$ e $i \ne j$. Si può rappresentare qualsiasi intero A in Z_M tramite una k-tupla i cui elementi sono in Z_{m_i} utilizzando la seguente corrispondenza:

$$A \leftrightarrow (a_1, a_2, \ldots, a_k) \tag{8.7}$$

dove $A \in Z_M$, $a_i \in Z_{m_i}$ e $a_i = A \bmod m_i$ per $1 \le i \le k$. Il teorema cinese del resto stabilisce due fatti.

1. Il mapping dell'Equazione 8.7 è una corrispondenza uno-a-uno (chiamata **biiezione**) fra Z_M e il prodotto cartesiano $Z_{m_1} \times Z_{m_2} \times \ldots \times Z_{m_k}$. Ovvero per ogni intero A tale che $0 \le A < M$ vi è una k-tupla univoca $(a_1, a_2, \ldots, a_k)$ con $0 \le a_i < m_i$ che lo rappresenta e per ogni k-tupla $(a_1, a_2, \ldots, a_k)$ vi è un A univoco in Z_M.
2. Le operazioni svolte sugli elementi di Z_M possono essere eseguite in modo equivalente sulle corrispondenti k-tuple svolgendo le operazioni in modo indipendente su ciascuna posizione coordinata nel sistema appropriato.

[7] Il Teorema cinese del resto è chiamato in questo modo poiché si ritiene sia stato scoperto dal matematico cinese SunTse intorno al 100 d.C.

Si proverà a dimostrare la *prima asserzione*. La trasformazione da A a $(a_1, a_2, \ldots, a_k)$ è ovviamente univoca; infatti ogni a_i è univocamente calcolato con $a_i = A \bmod m_i$. Il calcolo di A da $(a_1, a_2, \ldots, a_k)$ può essere effettuato nel modo seguente. Sia $M_i = M/m_i$ per $1 \le i \le k$. Si noti che $M_i = m_1 \times m_2 \times \ldots \times m_{i-1} \times m_{i+1} \times \ldots \times m_k$, tale che $M_i \equiv 0 \pmod{m_j}$ per tutti i $j \ne i$. Si definiscano:

$$c_i = M_i \times (M_i^{-1} \bmod m_i) \quad \text{per } 1 \le i \le k \tag{8.8}$$

Per definizione di M_i, questo è primo relativo di m_i e pertanto ha un unico inverso moltiplicativo modulo m_i. Dunque l'Equazione 8.8 è ben definita e produce un valore univoco c_i. Quindi si può calcolare:

$$A \equiv \left(\sum_{i=1}^{k} a_i c_i \right) \pmod{M} \tag{8.9}$$

Per dimostrare che il valore di A prodotto dall'Equazione 8.9 è corretto, occorre dimostrare che $a_i = A \bmod m_i$ per $1 \le i \le k$. Si noti che $c_j \equiv M_j \equiv 0 \pmod{m_i}$ se $j \ne i$ e che $c_i \equiv 1 \pmod{m_i}$. Ne consegue che $a_i = A \bmod m_i$.

La *seconda asserzione* del teorema cinese del resto, riguardante le operazioni aritmetiche, deriva dalle regole dell'aritmetica modulare.

La seconda asserzione può essere formulata nel seguente modo; se:

$$A \leftrightarrow (a_1, a_2, \ldots, a_k);\ B \leftrightarrow (b_1, b_2, \ldots, b_k)$$

allora:

$$(A + B) \bmod M \leftrightarrow ((a_1 + b_1) \bmod m_1, \ldots, (a_k + b_k) \bmod m_k)$$
$$(A - B) \bmod M \leftrightarrow ((a_1 - b_1) \bmod m_1, \ldots, (a_k - b_k) \bmod m_k)$$
$$(A \times B) \bmod M \leftrightarrow ((a_1 \times b_1) \bmod m_1, \ldots, (a_k \times b_k) \bmod m_k)$$

Una delle caratteristiche più utili del teorema cinese del resto è il fatto che fornisce un modo per manipolare numeri potenzialmente molto estesi modulo M in termini di tuple di numeri più piccoli. Questo può essere utile quando M è composto da 150 o più cifre. Si osservi che è necessario conoscere a priori la fattorizzazione di M.

Per rappresentare 973 mod 1813 come una coppia di numeri modulo 37 e 49, si definiscano[8]:

$$m_1 = 37$$
$$m_2 = 49$$
$$M = 1813$$
$$A = 973$$

Si ha anche $M_1 = 49$ e $M_2 = 37$. Utilizzando l'algoritmo esteso di Euclide si calcola $M_1^{-1} = 34 \bmod m_1$ e $M_2^{-1} = 4 \bmod m_2$. Si noti che basta calcolare ciascun M_i e ciascun M_i^{-1} una sola volta. Prendendo i resti modulo 37 e 49, la rappresentazione di 973 diventa (11, 42) poiché 973 mod 37 = 11 e 973 mod 49 = 42.

[8] Questo esempio è stato fornito dal professor Ken Calvert di Georgia Tech.

Ora si supponga di voler sommare 678 e 973. Cosa si può fare a (11, 42)? Innanzitutto si calcola (678) $\leftrightarrow$ (678 mod 37, 678 mod 49) = (12, 41). Poi si sommano le tuple, elemento per elemento, e si riduce (11 + 12 mod 37, 42 + 41 mod 49) = (23, 34). Per verificare che l'effetto è corretto, si calcola:

$$\begin{aligned}(23,34) &\leftrightarrow a_1 M_1 M_1^{-1} + a_2 M_2 M_2^{-1} \bmod M \\ &= [(23)(49)(34) + (34)(37)(4)] \bmod 1813 \\ &= 43350 \bmod 1813 \\ &= 1651\end{aligned}$$

e si controlla che sia uguale a (973 + 678) mod 1813 = 1651. Si ricordi che nella derivazione precedente, M_1^{-1} è l'inversa moltiplicativa di M_1 modulo m_1, mentre M_2^{-1} è l'inversa moltiplicativa di M_2 modulo m_2.

Si supponga di voler moltiplicare 1651 (mod 1813) per 73. Si moltiplica (23, 24) per 73 e si riduce per ottenere (23 × 73 mod 37, 34 × 73 mod 49) = (14, 32). È facile verificare che:

$$\begin{aligned}(14,32) &\leftrightarrow [(14)(49)(34) + (34)(37)(4)] \bmod 1813 \\ &= 865 \\ &= 1651 \times 73 \bmod 1813\end{aligned}$$

8.5 Logaritmi discreti

I logaritmi discreti sono fondamentali per vari algoritmi a chiave pubblica, fra cui l'algoritmo per lo scambio delle chiavi Diffie-Hellman e l'algoritmo per firme digitali DSA. Questa parte del capitolo offre una breve panoramica sui logaritmi discreti. Il lettore interessato ad approfondire questo argomento può consultare [ORE67] e [LEVE90].

Le potenze di un intero modulo *n*

Dal teorema di Eulero (Equazione 8.4) si sa che, per ogni a e n primi relativi vale che:

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

dove $\phi(n)$, la funzione toziente di Eulero, è il numero di interi positivi minori di n e primi relativi di n. Ora si consideri l'espressione più generale:

$$a^m \equiv 1 \pmod{n} \tag{8.10}$$

Se a e n sono primi relativi, allora vi è almeno un intero m che soddisfa l'Equazione 8.10, in particolare $m = \phi(n)$. Il più piccolo esponente positivo m per cui vale l'Equazione 8.10 può essere chiamato in vari modi:

- ordine di a (mod n);
- esponente cui appartiene a (mod n);

- lunghezza del periodo generato da a.

Per vedere quest'ultimo fatto, si considerino le potenze di 7, modulo 19:

$$7^1 \equiv 7 (\text{mod} 19)$$
$$7^2 = 49 = 2\times 19 + 11 \equiv 11 (\text{mod} 19)$$
$$7^3 = 343 = 18\times 19 + 1 \equiv 1 (\text{mod} 19)$$
$$7^4 = 2401 = 126\times 19 + 7 \equiv 7 (\text{mod} 19)$$
$$7^5 = 16807 = 884\times 19 + 11 \equiv 11 (\text{mod} 19)$$

Non vale la pena di continuare poiché la sequenza si ripete. Ciò può essere dimostrato notando che $7^3 \equiv 1 \pmod{19}$ e pertanto $7^{3+j} \equiv 7^3 7^j \equiv 7^j \pmod{19}$, e quindi tutte le potenze di 7 i cui esponenti differiscono di 3 (o multipli di 3) sono congruenti l'uno con l'altro (modulo 19). In altre parole, la sequenza è periodica e il periodo è il più piccolo esponente positivo m tale che $7^m \equiv 1 \pmod{19}$.

La Tabella 8. 3 mostra tutte le potenze di a, modulo 19, per tutti gli a positivi minori di 19. La lunghezza della sequenza per ciascun valore base è indicata dall'ombreggiatura. Si notino le seguenti proprietà.

1. Tutte le sequenze terminano con 1. Questo è coerente con il ragionamento dei precedenti paragrafi.
2. La lunghezza di una sequenza divide $\phi(19) = 18$. Ovvero in ogni riga della tabella si presenta un numero intero di sequenze.
3. Alcune delle sequenze sono di lunghezza 18. In questo caso si dice che l'intero base a genera (tramite le potenze) l'insieme degli interi diversi da 0 modulo 19. Ognuno di questi interi è chiamato radice primitiva del modulo 19.

Più in generale si può dire che il più alto esponente possibile cui un numero può appartenere modulo n è $\phi(n)$. Se un numero è di questo ordine, viene chiamato **radice primitiva** di n. L'importanza di questa nozione è il fatto che se a è una radice primitiva di n, allora le sue potenze:

$$a, a^2, \ldots, a^{\phi(n)}$$

sono distinte (modulo n) e sono tutte prime relative di n. In particolare, per un numero primo p, se a è una radice primitiva di p, allora:

$$a, a^2, \ldots, a^{p-1}$$

sono numeri distinti (modulo p). Per esempio, le radici primitive del numero primo 19 sono 2, 3, 10, 13, 14 e 15.

Non tutti gli interi hanno radici primitive. Infatti i soli interi con radici primitive sono nella forma 2, 4, p^α e $2p^\alpha$, dove p è un numero primo dispari e α è un intero positivo. La dimostrazione non è semplice ma si trova in numerosi testi di teoria dei numeri, tra i quali [ORE76].

Tabella 8.3 Le potenze degli interi modulo 19.

a	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^{15}	a^{16}	a^{17}	a^{18}
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1
3	9	8	5	15	7	2	6	18	16	10	11	14	4	12	17	13	1
4	16	7	9	17	11	6	5	1	4	16	7	9	17	11	6	5	1
5	6	11	17	9	7	16	4	1	5	6	11	17	9	7	16	4	1
6	17	7	4	5	11	9	16	1	6	17	7	4	5	11	9	16	1
7	11	1	7	11	1	7	11	1	7	11	1	7	11	1	7	11	1
8	7	18	11	12	1	8	7	18	11	12	1	8	7	18	11	12	1
9	5	7	6	16	11	4	17	1	9	5	7	6	16	11	4	17	1
10	5	12	6	3	11	15	17	18	9	14	7	13	16	8	4	2	1
11	7	1	11	7	1	11	7	1	11	7	1	11	7	1	11	7	1
12	11	18	7	8	1	12	11	18	7	8	1	12	11	18	7	8	1
13	17	12	4	14	11	10	16	18	6	2	7	15	5	8	9	3	1
14	6	8	17	10	7	3	4	18	5	13	11	2	9	12	16	15	1
15	16	12	9	2	11	13	5	18	4	3	7	10	17	8	6	14	1
16	9	11	5	4	7	17	6	1	16	9	11	5	4	7	17	6	1
17	4	11	16	6	7	5	9	1	17	4	11	16	6	7	5	9	1
18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1

I logaritmi dell'aritmetica modulare

Con i normali numeri reali positivi, la funzione logaritmo è l'inverso della funzione esponente. Esiste una funzione analoga per l'aritmetica modulare.

Si proverà a ricapitolare brevemente le proprietà dei normali logaritmi. Il logaritmo di un numero è la potenza cui deve essere elevata una determinata base positiva (tranne 1) per ottenere il numero. Ovvero, per la base x e per il valore y:

$$y = x^{\log_x(y)}$$

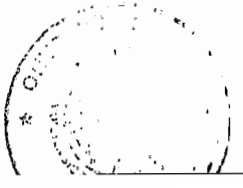
Fra le proprietà dei logaritmi vi sono le seguenti:

$$\begin{aligned} &\log_x(1) = 0 \\ &\log_x(x) = 1 \\ &\log_x(yz) = \log_x(y) + \log_x(z) \end{aligned} \tag{8.11}$$

$$\log_x(y^r) = r \times \log_x(y) \tag{8.12}$$

Si consideri una radice primitiva a per un numero primo p (l'argomento può essere sviluppato anche per numeri non primi). Allora si sa che le potenze di a da 1 a $(p-1)$ producono ciascun intero da 1 a $(p-1)$ esattamente una volta. Inoltre si sa che ogni intero b può essere espresso nella forma:

$$b \equiv r (\text{mod } p) \quad \text{per qualche } r \qquad \text{dove } 0 \le r \le (p-1)$$



Questo esponente i è chiamato **logaritmo discreto** del numero b per la base a (modulo p). Questo valore è denotato $\text{dlog}_{a,p}(b)$.[9]

Si notino le seguenti relazioni.

$$\text{dlog}_{a,p}(1) = 0, \text{ poiché } a^0 \bmod p = 1 \bmod p = 1 \tag{8.13}$$

$$\text{dlog}_{a,p}(a) = 1, \text{ poiché } a^1 \bmod p = a \tag{8.14}$$

Ecco un esempio che utilizza un modulo non primo, $n = 9$. Qui $\phi(n) = 6$ e $a = 2$ è una radice primitiva. Si calcolano le varie potenze di a e si trova che:

$$2^0 = 1$$
$$2^1 = 2$$
$$2^2 = 4$$
$$2^3 = 8$$
$$2^4 \equiv 7 (\text{mod } 9)$$
$$2^5 \equiv 5 (\text{mod } 9)$$
$$2^6 \equiv 1 (\text{mod } 9)$$

Questo fornisce la seguente tabella di numeri con determinati logaritmi discreti (modulo n) per la radice $a = 2$:

Logaritmo	0	1	2	3	4	5
Numero	1	2	4	8	7	5

Per ottenere facilmente il logaritmo discreto di un determinato numero si dispone la tabella come segue:

Indice	1	2	4	5	7	8
Logaritmo	0	1	2	5	4	3

Ora si consideri:

$$x = a^{\text{dlog}_{a,p}(x)} \bmod p$$
$$y = a^{\text{dlog}_{a,p}(y)} \bmod p$$
$$xy = a^{\text{dlog}_{a,p}(xy)} \bmod p$$

Utilizzando le regole della moltiplicazione modulare:

$$xy \bmod p = [(x \bmod p)(y \bmod p)] \bmod p$$
$$a^{\text{dlog}_{a,p}(xy)} \bmod p = [(a^{\text{dlog}_{a,p}(x)} \bmod p)(a^{\text{dlog}_{a,p}(y)} \bmod p)] \bmod p$$
$$= (a^{\text{dlog}_{a,p}(x) + \text{dlog}_{a,p}(y)}) \bmod p$$

[9] Il logaritmo discreto è chiamato, in numerosi testi, **indice**. Non vi è ancora una notazione universalmente adottata per questo concetto.

Ma si ora consideri il teorema di Eulero che afferma che, se a e n sono numeri primi relativi:

$$a^{\phi(n)} \equiv 1 (\text{mod } n)$$

Qualsiasi intero positivo z può essere espresso nella forma $z = q + k\phi(n)$, con $0 \le q < \phi(n)$. Pertanto, in base al teorema di Eulero:

$$a^z \equiv a^q \bmod n \qquad \text{se } z = q \bmod \phi(n)$$

Applicando questo risultato all'uguaglianza precedente, si ha:

$$\text{dlog}_{a,p}(xy) \equiv [\text{dlog}_{a,p}(x) + \text{dlog}_{a,p}(y)] (\text{mod } \phi(p))$$

e in generale:

$$\text{dlog}_{a,p}(y^r) \equiv [r \times \text{dlog}_{a,p}(y)] (\text{mod } \phi(p))$$

Questo dimostra l'analogia fra i veri logaritmi e i logaritmi discreti.

Si deve tenere in considerazione che i logaritmi discreti univoci modulo m di una base a esistono solo se a è una radice primitiva di m.

La Tabella 8.4, tratta direttamente dalla Tabella 8.3, mostra gli insiemi di logaritmi discreti che possono essere definiti modulo 19.

Calcolo dei logaritmi discreti

Si consideri l'equazione:

$$y = g^x \bmod p$$

Dati g, x e p, è immediato calcolare y. Nella peggiore delle ipotesi si devono eseguire x moltiplicazioni ma esistono algoritmi con un'efficienza superiore (vedere il Capitolo 9).

Tuttavia, dati y, g e p è, in generale, molto difficile calcolare x (il logaritmo discreto). La difficoltà sembra essere dello stesso ordine di grandezza della fattorializzazione dei primi richiesta per RSA. Al momento attuale l'algoritmo asintoticamente più veloce per calcolare i logaritmi discreti modulo un certo numero primo è dell'ordine di [BETH91]:

$$e^{((\ln p)^{1/3}(\ln(\ln p))^{2/3})}$$

ovvero il calcolo è impossibile per numeri primi di grandi dimensioni.

Tabella 8.4 Tabelle dei logaritmi discreti modulo 19.

A. Logaritmi discreti in base 2, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\text{dlog}_{2,19}(a)$	18	1	13	2	16	14	6	3	8	17	12	15	5	7	11	4	10	9

(segue)

Tabella 8.4 Tabelle dei logaritmi discreti modulo 19. *(continua)*

B. Logaritmi discreti in base 3, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\text{dlog}_{3,19}(a)$	18	7	1	14	4	8	6	3	2	11	12	15	17	13	5	10	16	9

C. Logaritmi discreti in base 10, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\text{dlog}_{10,19}(a)$	18	17	5	16	2	4	12	15	10	1	6	3	13	11	7	14	8	9

D. Logaritmi discreti in base 13, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\text{dlog}_{13,19}(a)$	18	11	17	4	14	10	12	15	16	7	6	3	1	5	13	8	2	9

E. Logaritmi discreti in base 14, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\text{dlog}_{14,19}(a)$	18	13	7	8	10	2	6	3	14	5	12	15	11	1	17	16	4	9

F. Logaritmi discreti in base 15, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\text{dlog}_{15,19}(a)$	18	5	11	10	8	16	12	15	4	13	6	3	7	17	1	2	14	9

8.6 Letture e siti Web consigliati

Esistono vari testi sulla teoria dei numeri che forniscono molti più dettagli di quanto i lettori di questo testo probabilmente desiderano. Un'introduzione forse un po' elementare ma utile è [ORE67]. Per il lettore interessato a una trattazione più approfondita, due testi eccellenti sull'argomento sono [KUMA98] e [ROSE00]. Anche [LEVE90] è un testo di facile lettura ma dettagliato. Questi volumi includono anche problemi con relative soluzioni che consentono di studiare l'argomento in modo autonomo.

Per i lettori disponibili a dedicarvi il tempo necessario, forse il modo migliore per acquisire una solida comprensione della teoria dei numeri è [BURN97], costituito unicamente da esercizi e soluzioni che conducono lo studente passo dopo passo nella teoria dei numeri; completare tutti gli esercizi equivale a terminare un corso di teoria dei numeri.

BURN97 R. Burn. *A Pathway to Number Theory*. Cambridge, England: Cambridge University Press, 1997.

KUMA98 R. Kumanduri e C. Romero. *Number Theory with Computer Applications*. Upper Saddle River, NJ: Prentice Hall, 1998.

LEVE90 W. Leveque. *Elementary Theory of Numbers*. New York: Dover, 1990.

ORE67 O. Ore. *Invitation to Number Theory*. Washington, DC: The Mathematical Association of America, 1967.

ROSE00 K. Rosen. *Elementary Number Theory and Its Applications*. Reading, MA: Addison-Wesley, 2000.

Sito Web consigliato

The Prime Pages: ricerche, note e risorse sui numeri primi.

8.7 Termini chiave, domande di ripasso e problemi

Termini chiave

Biiezione
Numero composito
Teorema cinese del resto
Logaritmo discreto
Teorema di Eulero
Funzione toziente di Eulero
Teorema di Fermat
Indice
Ordine
Numero primo
Radice primitiva

Domande di ripasso

8.1 Che cos'è un numero primo?
8.2 Qual è il significato dell'espressione *a divide b*?
8.3 Che cos'è il toziente di Eulero?
8.4 Il test di Miller-Rabin può determinare se un numero non è primo ma non può determinare se è primo. In quale modo si può utilizzare un tale algoritmo per verificare se un numero è primo?
8.5 Qual è la radice primitiva di un numero?
8.6 Qual è la differenza fra un indice e un logaritmo discreto?

Problemi

8.1 Lo scopo di questo problema è determinare la quantità di numeri primi. Si supponga che vi sia un totale di n numeri primi, elencati in sequenza:

$$p_1 = 2 < p_2 = 3 < p_3 = 5 < \ldots < p_n.$$

1. Definire $X = 1 + p_1 p_2 \ldots p_n$. In altre parole, X è uguale a 1 più il prodotto di tutti i primi. È possibile trovare un numero primo p_m che divide X?
2. Cosa si può dire di m?
3. Dedurre che il numero totale di primi non può essere finito.
4. Dimostrare che $p_{n+1} \leq 1 + p_1 p_2 \ldots p_n$.

8.2 Lo scopo di questo problema è quello di dimostrare che la probabilità che due numeri casuali siano primi relativi è pari a circa 0,6.

A. Sia $P = \Pr[\gcd(a,b) = 1]$. Dimostrare che $\Pr[\gcd(a,b) = d] = P/d^2$. *Suggerimento:* considerare la quantità $\gcd\left(\frac{a}{d}, \frac{b}{d}\right)$.

B. La somma del risultato della parte A per tutti i valori possibili di d è 1. Ovvero $\sum_{d \geq 1} \Pr[\gcd(a,b) = d] = 1$. Usare questa uguaglianza per determinare il valore di P. *Suggerimento:* utilizzare l'identità $\sum_{i=1}^{\infty} \frac{1}{i^2} = \frac{\pi^2}{6}$.

8.3 Perché $\gcd(n, n+1) = 1$ per due interi consecutivi n e $n+1$?

8.4 Utilizzando il teorema di Fermat, trovare 3^{201} modulo 11.

8.5 Utilizzare il teorema di Fermat per trovare un numero a compreso fra 0 e 72, con a congruente a 9794 modulo 73.

8.6 Utilizzare il teorema di Fermat per trovare un numero x compreso fra 0 e 28, con x^{85} congruente a 6 modulo 29. Nota: non è necessaria una ricerca esaustiva.

8.7 Utilizzare il teorema di Fermat per trovare un numero x compreso fra 0 e 9, tale che x sia congruente a 7^{1000} modulo 10. Si noti che questo corrisponde all'ultima cifra dell'espansione decimale di 7^{1000}.

8.8 Utilizzare il teorema di Fermat per trovare un numero x compreso fra 0 e 28, con x^{85} congruente a 6 modulo 35. Nota: non è necessaria una ricerca esaustiva.

8.9 Si noti nella Tabella 8.2 che $\phi(n)$ è pari per $n > 2$. Ciò vale per ogni $n > 2$. Spiegare brevemente perché.

8.10 Dimostrare la seguente asserzione.
Se t è primo, allora $\phi(p^i) = p^i - p^{i-1}$. *Suggerimento:* quali numeri hanno un fattore in comune con p^i?

8.11 Si può dimostrare (consultare un qualsiasi volume sulla teoria dei numeri) che se $\gcd(m,n) = 1$, allora $\phi(mn) = \phi(m)\phi(n)$. Utilizzando questa proprietà, la proprietà sviluppata nel problema precedente e la proprietà che $\phi(p) = p - 1$ per p primo, è facile determinare il valore di $\phi(n)$ per ogni n. Determinare i seguenti valori:

a. $\phi(41)$ **b.** $\phi(27)$ **c.** $\phi(231)$ **d.** $\phi(440)$

8.12 Dimostrare che per un intero positivo arbitrario a, $\phi(a)$ è dato da:

$$\phi(a) = \prod_{i=1}^{t} \left[p_i^{a_i - 1} (p_i - 1) \right]$$

dove a è dato dall'Equazione 8.1, ovvero $a = p_1^{a_1} p_2^{a_2} \ldots p_t^{a_t}$.

8.13 Considerare la funzione $f(n)$ = numero di elementi nell'insieme $\{a: 0 \leq a < n \text{ e } \gcd(a, n) = 1\}$ Di che funzione si tratta?

8.14 Sebbene gli antichi matematici cinesi abbiano fatto un buon lavoro nel formulare il teorema del resto, questo non sempre funziona. Avevano sviluppato un test per la primalità. Il test dice che n è primo se e solo se n divide $(2^n - 2)$.

A. Dare un esempio che soddisfa la condizione utilizzando un numero primo dispari.

B. La condizione è ovviamente vera per $n = 2$. Dimostrare che la condizione è vera se n è un numero primo dispari (dimostrando la condizione **se**).

C. Dare un esempio di un valore n dispari che non è primo e che non soddisfa la condizione. Si può svolgere l'operazione con numeri non primi fino a un valore molto esteso. Questo ha erroneamente portato i matematici cinesi a pensare che se la condizione è vera, allora n è primo.

D. Sfortunatamente, gli antichi cinesi non hanno mai provato $n = 341$ che non è primo ($341 = 11 \times 31$) e che comunque divide $2^{341} - 2$ senza resto. Dimostrare che $2^{341} \equiv 2 \pmod{341}$ (dimostrando che la condizione **solo se** è falsa). *Suggerimento:* non è necessario calcolare 2^{341}; basta lavorare con le congruenze.

8.15 Mostrare che se n è un intero composito dispari, allora il test di Miller-Rabin restituisce "incerto" per $a = 1$ e $a = (n - 1)$.

8.16 Se n è composito e passa il test di Miller-Rabin per la base a, allora n è chiamato *pseudoprimo forte per la base a*. Mostrare che 2047 è uno pseudoprimo forte per la base 2.

8.17 Una formulazione comune del teorema cinese del resto è la seguente: siano $m_1, \ldots, m_k$ numeri interi i quali siano a coppie primi relativi per $1 \leq i, j \leq k$ e $i \neq j$. Definire M come il prodotto di tutti gli m_i. Siano $a_1, \ldots, a_k$ dei valori interi. Allora l'insieme delle congruenze:

$$\begin{aligned} x &\equiv a_1 \pmod{m_1} \\ x &\equiv a_2 \pmod{m_2} \\ &\vdots \\ x &\equiv a_k \pmod{m_k} \end{aligned}$$

ha una soluzione univoca modulo M. Dimostrare che il teorema così formulato è valido.

8.18 L'esempio utilizzato da Sun-Tse per illustrare il teorema cinese del resto è il seguente:

$$x \equiv 2 \pmod 3; \quad x \equiv 3 \pmod 5; \quad x \equiv 2 \pmod 7$$

Risolvere per x.

8.19 Sei professori iniziano il proprio corso rispettivamente al lunedì, martedì, mercoledì, giovedì, venerdì e sabato e annunciano la propria intenzione di tenere lezioni rispet-

tivamente a intervalli di 2, 3, 4, 1, 6 e 5 giorni. I regolamenti universitari proibiscono di tenere lezioni di domenica (dunque la lezione di domenica deve essere saltata). Quando tutti i sei professori saranno costretti a saltare la lezione la prima volta? *Suggerimento:* utilizzare il teorema cinese del resto.

8.20 Trovare tutte le radici primitive di 25.

8.21 Dato 2 come radice primitiva di 29, costruire la tabella degli indici e utilizzarla per risolvere le seguenti congruenze:

A. $17x^2 \equiv 10 \pmod{29}$

B. $x^2 - 4x - 16 \equiv 0 \pmod{29}$

C. $x^7 \equiv 17 \pmod{29}$

Esercizi di programmazione

8.22 Sviluppare un programma che implementi l'esponenziazione veloce (quadrati successivi) modulo n.

8.23 Sviluppare un programma che implementi l'algoritmo di Miller-Rabin con n specificato dall'utente. Il programma dovrebbe consentire all'utente due opzioni: (1) specificare un possibile testimone a per la verifica con la procedura Witness, o (2) specificare un numero s di testimoni casuali per la verifica con Miller-Rabin.

Capitolo 9

Crittografia a chiave pubblica e RSA

Concetti essenziali

- La **crittografia asimmetrica** è un tipo di crittografia nella quale sia la crittografia sia la decrittografia utilizzano due chiavi differenti: una chiave pubblica e una privata. È chiamata anche crittografia a chiave pubblica.
- La crittografia asimmetrica trasforma il testo in chiaro in testo cifrato utilizzando l'algoritmo di crittografia con una delle due chiavi. Il testo in chiaro viene ottenuto dal testo cifrato utilizzando l'algoritmo di decrittografia con l'altra chiave.
- La crittografia asimmetrica può essere utilizzata per la segretezza, l'autenticazione o entrambe.
- Il sistema di crittografia a chiave pubblica più diffuso è **RSA**. La difficoltà nell'attaccare RSA dipende dalla difficoltà nell'identificare i fattori primi di un numero composto.

Lo sviluppo della crittografia a chiave pubblica è la più grande e forse l'unica vera rivoluzione nell'intera storia della crittografia. Dagli inizi e fino a tempi recenti, praticamente tutti i sistemi crittografici si sono basati sugli elementari strumenti della sostituzione e della permutazione. Dopo aver dovuto lavorare per millenni con algoritmi che dovevano fondamentalmente essere calcolati manualmente, si verificò un importante miglioramento nel campo della crittografia simmetrica con lo sviluppo della macchina di crittografia/decrittografia a rotazione. Il rotore elettromeccanico consentì lo sviluppo di sistemi di cifratura estremamente complessi. Con l'avvento dei computer, divenne possibile progettare sistemi ancora più complessi, il più notevole dei quali fu Lucifer sviluppato da IBM, culminato nello sviluppo dello standard di crittografia DES (Data Encryption Standard). Ma sia le macchine a rotore che DES, sebbene rappresentino importanti miglioramenti, si basavano comunque sui classici strumenti della sostituzione e della permutazione.

La crittografia a chiave pubblica si distacca in modo radicale da tutto ciò che l'ha preceduta. Innanzitutto gli algoritmi a chiave pubblica si basano su funzioni matematiche e non sulle operazioni di sostituzione e permutazione. Ancora più importante è il fatto che

la crittografia a chiave pubblica è asimmetrica e prevede l'uso di due chiavi distinte, mentre nel caso della crittografia simmetrica viene utilizzata una sola chiave. L'uso di due chiavi ha profonde conseguenze nel campo della segretezza, della distribuzione delle chiavi e dell'autenticazione.

Prima di procedere occorre sfatare alcuni luoghi comuni riguardanti la crittografia a chiave pubblica. Uno di questi è la credenza che la crittografia a chiave pubblica sia più resistente all'analisi crittografica rispetto alla crittografia simmetrica. In realtà la sicurezza di uno schema di crittografia dipende dalla lunghezza della chiave e dal calcolo necessario per violare l'algoritmo. Non vi è alcun elemento di principio che renda la crittografia simmetrica o la crittografia a chiave pubblica superiore rispetto all'altra dal punto di vista della resistenza all'analisi crittografica.

Un secondo luogo comune è il fatto che la crittografia a chiave pubblica sia una tecnica di utilizzo generale che ha reso obsoleta la crittografia simmetrica. Al contrario, dati i requisiti di calcolo degli schemi attuali di crittografia a chiave pubblica, non è previsto che la crittografia simmetrica venga abbandonata. Ecco quello che sostiene uno degli inventori della crittografia a chiave pubblica [DIFF88]: "la restrizione della crittografia a chiave pubblica all'ambito della gestione delle chiavi e delle firme è quasi universalmente accettata".

Infine vi è la sensazione che la distribuzione della chiave pubblica sia molto più semplice della procedura di distribuzione della chiave segreta per la crittografia simmetrica che prevede l'impiego di centri di distribuzione delle chiavi. In realtà è comunque necessario un protocollo che in genere prevede il collegamento con un ente centrale e le procedure coinvolte non sono né più semplici, né più efficienti rispetto a quelle richieste per la crittografia simmetrica (per esempio vedere l'analisi contenuta in [NEED78]).

Questo e il prossimo capitolo offrono una panoramica sulla crittografia a chiave pubblica, partendo dalla struttura concettuale. È molto interessante scoprire che i concetti di questa tecnica sono stati sviluppati e pubblicati prima che venisse dimostrata la praticità della loro adozione. Infine si esaminerà l'algoritmo RSA, l'algoritmo più importante nel campo della crittografia a chiave pubblica. Altri argomenti verranno esplorati nel Capitolo 10 e nell'Appendice F.

Molta della teoria dei sistemi di crittografia a chiave pubblica si basa sulla teoria dei numeri. Se si è pronti ad accettare i risultati forniti in questo capitolo, non sarà necessario avere una conoscenza della teoria dei numeri. Tuttavia, per apprezzare veramente gli algoritmi a chiave pubblica è opportuno conoscere, almeno a grandi linee, la teoria dei numeri, introdotta nel Capitolo 8.

9.1 I principi dei sistemi crittografici a chiave pubblica

Il concetto della crittografia a chiave pubblica sì è evoluto da un tentativo di attaccare due dei problemi più difficoltosi della crittografia simmetrica. Il primo problema, quello della distribuzione della chiave, è stato esaminato in dettaglio nel Capitolo 7.

Come si è detto, la distribuzione della chiave con la crittografia simmetrica richiede (1) che le due parti condividano già una chiave precedente oppure (2) l'uso di un centro di distribuzione delle chiavi. Whitfield Diffie, uno degli inventori della crittografia a chiave pubblica (insieme a Martin Hellman, anch'egli, a quel tempo, all'università di Stanford) ritenevano che questo secondo requisito negasse l'essenza stessa della crittografia: la capacità di garantire una totale segretezza delle comunicazioni. Come ha detto Diffie in [DIFF88], "che scopo ha sviluppare sistemi crittografici impenetrabili se poi gli utenti sono costretti a condividere le chiavi con un centro che può essere violato?".

Il secondo problema affrontato da Diffie, e apparentemente non correlato con il primo, era quello delle "firme digitali". Se l'uso della crittografia avesse dovuto espandersi dall'ambito militare al campo commerciale e privato, allora i messaggi e i documenti elettronici avrebbero dovuto contenere delle firme digitali analoghe a quelle impiegate nei documenti cartacei. In pratica si doveva escogitare un metodo che garantisse, con soddisfazione di tutte le parti, che un messaggio digitale fosse stato effettivamente prodotto da una determinata persona. Si tratta di un requisito più ampio rispetto a quello dell'autenticazione, e le sue implicazioni verranno esplorate nel Capitolo 13.

Diffie e Hellman ottennero un fondamentale risultato nel 1976 [DIFF76 a, b] presentando un metodo che risolveva entrambi i problemi e che era radicalmente differente da tutti gli approcci precedenti alla crittografia.[1]

Nella prossima parte del capitolo si discuterà il contesto globale della crittografia a chiave pubblica. Poi verranno esaminati i requisiti degli algoritmi di crittografia/decrittografia che formano il cuore di questo schema.

Sistemi crittografici a chiave pubblica

Gli algoritmi a chiave pubblica usano una chiave per la crittografia e un'altra chiave, differente ma correlata, per la decrittografia. Questi algoritmi hanno le seguenti peculiarità.

- È computazionalmente impossibile determinare la chiave di decrittografia conoscendo l'algoritmo di crittografia e la chiave di crittografia.

Inoltre, alcuni algoritmi, come l'algoritmo RSA, esibiscono anche la seguente caratteristica.

- Per la crittografia è possibile utilizzare una qualsiasi delle due chiavi correlate, nel qual caso, per la decrittografia. verrà impiegata l'altra chiave.

Uno schema di crittografia a chiave pubblica prevede sei componenti (vedere la Figura 9.1.A da confrontare con la Figura 2.1).

- **Testo in chiaro:** il messaggio o i dati leggibili inviati come input all'algoritmo.
- **Algoritmo di crittografia:** l'algoritmo di crittografia che opera varie trasformazioni sul testo in chiaro.

[1] Diffie e Hellman hanno introdotto *pubblicamente* per primi i concetti della crittografia a chiave pubblica nel 1976. Tuttavia questo non è stato il vero inizio. L'ammiraglio Bobby Inman, mentre era direttore della NSA (National Security Agency), sostenne che la crittografia a chiave pubblica era stata scoperta dalla NSA alla metà degli anni 60 [SIMM93]. La prima introduzione *documentata* di questi concetti risale al 1970 ed è stata prodotta dal Communications-Electronics Security Group, l'analogo britannico della NSA, in un rapporto segreto di James Ellis [ELLI70] che ha chiamato questa tecnica crittografia non segreta e descrive la scoperta in [ELLI99].

- **Chiavi pubblica e privata:** la coppia di chiavi scelte in modo che se una viene utilizzata per la crittografia, l'altra viene usata per la decrittografia. Le esatte trasformazioni svolte dall'algoritmo di crittografia dipendono dalla chiave pubblica o privata fornita come input.

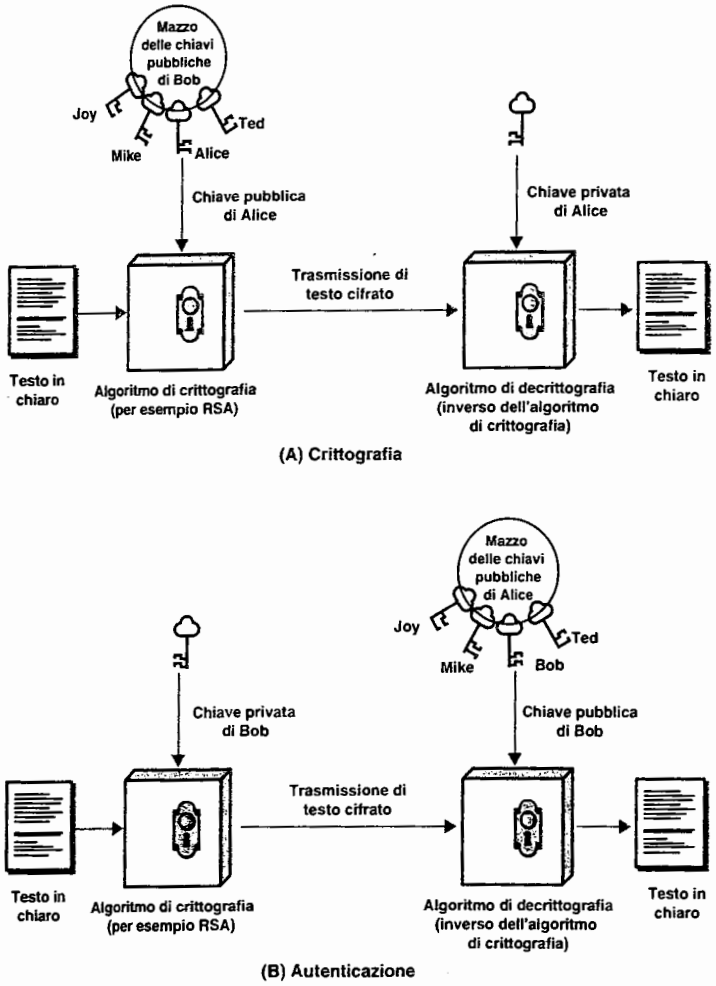


Figura 9.1 Crittografia a chiave pubblica.

- **Testo cifrato:** il messaggio codificato prodotto come output e che dipende dal testo in chiaro e dalla chiave. Per un determinato messaggio due diverse chiavi producono due testi cifrati differenti.
- **Algoritmo di decrittografia:** accetta il testo cifrato e la chiave corrispondente e rigenera il testo in chiaro.

Ecco le operazioni svolte.

1. Ciascun utente genera una coppia di chiavi da utilizzare per la crittografia e la decrittografia dei messaggi.
2. Ciascun utente inserisce una delle due chiavi in un registro pubblico o in un file accessibile. L'altra chiave viene mantenuta segreta, privata. Come suggerisce la Figura 9.1A, ciascun utente gestisce una sorta di "mazzo" di chiavi pubbliche degli altri utenti.
3. Se Bob vuole inviare un messaggio confidenziale ad Alice, deve crittografarlo utilizzando la chiave pubblica di Alice.
4. Quando Alice riceve il messaggio, lo deve decrittografare utilizzando la propria chiave privata. Nessun altro può decrittografare il messaggio poiché solo Alice conosce la propria chiave privata.

Seguendo questo approccio, le chiavi pubbliche saranno liberamente distribuibili mentre le chiavi private verranno generate localmente da ogni partecipante e non devono mai essere distribuite. Finché la chiave privata di un utente rimane protetta e segreta, tutte le comunicazioni in ingresso rimangono sicure. Un sistema può cambiare in qualsiasi momento la propria chiave privata e pubblicare la nuova chiave pubblica per sostituire la vecchia.

La Tabella 9.1 riepiloga alcuni degli aspetti più importanti della crittografia simmetrica e della crittografia a chiave pubblica. Per distinguerle, la chiave utilizzata per la crittografia simmetrica verrà generalmente chiamata **chiave segreta**. Le due chiavi utilizzate per la crittografia a chiave pubblica vengono invece chiamate **chiave pubblica** e **chiave privata**.[2] Anche la chiave privata deve essere mantenuta segreta ma è chiamata chiave privata per evitare confusioni con la crittografia simmetrica.

Ora si utilizzerà la Figura 9.2 (da confrontare con la Figura 2.2) per descrivere gli elementi fondamentali di uno schema di crittografia a chiave pubblica. Una sorgente A produce un messaggio in chiaro, $X = [X_1, X_2, \ldots, X_M]$. Gli M elementi di X sono lettere di un alfabeto finito. Il messaggio deve essere inviato alla destinazione B. B genera una coppia di chiavi correlate: una chiave pubblica, PU_b, e una chiave privata, PR_b. La chiave privata PR_b è nota solo a B mentre la chiave pubblica PU_b deve, ovviamente, essere accessibile ad A.

[2] Verrà utilizzata la seguente notazione. Una chiave segreta è rappresentata come K_m, dove m è un modificatore; per esempio K_a è la chiave segreta dell'utente A. Una chiave pubblica è rappresentata come PUa per l'utente A mentre la chiave privata corrispondente sarà PR_a. La crittografia del testo in chiaro può essere eseguita con una chiave segreta, una chiave pubblica o una chiave privata; le operazioni sono rappresentate rispettivamente come $E(K_a,X)$, $E(PU_a,X)$ e $E(PR_a)$. Analogamente, la decrittografia del testo cifrato C può essere eseguita con una chiave segreta, una chiave pubblica o una chiave privata; le operazioni sono rappresentate rispettivamente come $D(K_a,X)$, $D(PU_a,X)$ e $D(PR_a,X)$.

Tabella 9.1 Crittografia convenzionale e crittagrafia a chiave pubblica.

Crittografia convenzionale	Crittografia a chiave pubblica
Requisiti di funzionamento	*Requisiti di funzionamento*
1. Per la crittografia e decrittografia viene utilizzato lo stesso algoritmo con la stessa chiave.	1. Viene utilizzato un unico algoritmo per la crittografia e la decrittografia con una coppia di chiavi: una per la crittografia e una per la decrittografia.
2. Il mittente e il destinatario devono condividere l'algoritmo e la chiave.	2. Il mittente e il destinatario devono utilizzare una coppia di chiavi correlate ma distinte.
Requisiti per la sicurezza	*Requisiti per la sicurezza*
1. La chiave deve essere mantenuta segreta.	1. Una delle due chiavi deve essere mantenuta segreta.
2. Deve essere impossibile o quanto meno impraticabile decifrare un messaggio senza avere a disposizione altre informazioni.	2. Deve essere impossibile o quanto meno impraticabile decifrare un messaggio senza avere a disposizione altre informazioni.
3. La conoscenza dell'algoritmo e di campioni di testo cifrato non deve consentire di determinare la chiave.	3. La conoscenza dell'algoritmo, di una delle due chiavi e di campioni di testo cifrato non deve consentire di determinare l'altra chiave.

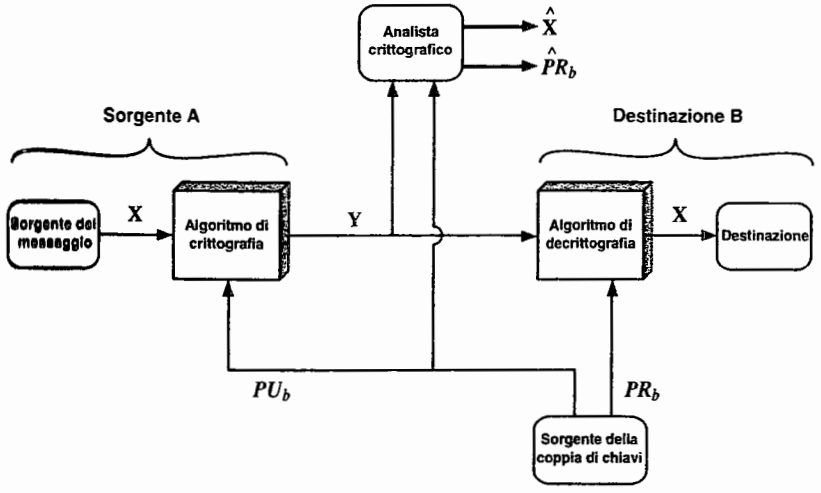


Figura 9.2 Sistemi crittografici a chiave pubblica: la segretezza.

Utilizzando come input il messaggio X e la chiave di crittografia KU_b, A genera il testo cifrato $Y = [Y_1, Y_2, \ldots, Y_N]$.

$$Y = E(PU_b, X)$$

Il destinatario, in possesso della chiave privata corrispondente, potrà invertire la trasformazione:

$$X = D(PR_b, Y)$$

Un avversario, osservando Y e avendo accesso a PU_b ma non avendo accesso a PR_b o X deve tentare di scoprire X e/o PR_b. Si suppone che l'avversario conosca anche gli algoritmi di crittografia E e di decrittografia D. Se l'avversario fosse interessato solo a un determinato messaggio, si concentrerebbe sul recupero di X generando una stima di testo in chiaro $\hat{X}$. Ma spesso l'avversario è interessato a leggere anche tutti i messaggi futuri, nel qual caso cercherà di individuare PR_b generando una stima PR_i.

Si è detto che per la crittografia è possibile utilizzare una qualsiasi delle due chiavi correlate, utilizzando l'altra per la decrittografia. Questo consente di implementare uno schema crittografico piuttosto differente. Mentre lo schema illustrato nella Figura 9.2 garantisce la segretezza, le Figure 9.1B e 9.3 mostrano l'utilizzo della crittografia a chiave pubblica per garantire l'autenticazione:

$$Y = E(PR_a, X)$$
$$X = D(PU_a, Y)$$

In questo caso, A prepara un messaggio per B e lo codifica utilizzando la propria chiave privata prima di trasmetterlo. B può decrittografare il messaggio utilizzando la chiave pub-

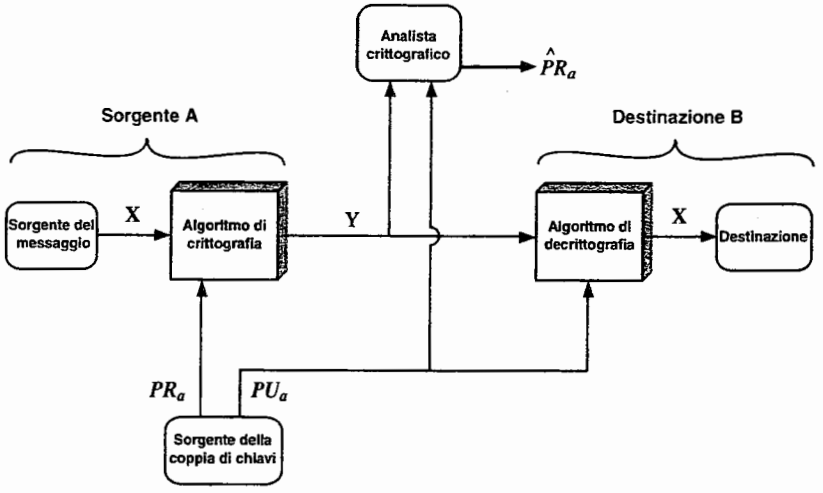


Figura 9.3 Sistema crittografico a chiave pubblica: autenticazioni.

blica di A. Poiché il messaggio è stato crittografato utilizzando la chiave privata di A, solo A può aver preparato il messaggio. Pertanto l'intero messaggio crittografato funge in sostanza da firma digitale. Inoltre è impossibile alterare il messaggio senza accedere alla chiave privata di A e dunque il messaggio è autenticato sia in termini di mittente che in termini di integrità dei dati.

Nello schema precedente viene crittografato l'intero messaggio; questo metodo, sebbene convalidi sia l'autore che il contenuto, richiede una grande area di memorizzazione. Ciascun documento deve essere mantenuto in chiaro per poter essere utilizzato. Una copia deve però essere conservata in testo cifrato in modo da poter verificare l'origine e il contenuto in caso di disputa. Un modo più efficiente per ottenere lo stesso risultato è quello di crittografare solo un piccolo blocco di bit che rappresenta una funzione del documento. Tale blocco di autenticazione (chiamato anche autenticatore) deve essere tale che sia impossibile modificare il documento senza modificare anche il blocco di autenticazione. Se l'autenticatore viene crittografato con la chiave privata del mittente, fungerà da firma digitale per verificare l'origine, il contenuto e la sequenza. Il Capitolo 13 esamina più in dettaglio questa tecnica.

È importante notare che il processo di crittografia riportato nelle Figure 9.1b e 9.3 non garantisce la segretezza. In pratica, il messaggio inviato è al sicuro contro le alterazioni ma non contro le intercettazioni. Questo è ovvio nel caso di una firma basata su una porzione del messaggio in quanto la parte rimanente del messaggio verrà trasmessa in chiaro. Anche nel caso di una crittografia completa, come indicato nella Figura 9.3, non vi è alcuna segretezza in quanto chiunque riuscisse a impossessarsi di una copia del messaggio potrà decrittografarla utilizzando la chiave pubblica del mittente.

È però possibile garantire sia la funzione di autenticazione che la segretezza tramite una doppia applicazione a chiave pubblica (Figura 9.4):

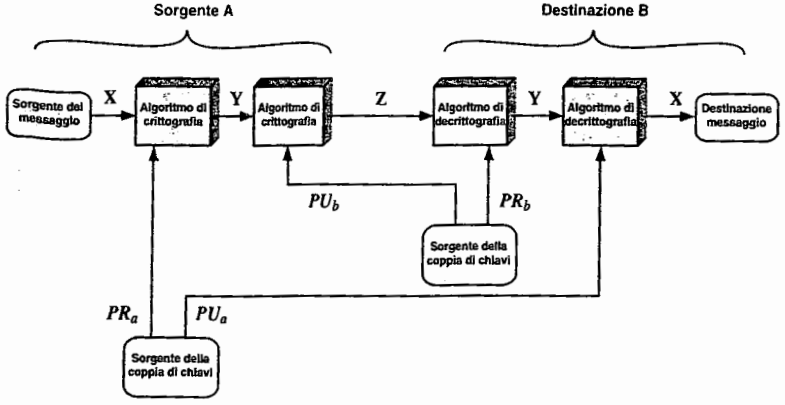


Figura 9.4 Sistemi crittografici a chiave pubblica: segretezza e autenticazione.

$$Z = E(PU_b, E(PR_a, X))$$
$$X = D(PU_a, D(PR_b, Z))$$

In questo caso, si comincia (come prima) a crittografare un messaggio utilizzando la chiave privata del mittente. Questo garantisce la firma digitale. Poi si esegue nuovamente la crittografia utilizzando la chiave pubblica del destinatario. In questo modo il testo cifrato prodotto potrà essere decrittografato solo utilizzando la chiave privata del destinatario. Questo pertanto garantisce la segretezza. Lo svantaggio di questo approccio consiste nel fatto che l'algoritmo a chiave pubblica, già di per sé complesso, deve essere applicato per quattro volte invece di due.

Applicazioni dei sistemi crittografici a chiave pubblica

Prima di procedere occorre chiarire un aspetto dei sistemi crittografici a chiave pubblica che potrebbe essere fonte di confusione. I sistemi a chiave pubblica sono caratterizzati dall'utilizzo di un algoritmo crittografico con due chiavi, una privata e una pubblica. A seconda dell'applicazione, il mittente utilizza la propria chiave privata, la chiave pubblica del destinatario o entrambe. In generale è possibile classificare i sistemi crittografici a chiave pubblica in tre categorie.

- **Crittografia/decrittografia:** il mittente esegue la crittografia di un messaggio con la chiave pubblica del destinatario.
- **Firma digitale:** il mittente "firma" un messaggio con la propria chiave privata. La firma è ottenuta applicando un algoritmo crittografico al messaggio o a un piccolo blocco di dati che rappresenta una funzione del messaggio.
- **Scambio delle chiave:** le due parti si scambiano una chiave di sessione. È possibile impiegare vari approcci che prevedono l'impiego delle chiavi private di una o di entrambe le parti.

Alcuni algoritmi possono essere impiegati per tutte e tre le applicazioni mentre altri possono essere utilizzati solo per una o due applicazioni. La Tabella 9.2 indica le applicazioni supportate dagli algoritmi trattati in questo volume.

Tabella 9.2 Applicazioni dei sistemi crittografici a chiave pubblica.

Algoritmo	Crittografia/Decrittografia	Firma digitale	Scambio delle chiavi
RSA	Sì	Sì	Sì
Curva ellittica	Sì	Sì	Sì
Diffie-Hellman	No	No	Sì
DSS	No	Sì	No

Requisiti della crittografia a chiave pubblica

Il sistema crittografico rappresentato nelle Figure da 9.2 a 9.4 dipende da un algoritmo crittografico che si basa su due chiavi correlate. Diffie e Hellman idearono questo sistema senza dimostrare l'esistenza di tali algoritmi; ciononostante definirono i requisiti che devono essere rispettati da tali algoritmi [DIFF76b].

1. Deve essere computazionalmente facile per B generare una coppia (chiave pubblica PU_b, chiave privata PR_b).
2. Deve essere computazionalmente facile per il mittente A, conoscendo la chiave pubblica e il messaggio da crittografare, M, generare il testo cifrato corrispondente:

$$C = E(PU_b, M)$$

3. Deve essere computazionalmente facile per il destinatario B decrittografare il testo cifrato risultante utilizzando la chiave privata:

$$M = D(PR_b, C) = D[PR_b, E(PU_b, M)]$$

4. Deve essere computazionalmente impossibile per un estraneo, conoscendo la chiave pubblica PU_b, determinare la chiave privata PR_b.
5. Deve essere computazionalmente impossibile per un estraneo, conoscendo la chiave pubblica PU_b e il testo cifrato C, ripristinare il messaggio originale M.

   Si può aggiungere anche un sesto requisito che, sebbene utile, non è necessario per tutte le applicazioni a chiave pubblica.

6. Le due chiavi possono essere applicate in qualsiasi ordine:

$$M = D[PU_b, E(PR_b, M)] = D[PR_b, E(PU_b, M)]$$

Si tratta di requisiti formidabili, come evidenziato dal fatto che pochi algoritmi (RSA, crittografia a curva ellittica, Diffie-Hellman, DSS) hanno ricevuto un'ampia accettazione da quando è stata proposta la crittografia a chiave pubblica.

Prima di discutere i motivi per cui questi requisiti sono così formidabili, è opportuno riformularli. Questi requisiti rimandano all'esigenza di avere una funzione trappola monodirezionale. Una funzione monodirezionale[3] mappa un dominio in un codominio in modo tale che ogni valore della funzione abbia un inverso univoco, con la condizione che il calcolo della funzione sia facile e il calcolo dell'inverso impossibile:

$Y = f(X)$ facile

$X = f^{-1}(Y)$ "impossibile"

Generalmente con "facile" si intende un problema che può essere risolto in un tempo polinomiale rispetto alla lunghezza dell'input. Pertanto se la lunghezza dell'input è di n bit, il tempo necessario per calcolare la funzione deve essere proporzionale a n^a dove a è una costante fissa. Tali algoritmi appartengono alla cosiddetta **Classe P**. Il termine *impossibile* è più difficile da definire. In generale si può dire che un problema è impossibile se l'impegno per risolverlo cresce più velocemente di un polinomio in funzione delle dimensioni dell'input. Per esempio, se la lunghezza dell'input è di n bit e il tempo necessario per calcolare la funzione è proporzionale a 2^n, il problema è considerato impossibile. Sfortunatamente è difficile determinare se un algoritmo esibisce questo livello di complessità. Inoltre, le nozioni tradizionali di complessità computazionale si concentrano sulla complessità dell'algoritmo nel caso peggiore o nel caso medio. Queste misure sono inadeguate per la crittografia che richiede l'impossibilità di invertire una funzione per virtualmente tutti gli input e non solo per il caso peggiore o il caso medio. Una breve introduzione a questi concetti si trova nell'Appendice 9.B.

[3] Da non confondere con le funzioni hash monodirezionali che prendono come argomento un campo arbitrariamente esteso e lo mappano in un output fisso. Tali funzioni vengono utilizzate per l'autenticazione (vedere il Capitolo 11).

Ora si vedrà che cos'è una funzione trappola monodirezionale, facile da calcolare in una direzione e impossibile da calcolare nell'altra direzione se non ottenendo alcune informazioni aggiuntive. Date le informazioni aggiuntive l'inversa può essere calcolata in un tempo polinomiale. Si può riepilogare il tutto nel seguente modo: una funzione trappola monodirezionale è una famiglia di funzioni invertibili f_k tali che:

$Y = f_k(X)$	facile se k e X sono noti
$X = f_k^{-1}(Y)$	facile se k e Y sono noti
$X = f_k^{-1}(Y)$	impossibile se Y è noto ma k è sconosciuto

Pertanto lo sviluppo di uno schema pratico a chiave pubblica dipende dalla scoperta di una funzione trappola monodirezionale adatta.

Analisi crittografica della chiave pubblica

Come la crittografia simmetrica, anche la crittografia a chiave pubblica è vulnerabile a un attacco a forza bruta. La contromisura è sempre la stessa: utilizzare chiavi di grandi dimensioni. Tuttavia occorre considerare un compromesso. I sistemi a chiave pubblica dipendono dall'uso di una funzione matematica invertibile. La complessità del calcolo di queste funzioni può non essere lineare rispetto al numero di bit della chiave ma può crescere più rapidamente. Pertanto le dimensioni della chiave devono essere sufficientemente grandi da rendere impossibile un attacco a forza bruta ma sufficientemente piccole da consentire l'esecuzione pratica della crittografia e decrittografia. In pratica le dimensioni delle chiavi che renderebbero impossibili gli attacchi a forza bruta generano velocità di crittografia/decrittografia troppo lente per gli utilizzi generali. Ma, come si è detto in precedenza, la crittografia a chiave pubblica è attualmente confinata alle sole applicazioni di gestione delle chiavi e di firma digitale.

Un'altra forma di attacco consiste nel trovare un modo per calcolare la chiave privata partendo dalla chiave pubblica. Attualmente non è matematicamente dimostrato che questa forma di attacco sia impossibile per un determinato algoritmo a chiave pubblica e pertanto qualsiasi algoritmo, compreso l'algoritmo RSA (così ampiamente utilizzato), è da considerarsi sospetto. La storia dell'analisi crittografica mostra che un problema che sem-

bra irrisolvibile da un punto di vista può avere una soluzione se lo si osserva da un'angolazione completamente differente.

Infine vi è una forma di attacco specifica contro i sistemi a chiave pubblica. Si tratterà, sostanzialmente, di un attacco a probabilità dei messaggi. Si supponga che venga inviato un messaggio costituito unicamente dalla chiave DES di 56 bit. Un estraneo potrebbe crittografare tutte le chiavi possibili utilizzando la chiave pubblica e potrebbe scoprire la chiave crittografata tramite un confronto con il testo cifrato trasmesso. Pertanto, indipendentemente dalle dimensioni della chiave pubblica, l'attacco si riduce a un attacco a forza bruta su una chiave a 56 bit. Questo attacco può essere vanificato aggiungendo dei bit casuali a tali messaggi particolarmente semplici.

9.2 L'algoritmo RSA

Il lavoro pioineristico di Diffie e Hellman [DIFF76b] ha introdotto un nuovo approccio alla crittografia e, in pratica, ha sfidato gli analisti crittografici a trovare un algoritmo crittografico che rispondesse ai requisiti dei sistemi a chiave pubblica. Una delle prime soluzioni venne sviluppata nel 1977 da Ron Rivest, Adi Shamir e Len Adleman del MIT ed è stata pubblicata per la prima volta nel 1978 [RIVE78].[4] Lo schema RSA (Rivest-Shamir-Adleman) è da allora l'approccio di carattere generale più ampiamente accettato e implementato nel campo della crittografia a chiave pubblica.

Lo schema RSA è una cifratura a blocchi in cui il testo in chiaro e il testo cifrato sono interi compresi fra 0 e $n - 1$ per un dato valore n. Normalmente n è pari a 1024 bit ovvero 309 cifre decimali. Ovvero, n è minore di 2^{1024}. In questa parte del capitolo si esaminerà in dettaglio l'algoritmo RSA e poi verranno esaminate alcune implicazioni computazionali e di analisi crittografica dell'algoritmo.

Descrizione dell'algoritmo

Lo schema sviluppato da Rivest, Shamir e Adleman utilizza un'espressione con valori esponenziali. Il testo in chiaro viene crittografato a blocchi, dove ciascun blocco è un valore binario minore di un certo numero n. In pratica il blocco deve avere dimensioni minori o uguali di $\log_2(n)$; ovvero deve avere dimensioni di k bit dove $2^k < n \leq 2^{k+1}$. La crittografia e la decrittografia di un determinato blocco di testo in chiaro M e del corrispondente blocco di testo cifrato C hanno la seguente forma:

$$C = M^e \bmod n$$
$$M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$$

[4] Apparentemente, il primo sistema a chiave pubblica impiegabile per la crittografia/decrittografia è stato sviluppato da Clifford Cocks del CESG britannico nel 1973 [COCK73]; il metodo di Cocks è praticamente identico a quello impiegato per RSA.

Il valore di n deve essere noto sia al mittente che al destinatario. Il mittente conosce il valore di e mentre solo il destinatario conosce il valore di d. Pertanto questo è un algoritmo con una chiave pubblica $PU = \{e, n\}$ e una chiave privata $PR = \{d, n\}$. Perché questo algoritmo possa essere soddisfacente per la crittografia a chiave pubblica, deve soddisfare i seguenti requisiti.

1. È possibile trovare i valori di e, d, n tali che $M^{ed} \bmod n = M$ per ogni $M < n$.
2. È relativamente facile calcolare $M^e \bmod n$ e $C^d \bmod n$ per tutti i valori di $M < n$.
3. È impossibile determinare d sulla base di e e n.

Per il momento ci si concentrerà sul primo requisito; gli altri problemi verranno affrontati in un secondo tempo. Si deve trovare una relazione della forma:

$$M^{ed} \bmod n = M$$

La relazione precedente è valida quando e e d sono inversi moltiplicativi modulo $\phi(n)$, dove $\phi(n)$ èla funzione toziente di Eulero. Si è illustrato nel Capitolo 8 che per p e q primi, si ha: $\phi(pq) = (p - 1)(q - 1)$. La relazione fra e e d può essere espressa come:

$$ed \bmod \phi(n) = 1 \qquad (9.1)$$

Questo equivale a dire:

$$ed \equiv 1 \bmod \phi(n)$$
$$d \equiv e^{-1} \bmod \phi(n)$$

In pratica, e e d sono inversi moltiplicativi mod $\phi(n)$. Si noti che, secondo le regole dell'aritmetica modulare, questo vale solo se d (e pertanto anche e) è primo relativo di $\phi(n)$. Di conseguenza $\gcd(\phi(n), d) = 1$. Si consulti l'Appendice 9A per la dimostrazione che l'equazione 9.1 soddisfa i requisiti di RSA.

Ora si può definire lo schema RSA. Ecco gli elementi.

p, q sono due numeri primi	(privati e scelti)
$n = pq$	(pubblico, calcolato)
e, con $\gcd(\phi(n), e) = 1$; $1 < e < \phi(n)$	(pubblico, scelto)
$d \equiv e^{-1} \pmod{\phi(n)}$	(privato, calcolato)

La chiave privata è costituita da $\{d, n\}$ e la chiave pubblica da $\{e, n\}$. Si supponga che l'utente A abbia pubblicato la propria chiave pubblica e che l'utente B voglia inviare ad A il messaggio M. Quindi B calcola $C = M^e \bmod n$ e trasmette C. Dopo aver ricevuto il testo cifrato C, l'utente A esegue la decrittografia calcolando $M = C^d \bmod n$.
La Figura 9.5 riepiloga il funzionamento dell'algoritmo RSA. Nella Figura 9.6 viene fornito un esempio tratto da [SING99]. Per questo esempio le chiavi vengono generate nel seguente modo.

1. Selezionare due numeri primi, $p = 17$ e $q = 11$.
2. Calcolare $n = pq = 17 \times 11 = 187$.
3. Calcolare $\phi(n) = (p - 1)(q - 1) = 16 \times 10 = 160$.
4. Selezionare e tale che sia primo relativo di $\phi(n) = 160$ e minore di $\phi(n)$. Si è scelto $e = 7$.

Generazione della chiave

Selezionare p, q	p e q entrambi primi e $p \neq q$.
Calcolare $n = p \times q$	
Calcolare $\phi(n) = (p-1)(q-1)$.	
Selezionare l'intero e	$\gcd(\phi(n), e) = 1;\ 1 < e < \phi(n)$
Calcolare d	$d \equiv e^{-1} \pmod{\phi(n)}$
Chiave pubblica	$PU = \{e, n\}$
Chiave privata	$PR = \{d, n\}$

Crittografia

Testo in chiaro:	$M < n$
Testo cifrato:	$C = M^e \bmod n$

Decrittografia

Testo cifrato:	C
Testo in chiaro:	$M = C^d \bmod n$

Figura 9.5 L'algoritmo RSA.

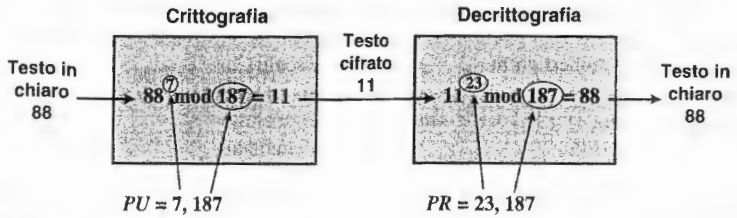


Figura 9.6 Esempio di algoritmo RSA.

5. Determinare d tale che $de \equiv 1 \pmod{160}$ e $d < 160$. Il valore corretto è $d = 23$ poiché $23 \times 7 = 161 = 10 \times 160 + 1$; d può essere calcolato utilizzando l'algoritmo esteso di Euclide (vedere il Capitolo 4).

Le chiavi risultanti sono la chiave pubblica $PU = \{7, 187\}$ e la chiave privata $PR = \{23, 187\}$. L'esempio mostra l'uso di queste chiavi per un input di testo in chiaro $M = 88$. Per la

crittografia occorre calcolare $C = 88^7 \bmod 187$. Sfruttando le proprietà dell'aritmetica modulare, si possono svolgere le seguenti operazioni:

$$88^7 \bmod 187 = [(88^4 \bmod 187) \times (88^2 \bmod 187) \times (88^1 \bmod 187)] \bmod 187$$
$$88^1 \bmod 187 = 88$$
$$88^2 \bmod 187 = 7744 \bmod 187 = 77$$
$$88^4 \bmod 187 = 59\,969\,536 \bmod 187 = 132$$
$$88^7 \bmod 187 = (88 \times 77 \times 132) \bmod 187 = 894\,432 \bmod 187 = 11$$

Per la decrittografia si calcola $M = 11^{23} \bmod 187$:

$$11^{23} \bmod 187 = [(11^1 \bmod 187) \times (11^2 \bmod 187) \times (11^4 \bmod 187) \times (11^8 \bmod 187) \times (11^8 \bmod 187)] \bmod 187$$
$$11^1 \bmod 187 = 11$$
$$11^2 \bmod 187 = 121$$
$$11^4 \bmod 187 = 14641 \bmod 187 = 55$$
$$11^8 \bmod 187 = 214\,458\,881 \bmod 187 = 33$$
$$11^{23} \bmod 187 = (11 \times 121 \times 55 \times 33 \times 33) \bmod 187 = 79\,720\,245 \bmod 187 = 88$$

Aspetti computazionali

Ora si valuterà la complessità di calcolo necessaria per utilizzare l'algoritmo RSA. Vi sono due elementi da considerare: la generazione della chiave e la crittografia/decrittografia. Innanzitutto si parlerà del processo di crittografia e decrittografia per poi considerare l'argomento della generazione della chiave.

Esponenziazione nell'aritmetica modulare

La crittografia e la decrittografia in RSA comportano l'elevamento di un intero a una potenza intera modulo n. Se l'esponenziazione venisse eseguita nel campo degli interi e poi ridotta modulo n, i valori intermedi diventerebbero immensi. Fortunatamente, come mostra l'esempio precedente, si può utilizzare una proprietà dell'aritmetica modulare:

$$[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$$

Pertanto si possono ridurre i risultati intermedi modulo n. Questo semplifica notevolmente il calcolo.

Una considerazione è l'efficienza dell'elevamento a potenza, poiché con RSA si ha a che fare con esponenti potenzialmente estesissimi. Per vedere come si può aumentare l'efficienza, si consideri che si desidera calcolare x^{16}. Un approccio immediato prevede l'impiego di 15 moltiplicazioni:

$$x^{16} = x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x$$

Tuttavia si può ottenere lo stesso risultato finale utilizzando solo quattro moltiplicazioni se si prende ripetutamente il quadrato di ciascun risultato parziale formando successivamente x^2, x^4, x^8, x^{16}. Come ulteriore esempio, si supponga di voler calcolare $x^{11} \bmod n$ per x e n interi. Si osservi che $x^{11} = x^{1+2+8} = (x)(x^2)(x^8)$. In questo caso si calcolano $x \bmod n$, $x^2 \bmod n$, $x^4 \bmod n$ e $x^8 \bmod n$ e si calcola infine $[(x \bmod n) \times (x^2 \bmod n) \times (x^8 \bmod n)] \bmod n$. Più in generale si supponga di voler trovare il valore a^b con a e b interi positivi. Se si esprime b come un numero binario $b_k, b_{k-1} \ldots b_0$, si avrà:

$$b = \sum_{b_i \neq 0} 2^i$$

Pertanto:

$$a^b = a^{\left(\sum_{b_i \neq 0} 2^i\right)} = \prod_{b_i \neq 0} a^{(2^i)}$$

$$a^b \bmod n = \left[\prod_{b_i \neq 0} a^{(2^i)}\right] \bmod n = \left(\prod_{b_i \neq 0} [a^{(2^i)} \bmod n]\right) \bmod n$$

Si può quindi sviluppare l'algoritmo[5] per il calcolo di $a^b \bmod n$, rappresentato nella Figura 9.7. La Tabella 9.3 mostra un esempio di esecuzione di questo algoritmo. Si noti che la variabile c non è necessaria: viene inclusa solo per scopi descrittivi. Il valore finale di c è il valore dell'esponente.

Impiego efficiente della chiave pubblica

Per velocizzare il funzionamento dell'algoritmo RSA con la chiave pubblica, solitamente si sceglie e in modo opportuno. La scelta più comune è 65 537 ($2^{16} - 1$), altre scelte possibili sono 3 e 17. Tutti questi valori hanno solo due bit a 1, consentendo quindi di minimizzare il numero di moltiplicazioni richieste nell'esponenziazione.

Tuttavia, con una chiave pubblica molto piccola, per esempio $e = 3$, RSA diviene vulnerabile a un semplice attacco. Si supponga di avere tre utenti RSA differenti che utilizzano tutti il valore $e = 3$ ma valori differenti di n: n_1, n_2 e n_3. Se l'utente A invia lo stesso messaggio M crittografato a tutti e tre gli utenti, i tre testi cifrati sono $C_1 = M^3 \bmod n_1$, $C_2 = M^3 \bmod n_2$ e $C_3 = M^3 \bmod n_3$. È probabile che n_1, n_2 e n_3 siano primi relativi a coppie: si potrebbe dunque utilizzare il teorema del resto cinese (CRT) per calcolare $M^3 \bmod (n_1 n_2 n_3)$. Per le regole dell'algoritmo RSA, M è inferiore a ciascuno degli n_i, dunque $M^3 < n_1 n_2 n_3$. L'attaccante deve quindi semplicemente calcolare la radice cubica di M^3. Questo attacco può essere evitato aggiungendo una stringa di bit pseudocasuali differente a ciascuna istanza del messaggio da crittografare. Questa strategia verrà discussa in seguito.

Il lettore potrebbe aver osservato che la definizione dell'algoritmo RSA (Figura 9.5) richiede che nella generazione della chiave l'utente selezioni un valore di e che sia primo relativo di $\phi(n)$. Per esempio se un utente ha pre-selezionato $e = 65\,537$ e ha poi generato i

[5] L'algoritmo ha una lunga storia; questa particolare formulazione è tratta da [CORM01].


```
c ← 0; f ← 1
for i ← k downto 0
    do  c ← 2 × c
        f ← (f × f) mod n
        if  bi = 1
            then  c ← c + 1
                  f ← (f × a) mod n
return f
```

Nota: L'intero b è espresso come numero binario $b_k b_{k-1} \ldots b_0$.

Figura 9.7 L'algoritmo di calcolo di $a^b \bmod n$.

Tabella 9.3 I risultati dell'algoritmo veloce di esponenziazione modulare per $a^b \bmod n$ dove $a = 7$, $b = 560 = 1000110000$, $n = 561$.

i	9	8	7	6	5	4	3	2	1	0
b_i	1	0	0	0	1	1	0	0	0	0
c	1	2	4	8	17	35	70	140	280	560
d	7	49	157	526	160	241	298	166	67	1

numeri primi p e q, potrebbe avvenire che $\gcd(\phi(n), e) \neq 1$. L'utente deve quindi rifiutare qualsiasi valore di p e q che non sia congruente a 1 (mod 65 537).

Impiego efficiente della chiave privata

In modo analogo non è possibile assegnare un piccolo valore a d con lo scopo di aumentare l'efficienza. Un piccolo valore di d è vulnerabile agli attacchi a forza bruta e a varie forme di analisi crittografica [WIEN90]. Vi è tuttavia un modo per velocizzare il calcolo utilizzando CRT. Si desidera calcolare il valore $M = C^d \bmod n$. Si definiscano i seguenti risultati intermedi:

$$V_p = C^d \bmod p; V_q = C^d \bmod q.$$

Secondo il CRT, l'Equazione 8.8 definisce i valori:

$$X_p = q \times (q^{-1} \bmod p); X_q = p \times (p^{-1} \bmod q).$$

Il CRT consente di derivare, tramite l'Equazione 8.9, che

$$M = (V_p X_p + V_q X_q) \bmod n.$$

È possibile semplificare ulteriormente il calcolo di V_p e V_q utilizzando il teorema di Fermat, il quale afferma che se a e p sono primi relativi, allora $a^{p-1} \equiv 1 \pmod p$. Con un semplice ragionamento ci si può convincere della validità delle equazioni seguenti:

$$V_p = C^d \bmod p = C^{d \bmod (p-1)} \bmod p; V_q = C^d \bmod q = C^{d \bmod (q-1)} \bmod q.$$

Le quantità $d \bmod (p-1)$ e $d \bmod (q-1)$ possono essere preventivamente calcolate. In questo modo il calcolo risulta circa quattro volte più veloce rispetto alla valutazione diretta di $M = C^d \bmod n$ [BONE02].

La generazione della chiave

Prima dell'applicazione del sistema a chiave pubblica, ciascun partecipante deve generare una coppia di chiavi. Ciò comporta le seguenti operazioni.

- Determinare due numeri primi p e q.
- Selezionare e o d e calcolare l'altro valore.

Innanzitutto si consideri la selezione di p e q. Poiché il valore di $n = pq$ sarà noto a chiunque, per impedire di scoprire p e q tramite metodi esaustivi, questi numeri primi devono essere scelti in un insieme sufficientemente esteso (p e q devono essere numeri molto estesi). Ma d'altra parte il metodo utilizzato per trovare numeri primi di grandi dimensioni deve essere ragionevolmente efficiente.

Al momento attuale non esistono tecniche utili che forniscano numeri primi arbitrariamente estesi ed è quindi necessario affrontare il problema in qualche altro modo. La procedura generalmente utilizzata consiste nello scegliere casualmente un numero dispari dell'ordine di grandezza desiderato e verificare se tale numero è primo. Se non lo è, si può scegliere un altro numero casuale fino a trovare un numero primo.

Sono stati sviluppati vari test che consentono di valutare se un numero è primo (vedere [KNUT98] per una descrizione di vari test). Quasi tutti i test sono probabilistici. In pratica il test può unicamente determinare che un determinato intero è *probabilmente* primo. Nonostante l'assenza di certezza, questi test possono essere eseguiti ripetutamente in modo da fare avvicinare il più possibile questa probabilità al valore 1,0. Uno degli algoritmi più efficienti e utilizzati, l'algoritmo Miller-Rabin, è descritto nel Capitolo 8. Con questo algoritmo e la maggior parte degli algoritmi di questo tipo, la procedura per verificare se un determinato intero n è primo consiste nello svolgere alcuni calcoli che coinvolgono n e un numero intero casuale a. Se n "fallisce" il test, allora non è primo. Se n "passa" il test, allora n può essere primo. Se n passa una grande quantità di test utilizzando valori di a scelti casualmente, allora si può avere un'elevata confidenza che sia veramente primo.

In breve, la procedura per la scelta di un numero primo è la seguente.

1. Scegliere casualmente un intero dispari n (utilizzando un generatore di numeri pseudocasuali).
2. Scegliere casualmente un intero $a < n$.
3. Eseguire un test di primalità, come per esempio il test di Miller-Rabin, con il parametro a. Se n fallisce il test, rifiutare il valore e tornare al passo 1.
4. Se n passa un numero sufficiente di test, accettare n, altrimenti tornare al passo 2.

Questa è una procedura per certi versi noiosa. Tuttavia, si deve ricordare che questa operazione non deve essere svolta molto frequentemente, solo quando è necessaria una nuova coppia (*PU*, *PR*).

Vale la pena notare quanti numeri verranno statisticamente rifiutati prima di trovare un numero primo. Un risultato tratto dalla teoria dei numeri, chiamato teorema del numero primo, stabilisce che i numeri primi attorno a N sono spaziati in media di circa $(\ln N)$ interi.

Pertanto, in media, sarà necessario eseguire il test di $\ln(N)$ interi prima di trovare un numero primo. In realtà, poiché tutti gli interi pari non sono certamente primi, il valore corretto è $\ln(N)/2$. Per esempio, se si cercasse un numero primo dell'ordine di grandezza di 2^{200}, per trovare un numero primo sarebbero necessari in media $\ln(2^{200})/2 = 70$ tentativi.

Dopo aver determinato i numeri primi p e q, il processo di generazione della chiave termina selezionando il valore di e e calcolando d o, alternativamente, selezionando il valore di d e calcolando e. Supponendo di impiegare la prima alternativa, occorre selezionare un valore e tale che $\gcd(\phi(n), e) = 1$ e poi calcolare $d \equiv e^{-1} \pmod{\phi(n)}$. Fortunatamente esiste un algoritmo in grado di calcolare il massimo comun divisore di due interi e, contemporaneamente, se gcd è uguale a 1, determinare l'inverso di uno degli interi modulo l'altro; si tratta dell'algoritmo di Euclide esteso, descritto nel Capitolo 8. Pertanto la procedura consiste nel generare una serie di numeri casuali, confrontarli con $\phi(n)$ fino a trovare un numero primo relativo di $\phi(n)$. Nuovamente ci si potrebbe chiedere: quanti numeri casuali occorre verificare per trovare un numero utilizzabile, ovvero un numero primo relativo di $\phi(n)$? Si può dimostrare con facilità che la probabilità che due numeri casuali siano primi relativi è circa 0,6; pertanto bastano pochi test per trovare un intero adatto (vedere il Problema 8.2).

La sicurezza di RSA

Tre possibili approcci per attaccare l'algoritmo RSA sono:

- **Forza bruta:** comporta l'applicazione di tutte le chiavi private possibili.
- **Attacchi matematici:** vi sono vari approcci, tutti sostanzialmente equivalenti, in termini di complessità, alla fattorializzazione del prodotto di due numeri primi.
- **Attacchi a tempo:** si basano sul tempo di esecuzione dell'algoritmo di decrittografia.
- **Attacchi a testo cifrato scelto:** sfruttano le caratteristiche dell'algoritmo RSA.

Le difese contro l'approccio a forza bruta non differiscono da quelle di altri sistemi crittografici, ovvero si deve utilizzare una chiave scelta in un grande spazio delle chiavi. Maggiore è il numero di bit contenuti in d e più resistente sarà l'algoritmo. Tuttavia, dati i calcoli necessari per la generazione della chiave e per la crittografia/decrittografia, maggiori sono le dimensioni della chiave, più lento sarà il sistema.

In questa parte del capitolo verrà fornita una panoramica sugli attacchi di tipo matematico e a tempo.

Il problema della fattorializzazione

Si possono identificare tre approcci di attacco matematico a RSA.

- Fattorializzare n nei suoi due fattori primi. Questo consente di calcolare $\phi(n) = (p-1) \times (q-1)$, che, a sua volta, consente di determinare $d \equiv e^{-1} \pmod{\phi(n)}$.
- Determinare direttamente $\phi(n)$, senza prima determinare p e q. Anche questo consente di determinare $d \equiv e^{-1} \pmod{\phi(n)}$.
- Determinare direttamente d senza prima determinare $\phi(n)$.

La maggior parte delle analisi crittografiche di RSA si concentra sulla fattorializzazione di n nei suoi due fattori primi. Determinare $\phi(n)$ dato n è equivalente a fattorializzare n [RIBE96]. Con gli algoritmi attualmente noti, determinare d sulla base di e e n sembra essere altrettanto dispendioso in termini di tempo quanto il problema della fattorializzazione [KALI95]. Pertanto si può utilizzare il tempo di fattorializzazione d come base di confronto per valutare la sicurezza di RSA.

Per un valore esteso di n, con grandi fattori primi, la fattorializzazione è un problema difficile ma non tanto come lo era un tempo. Ecco un esempio chiarificatore. Nel 1977, i tre inventori di RSA sfidarono i lettori di *Scientific American* a decodificare una cifratura che avevano pubblicato nelle colonne di "Mathematical Games" di Martin Gardner [GARD77] offrendo una ricompensa di 100 dollari a chi avrebbe fornito la sequenza di testo in chiaro, un evento che secondo le loro previsioni si sarebbe verificato solo dopo circa 40 quadrilioni di anni. Nell'aprile del 1994, un gruppo attivo in Internet reclamò il premio dopo soli otto mesi di lavoro [LEUT94]. Questa sfida utilizzava una chiave pubblica della lunghezza (n) di 129 cifre decimali (circa 428 bit). Nel frattempo, proprio come avevano fatto per DES, i Laboratories RSA emisero delle sfide per RSA con chiavi di 100, 110, 120 cifre e così via. L'ultima sfida da vincere è quella con una chiave della lunghezza di 200 cifre decimali, equivalenti a circa 663 bit. La Tabella 9.4 mostra i risultati dove lo sforzo è misurato in MIPS-anno: un processore da un milione di istruzioni per secondo in esecuzione per un intero anno, ovvero l'esecuzione di circa 3×10^{13} istruzioni. Un PC Pentium da 1 GHz è una macchina da circa 250 MIPS.

Tabella 9.4 I progressi nella fattorializzazione.

Numero di cifre decimali	Numero approssimativo di bit	Data violazione	MIPS-anno	Algoritmo
100	332	Aprile 1991	7	Quadratic sieve
110	365	Aprile 1992	75	Quadratic sieve
120	398	Giugno 1993	830	Quadratic sieve
129	428	Aprile 1994	5000	Quadratic sieve
130	431	Aprile 1996	1000	Generalized number field sieve
140	465	Febbraio 1999	2000	Generalized number field sieve
155	512	Agosto 1999	8000	Generalized number field sieve
160	530	Aprile 2003	-	Lattice sieve
174	576	Dicembre 2003	-	Lattice sieve
200	663	Maggio 2003	-	Lattice sieve

Un fatto interessante evidenziato dalla Tabella 9.4 riguarda il metodo utilizzato. Fino alla metà degli anni '90, gli attacchi a fattorializzazione sono stati eseguiti utilizzando un approccio a setaccio quadratico. L'attacco contro RSA-130 ha utilizzato un algoritmo più recente, chiamato GNFS (Generalized Number Field Sieve) ed è stato in grado di fattorializzare un numero più esteso rispetto a RSA-129 con un impegno in termini di calcolo pari a solo il 20%.

Le chiavi di grandi dimensioni sono doppiamente minacciate: da un lato la potenza di calcolo aumenta continuamente e dall'altro gli algoritmi di fattorializzazione vengono continuamente raffinati. Si è visto che l'impiego di un nuovo algoritmo ha prodotto una drastica accelerazione. Si possono prevedere ulteriori raffinamenti a GNFS e l'individuazione di algoritmi ancora migliori. Infatti un algoritmo correlato, chiamato SNFS (Special Number Field Sieve) può fattorializzare particolari numeri con una velocità molto superiore rispetto a GNFS. La Figura 9.8 confronta le prestazioni dei due algoritmi. È ragionevole attendersi dei cambiamenti repentini che consentiranno di ottenere prestazioni di fattorializzazione più meno nello stesso tempo di SNFS o con risultati ancora migliori [ODLY95]. Pertanto occorre scegliere con attenzione le dimensioni delle chiavi per RSA. Nel prossimo futuro sembra ragionevole l'impiego di chiavi da 1024 a 2048 bit.

Oltre a specificare le dimensioni di n, i ricercatori hanno suggerito altri vincoli. Per evitare i valori di n che possono essere fattorializzati con facilità, gli inventori dell'algoritmo suggeriscono di imporre su p e q i seguenti vincoli.

1. La lunghezza di p e q deve differire per poche cifre. Pertanto, per una chiave di 1024 bit (309 cifre decimali) sia p che q devono essere dell'ordine di grandezza che va da 10^{75} a 10^{100}.
2. Sia $(p-1)$ che $(q-1)$ devono contenere un grande fattore primo.
3. $\gcd(p-1, q-1)$ deve essere piccolo.

Inoltre è stato dimostrato che se $e < n$ e $d < n^{1/4}$, d può essere determinato con facilità [WIEN90].

Attacchi a tempo

Per capire quanto sia difficile valutare la sicurezza di un algoritmo crittografico, si può far riferimento agli attacchi di carattere temporale. Paul Kocher, un consulente crittografico, ha dimostrato che si può determinare una chiave privata analizzando il tempo impiegato dai computer per decifrare i messaggi [KOCH96, KALI96b]. Gli attacchi a valutazione temporale sono applicabili non solo a RSA ma anche ad altri sistemi di crittografia a chiave pubblica. Questo attacco preoccupa per due motivi: proviene da una direzione completamente inattesa e si basa unicamente sul testo cifrato.

Un attacco a tempo è per certi versi analogo a un ladro che cerca di indovinare la combinazione di sicurezza di una cassaforte osservando il tempo impiegato per ruotare la manopola. Si può spiegare l'attacco utilizzando l'algoritmo di esponenziazione modulare della Figura 9.7 ma l'attacco può essere adattato in modo da lavorare con qualsiasi implementazione che non operi in un tempo prefissato. In questo algoritmo l'esponenziazione modulare viene eseguita bit a bit, e viene eseguita una moltiplicazione modulare a ciascuna iterazione più un'ulteriore moltiplicazione modulare per ciascun bit a 1.

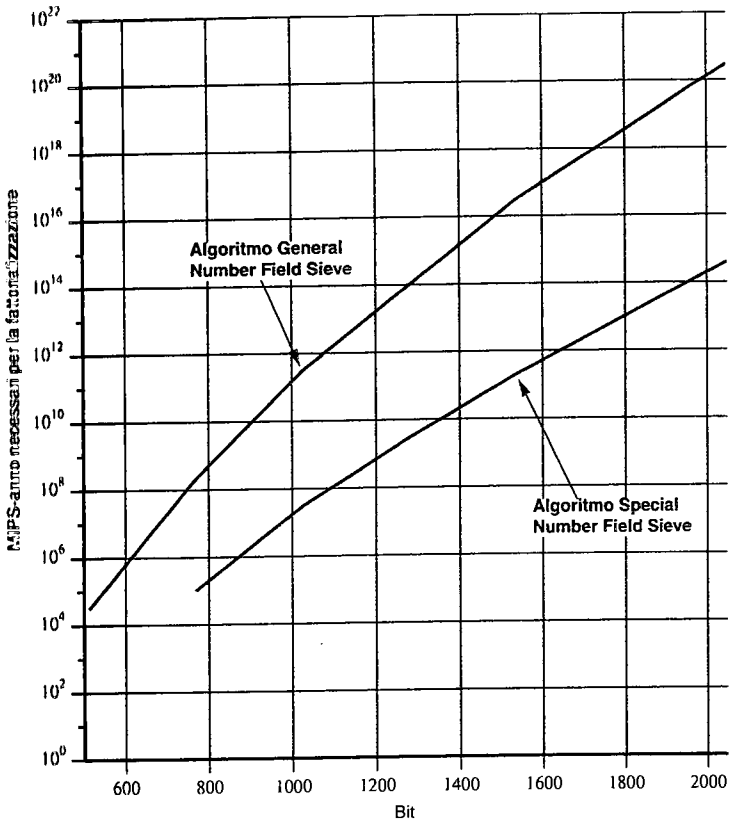


Figura 9.8 Grafico dei MIPS-anno necessari per la fattorializzazione.

Come evidenzia Kocher, è più facile comprendere l'attacco in un caso estremo. Si supponga che il sistema di destinazione utilizzi una funzione di moltiplicazione modulare molto veloce in quasi tutti i casi ma che in alcuni casi impieghi molto più tempo rispetto a un'intera esponenziazione modulare media. L'attacco procede bit dopo bit partendo dal bit più a sinistra, b_k. Si supponga che i primi j bit siano noti (si parte con $j = 0$ e si ripete l'attacco fino a conoscere l'intero esponente). Per un determinato testo cifrato, si possono completare le prime j iterazioni del ciclo **for**. L'operazione del passo successivo dipende dal primo bit sconosciuto dell'esponente. Se il bit è uguale a 1, verrà eseguito $d \leftarrow (d \times a) \bmod n$. Per alcuni valori di a e d, la moltiplicazione modulare sarà estremamente lenta e sarà possibile rilevare questa differenza. Pertanto, se il tempo impiegato per eseguire l'algoritmo di decifratura è sempre lento quando questa particolare iterazione è lenta con un bit uguale a 1, si può ragionevolmente supporre che questo bit sia 1. Se invece il tempo

di esecuzione dell'intero algoritmo è molto ridotto per un certo numero di osservazioni si può ragionevolmente supporre che questo bit sia 0.

In realtà le implementazioni a esponenziazione modulare non presentano variazioni temporali così estreme, tali che il tempo di esecuzione di una singola iterazione superi il tempo di esecuzione medio dell'intero algoritmo. Ciononostante esistono variabilità tali da rendere possibile questo tipo di attacco. Per tutti i dettagli consultare [KOCH96].

Sebbene l'attacco a tempo rappresenti una grave minaccia, contro tale attacco è possibile applicare alcune semplici contromisure fra cui le seguenti.

- **Tempo di esponenziazione costante:** assicurarsi che tutte le esponenziazioni richiedano lo stesso tempo prima di restituire un risultato. Si tratta di una correzione molto semplice che però introduce un degrado prestazionale.
- **Ritardo casuale:** si possono ottenere migliori prestazioni aggiungendo all'algoritmo di esponenziazione un ritardo casuale in modo da confondere l'attacco a tempo. Ma Kocher sostiene che se il rumore introdotto non è sufficiente, si può comunque riuscire a violare il codice raccogliendo un numero di misure sufficienti a compensare i ritardi casuali.
- **Tecnica Blinding:** si può moltiplicare il testo cifrato per un numero casuale prima di svolgere l'esponenziazione. Questa operazione impedisce di conoscere quali bit di testo cifrato vengono elaborati e pertanto impedisce l'analisi dei bit fondamentale per questo tipo di attacco.

RSA Data Security incorpora la tecnica Blinding in alcuni dei suoi prodotti. L'operazione a chiave privata $M = C^d \bmod n$ è implementata nel seguente modo.

1. Generare un numero casuale segreto r compreso fra 0 e $n - 1$.
2. Calcolare $C' = C(r^e)$ dove e è l'esponente pubblico.
3. Calcolare $M' = (C')^d \bmod n$ con la normale implementazione RSA.
4. Calcolare $M = M'r^{-1} \bmod n$. In questa equazione, r^{-1} è l'inverso moltiplicativo di r modulo n; per una discussione di questo concetto consultare il Capitolo 8. Si può dimostrare che questo è il risultato corretto osservando che $r^{ed} \bmod n = r \bmod n$.

RSA Data Security sostiene che la tecnica di Blinding introduce un degrado prestazionale valutabile fra il 2 e il 10%.

Attacco a testo cifrato scelto e padding ottimale per crittografia simmetrica

L'algoritmo RSA di base è vulnerabile agli attacchi a testo cifrato scelto (CCA). CCA è definito come un attacco nel quale l'avversario ha la possibilità di scegliere un certo numero di testi cifrati e ottenere i testi in chiaro corrispondenti, decrittografati con la chiave privata del suo obiettivo. In questo caso l'avversario potrebbe scegliere un testo in chiaro, crittografarlo con la chiave pubblica del suo obiettivo ed essere in grado di ottenerne il testo in chiaro corrispondente decrittografato con la chiave privata. Ovviamente questo non fornirebbe all'avversario alcuna informazione utile. Invece, l'avversario sfrutta le caratteristiche di RSA e seleziona dei blocchi di dati che, quando elaborati utilizzando la chiave privata dell'obiettivo, forniscono informazioni utili per l'analisi crittografica.

Un semplice esempio di attacco CCA contro RSA sfrutta la seguente caratteristica di RSA:

$$E(PU,M_1)\times E(PU,M_2)=E(PU,[M_1\times M_2]). \qquad (9.2)$$

È possibile decrittografare $C = M^e \bmod n$ utilizzando un attacco CCA nel modo seguente:

1. Calcolare $X = (C \times 2^e) \bmod n$.
2. Utilizzare X come testo cifrato scelto e ottenere $Y = X^d \bmod n$.

Ora si osservi:

$$X = (C \bmod n)\times(2^e \bmod n) = (M^e \bmod n)\times(2^e \bmod n) = (2M)^e \bmod n.$$

Si ottiene quindi $Y = (2M) \bmod n$, da cui è possibile dedurre M. Per neutralizzare questo tipo di attacco, i sistemi di crittografia reali basati su RSA aggiungono casualmente dei bit al testo in chiaro prima di crittografarlo. Questo modifica casualmente il testo cifrato in modo tale che l'Equazione 9.2 non sia più valida. Nonostante questo è possibile utilizzare attacchi CCA più complessi: si è così dimostrato che il semplice riempimento con un valore casuale non è sufficiente per garantire la sicurezza desiderata. Per neutralizzare questi attacchi, RSA Security Inc, un fornitore RSA leader nel settore e detentore in precedenza del brevetto RSA, raccomanda di modificare il testo in chiaro utilizzando una procedura nota come padding (riempimento) ottimale per crittografia asimmetrica (OAEP, Optimal Asymmetric Encryption Padding). La discussione dettagliata delle possibili minacce e di OAEP va oltre gli scopi di questo libro: consultare [POIN02] per un'introduzione e [BELL94a] per un'analisi rigorosa. In questo volume si riassume semplicemente la procedura OAEP.

La Figura 9.9 illustra la crittografia OEAP. Come primo passo vengono aggiunti i dati di riempimento al messaggio M da crittografare. Un insieme di parametri opzionali P viene trasformato da una funzione hash H[6]. Al risultato vengono aggiunti degli 0 di riempimento fino a ottenere la lunghezza totale del blocco dati (DB) desiderata. Successivamente viene generato un seme casuale che viene trasformato da un'altra funzione hash, chiamata funzione di generazione della maschera (MGF). Si calcola quindi lo XOR bit-a-bit fra il codice hash risultante e DB per produrre il valore maskedDB. Il valore risultante viene a sua volta trasformato con la MGF per ottenere un codice da utilizzare in una nuova operazione di XOR con il seme per ottenere il seme mascherato. La concatenazione del seme mascherato e del valore maskedDB costituisce il messaggio codificato EM. Si noti che EM include il messaggio e il riempimento mascherati con il seme, e il seme mascherato con maskedDB. EM viene infine crittografato con RSA.

9.3 Letture e siti Web consigliati

Le letture consigliate presentate nel Capitolo 3 riguardano la crittografia sia a chiave pubblica che simmetrica.

[6] Una funzione hash mappa un blocco di dati o un messaggio di lunghezza variabile in un valore di lunghezza fissa chiamato codice hash. Le funzioni hash sono trattate approfonditamente nei Capitoli 11 e 12.

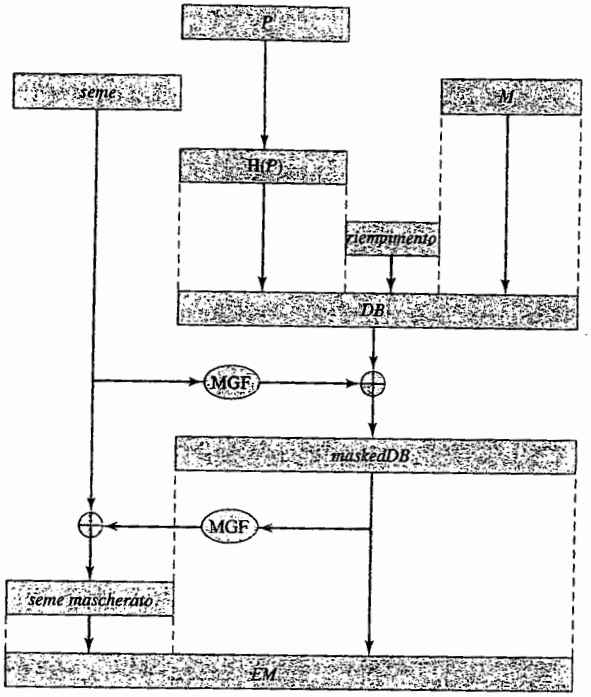


Figura 9.9 Crittografia utilizzando il riempimento ottimale per crittografia simmetrica (OAEP)..

[DIFF88] descrive in dettaglio i vari tentativi di sviluppare gli algoritmi crittografici a due chiavi e la graduale evoluzione di vari protocolli. [CORM01] fornisce un riepilogo conciso ma completo e leggibile di tutti gli algoritmi impiegati nella verifica, nel calcolo e nell'analisi crittografica di RSA. [BONE99] discute vari attacchi ad analisi crittografica contro RSA. [SHAM03] è una descrizione più recente.

BONE99 D. Boneh. "Twenty Years of Attacks on the RSA Cryptosystem". *Notices of the American Mathematical Society*, Febbraio 1999.

CORM01 T. Cormen, C. Leiserson, R. Rivest e C. Stein. *Introduction to Algorithms*. Cambridge, MA: MIT Press, 2001.

DIFF88 W. Diffie. "The First Ten Years of Public-Key Cryptography". *Proceedings of the IEEE*, Maggio 1988. Ristampato in [SIMM92].

SHAM03 A. Shamir e E. Tromer. "On the Cost of Factoring RSA-1024". *CryptoBytes*, Estate 2003. **http://www.rsasecurity.com/rsalabs**.

Siti Web consigliati

- **RSA Laboratories:** un'estesa raccolta di materiale tecnico su RSA e su altri argomenti della crittografia.

9.4 Termini chiave, domande di ripasso e problemi

Termini chiave

Attacco a tempo
Attacco a testo cifrato scelto (CCA)
Chiave privata
Chiave pubblica
Complessità temporale
Crittografia a chiave pubblica
Firma digitale
Funzione monodirezionale
Funzione trappola monodirezionale
Riempimento ottimale per crittografia simmetrica (OAEP)
RSA
Scambio di chiavi
Sistemi crittografici a chiave pubblica

Domande di ripasso

9.1 Quali sono i principali elementi di un sistema crittografico a chiave pubblica?
9.2 Quali sono i ruoli della chiave pubblica e della chiave privata?
9.3 Quali sono le tre categorie generali di applicazione per i sistemi crittografici a chiave pubblica?
9.4 A quali requisiti deve rispondere un sistema crittografico a chiave pubblica per essere considerato un algoritmo sicuro?
9.5 Che cos'è una funzione monodirezionale?
9.6 Che cos'è una funzione trappola monodirezionale?
9.7 Descrivere in termini generali una procedura efficiente per individuare un numero primo.

Problemi

9.1 Prima di scoprire gli schemi a chiave pubblica come RSA, venne sviluppata una prova della loro esistenza, il cui scopo era quello di dimostrare che la crittografia a chiave pubblica era in teoria possibile. Si considerino le funzioni $f_1(x_1) = z_1$; $f_2(x_2, y_2) = z_2$; $f_3(x_3, y_3) = z_3$, dove tutti i valori sono interi, con $1 \le x_i, y_i, z_i \le N$. La funzione f_1 può essere rappresentata da un vettore M1 di lunghezza N in cui la k-esima voce è il valore di $f_1(k)$. Analogamente $f_2(k)$ e $f_3(k)$ possono essere rappresentate con le matrici M2 e M3 di ordine $N \times N$. Lo scopo è quello di rappresentare il processo di crittografia/decrittografia tramite ricerche su tabelle con valori molto estesi di N. Tali tabelle sarebbero enormemente estese ma, in linea di principio, potrebbero essere costruite. Lo schema funziona nel seguente modo: si deve costruire M1 con una permutazione casuale di tutti gli interi compresi fra 1 e N; ovvero ciascun intero in M1 deve comparire esattamente una volta. Costruire M2 in modo che ciascuna riga contenga una permutazione casuale dei primi N interi. Infine costruire M3 in modo che soddisfi la seguente condizione:

$$f_3(f_2(f_1(k), p), k) = p \qquad \text{per tutti i } k, p \text{ con } 1 \le k, p \le N$$

In altri termini.

1. M1 prende come input k e produce come output x.
2. M2 prende come input x e p e fornisce z.
3. M3 prende come input z e k e produce p.

Le tre tabelle, una volta costruite, vengono rese pubbliche.

A. Dovrebbe essere chiaro che è possibile costruire M3 per soddisfare la condizione precedente. Come esempio, compilare M3 per il semplice caso seguente:

M1 =

5
4
2
3
1

M2 =

5	2	3	4	1
4	2	5	1	3
1	3	2	4	5
3	1	4	2	5
2	5	3	4	1

M3 =

Convenzione: l'i-esimo elemento di M1 corrisponde a $k = i$. La i-esima riga di M2 corrisponde a $x = i$; la j-esima colonna di M2 corrisponde a $p = j$. La i-esima riga di M3 corrisponde a $z = i$; la j-esima colonna di M3 corrisponde a $k = j$.

B. Descrivere l'uso di questo insieme di tabelle per eseguire la crittografia e la decrittografia fra due utenti.
C. Dimostrare che questo è uno schema sicuro.

9.2 Svolgere la crittografia e la decrittografia utilizzando l'algoritmo RSA, come illustrato nella Figura 9.6, per i seguenti valori.
A. $p = 3$; $q = 11$, $e = 7$; $M = 5$
B. $p = 5$; $q = 11$, $e = 3$; $M = 9$
C. $p = 7$; $q = 11$, $e = 17$; $M = 8$
D. $p = 11$; $q = 13$, $e = 11$; $M = 7$
E. $p = 17$; $q = 31$, $e = 7$; $M = 2$. *Suggerimento:* la decrittografia non è difficile come può sembrare, basta usare qualche stratagemma.

9.3 In un sistema a chiave pubblica che utilizza RSA, si intercetta il testo cifrato $C = 10$ inviato a un utente la cui chiave pubblica è $e = 5$, $n = 35$. Qual è il testo in chiaro M?

9.4 In un sistema RSA, la chiave pubblica di un determinato utente è $e = 31$, $n = 3599$. Qual è la chiave privata di questo utente?

Suggerimento: determinare prima di tutto p e q per tentativi, quindi utilizzare l'algoritmo di Euclide esteso per trovare l'inverso moltiplicativo di 31 modulo $\phi(n)$.

9.5 Nell'utilizzo dell'algoritmo RSA, se un piccolo numero di codifiche ripetute fornisce il testo in chiaro, qual è la causa probabile?

9.6 Si supponga di avere un insieme di blocchi codificati con l'algoritmo RSA e di non avere la chiave privata. Si supponga che la chiave pubblica sia $n = pq$, e. Si supponga anche che qualcuno dica di conoscere che uno dei blocchi di testo in chiaro ha un fattore comune con n. Questo può aiutare in qualche modo?

9.7 Nello schema di crittografia a chiave pubblica RSA, ciascun utente ha una chiave pubblica, e, e una chiave privata, d. Si supponga che Bob perda la propria chiave privata. Invece di generare un nuovo modulo, Bob decide di generare una nuova coppia di chiavi pubblica e privata. È sicura questa soluzione?

9.8 Supporre che Bob utilizzi il sistema di crittografia RSA con un modulo n molto esteso per il quale la fattorizzazione non possa essere effettuata entro limiti di tempo ragionevoli. Supporre che Alice invii un messaggio a Bob rappresentando ogni carattere alfabetico con un intero compreso fra 0 e 25 (A $\rightarrow$ 0, ..., Z$\rightarrow$25) e crittografando poi ciascun numero separatamente utilizzando RSA con entrambi n ed e estesi. Si tratta di un metodo sicuro? In caso negativo, descrivere l'attacco più efficiente contro questo metodo di crittografia.

9.9 Utilizzando un foglio elettronico (per esempio Excel) o una calcolatrice, calcolare le seguenti operazioni. Documentare i risultati di tutte le moltiplicazioni modulari intermedie. Determinare il numero di moltiplicazioni modulari per ciascuna trasformazione principale (crittografia, decrittografia, test di primalità ecc.).

1. Verificare tutti i numeri dispari nell'intervallo da 233 a 241 con il test di primalità Miller-Rabin con base 2.
2. Crittografare il messaggio $M = 2$ utilizzando RSA con i seguenti parametri: $e = 23$, $n = 233 \times 241$.
3. Calcolare una chiave privata (d, p, q) corrispondente alla chiave pubblica precedente (e, n).
4. Decrittografare il testo cifrato ottenuto con i due differenti metodi seguenti:
   a. senza utilizzare il teorema cinese del resto,
   b. utilizzando il teorema cinese del resto.

9.10 Supporre di generare un messaggio crittografato e autenticato applicando prima la trasformazione RSA determinata dalla propria chiave privata, e crittografando poi il messaggio utilizzando la chiave pubblica del destinatario (si noti che non si utilizza la funzione hash prima della prima trasformazione). Questo schema funzionerà correttamente [ovvero consentirà al destinatario di ricostruire il messaggio originale per tutte le possibili relazioni fra i moduli n_s del mittente e i moduli n_r del destinatario ($n_s > n_r$, $n_s < n_r$, $n_s = n_r$)]? Giustificare la risposta. In caso di risposta negativa, spiegare come correggere lo schema.

9.11 "Devo dire, Holmes", la voce del dottor Watson era entusiasta, "che le tue recenti attività nel campo della sicurezza delle reti hanno aumentato il mio interesse nella crittografia. Proprio ieri ho trovato un modo per rendere pratica la crittografia One-Time Pad".

"Davvero?" Il volto di Holmes perse il suo aspetto sonnolento.

"Sì, Holmes. L'idea è veramente semplice. Per una determinata funzione monodirezionale F, genero una lunga sequenza pseudocasuale di elementi applicando F a una sequenza standard di argomenti. L'analista crittografico può conoscere F e la natura generale della sequenza che può essere semplicemente S, S + 1, S + 2, ... ma non il valore segreto S. Data la natura monodirezionale di F nessuno potrà estrarre S dato F(S + i) per qualche i; pertanto, anche se qualcuno dovesse ottenere un determinato segmento della sequenza, non sarà in grado di ricostruire la parte rimanente".

"Mi spiace, Watson, ma la tua proposta ha dei difetti e F deve soddisfare alcune condizioni aggiuntive. Consideriamo, per esempio, la funzione di crittografia RSA, ovvero $F(M) = M^K \bmod N$, con K segreta. Questa funzione è ritenuta monodirezionale ma non consiglierei il suo uso sulla sequenza $M = 2,3,4,5,6, \ldots$".

"Ma perché, Holmes?" Il dottor Watson sembrava non capire. "Perché pensi che la sequenza risultante $2^K \bmod N$, $3^K \bmod N$, $4^K \bmod N$, ... non sia appropriata per una crittografia One-Time Pad se K rimane segreta?".

"Perché, caro Watson, anche se K è segreta è, almeno parzialmente, prevedibile. Hai detto che l'analista crittografico può conoscere F e la natura generale della sequenza. Ora supponiamo che riesca a ottenere un breve segmento della sequenza di output. Nei circoli crittografici, questa ipotesi è generalmente considerata plausibile. E per questa sequenza di output, la conoscenza dei soli primi due elementi consentirà di prevedere molto sui successivi elementi della sequenza, se non tutti, dunque questa sequenza non può essere considerata crittograficamente robusta. Con la conoscenza di un segmento ancora più lungo si potrebbero prevedere ancora più elementi della sequenza. Conoscendo la natura generale della sequenza e i suoi primi due elementi $2^K \bmod N$ e $3^K \bmod N$ è possibile calcolare con facilità tutti gli elementi successivi ". Illustrare questa tecnica.

9.12 Dimostrare come l'algoritmo RSA possa essere rappresentato dalle matrici M1, M2 e M3 del Problema 9.1.

9.13 Considerare il seguente schema.

1. Selezionare un numero dispari, E.
2. Selezionare due numeri primi, P e Q, dove $(P - 1)(Q - 1) - 1$ è divisibile precisamente per E.
3. Moltiplicare P e Q per ottenere N.
4. Calcolare $D = \dfrac{(P-1)(Q-1)(E-1)+1}{E}$

Questo schema è equivalente a RSA? Dimostrare perché sì o perché no.

9.14 Considerare il seguente schema in cui B esegue la crittografia di un messaggio per A.

1. A sceglie due numeri primi di grandi dimensioni P e Q che sono primi relativi di $(P - 1)$ e $(Q - 1)$.
2. A pubblica la propria chiave pubblica $N = PQ$.
3. A calcola P' e Q' tali che $PP' \equiv 1 \pmod{Q - 1}$ e $QQ' \equiv 1 \pmod{P - 1}$.
4. B esegue la crittografia del messaggio M tale che $C = M^N \bmod N$.
5. A trova M risolvendo $M \equiv C^{P'} \pmod{Q}$ e $M \equiv C^{Q'} \pmod{P}$.

A. Spiegare come funziona questo schema.

B. Perché è differente da RSA?

C. Vi è un particolare vantaggio di RSA rispetto a questo schema?
D. Mostrare come si possa rapprensentare questo schema con le matrici M1, M2 e M3 del Problema 9.1

9.15 "Questo è un caso molto interessante, Watson", disse Holmes: "Un ragazzo ama una ragazza ed è anche corrisposto. Ma il padre di lei è un tipo strano che insiste sul fatto che il suo genero debba progettare un protocollo semplice e sicuro per un sistema crittografico a chiave pubblica utilizzabile nella rete di computer della sua azienda. Il ragazzo produce il seguente protocollo per le comunicazioni fra due parti, per esempio l'utente A che voglia inviare il messaggio M all'utente B [i messaggi scambiati sono nel formato (nome mittente, testo, nome destinatario)]".

1. A invia a B il blocco $(A,E(PU_b,[M,A]),B)$.
2. B esegue l'acknowledgement della ricezione inviando ad A il blocco $(B,E(PU_a,[M,B]), A)$. Come si può vedere il protocollo è veramente molto semplice. Ma il padre della ragazza sostiene che il ragazzo non ha soddisfatto la richiesta di un protocollo semplice poiché la proposta contiene una certa ridondanza e può essere semplificata nel modo seguente".

1. A invia a B il blocco $(A,E(PU_b,M),B)$.
2. B esegue l'acknowledgement inviando ad A il blocco $(B,E(PU_a,M), A)$.

"Per questo motivo, il padre della ragazza si rifiuta di dare in sposa la figlia al ragazzo rendendo infelici entrambi. Il ragazzo è venuto a chiedermi aiuto".

"Hmm, non vedo come tu possa aiutarlo"; Watson era visibilmente infelice all'idea che questo ragazzo dovesse perdere la propria amata.

"Invece penso di poterlo aiutare. Come sai, Watson, la ridondanza talvolta è utile per garantire la sicurezza del protocollo. Pertanto la semplificazione proposta dal padre della ragazza potrebbe rendere il nuovo protocollo vulnerabile a un attacco cui invece poteva resistere il protocollo originario", sostiene Holmes. "Si, è così, Watson: guarda, basta che l'estraneo sia uno degli utenti della rete e sia pertanto in grado di intercettare i messaggi scambiati da A e B. Essendo un utente della rete, avrà una propria chiave di crittografia pubblica e sarà in grado di inviare i propri messaggi ad A o a B e di ricevere la risposta. Con l'aiuto del protocollo semplificato, potrebbe quindi ottenere il messaggio M che l'utente A ha precedentemente inviato a B utilizzando la seguente procedura". Completare la descrizione.

9.16 Utilizzare l'algoritmo di esponenziazione veloce della Figura 9.7 per determinare 5^{596} mod1234. Illustrare il procedimento.

9.17 Ecco un'altra implementazione dell'algoritmo di esponenziazione veloce. Dimostrare che è equivalente a quella descritto nella Figura 9.7.

```
1. f ← 1; T ← a; E ← b
2. if odd(e) then f ← f X T
3. E ← ⌊ E/2 ⌋
4. T ← T X T
5. if E > 0 then goto 2
6. output f
```


9.18 Questo problema illustra una semplice applicazione di attacco a testo cifrato scelto. Bob intercetta un testo cifrato C destinato ad Alice e crittografato con la chiave pubblica di Alice e. Bob vuole ottenere il messaggio originale $M = C^d \bmod n$. Sceglie un valore casuale r minore di n e calcola: $Z = r^e \bmod n$, $X = ZC \bmod n$, $t = r^{-1} \bmod n$. Successivamente Bob riesce a far autenticare (firmare) X da Alice con la sua chiave privata (come nella Figura 9.3), decrittografando quindi X. Alice restituisce $Y = X^d \bmod n$. Illustrare come Bob possa utilizzare le informazioni disponibili per determinare M.

9.19 Illustrare l'operazione di decodifica OAEP utilizzata nella decrittografia, che corrisponde alla procedura di codifica della Figura 9.9.

Nota: i problemi successivi fanno riferimento all'algoritmo knapsack a chiave pubblica descritto nell'Appendice F.

9.20 Migliorare l'algoritmo P1 contenuto nell'Appendice 9b.

A. Sviluppare un algoritmo che richieda $2n$ moltiplicazioni e $n + 1$ somme. *Suggerimento:* $x^{i+1} = x^i \times x$.
B. Sviluppare un algoritmo che richieda solo $n + 1$ moltiplicazioni e $n + 1$ somme. *Suggerimento:* $P(x) = a_0 + x \times q(x)$, dove $q(x)$ è un polinomio di grado $n - 1$.

9.21 Quali elementi vi sono nel knapsack della Figura F.1?

9.22 Effettuare la crittografia e la decrittografia utilizzando l'algoritmo knapsack nei casi seguenti:

1. $\mathbf{a}' = (1, 3, 5, 10)$; $w = 7$; $m = 20$; $\mathbf{x} = 1101$.
2. $\mathbf{a}' = (1, 3, 5, 11, 23, 46, 136, 263)$; $w = 203$; $m = 491$; $\mathbf{x} = 11101000$.
3. $\mathbf{a}' = (2, 3, 6, 12, 25)$; $w = 46$; $m = 53$; $\mathbf{x} = 11101$.
4. $\mathbf{a}' = (15, 92, 108, 279, 563, 1172, 2243, 4468)$; $w = 2393$; $m = 9291$; $\mathbf{x} = 10110001$.

9.23 Perché si impone il vincolo $m > \sum_{i=1}^{n} d_i$?

Appendice 9.A - Dimostrazione dell'algoritmo RSA

Gli elementi di base dell'algoritmo RSA possono essere riassunti nel modo seguente. Dati due numeri primi p e q, con $n = pq$ e un blocco di messaggio $M < n$, vengono scelti due interi e e d tali che

$$M^{ed} \bmod n = M.$$

Si è affermato nel Paragrafo 9.2. che la relazione precedente è valida se e e d sono inversi moltiplicativi modulo $\phi(n)$ dove $\phi(n)$ è la funzione toziente di Eulero. Si è visto nel Capitolo 8 che per p e q primi, $\phi(pq) = (p - 1)(q - 1)$. La relazione fra e e d può essere espressa come:

$$ed \bmod \phi(n) = 1.$$

Un modo equivalente per esprimere questo consiste nel dire che vi è un intero k tale che $ed = k\phi(n) + 1$. Quindi, si deve dimostrare che:

$$M^{k\phi(n)+1} \bmod n = M^{k(p-1)(q-1)+1} \bmod n = M. \qquad (9.3)$$

Risultati fondamentali

Prima di dimostrare l'Equazione 9.3, si riassumono alcuni risultati fondamentali. Nel Capitolo 4 si è indicato che una delle proprietà dell'aritmetica modulare è la seguente:

$$[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n.$$

Da questa, dovrebbe essere evidente che se si ha $x \bmod n = 1$, allora $x^2 \bmod n = 1$ e per qualunque intero y si ha $x^y \bmod n = 1$. Analogamente, se si ha $x \bmod n = 0$, per qualsiasi intero y si ha $x^y \bmod n = 0$.

Un'altra proprietà dell'aritmetica modulare è:

$$[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n.$$

L'altro risultato che sarà necessario è il teorema di Eulero, sviluppato nel Capitolo 8. Se due interi a e b sono primi relativi, allora $a^{\phi(n)} \bmod n = 1$.

Dimostrazione

Per prima cosa si dimostra che $M^{k(p-1)(q-1)+1} \bmod p = M \bmod p$. Vi sono due casi da considerare.

- **Caso 1:** M e p non sono primi relativi, ovvero p divide M. In questo caso $M \bmod p = 0$ e quindi $M^{k(p-1)(q-1)+1} \bmod p = 0$. Quindi, $M^{k(p-1)(q-1)+1} \bmod p = M \bmod p$.
- **Caso 2:** Se M e p sono primi relativi, per il teorema di Eulero si ha $M^{\phi(p)} \bmod p = 1$. Si procede nel modo seguente.

$M^{k(p-1)(q-1)+1} \bmod p = [(M)M^{k(p-1)(q-1)}] \bmod p$
$= [(M)(M^{(p-1)})^{k(q-1)}] \bmod p$
$= [(M)(M^{\phi(p)})^{k(q-1)}] \bmod p$
$= (M \bmod p) \times [(M^{\phi(p)}) \bmod p]^{k(q-1)}$
$= (M \bmod p) \times (1)^{k(q-1)}$ (per il teorema di Eulero)
$= M \bmod p.$

Si osservi ora che:

$$[M^{k(p-1)(q-1)+1} - M] \bmod p = [M^{k(p-1)(q-1)+1} \bmod p] - [M \bmod p] = 0.$$

Dunque p divide $[M^{k(p-1)(q-1)+1} - M]$. Con il medesimo ragionamento si può dimostrare che q divide $[M^{k(p-1)(q-1)+1} - M]$. Dato che p e q sono primi distinti, deve esistere un intero r che soddisfa:

$$[M^{k(p-1)(q-1)+1} - M] = (pq)r = nr.$$

Dunque p divide $[M^{k(p-1)(q-1)+1} - M]$, quindi $M^{k\phi(n)+1} \bmod = M^{k(p-1)(q-1)+1} \bmod n = M$.

Appendice 9.B La complessità degli algoritmi

L'elemento principale per valutare la resistenza di un algoritmo di crittografia all'analisi crittografica è il tempo necessario per svolgere l'attacco. In generale non si può essere certi di aver trovato l'algoritmo di attacco più efficiente. Tutt'al più si può dire che, per un determinato algoritmo, il livello di impegno per un attacco è di un determinato ordine di grandezza. Quindi si può confrontare tale ordine di grandezza con la velocità dei microprocessori disponibili, per determinare il livello di sicurezza di un dato algoritmo.

Una misura comune dell'efficienza di un algoritmo è la sua **complessità temporale**. Si definisce complessità temporale di un algoritmo la funzione f(n) tale che, per tutti gli input di lunghezza n, l'esecuzione dell'algoritmo richiede al più f(n) passi. Pertanto, per determinate dimensioni di input e determinate velocità del microprocessore, la complessità temporale è un limite superiore del tempo di esecuzione.

La definizione contiene numerose ambiguità. Innanzitutto, la definizione di passo non è precisa. Un passo potrebbe essere un'unica operazione di una macchina di Turing, un'istruzione del microprocessore, un'istruzione in un linguaggio di alto livello e così via. Tuttavia, queste differenti definizioni di "passo" dovrebbero essere tutte correlate con semplici costanti moltiplicative. Per valori molto estesi di n, queste costanti non sono importanti. Ciò che è importante è la velocità con cui il tempo di esecuzione relativo cresce. Per esempio se ci si chiede se il problema di utilizzare chiavi a 50 cifre ($n = 10^{50}$) o a 100 cifre ($n = 10^{100}$) per RSA, non è necessario (né realmente possibile) conoscere esattamente quanto tempo sarebbe necessario per violare le chiavi di ciascuna dimensione. Piuttosto si è interessati ai valori generici del livello di impegno e a conoscere qual è l'impegno relativo necessario per chiavi di dimensioni sempre più grandi.

Il secondo problema è che, in generale, non si può stabilire una formula esatta per f(n) ma solo una forma approssimata. Anche in questo caso si è principalmente interessati alla velocità di variazione di f(n) all'aumentare di n.

Vi è una notazione matematica standard per caratterizzare la complessità temporale degli algoritmi e molto utile in questo contesto: f(n) = O(g(n)) se e solo se esistono due numeri a e M tali che:

$$|f(n)| \le a \times |g(n)|, \qquad n \ge M \tag{9.4}$$

Un esempio aiuterà a chiarire l'uso di questa notazione. Si supponga di voler valutare un polinomio generale della forma:

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$$

Il semplice algoritmo seguente è tratto da [POHL81]:

```
algorithm P1;
   n, i, j: integer; x, polyval: real;
   a, S: array [0..100] of real;
   begin
      read(x, n);
      for i := 0 upto n do
      begin
            S[i] := 1; read(a[i]);
            for j := 1 upto i do S[i] := x × S[i];
            S[i] := a[i] × S[i]
      end;
      polyval := 0;
```

```
    for i := 0 upto n do polyval := polyval + S[i];
    write ('value at', x, 'is', polyval)
end.
```

In questo algoritmo, ogni sottoespressione viene valutata separatamente. Ciascun S[i] richiede (i + 1) moltiplicazioni: i per calcolare S[i] e 1 per moltiplicarla per a[i]. Il calcolo di tutti gli n termini richiede:

$$\sum_{i=o}^{n}(i+1)=\frac{(n+2)(n+1)}{2}$$

moltiplicazioni. Vi sono anche (n + 1) somme che possono essere ignorate dato il gran numero di moltiplicazioni. Pertanto la complessità temporale di questo algoritmo è f(n) = (n + 2)(n + 1)/2. Ora si dimostrerà che f(n) = O(n^2). Dalla definizione dell'Equazione 9.4, si vuole mostrare che per a = 1 e M = 4, vale la relazione per g(n) = n^2. Il ragionamento sarà per induzione su n. La relazione vale per n = 4 poiché (4 + 2)(4 + 1)/2 = 15 < 4^2 = 16. Ora si supponga che valga per tutti i valori di n fino a k (ovvero (k + 2)(k + 1)/2 < k^2]). Allora, per $n = k + 1$:

$$\frac{(n+2)(n+1)}{2}=\frac{(k+3)(k+2)}{2}=\frac{(k+2)(k+1)}{2}+k+2\le k^2+k+2\le k^2+2k+1=(k+1)^2=n^2$$

Pertanto il risultato è vero per $n = k + 1$.

Tabella 9.5 Livello di impegno per vari livelli di complessità.

Complessità	Dimensioni	Operazioni
$\log_2 n$	$2^{10^{12}} = 10^{3 \times 10^{11}}$	10^{12}
n	10^{12}	10^{12}
n^2	10^6	10^{12}
n^6	10^2	10^{12}
2^n	39	10^{12}
$n!$	15	10^{12}

In generale, questa notazione utilizza il termine che cresce più velocemente. Per esempio:

1. O($ax^7 + 3x^3 + \sin(x)$) = O(ax^7) = O(x^7)
2. O($e^n + an^{10}$) = O(e^n)
3. O($n! + n^{50}$) = O($n!$).

Vi sono molti altri aspetti legati a questa notazione, con ramificazioni affascinanti. Per il lettore interessato, uno dei testi più utili è [GRAH94] e [KNUT97].

Un algoritmo con un input di dimensioni n si dice:

- **lineare** se il tempo di esecuzione è O(n);
- **polinomiale** se il tempo di esecuzione è O(n^t) per una costante t;
- **esponenziale** se il tempo di esecuzione è O($t^{h(n)}$) per una costante t e un polinomio h(n).

In generale un problema che può essere risolto in un tempo polinomiale è considerato fattibile mentre tutto ciò che si comporta peggio di un tempo polinomiale, e in particolare in un tempo esponenziale, è considerato impossibile. Ma occorre prendere con attenzione questi termini. Innanzitutto, se le dimensioni dell'input sono sufficientemente piccole, anche gli algoritmi più complessi diventano fattibili. Si supponga, per esempio, di avere un sistema in grado di eseguire 10^{12} operazioni per unità temporale. La Tabella 9.5 mostra le dimensioni di input che possono essere gestite in un'unità temporale per algoritmi di varia complessità. Per gli algoritmi esponenziali o fattoriali, è possibile considerare valori di input solo molto piccoli.

Il secondo elemento cui bisogna fare attenzione è il modo in cui viene caratterizzato l'input. Per esempio, la complessità dell'analisi crittografica di un algoritmo di crittografia può essere caratterizzata ugualmente bene in termini del numero di possibili chiavi o della lunghezza della chiave. Per esempio, il numero di chiavi possibili per AES è 2^{128} e la lunghezza della chiave è di 128 bit. Se si considera che una singola crittografia è un "passo" e che il numero di chiavi possibili è $N = 2^n$, la complessità temporale dell'algoritmo è lineare in termini del numero delle chiavi [O(N)] ma esponenziale in termini di lunghezza della chiave [O(2^n)].

Capitolo 10

La gestione delle chiavi; altri sistemi crittografici a chiave pubblica

Concetti essenziali

- Gli schemi di crittografia a chiave pubblica sono sicuri solo se viene garantita l'autenticità della chiave pubblica. Gli schemi dei **certificati** di chiave pubblica forniscono la sicurezza necessaria.
- Un semplice algoritmo a chiave pubblica è lo **scambio di chiavi Diffie-Hellman**. Questo protocollo consente a due utenti di stabilire una chiave segreta utilizzando uno schema a chiave pubblica basato sui logaritmi discreti. Il protocollo è sicuro solo se viene garantita l'autenticità dei due partecipanti.
- L'aritmetica delle curve ellittiche può essere utilizzata per sviluppare numerosi schemi di crittografia a curva ellittica (**ECC**), tra i quali lo scambio delle chiavi, la crittografia e la firma digitale.
- Ai fini degli schemi ECC, l'aritmetica delle curve ellittiche comporta l'impiego di un'equazione di curva ellittica definita su un campo finito. I coefficienti e le variabili nell'equazione sono elementi del campo finito. Sono stati sviluppati schemi che utilizzano i campi Z_p e GF(2^n).

Questo capitolo prosegue la panoramica della crittografia a chiave pubblica. In particolare esamina la distribuzione e la gestione delle chiavi nei sistemi a chiave pubblica, compresa una discussione sullo scambio di chiavi Diffie-Hellman. Infine introduce la crittografia a curva ellittica.

10.1 La gestione delle chiavi

Nel Capitolo 7 è stato affrontato il problema della distribuzione delle chiavi segrete. Uno dei principali ruoli della crittografia a chiave pubblica è quello di risolvere il problema della distribuzione delle chiavi. A questo riguardo vi sono due diversi aspetti nell'utilizzo della crittografia a chiave pubblica.

- La distribuzione delle chiavi pubbliche.
- L'uso della crittografia a chiave pubblica per distribuire le chiavi segrete.

Distribuzione delle chiavi pubbliche

Sono state proposte varie tecniche per la distribuzione delle chiavi pubbliche. Praticamente tutte le proposte possono essere raggruppate nei seguenti schemi generali.

- Annuncio pubblico.
- Elenco pubblico.
- Autorità di distribuzione delle chiavi pubbliche.
- Certificati a chiavi pubbliche.

Annuncio pubblico

La crittografia a chiave pubblica si basa sul fatto che la chiave è "pubblica". Pertanto, se si trova un algoritmo a chiave pubblica ad ampia diffusione, per esempio RSA, chiunque potrà inviare la propria chiave pubblica a chiunque altro o trasmettere in broadcast la chiave a un'intera comunità (Figura 10.1). Per esempio, data la crescente popolarità del sistema PGP (Pretty Good Privacy), trattato nel Capitolo 15, che impiega proprio un algoritmo RSA, molti utenti PGP hanno adottato la pratica di aggiungere la propria chiave pubblica ai messaggi inviati ai forum pubblici, per esempio ai gruppi di discussione di USENET e alle mailing list.

Sebbene questo approccio sia comodo, presenta vari punti deboli. Chiunque è in grado di alterare a piacere questo tipo di annuncio pubblico. Ovvero chiunque potrebbe fingere di essere l'utente A e inviare una chiave pubblica a chiunque altro o trasmettere in broadcast tale chiave pubblica. Finché l'utente A non scoprirà l'inganno e avvertirà tutti gli altri, il responsabile sarà in grado di leggere tutti i messaggi crittografati destinati ad A e potrà utilizzare le chiavi create per l'autenticazione (vedere la Figura 9.3).

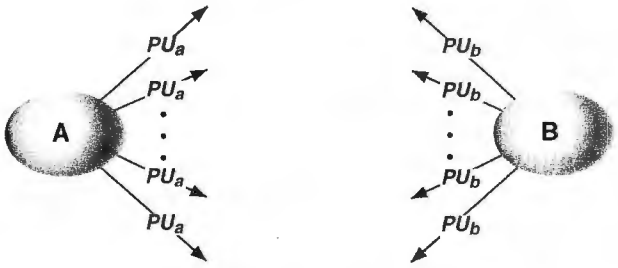


Figura 10.1 Distribuzione non controllata della chiave pubblica.

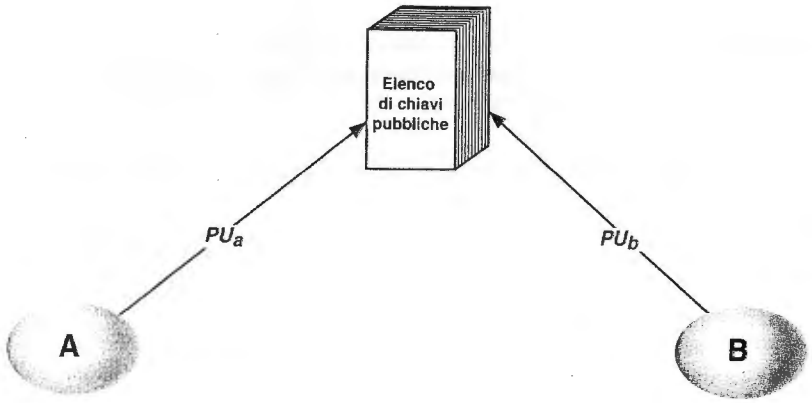


Figura 10.2 Pubblicazione della chiave pubblica.

Elenco pubblico

Si può ottenere un livello di sicurezza superiore impiegando un elenco dinamico e pubblico di tutte le chiavi pubbliche. La manutenzione e la distribuzione dell'elenco deve essere sotto la responsabilità di un'entità o un'organizzazione fidata (vedere la Figura 10.2). Tale schema prevede anche i seguenti elementi.

1. L'autorità gestisce un elenco con una voce [*nome,chiave-pubblica*] per ciascun partecipante.
2. Ciascun partecipante registra una chiave pubblica presso l'autorità di elenco. La registrazione deve essere svolta di persona o tramite una forma di autenticazione sicura.
3. Un partecipante può sostituire la propria chiave con una nuova chiave in qualsiasi momento, semplicemente perché desidera sostituire una chiave pubblica già utilizzata per una grande quantità di dati o perché la corrispondente chiave privata è stata violata.
4. I partecipanti possono accedere all'elenco anche in modo elettronico. Per questo scopo è obbligatorio impiegare una comunicazione autenticata e sicura fra l'autorità e i partecipanti.

Questo schema è chiaramente più sicuro rispetto agli annunci pubblici individuali ma presenta comunque dei punti deboli. Se un estraneo dovesse riuscire a ottenere o a calcolare la chiave privata dell'autorità dell'elenco, potrebbe distribuire chiavi pubbliche contraffatte e successivamente fingersi un qualsiasi partecipante e sottrarre i messaggi inviati a chiunque. Un altro modo per ottenere lo stesso risultato consiste nel violare i record gestiti dall'autorità.

Autorità di distribuzione delle chiavi pubbliche

Una maggiore sicurezza nella distribuzione delle chiavi pubbliche può essere ottenuta con un controllo più rigido sulla distribuzione delle chiavi pubbliche. Questa situazione è illustrata nella Figura 10.3 che si basa su una figura contenuta in [POPE79]. Anche in questo caso si presuppone che un'autorità centrale gestisca un elenco dinamico contenente le

chiavi pubbliche di tutti i partecipanti. Inoltre, ciascun partecipante conosce tramite un canale sicuro la chiave pubblica dell'autorità e solo l'autorità conosce la corrispondente chiave privata. Avvengono le seguenti operazioni (il numero coincide con quello indicato nella Figura 10.3).

1. A invia un messaggio con timestamp all'autorità delle chiavi pubbliche per richiedere la chiave pubblica corrente di B.
2. L'autorità risponde con un messaggio crittografato con la propria chiave privata, PR_{auth}. Pertanto A sarà in grado di decrittografare il messaggio utilizzando la chiave pubblica dell'autorità e quindi avrà la certezza che il messaggio proviene dall'autorità. Il messaggio include le seguenti informazioni.
   - La chiave pubblica di B, PU_b, che A può utilizzare per crittografare i messaggi destinati a B.
   - La richiesta originaria, per consentire ad A di confrontare questa risposta con la richiesta corrispondente e verificare che la richiesta originale non sia stata alterata prima di essere stata ricevuta dall'autorità.
   - Il timestamp originario, in modo che A possa determinare che non si tratti di un vecchio messaggio dall'autorità contenente una chiave diversa dalla chiave pubblica attuale di B.
3. A memorizza la chiave pubblica di B e la utilizza per crittografare un messaggio per B contenente il proprio identificatore (ID_A) e un valore nonce (N_1) che viene utilizzato per identificare univocamente questa transazione.
4, 5. B preleva la chiave pubblica di A dall'autorità così come A ha prelevato la chiave pubblica di B.

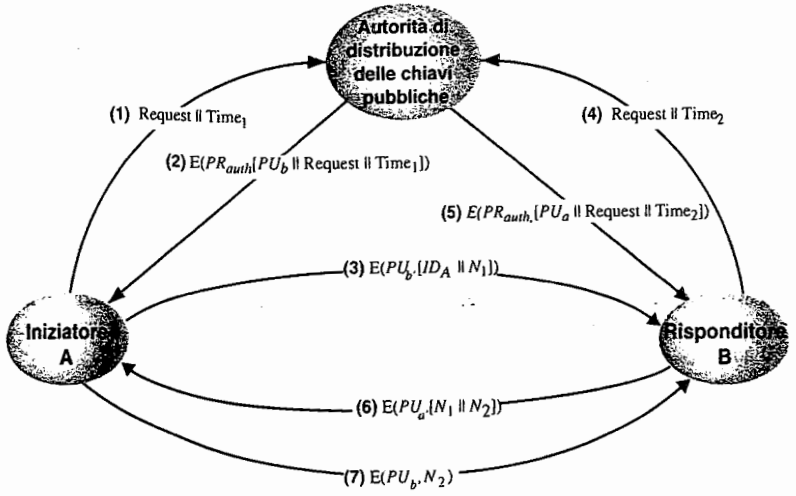


Figura 10.3 Distribuzione della chiave pubblica.

A questo punto le chiavi pubbliche sono state consegnate con sicurezza ad A e B e le parti possono iniziare uno scambio protetto. Tuttavia è opportuno svolgere due ulteriori passi.

6. B invia ad A un messaggio crittografato con PU_a contenente il valore nonce (N_1) di A e un nuovo valore nonce generato da B (N_2). Poiché solo B può aver decrittografato il messaggio (3), la presenza di N_1 nel messaggio (6) garantisce ad A che il corrispondente sia effettivamente B.
7. A restituisce N_2, crittografato utilizzando la chiave pubblica di B per assicurare a B che il suo corrispondente è A.

Pertanto, è necessario impiegare un totale di sette messaggi. Tuttavia, i primi quattro messaggi devono essere scambiati molto raramente poiché sia A che B possono salvare la chiave pubblica dell'altro, una tecnica chiamata memorizzazione cache. Periodicamente un utente dovrà richiedere una nuova copia delle chiavi pubbliche di tutti i suoi corrispondenti in modo da assicurarsi di avere un "mazzo di chiavi" aggiornato.

Certificati a chiave pubblica

La situazione rappresentata nella Figura 10.3 è interessante ma presenta ancora qualche difetto. L'autorità di gestione delle chiavi pubbliche potrebbe costituire il collo di bottiglia del sistema in quanto ciascun utente deve far riferimento all'autorità per ottenere la chiave pubblica di qualsiasi altro utente che vuole contattare. Come prima, l'elenco di nomi e chiavi pubbliche gestito dall'autorità è comunque vulnerabile ad attacchi esterni.

Un approccio alternativo, suggerito inizialmente da Kohnfelder [KOHN78], prevede l'impiego di **certificati** che possono essere utilizzati dai partecipanti per scambiarsi le chiavi senza contattare un'autorità di distribuzione delle chiavi pubbliche, ma non la medesima affidabilità. Sostanzialmente, un certificato consiste in una chiave pubblica e in un identificatore del suo proprietario, il tutto firmato da una terza parte fidata. Solitamente la terza parte è un'autorità di certificazione, per esempio un'agenzia governativa o un'istituzione finanziaria, che gode della fiducia della comunità di utenti. Un utente può sottoporre la propria chiave pubblica all'autorità con una modalità sicura per ottenere il certificato corrispondente, che potrà poi essere pubblicato. Chiunque abbia bisogno della chiave pubblica di questo utente potrà ottenere il certificato e verificarne la validità tramite la firma fidata a esso associata. Un partecipante può anche inviare le informazioni sulla propria chiave agli altri trasmettendo il proprio certificato. Gli altri partecipanti possono verificare che il certificato è stato creato dall'autorità. Ecco i requisiti di questo schema.

1. Qualsiasi partecipante può leggere un certificato per determinare il nome e la chiave pubblica del possessore del certificato.
2. Ogni partecipante può verificare che il certificato abbia avuto origine dall'autorità di certificazione e non sia stato contraffatto.
3. Solo l'autorità di certificazione può creare e aggiornare i certificati.

Questi requisiti sono soddisfatti dalla proposta originale di [KOHN78]. Denning [DENN83] ha aggiunto il seguente requisito.

4. Qualsiasi partecipante può verificare che il certificato sia aggiornato.

Questo schema di certificazione è rappresentato nella Figura 10.4. Ciascun partecipante fa riferimento all'autorità di certificazione fornendo una chiave pubblica e richiedendo un certificato. L'applicazione deve essere svolta di persona o tramite comunicazioni autenticate e sicure. Per il partecipante A, l'autorità fornisce un certificato nella forma:

$$C_a = E(PR_{auth}, [T \| ID_A \| KU_a])$$

dove PR_{auth} è la chiave privata utilizzata dall'autorità e T è un timestamp. A può quindi trasmettere questo certificato a qualsiasi altro partecipante che legge e verifica il certificato nel seguente modo:

$$D(PU_{auth}, C_A) = D(PU_{auth}, E(PR_{auth}, [T \| ID_A \| PU_a])) = (T \| ID_A \| PU_a)$$

Per decrittografare il certificato, il destinatario usa la chiave pubblica dell'autorità, PU_{auth}. Poiché il certificato è leggibile solo utilizzando la chiave pubblica dell'autorità, questo prova che il certificato proviene dall'autorità di certificazione. Gli elementi ID_A e PU_a forniscono al destinatario il nome e la chiave pubblica del detentore del certificato. Il valore timestamp T conferma che il certificato è aggiornato. Esso ha lo scopo di impedire la seguente situazione. La chiave privata di A viene scoperta da un estraneo. A genera una nuova coppia chiave privata/chiave pubblica e chiede all'autorità di certificazione un nuovo certificato. Nel frattempo l'estraneo invia il vecchio certificato a B, e potrà leggere gli eventuali messaggi crittografati da B utilizzando la vecchia chiave pubblica.

In questo contesto, la violazione di una chiave privata è paragonabile alla perdita di una carta di credito. Il proprietario annulla il numero di carta di credito ma rimane comunque a rischio finché tutti non saranno avvisati che la vecchia carta di credito è ormai obsoleta. Pertanto il valore timestamp funge da data di scadenza. Se un certificato è troppo vecchio, si presuppone che sia scaduto.

Lo standard X.509 è divenuto lo schema universalmente accettato per strutturare i certificati di chiave pubblica. I certificati X.509 sono utilizzati nella maggior parte delle applicazioni di sicurezza di rete, tra le quali IPSec, SSL (Secure Sockets Layer), SET (Secure Electronic Transactions) e S/MIME, e saranno tutti discussi nella Parte Seconda di questo volume. X.509 è affrontato in modo particolareggiato nel Capitolo 14.

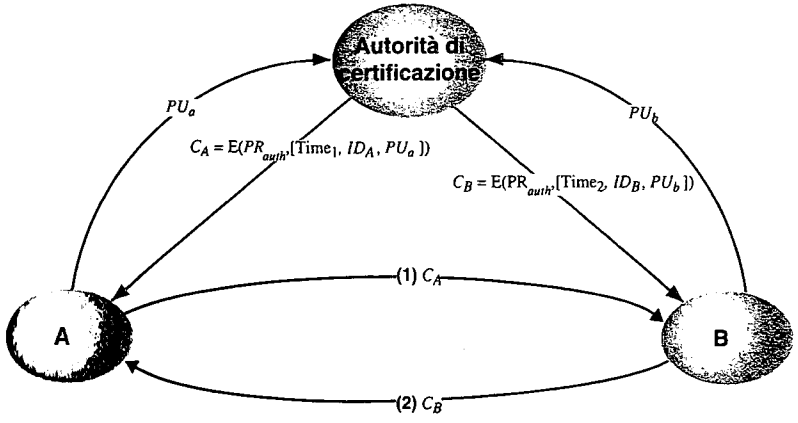


Figura 10.4 Scambio di certificati a chiave pubblica.

Distribuzione delle chiavi segrete tramite crittografia a chiave pubblica

Una volta che le chiavi pubbliche sono state distribuite o sono diventate accessibili, è possibile svolgere comunicazioni sicure in grado di resistere alle intercettazioni (Figura 9.2), alle falsificazioni (Figura 9.3) o a entrambe le cose (Figura 9.4). Tuttavia pochi utenti vorranno fare un uso esclusivo della crittografia a chiave pubblica per le comunicazioni, data la velocità relativamente modesta che è possibile raggiungere. Di conseguenza la crittografia a chiave pubblica è utilizzata più ragionevolmente per la distribuzione delle chiavi segrete da utilizzare per la crittografia convenzionale.

Semplice distribuzione della chiave segreta

Merkle [MERK79] ha progettato uno schema estremamente semplice, illustrato dalla Figura 10.5. Se A vuole comunicare con B, viene impiegata la seguente procedura.

1. A genera una coppia chiave pubblica/privata $\{PU_a, PR_a\}$ e trasmette un messaggio a B costituito da PU_a e dal proprio identificatore, ID_A.
2. B genera una chiave segreta, K_s e la trasmette ad A, crittografata con la chiave pubblica di A.
3. A calcola $D(PR_a, E(PU_a, K_s))$ per ottenere la chiave segreta. Poiché solo A può decrittografare il messaggio, solo A e B conosceranno il valore di K_s.
4. A elimina PU_a e PR_a e B elimina PU_a.

Ora A e B possono comunicare con sicurezza utilizzando la normale crittografia convenzionale e la chiave di sessione K_s. Al completamento dello scambio, sia A che B eliminano K_s. Nonostante la sua semplicità, questo è un protocollo molto interessante. Non esiste una chiave né prima della comunicazione né dopo il suo completamento. Pertanto il rischio di violazione delle chiavi è minimo. Contemporaneamente, le comunicazioni sono al sicuro contro ogni intercettazione.

Il protocollo illustrato nella Figura 10.5 è però vulnerabile a un avversario che possa intercettare i messaggi e ritrasmetterli successivamente o sostituirli con altri messaggi (Fi-

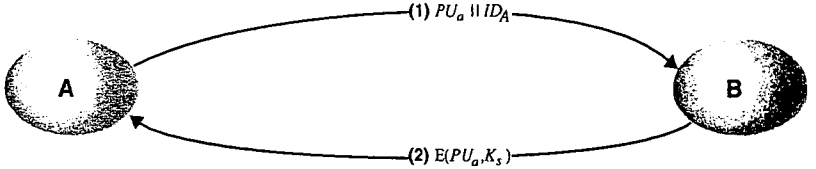


Figura 10.5 Semplice utilizzo della crittografia a chiave pubblica per concordare una chiave di sessione.

gura 14.c). Questo tipo di attacco è chiamato **attacco man-in-the-middle** [RIVE84]. In questo caso se un estraneo, E, ha il controllo di un canale di comunicazione intermedio, potrà violare le comunicazioni nel seguente modo senza essere rilevato.

1. A genera una coppia di chiavi pubblica/privata $\{PU_a, PR_a\}$ e trasmette un messaggio destinato a B costituito da PU_a e dal proprio identificatore ID_A.
2. E intercetta il messaggio, crea la propria coppia di chiavi pubblica/privata $\{PU_e, PR_e\}$ e trasmette $PU_e \| ID_A$ a B.
3. B genera una chiave segreta, K_s, e trasmette $E(PU_e, K_s)$.
4. E intercetta il messaggio e apprende K_s calcolando $D(PR_e, E(PU_e, K_s))$.
5. E trasmette ad A $E(PU_a, K_s)$.

Il risultato è che ora A e B conoscono K_s ma non sanno che K_s è nota anche a E. A e B possono scambiarsi messaggi utilizzando K_s. A questo punto E non interferisce più attivamente con il canale di comunicazione poiché gli basta intercettarlo. Conoscendo K_s, E sarà in grado di decrittografare tutti i messaggi senza che A e B possano rendersi conto del problema. Di conseguenza questo semplice protocollo è impiegabile solamente quando l'unica minaccia sia l'intercettazione passiva.

Distribuzione della chiave segreta con segretezza e autenticazione

La Figura 10.6, basata su un approccio suggerito in [NEED78], fornisce la protezione sia contro gli attacchi attivi che passivi. Si presume che A e B si siano scambiati le chiavi pubbliche tramite uno degli schemi descritti in precedenza. Quindi si svolgono le seguenti operazioni.

1. A utilizza la chiave pubblica di B per crittografare un messaggio per B contenente il proprio identificatore (ID_A) e un valore nonce (N_1) che viene utilizzato per identificare univocamente la transazione.
2. B invia ad A un messaggio crittografato con PU_a e contenente il valore nonce di A (N_1) e un nuovo valore nonce generato da B (N_2). Poiché solo B può aver decrittografato il messaggio (1), la presenza di N_1 nel messaggio (2) garantisce ad A che il corrispondente è in effetti B.
3. A restituisce N_2 crittografato con la chiave pubblica di B, per assicurare a B che sta effettivamente colloquiando con A.
4. A seleziona una chiave segreta K_s e invia a B $M = E(PU_b, E(PR_a, K_s))$. La crittografia di questo messaggio con la chiave pubblica di B garantisce che solo B possa leggerlo; la crittografia con la chiave privata di A garantisce che solo A possa averlo inviato.
5. B calcola $D(PU_a, D(PR_b, M))$ per recuperare la chiave segreta.

Si noti che i primi tre passi di questo schema sono uguali agli ultimi tre passi indicati nella Figura 10.3. Il risultato è che questo schema garantisce sia la segretezza che l'autenticazione nello scambio di una chiave segreta.

Uno schema ibrido

Esiste anche uno schema ibrido, utilizzato per distribuire le chiavi segrete nei mainframe IBM [LE93] tramite la crittografia a chiave pubblica. Questo schema utilizza un centro di distribuzione delle chiavi KDC (Key Distribution Center) che condivide una chiave master

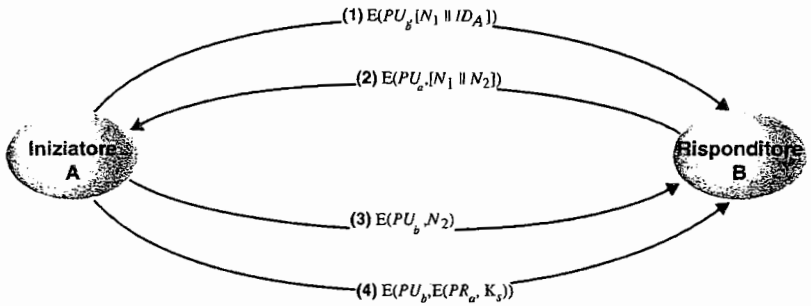


Figura 10.6 Distribuzione delle chiavi segrete tramite chiave pubblica.

segreta con ciascun utente e distribuisce le chiavi di sessione segrete crittografate con la chiave master. Per distribuire le chiavi master viene utilizzato uno schema a chiave pubblica. Viene utilizzato un approccio a tre livelli per i seguenti motivi.

- **Prestazioni:** vi sono molte applicazioni, specialmente quelle orientate alle transazioni, nelle quali le chiavi di sessione cambiano frequentemente. La distribuzione delle chiavi di sessione tramite crittografia a chiave pubblica produrrebbe un degrado globale delle prestazioni del sistema dato l'elevato carico computazionale della crittografia e decrittografia a chiave pubblica. Con una gerarchia a tre livelli, la crittografia a chiave pubblica viene utilizzata solo occasionalmente per aggiornare la chiave master fra un utente e il centro KDC.
- **Compatibilità all'indietro:** lo schema ibrido può essere facilmente sovrapposto a uno schema KDC esistente, richiedendo minime modifiche software.

L'aggiunta di un livello a chiave pubblica fornisce un mezzo sicuro ed efficiente per distribuire le chiavi master. Questo rappresenta un vantaggio in una configurazione in cui un unico centro KDC serve un'ampia base di utenti distribuiti.

10.2 Lo scambio di chiavi Diffie-Hellman

Il primo algoritmo a chiave pubblica pubblicato è comparso nel documento scritto da Diffie e Hellman che definiva la crittografia a chiave pubblica [DIFF76b] e pertanto è chiamato scambio di chiavi Diffie-Hellman.[1] Questa tecnica per lo scambio delle chiavi viene impiegata da vari prodotti commerciali.

Lo scopo dell'algoritmo è quello di consentire a due utenti di scambiarsi con sicurezza una chiave che possa poi essere utilizzata per la crittografia di ulteriori messaggi. L'algoritmo stesso è limitato allo scambio di valori segreti. L'efficacia dell'algoritmo di Diffie-Hellman dipende dalla difficoltà del calcolo dei logaritmi discreti. In breve si può definire il logarit-

[1] Williamson, del CESG britannico ha pubblicato lo stesso schema qualche mese prima in un documento segreto [WILL76] e sostiene di averlo scoperto molti anni prima; per informazioni consultare [ELLI99].

mo discreto nel seguente modo. Innanzitutto si definisce una radice primitiva di un numero primo p come un numero le cui potenze modulo p generano tutti gli interi compresi fra 1 e $p - 1$. In pratica, se a è radice primitiva del numero primo p, allora i numeri:

$$a \bmod p, a^2 \bmod p, \ldots, a^{p-1} \bmod p$$

sono distinti e sono costituiti dagli interi compresi fra 1 e $p - 1$ in una certa permutazione.

Per un qualsiasi intero b e una radice primitiva a di un numero primo p, si può trovare un esponente univoco i tale che:

$$b \equiv a^i \pmod p \qquad \text{dove } 0 \le i \le (p-1)$$

L'esponente i è chiamato logaritmo discreto di b per la base a, modulo p. Questo valore è denotato con $\text{dlog}_{a,p}(b)$. Per approfondire i logaritmi discreti consultare il Capitolo 8.

L'algoritmo

Lo scambio di chiavi Diffie-Hellman è riepilogato nella Figura 10.7. In questo schema vi sono due numeri pubblicamente noti: un numero primo q e un intero α che è radice primitiva di q. Si supponga che gli utenti A e B vogliano scambiarsi una chiave. L'utente A seleziona un intero casuale $X_A < q$ e calcola $Y_A = \alpha^{X_A} \bmod q$. Analogamente l'utente B seleziona indipendentemente un intero casuale $X_B < q$ e calcola $Y_B = \alpha^{X_B} \bmod q$. Ognuno dei due mantiene privato il valore X e rende il valore Y disponibile all'altro utente. L'utente A calcola la chiave come $K = (Y_B)^{X_A} \bmod q$ e l'utente B calcola la chiave come $K = (Y_A)^{X_B} \bmod q$. Questi due calcoli producono risultati identici:

$$\begin{aligned}
K &= (Y_B)^{X_A} \bmod q \\
&= (\alpha^{X_B} \bmod q)^{X_A} \bmod q \\
&= (\alpha^{X_B})^{X_A} \bmod q && \text{in base alle regole dell'aritmetica modulare} \\
&= \alpha^{X_B X_A} \bmod q \\
&= (\alpha^{X_A})^{X_B} \bmod q \\
&= (\alpha^{X_A} \bmod q)^{X_B} \bmod q \\
&= (Y_A)^{X_B} \bmod q
\end{aligned}$$

Il risultato è che le due parti si sono scambiate una chiave segreta. Inoltre, poiché X_A e X_B sono private, un estraneo potrà avere solo i seguenti dati con cui lavorare: q, α, Y_A e Y_B e pertanto sarà costretto a calcolare il logaritmo discreto per determinare la chiave. Per esempio, per determire la chiave privata dell'utente B, l'estraneo dovrà calcolare:

$$X_B = \text{dlog}_{\alpha,q}(Y_B)$$

Egli può quindi calcolare la chiave K nello stesso modo in cui la calcola l'utente B.

La sicurezza dello scambio di chiavi Diffie-Hellman si basa sul fatto che, mentre è relativamente facile calcolare dei valori esponenziali modulo un numero primo, è molto difficile calcolare i logaritmi discreti. Per valori primi molto estesi, quest'ultima operazione è considerata impossibile.

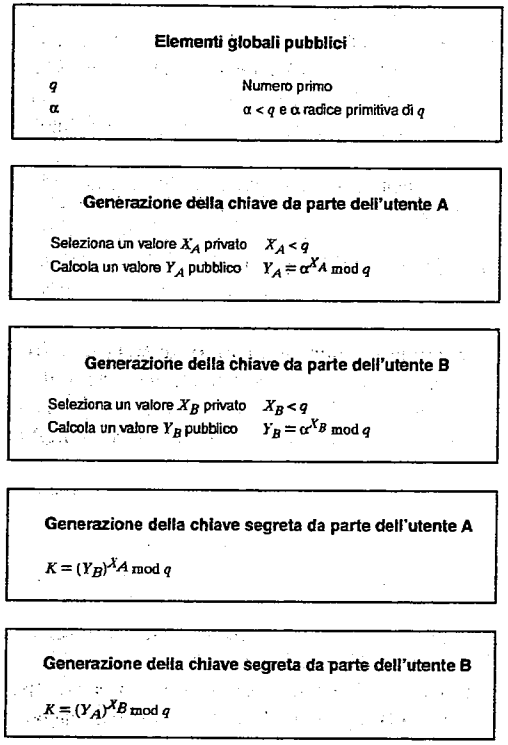


Figura 10.7 L'algoritmo di scambio di chiavi Diffie-Hellman.

Ecco un esempio. Lo scambio di chiavi si basa sull'uso del numero primo $q = 353$ e di una radice primitiva di 353, in questo caso $\alpha = 3$. A e B selezionano rispettivamente le chiavi segrete $X_A = 97$ e $X_B = 233$. Ognuno di essi calcola la propria chiave pubblica:

$$\text{A calcola } Y_A = 3^{97} \bmod 353 = 40$$
$$\text{B calcola } Y_B = 3^{233} \bmod 353 = 248$$

Dopo che si sono scambiati le chiavi pubbliche, ognuno di essi calcola la chiave segreta comune:

$$\text{A calcola } K = (Y_B)^{X_A} \bmod 353 = 248^{97} \bmod 353 = 160$$
$$\text{B calcola } K = (Y_A)^{X_B} \bmod 353 = 40^{233} \bmod 353 = 160$$

Si suppone che l'estraneo abbia a disposizione le seguenti informazioni:

$$q = 353; \quad \alpha = 3; \quad Y_A = 40; \quad Y_B = 248$$

In questo semplice esempio, sarebbe possibile determinare tramite forza bruta la chiave segreta 160. In particolare, un hacker E può determinare la chiave comune scoprendo una soluzione all'equazione $3^a \bmod 353 = 40$ o all'equazione $3^b \bmod 353 = 248$. L'approccio a forza bruta consiste nel calcolare le potenze di 3 modulo 353, fermandosi quando il risultato è uguale a 40 o 248. La risposta desiderata si ottiene con il valore esponenziale 97 che fornisce $3^{97} \bmod 353 = 40$.

Per numeri di grandi dimensioni questo problema diviene di difficile soluzione.

Protocolli per lo scambio delle chiavi

La Figura 10.8 mostra un semplice protocollo che utilizza il calcolo Diffie-Hellman. Si supponga che l'utente A voglia configurare una connessione con l'utente B e usare una chiave segreta per crittografare i messaggi di tale connessione. L'utente A può generare una chiave privata X_A mono-uso, calcolare Y_A e inviarla all'utente B. L'utente B risponde generando un valore privato X_B, calcolando Y_B e inviandola all'utente A. Entrambi gli utenti possono ora calcolare la chiave. I valori pubblici q e α devono essere noti prima di tutta quest'operazione. Alternativamente l'utente A può selezionare i valori di q e α e includerli nel primo messaggio.

Come esempio di un altro uso dell'algoritmo di Diffie-Hellman, si supponga che un gruppo di utenti (per esempio tutti gli utenti di una rete locale) generino un valore privato di lunga durata X_i (per l'utente i) e calcolino un valore pubblico Y_i. Questi valori pubblici, insieme ai valori pubblici globali q e α, vengono memorizzati in un elenco centrale. In qualsiasi momento l'utente j può accedere al valore pubblico di i, calcolare una chiave segreta e utilizzarla per inviare ad i un messaggio crittografato. Se l'elenco centrale è fidato, questa forma di comunicazione garantisce sia la segretezza che un certo livello di autenticazione. Poiché solo i e j possono determinare la chiave, nessun altro utente potrà leggere il messaggio (segretezza). Il destinatario i sa che solo l'utente j può aver creato il messaggio utilizzando questa

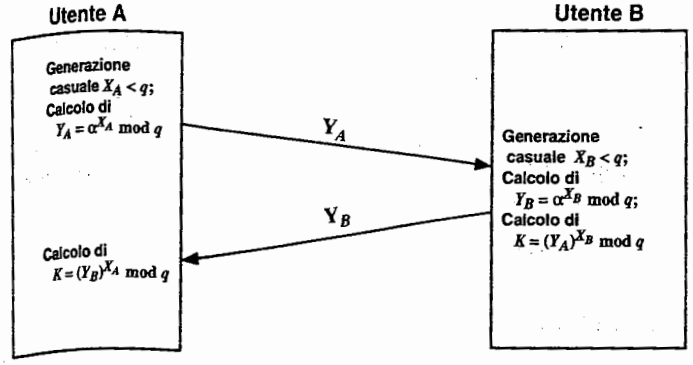


Figura 10.8 Lo scambio di chiavi Diffie-Hellman.

chiave (autenticazione). Tuttavia la tecnica non protegge contro gli attacchi a replica di messaggi (attacchi *replay*).

Attacco man-in-the-middle

Il protocollo schematizzato nella Figura 10.8 è vulnerabile all'attacco man-in-the-middle. Si supponga che Alice e Bob vogliano scambiarsi le chiavi e che Darth sia l'avversario. L'attacco procede nel modo seguente:

1. Darth si prepara all'attacco generando casualmente due chiavi private X_{D1} e X_{D2}, per poi calcolare le chiavi pubbliche corrispondenti Y_{D1} e Y_{D2}.
2. Alice trasmette Y_A a Bob.
3. Darth intercetta Y_A e trasmette Y_{D1} a Bob. Darth calcola anche $K2 = (Y_A)^{X_{D2}} \bmod q$.
4. Bob riceve Y_{D1} e calcola $K1 = (Y_{D1})^{X_B} \bmod q$.
5. Bob trasmette X_A ad Alice.
6. Darth intercetta X_A e trasmette Y_{D2} ad Alice. Darth calcola $K1 = (Y_B)^{X_{D1}} \bmod q$.
7. Alice riceve Y_{D2} e calcola $K2 = (Y_{D2})^{X_A} \bmod q$.

A questo punto Bob ed Alice credono di condividere una chiave segreta, mentre in realtà Bob e Darth condividono la chiave segreta $K1$ ed Alice e Darth condividono la chiave segreta $K2$. Tutte le comunicazioni successive fra Bob e Alice sono compromesse nel modo seguente:

1. Alice invia il messaggio crittografato M: $E(K2, M)$.
2. Darth intercetta il messaggio crittografato e ne esegue la decrittografia per recuperare M.
3. Darth invia a Bob $E(K1, M)$ o $E(K1, M')$, dove M' corrisponde a qualsiasi messaggio. Nel primo caso Darth vuole semplicemente spiare la comunicazione senza alterarla. Nel secondo caso Darth modifica il messaggio destinato a Bob.

Il protocollo di scambio della chiave è vulnerabile a tale attacco perché non autentica i partecipanti. Questo problema può essere evitato tramite l'impiego delle firme digitali e dei certificati di chiave pubblica, argomenti affrontati nei Capitoli 13 e 14.

10.3 Aritmetica a curva ellittica

La maggior parte dei prodotti e degli standard che utilizzano la crittografia a chiave pubblica per la crittografia e le firme digitali utilizza l'algoritmo RSA. Come si è visto, la lunghezza della chiave per l'utilizzo sicuro di RSA è aumentata nel corso degli ultimi anni e questo ha sottoposto le applicazioni che utilizzano RSA a un maggiore carico computazionale. Questo tipo di carico ha delle implicazioni, specialmente per i siti di commercio elettronico che svolgono una grande quantità di transazioni sicure. Recentemente si è reso disponibile un sistema concorrente a RSA: la crittografia a curva ellittica (ECC – Elliptic Curve Cryptography). La crittografia a curva ellittica è già stata sottoposta a vari tentativi di standardizzazione, fra cui lo standard IEEE P1363 per la crittografia a chiave pubblica.

Il principale interesse della crittografia a curva ellittica rispetto a RSA consiste nel fatto che sembra offrire la stessa sicurezza con chiavi di dimensioni di gran lunga inferiori riducendo pertanto il carico di elaborazione. D'altra parte, sebbene la teoria della crittografia a curva ellittica sia disponibile da qualche tempo, solo recentemente sono apparsi dei prodotti che impiegano la crittografia a curva ellittica e pertanto vi è stato un certo interesse nell'analisi crittografica che consentisse l'individuazione dei suoi punti deboli. Di conseguenza, il livello di fiducia nella crittografia a curva ellittica non è ancora così elevato quanto quello in RSA.

La crittografia a curva ellittica è fondamentalmente più difficile da spiegare rispetto alla crittografia RSA o Diffie-Hellman e la sua descrizione matematica non rientra negli scopi di questo volume. Questa parte del capitolo fornisce alcuni elementi di base sulle curve ellittiche e la crittografia a curva ellittica. Si comincerà con un breve ripasso del concetto di gruppo abeliano. Poi si parlerà di curve ellittiche definite sui numeri reali. Quindi si parlerà delle curve ellittiche definite sui campi finiti. Infine si potrà esaminare la crittografia a curva ellittica. Il lettore interessato può tornare a consultare il materiale sui campi finiti presentato nel Capitolo 4.

I gruppi abeliani

Come si è detto nel Capitolo 4, un **gruppo** abeliano G (rappresentato come $\{G, \bullet\}$), è un insieme di elementi con un'operazione binaria, rappresentata da $\bullet$, che associa a ciascuna coppia ordinata (a, b) di elementi di G un elemento $(a \bullet b)$ sempre in G tale che valgano i seguenti assiomi.[2]

(A1) Chiusura:	se a e b appartengono a G allora anche $a \bullet b$ appartiene a G.
(A2) Associatività:	$a \bullet (b \bullet c) = (a \bullet b) \bullet c$ per ogni a, b, c in G.
(A3) Elemento identità:	vi è un elemento e in G tale che $a \bullet e = e \bullet a = a$ per ogni a in G.
(A4) Elemento inverso:	per ciascun a in G vi è un elemento a' sempre in G tale che $a \bullet a' = a' \bullet a = e$.
(A5) Commutatività:	$a \bullet b = b \bullet a$ per ogni a, b in G.

Diversi algoritmi crittografici a chiave pubblica si basano sull'utilizzo di un gruppo abeliano. Per esempio, lo scambio di chiavi Diffie-Hellman prevede la moltiplicazione di coppie di interi diversi da zero modulo un numero primo q. Le chiavi vengono generate tramite esponenziazione sul gruppo (dove l'esponenziazione è definita come una moltiplicazione ripetuta). Per esempio, $a^k \bmod q = \underbrace{(a \times a \times \ldots \times a)}_{k \text{ volte}} \bmod q$. Per attaccare l'algoritmo Diffie-Hellman, si deve determinare k sulla base di a e a^k; si tratta di un problema riguardante i logaritmi discreti.

Per la crittografia a curva ellittica, viene utilizzata un'operazione chiamata somma sulle curve ellittiche. La moltiplicazione è definita come una somma ripetuta. Per esempio, $a \times k = \underbrace{(a + a + \ldots + a)}_{k \text{ volte}}$ dove la somma viene eseguita su una curva ellittica. L'analisi

[2] L'operatore $\bullet$ è generico e può essere la somma, la moltiplicazione o qualche altra operazione matematica.

crittografica comporta la ricerca di k sulla base di a e $(a \times k)$. Una curva ellittica è definita da un'equazione in due variabili con coefficienti. Per la crittografia, le variabili e i coefficienti sono elementi di un campo finito, condizione che porta alla definizione di un gruppo abeliano finito. Prima di vedere ciò si parlerà delle curve ellittiche in cui le variabili e i coefficienti sono numeri reali, un caso forse più facile da intuire.

Curve ellittiche sui numeri reali

Le curve ellittiche non sono ellissi. Si chiamano in questo modo poiché sono descritte da equazioni cubiche, simili a quelle utilizzate per calcolare la circonferenza di un'ellisse. In generale le equazioni cubiche delle curve ellittiche assumono la seguente forma:

$$y^2 + axy + by = x^3 + cx^2 + dx + e$$

dove a, b, c, d ed e sono numeri reali e x e y assumono valori reali.[3] Per gli scopi di questo volume, è sufficiente limitarsi alle equazioni nella forma:

$$y^2 = x^3 + ax + b \tag{10.1}$$

Tali equazioni sono dette cubiche o di grado 3 in quanto l'esponente più elevato è 3. Nella definizione di una curva ellittica vi è anche un singolo elemento denotato O e chiamato *punto infinito* o *punto zero* di cui si parlerà più avanti. Per tracciare una curva di questo tipo occorre calcolare:

$$y = \sqrt{x^3 + ax + b}$$

Per determinati valori di a e v, la curva è costituita da valori positivi e negativi di y per ciascun valore di x. Pertanto ciascuna curva risulta simmetrica rispetto a $y = 0$. La Figura 10.9 mostra due esempi di curve ellittiche. Come si può vedere, la formula produce a volte curve dall'aspetto un po' particolare.

Ora si consideri l'insieme E(a, b) costituito da tutti i punti (x, y) che soddisfano l'Equazione 10.1 insieme all'elemento *0*. Utilizzando un valore differente della coppia (a, b) si ottiene un insieme differente E(a, b). Impiegando questa terminologia, le due curve della Figura 10.9 rappresentano rispettivamente gli insiemi E$(-1, 0)$ ed E$(1, 1)$.

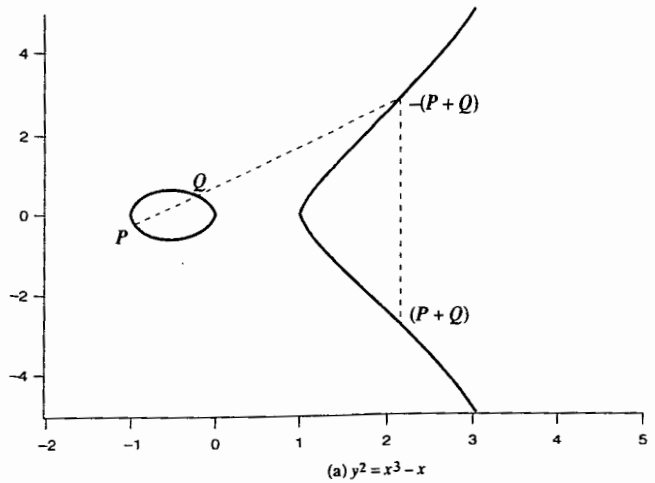
Descrizione geometrica della somma

Si può dimostrare che un gruppo può essere definito sulla base dell'insieme E(a, b) per valori specifici di a e b nell'equazione 10.1, posto che:

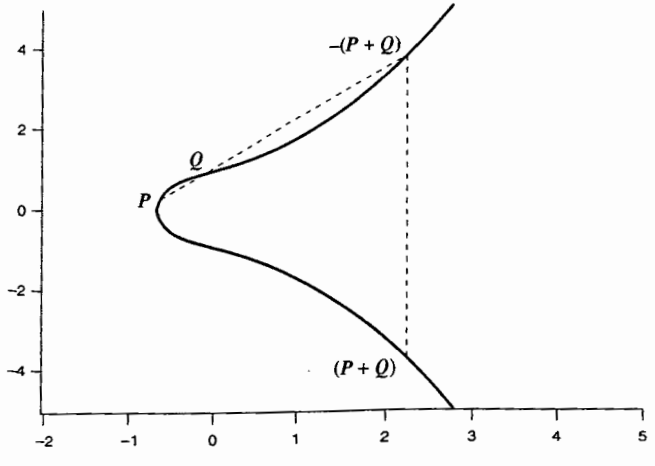
$$4a^3 + 27b^2 \neq 0 \tag{10.2}$$

Per definire il gruppo si deve definire un'operazione, chiamata somma e rappresentata dal simbolo +, per l'insieme E(a, b) dove a e b soddisfano l'Equazione 10.2. In termini geometrici, le regole della somma possono essere formulate nel seguente modo: se tre punti di

[3] Si noti che x e y sono vere variabili che assumono dei valori. Questo in contrasto con la discussione sugli anelli polinomiali e i campi presentata nel Capitolo 4 dove x veniva trattata come indeterminata.



(a) $y^2 = x^3 - x$



(b) $y^2 = x^3 + x + 1$

Figura 10.9 Esempi di curve ellittiche.

una curva ellittica sono disposti su una linea retta, la loro somma è zero. Da questa definizione si possono definire le regole della somma su una curva ellittica.

1. O funge da identità additiva. Pertanto $O = -O$; per ogni punto P della curva ellittica, $P + O = P$. In ciò che segue, si suppone che $P \neq O$ e $Q \neq O$.

2. Il negativo di un punto P è il punto con la stessa coordinata x ma con la coordinata y negativa; ovvero, se $P = (x, y)$, allora $-P = (x, -y)$. Si noti che questi due punti possono essere uniti da una linea verticale. Si noti che $P + (-P) = P - P = O$.
3. Per sommare P e Q con coordinate x differenti, tracciare una linea retta fra di esse e trovare il terzo punto di intersezione R. È facile vedere che vi è un unico punto di intersezione R (a meno che la linea sia tangente alla curva in P o in Q, nel qual caso si assumerà $R = P$ o $R = Q$ rispettivamente). Per formare una struttura di gruppo, occorre definire la somma su questi tre punti nel modo seguente: $P + Q = -R$. Ovvero si definisce $P + Q$ come l'immagine speculare (rispetto all'asse x) del terzo punto di intersezione. Questa costruzione è rappresentata nella Figura 10.9.
4. L'interpretazione geometrica dell'elemento precedente si applica anche a due punti P e $-P$ con la stessa coordinata x. I punti sono connessi da una linea verticale che può essere considerata come se intersecasse la curva a un punto infinito. Pertanto si avrà $P + (-P) = O$, coerente con l'elemento 2.
5. Per raddoppiare un punto Q si traccia la linea tangente e si trova l'altro punto di intersezione S. Allora $Q + Q = 2Q = -S$.

Con l'insieme di regole precedente, si può dimostrare che l'insieme E(a, b) è un gruppo abeliano.

Descrizione algebrica della somma

In questa parte del capitolo verranno presentati alcuni risultati che consentono di calcolare le somme sulle curve ellittiche.[4] Per due punti distinti $P = (x_P, y_P)$ e $Q = (x_Q, y_Q)$ che non siano l'uno il negativo dell'altro, la pendenza della retta l che li unisce è $\Delta = (y_Q - y_P)/(x_Q - x_P)$. Esiste esattamente un altro punto dove l interseca la curva ellittica e questo è il negativo della somma di P e Q. Dopo alcune manipolazioni algebriche, si può esprimere la somma $R = P + Q$ nel seguente modo:

$$\begin{aligned} x_R &= \Delta^2 - x_P - x_Q \\ y_R &= -y_P + \Delta(x_P - x_R) \end{aligned} \tag{10.3}$$

Occorre anche essere in grado di sommare un punto a se stesso: $P + P = 2P = R$. Per $y_P \neq 0$, le espressioni sono:

$$\begin{aligned} x_R &= \left(\frac{3x_P^2 + a}{2y_P}\right)^2 - 2x_P \\ y_R &= \left(\frac{3x_P^2 + a}{2y_P}\right)(x_P - x_R) - y_P \end{aligned} \tag{10.4}$$

[4] Per le derivazioni di questi risultati, vedere [KOBL94] o altre descrizioni matematiche delle curve ellittiche.

Curve ellittiche su Z_p

La crittografia a curva ellittica utilizza curve ellittiche le cui variabili e i cui coefficienti sono ristretti agli elementi di un campo finito. Nelle applicazioni crittografiche vengono utilizzate due famiglie di curve ellittiche: *curve prime* su Z_p e *curve binarie* su GF(2^n). Come **curva prima** su Z_p si utilizza un'equazione cubica nella quale le variabili e i coefficienti assumono valori nell'insieme di interi da 0 a $p - 1$ e i calcoli vengono effettuati modulo p. Le variabili e i coefficienti della **curva binaria** definita su $GF(2^m)$ assumono valori in $GF(2^n)$ e i calcoli sono effettuati in $GF(2^n)$. [FERN99] evidenzia che le curve prime sono le migliori per le applicazioni software in quanto non richiedono le numerose operazioni sui bit necessarie nel caso delle curve binarie; al contrario le curve binarie sono migliori per le applicazioni hardware dove bastano pochi elementi logici per creare un sistema crittografico veloce e potente. Queste due famiglie verranno esaminate in questo e nel prossimo paragrafo.

Non esiste alcuna interpretazione geometrica ovvia dell'aritmetica delle curve ellittiche sui campi finiti. L'interpretazione algebrica utilizzata per l'aritmetica delle curve ellittiche sui numeri reali è invece direttamente trasferibile e dunque verrà adottato questo approccio.

Per le curve ellittiche su Z_p, come nel caso dei numeri reali, ci si limita alle equazioni della forma riportata nell'Equazione 10.1 ma in questo caso con coefficienti e variabili limitati a Z_p:

$$y^2 \bmod p = (x^3 + ax + b) \bmod p \tag{10.5}$$

Per esempio, l'Equazione 10.5 è soddisfatta per $a = 1$, $b = 1$, $x = 9$, $y = 7$, $p = 23$.

$$7^2 \bmod 23 = (9^3 + 9 + 1) \bmod 23$$
$$49 \bmod 23 = 739 \bmod 23$$
$$3 = 3$$

Ora si consideri l'insieme $E_p(a, b)$ costituito da tutte le coppie di interi (x, y) che soddisfano l'Equazione 10.5, insieme a un punto all'infinito O. I coefficienti a e b e le variabili x e y sono tutti elementi di Z_p.

Per esempio, si supponga $p = 23$ e si consideri la curva ellittica $y^2 = x^3 + x + 1$. In questo caso, $a = b = 1$. Si noti che questa equazione è la stessa rappresentata nella Figura 10.9B. La figura mostra una curva continua con tutti i punti reali che soddisfano l'equazione. Per l'insieme $E_{23}(1, 1)$ si è interessati solo agli interi non negativi nel quadrante da (0, 0) a ($p - 1$, $p - 1$) che soddisfano l'equazione modulo p. La Tabella 10.1 elenca questi punti (ad eccezione di O) che fanno parte di $E_{23}(1, 1)$. La Figura 10.10 traccia i punti di $E_{23}(1, 1)$; si noti che i punti, con una eccezione, sono simmetrici rispetto a $y = 11{,}5$.

Si può dimostrare che può essere definito un gruppo finito abeliano basato sull'insieme $E_p(a, b)$ sempre che $(x^3 + ax + b) \bmod p$ non contenga fattori ripetuti. Ciò è equivalente alla condizione:

$$(4a^3 + 27b^2) \bmod p \neq 0 \bmod p \tag{10.6}$$

Si noti che l'Equazione 10.6 ha la stessa forma dell'Equazione 10.2.

Tabella 10.1 I punti sulla curva ellittica $E_{23}(1, 1)$.

(0, 1)	(6, 4)	(12, 19)
(0, 22)	(6, 19)	(13, 7)
(1, 7)	(7, 11)	(13, 16)
(1, 16)	(7, 12)	(17, 3)
(3, 10)	(9, 7)	(17, 20)
(3, 13)	(9, 16)	(18, 3)
(4, 0)	(11, 3)	(18, 20)
(5, 4)	(11, 20)	(19, 5)
(5, 19)	(12, 4)	(19, 18)

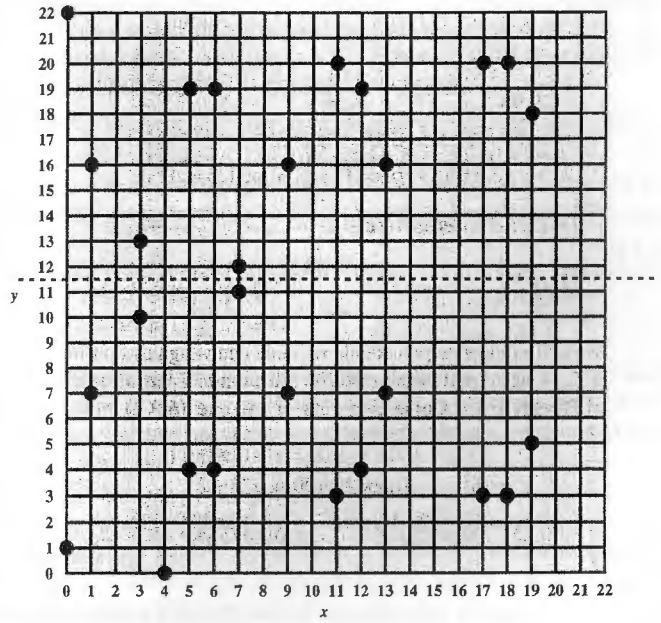


Figura 10.10 La curva ellittica $E_{23}(1, 1)$.

Le regole della somma su $E_p(a, b)$ corrispondono alla tecnica algebrica descritta per le curve ellittiche definite sui numeri reali. Per tutti i punti $P, Q \in E_p(a, b)$.

1. $P + O = P$.
2. Se $P = (x_P, y_P)$, allora $P + (x_P, -y_P) = O$. Il punto $(x_P, -y_P)$ è il negativo di P, rappresentato come $-P$. Per esempio, in $E_{23}(1, 1)$, per $P = (13, 7)$ si avrà $-P = (13, -7)$. Ma $-7 \bmod 23 = 16$. Pertanto $-P = (13, 16)$, anch'esso in $E_{23}(1, 1)$.
3. Se $P = (x_P, y_P)$ e $Q = (x_Q, y_Q)$ con $P \neq -Q$, allora $R = P + Q = (x_R, y_R)$ è determinato dalle seguenti regole:

$$x_R = (\lambda^2 - x_P - x_Q) \bmod p$$
$$y_R = (\lambda(x_P - x_R) - y_P) \bmod p$$

dove:

$$\lambda = \begin{cases} \left(\dfrac{y_Q - y_P}{x_Q - x_P}\right) \bmod p & \text{se } P \neq Q \\ \left(\dfrac{3x_P^2 + a}{2y_P}\right) \bmod p & \text{se } P = Q \end{cases}$$

4. La moltiplicazione è definita come una ripetizione di somme; per esempio $4P = P + P + P + P$.

Per esempio, sia $P = (3, 10)$ e $Q = (9, 7)$ in $E_{23}(1, 1)$. Allora:

$$\lambda = \left(\frac{7-10}{9-3}\right) \bmod 23 = \left(\frac{-3}{6}\right) \bmod 23 = \left(\frac{-1}{2}\right) \bmod 23 = 11$$
$$x_R = (11^2 - 3 - 9) \bmod 23 = 109 \bmod 23 = 17$$
$$y_R = (11(3 - 17) - 10) \bmod 23 = -164 \bmod 23 = 20$$

Pertanto $P + Q = (17, 20)$. Per trovare $2P$:

$$\lambda = \left(\frac{3(3^2)+1}{2\times 10}\right) \bmod 23 = \left(\frac{5}{20}\right) \bmod 23 = \left(\frac{1}{4}\right) \bmod 23 = 6$$

L'ultimo passo dell'equazione precedente richiede che venga preso l'inverso moltiplicativo di 4 in Z_{23}. Questo può essere ottenuto utilizzando l'algoritmo di Euclide esteso introdotto nel Paragrafo 4.4. Come conferma, si noti che $(6 \times 4) \bmod 23 = 24 \bmod 23 = 1$.

$$x_R = (6^2 - 3 - 3) \bmod 23 = 30 \bmod 23 = 7$$
$$y_R = (6(3 - 7) - 10) \bmod 23 = (-34) \bmod 23 = 12$$

e $2P = (7, 12)$.

Per determinare la sicurezza delle varie cifrature a curva ellittica, è interessante conoscere il numero di punti di un gruppo finito abeliano definito su una curva ellittica. Nel caso del gruppo finito $E_p(a, b)$, il numero di punti N è limitato da:

$$p + 1 - 2\sqrt{p} \leq N \leq p + 1 + 2\sqrt{p}$$

Si noti che per p di grandi dimensioni, il numero di punti in $E_p(a, b)$ è uguale approssimativamente al numero di elementi di Z_p, ovvero p elementi.

Curve ellittiche su GF(2^m)

Come si è detto nel Capitolo 4, un campo finito GF(2^m) è costituito da 2^m elementi insieme alle operazioni di somma e moltiplicazione che possono essere definite sui polinomi. Per le curve ellittiche su GF(2^m), si usa un'equazione cubica in cui le variabili e i coefficienti assumono i valori contenuti in GF(2^m) per qualche numero m, e in cui i calcoli vengono eseguiti utilizzando le regole dell'aritmetica in GF(2^m).

Ne consegue che la forma dell'equazione cubica appropriata per le applicazioni crittografiche a curva ellittica è leggermente differente per GF(2^m) rispetto a Z_p.

La forma è:

$$y^2 + xy = x^3 + ax^2 + b \tag{10.7}$$

dove le variabili x e y e i coefficienti a e b sono elementi di GF(2^m) e i calcoli vengono eseguiti in GF(2^m).

Ora si consideri l'insieme $E_{2^m}(a, b)$ costituito da tutte le coppie di interi (x, y) che soddisfano l'Equazione 10.7 insieme a un punto all'infinito O. Si utilizzi, per esempio, il campo GF(2^4) con il polinomio irriducibile $f(x) = x^4 + x + 1$. Questo produce un generatore che soddisfa $f(g) = 0$ per il valore $g^4 = g + 1$ o, in binario, 0010. Si possono sviluppare le potenze di g come segue:

g^0=0001	g^4=0011	g^8=0101	g^{12}=1111
g^1=0010	g^5=0110	g^9=1010	g^{13}=1101
g^2=0100	g^6=1100	g^{10}=0111	g^{14}=1001
g^3=1000	g^7=1011	g^{11}=1110	g^{15}=0001

Per esempio $g^5 = (g^4)g = g^2 + g = 0110$.

Si consideri ora la curva ellittica $y^2 + xy = x^3 + g^4x^2 + 1$. In questo caso $a = g^4$ e $b = g^0 = 1$. Un punto che soddisfa l'equazione è

$$(g^3)^2 + (g^5)(g^3) = (g^5)^3 + (g^4)(g^5)^2 + 1$$
$$g^6 + g^8 = g^{15} + g^{14} + 1$$
$$1100 + 0101 = 0001 + 1001 + 0001$$
$$1001 = 1001$$

La Tabella 10.2 elenca i punti (oltre a O) che fanno parte di $E_{24}(g^4, 1)$. La Figura 10.11 riporta i punti di $E_{24}(g^4, 1)$.

Si può dimostrare che può essere definito un gruppo abeliano finito sulla base dell'insieme $E_{2^m}(a, b)$ sempre che $b \neq 0$. Le regole della somma possono essere stabilite nel seguente modo. Per tutti i punti $P, Q \in E_{2^m}(a, b)$:

Tabella 10.2 Punti sulla curva ellittica $E_{24}(g^4, 1)$.

(0,1)	(g^5,g^3)	(g^9,g^{13})
$(1,g^6)$	(g^3,g^8)	(g^{10},g)
$(1,g^{13})$	(g^6,g^8)	(g^{10},g^8)
(g^3,g^8)	(g^6,g^{14})	$(g^{12},0)$
(g^3,g^{13})	(g^9,g^{10})	(g^{12},g^{12})

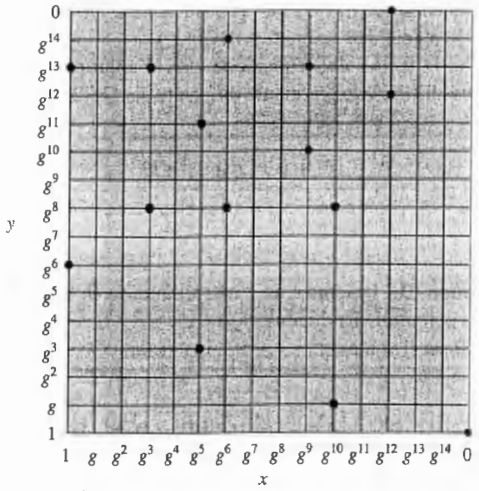


Figura 10.11 La curva ellittica $E_{24}(g^4, 1)$.

1. $P + O = P$.
2. Se $P = (x_P, y_P)$, allora $P + (x_P, x_P + y_P) = O$. Il punto $(x_P, x_P + y_P)$ è il negativo di P, rappresentato come $-P$.
3. Se $P = (x_P, y_P)$ e $Q = (x_Q, y_Q)$ con $P \neq -Q$ e $P \neq Q$, allora $R = P + Q = (x_R, y_R)$ è determinato dalle seguenti regole:

$$x_R = \lambda^2 + \lambda + x_P + x_Q + a$$
$$y_R = \lambda(x_P + x_R) + x_R + y_P$$

dove:

$$\lambda = \frac{y_Q + y_P}{x_Q + x_P}$$

4. Se $P = (x_P, y_P)$, allora $R = 2P = (x_R, y_R)$ è determinato dalle seguenti regole:

$$x_R = \lambda^2 + \lambda + a$$
$$y_R = x_P^2 + (\lambda + 1)x_R$$

dove:

$$\lambda = x_P + \frac{y_P}{x_P}$$

10.4 Crittografia a curva ellittica

L'operazione di somma nella crittografia a curva ellittica è l'analogo della moltiplicazione modulare in RSA e la somma multipla è analoga all'esponenziazione modulare. Per creare un sistema crittografico utilizzando le curve ellittiche occorre trovare un "problema difficile" corrispondente alla fattorializzazione del prodotto di due numeri primi o del calcolo del logaritmo discreto.

Si consideri l'equazione $Q = kP$ dove $Q, P \in E_p(a, b)$ e $k < p$. È relativamente facile calcolare Q dati k e P mentre è relativamente difficile determinare k dati Q e P. Questo viene chiamato il problema del logaritmo discreto per le curve ellittiche.

Si fornirà un esempio tratto dal sito Web Certicom (`www.certicom.com`). Si consideri il gruppo $E_{23}(9, 17)$. Questo è il gruppo definito dall'equazione $y^2 \bmod 23 = (x^3 + 9x + 17) \bmod 23$. Qual è il logaritmo discreto k di $Q = (4, 5)$ in base $P = (16, 5)$? Il metodo a forza bruta prevede di calcolare i multipli di P fino a trovare Q. Pertanto:

$$P = (16, 5); 2P = (20, 20); 3P = (14, 14); 4P = (19, 20); 5P = (13, 10);$$
$$6P = (7, 3); 7P = (8, 7); 8P = (12, 17); 9P = (4, 5)$$

Poiché $9P = (4, 5) = Q$, il logaritmo discreto $Q = (4, 5)$ della base $P = (16, 5)$ è $k = 9$. In un'applicazione vera e propria, k sarebbe così esteso da rendere impossibile l'approccio a forza bruta.

Di seguito si parlerà di due approcci alla crittografia a curva ellittica che sfruttano questa tecnica.

L'analogo dello scambio di chiavi Diffie-Hellman

Lo scambio di chiavi tramite curve ellittiche può essere eseguito nel seguente modo. Innanzitutto si deve scegliere un intero Q grande che sia un numero primo p o un intero nella forma 2^m e i parametri a e b per la curva ellittica dell'Equazione 10.5 o 10.7. Questa definisce il gruppo ellittico di punti $E_q(a, b)$. Poi si sceglie un *punto base* $G = (x_1, y_1)$ in $E_p(a, b)$ il cui ordine è un valore n molto esteso. L'**ordine** n di un punto G su una curva ellittica è il più piccolo intero positivo n tale che $nG = O$. $E_q(a, b)$ e G sono parametri del sistema crittografico noti a tutti i partecipanti.

Uno scambio di chiavi fra gli utenti A e B può essere ottenuto nel seguente modo (vedere la Figura 10.12).

1. A sceglie un intero n_A minore di n. Questa è la chiave privata di A. Quindi A genera una chiave pubblica $P_A = n_A \times G$; la chiave pubblica è un punto in $E_q(a, b)$.
2. Analogamente anche B sceglie una chiave privata n_B e calcola la chiave pubblica P_B.
3. A genera la chiave segreta $K = n_A \times P_B$. B genera la chiave segreta $K = n_B \times P_A$.

I due calcoli del passo 3 producono lo stesso risultato poiché:

$$n_A \times P_B = n_A \times (n_B \times G) = n_B \times (n_A \times G) = n_B \times P_A$$

Per violare questo schema, si dovrebbe riuscire a calcolare k dati G e kG, un problema molto difficile.

Come esempio[5], si prenda $p = 211$, $E_p(0, -4)$, (equivalente alla curva $y^2 = x^3 - 4$), e $G = (2, 2)$. Si può calcolare che $240G = O$. La chiave privata di A è $n_A = 121$ e dunque la chiave pubblica di A è $P_A = 121(2, 2) = (115,48)$. La chiave privata di B è $n_B = 203$ e dunque la chiave pubblica di B è $203(2, 2) = (130, 203)$. La chiave segreta condivisa è $121(130, 203) = 203(115, 48) = (161, 69)$.

Si noti che la chiave segreta è costituita da una coppia di numeri. Se questa chiave viene utilizzata come chiave di sessione per la crittografia convenzionale, deve essere generato un unico numero. Si possono semplicemente utilizzare le coordinate x o una semplice funzione della coordinata x.

Crittografia/decrittografia a curva ellittica

In letteratura sono stati analizzati vari approcci alla crittografia/decrittografia tramite curve ellittiche. In questa parte del capitolo si parlerà della forma forse più semplice. Il primo compito in questo sistema è quello di codificare il messaggio in chiaro m in modo da inviarlo come un punto P_m dato da $x–y$. Sarà il punto P_m ad essere crittografato come testo cifrato e successivamente decrittografato. Si noti che non è possibile codificare semplicemente il messaggio come le coordinate x o y di un punto poiché non tutte queste coordinate sono in $E_q(a, b)$; per esempio, si consulti la Tabella 10.1. Esistono vari approcci a questa codifica che non possono essere esaminati in questa sede; basti sapere che si possono utilizzare tecniche relativamente semplici.

Come si è detto per il sistema per lo scambio di chiavi, un sistema di crittografia/decrittografia richiede come parametri un punto G e un gruppo ellittico $E_q(a, b)$. Ciascun utente A seleziona una chiave privata n_A e genera una chiave pubblica $P_A = n_A \times G$.

Per crittografare e trasmettere un messaggio P_m a B, l'utente A sceglie un intero positivo casuale k e produce il testo cifrato C_m costituito dalla coppia di punti:

$$C_m = \{kG, P_m + kP_B\}$$

[5] Fornito da Ed Schaefer della Santa Clara University.

Elementi globali pubblici

$E_q(a, b)$	curva ellittica con i parametri a, b e q, dove q è un numero primo o un intero nella forma 2^m
G	punto sulla curva ellittica il cui ordine è un valore esteso n

Generazione della chiave da parte dell'utente A

Seleziona il valore privato n_A	$n_A < n$
Calcola il valore pubblico P_A	$P_A = n_A \times G$

Generazione della chiave da parte dell'utente B

Seleziona il valore privato n_B	$n_B < n$
Calcola il valore pubblico P_B	$P_B = n_B \times G$

Generazione della chiave segreta da parte dell'utente A

$K = n_A \times P_B$

Generazione della chiave segreta da parte dell'utente B

$K = n_B \times P_A$

Figura 10.12 Lo scambio della chiave ECC Diffie-Hellman.

Si noti che A ha utilizzato la chiave pubblica di B, P_B. Per decrittografare il testo cifrato, B moltiplica il primo punto della coppia per la propria chiave segreta e sottrae il risultato dal secondo punto:

$$P_m + kP_B - n_B(kG) = P_m + k(n_B G) - n_B(kG) = P_m$$

A ha mascherato il messaggio P_m sommandogli kP_B. Nessuno, tranne A, conosce il valore di k e dunque anche se P_B è una chiave pubblica, nessuno può rimuovere la maschera kP_B. Tuttavia, A ha anche introdotto un "indizio" che è sufficiente per rimuovere la maschera conoscendo la chiave privata n_B. Perché un estraneo possa recuperare il messaggio, dovrebbe calcolare k sulla base di G e kG, un problema che si presuppone essere difficile.

Come esempio del processo di crittografia, tratto da [KOBL94]), si prendano $p = 751$, $E_p(–1, 188)$, che è equivalente alla curva $y^2 = x^3 – x + 188$ e $G = (0, 376)$. Si supponga che A voglia inviare a B un messaggio codificato nel punto ellittico $P_m = (562, 201)$ e che A

selezioni il numero casuale $k = 386$. La chiave pubblica di B è $P_B = (201, 5)$. Si avrà quindi $386(0, 376) = (676, 558)$ e $(562, 201) + 386(201, 5) = (385, 328)$. Pertanto A trasmette il testo cifrato $\{(676, 558), (385, 328)\}$.

Sicurezza della crittografia a curva ellittica

La sicurezza della crittografia a curva ellittica dipende dalla difficoltà con cui è possibile determinare k dati kP e P. Questo è il cosiddetto problema del logaritmo della curva ellittica. La tecnica nota più veloce per calcolare il logaritmo della curva ellittica è chiamata metodo Pollard rho. La Tabella 10.3 confronta vari algoritmi riportando le dimensioni delle chiavi a parità di sforzo computazionale dell'analisi crittografica. Come si può vedere, si può usare una chiave di dimensioni molto inferiori rispetto a RSA. Inoltre, per chiavi di uguale lunghezza, l'impegno computazionale richiesto per la crittografia a curva ellittica e per RSA è confrontabile [JURI97]. Pertanto esiste un vantaggio computazionale nell'utilizzo della crittografia a curva ellittica con una chiave di lunghezza inferiore rispetto a una crittografia RSA di sicurezza analoga.

Tabella 10.3 Dimensioni delle chiavi a parità di sforzo computazionale dell'analisi crittografica.

Schema simmetrico (dimensioni della chiave in bit)	Schema basato su ECC (dimensione di n in bit)	RSA/DSA (dimensione del modulo in bit)
56	112	512
80	160	1024
112	224	2048
128	256	3072
192	384	7680
256	512	15360

Fonte: Certicom.

10.5 Letture e siti Web consigliati

Una trattazione piuttosto leggibile della crittografia a curva ellittica si trova in [ROSI99]; il testo pone l'accento soprattutto sull'implementazione software. [HANK04] è un altro volume leggibile ma rigoroso. Due altri buoni testi, che però contengono elementi matematici piuttosto complessi, sono [BLAK99] e [ENGE99]. Esistono anche altre buone descrizioni più compatte in [KUMA98], [STIN02] e [KOBL94]. Due interessanti indagini sull'argomento sono [FERN99] e [JURI97].

BLAK99 I. Blake, G. Seroussi e N. Smart. *Elliptic Curves in Cryptography*. Cambridge: Cambridge University Press, 1999.

ENGE99 A. Enge. *Elliptic Curves and Their Applications to Cryptography*. Norwell, MA: Kluwer Academic Publishers, 1999.

FERN99 A. Fernandes. "Elliptic Curve Cryptography". *Dr. Dobb's Journal*, Dicembre 1999.

HANK04 D. Hankerson, A. Menezes e S. Vanstone. *Guide to Elliptic Curve Cryptography*. New York: Sprinter, 2004.

JURI97 A. Jurisic e A. Menezes. "Elliptic Curves and Cryptography". *Dr. Dobb's Journal*, Aprile 1997.

KOBL94 N. Koblitz. *A Course in Number Theory and Cryptography*. New York: Springer-Verlag, 1994.

KUMA98 R. Kumanduri e C. Romero. *Number Theory with Computer Applications*. Upper Saddle River, NJ: Prentice Hall, 1998.

ROSI99 M. Rosing. *Implementing Elliptic Curve Cryptography*. Greeenwich, CT: Manning Publications, 1999.

STIN02 D. Stinson. *Cryptography: Theory and Practice*. Boca Raton, FL: CRC Press, 2002.

Sito Web consigliato

- **Certicom:** un'ampia raccolta di materiale tecnico sulla crittografia a curva ellittica e altri argomenti della crittografia.

10.6 Termini chiave, domande di ripasso e problemi

Termini chiave

Aritmetica a curva ellittica
Attacco man-in-the-middle
Campo finito
Certificato di chiave pubblica
Crittografia a curva ellittica
Curva ellittica
Curva prima
Distribuzione della chiave
Elenco di chiavi pubbliche
Equazione cubica
Gestione della chiave
Gruppo abeliano
Logaritmo discreto
Punto zero
Radice primitiva
Scambio di chiavi Diffie-Hellman

Domande di ripasso

10.1 Quali sono i due diversi utilizzi della crittografia a chiave pubblica relativi alla distribuzione della chiave?

10.2 Elencare quattro schemi generali per la distribuzione delle chiavi pubbliche.

10.3 Quali sono gli elementi fondamentali di un elenco di chiavi pubbliche?
10.4 Che cos'è un certificato a chiave pubblica?
10.5 Quali sono i requisiti per l'utilizzo di uno schema con certificati a chiave pubblica?
10.6 Descrivere brevemente lo scambio di chiavi Diffie-Hellman.
10.7 Che cos'è una curva ellittica?
10.8 Che cos'è il punto zero di una curva ellittica?
10.9 Qual è la somma di tre punti di una curva ellittica posizionati su una linea retta?

Problemi

10.1 Gli utenti A e B usano la tecnica di scambio di chiavi Diffie-Hellman con un numero primo comune $q = 71$ e una radice primitiva $\alpha = 7$.
   A. Se l'utente A ha la chiave privata $X_A = 5$, qual è la chiave pubblica di A, Y_A?
   B. Se l'utente B ha la chiave privata $X_B = 12$, qual è la chiave pubblica di B, Y_B?
   C. Qual è la chiave segreta condivisa?

10.2 Si consideri uno schema di Diffie-Hellman con un numero primo comune $q = 11$ e una radice primitiva $\alpha = 2$.
   A. Mostrare che 2 è una radice primitiva di 11.
   B. Se l'utente A ha la chiave pubblica $Y_A = 9$, qual è la chiave privata di A, X_A?
   C. Se l'utente B ha la chiave pubblica $Y_B = 3$, qual è la chiave segreta K, condivisa con A?

10.3 Nel protocollo Diffie-Hellman, ciascun partecipante sceglie un numero segreto x e invia agli altri partecipanti $\alpha^x \bmod q$ per qualche numero pubblico α. Cosa avverrebbe se i partecipanti si inviassero l'un l'altro x^α ? Fornire almeno un metodo che Alice e Bob potrebbero utilizzare per accordarsi su una chiave. Potrebbe Eva introdursi in tale sistema senza scoprire i numeri segreti? Potrebbe Eva scoprire i numeri segreti?

10.4 Questo problema illustra il fatto che il protocollo Diffie-Hellman non è sicuro senza il passo in cui si calcola il modulo, ovvero che il "Problema dei logaritmi indiscreti" non è un problema difficile! Tu sei Eva e hai catturato e imprigionato Alice e Bob. Ascolti il dialogo seguente:

Bob: Oh, non preoccupiamoci del numero primo nel protocollo Diffie-Hellman, sarà più semplice.

Alice: Va bene, ma dobbiamo comunque avere una base *á* da usare come potenza. Cosa ne dici di $g = 3$?

Bob: Perfetto, il mio risultato è dunque 27.

Alice: E il mio è 243.

Quali sono i valori segreti di Bob (X_B) e di Alice (X_A)? Qual è la loro chiave segreta combinata? Illustrare il procedimento.

10.5 Il Paragrafo 10.2 descrive un attacco man-in-the-middle al protocollo di scambio della chiave Diffie-Hellman, nel quale l'avversario genera due coppie di chiavi pubblica/privata per l'attacco. Sarebbe possibile effettuare lo stesso attacco con una sola coppia di chiavi? Spiegare.

10.6 Nel 1985, T. ElGamal annunciò uno schema a chiave pubblica che si basava sui logaritmi discreti e in stretta relazione con la tecnica Diffie-Hellman. Come Diffie-Hellman, gli elementi globali dello schema di ElGamal sono un numero primo q e α, una radice primitiva di q. Un utente A seleziona una chiave privata X_A e calcola una chiave pubblica Y_A come in Diffie-Hellman. L'utente A esegue la crittografia del testo in chiaro $M < q$ da inviare a B nel seguente modo.
   1. Sceglie un intero casuale k tale che $1 \le k \le q - 1$.
   2. Calcola $K = (Y_B)^k \pmod q$.
   3. Esegue la crittografia di M come una coppia di interi (C_1, C_2) dove:

$$C_1 = \alpha^k \bmod q \quad C_2 = KM \bmod q$$

   L'utente B ottiene il testo in chiaro nel seguente modo.
   1. Calcola $K = (C^1)^{X_B} \bmod q$
   2. Calcola $M = (C_2 K^{-1}) \bmod q$.

   Dimostrare che questo sistema funziona, ovvero mostrare che il processo di decrittografia risale al testo in chiaro.

10.7 Considerare uno schema ElGamal con un numero primo comune $q = 71$ e una radice primitiva $\alpha = 7$.
   A. Se B ha la chiave pubblica $Y_B = 3$ e A sceglie un intero casuale $k = 2$, quale sarà il testo cifrato di $M = 30$?
   B. Se A sceglie un altro valore di k in modo che la codifica di $M = 30$ sia $C = (59, C_2)$, quale sarà l'intero C_2?

10.8 La regola (5) dell'aritmetica nelle curve ellittiche sui numeri reali stabilisce che per raddoppiare un punto Q si traccia la tangente e si trova l'altro punto di intersezione S. Pertanto $Q + Q = 2Q = -S$. Se la linea tangente non è verticale, vi sarà un solo punto di intersezione. Ma si supponga che la linea tangente sia verticale. In tal caso quale sarà il valore $2Q$? Quale sarà il valore $3Q$?

10.9 Dimostrare che entrambe le curve ellittiche della Figura 10.9 soddisfano le condizioni di un gruppo sui numeri reali.

10.10 Il punto (4,7) appartiene alla curva ellittica $y^2 = x^3 - 5x + 5$ su numeri reali?

10.11 Nella curva ellittica sui numeri reali $y^2 = x^3 - 36x$, siano $P = (-3, 9)$ e $Q = (-2, 8)$. Trovare $P + Q$ e $2P$.

10.12 La curva ellittica di equazione $y^2 = x^3 + 10x + 5$ definisce un gruppo su Z_{17}?

10.13 Considerare la curva ellittica $E_{11}(1, 6)$; ovvero la curva definita da $y^2 = x^3 + x + 6$ con un modulo $p = 11$. Determinare tutti i punti di $E_{11}(1, 6)$. *Suggerimento:* iniziare calcolando il lato destro dell'equazione per tutti i valori di x.

10.14 Calcolare gli opposti dei seguenti punti su curva ellittica su Z_{17}: $P = (5, 8)$, $Q = (3, 0)$, $R = (0, 6)$.

10.15 Per $E_{11}(1, 6)$ considerare il punto $G = (2, 7)$. Calcolare i multipli di G da $2G$ a $13G$.

10.16 Questo problema svolge la crittografia/decrittografia a curva ellittica utilizzando lo schema introdotto nel Paragrafo 10.4. I parametri del sistema crittografico sono $E_{11}(1, 6)$ e $G = (2, 7)$. La chiave segreta di B è $n_B = 7$
   A. Trovare la chiave pubblica di B, P_B.

B. A vuole crittografare il messaggio $P_m = (10, 9)$ e sceglie il valore casuale $k = 3$. Determinare il testo cifrato C_m.
C. Mostrare il calcolo mediante il quale B risale a P_m a partire da C_m.

10.17 Si presenta un primo tentativo di firma elettronica basato su curve ellittiche. Si ha una curva ellittica globale, un numero primo p e un "generatore" G. Alice sceglie una chiave di firma privata X_A e crea la chiave pubblica di verifica $Y_A = X_A G$. Per firmare un messaggio M:

- Alice sceglie un valore k.
- Alice invia a Bob M, k e la firma $S = M - kX_A G$.
- Bob verifica che $M = S + kY_A$.

1. Dimostrare che questo schema funziona "correttamente". Ovvero, dimostrare che il processo di verifica produce un'uguaglianza quando la firma è valida.
2. Dimostrare che lo schema è inaccettabile descrivendo una semplice tecnica per creare la firma falsa di un utente su un qualsiasi messaggio.

10.18 Si presenta una versione migliorata dello schema proposto nel problema 10.17. Come nel caso precedente si ha una curva ellittica globale, un numero primo p e un "generatore" G. Alice sceglie una chiave di firma privata X_A e crea la chiave pubblica di verifica $Y_A = X_A G$. Per firmare un messaggio M:

- Bob sceglie un valore k.
- Bob invia ad Alice $C_1 = kG$.
- Alice invia a Bob M e la firma $S = M - X_A C_1$.
- Bob verifica che $M = S + kY_A$.

1. Dimostrare che questo schema funziona "correttamente". Ovvero, dimostrare che il processo di verifica produce un'uguaglianza quando la firma è valida.
2. Dimostrare che falsificare un messaggio con questo schema è difficile quanto violare la crittografia a curva ellittica ElGamal.
3. Questo schema presenta un passaggio ulteriore rispetto agli schemi di crittografia e firma digitale esaminati nel volume. Che svantaggi comporta?

Capitolo 11

Autenticazione dei messaggi e funzioni hash

Concetti essenziali

- L'**autenticazione dei messaggi** è un meccanismo o un servizio utilizzato per garantire l'integrità dei messaggi. L'autenticazione garantisce che i dati ricevuti siano identici ai dati inviati (ovvero che non vi siano apportate alterazioni, inserzioni, cancellazioni o repliche) e che l'identità del mittente indicato sia corretta.
- La crittografia simmetrica fornisce l'autenticazione fra coloro che condividono la chiave segreta. Anche la crittografia di un messaggio con la chiave privata del mittente fornisce una forma di autenticazione.
- Le due tecniche crittografiche più diffuse per l'autenticazione dei messaggi sono i codici MAC (Message Authentication Code - codici di autenticazione dei messaggi) e le funzioni hash sicure.
- Un algoritmo **MAC** richiede l'uso di una chiave segreta; esso riceve in ingresso un messaggio di lunghezza variabile e una chiave segreta per produrre il codice di autenticazione corrispondente. Il destinatario in possesso della chiave segreta può a sua volta generare il codice di autenticazione per verificare l'integrità del messaggio.
- Una funzione **hash** mappa un messaggio di lunghezza variabile in un valore hash di lunghezza fissa, chiamato anche **digest** del messaggio. Per l'autenticazione di un messaggio è necessario un metodo per combinare una funzione hash sicura con una chiave segreta.

L'area forse più complessa nel campo della sicurezza delle reti è quella dell'autenticazione dei messaggi, con l'argomento correlato delle firme digitali. Gli attacchi e le contromisure divengono così complessi che coloro che operano in questo campo ricordano un po' gli astronomi di tanto tempo fa che costruivano epicicli di epicicli nel tentativo di tenere conto di tutte le contingenze. Fortunatamente sembra che oggi i progettisti di protocolli crittografici, a differenza degli astronomi di un tempo, stiano utilizzando un modello fondamentalmente corretto.

Sarebbe impossibile, senza scrivere un intero volume, descrivere tutte le funzioni e i protocolli crittografici che sono stati proposti o implementati per l'autenticazione dei messaggi e le firme digitali. Ma lo scopo di questo capitolo e dei prossimi due è quello di fornire una panoramica generale sull'argomento e sviluppare una base sistematica per descrivere i vari approcci.

Questo capitolo si apre introducendo i requisiti dell'autenticazione e della firma digitale e i vari tipi di attacchi che devono essere affrontati. Poi esamina i principali approcci, fra cui l'importante area delle funzioni hash sicure. Il Capitolo 12 esaminerà invece alcune funzioni hash specifiche.

11.1 I requisiti per l'autenticazione

Nel contesto delle comunicazioni di rete, possono essere identificati i seguenti attacchi.

1. **Violazione:** il rilascio di contenuti a persone o processi che non possiedono la chiave crittografica appropriata.
2. **Analisi del traffico:** l'individuazione di schemi di traffico fra le parti. In un'applicazione orientata alla connessione, potrebbe essere possibile determinare la frequenza e la durata delle connessioni. Sia negli ambienti orientati alla connessione che in quelli non orientati alla connessione, potrebbe essere possibile determinare il numero e la lunghezza dei messaggi scambiati dalle parti.
3. **Mascheramento:** l'inserimento in rete di messaggi provenienti da una sorgente fasulla. Per esempio può trattarsi della creazione di messaggi da parte di un estraneo che sostiene di essere un'entità autorizzata. Può anche trattarsi di un acknowledgement fraudolento della ricezione di un messaggio o della mancata ricezione da parte di una persona differente dal destinatario del messaggio.
4. **Modifica dei contenuti:** l'alterazione dei contenuti di un messaggio fra cui l'inserimento, la cancellazione, la trasposizione e la modifica.
5. **Modifica della sequenza:** qualsiasi modifica a una sequenza di messaggi fra le parti comprendente l'inserimento, la cancellazione o il riordino.
6. **Modifica temporale:** il ritardo o la ripetizione dei messaggi. In un'applicazione orientata alla connessione, potrebbe essere possibile replicare un'intera sessione o una sequenza di messaggi provenienti da una sessione valida precedente o i singoli messaggi della sequenza potrebbero essere riproposti o ritardati. In un'applicazione senza connessione, potrebbe essere possibile ritardare o riprodurre un singolo messaggio (per esempio un datagram).
7. **Ripudiazione del mittente:** il mittente nega di aver trasmesso un messaggio.
8. **Ripudiazione del destinatario:** il destinatario nega di aver ricevuto il messaggio.

Le misure da adottare per i primi due attacchi riguardano la segretezza del messaggio e sono già state trattate nella Parte prima. Le misure invece da adottare per risolvere i problemi da 3 a 6 sono solitamente considerate relative all'autenticazione dei messaggi. I meccanismi da adottare per l'elemento 7 riguardano la firma digitale. In generale, la tecnica della firma digitale può anche rispondere ad alcuni o a tutti gli attacchi elencati nei punti da 3 a

6. Per risolvere il problema del punto 8 occorre impiegare una combinazione di firma digitale e protocolli specifici per questo tipo di attacco.

In breve, l'autenticazione dei messaggi è una procedura che verifica che i messaggi ricevuti provengano dalla sorgente indicata e che non siano stati modificati. L'autenticazione dei messaggi può anche verificare la corretta sequenza e temporizzazione. La firma digitale è una tecnica di autenticazione che include misure per impedire al mittente di negare di aver trasmesso un messaggio.

11.2 Le funzioni di autenticazione

Ogni meccanismo di autenticazione dei messaggi o di firma digitale può essere considerato come se fosse costituito fondamentalmente da due livelli. Al livello inferiore vi deve essere una funzione che produce un autenticatore, vale a dire un valore che verrà utilizzato per autenticare un messaggio. Questa funzione di basso livello viene poi utilizzata come primitiva in un protocollo di autenticazione di alto livello che consente al destinatario di verificare l'autenticità del messaggio.

Questa parte del capitolo riguarda le funzioni che possono essere utilizzate per produrre un autenticatore. Tali funzioni possono essere raggruppate in tre classi.

- **Crittografia dei messaggi:** come autenticatore viene utilizzata la crittografia dell'intero messaggio.
- **Codice MAC (Message Authentication Code):** una funzione del messaggio e una chiave segreta producono un valore di lunghezza fissa che funge da autenticatore.
- **Funzione hash:** una funzione mappa un messaggio di lunghezza arbitraria in un valore hash di lunghezza fissa che funge da autenticatore.

Si esamineranno ora brevemente ciascuno degli argomenti elencati. I codici MAC e le funzioni hash verranno analizzati più approfonditamente nei Paragrafi 11.3 e 11.4.

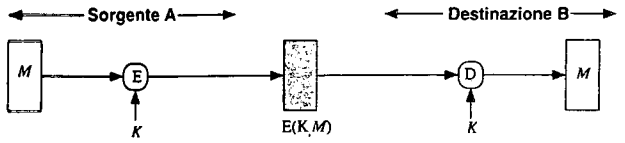
La crittografia dei messaggi

La stessa crittografia dei messaggi può servire come una misura di autenticazione. L'analisi varia per gli schemi simmetrici e a chiave pubblica.

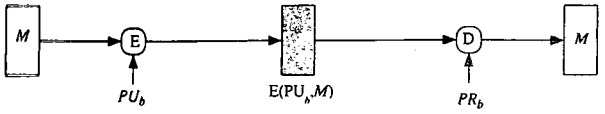
Crittografia simmetrica

Si consideri l'utilizzo ordinario della crittografia simmetrica (Figura 11.1A). Un messaggio M trasmesso dalla sorgente A alla destinazione B viene crittografato utilizzando una chiave segreta K condivisa da A e B. Nessun altro conosce la chiave e quindi la segretezza è garantita: nessun altro potrà risalire al messaggio in chiaro.

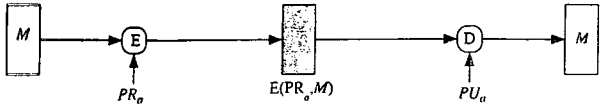
Inoltre, B è sicuro che il messaggio sia stato generato da A: il messaggio deve necessariamente provenire da A che è l'unico altro utente che possiede K e pertanto è l'unico in possesso delle informazioni necessarie per costruire il testo cifrato che può essere decrittografato con K. Inoltre, se B può recuperare il messaggio M, saprà anche che nessuno dei bit di M è stato alterato in quanto un estraneo che non conoscesse K non saprebbe



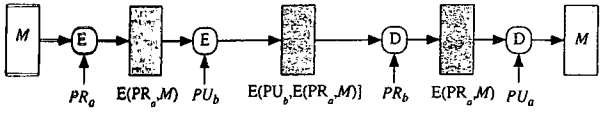
(A) Crittografia simmetrica: segretezza e autenticazione



(B) Crittografia a chiave pubblica: segretezza



(C) Crittografia a chiave pubblica: autenticazione e firma



(D) Crittografia a chiave pubblica: segretezza, autenticazione e firma

Figura 11.1 Semplici impieghi della crittografia dei messaggi.

come modificare i bit del testo cifrato per produrre le modifiche desiderate nel testo in chiaro.

Pertanto la crittografia simmetrica garantisce l'autenticazione e la segretezza. Tuttavia questa semplice affermazione deve essere qualificata un po' meglio. Si consideri esattamente ciò che accade in B. Data una funzione di decrittografia D e una chiave segreta K, la destinazione accetterà *qualsiasi* input X e produrrà l'output $Y = D(K, X)$. Se X è il testo cifrato di un messaggio legittimo M prodotto dalla funzione di crittografia, allora Y è un messaggio in chiaro M. Altrimenti, Y sarebbe una sequenza di bit senza significato. Vi può essere la necessità un mezzo automatico per determinare in B se Y è un testo in chiaro legittimo e pertanto se proviene da A.

Le implicazioni di questo ragionamento sono profonde dal punto di vista dell'autenticazione. Si supponga che il messaggio M possa essere una sequenza arbitraria di bit. In tal caso non vi sarà alcun modo per determinare alla destinazione se il messaggio in arrivo è

testo cifrato di un messaggio legittimo. La conclusione è incontrovertibile: se M può essere una sequenza qualsiasi di bit, allora, indipendentemente dal valore di X, $Y = D(K, X)$ è una determinata sequenza di bit e pertanto deve essere accettata come testo in chiaro.

Pertanto, in generale, si richiede che possa essere considerato solo un piccolo sottoinsieme di tutte le possibili sequenze di bit come testo in chiaro legittimo. In tal caso, un testo cifrato errato difficilmente produrrà un testo in chiaro legittimo. Per esempio, si supponga che venga considerata legittima una sola sequenza di bit su 10^6. Allora la probabilità che una sequenza casuale di bit trattata come testo cifrato produca un messaggio in chiaro legittimo è solo 10^{-6}.

Per molte applicazioni e molti schemi di crittografia, ovviamente prevalgono le condizioni desiderate. Per esempio, si supponga di trasmettere un messaggio in lingua inglese utilizzando la cifratura di Cesare con uno scorrimento pari a 1 ($K = 1$). A invia il seguente testo cifrato legittimo:

```
nbsftfbupbutboeepftfbupbutboemjuumfmbnctfbujwz
```

B esegue la decrittografia per produrre il seguente testo in chiaro:

```
mareseatoatsanddoeseatoatsandlittlelambseativy
```

Una semplice analisi della frequenza conferma che questo messaggio ha il profilo della lingua inglese. Al contrario, se un estraneo generasse la seguente sequenza di lettere casuale:

```
zuvrsoevgqxlzwigamdvnmhpmccxiuureosfbcebtqxsxq
```

verrà prodotto il testo in chiaro:

```
ytuqrndufpwkyvhfzlcumlgolbbwhttqdnreabdaspwrwp
```

che non rientra nel profilo della normale lingua inglese.

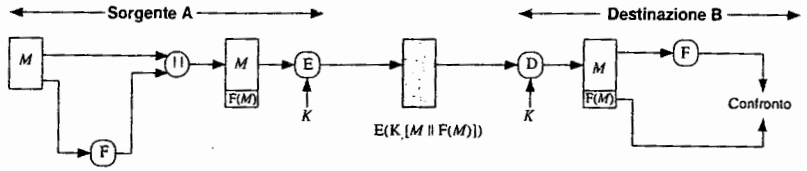
Può essere difficile determinare *automaticamente* se il testo cifrato in arrivo produce testo in chiaro intelligibile. Se il testo in chiaro è, per esempio, costituito da un oggetto binario o da una digitalizzazione, può essere difficile determinare se il testo in chiaro prodotto sia corretto e pertanto autentico. Un estraneo potrebbe quindi riuscire a produrre un determinato livello di disturbo semplicemente emettendo messaggi con contenuti casuali e sostenendo che provengono da un utente legittimo.

Una soluzione a questo problema consiste nel costringere il testo in chiaro a rispettare una determinata struttura che possa essere facilmente riconosciuta ma che non possa essere replicata senza ricorrere alla funzione di crittografia. Per esempio, prima della crittografia, si potrebbe aggiungere a ciascun messaggio un codice di rilevamento degli errori FCS (Frame Check Sequence) o checksum (vedere la Figura 11.2A). A prepara un messaggio in chiaro M e lo invia come input alla funzione F che produce il codice FCS. Questo codice viene poi aggiunto a M e quindi viene crittografato l'intero blocco. Alla destinazione, B

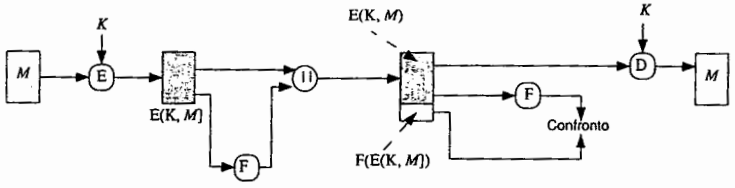
esegue la decrittografia del blocco costituito dal messaggio con l'aggiunta del codice FCS. Quindi B applica la stessa funzione F per riprodurre il codice FCS. Se il codice FCS calcolato è uguale a quello ricevuto, allora il messaggio è autentico. È molto improbabile che una sequenza di bit casuali esibisca questa relazione.

Si noti che l'ordine in cui vengono svolte le funzioni FCS di crittografia è fondamentale. La sequenza rappresentata nella Figura 11.2A è chiamata [DIFF79] *controllo interno degli errori* per distinguerla dal *controllo esterno degli errori* (vedere la Figura 11.2B). Con il controllo interno degli errori, l'autenticazione è garantita poiché un estraneo non riuscirebbe a generare il testo cifrato che, decrittografato, presentasse un codice di controllo valido. Se il codice FCS fosse esterno, un estraneo potrebbe costruire messaggi che producono codici di controllo degli errori validi. Sebbene un estraneo non possa risalire al testo in chiaro decrittografato, potrà comunque cercare di creare confusione.

Un codice di controllo degli errori è semplicemente un esempio: infatti qualsiasi forma di strutturazione aggiunta al messaggio trasmesso è utile per rafforzare le capacità di autenticazione. Tale struttura è fornita dall'uso di un'architettura di comunicazione costituita da protocolli a livelli. Per esempio si consideri la struttura dei messaggi trasmessi utilizzando l'architettura TCP/IP. La Figura 11.3 mostra il formato di un segmento TCP, illustrandone l'intestazione. Ora si supponga che ciascuna coppia di host condivida una chiave segreta univoca e che pertanto tutti gli scambi che si verificano fra una coppia di host utilizzino la stessa chiave indipendentemente dall'applicazione. Allora si potrebbe semplicemente crittografare tutto il segmento ad eccezione dell'intestazione IP (vedere la Figura 7.5). Anche in questo caso, se un estraneo dovesse sostituire una qualsiasi sequenza di bit del segmento TCP crittografato, il messaggio in chiaro risultante non includerebbe un'intestazione che avesse significato. In questo caso, l'intestazione include non solo un checksum (che copre l'intestazione) ma anche altre informazioni utili, per esempio il numero di sequenza. Poiché i segmenti TCP di una



(A) Controllo degli errori interno



(B) Controllo degli errori esterno

Figura 11.2 Controllo degli errori interno ed esterno.

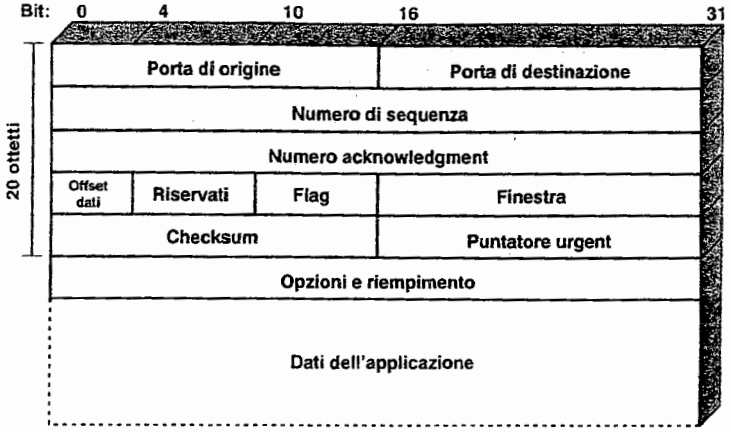


Figura 11.3 Un segmento TCP.

connessione sono numerati sequenzialmente, la crittografia garantisce che nessuno possa ritardare, cambiare l'ordine o cancellare un segmento.

Crittografia a chiave pubblica

L'uso ordinario della crittografia a chiave pubblica (Figura 11.1B) garantisce la segretezza ma non l'autenticazione. Per crittografare M, la sorgente (A) usa la chiave pubblica PU_b della destinazione (B). Poiché solo B ha la chiave privata corrispondente, PR_b, solo B potrà decrittografare il messaggio. Questo schema non garantisce l'autenticazione poiché chiunque potrebbe usare la chiave pubblica di B per crittografare il messaggio, sostenendo di essere l'utente A.

Per garantire l'autenticazione, A esegue la crittografia del messaggio con la propria chiave privata e B esegue la decrittografia con la chiave pubblica di A (Figura 11.1C). Questo garantisce l'autenticazione utilizzando lo stesso tipo di ragionamento impiegato nel caso della crittografia simmetrica: il messaggio deve necessariamente provenire da A poiché A è l'unico che possiede PR_a e pertanto è l'unico dotato delle informazioni necessarie per costruire il testo cifrato che può essere decrittografato con PU_a. Si applica anche lo stesso ragionamento precedente: il testo in chiaro deve avere una struttura interna tale che il ricevitore possa distinguerlo da una sequenza di bit casuali.

Supponendo che questa struttura interna esista, allora lo schema della Figura 11.1C garantisce l'autenticazione. Fornisce anche una firma digitale.[1] Solo A può aver costruito il testo cifrato poiché solo A possiede PR_a. Nemmeno B, il destinatario, avrebbe potuto costruire il testo cifrato. Pertanto, se B è in possesso del testo cifrato, B può dimostrare che il messaggio proviene da A. In pratica, A ha "firmato" il messaggio utilizzando per la crittografia la propria chiave privata.

[1] Come si vedrà, le firme digitali usano lo stesso principio ma vengono costruite in un altro modo.

Si noti che questo schema non garantisce però la segretezza. Chiunque sia in possesso della chiave pubblica di A può decrittografare il testo cifrato.

Per fornire sia la segretezza che l'autenticazione, A può crittografare M utilizzando innanzitutto la propria chiave privata per garantire la firma digitale e poi la chiave pubblica di B che garantisce la segretezza (Figura 11.1D). Lo svantaggio di questo approccio è il fatto che l'algoritmo a chiave pubblica, che è complesso, deve essere applicato quattro volte invece di due.

La Tabella 11.1 riepiloga le implicazioni di segretezza e autenticazione dei vari approcci alla crittografia dei messaggi.

Il codice MAC (Message Authentication Code)

Una tecnica di autenticazione alternativa prevede l'uso di una chiave segreta per generare un piccolo blocco di dati di dimensioni fisse chiamato checksum crittografico o codice MAC che viene poi aggiunto al messaggio. Questa tecnica presuppone che le due parti, A e B, condividano una chiave segreta comune K. Quando A deve inviare un messaggio a B, calcola il codice MAC in funzione del messaggio e della chiave: MAC = $C(K, M)$, dove:

M = messaggio di input
C = funzione MAC
K = chiave segreta condivisa
MAC = codice MAC

Il messaggio più il codice MAC vengono trasmessi al destinatario. Il destinatario svolge sul messaggio ricevuto lo stesso calcolo utilizzando la stessa chiave segreta e genera il proprio codice MAC. Il codice MAC ricevuto viene confrontato con quello calcolato (Figura 11.4A). Se si è certi che solo il destinatario e il mittente conoscano la chiave segreta e se il codice MAC di destinazione ricevuto corrisponde a quello calcolato, allora valgono i seguenti fatti.

1. Il destinatario è sicuro che il messaggio non è stato alterato. Se un estraneo dovesse alterare il messaggio senza modificare il MAC, il codice MAC calcolato dal destinatario risulterebbe differente da quello ricevuto. Poiché l'estraneo non può conoscere la chiave segreta, non potrà alterare il codice MAC sulla base delle alterazioni apportate al messaggio.
2. Il destinatario è sicuro che il messaggio proviene dal mittente indicato. Poiché nessun altro conosce la chiave segreta, nessun altro potrebbe preparare un messaggio con il codice MAC corretto.
3. Se il messaggio include un numero di sequenza (come quello utilizzato con HDLC, X.25 e TCP) allora il destinatario è sicuro che la sequenza è corretta poiché un estraneo non potrebbe modificare con successo il numero sequenziale.

Una funzione MAC è simile alla crittografia. Una differenza consiste nel fatto che l'algoritmo MAC non deve essere reversibile, come avviene nel caso della decrittografia. In generale, la funzione MAC è molti-a-uno. Il dominio della funzione è costituito dai messaggi di lunghezza arbitraria mentre il codominio è costituito da tutti i codici MAC possibili e dalle relative chiavi. Se viene utilizzato un codice MAC di n bit, allora vi saranno 2^n possibili codici MAC mentre vi sono N possibili messaggi con $N >> 2^n$. Inoltre, con una chiave di k bit, vi saranno 2^k chiavi possibili.

Per esempio, si supponga di utilizzare messaggi da 100 bit e un codice MAC da 10 bit. Vi sarà un totale di 2^{100} messaggi ma solo 2^{10} codici MAC differenti. Pertanto, in media, ciascun valore MAC verrà generato da un totale di $2^{100}/2^{10} = 2^{90}$ messaggi differenti.

Se viene utilizzata una chiave da 5 bit, allora vi saranno $2^5 = 32$ mapping differenti dall'insieme dei messaggi all'insieme dei valori MAC.

A causa delle proprietà matematiche della funzione di autenticazione, questa è meno vulnerabile alla violazione rispetto alla crittografia.

Tabella 11.1 Segretezza e autenticazione nella crittografia dei messaggi (Figura 11.1).

$A \rightarrow B$: $E(K, M)$

- Garantisce la segretezza.
  - Solo A e B condividono K.
- Garantisce un certo grado di autenticazione.
  - Può provenire solo da A.
  - Non è stato alterato durante il transito.
  - Richiede una certa strutturazione/ridondanza.
- Non garantisce la firma.
  - Il destinatario può modificare il messaggio.
  - Il mittente può negare di aver inviato il messaggio.

(A) Crittografia simmetrica.

$A \rightarrow B$: $E(PU_b, M)$

- Garantisce la segretezza.
  - Solo B ha PR_b per eseguire la decrittografia.
- Non fornisce autenticazione.
  - Chiunque può utilizzare PU_b per crittografare un messaggio e sostenere di essere A.

(B) Crittografia a chiave pubblica (asimmetrica).

$A \rightarrow B$: $E(PR_a, M)$

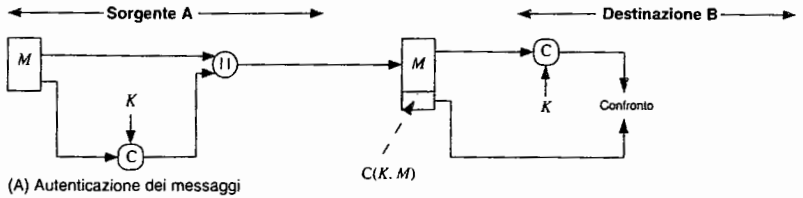
- Garantisce l'autenticazione e la firma.
  - Solo A ha PR_a per crittografare.
  - Non è stato alterato durante il transito.
  - Richiede una certa strutturazione/ridondanza.
  - Chiunque può utilizzare PU_a per verificare la firma.

(C) Crittografia a chiave pubblica: autenticazione e firma.

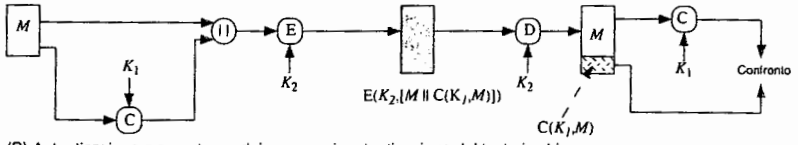
$A \rightarrow B$: $E(PU_b, E(PR_a, M))$

- Garantisce la segretezza grazie a PU_b.
- Garantisce l'autenticazione e la firma grazie a PR_a.

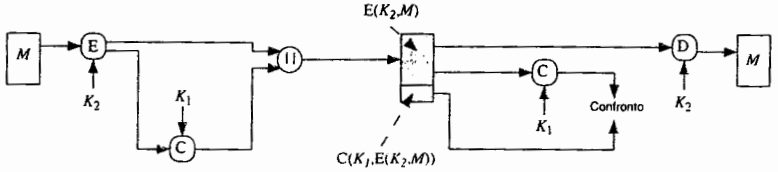
(D) Crittografia a chiave pubblica: segretezza, autenticazione e firma.



(A) Autenticazione dei messaggi



(B) Autenticazione e segretezza dei messaggi; autenticazione del testo in chiaro



(C) Autenticazione e segretezza dei messaggi; autenticazione del testo cifrato

Figura 11.4 Utilizzi del codice MAC (Message Authentication Code).

Il processo rappresentato nella Figura 11.4A garantisce l'autenticazione ma non la segretezza poiché il messaggio viene trasmesso interamente in chiaro. La segretezza può essere garantita svolgendo la crittografia del messaggio dopo (Figura 11.4B) o prima (Figura 11.4C) dell'algoritmo MAC. In entrambi i casi, sono necessarie due diverse chiavi condivise dal mittente e dal destinatario. Nel primo caso il codice MAC viene calcolato usando il messaggio come input e poi viene concatenato al messaggio. L'intero blocco viene poi crittografato. Nel secondo caso il messaggio viene innanzitutto crittografato, poi viene calcolato il codice MAC utilizzando il testo cifrato risultante. Il codice MAC viene concatenato al testo cifrato per costruire il blocco trasmesso. In genere è preferibile collegare l'autenticazione direttamente al testo in chiaro e dunque viene utilizzato il metodo rappresentato nella Figura 11.4B.

Poiché la crittografia simmetrica garantisce l'autenticazione e poiché è ampiamente utilizzata da molti prodotti commerciali, perché non utilizzare semplicemente questo metodo invece di un codice distinto per l'autenticazione dei messaggi come MAC? [DAVI89] suggerisce tre situazioni in cui è preferibile utilizzare il codice MAC.

1. Vi sono varie applicazioni che prevedono che lo stesso messaggio venga trasmesso in broadcast a più destinazioni. Gli esempi sono la notifica agli utenti che la rete non è disponibile o un segnale d'allarme in un centro di controllo militare. È più economico e affidabile avere un'unica destinazione responsabile del controllo dell'autenticità. Pertanto il messaggio deve essere trasmesso in broadcast come testo in chiaro con il codice MAC associato. Il sistema responsabile ha la chiave segreta e svolge l'autenticazione. Se si verifica una violazione, gli altri sistemi di destinazione saranno avvertiti tramite un allarme generale.
2. Un'altra situazione possibile è rappresentata da uno scambio di dati in cui un lato ha un notevole carico e non può permettersi di decrittografare tutti i messaggi in arrivo. L'autenticazione viene svolta in modo selettivo e il controllo viene effettuato su messaggi scelti casualmente.
3. L'autenticazione di un programma come testo in chiaro è un servizio interessante. Il programma può essere eseguito senza doverlo decrittografare ogni volta, sprecando pertanto risorse del microprocessore. Tuttavia, se al programma venisse associato il codice di autenticazione MAC, questo potrebbe essere controllato ogni volta che fosse necessario verificare l'integrità del programma.

Possono essere aggiunti altri tre elementi.

4. Per alcune applicazioni, può non essere importante mantenere segreti i messaggi ma piuttosto garantire la loro autenticazione. Un esempio è costituito dal protocollo SNMPv3 (Simple Network Management Protocol Version 3) che separa le funzioni di segretezza e autenticazione. Per questa applicazione è normalmente importante che un sistema autentichi i messaggi SNMP in arrivo, in particolare se il messaggio contiene un comando che altera i parametri del sistema. D'altra parte può non essere necessario rendere segreto il traffico SNMP.
5. La separazione delle funzioni di autenticazione e segretezza aumenta la flessibilità dell'architettura. Per esempio, può essere utile eseguire l'autenticazione a livello dell'applicazione e garantire la segretezza a un livello inferiore, per esempio al livello di trasporto.
6. Un utente può voler prolungare il periodo di protezione oltre il breve tempo della ricezione e consentire ciononostante l'elaborazione del contenuto del messaggio. Con la crittografia del messaggio, la protezione viene persa quando il messaggio viene decrittografato e dunque il messaggio risulta protetto contro le modifiche fraudolente solo durante il transito ma non quando si trova nel sistema di destinazione.

Infine si deve notare che il codice MAC non fornisce la firma digitale poiché il mittente e il destinatario condividono la stessa chiave.

La Tabella 11.2 riepiloga le implicazioni in termini di segretezza e autenticazione degli approcci illustrati nella Figura 11.4.

Tabella 11.2. Utilizzi di base del codice MAC C (Figura 11.4).

$A \rightarrow B$: $M \| C(K, M)$
• Garantisce l'autenticazione. – Solo A e B condividono K.

(A) Autenticazione del messaggio.

(segue)

Tabella 11.2. Utilizzi di base del codice MAC C (Figura 11.4). *(continua)*

A → B: $E(K_2, (M \| C(K_1, M)))$ • Garantisce l'autenticazione. – Solo A e B condividono K_1. • Garantisce la segretezza. – Solo A e B condividono K_2.
(B) Autenticazione e segretezza del messaggio: autenticazione del testo in chiaro.
A → B: $E(K_2, M) \| C(K_1, E(K_2, M))$ • Garantisce l'autenticazione. – Utilizzando K_1 • Garantisce la segretezza. – Utilizzando K_2
(C) Autenticazione e segretezza del messaggio: autenticazione del testo cifrato.

La funzione hash

Una variante del codice di autenticazione del messaggio MAC è la funzione hash monodirezionale. Come il codice MAC, anche la funzione hash accetta come input un messaggio M di lunghezza variabile e produce un output di lunghezza fissa chiamato **codice hash** H(M). A differenza del codice MAC, il codice hash non utilizza una chiave ma è funzione solo del messaggio di input. Il codice hash è anche chiamato **message digest** o valore hash. Il codice hash è una funzione che considera tutti i bit del messaggio e fornisce una funzionalità di rilevamento degli errori: una variazione in uno o più bit del messaggio produce una variazione nel codice hash.

La Figura 11.5 illustra vari modi in cui si può utilizzare un codice hash per garantire l'autenticazione del messaggio.

A. Il messaggio e il codice hash concatenati vengono crittografati utilizzando la crittografia simmetrica. Questa procedura presenta una struttura identica alla strategia di controllo degli errori interna rappresentata nella Figura 11.2A. Si applica lo stesso tipo di ragionamento: poiché solo A e B condividono la chiave segreta, il messaggio deve provenire da A e non è stato modificato. Il codice hash fornisce la ridondanza necessaria per ottenere l'autenticazione. Poiché la crittografia viene applicata all'intero messaggio e al codice hash, viene garantita anche la segretezza.

B. Viene crittografato solo il codice hash utilizzando la crittografia simmetrica. Questo riduce il carico di elaborazione per quelle applicazioni che non richiedono la segretezza. Si noti che la combinazione di hash e crittografia produce una funzione che, complessivamente, genera un codice MAC (Figura 11.4A). Ovvero E(K, H(M)) è funzione di un messaggio M di lunghezza variabile e di una chiave segreta K e produce un output di lunghezza fissa che è sicuro contro chiunque non conosca la chiave segreta.

C. Viene crittografato solo il codice hash utilizzando la crittografia a chiave pubblica con la chiave privata del mittente. Come nel caso precedente (B) viene assicurata l'autenti-

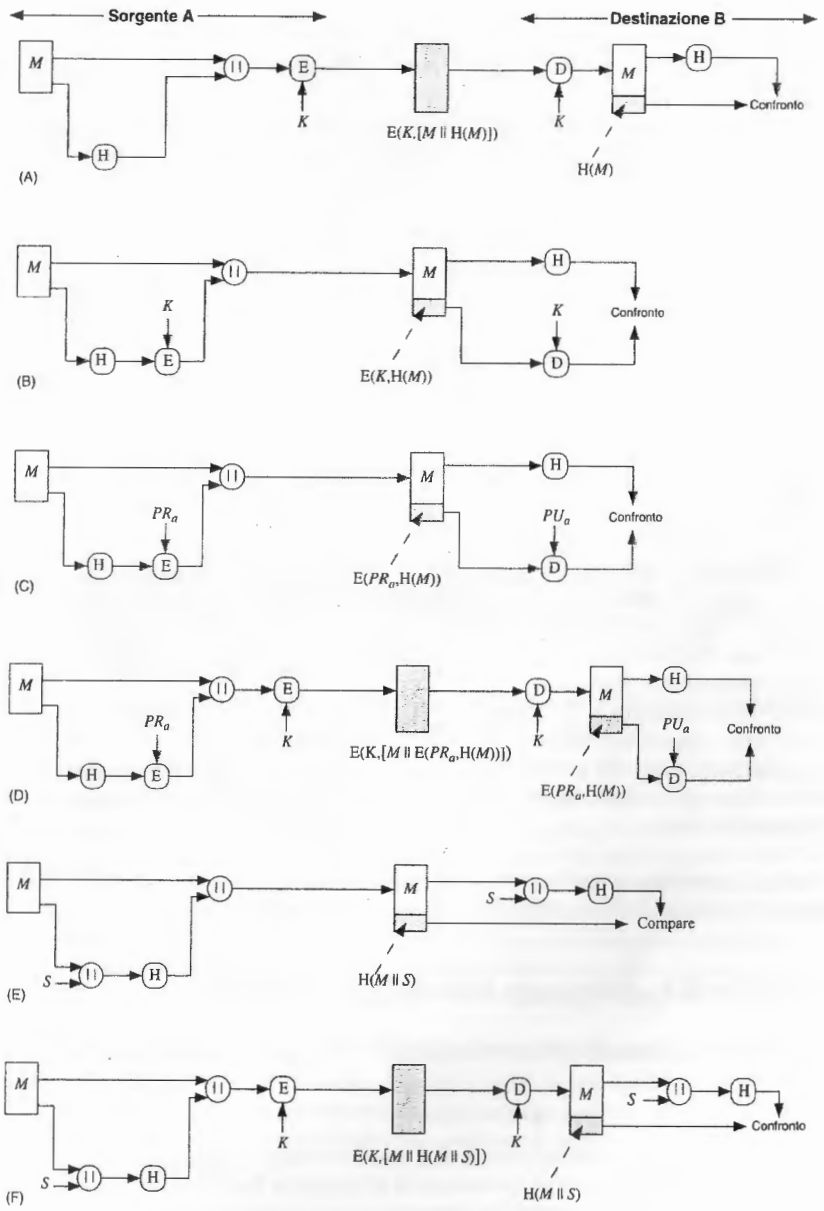


Figura 11.5 Utilizzi base di una funzione hash.

cazione. Inoltre viene garantita la firma digitale poiché solo il mittente può avere prodotto il codice hash crittografato. In effetti si tratta essenzialmente della tecnica a firma digitale.

D. Se si richiede la segretezza oltre che la firma digitale, il messaggio e il codice hash crittografato con la chiave pubblica possono essere crittografati utilizzando una chiave segreta simmetrica. Questa è una tecnica molto utilizzata.

E. È possibile utilizzare una funzione hash senza crittografia per l'autenticazione del messaggio. La tecnica presuppone che le due parti condividano un valore segreto comune S. A calcola il valore hash sulla concatenazione di M e S e aggiunge al messaggio M il valore hash risultante. Poiché B possiede S, potrà ricalcolare il valore hash per verificarlo. Poiché il valore segreto non viene inviato, nessuno potrà modificare un messaggio intercettato e generare un falso messaggio.

F. Si può aggiungere la segretezza all'approccio E crittografando l'intero messaggio e il codice hash.

Quando non è richiesta la segretezza, i metodi A e B presentano il vantaggio rispetto a quelli che eseguono la crittografia dell'intero messaggio, di richiedere un minor impegno computazionale. Ciononostante, vi è un crescente interesse nelle tecniche che evitano la crittografia (Figura 11.5E). I motivi di questo interesse sono evidenziati in [TSUD92].

- La crittografia software è piuttosto lenta. Anche se il volume di dati da crittografare per ogni messaggio è limitato, in un sistema vi può essere un notevole flusso di messaggi.
- I costi dell'hardware di crittografia non sono trascurabili. Esistono implementazioni a basso costo di DES ma i costi si sommano nel caso in cui tutti i nodi di una rete debbano essere dotati di questa funzionalità.
- L'hardware di crittografia è ottimizzato per grandi volumi di dati. Per piccoli blocchi di dati si spreca troppo tempo nella fase di inizializzazione/chiamata.
- Gli algoritmi di crittografia possono essere coperti da licenze d'uso. Per esempio, prima che il brevetto scadesse, era necessario utilizzare RSA su licenza, incrementando ulteriormente i costi.

La Tabella 11.3 riepiloga le implicazioni di segretezza e autenticazione degli approcci illustrati nella Figura 11.5. Ora si parlerà più in dettaglio dei codici MAC e hash.

11.3 I codici MAC (Message Authentication Code)

Un codice MAC, chiamato anche checksum crittografico, viene generato da una funzione C della forma:

$$MAC = C(K, M)$$

dove M è un messaggio di lunghezza variabile, K è una chiave segreta condivisa solo dal mittente e dal destinatario e $C(K, M)$ è l'autenticatore di lunghezza fissa. Il codice MAC viene aggiunto al messaggio alla sorgente nel momento in cui il messaggio è sicuramente corretto. Il destinatario autentica il messaggio ricalcolando il codice MAC.

Tabella 11.3 Utilizzi della funzione hash H (Figura 11.5).

A → B: $E(K, (M \| H(M)))$

- Garantisce la segretezza.
  - Solo A e B condividono K.
- Garantisce l'autenticazione.
  - $H(M)$ è protetta crittagraficamente.

(A) Crittografia del messaggio e del codice hash.

A → B: $M \| E(K, H(M))$

- Garantisce l'autenticozione.
  - $H(M)$ è protetta crittograficamente.

(B) Crittografia del codice hash; chiave segreta condivisa.

A → B: $M \| E(PR_a, H(M))$

- Garantisce l'autenticaziane e la firma digitale.
  - $H(M)$ è protetta crittograficamente.
  - Solo A può creare $E(PR_a, H(M))$.

(C) Crittografia del codice hash; chiave privata del mittente.

A → B: $E(K, (M \| E(PR_a, H(M))))$

- Garantisce l'autenticazione e la firma digitale.
- Garantisce lo segretezza
  - Solo A e B condividono K.

(D) Crittografia dei risultati di (C); chiave segreta condiviso.

A → B: $M \| H(M \| S)$

- Garantisce l'autenticaziane.
  - Solo A e B condividono S.

(E) Calcolo del codice hash del messaggio più il valore segreto.

A → B: $E(K, (M \| H(M) \| S))$

- Garantisce l'autenticazione.
  - Solo A e B condividono S.
- Garantisce lo segretezza
  - Solo A e B condividono K.

(F) Crittografia dei risultati di (E).

In questa parte del capitolo si vedranno i requisiti della funzione C e si esaminerà un esempio specifico. Un altro esempio verrà trattato nel Capitolo 12.

Requisiti per i codici MAC

Quando per ottenere la segretezza viene crittografato l'intero messaggio, utilizzando la crittografia simmetrica o asimmetrica, la sicurezza dello schema dipende generalmente dalla lunghezza in bit della chiave. Non trovando punti deboli nell'algoritmo, un estraneo dovrebbe ricorrere a un attacco a forza bruta utilizzando tutte le chiavi possibili. In media tale attacco richiederà $2^{(k-1)}$ tentativi per una chiave di k bit. In particolare, per un attacco a testo cifrato, l'estraneo, avendo a disposizione solo il testo cifrato C, dovrebbe eseguire $P_i = D(K_i, C)$ per tutti i valori K_i possibili fino a produrre un valore P_i corrispondente a un testo in chiaro accettabile.

Nel caso di un codice MAC, le considerazioni sono completamente differenti. In generale, la funzione MAC è una funzione molti-a-uno a causa della sua stessa natura. Utilizzando il metodo a forza bruta, come potrebbe un estraneo tentare di scoprire la chiave? Se non viene impiegata la segretezza, l'estraneo avrà accesso ai messaggi in chiaro e ai relativi codici MAC. Si supponga che $k > n$, ovvero si supponga che la chiave abbia dimensioni maggiori rispetto al codice MAC. Allora, noti M_1 e MAC_1, con $MAC_1 = C(K_1, M_1)$, l'analista crittografico può eseguire $MAC_i = C(K_i, M_1)$ per tutti i valori possibili della chiave K_i. Almeno una chiave produrrà una corrispondenza $MAC_i = MAC_1$. Si noti che verranno prodotti 2^k codici MAC ma esistono solo $2^n < 2^k$ codici MAC differenti. Pertanto più chiavi possono produrre il codice MAC corretto e l'estraneo non può in alcun modo sapere qual è la chiave corretta. In media la corrispondenza verrà trovata su $2^k/2^n = 2^{(k-n)}$ chiavi. L'estraneo deve quindi iterare l'attacco.

- **Prima fase**
  Dati: M_1, $MAC_1 = C(K, M_1)$.
  Calcolare $MAC_i = C(K_i, M_1)$ per tutte le 2^k chiavi.
  Numero di corrispondenze = $2^{(k-n)}$.
- **Seconda fase**
  Dati: M_2, $MAC_2 = C(K, M_2)$.
  Calcolare $MAC_i = C(K_i, M_2)$ per le $2^{(k-n)}$ chiavi ottenute dalla prima fase.
  Numero di corrispondenze $\approx 2^{(k-2\times n)}$.

e così via. In media, se $k = \alpha \times n$, saranno necessarie α fasi. Per esempio, se venisse utilizzata una chiave di 80 bit e un codice MAC di 32 bit, la prima fase produrrebbe circa 2^{48} possibili chiavi. La seconda fase restringerebbe il campo a 2^{16} possibilità. La terza fase dovrebbe produrre un'unica chiave che deve essere quella utilizzata dal mittente.

Se la lunghezza della chiave è minore o uguale della lunghezza del codice MAC, è probabile che una prima fase produrrebbe un'unica corrispondenza. È possibile che tale corrispondenza venga individuata da più chiavi, nel qual caso l'estraneo dovrà svolgere lo stesso test su una nuova coppia (messaggio, MAC).

Pertanto, un tentativo a forza bruta per scoprire la chiave di autenticazione non è più leggero e può richiedere un maggiore impegno rispetto a quello richiesto per scoprire una chiave di decrittografia della stessa lunghezza. Tuttavia è possibile impiegare anche altri attacchi che non richiedono la scoperta della chiave.

Si consideri il seguente algoritmo MAC. Sia $M = (X_1 \| X_2 \| \ldots \| X_m)$ un messaggio che viene trattato come la concatenazione di blocchi di 64 bit X_i. Si definiscano:

$$\Delta(M) = X_1 \oplus X_2 \oplus \ldots \oplus X_m$$
$$C(K, M) = E(K, \Delta(M))$$

Dove $\oplus$ è l'operazione di OR esclusivo (XOR) e l'algoritmo di crittografia impiegato è DES in modalità Electronic Codebook. In questo caso, la lunghezza della chiave è di 56 bit e la lunghezza del codice MAC è di 64 bit. Se un estraneo intercetta $\{M \| C(K, M)\}$, un tentativo a forza bruta di determinare K richiederà almeno 2^{56} crittografie. Ma l'estraneo può attaccare il sistema sostituendo da X_1 a X_{m-1} con qualsiasi valore desiderato da Y_1 a Y_{m-1} e sostituendo X_m con Y_m dove Y_m viene calcolato nel seguente modo:

$$Y_m = Y_1 \oplus Y_2 \oplus \ldots \oplus Y_{m-1} \oplus \Delta(M)$$

L'estraneo può ora concatenare il nuovo messaggio, costituito da $Y_1 - Y_m$ con il codice MAC originario per formare un messaggio che verrà accettato come autentico dal destinatario. Con questa tattica, qualsiasi messaggio di lunghezza di $64 \times (m - 1)$ bit può essere inserito in modo fraudolento.

Pertanto, nella valutazione della sicurezza di una funzione MAC, occorre considerare i vari tipi di attacchi che possono essere escogitati. Tenendo in considerazione questo fatto, si possono stabilire i requisiti della funzione. Si supponga che un estraneo conosca la funzione MAC C ma non conosca K. La funzione MAC dovrebbe soddisfare i seguenti requisiti.

1. Anche se un estraneo osservasse M e $C(K, M)$, dovrebbe essere computazionalmente impossibile costruire un messaggio M' tale che $C(K, M') = C(K, M)$.
2. $C(K, M)$ dovrebbe essere distribuito uniformemente nel senso che per messaggi scelti casualmente, M e M', la probabilità che $C(K, M) = C(K, M')$ deve essere 2^{-n}, dove n è il numero di bit che compongono il codice MAC.
3. Se M' è uguale a una trasformazione nota di M, ovvero $M' = f(M)$ (per esempio f potrebbe invertire uno o più bit), allora $\Pr[C(K, M) = C(K, M')] = 2^{-n}$.

Il primo requisito fa riferimento all'esempio precedente in cui l'estraneo è in grado di costruire un nuovo messaggio coincidente con un determinato codice MAC, anche se l'estraneo non conosce e non può individuare la chiave. Il secondo requisito fa riferimento alla necessità di impedire un attacco a forza bruta sulla base di un testo in chiaro scelto. Ovvero, se si suppone che l'estraneo non conosca K ma abbia accesso alla funzione MAC e possa presentare messaggi per la generazione dei codici MAC corrispondenti, l'estraneo potrebbe tentare vari messaggi fino a trovarne uno che coincida con un determinato MAC. Se la funzione MAC esibisse una distribuzione uniforme, un metodo a forza bruta richiederebbe, in media, $2^{(n-1)}$ tentativi prima di trovare un messaggio che generi un determinato codice MAC.

L'ultimo requisito stabilisce che l'algoritmo di autenticazione non sia più debole in determinate parti o bit del messaggio rispetto ad altre. Se non fosse così, un estraneo che conoscesse M e $C(K, M)$ potrebbe tentare delle varianti su M nei "punti deboli" noti con una maggiore probabilità di successo nella creazione di un nuovo messaggio che generi il vecchio codice MAC.

Codici MAC basati su DES

Il Data Authentication Algorithm, basato su DES, è stato uno dei codici MAC più diffusi per numerosi anni. L'algoritmo è una pubblicazione FIPS (FIPS PUB 113) e uno standard ANSI (X9.17). Tuttavia, come si vedrà nel Capitolo 12, in questo algoritmo sono state scoperte alcune lacune di sicurezza e sta quindi per essere rimpiazzato da algoritmi più recenti e robusti.

L'algoritmo può essere definito come una modalità CBC (Cipher Block Chaining) di DES (Figura 6.4) con un vettore di inizializzazione 0 (zero). I dati (messaggi, record, file o programmi) da autenticare sono raggruppati in blocchi di 64 bit contigui: $D_1, D_2, \ldots, D_N$. Se necessario, l'ultimo blocco viene completato con una sequenza di 0 per formare un blocco da 64 bit. Utilizzando l'algoritmo di crittografia DES, E, e una chiave segreta, K, il codice di autenticazione DAC (Data Authentication Code) viene calcolato nel seguente modo (Figura 11.6):

$$
\begin{aligned}
O_1 &= E(K, D_1) \\
O_2 &= E(K, D_2 \oplus O_1) \\
O_3 &= E(K, D_3 \oplus O_2) \\
&\vdots \\
O_N &= E(K, D_N \oplus O_{N-1})
\end{aligned}
$$

Il codice DAC è costituito dall'intero blocco O_N o dai primi M bit del blocco, con $16 \leq M \leq 64$.

11.4 Le funzioni hash

Un valore hash h viene generato da una funzione H con la seguente forma:

$$h = H(M)$$

dove M è un messaggio di lunghezza variabile e $H(M)$ è il valore hash di lunghezza fissa. Il valore hash viene aggiunto al messaggio alla sorgente nel momento in cui il messaggio viene considerato corretto. Alla ricezione si autentica tale messaggio ricalcolando il valore hash. Poiché la funzione hash non è segreta, è necessario un modo per proteggere il valore hash (Figura 11.5).

Per iniziare si esaminano i requisiti di una funzione hash da utilizzare per l'autenticazione del messaggio. Poiché le funzioni hash sono, normalmente, piuttosto complesse, è utile esaminare qualche semplice funzione per avere un'idea dei problemi coinvolti. Quindi si vedranno vari approcci alla progettazione della funzione hash.

I requisiti di una funzione hash

Lo scopo di una funzione hash è quello di produrre una sorta di "impronta digitale" di un file, messaggio o altro blocco di dati. Per poter essere utile per l'autenticazione dei messaggi, una funzione hash H deve avere le seguenti proprietà (adattate da un elenco presentato in [NECH92]).

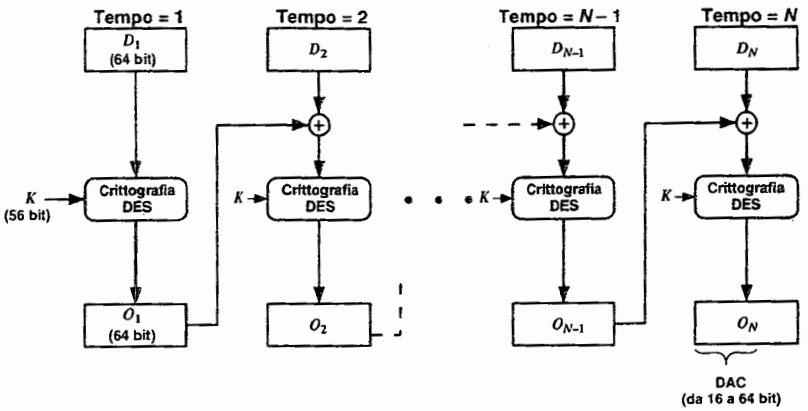


Figura 11.6 L'algoritmo Data Authentication Algorithm (FIPS PUB 113).

1. H può essere applicata a un blocco di dati di qualsiasi dimensione.
2. H produce un output di lunghezza fissa.
3. $H(x)$ è relativamente facile da calcolare, consentendo quindi un'implementazione sia hardware che software.
4. Per un determinato valore h è computazionalmente impossibile trovare un valore x tale che $H(x) = h$. In genere questa proprietà è detta **monodirezionalità**.
5. Per un determinato blocco x è computazionalmente impossibile trovare un valore $y \neq x$ tale che $H(y) = H(x)$. Questa proprietà è normalmente chiamata **resistenza debole alle collisioni**.
6. È computazionalmente impossibile trovare una coppia (x, y) tale che $H(y) = H(x)$. Questa viene normalmente chiamata **resistenza forte alle collisioni**.[2]

Le prime tre proprietà sono requisiti per l'applicazione pratica di una funzione hash all'autenticazione dei messaggi.

La quarta proprietà, la monodirezionalità, afferma che è facile generare un codice sulla base di un messaggio mentre è praticamente impossibile generare un messaggio sulla base di un codice. Questa proprietà è importante se la tecnica di autenticazione prevede l'uso di un valore segreto (Figura 11.5E). Il valore segreto non viene inviato, ma se la funzione hash non fosse monodirezionale, un estraneo potrebbe con facilità scoprire il valore segreto: se l'estraneo potesse osservare o intercettare una trasmissione, otterrebbe il messaggio M e il codice hash $C = H(S_{AB} \| M)$. A questo punto l'estraneo potrebbe invertire la funzione hash per ottenere $S_{AB} \| M = H^{-1}(C)$. Poiché ora l'estraneo avrebbe sia M che $S_{AB} \| M$, potrebbe facilmente recuperare S_{AB}.

[2] Sfortunatamente questi termini non sono utilizzati in modo uniforme. Fra i termini utilizzati in letteratura vi sono *funzione hash monodirezionale* (proprietà 4 e 5); *funzione hash resistente alle collisioni* (proprietà 4, 5 e 6); *funzione hash monodirezionale debole* (proprietà 4 e 5); *funzione hash monodirezionale forte* (proprietà 4, 5 e 6). Nel consultare la letteratura è quindi necessario prestare attenzione al significato reale dei termini utilizzati.

La quinta proprietà garantisce che non sia possibile trovare un altro messaggio in grado di produrre lo stesso valore hash. Questo impedisce ogni alterazione del messaggio quando viene utilizzato il codice hash (Figura 11.5B e C). Per questi casi, l'estraneo può leggere il messaggio e pertanto generare il proprio codice hash. Tuttavia, poiché l'estraneo non ha la chiave segreta, non dovrebbe essere in grado di alterare il messaggio senza essere rilevato. Se questa proprietà non fosse vera, un estraneo sarebbe in grado di svolgere le seguenti operazioni: innanzitutto, potrebbe osservare o intercettare un messaggio e il codice hash crittografato; poi potrebbe generare un codice hash non crittografato a partire dal messaggio; infine potrebbe generare un altro messaggio che utilizzi lo stesso codice hash.

La sesta proprietà fa riferimento alla resistenza della funzione hash a una classe di attacchi denominati *attacchi a compleanno*, di cui si parlerà fra breve.

Semplici funzioni hash

Tutte le funzioni hash utilizzano i seguenti principi generali. L'input (messaggio, file o quant'altro) viene considerato come una sequenza di blocchi di n bit. L'input viene elaborato un blocco alla volta in modo iterativo per produrre un valore hash di n bit.

Una delle funzioni hash più semplici è l'operazione di OR esclusivo bit a bit (XOR) applicata a ciascun blocco. L'operazione può essere espressa nel seguente modo:

$$C_i = b_{i1} \oplus b_{i2} \oplus \ldots \oplus b_{im}$$

dove:

C_i = i-esimo bit del codice hash, $1 \le i \le n$.
m = numero di blocchi di n bit di input.
b_{ij} = i-esimo bit nel j-esimo blocco.
$\oplus$ = operazione XOR

Questa operazione produce un semplice bit di parità per ciascuna posizione di bit. Questo codice è noto come controllo di ridondanza longitudinale. È ragionevolmente efficace come controllo di integrità dei dati per dati casuali. Ogni valore hash di n bit è ugualmente probabile. Pertanto la probabilità che un errore nei dati alteri il valore hash è pari a 2^{-n}. Con dati strutturati in modo più prevedibile, la funzione è meno efficace. Per esempio, nella maggior parte dei file di testo, il bit di ordine più elevato di ciascun ottetto è sempre 0 (zero). Pertanto se viene utilizzato un valore hash di 128 bit, invece di un'efficacia di 2^{-128}, la funzione hash su questo tipo di dati avrà un'efficacia pari a 2^{-112}.

Un modo semplice per migliorare le cose consiste nell'applicare uno scorrimento circolare, ovvero una rotazione, sul valore hash dopo avere elaborato ciascun blocco. La procedura può essere riepilogata nel seguente modo.

1. Impostare inizialmente a 0 gli n bit del valore hash.
2. Elaborare ciascun blocco successivo di n bit di dati nel seguente modo.
   A. Ruotare a sinistra di un bit il valore hash corrente.
   B. Eseguire l'operazione di XOR del blocco nel valore hash.

Questo ha l'effetto di "casualizzare" l'input, ovvero di eliminare ogni regolarità presente nell'input. La Figura 11.7 illustra questi due tipi di funzioni per valori hash a 16 bit.

Sebbene la seconda procedura garantisca in buona misura l'integrità dei dati, è praticamente inutile per la sicurezza quando viene utilizzato un codice hash crittografato con un messaggio in chiaro, come nella Figura 11.5B e C. Dato un messaggio, è facile produrre un nuovo messaggio che fornisce lo stesso codice hash: basta preparare il messaggio alternativo desiderato e aggiungere un blocco di n bit che costringa il nuovo messaggio a fornire tale codice hash.

Sebbene una semplice operazione di XOR o XOR ruotato (RXOR) sia insufficiente se viene crittografato solo il codice hash, si potrebbe comunque ritenere che tale semplice funzione possa essere utile quando vengono crittografati sia il messaggio che il codice hash (Figura 11.5A). Occorre però fare attenzione. Una tecnica originariamente proposta dal National Bureau of Standards usava un semplice XOR applicato a blocchi di 64 bit del messaggio e poi una crittografia dell'intero messaggio che utilizzava la modalità CBC (Cipher Block Chaining). Si può definire lo schema nel seguente modo: dato un messaggio

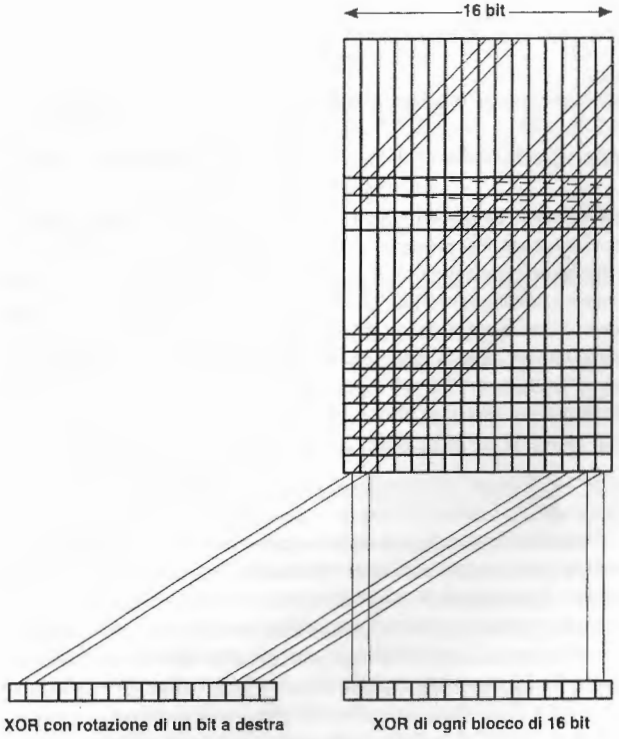


Figura 11.7 Due semplici funzioni hash.

costituito da una sequenza di blocchi di 64 bit $X_1, X_2, \ldots, X_N$, definire il codice hash C come lo XOR blocco per blocco di tutti i blocchi aggiungendo il codice hash come blocco finale:

$$C = X_{N+1} = X_1 \oplus X_2 \oplus \ldots \oplus X_N$$

Quindi crittografare l'intero messaggio più il codice hash utilizzando la modalità CBC per produrre il messaggio crittografato $Y_1, Y_2, \ldots, Y_{N+1}$. [JUEN85] indica vari modi in cui può essere manipolato il testo cifrato di questo messaggio in modo non rilevabile dal codice hash. Per esempio, per definizione di CBC (Figura 6.4), si avrà:

$$X_1 = IV \oplus D(K, Y_1)$$
$$X_i = Y_{i-1} \oplus D(K, Y_i)$$
$$X_{N+1} = Y_N \oplus D(K, Y_{N+1})$$

Ma X_{N+1} è il codice hash:

$$X_{N+1} = X_1 \oplus X_2 \oplus \ldots \oplus X_N = (IV \oplus D(K, Y_1)) \oplus (Y_1 \oplus D(K, Y_2)) \oplus \ldots \oplus (Y_{N-1} \oplus D(K, Y_N))$$

Poiché i termini dell'equazione precedente possono essere sottoposti a uno XOR in qualsiasi ordine, ne consegue che il codice hash non cambierà nel caso in cui vengano permutati i blocchi di testo cifrato.

Attacchi a compleanno

Si supponga che venga utilizzato un codice hash a 64 bit. Si potrebbe pensare che questa tecnica sia sufficientemente sicura. Per esempio, se un codice hash crittografato C venisse trasmesso con il corrispondente messaggio non crittografato M (Figura 11.5B o C), un estraneo dovrebbe trovare un messaggio M' tale che $H(M') = H(M)$ per sostituire il messaggio originale e ingannare il destinatario. In media, l'estraneo dovrebbe provare circa 2^{63} messaggi per trovarne uno che produca lo stesso codice hash del messaggio intercettato; a tale proposito consultare l'Equazione 11.1 nell'Appendice 11.A.

Tuttavia è possibile impiegare anche un altro tipo di attacco, basato sul paradosso del compleanno (Appendice 11.A). Yuval ha proposto la seguente strategia [YUVA79].

1. Il mittente, A, si prepara a "firmare" un messaggio aggiungendo il codice hash di m bit appropriato e crittografando tale codice hash con la propria chiave privata (Figura 11.5C).
2. L'estraneo genera $2^{m/2}$ varianti del messaggio, ognuna delle quali ha fondamentalmente lo stesso significato. L'estraneo prepara un ugual numero di messaggi che rappresentano varianti del messaggio fraudolento da sostituire al vero messaggio.
3. I due insiemi di messaggi vengono confrontati per trovare una coppia di messaggi che producono lo stesso codice hash. La probabilità di successo, per il paradosso del compleanno, è maggiore di 0,5. Se non viene trovata alcuna corrispondenza, vengono generati altri messaggi validi e fraudolenti, fino a trovare una corrispondenza.
4. L'estraneo offre la variante valida ad A per la firma. Questa firma può essere associata alla variante fraudolenta e trasmessa al destinatario. Poiché le due varianti hanno lo stesso codice hash, produrranno la stessa firma; questo garantirà all'estraneo ogni probabilità di successo anche se la chiave di crittografia non è nota.

Pertanto, se viene utilizzato un codice hash di 64 bit, l'impegno richiesto è solo dell'ordine di 2^{32} (vedere l'Equazione 11.7 nell'Appendice 11.A).

Non è difficile generare più varianti che forniscono lo stesso significato. Per esempio l'estraneo potrebbe inserire fra le parole del documento varie coppie di caratteri "spazio-spazio-backspace".

Le varianti potrebbero quindi essere generate sostituendo le sequenze "spazio-backspace-spazio" nelle varie istanze. Alternativamente l'estraneo potrebbe semplicemente riformulare il messaggio conservando il significato. Un esempio, tratto da [DAVI89], si trova nella Figura 11.8.

La conclusione da trarre è che la lunghezza del codice hash deve essere notevole. L'argomento verrà approfondito nel Paragrafo 11.5.

Dear Anthony,

{This letter is / I am writing} to introduce {you to / to you} {Mr. / --} Alfred {P. / --}
Barton, the {new / newly appointed} {chief / senior} jewellery buyer for {our / the}
Northern {European / Europe} {area / division}. He {will take / has taken} over {the / --}
responsibility for {all / the whole of} our interests in {watches and jewellery / jewellery and watches}
in the {area / region}. Please {afford / give} him {every / all the} help he {may need / needs}
to {seek out / find} the most {modern / up to date} lines for the {top / high} end of the
market. He is {empowered / authorized} to receive on our behalf {samples / specimens} of the
{latest / newest} {watch and jewellery / jewellery and watch} products, {up / subject} to a {limit / maximum}
of ten thousand dollars. He will {carry / hold} a signed copy of this {letter / document}
as proof of identity. An order with his signature, which is {appended / attached}
{authorizes / allows} you to charge the cost to this company at the {above / head office}
address. We {fully / --} expect that our {level / volume} of orders will increase in
the {following / next} year and {trust / hope} that the new appointment will {be / prove}
{advantageous / an advantage} to both our companies.

Figura 11.8 Una lettera in 2^{37} varianti (DAVI89).

Tecniche di concatenamento a blocchi

Sono state fatte varie proposte relative a funzioni hash basate sull'uso di una cifratura con tecnica di concatenamento a blocchi senza chiave segreta. Una delle prime proposte fu quella di Rabin [RABI78]. Si deve dividere un messaggio M in blocchi di dimensioni fisse $M_1, M_2, \ldots, M_N$ e utilizzare un sistema di crittografia simmetrico come DES per calcolare il codice hash G nel seguente modo:

$H_0 = \text{valore iniziale}$
$H_i = E(M_i, H_{i-1})$
$G = H_N$

Si tratta di una tecnica simile alla CBC ma in questo caso non vi è alcuna chiave segreta. Come per ogni altro codice hash, questo schema è soggetto all'attacco a compleanno e se l'algoritmo di crittografia è DES e viene prodotto un codice hash di soli 64 bit, allora il sistema è vulnerabile. Inoltre esiste un'altra versione dell'attacco a compleanno utilizzabile anche se l'estraneo avesse accesso a un solo messaggio e alla relativa firma valida e non potesse ottenere altre firme. Si supponga che l'estraneo intercetti un messaggio con una firma sotto forma di un codice hash crittografato e che il codice hash non crittografato sia lungo m bit.

1. Usare l'algoritmo definito all'inizio di questa parte del capitolo per calcolare il codice hash non crittografato G.
2. Costruire il messaggio desiderato nella forma $Q_1, Q_2, \ldots, Q_{N-2}$.
3. Calcolare $H_i = E(Q_i, H_{i-1})$ per $1 \le i \le (N-2)$.
4. Generare $2^{m/2}$ blocchi casuali; per ciascun blocco X, calcolare $E(X, H_{N-2})$. Generare ulteriori $2^{m/2}$ blocchi casuali. Per ciascun blocco Y, calcolare $D(Y, G)$, dove D è la funzione di decrittografia corrispondente a E.
5. Sulla base del paradosso del compleanno, vi è un'elevata probabilità che vi siano un X e un Y tali che $E(X, H_{N-2}) = D(Y, G)$.
6. Costruire il messaggio $Q_1, Q_2, \ldots, Q_{N-2}, X, Y$. Questo messaggio contiene il codice hash G e pertanto può essere utilizzato con la firma crittografata intercettata.

Questa forma di attacco è chiamata attacco "Meet-in-the-Middle". Vari ricercatori hanno proposto dei raffinamenti che hanno lo scopo di rafforzare l'approccio di base. Per esempio, Davies e Price [DAVI89] descrivono la seguente variante:

$$H_i = E(M_i, H_{i-1}) \oplus H_{i-1}$$

Ecco un'altra variante proposta in [MEYE88]:

$$H_i = E(H_{i-1}, M_i) \oplus M_i$$

Tuttavia è stato dimostrato che entrambi questi schemi sono vulnerabili a vari attacchi [MIYA90]. Più in generale, si può dimostrare che una qualche forma di attacco a compleanno ha successo contro qualsiasi schema hash che faccia uso di un concatenamento a blocchi cifrati senza una chiave segreta, sempre che il codice hash prodotto sia di dimensioni sufficientemente piccole (per esempio 64 bit o meno) o che un codice hash di grandi dimensioni possa essere decomposto in sottocodici indipendenti [JUEN87].

Pertanto si è rivolta l'attenzione alla ricerca di altri approcci al calcolo della funzione hash. Molti di questi hanno già manifestato dei punti deboli [MITC92]. Alcune funzioni hash più resistenti verranno trattate nel Capitolo 12.

11.5 La sicurezza delle funzioni hash e dei codici MAC

Come nel caso della crittografia simmetrica e a chiave pubblica, gli attacchi alle funzioni hash e ai codici MAC possono essere raggruppati in due categorie: attacchi a forza bruta e ad analisi crittografica.

Attacchi a forza bruta

La natura degli attacchi a forza bruta è differente per le funzioni hash e i codici MAC.

Funzioni hash

La resistenza di una funzione hash contro gli attacchi a forza bruta dipende esclusivamente dalla lunghezza del codice hash prodotto dall'algoritmo. Come si è detto in precedenza, le funzioni hash dovrebbero soddisfare tre proprietà.

- **Monodirezionalità:** per un determinato codice **h**, è computazionalmente impossibile trovare **x** tale che $H(x) = h$.
- **Resistenza debole alle collisioni:** per un determinato blocco x, è computazionalmente impossibile trovare $y \neq x$ tale che $H(y) = H(x)$.
- **Resistenza forte alle collisioni:** è computazionalmente impossibile trovare una coppia (x, y) tale che $H(x) = H(y)$.

Per un codice hash di lunghezza n, l'impegno richiesto, come si è visto, è proporzionale ai seguenti valori.

Monodirezionalità	2^n
Resistenza debole alle collisioni	2^n
Resistenza forte alle collisioni	$2^{n/2}$

Se è richiesta la resistenza forte alle collisioni (proprietà desiderabile per un codice hash sicuro di utilizzo generale), il valore $2^{n/2}$ determina la resistenza del codice hash contro gli attacchi a forza bruta. Oorschot e Wiener [VANO94] hanno presentato un progetto per una macchina per la ricerca di collisioni da 10 milioni di dollari per MD5 che usa un codice hash della lunghezza di 128 bit e che potrebbe trovare una collisione in 24 giorni. Pertanto un codice di 128 bit può essere considerato inadeguato. Il passo successivo, considerando incrementi di 32 bit, è un codice hash della lunghezza di 160 bit. In questo caso, la stessa macchina richiederebbe oltre quattromila anni per trovare una collisione. Tuttavia, anche 160 bit sono ora considerati inadeguati. Si ritornerà sull'argomento nel Capitolo 12.

Codici MAC

Un attacco a forza bruta contro un codice MAC è un'impresa più difficile poiché richiede la conoscenza di coppie messaggio/codice MAC. Ora si vedrà il motivo di tale difficoltà. Per attaccare un codice hash si può procedere nel seguente modo. Dato un messaggio fisso x con hash di n bit $h = H(x)$, il metodo a forza bruta per la ricerca di una collisione consiste nello scegliere una stringa di bit casuali y e verificare se $H(y) = H(x)$. L'estraneo può eseguire ripetutamente questa operazione offline.

Il fatto che contro un algoritmo MAC possa essere utilizzato un attacco offline, dipende dalle dimensioni relative della chiave e del codice MAC. Per procedere, occorre definire la proprietà desiderata per l'algoritmo MAC, che può essere espressa nel modo seguente.

- **Resistenza computazionale:** date una o più coppie testo/codice MAC $[x_i, C(K, x_i)]$, è computazionalmente impossibile calcolare una coppia testo/codice MAC $[x, C(K, x)]$ per ogni nuovo input $x \neq x_i$.

In altre parole, l'estraneo vorrebbe determinare il codice MAC corretto per un determinato messaggio x. Può utilizzare due diverse strategie di attacco: attaccare lo spazio delle chiavi o attaccare il codice MAC. Verranno esaminate entrambe le possibilità.

Se un estraneo riesce a determinare la chiave MAC, allora potrà generare un valore MAC valido per qualsiasi input x. Si supponga che le dimensioni della chiave siano pari a k bit e che l'estraneo abbia a disposizione una coppia testo/codice MAC. In questo caso potrà calcolare il codice MAC a n bit sul testo noto per tutte le chiavi possibili. Almeno una di esse produrrà il codice MAC corretto: questa è la chiave inizialmente utilizzata per produrre la coppia di testo/codice MAC nota. Questa fase dell'attacco richiede un impegno proporzionale a 2^k (ovvero un'operazione per ognuno dei 2^k possibili valori della chiave). Tuttavia, come si è detto in precedenza, poiché il codice MAC è un mapping molti-a-uno, vi possono essere anche altre chiavi che producono il valore corretto. Pertanto, se viene trovata più di una chiave che produce il valore corretto, sarà necessario verificare altre coppie testo/codice MAC. Si può dimostrare che il livello di impegno cala rapidamente con ogni coppia di testo/codice MAC addizionale e che il livello di impegno globale è pari a circa 2^k [MENE97].

Un estraneo può anche lavorare sul codice MAC senza tentare di ottenere la chiave. In questo caso l'obiettivo è generare un valore MAC valido per un determinato messaggio o trovare un messaggio corrispondente a un determinato valore MAC. In entrambi i casi, il livello di impegno è paragonabile a quello necessario per allocare la proprietà monodirezionale o la resistenza debole alle collisioni di un codice hash, ovvero 2^n. Nel caso di MAC, l'attacco non può essere condotto offline senza ulteriori input; l'estraneo dovrebbe avere coppie scelte testo/codice MAC o conoscere la chiave.

Per riepilogare, il livello di impegno per l'attacco a forza bruta contro un algoritmo MAC può essere espresso come $\min(2^k, 2^n)$. La valutazione della resistenza è simile a quella degli algoritmi di crittografia simmetrici. Sembrerebbe ragionevole richiedere che la lunghezza della chiave e la lunghezza del codice MAC soddisfino una relazione come $\min(k, n) \geq N$, dove N è dell'ordine dei 128 bit.

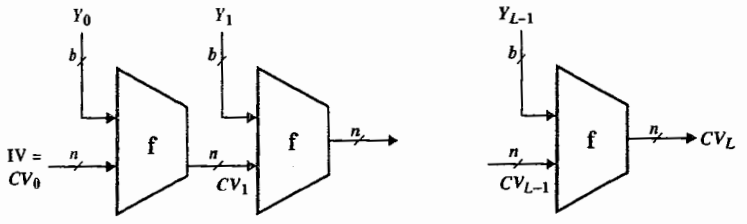
Analisi crittografica

Come nel caso degli algoritmi di crittografia, gli attacchi ad analisi crittografica contro le funzioni hash e gli algoritmi MAC cercano di sfruttare certe proprietà dell'algoritmo per svolgere attacchi diretti senza impiegare una ricerca esaustiva. Il modo per misurare la resistenza di un algoritmo hash o MAC all'analisi crittografica consiste nel confrontare la sua resistenza all'impegno necessario per un attacco a forza bruta. Un algoritmo hash o MAC ideale richiede un impegno per l'analisi crittografica maggiore o uguale rispetto a quello di un attacco a forza bruta.

Funzioni hash

Negli ultimi anni, vi sono stati numerosi sforzi e anche qualche successo nello sviluppo di attacchi ad analisi crittografica contro le funzioni hash. Per comprenderli occorre considerare la struttura generale di una tipica funzione hash sicura, rappresentata nella Figura 11.9. Questa struttura, chiamata funzione hash iterata, è stata proposta da Merkle [MERK79, MERK89] ed è la struttura della maggior parte delle funzioni hash attualmente utilizzate, fra cui SHA e Whirlpool che verranno descritte nel Capitolo 12. La funzione hash prende un messaggio di input e lo suddivide in L blocchi di dimensioni fisse pari a b bit. Se necessario, l'ultimo blocco viene completato in modo da ottenere b bit. L'ultimo blocco contiene anche la lunghezza totale dell'input della funzione hash. L'inclusione della lunghezza complica il lavoro di violazione. Infatti l'estraneo deve trovare due messaggi di uguale lunghezza che producono lo stesso valore o due messaggi di lunghezza differente che, insieme ai valori di lunghezza, producono tramite la funzione hash lo stesso valore.

L'algoritmo hash comporta la ripetizione di una *funzione di compressione*, f, che accetta due input (un input di n bit tratto dal passo precedente chiamato *variabile di concatenamento* e un blocco di b bit) e produce un output di n bit. All'inizio del calcolo



IV = valore iniziale
CV = variabile di concatenamento
Y_i = i-esimo blocco di input
f = algoritmo di compressione
L = numero di blocchi di input
n = lunghezza del codice hash
b = lunghezza del blocco di input

Figura 11.9 Struttura generale di un codice hash sicuro.

della funzione hash, la variabile di concatenamento ha un valore iniziale specificato dall'algoritmo. Il valore finale della variabile di concatenamento è il valore hash. Normalmente $b > n$; da qui il termine *compressione*. La funzione hash può essere riepilogata nel seguente modo:

$$CV_0 = IV = \text{valore iniziale di } n \text{ bit}$$
$$CV_i = f(CV_{i-1}, Y_{i-1}) \qquad 1 \le i \le L$$
$$H(M) = CV_L$$

dove l'input della funzione hash è un messaggio M costituito dai blocchi $Y_0, Y_1, \ldots, Y_{L-1}$.

La motivazione della struttura iterativa deriva dall'osservazione di Merkle [MERK89] e Damgard [DAMG89] che se la funzione di compressione è resistente alle collisioni, lo sarà anche la funzione hash iterata risultante.[3] Pertanto, la struttura può essere utilizzata per produrre una funzione hash sicura per messaggi di qualsiasi lunghezza. Il problema della progettazione di una funzione hash sicura si riduce alla progettazione di una funzione di compressione resistente alle collisioni che operi su input di dimensioni fisse.

L'analisi crittografica delle funzioni hash si concentra sulla struttura interna di f e si basa sui tentativi di trovare tecniche efficienti per produrre collisioni con un'unica esecuzione di f. L'attacco deve poi tenere in considerazione il valore fisso di IV. L'attacco contro f dipende dallo sfruttamento della sua struttura interna. Tipicamente, come nelle cifrature simmetriche a blocchi, f è costituita da una serie di fasi di elaborazione: l'attacco comporta quindi l'analisi della sequenza di variazioni dei bit da fase a fase.

Si deve tenere in considerazione che ogni funzione hash presenta necessariamente delle collisioni in quanto deve mappare un messaggio di lunghezza quanto meno uguale al doppio delle dimensioni del blocco b (perchè si deve aggiungere un campo dimensione) in un codice hash di lunghezza n dove $b \ge n$. Ciò che si richiede è che sia computazionalmente impossibile trovare delle collisioni.

Gli attacchi applicati alle funzioni hash sono piuttosto complessi e non rientrano negli scopi di questo volume. Il lettore interessato può consultare [DOBB96a] e [BELL97].

Codici MAC

La struttura dei codici MAC è ancora più varia rispetto alle funzioni hash e dunque è ancora più difficile fare generalizzazioni sull'analisi crittografica dei codici MAC. Inoltre è stato svolto molto meno lavoro nello sviluppo di questi attacchi. Un'utile e recente indagine di alcuni metodi specifici per MAC si trova in [PREN96].

11.6 Letture e siti Web consigliati

[JUEN85] e [JUEN87] forniscono informazioni di base sull'autenticazione dei messaggi concentrandosi sui codici MAC e le funzioni hash. Per una trattazione approfondita delle funzioni hash e dei codici MAC, consultare [STIN02] e [MENE97]. Un'indagine ottima e recente si trova in [PREN99].

[3] Non necessariamente vale il contrario.

JUEN85 R. Jueneman, S. Matyas e C. Meyer. "Message Authentication". *IEEE Communications Magazine*, Settembre 1988.

JUEN87 R. Jueneman. "Electronic Document Authentication". *IEEE Network Magazine*, Aprile 1987.

MENE97 A. Menezes, P. Oorshcot e S. Vanstone. *Handbook of Applied Cryptography*. Boca Raton, FL: CRC Press, 1997.

PREN99 B. Preneel. "The State of Cryptographic Hash Functions". *Proceedings, EUROCRYPT '96*, 1996; pubblicato da Springer-Verlag.

STIN02 D. Stinson. *Cryptography: Theory and Practice*. Boca Raton, FL: CRC Press, 2002.

11.7 Termini chiave, domande di ripasso e problemi

Termini chiave

Attacco a compleanno
Autenticatore
Autenticazione del messaggio
Checksum crittografico
Codice hash
Codice MAC
Funzione di compressione
Funzione hash
Funzione hash monodirezionale
Message digest
Paradosso del compleanno
Resistenza debole alle collisioni
Resistenza forte alle collisioni
Valore hash

Domande di ripasso

11.1 Quali tipi di attacchi sono sventati dall'autenticazione dei messaggi?
11.2 Quali livelli di funzionalità comprendono i meccanismi di autenticazione dei messaggi e di firma digitale?
11.3 Quali sono gli approcci all'autenticazione dei messaggi?
11.4 Quando viene utilizzata una combinazione di crittografia simmetrica e di codice di controllo degli errori per l'autenticazione dei messaggi, in quale ordine devono essere svolte le due funzioni?
11.5 Che cos'è il codice MAC?
11.6 Qual è la differenza fra un codice MAC e una funzione hash monodirezionale?
11.7 In quali modi un valore hash può essere reso sicuro in modo da fornire la funzione di autenticazione dei messaggi?

11.8 È necessario conoscere la chiave segreta per attaccare un algoritmo MAC?
11.9 Quali caratteristiche deve avere una funzione hash sicura?
11.10 Qual è la differenza fra resistenza alle collisioni debole e forte?
11.11 Qual è il ruolo della funzione di compressione in una funzione hash?

Problemi

11.1 Se F è una funzione di rilevamento degli errori, sia l'utilizzo interno che esterno (Figura 11.2) fornirà la funzionalità di rilevamento degli errori. Se viene modificato un bit del messaggio trasmesso, questo produrrà una differenza fra l'FCS ricevuto e l'FCS calcolato indipendentemente dal fatto che la funzione FCS venga eseguita all'interno o all'esterno della funzione di crittografia. Alcuni codici fomiscono anche una funzionalità di correzione degli errori. A seconda della natura della funzione, se un ridotto numero di bit viene alterato nel transito, il codice di correzione degli errori conterrà sufficienti informazioni ridondanti per determinare il bit o i bit errati e correggerli. Chiaramente il codice di correzione degli errori utilizzato esternamente alla funzione di crittografia garantisce la capacità di correzione. Sarebbe possibile fornire questa funzionalità anche se venisse impiegato internamente alla funzione di crittografia?

11.2 L'algoritmo di autenticazione dei dati descritto nel Paragrafo 11.3 può essere definito come la modalità di funzionamento CBC (Cipher Block Chaining) di DES con un vettore di inizializzazione 0 (Figura 11.7). Dimostrare che si può produrre lo stesso risultato utilizzando la modalità feedback.

11.3 Il protocollo di trasporto ad alta velocità XTP (Xpress Transfer Protocol) utilizza una funzione checksum a 32 bit definita come il concatenamento di due funzioni a 16 bit: XOR e RXOR, indicate nel Paragrafo 11.4 come "due semplici funzioni hash" (vedere la Figura 11.7).

A. Il checksum è in grado di rilevare tutti gli errori provocati da un numero di errori dispari nei bit? Spiegare.
B. Questo checksum è in grado di rilevare tutti gli errori provocati da un numero di errori pari nei bit? In caso contrario caratterizzare gli errori che provocano una segnalazione di errore nel checksum.
C. Commentare l'efficacia di questa funzione come funzione hash per l'autenticazione.

11.4 A Considerare lo schema del codice hash di Davies e Price descritto nel Paragrafo 11.4 e supporre che come algoritmo di crittografia venga utilizzato DES:

$$H_i = H_{i-1} \oplus E(M_i, H_{i-1})$$

Si ricordi la proprietà di complementarietà di DES (Problema 3.14): se $Y = E(K, X)$, allora $Y' = E(K', X')$. Usare questa proprietà per mostrare come un messaggio costituito dai blocchi $M_1, M_2, \ldots, M_N$ possa essere modificato senza alterare il suo codice hash.

B. Mostrare che un attacco simile avrebbe successo contro lo schema proposto in [MEYE88]:

$$H_i = M_i \oplus E(H_{i-1}, M_i)$$

11.5 A. Si consideri la seguente funzione hash. I messaggi sono espressi come sequenza di numeri decimali $M = (a_1, a_2, \ldots a_t)$. Il valore hash h è calcolato come $\left(\sum_{i=1}^{t} a_i\right) \bmod n$, per qualche valore predefinito di n. Questa funzione soddisfa qualche requisito fra quelli elencati nel Paragrafo 11.4 per le funzioni hash? Motivare le risposta.

B. Ripetere la parte (1) per la funzione hash $\left(\sum_{i=1}^{t} (a_i)^2\right) \bmod n$.

C. Calcolare la funzione hash della parte (2) per $M = (189, 632, 900, 722, 349)$ e $n = 989$.

11.6 È possibile utilizzare una funzione hash per costruire una cifratura a blocchi con una struttura simile a DES. Dato che una funzione hash è monodirezionale e una cifratura a blocchi deve essere invertibile (per la decrittografia), come è possibile ciò?

11.7 Ora si consideri il problema opposto: l'uso di un algoritmo di crittografia per costruire una funzione hash monodirezionale. Per esempio, si consideri l'impiego di RSA con una chiave nota. Elaborare un messaggio costituito da una sequenza di blocchi nel seguente modo: crittografare il primo blocco, eseguire l'operazione di XOR del risultato con il secondo blocco e crittografarlo ancora e così via. Mostrare che questo schema non è sicuro risolvendo il seguente problema. Dato un messaggio di due blocchi B1 e B2 e la funzione hash:

$$RSAH(B1,B2) = RSA(RSA(B1) \oplus B2)$$

e dato un blocco arbitrario C1, scegliere C2 in modo che RSAH(C1, C2) = RSAH(B1, B2).

11.8 Si supponga che $H(m)$ sia una funzione hash resistente alle collisioni, che mappi un messaggio di lunghezza (in bit) arbitraria in un valore hash di n bit. È vero che, per tutti i messaggi x, x' con $x \neq x'$ si ha $H(x) \neq H(x')$? Motivare la risposta.

Appendice 11.A Le basi matematiche dell'attacco a compleanno

In questa appendice viene fornita la giustificazione matematica dell'attacco a compleanno. Si partirà con un problema correlato per poi discutere il problema da cui deriva il nome "attacco a compleanno".

Un problema correlato

Ecco un problema generale relativo alle funzioni hash. Data una funzione hash H con n possibili output e un determinato valore $H(x)$, se H viene applicato a k input casuali, quale deve essere il valore di k tale che la probabilità che almeno un input y soddisfi $H(y) = H(x)$ sia 0,5?

Per un determinato valore di y, la probabilità che $H(y) = H(x)$ è solo $1/n$. Al contrario, la probabilità che $H(y) \neq H(x)$ è $[1 - (1/n)]$. Se si generano k valori casuali di y, allora la probabilità che nessuno di essi verifichi l'uguaglianza è semplicemente il prodotto delle probabilità che ciascun singolo valore non verifichi l'uguaglianza o $[1 - (1/n)]^k$. Pertanto la probabilità che vi sia almeno una corrispondenza è $1 - [1 - (1/n)]^k$.

Il teorema binomiale può essere formulato nel seguente modo:

$$(1-a)^k = 1 - ka + \frac{k(k-1)}{2!}a^2 - \frac{k(k-1)(k-2)}{3!}a^3 \ldots$$

Per valori molto piccoli di a, questo polinomio può essere approssimato a $(1 - ka)$. Pertanto la probabilità di almeno una corrispondenza è approssimata a $1 - [1 - (1/n)]^k \approx 1 - [1 - (k/n)] = k/n$. Per una probabilità di 0,5 si ha $k = n/2$.

In particolare, per un codice hash di m bit, il numero di codici possibili è 2^m e il valore di k che produce una probabilità di 1/2 è:

$$k = 2^{(m-1)} \tag{11.1}$$

Il paradosso del compleanno

Il paradosso del compleanno viene frequentemente presentato nei corsi elementari di probabilità per dimostrare che la probabilità ha risultati talvolta tutt'altro che intuitivi. Il problema può essere formulato nel seguente modo: qual è il valore minimo di k tale che la probabilità che almeno due persone in un gruppo di k persone abbiano lo stesso compleanno sia maggiore di 0,5? Si deve ignorare il 29 febbraio e supporre che ciascun compleanno sia ugualmente probabile. Per rispondere si può definire:

$P(n, k)$ = probabilità che esista almeno un duplicato in k elementi quando ciascun elemento può assumere un valore ugualmente probabile fra 1 e n.

Pertanto si sta cercando il più piccolo valore di k tale che $P(365, k) \geq 0{,}5$. È più facile derivare innanzitutto la probabilità che non vi siano duplicati che si indica $Q(365, k)$.
Se $k > 365$, allora è impossibile che tutti i valori siano differenti. Pertanto si supporrà che $k \leq 365$. Ora si consideri il numero dei vari modi, N, in cui vi possono essere k valori senza duplicati. Si può scegliere uno qualsiasi dei 365 valori come primo elemento, uno qualsiasi dei rimanenti 364 numeri come secondo elemento e così via. Pertanto il numero di modi diversi sarà:

$$N = 365 \times 364 \times \ldots (365 - k + 1) = \frac{365!}{(365-k)!} \tag{11.2}$$

Se si rimuove la restrizione che non vi siano duplicazioni, allora ciascun elemento potrà essere uno qualsiasi dei 365 valori e il numero totale di possibilità è 365^k. Pertanto le probabilità che non esistano duplicati è semplicemente la frazione degli insiemi dei valori che non hanno duplicati tra tutti i possibili insiemi di valori:

$$Q(365,k) = \frac{365!/(365-k)!}{(365)^k} = \frac{365!}{(365-k)!(365)^k}$$

e:

$$P(365,k) = 1 - Q(365,k) = 1 - \frac{365!}{(365-k)!(365)^k} \tag{11.3}$$

Questa funzione è rappresentata nella Figura 11.10. Le probabilità possono essere sorprendentemente elevati per chiunque non abbia considerato il problema in precedenza. Molti possono ritenere che per avere una probabilità maggiore di 0,5 che vi sia almeno un duplicato, il numero di componenti nel gruppo debba essere di circa 100. In realtà il numero è 23, con $P(365, 23) = 0{,}5073$. Per $k = 100$, la probabilità che vi sia almeno un duplicato è 0,9999997.

Questo risultato sembra sorprendente perché se si considera una determinata persona in un gruppo, la probabilità che qualche altra persona del gruppo abbia lo stesso compleanno è molto ridotta. Ma la probabilità cui si è interessati è la probabilità che una *qualsiasi* coppia di persone del gruppo abbia lo stesso compleanno. In un gruppo di 23 persone, vi sono $(23(23-1))/2 = 253$ diverse coppie di persone. Da qui le elevate probabilità.

Una disuguaglianza utile

Prima di sviluppare una generalizzazione del problema del compleanno, si deriva una disuguaglianza che sarà necessaria:

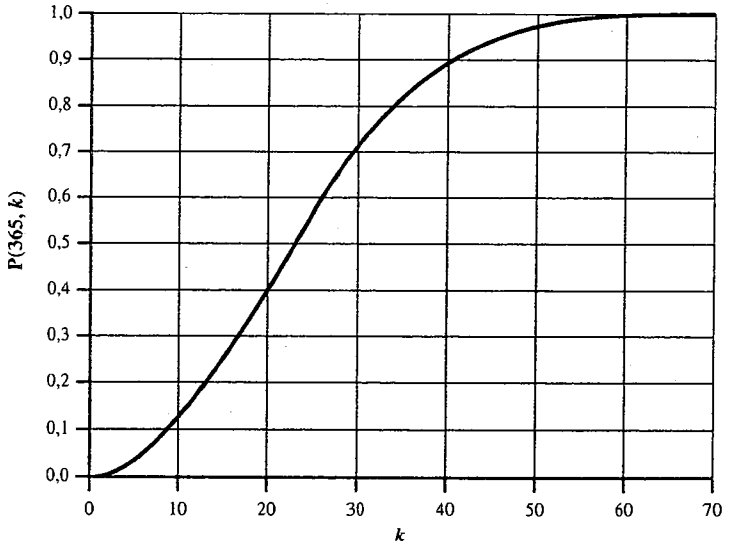


Figura 11.10 Il paradosso del compleanno.

$$(1-x) \le e^{-x} \qquad \text{per tutti gli } x \ge 0 \tag{11.4}$$

La Figura 11.11 illustra questa disuguaglianza. Per dimostrarla, si noti che la linea inferiore è tangente a e^{-x} in $x = 0$. La pendenza di tale linea è semplicemente la derivata di e^{-x} in $x = 0$:

$$f(x) = e^{-x}$$
$$f'(x) = \frac{d}{dx} e^{-x} = -e^{-x}$$
$$f'(0) = -1$$

La tangente è una retta della forma $ax + b$, con $a = -1$, e la tangente a $x = 0$ deve essere uguale a $e^{-0} = 1$. Pertanto la tangente è la funzione $(1 - x)$, confermando la disuguaglianza dell'Equazione 11.4. Inoltre si noti che per valori piccoli di x, si ha $(1 - x) \approx e^{-x}$.

Il caso generale delle duplicazioni

Il problema del compleanno può essere generalizzato dal seguente problema: data una variabile casuale intera con una distribuzione uniforme fra 1 e n e una scelta di k istanze ($k \le n$)) della variabile casuale, qual è la probabilità, P(n, k), che vi sia almeno un duplicato? Il problema del compleanno è semplicemente un caso speciale per $n = 365$. Con lo stesso ragionamento precedente, si ha la generalizzazione dell'Equazione 11.3:

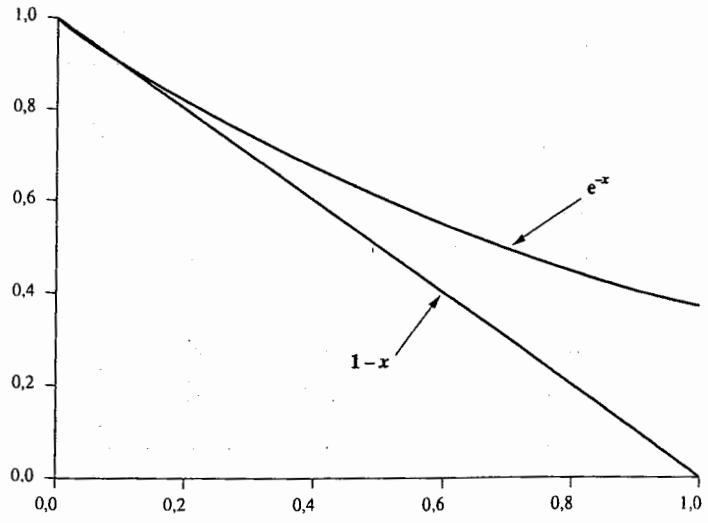


Figura 11.11 Un'utile disuguaglianza.

$$P(n,k) = 1 - \frac{n!}{(n-k)!\,n^k} \tag{11.5}$$

che può essere riscritta come:

$$P(n,k) = 1 - \frac{n \times (n-1) \times \ldots \times (n-k+1)}{n^k}$$
$$= 1 - \left[\frac{n-1}{n} \times \frac{n-2}{n} \times \ldots \times \frac{n-k+1}{n}\right]$$
$$= 1 - \left[\left(1-\frac{1}{n}\right) \times \left(1-\frac{2}{n}\right) \times \ldots \times \left(1-\frac{k-1}{n}\right)\right]$$

Utilizzando la disuguaglianza dell'Equazione 11.4:

$$P(n,k) > 1 - [(e^{-1/n}) \times (e^{-2/n}) \times \ldots \times (e^{-(k-1)/n})]$$
$$> 1 - e^{-[(1/n)+(2/n)+\ldots+((k-1)/n)]}$$
$$> 1 - e^{-(k \times (k-1))/2n}$$

Ora: qual è il valore di k tale che P(n, k) = 0,5? Per soddisfare il requisito si ha:

$$1/2 = 1 - e^{-(k \times (k-1))/2n}$$
$$2 = e^{(k \times (k-1))/2n}$$
$$\ln(2) = \frac{k \times (k-1)}{2n}$$

Per k di grandi dimensioni si può sostituire $k \times (k - 1)$ con k^2 ottenendo:

$$k = \sqrt{2(\ln 2)n} = 1{,}18\sqrt{n} \approx \sqrt{n} \tag{11.6}$$

Per verificarlo, per $n = 365$, si avrà $k = 1{,}18 \times \sqrt{365} = 22{,}54$ che è molto vicino alla risposta corretta, ovvero 23.

Ora si può stabilire la base dell'attacco a compleanno nei seguenti termini. Si supponga di avere una funzione H con 2^m possibili output (ovvero un output di m bit). Se H è applicata a k input casuali, quale deve essere il valore di k tale che la probabilità di almeno un duplicato (H(x) = H(y) per qualche input x, y) sia > 0,5? Utilizzando l'approssimazione nell'equazione 11.6:

$$k = \sqrt{2^m} = 2^{m/2} \tag{11.7}$$

Sovrapposizione fra due insiemi

Vi è un problema correlato al caso generale delle duplicazioni che è altrettanto importante per la discussione. Il problema è il seguente: data una variabile casuale intera con una distribuzione uniforme fra 1 e n e due insiemi di k istanze ($k \le n$) della variabile casuale,

qual è la probabilità, R(n , k), che i due insiemi non siano disgiunti; ovvero qual è la probabilità che vi sia almeno un valore presente in entrambi gli insiemi?

Si definiscano i due insiemi X e Y con gli elementi $\{x_1, x_2, \ldots, x_k\}$ e $\{y_1, y_2, \ldots, y_k\}$. Dato il valore di x_1, la probabilità che $y_1 = x_1$ è semplicemente $1/n$ e pertanto la probabilità che y_1 non corrisponda a x_1 è $[1 - (1/n)]$. Se si generano k valori casuali di Y, la probabilità che nessuno di questi valori sia uguale a x_1 è $[1 - (1/n)]^k$. Pertanto la probabilità che vi sia almeno una corrispondenza con x_1 è $1 - [1 - (1/n)]^k$.

Per procedere si supponga che tutti gli elementi di X si siano distinti. Se n è esteso e k è anch'esso esteso (dell'ordine di $\sqrt{n}$), si tratta di una buona approssimazione. In realtà vi possono essere alcune duplicazioni ma la maggior parte dei valori saranno distinti. Data questa supposizione si possono creare le seguenti derivazioni:

$$\Pr[\text{nessuna corrispondenza in Y con } x_1] = \left(1 - \frac{1}{n}\right)^k$$

$$\Pr[\text{nessuna corrispondenza in Y con X}] = \left(\left(1 - \frac{1}{n}\right)^k\right)^k = \left(1 - \frac{1}{n}\right)^{k^2}$$

$$R(n,k) = \Pr[\text{almeno una corrispondenza in Y con X}] = 1 - \left(1 - \frac{1}{n}\right)^{k^2}$$

Utilizzando la disuguaglianza dell'Equazione 11.4:

$$R(n,k) > 1 - (e^{-1/n})^{k^2}$$

$$R(n,k) > 1 - (e^{-k^2/n})$$

Ora, per quale valore di k vale $R(n, k) > 0{,}5$? Si avrà:

$$1/2 = 1 - (e^{-k^2/n})$$

$$2 = e^{k^2/n}$$

$$\ln(2) = \frac{k^2}{n}$$

$$k = \sqrt{(\ln(2))n} = 0{,}83\sqrt{n} \approx \sqrt{n} \qquad (11.8)$$

Si può stabilire questo fatto in termini correlati all'attacco a compleanno nel modo seguente. Si supponga di avere una funzione H con 2^m possibili output (per esempio un output di m bit). Si applichi H a k input casuali per produrre l'insieme X e nuovamente ad altri k input casuali per produrre l'insieme Y. Quale deve essere il valore di k tale che vi sia la probabilità di almeno 0,5 che vi sia una corrispondenza fra i due insiemi (ovvero $H(x) = H(y)$ per qualche $x \in X$, $y \in Y$)? Utilizzando l'approssimazione dell'Equazione 11.8 si ha

$$k = \sqrt{2^m} = 2^{m/2}$$

Capitolo 12

Algoritmi hash e MAC

Concetti essenziali

- Praticamente tutti gli algoritmi **hash** sicuri rispettano la struttura generale illustrata nella Figura 11.9.
- La **funzione di compressione** utilizzata negli algoritmi hash sicuri può essere di due tipi: o una funzione appositamente progettata per la funzione hash (per esempio SHA) o una cifratura simmetrica a blocchi (come Whirlpool).
- Anche i **codici di autenticazione MAC** ricadono in due possibili categorie: quelli che utilizzano un algoritmo hash sicuro (per esempio HMAC) e quelli che utilizzano una cifratura simmetrica a blocchi (come CMAC).

In questo capitolo si esaminano alcuni esempi notevoli di algoritmi hash sicuri e di codici di autenticazione MAC. Le principali funzioni hash più recenti seguono la struttura di base schematizzata nella Figura 11.9, che si è dimostrata fondamentalmente solida: i nuovi progetti non fanno altro che raffinarla e incrementare la lunghezza del codice hash. Nel contesto di questa struttura di base, sono stati seguiti due approcci nella progettazione della funzione di compressione, che costituisce l'elemento base fondamentale della funzione hash. Nel passato, la maggior parte delle funzioni hash più diffuse si è affidata a una funzione di compressione appositamente progettata per la particolare funzione hash. Solitamente questa funzione di compressione utilizza l'aritmetica modulare e le operazioni logiche binarie. Il secondo approccio consiste nell'utilizzare una cifratura a blocchi simmetrica come funzione di compressione. In questo capitolo si esamina l'esempio forse più importante di ciascun approccio: SHA (Secure Hash Algorithm) e Whirlpool.

Anche i codici MAC possono essere catalogati in due classi sulla base del loro elemento costitutivo fondamentale. Un approccio diffuso consiste nell'utilizzare un algoritmo hash, per esempio SHA, come nucleo fondamentale dell'algoritmo MAC. Un secondo approccio consiste nell'utilizzare una cifratura simmetrica a blocchi in modalità CBC. Anche in questo caso si esamina l'esempio forse più importante di ciascun approccio: HMAC e CMAC.

12.1 L'algoritmo SHA

L'algoritmo SHA (Secure Hash AIgorithm) è stato sviluppato dal NIST (National Institute of Standards and Technology) e pubblicato nel 1993 come standard FIPS 180 (Federal Information Processing Standard); nel 1995 è stata emessa una versione aggiornata, FIPS 180-1, chiamata normalmente SHA-1. Il documento di standardizzazione è in realtà intitolato Secure Hash Standard. SHA si basa sulla funzione hash MD4 e in effetti replica da vicino la struttura di MD4. SHA-1 è anche specificato nel documento RFC 3174 che fondamentalmente riutilizza il materiale contenuto in FIPS 180-1 cui aggiunge un'implementazione in codice C.

SHA-1 produce un valore hash di 160 bit. NIST ha emesso nel 2002 una versione aggiornata dello standard, FIPS 180-2, che definisce tre nuove versioni di SHA con lunghezze del valore hash di 256, 384 e 512 bit, indicate con SHA-256, SHA-384 e SHA-512 (Tabella 12.1).

Tabella 12.1 Confronto dei parametri SHA.

	SHA-1	SHA-256	SHA-384	SHA-512
Lunghezza del codice digest	160	256	384	512
Lunghezza del messaggio	$<2^{64}$	$<2^{64}$	$<2^{128}$	$<2^{128}$
Dimensioni del blocco	512	512	1024	1024
Lunghezza della parola	32	32	64	64
Numero di passi	80	64	80	80
Sicurezza	80	128	192	256

Note: 1. Le dimensioni sono indicate in bit.
2. Il parametro *sicurezza* si riferisce al fatto che un attacco a compleanno su un codice digest di dimensione *n* genera una collisione con una complessità dell'ordine di $2^{n/2}$.

Queste nuove versioni hanno la medesima struttura sottostante e utilizzano gli stessi tipi di aritmetica modulare e operazioni logiche binarie di SHA-1. Nel 2005 NIST ha annunciato l'intenzione di abolire gradualmente SHA-1 per passare alle altre versioni SHA entro il 2010. Poco tempo dopo un gruppo di ricercatori ha descritto un attacco per individuare due messaggi separati che producono lo stesso codice hash SHA-1 con 2^{69} operazioni, molte meno delle 2^{80} che si credevano in precedenza necessarie per trovare una collisione con SHA-1 [WANG05]. Questo risultato dovrebbe affrettare la transizione alle altre versioni di SHA.

In questo paragrafo si descrive SHA-512: le altre versioni sono molto simili.

La logica di funzionamento di SHA-512

L'algoritmo prende come input un messaggio con una lunghezza massima di meno di 2^{128} bit e produce in output un codice digest di 512 bit. L'input viene elaborato in blocchi di 1024 bit.

La Figura 12.1 schematizza l'elaborazione generale di un messaggio per ottenere il codice digest corrispondente. L'elaborazione segue la struttura generale illustrata nella Figura 11.9 ed è costituita dai seguenti passi.

- **Passo 1: aggiunta dei bit di riempimento.** Il messaggio viene esteso in modo che la sua lunghezza sia congruente a 896 modulo 1024 [lunghezza ≡ 896 (mod 1024)]. Questi bit vengono sempre aggiunti, anche se il messaggio è già della lunghezza desiderata. Pertanto il numero di bit di riempimento sarà sempre compreso fra 1 e 1024. I bit di riempimento sono costituiti da un bit a "1" seguito da una sequenza di bit a "0".
- **Passo 2: aggiunta della lunghezza.** Al messaggio viene aggiunto un blocco di 128 bit che viene trattato come un intero senza segno di 128 bit (per primo viene il byte più significativo) e contenente la lunghezza del messaggio originario (prima dell'operazione di riempimento).
  Il risultato dei primi due passi consiste in un messaggio di lunghezza multipla di 1024 bit. Il messaggio espanso è rappresentato nella Figura 12.1 come sequenza di blocchi di 1024 bit $M_1, M_2, \ldots, M_N$, in modo che la lunghezza totale del messaggio sia N volte 1024 bit.
- **Passo 3: inizializzazione del buffer hash.** Per contenere i risultati intermedi e finale della funzione hash viene utilizzato un buffer di 512 bit. Il buffer può essere rappresen-

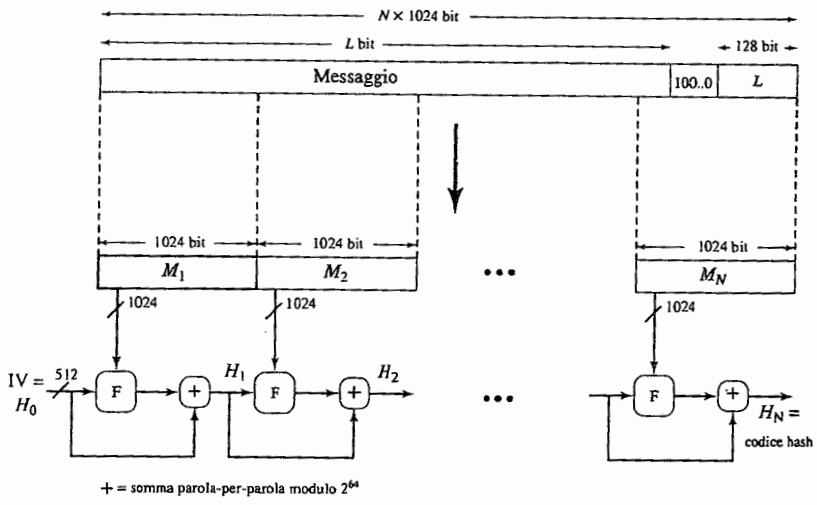


Figura 12.1 Generazione del codice digest utilizzando SHA-512.

tato come un insieme di otto registri da 64 bit (A, B, C, D, E, F, G, H). Questi registri vengono inizializzati con i seguenti valori a 64 bit (esadecimali):

A = 6A09E667F3BCC908
B = BB67AE8584CAA73B
C = 3CEF372FE94F82B
D = A54FF53A5F1D36F1
E = 510E527FADE682D1
F = 9B05688C2B3E6C1F
G = 1F83D9ABFB41BD6B
H = 5BE0CD19137E2179

Questi valori vengono memorizzati in formato big-endian, ovvero il byte più significativo di una word si trova nell'indirizzo più basso. Le parole sono state generate prendendo i primi 64 bit della parte frazionaria della radice quadrata dei primi otto numeri primi.

- **Passo 4: elaborazione del messaggio in blocchi di 1024 bit (128 word).** Il cuore dell'algoritmo è costituito da 80 fasi (modulo F nella Figura 12.1). La logica è illustrata nella Figura 12.2.
  Ogni fase prende come input il valore ABCDEFGH del buffer a 512 bit e aggiorna il contenuto del buffer. All'ingresso della prima fase, il buffer contiene il valore hash intermedio H_{i-1}. Ogni fase t utilizza il valore W_t di 64 bit derivato dal blocco di 1024 bit in elaborazione (M_i). La tecnica utilizzata per determinare tale valore verrà descritta successivamente.
  Ogni fase utilizza anche una costante additiva K_t dove $0 \le t \le 79$ indica una delle 80 fasi. Queste parole rappresentano i primi 64 bit della parte frazionaria della radice cubica dei primi ottanta numeri primi. Tali costanti costituiscono un insieme di configurazioni di 64 bit del tutto "casuali", che dovrebbero eliminare qualsiasi regolarità nei dati di ingresso.
  L'output della ottantesima fase viene aggiunto all'input della prima fase (H_{i-1}) per produrre H_i. La somma viene svolta in modo indipendente per ognuna delle otto word del buffer con ognuna delle corrispondenti word di H_{i-1}, utilizzando la somma modulo 2^{64}.
- **Passo 5: output.** Dopo che sono stati elaborati tutti gli N blocchi da 1024 bit, l'output dello stadio N-esimo è il codice digest di 512 bit.

Si può riassumere il comportamento di SHA-512 nel seguente modo:

$$H_0 = \text{IV}$$
$$H_i = \text{SUM}_{64}(H_{i-1}, \text{ABCDEFGH}_i)$$
$$MD = H_N$$

dove:

IV = valore iniziale del buffer ABCDEFGH, definito nel passo 3.
ABCDEFGH_i = output dell'ultima fase di elaborazione del i-esimo blocco del messaggio.
N = numero di blocchi del messaggio (inclusi i campi di riempimento e di lunghezza).

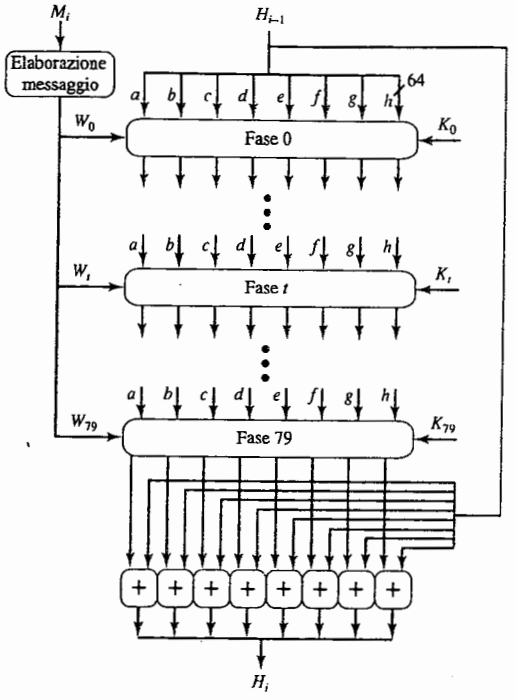


Figura 12.2 Eloborazione SHA-512 di un singolo blocco di 1024 bit.

SUM_{64} = somma modulo 2^{64} eseguita separatamente su ciascuna word della coppia di input.
MD = valore finale del codice digest.

La funzione di fase di SHA-512

Ora si vedrà più in dettaglio la logica di ognuno degli 80 passi di elaborazione di un blocco di 512 bit (Figura 12.3). Ogni fase è definita dal seguente insieme di equazioni.

$$T_1 = h + Ch(e,f,g) + (\sum_1^{512} e) + W_t + K_t$$
$$T_2 = (\sum_0^{512} a) + Maj(a,b,c)$$
$$a = T_1 + T_2$$
$$b = a$$
$$c = b$$
$$d = c$$

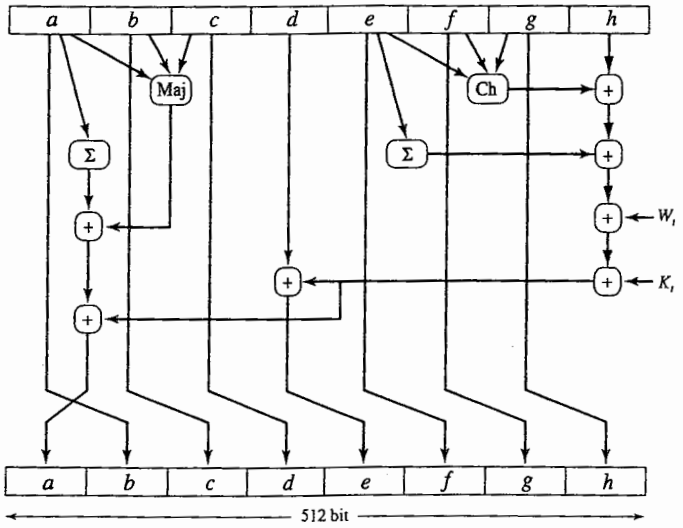


Figura 12.3 Operazione elementare in SHA-512 (singola fase).

$$e = d + T_1$$
$$f = e$$
$$g = f$$
$$h = g$$

dove:

t	= numero della fase, $0 \le t \le 79$
$Ch(e, f, g)$	= (e AND f) ⊕ (NOT e AND g) *la funzione condizionale: If e then f else g.*
Maj (a, b, c)	= (a AND b) ⊕ (a AND c) ⊕ (b AND c) *la funzione è vera solo se la maggioranza (due o tre) degli argomenti è vera.*
$(\sum_0^{512} a)$	= $\text{ROTR}^{28}(a) \oplus \text{ROTR}^{34}(a) \oplus \text{ROTR}^{39}(a)$
$(\sum_1^{512} e)$	= $\text{ROTR}^{14}(e) \oplus \text{ROTR}^{18}(e) \oplus \text{ROTR}^{41}(e)$
$\text{ROTR}^n(x)$	= rotazione circolare a destra di n bit dell'argomento x di 64 bit
W_t	= parola di 64 bit derivata dal blocco di ingresso di 512 bit in elaborazione
K_t	= costante additiva di 64 bit
+	= somma modulo 2^{64}

Rimane da indicare il modo in cui i valori W_t delle word a 64 bit vengono derivati dal messaggio a 1024 bit. La Figura 12.4 illustra il mapping eseguito. I primi 16 valori di W_t

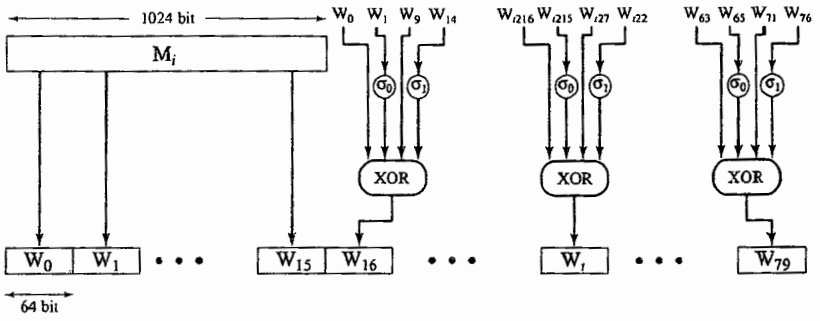


Figura 12.4 Generazione della sequenza di input di 80 parole per l'elaborazione SHA-512 di un singolo blocco.

vengono tratti direttamente dalle 16 word del blocco corrente. I valori rimanenti sono definiti nel seguente modo:

$$W_t = \sigma_1^{512}(W_{t-2}) + W_{t-7} + \sigma_0^{512}(W_{t-15}) + W_{t-16}$$

dove:

$\sigma_0^{512}(x)$	= $\text{ROTR}^1(x) \oplus \text{ROTR}^8(x) \oplus \text{SHR}^7(x)$
$\sigma_1^{512}(x)$	= $\text{ROTR}^{19}(x) \oplus \text{ROTR}^{61}(x) \oplus \text{SHR}^6(x)$
$\text{ROTR}^n(x)$	= rotazione circolare a destra di n bit dell'argomento x di 64 bit
$\text{SHR}^n(x)$	= scorrimento a sinistra di n bit dell'argomento x di 64 bit, con riempimento sulla destra di valori zero.

Pertanto, nei primi 16 passi di elaborazione, il valore di W_t è uguale alla word corrispondente nel blocco del messaggio. Per i rimanenti 64 passi, il valore di W_t è costituito da uno scorrimento circolare a sinistra di un bit dell'operazione di XOR su quattro dei precedenti valori di W_t, due dei quali sottoposti a operazioni di shift e rotazione. Questo introduce una notevole ridondanza e interdipendenza nei blocchi del messaggio che vengono compressi, complicando l'individuazione di un blocco del messaggio che produca lo stesso output della funzione di compressione.

12.2 Whirlpool[1]

In questo paragrafo si esamina la funzione hash Whirlpool [BARR03], uno dei cui progettisti è anche co-inventore di Rijndael, adottato come Advanced Encryption Standard (AES). Whirlpool è una delle due sole funzioni hash approvate da NESSIE (New European Schemes for Signature, Integrity, and Encryption, o "Nuovi schemi europei di firma, integrità e crittografia")[2]. Il progetto NESSIE è sponsorizzato dall'Unione Europea con l'obiettivo di identificare un portafoglio di vari tipi di primitive crittografiche robuste.

[1] La maggior parte del materiale in questo paragrafo è stato pubblicato originariamente in [STAL06].
[2] L'altro schema approvato è costituito dalle tre varianti di SHA: SHA-256, SHA-384 e SHA-512.

Whirlpool si basa sull'impiego di una cifratura a blocchi come funzione di compressione. Come menzionato nel Capitolo 11, vi è stato un interesse limitato nell'uso di funzioni hash basate sulla cifratura a blocchi a causa della vulnerabilità della struttura. Esistono i seguenti potenziali svantaggi.

1. Gli algoritmi di cifratura a blocchi non godono delle proprietà delle funzioni che generano numeri casuali: per esempio sono invertibili. Questa mancanza di casualità potrebbe comportare delle debolezze potenzialmente sfruttabili.
2. Le cifratura a blocchi hanno solitamente altre regolarità o punti deboli. Per esempio, [MIYA90] dimostra come compromettere numerosi schemi hash sulla base delle proprietà della cifratura a blocchi utilizzata.
3. Solitamente, le funzioni hash basate sulle cifrature a blocchi sono significativamente più lente delle funzioni hash basate su una funzione di compressione appositamente progettata per la funzione hash.
4. Un indicatore fondamentale della robustezza di una funzione hash è la lunghezza in bit del codice hash. Le proposte di codici hash basati su cifrature a blocchi sono caratterizzate da una lunghezza del codice hash pari alla dimensione di blocco della cifratura o al suo doppio. Tradizionalmente la lunghezza di blocco delle cifrature si è limitata a 64 bit (per esempio DES e triple DES), comportando un codice hash di dubbia robustezza.

Tuttavia, dopo l'adozione di AES vi è stato un rinnovato interesse a sviluppare una funzione hash sicura basata su un algoritmo di cifratura a blocchi robusto e caratterizzato da buone prestazioni. Whirlpool è una funzione hash basata sulla cifratura a blocchi capace di fornire livelli di sicurezza e prestazioni comparabili, se non addirittura migliori, a quelli di funzioni hash non basate su cifrature a blocchi, come SHA. Whirlpool ha le seguenti caratteristiche.

1. La dimensione del codice hash è di 512 bit, equivalente al codice più lungo disponibile con SHA.
2. Si è dimostrato che la struttura complessiva della funzione hash è resistente agli attacchi tipici contro i codici hash basati su cifrature a blocchi.
3. La cifratura a blocchi sottostante è basata su AES ed è progettata per poter essere implementata sia in software che con hardware, in modo compatto e con buone prestazioni.

Il progetto di Whirlpool stabilisce i seguenti obiettivi di sicurezza; si supponga di prendere come risultato hash il valore di qualsiasi sottostringa di n bit dell'output complessivo di Whirlpool.

- Il carico computazionale atteso per generare una collisione è dell'ordine di $2^{n/2}$ esecuzioni di Whirlpool.
- Dato un valore di n bit, il carico computazionale atteso per trovare un messaggio il cui codice hash corrisponda a tale valore è dell'ordine di 2^n esecuzioni di Whirlpool.
- Dato un messaggio e il corrispondente codice hash di n bit, il carico computazionale atteso per trovare un secondo messaggio con lo stesso codice hash è dell'ordine di 2^n esecuzioni di Whirlpool.
- Non è praticamente possibile identificare correlazioni sistematiche fra qualsiasi combinazione lineare di bit di ingresso e qualsiasi combinazione di bit del risultato hash, o predire quali bit del risultato hash cambino all'invertire determinati bit di input (questo significa resistenza agli attacchi lineari e differenziali).

I progettisti esprimono completa fiducia nel fatto che questi obiettivi siano stati soddisfatti con ampi margini di sicurezza. Ciò nonostante non è possibile dimostrarlo formalmente.

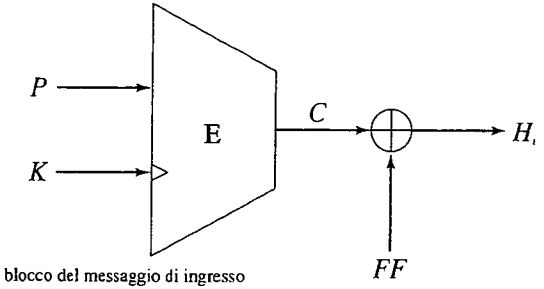
Si inizia con l'esame della struttura della funzione hash complessiva, per poi esaminare la cifratura a blocchi utilizzata come elemento di base.

Struttura hash di Whirlpool

Informazioni preliminari

La struttura generale hash iterativa proposta da Merkle (Figura 11.9) è utilizzata praticamente in tutte le funzioni hash sicure. Tuttavia, come già discusso, vi sono delle difficoltà nel progettare una funzione hash iterativa realmente sicura quando la funzione di compressione è costituita da una cifratura a blocchi. Prenel [PREN93a, PREN93b] ha eseguito un'analisi sistematica delle funzioni hash basate su cifrature a blocchi, usando il modello schematizzato nella Figura 12.5.

In questo modello la lunghezza del codice hash è uguale alla dimensione del blocco della cifratura. Se la lunghezza del codice hash fosse superiore alla dimensione del blocco della cifratura, si introdurrebbero ulteriori problemi di sicurezza e l'analisi si complica. Prenel ha considerato 64 possibili permutazioni del modello di base, che dipendono da quale input viene utilizzato come chiave di crittografia e quale viene utilizzato come testo in chiaro, e a seconda di quale input, eventualmente, viene combinato con il testo cifrato per produrre il codice hash intermedio. Sulla base di questa analisi ha concluso che solo gli schemi nei quali il testo in chiaro viene reintrodotto all'uscita di ogni iterazione (FF, o *feed forward*) e combinato con il testo cifrato sono sicuri. Tale configurazione rende la funzione



m_i = i-esimo blocco del messaggio di ingresso
H_i = i-esimo valore hash intermedio
P = testo in chiaro; K = chiave di crittografia; C = testo cifrato
FF = valore feed forward
P, K e FF possono essere scelti dall'insieme $(0, m_i, H_{i-1}, m_i \oplus H_{i-1})$

Nota: i simboli triangolari indicano l'ingresso della chiave di crittografia.

Figura 12.5 Modello di una singola iterazione della funzione hash (lunghezze del codice hash e del blocco identiche).

di compressione difficile da invertire. [BLAC02] ha confermato questi risultati, ma ha individuato un problema di sicurezza derivato dall'utilizzo di una cifratura a blocchi esistente come AES. Il codice hash a 128 bit risultante dall'impiego di AES o di altro schema con la medesima dimensione del blocco potrebbe non essere sufficientemente sicuro.

La logica di Whirlpool

Dato un messaggio composto da una sequenza di blocchi $m_1, m_2, \ldots, m_t$, la funzione hash Whirlpool è espressa nel modo seguente:

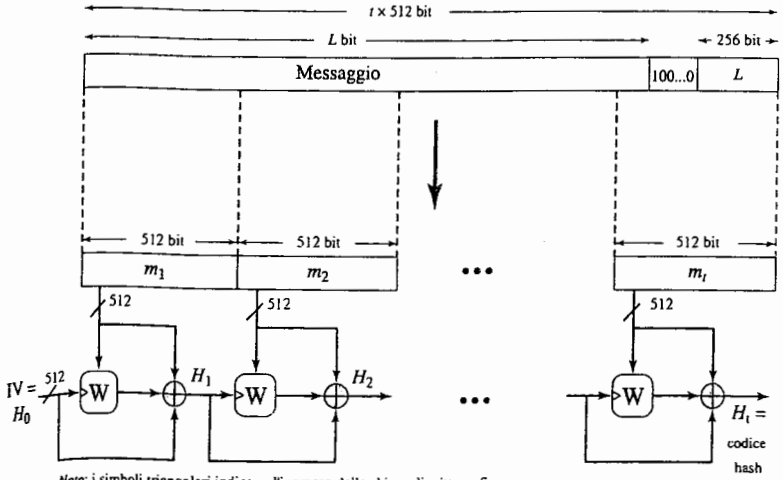
H_0 = valore iniziale
$H_i = E(H_{i-1}, m_i) \oplus H_{i-1} \oplus m_i$ = valore intermedio
H_t = valore del codice hash

In termini del modello di Figura 12.5, la chiave di crittografia di ingresso a ogni iterazione è il valore hash intermedio risultante dall'iterazione precedente, il testo in chiaro corrisponde al blocco del messaggio corrente e il valore FF è il risultato dello XOR bit-a-bit fra il blocco del messaggio corrente e il valore hash intermedio dall'iterazione precedente.

L'algoritmo riceve in ingresso un messaggio con una lunghezza massima minore di 2^{256} bit e produce in uscita un codice digest di 512 bit. L'input è elaborato in blocchi di 512 bit. La Figura 12.6 illustra l'elaborazione generale di un messaggio per produrre il digest. Questo segue la struttura generale schematizzata nella Figura 11.9.

L'elaborazione consiste nei passi seguenti.

- **Passo 1: aggiunta dei bit di riempimento.** Vengono aggiunti bit di riempimento in coda al messaggio in modo che la sua lunghezza in bit sia un multiplo dispari di 256.



Nota: i simboli triangolari indicano l'ingresso della chiave di crittografia.

Figura 12.6 Generazione del codice digest utilizzando Whirlpool.

Questa operazione viene sempre effettuata, anche nel caso in cui il messaggio sia della lunghezza desiderata. Per esempio, se il messaggio fosse lungo 256 × 3 = 768 bit, verrebbero aggiunti 512 bit di riempimento per raggiungere la lunghezza di 256 × 5 = 1280 bit. Il numero di bit di riempimento varia dunque da 1 a 512.

Il riempimento consiste in un singolo bit di valore 1 seguito dal numero necessario di bit 0.

- **Passo 2: accodamento della lunghezza.** Viene aggiunto in coda al messaggio un blocco di 256 bit. Questo blocco viene interpretato come intero senza segno di 256 bit (iniziando con il byte più significativo) e indica la lunghezza in bit del messaggio originale (prima del riempimento).

  Il risultato dei primi due passi genera un messaggio la cui lunghezza è un multiplo intero di 512. Nella Figura 12.6 il messaggio espanso è rappresentato dalla sequenza di blocchi di 512 bit $m_1, m_2, \ldots, m_t$; la lunghezza totale del messaggio espanso è dunque t × 512 bit. Questi blocchi sono visti esternamente come array di byte, raggruppando sequenzialmente i bit in gruppi di otto. Internamente, tuttavia, lo stato hash H_i è visto come matrice di 8 × 8 byte. La trasformazione fra i due verrà indicata successivamente.
- **Passo3: inizializzazione della matrice hash.** Per memorizzare i risultati intermedio e finale della funzione hash viene utilizzata una matrice di 8 × 8 byte, inizializzata con valori nulli.
- **Passo 4: elaborazione del messaggio in blocchi di 512 bit (64 byte).** Il cuore dell'algoritmo è costituito dalla cifratura a blocchi W.

Algoritmo di cifratura a blocchi W

A differenza delle altre proposte di funzioni hash basate su cifrature a blocchi, Whirlpool utilizza una cifratura a blocchi appositamente progettata per essere utilizzata nella funzione hash e che difficilmente verrà mai utilizzata come funzione di crittografia indipendente. Il motivo risiede nel fatto che i progettisti hanno voluto utilizzare una cifratura a blocchi con la sicurezza e l'efficienza di AES, ma con una lunghezza hash tale da fornire una sicurezza equivalente a SHA-512. Il risultato è la cifratura a blocchi W, che ha una struttura simile e utilizza le medesime funzioni elementari di AES, ma che utilizza dimensioni del blocco e lunghezza della chiave di 512 bit. La Tabella 12.2 confronta AES e W.

Sebbene W sia simile ad AES, non è semplicemente una sua estensione. Si ricordi che la proposta Rijndael per AES definiva una cifratura nella quale la lunghezza del blocco e la lunghezza della chiave potevano essere specificate in modo indipendente con valore di 128, 192 o 256 bit. La specifica di AES utilizza le stesse alternative per la dimensione della chiave ma limita la lunghezza del blocco a 128 bit. AES opera su uno stato di 4 × 4 byte. Rijndael con lunghezza di blocco di 192 bit lavora con uno stato di 4 × 6 byte, con lunghezza di blocco di 256 bit opera con uno stato di 4 × 8 byte. W lavora con uno stato di 8 × 8 byte. Più la rappresentazione dello stato differisce da un quadrato, più lenta diviene la diffusione rendendo necessarie più fasi nella cifratura. Con una lunghezza di blocco di 512 bit, i progettisti di Whirlpool potrebbero aver definito un Rijndael operante su uno stato di 4 × 16 byte, ma tale cifratura avrebbe richiesto molte fasi e sarebbe risultata molto lenta.

Tabella 12.2 Confronto fra le cifrature a blocchi W di Whirlpool e AES.

	W	AES
Dimensioni del blocco (bit)	512	128
Lunghezza della chiave (bit)	512	128, 192 o 256
Orientazione della matrice	L'input viene organizzato per righe	L'input viene organizzato per colonne
Numero di fasi	10	10, 12 o 14
Espansione della chiave	Funzione di fase W	Algoritmo di espansione dedicato
Polinomio in GF(2^8)	$x^8 + x^4 + x^3 + x^2 + 1$ (011D)	$x^8 + x^4 + x^3 + x^2 + 1$ (011B)
Origine della S-box	Struttura ricorsiva	Inverso moltiplicativo in GF(2^8) più trasformazione affine
Origine delle costanti di fase	Entry successive della S-box	Elementi 2^i in GF(2^8)
Livello di diffusione	Moltiplicazione destra con matrice MDS circolante 8 × 8 (1, 1, 4, 1, 8, 5, 2, 9) – miscelazione righe	Moltiplicazione sinistra con matrice MDS circolante 4 × 4 (2, 3, 1, 1) – miscelazione colonne
Permutazione	Scorrimento colonne	Scorrimento righe

Come indicato nella Tabella 12.2, W utilizza una matrice orientata per righe mentre AES utilizza una matrice orientata per colonne. Non vi è alcuna ragione tecnica per preferire un'orientazione rispetto all'altra, dato che è facile costruire una descrizione equivalente della medesima cifratura, scambiando le righe con le colonne.

Struttura generale

La Figura 12.7 illustra la struttura generale di W.

L'algoritmo di crittografia riceve in ingresso un blocco di 512 bit di testo in chiaro e una chiave di 512 bit, per produrre in uscita un blocco di 512 bit di testo cifrato. L'algoritmo di crittografia comporta l'impiego di quattro funzioni, o trasformazioni, differenti: Add Key (AK), Substitute Bytes (SB), Shift Columns (SC) e Mix Rows (MR), il cui funzionamento verrà presto spiegato. W prevede una singola applicazione di AK seguita da dieci fasi che utilizzano tutte e quattro le funzioni. Si può esprimere concisamente il funzionamento di una singola fase r come funzione di fase RF composta:

$$\mathrm{RF}(\boldsymbol{K}_r) = \mathrm{AK}[\boldsymbol{K}_r] \circ \mathrm{MR} \circ \mathrm{SC} \circ \mathrm{SB} \qquad (12.1)$$

dove $\boldsymbol{K}_r$ è la matrice della chiave della fase r. L'algoritmo generale, con chiave di ingresso **K**, può essere definito nel modo seguente:

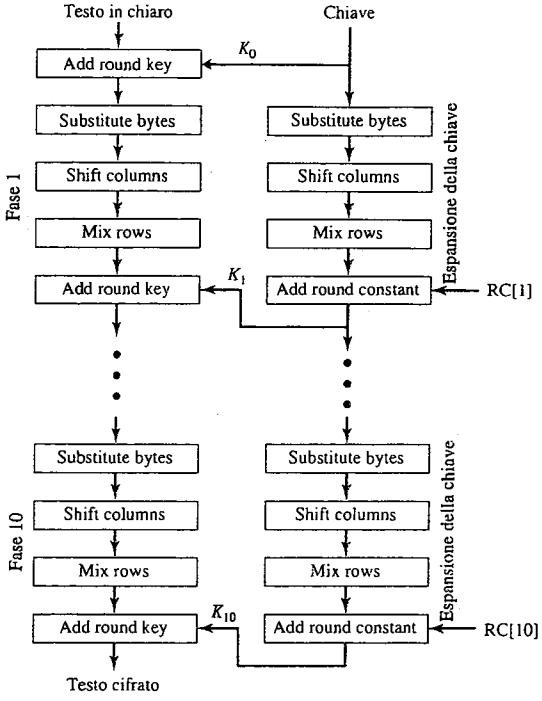


Figura 12.7 La cifratura W di Whirlpool.

$$W(K) = \left(\bigcirc_{r=1}^{10} RF(\boldsymbol{K}_r) \circ AK(\boldsymbol{K}_0) \right)$$

dove O indica l'iterazione della funzione composta con indice r che varia da 1 a 10.

Il testo in chiaro in ingresso a W è un singolo blocco di 512 bit. Questo blocco viene trattato come una matrice quadrata di 8 × 8 byte, chiamata **CState**. La Figura 12.8 indica che l'ordinamento dei byte nella matrice è per riga. Quindi, per esempio, i primi otto dei 512 byte di testo in chiaro in ingresso alla cifratura occupano la prima riga della matrice interna **CState**, i secondi otto byte occupano la seconda riga e via dicendo.

La rappresentazione del flusso lineare di byte come matrice quadrata può essere espressa concisamente come funzione di mapping. Dato un array lineare X di byte con elementi x_k ($0 \le k \le 63$), la matrice corrispondente **A** con elementi a_{ij} ($0 \le i,j \le 7$) è così definita:

$$\mathbf{A} = \mu(\mathrm{X}) \Leftrightarrow a_{i,j} = x_{8i+j}$$

Analogamente, la chiave di 512 bit viene interpretata come matrice quadrata di byte **KState**. Questa chiave è utilizzata come input alla funzione AK iniziale. La chiave è inoltre espansa in un insieme di 10 chiavi di fase, come successivamente esposto.

Si esaminano ora le specifiche funzioni che fanno parte di W.

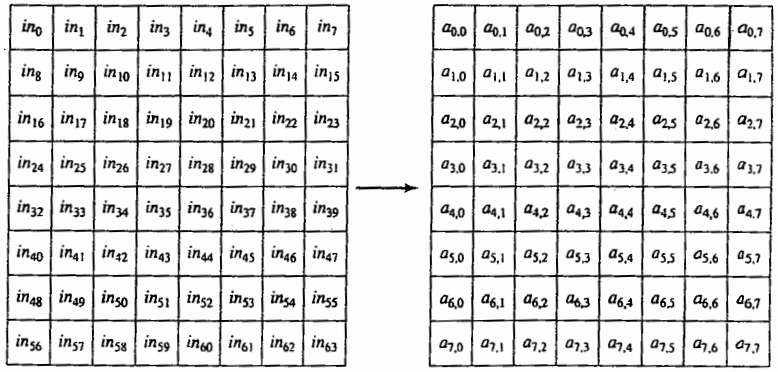


Figura 12.8 Struttura della matrice di Whirlpool.

Livello non lineare SB

La funzione SB (Substitute Byte) è una semplice tabella di lookup che fornisce un mapping non lineare. W definisce una matrice di 16 × 16 byte, chiamata S-box (Tabella 12.3), che contiene una permutazione di tutti i possibili 256 valori di 8 bit.

Ciascun byte di **CState** viene mappato in un nuovo byte nel modo seguente. I primi 4 bit più a sinistra del byte sono utilizzati come valore di riga e i 4 bit più a destra come valore di colonna. Questi due valori vengono impiegati come indici nella S-box per selezionare un valore unico di 8 bit in uscita. Per esempio, il valore esadecimale[3] {95} si riferisce al byte di riga 9 e colonna 5 della S-box, che contiene il valore {BA}. La funzione SB può essere espressa dalla seguente corrispondenza, dove **A** indica la matrice di input e **B** quella di output:

$$\mathbf{B} = \mathrm{SB}(\mathrm{A}) \Leftrightarrow b_{ij} = \mathrm{S}[a_{ij}],\ 0 \le i,j \le 7$$

S[x] indica il mapping del byte di input x nel byte di output S[x] tramite la S-box.

La S-box può essere generata dalla struttura della Figura 12.9.

Essa consiste in due livelli non lineari, ciascuno dei quali contiene due S-box separate da una box 4 × 4 generata casualmente. Ciascuna box mappa l'ingresso di 4 bit nell'uscita di 4 bit. La box E è definita come $E(u) = \{B\}^u$ se $u \neq \{F\}$ ed $E(\{F\}) = 0$, con aritmetica calcolata sul campo finito GF(2^4) con polinomio irriducibile $f(x) = x^4 + x + 1$.

La funzione SB è progettata con lo scopo di introdurre la non-linearità nell'algoritmo. Questo significa che la funzione SB non dovrebbe presentare alcuna correlazione fra combinazioni lineari di bit di ingresso e combinazioni lineari di bit di uscita. Inoltre, le differenze fra insiemi di bit di ingresso non si dovrebbero propagare in differenze simili fra i corrispondenti bit di uscita. In altre parole, piccole variazioni nell'ingresso dovrebbero produrre grandi variazioni nell'uscita. Queste due proprietà contribuiscono a rendere W resistente agli attacchi ad analisi lineare e differenziale.

[3] Come già fatto nel caso di AES, i numeri esadecimali sono indicati fra parentesi graffe ogniqualvolta sia opportuno, per motivi di chiarezza.

Tabella 12.3 La S-box di Whirlpool.

(a) S-box

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	18	23	C6	E8	87	B8	01	4F	36	A6	D2	F5	79	6F	91	52
1	60	BC	9B	8E	A3	0C	7B	35	1D	E0	D7	C2	2E	4B	FE	57
2	15	77	37	E5	9F	F0	4A	CA	58	C9	29	0A	B1	A0	6B	85
3	BD	5D	10	F4	CB	3E	05	67	E4	27	41	8B	A7	7D	95	C8
4	FB	EE	7C	66	DD	17	47	9E	CA	2D	BF	07	AD	5A	83	33
5	63	02	AA	71	C8	19	49	C9	F2	E3	5B	88	9A	26	32	B0
6	E9	0F	D5	80	BE	CD	34	48	FF	7A	90	5F	20	68	1A	AE
7	B4	54	93	22	64	F1	73	12	40	08	C3	EC	DB	A1	8D	3D
8	97	00	CF	2B	76	82	D6	1B	B5	AF	6A	50	45	F3	30	EF
9	3F	55	A2	EA	65	BA	2F	C0	DE	1C	FD	4D	92	75	06	8A
A	B2	E6	0E	1F	62	D4	A8	96	F9	C5	25	59	84	72	39	4C
B	5E	78	38	8C	D1	A5	E2	61	B3	21	9C	1E	43	C7	FC	04
C	51	99	6D	0D	FA	DF	7E	24	3B	AB	CE	11	8F	4E	B7	EB
D	3C	81	94	F7	B9	13	2C	D3	E7	6E	C4	03	56	44	7F	A9
E	2A	BB	C1	53	DC	0B	9D	6C	31	74	F6	46	AC	89	14	E1
F	16	3A	69	09	70	B6	C0	ED	CC	42	98	A4	28	5C	F8	86

(b) Mini-box E

u	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$E(u)$	1	B	9	C	D	6	F	3	E	8	7	4	A	2	5	0

(c) Mini-box E^{-1}

u	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$E^{-1}(u)$	F	0	D	7	B	E	5	A	9	2	C	1	3	4	8	6

(d) Mini-box R

u	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$R(u)$	7	C	B	D	E	4	9	F	6	3	8	A	2	5	1	0

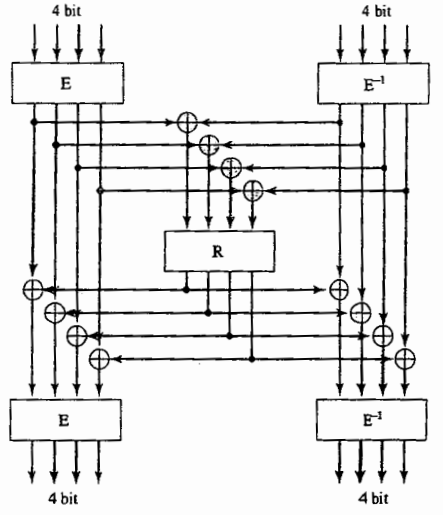


Figura 12.9 Implementazione della S-box Whirlpool.

Livello di permutazione SC

Il livello di permutazione (Shift Columns) esegue una rotazione circolare verso il basso di ciascuna colonna di **CState**, ad eccezione della prima colonna. La terza colonna viene ruotata di un byte, la terza colonna di due byte e così via. La funzione SC può essere espressa tramite la seguente corrispondenza, dove **A** indica la matrice di input e **B** quella di output:

$$\mathbf{B} = \mathrm{SC}(\mathbf{A}) \Leftrightarrow b_{ij} = a_{(i-j) \bmod 8j}, \quad 0 \le i,j \le 7$$

La trasformazione Shift Columns è più sostanziale di quanto potrebbe apparire a prima vista. Infatti **CState** è trattato come un array bidimensionale o matrice di 8 righe di 8 byte. Dunque, nella crittografia, i primi 8 byte del testo in chiaro vengono copiati nella prima riga di **CState**, e così via. La rotazione delle colonne sposta ciascun byte da una riga a un'altra, distante un multiplo di 8 byte. Si noti anche che la trasformazione assicura che gli 8 byte di una riga siano dispersi in otto righe differenti.

Livello di diffusione MR

Si ricorderà dal Capitolo 3 che nelle funzioni caratterizzate dalla proprietà di diffusione, la struttura statistica dell'input viene diffusa in statistiche di lungo termine dell'output. Questo viene ottenuto facendo in modo che ogni bit dell'input influenzi il valore di molti bit dell'output. Il livello di diffusione (Mix Rows) ottiene la diffusione separatamente in ciascuna riga. Ogni byte di una riga viene mappato in un nuovo valore che è funzione di tutti gli otto byte della medesima riga. La trasformazione può essere definita tramite la moltiplicazione tra matrici $\mathbf{B} = \mathbf{AC}$, dove **A** indica la matrice di input, **B** quella di output e **C** è la seguente matrice di trasformazione:

$$C = \begin{pmatrix} 01 & 01 & 04 & 01 & 08 & 05 & 02 & 09 \\ 09 & 01 & 01 & 04 & 01 & 08 & 05 & 02 \\ 02 & 09 & 01 & 01 & 04 & 01 & 09 & 05 \\ 05 & 02 & 09 & 01 & 01 & 04 & 01 & 08 \\ 08 & 05 & 02 & 09 & 01 & 01 & 04 & 01 \\ 01 & 08 & 05 & 02 & 00 & 01 & 01 & 04 \\ 04 & 01 & 08 & 05 & 02 & 09 & 01 & 01 \\ 01 & 04 & 01 & 08 & 05 & 02 & 09 & 01 \end{pmatrix}$$

Ogni elemento nella matrice prodotto è la somma dei prodotti degli elementi di una riga e una colonna. In questo caso le somme e le moltiplicazioni[4] sono effettuate in GF(2^8) con il polinomio irriducibile $f(x) = x^8 + x^4 + x^3 + x^2 + 1$. Come esempio di calcolo relativo al prodotto di matrici, il primo elemento della matrice di uscita è:

$$b_{0,0} = a_{0,0} \oplus (9 \bullet a_{0,1}) \oplus (2 \bullet a_{0,2}) \oplus (5 \bullet a_{0,3}) \oplus (8 \bullet a_{0,4}) \oplus a_{0,5} \oplus (4 \bullet a_{0,6}) \oplus a_{0,7}$$

[4] Come già fatto per AES, si utilizzano il simbolo o per indicare la moltiplicazione nel campo finito GF(2^8) e il simbolo ⊕ per indicare lo XOR bit-a-bit, che corrisponde alla somma in GF(2^8).

Si noti che ciascuna riga di C viene costruita tramite lo scorrimento circolare a destra della riga precedente. **C** è progettata per essere una matrice **separabile a distanza massima** (MDS). Nell'ambito dei codici a correzione degli errori, un codice MDS riceve come input una stringa di bit di dimensione fissa e produce una stringa di uscita espansa tale da massimizzare la distanza di Hamming fra le coppie di stringhe di uscita. Con un codice MDS, anche errori multipli nei bit producono un codice più vicino al valore corretto che a qualsiasi altro valore. Nel contesto della cifratura a blocchi, una matrice di trasformazione costruita con un codice MDS fornisce un elevato livello di diffusione [JUN04]. L'uso dei codici MDS per ottenere una diffusione elevata è stato inizialmente proposto in [RIJM96].

La matrice C è una matrice MDS che ha il massimo numero possibile di elementi 1 (3 per riga). Complessivamente, gli elementi in C garantiscono un'implementazione hardware efficiente.

Livello Add Key (AK)

Nel livello Add Key, si calcola lo XOR bit-a-bit fra i 512 bit di **CState** e i 512 bit della chiave di fase. La funzione AK può essere espressa tramite la seguente corrispondenza, dove **A** indica la matrice di input, **B** quella di output e K_i la chiave di fase:

$$\mathbf{B} = \mathrm{AK}[K_i](\mathrm{A}) \Leftrightarrow b_{i,j} = a_{i,j} \oplus k_{i,j}, \; 0 \le i,j \le 7$$

Espansione della chiave per la cifratura a blocchi W

Come illustrato nella Figura 12.7, l'espansione della chiave è ottenuta utilizzando la cifratura a blocchi stessa, con una costante di fase che agisce da chiave di fase per l'espansione. La costante per la fase r ($1 \le r \le 10$) è una matrice $\mathbf{RC}[r]$ nella quale solo la prima riga non è nulla, ed è definita nel modo seguente:

$$\mathrm{rc}[r]_{0,j} = \mathrm{S}[8(r-1)+j], \quad 0 \le j \le 7, \; 1 \le r \le 10$$

$$\mathrm{rc}[r]_{i,j} = 0, \quad 1 \le i \le 7, \; 0 \le j \le 7, \; 1 \le r \le 10$$

Ogni elemento della prima riga è un mapping che utilizza la S-box. Quindi, la prima riga di $\mathbf{RC}[1]$ è:

S[0]	S[1]	S[2]	S[3]	S[4]	S[5]	S[6]	S[7]
=							
18	23	C6	E8	87	B8	01	4F

Utilizzando le costanti di fase, la chiave di cifratura K di 512 bit viene espansa nella sequenza di chiavi di fase $\mathbf{K}_0, \mathbf{K}_1, \ldots, \mathbf{K}_{10}$:

$$\mathbf{K}_0 = \boldsymbol{K}$$

$$\mathbf{K}_r = \mathrm{RF}[\mathbf{RC}[r]](\mathbf{K}_{r-1})$$

dove RF è la funzione di fase definita dall'Equazione 12.1. Si noti che nella trasformazione AK di ogni fase, viene alterata solamente la prima riga di **KState**.

Le prestazioni di Whirlpool

L'esperienza con implementazioni Whirlpool è a tutt'oggi limitata. Si ricordi che i criteri di valutazione NIST richiedevano che Rijndael esibisse buone prestazioni nelle implementazioni sia hardware che software e che fosse adatto a situazioni con limitata disponibilità di memoria. Questi criteri sono stati determinanti nella scelta di Rijndael per AES. Whirlpool utilizza gli stessi blocchi funzionali di AES e ha la medesima struttura: si può dunque ritenere che abbia prestazioni e caratteristiche di occupazione di memoria simili.

Un'analisi delle prestazioni di Whirlpool è riportata in [KITS04], dove gli autori confrontano Whirlpool con varie altre funzioni hash sicure, tra le quali tutte le versioni di SHA. Gli autori, dopo aver sviluppato più implementazioni hardware di ciascuna funzione hash, concludono che Whirlpool, rispetto a SHA-512, richiede più risorse hardware ma raggiunge prestazioni notevolmente superiori.

12.3 HMAC

Nel Capitolo 11, si è visto un esempio di codice MAC (Message Authentication Code) basato sull'uso della cifratura simmetrica a blocchi, in particolare l'algoritmo Data Authentication definito in FIPS PUB 113. Questo è stato tradizionalmente l'approccio più comune alla costruzione di un codice MAC. In anni più recenti, vi è stato un sempre maggiore interesse allo sviluppo di codici MAC derivati da una funzione hash crittografica. Ecco i motivi di questo interesse.

1. Le funzioni hash crittografiche come MD5 e SHA-1 sono generalmente più veloci in software rispetto alle cifrature simmetriche a blocchi come DES.
2. Sono ampiamente disponibili delle librerie per funzioni hash crittografiche.

Lo sviluppo di AES e la crescente disponibilità di codici di algoritmi di crittografia hanno reso meno significative queste considerazioni, ma i codici MAC basati su funzioni hash continuano a essere ampiamente utilizzati.

Una funzione hash come SHA non è stata progettata per essere utilizzata come codice MAC e non può essere utilizzata direttamente per tale scopo in quanto non si basa su una chiave segreta. Vi sono state numerose proposte per l'incorporazione di una chiave segreta in un algoritmo hash esistente. L'approccio che ha ricevuto il maggior sostegno è HMAC [BELL96a, BELL96b]. HMAC è stato emesso come documento RFC 2104 ed è stato scelto come codice MAC obbligatorio per la sicurezza IP e viene utilizzato anche in altri protocolli Internet come SSL. HMAC è anche stato emesso come standard FIPS (198).

Obiettivi progettuali di HMAC

Il documento RFC 2104 elenca i seguenti obiettivi progettuali per HMAC.

- Utilizzare, senza modifiche, le funzioni hash disponibili, in particolare quelle che manifestano un buon comportamento in software e il cui codice è liberamente e ampiamente disponibile.
- Consentire la facile sostituzione delle funzioni hash incorporate nel caso in cui venissero trovate o si rendessero necessarie funzioni hash più veloci o sicure.
- Garantire le prestazioni originarie delle funzioni hash senza introdurre significativi degradi prestazionali.
- Utilizzare e gestire le chiavi in modo semplice.
- Avere un'analisi crittografica della robustezza del meccanismo di autenticazione ben compresa sulla base di supposizioni ragionevoli riguardanti la funzione hash incorporata.

I primi due obiettivi sono importanti per l'accettabilità di HMAC. HMAC tratta la funzione hash come una sorta di "scatola nera". Questo presenta due vantaggi. Innanzitutto consente di utilizzare l'implementazione di una funzione hash come modulo per l'implementazione di HMAC. In questo modo, il nucleo centrale del codice di HMAC risulta già pronto all'uso senza alcuna modifica. Se si rendesse necessario sostituire una determinata funzione hash in un'implementazione HMAC, basterebbe rimuovere il modulo della funzione hash e inserire il nuovo modulo. Questa operazione potrebbe essere necessaria nel caso in cui si volesse impiegare una funzione hash più veloce. Ancora più importante: se la sicurezza della funzione hash incorporata venisse violata, sarebbe possibile mantenere la sicurezza di HMAC semplicemente sostituendo la funzione hash con una nuova versione più sicura (per esempio si potrebbe sostituire SHA con Whirlpool).

L'ultimo obiettivo progettuale dell'elenco precedente è, in realtà, il vantaggio principale di HMAC rispetto agli altri schemi proposti basati su funzioni hash. HMAC rimane sicuro sempre che la funzione hash incorporata abbia una resistenza crittografica ragionevole. Si tornerà su questo punto più avanti ma innanzitutto si deve esaminare la struttura di HMAC.

L'algoritmo HMAC

La Figura 12.10 illustra il funzionamento generale di HMAC. Occorre definire i seguenti termini.

H = funzione hash incorporata (per esempio MD5, SHA-1, RIPEMD-160).
IV = valore di input iniziale della funzione hash.
M = messaggio di input per HMAC (comprendente gli elementi di riempimento specificati nella funzione hash incorporata).
Y_i = i-esimo blocco di M, $0 \le i \le (L - 1)$
L = numero di blocchi di M.
b = numero di bit di un blocco.
n = lunghezza del codice hash prodotto dalla funzione hash incorporata.
K = chiave segreta; se la chiave è più lunga di b viene utilizzata la funzione hash per ridurla a n bit; la lunghezza consigliata è $\ge n$.

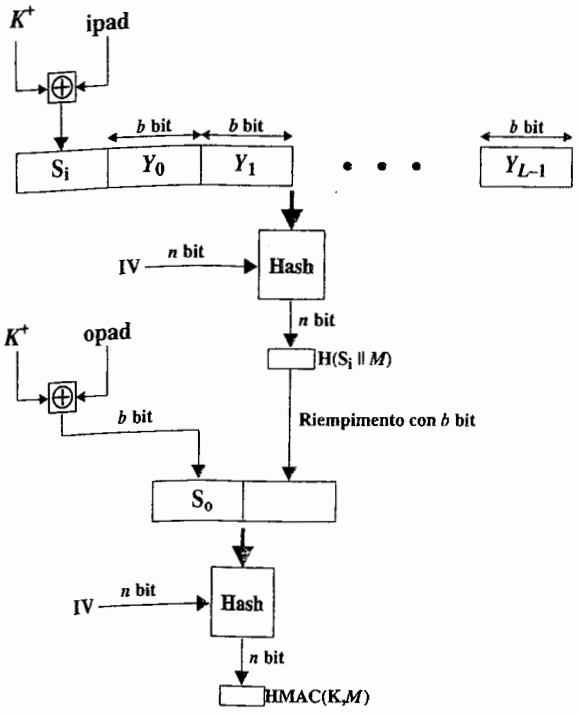


Figura 12.10 La struttura di HMAC.

K^+ = K riempito con una serie di "0" a sinistra in modo che il risultato abbia una lunghezza di b bit.
ipad = 00110110 (36 in esadecimale) ripetuto $b/8$ volte.
opad = 01011100 (5C in esadecimale) ripetuto $b/8$ volte.

Pertanto HMAC può essere espresso nel seguente modo:

$$\text{HMAC}(K,M) = \text{H}[(K^+ \oplus \text{opad}) \,||\, \text{H}[(K^+ \oplus \text{ipad}) \,||\, M]]$$

Ecco come si può formulare l'algoritmo.

1. Aggiungere una sequenza di "0" all'estremità sinistra di K per creare una stringa di b bit K^+ (per esempio se K ha una lunghezza di 160 bit e $b = 512$, allora a K verranno aggiunti 44 byte a "0" (0x00)).
2. Eseguire uno XOR (OR esclusivo bit a bit) di K^+ con ipad per produrre il blocco di b bit S_i.
3. Aggiungere M a S_i.
4. Applicare H al flusso generato nel passo 3.

5. Eseguire l'operazione di XOR di K^+ con opad per produrre il blocco di b bit S_o.
6. Aggiungere il risultato della hash dal passo 4 a S_o.
7. Applicare H al flusso generato nel passo 6 e produrre in output il risultato.

Si noti che l'operazione di XOR con ipad fa in modo che cambi stato la metà dei bit di K. Analogamente, l'operazione di XOR con opad cambia stato a metà dei bit di K ma operando su un insieme di bit differente. In pratica, facendo passare S_i e S_o attraverso la funzione di compressione dell'algoritmo hash, si sono generate in modo pseudocasuale due chiavi partendo da K.

HMAC dovrebbe essere eseguibile più o meno nello sesso tempo della funzione hash incorporata, quanto meno per messaggi di una certa lunghezza. HMAC aggiunge tre esecuzioni della funzione di compressione hash (per S_i, S_o e il blocco prodotto dall'operazione hash interna).

È possibile utilizzare un'implementazione più efficiente, rappresentata nella Figura 12.11. In questo caso vengono precalcolate due quantità:

$$f(\text{IV},(K^+ \oplus \text{ipad}))$$
$$f(\text{IV},(K^+ \oplus \text{opad}))$$

dove f(cv, block) è la funzione di compressione per la funzione hash, che prende come argomenti una variabile di concatenamento di n bit e un blocco di b bit e produce una variabile di concatenamento di n bit. Queste quantità devono essere calcolate solo inizialmente e ogni volta che cambia la chiave. In pratica le quantità precalcolate sostituiscono il valore iniziale (IV) nella funzione hash. Con questa implementazione, all'elaborazione normalmente prodotta dalla funzione hash viene aggiunta una sola istanza della funzione di compressione. Questa implementazione più efficiente è particolarmente interessante se la maggior parte dei messaggi per cui viene calcolato il codice macchina è piuttosto breve.

La sicurezza di HMAC

La sicurezza di qualsiasi funzione MAC basata su una funzione hash incorporata dipende dalla resistenza crittografica della funzione hash sottostante. L'aspetto interessante di HMAC è che i suoi progettisti sono stati in grado di dimostrare una relazione esatta fra la resistenza della funzione hash incorporata e la resistenza di HMAC.

La sicurezza di una funzione MAC è generalmente espressa in termini della probabilità che una falsificazione abbia successo sulla base di un determinato tempo speso per l'operazione e un determinato numero di coppie messaggio/MAC create con la stessa chiave. In effetti si dimostra, [BELL96a], che per un determinato livello di impegno (tempo, coppie messaggio/MAC) e su messaggi generati da un utente legittimo e intercettati, la probabilità che un attacco contro HMAC abbia successo è equivalente a uno dei seguenti attacchi alla funzione hash incorporata.

1. L'estraneo è in grado di calcolare l'output della funzione di compressione anche con un IV casuale, segreto e sconosciuto all'estraneo.
2. L'estraneo trova collisioni nella funzione hash anche quando il valore iniziale IV è casuale e segreto.

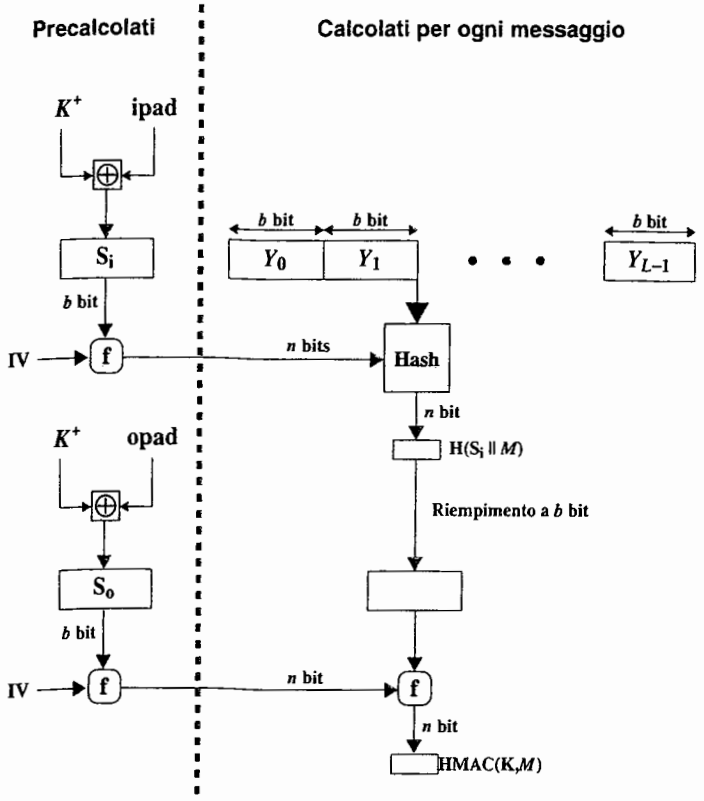


Figura 12.11 Un'implementazione più efficiente di HMAC.

Nel primo attacco si può vedere la funzione di compressione come equivalente alla funzione hash applicata a un messaggio costituito da un unico blocco di b bit. Per questo attacco, il valore iniziale IV della funzione hash è sostituito da un valore segreto e casuale di n bit. Un attacco contro questa funzione hash richiede o un attacco a forza bruta sulla chiave, con un livello di impegno dell'ordine di 2^n, o un attacco a compleanno, che è un caso speciale del secondo attacco di cui si parlerà di seguito.

Nel secondo attacco, l'estraneo ricerca due messaggi M e M' che producono lo stesso codice hash: H(M) = H(M'). Questo è l'attacco a compleanno discusso nel Capitolo 11. Si è dimostrato che questo richiede un livello di impegno pari a $2^{n/2}$ per una lunghezza hash pari a n. Con questo valore, la sicurezza di MD5 vacilla poiché un livello di impegno pari a 2^{64} è fattibile con le tecnologie odierne. Questo non significa che una funzione hash a 128 bit come MD5 è inadatta per HMAC per il seguente motivo. Per attaccare MD5, l'estraneo

può scegliere qualsiasi insieme di messaggi e lavorare su questi messaggi offline su un sistema dedicato fino a trovare una collisione. Poiché l'estraneo conosce l'algoritmo hash e il valore iniziale standard, potrà generare il codice hash per tutti i messaggi generati. Al contrario, quando si attacca HMAC, l'estraneo non può generare offline delle coppie messaggio/codice poiché non conosce K. Pertanto deve osservare una sequenza di messaggi generati da HMAC con la stessa chiave e svolgere l'attacco su questi messaggi noti. Per un codice hash di 128 bit, questo richiede l'osservazione di 2^{64} blocchi (2^{73} bit) generati con la stessa chiave. In un collegamento a 1 Gbit/s, sarebbe necessario osservare un flusso continuo di messaggi senza alcun cambio nella chiave per circa 150 mila anni per avere successo. Pertanto, se si è interessati principalmente alla velocità, è perfettamente accettabile utilizzare MD5 invece di SHA-1 come funzione hash incorporata di HMAC.

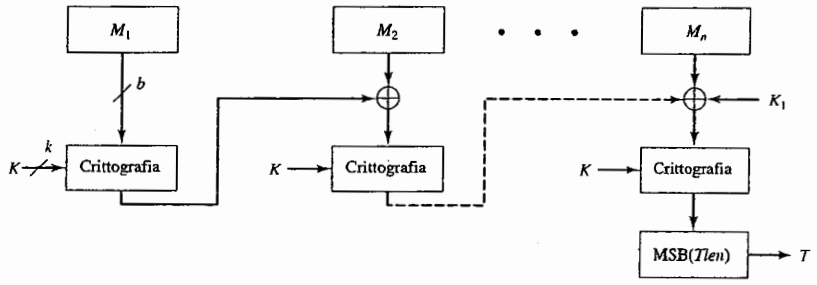
12.4 CMAC

L'algoritmo di autenticazione dati definito nello standard FIPS PUB 113, noto anche come CBC-MAC, è descritto nel Capitolo 11. Questo MAC basato su cifratura a blocchi è stato ampiamente adottato a livello governativo e industriale. [BEL00] ha dimostrato che questo MAC è sicuro, considerato un ragionevole insieme di criteri di sicurezza, ma con una restrizione: vengono elaborati solo i messaggi di lunghezza fissa di mn bit, dove n è la dimensione del blocco della cifratura e m è un intero positivo fisso. Come semplice esempio, si noti che dato il codice MAC CBC di un messaggio di un blocco X, $T = \text{MAC}(K, X)$, l'avversario conoscerebbe immediatamente il codice MAC del messaggio di due blocchi $X||(X \oplus T)$ dato che vale sempre T.

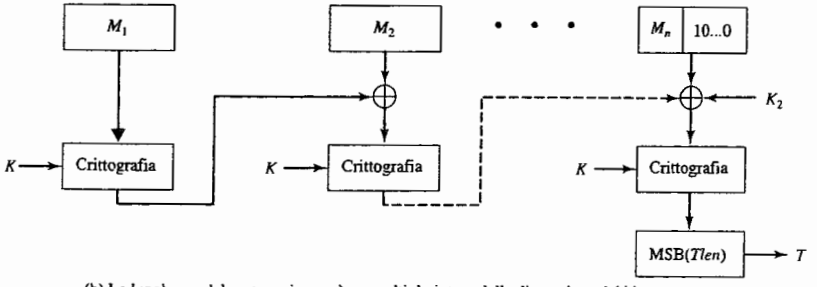
Black e Rogaway [BLAC00] hanno dimostrato che questa limitazione potrebbe essere superata usando tre chiavi: una chiave di lunghezza k da utilizzare a ogni passo del CBC e due chiavi di lunghezza n, dove k è la lunghezza della chiave e n è la lunghezza del blocco della cifratura. Questa proposta è stata perfezionata da Iwata e Kurosawa in modo da derivare le due chiavi di n bit dalla chiave di crittografia, senza necessità di doverle fornire separatamente [IWAT03]. Questa miglioria è stata adottata da NIST nella modalità operativa CMAC, per l'impiego con AES e triple DES. È specificata nella pubblicazione speciale NIST 800-38B.

Si consideri, per prima cosa, il funzionamento di CMAC quando il messaggio è un intero n multiplo della lunghezza del blocco della cifratura b. Nel caso di AES b vale 128, nel caso di triple DES vale 64. Il messaggio viene diviso in n blocchi $M_1, M_2, \ldots, M_n$. L'algoritmo utilizza una chiave di crittografia di k bit e una costante K_1 di n bit. Nel caso di AES la dimensione della chiave k è 128, 192 o 256 bit; nel caso di triple DES è 112 o 168 bit. CMAC viene calcolato nel modo seguente (Figura 12.12a).

$$C_1 = \text{E}(K, M_1)$$
$$C_2 = \text{E}(K, [M_2 \oplus C_1])$$
$$C_3 = \text{E}(K, [M_3 \oplus C_2])$$
$$\vdots$$



(a) La lunghezza del messaggio è un multiplo intero della dimensione del blocco



(b) La lunghezza del messaggio non è un multiplo intero della dimensione del blocco

Figura 12.12 CMAC (MAC basato su cifratura).

$$C_n = \mathrm{E}(K, [M_n \oplus C_{n-1} \oplus K_1])$$
$$T = \mathrm{MSB}_{Tlen}(C_n)$$

Dove:

T	= codice di autenticazione del messaggio, chiamato anche *tag* ("etichetta")
$Tlen$	= lunghezza di T in bit
$\mathrm{MSB}_s(X)$	= gli s bit più a sinistra della stringa di bit X

Se il messaggio non è un multiplo intero della lunghezza del blocco della cifratura, è necessario aggiungere dei bit di riempimento alla destra dell'ultimo blocco (i bit meno significativi): un bit a 1 seguito dal numero necessario di bit a 0 in modo che anche il blocco finale abbia lunghezza b. L'operazione CMAC procede quindi come nel caso precedente, ma viene utilizzata una chiave K_2 di n bit differente da K_1.

Le due chiavi di n bit vengono ricavate dalla chiave di crittografia di k bit nel modo seguente:

$$L = \mathrm{E}(K, 0_n)$$
$$K_1 = L \bullet x$$
$$K_2 = L \bullet x^2 = (L \bullet x) \bullet x$$

dove la moltiplicazione (•) viene eseguita nel campo finito $\mathrm{GF}(2^n)$, e x e x^2 sono i polinomi di primo e secondo grado elementi di $\mathrm{GF}(2^n)$. La rappresentazione binaria di x consiste quindi in $n - 2$ zero seguiti da 10; la rappresentazione binaria di x^2 consiste di $n - 3$ zero seguiti da 100. Il campo finito viene stabilito in relazione a un polinomio irriducibile che è il primo, in ordine lessicografico, fra tutti i polinomi con il minimo numero possibile di termini nulli. I polinomi, per le due dimensioni di blocco approvate, sono $x^{64} + x^4 + x^3 + x + 1$ e $x^{128} + x^7 + x^2 + x + 1$.

Per generare K_1 e K_2 si applica la cifratura al blocco costituito unicamente da bit nulli. La prima sottochiave viene ottenuta dal testo cifrato risultante con uno scorrimento a sinistra di un bit ed eventualmente dallo XOR con una costante che dipende dalla dimensione del blocco. La seconda sottochiave viene derivata nello stesso modo dalla prima sottochiave. Questa proprietà dei campi finiti della forma $\mathrm{GF}(2^n)$ è stata spiegata nella discussione della trasformazione MixColumns nel Capitolo 5.

12.5 Letture e siti Web consigliati

[GILB03] esamina la sicurezza degli algoritmi da SHA-256 a SHA-512. [BELL96a] e [BELL96b] forniscono una panoramica di HMAC.

BELL96a M. Bellare, R. Canetti e H. Krawczyk. "Keying Hash Functions for Message Authentication". *Proceedings, CRYPTO 96*, Agosto 1996. Pubblicato da Springer-Verlag. Una versione più estesa è disponibile a: **http://www-cse.ucsd.edu/users/mihir**.

BELL96b M. Bellare, R. Canetti e H. Krawczyk. "The HMAC Construction". *CryptoBytes*, Primavera 1996.

GILB03 H. Gilbert e H. Handschuh. "Security Analysis of SHA-256 and Sisters". *Proceedings, CRYPTO 03*, 2003. Pubblicato da Springer-Verlag.

Siti Web consigliati

- **NIST Secure Hashing Page:** SHA FIPS e documenti correlati.
- **Whirlpool:** informazioni varie relative a Whirlpool.
- **Block cipher modes of operation:** pagina NIST con tutte le informazioni relative a CMAC.

12.6 Termini chiave, domande di ripasso e problemi

Termini chiave

big endian
CMAC
Funzione di compressione

little endian
HMAC
MD4
MD5
RIPEMD-160
SHA-1
SHA-256
SHA-384
SHA-512
Whirlpool

Domande di ripasso

12.1 Qual è la differenza fra i formati little-endian e big-endian?
12.2 Quali funzioni aritmetiche e logiche di base vengono utilizzate in SHA?
12.3 Quali funzioni aritmetiche e logiche di base vengono utilizzate in Whirlpool?
12.4 Perché vi è stato maggiore interesse nello sviluppo di un codice di autenticazione dei messaggi MAC derivato da una funzione hash crittografica invece che da una cifratura simmetrica?
12.5 Quali modifiche è necessario apportare a HMAC per sostituire una funzione hash con un'altra?

Problemi

12.1 Nella Figura 12.4, si suppone che sia disponibile un array di 80 word di 64 bit per memorizzare i valori di W_t in modo che possano essere precalcolati all'inizio dell'elaborazione di un blocco. Ora si supponga che lo spazio sia limitato. Come alternativa si consideri l'uso di un buffer circolare di 16 word che inizialmente contiene i valori da W_0 a W_{15}. Progettare un algoritmo che, per ogni passo t calcoli il valore di input richiesto W_t.
12.2 Per SHA-512, mostrare le equazioni per i valori di W_{16}, W_{17}, W_{18}, W_{19}.
12.3 Si supponga che $a_1a_2a_3a_4$ siano i quattro byte di una word di 32 bit. Ogni a_i può essere considerato come un intero compreso fra 0 e 255 rappresentato in codice binario. In un'architettura big-endian, questa word rappresenta l'intero:

$$a_1 2^{24} + a_2 2^{16} + a_3 2^8 + a_4$$

In un'architettura little-endian, questa word rappresenta l'intero:

$$a_4 2^{24} + a_3 2^{16} + a_2 2^8 + a_1$$

A. Alcune funzioni hash, tra le quali MD5, presuppongono l'architettura little-endian. È importante che il codice digest sia indipendente dall'architettura sottostante. Pertanto, per svolgere l'operazione di somma modulo 2 di MD5 o RIPEMD-160 su un'architettura big-endian, è necessario apportare delle modifiche. Si supponga che $X = x_1x_2x_3x_4$ e $Y = y_1y_2y_3y_4$. Mostrare l'esecuzione di una somma $(X + Y)$ su una macchina big-endian.
B. SHA-1 usa un'architettura big-endian. Mostrare come viene svolta l'operazione $(X + Y)$ per SHA-1 su una macchina little-endian.

12.4 Questo problema introduce una funzione hash simile, nel principio, a SHA, che opera però su lettere anziché su dati binari. Viene chiamata TTH (Toy Tetragraph Hash)[5]. Dato un messaggio composto da una sequenza di lettere, TTH produce un valore hash di quattro lettere. Per prima cosa, TTH divide il messaggio in blocchi di 16 lettere ignorando spazi, punteggiatura e maiuscole/minuscole. Se la lunghezza del messaggio non è divisibile per 16, vengono aggiunti dei caratteri nulli. Viene mantenuto un totale temporaneo di quattro numeri inizializzato al valore (0, 0, 0, 0), utilizzato come input alla funzione di compressione per elaborare il primo blocco. La funzione di compressione è costituita da due fasi. **Fase 1:** ottiene il blocco di testo successivo, lo compone per righe in un blocco di testo 4 × 4 e lo converte a numeri (A = 0, B = 1 ecc.). Per esempio, dal blocco ABCDEFGHIJKLMNOP si ottiene:

A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Successivamente somma ogni colonna modulo 26 e aggiunge il risultato al totale temporaneo. In questo esempio il totale temporaneo vale (24, 2, 6, 19). **Fase 2:** utilizzando la matrice dalla fase 1, ruota la prima riga a sinistra di una posizione, la seconda riga a sinistra di 2, la terza riga a sinistra di 3 e inverte l'ordine della quarta colonna. Nell'esempio:

B	C	D	A
G	H	E	F
L	I	J	K
P	O	N	M

1	2	3	0
6	7	4	5
11	8	9	10
15	14	13	12

Ora somma ogni colonna modulo 26 e aggiunge il risultato al totale temporaneo. Il nuovo totale vale dunque (5, 7, 9, 11). Il totale temporaneo così ottenuto è ora l'input alla prima fase della funzione di compressione per il blocco di testo successivo. Dopo l'elaborazione dell'ultimo blocco, converte il totale temporaneo finale in lettere. Per esempio, se il messaggio fosse ABCDEFGHIJKLMNOP, il risultato sarebbe FHJL.
A. Disegnare figure simili alle Figure 12.1 e 12.2 per schematizzare la logica generale di TTH e la logica della funzione di compressione.
B. Calcolare la funzione hash del messaggio di 48 lettere "I leave twenty million dollars to my friendly cousin Bill".

[5] L'autore ringrazia William K. Mason, del personale della rivista *The Cryptogram*, per aver fornito questo esempio.

C. Per dimostrare la debolezza di TTH, trovare un blocco di 48 lettere che produce un valore hash identico a quello appena ottenuto. *Suggerimento*: utilizzare molte lettere A.

12.5 Sviluppare una tabella simile alla Tabella 4.8 per GF(2^8) con $m(x) = x^8 + x^4 + x^3 + x^2 + 1$.

12.6 Riscrivere le mini-box E ed E^{-1} della Tabella 12.3 nel formato tradizionale di matrice quadrata delle S-box, come quello proposto nella Tabella 5.4.

12.7 Verificare che la Figura 12.9 rappresenti un'implementazione valida della S-box riportata nella Tabella 12.3a, illustrando i calcoli richiesti per i tre valori di ingresso: 00, 55, 1E.

12.8 Fornire un'espressione booleana che definisca il comportamento equivalente alla S-box illustrata nella Figura 12.9.

12.9 Whirlpool utilizza la formula $H_i = E(H_{i-1}, m_i) \oplus H_{i-1} \oplus M_i$. Un'altra formula che è stata dimostrata sicura da Preneel è $H_i = E(H_{i-1}, M_i) \oplus M_i$. Si noti ora che la tecnica di generazione delle chiavi di Whirlpool è simile alla crittografia della chiave di cifratura con una pseudo-chiave definita dalle costanti di fase, in modo tale che il nucleo del processo hash potrebbe essere formalmente considerato come due linee di crittografia interagenti. Si consideri la crittografia $E(H_{i-1}, M_i)$. Si potrebbe scrivere la chiave di fase finale di questo blocco come $K_{10} = E(RC, H_{i-1})$. Si dimostri ora che le due formule hash sono sostanzialmente equivalenti a causa della particolare tecnica di generazione delle chiavi adottata.

12.10 All'inizio del Paragrafo 12.4 si è osservato che dato il MAC CBC di un messaggio di un blocco X, $T = \text{MAC}(K, X)$, l'avversario conoscerebbe immediatamente il codice MAC del messaggio di due blocchi $X \| (X \oplus T)$ dato che vale sempre T. Giustificare l'affermazione.

12.11 In questo problema si dimostra che, nel caso di CMAC, la variante che calcola lo XOR della seconda chiave dopo aver applicato la crittografia finale non funziona correttamente. Si consideri il caso di un messaggio multiplo intero della dimensione del blocco. La variante può essere espressa come $\text{VMAC}(K, M) = \text{CBC}(K, M) \oplus K_1$. Si supponga ora che un avversario possa ottenere i codici MAC di tre messaggi: il messaggio $\mathbf{0} = 0^n$, dove n è la dimensione del blocco di cifratura; il messaggio $\mathbf{1} = 1^n$ e il messaggio $\mathbf{1} \| \mathbf{0}$. Come risultato di queste tre richieste l'avversario ottiene $T_0 = \text{CBC}(K, \mathbf{0}) \oplus K_1$; $T_1 = \text{CBC}(K, \mathbf{1}) \oplus K_1$ e $T_2 = \text{CBC}(K, [\text{CBC}(K, \mathbf{1})]) \oplus K_1$. Dimostrare che l'avversario può ora calcolare il MAC corretto del nuovo messaggio $\mathbf{0} \| (T_0 \oplus T_1)$.

12.12 Nella discussione della generazione delle sottochiavi in CMAC, si è affermato che si applica la cifratura al blocco costituito esclusivamente da bit 0. La prima sottochiave viene derivata dalla stringa risultante con uno scorrimento a sinistra di un bit ed, eventualmente, calcolandone lo XOR con una costante che dipende dalla dimensione del blocco. La seconda sottochiave viene derivata nello stesso modo dalla prima sottochiave.

A. Quali sono le costanti necessarie per le dimensioni del blocco di 64 e di 128 bit?

B. Spiegare come il risultato desiderato sia ottenuto tramite lo scorrimento a sinistra e l'operazione di XOR.

Capitolo 13

Firme digitali e protocolli di autenticazione

Concetti essenziali

- Per **firma digitale** si intende un meccanismo di autenticazione che consente all'autore di un messaggio di aggiungere un codice che serva da firma. La firma viene creata calcolando il codice hash del messaggio e crittografandolo con la chiave privata dell'autore. La firma garantisce l'origine e l'integrità del messaggio.
- I **protocolli di mutua autenticazione** consentono alle entità comunicanti di garantire l'una all'altra la propria identità e di scambiare delle chiavi di sessione.
- L'**autenticazione a una via** garantisce al destinatario che il messaggio proviene effettivamente dal mittente indicato.
- **DSS** (Digital Signature Standard) è uno standard NIST che utilizza l'algoritmo SHA (Secure Hash Algorithm).

Lo sviluppo più importante della crittografia a chiave pubblica è rappresentato dalla firma digitale. La firma digitale offre un insieme di funzionalità di sicurezza che sarebbe difficile implementare in altro modo. Questo capitolo si apre con una panoramica della firma digitale. Poi si parlerà dei protocolli di autenticazione, molti dei quali dipendono dall'uso della firma digitale. Infine verrà introdotto lo standard DSS (Digital Signature Standard).

13.1 Le firme digitali

Requisiti

L'autenticazione dei messaggi protegge le due parti che si scambiano i messaggi da una terza parte. Tuttavia non protegge le due parti l'una dall'altra. Sono possibili varie forme di dispute fra le due parti.

Per esempio, si supponga che John invii a Mary un messaggio autenticato, utilizzando uno degli schemi rappresentati nella Figura 11.4. Potrebbero sorgere le seguenti dispute.

1. Mary può creare un messaggio e sostenere che proviene da John. Mary dovrebbe semplicemente creare il messaggio e aggiungergli un codice di autenticazione utilizzando la chiave condivisa da John e Mary.
2. John può negare di avere inviato il messaggio. Poiché Mary può anche creare un messaggio fasullo, non esiste alcun modo per dimostrare che sia stato veramente John a inviare il messaggio originale.

Entrambe le situazioni rappresentano un grave problema. Ecco un esempio della prima situazione: si svolge un trasferimento elettronico di una somma di denaro e il destinatario aumenta la somma di denaro trasferita sostenendo che è stato il mittente a inviare la somma di denaro maggiorata. Un esempio della seconda situazione può esser rappresentato da un messaggio di posta elettronica al quale si chiede a un agente di borsa di eseguire una transazione che successivamente si rivela in perdita. Il mittente può successivamente sostenere di non avere mai inviato il messaggio.
Nelle situazioni come queste, in cui non esiste la completa fiducia fra il mittente e il destinatario, è necessario qualcosa di più della semplice autenticazione. La soluzione più interessante a questo problema è rappresentata dalla firma digitale. La firma digitale è analoga a una firma manuale autografa e deve avere le seguenti proprietà.

- Deve certificare l'autore e la data/ora della firma.
- Deve autenticare il contenuto nel momento in cui è stata apposta la firma.
- Deve essere verificabile da terzi in modo da consentire di risolvere eventuali dispute.

Pertanto la funzione di firma digitale include la funzione di autenticazione.

Sulla base di queste proprietà, si possono formulare i seguenti requisiti per una firma digitale.

- La firma deve essere una configurazione di bit che dipende dal messaggio firmato.
- La firma deve utilizzare informazioni specifiche del mittente in modo da impedirgli sia modifiche al messaggio che la possibilità di negare di aver inviato il messaggio.
- Deve essere relativamente facile produrre la firma digitale.
- Deve essere relativamente facile riconoscere e verificare la firma digitale.
- Deve essere computazionalmente impossibile falsificare una firma digitale costruendo un nuovo messaggio per una firma digitale esistente oppure costruendo una firma digitale fraudolenta a partire da un determinato messaggio.
- Deve essere possibile conservare una copia della firma digitale.

Una funzione hash sicura, incorporata in uno schema come quelli rappresentati nella Figura 11.5C o D soddisfa questi requisiti.

Sono stati proposti vari approcci per la funzione di firma digitale. Questi approcci rientrano in due categorie: diretti e arbitrati.

Firma digitale diretta

La firma digitale diretta coinvolge solo le parti che svolgono la comunicazione, ovvero il mittente e il destinatario. Si presuppone che il destinatario conosca la chiave pubblica del mittente. Una firma digitale può essere creata crittografando l'intero messaggio con la chiave privata del mittente (Figura 11.1C) oppure crittografando un codice hash del messaggio tramite la chiave privata del mittente (Figura 11.5C).

La segretezza può essere garantita crittografando ulteriormente l'intero messaggio più la firma con la chiave pubblica del destinatario (crittografia a chiave pubblica) o con una chiave segreta condivisa (crittografia simmetrica); per esempio si considerino le Figure 11.1D e 11.5D. Si noti che è importante applicare prima la firma e poi una funzione di segretezza esterna. In caso di disputa, vi è la possibilità che una terza parte debba poter osservare il messaggio e la sua firma. Se la firma venisse calcolata sul messaggio crittografato, la parte terza dovrebbe necessariamente avere la chiave di decrittografia per leggere il messaggio originale. Se invece la firma è interna, il destinatario potrà memorizzare il messaggio in chiaro con la relativa firma, per poterli riutilizzare in seguito nel caso in cui fosse necessario risolvere una disputa.
Tutti gli schemi descritti finora hanno un punto debole in comune. La validità dello schema dipende dalla sicurezza della chiave privata del mittente. Se un mittente in seguito volesse negare di aver inviato un determinato messaggio, potrebbe sostenere che la chiave privata sia stata persa o rubata e che dunque qualcun altro ha falsificato la firma. Per rendere impossibile o comunque molto improbabile questa situazione, possono essere impiegati dei controlli amministrativi sulla sicurezza delle chiavi private ma questo problema non può comunque essere eliminato. Un esempio consiste nel richiedere che ogni messaggio firmato includa un'indicazione temporale timestamp (data e ora) e richiedere che la violazione delle chiavi venga prontamente segnalata a un'autorità centrale.

Un altro problema è dovuto al fatto che la chiave privata potrebbe effettivamente essere rubata a X nel tempo T. Un estraneo potrebbe quindi inviare un messaggio con la firma digitale di X e contrassegnarlo con un timestamp precedente o coincidente con T.

Firma digitale arbitrata

I problemi della firma digitale diretta possono essere risolti utilizzando un arbitro.

Come negli schemi di firma diretta, vi sono vari schemi a firma arbitrata. In termini generali, questi schemi si comportano nel seguente modo. Qualsiasi messaggio firmato da un mittente X a un destinatario Y attraversa innanzitutto un arbitro A che sottopone il messaggio e la sua firma a una serie di test per verificare la sua origine e il suo contenuto. Il messaggio viene quindi datato e inviato a Y indicando che è già stato verificato dall'arbitro. La presenza di A risolve il problema degli schemi a firma diretta e X non può negare di aver inviato il messaggio.

L'arbitro gioca un ruolo delicato e cruciale in questo tipo di schema ed entrambe le parti devono riporre una notevole fiducia sul fatto che il meccanismo di arbitraggio funzioni correttamente. L'uso di un sistema fidato, descritto nel Capitolo 20, potrebbe soddisfare questo requisito.

La Tabella 13.1, basata sulle situazioni descritte in [AKL83] e [MITC92], fornisce numerosi esempi di firme digitali arbitrate.[1] Nel primo viene utilizzata la crittografia sim-

[1] Viene utilizzato il seguente formato. Un passo di comunicazione in cui P invia a Q un messaggio M viene rappresentato come $P \rightarrow Q$: M.

metrica. Si presuppone che il mittente X e l'arbitro A condividano una chiave segreta K_{xa} e che A e Y condividano la chiave segreta K_{ay}. X costruisce un messaggio M e calcola il suo valore hash H(M). Quindi X trasmette ad A il messaggio con una firma. La firma è costituita dall'identificatore ID_X di X più il valore hash, il tutto crittografato con K_{xa}. A esegue la decrittografia della firma e controlla il valore hash per convalidare il messaggio. Quindi A trasmette un messaggio a Y crittografato con K_{ay}. Il messaggio include ID_X, il messaggio originale ricevuto da X, la firma e un timestamp. Y può decrittografarlo per recuperare il messaggio e la firma. Il timestamp informa Y che questo messaggio non è una ripetizione di un messaggio precedente. Y può memorizzare M e la firma. In caso di disputa, Y, che sostiene di aver ricevuto il messaggio M da X, invia ad A il seguente messaggio:

$$E(K_{ay},[ID_X \| M \| E(K_{xa},[ID_X \| H(M)])])$$

L'arbitro usa K_{ay} per ripristinare ID_X, M e la firma e quindi utilizza K_{xa} per decrittografare la firma e verificare il codice hash. In questo schema, Y non può controllare direttamente la firma di X; la firma è presente unicamente per risolvere le dispute. Y considera il messaggio proveniente da X autentico poiché gli è stato inviato da A. In questa situazione, entrambi devono riporre un elevato livello di fiducia in A.

- X deve confidare sul fatto che A non riveli K_{xa} e non generi false firme nella forma $E(K_{xa},[ID_X \| H(M)])$.
- Y deve fidarsi che A invii $E(K_{ay},[ID_X \| M \| E(K_{xa},[ID_X \| H(M)] \| T])$ solo se il valore hash è corretto e la firma è stata effettivamente generata da X.
- Entrambi devono fidarsi di A per la soluzione delle dispute.

Tabella 13.1 Tecniche a firma digitale arbitrata.

A. Crittografia convenzionale: l'arbitro vede il messaggio

(1) $X \Rightarrow A : M \| E(K_{xa},[ID_X \| H(M)])$
(2) $A \Rightarrow Y : E(K_{ay},[ID_X \| M \| E(K_{xa},[ID_X \| H(M)] \| T])$

B. Crittografia convenzionale: l'arbitro non vede il messaggio

(1) $X \Rightarrow A : ID_X \| E(K_{xy},M) \| E(K_{xa},[ID_X \| E(K_{xy},M)])$
(2) $A \Rightarrow Y : E(K_{ay},[ID_X \| E(K_{xy},M)] \| E(K_{xa},[ID_X \| H(E(K_{xy},M)) \| T])$

C. Crittografia a chiave pubblica: l'arbitro non vede il messaggio

(1) $X \Rightarrow A : ID_X \| E(PR_x,[ID_X \| E(PU_y,E(PR_x,M))])$
(2) $A \Rightarrow Y : E(PR_a,[ID_X \| E(PU_y,E(PR_x,M)) \| T])$

Notazione: X = mittente, Y = destinatario, A = arbitro, M = messaggio, T = timestamp

Se l'arbitro è fidato, allora X saprà che nessuno può falsificare la sua firma e Y sarà sicuro che X non potrà negare di aver apposto la propria firma.

La situazione precedente implica anche che A sia in grado di leggere i messaggi da X e Y e, in pratica, che la stessa cosa possa essere fatta da un estraneo. La Tabella 13.1.B mostra una situazione che sfrutta l'arbitraggio ma garantisce anche la segretezza. In questo caso, si suppone che X e Y condividano la chiave segreta K_{xy}. Ora X trasmette ad A un identificatore, una copia del messaggio crittografato con K_{xy} e una firma. La firma è costituita dall'identificatore e dal valore hash del messaggio crittografato, il tutto crittografato utilizzando K_{xa}. Come prima, A esegue la decrittografia della firma e controlla il valore hash per convalidare il messaggio. In questo caso, A lavora solo con la versione crittografata del messaggio e non può leggerla. A questo punto A trasmette a Y tutto ciò che ha ricevuto da X più un codice timestamp, il tutto crittografato con K_{ay}.

Anche se non può leggere il messaggio, l'arbitro sarà comunque nella posizione di prevenire le frodi da parte di X o Y. Un altro problema, questo in comune con la prima situazione, è che l'arbitro potrebbe allearsi con il mittente negando di avere inviato un messaggio firmato o con il destinatario per falsificare la firma del mittente.

Tutti i problemi appena trattati possono essere risolti utilizzando uno schema a chiave pubblica, di cui una versione si trova nella Tabella 13.1C. In questo caso, X esegue una doppia crittografia del messaggio M, prima con la propria chiave privata, PR_x, e poi con la chiave pubblica di Y, PU_y. Si tratta quindi di una versione firmata e segreta del messaggio. Questo messaggio firmato, insieme all'identificatore di X, viene nuovamente crittografato con PR_x e insieme con ID_X viene rinviato ad A. Il messaggio interno, a doppia crittografia, è protetto contro chiunque, ad eccezione di Y. Tuttavia A può decrittografare la crittografia esterna per assicurarsi che il messaggio provenga da X (poiché solo X conosce PR_x). A controlla che la coppia chiave pubblica/privata di X sia tuttora valida e, in caso affermativo, verifica il messaggio. Poi A trasmette un messaggio a Y, crittografato con PR_a. Il messaggio include ID_X, il messaggio a doppia crittografia e un timestamp. Questo sistema presenta vari vantaggi rispetto ai due schemi precedenti. Innanzitutto non è necessario che le parti condividano delle informazioni prima della comunicazione, impedendo quindi alleanze fra le parti. In secondo luogo, non si può inviare alcun messaggio con data errata, anche se PR_x venisse violata, sempre che non venga violata anche PR_a. Infine, il contenuto del messaggio inviato da X a Y è segreto sia per A che per chiunque altro. Tuttavia quest'ultimo schema prevede una doppia crittografia del messaggio con un algoritmo a chiave pubblica. Di seguito si affronteranno degli approcci più pratici.

13.2 I protocolli di autenticazione

Gli strumenti di base descritti nel Capitolo 11 vengono utilizzati in varie applicazioni, fra cui la firma digitale di cui si è parlato nel Paragrafo 13.1. Si stanno però sviluppando numerosi altri usi. In questa parte del capitolo ci si concentrerà su due aree di tipo generale (autenticazione reciproca e autenticazione monodirezionale) esaminando alcune implicazioni delle tecniche di autenticazione impiegate.

Autenticazione mutua

Un'importante area applicativa è rappresentata dai protocolli di autenticazione mutua. Tali protocolli consentono alle due parti di verificare vicendevolmente la propria identità e scambiarsi le chiavi di sessione. Questo argomento è stato esaminato nel Paragrafo 7.3 (tecniche simmetriche) e nel Paragrafo 10.1 (tecniche a chiave pubblica), ponendo l'attenzione sulla distribuzione delle chiavi. Ora si tornerà su questo argomento per considerare in generale le implicazioni dell'autenticazione.

Nel campo dello scambio di chiavi autenticate vi sono fondamentalmente due problemi: la segretezza e l'aggiornamento. Per evitare che qualcuno si finga qualcun altro e per evitare la violazione delle chiavi di sessione, le informazioni essenziali per l'identificazione e le chiavi di sessione devono essere comunicate in forma crittografata. Questo richiede la pre-esistenza di chiavi pubbliche o segrete che possano essere utilizzate per questo scopo. Il secondo problema, correttezza temporale, è importante data la possibilità che un messaggio venga riproposto (attacco a replay). Una riproposizione potrebbe, nella peggiore delle ipotesi, consentire a un estraneo di violare una chiave di sessione o impersonare con successo una delle parti. Come minimo un replay può confondere le operazioni presentando messaggi che sembrano legittimi ma che in realtà non lo sono.

[GONG93] elenca i seguenti esempi di attacco a replay.

- **Ripetizione semplice:** l'avversario non fa altro che copiare un messaggio e riproporlo successivamente.
- **Ripetizione registrabile:** l'estraneo ripropone un messaggio dotato di timestamp entro il periodo di validità temporale.
- **Ripetizione non rilevabile:** questa situazione può sorgere perché il messaggio originale può essere stato soppresso e pertanto non è arrivato a destinazione; arriva solo il messaggio ripetuto.
- **Ripetizione all'indietro senza modifiche:** questo è un rinvio del messaggio al mittente. Questo attacco è possibile se viene utilizzata la crittografia simmetrica e se il mittente non può riconoscere con facilità la differenza fra i messaggi inviati e i messaggi ricevuti sulla base del contenuto.

Una stragegia per rispondere agli attacchi a replay consiste nell'associare un numero sequenziale a ciascun messaggio utilizzato in uno scambio di autenticazione. Un nuovo messaggio verrà accettato solo se il numero di sequenza è corretto. La difficoltà di questo approccio consiste nel fatto che richiede che entrambe le parti registrino l'ultimo numero di sequenza per ciascun'altra parte con la quale hanno comunicato. Dato questo sovraccarico, i numeri sequenziali non vengono normalmente utilizzati per l'autenticazione e lo scambio delle chiavi. Al loro posto viene utilizzato uno dei due approcci generali seguenti.

- **Timestamp:** la parte A accetta un messaggio solo se questo contiene un timestamp che, secondo A, è sufficientemente vicino al tempo corrente da essa misurato. Questo approccio richiede che i clock fra i vari partecipanti siano ben sincronizzati.
- **Challenge/response:** la parte A, attendendo un messaggio da B, le invia un messaggio nonce (challenge) e richiede che il messaggio successivo (response) che riceve da B contenga il valore nonce corretto.

Si può argomentare (vedere per esempio [LAM92a]) che l'approccio a timestamp non dovrebbe essere utilizzato per le applicazioni orientate alla connessione proprio per le difficoltà implicite di questa tecnica. Innanzitutto è necessario utilizzare un determinato protocollo per mantenere la sincronizzazione fra i vari clock dei processi coinvolti. Questo protocollo deve essere resistente ai guasti, per poter gestire gli errori della rete, ma anche sicuro (per rispondere agli attacchi ostili). In secondo luogo, sorgerebbe l'opportunità di un attacco nel caso in cui vi fosse una temporanea perdita di sincronizzazione dovuta a un guasto nel meccanismo di clock di una delle due parti. Infine, data la natura variabile e imprevedibile dei ritardi di rete, quando gli orologi sono distribuiti non si può garantire una sincronizzazione precisa. Pertanto, qualsiasi procedura basata sul timestamp deve prevedere una finestra temporale sufficientemente estesa per considerare i ritardi di rete ma sufficientemente ridotta per ridurre il più possibile le opportunità di attacco.

L'approccio a challenge/response è inutilizzabile per un'applicazione senza connessione in quanto richiede un handshake prima di ogni trasmissione negando, di fatto, la caratteristica principale di una transazione senza connessione. Per tali applicazioni, l'approccio migliore può essere utilizzare un server temporale sicuro e fare in modo che le varie parti tentino di mantenersi il più possibile sincronizzate con questo server (vedere per esempio [LAM92b]).

Approcci a crittografia simmetrica

Come si è detto nel Paragrafo 7.3, per garantire la segretezza delle comunicazioni in un ambiente distribuito si possono usare delle chiavi di crittografia simmetrica gerarchiche, a due livelli. In generale, questa strategia prevede l'uso di un centro di distribuzione delle chiavi (KDC). Ogni parte nella rete condivide con il centro KDC una chiave segreta, chiamata chiave master. Il centro KDC è poi responsabile della generazione delle chiavi che verranno utilizzate per un breve periodo durante la connessione fra le due parti (le chiavi di sessione) e della distribuzione protetta di tali chiavi tramite le chiavi master. Questo approccio è piuttosto comune: per esempio si vedrà il sistema Kerberos nel Capitolo 14. La discussione in questa parte del capitolo è utilissima per comprendere i meccanismi di funzionamento di Kerberos.

La Figura 7.9 illustra una proposta inizialmente avanzata da Needham e Schroeder [NEED78] per la distribuzione della chiave segreta tramite un centro KDC che, come si è detto nel Capitolo 7, include anche delle funzioni di autenticazione. Il protocollo può essere riassunto nel seguente modo.

1. $A \rightarrow KDC:$ $ID_A \| ID_B \| N_1$
2. $KDC \rightarrow A:$ $E(K_a, [K_s \| ID_B \| N_1 \| E(K_b, [K_s \| ID_A])])$
3. $A \rightarrow B:$ $E(K_b, [K_s \| ID_A])$
4. $B \rightarrow A:$ $E(K_s, N_2)$
5. $A \rightarrow B:$ $E(K_s, f(N_2))$

Le chiavi segrete K_a e K_b sono condivise rispettivamente fra la parte A e il centro KDC e fra la parte B e il centro KDC. Lo scopo del protocollo è quello di distribuire in modo sicuro ad A e B la chiave di sessione K_s. A acquisisce in modo sicuro una nuova chiave di sessione

nel passo 2. Il messaggio nel passo 3 può essere decrittografato, e pertanto compreso, solo da B. Il passo 4 riflette il fatto che B conosce K_s e il passo 5 garantisce a B che A conosce K_s e garantisce a B che si tratta di un messaggio nuovo grazie all'uso del nonce N_2. Come si è detto nel Capitolo 7, lo scopo dei passi 4 e 5 è quello di impedire un determinato tipo di attacco a replay. In particolare, se un estraneo fosse in grado di catturare il messaggio nel passo 3 e riproporlo, potrebbe provocare problemi alla parte B.

Nonostante l'operazione di handshake dei passi 4 e 5, il protocollo rimane vulnerabile a una forma di attacco a replay. Si supponga che un estraneo, X, sia stato in grado di violare una vecchia chiave di sessione. Obiettivamente si tratta di un caso molto più improbabile rispetto al fatto che un estraneo possa aver semplicemente osservato e registrato i dati trasmessi al passo 3. Ciononostante, si tratta di un potenziale rischio di sicurezza. A questo punto X può impersonare A e convincere B a utilizzare la vecchia chiave di sessione riproponendo il passo 3. A meno che B ricordi indefinitamente tutte le precedenti chiavi di sessione utilizzate con A, non sarà in grado di determinare che si tratta di un replay. Se X può intercettare il messaggio di handshake, al passo 4, potrà impersonare la risposta di A, al passo 5. Da questo punto in avanti, X potrà inviare a B tutti i messaggi che desidera e B supporrà che provengano da A in quanto utilizzano una chiave di sessione autenticata.

Denning [DENN81, DENN82] propone una soluzione per questo punto debole tramite una modifica al protocollo Needham/Schroeder che include l'aggiunta di un timestamp nei passi 2 e 3. La sua proposta presuppone che le chiavi master, K_a e K_b, siano sicure ed è costituita dai seguenti passi.

1. $A \rightarrow KDC$: $ID_A \| ID_B$
2. $KDC \rightarrow A$: $E(K_a, [K_s \| ID_B \| T \| E(K_b, [K_s \| ID_A \| T])])$
3. $A \rightarrow B$: $E(K_b, [K_s \| ID_A \| T])$
4. $B \rightarrow A$: $E(K_s, N_1)$
5. $A \rightarrow B$: $E(K_s, f(N_1))$

T è un timestamp che garantisce ad A e B che la chiave di sessione sia appena stata generata. Pertanto sia A che B sanno che la distribuzione della chiave è un'operazione recente. A e B possono verificare questo fatto controllando che:

$$|\text{Clock} - T| < \Delta t_1 + \Delta t_2$$

dove Δt_1 è la discrepanza stimata normale fa il clock del centro KDC e il clock locale (in A o B) e Δt_2 è il tempo di ritardo previsto nella rete. Ciascun nodo può impostare il proprio clock facendo riferimento a una determinata fonte standard. Poiché il timestamp T è crittografato utilizzando le chiavi master sicure, un estraneo, anche conoscendo una vecchia chiave di sessione, non potrà avere successo in quanto un messaggio replay del passo 3 verrebbe immediatamente rilevato da B come non sincronizzato.

Un ultimo punto: i passi 4 e 5 non sono stati inclusi nella presentazione originale [DENN81] ma sono stati aggiunti successivamente in [DENN82]. Questi passi confermano la ricezione della chiave di sessione in B.

Il protocollo Denning sembra offrire un livello di sicurezza superiore rispetto al protocollo Needham/Schroeder. Tuttavia sorge un nuovo problema: il nuovo schema richiede che tutti i sistemi della rete abbiano clock sincronizzati. [GONG92] evidenzia un rischio potenziale che sorge in questo caso. Il rischio si basa sul fatto che i clock distribuiti divengano non sincronizzati a causa di un sabotaggio o di un guasto negli orologi o nel meccanismo di sincronizzazione.[2] Il problema si verifica quando l'orologio del mittente anticipa quello del destinatario. In questo caso, un estraneo potrebbe intercettare un messaggio fornito dal mittente e riprodurlo successivamente quando il timestamp del messaggio diviene corretto per il sito del destinatario. Questo attacco può produrre risultati imprevedibili. Gong chiama questi attacchi **Suppress-replay**.

Un modo per contrastare gli attacchi suppress-replay è quello di imporre il requisito che le parti confrontino regolarmente il proprio clock con il clock del centro KDC. L'altra alternativa, che evita la necessità di sincronizzazione del clock, è quella di sfruttare i protocolli di handshaking tramite nonce. Quest'ultima alternativa non è vulnerabile agli attacchi suppress-replay poiché i valori nonce che verranno scelti dal destinatario nel futuro sono imprevedibili per il mittente. Il protocollo Needham/Schroeder, fa affidamento solo sul nonce ma, come si è visto, ha altri punti deboli.

In [KEHN92] è stato fatto un tentativo di rispondere alle preoccupazioni relative agli attacchi suppress-replay e contemporaneamente correggere i problemi nel protocollo Needham/Schroeder. Successivamente è stata notata un'incoerenza in quest'ultimo protocollo ed è stata presentata una nuova strategia in [NEUM93a].[3] Il protocollo è il seguente.

1. $A \rightarrow B$: $ID_A \| N_a$
2. $B \rightarrow KDC$: $ID_B \| N_b \| E(K_b, [ID_A \| N_a \| T_b])$
3. $KDC \rightarrow A$: $E(K_a, [ID_B \| N_a \| K_s \| T_b]) \| \{E(K_b, [ID_A \| K_s \| T_b]) \| N_b$
4. $A \rightarrow B$: $E(K_b, [ID_A \| K_s \| T_b]) \| E(K_s, N_b)$

Ecco come funziona passo per passo lo scambio delle chiavi.

1. A inizia lo scambio di autenticazione generando il valore nonce, N_a, che invia in chiaro a B insieme al proprio identificatore. Questo valore nonce verrà restituito ad A in un messaggio crittografato che include la chiave di sessione che garantisce ad A la corretta sincronizzazione.
2. B avverte il centro KDC che è necessaria una chiave di sessione. Il suo messaggio include il proprio identificatore e un valore nonce N_b. Questo valore nonce verrà restituito a B in un messaggio crittografato che include la chiave di sessione, garantendo a B la corretta sincronizzazione del messaggio. Il messaggio di B al centro KDC include anche un blocco crittografato con la chiave segreta condivisa da B e dal centro KDC. Questo blocco viene utilizzato per chiedere al centro KDC di inviare le credenziali ad A; il blocco specifica il destinatario delle credenziali, un tempo di scadenza suggerito per le credenziali e il valore nonce ricevuto da A.
3. Il centro KDC passa ad A il valore nonce di B e un blocco crittografato con la chiave segreta che B condivide con il centro KDC. Il blocco funge in pratica da "ticket" che, come si vedrà, può essere utilizzato da A per le successive autenticazioni. Il centro

[2] Negli ultimi anni in molti computer e in altri sistemi elettronici sono stati utilizzati dei chip difettosi per registrare l'ora e la data. Questi chip hanno la tendenza di anticipare di un giorno [NEUM90].
[3] È veramente difficile non commettere errori in questo campo.

KDC invia ad A anche un blocco crittografato con la chiave segreta condivisa fra A e il centro KDC. Questo blocco verifica che B abbia ricevuto il messaggio iniziale di A (ID_B) e che si tratti di un messaggio sincronizzato e non di un replay (N_a) e fornisce ad A una chiave di sessione K_s e il tempo limite per il suo uso T_b.

4. A trasmette il ticket a B insieme al valore nonce di B, quest'ultimo crittografato con la chiave di sessione. Il ticket fornisce a B la chiave segreta che viene utilizzata per decrittografare $E(K_s, N_b)$ per recuperare il valore nonce. Il fatto che il nonce di B sia crittografato con la chiave di sessione comprova che il messaggio proveniva da A e non era un replay.

Questo protocollo fornisce un mezzo efficace e sicuro affinchè A e B possano stabilire una sessione con una chiave di sessione sicura. Inoltre il protocollo lascia A in possesso di una chiave che può essere utilizzata per una successiva autenticazione a B, evitando di contattare ripetutamente il server di autenticazione. Si supponga che A e B stabiliscano una sessione con il protocollo menzionato in precedenza e poi concludano tale sessione. Successivamente, ma sempre entro il tempo limite stabilito dal protocollo, A desidera attivare con B una nuova sessione. Si può utilizzare il protocollo seguente.

1. $A \rightarrow B$: $E(K_b, [ID_A \| K_s \| T_b]), N'_a$
2. $B \rightarrow A$: $N'_b, E(K_s, N'_a)$
3. $A \rightarrow B$: $E(K_s, N'_b)$

Quando B riceve il messaggio nel passo 1, verifica che il ticket non sia scaduto. I nonce Na' e N_b' appena generati garantiscono alle parti che non si tratta di un replay. Il tempo specificato in T_b è relativo al clock di B. Pertanto questo valore timestamp non richiede clock sincronizzati in quanto B controlla solamente i timestamp auto-generati.

Approcci di crittografia a chiave pubblica

Nel Capitolo 10 è stato presentato un approccio all'utilizzo della crittografia a chiave pubblica con lo scopo di distribuire la chiave di sessione (Figura 10.6). Questo protocollo presuppone che ognuna delle due parti sia in possesso della chiave pubblica aggiornata dell'altra parte. Talvolta è poco pratico richiedere questo requisito.

Ecco un protocollo che utilizza timestamp descritto in [DENN81].

1. $A \rightarrow AS$: $ID_A \| ID_B$
2. $AS \rightarrow A$: $E(PR_{as}, [ID_A \| PU_a \| T]) \| E(PR_{as}, [ID_B \| PU_b \| T])$
3. $A \rightarrow B$: $E(PR_{as}, [ID_A \| PU_a \| T]) \| E(PR_{as}, [ID_B \| PU_b \| T]) \| E(PU_b, E(PR_a, [K_s \| T]))$

In questo caso il sistema centrale è chiamato server di autenticazione (AS - Authentication Server), poiché tale sistema non è in realtà responsabile della distribuzione della chiave segreta. Piuttosto il server di autenticazione fornisce i certificati di chiave pubblica. La chiave di sessione viene scelta e crittografata da A, pertanto non vi è alcun rischio di furto dal server di autenticazione. I timestamp proteggono contro il riutilizzo di chiavi violate.

Questo protocollo è compatto ma richiede anch'esso la sincronizzazione degli orologi. Un altro approccio, proposto da Woo e Lam [WOO92a], utilizza invece dei codici nonce. Il protocollo è costituito dai seguenti passi.

1. $A \rightarrow KDC$: $ID_A \| ID_B$
2. $KDC \rightarrow A$: $E(PR_{auth}, [ID_B \| PU_b])$
3. $A \rightarrow B$: $E(PU_b, [N_a \| ID_A])$
4. $B \rightarrow KDC$: $ID_A \| ID_B \| E(PU_{auth}, N_a)$
5. $KDC \rightarrow B$: $E(PR_{auth}, [ID_A \| PU_a]) \| E(PU_b, E(PR_{auth}, [N_a \| K_s \| ID_B]))$
6. $B \rightarrow A$: $E(PU_a, E(PR_{auth}, [(N_a \| K_s \| ID_B) \| N_b]))$
7. $A \rightarrow B$: $E(K_s, N_b)$

Nel passo 1, A informa il centro KDC della sua intenzione di attivare una connessione sicura con B. Il centro KDC restituisce ad A una copia del certificato a chiave pubblica di B (passo 2). Utilizzando la chiave pubblica di B, A informa B del suo desiderio di comunicare e invia il codice nonce N_a (passo 3). Nel passo 4, B chiede al centro KDC il certificato a chiave pubblica di A e richiede una chiave di sessione; B include il codice nonce di A in modo che il centro KDC possa contrassegnare la chiave di sessione con tale codice nonce. Il codice nonce è protetto utilizzando la chiave pubblica del centro KDC. Nel passo 5 il centro KDC restituisce a B una copia del certificato a chiave pubblica di A più l'informazione $\{N_a, K_s, ID_B\}$. Queste informazioni dicono fondamentalmente che K_s è una chiave segreta generata dal centro KDC per conto di B e associata a N_a; l'associazione fra K_s e N_a garantisce ad A che K_s sia recente. Questa tripletta viene crittografata utilizzando la chiave privata del centro KDC in modo da consentire a B di verificare che tali dati provengano realmente dal centro KDC. Inoltre la tripletta viene crittografata utilizzando la chiave pubblica di B in modo che nessun'altra entità possa utilizzarla per tentare di attivare una connessione fraudolenta con A. Nel passo 6 la tripletta $\{N_a, K_s, ID_B\}$, sempre crittografata con la chiave privata del centro KDC, viene rinviata ad A insieme al codice nonce N_b generato da B. Il tutto viene crittografato con la chiave pubblica di A. A preleva la chiave di sessione K_s e la utilizza per crittografare N_b e restituirlo a B. Quest'ultimo messaggio garantisce a B che A conosca la chiave di sessione. Questo sembra essere un protocollo sicuro che tiene in considerazione vari tipi di attacchi. Tuttavia gli stessi autori hanno individuato un difetto e hanno prodotto una nuova versione dell'algoritmo in [WOO92b].

1. $A \rightarrow KDC$: $ID_A \| ID_B$
2. $KDC \rightarrow A$: $E(PR_{auth}, [ID_B \| PU_b])$
3. $A \rightarrow B$: $E((PU_b, [N_a \| ID_A])$
4. $B \rightarrow KDC$: $ID_A \| ID_B \| E(PU_{auth}, N_a)$
5. $KDC \rightarrow B$: $E(PR_{auth}, [ID_A \| PU_a] \| E(PU_b, E(PR_{auth}, [N_a \| K_s \| ID_A \| ID_B]))$
6. $B \rightarrow A$: $E(PU_a, E(PR_{auth}, [(N_a \| K_s \| ID_A \| ID_B) \| N_b]))$
7. $A \rightarrow B$: $E(K_s, N_b)$

L'identificatore di A, ID_A, viene aggiunto all'insieme di elementi crittografati con la chiave privata del centro KDC nei passi 5 e 6. Questo associa la chiave di sessione K_s agli identificatori delle due parti coinvolte nella sessione di comunicazione. Questa inclusione

di ID_A tiene in considerazione il fatto che il valore nonce N_a è considerato univoco solo fra tutti i nonce generati da A e non fra tutti i nonce generati da tutte le parti che possono essere in comunicazione. Pertanto, è proprio la coppia $\{ID_A, N_a\}$ che identifica univocamente la richiesta di connessione di A.

Sia in questo esempio che nei protocolli descritti in precedenza è accaduto che protocolli ritenuti sicuri venissero riveduti dopo avere svolto nuove analisi. Questi esempi evidenziano la difficoltà di ottenere risultati veramente sicuri nel campo dell'autenticazione.

Autenticazione monodirezionale

Un'applicazione per cui la crittografia sta acquisendo sempre più popolarità è la posta elettronica (e-mail). La natura intrinseca della posta elettronica e il suo principale vantaggio, consiste nel fatto che non è necessario che il mittente e il destinatario siano contemporaneamente online. Il messaggio di posta elettronica viene infatti inoltrato alla casella postale elettronica del destinatario dove rimane in attesa finché il destinatario stesso non vorrà leggerla.

La "busta", ovvero l'intestazione del messaggio di posta elettronica, deve essere in chiaro, in modo che il messaggio possa essere gestito tramite un protocollo di memorizzazione e inoltro come SMTP (Simple Mail Transfer Protocol) o X.400. Tuttavia spesso è preferibile che il protocollo di gestione della posta elettronica non richieda l'accesso alla forma in chiaro del messaggio poiché questo richiederebbe la completa fiducia nel meccanismo di gestione della posta elettronica. Di conseguenza il messaggio di posta elettronica dovrebbe essere crittografato in modo che il sistema di gestione della posta elettronica non richieda il possesso della chiave di decrittografia.

Il secondo requisito è quello dell'autenticazione. In genere il destinatario vuole la garanzia che il messaggio provenga dal mittente indicato.

Approccio a crittografia simmetrica

Utilizzando la crittografia simmetrica, la situazione di distribuzione decentralizzata della chiave illustrata nella Figura 7.11 si rivela impraticabile. Questo schema richiede che il mittente emetta una richiesta per il destinatario indicato e attenda una risposta comprendente la chiave di sessione; solo allora il mittente potrà inviare il messaggio.

Con qualche miglioramento, la strategia a centro KDC illustrata nella Figura 7.9, può essere impiegata per la crittografia della posta elettronica. Poiché si vuole evitare di richiedere che il destinatario (B) sia online contemporaneamente al mittente (A), i passi 4 e 5 devono essere eliminati. Per un messaggio M, la sequenza sarà quindi la seguente.

1. $A \rightarrow KDC: \quad ID_A \| ID_B \| N_1$
2. $KDC \rightarrow A: \quad E(K_a,[K_s \| ID_B \| N_1 \| E(K_b,[K_s \| ID_A])])$
3. $A \rightarrow B: \quad E(K_b,[K_s \| ID_A]) \| E(K_s,M)$

Questo approccio garantisce che il messaggio possa essere letto solo dal destinatario indicato. Inoltre fornisce un certo livello di autenticazione che il mittente sia in effetti A. Così come è specificato, il protocollo non protegge contro gli attacchi a replay. Si potrebbe

fornire una certa forma di difesa includendo nel messaggio un timestamp. Tuttavia, dati i potenziali ritardi di elaborazione della posta elettronica, tale timestamp avrebbe un'utilità molto limitata.

Approcci a crittografia a chiave pubblica

Sono già stati presentati degli approcci di crittografia a chiave pubblica adatti per la posta elettronica, compresa la crittografia dell'intero messaggio per ottenere la segretezza (Figura 11.1B), e l'autenticazione (Figura 11.1C) o entrambe le cose (Figura 11.1D). Questi approcci richiedono che il mittente conosca la chiave pubblica del destinatario (segretezza), che il destinatario conosca la chiave pubblica del mittente (autenticazione) o entrambe le cose (segretezza e autenticazione). Inoltre l'algoritmo a chiave pubblica deve essere applicato una o due volte a un messaggio che può essere anche piuttosto lungo.
Se la segretezza è fondamentale, può essere più efficiente la seguente via:

$$A \rightarrow B: E(PU_b, K_s) \| E(K_s, M)$$

In questo caso il messaggio viene crittografato con una chiave segreta monouso. Inoltre A esegue la crittografia di questa chiave monouso con la chiave pubblica di B. Solo B sarà in grado di utilizzare la chiave privata corrispondente per ripristinare la chiave monouso e poi utilizzare tale chiave per decrittografare il messaggio. Questo schema è più efficiente rispetto alla semplice crittografia dell'intero messaggio con la chiave pubblica di B.

Se l'autenticazione è fondamentale, allora può bastare una firma digitale come indicato nella Figura 11.5C:

$$A \rightarrow B: M \| E(PR_a, H(M))$$

Questo metodo garantisce che A non possa in un secondo tempo negare di aver inviato il messaggio. Tuttavia questa tecnica è soggetta a un altro tipo di frode. Bob compone un messaggio per il suo capo Alice che contiene un'idea che consentirebbe alla società di risparmiare ingenti somme. Appone la propria firma digitale e invia il messaggio tramite il servizio di posta elettronica. Alla fine il messaggio raggiunge la casella postale di Alice. Ma Max ha sentito l'idea di Bob e riesce ad avere accesso alla coda della posta elettronica prima della consegna. Trova il messaggio di Bob, estrae la sua firma, aggiunge la propria e rimette in coda il messaggio per consegnarlo ad Alice. In questo modo Max ha sottratto l'idea a Bob.

Per risolvere questo problema, sia il messaggio che la firma possono essere crittografati utilizzando la chiave pubblica del destinatario:

$$A \rightarrow B: E(PU_b,[M \| E(PR_a, H(M))])$$

Questi ultimi due schemi richiedono che B conosca la chiave pubblica di A e che questa sia aggiornata. Un modo efficace per garantirlo è il certificato digitale descritto nel Capitolo 10. Ora si ha:

$$A \rightarrow B: M \| E(PR_a, H(M)) \| E(PR_{as},[T \| ID_A \| PU_a])$$

Oltre al messaggio, A invia a B la firma digitale, crittografata con la chiave privata di A e il certificato di A crittografato con la chiave privata del server di autenticazione. Il destinatario del messaggio utilizza il certificato per ottenere la chiave pubblica del mittente e verificare

che sia autentica e poi utilizza la chiave pubblica per verificare il messaggio. Se è richiesta la segretezza, allora l'intero messaggio può essere crittografato con la chiave pubblica di B. Alternativamente l'intero messaggio può essere crittografato con una chiave segreta monouso; la chiave segreta viene anch'essa trasmessa, crittografata con la chiave pubblica di B. Questo approccio viene esplorato nel Capitolo 15.

13.3 Lo standard DSS (Digital Signature Standard)

Il NIST (National Institute of Standards and Technology) ha pubblicato lo standard FIPS 186, chiamato anche DSS (Digital Signature Standard). Lo standard DSS utilizza l'algoritmo SHA (Secure Hash Algorithm) descritto nel Capitolo 12 e presenta una nuova tecnica di firma digitale, l'algoritmo DSA (Digital Signature Algorithm). Lo standard DSS è stato originariamente proposto nel 1991 e riveduto nel 1993 in risposta a un'indagine pubblica riguardante la sicurezza di questo schema. Vi è stata un'ulteriore revisione di minore entità nel 1996. Nel 2000 è stata emessa una versione estesa dello standard con il nome di FIPS 186-2. Quest'ultima versione incorpora anche gli algoritmi di firma digitale basati su RSA e sulla crittografia a curva ellittica. In questa parte del capitolo si parlerà dell'algoritmo DSS originario.

L'approccio DSS

Lo standard DSS utilizza un algoritmo progettato per fornire solo la funzionalità di firma digitale. A differenza di RSA, non può essere utilizzato per la crittografia o per lo scambio delle chiavi. Ciononostante si tratta di una tecnica a chiave pubblica.

La Figura 13.1 confronta l'approccio DSS per la generazione delle firme digitali con quello utilizzato da RSA. Nell'approccio RSA, il messaggio da firmare viene inviato a una funzione hash che produce un codice hash sicuro di lunghezza fissa. Questo codice hash viene poi crittografato utilizzando la chiave privata del mittente in modo da formare la firma. Vengono quindi trasmessi sia il messaggio che la firma. Il destinatario prende il messaggio e produce un codice hash. Inoltre il destinatario esegue la decrittografia della firma utilizzando la chiave pubblica del mittente. Se il codice hash calcolato coincide con la firma decrittografata, questa viene accettata come valida. Poiché solo il mittente conosce la chiave privata, solo lui può aver prodotto una firma valida.

Anche l'approccio DSS utilizza una funzione hash. Il codice hash viene fornito come input a una funzione di firma insieme a un numero casuale k generato per questa particolare firma. La funzione di firma dipende anche dalla chiave privata del mittente (PR_a) e da un insieme di parametri noti a un gruppo di responsabili della trasmissione. In pratica questo insieme costituisce una chiave pubblica globale (PU_G).[4] Il risultato è una firma costituita da due componenti: s e r.

[4] È anche possibile fare in modo che questi parametri aggiuntivi varino per ciascun utente in modo che facciano parte della chiave pubblica di un utente. In pratica è molto più probabile che venga utilizzata una chiave pubblica globale distinta dalla chiave pubblica di ciascun utente.

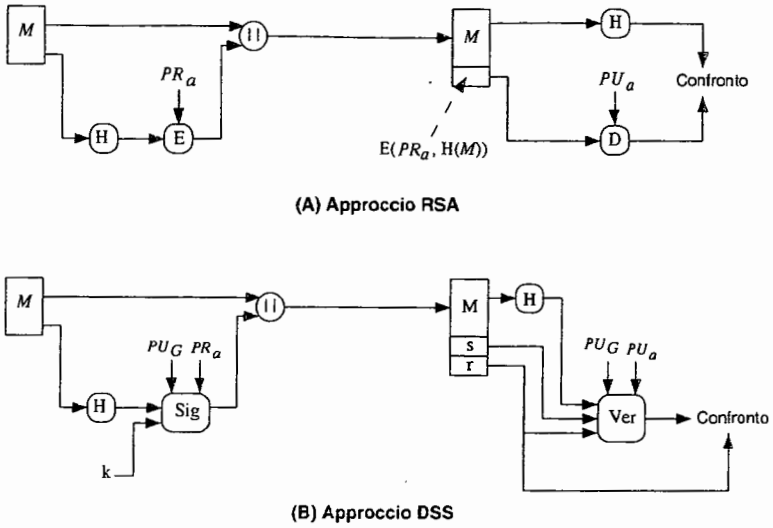


Figura 13.1 Due approcci alle firme digitali.

All'estremità ricevente, viene generato il codice hash del messaggio in arrivo. Questo e la firma vengono inviati in input alla funzione di verifica. La funzione di verifica dipende dalla chiave pubblica globale e dalla chiave pubblica del mittente (PU_a) che viene associata alla chiave privata del mittente. Se la firma è valida l'output della funzione di verifica coincide con il valore della componente della firma r. La funzione di firma è tale che solo il mittente, conoscendo la chiave privata, possa aver prodotto una firma valida.

Ora si analizzeranno i dettagli dell'algoritmo.

L'algoritmo DSA (Digital Signature Algorithm)

L'algoritmo DSA si basa sulla difficoltà di calcolare i logaritmi discreti (vedere il Capitolo 8) e impiega gli schemi originariamente presentati da ElGamal [ELGA85] e Schnorr [SCHN91].

La Figura 13.2 riepiloga il funzionamento dell'algoritmo. Vi sono tre parametri pubblici che possono essere comuni a un gruppo di utenti. Viene scelto un numero primo q di 160 bit. Successivamente viene scelto un numero primo p con una lunghezza compresa fra 512 e 1024 bit tale che q divida $(p - 1)$. Infine viene scelto g nella forma $h^{(p-1)/q} \bmod p$, dove h è un intero compreso fra 1 e $(p - 1)$ con g maggiore di 1.[5]

Avendo a disposizione questi numeri, ciascun utente seleziona una chiave privata e genera una chiave pubblica. La chiave privata x deve essere compresa fra 1 e $(q - 1)$ e

[5] In termini di teoria dei numeri, g è dell'ordine di q modulo p; vedere il Capitolo 8.

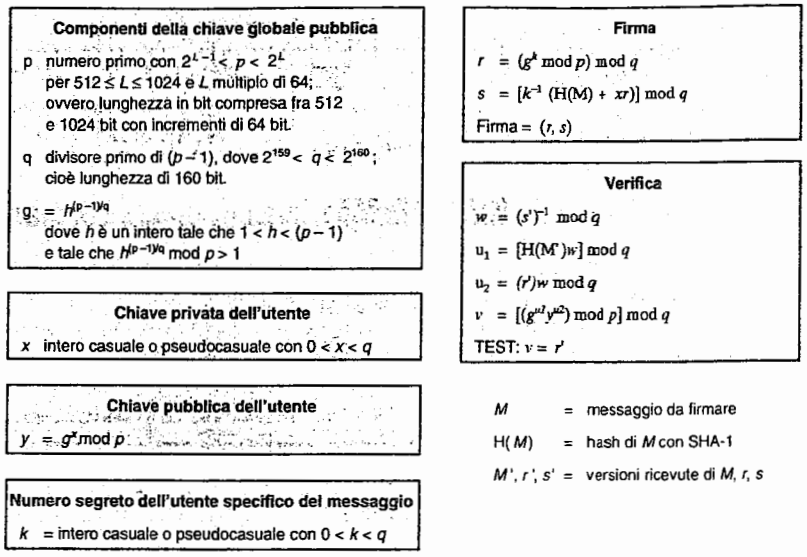


Figura 13.2 L'algoritmo DSA (Digital Signature Algorithm).

dovrebbe essere scelta casualmente o pseudocasualmente. La chiave pubblica viene calcolata dalla chiave privata come $y = g^x \bmod p$. Il calcolo di y sulla base di x è relativamente facile. Tuttavia, data la chiave pubblica y, si ritiene computazionalmente impossibile determinare x che è il logaritmo discreto di y in base g modulo p (vedere il Capitolo 8).

Per creare una firma, un utente calcola due quantità, r e s, che sono funzioni dei componenti della chiave pubblica (p, q, g), della chiave privata dell'utente (x), del codice hash del messaggio, $H(M)$, e di un intero aggiuntivo k che dovrebbe essere generato casualmente o pseudocasualmente ed essere univoco in ciascuna operazione di firma.

All'estremità ricevente, la verifica viene effettuata utilizzando le formule rappresentate nella Figura 13.2. Il destinatario genera una quantità v che è funzione delle componenti della chiave pubblica globale, della chiave pubblica del mittente e del codice hash del messaggio in arrivo. La firma è valida se questa quantità corrisponde alla componente r della firma.

La Figura 13.3 rappresenta le funzioni di firma e verifica.

La struttura dell'algoritmo, come indicato nella Figura 13.3, è piuttosto interessante. Si noti che il test finale è sul valore r che non dipende affatto dal messaggio. Al contrario r è funzione di k e dei tre componenti della chiave pubblica globale. L'inverso moltiplicativo di k (modulo q) viene passato a una funzione che usa come input anche il codice hash del messaggio e la chiave privata dell'utente. La struttura di questa funzione è tale che il destinatario possa recuperare r utilizzando il messaggio in arrivo e la firma, la chiave pubblica dell'utente e la chiave pubblica globale. Certamente non è evidente dalle Figure 13.2 e 13.3 che tale schema possa funzionare, ma la dimostrazione è disponibile nel sito Web di questo volume.

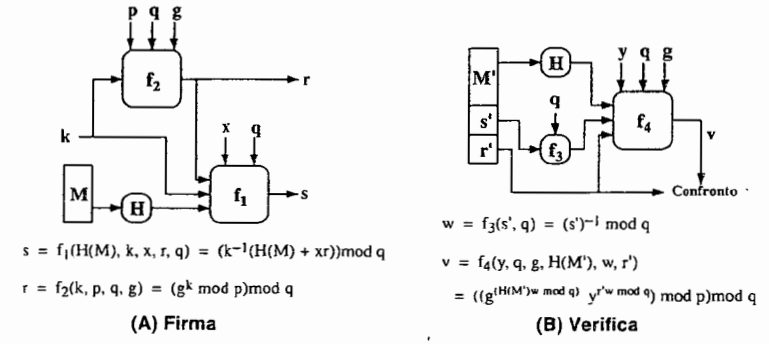


Figura 13.3 Firma e verifica DSS.

Data la difficoltà di calcolare i logaritmi discreti, è impossibile per un estraneo risalire a k partendo da r oppure risalire a x partendo da s.

Un altro elemento degno di nota è che l'unica operazione computazionalmente pesante nella generazione della firma è il calcolo esponenziale $g^k \bmod p$. Poiché questo valore non dipende dal messaggio che deve essere firmato, può anche essere precalcolato. Pertanto un utente potrebbe precalcolare un certo numero di valori di r da utilizzare per firmare i documenti ogni volta che se ne presenti l'occasione. L'unica altra operazione piuttosto pesante è la determinazione dell'inverso moltiplicativo k^{-1}. Anche in questo caso è però possibile precalcolare alcuni di questi valori.

13.4 Letture e siti Web consigliati

[AKL83] è il classico documento sulle firme digitali ed è tuttora molto importante. Un'indagine più recente e comunque eccellente è [MITC92].

AKL83 S. Akl. "Digital Signatures: A Tutorial Survey". *Computer*, Febbraio 1983.
MITC92 C. Mitchell, F. Piper e P. Wild. "Digital Signatures". In [SIMM92a].

Sito Web consigliato

- **Digital Signatures:** pagina NIST con informazioni relative alle opzioni di firma digitale approvate da NIST.

13.5 Termini chiave, domande di ripasso e problemi

Termini chiave

Algoritmo DSA (Digital Signature Algorithm)
Arbitro
Attacco a replay
Attacco suppress-replay
Autenticazione monodirezionale
Autenticazione reciproca
Firma digitale
Firma digitale arbitrata
Firma digitale diretta
Nonce
Standard DSS (Digital Signature Standard)
Timestamp

Domande di ripasso

13.1 Elencare due dispute che possono sorgere nel contesto dell'autenticazione dei messaggi.
13.2 Quali sono le proprietà che deve avere una firma digitale?
13.3 Quali requisiti deve soddisfare uno schema di firma digitale?
13.4 Qual è la differenza fra firma digitale diretta e arbitrata?
13.5 In quale ordine devono essere applicate le funzioni di firma e di segretezza di un messaggio? Motivare la risposta.
13.6 Quali sono le minacce associate allo schema a firma digitale diretta?
13.7 Fornire esempi di attacchi a replay.
13.8 Elencare tre approcci generali per rispondere agli attacchi a replay.
13.9 Che cos'è un attacco suppress-replay?

Problemi

13.1 Modificare le tecniche a firma digitale della Tabella 13.1A e 13.1B per consentire al destinatario di verificare la firma.
13.2 Modificare la tecnica a firma digitale della Tabella 13.1C per evitare la tripla crittografia dell'intero messaggio.
13.3 Nella discussione relativa alla Tabella 13.1C, si è detto che è impossibile creare alleanze fraudolente. In realtà esiste una possibilità. Descriverla e spiegare perché la sua credibilità sarebbe talmente scarsa che tale possibilità può essere tranquillamente ignorata.

13.4 Nel Paragrafo 13.2 è stato trattato brevemente lo schema a chiave pubblica proposto in [WOO92a] per la distribuzione delle chiavi segrete. La versione riveduta include ID_A nei passi 5 e 6. Questa revisione aveva lo scopo di sventare quale attacco?
13.5 Il protocollo cui si fa riferimento nel Problema 13.1 può essere ridotto da 7 a 5 passi con la seguente sequenza.

(1) A → B:
(2) B → KDC:
(3) KDC → B:
(4) B → A:
(5) A → B:

Mostrare il messaggio trasmesso in ciascun passo. *Suggerimento:* il messaggio finale in questo protocollo è uguale al messaggio finale del protocollo originario.
13.6 Far riferimento all'attacco suppress-replay descritto nel Paragrafo 13.2.
A. Dare un esempio di attacco quando l'orologio di una delle parti è in anticipo rispetto a quello del centro KDC.
B. Dare un esempio di attacco quando l'orologio di una delle parti è in anticipo rispetto a quello dell'altra parte.
13.7 Esistono tre modi tipici per utilizzare i nonce per la verifica. Supporre che N_a sia un nonce generato da A, che A e B condividano la chiave K e che f () sia una funzione come per esempio un incremento. I tre modi sono i seguenti.

Modo 1	Modo 2	Modo 3
(1) A → B: N_a	(1) A → B: $E(K, N_a)$	(1) A → B: $E(K, N_a)$
(2) B → A: $E(K, N_a)$	(2) B → A: N_a	(2) B → A: $E(K, f(N_a))$

Descrivere le situazioni in cui ciascuna modalità è appropriata.
13.8 Il dottor Watson attese pazientemente che Sherlock Holmes finisse. "Qualche interessante problema da risolvere, Holmes?" Chiese.
"Oh, non esattamente. Ho semplicemente controllato la posta elettronica e ho fatto un paio di esperimenti di rete invece dei soliti esperimenti chimici. Al momento ho un solo cliente e ho già risolto il suo problema. Se mi ricordo bene, una volta mi hai detto che hai l'hobby della criptologia e dunque questo argomento forse ti può interessare".
"Io sono solo un appassionato di crittografia, Holmes, ma naturalmente sono interessato al problema. Di che cosa si tratta?".
"Il mio cliente è Mr. Hosgrave, il direttore di una banca piccola ma dinamica. La banca è completamente computerizzata e, naturalmente, fa largo uso delle reti di comunicazione. La banca usa già l'algoritmo RSA per proteggere i propri dati e per firmare i documenti comunicati. Ora la banca vuole introdurre qualche variazione

nelle proprie procedure; in particolare deve poter firmare alcuni documenti con *due* firme in modo che:

1. Il primo firmatario prepara il documento, appone la sua firma e passa il documento al secondo firmatario.
2. Il secondo firmatario verifica innanzitutto che il documento sia stato effettivamente firmato dal primo firmatario. A questo punto appone la propria firma al documento in modo che il destinatario, e qualsiasi altro membro pubblico possa verificare che il documento è stato firmato da entrambi. Inoltre, solo il secondo firmatario deve essere in grado di verificare la firma del documento dopo il passo 1; in pratica il destinatario (o qualsiasi membro pubblico) deve essere in grado di verificare solo il documento completo con entrambe le firme ma non il documento nella sua forma intermedia, con una sola firma. Inoltre la banca vorrebbe utilizzare i moduli esistenti che supportano le firme digitali in stile RSA".

"Hmmm, so che RSA può essere utilizzato per firmare una sola volta i documenti, Holmes. Immagino che tu abbia risolto il problema di Mr. Hosgrave tramite una generalizzazione appropriata delle firme digitali RSA".

"Esattamente, Watson" rispose Sherlock Holmes. "Originariamente la firma digitale RSA aveva lo scopo di crittografare il documento tramite la chiave di crittografia privata del firmatario "d" e la firma poteva essere verificata da chiunque decrittografandola con la chiave di crittografia pubblica "e". Si può verificare che la firma S è stata eseguita dalla persona che conosce d e che sarà l'unico firmatario. Ora il problema di Mr. Hosgrave può essere risolto nello stesso modo con una leggera generalizzazione del processo, ovvero .....". Completare la descrizione

13.9 L'algoritmo DSA specifica che se il processo di generazione della firma produce un valore $s = 0$, si deve generare un nuovo valore di k e la firma deve essere ricalcolata. Perché?

13.10 Cosa accade se viene violato un valore k utilizzato nella creazione di una firma DSA?

13.11 Il documento DSS include un algoritmo consigliato per controllare la primalità di un numero.

1. **Scegliere w:** sia w un intero casuale dispari. Quindi $(w - 1)$ è pari e può essere espresso sotto forma di $2^a m$ con m dispari. Ovvero 2^a è la massima potenza di 2 che divide $(w - 1)$.
2. **Generare b:** un numero casuale tale che $1 < b < w$.
3. **Esponenziazione:** assegnare $j = 0$ e $z = b^m \bmod w$.
4. **Test OK?** Se $j = 0$ e $z = 1$ o se $z = w - 1$, allora w passa il test e può essere primo; procedere con il passo 8.
5. **Terminare?** Se $j > 0$ e $z = 1$, allora w non è primo; terminare l'algoritmo per questo w.
6. **Incrementare j:** $j = j + 1$. Se $j < a$, assegnare $z = z^2 \bmod w$ e procedere con il passo 4.
7. **Terminare:** w non è primo; terminare l'algoritmo per questo w.
8. **Nuovo test:** se è stato controllato un numero sufficiente di valori casuali di b, accettare w come primo e chiudere l'algoritmo, altrimenti procedere con il passo 2.
   A. Spiegare come funziona questo algoritmo.
   B. Mostrare che è equivalente al test di Miller-Rabin descritto nel Capitolo 8.

13.12 Con DSS, poiché viene generato un valore di k differente per ogni firma, anche se lo stesso messaggio venisse firmato due volte in due diverse occasioni, le firme sarebbero differenti. Questo non accade nelle firme RSA. Quali sono le implicazioni pratiche di questa differenza?

13.13 Si consideri il problema della creazione dei parametri di dominio per DSA. Si supponga di avere già identificato i numeri primi p e q tali che $q|(p - 1)$. Ora si deve trovare $g \in Z_p$ con g dell'ordine di $q \bmod p$. Si considerino i due algoritmi seguenti:

Algoritmo 1	Algoritmo 2
repeat	**repeat**
select $g \in Z_p$	**select** $h \in Z_p$
$h \leftarrow g^q \bmod p$	$g \leftarrow h^{(p-1)/p} \bmod p$
until ($h = 1$ and $g \neq 1$)	**until** ($g \neq 1$)
return g	**return** g

A. Dimostrare che il valore restituito dal primo algoritmo è di ordine q.
B. Dimostrare che il valore restituito dal secondo algoritmo è di ordine q.
C. Supporre $p = 40\,193$ e $q = 157$. Quante iterazioni verranno eseguite dal primo algoritmo prima di trovare un generatore?
D. Se p è di 1024 bit e q è di 160 bit, sarebbe consigliabile il primo algoritmo per trovare g? Spiegare.
   Supporre $p = 40\,193$ e $q = 157$. Qual è la probabilità che il secondo algoritmo calcoli un generatore nella prima iterazione del suo ciclo? Per rispondere al quesito può essere utile ricordare: $\sum_{d|n} \varphi(d) = n$.

13.14 Si potrebbe voler sviluppare una variante di Diffie-Hellman che possa essere utilizzata come firma digitale. Ecco una variante più semplice di DSA e che non richiede un numero casuale segreto oltre alla chiave privata.

Elementi pubblici

q numero primo
α $\alpha < q$ e α è una radice primitiva di q.

Chiave privata

X $X < q$

Chiave pubblica

$Y = \alpha^X \bmod q$

Per firmare un messaggio M, calcolare $h = H(M)$, il codice hash del messaggio. Si richiede che $\gcd(h, q - 1)) = 1$. In caso negativo, aggiungere il codice hash al messaggio e calcolare un nuovo valore hash. Ripetere queste operazioni fino a produrre un codice hash che sia primo relativo di $(q - 1)$. Poi calcolare Z per soddisfare $Z \times h \equiv X \pmod{q - 1}$. La firma del messaggio è α^Z. Per verificare la firma, un utente verifica che $Y = (\alpha^Z)^h = \alpha^X \bmod q$.

A. Mostrare che questo schema funziona. Ovvero, mostrare che il processo di verifica produce un'uguaglianza se la firma è valida.

B. Mostrare che lo schema è inaccettabile descrivendo una semplice tecnica per falsificare la firma di un utente in un messaggio arbitrario.

13.15 Una vecchia proposta per uno schema di firma digitale con crittografia simmetrica è il seguente. Per firmare un messaggio di n bit, il mittente genera casualmente in anticipo $2n$ chiavi crittografiche di 56 bit:

$$k1, K1, k2, K2, \ldots, kn, Kn$$

che sono mantenute segrete. Il mittente prepara anche due insiemi di parametri di validazione non segreti corrispondenti, che sono resi pubblici:

$$u1, U1, u2, U2, \ldots un, Un \text{ e } v1, V1, v2, V2, \ldots, vn, Vn$$

dove

$$vi = \mathrm{E}(ki, ui),\ Vi = \mathrm{E}(ki, Ui)$$

Il messaggio M viene così firmato: all'$i$esimo bit del messaggio viene associato ki o Ki a seconda che il bit sia 0 o 1. Per esempio, se i primi tre bit del messaggio sono 011, le prime tre chiavi della firma saranno $k1$, $K2$, $K3$.

A. Come può, il destinatario, validare il messaggio?
B. Si tratta di una tecnica sicura?
C. Quante volte è possibile utilizzare in sicurezza lo stesso insieme di chiavi segrete per messaggi differenti?
D. Che problemi pratici presenta questo schema?

Parte terza

Applicazioni per la sicurezza delle reti

Nelle prime due parti del volume sono stati esaminati vari metodi di crittografia e il loro impiego nel campo della segretezza, dell'autenticazione, dello scambio delle chiavi e di tutte le funzioni correlate. La Parte terza tratta invece alcune importanti applicazioni per la sicurezza della rete che impiegano queste funzioni. Queste applicazioni possono essere utilizzate nell'ambito di una singola rete, di una intranet aziendale o dell'intera Internet.

Capitolo 14: Applicazioni per l'autenticazione

Il Capitolo 14 riguarda due delle più importanti specifiche di autenticazione attualmente in uso. Kerberos è un protocollo di autenticazione basato sulla crittografia convenzionale, che ha ricevuto un ampio supporto ed è utilizzato in molti sistemi. X.509 specifica un algoritmo di autenticazione e definisce una funzione di certificazione. Quest'ultima consente di ottenere certificati delle chiavi pubbliche in modo che la comunità di utenti possa aver fiducia nella loro validità. Questa funzionalità è impiegata come elemento costitutivo di varie applicazioni.

Capitolo 15: Sicurezza della posta elettronica

L'applicazione distribuita più utilizzata in assoluto è la posta elettronica; pertanto vi è un interesse crescente nella fornitura di servizi di autenticazione e segretezza della posta elettronica. Il Capitolo 15 esamina due approcci che probabilmente domineranno il campo della sicurezza della posta elettronica nel prossimo futuro. PGP (Pretty Good Privacy) è un meccanismo ampiamente utilizzato che non dipende da alcuna organizzazione o autorità. Pertanto è particolarmente adatto sia a singoli utenti che a essere incorporato nelle configurazioni di rete impiegate dalle grandi aziende. S/MIME (Secure/Multipurpose Internet Mail Extension) è stato sviluppato con lo scopo di divenire uno standard per Internet.

Capitolo 16: La sicurezza IP

Il protocollo IP (Internet Protocol) è l'elemento fondamentale su cui si basano Internet e le reti intranet private. La sicurezza al livello IP è, di conseguenza, fondamentale per progettare qualsiasi meccanismo di sicurezza nelle reti interconnesse. Il Capitolo 16 esamina il meccanismo di sicurezza IP, sviluppato per poter operare sia con la versione corrente di IP che con la sua versione successiva, l'emergente IPv6.

Capitolo 17: La sicurezza nel Web

La crescita esplosiva nell'impiego del World Wide Web per il commercio elettronico e per la distribuzione delle informazioni ha generato la necessità di una forte sicurezza Web. Il Capitolo 17 offre un'analisi di questo importante campo della sicurezza esaminando due standard disponibili: SSL (Secure Sockets Layer) e SET (Secure Electronic Transaction).

Capitolo 14

Applicazioni per l'autenticazione

Concetti essenziali

- **Kerberos** è un servizio di autenticazione progettato per l'impiego in ambienti distribuiti.
- Kerberos utilizza un servizio di autenticazione fidato che consente ai client e ai server di stabilire delle comunicazioni autenticate.
- La raccomandazione **X.509** definisce il formato dei certificati a chiave pubblica. Questo formato è ampiamente utilizzato in numerose applicazioni.
- Per **PKI**, o "infrastruttura per le chiavi pubbliche", si intende l'insieme dell'hardware, del software, degli utenti, delle politiche e delle procedure necessarie per creare, gestire, memorizzare, distribuire e revocare certificati digitali tramite tecniche di crittografia a chiave pubblica.
- Le implementazioni PKI utilizzano solitamente i certificati X.509.

Questo capitolo esamina alcune delle funzioni che sono state sviluppate per supportare l'autenticazione delle applicazioni e le firme digitali.

Si partirà da uno dei primi e più diffusi servizi, Kerberos. Poi verrà esaminato il servizio di autenticazione a directory X.509. Questo standard è importante come parte del servizio di directory che supporta ma anche in quanto è l'elemento costitutivo di altri standard, tra i quali S/MIME trattato nel Capitolo 15. Infine si analizzerà il concetto di infrastruttura per la gestione delle chiavi pubbliche (PKI - Public Key Infrastructure).

14.1 Kerberos

Kerberos è un servizio di autenticazione sviluppato al MIT nell'ambito del progetto Athena. Il problema risolto da Kerberos è il seguente: si immagini un ambiente distribuito aperto, in cui gli utenti, seduti alle proprie workstation, vogliano accedere a servizi su server distribuiti nella rete. I server dovrebbero essere in grado di limitare l'accesso agli utenti autoriz-

zati e di autenticare le richieste di servizi. In questo ambiente non si può lasciare alle workstation la responsabilità di identificare correttamente gli utenti dei servizi della rete. In particolare esistono tre minacce.

- Un utente potrebbe guadagnare l'accesso a una determinata workstation e fingersi un altro utente normalmente operante da quella workstation.
- Un utente potrebbe modificare l'indirizzo di rete della propria workstation in modo che la richiesta inviata dalla sua workstation sembri provenire da un'altra workstation.
- Un utente potrebbe intercettare gli scambi e utilizzare un attacco a replay per acquisire l'ingresso a un server o per disturbare il funzionamento della rete.

In ognuno di questi casi, un utente non autorizzato può essere in grado di acquisire l'accesso a servizi e dati riservati. Invece di implementare protocolli di autenticazione elaborati su ciascun server, Kerberos[1] fornisce un server di autenticazione centralizzato la cui funzione è quella di autenticare gli utenti per i server e i server per gli utenti. A differenza della maggior parte degli altri meccanismi di autenticazione descritti in questo volume, Kerberos si affida esclusivamente alla crittografia simmetrica e non alla crittografia a chiave pubblica.

Attualmente vengono utilizzate due versioni di Kerberos. La versione 4 [MILL88, STEI88] è ancora utilizzata. La Versione 5 [KOHL94] risolve alcuni dei problemi di sicurezza della versione 4 ed è stata proposta come standard per Internet (RFC 1510).[2]

Innanzitutto occorre spiegare le motivazioni che hanno portato all'approccio Kerberos. Poi, data la complessità di Kerberos, si partirà da una descrizione del protocollo di autenticazione impiegato nella versione 4. Questo consentirà di comprendere la strategia di Kerberos senza dover considerare tutti i dettagli necessari per poter resistere agli attacchi più subdoli. Infine verrà esaminata la versione 5.

Motivazioni

Se un insieme di utenti opera su personal computer dedicati, senza connessioni di rete, le risorse e i file potranno essere protetti semplicemente adottando delle misure fisiche di sicurezza su ciascun personal computer. Quando invece gli utenti sono serviti da un sistema time-sharing centralizzato, deve essere il sistema operativo di questo sistema a fornire le funzionalità di sicurezza. Il sistema operativo può prevedere delle politiche di controllo degli accessi basate sull'identità dell'utente e quindi utilizzare delle procedure di login per identificare gli utenti.

Al giorno d'oggi nessuna di queste due situazioni è molto diffusa. In genere si impiega un'architettura distribuita costituita da workstation dedicate (client) e server distribuiti o centralizzati. In questo ambiente si possono prevedere tre approcci alla sicurezza.

[1] Nella mitologia greca, Cerbero è un cane a più teste, normalmente tre, in genere con la coda di serpente, ed è posto a guardia dell'ingresso dell'Ade. Come Cerbero aveva tre teste, anche il sistema Kerberos prevedeva tre componenti per controllare gli ingressi in una rete: l'autenticazione, l'accounting e l'auditing. Queste ultime due "teste" non sono mai state implementate.

[2] Le versioni da 1 a 3 erano versioni di sviluppo, impiegate internamente. La versione 4 è la versione "originale" di Kerberos.

1. Contare sulle singole workstation client per garantire l'identità degli utenti e contare su ciascun server per imporre una politica di sicurezza basata sull'identificazione degli utenti.
2. Richiedere che i sistemi client si autentichino ai server confidando sui sistemi client per la verifica dell'identità degli utenti.
3. Chiedere all'utente di dimostrare la propria identità per ogni servizio utilizzato. Inoltre i server devono garantire la propria identità ai client.

In un piccolo ambiente chiuso, in cui tutti i sistemi sono gestiti da un'unica azienda, potrebbe bastare la prima o la seconda strategia.[3] Ma in un ambiente più aperto, in cui sono supportate connessioni di rete con altre macchine, è necessario adottare il terzo approccio per proteggere le informazioni degli utenti e le risorse ospitate dai server. Questo è l'approccio supportato da Kerberos. Kerberos presuppone la presenza di un'architettura distribuita client/server e impiega uno o più server Kerberos per fornire il servizio di autenticazione.
Il primo rapporto pubblicato su Kerberos [STEI88] elencava i seguenti requisiti:

- **Sicurezza:** un intercettatore sulla rete non deve essere in grado di ottenere le informazioni necessarie per potersi sostituire a un altro utente. Più in generale, Kerberos deve fare in modo che un potenziale avversario non lo trovi l'anello debole della rete.
- **Affidabilità:** per tutti i servizi che fanno affidamento su Kerberos per il controllo degli accessi, l'indisponibilità del servizio Kerberos significa indisponibilità dei servizi supportati. Pertanto Kerberos deve essere altamente affidabile e impiegare un'architettura a server distribuiti, in cui un sistema è in grado di supplire ai problemi di un altro.
- **Trasparenza:** teoricamente l'utente non dovrebbe essere consapevole del processo di autenticazione, tranne per il fatto di dover introdurre una password.
- **Scalabilità:** il sistema deve essere in grado di supportare un gran numero di client e server. Questo suggerisce l'impiego di un'architettura modulare e distribuita.

Per supportare questi requisiti, il meccanismo di Kerberos prevede che venga impiegato un servizio di autenticazione esterno fidato che impieghi un protocollo basato sulla proposta di Needham e Schroeder [NEED78] discussa nel Capitolo 7. Si tratta di un sistema fidato, nel senso che i client e i server confidano in Kerberos per effettuare la loro reciproca autenticazione. Supponendo che il protocollo Kerberos sia ben realizzato e che il server Kerberos stesso sia sicuro, allora il servizio di autenticazione può essere considerato sicuro.[4]

[3] Tuttavia anche in un ambiente chiuso vi può essere il rischio rappresentato da un dipendente insoddisfatto.

[4] La sicurezza del server Kerberos non deve essere data per scontata ma deve essere garantita con la massima cura (per esempio il server deve essere situato in un locale ad accesso controllato). Vale forse la pena di ricordare il destino di Cerbero che doveva essere catturato da Ercole sotto ordine di Euristeo come dodicesima fatica: Ercole trovò il grosso cane incatenato e lo afferrò per la gola. Immediatamente le tre teste tentarono di attaccarlo mentre Cerbero lo sferzava con la sua potente coda. Ercole non lasciò la morsa e Cerbero venne meno. Euristeo si sorprese di vedere Ercole ancora vivo, quando vide le tre teste e il grosso cane cui appartenevano si spaventò tanto da tornare al sicuro nella sua giara di bronzo.

Kerberos versione 4

La versione 4 di Kerberos utilizza DES in un protocollo piuttosto elaborato per garantire il servizio di autenticazione. Considerando il protocollo nella sua globalità, è difficile vedere la necessità di impiegare tutti gli elementi in esso contenuti. Pertanto si adotterà la strategia impiegata da Bill Bryant del progetto Athena [BRYA88], che costruisce il protocollo analizzando prima vari ipotetici dialoghi. Ogni successivo dialogo aggiunge ulteriori complessità per correggere i punti deboli individuati dal dialogo precedente.

Dopo avere esaminato il protocollo si vedranno altri elementi della versione 4.

Un semplice dialogo di autenticazione

In un ambiente di rete non protetto, qualsiasi client può chiedere un servizio a qualsiasi server. Il rischio ovvio per la sicurezza è quello che un utente si finga un altro utente. Un estraneo può fingersi un altro client e ottenere privilegi riservati sulle macchine server. Per risolvere questo problema, i server devono essere in grado di confermare l'identità dei client che richiedono un servizio. Ciascun server potrebbe dover svolgere questa operazione per ogni interazione client/server sebbene in un ambiente aperto questo significhi gravare ciascun server di un sostanziale carico elaborativo.

Un'alternativa è costituita dall'impiego di un server di autenticazione (AS – Authentication Server) che conosce le password di tutti gli utenti e le memorizza in un database centralizzato. Inoltre il server di autenticazione condivide una chiave segreta univoca con ciascun server. Queste chiavi vengono distribuite fisicamente oppure utilizzando un altro metodo sicuro. Si consideri il seguente ipotetico dialogo.[5]

$$(1)\ C \rightarrow AS: \quad ID_C \| P_C \| ID_V$$
$$(2)\ AS \rightarrow C: \quad Ticket$$
$$(3)\ C \rightarrow V: \quad ID_C \| Ticket$$
$$Ticket = E(K_V, [ID_C \| AD_C \| ID_V])$$

dove:

- C = client
- AS = server di autenticazione
- V = server
- ID_C = identificatore dell'utente su C
- ID_V = identificatore di V
- P_C = password dell'utente su C
- AD_C = indirizzo di rete di C
- K_V = Chiave di crittografia segreta condivisa da AS e V.

In questa situazione, l'utente si connette a una workstation e chiede l'accesso al server V. Il modulo client C nella workstation richiede la password all'utente e poi invia un messaggio

[5] La parte sinistra (prima dei due punti) indica il mittente e il destinatario; la parte destra indica il contenuto del messaggio, il simbolo || indica il concatenamento.

al server di autenticazione AS che include il codice ID e la password dell'utente, e il codice ID del server. Il server di autenticazione AS consulta il proprio database per vedere se l'utente ha fornito la password corretta e se può accedere al server V. Se l'esito è positivo, il server di autenticazione AS accetta l'utente; ora deve convincere il server che questo utente è autentico. A tale scopo, il server di autenticazione AS crea un ticket contenente il codice ID e l'indirizzo di rete dell'utente e il codice ID del server. Questo ticket viene crittografato utilizzando la chiave segreta condivisa dal server di autenticazione AS e dal server che offre il servizio. Questo ticket viene poi rinviato a C. Poiché il ticket è crittografato, non può essere modificato da C o da un estraneo.

Con questo ticket, C può quindi richiedere un servizio a V. C invia un messaggio a V contenente il proprio codice ID e il ticket. V esegue la decrittografia del ticket e verifica che l'utente indicato nel ticket sia lo stesso indicato nel codice ID non crittografato dell'utente. Se i due valori coincidono, il server considera l'utente autenticato e gli concede il servizio richiesto.

Tutti i componenti del messaggio (3) sono importanti. Il ticket viene crittografato per evitare che possa essere modificato o falsificato. Nel ticket è incluso il codice ID del server (ID_V) in modo che il server possa verificare di aver decrittografato correttamente il ticket. Nel ticket è incluso anche il codice ID_C per indicare che questo ticket è stato emesso per conto di C. Infine, il codice AD_C serve ad evitare l'attacco seguente. Un estraneo potrebbe catturare il ticket trasmesso nel messaggio (2) e utilizzare il codice ID_C trasmettendo un messaggio nella forma (3) da un'altra workstation. Il server riceverebbe un ticket valido corrispondente al codice ID dell'utente e consentirebbe l'accesso all'utente situato su quest'altra workstation. Per impedire questo attacco, il server di autenticazione AS include nel ticket l'indirizzo di rete da cui proviene la richiesta originaria. Dunque il ticket è valido solo se viene trasmesso dalla stessa workstation che lo ha richiesto.

Un dialogo di autenticazione più sicuro

Sebbene lo scenario proposto risolva alcuni dei problemi di autenticazione di un ambiente di rete aperto, ne rimangono comunque altri. In particolare occorre evidenziarne due. Innanzitutto si vorrebbe ridurre al minimo il numero di volte che un utente deve introdurre la password. Si supponga che ciascun ticket possa essere utilizzato una sola volta. Se l'utente C si connette alla workstation al mattino e desidera controllare la propria posta su un server di posta elettronica, dovrà fornire una password per ottenere un ticket per il server di posta elettronica. Se poi C vuole controllare la posta elettronica più volte durante il giorno, ogni tentativo richiederà la reintroduzione della password. Si possono migliorare le cose consentendo di riutilizzare i ticket. Per una singola sessione di login, la workstation può memorizzare il ticket del server di posta elettronica dopo averlo ricevuto e utilizzarlo per l'utente ogni volta che vorrà accedere al server di posta elettronica.

Tuttavia anche in questa situazione l'utente avrà bisogno di un nuovo ticket per ogni servizio impiegato. Se l'utente deve accedere a un server di stampa, a un server di posta elettronica, a un server di file e così via, la prima istanza di ogni accesso comporterà la richiesta di un nuovo ticket e pertanto la reintroduzione della password.

Il secondo problema è che la situazione precedente prevede la trasmissione in chiaro della password nel messaggio (1). Un estraneo potrebbe intercettare la password e utilizzarla per accedere ai servizi della vittima.

Per risolvere questi ulteriori problemi, si può introdurre uno schema che evita l'impiego di password in chiaro e la richiesta della password per ogni nuovo servizio, ovvero l'impiego di un server TGS (Ticket–Granting Server). La nuova situazione, ipotetica, è la seguente:

Per ogni sessione di login dell'utente.

$$(1)\ C \rightarrow AS: \quad ID_C \| ID_{tgs}$$
$$(2)\ AS \rightarrow C: \quad E(K_C, Ticket_{tgs})$$

Per ogni tipo di servizio:

$$(3)\ C \rightarrow TGS: \quad ID_C \| ID_V \| Ticket_{tgs}$$
$$(4)\ TGS \rightarrow C: \quad Ticket_v$$

Per ogni sessione di servizio:

$$(5)\ C \rightarrow V: \quad ID_C \| Ticket_v$$

$$Ticket_{tgs} = E(K_{tgs}, [ID_C \| AD_C \| ID_{tgs} \| TS_1 \| Lifetime_1])$$
$$Ticket_v = E(K_v, [ID_C \| AD_C \| ID_v \| TS_2 \| Lifetime2])$$

Il nuovo servizio, TGS, emette i ticket per gli utenti che sono stati autenticati dal server di autenticazione AS. Pertanto l'utente deve innanzitutto richiedere al server di autenticazione AS un ticket $Ticket_{tgs}$. Questo ticket viene salvato dal modulo client nella workstation dell'utente. Ogni volta che l'utente richiede l'accesso a un nuovo servizio, il client impiega il server TGS utilizzando il ticket per autenticarsi. Il server TGS assegna quindi un ticket per questo specifico servizio. Il client salva ciascun ticket di servizio e lo utilizza per autenticare il proprio utente a un server ogni volta che questi richiede un determinato servizio. Ecco come funziona questo meccanismo.

1. Il client richiede un ticket di assegnamento ticket per conto dell'utente, inviando al server di autenticazione AS il codice ID e la password dell'utente insieme al codice ID del server TGS per indicare una richiesta di utilizzo del servizio TGS.
2. Il server di autenticazione AS risponde con un ticket crittografato con una chiave che deriva dalla password dell'utente. Quando questa risposta arriva al client, questi chiede all'utente di specificare la propria password, genera la chiave e tenta di decrittografare il messaggio in arrivo. Se la password fornita è corretta, il ticket viene individuato con successo.

Poiché solo l'utente conosce la propria password, solo lui potrà recuperare il ticket. Pertanto la password consente di ottenere le credenziali da Kerberos senza dover trasmettere la password in chiaro. Il ticket è costituito dal codice ID e dall'indirizzo di rete dell'utente, e dal codice ID del server TGS. Però questo coincide con la prima situazione. L'idea è che questo ticket possa essere utilizzato dal client per richiedere più ticket di servizio, pertanto deve essere riutilizzabile. Tuttavia non si vuole che un estraneo sia in grado di catturare e utilizzare il ticket. Si consideri la seguente situazione: un estraneo cattura il ticket di login e attende che l'utente si disconnetta dalla propria workstation. Poi si siede alla stessa

workstation oppure configura la propria workstation con lo stesso indirizzo di rete della vittima. L'estraneo sarà quindi in grado di riutilizzare il ticket per ingannare il server TGS. Per risolvere questo problema, il ticket include un valore timestamp che indica la data in cui è stato emesso e la sua durata, indicando il periodo di validità (per esempio 8 ore). Pertanto ora il client ha un ticket riutilizzabile e non deve preoccuparsi di richiedere la password all'utente per ogni nuova richiesta di servizio. Infine si noti che il ticket di servizio è crittografato con una chiave segreta, nota solo al server di autenticazione AS e al server TGS. Questo impedisce ogni modifica al ticket. Questo ticket viene ricrittografato con una chiave basata sulla password dell'utente. Questo garantisce che il ticket possa essere recuperato solo dall'utente corretto, garantendo quindi l'autenticazione.

Ora che il client è dotato di un ticket di assegnamento ticket, l'utente potrà accedere al server tramite i passi 3 e 4.

3. Il client richiede un ticket di servizio per conto dell'utente. Per questo scopo, il client trasmette al server TGS un messaggio contenente il codice ID dell'utente, il codice ID del servizio desiderato e il ticket di assegnamento ticket.
4. Il server TGS esegue la decrittografia del ticket in ingresso e verifica il successo della decrittografia grazie alla presenza del proprio codice ID. Poi controlla che il ticket non sia scaduto. Quindi confronta il codice ID dell'utente e l'indirizzo di rete con le informazioni in arrivo, in modo da autenticare l'utente. Se all'utente è consentito l'accesso al server V, il server TGS emette un ticket per consentire l'accesso al servizio richiesto.

Il ticket di servizio ha la stessa struttura del ticket di assegnamento ticket. Infatti, poiché anche il server TGS è comunque un server, ci si può aspettare che per autenticare un client sul server TGS o su un server di applicazioni vengano impiegati gli stessi elementi. Anche questo ticket contiene sia il timestamp che la durata. Se l'utente vuole accedere allo stesso servizio in un secondo tempo, il client potrà semplicemente utilizzare il precedente ticket di servizio senza richiedere la reintroduzione della password. Si noti che il ticket è crittografato con una chiave segreta K_v nota solo al server TGS e al server di applicazioni, impedendo così ogni modifica.

Infine, con un determinato ticket di servizio, il client può avere accesso al servizio corrispondente nel passo (5).

5. Il client richiede l'accesso a un servizio per conto dell'utente trasmettendo un messaggio al server contenente il codice ID dell'utente e il ticket di servizio. Il server autentica l'utente utilizzando il contenuto del ticket.

Questa nuova situazione soddisfa i due requisiti che prevedono l'impiego di una sola richiesta di password per sessione utente e la protezione della password.

Dialogo di autenticazione per la versione 4

Sebbene la situazione precedente migliori la sicurezza rispetto al primo tentativo, rimangono altri due problemi. Il cuore del primo problema è la durata del ticket di assegnamento ticket. Se la sua durata è molto breve (pochi minuti) l'utente dovrà comunque reintrodurre più volte la password. Se invece la durata è più lunga (varie ore), un estraneo avrà più opportunità di sferrare un attacco a replay. L'estraneo potrebbe intercettare una copia del ticket di assegnamento ticket e poi attendere che l'utente legittimo si disconnetta. A questo

punto potrebbe falsificare l'indirizzo di rete dell'utente legittimo e inviare al server TGS il messaggio del passo (3). Questo garantirebbe all'estraneo un accesso illimitato alle risorse e ai file disponibili per l'utente legittimo.

Analogamente, se un estraneo dovesse catturare un ticket di servizio e utilizzarlo prima della scadenza, avrebbe accesso al servizio corrispondente.

Pertanto occorre un ulteriore requisito. Un servizio di rete (il servizio TGS o un servizio applicativo) deve essere in grado di dimostrare che la persona che utilizza un ticket è quella per cui il ticket è stato emesso.

Il secondo problema è che vi può essere un requisito di autenticazione dei server presso gli utenti. Senza questa autenticazione, un estraneo potrebbe sabotare la configurazione in modo che i messaggi inviati al server giungano altrove. Il falso server si comporterebbe come il server vero e proprio, catturando tutte le informazioni introdotte dall'utente e negandogli il servizio reale.

Si esamineranno questi problemi facendo riferimento alla Tabella 14.1 che illustra il funzionamento del protocollo Kerberos.

Innanzitutto si consideri il problema della cattura dei ticket di assegnamento ticket e della necessità di determinare che colui che presenta il ticket sia lo stesso per cui è stato emesso. Si teme che un estraneo possa sottrarre il ticket e utilizzarlo prima della scadenza. Per aggirare questo problema, si fa in modo che il server di autenticazione TGS fornisca sia al client che al server TGS un'informazione segreta in modo sicuro. Poi il client potrà dimostrare la propria identità al server TGS rivelando l'informazione segreta, ancora in modo sicuro. Un modo efficiente per ottenere ciò consiste nell'utilizzare come informazione sicura una chiave di crittografia; in Kerberos questa chiave è chiamata chiave di sessione.

La Tabella 14.1A mostra la tecnica di distribuzione della chiave di sessione. Come prima, il client invia un messaggio al server di autenticazione AS richiedendo l'accesso al server TGS. Il server AS risponde con un messaggio contenente il ticket, crittografato con una chiave derivata dalla password dell'utente (K_c). Il messaggio crittografato contiene anche una copia della chiave di sessione, $K_{c,tgs}$; gli indici indicano che si tratta di una chiave di sessione per C e il server TGS. Poiché questa chiave di sessione si trova all'interno del messaggio crittografato con K_c, solo il client dell'utente potrà leggerla. La stessa chiave di sessione viene inclusa nel ticket, che può essere letto solo dal server TGS. Pertanto la chiave di sessione è stata consegnata in modo sicuro sia a C che al server TGS.

Si noti che sono stati aggiunti vari elementi alla prima fase del dialogo. Il messaggio (1) include un valore timestamp in modo che il server AS sappia che il messaggio è recente. Il messaggio (2) include vari elementi del ticket in una forma accessibile a C. Questo consente a C di confermare che questo è il ticket per il server TGS e di conoscere la sua scadenza.

Dotato del ticket e della chiave di sessione, C è pronto per contattare il server TGS. Come prima, C invia al server TGS un messaggio che include il ticket più il codice ID del servizio richiesto (il messaggio (3) nella Tabella 14.1B). Inoltre, C trasmette un autenticatore che include il codice ID e l'indirizzo dell'utente di C più un valore timestamp. A differenza del ticket, che è riutilizzabile, l'autenticatore può essere utilizzato una sola volta e ha una durata molto breve. Il server TGS può decrittografare il ticket con la chiave che condivide con il server AS. Il ticket indica che l'utente C ha ricevuto la chiave di sessione $K_{c,tgs}$. In pratica il ticket dice: "Chi usa $K_{c,tgs}$ deve necessariamente essere C". Il server TGS utilizza la chiave di sessione per decrittografare l'autenticatore.

Tabella 14.1 Riepilogo degli scambi di messaggi in Kerberos versione 4.

A. Scambio per l'autenticazione del servizio: per ottenere il ticket di assegnamento ticket

(1) C → AS : $ID_c \| ID_{tgs} \| TS_1$

(2) AS → C : $E(K_c, [K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}])$

$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2])$

B. Scambio per l'ottenimento del ticket di servizio

(3) C → TGS : $ID_v \| Ticket_{tgs} \| Autenticatore_c$

(4) TGS → C : $E(K_{c,tgs}, [K_{c,v} \| ID_v \| TS_4 \| Ticket_v])$

$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2])$

$Ticket_v = E(K_v, [K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4])$

$Autenticatore_c = E(K_{c,tgs}, [ID_C \| AD_C \| TS_3])$

C. Scambio di autenticazione client/server per ottenere il servizio

(5) C → V : $Ticket_v \| Autenticatore_c$

(6) V → C : $E(K_{c,v}, [TS_5 + 1])$ (per la mutua autenticazione)

$Ticket_v = E(K_v, [K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4])$

$Autenticatore_c = E(K_{c,v}, [ID_C \| AD_C \| TS_5])$

Quindi il server TGS può verificare il nome e l'indirizzo dall'autenticatore con quello del ticket e con l'indirizzo di rete del messaggio in arrivo. Se tutto corrisponde, il server TGS è sicuro che il mittente del ticket sia in effetti il vero possessore del ticket. In pratica l'autenticatore dice: "Nel momento TS_3 io uso $K_{c,tgs}$". Si noti che il ticket non dimostra l'identità di nessuno ma consente di distribuire in modo sicuro le chiavi. È l'autenticatore che dimostra l'identità del client. Poiché l'autenticatore può essere utilizzato una sola volta e ha una breve durata, non vi è il rischio che un estraneo possa sottrarre sia il ticket che l'autenticatore per ripresentarli in un secondo tempo.

La risposta del server TGS nel messaggio (4) segue la struttura del messaggio (2). Il messaggio viene crittografato con la chiave di sessione condivisa dal server TGS e dal client C e include la chiave di sessione che verrà condivisa fra C e il server V, il codice ID di V e il timestamp del ticket. Il ticket include la stessa chiave di sessione.

Ora C ha un ticket di servizio riutilizzabile per V. Quando C presenta questo ticket, come indicato nel messaggio 5, invia anche un autenticatore. Il server può decrittografare il ticket, recuperare la chiave di sessione e decrittografare l'autenticatore.

Se è necessaria la reciproca autenticazione, il server può rispondere come indicato nel messaggio (6) della Tabella 14.1. Il server restituisce il valore del timestamp tratto dall'autenticatore, incrementato di 1 e crittografato con la chiave di sessione. C può decrittografare questo messaggio per recuperare il timestamp incrementato. Poiché il messaggio è stato crittografato dalla chiave di sessione, C sa che può essere stato creato solo da V. Il contenuto del messaggio garantisce a C che non sia un replay di una vecchia risposta.

Alla conclusione dell'operazione, il client e il server condividono una chiave segreta. Questa chiave può essere utilizzata per crittografare i futuri messaggi scambiati dai due o per scambiare una nuova chiave di sessione casuale per questo scopo.

La Tabella 14.2 riepiloga le motivazioni di ciascuno degli elementi impiegati nel protocollo Kerberos mentre la Figura 14.1 rappresenta le operazioni svolte.

Tabella 14.2 Descrizione delle motivazioni di ciascuno degli elementi che compongono il protocollo Kerberos versione 4.

A. Scambio per il servizio di autenticazione	
Messaggio (1)	Il client richiede il ticket di assegnamento ticket.
ID_C:	Comunica al server AS l'identità dell'utente di questo client.
ID_{tgs}:	Comunica al server AS che l'utente richiede l'accesso al server TGS.
TS_1:	Consente al server AS di verificare che il clock del client sia sincronizzato con quello del server AS.
Messaggio (2)	Il server AS restituisce il ticket di assegnamento ticket.
K_c:	La crittografia si basa sulla password dell'utente, consentendo al server AS e al client di verificare la password e di proteggere il contenuto del messaggio (2).
$K_{c,tgs}$:	Copia della chiave di sessione accessibile al client; creata dal server AS per consentire scambi sicuri fra il client e il server TGS senzo che essi debbono condividere uno chiave permanente.
ID_{tgs}:	Confermo che questo ticket è per il server TGS.
TS_2:	Informa il client dell'istante in cui è stato emesso questo ticket.
$Lifetime_2$:	Informa il client della durata di questo ticket.
$Ticket_{tgs}$:	Ticket do utilizzare dal client per accedere al server TGS.
B. Scambio del ticket di assegnamento ticket	
Messaggio (3)	Il client richiede il ticket di assegnamento ticket.
ID_V:	Comunica al server TGS che l'utente richiede l'accesso al server V.
$Ticket_{tgs}$:	Assicura al server TGS che questo utente è stato autenticato dal server AS.
$Autenticatore_c$:	Generato dal client per convalidare il ticket.
Messaggio (4)	Il server TGS restituisce il ticket di assegnamento ticket.
$K_{c,tgs}$	Chiave condivisa solo da C e dal server TGS; protegge il contenuto del messaggio (4).
$K_{c,v}$:	Copia dello chiave di sessione accessibile al client; creata dal server TGS per consentire scambi sicuri fra il client e il server senza richiedere la condivisione di una chiave permanente.
ID_V:	Confermo che questo ticket è per il server V.
TS_4:	Informa il client dell'istante in cui è stato emesso il ticket.
$Ticket_V$:	Ticket da utilizzare dal client per accedere al server V.
$Ticket_{tgs}$:	Riutilizzabile in modo che l'utente non debba ripetere l'introduzione dello password.
K_{tgs}:	Il ticket è crittografato con una chiave nota solo al server AS e al server TGS per evitare falsificazioni.
$K_{c,tgs}$:	Copia della chiave di sessione accessibile al server TGS; utilizzata per decrittografare l'autenticatore, autenticando pertonto il ticket.
ID_C:	Indica il vero proprietario di questo ticket.
AD_C:	Impedisce l'uso del ticket da un workstation diverso da quella che ha inizialmente richiesto il ticket.
ID_{tgs}:	Garantisce al server che il ticket sia stato decrittografato correttamente.

(segue)

Tabella 14.2 Descrizione delle motivazioni di ciascuno degli elementi che compongono il protocollo Kerberos versione 4 *(continua)*.

B. Scambio del ticket di assegnamento ticket	
TS_2:	Informa il server TGS del momento in cui è stato emesso questo ticket.
$Lifetime_2$:	Impedisce attacchi a reploy dopo che il ticket è scaduto.
$Autenticatore_c$:	Garantisce al server TGS che il presentatore del ticket è lo stesso client per il quale è stato emesso il ticket; ha una durota molto breve per impedire gli attacchi a reploy.
$K_{c,tgs}$:	L'autenticatore viene crittografato con una chiave nota sola al client e al server TGS per impedire falsificazioni.
ID_C:	Deve corrispondere al codice ID nel ticket per autenticorlo.
AD_C:	Deve corrispondere con l'indirizzo nel ticket per autenticarlo.
TS_3:	Informa il server TGS del momento in cui è stato generoto questo autenticatore.
C. Scambio dell'autenticazione client/server	
Messaggio (5)	Il client richiede il servizio.
$Ticket_v$:	Garantisce al server che questo utente è stato autenticato dal server AS.
$Autenticatore_c$:	Generato dal client per convalidare il ticket.
Messaggio (6)	Autenticazione opzionale del server per il client.
$K_{c,v}$:	Garantisca a C che questo messaggio proviene da V.
$TS_5 + 1$:	Garantisca a C che questo non è una ripetizione di una vecchio risposta.
$Ticket_v$:	Riutilizzabile, in modo che il client non debba richiedere un nuovo ticket dal server TGS per ogni accesso allo stesso server.
K_v:	Il ticket viene crittografato con una chiave nota solo al server TGS e al server, per impedire falsificazioni.
$K_{c,v}$:	Copia della chiave di sessione accessibile al client; utilizzata per decrittografare l'autenticatore autenticondo pertanto il ticket.
ID_C:	Indica il possessore legittimo di questo ticket.
AD_C:	Impedisce l'uso del ticket da workstation diverse da quella per la quale è stato emesso il ticket.
ID_V:	Garantisce al server che il ticket è stato decrittografato correttomente.
TS_4:	Informa il server del momento in cui è stato emesso questo ticket.
$Lifetime_4$:	Impedisce attacchi o replay dopo la scadenza del ticket.
$Autenticator_c$:	Garantisce il server che il presentatore del ticket è lo stesso client per cui è stato emesso il ticket; ha una durata molto breve per impedire gli attacchi a replay.
$K_{c,v}$:	L'autenticatore viene crittografato con una chiove nota solo al client e al server per impedire falsificazioni.
ID_C:	Deve corrispondere al codice ID nel ticket per autenticarlo.
AD_C:	Deve corrispondere all'indirizzo nel ticket per autenticorlo.
TS_5:	Informo il server del momento in cui è stato generato questo autenticatore.

I realm Kerberos e l'impiego di più Kerberos.

L'ambiente Kerberos completo è costituito da un server Kerberos, una serie di client e una serie di server di applicazioni con i seguenti requisiti.

1. Il server Kerberos deve conservare in un database il codice utente e la codifica hash della password di tutti gli utenti che sono registrati sul server Kerberos.

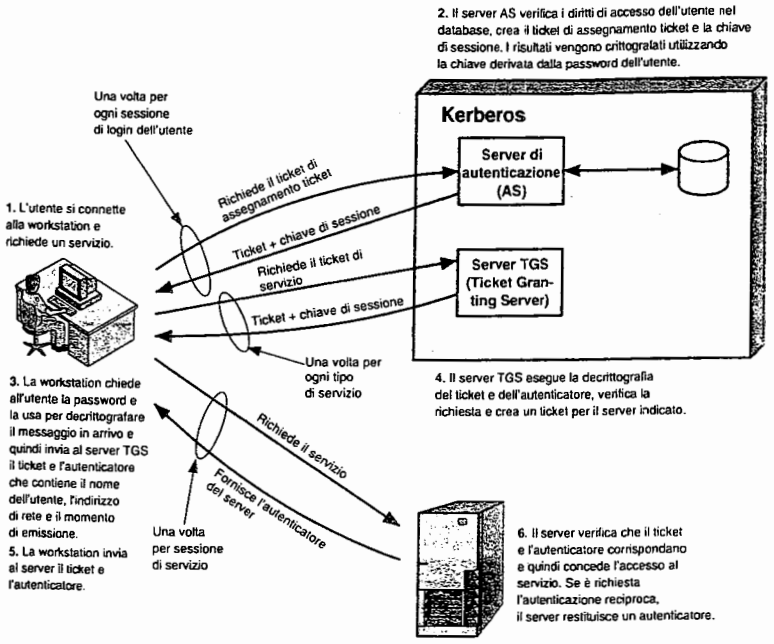


Figura 14.1 Il funzionamento di Kerberos.

2. Il server Kerberos deve condividere una chiave segreta con ciascun server. Tutti i server sono registrati sul server Kerberos.

Questo tipo di ambiente è chiamato **realm** Kerberos. Il concetto di realm può essere spiegato nel modo seguente. Un realm Kerberos è un insieme di nodi gestiti che condividono il medesimo database Kerberos. Questo database risiede sul computer master del sistema Kerberos, il quale deve essere protetto in un luogo ad accesso controllato. È possibile che esistano altre copie di sola lettura del database Kerberos su altri computer del sistema, tuttavia qualsiasi modifica al database deve essere introdotta sul computer master. L'accesso e la modifica ai contenuti di un database Kerberos richiedono la password Kerberos master.

Altro concetto collegato è quello di *principal* Kerberos: un servizio o un utente noto al sistema Kerberos. Un principal viene identificato tramite il proprio *nome principal*. I nomi principal sono costituiti da tre parti: servizio o nome utente, nome istanza e nome realm.

Le reti di client e server operanti in organizzazioni amministrativamente differenti costituiscono solitamente realm differenti. In pratica è sconsigliabile o quantomeno non conforme alla politica di amministrazione, fare in modo che gli utenti e i server in un dominio

amministrativo siano registrati su un altro server Kerberos. Tuttavia gli utenti di un realm possono dover accedere ai server presenti in altri realm e alcuni server possono voler fornire servizi agli utenti di altri realm, sempre che tali utenti siano autenticati.

Kerberos fornisce un meccanismo per supportare tale autenticazione fra realm differenti. Perché due realm supportino l'autenticazione, viene aggiunto un terzo requisito.

3. Il server Kerberos di ciascun realm interoperante condivide una chiave segreta con il server dell'altro realm. I due server Kerberos sono registrati l'uno con l'altro.

Il meccanismo richiede che il server Kerberos di un realm lasci al server Kerberos dell'altro realm la responsabilità di autenticare i suoi utenti. Inoltre i server del secondo realm devono confidare nel server Kerberos del primo realm.

Sulla base di queste regole, si può descrivere il meccanismo nel seguente modo (Figura 14.2): un utente che desidera un servizio da un server di un altro realm ha bisogno di un ticket per tale server. Il client dell'utente segue la normale procedura per accedere al server TGS locale e richiede un ticket di assegnamento ticket per un server TGS remoto (il server TGS dell'altro realm). A questo punto il client può richiedere al server TGS remoto un ticket di servizio per il server desiderato che si trova nel realm del server TGS remoto.

I dettagli sugli scambi illustrati nella Figura 14.2 sono i seguenti (da confrontare con la Tabella 14.1).

$$(1)\ C \rightarrow AS: \quad ID_C \| ID_{tgs} \| TS_1$$

$$(2)\ AS \rightarrow C: \quad E(K_C, [K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}])$$

$$(3)\ C \rightarrow TGS: \quad ID_{tgsrem} \| Ticket_{tgs} \| Autenticatore_c$$

$$(4)\ TGS \rightarrow C: \quad E(K_{c,tgs}, [K_{c,tgsrem} \| ID_{tgsrem} \| TS_4 \| Ticket_{tgsrem}])$$

$$(5)\ C \rightarrow TGS_{rem}: \quad ID_{vrem} \| Ticket_{tgsrem} \| Autenticatore_c$$

$$(6)\ TGS_{rem}: \quad E(K_{c,tgsrem}, [K_{c,vrem} \| ID_{vrem} \| TS_6 \| Ticket_{vrem}])$$

$$(7)\ C \rightarrow V_{rem}: \quad Ticket_{vrem} \| Autenticatore_c$$

Il ticket presentato al server remoto (V_{rem}) indica il realm in cui è stato originariamente autenticato l'utente. Il server quindi decide se onorare o meno la richiesta remota.

Un problema di questo approccio è il fatto che non è particolarmente adatto a un numero elevato di realm. Se vi sono N realm, vi devono essere $N(N-1)/2$ scambi di chiavi sicure per fare in modo che ciascun realm Kerberos possa interrogare ogni altro realm Kerberos.

Kerberos versione 5

La versione 5 di Kerberos è specificata nel documento RFC 1510 e presenta vari miglioramenti rispetto alla versione 4 [KOHL94]. Per iniziare si fornirà una panoramica dei cambiamenti intercorsi fra la versione 4 e la versione 5 e quindi si parlerà della versione 5 del protocollo.

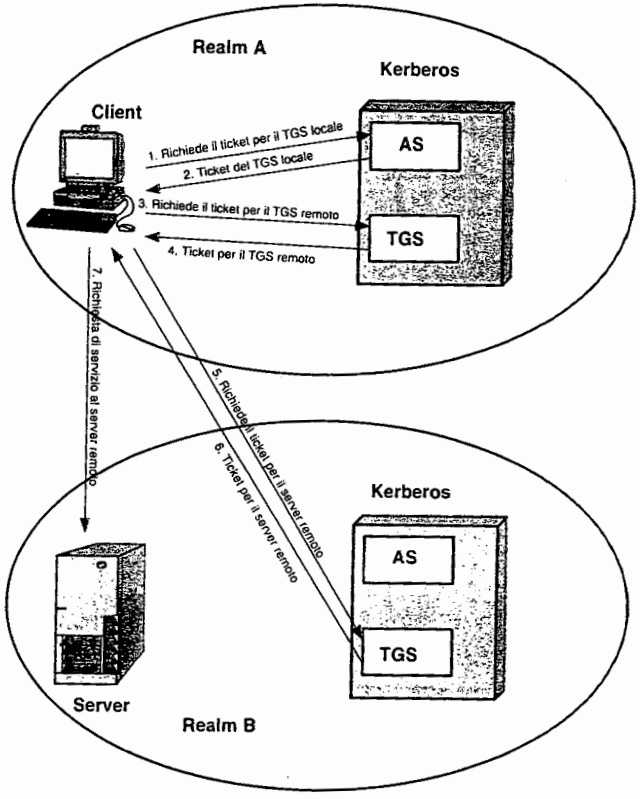


Figura 14.2 Richiesta di servizio in un altro realm.

Differenze fra le versioni 4 e 5

La versione 5 di Kerberos ha lo scopo di risolvere i limiti della versione 4 in due campi: problemi ambientali e problemi tecnici. Ecco brevemente i miglioramenti in questi campi.[6]

La versione 4 di Kerberos è stata sviluppata per l'impiego nell'ambito del progetto Athena e, di conseguenza, non rispondeva completamente alle necessità di un utilizzo generale. Questo ha portato ai seguenti **problemi ambientali**.

1. **Dipendenza dal sistema di crittografia:** la versione 4 richiede l'impiego di DES. Le restrizioni all'esportazione di DES e i dubbi sulla sua resistenza rappresentavano pertanto dei problemi. Nella versione 5, il testo cifrato è contrassegnato con un identificatore del tipo di crittografia: questo consente di impiegare qualsiasi tecnica di crittografia. Le chiavi di crittografia sono contrassegnate con il tipo e la lunghezza, consentendo di utilizzare la stessa chiave in algoritmi differenti e di specificare più varianti di un determinato algoritmo.
2. **Dipendenza dal protocollo IP:** la versione 4 richiede l'impiego di indirizzi IP. Altri tipi di indirizzi, come gli indirizzi di rete ISO, non vengono considerati. La versione 5 contrassegna gli indirizzi di rete con tipo e lunghezza e pertanto consente di impiegare qualsiasi tipo di indirizzo di rete.
3. **Ordinamento dei byte nel messaggio:** nella versione 4, il mittente di un messaggio impiega un ordinamento dei byte a scelta e contrassegna il messaggio per indicare il byte meno significativo nell'indirizzo più basso oppure il byte più significativo nell'indirizzo più basso. Queste tecniche funzionano ma non seguono le convenzioni standard. Nella versione 5, tutte le strutture dei messaggi sono definite utilizzando le notazioni ASN.1 (Abstract Syntax Notation One) e BER (Basic Encoding Rules) che definiscono un ordinamento non ambiguo dei byte.
4. **Durata del ticket:** nella versione 4 la durata dei ticket è codificata in 8 bit con unità di cinque minuti. Pertanto la durata massima può essere $2^8 \times 5 = 1280$ minuti, poco oltre 21 ore. Si tratta di una soluzione non adeguata per alcune applicazioni (per esempio lunghe simulazioni che richiedono credenziali Kerberos valide per tutta la durata dell'esecuzione). Nella versione 5 i ticket includono un tempo di inizio e un tempo di fine espliciti, consentendo ticket di durata arbitraria.
5. **Inoltro dell'autenticazione:** la versione 4 non consente di inoltrare le credenziali emesse per un determinato client a qualche altro host o l'impiego in altri client. Questo consentirebbe a un client di accedere a un server che a sua volta potrebbe accedere a un altro server per conto del client. Per esempio, un client emette una richiesta per il server di stampa che poi accede al file del client da un server di file, utilizzando per l'accesso le credenziali del client. La versione 5 fornisce questa funzionalità.
6. **Autenticazione fra realm differenti:** nella versione 4 l'interoperabilità fra N realm richiede un numero di relazioni fra server Kerberos che, come si è detto in precedenza, è dell'ordine di N^2. Come si vedrà fra breve, la versione 5 supporta un metodo che richiede meno relazioni.

Oltre a questi problemi ambientali, esistono anche dei **problemi tecnici** insiti nel protocollo versione 4. La maggior parte di questi problemi è stata documentata in [BELL90] e la versione 5 tenta di porvi rimedio. Ecco i problemi riscontrati.

1. **Doppia crittografia:** nella Tabella 14.1 si può notare (messaggi 2 e 4) che i ticket forniti ai client vengono crittografati due volte, una volta con la chiave segreta del server di destinazione e poi nuovamente con una chiave segreta nota al client. La seconda crittografia non è necessaria e rappresenta uno spreco computazionale.
2. **Crittografia PCBC:** la crittografia nella versione 4 utilizza una modalità non standard di DES chiamata PCBC (Propagating Cipher Block Chaining).[7] È stato dimostrato che questa modalità è vulnerabile a un attacco con scambio di blocchi di testo cifrato [KOHL89]. PCBC aveva lo scopo di fornire un controllo di integrità come parte del-

[6] Questa discussione segue la presentazione contenuta in [KOHL94].

[7] Se ne parlerà nell'Appendice 14.A.

l'operazione di crittografia. La versione 5 fornisce meccanismi di integrità espliciti che consentono di utilizzare per la crittografia la modalità standard CBC. In particolare si aggiunge al messaggio un checksum o un codice hash, prima di crittografarlo in modalità CBC.

3. **Chiavi di sessione:** ciascun ticket include una chiave di sessione che viene utilizzata dal client per crittografare l'autenticatore inviato al servizio associato al ticket. Inoltre la chiave di sessione può essere successivamente utilizzata dal client e dal server per proteggere i messaggi scambiati durante tale sessione. Tuttavia, poiché lo stesso ticket può essere utilizzato ripetutamente per acquisire un servizio da un determinato server, vi è il rischio che un estraneo possa riprodurre i messaggi di una vecchia sessione inviandoli al client o al server. Nella versione 5 il client e il server possono negoziare una chiave di sottosessione che viene utilizzata solo per una connessione. Un nuovo accesso da parte del client provoca l'impiego di una nuova chiave di sottosessione.
4. **Attacchi alle password:** entrambe le versioni sono vulnerabili agli attacchi alle password. Il messaggio inviato dal server AS al client comprende materiale crittografato con una chiave che si basa sulla password del client.[8] Un estraneo può catturare questo messaggio e tentare di decrittografarlo provando varie password. Se il risultato della decrittografia ha l'aspetto corretto, l'estraneo ha scoperto la password del client e potrà successivamente utilizzarla per acquisire le credenziali di autenticazione da Kerberos. Questo è lo stesso tipo di attacco alla password descritto nel Capitolo 18, che prevede lo stesso tipo di contromisure. La versione 5 fornisce un meccanismo chiamato preautenticazione che dovrebbe complicare, ma non impedire, gli attacchi alle password.

Dialogo di autenticazione in Kerberos versione 5

La Tabella 14.3 riepiloga il dialogo di base che si verifica nella versione 5. È più facile descriverlo confrontandolo con quello che si svolge nella versione 4 (Tabella 14.1).

Innanzitutto si consideri lo **scambio di servizio per l'autenticazione**. Il messaggio (1) è la richiesta di ticket di assegnamento ticket da parte del client. Come prima, include i codici ID dell'utente e dal server TGS. Vengono però aggiunti i seguenti elementi.

- **Realm:** identifica il realm dell'utente.
- **Opzioni:** usate per richiedere l'impostazione di determinati flag nel ticket restituito.
- **Tempi:** utilizzati dal client per richiedere le seguenti impostazioni temporali nel ticket.
  - from: il tempo di inizio della validità del ticket richiesto.
  - till: il tempo di scadenza della validità del ticket richiesto.
  - rtime: il tempo di rinnovo richiesto.
- **Nonce:** un valore casuale che verrà ripetuto nel messaggio (2) per assicurarsi che la risposta non sia un replay inviato da un estraneo.

Il messaggio (2) restituisce un ticket di assegnamento ticket che identifica le informazioni per il client e un blocco crittografato utilizzando la chiave di crittografia derivata dalla password dell'utente. Questo blocco include la chiave di sessione che verrà utilizzata fra il client e il server TGS, i tempi specificati nel messaggio (1), il valore nonce del messaggio (1) e le informazioni per l'identificazione del server TGS. Il ticket stesso include la chiave

[8] L'Appendice 14.A descrive il mapping delle password sulle chiavi di crittografia.

di sessione, le informazioni di identificazione del client, i valori temporali richiesti e i flag che riflettono lo stato di questo ticket e le opzioni richieste. Questi flag introducono nuove funzionalità significative nella versione 5. Per il momento si rimanda la discussione su questi flag per concentrarsi sulla struttura generale del protocollo versione 5.

Innanzitutto si deve confrontare lo **scambio del ticket di servizio** per le versioni 4 e 5. Come si può vedere il messaggio (3) di entrambe le versioni include un autenticatore, un ticket e il nome del servizio richiesto. Inoltre la versione 5 fornisce i tempi e le opzioni richieste per il ticket più un valore nonce, tutti con funzioni simili a quelli del messaggio (1). L'autenticatore stesso è fondamentalmente uguale a quello utilizzato nella versione 4.

Il messaggio (4) ha la stessa struttura del messaggio 2 e restituisce un ticket più le informazioni richieste dal client, queste ultime crittografate con la chiave di sessione ora condivisa dal client e dal server TGS.

Infine lo **scambio di autenticazione client/server** introduce nella versione 5 numerose novità. Nel messaggio (5) il client può richiedere come opzione l'autenticazione reciproca. L'autenticatore include nuovi campi.

- **Sottochiave:** la chiave di crittografia scelta dal client che verrà utilizzata per proteggere questa specifica sessione. Se questo campo viene omesso, verrà utilizzata la chiave di sessione del ticket ($K_{c,v}$).
- **Numero sequenziale:** un campo opzionale che specifica il numero di avvio della sequenza che verrà utilizzato dal server per i messaggi inviati al client durante questa sessione. I messaggi possono essere numerati in sequenza per rilevare le ripetizioni.

Tabella 14.3 Riepilogo dello scambio di messaggi in Kerberos versione 5.

A. Scambio di servizio per l'autenticazione, per ottenere il ticket di assegnamento ticket

(1) $C \rightarrow AS$: $Opzioni \| ID_c \| Realm_c \| ID_{tgs} \| Times \| Nonce_1$

(2) $AS \rightarrow C$: $Realm_c \| ID_c \| Ticket_{tgs} \| E(K_c, [K_{c,tgs} \| Times \| Nonce_1 \| Realm_{tgs} \| ID_{tgs}])$

$Ticket_{tgs} = E(K_{tgs}, [Flags \| K_{c,tgs} \| Realm_c \| ID_C \| AD_C \| Times])$

B. Scambio di assegnamento del ticket di servizio

(3) $C \rightarrow TGS$: $Opzioni \| ID_v \| Times \| Nonce_2 \| Ticket_{tgs} \| Autenticatore_c$

(4) $TGS \rightarrow C$: $Realm_c \| ID_c \| Ticket_v \| E(K_{tgs}, [K_{c,v} \| Times \| Nonce_2 \| Realm_v \| ID_v])$

$Ticket_{tgs} = E(K_{tgs}, [Flags \| K_{c,tgs} \| Realm_c \| ID_C \| AD_C \| Times])$

$Ticket_v = E(K_v, [Flags \| K_{c,v} \| Realm_c \| ID_C \| AD_C \| Times])$

$Autenticatore_c = E(K_{c,tgs}, [ID_C \| Realm_c \| TS_1])$

C. Scambio di autenticazione client/server: per ottenere il servizio

(5) $C \rightarrow V$: $Opzioni \| Ticket_v \| Autenticatore_c$

(6) $V \rightarrow C$: $E_{K_{c,v}}[TS_2 \| Subkey \| Seq\#]$

$Ticket_v = E(K_v, [Flags \| K_{c,v} \| Realm_c \| ID_C \| AD_C \| Times])$

$Autenticatore_c = E(K_{c,v}, [ID_C \| Realm_c \| TS_2 \| Subkey \| Seq\#])$

Se è richiesta l'autenticazione reciproca, il server risponde con il messaggio (6). Questo messaggio include il valore timestamp dell'autenticatore. Si noti che nella versione 4 il valore timestamp era incrementato di un'unità. Questo non è necessario nella versione 5 data la natura del formato dei messaggi, in quanto un estraneo non può creare un messaggio (6) senza conoscere le chiavi di crittografia appropriate. Il campo della sottochiave, se presente, sostituisce l'eventuale campo della sottochiave nel messaggio (5). Il campo opzionale Starting Number indica il numero sequenziale di partenza che verrà utilizzato dal client.

I flag del ticket

Il campo dei flag incluso nei ticket della versione 5 supporta nuove funzionalità rispetto alla versione 4. La Tabella 14.4 elenca i flag che possono essere inclusi nel ticket.

Il flag INITIAL indica che questo ticket è stato emesso dal server AS e non dal server TGS. Quando un client richiede un ticket di servizio dal server TGS, presenta un ticket di assegnamento ticket ottenuto dal server di autenticazione AS. Nella versione 4 questo era l'unico modo per ottenere un ticket di servizio. Nella versione 5 il client può ottenere un ticket di servizio direttamente dal server AS. L'utilità di ciò è la seguente: un server, per esempio un server per il cambio di password, può voler sapere se la password del client è stata controllata recentemente.

Il flag PREAUTHENT, se attivo, indica che quando il server AS ha ricevuto la richiesta iniziale (messaggio 1) ha autenticato il client prima di emettere un ticket. La forma esatta di questa preautenticazione non viene specificata. Come esempio, l'implementazione MIT della versione 5 attiva, nella configurazione predefinita, la preautenticazione del timestamp crittografato. Quando un utente vuole ottenere un ticket, deve inviare al server AS un blocco di preautenticazione contenente un valore casuale, il numero di versione e un codice timestamp, crittografati con la chiave basata sulla password del client. Il server AS esegue la decrittografia del blocco e non restituirà un ticket di assegnamento ticket a meno che il timestamp nel blocco di preautenticazione sia nella finestra temporale consentita (l'intervallo di tempo che considera le derive dal clock e i ritardi della rete). Un'altra possibilità prevede l'uso di una smart card che genera continuamente nuove password che vengono incluse nei messaggi preautenticati. Le password generate dalla card possono basarsi sulla password utente ma devono essere trasformate dalla card in modo che, in pratica, vengano impiegate password arbitrarie. Questo evita un attacco che sfrutta password facili da indovinare. Se viene impiegata una smart card o un dispositivo simile, ciò viene indicato con il flag HW-AUTHENT.

Quando un ticket ha una lunga durata, vi è la possibilità che venga rubato o utilizzato da un estraneo per un periodo di tempo considerevole. Se viene utilizzata una durata molto breve, per ridurre i rischi, occorrerà considerare il sovraccarico dovuto all'acquisizione di nuovi ticket. Nel caso di un ticket di assegnamento ticket, il client dovrà memorizzare la chiave segreta dell'utente, che è chiaramente a rischio, oppure richiedere ripetutamente all'utente la password. Un meccanismo di compromesso consiste nell'impiego di ticket rinnovabili. Un ticket con il flag RENEWABLE include due date di scadenza: una per il ticket vero e proprio e una che rappresenta l'ultimo valore consentito come data di scadenza. Un client può rinnovare un ticket ripresentandolo al server TGS con la richiesta di una

nuova data di scadenza. Se la nuova scadenza si trova entro i limiti dell'ultimo valore consentito, il server TGS può emettere un nuovo ticket con un nuovo tempo di sessione e una nuova scadenza. Il vantaggio di questo meccanismo è il fatto che il server TGS può rifiutarsi di rinnovare un ticket segnalato come rubato.

Un client può richiedere che il server AS fornisca un ticket di assegnamento ticket con il flag MAY-POSTDATE impostato. Il client potrà poi usare questo ticket per richiedere al server TGS un ticket contrassegnato con i flag POSTDATED e INVALID. Successivamente, il client potrà inoltrare il ticket postdatato per la convalida. Questo meccanismo può essere utile per eseguire una lunga operazione batch su un server che richiede periodicamente un ticket. Il client può ottenere contemporaneamente un certo numero di ticket di sessione con valori temporali ben spaziati. Inizialmente tutti i ticket non sono validi, tranne il primo. Quando l'esecuzione raggiunge un punto in cui si richiede un nuovo ticket, il client può ottenere la convalida del ticket appropriato. Con questo approccio, il client non dovrà usare ripetutamente il suo ticket di assegnamento ticket per ottenere un ticket di servizio.

Tabella 14.4 I flag di Kerberos versione 5.

INITIAL	Questo ticket è stato emesso utilizzando il protocollo AS: non è stato emesso utilizzando un ticket di assegnamento ticket.
PRE-AUTHENT	Nell'autenticazione iniziale, il client è stato autenticato dal KDC prima che fosse emesso un ticket.
HW-AUTHENT	Il protocollo utilizzato per l'autenticazione iniziale ha richiesto l'impiego di hardware che si presume disponibile solo al client indicato.
RENEWABLE	Indica al TGS che il ticket può essere utilizzato per ottenere un nuovo ticket con scadenza posteriore.
MAY-POSTDATE	Indica al TGS che può emettere un ticket post-datato sulla base di questo ticket di assegnamento ticket.
POSTDATED	Segnala che questo ticket è stato post-datato; il server finale può controllare il campo *authtime* per controllare quando è avvenuta l'autenticazione originale.
INVALID	Questo ticket non è valido e deve essere validato dal KDC prima di poter essere utilizzato.
PROXIABLE	Indica al TGS che è possibile emettere un nuovo ticket di assegnamento ticket con indirizzo di rete differente, sulla base del ticket presentato.
PROXY	Segnala che questo ticket è un proxy.
FORWARDABLE	Segnala al TGS che è possibile emettere un nuovo ticket di assegnamento ticket con indirizzo di rete differente, sulla base di questo ticket di assegnamento ticket.
FORWARDED	Indica che questo ticket è stato inoltrato oppure è stato emesso sulla base di un'autenticazione che ha comportato l'impiego di un ticket di assegnamento ticket inoltrato.

Nella versione 5 un server può fungere da proxy per un client, adottando in pratica le credenziali e i privilegi del client per richiedere un servizio a un altro server. Se un client desidera utilizzare questo meccanismo, richiede un ticket di assegnamento ticket con il flag PROXIABLE impostato. Quando il server TGS riceve questo ticket, gli viene permesso di emettere un ticket di servizio con un indirizzo di rete differente; quest'ultimo ticket avrà impostato il flag PROXY. Un'applicazione che riceve questo ticket può accettarlo o richiedere ulteriori autenticazioni per garantire l'auditing.[9]

Il concetto di proxy è un caso speciale della procedura più potente di inoltro (forwarding). Se un ticket ha il flag FORWARDABLE impostato, un server TGS può emettere al richiedente un ticket di assegnamento ticket con un indirizzo di rete diverso e il flag FORWARDED impostato. Questo ticket può essere presentato a un server TGS remoto. Questo consente a un client di acquisire l'accesso a un server di un altro realm senza richiedere che ciascun server Kerberos mantenga una chiave segreta con i server Kerberos di altri realm. Per esempio, i realm possono essere strutturati in modo gerarchico. A questo punto un client potrebbe risalire l'albero fino a giungere a un nodo comune e poi ridiscendere fino a raggiungere il realm di destinazione. Ciascun movimento richiederebbe l'inoltro di un ticket di assegnamento ticket al successivo server TGS del percorso.

14.2 Il servizio di autenticazione X.509

La raccomandazione ITU-T X.509 fa parte della serie di raccomandazioni X.500 che definiscono un servizio di directory. La directory è, in pratica, un server (o un insieme distribuito di server) che gestisce un database di informazioni sugli utenti. Le informazioni includono un mapping fra il nome utente e l'indirizzo di rete e altri attributi e informazioni sugli utenti.

X.509 definisce una struttura con lo scopo di fornire servizi di autenticazione tramite la directory X.500. La directory può fungere da archivio di certificati a chiave pubblica del tipo trattato nel Capitolo 9. Ciascun certificato contiene la chiave pubblica di un utente ed è firmato con la chiave privata di un'autorità di certificazione fidata. Inoltre, X.509 definisce dei protocolli di autenticazione alternativi basati sull'uso di certificati a chiave pubblica.

X.509 è uno standard importante in quanto la struttura dei certificati e i protocolli di autenticazione definiti in X.509 vengono utilizzati in vari contesti. Per esempio il formato dei certificati X.509 viene utilizzato in S/MIME (Capitolo 15), IP Security (Capitolo 16) e SSL/TLS e SET (Capitolo 17).

X.509 venne emesso inizialmente nel 1988. Lo standard venne successivamente riveduto per risolvere alcuni dei problemi di sicurezza documentati in [IANS90] e [MITC90]. Fu emessa una raccomandazione riveduta nel 1993 e una terza versione nel 1995 a sua volta riveduta nel 2000.

X.509 si basa sull'uso della crittografia a chiave pubblica e delle firme digitali. Lo standard non prevede l'uso di un algoritmo determinato ma consiglia l'impiego di RSA. Il meccanismo di firma digitale richiede l'impiego di una funzione hash. Anche in questo caso lo standard

[9] Per informazioni sui possibili usi della funzionalità proxy, consultare [NEUM93b].

non prevede l'impiego di un determinato algoritmo hash. La raccomandazione del 1988 includeva la descrizione di un algoritmo hash consigliato; questo algoritmo si è però dimostrato non sicuro ed è stato eliminato dalla raccomandazione del 1993. La Figura 14.3 illustra la generazione di un certificato a chiave pubblica.

I certificati

Il cuore del meccanismo X.509 è il certificato a chiave pubblica di ciascun utente. Si presuppone che questi certificati utente siano creati da qualche autorità di certificazione e inseriti nella directory da tale autorità o dall'utente. Il server contenente la directory non è quindi responsabile della creazione delle chiavi pubbliche o della funzione di certificazione; fornisce semplicemente una locazione accessibile per ottenere i certificati.

La Figura 14.4A mostra il formato generale di un certificato comprendente i seguenti elementi.

- **Versione:** distingue fra le varie versioni del formato dei certificati: la versione predefinita è la 1. Se sono presenti i campi Issuer Unique Identifier o Subject Unique Identifier, il valore deve essere 2. Se sono presenti una o più estensioni, la versione deve essere 3.

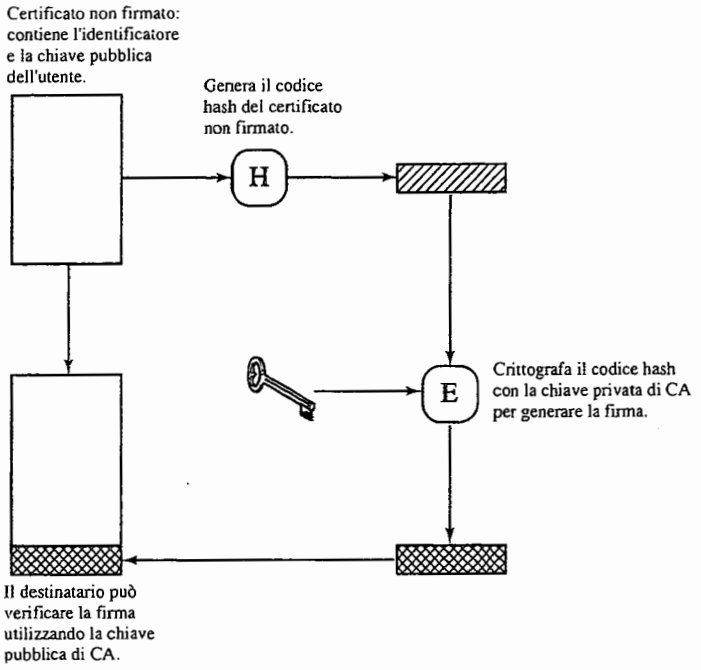


Figura 14.3 Impiego di un certificato a chiave pubblica.

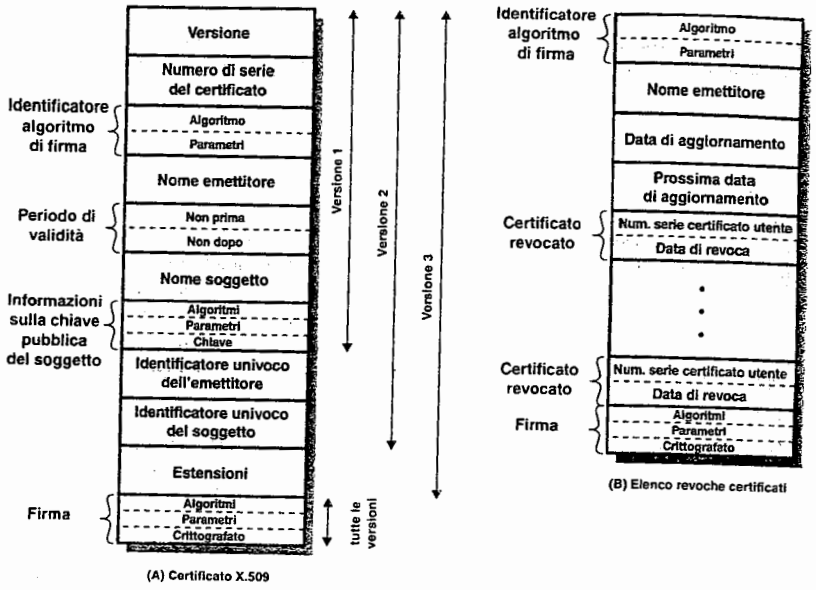


Figura 14.4 I formati specificati do X.509.

- **Numero di serie:** un valore intero univoco nell'ambito dell'autorità di certificazione che viene assegnato in modo non ambiguo a questo certificato.
- **Identificatore algoritmo di firma:** l'algoritmo utilizzato per firmare il certificato, con tutti i relativi parametri. Poiché questa informazione viene ripetuta nel campo della firma alla fine del certificato, questo campo ha in realtà una scarsissima utilità.
- **Nome emettitore:** il nome X.500 dell'autorità di certificazione che ha creato e firmato questo certificato.
- **Periodo di validità:** è costituito da due date: la prima e l'ultima data di validità del certificato.
- **Nome soggetto:** il nome dell'utente cui fa riferimento il certificato. In pratica questo certificato certifica la chiave pubblica del soggetto che ha la corrispondente chiave privata.
- **Informazioni sulla chiave pubblica del soggetto:** la chiave pubblica del soggetto, più un identificatore dell'algoritmo per il quale verrà utilizzata la chiave, insieme a tutti i parametri ad essa associati.
- **Identificatore univoco dell'emettitore:** una stringa di bit opzionale utilizzata per identificare univocamente l'autorità di certificazione emettitrice nel caso in cui il nome X.500 venga riutilizzato da più entità.
- **Identificatore univoco del soggetto:** una stringa di bit opzionale utilizzata per identificare univocamente il soggetto nel caso in cui il nome X.500 sia stato riutilizzato per varie entità.

- **Estensioni:** un insieme di uno o più campi di estensione. Le estensioni sono state aggiunte nella versione 3 e verranno trattate più avanti in questo stesso capitolo.
- **Firma:** copre tutti gli altri campi del certificato; contiene il codice hash degli altri campi crittografato con la chiave privata dell'autorità di certificazione. Questo campo include l'identificatore dell'algoritmo di firma.

I campi univoci di identificazione sono stati aggiunti nella versione 2 per gestire la possibilità di riutilizzo dei nomi del soggetto e/o dell'emettitore. Questi campi vengono però utilizzati raramente.

Lo standard utilizza la seguente notazione per definire un certificato:

$$CA\langle\langle A\rangle\rangle = CA\{V, SN, AI, CA, T_A, A, Ap\}$$

dove:

$Y\langle\langle X\rangle\rangle$ = il certificato dell'utente X emesso dall'autorità di certificazione Y.

$Y\{I\}$ = la firma di I emessa da Y. È costituita da I con l'aggiunta di un codice hash crittografato.

L'autorità di certificazione firma il certificato con la propria chiave privata. Se l'utente conosce la chiave pubblica corrispondente, potrà verificare che il certificato firmato dall'autorità di certificazione è valido. Questo è il tipico approccio a firma digitale illustrato nella Figura 11.5C.

Come ottenere un certificato utente

I certificati utente generati da un'autorità di certificazione hanno le seguenti caratteristiche.

- Ogni utente con accesso alla chiave pubblica dell'autorità di certificazione può verificare la chiave pubblica dell'utente che è stata certificata.
- Nessuno, ad eccezione dell'autorità di certificazione, può modificare il certificato senza essere rilevato.

Poiché i certificati non possono essere falsificati, possono essere inseriti in una directory senza alcuna necessità di proteggerli.

Se tutti gli utenti si abbonano alla stessa autorità di certificazione, vi sarà una fiducia comune su tale autorità. Tutti i certificati utente possono essere inseriti nella directory e resi accessibili a tutti gli utenti. Inoltre, un utente può trasmettere il proprio certificato direttamente ad altri utenti. In ogni caso, una volta che B è in possesso del certificato di A, sarà sicuro che i messaggi crittografati con la chiave pubblica di A saranno al sicuro da qualsiasi intercettazione e che i messaggi firmati con la chiave privata di A non possano essere falsificati.

Se la comunità di utenti è estesa, è sconsigliabile che tutti gli utenti si abbonino alla stessa autorità di certificazione. Poiché è l'autorità di certificazione che firma i certificati, per poter verificare le firme ogni utente deve avere una copia della chiave pubblica dell'autorità di certificazione. Questa chiave pubblica deve essere fornita a ciascun utente in modo assolutamente sicuro (per quanto riguarda l'integrità e l'autenticità) in modo che l'utente possa fidarsi dei relativi certificati. Pertanto, se gli utenti sono numerosi, è più pratico impiegare più autorità di certificazione, ognuna delle quali fornisce in modo sicuro la propria chiave pubblica a una determinata frazione degli utenti.

Ora si supponga che A abbia ottenuto un certificato dall'autorità di certificazione X_1 e che B abbia ottenuto un certificato dall'autorità di certificazione X_2. Se A non conosce in modo sicuro la chiave pubblica di X_2, il certificato di B, emesso da X_2, risulterà inutile per A. A può leggere il certificato di B ma non può verificarne la firma. Tuttavia, se le due autorità di certificazione si sono scambiate la propria chiave pubblica in modo sicuro, la seguente procedura consentirà ad A di ottenere la chiave pubblica di B.

1. A ottiene, dalla directory, il certificato di X_2 firmato da X_1. Poiché A conosce in modo sicuro la chiave pubblica di X_1, può ottenere la chiave pubblica di X_2 dal suo certificato e verificarla per mezzo della firma di X_1 sul certificato.
2. Quindi A ritorna alla directory e ottiene il certificato di B firmato da X_2. Poiché ora A ha una copia sicura della chiave pubblica di X_2, potrà verificare la firma e ottenere con sicurezza la chiave pubblica di B.

A ha utilizzato una catena di certificati per ottenere la chiave pubblica di B. Nella notazione X.509, questa catena è espressa come:

$$X_1\langle\langle X_2\rangle\rangle X_2\langle\langle B\rangle\rangle$$

Analogamente, B può ottenere la chiave pubblica di A con la catena inversa:

$$X_2\langle\langle X_1\rangle\rangle X_i\langle\langle A\rangle\rangle$$

Questo schema non si limita a una catena di due certificati. Si può produrre una catena di un numero arbitrariamente lungo di autorità di certificazione. Una catena di *N* elementi verrà espressa come:

$$X_1\langle\langle X_2\rangle\rangle X_2\langle\langle X_3\rangle\rangle \ldots X_N\langle\langle B\rangle\rangle$$

In questo caso, ciascuna coppia di autorità di certificazione della catena (X_i, X_{i+1}) deve aver creato vicendevolmente i certificati.

Tutti i certificati delle autorità di certificazione per altre autorità di certificazione devono comparire nella directory e l'utente deve conoscere il modo in cui esse sono collegate fra loro per seguire un percorso verso il certificato a chiave pubblica di un altro utente. X.509 suggerisce che le autorità di certificazione vengano disposte in modo gerarchico in modo da semplificarne la navigazione.

La Figura 14.5, tratta da X.509, rappresenta un esempio di questa gerarchia. I cerchi connessi indicano le relazioni gerarchiche fra le autorità di certificazione; i riquadri indicano i certificati mantenuti nella directory per ciascuna voce dell'autorità di certificazione. Ogni voce di directory conservata dall'autorità di certificazione include due tipi di certificati.

- **Certificati in avanti:** certificati di X generati da altre autorità di certificazione.
- **Certificati inversi:** certificati generati da X per altre autorità di certificazione.

In questo esempio l'utente A può acquisire i seguenti certificati dalla directory per stabilire un percorso di certificazione con B:

$$X\langle\langle W\rangle\rangle W\langle\langle V\rangle\rangle V\langle\langle Y\rangle\rangle Y\langle\langle Z\rangle\rangle Z\langle\langle B\rangle\rangle$$

Quando A ha ottenuto questi certificati, può seguire il percorso di certificazione in sequenza per ottenere una copia fidata della chiave pubblica di B. Utilizzando questa chiave pubblica, A può inviare a B i messaggi crittografati. Se A vuole ricevere i messaggi crittografati

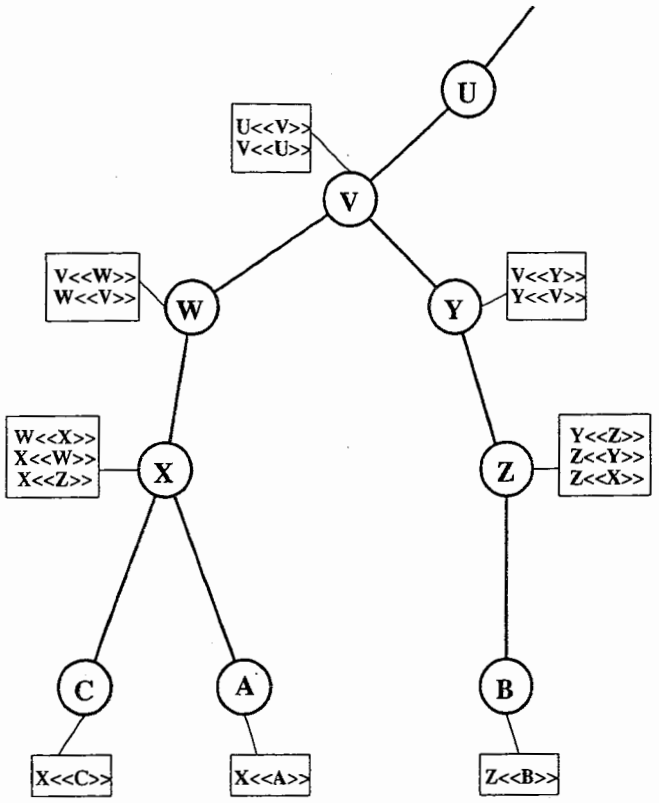


Figura 14.5 Un esempio ipotetica di una gerarchia X.509.

da B o firmare i messaggi inviati a B, allora B deve conoscere la chiave pubblica di A, che può essere ottenuta con il seguente percorso di certificazione:

$$Z\langle\langle Y\rangle\rangle Y\langle\langle V\rangle\rangle V\langle\langle W\rangle\rangle W\langle\langle X\rangle\rangle X\langle\langle A\rangle\rangle$$

B può ottenere questo insieme di certificati dalla directory oppure A può fornirlo nell'ambito del suo messaggio iniziale a B.

Revoca dei certificati

Come si può vedere nella Figura 14.4, ogni certificato include un periodo di validità, un po' come una carta di credito. In genere viene emesso un nuovo certificato appena prima della scadenza del precedente. Talvolta è preferibile revocare un certificato prima della scadenza per uno dei seguenti motivi.

1. La chiave privata dell'utente è stata violata.
2. L'utente non è più certificato da questa autorità di certificazione.
3. Il certificato dell'autorità di certificazione è stato violato.

Ogni autorità di certificazione deve gestire un elenco di tutti i certificati revocati ma non scaduti da lei emessi, inclusi sia quelli emessi per gli utenti che quelli emessi per altre autorità di certificazione. Questi elenchi devono essere disponibili anche nella directory.

Ciascuna lista di revoca dei certificati inviata alla directory è firmata dall'emettitore e include (Figura 14.4B) il nome dell'emettitore, la data in cui è stata creata la lista, la data di programmazione della prossima lista di revoca dei certificati e una voce per ciascun certificato revocato. Ogni voce è costituita dal numero di serie e dalla data di revoca del certificato. Poiché il numero di serie è univoco nell'ambito di un'autorità di certificazione, è sufficiente per identificare il certificato.

Quando un utente riceve un certificato in un messaggio, deve determinare se il certificato è stato revocato. L'utente potrebbe controllare la directory ogni volta che viene ricevuto un certificato. Per evitare i ritardi (e i costi) associati alle ricerche nella directory, è probabile che l'utente voglia gestire un archivio locale dei certificati e delle liste dei certificati revocati.

Procedure di autenticazione

X.509 include tre procedure di autenticazione alternative destinate a vari tipi di applicazioni. Tutte queste procedure utilizzano firme a chiave pubblica. Si presuppone che le due parti conoscano la reciproca chiave pubblica o avendo ottenenuto i certificati dell'altra parte tramite la directory o perché il certificato è stato incluso nel messaggio iniziale inviato dall'altro capo.

La Figura 14.6 illustra le tre procedure.

Autenticazione monodirezionale

L'autenticazione monodirezionale prevede un unico trasferimento delle informazioni dall'utente A all'utente B e stabilisce.

1. L'identità di A e il fatto che il messaggio sia stato generato da A.
2. Il fatto che il messaggio era destinato a B.
3. L'integrità e originalità del messaggio (ovvero non sono state inviate più copie).

Si noti che in questa operazione viene verificata solo l'identità dell'entità che inizia lo scambio e non quella dell'entità che risponde.

Come minimo il messaggio include un valore timestamp t_A, un valore nonce r_A e l'identità di B e viene firmato con la chiave privata di A. Il codice timestamp è costituito da un tempo di generazione opzionale e da un tempo di scadenza. Questo impedisce la consegna ritardata dei messaggi. Il valore nonce può essere utilizzato per rilevare gli attacchi a replay. Esso deve essere univoco nell'ambito del tempo di scadenza del messaggio. Pertanto, B può memorizzare il valore nonce fino alla scadenza e rifiutare ogni nuovo messaggio con lo stesso valore nonce.

Per svolgere la sola autenticazione, il messaggio viene utilizzato semplicemente per presentare le credenziali a B. Il messaggio può anche includere le informazioni da trasmettere.

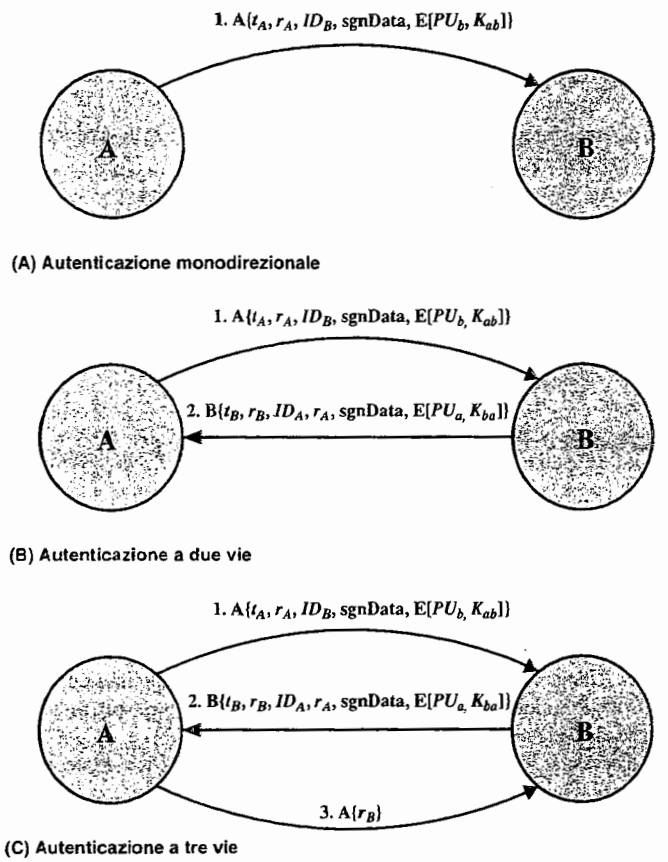


Figura 14.6 Procedure di autenticazione forte in X.509.

Queste informazioni, sgnData, vengono incluse nell'ambito della firma, garantendo la loro autenticità e integrità. Il messaggio può anche essere utilizzato per inviare a B una chiave di sessione, crittografata con la chiave pubblica di B.

Autenticazione a due vie

Oltre ai tre elementi elencati in precedenza, l'autenticazione a due vie stabilisce i seguenti elementi.

4. L'identità di B e il fatto che il messaggio di risposta sia generato da B.
5. Il fatto che il messaggio era destinato a A.
6. L'integrità e l'originalità della risposta.

L'autenticazione a due vie consente pertanto a entrambe le parti di verificare l'identità dell'altra parte.

Il messaggio di risposta include il codice nonce ricevuto da A, per convalidare la risposta. Inoltre include il valore timestamp e un valore nonce generato da B. Come prima il messaggio può includere ulteriori informazioni firmate e una chiave di sessione crittografata con la chiave pubblica di A.

Autenticazione a tre vie

Nell'autenticazione a tre vie viene incluso un messaggio finale da A a B che contiene una copia firmata del valore nonce r_B. Lo scopo di questo meccanismo è evitare di controllare i timestamp: poiché entrambi i nonce vengono rinviati all'altro lato, ciascun lato può controllare il nonce restituito per individuare gli attacchi a replay. Questo approccio è necessario quando i clock non sono sincronizzati.

X.509 versione 3

Il formato della versione 2 di X.509 non fornisce tutte le informazioni dimostratesi necessarie dall'esperienza di progettazione e di implementazione più recente. [FORD95] elenca i seguenti requisiti non soddisfatti dalla versione 2.

1. Il campo "Nome soggetto" è inadeguato per fornire l'identità del possessore di una chiave a un utente a chiave pubblica. I nomi X.509 possono essere relativamente brevi e possono mancare certi dettagli di identificazione richiesti dall'utente.
2. Il campo "Nome soggetto" è anche inadeguato per molte applicazioni che in genere riconoscono le entità tramite l'indirizzo di posta elettronica Internet, un indirizzo URL o qualche altra identificazione Internet.
3. Vi è la necessità di indicare informazioni relative alla politica di sicurezza. Questo consente all'applicazione o funzione di sicurezza come IPSec, di correlare un certificato X.509 a una determinata politica.
4. Vi è la necessità di limitare il danno che può essere provocato da un guasto o da un'autorità di certificazione disonesta imponendo dei vincoli sull'applicabilità di un determinato certificato.
5. È importante essere in grado di identificare separatamente le varie chiavi utilizzate dallo stesso possessore in momenti differenti. Questa funzionalità supporta la gestione del ciclo di vita della chiave, in particolare la possibilità di aggiornare regolarmente o in alcune circostanze eccezionali le coppie di chiavi per gli utenti e le autorità di certificazione.

Invece di continuare ad aggiungere campi a un formato fisso, gli sviluppatori dello standard hanno ritenuto di dover adottare un approccio più flessibile. Pertanto la versione 3 include varie estensioni opzionali che possono essere aggiunte al formato della versione 2. Ogni estensione è costituita da un identificatore, un indicatore di importanza e un valore. L'indicatore di importanza indica se una estensione può essere ignorata o meno. Se l'indicatore ha il valore TRUE e l'implementazione non riconosce l'estensione, deve trattare il certificato come non valido.

Le estensioni del certificato rientrano in tre categorie: informazioni sulla chiave e la politica, attributi del soggetto e dell'emettitore e vincoli nel percorso di certificazione.

Informazioni sulla chiave e la politica

Queste estensioni forniscono informazioni aggiuntive sulle chiavi del soggetto e dell'emettitore più gli indicatori della politica del certificato. Una politica del certificato è costituita da un insieme di regole che indicano l'applicabilità di un certificato a una determinata comunità e/o classe di applicazione con requisiti di sicurezza comuni. Per esempio, una politica può indicare che il certificato è applicabile all'autenticazione di transazioni EDI (Electronic Data Interchange) per il commercio di beni all'interno di un determinato intervallo di prezzi.

L'area include i seguenti elementi.

- **Identificatore della chiave dell'autorità:** identifica la chiave pubblica che deve essere utilizzata per verificare la firma su questo certificato o CRL. Consente di distinguere le chiavi emesse dalla stessa autorità di certificazione. Un uso di questo campo è la gestione degli aggiornamenti delle coppie di chiavi dell'autorità di certificazione.
- **Identificatore della chiave del soggetto:** identifica la chiave pubblica certificata. Utile per l'aggiornamento delle coppie di chiavi del soggetto. Inoltre un soggetto può avere più coppie di chiavi e, di conseguenza, certificati differenti per scopi differenti (per esempio accordo sulla firma digitale e la chiave di crittografia).
- **Uso della chiave:** indica una restrizione riguardante lo scopo e le politiche d'uso della chiave pubblica certificata. Può indicare una o più delle seguenti funzionalità: firma digitale, non ripudiabilità, crittografia della chiave, crittografia dei dati, accordo sulla chiave, verifica della firma dell'autorità di certificazione sui certificati, verifica della firma dell'autorità di certificazione sui CRL.
- **Periodo d'uso della chiave privata:** indica il periodo d'uso della chiave privata corrispondente alla chiave pubblica. In genere la chiave privata viene utilizzata per un periodo di validità differente rispetto alla chiave pubblica. Per esempio, per le chiavi per la firma digitale, il periodo d'uso della chiave privata è normalmente più breve di quello della chiave pubblica di verifica.
- **Politiche dei certificati:** i certificati possono essere utilizzati in ambienti in cui si applicano più politiche. Questa estensione elenca le politiche supportate dal certificato insieme a informazioni opzionali di qualifica.
- **Mapping delle politiche:** utilizzati solo nei certificati per le autorità di certificazione emessi da altre autorità di certificazione. I mapping delle politiche consentono a un'autorità di certificazione emettitrice di indicare che una o più delle sue politiche possono essere considerate equivalenti ad altre politiche utilizzate nel dominio dell'autorità di certificazione del soggetto.

Attributi del soggetto e dell'emettitore del certificato

Queste estensioni supportano nomi alternativi, in formati alternativi, per il soggetto o l'emettitore di un certificato e possono fornire informazioni aggiuntive sul soggetto del certificato, per aumentare la sicurezza che il soggetto del certificato sia una determinata persona o entità. Per esempio informazioni quali l'indirizzo postale, la posizione aziendale o un'immagine.

Fra i campi di estensione di quest'area vi sono i seguenti.

- **Nome alternativo del soggetto:** contiene uno o più nomi alternativi, in vari formati. Questo campo è importante per supportare determinate applicazioni quali la posta elettronica, EDI e IPSec che possono impiegare specifici formati di denominazione.
- **Nome alternativo dell'emettitore:** contiene uno o più nomi alternativi utilizzando vari formati.
- **Attributi della directory del soggetto:** fornisce ogni attributo desiderato nella directory X.500 per il soggetto di questo certificato.

Vincoli del percorso di certificazione

Queste estensioni consentono di specificare dei vincoli da includere nei certificati di autorità di certificazione emessi da altre autorità di certificazione. I vincoli possono limitare i tipi di certificati che possono essere emessi dall'autorità di certificazione o emessi successivamente in una catena di certificazione.

I campi di estensione in quest'area comprendono i seguenti.

- **Vincoli di base:** indica se il soggetto può fungere da autorità di certificazione. In caso affermativo può essere specificato un vincolo riguardante la lunghezza del percorso di certificazione.
- **Vincoli sui nomi:** indica uno spazio dei nomi all'interno del quale devono trovarsi tutti i nomi dei certificati successivi in un percorso di certificazione.
- **Vincoli di politica:** specifica i vincoli che possono richiedere l'identificazione esplicita della politica del certificato o che inibiscono il mapping della politica per la parte rimanente del percorso di certificazione.

14.3 L'infrastruttura per la chiave pubblica

Il documento RFC 2822 (*Internet Security Glossary*) definisce l'infrastruttura per la chiave pubblica (PKI) come l'insieme dell'hardware, software, utenti, politiche e procedure necessari per creare, gestire, memorizzare, distribuire e revocare i certificati digitali relativi alla crittografia asimmetrica. L'obiettivo principale per sviluppare una PKI è consentire l'acquisizione sicura, conveniente ed efficiente delle chiavi pubbliche. Il gruppo di lavoro *Public Key Infrastructure X.509* (PKIX) della IETF ha consentito di definire un modello formale (e generico) basato su X.509, idoneo alla messa in opera su Internet di un'architettura basata su certificati. Questo paragrafo descrive il modello PKIX.

La Figura 14.7 illustra la relazione fra gli elementi chiave del modello PKIX.

Questi elementi sono:

- **Entità finale:** termine generico utilizzato per indicare gli utenti finali, i dispositivi (server, router ecc.) o qualsiasi altra entità che possa essere indicata nel campo soggetto di un certificato a chiave pubblica. Le entità finali solitamente consumano e/o supportano i servizi PKI.

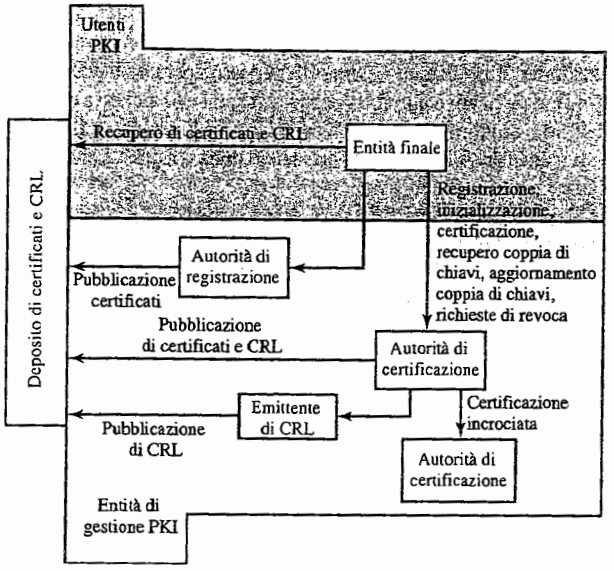


Figura 14.7 Modello architetturale PKIX.

- **Autorità di certificazione (CA):** l'emittente dei certificati e (solitamente) degli elenchi di revoca dei certificati (CRL). Può anche supportare altre funzioni amministrative, sebbene queste vengano spesso delegate a una o più autorità di registrazione.
- **Autorità di registrazione (RA):** componente opzionale che si può occupare di varie funzioni amministrative per conto della CA. L'RA si occupa spesso del processo di registrazione delle entità finali, ma può essere di supporto anche in varie altre aree.
- **Emittente delle CRL:** componente opzionale alla quale una CA può delegare il compito di pubblicare le CRL.
- **Deposito (repository):** termine generico utilizzato per indicare qualsiasi metodo per memorizzare certificati e CRL in modo che possano essere ottenute dalle entità finali.

Le funzioni di gestione PKIX

PKIX identifica le funzioni di gestione che richiedono potenzialmente di essere supportate da protocolli di gestione. Esse sono indicate nella Figura 14.7 e comprendono:

- **Registrazione:** il processo che consente a un utente di presentarsi a una CA (direttamente o tramite una RA), affinché questa possa emettere un certificato per tale utente. La registrazione consente di iniziare il processo di ammissione in una PKI. La registrazione comporta solitamente alcune procedure in linea o fuori linea per la mutua

autenticazione. Tipicamente si consegnano all'entità finale una o più chiavi segrete condivise, utilizzate per la successiva autenticazione.

- **Inizializzazione:** prima che un sistema client possa operare in sicurezza, è necessario installare materiale relativo alle chiavi opportunamente associato a chiavi memorizzate altrove nell'infrastruttura. Per esempio, il client deve poter essere inizializzato in sicurezza con la chiave pubblica e altre informazioni della CA fidata, che verranno utilizzate per validare percorsi di certificati.
- **Certificazione:** processo con il quale la CA emette un certificato per la chiave pubblica di un utente, lo comunica al sistema client dell'utente e/o lo memorizza nel deposito.
- **Recupero della coppia di chiavi:** le coppie di chiavi possono essere utilizzate per supportare la creazione e la verifica di firme digitali, la crittografia/decrittografia, o entrambe. Quando una coppia di chiavi viene utilizzata per la crittografia/decrittografia, è importante prevedere un meccanismo per il recupero della chiave di decrittografia anche quando non sia più possibile l'accesso normale alle chiavi, pena l'impossibilità di recuperare le informazioni crittografate. La perdita della chiave di decrittografia potrebbe essere il risultato di una password o un PIN dimenticati, unità disco rovinate, unità hardware danneggiate e simili. Il recupero delle coppie di chiavi consente alle entità finali di recuperare la propria coppia di chiavi di crittografia/decrittografia da un sistema autorizzato di backup delle chiavi (tipicamente la CA che ha emesso il certificato dell'entità finale).
- **Aggiornamento della coppia di chiavi:** tutte le coppie di chiavi devono essere regolarmente aggiornate (ovvero sostituite da una nuova coppia di chiavi) e si devono emettere i nuovi certificati corrispondenti. L'aggiornamento è richiesto alla scadenza e alla revoca del certificato.
- **Richiesta di revoca:** un utente autorizzato comunica a una CA una situazione anomala che richiede la revoca di un certificato. Possibili motivi di revoca sono la perdita di segretezza della chiave privata, cambiamenti di affiliazione e sostituzioni di nome.
- **Certificazione incrociata:** due CA si scambiano informazioni utilizzate per creare un certificato incrociato. Un *certificato incrociato* è emesso da una CA per una seconda CA, che contiene una chiave di firma utilizzata per emettere certificati.

Protocolli di gestione PKIX

Il gruppo di lavoro PKIX ha definito due protocolli di gestione alternativi fra entità PKIX che supportano le funzioni di gestione elencate nel paragrafo precedente. La RFC 2510 definisce i protocolli di gestione dei certificati (CMP). Nel contesto CMP, le funzioni di gestione vengono esplicitamente identificate da una serie di interazioni. CMP è progettato per essere un protocollo flessibile capace di adattarsi a vari modelli tecnici, operazionali e commerciali.

La RFC 2797 definisce i messaggi di gestione dei certificati (CMC) sopra CMS, dove CMS si riferisce alla RFC 2630 relativa alla sintassi dei messaggi di crittografia. CMC è l'evoluzione di lavori precedenti e cerca di valorizzare le implementazioni esistenti. Sebbene supporti tutte le funzioni PKIX, non tutte corrispondono a specifiche interazioni del protocollo.

14.4 Letture e siti Web consigliati

Un modo semplice per comprendere i concetti che stanno alla base di Kerberos è [BRYA88]. Uno dei migliori trattati su Kerberos è [KOHL94]. [TUNG99] descrive Kerberos dal punto di vista dell'utente. [PERL99] propone una rassegna di vari modelli di trust che possono essere utilizzati in PKI. [GUTM02] evidenzia le difficoltà insite nell'impiego di PKI e propone raccomandazioni per un uso efficace.

BRYA88 W. Bryant. *Designing an Authentication System: A Dialogue in Four Scenes.* Project Athena document, Febbraio 1988. Disponibile presso il sito `http://web.mit.edu/kerberos/www/dialogue.html`.

GUTM02 P. Gutmann. "PKI: It's Not Dead, Just Resting". *Computer*, Agosto 2002.

KOHL94 J. Kohl, B. Neuman e T. Ts'o. "The Evolution of the Kerberos Authentication Service". in F. Brazier and Johansen, D. Distributed Open Systems. Los Alamitos, CA: IEEE Computer Society Press, 1994. Disponibile presso il sito `http://web.mit.edu/kerberos/www/papers.html`.

PERL99 R. Perlman. "An Overview of PKI Trust Models". *IEEE Network*, Novembre/Dicembre 1999.

TUNG99 B. Tung. *Kerberos: A Network Authentication System.* Reading, MA: Addison-Wesley, 1999.

Siti Web consigliati

- **MIT Kerberos Site:** informazioni su Kerberos fra cui documenti FAQ, altri documenti, articoli e puntatori a siti di prodotti commerciali.
- **USC/ISI Kerberos Page:** un'altra buona fonte di materiali su Kerberos.
- **Public-Key Infrastructure Working Group:** il gruppo IETF che sviluppa standard basati su X.509 versione 3.
- **Verisign:** uno dei principali produttori commerciali di prodotti basati su X.509; contiene documenti e altro materiale interessante.
- **Kerberos Working Group:** il gruppo IETF che sviluppa gli standard relativi a Kerberos.
- **NIST PKI Program:** ottima fonte di informazioni.

14.5 Termini chiave, domande di ripasso e problemi

Termini chiave

Autenticazione
Certificato a chiave pubblica

Certificato X.509
Durata
Kerberos
Modalità PCBC (Propagating Cipher Block Chaining)
Nonce
Numero sequenziale
Realm Kerberos
Server di autenticazione
Server TGS (Ticket-Granting Server)
Sottochiave
Ticket

Domande di ripasso

14.1 Quali problemi deve risolvere Kerberos?
14.2 Indicare tre rischi associati all'autenticazione degli utenti tramite una rete o Internet.
14.3 Elencare tre approcci per rendere sicura l'autenticazione degli utenti in un ambiente distribuito.
14.4 Quali sono i quattro requisiti definiti per Kerberos?
14.5 Quali entità costituiscono un ambiente Kerberos completo?
14.6 Nel contesto di Kerberos, che cos'è un realm?
14.7 Quali sono le principali differenze fra le versioni 4 e 5 di Kerberos?
14.8 Qual è lo scopo dello standard X.509?
14.9 Che cos'è una catena di certificati?
14.10 Come viene revocato un certificato X.509?

Problemi

14.1 Mostrare che in modalità PCBC un errore casuale in un blocco di testo cifrato si propaga a tutti i blocchi successivi di testo in chiaro (Figura 14.9).
14.2 Si supponga che, in modalità PCBC, i blocchi C_i e $C_{(i+1)}$ vengano scambiati durante la trasmissione. Mostrare che ciò coinvolge solo i blocchi decrittografati P_i e P_{i+1} ma non i blocchi successivi.
14.3 La procedura di autenticazione originale a tre vie per X.509 illustrata nella Figura 14.6 C ha una lacuna di sicurezza. Fondamentalmente il protocollo funziona nel seguente modo:

$$\begin{aligned} A \rightarrow B: &\quad A\{t_A, r_A, ID_B\} \\ B \rightarrow A: &\quad B\{t_B, r_B, ID_A, r_A\} \\ A \rightarrow B: &\quad A\{r_B\} \end{aligned}$$

Il testo di X.509 stabilisce che il controllo dei timestamp t_A e t_B è opzionale per l'autenticazione a tre vie ma si consideri il seguente esempio. Si supponga che A e B abbiano usato il protocollo in una precedente occasione e che un estraneo C abbia intercettato i tre messaggi precedenti. Si supponga inoltre che i timestamp non vengano utilizzati e siano tutti uguali a zero. Infine si supponga che C voglia impersonare A interagendo con B. Inizialmente C invia a B il primo messaggio catturato:

$$C \rightarrow B: \quad A\{0, r_A, ID_B\}$$

B risponde credendo di comunicare con A ma in realtà sta comunicando con C:

$$B \rightarrow C: \quad B\{0, r'_B, ID_A, r_A\}$$

Nel frattempo C fa in modo che A inizi l'autenticazione con C. Come risultato, A invia a C il seguente messaggio:

$$A \rightarrow C: \quad A\{0, r'_A, ID_C\}$$

C risponde ad A utilizzando lo stesso codice nonce fornito a C da B:

$$C \rightarrow A: \quad C\{0, r'_B, ID_A, r'_A\}$$

A risponde con:

$$A \rightarrow C: \quad A\{r'_B\}$$

Questo è esattamente ciò di cui C ha bisogno per convincere B che sta comunicando con A e dunque C ripete il messaggio in arrivo inviandolo a B:

$$C \rightarrow B: \quad A\{r'_B\}$$

In questo modo B è convinto di comunicare con A mentre in realtà sta comunicando con C. Suggerire una semplice soluzione a questo problema che non preveda l'uso dei codici timestamp.
14.4 Nella versione X.509 del 1988 sono elencate le proprietà che le chiavi RSA devono soddisfare per essere sicure in base alle conoscenze dell'epoca relative alla difficoltà di fattorializzare grossi numeri interi. La discussione si conclude con un vincolo sull'esponente pubblico e il modulo n:
Si deve assicurare che $e > \log_2(n)$ per impedire attacchi che possano calcolare la radice e-esima modulo n per decrittografare il testo in chiaro.
Sebbene il vincolo sia corretto, il motivo della richiesta è errato. Che cosa vi è di sbagliato nel motivo indicato e qual è il vero motivo?

Appendice 14.A Le tecniche di crittografia Kerberos

Kerberos include una libreria di crittografia che supporta varie operazioni di crittografia.

Queste erano state incluse nelle specifiche di Kerberos 5 e sono diffuse nelle implementazioni commerciali. Nel febbraio del 2005, la IETF ha rilasciato le RFC 3961 e 3962 che estendono le opzioni delle tecniche crittografiche. In questa appendice si presentano le tecniche contenute nella RFC 1510.

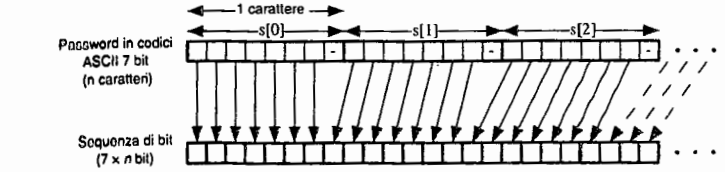
Trasformazione della password in chiave

Le password in Kerberos possono impiegare solamente caratteri rappresentabili in formato ASCII a 7 bit. Una password, di lunghezza arbitraria, viene convertita in una chiave di crittografia conservata nel database Kerberos. La Figura 14.8 illustra la procedura.

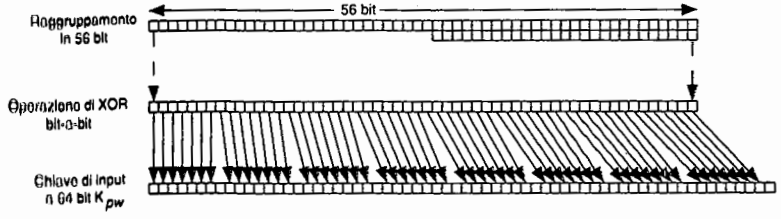
Innanzitutto la stringa di caratteri s viene trasformata in una stringa di bit, b, in modo che il primo carattere sia contenuto nei primi sette bit, il secondo carattere nei secondi 7 bit e così via. Il tutto può essere espresso nel seguente modo:

b[0] = bit 0 di s[0]

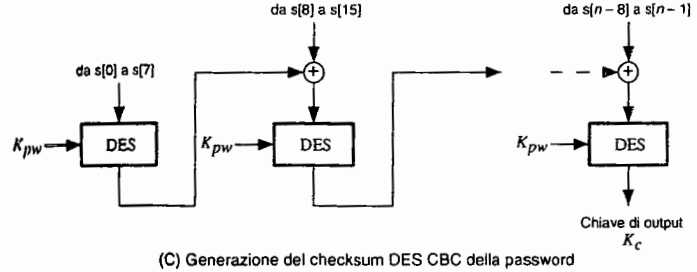
.
.
.



(A) Conversione della password in un flusso di bit



(B) Conversione del flusso di bit nella chiave di input



(C) Generazione del checksum DES CBC della password

Figura 14.8 Generazione della chiave di crittografia a partire dalla password.

b[6] = bit 6 di s[0]
b[7] = bit 0 di s[1]

.
.
.

$$b[7i + m] = \text{bit } m \text{ di } s[i] \quad 0 \le m \le 6$$

La stringa di bit viene poi compattata in 56 bit allineando i bit in righe di 56 bit a "ventaglio" e svolgendo un'operazione di XOR tra i bit delle righe. Per esempio, se la stringa di bit è di lunghezza 59, allora:

b[55] = b[55] ⊕ b[56]
b[54] = b[54] ⊕ b[57]
b[53] = b[53] ⊕ b[58]

Questo crea una chiave DES a 56 bit. Per adattarsi al formato a 64 bit della chiave, la stringa viene trattata come una sequenza di otto blocchi di 7 bit e mappata in otto blocchi di 8 bit per formare una chiave di input K_{pw}.

Infine la password originaria viene crittografata utilizzando la modalità CBC (Cipher Block Chaining) di DES con la chiave K_{pw}. L'ultimo blocco di 64 bit restituito da questo processo, chiamato checksum CBC, è la chiave di output associata alla password.

L'intero algoritmo può esser considerato come una funzione hash che mappa una password arbitraria in un codice hash di 64 bit.

Modalità PCBC

Nel Capitolo 6 si è detto che, nella modalità CBC di DES, l'input dell'algoritmo DES in ciascuna fase è costituito dallo XOR del blocco di testo in chiaro corrente e del blocco di testo cifrato precedente utilizzando la stessa chiave per ciascun blocco (Figura 6.4). Il vantaggio di questa modalità rispetto alla modalità ECB (Electronic CodeBook) in cui ciascun blocco di testo in chiaro viene crittografato in modo indipendente è il seguente: con CBC lo stesso blocco di testo in chiaro, se ripetuto, produce blocchi di testo cifrato differenti.

In CBC, un eventuale errore di trasmissione del blocco di testo cifrato C_i si propaga ai blocchi di testo in chiaro P_i e P_{i+1}.

La versione 4 di Kerberos usa un'estensione di CBC, chiamata modalità PCBC (Propagating Cipher Block Chaining) [MEYE82]. In questa modalità un errore in un blocco di testo cifrato si propaga a tutti i blocchi decrittografati successivi del messaggio, rendendoli tutti inutilizzabili. Pertanto la crittografia e l'integrità vengono combinate in un'unica operazione (per un'eccezione, vedere il problema 14.2). La modalità PCBC è illustrata nella Figura 14.9.

In questo schema, l'input dell'algoritmo di crittografia è lo XOR del blocco di testo in chiaro corrente, del blocco di testo cifrato precedente e del blocco di testo in chiaro precedente:

$$C_n = E(K, [C_{n-1} \oplus P_{n-1} \oplus P_n])$$

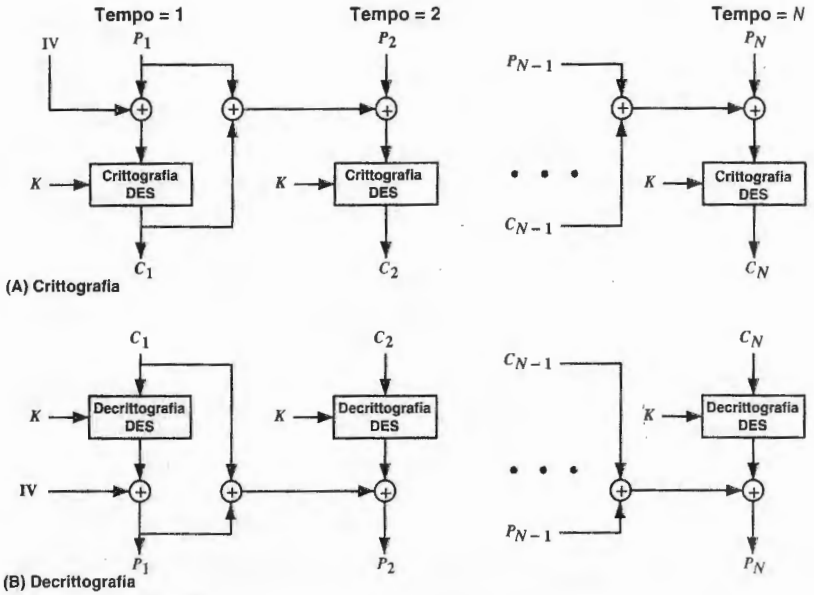


Figura 14.9 La modalità PCBC (Propagating Cipher Block Chaining).

Alla decrittografia, ciascun blocco di testo cifrato viene elaborato dall'algoritmo di decrittografia. Quindi l'output subisce uno XOR con il blocco di testo cifrato precedente e il blocco di testo in chiaro precedente. Si può dimostrare che questo schema funziona:

$$D(K, C_n) = D(K, \mathrm{E}(\mathrm{K}, [C_{n-1} \oplus P_{n-1} \oplus P_n]))$$
$$D(K, C_n) = C_{n-1} \oplus P_{n-1} \oplus P_n$$
$$C_{n-1} \oplus P_{n-1} \oplus D(K, C_n) = P_n$$

Capitolo 15

Sicurezza della posta elettronica

Concetti essenziali

- **PGP** è un package software open-source, pubblicamente disponibile, per la gestione della sicurezza della posta elettronica. Supporta l'autenticazione tramite la firma digitale, la segretezza tramite la crittografia a blocchi simmetrica, la compressione tramite l'algoritmo ZIP, la compatibilità tramite lo schema di codifica radix-64 e la segmentazione/ricomposizione per gestire i messaggi di grandi dimensioni.
- PGP incorpora gli strumenti necessari per sviluppare un modello di fiducia a chiave pubblica e per gestire i certificati di chiave pubblica.
- Lo standard Internet **S-MIME** rappresenta un altro approccio alla sicurezza della posta elettronica con le medesime funzionalità di PGP.

La posta elettronica è praticamente l'applicazione di rete più diffusa in qualsiasi ambiente distribuito. Inoltre, la posta elettronica è l'unica applicazione distribuita che viene ampiamente utilizzata attraverso qualsiasi architettura e piattaforma. Gli utenti possono inviare posta ad altri utenti connessi direttamente o indirettamente a Internet, indipendentemente dal sistema operativo o dal pacchetto di comunicazione utilizzati.

Proprio per il fatto che si conta sempre di più sulla posta elettronica per ogni scopo concepibile, cresce anche la richiesta di servizi di autenticazione e segretezza. In particolare vi sono due approcci che godono di un'ampia diffusione: PGP (Pretty Good Privacy) e S/MIME che rappresentano l'argomento di questo capitolo.

15.1 PGP (Pretty Good Privacy)

PGP è un fenomeno decisamente interessante. È un servizio di segretezza e autenticazione dovuto fondamentalmente all'iniziativa di una persona, Phil Zimmermann, che può essere utilizzato per la posta elettronica e per la memorizzazione dei file. Sostanzialmente, Zimmermann:

1. Ha impiegato i migliori algoritmi crittografici disponibili.
2. Ha integrato questi algoritmi in un'applicazione generica indipendente dal sistema operativo e dal microprocessore e basata su un piccolo insieme di comandi di facile uso.
3. Ha reso il pacchetto e la sua documentazione, compreso il codice sorgente, liberamente disponibili via Internet e altri mezzi.
4. Ha stretto accordi con una società (Viacrypt, ora Network Associates) per fornire una versione commerciale pienamente compatibile e a basso costo di PGP.

Da allora il sistema PGP è cresciuto in modo esplosivo ed è ora ampiamente utilizzato. Questa rapida crescita è dovuta a vari fattori.

1. È disponibile gratuitamente in tutto il mondo in versioni per molteplici piattaforme, fra cui Windows, Unix, Macintosh e molte altre. Inoltre la versione commerciale soddisfa gli utenti che desiderano un prodotto dotato di supporto da parte del produttore.
2. Si basa su algoritmi che hanno sostenuto ampie revisioni pubbliche e sono considerati estremamente sicuri. In particolare il pacchetto include gli algoritmi RSA, DSS e Diffie-Hellman per la crittografia a chiave pubblica, CAST-128, IDEA e 3DES per la crittografia simmetrica e SHA-1 per la codifica hash.
3. Ha un'ampia applicabilità, dalle grandi aziende che vogliono scegliere e adottare uno schema standardizzato per la crittografia dei file e dei messaggi fino ai singoli utenti che vogliono comunicare in modo sicuro con altri utenti in tutto il mondo attraverso Internet e altre reti.
4. Non è stato sviluppato e non è sotto il controllo di un'organizzazione governativa o di standardizzazione. Per coloro che hanno un'istintiva sfiducia verso le "cose ufficiali" PGP risulta molto interessante.
5. PGP sta ora diventando uno standard per Internet (documento RFC 3156). Ciononostante, PGP mantiene intatta la sua aura di strumento non "ufficiale".

Si partirà descrivendo in generale il funzionamento di PGP. Poi si vedrà come vengono create e memorizzate le chiavi crittografiche. Quindi si parlerà dell'argomento fondamentale della gestione della chiave pubblica.

Notazione

La maggior parte delle notazioni impiegate in questo capitolo è già stata introdotta in precedenza, ma alcuni termini sono nuovi. È forse meglio riepilogarli tutti all'inizio. Verranno utilizzati i seguenti simboli.

K_s = chiave di sessione utilizzata nello schema di crittografia simmetrica.
PR_a = chiave privata dell'utente A, utilizzata nello schema di crittografia a chiave pubblica.
PU_a = chiave pubblica dell'utente A, utilizzata nello schema di crittografia a chiave pubblica.
EP = crittografia a chiave pubblica.
DP = decrittografia a chiave pubblica.
EC = crittografia simmetrica.
DC = decrittografia simmetrica.
H = funzione hash.
|| = concatenamento.
Z = compressione con l'algoritmo ZIP.
R64 = conversione al formato ASCII radix-64.

La documentazione di PGP usa frequentemente il termine "chiave segreta" per far riferimento alla chiave associata a una chiave pubblica in uno schema di crittografia a chiave pubblica. Come si è detto in precedenza, questo termine rischia di entrare in conflitto con la chiave segreta utilizzata nella crittografia simmetrica. Pertanto in questo capitolo verrà preferito il termine chiave privata.

Descrizione operativa

Il funzionamento di PGP comprende cinque servizi: autenticazione, segretezza, compressione, compatibilità con la posta elettronica e segmentazione (Tabella 15.1). I seguenti paragrafi esaminano uno per uno questi servizi.

Tabella 15.1 Riepilogo dei servizi PGP.

Funzione	Algoritmo utilizzato	Descrizione
Firma digitale	DSS/SHA o RSA/SHA	Viene creato il codice hash del messaggio utilizzando SHA-1. Questo codice viene crittografato utilizzando DSS o RSA con la chiave privata del mittente; quindi viene incluso nel messaggio.
Crittografia del messaggio	CAST o IDEA o Triple DES a tre chiavi con Diffie-Hellman o RSA	Il messaggio viene crittografato utilizzando l'algoritmo CAST-128 o IDEA o 3DES con una chiave di sessione monouso generata dal mittente. La chiave di sessione viene crittografata utilizzando Diffie-Hellmon o RSA con la chiave pubblica del destinatario e viene inclusa nel messaggio.
Compressione	ZIP	Il messaggio può essere compresso, per agevolarne lo memorizzazione o la trasmissione, utilizzando l'algoritmo ZIP.
Compatibilità con la posta elettronica	Conversione radix-64	Per garantire la trasparenza per le applicazioni di posta elettronica, il messaggio crittografato può essere convertito in una stringa hash utilizzando la conversione radix-64.
Segmentazione	–	Per evitare i problemi dovuti alle dimensioni del messaggio, PGP comprende le operazioni di segmentazione e riassemblaggio.

Autenticazione

La Figura 15.1A illustra il servizio di firma digitale fornito da PGP. Si tratta dello schema a firma digitale trattato nel Capitolo 13 e illustrato nella Figura 11.5C. Ecco la sequenza impiegata.

1. Il mittente crea un messaggio.
2. Viene utilizzato SHA-1 per generare un codice hash a 160 bit dell'intero messaggio.
3. Il codice hash viene crittografato con RSA utilizzando la chiave privata del mittente e il risultato viene posizionato davanti al messaggio.
4. Il destinatario usa RSA per decrittografare il codice hash utilizzando la chiave pubblica del mittente.
5. Il destinatario genera un nuovo codice hash per il messaggio e lo confronta con il codice hash decrittografato. Se i due codici coincidono, il messaggio viene accettato come autentico.

La combinazione di SHA-1 e RSA fornisce un meccanismo efficace di firma digitale. Data la resistenza di RSA, il destinatario sa che solo il possessore della chiave privata corrispondente può generare la firma. Grazie alla resistenza di SHA-1, il destinatario sa che nessun altro potrebbe aver generato un nuovo messaggio corrispondente al codice hash e, pertanto, neanche la firma del messaggio originale.

In alternativa le firme possono essere generate utilizzando DSS/SHA-1.

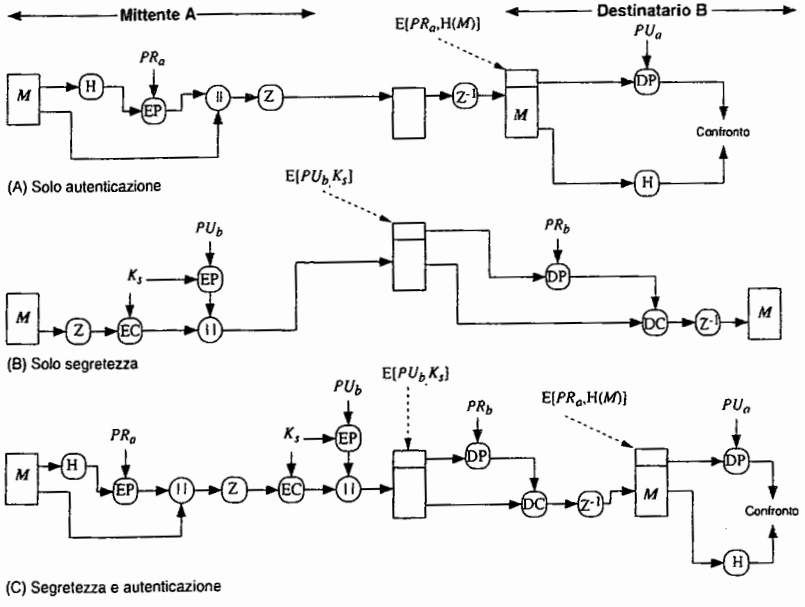


Figura 15.1 Funzioni crittografiche di PGP

Sebbene le firme vengano normalmente allegate al messaggio o al file firmato, non sempre le cose stanno così: è anche possibile prevedere firme separate. Una firma separata può essere memorizzata e trasmessa in modo distinto rispetto al messaggio firmato. Questa possibilità è utile in vari contesti. Un utente può voler conservare una registrazione separata delle firme di tutti i messaggi inviati o ricevuti. Una firma separata di un programma eseguibile consente di rilevare l'eventuale infezione di un virus. Infine la possibilità di avere firme separate può essere utilizzata quando un documento deve essere firmato da più parti, come nel caso di un contratto legale. Ogni firma è indipendente e pertanto viene applicata solo al documento. Altrimenti sarebbe necessario produrre un annidamento di firme dove il secondo firmatario firma sia il documento che la prima firma e così via.

Segretezza

Un altro servizio fondamentale offerto da PGP è la segretezza, garantita dalla crittografia dei messaggi da trasmettere o da memorizzare localmente come file. In entrambi i casi può essere utilizzato l'algoritmo a crittografia simmetrica CAST-128. Alternativamente si possono utilizzare IDEA o 3DES. Viene utilizzata la modalità di cifratura a 64 bit CFB (Cipher FeedBack).

Come sempre si deve risolvere il problema della distribuzione della chiave. In PGP, ogni chiave simmetrica viene utilizzata una sola volta. Ovvero per ciascun messaggio viene generata una nuova chiave sotto forma di numero casuale a 128 bit. Pertanto, sebbene nella documentazione si parli di chiave di sessione, in realtà si tratta di una chiave monouso. Poiché viene utilizzata una sola volta, la chiave di sessione è legata al messaggio e trasmessa insieme ad esso. Per proteggere la chiave, questa viene crittografata con la chiave pubblica del destinatario. La Figura 15.1B illustra la sequenza utilizzata che può essere descritta nel seguente modo.

1. Il mittente genera un messaggio e un numero casuale a 128 bit che verrà utilizzato come chiave di sessione unicamente per questo messaggio.
2. Il messaggio viene crittografato utilizzando l'algoritmo CAST-128 (o IDEA o 3DES) con la chiave di sessione.
3. La chiave di sessione viene crittografata con RSA utilizzando la chiave pubblica del destinatario e viene fatta precedere al messaggio.
4. Il destinatario usa RSA con la propria chiave privata per decrittografare ed estrarre la chiave di sessione.
5. La chiave di sessione viene utilizzata per decrittografare il messaggio.

In alternativa a RSA per la crittografia della chiave, PGP fornisce un'opzione chiamata Diffie-Hellman. Come si è detto nel Capitolo 10, Diffie-Hellman è un algoritmo per lo scambio della chiave. In realtà, PGP usa una variante di Diffie-Hellman, chiamata ElGamal, che fornisce anche la crittografia/decrittografia (vedere il Problema 10.6). Si possono fare varie osservazioni. Innanzitutto, per ridurre il tempo di crittografia si preferisce utilizzare la combinazione di crittografia simmetrica e a chiave pubblica, piuttosto che usare semplicemente RSA o ElGamal per crittografare direttamente il messaggio: CAST-128 e gli altri algoritmi simmetrici sono notevolmente più veloci di RSA o ElGamal. In secondo luogo, l'uso dell'algoritmo a chiave pubblica risolve il problema della distribuzione della chiave di sessione, poiché solo il destinatario sarà in grado di recuperare la chiave di sessione

associata al messaggio. Si noti che non è necessario un protocollo per lo scambio delle chiavi del tipo trattato nel Capitolo 10, poiché non si sta iniziando una sessione. Piuttosto, ciascun messaggio è un evento unico e indipendente con la propria chiave. Inoltre, data la natura di memorizzazione e inoltro della posta elettronica, non si può utilizzare l'handshaking per garantire che entrambi i lati abbiano la stessa chiave di sessione. Infine, l'uso delle chiavi simmetriche monouso rafforza l'approccio a crittografia simmetrica già sufficientemente resistente. Ciascuna chiave serve per crittografare solo una piccola quantità di testo in chiaro e non esiste alcuna relazione fra le chiavi. Pertanto, se l'algoritmo a chiave pubblica è sicuro, l'intero meccanismo risulterà sicuro. A questo proposito, PGP fornisce all'utente l'opzione di utilizzare chiavi comprese fra 768 e 3072 bit (la chiave DSS per le firme è limitata a 1024 bit).

Segretezza e autenticazione

Come si può vedere nella Figura 15.1C, è possibile applicare allo stesso messaggio entrambi i servizi. Innanzitutto viene generata una firma per il messaggio in chiaro che viene fatta precedere al messaggio. Poi il messaggio di testo in chiaro più la firma vengono crittografati utilizzando l'algoritmo CAST-128 (o IDEA o 3DES) e la chiave di sessione viene crittografata utilizzando RSA (o ElGamal). Questa sequenza è preferibile all'inversa (la crittografia del messaggio e poi la generazione di una firma per il messaggio crittografato); in generale è più comodo memorizzare una firma con la versione di testo in chiaro del messaggio. Inoltre, per consentire verifiche da parte di terzi, se per prima venisse eseguita la firma, una terza parte non dovrebbe avere necessariamente la chiave simmetrica per verificare la firma.

In breve, quando vengono impiegati entrambi i servizi, il mittente firma il messaggio con la propria chiave privata, poi esegue la crittografia del messaggio con una chiave di sessione e infine esegue la crittografia della chiave di sessione con la chiave pubblica del destinatario.

Compressione

Come impostazione predefinita, PGP esegue la compressione del messaggio dopo aver applicato la firma ma prima della crittografia. Questo presenta il vantaggio di risparmiare spazio sia per la trasmissione della posta elettronica che per la memorizzazione dei file.

La posizione dell'algoritmo di compressione, indicato nella Figura 15.1 da Z per la compressione e Z^{-1} per la decompressione è fondamentale.

1. La firma viene generata prima della compressione per due motivi.
   A. È preferibile firmare un messaggio non compresso in modo da poter memorizzare solo il messaggio non compresso insieme alla firma per successive operazioni di verifica. Se si firmasse un documento compresso, sarebbe necessario memorizzare una versione compressa del messaggio per l'eventuale verifica successiva oppure ricomprimere il messaggio ogni volta che occorre verificarlo.
   B. Anche se si volesse generare dinamicamente un messaggio ricompresso per la verifica sorgerebbe una difficoltà con l'algoritmo di compressione di PGP. L'algoritmo non è deterministico: le varie implementazioni dell'algoritmo ottengono compromessi differenti per quanto riguarda la velocità di esecuzione e il rapporto di compressione e, pertanto, possono produrre forme compresse differenti. Tuttavia, questi diversi algoritmi di compressione sono interoperabili poiché ogni versione dell'algoritmo può decomprimere correttamente l'output prodotto da ogni altra versione. L'applicazione della funzione hash e della firma dopo la compressione costringerebbe tutte le implementazioni di PGP a utilizzare la stessa versione dell'algoritmo di compressione.
2. La crittografia del messaggio viene applicata dopo la compressione per rafforzare la sua sicurezza crittografica. Poiché il messaggio compresso contiene meno ridondanza rispetto al testo in chiaro originale, l'analisi crittografica risulterà più difficile.

L'algoritmo di compressione utilizzato è ZIP, descritto nell'Appendice 15.A.

Compatibilità con la posta elettronica

Quando viene utilizzato PGP, viene crittografata quanto meno una parte del blocco da trasmettere. Se viene utilizzato solo il servizio di firma digitale, viene crittografato il codice digest (con la chiave privata del mittente). Se viene utilizzato il servizio di segretezza, vengono crittografati sia il messaggio che l'eventuale firma utilizzando una chiave simmetrica monouso. Pertanto una parte o l'intero blocco risultante è costituito da un flusso di ottetti arbitrari di 8 bit. Tuttavia molti sistemi di posta elettronica consentono di impiegare solo blocchi costituiti da testo ASCII. Per tenere conto di questo vincolo, PGP fornisce il servizio di conversione del flusso grezzo di 8 bit binari in un flusso di caratteri ASCII stampabili.

Il meccanismo utilizzato per questo scopo è la conversione radix-64. Ciascun gruppo di tre ottetti di dati binari viene mappato in quattro caratteri hash. Questo formato aggiunge anche un codice CRC per il rilevamento degli errori di trasmissione. Per una descrizione si veda l'Appendice 15.B.

L'uso della conversione radix-64 espande un messaggio del 33%. Fortunatamente le porzioni relative alla chiave di sessione e alla firma sono relativamente compatte e il messaggio in chiaro è compresso. In effetti la compressione dovrebbe essere più che sufficiente per compensare l'espansione dovuta alla conversione radix-64. Per esempio, [HELD96] indica un rapporto di compressione medio di circa 2,0 con ZIP. Se si ignorano la firma e i componenti della chiave relativamente compatti, l'effetto globale medio della compressione e dell'espansione di un file di lunghezza X sarà $1{,}33 \times 0{,}5 \times X = 0{,}665 \times X$. Pertanto si otterrà comunque una compressione di circa un terzo.

Un aspetto degno di nota dell'algoritmo radix-64 è il fatto che converte in modo "cieco" il flusso di input in un formato radix-64, indipendentemente dal contenuto, anche se l'input è costituito da testo ASCII. Pertanto, se un messaggio viene firmato ma non crittografato e la conversione viene applicata all'intero blocco, l'output risulterà illeggibile per l'osservatore casuale, cosa che questo offre un certo livello di segretezza. Opzionalmente PGP può essere configurato per convertire in formato radix-64 solo la firma dei messaggi in chiaro firmati. Questo consente al destinatario di leggere il messaggio senza utilizzare PGP. PGP dovrà comunque essere utilizzato per verificare la firma.

La Figura 15.2 mostra la relazione esistente fra i quattro servizi trattati finora. In trasmissione, se necessario, viene generata una firma utilizzando un codice hash del testo in chiaro non compresso. Poi il testo in chiaro più l'eventuale firma vengono compressi.

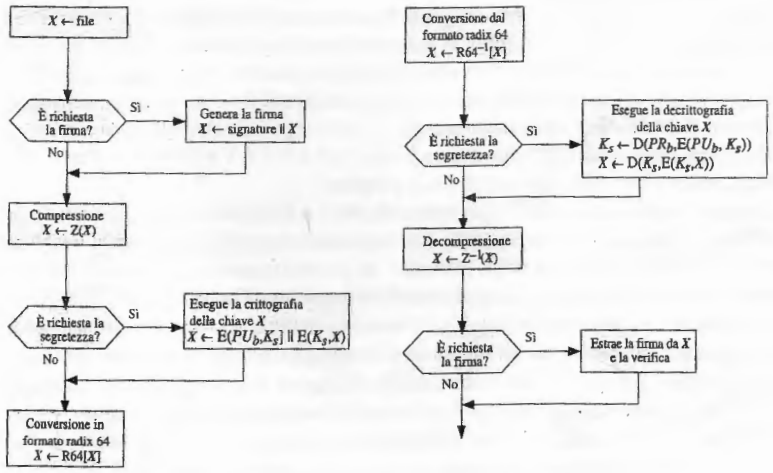
Quindi, se è richiesta la segretezza, il blocco (testo in chiaro compresso o firma più testo in chiaro compressi) viene crittografato e gli viene fatta precedere la chiave di crittografia simmetrica, crittografata con la chiave pubblica. Infine l'intero blocco viene convertito in formato radix-64.

Alla ricezione, il blocco in ingresso viene innanzitutto convertito nuovamente dal formato radix-64 al formato binario. Poi, se il messaggio è stato crittografato, il destinatario ripristina la chiave di sessione ed esegue la decrittografia del messaggio. Il blocco risultante viene poi decompresso. Se il messaggio è firmato, il destinatario ripristina il codice hash trasmesso e lo confronta con quello calcolato localmente.

Frammentazione e riassemblaggio

I sistemi di posta elettronica pongono spesso dei limiti alla lunghezza massima dei messaggi. Per esempio, molti dei sistemi accessibili via Internet impongono una lunghezza massima di 50 000 ottetti. Ogni messaggio più lungo deve essere suddiviso in più segmenti che devono essere inviati separatamente.

Per poter gestire questo limite, PGP suddivide automaticamente un messaggio troppo esteso in segmenti di dimensioni sufficientemente ridotte per la trasmissione via posta elettronica. La segmentazione viene svolta dopo ogni altra elaborazione, compresa la conversione radix-64. Pertanto la chiave di sessione e la firma compariranno una sola volta all'inizio del primo segmento. All'estremità ricevente, PGP deve eliminare tutte le intestazioni di posta elettronica e assemblare l'intero blocco originale prima di svolgere le operazioni illustrate nella Figura 15.2B.



(A) Schema di trasmissione generica da A

(B) Schema di ricezione generica da B

Figura 15.2 Trasmissione e ricezione dei messaggi PGP

Chiavi crittografiche e key ring

PGP usa quattro tipi di chiavi: chiavi simmetriche di sessione monouso, chiavi pubbliche, chiavi private e chiavi simmetriche a frase segreta (se ne parlerà più avanti). Queste chiavi rispondono a tre diversi requisiti.

1. È necessario un modo per generare chiavi di sessione non prevedibili.
2. Si vorrebbe consentire a un utente di avere più coppie chiave pubblica/chiave privata. Una delle ragioni è che l'utente può voler cambiare la propria coppia di chiavi di tanto in tanto. In questi casi, tutti i messaggi in corso verranno costruiti con una chiave obsoleta. Inoltre i destinatari conosceranno solo la vecchia chiave pubblica finché non saranno raggiunti dall'aggiornamento. Oltre alla necessità di cambiare le chiavi di tanto in tanto, un utente può anche voler avere più coppie di chiavi per interagire con gruppi di corrispondenti differenti o semplicemente per migliorare la sicurezza limitando la quantità di materiale crittografato con una stessa chiave. Il risultato è che non esiste una corrispondenza uno-a-uno fra gli utenti e le loro chiavi pubbliche. Pertanto è necessario un mezzo per identificare le chiavi.
3. Ogni entità PGP deve conservare un file delle proprie coppie di chiavi pubbliche/private e un file delle chiavi pubbliche dei corrispondenti.

I seguenti paragrafi esaminano uno per uno questi requisiti.

Generazione della chiave di sessione

Ogni chiave di sessione è associata a un unico messaggio e viene utilizzata con il solo scopo di crittografare e decrittografare tale messaggio. La crittografia/decrittografia del messaggio viene eseguita con un algoritmo di crittografia simmetrico. CAST-128 e IDEA usano chiavi a 128 bit; 3DES usa una chiave a 168 bit. In questo caso si supporrà l'impiego di CAST-128.

CAST-128 consente di generare numeri casuali a 128 bit. L'input del generatore di numeri casuali è costituito da una chiave di 128 bit e due blocchi di 64 bit che vengono trattati come testo in chiaro da crittografare. Utilizzando la modalità cipher feedback, la crittografia CAST-128 produce due blocchi di testo cifrato da 64 bit che vengono concatenati per formare la chiave di sessione a 128 bit. L'algoritmo usato si basa su quello specificato in ANSI X12.17.

Il "testo in chiaro" che funge da input del generatore di numeri casuali, costituito da due blocchi di 64 bit, deriva esso stesso da un flusso di numeri casuali di 128 bit. Questi numeri si basano sulle informazioni introdotte alla tastiera dall'utente. Per generare il flusso casuale vengono utilizzati sia i tasti effettivamente premuti dall'utente che la durata delle pause fra le varie pressioni. Pertanto, se l'utente preme tasti arbitrari alla propria velocità normale, verrà generato un input ragionevolmente "casuale". Questo input casuale viene combinato con la precedente chiave di sessione prodotta da CAST-128 per creare la chiave di input per il generatore. Il risultato, data l'efficacia di CAST-128, produce una sequenza di chiavi di sessione assolutamente imprevedibili.

L'Appendice 15.C discute più in dettaglio le tecniche per la generazione di numeri casuali di PGP.

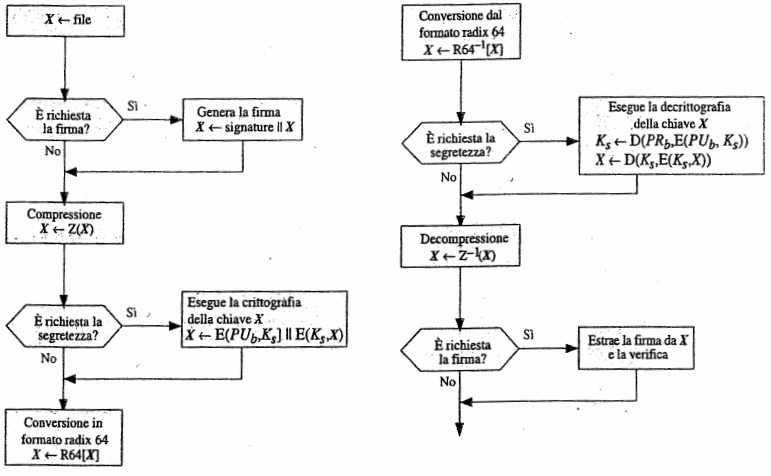
Quindi, se è richiesta la segretezza, il blocco (testo in chiaro compresso o firma più testo in chiaro compressi) viene crittografato e gli viene fatta precedere la chiave di crittografia simmetrica, crittografata con la chiave pubblica. Infine l'intero blocco viene convertito in formato radix-64.

Alla ricezione, il blocco in ingresso viene innanzitutto convertito nuovamente dal formato radix-64 al formato binario. Poi, se il messaggio è stato crittografato, il destinatario ripristina la chiave di sessione ed esegue la decrittografia del messaggio. Il blocco risultante viene poi decompresso. Se il messaggio è firmato, il destinatario ripristina il codice hash trasmesso e lo confronta con quello calcolato localmente.

Frammentazione e riassemblaggio

I sistemi di posta elettronica pongono spesso dei limiti alla lunghezza massima dei messaggi. Per esempio, molti dei sistemi accessibili via Internet impongono una lunghezza massima di 50 000 ottetti. Ogni messaggio più lungo deve essere suddiviso in più segmenti che devono essere inviati separatamente.

Per poter gestire questo limite, PGP suddivide automaticamente un messaggio troppo esteso in segmenti di dimensioni sufficientemente ridotte per la trasmissione via posta elettronica. La segmentazione viene svolta dopo ogni altra elaborazione, compresa la conversione radix-64. Pertanto la chiave di sessione e la firma compariranno una sola volta all'inizio del primo segmento. All'estremità ricevente, PGP deve eliminare tutte le intestazioni di posta elettronica e assemblare l'intero blocco originale prima di svolgere le operazioni illustrate nella Figura 15.2B.



(A) Schema di trasmissione generica da A

(B) Schema di ricezione generica da B

Figura 15.2 Trasmissione e ricezione dei messaggi PGP

Chiavi crittografiche e key ring

PGP usa quattro tipi di chiavi: chiavi simmetriche di sessione monouso, chiavi pubbliche, chiavi private e chiavi simmetriche a frase segreta (se ne parlerà più avanti). Queste chiavi rispondono a tre diversi requisiti.

1. È necessario un modo per generare chiavi di sessione non prevedibili.
2. Si vorrebbe consentire a un utente di avere più coppie chiave pubblica/chiave privata. Una delle ragioni è che l'utente può voler cambiare la propria coppia di chiavi di tanto in tanto. In questi casi, tutti i messaggi in corso verranno costruiti con una chiave obsoleta. Inoltre i destinatari conosceranno solo la vecchia chiave pubblica finché non saranno raggiunti dall'aggiornamento. Oltre alla necessità di cambiare le chiavi di tanto in tanto, un utente può anche voler avere più coppie di chiavi per interagire con gruppi di corrispondenti differenti o semplicemente per migliorare la sicurezza limitando la quantità di materiale crittografato con una stessa chiave. Il risultato è che non esiste una corrispondenza uno-a-uno fra gli utenti e le loro chiavi pubbliche. Pertanto è necessario un mezzo per identificare le chiavi.
3. Ogni entità PGP deve conservare un file delle proprie coppie di chiavi pubbliche/private e un file delle chiavi pubbliche dei corrispondenti.

I seguenti paragrafi esaminano uno per uno questi requisiti.

Generazione della chiave di sessione

Ogni chiave di sessione è associata a un unico messaggio e viene utilizzata con il solo scopo di crittografare e decrittografare tale messaggio. La crittografia/decrittografia del messaggio viene eseguita con un algoritmo di crittografia simmetrico. CAST-128 e IDEA usano chiavi a 128 bit; 3DES usa una chiave a 168 bit. In questo caso si supporrà l'impiego di CAST-128.

CAST-128 consente di generare numeri casuali a 128 bit. L'input del generatore di numeri casuali è costituito da una chiave di 128 bit e due blocchi di 64 bit che vengono trattati come testo in chiaro da crittografare. Utilizzando la modalità cipher feedback, la crittografia CAST-128 produce due blocchi di testo cifrato da 64 bit che vengono concatenati per formare la chiave di sessione a 128 bit. L'algoritmo usato si basa su quello specificato in ANSI X12.17.

Il "testo in chiaro" che funge da input del generatore di numeri casuali, costituito da due blocchi di 64 bit, deriva esso stesso da un flusso di numeri casuali di 128 bit. Questi numeri si basano sulle informazioni introdotte alla tastiera dall'utente. Per generare il flusso casuale vengono utilizzati sia i tasti effettivamente premuti dall'utente che la durata delle pause fra le varie pressioni. Pertanto, se l'utente preme tasti arbitrari alla propria velocità normale, verrà generato un input ragionevolmente "casuale". Questo input casuale viene combinato con la precedente chiave di sessione prodotta da CAST-128 per creare la chiave di input per il generatore. Il risultato, data l'efficacia di CAST-128, produce una sequenza di chiavi di sessione assolutamente imprevedibili.

L'Appendice 15.C discute più in dettaglio le tecniche per la generazione di numeri casuali di PGP.

Gli identificatori delle chiavi

Come si è detto in precedenza, un messaggio crittografato è accompagnato dalla versione crittografata della chiave di sessione utilizzata per crittografare il messaggio. La stessa chiave di sessione viene crittografata con la chiave pubblica del destinatario. Pertanto, solo il destinatario sarà in grado di recuperare la chiave di sessione e quindi il messaggio. Se ogni utente impiegasse un'unica coppia di chiavi pubbliche/private, il destinatario saprebbe automaticamente quale chiave utilizzare per decrittografare la chiave di sessione: la sua unica chiave privata. Tuttavia uno dei requisiti prevede che ciascun utente possa avere più coppie di chiavi pubbliche/private.

Detto questo, come può il destinatario sapere quale delle sue chiavi pubbliche è stata utilizzata per crittografare la chiave di sessione? Una semplice soluzione consisterebbe nel trasmettere la chiave pubblica insieme al messaggio. Il destinatario potrebbe quindi verificare che questa è in effetti una delle sue chiavi pubbliche e procedere. Questo meccanismo funzionerebbe ma rappresenta un inutile spreco di spazio. Una chiave pubblica RSA può avere una lunghezza di centinaia di cifre decimali. Un'altra soluzione consisterebbe nell'associare un identificatore a ciascuna chiave pubblica, che è univoca per ciascun utente. La combinazione codice utente/codice della chiave sarebbe sufficiente per identificare in modo univoco la chiave. In questo modo sarebbe sufficiente trasmettere il codice ID della chiave, molto più breve. Ma questa soluzione solleva un problema di gestione e sovraccarico: sarebbe necessario assegnare un codice ID alla chiave, che dovrebbe essere memorizzato sia dal mittente che dal destinatario per poterlo associare alla chiave pubblica. Si tratta di un meccanismo inutilmente complesso.

La soluzione adottata da PGP consiste nell'assegnare un identificatore (key ID) a ciascuna chiave pubblica che risulti, con elevate probabilità, univoca per ciascun utente.[1] Il key ID associato a ciascuna chiave pubblica è costituito dai suoi 64 bit meno significativi. Ovvero il key ID della chiave pubblica PU_a è (PU_a mod 2^{64}). Si tratta di una lunghezza sufficiente per ridurre la probabilità che i key ID di due chiavi siano uguali.

È necessario anche un identificatore della chiave per la firma digitale PGP. Poiché un mittente può usare più chiavi private per crittografare il message digest, il destinatario deve sapere quale chiave pubblica utilizzare. Di conseguenza la componente della firma digitale di un messaggio include il key ID a 64 bit della chiave pubblica richiesta. Quando viene ricevuto il messaggio, il destinatario verifica che il key ID è quello di una determinata chiave pubblica di tale mittente e quindi procede a verificare la firma.

Dopo avere introdotto il concetto di key ID, si può dare una descrizione più dettagliata del formato di un messaggio trasmesso, rappresentato nella Figura 15.3. Un messaggio è costituito da tre componenti: il messaggio, una firma opzionale e una chiave di sessione opzionale.

La componente **messaggio** include i dati che devono essere memorizzati o trasmessi, oltre al nome del file e a un timestamp che indica il momento della creazione.

La componente **firma** include i seguenti elementi.

[1] Si è già parlato nel Paragrafo 8.3 dei concetti di calcolo delle probabilità necessari per determinare se un numero è primo. Nella progettazione degli algoritmi si usano frequentemente delle tecniche probabilistiche che generano soluzioni più veloci o meno complesse, o entrambe le cose insieme.

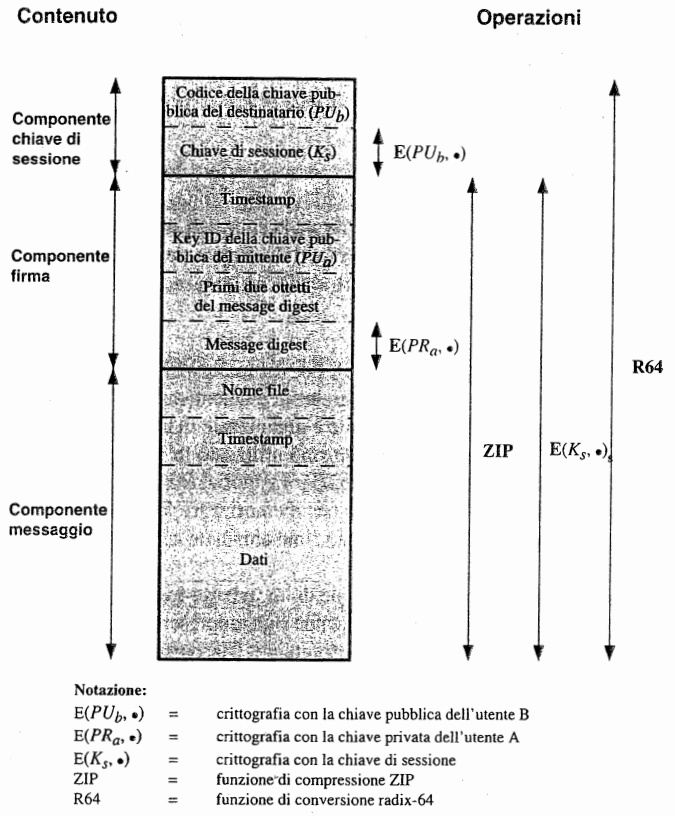


Figura 15.3 Il formato generale di un messaggio PGP (da A a B).

- **Timestamp:** il momento in cui è stata apposta la firma.
- **Message digest:** il codice SHA-1 a 160 bit crittografato con la chiave privata del mittente. Il codice viene calcolato in base al valore timestamp della firma concatenato con la porzione dati del messaggio. L'inclusione del valore timestamp della firma nel codice digest garantisce contro gli attacchi a replay. L'esclusione del nome del file e del codice timestamp della componente del messaggio garantisce che le firme separate siano esattamente le stesse firme allegate che precedono il messaggio. Le firme separate vengono calcolate su un file distinto che non ha nessuno dei campi di intestazione del messaggio.
- **Primi due ottetti del message digest:** per consentire al destinatario di determinare se è stata utilizzata la chiave pubblica corretta per decrittografare il codice message digest per l'autenticazione, confrontando questa copia dei primi due ottetti di testo in chiaro

con i primi due ottetti del digest decrittografato. Questi ottetti rappresentano anche il codice di rilevazione degli errori a 16 bit del messaggio.
- **Key ID della chiave pubblica del mittente:** identifica la chiave pubblica che deve essere utilizzata per decrittografare il message digest e, pertanto, identifica la chiave privata utilizzata per crittografarlo.

Le componenti messaggio e firma opzionale possono essere compressi utilizzando ZIP e possono essere crittografati utilizzando una chiave di sessione.

La componente **chiave di sessione** include sia la chiave di sessione che l'identificatore della chiave pubblica del destinatario utilizzata dal mittente per crittografare la chiave di sessione. L'intero blocco viene normalmente codificato con radix-64.

Key ring

Si è visto come i key ID siano fondamentali per il funzionamento di PGP e che in ogni messaggio PGP che fornisce i servizi di segretezza e autenticazione vengono inclusi due key ID. Le chiavi devono essere memorizzate e organizzate in modo sistematico per poter essere utilizzate in modo efficace. Il meccanismo utilizzato in PGP consiste nel fornire a ciascun nodo una coppia di strutture dati, una per memorizzare le coppie di chiavi pubbliche/private di proprietà di tale nodo e una per memorizzare le chiavi pubbliche di altri utenti noti a questo nodo. Queste strutture dati sono chiamate, rispettivamente, private-key ring e public-key ring.

La Figura 15.4 mostra la struttura generale di un **private-key ring**. Si può considerare questo elemento come una tabella in cui ciascuna riga rappresenta una delle coppie chiave pubblica/chiave privata di proprietà di questo utente. Ciascuna riga contiene le seguenti voci.

- **Timestamp:** la data/ora in cui è stata generata la coppia di chiavi di questa voce.
- **Key ID:** i 64 bit meno significativi della chiave pubblica di questa voce.
- **Chiave pubblica:** la porzione pubblica della coppia di chiavi.
- **Chiave privata:** la porzione privata della coppia di chiavi; questo campo è crittografato.
- **Codice utente:** normalmente si tratta dell'indirizzo di posta elettronica dell'utente. Tuttavia l'utente può scegliere di associare un nome specifico a ciascuna coppia o di utilizzare lo stesso codice utente più volte.

Il private-key ring può essere indicizzato in base al codice utente o al key ID; più avanti si vedrà la necessità di utilizzare entrambi i metodi di indicizzazione.

Sebbene il private-key ring debba essere memorizzato solo sulla macchina dell'utente che ha creato e che possiede le coppie di chiavi e sebbene debba essere accessibile solo a a tale utente, è opportuno rendere il più sicuro possibile il valore della chiave privata. Di conseguenza la chiave privata non viene memorizzata in chiaro nel key ring ma viene crittografata utilizzando l'algoritmo CAST-128 (o IDEA o 3DES). La procedura è la seguente.

1. L'utente seleziona una frase segreta che verrà utilizzata per la crittografia delle chiavi private.

Private-key ring

Timestamp	Key ID*	Chiave pubblica	Chiave privata crittografata	Codice* utente
•	•	•	•	•
•	•	•	•	•
•	•	•	•	•
T_i	$PU_i \bmod 2^{64}$	PU_i	$E(H(P_i), PR_i)$	Utente i
•	•	•	•	•
•	•	•	•	•
•	•	•	•	•

Public-key ring

Timestamp	Key ID*	Chiave pubblica	Trust del proprietario	Codice* utente	Legittimità chiave	Firma	Trust della firma
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•
T_i	$PU_i \bmod 2^{64}$	PU_i	$trust\_flag_i$	$Utente_i$	$trust\_flag_i$		
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•

* = Campo utilizzato per indicizzare la tabella

Figura 15.4 Struttura dei private-key ring e public-key ring.

2. Quando il sistema genera una nuova coppia chiave pubblica/chiave privata utilizzando RSA, richiede all'utente la frase segreta. Utilizzando SHA-1, viene generato un codice hash di 160 bit a partire dalla frase segreta che poi viene eliminata.
3. Il sistema esegue la crittografia della chiave privata utilizzando l'algoritmo CAST-128 utilizzando come chiave i 128 bit del codice hash. Il codice hash viene poi eliminato e la chiave privata crittografata viene conservata nel private-key ring.

Successivamente, quando un utente accede al private-key ring per prelevare una chiave privata, deve fornire la frase segreta. PGP preleva la chiave privata crittografata, genera il codice hash per la frase segreta ed esegue la decrittografia della chiave privata utilizzando l'algoritmo CAST-128 con il codice hash.

Si tratta di un meccanismo molto compatto ed efficace. Come in ogni sistema basato su password, la sicurezza del sistema dipende dalla sicurezza della password. Per evitare la tentazione di trascriverla, l'utente dovrebbe utilizzare una frase segreta difficile da indovinare ma facile da ricordare.

La Figura 15.4 mostra anche la struttura generale del **public-key ring**. Questa struttura dati viene utilizzata per memorizzare le chiavi pubbliche di altri utenti noti all'utente. Per il momento, si possono ignorare alcuni campi della tabella e descrivere solo i seguenti:

- **Timestamp:** la data/ora in cui è stata generata la voce.

- **Key ID:** i 64 bit meno significativi della chiave pubblica di questa voce.
- **Chiave pubblica:** la chiave pubblica di questa voce.
- **Codice utente:** identifica il proprietario della chiave. A una sola chiave pubblica possono essere associati più codici utente.

Il public-key ring può essere indicizzato in base al codice utente o al key ID; più avanti si vedrà la necessità di impiegare entrambe queste indicizzazioni.

Ora si può vedere come questi key ring vengono utilizzati nella trasmissione e ricezione di un messaggio. Per semplicità si ignorerà la compressione e la conversione radix-64. Innanzitutto si consideri la trasmissione del messaggio (Figura 15.5) supponendo che questo sia da firmare e da crittografare. L'entità PGP del mittente svolge le seguenti operazioni.

1. Firma del messaggio.
   A. PGP estrae la chiave privata del mittente dal private-key ring utilizzando come indice il codice dell'utente. Se il codice utente non viene fornito, viene estratta la prima chiave privata del key ring.
   B. PGP chiede all'utente la fase segreta per ripristinare in chiaro la chiave privata non crittografata.
   C. Viene costruita la componente di firma del messaggio.
2. Crittografia del messaggio.

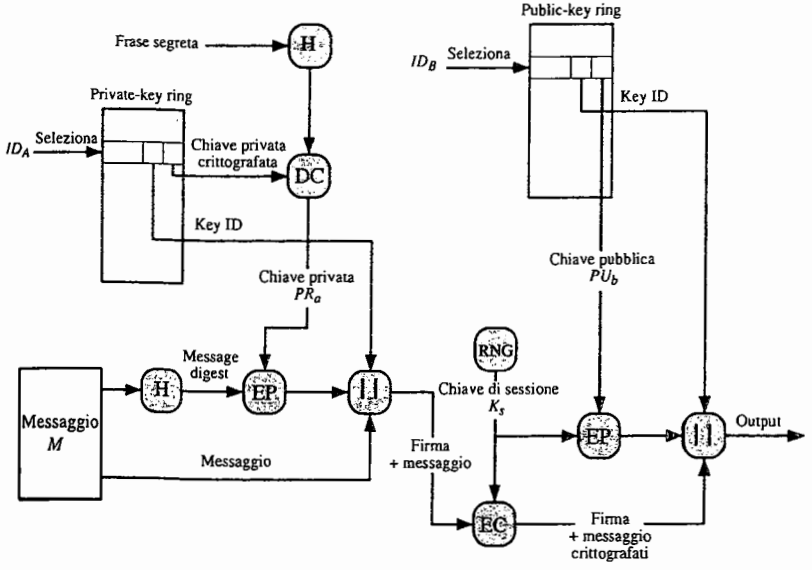


Figura 15.5 Generazione del messaggio PGP (dall'utente A all'utente B senza considerare la compressione e la conversione radix-64).

   A. PGP genera una chiave di sessione ed esegue la crittografia del messaggio.
   B. PGP preleva la chiave pubblica del destinatario dal public-key ring utilizzando come indice il suo codice utente.
   C. Viene costruita la componente chiave di sessione del messaggio.

L'entità PGP ricevente svolge le seguenti operazioni (Figura 15.6).

1. Decrittografia del messaggio.
   A. PGP preleva la chiave privata del destinatario dal private-key ring utilizzando come indice il key ID nella componente della chiave di sessione del messaggio.
   B. PGP chiede all'utente la frase segreta per ripristinare in chiaro la chiave privata.
   C. PGP recupera la chiave di sessione ed esegue la decrittografia del messaggio.
2. Autenticazione del messaggio.
   A. PGP preleva la chiave pubblica del mittente dal public-key ring, utilizzando come indice il key ID nella firma del messaggio.
   B. PGP recupera il message digest trasmesso.
   C. PGP calcola il message digest del messaggio ricevuto e lo confronta con quello trasmesso per autenticarlo.

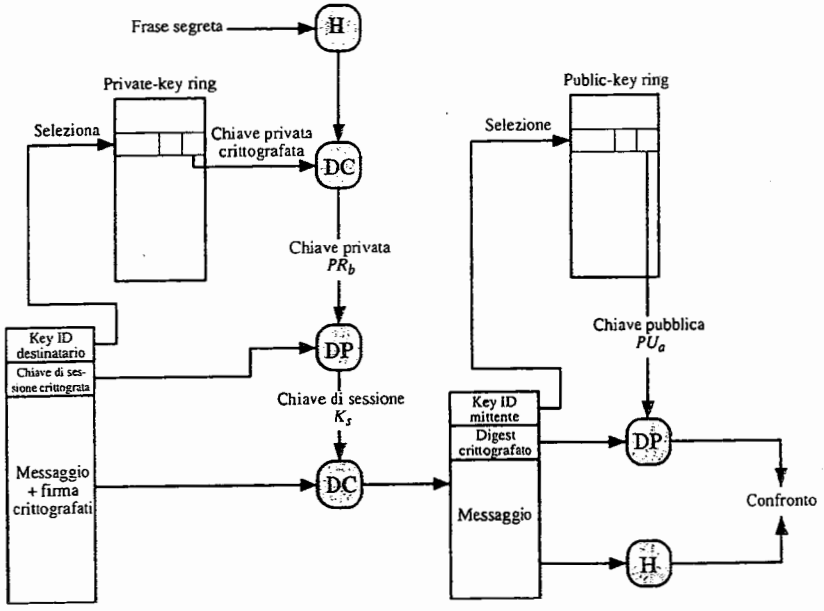


Figura 15.6 Ricezione PGP (messaggio dall'utente A all'utente B, senza compressione e conversione radix-64).

Gestione della chiave pubblica

Come si è visto finora, PGP è un insieme interessante, efficiente e interconnesso di funzioni e formati in grado di garantire un efficace servizio di segretezza e autenticazione. Per completare il sistema, occorre considerare anche il problema della gestione della chiave pubblica. La documentazione di PGP evidenzia l'importanza di questo campo:

Il fatto di proteggere le chiavi pubbliche contro ogni manomissione rappresenta il problema di gran lunga più difficile nell'applicazione pratica delle chiavi pubbliche. È in sostanza il "tallone d'Achille" della crittografia a chiave pubblica e in gran parte la complessità del software è dovuta alla soluzione di questo problema.

PGP fornisce una struttura per risolvere questo problema, suggerendo la possibilità di utilizzare varie opzioni. Poiché PGP deve poter essere utilizzato in vari ambienti formali e informali, non è previsto alcun meccanismo di gestione rigido delle chiavi pubbliche come quello che si vedrà per S/MIME più avanti in questo stesso capitolo.

Approcci alla gestione della chiave pubblica

Sostanzialmente il problema è il seguente: l'utente A deve costruire un public-key ring contenente le chiavi pubbliche di tutti gli altri utenti con cui si trova a operare con PGP. Si supponga che il key ring di A contenga una chiave pubblica attribuita a B ma che, in realtà, appartiene a C. Questo può accadere se, per esempio, A preleva una chiave da un messaggio in un newsgroup che è stato usato da B per inviare la chiave pubblica ma che è stato violato da C. Il risultato è che ora esistono due problemi. Innanzitutto C può inviare messaggi ad A e falsificare la firma di B in modo che A accetti il messaggio come proveniente da B. In secondo luogo, ogni messaggio crittografato inviato da A a B potrà essere letto anche da C.

Si possono adottare vari approcci per ridurre i rischi che il public-key ring di un utente contenga chiavi false. Si supponga che A voglia ottenere da B una chiave pubblica affidabile. Ecco gli approcci utilizzabili.

1. Richiedere fisicamente la chiave a B. B potrebbe memorizzare la propria chiave pubblica (PU_b) su un disco floppy e consegnarlo ad A. A questo punto A potrebbe caricare la chiave nel proprio sistema dal disco floppy. Si tratta di un metodo molto sicuro ma dagli ovvi limiti pratici.
2. Verificare la chiave telefonicamente. Se A è in grado di riconoscere B al telefono, A potrà chiamare B e chiedergli di dettare la chiave in formato radix-64. Ecco un'alternativa più pratica: B potrebbe trasmettere la propria chiave ad A in un messaggio di posta elettronica. A potrebbe fare in modo che PGP generi un codice SHA-1 a 160 bit della chiave e lo visualizzi in formato esadecimale; questa viene chiamata "l'impronta digitale" della chiave. A questo punto, A potrebbe chiamare B e chiedergli di dettare per telefono l'impronta digitale. Se le due impronte digitali coincidono, la chiave è verificata.
3. Ottenere la chiave pubblica di B tramite un intermediario fidato D. Per questo scopo, D crea un certificato firmato. Il certificato include la chiave pubblica di B, il momento di creazione della chiave e il periodo di validità della chiave. D genera un codice digest SHA-1 di questo certificato, ne esegue la crittografia con la propria chiave privata e poi allega al certificato la firma. Poiché solo D può aver creato la firma, nessun altro potrebbe creare una chiave pubblica falsa e sostenere che sia stata firmata da D. Il certificato firmato può essere inviato direttamente ad A da B o da D o può essere inviato a un servizio pubblico come un newsgroup.
4. Ottenere la chiave pubblica di B da un'autorità di certificazione fidata. Anche in questo caso viene creato un certificato della chiave pubblica firmato dall'autorità. A questo punto A potrebbe accedere all'autorità, fornire il nome-utente di B e ricevere un certificato firmato.

Per i casi 3 e 4, A avrà già una copia della chiave pubblica dell'intermediario e confiderà nella sua validità. Alla fine, è compito di A assegnare un livello di fiducia all'intermediario.

L'uso dei trust

Sebbene PGP non includa alcuna specifica per definire delle autorità di certificazione o per stabilire il livello di fiducia (*trust*), fornisce un mezzo comodo per utilizzare i trust, associandoli alle chiavi pubbliche.

La struttura di base è la seguente. Come si è detto in precedenza, ogni voce del public-key ring è un certificato della chiave pubblica. A ognuna di queste voci è associato un **campo di legittimità della chiave** che indica quanto PGP può fidarsi della validità di questa chiave pubblica per questo utente; più elevato è il livello di fiducia, più forte sarà l'associazione di questo codice utente alla sua chiave. Questo campo viene calcolato da PGP. Associate alla voce vi sono anche zero o più firme che il possessore del public-key ring ha raccolto e che firmano questo certificato. A sua volta a ciascuna firma è associato un **campo di fiducia della firma** che indica il grado di fiducia dell'utente PGP nei confronti del firmatario. Il campo di legittimità della chiave deriva dalla serie di campi di fiducia delle firme di questa voce. Infine ciascuna voce definisce una chiave pubblica associata a un determinato proprietario e un campo di fiducia del proprietario che indica il grado di fiducia della chiave pubblica per firmare altri certificati di chiave pubblica; questo livello di fiducia viene assegnato dall'utente. In pratica i campi di fiducia della firma sono come delle copie del campo di fiducia del proprietario presenti in altre voci.

I tre campi menzionati nel paragrafo precedente sono contenuti in una struttura chiamata byte dei flag di trust. Il contenuto di questo flag trust per ognuno dei tre utilizzi è rappresentato nella Tabella 15.2. Si supponga di considerare il public-key ring dell'utente A. Si può descrivere il funzionamento dell'elaborazione dei trust nel seguente modo.

1. Quando A inserisce una nuova chiave pubblica nel public-key ring, PGP deve assegnare un valore al flag di trust relativo al proprietario di questa chiave pubblica. Se il proprietario è A (e pertanto questa chiave pubblica compare anche nel private-key ring), gli viene automaticamente assegnato il valore di trust massimo (ultimate trust). Altrimenti, PGP chiede ad A di indicare il livello di fiducia da assegnare al possessore di questa chiave. L'utente può specificare che questo proprietario è sconosciuto, non fidato, parzialmente fidato o completamente fidato.
2. Quando si introduce la nuova chiave pubblica, è possibile allegare una o più firme. Successivamente sarà possibile aggiungere ulteriori firme. Quando una firma viene inserita nella voce, PGP verifica nel public-key ring se l'autore di questa firma è fra i possessori noti di chiavi pubbliche. In questo caso, al campo SIGTRUST di questa

firma viene assegnato il valore OWNERTRUST di questo proprietario. In caso contrario viene assegnato il valore *unknown user* (utente sconosciuto).

3. Il valore del campo di legittimità della chiave viene calcolato sulla base dei campi di trust della firma presenti in questa voce. Se almeno una delle firme ha un livello di trust *ultimate*, allora il valore di legittimità della chiave sarà *complete*. Altrimenti PGP calcola una somma pesata dei valori di trust. Viene assegnato un peso pari a $1/X$ alle firme già verificate (*always trusted*) e $1/Y$ alle firme che vengono normalmente verificate (*usually trusted*) dove X e Y sono parametri configurabili dall'utente. Quando il totale dei pesi di coloro che hanno introdotto una combinazione chiave/codice utente raggiunge 1, l'associazione viene considerata fidata e il valore di legittimità della chiave viene impostato a complete. Pertanto, in assenza di un livello di trust ultimate, sono necessarie almeno X firme always trusted o Y usually trusted o una combinazione di esse.

Tabella 15.2 Contenuto dei byte dei flag di trust.

(A) Trust assegnato al possessore della chiave pubblica (specificato dopo il pacchetto della chiave; definito dall'utente)	**(B) Livello di trust assegnato a coppie chiave pubblica/Codice utente (specificato dopo il pacchetto del codice utente; calcolato da PGP)**	**(C) Livello di trust assegnato alla firma (compare dopo il pacchetto della firma; copia del OWNERTRUST per il firmatario).**
Campo OWNERTRUST - Trust non definito - Utente sconosciuto - Normalmente non fidato per la firma di altre chiavi - Normalmente fidato per la firma di altre chiavi - Sempre fidato per la firma di altre chiavi - Questa chiave è presente nel private-key ring (livello di trust ultimate) Bit BUCKSTOP - Impostato se questa chiave compare nel private-key ring.	Campo KEYLEGIT - Trust sconosciuto o non definito - Proprietà della chiave non fidata - Fiducia marginale nella proprietà della chiave. - Fiducia completa nella proprietà della chiave Bit WARNONLY - Impostato se l'utente vuole solo essere avvertito quando per la crittografia viene utilizzata una chiave che non è completamente convalidata.	Campo SIGTRUST - Trust non definito - Utente sconosciuto - Normalmente non fidato per la firma di altre chiavi - Normalmente fidato per la firma di altre chiavi - Sempre fidato per la firma di altre chiavi - Questa chiave è presente nel private-key ring (livello di trust ultimate) Bit CONTIG - Impostato se la firma conduce a un percorso di certificazione contiguo fidato che rimanda al possessore fidato del key ring.

Periodicamente, PGP elabora il public-key ring per verificarne la coerenza. Si tratta essenzialmente di un processo top-down. Per ciascun campo OWNERTRUST, PGP esegue la scansione del ring alla ricerca di tutte le firme prodotte da tale proprietario e aggiorna il campo SIGTRUST assegnandogli il contenuto del campo OWNERTRUST. Questa operazione inizia con le chiavi per le quali vi è un livello di trust ultimate. Poi vengono calcolati tutti i campi KEYLEGIT sulla base delle firme allegate.

La Figura 15.7 fornisce un esempio delle relazioni che legano la fiducia della firma e la legittimità della chiave.[2] La figura mostra la struttura di un public-key ring. L'utente ha acquisito un certo numero di chiavi pubbliche, alcune direttamente dai proprietari e altre da terze parti, per esempio un server di chiavi.

Il nodo più in alto fa riferimento alla voce del public-key ring corrispondente a questo utente. Questa chiave è legittima e il valore di OWNERTRUST è ultimate trust. Ogni altro nodo del key ring ha il valore OWNERTRUST indefinito, a meno che l'utente abbia assegnato altri valori. In questo esempio, questo utente ha specificato che si fida sempre dei seguenti utenti per la firma di altre chiavi: D, E , F, L, e che si fida parzialmente degli utenti A e B.

Pertanto l'ombreggiatura (o la mancanza) dei nodi nella Figura 15.7 indica il livello di fiducia assegnato dall'utente. La struttura ad albero indica quali chiavi sono state firmate da quali altri utenti. Se una chiave è firmata da un utente la cui chiave è già presente nel key ring, la freccia congiunge la chiave firmata al firmatario. Se la chiave è firmata da un utente la cui chiave non è presente nel key ring, la freccia congiunge la chiave firmata al punto interrogativo per indicare che il firmatario è sconosciuto a questo utente.

La Figura 15.7 illustra vari punti.

1. Si noti che tutte le chiavi i cui proprietari sono completamente o parzialmente fidati per questo utente sono state firmate da questo utente, con l'eccezione del nodo L. La firma dell'utente non è sempre necessaria, come indica la presenza del nodo L, ma in pratica la maggior parte degli utenti firmerà probabilmente le chiavi della maggior parte dei proprietari di cui si fida. Pertanto, per esempio, anche se la chiave di E è già firmata dall'intermediario fidato F, l'utente sceglie di firmarla direttamente.
2. Si suppone che due firme parzialmente fidate siano sufficienti per certificare una chiave. Pertanto, la chiave per l'utente H è considerata legittima da PGP poiché è firmata da A e B, entrambi parzialmente fidati.
3. Una chiave può essere determinata come legittima poiché è firmata da un firmatario completamente fidato o da due firmatari parzialmente fidati ma il suo proprietario può non essere considerato fidato per firmare altre chiavi. Per esempio, la chiave di N è legittima poiché è firmata da E di cui questo utente si fida ma N non è fidato per firmare altre chiavi poiché questo utente non ha assegnato a N questo livello di fiducia. Pertanto, sebbene la chiave di R sia firmata da N, PGP non considera legittima la chiave di R. Questa situazione ha perfettamente senso. Se si vuole inviare un messaggio privato a qualcuno, non è necessario fidarsi di questa persona. È solo necessario assicurarsi di avere la chiave pubblica corretta per questa persona.
4. La Figura 15.7 mostra anche un esempio di nodo (S) isolato, od "orfano" con due firme sconosciute. Tale chiave potrebbe essere stata ottenuta da un server di chiavi. PGP non può presupporre che questa chiave sia legittima semplicemente perché ottenuta da un server fidato. L'utente deve dichiarare che la chiave è legittima o firmandola o indicando a PGP la sua volontà di considerare completamente fidato uno dei firmatari della chiave.

Un'ultima considerazione: in precedenza si è detto che a una chiave pubblica di un ring di chiavi pubbliche possono essere associati più codici utente. Questo può verificarsi perché

[2] Figura fornita all'autore da Phil Zimmermann.

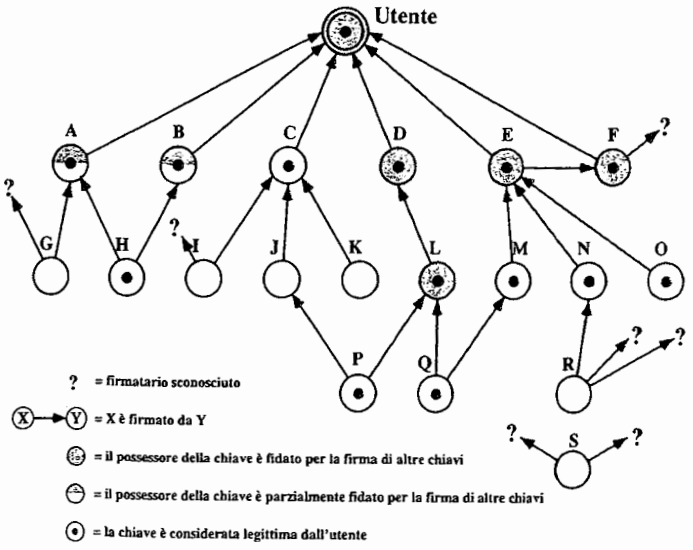


Figura 15.7 Esempio di modello di fiducia in PGP

una persona ha cambiato nome o è stata introdotta tramite firma sotto più nomi, per esempio indicando indirizzi di posta elettronica differenti. Pertanto si può considerare la chiave pubblica come la radice di un albero. A una chiave pubblica sono associati più codici utente con un certo numero di firme sotto ciascun codice. L'associazione fra un determinato codice utente e una chiave dipende dalle firme associate a tale codice e a tale chiave mentre il livello di fiducia su questa chiave (per l'impiego nella firma di altre chiavi) è funzione di tutte le firme dipendenti.

Revoca delle chiavi pubbliche

Un utente può voler revocare la propria chiave pubblica perché violata o semplicemente per evitare di utilizzare la stessa chiave per un lungo periodo. Si noti che la violazione di una chiave richiederebbe che un avversario abbia in qualche modo ottenuto una copia della chiave privata non crittografata o che abbia ottenuto sia la chiave privata dal ring delle chiavi private che la frase segreta.

La convenzione per la revoca di una chiave pubblica è che il proprietario emetta un certificato di revoca della chiave, firmato dal proprietario stesso. Il certificato ha lo stesso formato del normale certificato di firma ma include un indicatore del fatto che si tratta di un certificato di revoca dell'uso di questa chiave pubblica. Si noti che per firmare un certificato che revoca una chiave pubblica deve essere utilizzata la chiave privata corrispondente. Il proprietario dovrebbe quindi tentare di distribuire questo certificato il più ampliamente e il più rapidamente possibile per consentire ai potenziali corrispondenti di aggiornare i propri ring delle chiavi pubbliche.

Si noti che questo certificato può essere emesso anche da un estraneo che abbia violato la chiave privata del proprietario. Tuttavia questo impedirebbe all'estraneo oltre che al legittimo proprietario di utilizzare la chiave pubblica e pertanto sembra una minaccia meno probabile rispetto all'utilizzo fraudolento di una chiave privata rubata.

15.2 S/MIME

S/MIME (Secure/Multipurpose Internet Mail Extension) è un'estensione dedicata alla sicurezza dello standard del formatto della posta elettronica per Internet MIME, basato sulla tecnologia RSA Data Security. Sebbene sia PGP che S/MIME stiano seguendo il processo di standardizzazione IETF, sembra probabile che S/MIME emergerà quale standard per l'utilizzo commerciale e aziendale mentre PGP rimarrà la scelta per la sicurezza personale della posta elettronica per molti utenti. S/MIME è definito in vari documenti, i più importanti dei quali sono i documenti RFC 3369, 3370 e 3851.

Per comprendere S/MIME occorre innanzitutto conoscere in generale il formato su cui si basa, ovvero MIME. Ma per conoscere il significato di MIME occorre risalire allo standard tradizionale del formatto della posta elettronica, RFC 822, tuttora comunemente utilizzato. Di conseguenza, questa parte del capitolo fornirà innanzitutto un'introduzione a questi standard antecedenti per poi trattare S/MIME.

Il documento RFC 822

Il documento RFC 822 definisce un formato per i messaggi di testo inviati utilizzando la posta elettronica. RFC 822 è stato per lungo tempo lo standard per i messaggi di posta elettronica testuali in Internet e rimane tuttora in uso. Nel contesto RFC 822, i messaggi sono costituiti da una busta e da un contenuto. La busta contiene tutte le informazioni necessarie per la trasmissione e la consegna. Il contenuto è l'oggetto da consegnare al destinatario. Lo standard RFC 822 riguarda solo il contenuto. Tuttavia questo standard include un insieme di campi di intestazione che possono essere utilizzati dal sistema di posta elettronica per creare la "busta" e lo standard ha lo scopo di facilitare l'acquisizione di tali informazioni da parte dei programmi.

La struttura generale di un messaggio conforme al documento RFC 822 è molto semplice. Un messaggio è costituito da un certo numero di righe di intestazione seguite da testo libero (il *corpo* del messaggio). L'intestazione è separata dal corpo da una riga vuota. In altre parole un messaggio è un testo ASCII e tutte le righe fino alla prima riga vuota sono l'intestazione utilizzata dall'agente utente per il sistema di posta elettronica.

Una riga di intestazione è normalmente costituita da una parola riservata, seguita dal segno di due punti e dall'argomento; il formato consente di spezzare le righe più lunghe in più righe. Le parole riservate più utilizzate sono *From*, *To*, *Subject* e *Date*. Ecco un esempio di messaggio:

```
Date: Tue, 16 Jan 1998 10:37:17 (EST)
From: "William Stallings" <ws@shore.net>
```

```
Subject: The Syntax in RFC 822
To: Smith@Other-host.com
Cc: Jones@Yet-Another-Host.com

Hello. This section begins the actual message body, which is
delimited from the message heading by a blank line.
```

Un altro campo comunemente utilizzato nelle intestazioni RFC 822 è *Message-ID* che contiene un identificatore univoco associato al messaggio.

MIME (Multipurpose Internet Mail Extensions)

MIME è un'estensione della struttura RFC 822 che ha lo scopo di risolvere alcuni dei problemi e dei limiti del protocollo SMTP (Simple Mail Transfer Protocol), di altri protocolli di trasferimento della posta elettronica e di RFC 822. [RODR02] elenca i seguenti problemi dello schema SMTP/RFC 822.

1. SMTP non è in grado di trasmettere file eseguibili o altri oggetti binari. Esistono vari sistemi per convertire i file binari in un formato testuale che possa essere utilizzato dai sistemi di posta elettronica SMTP, fra cui il noto schema di Unix UUencode/UUdecode. Tuttavia nessuno di questi è uno standard, nemmeno uno standard "di-fatto".
2. SMTP non può trasmettere dati testuali comprendenti caratteri internazionali poiché questi sono rappresentati da codici a 8 bit con valori superiori a 128 e SMTP può utilizzare solo caratteri ASCII a 7 bit.
3. I server SMTP possono rifiutare i messaggi di posta elettronica che superano determinate dimensioni.
4. I gateway SMTP che effettuano la traduzione fra la codifica ASCII e la codifica EBCDIC non utilizzano forme di conversione coerenti, provocando problemi di traduzione.
5. I gateway SMTP verso reti di posta elettronica X.400 non sono in grado di gestire i dati non testuali inclusi nei messaggi X.400.
6. Alcune implementazioni di SMTP non aderiscono completamente allo standard definito dal documento RFC 821. Ciò provoca vari problemi tra i quali i seguenti:
   - Cancellazione, aggiunta o riposizionamento dei codici Carriage Return e Line Feed.
   - Troncamento o invio a capo delle righe più lunghe di 76 caratteri.
   - Rimozione dello spazio a inizio riga (spazio e carattere di tabulazione).
   - Aggiunta di caratteri alle righe per raggiungere la stessa lunghezza.
   - Conversione dei caratteri di tabulazione in sequenze di spazi.

MIME ha lo scopo di risolvere questi problemi in un modo compatibile con le implementazioni RFC 822 esistenti. Le specifiche sono contenute nei documenti RFC da 2045 a 2049.

Panoramica

Le specifiche MIME includono i seguenti elementi.

1. Vengono definiti cinque nuovi campi di intestazione dei messaggi che possono essere inclusi nell'intestazione RFC 822. Questi campi forniscono informazioni sul corpo del messaggio.
2. Sono definiti nuovi formati dei contenuti, standardizzando le rappresentazioni che supportano la posta elettronica multimediale.
3. Sono definite delle codifiche di trasferimento che consentono la conversione di ogni formato in una forma protetta da modifiche da parte del sistema di posta elettronica.

Verranno ora introdotti i cinque campi di intestazione dei messaggi. Successivamente si parlerà del formato dei contenuti e delle codifiche di trasferimento. Ecco i cinque campi di intestazione definiti da MIME.

- **MIME-Version:** deve contenere il valore 1.0. Ciò indica che il messaggio è conforme ai documenti RFC 2045 e 2046.
- **Content-Type:** descrive i dati contenuti nel corpo con un livello di dettaglio tale da consentire all'agente di ricezione di richiamare l'agente o il meccanismo appropriato per presentare i dati all'utente o comunque per gestire i dati in modo appropriato.
- **Content-Transfer-Encoding:** indica il tipo di trasformazione che è stata utilizzata per rappresentare il corpo del messaggio in modo accettabile per il sistema di trasporto della posta elettronica.
- **Content-ID:** identifica univocamente le entità MIME in più contesti.
- **Content-Description:** una descrizione testuale dell'oggetto con il suo corpo; è utile quando l'oggetto non è leggibile (come nel caso dei dati audio).

Questi campi possono essere dunque presenti in una normale intestazione RFC 822. Un'implementazione compatibile deve supportare i campi MIME-Version, Content-Type e Content-Transfer-Encoding; i campi Content-ID e Content-Description sono opzionali e possono essere ignorati dall'implementazione del destinatario.

I tipi di contenuti MIME

Il nucleo centrale delle specifiche MIME riguarda la definizione dei vari tipi di contenuti. Questo riflette la necessità di fornire uno standard per il trattamento dell'ampia varietà di informazioni possibili in un ambiente multimediale.

La Tabella 15.3 elenca i tipi di contenuti specificati nel documento RFC 2046. Si tratta di 7 tipi principali e 15 sottotipi. In generale un tipo dichiara il tipo generico dei dati mentre il sottotipo specifica il formato dei dati.

Per il tipo **text**, non è necessario alcun software particolare per ottenere il testo, tranne il supporto del set di caratteri indicato. Il principale sottotipo è *plain text* che indica semplicemente una stringa di caratteri ASCII o ISO 8859. Il sottotipo *enriched* offre una maggiore flessibilità di formattazione.

Il tipo **multipart** indica che il corpo contiene più parti indipendenti. Il campo di intestazione Content-Type include un parametro, boundary, che definisce il delimitatore fra le parti del corpo. Questo parametro non deve comparire in nessuna parte del messaggio. Ogni boundary inizia una nuova riga ed è costituito da due trattini seguiti dal valore del delimitatore. Il boundary finale, che indica la fine dell'ultima parte, ha anche un suffisso di due trattini. All'interno di ciascuna parte vi può essere una normale intestazione MIME opzionale.

Tabella 15.3 I tipi di contenuti MIME.

Tipo	Sottotipo	Descrizione
Text	Plain	Testo non formattato; può essere in formato ASCII o ISO 8859.
	Enriched	Offre una maggiore flessibilità di formattazione.
Multiport	Mixed	Le varie parti sono indipendenti ma devono essere trasmesse insieme. Devono essere presentate al destinatario nell'ordine in cui compaiono nel messaggio di posta elettronica.
	Parallel	Differisce dal sottotipo Mixed per il fatto che non è definito alcun ordine per la consegna delle parti al destinatario.
	Alternative	Le varie parti sono versioni alternative della stessa informazione. Sono ordinate in base alla loro fedeltà all'originale e il sistema di posta elettronica del destinatario dovrà visualizzare all'utente la versione "migliore".
	Digest	Simile a Mixed ma il tipo/sottotipo di ciascuna parte è message/rfc822.
Message	rfc822	Il corpo è un messaggio incapsulato conforme al documento RFC 822.
	Partial	Usato per consentire la frammentazione di messaggi di grandi dimensioni in modo trasparente al destinatario.
	External-body	Contiene un puntatore o un oggetto che si trova altrove.
Image	jpeg	Immagine in formato JPEG, codifica JFIF.
	gif	Immagine in formato GIF.
Video	mpeg	Formato MPEG
Audio	Basic	Audio mono-canale a 8 bit ISDN con codifica mu-law a una frequenza di campionamento di 8 kHz.
Application	PostScript	Messaggio Adobe PostScript
	Octet-stream	Dati binari generici costituiti da byte di 8 bit.

Ecco un semplice esempio di messaggio multipart contenente due parti, entrambe costituite da semplice testo (tratto da RFC 2046):

```
From: Nathaniel Borenstein <nsb@bellcore.com>
To: Ned Freed <ned@innosoft.com>
Subject: Sample message
MIME-Version: 1.0
Content-type: multipart/mixed; boundary="simple boundary"

This is the preamble. It is to be ignored, though it is a handy
place for mail composers to include an explanatory note to non-
MIME conformant readers.--simple boundary

This is implicitly typed plain ASCII text. It does NOT end with a
linebreak.--simple boundary
```



```
Content-type: text/plain; charset=us-ascii
This is explicitly typed plain ASCII text. It DOES end with a
linebreak.

--simple boundary--
This is the epilogue. It is also to be ignored.
```

Il tipo multipart prevede quattro sottotipi che utilizzano la stessa sintassi generale. Il sottotipo **multipart/mixed** viene utilizzato quando il corpo è formato da più parti indipendenti che devono essere raggruppate in un ordine ben preciso. Nel sottotipo **multipart/parallel** l'ordine delle parti non è importante. Se il sistema del destinatario è appropriato, le parti possono essere presentate in parallelo. Per esempio, un'immagine o un testo possono essere affiancati da un commento vocale che viene riprodotto durante la visualizzazione del testo o dell'immagine.

Il sottotipo **multipart/alternative** prevede che le varie parti siano rappresentazioni differenti della stessa informazione. Ecco un esempio:

```
From: Nathaniel Borenstein <nsb@bellcore.com>
To: Ned Freed <ned@innosoft.com>
Subject: Formatted text mail
MIME-Version: 1.0
Content-Type: multipart/alternative; boundary=boundary42
--boundary42

Content-Type: text/plain; charset=us-ascii

... plain text version of message goes here ....

--boundary42
Content-Type: text/enriched

.... RFC 1896 text/enriched version of same message goes here ...
--boundary42--
```

In questo sottotipo, le varie parti del corpo sono ordinate in base alla preferenza. In questo esempio, se il sistema di destinazione è in grado di visualizzare il messaggio in formato text/enriched, lo farà, altrimenti utilizzerà il formato plain text.

Il sottotipo **multipart/digest** viene utilizzato quando le varie parti del corpo sono interpretate come messaggi RFC 822 con intestazioni. Questo sottotipo consente di costruire un messaggio le cui parti sono a loro volta messaggi. Per esempio, il moderatore di un gruppo potrebbe raccogliere i messaggi di posta elettronica dei partecipanti, riunirli in un messaggio e poi inviarli in un unico messaggio MIME.

Il tipo **message** fornisce varie importanti funzionalità in MIME. Il sottotipo **message/rfc822** indica che il corpo è un messaggio completo comprendente l'intestazione e il cor-

po. Nonostante il nome di questo sottotipo, il messaggio incapsulato può anche non essere un messaggio RFC 822 ma un qualsiasi messaggio MIME.

Il sottotipo **message/partial** consente di frammentare un messaggio voluminoso in più parti che devono poi essere riassemblate alla destinazione. Per questo sottotipo vengono specificati tre parametri nel campo Content-Type: Message/Partial: un codice *ID* comune a tutti i frammenti dello stesso messaggio, un *numero sequenziale* univoco per ciascun frammento e il numero *totale* di frammenti.

Il sottotipo **message/external-body** indica che i dati che devono essere trasferiti in questo messaggio non sono contenuti nel corpo. Al contrario il corpo contiene le informazioni necessarie per accedere ai dati. Come per altri tipi di messaggi, il sottotipo message/external-body ha un'intestazione esterna e un messaggio incapsulato con la propria intestazione. L'unico campo necessario nell'intestazione esterna è Content-Type che identifica il sottotipo message/external-body. L'intestazione interna è l'intestazione del messaggio incapsulato. Il campo Content-Type dell'intestazione esterna deve includere un parametro access-type che indica il metodo di accesso, per esempio FTP (File Transfer Protocol).

Il tipo **application** fa riferimento ad altri tipi di dati, normalmente interpretati come dati binari o comunque informazioni che devono essere elaborate da un'applicazione.

Codifiche di trasferimento MIME

L'altra componente principale di MIME, oltre alla specifica del tipo di contenuto, riguarda la definizione delle codifiche di trasferimento del corpo del messaggio. L'obiettivo è quello di fornire un metodo di consegna affidabile in un'ampia gamma di ambienti.

Lo standard MIME definisce due metodi di codifica dei dati. Il campo Content-Transfer-Encoding può assumere 6 valori, elencati nella Tabella 15.4. Tuttavia tre di questi valori (7 bit, 8 bit e binary) indicano che non è stata effettuata alcuna codifica ma forniscono informazioni riguardanti la natura dei dati. Per il trasferimento SMTP è sicuro utilizzare il formato 7 bit. I formati 8 bit e binary possono essere utilizzati in altri contesti di trasporto della posta elettronica. Un altro valore del campo Content-Transfer-Encoding è x-token che indica che viene utilizzato un altro schema di codifica per il quale deve essere fornito un nome. Può trattarsi di uno schema di codifica specifico di un produttore o di un'applicazione. I due

Tabella 15.4 Codifiche di trasferimento MIME.

7 bit	I dati sono codificati con righe brevi di caratteri ASCII.
8 bit	Le righe sono brevi ma vi possono essere caratteri non-ASCII (ovvero byte con il bit più significativo impostato.
binary	Può contenere caratteri non-ASCII e le righe possono essere più lunghe del limite previsto da SMTP.
quoted-printable	Codifica i dati in modo tale che, se si tratta prevolementemente di testo ASCII, la forma codificata rimanga riconoscibile dagli utenti.
base 64	Codifica i dati associando blocchi di 6 bit di input o blocchi di 8 bit di output, i quali sono caratteri ASCII stampabili.
x-token	Nome di una codifica non standard.

schemi di codifica realmente definiti sono quoted-printable e base64. Vengono definiti due schemi per offrire una scelta fra una tecnica di trasferimento fondamentalmente orientata alla lettura umana e una sicura per tutti i tipi di dati e ragionevolmente compatta.

La codifica di trasferimento **quoted-printable** è utile quando i dati sono costituiti fondamentalmente da ottetti corrispondenti a caratteri ASCII stampabili. In pratica trasforma i caratteri non sicuri nella rappresentazione esadecimale del loro codice e introduce delle interruzioni di riga reversibili (soft) per limitare le righe dei messaggi a un massimo di 76 caratteri.

La **codifica di trasferimento base64**, chiamata anche radix-64, è comune per la codifica di dati binari arbitrari in modo che non vengano modificati dai programmi di trasporto della posta elettronica. Viene utilizzata anche in PGP ed è descritta nell'Appendice 15.B.

Un esempio multipart

La Figura 15.8, tratta dal documento RFC 2045, rappresenta la struttura di un complesso messaggio multipart. Il messaggio comprende cinque parti da visualizzare sequenzialmente: due parti introduttive di semplice testo, un messaggio multipart incorporato, una parte richtext e un messaggio di testo incapsulato di chiusura in un set di caratteri non ASCII. Il messaggio multipart incorporato comprende due parti da visualizzare in parallelo: un'immagine e un frammento audio.

Forma canonica

Un concetto importante in MIME e S/MIME è la forma canonica. La forma canonica è un formato, adatto al contenuto, che è standardizzato per l'utilizzo in vari sistemi. Il suo opposto è la forma nativa che è un formato che può essere specifico di un particolare sistema. La Tabella 15.5, tratta dal documento RFC 2049, dovrebbe aiutare a chiarire questo argomento.

Tabella 15.5 Forma nativa e forma canonica.

Forma nativa	Il corpo da trasmettere viene creata nel formato nativo del sistema. Viene utilizzato il set di caratteri nativo e, ove appropriato, vengono utilizzate anche le convenzioni locali riguardanti i codici di fine riga. Il corpo può essere un file di testo stile Unix, un'immagine raster Sun, un file indicizzato VMS, dati audio in un formato dipendente del sistema o qualsiasi altra cosa che corrisponda al modello locale di rappresentazione di determinate informazioni. Fondamentalmente i dati vengono creati nella forma "nativa" che corrisponde al tipo di media specificato.
Forma canonico	L'intero corpo, comprese le informazioni "fuori banda" come la lunghezza dei record ed eventuali informazioni sugli attributi dei file, viene convertito in una forma canonica universale. Il tipo di media specifico del corpo e gli attributi ad esso associati indicono la forma canonico utilizzata. La conversione nella forma canonica corretta può prevedere la conversione del set di caratteri, la trasformazione dei dati audio, la compressione e varie altre operazioni specifiche dei vari tipi di media. Se viene eseguita una conversione del set di caratteri, occorre prestare attenzione alla semantico del tipo di media, che può avere forti implicazioni per la conversione del set di caratteri (per esempio rispetto ai caratteri sintatticamente significativi in un sottotipo di Text diverso da "plain").

```
MIME-Version: 1.0
From: Nathaniel Borenstein <nsb@bellcore.com>
To: Ned Freed <ned@innosoft.com>
Subject: A multipart example
Content-Type: multipart/mixed;
    boundary=unique-boundary-1

This is the preamble area of a multipart message. Mail readers that understand multipart format should ignore
this preamble. If you are reading this text, you might want to consider changing to a mail reader that understands
how to properly display multipart messages.

--unique-boundary-1

   ...Some text appears here...
[Note that the preceding blank line means no header fields were given and this is text, with charset US ASCII.
It could have been done with explicit typing as in the next part.]

--unique-boundary-1
Content-type: text/plain; charset=US-ASCII

This could have been part of the previous part, but illustrates explicit versus implicit typing of body parts.

--unique-boundary-1
Content-Type: multipart/parallel;  boundary=unique-boundary-2

--unique-boundary-2
Content-Type: audio/basic
Content-Transfer-Encoding: base64

   ... base64-encoded 8000 Hz single-channel mu-law-format audio data goes here....

--unique-boundary-2
Content-Type: image/jpeg
Content-Transfer-Encoding: base64

   ... base64-encoded image data goes here....

--unique-boundary-2--

--unique-boundary-1
Content-type: text/enriched

This is <bold><italic>richtext.</italic></bold> <smaller>as defined in RFC 1896</smaller>

Isn't it <bigger><bigger>cool?</bigger></bigger>

--unique-boundary-1
Content-Type: message/rfc822

From: (mailbox in US-ASCII)
To: (address in US-ASCII)
Subject: (subject in US-ASCII)
Content-Type: Text/plain; charset=ISO-8859-1
Content-Transfer-Encoding: Quoted-printable

   ... Additional text in ISO-8859-1 goes here ...

--unique-boundary-1--
```

Figura 15.8 Un esempio di messaggio MIME multipart.

Le funzionalità di S/MIME

In termini di funzionalità generali, S/MIME è molto simile a PGP. Entrambi offrono la possibilità di firmare e/o crittografare i messaggi. In questa parte del capitolo verranno riepilogate brevemente le funzionalità di S/MIME. Poi si osserveranno più in dettaglio le funzionalità esaminando i formati dei messaggi e la preparazione dei messaggi.

Funzioni

S/MIME fornisce le seguenti funzioni.

- **Dati enveloped (imbustati):** è costituito da contenuti crittografati di qualsiasi tipo e chiavi di crittografia crittografate per uno o più destinatari.
- **Dati firmati:** la firma digitale viene costituita crittografando con la chiave privata del firmatario il codice message digest del contenuto da firmare. Il contenuto e la firma vengono poi codificati base64. Un messaggio a dati firmati può essere visualizzato solo da un destinatario dotato di funzionalità S/MIME.
- **Dati in chiaro firmati:** come per i dati firmati, viene creata una firma digitale del contenuto del messaggio. Tuttavia, in questo caso, solo la firma digitale viene codificata base64. Come risultato, i destinatari senza funzionalità S/MIME potranno visualizzare il contenuto del messaggio sebbene non possano verificarne la firma.
- **Dati firmati ed enveloped:** le entità firmate ed enveloped possono essere nidificate in modo che i dati crittografati possano essere firmati e i dati firmati (o firmati in chiaro) possano essere crittografati.

Algoritmi crittografici

La Tabella 15.6 riepiloga gli algoritmi crittografici utilizzati in S/MIME. S/MIME usa la seguente terminologia, tratta dal documento RFC 2119, per specificare il livello dei requisiti:

- **DEVE:** per poter essere conforme alle specifiche, un'implementazione deve includere questa funzionalità.
- **DOVREBBE:** vi potrebbero essere motivi validi in particolari circostanze per ignorare questa funzionalità ma è consigliabile che l'implementazione la includa.

S/MIME incorpora tre algoritmi a chiave pubblica. Lo standard DSS (Digital Signature Standard), di cui si è parlato nel Capitolo 13, è l'algoritmo preferenziale per la firma digitale. S/MIME indica Diffie-Hellman come algoritmo preferenziale per la crittografia delle chiavi di sessione; in realtà, S/MIME usa una variante di Diffie-Hellman che fornisce la crittografia/decrittografia, chiamata ElGamal (vedere il Problema 10.6). In alternativa, sia per le firme che per la crittografia della chiave di sessione può essere utilizzato RSA, descritto nel Capitolo 9. Si tratta degli stessi algoritmi impiegati in PGP, che offrono un elevato livello di sicurezza. Per la funzione hash utilizzata per creare la firma digitale, le specifiche richiedono l'impiego di SHA-1 a 160 bit ma consigliano che l'agente di ricezione supporti MD5 a 128 bit per garantire la compatibilità all'indietro con le versioni precedenti di S/MIME. Come si è detto nel Capitolo 12, esistono dei dubbi giustificati sulla sicurezza di MD5: dunque SHA-1 rappresenta chiaramente l'alternativa preferibile.

Tabella 15.6 Algoritmi crittagrafici usati in S/MIME.

Funzione	Requisito
Creazione del codice message digest da utilizzare nella creazione della firma digitale.	*Deve* supportare SHA-1. L'agente di ricezione *dovrebbe* supportare MD5 per la compatibilità all'indietro.
Crittografia del messoge digest per creare la firma digitale.	Gli agenti di invio e ricezione *devono* supportare DSS. Gli agenti di invio *dovrebbero* supportare la crittografia RSA. Gli agenti di ricezione *dovrebbero* supportare la verifica delle firme RSA con chiavi comprese fra 512 e 1024 bit.
Crittografia della chiove di sessione per la trasmissione insieme al messoggio.	Gli agenti di invio e ricezione *dovrebbero* supportare Diffie-Hellman. Gli agenti di invio e ricezione *devono* supportare la crittografia RSA con chiavi di dimensioni comprese fra 512 e 1024 bit.
Crittografia del messaggio per la trasmissione con una chiave di sessione monouso.	Gli agenti di invio e ricezione *devono* supportore lo crittografia con 3DES. Gli agenti di invio *dovrebbero* supportare lo crittografia con AES. Gli agenti di invio *dovrebbero* supportare la crittografia con RC2/40
Creazione del codice di autenticazione del messaggio.	Gli agenti di ricezione *devono* supportare HMAC con SHA-1. Gli agenti di ricezione *dovrebbero* supportare HMAC con HSA-1.

Per la crittografia dei messaggi, è consigliato l'impiego di DES a 3 chiavi (tripleDES) ma le implementazioni compatibili devono supportare RC2 a 40 bit. Quest'ultimo è un algoritmo di crittografia molto debole che però supera i controlli di esportazione dagli Stati Uniti.

Le specifiche S/MIME includono una discussione della procedura per decidere quali algoritmi di crittografia dei contenuti utilizzare. Essenzialmente, un agente di trasmissione deve prendere due decisioni. Innanzitutto deve determinare se l'agente di ricezione è in grado di eseguire la decrittografia utilizzando un determinato algoritmo. In secondo luogo, se l'agente di ricezione è in grado di accettare solo contenuti crittografati in modo debole, l'agente di trasmissione deve decidere se è accettabile inviare le informazioni utilizzando la crittografia debole. Per supportare questo processo di decisione, un agente di trasmissione può annunciare le proprie capacità di decrittografia in ordine di preferenza in ogni messaggio che trasmette. Un agente di ricezione può memorizzare queste informazioni per ogni utilizzo futuro.

Un agente di trasmissione deve seguire le seguenti regole in ordine di importanza.

1. Se l'agente di trasmissione ha l'elenco di funzionalità di decrittografia preferite da un destinatario, *dovrebbe* scegliere la prima funzionalità (preferenza più elevata) fra quelle che è in grado di usare.

2. Se l'agente di trasmissione non ha questo elenco di funzionalità dal destinatario ma ha ricevuto uno o più messaggi da tale destinatario, il messaggio in uscita *dovrebbe* utilizzare lo stesso algoritmo di crittografia utilizzato dall'ultimo messaggio ricevuto firmato e crittografato da quel destinatario.
3. Se l'agente di trasmissione non conosce nulla sulle funzionalità di decrittografia del destinatario previsto e accetta il rischio che il destinatario non sia in grado di decrittografare il messaggio, *dovrebbe* utilizzare tripleDES.
4. Se l'agente di trasmissione non conosce le funzionalità di decrittografia del destinatario previsto e non vuole rischiare che il destinatario non riesca a decrittografare il messaggio, allora *deve* utilizzare RC2/40.

Se un messaggio deve essere inviato a più destinatari e non è possibile scegliere un algoritmo di crittografia comune per tutti, l'agente di trasmissione dovrà inviare più messaggi. Tuttavia, in tal caso, è importante notare che il messaggio viene reso vulnerabile dalla trasmissione della copia con la sicurezza più bassa.

Messaggi S/MIME

S/MIME utilizza vari nuovi tipi di contenuti MIME, rappresentati nella Tabella 15.7. Tutti i nuovi tipi di applicazione usano la designazione PKCS che fa riferimento a un insieme di specifiche di crittografia a chiave pubblica emesse da RSA Laboratories e rese disponibili per S/MIME. Questi tipi e sottotipi verranno esaminati dopo aver descritto le procedure generali per la preparazione di un messaggio S/MIME.

Tabella 15.7 Tipi di contenuti S/MIME.

Tipo	Sottotipo	Parametro S/MIME	Descrizione
Multipart	Signed		Messaggio firmato in chiaro in due parti: una è il messoggio e l'altra è la firma.
Application	pkcs7-mime	signedData	Entità S/MIME firmoto.
	pkcs7-mime	envelopedData	Entitò S/MIME crittografata.
	pkcs7-mime	degenerate signedDota	Entità contenente solo certificoti di chiove pubblica.
	Pkcs7-mime	CompressedDato	Entità S/MIME compressa.
	pkcs7-signature	–	Tipo di contenuto corrispondente alla sottoparte firma di un messaggio firmato multipart.

Sicurezza di un'entità MIME

S/MIME rende sicura un'entità MIME con la firma, la crittografia o entrambe le funzionalità. Un'entità MIME può essere un intero messaggio (ad eccezione dell'intestazione RFC 822) o, se il tipo di contenuto MIME è multipart, allora l'entità MIME è una o più delle

sottoparti del messaggio. L'entità MIME viene preparata in base alle normali regole della preparazione dei messaggi MIME. Quindi l'entità MIME più alcuni dati di sicurezza (come gli identificatori dell'algoritmo e i certificati) vengono elaborati da S/MIME per produrre un cosiddetto oggetto PKCS. L'oggetto PKCS viene poi trattato come contenuto del messaggio e incorporato in MIME (dotato delle intestazioni MIME appropriate). Questa operazione risulterà più chiara analizzando gli oggetti coinvolti e fornendo alcuni esempi.
In ogni caso, il messaggio da inviare viene convertito in forma canonica. In particolare, dati un tipo e un sottotipo, per il contenuto del messaggio viene utilizzata la forma canonica appropriata. Per un messaggio multipart, viene utilizzata la forma canonica appropriata per ciascuna sottoparte.

L'uso della codifica di trasferimento richiede particolare attenzione. Per la maggior parte dei casi, il risultato dell'applicazione dell'algoritmo di sicurezza produrrà un oggetto parzialmente o totalmente rappresentato da dati binari arbitrari. Questo messaggio verrà incorporato in un messaggio MIME esterno e a questo punto si potrà applicare la codifica di trasferimento, normalmente base64. Tuttavia, nel caso di un messaggio firmato multipart, descritto in dettaglio più avanti, il contenuto del messaggio di una delle sottoparti non viene modificato dal processo di sicurezza tale contenuto, a meno che sia 7bit, dovrebbe essere codificato per il trasferimento utilizzando base64 o quoted-printable in modo che non vi sia il pericolo di modificare il contenuto a cui è stata applicata la firma.

Ora si descriveranno i vari tipi di contenuti S/MIME.

EnvelopedData

Il sottotipo application/pkcs7-mime viene utilizzato per quattro categorie di elaborazione S/MIME, ognuna delle quali ha un parametro smime-type univoco. In ogni caso l'entità risultante, chiamata *oggetto*, viene rappresentata in una forma chiamata BER (Basic Encoding Rules) definita nella raccomandazione ITU-T X.209. Il formato BER è costituito da stringhe di ottetti arbitrari ed è pertanto costituito da dati binari. Tale oggetto deve essere trasferito codificato con base64 in un messaggio MIME esterno.

Ecco le operazioni necessarie per preparare un entità MIME envelopedData.

1. Generare una chiave di sessione pseudocasuale per un dato algoritmo di crittografia simmetrico (RC2/40 o tripleDES).
2. Per ogni destinatario, crittografare la chiave di sessione con la chiave pubblica RSA del destinatario.
3. Per ogni destinatario, preparare un blocco chiamato RecipientInfo che contiene un identificatore del certificato della chiave pubblica del destinatario[3], un identificatore dell'algoritmo utilizzato per crittografare la chiave di sessione e la chiave di sessione crittografata.
4. Crittografare il contenuto del messaggio con la chiave di sessione.

I blocchi RecipientInfo seguiti dal contenuto crittografato costituiscono la parte envelopedData. Questa informazione viene poi codificata in base64. Ecco un esempio di messaggio (con l'esclusione delle intestazioni RFC 822):

[3] Si tratta di un certificato X.509, di cui si parlerà più avanti.


```
Content-Type: application/pkcs7-mime; smime-type=enveloped-data;
      name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m

rfvbnj756tbBghyHhHUujhJhjH77n8HHGT9HG4VQpfyF467GhIGfHfYT6
7n8HHGghyHhHUujhJh4VQpfyF467GhIGfHfYGTrfvbnjT6jH7756tbB9H
f8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
QGhIGfHfQbnj756YT64V
```

Per ripristinare il messaggio crittografato, il destinatario deve innanzitutto eliminare la codifica base64. Poi utilizza la propria chiave privata del destinatario per ripristinare la chiave di sessione. Infine il contenuto del messaggio viene decrittografato con la chiave di sessione.

SignedData

Il tipo signedData può essere utilizzato con uno o più firmatari. Per chiarezza, si limita la descrizione al caso di un'unica firma digitale. Ecco le operazioni da svolgere per preparare un'entità MIME signedData.

1. Selezionare un algoritmo message digest (SHA o MD5).
2. Calcolare il valore message digest (o la funzione hash) del contenuto da firmare.
3. Crittografare il message digest con la chiave privata del firmatario.
4. Preparare un blocco, chiamato SignerInfo, contenente il certificato della chiave pubblica del firmatario, l'identificatore dell'algoritmo message digest, l'identificatore dell'algoritmo utilizzato per crittografare il message digest e il message digest crittografato.

L'entità signedData è costituita da una serie di blocchi, tra i quali l'identificatore dell'algoritmo message digest, il messaggio da firmare e SignerInfo. L'entità signedData può anche includere un insieme di certificati di chiave pubblica tali da creare una catena da una radice riconosciuta o da un'autorità di certificazione di alto livello che giunge fino al firmatario. Questa informazione viene poi codificata in base64. Ecco un esempio di messaggio (escluse le intestazioni RFC 822):

```
Content-Type: application/pkcs7-mime; smime-type=signed-data;
      name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m

567GhIGfHfYT6ghyHhHUujpfyF4f8HHGTrfvhJhjH776tbB9HG4VQbnj7
77n8HHGT9HG4VQpfyF467GhIGfHfYT6rfvbnj756tbBghyHhHUujhJhjH
HUujhJh4VQpfyF467GhIGfHfYGTrfvbnjT6jH7756tbB9H7n8HHGghyHh
6YT64VOGhIGfHfQbnj75
```

Per ripristinare il messaggio firmato e verificare la firma, il destinatario deve innanzitutto eliminare la codifica base64. Poi usa la chiave pubblica del firmatario per decrittografare il message digest. Il destinatario calcola in modo indipendente il codice message digest e lo confronta con il message digest decrittografato per verificare la firma.

Firma in chiaro (Clear Signing)

La firma in chiaro è ottenuta utilizzando il tipo di contenuto multipart con il sottotipo signed. Come si è detto, questo processo di firma non prevede la trasformazione del messaggio da firmare e quindi il messaggio viene inviato "in chiaro". Pertanto i destinatari dotati di funzionalità MIME ma non S/MIME saranno in grado di leggere il messaggio in arrivo.

Un messaggio multipart/signed è composto da due parti. La prima può essere un tipo MIME qualsiasi ma deve essere preparata in modo che non venga modificata durante il trasferimento dall'origine alla destinazione. Questo significa che se la prima parte non è 7bit, dovrà essere codificata utilizzando base64 o quoted-printable. Questa parte viene poi elaborata nello stesso modo di signedData ma in questo caso viene creato un oggetto in formato signedData che ha un campo content del messaggio vuoto. Questo oggetto è una firma separata. Viene poi codificato utilizzando base64 per diventare la seconda parte del messaggio multipart/signed. Questa seconda parte ha un tipo di contenuto MIME application e un sottotipo pkcs7-signature. Ecco un esempio di messaggio.

```
Content-Type: multipart/signed;
      protocol="application/pkcs7-signature";
      micalg=sha1; boundary=boundary42

--boundary42
Content-Type: text/plain

This is a clear-signed message.

--boundary42
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s
ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
/GhIGfHfYT64VQbnj756
--boundary42--
```

Il parametro protocol indica che si tratta di un'entità a due parti firmata in chiaro. Il parametro micalg indica il tipo di message digest utilizzato. Il destinatario potrà verificare la firma prendendo il message digest della prima parte e confrontandolo con il message digest estratto dalla firma nella seconda parte.

Richiesta di registrazione

In genere un'applicazione o un utente richiede il certificato di chiave pubblica a un'autorità di certificazione. Per trasferire la richiesta di certificazione viene utilizzata l'entità S/MIME application/pkcs10. La richiesta di certificazione include il blocco certificationRequestInfo, l'identificatore dell'algoritmo di crittografia a chiave pubblica e la firma del blocco certificationRequestInfo, creata utilizzando la chiave privata del mittente. Il blocco certificationRequestInfo include il nome del soggetto del certificato (l'entità la cui chiave pubblica deve essere certificata) e una rappresentazione a stringa di bit della chiave pubblica dell'utente.

Messaggi di sola certificazione

In risposta a una richiesta di registrazione può essere inviato un messaggio contenente solo certificati o una lista di revoca dei certificati (CLR – Certificate Revocation List). Il messaggio è di tipo application/pkcs7-mime con parametro smime-type degenerate. Le operazioni previste sono le stesse già viste per la creazione di un messaggio signedData, tranne per il fatto che non esiste contenuto del messaggio e il campo signerInfo è vuoto.

Elaborazione dei certificati S/MIME

S/MIME utilizza certificati di chiave pubblica conformi alla versione 3 di X.509 (vedere il Capitolo 14). Il meccanismo di gestione della chiave utilizzato da S/MIME è per certi versi un ibrido fra la rigida gerarchia di certificazione X.509 e la rete di trust di PGP. Come nel modello PGP, i manager e/o gli utenti S/MIME devono configurare ciascun client con un elenco di chiavi fidate e con liste di revoca dei certificati. In pratica la responsabilità della gestione dei certificati necessari per verificare le firme in ingresso e per crittografare i messaggi in uscita è locale. I certificati sono invece firmati dalle autorità di certificazione.

Il ruolo dell'agente utente

Un utente S/MIME deve svolgere varie funzioni di gestione della chiave.

- **Generazione della chiave:** l'utente di un programma di servizio amministrativo (per esempio associato alla gestione della rete locale) *deve* essere in grado di generare coppie di chiavi distinte Diffie-Hellman e DSS e *dovrebbe* essere in grado di generare coppie di chiavi RSA. Ciascuna coppia di chiavi *deve* essere generata da una buona fonte di input casuale non deterministica ed essere protetta in modo sicuro. Un agente utente *dovrebbe* generare coppie di chiavi RSA con una lunghezza compresa fra 768 e 1024 bit ma che *non deve* essere minore di 512 bit.
- **Registrazione:** la chiave pubblica di un utente dovrebbe essere registrata da un'autorità di certificazione per poter ricevere un certificato di chiave pubblica X.509.
- **Memorizzazione e ricerca dei certificati:** per poter verificare le firme in ingresso e per crittografare i messaggi in uscita, un utente deve accedere a un elenco locale di certificati. Tale elenco può essere gestito dall'utente o da un'entità amministrativa locale per conto di più utenti.

Certificati VeriSign

Esistono varie aziende che forniscono servizi di certificazione. Per esempio, Nortel ha progettato una soluzione di certificazione aziendale e può fornire il supporto S/MIME all'interno di un'azienda. Vi sono varie autorità di certificazione in Internet fra cui VeriSign, GTE e il servizio postale statunitense. Di queste, la più utilizzata è VeriSign, della quale verrà fornita una breve descrizione.

VeriSign fornisce un servizio di certificazione compatibile con S/MIME e con varie altre applicazioni. VeriSign emette certificati X.509 con il nome di prodotto VeriSign Server Digital ID. All'inizio del 1998, più di 35 mila siti Web commerciali utilizzavano VeriSign Server Digital ID e sono stati emessi oltre un milione di codici digitali per utenti dei browser Netscape e Microsoft. Le informazioni contenute in un Digital ID dipendono dal tipo di Digital ID e dal suo uso. Come minimo, ciascun Digital ID contiene le seguenti informazioni.

- La chiave pubblica del proprietario.
- Il nome o l'alias del proprietario.
- La data di scadenza del Digital ID.
- Il numero di serie del Digital ID.
- Il nome dell'autorità di certificazione che ha emesso il Digital ID.
- La firma digitale dell'autorità di certificazione che ha emesso il Digital ID.

I codici Digital ID possono anche contenere altre informazioni fornite dall'utente, fra cui le seguenti.

- Indirizzo
- Indirizzo di posta elettronica.
- Semplici informazioni di registrazione (paese, codice, CAP, età e genere).

VeriSign fornisce tre livelli o classi di sicurezza per i certificati di chiave pubblica (vedere la Tabella 15.8). Un utente richiede online un certificato al sito Web VeriSign o ad altri siti Web consociati. Le richieste di Classe 1 e Classe 2 vengono elaborate online e nella maggior parte dei casi approvate in pochi secondi. Brevemente ecco le procedure utilizzate.

- Per i codici Digital ID di Classe 1, VeriSign conferma l'indirizzo di posta elettronica dell'utente inviando all'indirizzo di posta elettronica fornito all'applicazione un codice PIN e le informazioni per il prelevamento del codice Digital ID.
- Per codici Digital ID di Classe 2, VeriSign verifica le informazioni nell'applicazione tramite un confronto automatico con un database di consumatori oltre a svolgere tutti i controlli associati ai codice Digital ID di Classe 1. Infine invia una conferma all'indirizzo postale specificato avvertendo l'utente che è stato emesso un codice Digital ID a suo nome.
- Per i codici Digital ID di Classe 3, VeriSign richiede un livello più elevato di verifica dell'identità. Una persona deve dimostrare la propria identità fornendo credenziali autenticate o comparendo di persona.

Tabella 15.8 Le classi di certificati di chiave pubblica VeriSign.

	Conferma dell'identità	Protezione della chiave privata dell'autorità emettitrice	Protezione della chiave privata dell'abbonato e del richiedente il certificato	Applicazioni implementate dagli utenti
Classe 1	Ricerca automatizzata dell'unicità del nome e dell'indirizzo di posta elettronica	PCA: hardware fidato; CA: hardware o software fidato	Software di crittografia (protetto da PIN) consigliato ma non richiesto.	Navigazione Web e uso della posta elettronica
Classe 2	Come la Classe 1 più verifica automatica delle informazioni fornite e verifica automatica dell'indirizzo	PCA e CA: hardware fidato	Software di crittografia (protetto da PIN).	Posta elettronica individuale interno ed esterno all'azienda, sottoscrizioni online, sostituzione password e convalida del software.
Classe 3	Come la Classe 1 più presenza personale e documentaziane di identità e più i controlli automatizzati di Classe 2 per le singole persone; registrazioni commerciali per le aziende.	PCA e CA: hardware fidato	Software di crittografia (protetto da PIN); token hardware consigliato ma non richiesto.	Applicazioni di banco elettronica, accesso a database aziendali, gestiane bancaria personale, servizi online per soli membri, servizi di integrità dei contenuti, server di commercio elettronico, convalida del software; autenticazione LRAA, crittografia forte per alcuni server.

CA: autorità di certificazione
PCA: VeriSign Public Primary Certification Authority
PIN: Personal Identification Number
LRAA: Local Registration Authority Administrator

Servizi di sicurezza avanzati

Al momento attuale, sono stati proposti tre servizi di sicurezza avanzati come standard Internet. Trattandosi di una proposta i dettagli potrebbero anche cambiare e potrebbero essere aggiunti nuovi servizi. I tre servizi sono i seguenti.

- **Ricevute firmate:** in un oggetto SignedData può essere richiesta una ricevuta firmata. La restituzione di una ricevuta firmata prova la consegna di un messaggio e consente al mittente di dimostrare a terzi che il destinatario ha ricevuto il messaggio. Sostanzial-

mente il destinatario firma l'intero messaggio originale più la firma originale del mittente aggiungendo la nuova firma per formare un nuovo messaggio S/MIME.

- **Etichette di sicurezza:** negli attributi autenticati di un oggetto SignedData può essere inclusa un'etichetta di sicurezza, costituita da un insieme di informazioni di sicurezza riguardanti la segretezza dei contenuti protetti dall'incapsulazione S/MIME. Le etichette possono essere utilizzate per il controllo degli accessi, indicando quali utenti possono accedere a un oggetto. Altri usi includono il livello di segretezza o il ruolo, descrivendo quali classi di utenti possono consultare le informazioni (per esempio il team sanitario di un paziente, gli addetti alla fatturazione medica e così via).
- **Mailing list sicure:** quando un utente invia un messaggio a più destinatari, è richiesta una certa elaborazione specifica per ogni destinatario, incluso l'uso della chiave pubblica di ciascun destinatario. L'utente può evitare questo carico impiegando i servizi di un MLA (Mail List Agent) S/MIME. Un MLA può prendere un messaggio in arrivo, svolgere la crittografia specifica per ciascun destinatario e inoltrare il messaggio. Il mittente del messaggio deve semplicemente inviare il messaggio al MLA crittografato utilizzando la chiave pubblica del MLA.

15.3 Letture e siti Web consigliati

- **PGP Home Page:** sito Web PGP gestito da PGP Corp., il principale produttore commerciale di PGP.
- **International PGP Home Page:** pagina dedicata alla promozione di PGP a livello internazionale. Contiene documenti e collegamenti interessanti.
- **PGP Charter:** contiene le RFC più recenti e le bozze Internet relative alle "specifiche aperte" di PGP.
- **MIT Distribution Site for PGP:** il principale distributore di PGP freeware. Contiene documenti FAQ, altre informazioni e link verso altri siti dedicati a PGP.
- **S/MIME Charter:** gli ultimi documenti RFC e le ultime bozze per S/MIME.

15.4 Termini chiave, domande di ripasso e problemi

Termini chiave

Chiave di sessione
Firma separata
MIME (Multipurpose Internet Mail Extensions)
PGP (Pretty Good Privacy)
Posta elettronica
radix64
S/MIME
Trust
ZIP

Domande di ripasso

15.1 Quali sono i cinque principali servizi forniti da PGP?
15.2 Qual è l'utilità di una firma separata?
15.3 Perché PGP genera la firma prima di applicare la compressione?
15.4 Che cos'è la conversione R64?
15.5 Perché la conversione R64 è utile per un'applicazione di posta elettronica?
15.6 Perché in PGP è richiesta la funzione di segmentazione e assemblaggio?
15.7 In quale modo viene utilizzato il concetto di trust (fiducia) in PGP?
15.8 Che cos'è la RFC 822?
15.9 Che cosa è MIME?
15.10 Che cos'è S/MIME?

Problemi

15.1 PGP utilizza la modalità CFB (Cipher Feedback) di CAST-128 mentre la maggior parte delle applicazioni a crittografia simmetrica (tranne per la crittografia della chiave) usa la modalità CBC (Cipher Block Chaining). Si ha:

$$\text{CBC}: C_i = E(K, [C_{i-1} \oplus P_i]); \qquad P_i = C_{i-1} \oplus D(K, C_i)$$
$$\text{CFB}: C_i = P_i \oplus E(K, C_{i-1}); \qquad P_i = C_i \oplus E(K, C_{i-1})$$

Questi due schemi sembrano offrire la stessa sicurezza. Suggerire un motivo per cui PGP usa la modalità CFB.

15.2 Nello schema PGP, qual è statisticamente il numero di chiavi di sessione generate prima che venga prodotta una chiave di sessione già creata in precedenza?

15.3 In PGP, qual è la probabilità che un utente con N chiavi pubbliche abbia almeno un codice ID della chiave duplicato?

15.4 I primi 16 bit del message digest di una firma PGP vengono tradotti in chiaro.
A. Fino a che punto ciò compromette la sicurezza dell'algoritmo hash?
B. Fino a che punto questo svolge la sua funzione prevista, ovvero aiuta a determinare se per decrittografare il messagge digest è stata utilizzata la chiave RSA corretta?

15.5 Nella Figura 15.4, ciascuna voce di un ring di chiavi pubbliche contiene un campo che indica il grado di fiducia relativo al proprietario di questa chiave pubblica. Perché non è sufficiente? Ovvero, se il proprietario è fidato e si suppone che sia veramente il proprietario della chiave pubblica, perché ciò non è sufficiente per consentire a PGP di utilizzare questa chiave pubblica?

15.6 Considerare la conversione radix-64 come una forma di crittografia. In questo caso non vi è alcuna chiave. Ma si supponga che un estraneo conosca solo che è stato utilizzato un algoritmo a sostituzione per crittografare testo in inglese, senza che sappia che è stato crittografato R64. Quanto sarebbe efficace questo algoritmo contro l'analisi crittografica?

15.7 Phil Zimmermann ha scelto IDEA, triple DES a tre chiavi e CAST-128 come algoritmi di crittografia simmetrici per PGP. Fornire i motivi per cui i seguenti algoritmi di crittografia simmetrica descritti nel volume sono adatti o meno per PGP: DES, triple DES a due chiavi e AES.

Appendice 15.A Compressione dei dati con ZIP

PGP utilizza il pacchetto di compressione ZIP, scritto da Jean-lup Gailly, Mark Adler e Richard Wales. ZIP è un pacchetto freeware scritto in C, utilizzabile in Unix e altri sistemi, funzionalmente equivalente a PKZIP, un pacchetto shareware ampiamente disponibile per sistemi Windows sviluppato da PKWARE, Inc. L'algoritmo ZIP è forse la tecnica di compressione inter-piattaforma più utilizzata; esistono versioni freeware e shareware di ZIP per Macintosh, Windows, Unix e altri sistemi.

ZIP e altri algoritmi simili derivano dalla ricerca svolta da Jacob Ziv e Abraham Lempel. Nel 1977, essi descrissero una tecnica basata su un buffer a finestra di scorrimento che contiene il test elaborato più recentemente [ZIV77]. Questo algoritmo è generalmente chiamato LZ77. Lo schema di compressione ZIP (PKZIP, gzip, zipit e così via) utilizza una variante di questo algoritmo.

LZ77 e le sue varianti sfruttano il fatto che le parole e le frasi in un flusso di testo (o le sequenze di pixel nelle immagini GIF) spesso si ripetono. Quando si verifica una ripetizione, la sequenza ripetuta viene sostituita da un breve codice. Il programma di compressione individua queste ripetizioni e sviluppa dei codici "al volo" per sostituire la sequenza ripetuta. Nel corso del tempo i codici vengono riutilizzati per catturare nuove sequenze. L'algoritmo deve essere definito in modo che il programma di decompressione sia in grado di individuare le associazioni fra i codici e le sequenze presenti nei dati.

Prima di analizzare i dettagli di LZ77, si può osservare un semplice esempio.[4] Si consideri la seguente frase insensata:

```
the brown fox jumped over the brown foxy jumping frog
```

La frase è costituita da 53 ottetti, ovvero da 424 bit. L'algoritmo elabora questo testo da sinistra a destra. Inizialmente ciascun carattere viene mappato in uno schema di 9 bit costituito da un 1 binario seguito dalla rappresentazione ASCII a 8 bit del carattere. A mano a mano che procede l'elaborazione, l'algoritmo individua delle sequenze ripetute. Quando incontra una ripetizione, l'algoritmo procede con la scansione fino al termine della ripetizione. In altre parole, ogni volta che si verifica una ripetizione, l'algoritmo include quanti più caratteri possibili. La prima sequenza di questo tipo incontrata è **the brown fox**. Questa sequenza viene sostituita da un puntatore alla prima sequenza e dalla lunghezza della sequenza. In questo caso, la sequenza precedente di **the brown fox** si trova 26 caratteri prima e la lunghezza della sequenza è di 13 caratteri. Per questo esempio, si supponga l'impiego di due opzioni di codifica; un puntatore a 8 bit e una lunghezza di 4 bit oppure un puntatore

[4] Si basa su un esempio presente in [WEIS93].

di 12 bit e una lunghezza di 6 bit. Un'intestazione di 2 bit indica quale opzione viene utilizzata, dove 00 indica la prima opzione e 01 la seconda opzione. Pertanto la seconda occorrenza della frase **the brown fox** verrà codificata come $<00_b><26_d><13_d>$ o 00 00011010 1101.

Le parti rimanenti del messaggio compresso sono la lettera **y**, la sequenza $<00_b><27_d><5_d>$ che sostituisce la sequenza costituita dal carattere di spazio seguita dalla parola **jump** e la sequenza di caratteri **ing frog**.

La Figura 15.9 illustra il funzionamento della compressione. Il messaggio compresso è costituito da 35 caratteri di 9 bit e due codici per un totale di $35 \times 9 + 2 \times 14 = 343$ bit. Se paragonato alla lunghezza di 424 bit del messaggio non compresso, si ottiene un rapporto di compressione pari a 1,24.

L'algoritmo di compressione

L'algoritmo di compressione di LZ77 e delle sue varianti utilizza due buffer. Un buffer a scorrimento "storico" contiene gli ultimi N caratteri del codice di origine che è stato elaborato e il buffer **look-ahead** contiene i successivi L caratteri da elaborare (Figura 15.10A). L'algoritmo tenta di individuare due o più caratteri dall'inizio del buffer look-ahead in una stringa del buffer storico a scorrimento. Se non viene trovata alcuna corrispondenza, il primo carattere del buffer look-ahead viene inviato in output come un carattere di 9 bit e la finestra di scorrimento viene spostata eliminando il carattere meno recente. Se viene trovata una corrispondenza, l'algoritmo procede con la scansione per individuare la corrispondenza più lunga. Poi la stringa individuata viene inviata in output come una tripletta (indicatore, puntatore, lunghezza). Per una stringa di K caratteri, vengono fatti fuoriuscire dalla finestra a scorrimento i K caratteri meno recenti e i K caratteri della stringa codificata vengono fatti scorrere all'interno della finestra.

La Figura 15.10B mostra il funzionamento di questo meccanismo sulla semplice sequenza di esempio impiegata in precedenza. L'illustrazione presuppone una finestra a scorrimento di 39 caratteri e un buffer look-ahead di 13 caratteri. Nella parte superiore dell'esempio, i primi 40 caratteri sono stati già elaborati e nella finestra a scorrimento si trova la versione non compressa dei 39 caratteri più recenti. La parte rimanente del testo sorgente si trova nella finestra look-ahead. L'algoritmo di compressione determina la corrispondenza successiva, esegue uno scorrimento di cinque caratteri dal buffer look-ahead nella finestra a scorrimento e produce in output il codice per questa stringa. Lo stato del buffer dopo queste operazioni è illustrato nella parte inferiore dell'esempio.

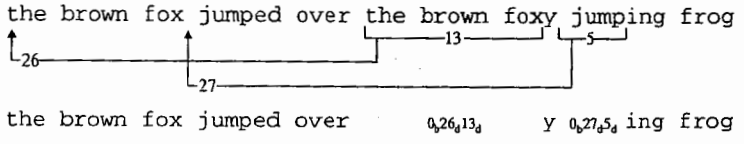


Figura 15.9 Funzionamento dello schema LZ77.

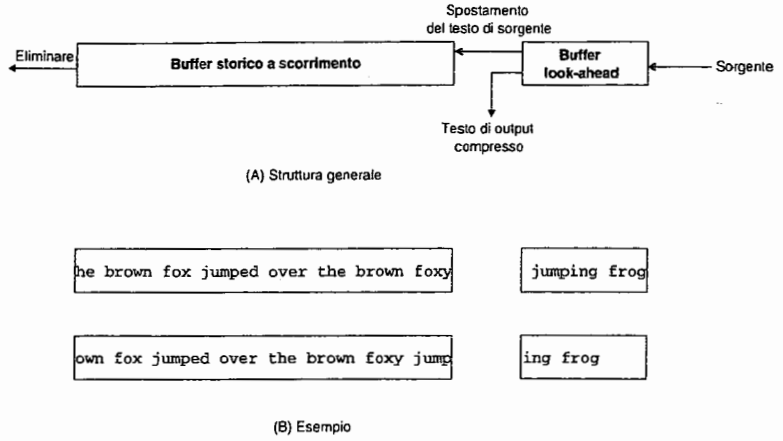


Figura 15.10 Il meccanismo di funzionamento di LZ77.

LZ77 è efficace e si adatta alla natura dell'input corrente, ma presenta alcuni difetti. Per individuare le corrispondenze nel testo l'algoritmo usa una finestra di dimensioni finite. Per un blocco di testo molto lungo, rispetto alle dimensioni della finestra, ciò impedisce di individuare molte potenziali corrispondenze. È possibile aumentare le dimensioni della finestra ma questo impone due penalità: (1) aumenta il tempo di elaborazione dell'algoritmo poiché deve svolgere un confronto fra stringhe rispetto al buffer look-ahead per ogni posizione della finestra di scorrimento e (2) il campo del puntatore deve essere di maggiori dimensioni per poter codificare salti di maggior lunghezza.

L'algoritmo di decompressione

La decompressione del testo compresso con LZ77 è molto semplice. L'algoritmo di decompressione deve salvare gli ultimi *N* caratteri dell'output decompresso. Quando incontra una stringa codificata, l'algoritmo di decompressione usa i campi puntatore e lunghezza per sostituire il codice con la stringa di testo.

Appendice 15.B La conversione radix-64

Sia PGP che S/MIME utilizzano la tecnica di conversione radix-64. Questa tecnica associa l'input binario a un output costituito da caratteri stampabili. Questa codifica ha le seguenti caratteristiche.

1. Il codominio (l'output) della funzione è un set di caratteri universalmente rappresentabile ovunque, non una codifica binaria specifica di un determinato set di caratteri. Pertanto i caratteri stessi possono essere codificati in qualsiasi forma necessaria per un determinato sistema. Per esempio, il carattere "E" è rappresentato in ASCII con il codice esadecimale 45 e in EBCDIC con il codice esadecimale C5.
2. Il set di caratteri è costituito da 65 caratteri stampabili, uno dei quali viene utilizzato per il riempimento. Con $2^6 = 64$ caratteri disponibili, ciascun carattere può essere utilizzato per rappresentare 6 bit di input.
3. Nel set non è incluso alcun carattere di controllo. Pertanto un messaggio codificato in radix-64 può attraversare i sistemi di gestione della posta elettronica che eseguono la scansione del flusso dei dati alla ricerca dei caratteri di controllo.
4. Il carattere di trattino ("-") non viene usato. Questo carattere ha infatti un significato nel formato RFC 822 e deve pertanto essere evitato.

La Tabella 15.9 mostra il mapping fra i valori di input a 6 bit e i corrispondenti caratteri. Il set di caratteri è costituito dai caratteri alfanumerici più i simboli "+" e "/". Come carattere di riempimento viene utilizzato il carattere "=".

La Figura 15.11 illustra il semplice schema di mapping. L'input binario viene elaborato in blocchi di tre ottetti o 24 bit. Ciascun insieme di 6 bit nel blocco di 24 bit viene associato a un carattere. Nella figura i caratteri sono codificati in quantità di 8 bit. In questo caso tipico, ciascun input di 24 bit viene espanso in 32 bit di output.

Per esempio, si consideri la sequenza di 24 bit 00100011 01011100 10010001 che può essere espressa in esadecimale come 235C91. Si può disporre questo input in blocchi di 6 bit:

001000 110101 110010 010001

I valori decimali di 6 bit estratti sono 8, 53, 50, 17. Ricercandoli nella Tabella 15.9, si ottiene la codifica radix-64 con i seguenti caratteri: I1yR. Se questi caratteri vengono memorizzati in formato ASCII a 8 bit con il bit di parità a 0, si avrà:

01001001 00110001 01111001 01010010

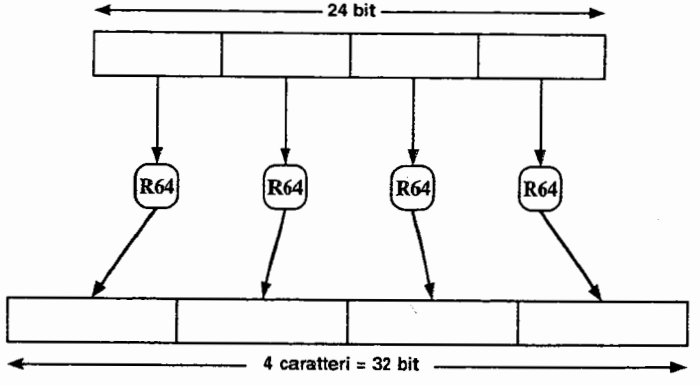


Figura 15.11 Codifica stampabile dei dati binari in formato radix-64.

Tabella 15.9 La codifica radix-64.

Valore di 6 bit	Carattere corrispondente	Valore di 6 bit	Carattere corrispondente	Valore di 6 bit	Carattere corrispondente	Valore di 6 bit	Carattere corrispondente
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	-
15	P	31	f	47	v	63	/
						(pad)	=

In esadecimale si tratta del valore 49317952. Per riepilogare:

Dati di input	
Rappresentazione binaria	00100011 01011100 10010001
Rappresentazione esadecimale	235C91
Codifica radix-64 dei dati di input	
Rappresentazione a caratteri	I1yR
Codice ASCII (8 bit parità 0)	01001001 00110001 01111001 01010010
Rappresentazione esadecimale	49317952

Appendice 15.C Generazione di numeri casuali in PGP

PGP usa un meccanismo complesso e potente per generare i numeri casuali e pseudocasuali per vari scopi. PGP genera i numeri casuali utilizzando i tasti digitati dall'utente e misurando le pause fra le pressioni e genera numeri pseudocasuali utilizzando un algoritmo basato su ANSI X9.17. PGP usa questi numeri per i seguenti scopi.

- Veri numeri casuali:
  - usati per generare coppie di chiavi RSA;
  - forniscono il seme iniziale per il generatore di numeri pseudocasuali;
  - forniscono ulteriori input durante la generazione di numeri pseudocasuali.
- Numeri pseudocasuali:
  - usati per generare le chiavi di sessione;
  - usati per generare i vettori di inizializzazione, impiegati con la chiave di sessione nella crittografia in modalità CFB.

Veri numeri casuali

PGP gestisce un buffer di 256 byte di bit casuali. Ogni volta che PGP si attende la pressione di un tasto, registra l'istante in cui inizia ad aspettare in formato a 32 bit. Quando viene premuto il tasto, registra nuovamente l'istante di tempo e il valore a 8 bit del tasto. Le informazioni così rilevate (temporali + codice tasto) vengono poi utilizzate per generare una chiave che, a sua volta, viene utilizzata per crittografare il valore corrente del buffer di bit casuali.

Numeri pseudocasuali

La generazione di numeri pseudocasuali utilizza un seme di 24 ottetti e produce una chiave di sessione di 16 ottetti, un vettore di inizializzazione di 8 ottetti e un nuovo seme da utilizzare per la generazione del numero pseudocasuale sucessivo. L'algoritmo si basa sull'algoritmo X9.17 descritto nel Capitolo 7 (vedere la Figura 7.14) ma per la crittografia utilizza CAST-128 invece di triple DES. L'algoritmo usa le seguenti strutture dati.

1. Input
   - randseed.bin (24 ottetti): se questo file è vuoto, viene riempito con 24 ottetti casuali.
   - Messaggio: la chiave di sessione e il vettore di inizializzazione che verranno utilizzati per crittografare il messaggio sono essi stessi una funzione del messaggio. Questo contribuisce ulteriormente alla casualità della chiave e del vettore di inizializzazione; se un estraneo dovesse conoscere già il contenuto in chiaro del messaggio, non avrà alcuna necessità apparente per catturare la relativa chiave di sessione monouso.
2. Output
   - K (24 ottetti): i primi 16 ottetti, K[0..15], contengono la chiave di sessione e gli ultimi otto ottetti, K[16..23], contengono il vettore di inizializzazione.

- randseed.bin (24 ottetti): in questo file viene inserito il nuovo valore del seme.

3. Strutture dati interne
   - dtbuf (8 ottetti): i primi quattro ottetti, dtbuf[0..3], vengono inizializzati con il valore della data/ora corrente. Questo buffer è equivalente alla variabile DT dell'algoritmo X12.17.
   - rkey (16 ottetti): chiave di crittografia CAST-128 utilizzata in tutte le fasi dell'algoritmo.
   - rseed (8 ottetti): equivalente alla variabile V_i di X12.17.
   - rbuf (8 ottetti): un numero pseudocasuale generato dall'algoritmo. Questo buffer è equivalente alla variabile R_i di X12.17.
   - K´ (24 ottetti): buffer temporaneo per il nuovo valore di randseed.bin.

L'algoritmo è costituito da nove passi, da G1 a G9. Il primo e l'ultimo sono passi di confusione che hanno lo scopo di ridurre il valore di un eventuale file randseed.bin catturato da un estraneo. I rimanenti passi sono fondamentalmente equivalenti a tre iterazioni dell'algoritmo X12.17 e sono illustrati nella Figura 15.12 (da confrontare con la Figura 7.14). Per riepilogare, vengono svolte le seguenti operazioni.

G1. Prelavaggio del seme precedente.

A. Copiare randseed.bin in K[0..23].
B. Prendere il codice hash del messaggio (è già stato generato se il messaggio è stato firmato, altrimenti vengono utilizzati i primi 4K ottetti del messaggio). Usare il risultato come chiave, usare un vettore di inizializzazione nullo e crittografare K in modalità CFB. Memorizzare nuovamente il risultato in K.

G2. Impostazione del seme iniziale.

A. Assegnare a dtbuf[0..3] il tempo locale come valore di 32 bit. Assegnare a dtbuf[4..7] una sequenza di "0". Copiare rkey ← K[0..15]. Copiare rseed ← K[16..23].
B. Crittografare il dtbuf a 64 bit utilizzando la chiave rkey di 128 bit in modalità ECB. Memorizzare nuovamente il risultato in dtbuf.

G3. Preparazione alla generazione degli ottetti casuali

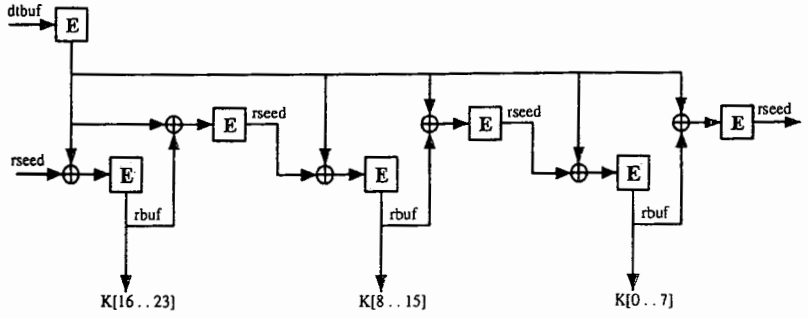


Figura 15.12 La generazione della chiave di sessione e del vettore di inizializzazione in PGP (passi da G2 a G8).

Effettuare gli assegnamenti rcount ← 0 e k ← 23. Il ciclo di passi da G4 a G7 verrà eseguito 24 volte (k = 23 ... 0), uno per ogni ottetto casuale prodotto e inserito in K. La variabile rcount è il numero di ottetti casuali non ancora utilizzati in rbuf. Conterà da 8 a 0 per tre volte per generare i 24 ottetti.

G4. Byte disponibili?

Se rcount = 0, goto G5 else goto G7. I passi G5 e G6 svolgono un'istanza dell'algoritmo X12.17 per generare un nuovo gruppo di otto ottetti casuali.

G5. Generazione di nuovi ottetti casuali

A. rseed ← rseed ⊕ dtbuf
B. rbuf ← E_{rkey}[rseed] in modalità ECB.

G6. Generazione del seme successivo

A. rseed ← rbuf ⊕ dtbuf
B. rseed ← E_{rkey}[rseed] in modalità ECB.
C. Eseguire l'assegnamento rcount ← 8.

G7. Trasferimento di un byte alla volta da rbuf a K

A. Eseguire l'assegnamento rcount ← rcount – 1.
B. Generare un vero byte casuale b ed eseguire l'assegnamento K[k] ← rbuf[rcount] ⊕ b.

G8. Fine? If k = 0 goto G9 else k ← k – 1 e goto G4

G9. Post lavaggio del seme e restituzione del risultato

A. Generare altri 24 byte con il metodo dei passi G4-G7 ma non eseguire l'operazione di XOR con il byte casuale in G7. Inserire il risultato nel buffer K´.
B. Crittografare K´ con la chiave K[0..15] e il vettore di inizializzazione K[16..23] in modalità CFB; memorizzare il risultato in randseed.bin.
C. Restituire K.

Non dovrebbe essere possibile determinare la chiave di sessione partendo dai 24 nuovi ottetti generati nel passo G9.A. Tuttavia, per assicurarsi che il file randseed.bin memorizzato non fornisca informazioni sulla chiave di sessione più recente, i 24 nuovi ottetti vengono crittografati e il risultato viene memorizzato come nuovo seme. Questo elaborato algoritmo dovrebbe fornire numeri pseudocasuali crittograficamente resistenti.

Capitolo 16

La sicurezza IP

Concetti essenziali

- **IPSec** (sicurezza IP) costituisce un insieme di funzioni di sicurezza con le quali è possibile estendere sia il Protocollo Internet versione attuale (IPv4), sia versione 6 (IPv6), tramite intestazioni addizionali.
- IPSec comprende tre aree funzionali: autenticazione, segretezza e gestione delle chiavi.
- L'**autenticazione** si basa sul codice di autenticazione dei messaggi HMAC. L'autenticazione può essere applicata all'intero pacchetto IP originale (*modalità tunnel*) o al pacchetto esclusa l'intestazione IP (*modalità transport*).
- La **segretezza** si basa sul formato di crittografia noto come ESP (Encapsulating Security Payload). Possono essere gestite entrambe le modalità tunnel e trasporto.
- IPSec definisce varie tecniche per la **gestione delle chiavi**.

La comunità Internet ha sviluppato dei meccanismi di sicurezza specifici per varie attività, fra cui la posta elettronica (S/MIME, PGP), le comunicazioni client/server (Kerberos), l'accesso al Web (SSL – Secure Sockets Layer) e altri. Tuttavia gli utenti si preoccupano della sicurezza di ciò che attraversa i vari livelli di protocolli. Per esempio, un'azienda può impiegare una rete TCP/IP privata sicura impedendo il collegamento a siti Web non fidati, eseguendo la crittografia dei pacchetti in uscita e l'autenticazione dei pacchetti in entrata. Implementando la sicurezza a livello IP, un'azienda può assicurarsi una gestione sicura della rete non solo per le applicazioni dotate di meccanismi di sicurezza ma per le molte applicazioni che non si preoccupano della sicurezza.

La sicurezza IP riguarda tre aree funzionali: l'autenticazione, la segretezza e la gestione delle chiavi. Il meccanismo di autenticazione garantisce che un pacchetto ricevuto sia stato realmente trasmesso dalla sorgente indicata nell'intestazione del pacchetto stesso. Inoltre, questo meccanismo garantisce che il pacchetto non sia stato alterato durante il

transito. La funzionalità di segretezza consente ai nodi di comunicazione di crittografare i messaggi per impedire intercettazioni da parte di estranei. Il sistema di gestione della chiave si occupa dello scambio sicuro delle chiavi.

Il capitolo si apre con una panoramica sulla sicurezza IP (IPSec) e un'introduzione all'architettura IPSec. Quindi si parlerà più in dettaglio delle tre aree funzionali. L'appendice di questo capitolo riesamina i protocolli per Internet.

16.1 Panoramica sulla sicurezza IP

Nel 1994, lo IAB (Internet Architecture Board) emise un rapporto intitolato "Security in the Internet Architecture" (RFC 1636). Il rapporto affermava il generale consenso sul fatto che fosse necessario migliorare il livello di sicurezza di Internet e identificava le aree chiave per i meccanismi di sicurezza. Fra queste, la necessità di rendere sicura l'infrastruttura della rete da un monitoraggio non autorizzato del traffico e la necessità di rendere sicuro il traffico in transito fra gli utenti finali utilizzando meccanismi di autenticazione e crittografia.

Queste preoccupazioni sono pienamente giustificate. Come conferma, il rapporto annuale del 2003 del CERT (Computer Emergency Response Team) elencava più di 137 000 incidenti alla sicurezza. Fra gli attacchi di tipo più grave si citano lo spoofing IP, in cui un estraneo crea pacchetti con indirizzi IP fasulli e sfrutta le applicazioni che usano l'autenticazione basata su IP, e varie forme di intercettazione dei pacchetti in cui un estraneo legge le informazioni trasmesse, comprese le informazioni di login e il contenuto di database.

In risposta a questi problemi, l'IAB ha incluso l'autenticazione e la crittografia nelle funzionalità di sicurezza necessarie per la prossima versione del protocollo IP, IPv6. Fortunatamente queste funzionalità di sicurezza sono state progettate per essere utilizzabili sia nella versione corrente del protocollo, IPv4, che nella versione futura IPv6. Questo significa che i produttori possono iniziare a offrire fin da ora queste funzionalità: molti infatti offrono già oggi nei propri prodotti alcune funzionalità IPSec.

Le applicazioni di IPSec

IPSec offre la possibilità di rendere sicure le comunicazioni all'interno di una rete locale, fra reti geografiche private e pubbliche e in Internet. Ecco alcuni esempi.

- **Connettività sicura delle sedi locali via Internet:** un'azienda può realizzare una rete privata virtuale sicura attraverso Internet o una rete geografica pubblica. Questo consente all'azienda di sfruttare Internet per ridurre la necessità di reti private, risparmiando in costi e attività di gestione della rete.
- **Accesso remoto sicuro via Internet:** un utente finale il cui sistema è dotato di protocolli di sicurezza IP può eseguire una chiamata locale a un provider Internet e acquisire un accesso sicuro alla rete aziendale. Questo riduce i costi telefonici per i dipendenti dell'azienda che si trovano frequentemente in viaggio e per i tele-lavoratori.

- **Attivazione della connettività extranet e intranet con partner commerciali:** IPSec può essere utilizzato per rendere sicure le comunicazioni con le altre aziende, garantendo i servizi di autenticazione e segretezza e fornendo un meccanismo di scambio delle chiavi.
- **Miglioramento della sicurezza del commercio elettronico:** anche se alcune applicazioni Web e di commercio elettronico sono dotate di protocolli interni per la sicurezza, l'impiego di IPSec migliora questo livello di sicurezza.

La caratteristica principale di IPSec che gli consente di supportare queste differenti applicazioni è il fatto che può crittografare e/o autenticare tutto il traffico a livello IP. Pertanto possono essere rese sicure tutte le applicazioni distribuite, fra cui i login remoti, le comunicazioni client/server, la posta elettronica, il trasferimento di file, l'accesso al Web.

La Figura 16.1 mostra un tipico impiego del protocollo IPSec. Un'azienda è dotata di varie reti locali in filiali remote. All'interno di ciascuna rete locale il traffico IP non è sicuro. Per il traffico che esce dai limiti della sede, sfruttando una rete geografica privata o pubblica, vengono utilizzati i protocolli IPSec. Questi protocolli operano direttamente nei dispositivi di rete, come i router o i firewall che connettono ciascuna rete locale al mondo esterno. In genere il dispositivo di rete IPSec esegue la crittografia e la compressione di tutto il traffico in ingresso nella rete geografica e la decrittografia e la decompressione del traffico proveniente dalla rete geografica. Queste operazioni sono trasparenti per le workstation e i server della rete locale. È anche possibile attivare connessioni sicure con singoli utenti che si collegano alla rete geografica. Per garantire la sicurezza, le workstation di questi utenti devono implementare i protocolli IPSec.

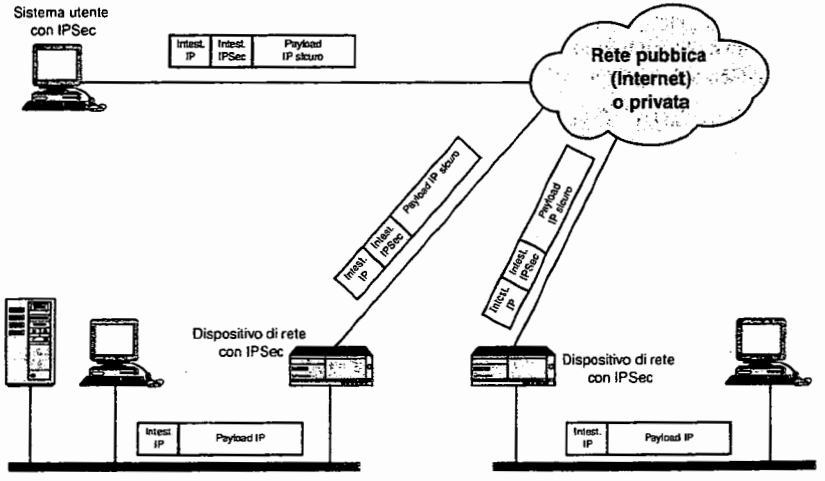


Figura 16.1 Uno scenario d'uso di IPSec.

I vantaggi di IPSec

[MARK97] elenca i seguenti vantaggi di IPSec.

- Quando IPSec viene implementato in un firewall o in un router, fornisce un livello di sicurezza elevato che può essere applicato a tutto il traffico che attraversa il perimetro. Il traffico interno dell'azienda o del gruppo di lavoro non subisce il sovraccarico rappresentato dall'elaborazione dedicata alla sicurezza.
- L'installazione di IPSec in un firewall risulterà difficile da eludere se tutto il traffico proveniente dall'esterno deve utilizzare il protocollo IP e il firewall è l'unico mezzo di ingresso da Internet verso l'azienda.
- IPSec si situa sotto il livello di trasporto (TCP, UDP) e pertanto risulta trasparente alle applicazioni. Non vi sarà alcuna necessità di cambiare il software sul sistema dell'utente o sul server a causa dell'implementazione di IPSec nel firewall o nel router. Anche se IPSec viene implementato nei sistemi finali, il software di livello superiore, comprese le applicazioni, non verrà interessato dalla modifica.
- IPSec può essere trasparente agli utenti finali. Non vi è alcuna necessità di addestrare gli utenti sui meccanismi di sicurezza, di rilasciare informazioni sulle chiavi per gli utenti o di revocare tali informazioni quando gli utenti lasciano l'azienda.
- IPSec può garantire anche la sicurezza dei singoli utenti. Questa soluzione è utile per coloro che lavorano fuori sede e per configurare una sottorete virtuale sicura all'interno di un'azienda per le applicazioni riservate.

Le applicazioni di routing

Oltre a supportare gli utenti finali e a proteggere i sistemi e le reti aziendali, IPSec può giocare un ruolo fondamentale nell'architettura di routing necessaria per l'interconnessione fra reti. Come descritto in [HUIT98], IPSec può garantire le seguenti funzionalità.

- Che il messaggio di pubblicizzazione di un router (un nuovo router che notifica la propria presenza) provenga da un router autorizzato.
- Che una pubblicizzazione neighbor (un router che cerca di stabilire o mantenere una relazione di "vicinanza" con un router di un altro dominio) provenga da un router autorizzato.
- Che un messaggio di redirezione provenga dal router al quale è stato inviato il pacchetto iniziale.
- Che l'aggiornamento delle informazioni di routing non venga falsificato.

Senza questi meccanismi di sicurezza, un estraneo potrebbe pregiudicare le comunicazioni o dirottare una parte del traffico. I protocolli di routing come OSPF dovrebbero operare sopra le associazioni di sicurezza fra i router definiti da IPSec.

16.2 L'architettura di IPSec

Le specifiche di IPSec sono diventate piuttosto complesse. Per avere un'idea globale dell'architettura, si inizierà a indicare i documenti che definiscono IPSec. Poi si affronteranno i servizi di IPSec e si introdurrà il concetto di associazione di sicurezza.

I documenti di IPSec

Le specifiche di IPSec sono costituite da numerosi documenti. I più importanti di questi, emessi nel novembre del 1998 sono i documenti RFC 2401, 2402, 2406 e 2408.[1]

- RFC 2401: Panoramica dell'architettura di sicurezza.
- RFC 2402: Descrizione dell'estensione per l'autenticazione dei pacchetti in IPv4 e in IPv6.
- RFC 2406: Descrizione dell'estensione per la crittografia dei pacchetti in IPv4 e in IPv6.
- RFC 2408: Specifiche delle funzionalità di gestione delle chiavi.

Il supporto di queste funzionalità è obbligatorio in IPv6 e opzionale in IPv4. In entrambi i casi, le funzionalità di sicurezza sono implementate come *extension header* che seguono l'intestazione principale IP. L'extension header per l'autenticazione è chiamata Authentication Header; quella per la crittografia è chiamata ESP (Encapsulating Security Payload). Oltre a questi quattro documenti RFC, il gruppo IP Security Protocol Working Group di IETF ha pubblicato molte altre bozze di documenti. I documenti sono suddivisi in sette gruppi, come indicato nella Figura 16.2 (RFC 2401).

- **Architettura:** descrive i concetti generali, i requisiti di sicurezza, le definizioni e i meccanismi che definiscono la tecnologia IPSec.
- **ESP (Encapsulating Security Payload):** descrive il formato del pacchetto e altri elementi generali relativi all'uso di ESP per la crittografia e, opzionalmente, l'autenticazione.
- **AH (Authentication Header):** descrive il formato del pacchetto e gli elementi generali relativi all'autenticazione del pacchetto.
- **Algoritmi di crittografia:** documenti che descrivono il modo in cui i vari algoritmi di crittografia vengono utilizzati per ESP.
- **Algoritmi di autenticazione:** documenti che descrivono il modo in cui i vari algoritmi di autenticazione vengono utilizzati per AH e per l'opzione di autenticazione di ESP.
- **Gestione delle chiavi:** documenti che descrivono i meccanismi di gestione delle chiavi.
- **DOI (Domain of Interpretation):** contiene i valori che fanno riferimento alle relazioni fra i documenti. Fra questi vi sono gli identificatori per gli algoritmi di crittografia e autenticazione approvati, più vari parametri operativi, per esempio la durata delle chiavi.

[1] Nel Dicembre 2005 l'IETF ha rilasciato aggiornamenti importanti a questi RFC. I nuovi documenti sono, rispettivamente: RFC 4301, 4302, 4303, 4306. Nonostante il presente testo sia basato sui documenti standard precedenti, in quanto essi sono i più implementati al momento, si rimanda il lettore ai nuovi RFC per la comprensione di come la suite protocollare di IPSec evolverà nel futuro.

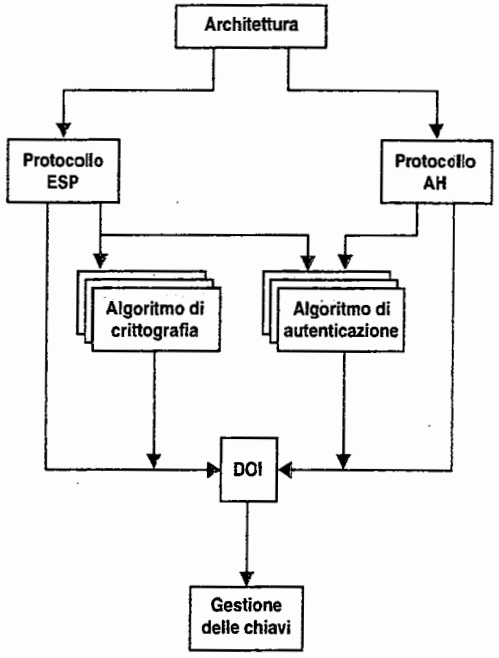


Figura 16.2 Panoramica dei documenti IPSec.

I servizi di IPSec

IPSec fornisce i servizi di sicurezza a livello IP consentendo a un sistema di selezionare i protocolli di sicurezza richiesti, di determinare gli algoritmi da utilizzare per i servizi e di gestire le chiavi crittografiche necessarie per fornire i servizi richiesti. Per garantire la sicurezza vengono impiegati due protocolli: un protocollo di autenticazione stabilito dall'intestazione del protocollo, AH (Authentication Header), e un protocollo combinato di crittografia/autenticazione stabilito dal formato del pacchetto per tale protocollo, ESP (Encapsulating Security Payload). I servizi offerti sono i seguenti:

- controllo degli accessi;
- integrità dei dati con protocolli senza connessione;
- autenticazione dell'origine dei dati;
- rifiuto dei pacchetti a replay (una forma di integrità parziale della sequenza);
- segretezza (crittografia);
- segretezza parziale del flusso di traffico.

La Tabella 16.1 mostra i servizi offerti dai protocolli AH e ESP. Per ESP, vi sono due casi: con e senza opzione di autenticazione. Sia AH che ESP consentono il controllo degli accessi sulla base della distribuzione delle chiavi crittografiche e della gestione dei flussi di traffico relativi a questi protocolli di sicurezza.

Tabella 16.1 I servizi offerti da IPSec.

	AH	ESP (solo crittografia)	ESP (crittografia e autenticazione)
Controllo degli accessi	✔	✔	✔
Integrità senza connessione	✔		✔
Autenticazione dell'origine dei dati	✔		✔
Rifiuto dei pacchetti a replay	✔	✔	✔
Segretezza		✔	✔
Segretezza parziale del flusso di traffico		✔	✔

Le associazioni di sicurezza

Un concetto chiave nei meccanismi di autenticazione e segretezza IP è quello di associazione di sicurezza (SA – Security Association). Un'associazione è una relazione monodirezionale fra un mittente e un destinatario che riguarda i servizi di sicurezza relativi al traffico trasportato.

Se è richiesto uno scambio sicuro bidirezionale sono necessarie due associazioni di sicurezza. I servizi di sicurezza sono assegnati a un'associazione di sicurezza per l'impiego da parte dei protocolli AH o ESP ma non di entrambi.

Un'associazione di sicurezza è identificata univocamente da tre parametri.

- **SPI (Security Parameters Index):** stringa di bit assegnata all'associazione di sicurezza e con significato esclusivamente locale. Il valore SPI è trasportato nelle intestazioni AH e ESP per consentire al sistema di destinazione di selezionare l'associazione di sicurezza in base alla quale elaborare il pacchetto ricevuto.
- **Indirizzo IP di destinazione:** attualmente è consentito solo l'impiego di indirizzi unicast: l'indirizzo della destinazione finale dell'associazione di sicurezza; ovvero il sistema di un utente finale o un sistema di rete come un firewall o un router.
- **Identificatore del protocollo di sicurezza:** indica se l'associazione di sicurezza è di tipo AH o ESP.

Pertanto in ogni pacchetto IP[2], l'associazione di sicurezza è identificata univocamente dall'indirizzo di destinazione nell'intestazione IPv4 o IPv6 e dal parametro SPI nell'intestazione di estensione (AH o ESP).

[2] In questo capitolo il termine pacchetto IP fa riferimento a un datagram IPv4 o a un pacchetto IPv6.

I parametri di un'associazione di sicurezza

In ogni implementazione di IPSec esiste un Security Association Database nominale[3] che definisce i parametri relativi a ciascuna associazione di sicurezza. Un'associazione di sicurezza è definita normalmente dai seguenti parametri.

- **Sequence Number Counter:** valore a 32 bit utilizzato per generare il campo Sequence Number nell'intestazione AH o ESP (obbligatorio per tutte le implementazioni) descritta nel Paragrafo 16.3.
- **Sequence Counter Overflow:** flag che indica se un overflow del campo Sequence Number Counter deve generare un evento di audit e impedire l'ulteriore trasmissione di pacchetti su questa associazione di sicurezza (obbligatorio per tutte le implementazioni).
- **Anti-Replay Window:** utilizzato per determinare se un pacchetto AH o ESP in ingresso è un replay; descritto nel Paragrafo 16.3 (obbligatorio per tutte le implementazioni).
- **AH Information:** algoritmo di autenticazione, chiavi, durata delle chiavi e altri parametri correlati utilizzati con AH (obbligatorio per le implementazioni AH).
- **ESP Information:** algoritmo di crittografia e autenticazione, chiavi, valori di inizializzazione, durata delle chiavi e parametri relativi utilizzati con ESP (obbligatorio per le implementazioni ESP).
- **Lifetime of This Security Association:** intervallo di tempo, o conteggio in byte, trascorso il quale un'associazione di sicurezza deve essere sostituita da una nuova associazione di sicurezza (e un nuovo SPI) o chiusa, più l'indicazione dell'azione che da eseguire alla scadenza (obbligatorio per tutte le implementazioni).
- **IPSec Protocol Mode:** tunnel, trasporto o wildcard (obbligatorio per tutte le implementazioni). Queste modalità verranno trattate più avanti in questa stessa parte del capitolo.
- **Path MTU:** massima unità di trasmissione sul percorso (dimensioni massime del pacchetto che può essere trasmesso senza frammentazione) e variabili relativi alla durata di validità (obbligatorio per tutte le implementazioni).

Il meccanismo di gestione della chiave utilizzato per distribuire le chiavi è affiancato da meccanismi di autenticazione e privacy unicamente tramite il Security Parameters Index. Pertanto i meccanismi di autenticazione e privacy risultano indipendenti da qualsiasi meccanismo specifico di gestione della chiave.

Selettori delle associazioni di sicurezza

IPSec fornisce all'utente una notevole flessibilità sul modo in cui i servizi IPSec vengono applicati al traffico IP. Come si vedrà più avanti, le associazioni di sicurezza (SA) possono essere combinate in vari modi per fornire la configurazione desiderata. Inoltre IPSec fornisce un elevato livello di granularità nella discriminazione fra il traffico soggetto alla protezione IPSec e il traffico non soggetto.

Il traffico IP viene associato alle SA tramite il SPD (Security Policy Database). Il traffico che può aggirare IPSec non viene associato ad alcuna SA. Nella sua forma più semplice, un database SPD contiene delle voci, ognuna delle quali definisce un sottoinsieme del traffico IP e punta a un'associazione di sicurezza per tale traffico. In ambienti più complessi, vi possono essere più voci potenzialmente correlate alla medesima associazione di sicurezza o più associazioni di sicurezza relative a una sola voce del database. Per una discussione più approfondita si rimanda ai documenti specifici su IPSec.

Ogni voce del database SPD è definita da un insieme di campi IP e di livello superiore chiamati *selettori*. In pratica i selettori consentono di filtrare il traffico in uscita per associarlo a una determinata associazione di sicurezza. L'elaborazione in uscita di ciascun pacchetto IP procede secondo la seguente sequenza generale.

1. Confronta i valori dei campi appropriati del pacchetto (i campi del selettore) con quelli del database SPD per trovare la voce corrispondente che punta a zero o più associazioni di sicurezza.
2. Determina l'eventuale associazione di sicurezza relativa a questo pacchetto e il corrispondente SPI.
3. Svolge l'elaborazione IPSec richiesta (ovvero l'elaborazione AH o ESP).

Una voce SPD è determinata dai seguenti selettori.

- **Destination IP Address:** può trattarsi di un singolo indirizzo IP, di un elenco o di un intervallo di indirizzi o di un insieme di indirizzi specificati con una maschera. Questi ultimi due sono necessari per supportare le situazioni in cui più sistemi di destinazione condividono la stessa associazione di sicurezza (per esempio quando i sistemi si trovano dietro un firewall).
- **Source IP Address:** può trattarsi di un unico indirizzo IP, di un elenco o di un intervallo di indirizzi o di un insieme di indirizzi specificati con una maschera. Le ultime due forme di indirizzamento sono necessarie per supportare le situazioni in cui più sistemi di origine condividono la stessa associazione di sicurezza (per esempio dietro un firewall).
- **UserID:** identificatore dell'utente ottenuto dal sistema operativo. Non si tratta di un campo IP o di livelli superiori ma è disponibile se IPSec utilizza lo stesso sistema operativo dell'utente.
- **Data Sensitivity Level:** usato per i sistemi che gestiscono la sicurezza del flusso di informazioni (per esempio Secret o Unclassified).
- **Transport Layer Protocol:** ottenuto dai campi IPv4 Protocol o IPv6 Next Header. Può essere un singolo numero di protocollo, un elenco di numeri di protocolli o un intervallo di numeri di protocolli.
- **Source Ports e Destination Ports:** può trattarsi di singole porte TCP o UDP, di un elenco di porte o di un'indicazione a maschera delle porte.

Le modalità *transport* e *tunnel*

AH e ESP supportano due modalità di funzionamento: transport e tunnel. È più facile comprendere il funzionamento di queste due modalità nel contesto della descrizione di AH e ESP trattati rispettivamente nei Paragrafi 16.3 e 16.4. Qui verrà offerta solo una breve panoramica.

[3] Con "nominale" si intende il fatto che la funzionalità fornita dal Security Association Database deve essere disponibile in ogni implementazione IPSec, ma il modo in cui essa viene fornita è lasciato all'implementazione.

La modalità transport

La modalità transport fornisce fondamentalmente la protezione dei protocolli di livello superiore. In altre parole la protezione della modalità transport riguarda il payload del pacchetto IP. Fra gli esempi vi sono i segmenti TCP o UDP o i pacchetti ICMP che operano direttamente sopra il livello IP nello stack di protocolli dell'host. In genere la modalità transport viene utilizzata per le comunicazioni end-to-end fra due host (per esempio un client e un server o due workstation). Quando un host utilizza AH o ESP su IPv4, il payload è rappresentato dai dati che normalmente seguono l'intestazione IP. Per IPv6, il payload è rappresentato dai dati che normalmente seguono sia l'intestazione IP che eventuali intestazioni di estensione IPv6 presenti, con la possibile eccezione dell'intestazione delle opzioni della destinazione, che può essere inclusa nella protezione.

ESP in modalità transport esegue la crittografia e, opzionalmente, autentica il payload IP ma non l'intestazione IP. AH in modalità transport autentica il payload IP e determinate parti dell'intestazione IP.

Modalità tunnel

La modalità tunnel fornisce la protezione dell'intero pacchetto IP. Per ottenere ciò, dopo che al pacchetto IP sono stati aggiunti i campi AH o ESP, l'intero pacchetto e i campi di sicurezza vengono trattati come payload del nuovo pacchetto IP "esterno" con una nuova intestazione IP esterna. L'intero pacchetto originario, quello interno, viaggia in una sorta di "tunnel" da un punto all'altro della rete IP; nessun router lungo il percorso sarà in grado di esaminare l'intestazione IP interna. Poiché il pacchetto originale è incapsulato, il nuovo pacchetto esterno può avere indirizzi di destinazione e di origine completamente differenti, in modo da migliorare ulteriormente la sicurezza. La modalità tunnel viene utilizzata quando una o entrambe le estremità di un'associazione di sicurezza sono costituite da gateway di sicurezza, per esempio un firewall o un router che implementa IPSec. Con la modalità tunnel, gli host posti su reti protette da firewall possono intrattenere comunicazioni sicure senza implementare IPSec. I pacchetti non protetti generati da questi host vengono convogliati in un "tunnel" attraverso le reti esterne grazie alle associazioni di sicurezza in modalità tunnel, configurate dal software IPSec nel firewall o nel router di sicurezza che si trova al confine della rete locale.

Ecco un esempio di funzionamento della modalità tunnel di IPSec. L'host A di una rete genera un pacchetto IP con l'indirizzo di destinazione dell'host B che si trova su un'altra rete. Questo pacchetto viene indirizzato dall'host di origine a un firewall o a un router di sicurezza posto ai confini della rete di A. Il firewall filtra tutti i pacchetti in uscita per determinare la necessità di elaborazioni IPSec. Se questo pacchetto da A a B richiede IPSec, il firewall svolge l'elaborazione IPSec e incapsula il pacchetto in un'intestazione IP esterna. L'indirizzo IP di origine di questo pacchetto IP esterno è questo firewall e l'indirizzo di destinazione deve essere il firewall posto sul confine della rete locale di B. Questo pacchetto viene poi instradato al firewall di B tramite i router intermedi che esaminano solo l'intestazione IP esterna. Giunto al firewall di B, l'intestazione IP esterna del pacchetto viene eliminata e a B viene consegnato il pacchetto interno.

ESP in modalità tunnel esegue la crittografia e, opzionalmente, l'autenticazione, dell'intero pacchetto IP interno, compresa l'intestazione IP interna. AH in modalità tunnel autentica l'intero pacchetto IP interno e determinate porzioni dell'intestazione IP esterna. La Tabella 16.2 riepiloga le funzionalità delle modalità transport e tunnel.

16.3 Authentication Header

Authentication Header (AH) fornisce il supporto per l'integrità dei dati e per l'autenticazione dei pacchetti IP. La funzionalità di integrità dei dati garantisce che non sia possibile modificare il contenuto del pacchetto in transito senza che tale operazione venga rilevata. La funzionalità di autenticazione consente a un sistema terminale o a un dispositivo di rete di autenticare l'utente o l'applicazione e filtrare il traffico di conseguenza. Inoltre impedisce gli attacchi a indirizzo IP fasullo (IP spoofing), attualmente molto diffusi in Internet. Infine Authentication Header mette al riparo dagli attacchi a replay di cui si parlerà più avanti in questa parte del capitolo.

L'autenticazione si basa su un codice MAC (Message Authentication Code) come descritto nel Capitolo 11, pertanto le due parti devono condividere una chiave segreta. L'intestazione Authentication Header è costituita dai seguenti campi (Figura 16.3).

- **Next Header (8 bit):** identifica il tipo di intestazione che segue immediatamente questa intestazione.
- **Payload Length (8 bit):** lunghezza di Authentication Header in word di 32 bit, meno 2. Per esempio, la lunghezza standard del campo dati di autenticazione è di 96 bit ovvero tre word di 32 bit. Con un'intestazione fissa di tre word, nell'intestazione vi saranno in totale 6 word e il campo Payload Length conterrà il valore 4.
- **Reserved (16 bit):** riservato per utilizzi futuri.
- **Security Parameters Index (32 bit):** identifica un'associazione di sicurezza.
- **Sequence Number (32 bit):** un contatore incrementale monotonico di cui si parlerà più avanti.

Tabella 16.2 Le funzionalità delle modalità transport e tunnel.

	Modalità transport	Modalità tunnel
AH	Autentica il payload IP e determinate parti dell'intestazione IP e delle intestazioni di estensione IPv6.	Autentica l'intero pacchetto IP interno (intestazione interna più payload IP) più determinate parti dell'intestazione IP esterno e delle intestazioni di estensione IPv6 esterna.
ESP	Esegue la crittografia del payload IP e di ogni intestazione di estensione IPv6 che segue l'intestazione ESP.	Esegue la crittografia del pacchetto IP interno.
ESP con autenticazione	Esegue la crittografia del payload IP e di ogni intestazione di estensione IPv6 che segue l'intestazione ESP. Autentica il payload IP ma non l'intestazione IP.	Esegue la crittografia del pacchetto IP interno. Autentica il pacchetto IP interno.

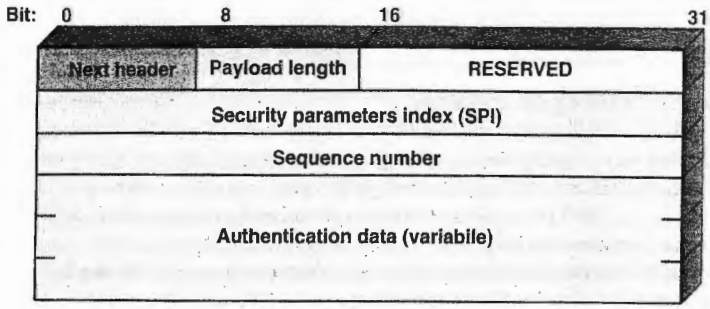


Figura 16.3 Authentication Header di IPSec.

- **Authentication Data (variabile):** un campo di lunghezza variabile (deve essere però un multiplo intero di word di 32 bit) che contiene il valore ICV (Integrity Check Value) o MAC di questo pacchetto; se ne parlerà più avanti.

Il servizio anti-replay

In un attacco a replay, un estraneo ottiene una copia di un pacchetto autenticato che trasmette successivamente alla destinazione prevista. La ricezione di pacchetti IP autenticati duplicati può disturbare il funzionamento del servizio o può avere altre conseguenze indesiderate. Il campo Sequence Number ha lo scopo di sventare questo genere di attacchi. Innanzitutto si parlerà della generazione del numero sequenziale da parte del mittente e poi si vedrà come questo valore venga elaborato dal destinatario.

Quando viene stabilita una nuova associazione di sicurezza, il **mittente** inizializza un contatore a 0. Ogni volta che un pacchetto viene inviato in questa associazione di sicurezza, il mittente incrementa il contatore e inserisce il valore nel campo Sequence Number. Pertanto il primo valore da usare è 1. Se è attiva la difesa contro gli attacchi a replay (impostazione predefinita), il mittente non deve consentire che il numero di sequenza ricominci ciclicamente dopo $2^{32} - 1$ tornando a 0. Altrimenti vi sarebbero più pacchetti validi con lo stesso numero di sequenza. Se viene raggiunto il limite $2^{32} - 1$, il mittente dovrà chiudere questa associazione di sicurezza e negoziare una nuova associazione con una nuova chiave.

Poiché il protocollo IP offre un servizio senza connessione non affidabile, non garantisce che i pacchetti vengano consegnati in ordine e non garantisce neppure che vengano consegnati tutti i pacchetti. Pertanto il documento di autenticazione IPSec stabilisce che il **ricevitore** implementi una finestra di dimensioni W, normalmente $W = 64$. Il margine destro della finestra rappresenta il numero sequenziale più elevato, N, ricevuto finora per un pacchetto valido. Per ogni pacchetto con numero sequenziale compreso fra $N - W + 1$ e N ricevuto correttamente (ovvero autenticato correttamente), viene contrassegnata la posizione corrispondente nella finestra (vedere la Figura 16.4). Quando il pacchetto viene ricevuto, l'elaborazione si svolge nel seguente modo.

1. Se il pacchetto ricevuto rientra nella finestra ed è nuovo, viene controllato il codice MAC. Se il pacchetto è autenticato, viene contrassegnata la posizione corrispondente nella finestra.
2. Se il pacchetto ricevuto si trova a destra della finestra ed è nuovo, viene controllato il codice MAC. Se il pacchetto è autenticato, la finestra viene fatta avanzare in modo che questo numero di sequenza rappresenti il nuovo margine destro della finestra e quindi viene contrassegnata la posizione corrispondente nella finestra.
3. Se il pacchetto ricevuto si trova a sinistra della finestra o se non passa l'autenticazione, il pacchetto viene eliminato; si tratta di un evento registrabile ai fini dell'eventuale auditing.

L'Integrity Check Value

Il campo Authentication Data contiene un valore chiamato Integrity Check Value, un codice di autenticazione del messaggio o una versione troncata di un codice prodotto da un algoritmo MAC. Le specifiche correnti stabiliscono che un'implementazione compatibile debba supportare:

- HMAC-MD5-96
- HMAC-SHA-1-96

Entrambi usano l'algoritmo HMAC, il primo con codice hash MD5 e il secondo con codice hash SHA-1 (questi algoritmi sono descritti nel Capitolo 12). In entrambi i casi viene calcolato il valore HMAC completo ma questo valore viene poi troncato utilizzando i primi 96 bit che rappresentano la lunghezza standard del campo Authentication Data.

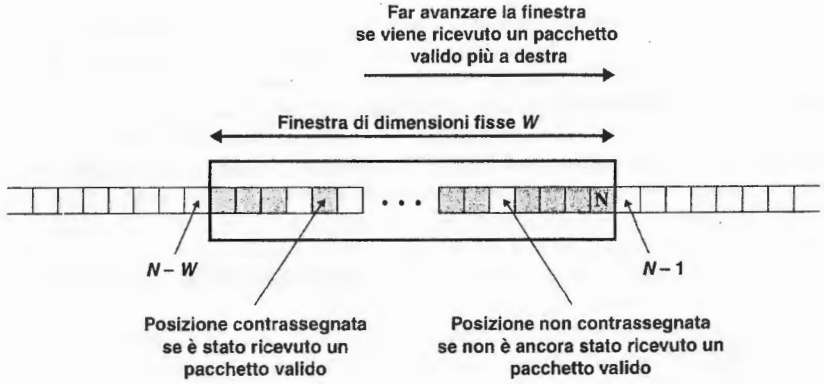


Figura 16.4 Il meccanismo anti-replay.

Il codice MAC viene calcolato in base ai seguenti elementi.

- I campi dell'intestazione IP che non cambiano durante la trasmissione (sono immutabili) o che hanno un valore prevedibile all'arrivo al punto terminale dell'associazione di sicurezza AH. I campi che possono invece cambiare durante il transito e il cui valore all'arrivo è imprevedibile, vengono considerati a 0 per poter svolgere gli stessi calcoli sia all'origine che alla destinazione.
- L'intestazione AH, tranne il campo Authentication Data. Il campo Authentication Data viene considerato a 0 per i calcoli sia all'origine che alla destinazione.
- Tutti i dati del protocollo di livello superiore che si presuppone siano immutabili nel transito (per esempio un segmento TCP o un pacchetto IP interno in modalità tunnel).

Per IPv4, fra i campi immutabili vi sono Internet Header Length e Source Address. Un esempio di campo che può cambiare ma è prevedibile è Destination Address (con routing lasco o rigido). Tra gli esempi di campi mutabili che vengono azzerati prima del calcolo ICV vi sono Time to Live e Header Checksum. Si noti che i campi degli indirizzi di origine e destinazione vengono protetti per evitare l'IP spoofing.

Tra gli esempi nell'intestazione di base IPv6 vi sono Version (immutabile), Destination Address (mutevole ma prevedibile) e Flow Label (mutevole e quindi azzerato per il calcolo).

Le modalità transport e tunnel

La Figura 16.5 mostra due modi in cui può essere utilizzato il servizio di autenticazione di IPSec. In un caso l'autenticazione viene fornita direttamente fra il server e le workstation client. La workstation può essere sulla stessa rete del server o su una rete esterna. Se la workstation e il server condividono una chiave segreta protetta, il processo di autenticazione è sicuro. Questo caso utilizza un'associazione di sicurezza in modalità transport. Nell'altro caso, una workstation remota si autentica presso il firewall aziendale per accedere all'intera rete interna o perché il server richiesto non supporta la funzionalità di autenticazione. Questo caso usa l'associazione di sicurezza in modalità tunnel.

In questa parte del capitolo si parlerà del raggio d'azione dell'autenticazione garantita da AH e della posizione dell'intestazione di autenticazione per le due modalità. Le considerazioni sono leggermente differenti per IPv4 e IPv6. La Figura 16.6A mostra dei tipici pacchetti IPv4 e IPv6. In questo caso il payload IP è un segmento TCP, ma potrebbe anche trattarsi di unità dati di qualsiasi altro protocollo che utilizza IP, per esempio UDP o ICMP.

Per **AH** in modalità transport in IPv4, l'intestazione AH viene inserita dopo l'intestazione IP originaria e prima del payload IP (per esempio un segmento TCP); questa situazione è rappresentata nella parte superiore della Figura 16.6B. L'autenticazione copre l'intero pacchetto, esclusi i campi mutevoli dell'intestazione IPv4 che vengono impostati a zero per il calcolo del codice MAC.

Nel contesto di IPv6, l'intestazione AH viene considerata come un payload end-to-end, ovvero non viene esaminata o elaborata dai router intermedi. Pertanto l'intestazione AH viene specificata dopo l'intestazione base IPv6 e le extension header hop-by-hop, routing e fragment. L'extension header delle opzioni di destinazione può trovarsi prima o dopo l'intestazione AH a seconda della semantica desiderata. In ogni caso l'autenticazione co-

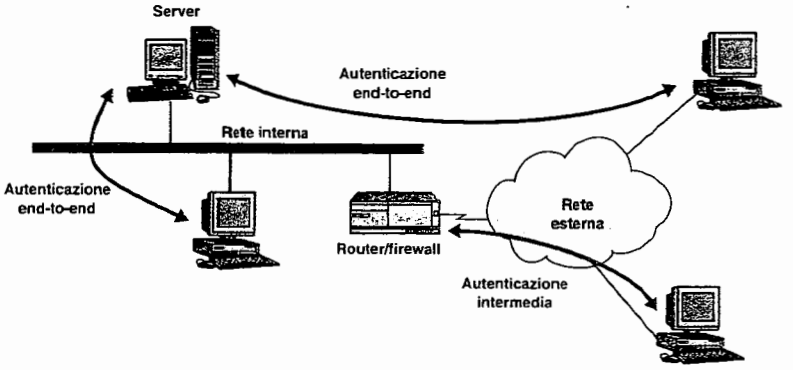


Figura 16.5 Confronto fra autenticazione end-to-end e intermedia.

pre l'intero pacchetto, ad eccezione dei campi mutevoli che vengono impostati a zero per il calcolo del codice MAC.

Per **AH** in modalità tunnel, viene autenticato l'intero pacchetto IP di origine e l'intestazione AH viene inserita fra l'intestazione di origine IP e la nuova intestazione IP esterna (Figura 16.6C). L'intestazione IP interna trasporta gli indirizzi di origine e di destinazione finali mentre l'intestazione IP esterna può contenere indirizzi IP differenti (per esempio gli indirizzi dei firewall o di altri gateway di sicurezza).

Con la modalità tunnel, l'intero pacchetto IP interno, compresa l'intestazione IP interna completa, risulta protetto dal AH. L'intestazione IP esterna (e nel caso di IPv6, le intestazioni di estensione IP esterne) è protetta ad eccezione dei campi mutevoli e imprevedibili.

16.4 Encapsulating Security Payload

Encapsulating Security Payload (ESP) fornisce servizi di segretezza, fra cui la segretezza del contenuto del messaggio e una parziale segretezza del flusso di traffico. Opzionalmente ESP può fornire un servizio di autenticazione.

Il formato dei pacchetti ESP

La Figura 16.7 mostra il formato di un pacchetto ESP. Il pacchetto contiene i seguenti campi.

- **Security Parameters Index (32 bit):** identifica un'associazione di sicurezza.
- **Sequence Number (32 bit):** contatore incrementale monotonico utilizzato per la funzione anti-replay, come già visto per l'intestazione AH.
- **Payload Data (variabile):** il segmento di trasporto dei dati (modalità transport) o il pacchetto IP (modalità tunnel) protetto dalla crittografia.

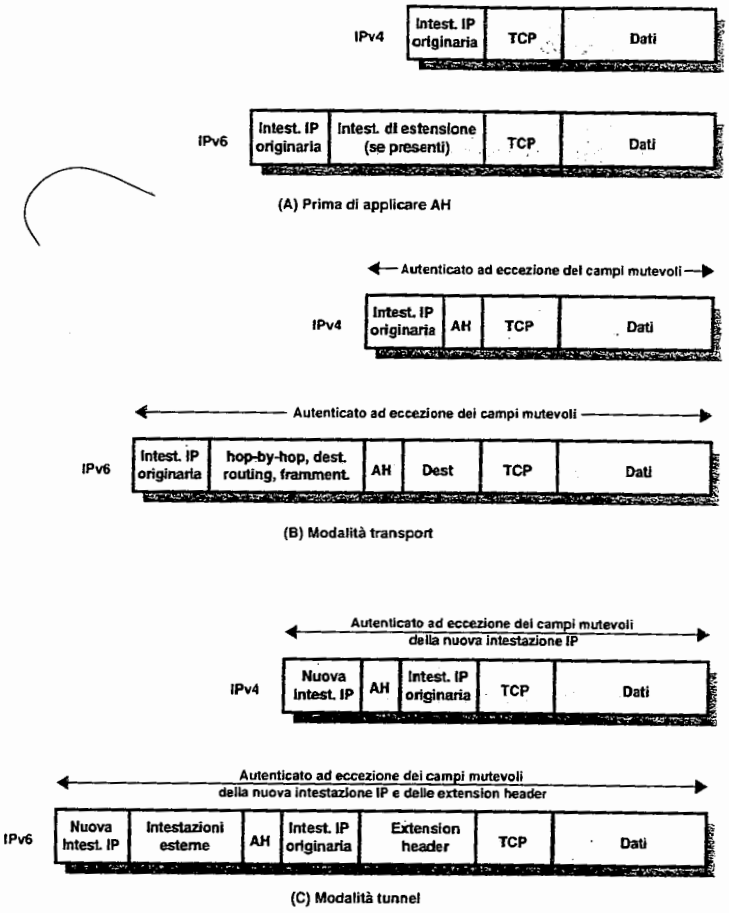


Figura 16.6 Autenticazione AH.

- **Padding (0-255 byte):** lo scopo di questo campo verrà trattato più avanti.
- **Pad Length (8 bit):** indica il numero di byte di riempimento che precedono questo campo.
- **Next Header (8 bit):** identifica il tipo di dati contenuti nel campo Payload Data identificando la prima intestazione contenuta (per esempio un'extension header IPv6 o un protocollo di livello superiore come TCP).
- **Authentication Data (variabile):** campo di lunghezza variabile (ma costituito da un numero intero di word di 32 bit) che contiene il valore Integrity Check Value calcolato sul pacchetto ESP, meno il campo Authentication Data.

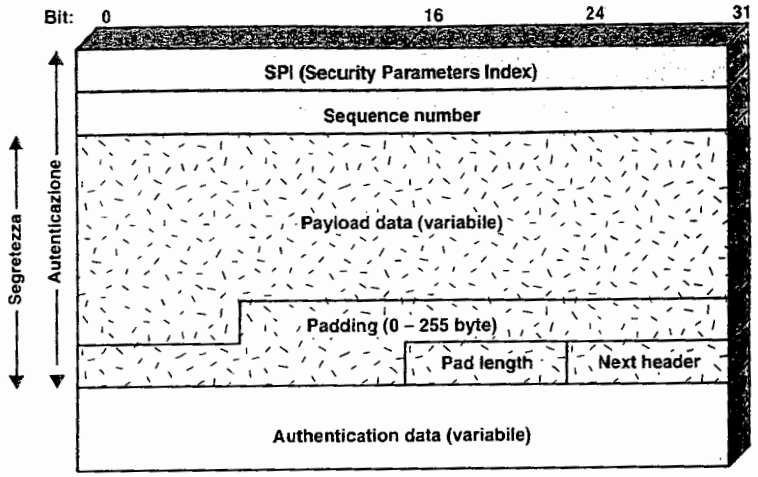


Figura 16.7 Il formato di un pacchetto ESP in IPSec.

Gli algoritmi di crittografia e autenticazione

I campi Payload Data, Padding, Pad Length e Next Header sono crittografati dal servizio ESP. Se l'algoritmo utilizzato per crittografare il payload richiede dei dati di sincronizzazione crittografica, per esempio un vettore di inizializzazione (IV), questi dati devono essere trasportati esplicitamente all'inizio del campo Payload Data. Se incluso, il vettore di inizializzazione non viene normalmente crittografato, sebbene spesso si consideri parte del testo cifrato.

Le specifiche correnti dicono che un'implementazione compatibile deve supportare l'algoritmo DES in modalità CBC (Cipher Block Chaining) di cui si è parlato nel Capitolo 3. Nel documento DOI sono stati assegnati anche gli identificatori di alcuni altri algoritmi che possono essere utilizzati per la crittografia, fra cui:

- Triple DES a tre chiavi
- RC5
- IDEA
- Triple IDEA a tre chiavi
- CAST
- Blowfish

Molti di questi algoritmi sono stati descritti nel Capitolo 6.

Anche ESP (come AH) supporta l'uso di un codice MAC con una lunghezza standard di 96 bit. Sempre come AH, le specifiche correnti dicono che un'implementazione compatibile deve supportare HMAC-MD5-96 e HMAC-SHA-1-96.

Padding

Il campo Padding ha molte funzioni.

- Se un algoritmo di crittografia richiede che il testo in chiaro sia un multiplo di un determinato numero di byte (per esempio il multiplo di un blocco per la cifratura a blocchi), il campo Padding consente di espandere il testo in chiaro (costituito dai campi Payload Data, Padding, Pad Length e Next Header) fino a raggiungere la lunghezza desiderata.
- Il formato di ESP richiede che i campi Pad Length e Next Header siano allineati a destra all'interno di una word di 32 bit. In modo equivalente, il testo cifrato deve essere un multiplo intero di 32 bit. Il campo Padding consente di garantire questo allineamento.
- Può essere previsto un ulteriore riempimento per offrire un servizio di segretezza parziale del flusso di traffico, celando l'effettiva lunghezza del payload.

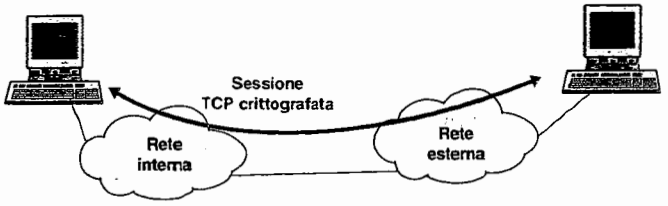
Le modalità transport e tunnel

La Figura 16.8 mostra due modalità di impiego del servizio ESP di IPSec. Nella parte superiore della figura, la crittografia (e opzionalmente l'autenticazione) è impiegata direttamente fra i due host. La Figura 16.8B mostra come la modalità tunnel possa essere utilizzata per creare una rete privata virtuale. In questo esempio un'azienda ha quattro reti private interconnesse tramite Internet. Gli host delle reti interne usano Internet per trasportare i dati ma senza interagire con altri host in Internet. Facendo arrivare i tunnel nei gateway di sicurezza di ciascuna rete interna, questa configurazione consente di evitare di implementare negli host le funzionalità di sicurezza. La prima tecnica è supportata da un'associazione di sicurezza in modalità transport mentre l'ultima tecnica utilizza la modalità tunnel. In questa parte del capitolo si vedrà l'impiego delle due modalità ESP. Le considerazioni sono per certi versi differenti per IPv4 e IPv6. Come si è visto nella discussione per AH, si utilizzerà come punto di partenza il formato dei pacchetti rappresentato nella Figura 16.6A.

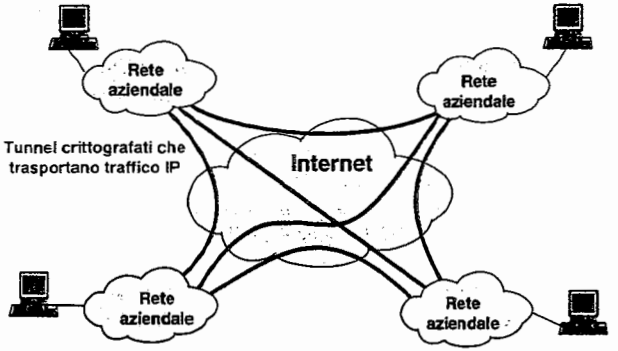
ESP in modalità transport

La modalità transport di ESP consente di crittografare e opzionalmente autenticare i dati trasportati da IP (per esempio un segmento TCP) come indicato nella Figura 16.9A. Per questa modalità in IPv4, l'intestazione ESP viene inserita nel pacchetto IP immediatamente prima dell'intestazione del livello di trasporto (per esempio TCP, UDP, ICMP) e dopo il pacchetto IP viene inserita una coda ESP (i campi Padding, Pad Length e Next Header); se è richiesta anche l'autenticazione, dopo la coda ESP viene aggiunto il campo ESP Authentication Data. Vengono crittografati l'intero segmento del livello di trasporto più la coda ESP. L'autenticazione copre tutto il testo cifrato più l'intestazione ESP.

Nel contesto di IPv6, ESP è considerato come un payload end-to-end, ovvero non viene né esaminato né elaborato dai router intermedi. Pertanto l'intestazione ESP si presenta dopo l'intestazione IPv6 e le extension header relative al funzionamento hop-by-hop, al routing e alla frammentazione. L'extension header delle opzioni di destinazione può trovarsi prima o dopo l'intestazione ESP, a seconda della semantica desiderata. In IPv6, la



(A) Sicurezza a livello di trasporto



(B) Una rete privata virtuale realizzata tramite la modalità tunnel

Figura 16.8 La crittografia nella modalità transport e nella modalità tunnel.

crittografia copre l'intero segmento di livello trasporto, più la coda ESP, più l'extension header delle opzioni di destinazione nel caso in cui si trovi dopo l'intestazione ESP. Anche in questo caso l'autenticazione copre il testo cifrato più l'intestazione ESP.

Il funzionamento della modalità transport può essere riepilogato nel modo seguente.

1. Alla sorgente, viene crittografato il blocco di dati costituito dalla coda ESP più l'intero segmento di livello trasporto; il testo in chiaro di questo blocco viene sostituito dal testo cifrato per formare il pacchetto IP per la trasmissione. Opzionalmente può essere aggiunta l'autenticazione.
2. Il pacchetto viene instradato alla destinazione. Ogni router intermedio deve esaminare ed elaborare l'intestazione IP più eventuali estensioni IP in chiaro ma non ha necessità di esaminare il testo cifrato.
3. Il nodo di destinazione esamina ed elabora l'intestazione IP più ogni intestazione di estensione IP in chiaro. Poi, sulla base del campo SPI dell'intestazione ESP, il nodo di

destinazione esegue la decrittografia della parte rimanente del pacchetto per recuperare il segmento di livello trasporto in chiaro.

La modalità transport fornisce funzionalità di segretezza a qualsiasi applicazione, evitando così la necessità di implementare la segretezza in ogni singola applicazione. Questa modalità di funzionamento è anche ragionevolmente efficiente in quanto aumenta solo leggermente la lunghezza del pacchetto IP. Un difetto di questa modalità è il fatto che è possibile svolgere un'analisi del traffico dei pacchetti trasmessi.

ESP in modalità tunnel

ESP in modalità tunnel viene utilizzato per crittografare l'intero pacchetto IP (Figura 16.9B). Con questa modalità, l'intestazione ESP precede il pacchetto e vengono crittografati il pacchetto più la coda ESP. Questo metodo può essere utilizzato per sventare gli attacchi ad analisi del traffico.

Poiché l'intestazione IP contiene l'indirizzo di destinazione e talvolta delle direttive di source routing e informazioni sulle opzioni hop by hop, non è possibile trasmettere semplicemente il pacchetto IP crittografato preceduto dall'intestazione ESP, altrimenti i router intermedi non sarebbero in grado di elaborare il pacchetto. Pertanto è necessarie incapsulare l'intero blocco (intestazione ESP più testo cifrato più eventuali dati Authentication Data) con una nuova intestazione IP che contiene informazioni sufficienti per svolgere le operazioni di routing ma non l'analisi del traffico.

Mentre la modalità transport è adatta per proteggere le connessioni fra host che supportano la funzionalità ESP, la modalità tunnel è utile in una configurazione che include un firewall o qualche altro tipo di gateway di sicurezza che protegge una rete fidata dalle reti esterne. In quest'ultimo caso, la crittografia si verifica solo fra gli host esterni e il gateway o fra due gateway di sicurezza. Questo evita agli host della rete interna la necessità di svolgere la crittografia e semplifica l'operazione di distribuzione delle chiavi riducendo il numero di chiavi necessarie. Inoltre impedisce le analisi del traffico basate sulla destinazione finale.

Si consideri il caso in cui un host esterno voglia comunicare con un host che si trova su una rete interna, protetto da un firewall. Si supponga che ESP sia implementato nell'host esterno e nei firewall. Ecco le operazioni per trasferire un segmento di livello trasporto dall'host esterno all'host interno.

1. La sorgente prepara un pacchetto IP (interno) con l'indirizzo di destinazione dell'host interno. Questo pacchetto viene fatto precedere da un'intestazione ESP, quindi il pacchetto e la coda ESP vengono crittografati con l'eventuale aggiunta di dati Authentication Data. Il blocco prodotto viene incapsulato in una nuova intestazione IP (l'intestazione base più le estensioni opzionali come per esempio le opzioni di routing in IPv6) il cui indirizzo di destinazione è il firewall; si forma così il pacchetto IP esterno.
2. Il pacchetto esterno viene instradato al firewall di destinazione. Ciascun router intermedio deve esaminare ed elaborare l'intestazione IP esterna più le eventuali extension header IP ma non ha necessità di esaminare il testo cifrato.
3. Il firewall di destinazione esamina ed elabora l'intestazione IP esterna più le eventuali extension header IP esterne. Poi, sulla base del campo SPI contenuto nell'intestazione ESP, il nodo di destinazione esegue la decrittografia della parte rimanente del pacchetto per recuperare il pacchetto IP interno; questo pacchetto viene poi trasmesso nella rete interna.
4. Il pacchetto interno viene inoltrato, attraverso zero o più router nella rete interna, fino a raggiungere l'host di destinazione.

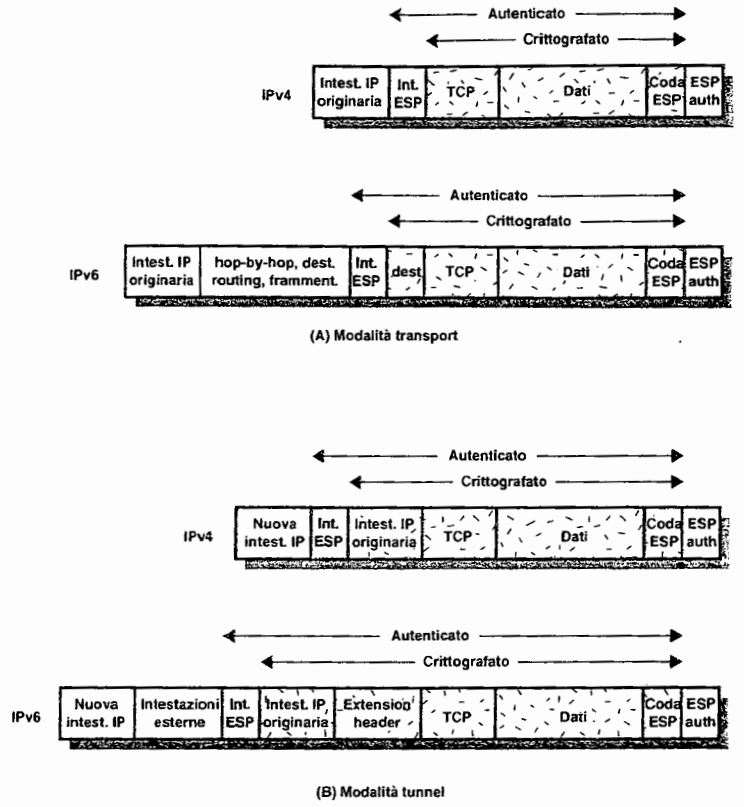


Figura 16.9 Campi nella crittografia e autenticazione ESP

16.5 Combinazione di più associazioni di sicurezza

Una singola associazione di sicurezza può specificare il protocollo AH o ESP ma non entrambi. Talvolta invece un determinato flusso di traffico richiede sia servizi forniti da AH che da ESP. Inoltre un determinato flusso di traffico può richiedere i servizi IPSec fra gli host e, per lo stesso flusso, servizi distinti fra i gateway di sicurezza come i firewall. In tutti

questi casi, per ottenere i servizi di IPSec desiderati, occorre impiegare più associazioni di sicurezza per lo stesso flusso di traffico. In questi casi si crea un gruppo di associazioni di sicurezza rispetto alle quali il traffico deve essere elaborato per fornire il particolare insieme di servizi IPSec desiderato. Le associazioni di sicurezza di questo gruppo possono terminare negli stessi punti o in punti differenti.

Le associazioni di sicurezza possono essere combinate in due modi.

- **Adiacenza di trasporto:** allo stesso pacchetto IP vengono applicati più protocolli di sicurezza senza utilizzare la modalità tunnel. Questo approccio, per combinare AH e ESP, offre un solo livello di combinazione; ulteriori nidificazioni non forniscono maggiori vantaggi in quanto l'elaborazione viene svolta su una sola istanza di IPSec: la destinazione finale.
- **Tunnel iterato:** vengono applicati più livelli di protocolli di sicurezza tramite tunnel IP. Questo approccio consente più livelli di nidificazione in quanto ciascun tunnel può avere un'origine differente o può terminare in un sito IPSec differente lungo il percorso.

I due approcci possono essere combinati, per esempio facendo in modo che una SA di trasporto fra due host compia parte del suo percorso in un tunnel SA fra gateway di sicurezza. Un problema interessante che sorge quando si considerano gruppi di associazioni di sicurezza è l'ordine in cui devono essere applicate le funzionalità di autenticazione e crittografia in una determinata coppia di punti terminali e i modi con cui svolgere questa operazione. Questo argomento verrà trattato tra breve. Poi si vedranno le varie combinazioni di associazioni di sicurezza che coinvolgono quanto meno un tunnel.

Autenticazione e segretezza

La crittografia e l'autenticazione possono essere combinate per trasmettere un pacchetto IP dotato di funzionalità di segretezza e autenticazione fra gli host. Verranno introdotti vari approcci.

ESP con opzione di autenticazione

Questo approccio è illustrato nella Figura 16.9. In questo approccio l'utente applica innanzitutto ESP ai dati che devono essere protetti e quindi aggiunge il campo dei dati di autenticazione. Esistono due possibilità.

- **ESP in modalità transport:** l'autenticazione e la crittografia si applicano al payload IP consegnato all'host mentre l'intestazione IP non viene protetta.
- **ESP in modalità tunnel:** l'autenticazione si applica all'intero pacchetto IP consegnato all'indirizzo di destinazione IP esterno (per esempio un firewall) dove viene effettuata l'autenticazione. L'intero pacchetto IP interno è protetto dal meccanismo di privacy per la consegna alla destinazione IP.

In entrambi i casi, l'autenticazione si applica al testo cifrato e non al testo in chiaro.

Adiacenza di trasporto

Un altro modo per applicare l'autenticazione dopo la crittografia consiste nell'utilizzare due associazioni di sicurezza di trasporto, dove quella interna è una associazione di sicurezza ESP e quella esterna è un'associazione di sicurezza AH. In questo caso ESP viene utilizzato senza la sua opzione di autenticazione. Poiché l'associazione di sicurezza interna è in modalità transport, la crittografia viene applicata al payload IP. Il pacchetto risultante consiste in un'intestazione IP (ed eventualmente delle intestazioni di estensione IPv6) seguita da un ESP. AH viene quindi applicato in modalità transport in modo che l'autenticazione copra l'intestazione ESP più l'intestazione originale IP (e relative estensioni) ad eccezione dei campi mutevoli. Il vantaggio di questo approccio rispetto al semplice utilizzo dell'associazione di sicurezza ESP con opzione di autenticazione, consiste nel fatto che l'autenticazione copre più campi, compresi gli indirizzi IP di sorgente e di destinazione. Lo svantaggio è il sovraccarico dovuto al fatto di impiegare due associazioni di sicurezza invece di una.

Raggruppamento transport-tunnel

L'uso dell'autenticazione prima della crittografia può essere preferibile per vari motivi. Innanzitutto, poiché i dati di autenticazione sono protetti dalla crittografia, sarà impossibile intercettare il messaggio e modificare i dati di autenticazione senza che venga rilevato. In secondo luogo può essere conveniente poter memorizzare le informazioni di autenticazione insieme al messaggio per eventuali riferimenti futuri. È più comodo svolgere questa operazione se le informazioni di autenticazione si applicano al messaggio non crittografato; in caso contrario il messaggio dovrebbe essere ricrittografato ogni volta che fosse necessario verificare le informazioni di autenticazione.

Una possibilità per applicare l'autenticazione prima della crittografia fra due host prevede l'impiego di un raggruppamento costituito da un'associazione di sicurezza di trasporto AH interna e un'associazione di sicurezza a tunnel ESP esterna. In questo caso l'autenticazione viene applicata al payload IP e all'intestazione IP (e relative estensioni) ad esclusione dei campi mutevoli. Il pacchetto IP risultante viene poi elaborato in modalità tunnel da ESP; il risultato è che l'intero pacchetto interno autenticato viene crittografato e viene aggiunta una nuova intestazione IP esterna con le relative estensioni.

Combinazioni di base delle associazioni di sicurezza

Il documento IPSec Architecture elenca quattro esempi di combinazioni di associazioni di sicurezza che devono essere supportate dagli host compatibili IPSec (per esempio workstation e server) o dai gateway di sicurezza (per esempio firewall e router). Questi esempi sono illustrati nella Figura 16.10. La parte inferiore di ciascun caso della figura rappresenta la connettività fisica degli elementi; la parte superiore rappresenta la connettività logica attraverso una o più associazioni di sicurezza nidificate. Ciascuna associazione di sicurezza può essere AH o ESP. Per le associazioni di sicurezza fra host, la modalità può essere di trasporto o tunnel; altrimenti si tratta della modalità tunnel.

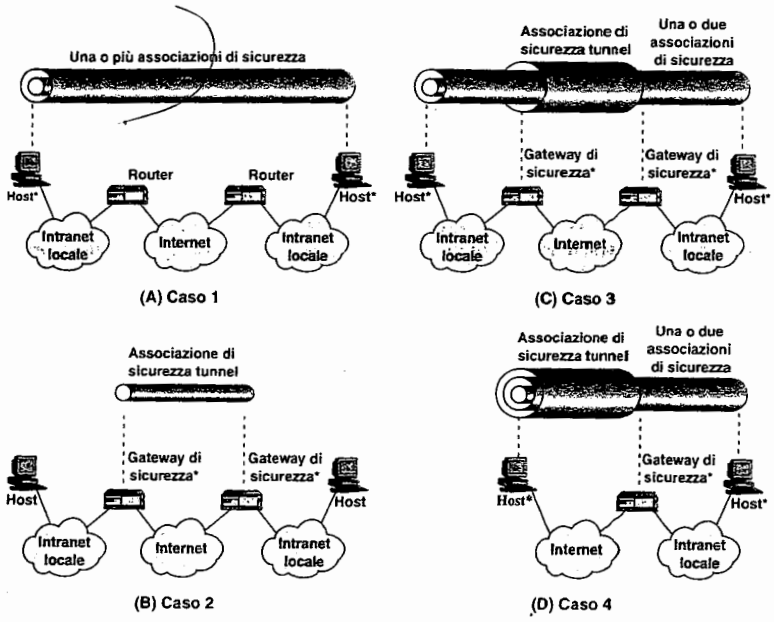


Figura 16.10 Combinazioni di base delle associazioni di sicurezza.

Nel **Caso 1** la sicurezza viene fornita dai sistemi terminali che implementano IPSec. Perché due sistemi terminali possano comunicare tramite un'associazione di sicurezza, devono condividere le chiavi segrete appropriate. Ecco alcune combinazioni possibili.

A. AH in modalità transport.
B. ESP in modalità transport.
C. AH seguito da ESP in modalità transport (un'associazione di sicurezza ESP all'interno di un'associazione di sicurezza AH).
D. A, B o C all'interno di AH o ESP in modalità tunnel.

Si è già analizzato come queste combinazioni possano essere utilizzate per supportare l'autenticazione, la crittografia, l'autenticazione prima della crittografia e l'autenticazione dopo la crittografia.

Nel **Caso 2** la sicurezza viene fornita solo fra gateway (router, firewall e così via) mentre nessun host implementa IPSec. Questo caso illustra il supporto di una semplice rete privata virtuale. Il documento di architettura della sicurezza specifica che in questo caso è necessaria un'unica associazione di sicurezza a tunnel. Il tunnel può supportare AH, ESP o ESP con opzione di autenticazione. Non è necessario utilizzare tunnel nidificati, in quanto i servizi in IPSec si applicano all'intero pacchetto interno.

Il **Caso 3** estende il Caso 2 aggiungendo la sicurezza end-to-end. Sono consentite le stesse combinazioni trattate per i Casi 1 e 2. Il tunnel gateway-gateway fornisce l'autenticazione, la segretezza o entrambe le funzionalità per tutto il traffico fra i sistemi terminali. Quando il tunnel gateway-gateway è di tipo ESP, fornisce anche una forma parziale di segretezza del traffico. I singoli host possono implementare eventuali servizi aggiuntivi IPSec necessari per determinate applicazioni o per determinati utenti, tramite associazioni di sicurezza end-to-end.

Il **Caso 4** fornisce il supporto per un host remoto che utilizza Internet per raggiungere il firewall di un'azienda per poi accedere a qualche server o workstation protetto dal firewall. Fra l'host remoto e il firewall è necessaria solo la modalità tunnel. Come nel Caso 1, fra l'host remoto e l'host locale possono essere utilizzate una o due associazioni di sicurezza.

16.6 Gestione delle chiavi

La gestione delle chiavi in IPSec comporta la scelta e la distribuzione delle chiavi segrete. In genere sono richieste quattro chiavi per le comunicazioni fra due applicazioni: due coppie di chiavi di trasmissione e ricezione sia per AH che per ESP. Il documento IPSec Architecture richiede obbligatoriamente il supporto di due tipi di gestione delle chiavi.

- **Manuale:** l'amministratore di sistema configura manualmente ciascun sistema con le proprie chiavi e con le chiavi degli altri sistemi. Si tratta di una soluzione pratica per ambienti piccoli e relativamente statici.
- **Automatica:** un sistema automatico consente la creazione su richiesta delle chiavi per le associazioni di sicurezza e ne facilita l'uso in sistemi distribuiti di grande estensione con configurazione dinamica.

Il protocollo standard per la gestione automatizzata delle chiavi per IPSec è chiamato ISAKMP/Oakley ed è costituito dai seguenti elementi.

- **Oakley Key Determination Protocol:** Oakley è un protocollo per lo scambio delle chiavi basato sull'algoritmo Diffie-Hellman, che però fornisce una sicurezza più elevata. Oakley è un protocollo generico, per il fatto che non stabilisce alcun formato specifico.
- **ISAKMP (Internet Security Association and Key Management Protocol):** ISAKMP fornisce una struttura per la gestione delle chiavi in Internet e il supporto, fra cui i formati, per la negoziazione degli attributi di sicurezza.

ISAKMP non impone un algoritmo specifico per lo scambio delle chiavi ma è piuttosto un insieme di tipi di messaggi che consentono di utilizzare vari algoritmi per lo scambio delle chiavi. Oakley era l'algoritmo per lo scambio delle chiavi obbligatorio nella versione iniziale di ISAKMP.

Si inizierà con una panoramica di Oakley per poi parlare di ISAKMP.

Il protocollo Oakley

Oakley è un raffinamento dell'algoritmo per lo scambio delle chiavi Diffie-Hellman. Come si ricorderà, Diffie-Hellman prevede le seguenti interazioni fra gli utenti A e B. Vi è un accordo iniziale sui parametri globali: q, un numero primo esteso e α, una radice primitiva di q. A seleziona un intero casuale X_A come chiave privata e trasmette a B la sua chiave pubblica $Y_A = \alpha^{X_A} \bmod q$. Analogamente, B seleziona un intero casuale X_B come propria chiave privata e trasmette ad A la propria chiave pubblica $Y_B = \alpha^{X_B} \bmod q$. Ognuno dei due lati può quindi calcolare la chiave segreta di sessione:

$$K = (Y_B)^{X_A} \bmod q = (Y_A)^{X_B} \bmod q = \alpha^{X_A X_B} \bmod q$$

L'algoritmo Diffie-Hellman ha due caratteristiche interessanti.

- Le chiavi segrete vengono create solo quando è necessario. Non vi è alcuna necessità di memorizzare le chiavi segrete per un lungo periodo di tempo, cosa che le renderebbe più vulnerabili.
- Lo scambio non richiede alcuna infrastruttura preesistente, tranne un accordo globale sui parametri.

Tuttavia, l'algoritmo Diffie-Hellman presenta anche vari punti deboli, come indicato in [HUIT98].

- Non fornisce informazioni sull'identità delle parti.
- È soggetto all'attacco man-in-the-middle, in cui un estraneo C si finge B mentre comunica con A e si finge A mentre comunica con B. Sia A che B si trovano a negoziare una chiave con C che può quindi ascoltare e inoltrare il traffico. L'attacco man-in-the-middle si svolge nel seguente modo.
  1. B invia la sua chiave pubblica Y_B in un messaggio indirizzato ad A (vedere la Figura 10.8).
  2. L'estraneo (E) intercetta questo messaggio. E salva la chiave pubblica di B e invia un messaggio ad A che ha il codice utente di B ma la chiave pubblica di E, Y_E. Questo messaggio viene inviato in modo che sembri provenire dal sistema host di B. A riceve il messaggio di E e memorizza la chiave pubblica di E con il codice utente di B. Analogamente, E invia un messaggio a B con la propria chiave pubblica, facendo in modo che sembri provenire da A.
  3. B calcola una chiave segreta K_1 basata sulla propria chiave privata e su Y_E. A calcola una chiave segreta K_2 basata sulla propria chiave privata e su Y_E. E calcola K_1 utilizzando la propria chiave segreta X_E e Y_B e calcola K_2 utilizzando la propria chiave segreta X_E e Y_A.
  4. D'ora in poi, E è in grado di rinviare i messaggi fra A e B e fra B e A, cambiando in modo appropriato la cifratura lungo il percorso in modo che né A né B possano rendersi conto che stanno condividendo la comunicazione con E.
- È computazionalmente intensivo, ed è quindi vulnerabile a un attacco clogging, in cui un estraneo richiede un elevato numero di chiavi. La vittima dedica una notevole quantità di risorse di calcolo a svolgere inutili calcoli esponenziali modulari.

Oakley è progettato per mantenere i vantaggi di Diffie-Hellman evitando nel contempo i suoi punti deboli.

Le caratteristiche di Oakley

L'algoritmo Oakley è caratterizzato da cinque importanti caratteristiche.

1. Impiega un meccanismo di cookie per evitare gli attacchi clogging.
2. Consente alle due parti di negoziare un *gruppo*; questo, in pratica, specifica i parametri globali dello scambio delle chiavi Diffie-Hellman.
3. Utilizza dei nonce per difendersi dagli attacchi a replay.
4. Consente lo scambio delle chiavi pubbliche Diffie-Hellman.
5. Autentica lo scambio Diffie-Hellman per sventare l'attacco man-in-the-middle.

Si è già parlato del funzionamento dello scambio di chiavi Diffie-Hellman. Ora si parlerà più in dettaglio degli altri elementi. Innanzitutto si consideri il problema degli attacchi clogging. In questo attacco, un estraneo falsifica l'indirizzo di origine di un utente legittimo e invia alla vittima una chiave pubblica Diffie-Hellman. La vittima esegue un'esponenziazione modulare per calcolare la chiave segreta. Una serie ripetuta di messaggi di questo tipo può mettere in ginocchio il sistema della vittima. Lo **scambio di cookie** richiede che ciascun lato invii nel messaggio iniziale un numero pseudocasuale, il cookie; l'altro capo della trasmissione ne confermerà la ricezione. Questa conferma deve essere ripetuta nel primo messaggio dello scambio di chiavi Diffie-Hellman. Se l'indirizzo di origine è stato falsificato, l'estraneo non otterrà alcuna risposta. Pertanto l'estraneo potrà solo costringere l'utente a generare gli acknowledgement e non più a svolgere il complesso calcolo Diffie-Hellman.

ISAKMP richiede che la generazione di cookie soddisfi tre requisiti fondamentali.

1. Il cookie deve dipendere dalle specifiche parti. Questo impedisce a un estraneo di derivare un cookie da un indirizzo IP e una porta UDP e inondare quindi la vittima con richieste da indirizzi IP o porte scelte casualmente.
2. Non deve essere possibile per nessuno, tranne che per l'entità emettitrice, generare dei cookie che vengano accettati da tale entità. Questo implica che nella generazione e successivamente nella verifica del cookie l'entità emettitrice utilizzerà delle informazioni segrete locali. Non deve essere possibile dedurre queste informazioni segrete da un determinato cookie. Questo requisito consente all'entità emettitrice di non dover salvare delle copie dei propri cookie che risulterebbero più vulnerabili ad eventuali attacchi ma di poter comunque verificare la conferma di un cookie in arrivo ogni volta che se ne presenti la necessità.
3. I metodi di generazione e verifica dei cookie devono essere veloci per evitare attacchi che mirano a consumare le risorse di elaborazione.

Il metodo consigliato per creare il cookie consiste nello svolgere un calcolo hash veloce (per esempio MD5) utilizzando gli indirizzi IP di origine e destinazione, le porte UDP di origine e di destinazione e un valore segreto generato localmente.

Oakley supporta l'impiego di vari **gruppi** per lo scambio delle chiavi Diffie-Hellman. Ciascun gruppo include la definizione dei due parametri globali e l'identità dell'algoritmo. Attualmente le specifiche includono i seguenti gruppi.

- Esponenziazione modulare con un modulo a 768 bit.

$$q = 2^{768} - 2^{704} - 1 + 2^{64} \times (\lfloor 2^{638} \times \pi \rfloor + 149686)$$
$$\alpha = 2$$

- Esponenziazione modulare con modulo a 1024 bit.

$$q = 2^{1024} - 2^{960} - 1 + 2^{64} \times (\lfloor 2^{894} \times \pi \rfloor + 129093)$$
$$\alpha = 2$$

- Esponenziazione modulare con modulo a 1536 bit.
  - Parametri da determinare
- Curva ellittica su 2^{155}.
  - Generatore (esadecimale): X = 7B, Y = 1C8.
  - Parametri della curva ellittica (esadecimali): A = 0, Y = 7338F.
- Curva ellittica su 2^{185}.
  - Generatore (esadecimale): X = 18, Y = D.
  - Parametri della curva ellittica (esadecimali): A = 0, Y = 1EE9.

I primi tre gruppi sono il classico algoritmo Diffie-Hellman con l'esponenziazione modulare. Gli ultimi due gruppi usano la curva ellittica analoga a Diffie-Hellman di cui si è parlato nel Capitolo 10.

Oakley impiega dei **nonce** per difendersi dagli attacchi a replay. Ciascun nonce è un numero pseudocasuale generato localmente. I nonce compaiono nelle risposte e vengono crittografati in determinati punti dello scambio per renderne l'impiego sicuro.

Con Oakley possono essere utilizzati tre diversi metodi di **autenticazione**.

- **Firme digitali:** lo scambio è autenticato firmando un codice hash ottenibile vicendevolmente; ciascuna parte esegue la crittografia del codice hash con la propria chiave privata. Il codice hash viene generato impiegando parametri notevoli, quali il codice utente e i valori nonce.
- **Crittografia a chiave pubblica:** lo scambio è autenticato eseguendo la crittografia di parametri quali i codici di identificazione e il nonce con la chiave privata del mittente.
- **Crittografia a chiave simmetrica:** la chiave deriva da un meccanismo fuori banda e può essere utilizzata per autenticare lo scambio tramite la crittografia simmetrica dei parametri scambiati.

Esempio di scambio delle chiavi Oakley

Le specifiche Oakley includono vari esempi di scambi consentiti dal protocollo. Per dare un'idea del funzionamento di Oakley, si presenterà un esempio che nelle specifiche è chiamato *scambio aggressivo delle chiavi* poiché vengono scambiati solo tre messaggi.

La Figura 16.11 mostra il protocollo di scambio aggressivo delle chiavi. Nel primo passo, l'iniziatore (I) trasmette un cookie, il gruppo da utilizzare e la propria chiave pubblica Diffie-Hellman per questo scambio. I indica anche gli algoritmi di crittografia a chiave pubblica, di calcolo hash e di autenticazione offerti da utilizzare in questo scambio. Sempre nel messaggio sono inclusi gli identificatori di I e del risponditore R più il valore nonce di I per questo scambio. Infine, I aggiunge una firma utilizzando la propria chiave privata

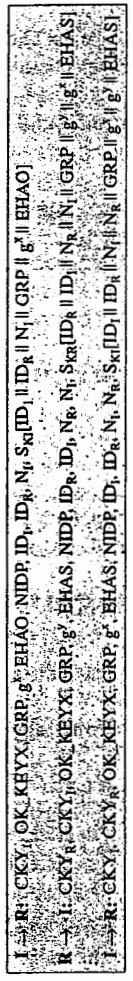


Figura 16.11 Esempio di scambio delle chiavi Oakley aggressivo.

che firma i due identificatori, il nonce, il gruppo, la chiave pubblica Diffie-Hellman e gli algoritmi offerti.

Quando R riceve il messaggio, verifica la firma utilizzando la chiave pubblica di I. R conferma la ricezione del messaggio rimandando a I il cookie, l'identificatore e il nonce insieme al gruppo. R include nel messaggio anche un cookie, la propria chiave pubblica Diffie-Hellman, gli algoritmi selezionati (che devono essere fra gli algoritmi offerti), il proprio identificatore e il proprio valore nonce per questo scambio. Infine R aggiunge una firma utilizzando la propria chiave privata che firma i due identificatori, i due nonce, il gruppo, le due chiavi pubbliche Diffie-Hellman e gli algoritmi selezionati.

Quando I riceve il secondo messaggio, verifica la firma utilizzando la chiave pubblica di R. I valori nonce contenuti nel messaggio garantiscono che questo non sia un attacco a replay che impiega un vecchio messaggio. Per completare lo scambio, I deve inviare un messaggio a R per consentirgli di verificare che I abbia ricevuto la chiave pubblica di R.

ISAKMP

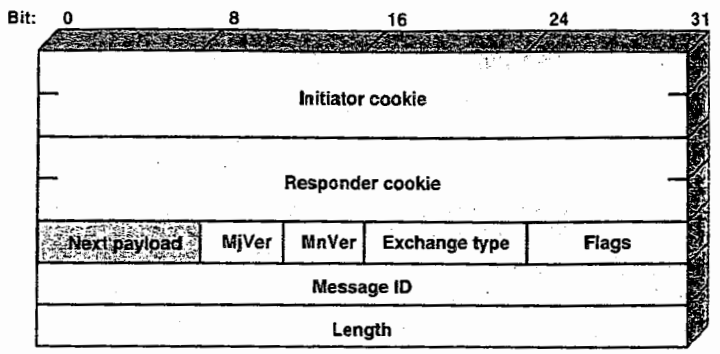
ISAKMP definisce le procedure e i formati dei pacchetti necessari per attivare, negoziare, modificare e cancellare le associazioni di sicurezza. Nell'ambito dell'attivazione di un'associazione di sicurezza, ISAKMP definisce il payload per lo scambio dei dati di generazione e autenticazione delle chiavi. Questi formati di payload costituiscono una struttura indipendente dallo specifico protocollo di scambio delle chiavi, dall'algoritmo di crittografia e dal meccanismo di autenticazione.

Il formato dell'intestazione ISAKMP

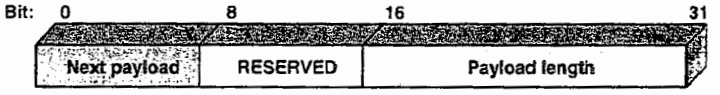
Un messaggio ISAKMP è costituito da un'intestazione ISAKMP seguita da uno o più carichi utili. Tutto ciò viene trasportato in un protocollo di trasporto. Le specifiche stabiliscono che le implementazioni debbano supportare l'impiego del protocollo di trasporto UDP.

La Figura 16.12A mostra il formato dell'intestazione di un messaggio ISAKMP. L'intestazione è costituita dai seguenti campi.

- **Initiator Cookie (64 bit):** cookie dell'entità che ha iniziato l'attivazione, la notifica o la cancellazione dell'associazione di sicurezza.
- **Responder Cookie (64 bit):** cookie dell'entità rispondente; nullo nel primo messaggio dell'iniziatore.
- **Next Payload (8 bit):** indica il tipo del primo payload del messaggio; si parlerà dei carichi utili più avanti.
- **Major Version (4 bit):** indica la versione (major) di ISAKMP utilizzata.
- **Minor Version (4 bit):** indica la versione (minor) di ISAKMP utilizzata.
- **Exchange Type (8 bit):** indica il tipo di scambio; se ne parlerà più avanti.
- **Flags (8 bit):** indica le opzioni impostate per questo scambio ISAKMP. I due bit attualmente definiti sono: il bit Encryption è impostato se tutti i carichi utili che seguono l'intestazione sono crittografati utilizzando l'algoritmo di crittografia di questa associazione di sicurezza. Il bit Commit viene utilizzato per garantire che il materiale



(A) Intestazione ISAKMP



(B) Intestazione generica del carico utile

Figura 16.12 I formati ISAKMP

crittografato non venga ricevuto prima del completamento dell'attivazione dell'associazione di sicurezza.

- **Message ID (32 bit):** codice ID univoco di questo messaggio.
- **Length (32 bit):** lunghezza totale del messaggio (intestazione più tutti i carichi utili) misurata in ottetti.

I tipi di payload ISAKMP

Tutti i carichi utili ISAKMP iniziano con la stessa intestazione generica rappresentata nella Figura 16.12B. Il campo Next Payload ha il valore 0 se questo è l'ultimo payload del messaggio altrimenti il suo valore è il tipo del payload successivo. Il campo Payload Length indica la lunghezza in ottetti del payload, compresa l'intestazione generica.

La Tabella 16.3 riepiloga i tipi di payload definiti per ISAKMP ed elenca i campi o parametri che fanno parte di ciascun payload. Il payload **SA** inizia l'attivazione di un'associazione di sicurezza. In questo caso, il parametro Domain of Interpretation (DOI) identifica il dominio nel quale si sta svolgendo la negoziazione. Il DOI IPSec è un esempio ma ISAKMP può essere utilizzato anche in altri contesti. Il parametro Situation definisce la politica di sicurezza di questa negoziazione; in pratica vengono specificati i livelli di sicurezza necessari per la crittografia e la segretezza (per esempio il livello di segretezza o il compartimento di sicurezza).

Il payload **Proposal** contiene informazioni utilizzate durante la negoziazione dell'associazione di sicurezza. Il payload indica il protocollo di questa associazione di sicurezza (ESP o AH) per il quale vengono negoziati i servizi e i meccanismi. Il payload include

Tabella 16.3 I tipi di payload ISAKMP

Tipo	Parametri	Descrizione
SA (Security Association)	Domain of Interpretation, Situation	Usato per negoziare gli attributi di sicurezza e indicare il dominio di interpretazione e la situazione nei quali si sta svolgendo la negoziazione.
P (Proposal)	Proposal #, Protocol-ID, SPI Size, # of Transforms, SPI	Usato durante la negoziazione di un'associazione di sicurezza: indico il protocollo da utilizzare e il numero di trasformazioni.
T (Transform)	Transform #, Transform-ID, SA Attributes	Usato durante la negoziazione di un'associazione di sicurezza: indico gli attributi dello trasformazione e dello relativa associazione di sicurezzo.
KE (Key Exchange)	Key Exchange Data	Supporta varie tecniche di scambio delle chiavi.
ID (Identification)	ID Type, ID Dato	Usato per scambiare le informazioni di identificazione.
CERT (Certificate)	Cert Encoding, Certificate Data	Usato per trasportare i certificati e altre informazioni correlate.
CR (Certificate Request)	# Cert Types, Certificate Types, # Cert Auths, Certificate Authorities	Usato per richiedere i certificati: indica i tipi di certificati richiesti e le autorità di certificazione accettate.
HASH (Hash)	Hash Dato	Contiene i dati generati da uno funzione hash.
SIG (Signature)	Signature Data	Contiene i dati generati da una funzione di firma digitale.
NONCE (nonce)	Nonce Data	Contiene un codice nonce.
N (Notificotion)	DOI, Protocol-ID, SPI Size, Notify Message Type, SPI, Notification Dato	Usato per trasmettere i dati di notifica, come per esempio una condizione d'errore.
D (Delete)	DOI, Protocol-ID, SPI Size, # of SPIs, SPI (uno o più)	Indica che un'associazione di sicurezza non è più valida.

anche l'identificatore SPI dell'iniziatore e il numero di trasformazioni. Ogni trasformazione è contenuta in un apposito payload. L'uso di più carichi utili di trasformazione consente all'iniziatore di offrire più possibilità, fra le quali il risponditore deve sceglierne una o rifiutare l'offerta.

Il payload **Transform** definisce la trasformazione di sicurezza da utilizzare per rendere sicuro il canale di comunicazione per il protocollo indicato. Il parametro Transform # identifica questo specifico payload in modo che il risponditore possa utilizzarlo per indicare l'accettazione di questa particolare trasformazione. I campi Transform-ID e Attributes

identificano una specifica trasformazione (per esempio 3DES per ESP, HMAC-SHA-1-96 per AH) con i relativi attributi (per esempio la lunghezza del codice hash).

Il payload **Key Exchange** può essere utilizzato per varie tecniche di scambio delle chiavi fra cui Oakley, Diffie-Hellman e le tecniche RSA utilizzate da PGP. Il campo dati Key Exchange contiene i dati necessari per generare una chiave di sessione e dipende dall'algoritmo di scambio delle chiavi utilizzato.

Il payload **Identification** viene utilizzato per determinare l'identità dei nodi in comunicazione e può essere utilizzato per valutare l'autenticità delle informazioni. In genere il campo ID Data contiene un indirizzo IPv4 o IPv6.

Il payload **Certificate** trasferisce un certificato a chiave pubblica. Il campo Certificate Encoding indica il tipo di certificato o di informazioni correlate e può comprendere i seguenti elementi:

- Certificato PKCS #7 wrapped X.509
- Certificato PGP
- Chiave firmata DNS
- Certificato X.509; firma
- Certificato X.509; scambio delle chiavi
- Token Kerberos
- Certificate Revocation List (CRL)
- Authority Revocation List (ARL)
- Certificato SPKI

In qualsiasi momento di uno scambio ISAKMP, il mittente può includere un payload **Certificate Request** per richiedere il certificato dell'altra entità in comunicazione. Il payload può elencare più tipi di certificati e più autorità di certificazione accettabili.

Il payload **Hash** contiene i dati generati da una funzione hash su una determinata parte del messaggio e/o lo stato ISAKMP. Questo payload può essere utilizzato per verificare l'integrità dei dati in un messaggio o per autenticare le entità in negoziazione.

Il payload **Signature** contiene i dati generati da una funzione di firma digitale su una parte del messaggio e/o dello stato ISAKMP. Questo payload viene utilizzato per verificare l'integrità dei dati in un messaggio e può essere utilizzato per servizi di non-ripudiabilità.

Il payload **Nonce** contiene dati casuali utilizzati per garantire l'attualità di uno scambio e proteggersi da attacchi a replay.

Il payload **Notification** contiene informazioni di errore o di stato relative a questa associazione di sicurezza o a questa negoziazione dell'associazione di sicurezza. ISAKMP definisce i seguenti messaggi d'errore:

Invalid Payload Type
DOI Not Supported
Situation Not Supported
Invalid Cookie
Invalid Major Version
Invalid Minor Version
Invalid Exchange Type
Invalid Flags
Invalid Message ID
Invalid Protocol ID
Invalid SPI
Invalid Transform ID
Attributes Not Supported
No Proposal Chosen
Bad Proposal Syntax
Payload Malformed

Invalid Key Information
Invalid Cert Encoding
Invalid Certificate
Bad Cert Request Syntax
Invalid Cert Authority
Invalid Hash Information
Authentication Failed
Invalid Signature
Address Notification

L'unico messaggio di stato ISAKMP attualmente definito è Connected. Oltre a queste notifiche ISAKMP, vengono utilizzate delle notifiche relative al particolare di DOI. Per IPSec, sono definiti i seguenti messaggi di stato aggiuntivi.

- **Responder-Lifetime:** comunica la durata dell'associazione di sicurezza scelta dal risponditore.
- **Replay-Status:** usato per confermare la scelta del risponditore di attivare o meno il rilevamento anti-replay.
- **Initial-Contact:** informa l'altra parte che questa è la prima associazione di sicurezza attivata con il sistema remoto. Il destinatario di questa notifica può quindi cancellare le eventuali associazioni di sicurezza attive con il sistema mittente, supponendo che questo sia stato riavviato e non abbia più accesso a queste associazioni di sicurezza.

Il payload **Delete** indica una o più associazioni di sicurezza che il mittente ha cancellato dal proprio database e che pertanto non sono più valide.

Scambi ISAKMP

ISAKMP fornisce una struttura per lo scambio dei messaggi, dove i tipi di payload servono come elementi costitutivi. Le specifiche identificano cinque tipi di scambio standard che dovrebbero sempre essere supportati. Essi sono riepilogati nella Tabella 16.4, dove l'indicazione SA fa riferimento al payload dell'associazione di sicurezza con i relativi payload Protocol e Transform.

Lo scambio **Base** consente lo scambio contemporaneo delle chiavi e delle informazioni di autenticazione. Questo riduce il numero di scambi ma presenta il difetto di non proteggere l'identità. I primi due messaggi forniscono i cookie e attivano un'associazione di sicurezza le trasformazioni su e il protocollo concordati; entrambe le parti usano un codice nonce per proteggersi dagli attacchi a replay. Gli ultimi due messaggi scambiano le informazioni delle chiavi e i codici ID utente con un meccanismo di autenticazione utilizzato per autenticare le chiavi, le identità e i codici nonce dei primi due messaggi.

Lo scambio **Identity Protection** espande lo scambio Base per proteggere le identità degli utenti. I primi due messaggi attivano l'associazione di sicurezza. I due messaggi successivi eseguono lo scambio delle chiavi utilizzando codici nonce per evitare attacchi a replay. Una volta calcolata la chiave di sessione, le due parti si scambiano dei messaggi crittografati che contengono le informazioni di autenticazione come le firme digitali e opzionalmente i certificati di convalida delle chiavi pubbliche.

Lo scambio **Authentication Only** viene utilizzato per svolgere la reciproca autenticazione senza scambio di chiavi. I primi due messaggi attivanó l'associazione di sicurezza. Inoltre il risponditore utilizza il secondo messaggio per trasferire il proprio codice utente e utilizza l'autenticazione per proteggere il messaggio. L'iniziatore invia il terzo messaggio per trasmettere il proprio codice utente autenticato.

Tabella 16.4 I tipi di scambio ISAKMP

Scambio	Nota
A. Scambio Base	
(1) I → R: SA; NONCE	Inizia la negoziazione dell'associazione di sicurezza ISAKMP.
(2) R → I: SA; NONCE	Associazione di sicurezza base concordata.
(3) I → R: KE; ID_i; AUTH	Chiave generata; identità dell'iniziatore verificata dal risponditore.
(4) R → I: KE; ID_r; AUTH	Identità del risponditore verificata dall'iniziatore; chiave generoto; associazione di sicurezza attivata.
B. Scambio Identity Protection	
(1) I → R: SA	Inizia la negoziazione dell'associazione di sicurezza ISAKMP.
(2) R → I: SA	Associazione di sicurezza base accordata.
(3) I → R: KE; NONCE	Chiove generata.
(4) R → I: KE; NONCE	Chiave generata.
(5)* I → R: ID_i; AUTH	Identità dell'iniziatore verificata dol risponditore.
(6)* R → I: ID_r; AUTH	Identità del risponditore verificata dall'iniziatore; associazione di sicurezza attivata.
C. Scambio Authentication Only.	
(1) I → R: SA; NONCE	Inizia la negoziazione dell'associazione di sicurezza ISAKMP.
(2) R → I: SA; NONCE; IDR; AUTH	Associazione di sicurezza base accordata; identità del risponditore verificata dall'iniziatore.
(3) I → R: ID_i; AUTH	Identitò del risponditore verificata dall'iniziatore; associazione di sicurezza attivata.
D. Scambi Aggressive	
(1) I → R: SA; KE; NONCE; IDI	Inizia la negoziazione dell'associazione di sicurezza ISAKMP e lo scambio delle chiavi.
(2) R → I: SA; KE; NONCE; IDR; AUTH	Identità del risponditore verificata dall'iniziatore; chiave generata; associazione di sicurezza base accordata.
(3)* I → R: AUTH	Identità del risponditore verificato dall'iniziatore; associazione di sicurezza attivata.
E. Scambio Informational	
(1) I → R: N/D;	Cancellazione o notifica di errore o di stato.

Notazione
I = iniziatore
R = risponditore
* = crittografia del payload dopo l'intestazione ISAKMP.
AUTH = meccanismo di autenticazione impiegata.

Lo scambio **Aggressive** riduce il numero di scambi ma non garantisce la protezione dell'identità. Nel primo messaggio l'iniziatore propone un'associazione di sicurezza offrendo dei protocolli e delle opzioni di trasformazione. L'iniziatore attiva anche lo scambio

della chiave e fornisce il proprio codice utente. Nel secondo messaggio, il risponditore indica se ha accettato l'associazione di sicurezza con un determinato protocollo e una determinata trasformazione, completa lo scambio della chiave e autentica le informazioni trasmesse. Nel terzo messaggio l'iniziatore autentica le informazioni precedenti, crittografandole con la chiave di sessione segreta condivisa.

Lo scambio **Informational** viene utilizzato per la trasmissione monodirezionale di informazioni per la gestione dell'associazione di sicurezza.

16.7 Letture e siti Web consigliati

IPv6 e IPv4 sono trattati in dettaglio in [STAL04]. [CHEN98] fornisce una buona discussione sulla progettazione di IPSec. [FRAN01] e [DORA03] sono trattazioni più complete di IPSec.

CHEN98 P. Cheng, et al. "A Security Architecture for the Internet Protocol." *IBM Systems Journal*, Numero 1, 1998.

DORA03 N. Doraswamy e D. Harkins. *IPSec.* Upper Saddle River, NJ: Prentice Hall, 2003.

FRAN01 S. Frankel. *Demystifying the IPSec Puzzle.* Boston: Artech House, 2001.

STAL04 W. Stallings. *Computer Networking with Internet Protocols and Technology.* Upper Saddle River, NJ: Prentice Hall, 2004.

Siti Web consigliati

- **NIST IPSEC Project:** articoli, presentazioni e implementazioni di riferimento.

16.8 Termini chiave, domande di ripasso e problemi

Termini chiave

AH (Authentication Header)
Associazione di sicurezza (SA)
Attacco a replay
ESP (Encapsulating Security Payload)
IPSec (IP Security)
IPv4
IPv6
ISAKMP (Internet Security Association and Key Management Protocol)
Modalità transport
Modalità tunnel
Protocollo Oakley (protocollo di determinazione della chiave)
Servizio anti-replay

Domande di ripasso

16.1 Fornire esempi di applicazioni di IPSec.
16.2 Quali servizi offre IPSec?
16.3 Quali parametri identificano un'associazione di sicurezza e quali parametri caratterizzano la natura di una determinata associazione di sicurezza?
16.4 Qual è la differenza fra modalità transport e modalità tunnel?
16.5 Che cos'è un attacco a replay?
16.6 Perché il protocollo ESP include un campo di padding?
16.7 Quali sono gli approcci di base al raggruppamento di associazioni di sicurezza?
16.8 Quali sono i ruoli del protocollo di determinazione della chiave Oakley e di ISAKMP in IPSec?

Problemi

16.1 Nella discussione dell'elaborazione AH si è detto che non tutti i campi di un intestazione IP vengono inclusi nel calcolo del codice MAC.
   A. Per ognuno dei campi dell'intestazione IPv4, indicare se il campo è immutabile, mutabile ma prevedibile o mutabile (azzerato prima del calcolo ICV).
   B. Svolgere la stessa operazione per l'intestazione IPv6.
   C. Svolgere la stessa operazione per le intestazioni di estensione IPv6.
   Giustificare le decisioni per ciascun campo.

16.2 Quando viene utilizzata la modalità tunnel, viene costruita una nuova intestazione IP esterna. Per IPv4 e IPv6 indicare la relazione fra ciascun campo dell'intestazione IP esterna e ciascuna intestazione di estensione del pacchetto esterno con il campo o l'intestazione di estensione corrispondenti nel pacchetto IP interno. Ovvero indicare quali valori esterni derivano dai valori interni e quali vengono costruiti in modo indipendente dai valori interni.

16.3 Due host richiedono l'autenticazione e la crittografia end-to-end. Schematizzare con le figure simili alle Figure 16.6 e 16.9 i casi seguenti.
   A. Adiacenza di trasporto, con crittografia applicata prima dell'autenticazione.
   B. Un'associazione di sicurezza di trasporto nidificata in un'associazione di sicurezza a tunnel con la crittografia applicata prima dell'autenticazione.
   C. Un'associazione di sicurezza di trasporto nidificata in un'associazione di sicurezza a tunnel con l'autenticazione applicata prima della crittografia

16.4 Il documento IPSec Architecture stabilisce che quando due associazioni di sicurezza in modalità transport vengono raggruppate per consentire l'impiego di entrambi i protocolli AH e ESP sullo stesso flusso end-to-end, solo un ordinamento dei protocolli di sicurezza sembra appropriato: applicare il protocollo ESP prima del protocollo AH. Perché si consiglia di adottare questo approccio invece di applicare l'autenticazione prima della crittografia?

16.5 A. A quale dei tipi di scambio ISAKMP (Tabella 16.4) corrisponde lo scambio di chiavi Oakley aggressivo (Figura 16.11)?
   B. Per lo scambio di chiavi Oakley aggressivo, indicare quali parametri in ciascun messaggio vanno in quali tipi di payload ISAKMP.

Appendice 16.A Protocolli di interconnessione fra reti e Internet

Questa appendice fornisce una panoramica sui protocolli di interconnessione fra reti. Si partirà con un riepilogo del ruolo dei protocolli Internet per garantire l'interconnessione fra le reti. Poi si parlerà dei due principali protocolli di questo tipo: IPv4 e IPv6.

Il ruolo di un protocollo IP

Il protocollo Internet (IP) fornisce le funzionalità di interconnessione dei sistemi posti su reti differenti. Per questo scopo, IP è implementato in ciascun sistema terminale e router (i dispositivi che garantiscono la connessione fra reti differenti). I dati di alto livello del sistema terminale di origine vengono incapsulati per la trasmissione in un'unità dati chiamata PDU (Protocol Data Unit) di livello IP. La PDU IP passa poi attraverso una o più reti e router di connessione per raggiungere il sistema terminale di destinazione.

Il router deve essere in grado di gestire le eventuali differenze fra le reti, fra le quali seguenti.

- **Schemi di indirizzamento:** le reti possono usare schemi differenti per assegnare gli indirizzi ai dispositivi. Per esempio, una rete locale IEEE 802 usa indirizzi binari a 16 o 48 bit per ciascun dispositivo connesso; una rete pubblica X.25 a commutazione di pacchetto usa indirizzi decimali a 12 cifre (codificati come quattro bit per cifra decimale per un indirizzo totale di 48 bit). Occorre una forma di indirizzamento di rete globale e un servizio di directory.
- **Dimensioni massime dei pacchetti:** i pacchetti di una rete possono dover essere suddivisi in frammenti più piccoli per poter essere trasmessi su un'altra rete, con un processo chiamato frammentazione. Per esempio, Ethernet impone pacchetti delle dimensioni massime di 1500 byte, mentre sulle reti X.25 vengono frequentemente utilizzati pacchetti di 1000 byte. Un pacchetto trasmesso su un sistema Ethernet e raccolto da un router per la ritrasmissione su una rete X.25 può dover quindi essere frammentato in due parti.
- **Interfacce:** le interfacce hardware e software delle varie reti sono differenti. Il router deve poter gestire queste differenze.
- **Affidabilità:** i vari servizi di rete possono variare da servizi affidabili a circuito virtuale end-to-end o a servizi inaffidabili. Il funzionamento dei router non deve fare affidamento sull'affidabilità della rete.

Il funzionamento di un router, come indicato nella Figura 16.13, dipende dal protocollo internet. In questo esempio si utilizza il protocollo IP della famiglia di protocolli TCP/IP. IP deve essere implementato in tutti i sistemi terminali di tutte reti e in tutti i router. Inoltre ciascun sistema terminale deve avere protocolli sopra IP compatibili per poter comunicare. I router intermedi devono invece essere solamente dotati dei protocolli fino a IP.

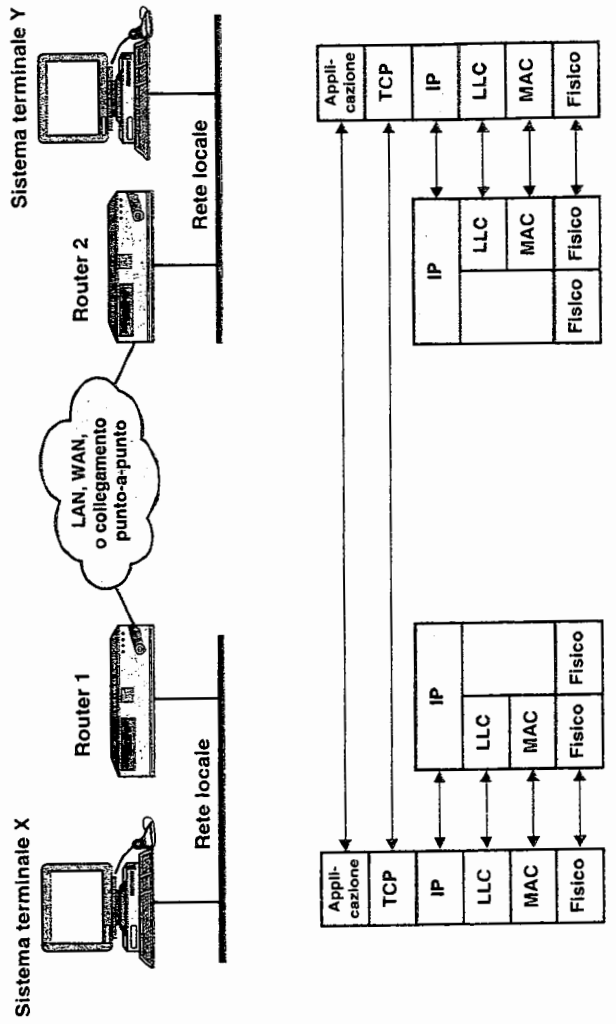


Figura 16.13 Esempio di configurazione per TCP/IP

Si consideri il trasferimento di un blocco di dati dal sistema terminale *X* al sistema terminale *Y* (vedere la Figura 16.13). Il livello IP di *X* riceve i blocchi di dati da inviare a *Y* dal livello TCP.

Il livello IP associa un'intestazione che specifica l'indirizzo internet globale di *Y*. Questo indirizzo è costituito da due parti: l'identificatore della rete e l'identificatore del sistema terminale. Questo blocco forma il pacchetto IP. Quindi IP trova che la destinazione (*Y*) si trova su un'altra sottorete. Dunque il primo passo è quello di inviare il pacchetto a un router, in questo caso il router 1. Per fare ciò, invia l'unità dati IP al livello LLC fornendo le informazioni di indirizzamento appropriate. L'LLC crea un'unità PDU che viene inviata al livello MAC. Il livello MAC costruisce un pacchetto MAC la cui intestazione contiene l'indirizzo del router 1.

Quindi il pacchetto attraversa la rete locale fino a giungere al router 1. Il router elimina l'intestazione e la coda LLC e analizza l'intestazione IP per determinare la destinazione finale dei dati, in questo caso *Y*. A questo punto il router deve prendere una decisione di routing. Vi sono due possibilità.

1. Il sistema terminale di destinazione *Y* è connesso direttamente a una delle sottoreti cui è connesso il router.
2. Per raggiungere la destinazione devono essere attraversati uno o più router aggiuntivi.

In questo esempio il pacchetto deve passare attraverso il router 2 prima di raggiungere la destinazione. Pertanto il router 1 passa il pacchetto IP al router 2 tramite una rete intermedia, utilizzando quindi i protocolli di tale rete. Per esempio, se la rete intermedia è di tipo X.25, l'unità dati IP viene imbustata in un pacchetto X.25 dotato delle informazioni di indirizzamento appropriate per raggiungere il router 2. Quando questo pacchetto arriva al router 2, l'intestazione del pacchetto viene eliminata. Il router determina che questo pacchetto IP è destinato a *Y* che è connesso direttamente a una sottorete cui è connesso anche il router. Pertanto il router crea un pacchetto con l'indirizzo di destinazione di *Y* e lo invia attraverso la rete locale. Alla fine i dati raggiungono *Y* dove le informazioni LLC e IP possono essere eliminate.

Questo servizio offerto dal protocollo IP è di tipo inaffidabile. In pratica IP non garantisce che tutti i dati vengano consegnati o che i dati consegnati arriveranno nell'ordine corretto. Sarà compito del livello più elevato, in questo caso TCP, individuare e gestire eventuali errori. Questo approccio offre un elevato livello di flessibilità. Poiché non è garantita la consegna, non esistono particolari requisiti di affidabilità in nessuna delle sottoreti. Pertanto il protocollo funziona con qualsiasi combinazione di sottoreti. Poiché la sequenza di consegna non è garantita, i pacchetti possono anche seguire percorsi differenti in internet. Questo consente al protocollo di reagire a problemi di congestione o a guasti semplicemente cambiando il percorso di rete.

Il protocollo IPv4

Per decenni, il fondamento dell'architettura TCP/IP si è basata sul protocollo IP (Internet Protocol) versione 4. La Figura 16.14A mostra il formato dell'intestazione IP costituita da un minimo di 20 ottetti (160 bit). Ecco i campi utilizzati.

- **Version (4 bit):** indica il numero di versione per consentire l'evoluzione del protocollo; il valore è 4.

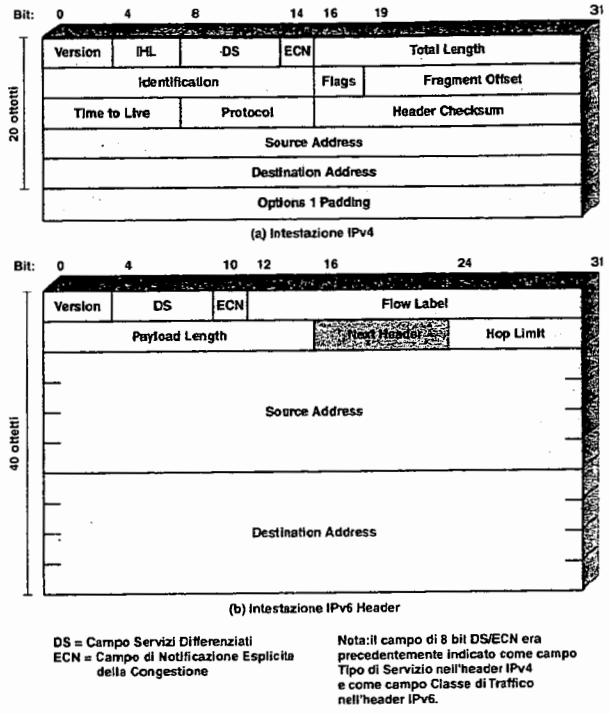


Figura 16.14 Le intestazioni IP

- **Internet Header Length (IHL) (4 bit):** lunghezza dell'intestazione in word di 32 bit. Il valore minimo è 5 per una lunghezza minima dell'intestazione di 20 ottetti.
- **DS/ECN (8 bit):** prima dell'introduzione dei servizi differenziati, questo campo veniva chiamato *Type of Service* e definiva i parametri di affidabilità, precedenza e prestazioni. Questa interpretazione è ora obsoleta. I primi 6 bit del campo TOS costituiscono attualmente il campo DS (*Differentiated Services*, o Servizi Differenziati); i due bit rimanenti sono utilizzati come campo ECN (*Explicit Congestion Notification*, o Notificazione Esplicita della Congestione).
- **Total Length (16 bit):** lunghezza totale del pacchetto IP, in ottetti.
- **Identification (16 bit):** numero sequenziale che, insieme all'indirizzo di sorgente, all'indirizzo di destinazione e al protocollo utente, identifica univocamente un pacchetto. Pertanto l'identificatore dovrà essere univoco per l'indirizzo di sorgente del pacchetto, l'indirizzo di destinazione del pacchetto e il protocollo utente per tutto il tempo in cui il pacchetto rimane in internet.
- **Flag (3 bit):** sono definiti solo due dei bit. Quando un pacchetto viene frammentato, il bit More indica se questo è l'ultimo frammento del pacchetto originale. Il bit Don't

Fragment proibisce la frammentazione. Questo bit può essere utile se la destinazione non è in grado di riassemblare i frammenti. Tuttavia, quando questo bit è impostato a 1, il pacchetto verrà eliminato se supera la lunghezza massima di una sottorete del percorso. Pertanto, quando questo bit è impostato, può essere utile impiegare il source routing per evitare le sottoreti che usano pacchetti di dimensioni minori.

- **Fragment Offset (13 bit):** indica la posizione di questo frammento all'interno del pacchetto originale misurata in unità di 64 bit. Questo implica che i frammenti diversi dall'ultimo devono contenere un campo dati la cui lunghezza è un multiplo di 64 bit.
- **Time to Live (8 bit):** specifica la durata, in secondi, per la quale un pacchetto può rimanere in Internet. Ogni router che elabora un pacchetto deve ridurre il valore TTL di almeno un'unità e dunque il campo TTL è per certi versi simile a un conteggio delle tratte attraversate.
- **Protocol (8 bit):** indica il protocollo di alto livello che riceverà i dati alla destinazione; pertanto questo campo identifica il tipo dell'intestazione nel pacchetto successiva all'intestazione IP.
- **Header Checksum (16 bit):** codice di rilevamento degli errori applicato alla sola intestazione. Poiché alcuni campi dell'intestazione possono cambiare durante il transito (per esempio i campi Time to Live e quelli di frammentazione), questo valore viene riverificato e ricalcolato da ciascun router. Il campo di checksum è una somma a 16 bit in complemento a 1 di tutte le word di 16 bit contenute nell'intestazione. Il campo del checksum viene inizializzato al valore 0.
- **Source Address (32 bit):** codificato in modo da consentire un'allocazione variabile di bit per specificare la rete e il sistema terminale connesso alla rete specificata (7 e 24 bit, 14 e 16 bit o 21 e 8 bit).
- **Destination Address (32 bit):** stesse caratteristiche dell'indirizzo sorgente.
- **Options (variabile):** codifica le opzioni richieste dal mittente; possono includere l'etichetta di sicurezza, il source routing, il record routing e il timestamp.
- **Padding (variabile):** utilizzato per garantire che l'intestazione del pacchetto abbia una lunghezza multipla di 32 bit.

Il protocollo IPv6

Nel 1995 la IETF (Internet Engineering Task Force), che sviluppa gli standard per i protocolli Internet, emise una specifica per la "prossima generazione" (Next Generation) del protocollo IP, chiamata IPng. Questa specifica venne trasformata in standard nel 1996 con il nome IPv6. IPv6 offre vari miglioramenti funzionali rispetto al protocollo IP esistente (IPv4), che hanno lo scopo di rispondere alle esigenze delle attuali reti ad alta velocità e dei vari tipi di flussi di dati che comprendono grafica e video e che stanno diventando sempre più importanti. Ma ciò che ha portato allo sviluppo del nuovo protocollo è stata la necessità di un maggior numero di indirizzi. IPv4 identifica la sorgente e la destinazione tramite indirizzi di 32 bit. Data la crescita esplosiva di Internet e delle reti private connesse a Internet, questa lunghezza è diventata insufficiente per comprendere tutti i sistemi che richiedono un indirizzo. Come indica la Figura 16.14B, IPv6 prevede campi per gli indirizzi di 128 bit. Alla fine tutte le installazioni che utilizzano TCP/IP migreranno dalla versione corrente di IP alla versione IPv6, ma questo processo richiederà molti anni, se non decenni.

L'intestazione IPv6

L'intestazione IPv6 ha una lunghezza fissa di 40 ottetti costituita dai seguenti campi (Figura 16.14B).

- **Version (4 bit):** numero di versione del protocollo; il valore è 6.
- **DS/ECN (8 bit):** prima dell'introduzione dei servizi differenziati, questo campo veniva chiamato *Traffic Class* (Classe di Traffico) ed era riservato all'impiego da parte del nodo mittente e/o dei router di inoltro per identificare e distinguere le varie classi o priorità dei pacchetti IPv6. I primi 6 bit del campo Traffic Class costituiscono ora il campo DS (Servizi Differenziati); i due bit rimanenti sono utilizzati come campo ECN (Notificazione Esplicita della Congestione).
- **Flow Label (20 bit):** può essere utilizzato dall'host per etichettare quei pacchetti per i quali è richiesta una particolare gestione da parte dei router della rete. Questo campo può assistere nella prenotazione delle risorse e nell'elaborazione del traffico in tempo reale.
- **Payload Length (16 bit):** lunghezza della parte rimanente del pacchetto IPv6 che segue l'intestazione, misurata in ottetti. In altre parole è la lunghezza totale di tutte le intestazioni di estensione e della PDU di livello trasporto.
- **Next Header (8 bit):** identifica il tipo di intestazione che segue immediatamente l'intestazione IPv6; può trattarsi di un'intestazione di estensione IPv6 o di un'intestazione di livello più elevato, per esempio TCP o UDP.
- **Hop Limit (8 bit):** numero massimo di tratte rimanenti per questo pacchetto. Il limite viene impostato al valore massimo dalla sorgente e poi decrementato di una unità da ciascun nodo mentre il pacchetto viene inoltrato verso la destinazione. Il pacchetto viene eliminato nel caso in cui il campo Hop Limit raggiunga il valore 0.
- **Source Address (128 bit):** indirizzo di sorgente del pacchetto.
- **Destination Address (128 bit):** indirizzo del destinatario previsto del pacchetto. Questa può anche non essere la destinazione finale del pacchetto se è presente l'intestazione di estensione Routing, come si vedrà più avanti.

Sebbene l'intestazione IPv6 sia più lunga della porzione obbligatoria di IPv4 (40 ottetti contro 20), contiene meno campi (8 contro 12). Pertanto i router avranno un carico computazionale minore per ogni intestazione, cosa che dovrebbe accelerare l'operazione di inoltro.

Le extension header di IPv6

Un pacchetto IPv6 include l'intestazione IPv6, di cui si è appena parlato e zero o più extension header. Oltre a IPSec, sono state definite le seguenti extension header.

- **Intestazione Hop-by-Hop Options:** definisce le opzioni speciali che richiedono un'elaborazione durante le varie tratte del percorso.
- **Intestazione Routing:** fornisce un routing esteso, simile al source routing di IPv4.
- **Intestazione Fragment:** contiene le informazioni di frammentazione e di assemblaggio.

- **Intestazione Authentication Header:** fornisce le funzionalità di integrità e di autenticazione dei pacchetti.
- **Intestazione Encapsulating Security Payload:** fornisce la funzione di privacy.
- **Intestazione Destination Options:** contiene informazioni opzionali che devono essere esaminate dal nodo di destinazione.

Lo standard IPv6 raccomanda che, quando vengono utilizzate più extension header, le intestazioni IPv6 compaiano nel seguente ordine.

1. Intestazione IPv6: obbligatoria, deve sempre comparire per prima.
2. Intestazione Hop-by-Hop Options.
3. Intestazione Destination Options: per le opzioni che devono essere elaborate dalla prima destinazione che compare nel campo IPv6 Destination Address più le successive destinazioni elencate nell'intestazione Routing.
4. Intestazione Routing.
5. Intestazione Fragment.
6. Intestazione Authentication Header.
7. Intestazione Encapsulating Security Payload.
8. Intestazione Destination Options: per le opzioni che devono essere elaborate solo dalla destinazione finale del pacchetto.

La Figura 16.15 mostra un esempio di pacchetto IPv6 che include un'istanza di ogni intestazione non relativa alla sicurezza. Si noti che l'intestazione IPv6 e ciascuna extension header include un campo Next Header. Questo campo identifica il tipo dell'intestazione immediatamente seguente. Se l'estensione successiva è un'extension header, questo campo ne contiene l'identificatore. Altrimenti questo campo contiene l'identificatore del protocollo di livello superiore che utilizza IPv6 (in genere un protocollo del livello di trasporto), codificato con gli stessi valori del campo Protocol di IPv4. Nella figura, il protocollo di livello superiore è TCP, quindi i dati di livello superiore trasportati dal pacchetto IPv6 sono costituiti da un'intestazione TCP seguita da un blocco dei dati di livello applicazione.

L'intestazione **Hop-by-Hop Options** trasporta informazioni opzionali che, se presenti, devono essere esaminate da ogni router lungo il percorso. L'intestazione è costituita dai seguenti campi.

- **Next Header (8 bit):** identifica il tipo di intestazione che segue immediatamente questa intestazione.
- **Header Extension Length (8 bit):** lunghezza di questa intestazione in unità di 64 bit, esclusi i primi 64 bit.
- **Options:** contiene una o più opzioni. Ogni opzione è costituita da tre sottocampi: un tag che indica il tipo di opzione, una lunghezza e un valore.

Finora è stata definita la sola opzione Jumbo Payload, utilizzata per inviare pacchetti IPv6 con un payload più lungo di $2^{16} - 1 = 65\,535$ ottetti. Il campo Option Data di questa opzione è lungo 32 bit e indica la lunghezza del pacchetto in ottetti, esclusa l'intestazione IPv6. Per questi pacchetti, il campo Payload Length dell'intestazione IPv6 deve essere impostato a 0 e non vi deve essere intestazione Fragment. Con questa opzione, IPv6 supporta pacchetti di dimensione massima superiore a quattro miliardi di ottetti. Questo facilita la

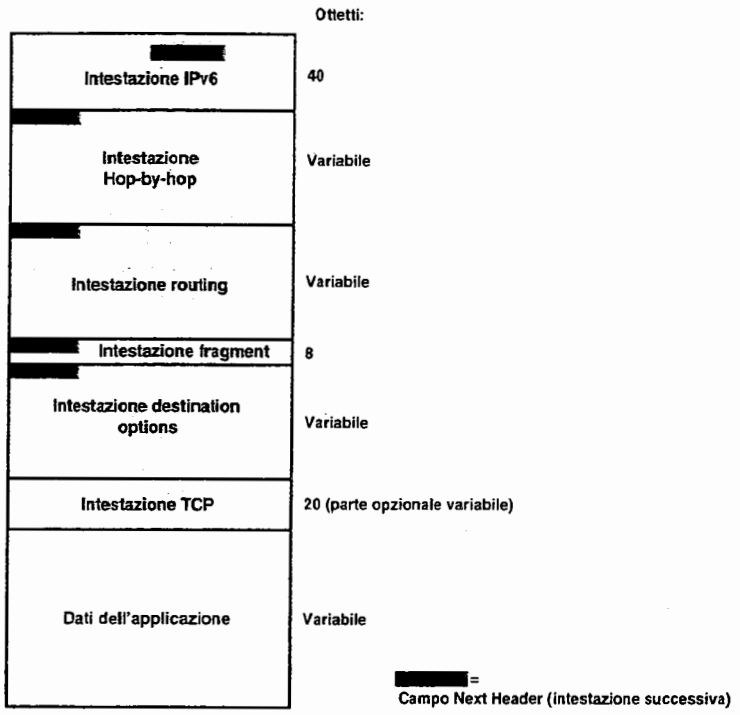


Figura 16.15 Un pacchetto IPv6 con le intestazioni di estensione (contiene un segmento TCP).

trasmissione di grossi pacchetti video e consente a IPv6 di ottimizzare l'impiego della capacità disponibile in qualsiasi mezzo di trasmissione.

L'intestazione **Routing** contiene un elenco di uno o più nodi intermedi da visitare per giungere alla destinazione. Tutte le intestazioni di routing cominciano con un blocco di 32 bit costituito da quattro campi di 8 bit, seguito da dati di routing specifici a un determinato tipo di routing. I quattro campi di 8 bit sono Next Header, Header Extension Length e i seguenti.

- **Routing Type:** identifica una determinata variante dell'intestazione di Routing. Se un router non riconosce il valore Routing Type, deve eliminare il pacchetto.
- **Segments Left:** numero di nodi intermedi elencati esplicitamente ancora da visitare prima di raggiungere la destinazione finale.

Oltre a questa definizione dell'intestazione generale, le specifiche IPv6 definiscono l'intestazione Type 0 Routing. Quando si usa questa intestazione, il nodo sorgente non inserisce nell'intestazione IPv6 l'indirizzo della destinazione finale. Tale indirizzo corrisponde invece all'ultimo indirizzo elencato nell'intestazione Routing mentre l'intestazione IPv6 con-

tiene l'indirizzo di destinazione del primo router del percorso. L'intestazione Routing non verrà esaminata finché il pacchetto non giunge al nodo identificato dall'intestazione IPv6. A questo punto, i contenuti delle intestazioni IPv6 e Routing vengono aggiornati e il pacchetto viene inoltrato. L'aggionamento è costituito dall'inserimento nell'intestazione IPv6 dell'indirizzo successivo da visitare e nel decrementare il campo Segments Left dell'intestazione Routing.

IPv6 richiede che un nodo IPv6 che riceve un pacchetto contenente un'intestazione Routing inverta il percorso nel caso in cui debba rinviare il pacchetto al mittente.

L'intestazione **Fragment** viene utilizzata dalla sorgente quando è richiesta la frammentazione. In IPv6 la frammentazione può essere eseguita solo dai nodi sorgente e non dai router che si trovano lungo il percorso di consegna del pacchetto. Per sfruttare al meglio l'ambiente di interconnessione fra reti, un nodo deve eseguire un algoritmo di ricerca del percorso che gli consenta di conoscere la più piccola unità di trasmissione massima (MTU – Maximum Transmission Unit) supportata dalle sottoreti che si trovano lungo il percorso. In altre parole, l'algoritmo di ricerca del percorso consente a un nodo di individuare il valore MTU della sottorete che rappresenta il "collo di bottiglia" del percorso. Ottenuta questa informazione, il nodo sorgente frammenterà il pacchetto a seconda della sua destinazione. Altrimenti la sorgente deve limitare tutti i pacchetti a 1280 ottetti, il minimo valore MTU che deve essere supportato da qualsiasi sottorete.

Oltre al campo Next Header, l'intestazione Fragment include i seguenti campi.

- **Fragment Offset (13 bit):** indica la posizione del payload di questo frammento all'interno del pacchetto originario. È misurato in unità di 64 bit. Questo implica che i frammenti (tranne l'ultimo) devono contenere un campo dati multiplo di 64 bit.
- **Res (2 bit):** riservato per utilizzi futuri.
- **Flag (1 bit):** 1 = seguono altri frammenti; 0 = ultimo frammento.
- **Identification (32 bit):** identifica univocamente il pacchetto originario. L'identificatore deve essere univoco per l'indirizzo sorgente e di destinazione del pacchetto per tutta la durata in cui il pacchetto rimarrà nella rete internet. Tutti i frammenti con lo stesso identificatore, indirizzo di origine e indirizzo di destinazione vengono assemblati per formare il pacchetto originario.

L'intestazione **Destination Options** trasporta informazioni opzionali che, quando presenti, vengono esaminate solo dal nodo di destinazione del pacchetto. Il formato di questa intestazione è lo stesso impiegato per l'intestazione Hop-by-Hop Options.

Capitolo 17

La sicurezza nel Web

Concetti essenziali

- **SSL** (Secure Socket Layer) fornisce i servizi di sicurezza fra TCP e le applicazioni che utilizzano TCP. La versione standard Internet si chiama **TLS** (Transport Layer Security, o "sicurezza a livello trasporto").
- SSL/TLS garantisce la segretezza utilizzando la crittografia simmetrica; garantisce l'integrità dei messaggi utilizzando un codice di autenticazione MAC.
- SSL/TLS include i meccanismi/protocolli per consentire a due utenti TCP di negoziare i meccanismi e i servizi di sicurezza da utilizzare.
- **SET** (Secure Electronic Transaction) è una specifica aperta per la crittografia e la sicurezza, progettata per proteggere le transazioni su Internet con carte di credito.

Al giorno d'oggi praticamente tutte le aziende, la maggior parte degli enti statali e anche molti individui sono dotati del proprio sito Web. Il numero di utenti e aziende dotate di accesso a Internet cresce rapidamente e tutti sono dotati di browser Web grafici. Il risultato è che le aziende sono entusiaste della possibilità di creare sistemi Web per il commercio elettronico elettronico. Ma la realtà è che Internet e il Web sono estremamente vulnerabili. A mano a mano che le aziende si rendono conto di questa realtà, cresce la domanda di servizi Web sicuri.

L'argomento della sicurezza nel Web è molto ampio e da solo basterebbe a riempire un intero volume (alcuni di questi sono consigliati alla fine del capitolo). In questo capitolo si introdurranno i requisiti generali della sicurezza nel Web e quindi ci si concentrerà su due meccanismi standard che stanno assumendo un'importanza crescente nel commercio via Web: SSL/TLS e SET.

17.1 Considerazioni sulla sicurezza del Web

Il World Wide Web è fondamentalmente un'applicazione client/server operante su Internet e reti intranet TCP/IP. Per questo motivo, gli strumenti di sicurezza e i vari approcci di cui

si è parlato finora sono tutti applicabili al problema della sicurezza del Web. Ma come evidenziato in [GARF97], il Web presenta nuove sfide che generalmente non sono valutate adeguatamente nel contesto della sicurezza delle reti e dei computer.

- Internet è bidirezionale. A differenza dei tradizionali mezzi di diffusione, inclusi sistemi di pubblicazione elettronica come il teletext e i risponditori telefonici automatici, il Web è vulnerabile agli attacchi contro i server via Internet.
- Il Web sta diventando sempre più una fonte di promozione particolarmente visibile per le aziende e per i loro prodotti e dunque si trasforma sempre più in una piattaforma per transazioni commerciali. In caso di attacco ai server Web si corrono gravi rischi sia economici che in termini di reputazione.
- Sebbene i browser Web siano molto facili da usare, i server Web siano relativamente facili da configurare e gestire e i contenuti Web siano sempre più facili da sviluppare, il software su cui si basa il tutto è straordinariamente complesso. Questa complessità può nascondere molte potenziali lacune di sicurezza. La breve storia del Web è ricca di esempi di sistemi nuovi, aggiornati e correttamente installati ma ciononostante vulnerabili.
- Un server Web può essere sfruttato come una sorta di trampolino di lancio verso il complesso di computer dell'intera azienda. Una volta ottenuto l'accesso al server Web, un hacker può ottenere l'accesso ai dati e sistemi privati che non fanno parte del servizio Web ma che sono localmente collegati al server.
- Spesso i clienti dei servizi Web sono utenti casuali non addestrati (in termini di sicurezza). Tali utenti non sono necessariamente al corrente dei problemi di sicurezza che esistono e non sono dotati degli strumenti o delle conoscenze indispensabili per prendere le contromisure necessarie.

Minacce alla sicurezza del Web

La Tabella 17.1 presenta un riepilogo dei vari problemi di sicurezza che si devono affrontare quando si usa il Web. Un modo per raggruppare queste minacce consiste nel suddividerle fra attacchi passivi e attivi. Fra gli attacchi passivi vi sono l'intercettazione del traffico di rete fra browser e server e l'accesso a informazioni su un server Web che dovrebbe essere riservato. Fra gli attacchi attivi vi sono la simulazione di altri utenti, la modifica dei messaggi in transito fra client e server e l'alterazione delle informazioni contenute in un sito Web.

Un altro modo per classificare i problemi di sicurezza del Web è in termini di posizione della minaccia: nel server Web, nel browser Web e nel traffico di rete in transito fra il server e il browser. I problemi di sicurezza del server e del browser rientrano nell'ambito della sicurezza dei computer; la prossima parte di questo volume affronta l'argomento della sicurezza dei sistemi in generale ma riguarda anche la sicurezza dei sistemi Web. I problemi di sicurezza del traffico rientrano nell'ambito della sicurezza delle reti e verranno pertanto trattati in questo capitolo.

Tabella 17.1 Le minacce nel Web (RUBI97).

	Minacce	Conseguenze	Contromisure
Integrità	• Modifica dei dati utente • Cavalli di Troia • Modifiche della memoria • Modifiche dei messaggi in transito	• Perdita di informazioni • Violazione della macchina • Vulnerabilità ad altre minacce	• Checksum crittografici
Segretezza	• Intercettazioni in rete • Furto di informazioni dal server • Furto di dati dal client • Informazioni sulla configurazione della rete • Informazioni sulle comunicazioni fra clienti e server	• Perdita di informazioni • Perdita della privacy	• Crittografia, proxy Web
Attacchi denial of service	• Terminazione dei processi degli utenti • Intasamento della macchina con una grande quantità di richieste • Saturazione dei dischi e della memoria • Isolamento della macchina tramite attacchi DNS	• Impedisce all'utente di completare il proprio lavoro	• Difficile da prevenire
Autenticazione	• Simulazione di utenti legittimi • Falsificazione dei dati	• Errata rappresentazione dell'utente • Accettazione di informazioni false ma ritenute valide	• Tecniche crittografiche

Approcci alla sicurezza del traffico Web

È possibile adottare vari approcci alla sicurezza nel Web. Questi approcci offrono servizi simili e, in un certo senso, usano anche meccanismi simili ma differiscono per la loro applicabilità e la loro posizione nell'ambito dello stack di protocolli TCP/IP.

La Figura 17.1 illustra questa differenza. Un modo per fornire la sicurezza nel Web è quello di utilizzare IPSec (Figura 17.1A). Il vantaggio di IPSec consiste nel fatto che è trasparente agli utenti finali e alle applicazioni e rappresenta una soluzione di carattere generale. Inoltre IPSec include una funzionalità di filtraggio, in modo che solo una parte del traffico debba incorrere nel sovraccarico dovuto all'elaborazione IPSec.

Un'altra soluzione di carattere relativamente generale consiste nell'implementare la sicurezza appena sopra il protocollo TCP (Figura 17.1B). Il principale esempio di questo approccio è SSL (Secure Sockets Layer) e il relativo standard Internet chiamato TLS (Transport Layer Security). A questo livello vi sono due possibili implementazioni. Per la soluzione più generale, si può impiegare SSL (o TLS) nell'ambito della famiglia di protocolli sottostanti e

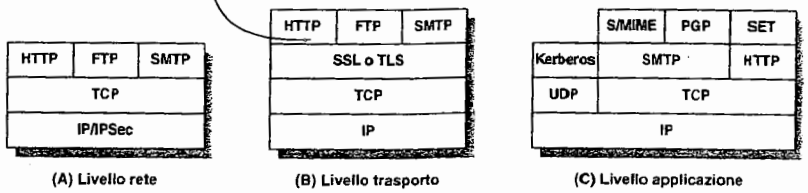


Figura 17.1 Posizione dei sistemi di sicurezza nello stack di protocolli TCP/IP

pertanto in modo trasparente alle applicazioni. Alternativamente, SSL può essere incorporato in specifici pacchetti. Per esempio, i browser Web Netscape e Microsoft Explorer sono dotati di SSL e la maggior parte dei server Web hanno implementato questo protocollo.

I servizi di sicurezza specifici delle applicazioni sono incorporati nelle applicazioni. La Figura 17.1C mostra alcuni esempi di questa architettura. Il vantaggio di questo approccio è che il servizio può essere personalizzato in base alle esigenze specifiche di una determinata applicazione. Nel contesto della sicurezza Web, un esempio importante di questo approccio è SET (Secure Electronic Transaction).[1]

La parte rimanente di questo capitolo è dedicata alla discussione di SSL/TLS e SET.

17.2 I protocolli SSL (Secure Socket Layer) e TLS (Transport Layer Security)

SSL è stato sviluppato da Netscape. La versione 3 del protocollo è stata progettata con un processo di revisione pubblica e sulla base di input dal mondo delle aziende ed è stata pubblicata come bozza Internet. Successivamente, raggiunto un certo consenso per proporre il protocollo come standard Internet, venne costituito il gruppo di lavoro TLS nell'ambito di IETF per sviluppare uno standard comune. Questa prima versione di TLS può essere considerata fondamentalmente come la versione 3.1 di SSL ed è molto simile e compatibile all'indietro con SSL versione 3.

Questa parte del capitolo è dedicata fondamentalmente a SSL versione 3 (SSLv3). Verso la fine di questa parte del capitolo verranno descritte le principali differenze fra SSLv3 e TLS.

L'architettura SSL

L'architettura SSL è stata progettata per impiegare TCP con un servizio affidabile end-to-end. SSL non è un unico protocollo ma è costituito da due livelli di protocolli, come indicato nella Figura 17.2.

[1] La Figura 17.1C mostra SET sopra HTTP, un'implementazione molto comune. In alcune implementazioni, invece, SET utilizza direttamente TCP.

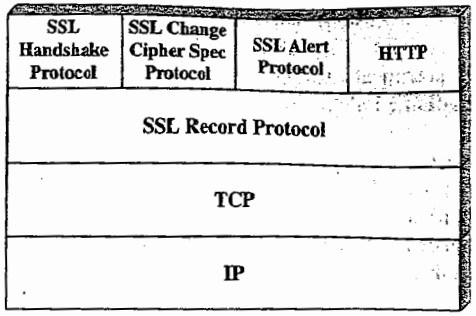


Figura 17.2 Lo stack di protocolli SSL.

Il protocollo SSL Record Protocol fornisce i servizi di sicurezza di base per i vari protocolli di livello superiore. In particolare, il protocollo HTTP (Hypertext Transfer Protocol), che fornisce il servizio di trasferimento per le interazioni Web client/server, può operare sopra SSL. Nell'ambito di SSL sono definiti tre protocolli di più alto livello: Handshake Protocol, Change Cipher Spec Protocol e Alert Protocol. Questi protocolli SSL vengono utilizzati nella gestione degli scambi SSL e verranno trattati più avanti in questa stessa parte del capitolo.

Due concetti importanti in SSL sono la sessione SSL e la connessione SSL che sono definiti nel seguente modo.

- **Connessione:** una connessione è una forma di trasporto (nella definizione del modello a livelli OSI) che fornisce un determinato tipo di servizio. Per SSL, tali connessioni sono relazioni fra nodi paritari. Le connessioni sono transitorie. Ogni connessione è associata a una sola sessione.
- **Sessione:** una sessione SSL è un'associazione fra un client e un server. Le sessioni vengono create dal protocollo Handshake e definiscono un insieme di parametri di sicurezza crittografica che possono essere condivisi fra più connessioni. Le sessioni vengono utilizzate per evitare di svolgere la costosa negoziazione di nuovi parametri di sicurezza per ciascuna connessione.

Ogni coppia di parti (applicazioni come HTTP su client e server) può intrattenere più connessioni sicure. In teoria vi potrebbero anche essere più sessioni simultanee fra le parti ma in realtà questa funzionalità non viene utilizzata.

A ciascuna sessione vengono associati più stati. Una volta attivata una sessione, vi è uno stato operativo corrente per la lettura e la scrittura (ovvero per la ricezione e l'invio). Inoltre, durante il protocollo di Handshake, vengono creati degli stati provvisori di lettura e scrittura. Alla conclusione del protocollo di Handshake, gli stati provvisori diventano stati correnti.

Uno stato di sessione è definito dai seguenti parametri (definizioni tratte dalle specifiche SSL).

- **Session identifier:** una sequenza di byte arbitraria scelta dal server per identificare lo stato di una sessione attiva o riattivabile.
- **Peer certificate:** il certificato X509.v3 del nodo. Questo elemento può essere nullo.
- **Compression method:** l'algoritmo utilizzato per comprimere i dati prima della crittografia.
- **Cipher spec:** specifica l'algoritmo di crittografia dei dati (null, AES e così via) e l'algoritmo hash (come MD5 o SHA-1) utilizzato per il calcolo del codice MAC. Inoltre definisce gli attributi crittografici, per esempio hash_size.
- **Master secret:** un codice segreto di 48 byte condiviso dal client e dal server.
- **Is resumable:** un flag che indica se la sessione può essere utilizzata per iniziare nuove connessioni.

Lo stato di una connessione è definito dai seguenti parametri.

- **Server and client random:** sequenze di byte scelte dal server e dal client per ciascuna connessione.
- **Server write MAC secret:** la chiave segreta utilizzata nelle operazioni MAC per i dati inviati dal server.
- **Client write MAC secret:** la chiave segreta utilizzata nelle operazioni MAC sui dati inviati dal client.
- **Server write key:** la chiave di crittografia convenzionale per i dati crittografati dal server e decrittografati dal client.
- **Client write key:** la chiave di crittografia convenzionale per i dati crittografati dal client e decrittografati dal server.
- **Initialization vectors:** quando viene usata una cifratura a blocchi in modalità CBC, per ciascuna chiave viene mantenuto un vettore di inizializzazione (IV). Questo campo viene inizializzato dal protocollo SSL Handshake. Successivamente il blocco di testo cifrato finale di ciascun record viene preservato per essere utilizzato come vettore di inizializzazione del record seguente.
- **Sequence numbers:** ciascuna parte gestisce numeri sequenziali distinti per i messaggi trasmessi e ricevuti per ciascuna connessione. Quando una parte invia o riceve un messaggio di change cipher spec, il numero di sequenza appropriato viene impostato a 0. I numeri di sequenza non devono superare il valore $2^{64} - 1$.

Il protocollo SSL Record

Il protocollo SSL Record fornisce due servizi per le connessioni SSL.

- **Segretezza:** il protocollo Handshake definisce una chiave segreta condivisa utilizzata per la crittografia convenzionale del payload SSL.
- **Integrità del messaggio:** il protocollo Handshake definisce anche una chiave segreta condivisa che viene utilizzata per creare un codice MAC (Message Authentication Code).

La Figura 17.3 indica il funzionamento generale del protocollo SSL Record. Questo protocollo accetta il messaggio da trasmettere, frammenta i dati in blocchi di dimensioni appropriate, opzionalmente comprime i dati, applica un codice MAC, esegue la crittografia, aggiunge l'intestazione e trasmette l'unità risultante in un segmento TCP. I dati ricevuti

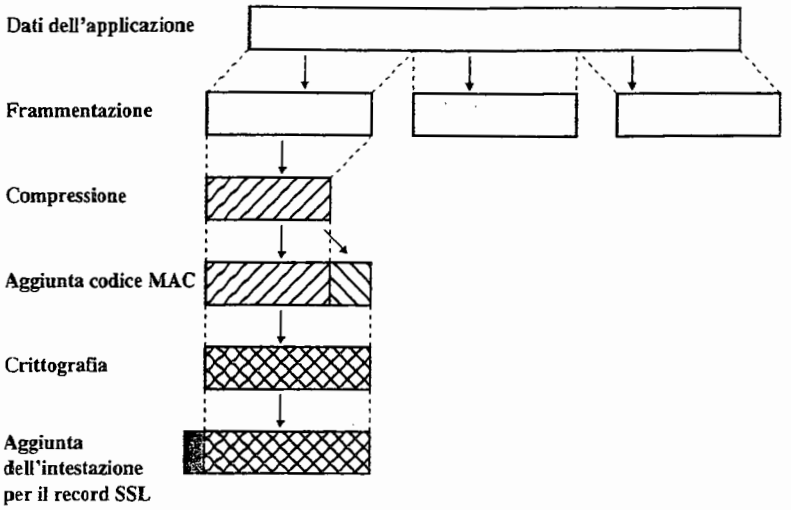


Figura 17.3 Funzionamento del protocollo SSL Record.

vengono decrittografati, verificati, decompressi e assemblati e quindi consegnati agli utenti di livello più elevato.

Il primo passo è la **frammentazione**. Ciascun messaggio dal livello superiore viene frammentato in blocchi di 2^{14} byte (16 384 byte) o meno. Poi viene eventualmente applicata la **compressione**. La compressione non deve perdere informazioni e non deve aumentare la lunghezza del contenuto di più di 1024 byte.[2] In SSLv3 (e nella versione corrente di TLS) non è specificato alcun algoritmo di compressione e dunque l'algoritmo di compressione predefinito è nullo.

Il passo successivo consiste nel calcolo del codice **MAC** (Message Authentication Code) sui dati compressi. A tale scopo viene utilizzata una chiave segreta. Il calcolo è definito nel seguente modo.

hash(MAC_write_secret || pad_2 || hash(MAC_write_secret || pad_1 || seq_num || SSLCompressed.type || SSLCompressed.length || SSLCompressed.fragment))

dove:

|| =	concatenamento
MAC_write_secret =	chiave segreta condivisa
hash =	algoritmo crittografico hash; MD5 o SHA-1.
pad_1 =	il byte 0x36 (0011 0110) ripetuto 48 volte (384 bit) per MD5 e 40 volte (320 bit) per SHA-1.

[2] Naturalmente la compressione dovrebbe ridurre e non aumentare le dimensioni dei dati. Tuttavia, per blocchi molto piccoli, le convenzioni di formattazione possono fare in modo che l'algoritmo di compressione produca un output di maggiori dimensioni dell'input.

pad_2 = il byte 0x5C (0101 1100) ripetuto 48 volte per MD5 e 40 volte per SHA-1.
seq_num = il numero sequenziale di questo messaggio.
SSLCompressed.type = il protocollo di livello superiore utilizzato per elaborare il frammento.
SSLCompressed.length = la lunghezza del frammento compresso.
SSLCompressed.fragment = il frammento compresso (se non viene usata la compressione, si tratta del frammento di testo in chiaro).

Si noti che questo è molto simile all'algoritmo HMAC definito nel Capitolo 12. La differenza consiste nel fatto che i due "pad" in SSLv4 sono concatenati mentre in HMAC sono sottoposti a uno XOR. L'algoritmo MAC di SSLv3 si basa sulla bozza Internet originale per HMAC che usava il concatenamento. La versione finale di HMAC, definita nel documento RFC 2104, usa l'operatore XOR.

Quindi il messaggio compresso più il codice MAC vengono **crittografati** utilizzando la crittografia simmetrica. La crittografia non può aumentare la lunghezza del contenuto di più di 1024 byte e dunque la lunghezza totale non può superare $2^{14} + 2048$. Possono essere utilizzati i seguenti algoritmi di crittografia.

Crittografia a blocchi		Crittografia di flussi	
Algoritmo	**Dimensione chiave**	**Algoritmo**	**Dimensione chiave**
AES	128, 256	RC4-40	40
IDEA	128	RC4-128	128
RC2-40	40		
DES-40	40		
DES	56		
3DES	168		
Fortezza	80		

Fortezza può essere utilizzato in uno schema di crittografia per Smart-Card.

Per la crittografia di flussi, vengono crittografati il messaggio compresso più il codice MAC. Si noti che il codice MAC viene calcolato prima della crittografia e poi viene crittografato insieme al testo in chiaro semplice o compresso.

Per la crittografia a blocchi, possono essere aggiunti dei pad dopo il codice MAC prima della crittografia. La parte di riempimento è costituita dal numero di byte di riempimento seguiti da un byte che indica la lunghezza del riempimento. Il numero di byte di riempimento è il più piccolo valore tale che la dimensione totale dei dati da crittografare (testo in chiaro più codice MAC più riempimento) sia un multiplo della lunghezza del blocco di testo cifrato. Un esempio è un testo in chiaro (o testo compresso se viene utilizzata la compressione) di 58 byte con un codice MAC di 20 byte (utilizzando SHA-1) che viene

crittografato utilizzando un blocco della lunghezza di 8 byte (per esempio in DES). Aggiungendo il byte padding.length, si ottiene un totale di 79 byte. Per rendere il totale un multiplo intero di 8, viene aggiunto un solo byte.

L'ultimo passo di elaborazione del protocollo SSL Record consiste nell'aggiunta di un'intestazione iniziale costituita dai seguenti campi.

- **Content Type (8 bit):** il protocollo di livello superiore utilizzato per elaborare il frammento incluso.
- **Major Version (8 bit):** la versione major di SSL in uso. Per SSLv3 il valore è 3.
- **Minor Version (8 bit):** la versione minor in uso. Per SSLv3 il valore è 0.
- **Compressed Length (16 bit):** la lunghezza in byte del frammento di testo in chiaro (o del frammento compresso se viene usata la compressione). Il valore massimo è $2^{14} + 2048$.

I tipi di contenuti definiti sono change_cipher_spec, alert, handshake e application_data. I primi tre sono protocolli specifici di SSL di cui si parlerà fra breve. Si noti che non viene fatta alcuna distinzione fra le varie applicazioni (per esempio HTTP) che possono usare SSL. Il contenuto dei dati creati da tali applicazioni è totalmente indipendente da SSL.

La Figura 17.4 mostra il formato del record SSL.

Il protocollo Change Cipher Spec

Il protocollo Change Cipher Spec è uno dei tre protocolli specifici di SSL che utilizzano il protocollo SSL Record, ed è anche il più semplice. Questo protocollo è costituito da un unico messaggio (vedere la Figura 17.5A) costituito da un unico byte contenente il valore 1. L'unico scopo di questo messaggio è quello di fare in modo che lo stato provvisorio

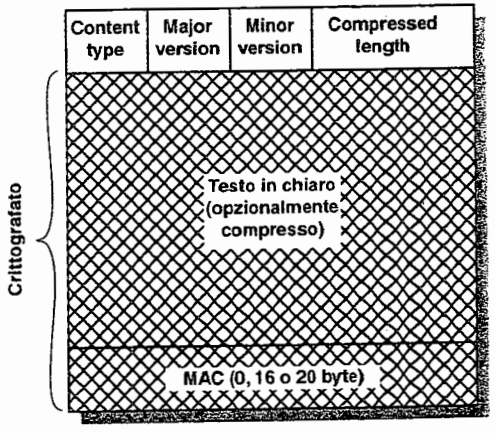


Figura 17.4 Il formato del record SSL.

venga copiato nello stato corrente, aggiornando e quindi attivando la cifratura che verrà utilizzata in questa connessione.

Il protocollo Alert

Il protocollo Alert viene utilizzato per trasmettere allarmi SSL all'entità peer. Come per altre applicazioni che utilizzano SSL, i messaggi alert vengono compressi e crittografati, come specificato dallo stato corrente.

Ciascun messaggio di questo protocollo è costituito da due byte (Figura 17.5B). Il primo assume il valore warning(1) o fatal(2) per indicare la gravità del messaggio. Se il livello è fatal, SSL chiude immediatamente la connessione. Altre connessioni nella stessa sessione possono continuare ma non si possono attivare nuove connessioni in questa sessione. Il secondo byte contiene un codice che indica l'allarme specificato. Innanzitutto si elencano gli allarmi che sono sempre irreversibili (definizioni tratte dalle specifiche SSL).

- **unexpected_message:** è stato ricevuto un messaggio inappropriato.
- **bad_record_mac:** è stato ricevuto un codice MAC errato.
- **decompression_failure:** la funzione di decompressione ha ricevuto un input errato (per esempio incapacità di comprimere o decomprimere a una dimensione superiore alla massima consentita).
- **handshake_failure:** il mittente non è stato in grado di negoziare un insieme di parametri di sicurezza accettabile date le opzioni disponibili.
- **illegal_parameter:** un campo in un messaggio di handshake era oltre i limiti consentiti o era incoerente rispetto agli altri campi.

Ecco gli altri allarmi.

- **close_notify:** notifica al destinatario che il mittente non invierà altri messaggi in questa connessione. Ciascuna parte deve inviare un allarme close_notify prima di chiudere il lato di scrittura di una connessione.

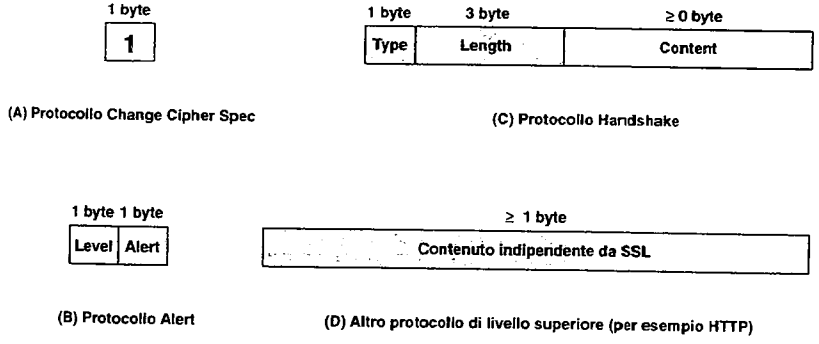


Figura 17.5 Il payload del protocollo SSL Record.

- **no_certificate:** può essere inviato in risposta a una richiesta di certificato nel caso in cui non sia disponibile alcun certificato appropriato.
- **bad_certificate:** un certificato ricevuto era alterato (per esempio conteneva una firma che non poteva essere verificata).
- **unsupported_certificate:** il tipo del certificato ricevuto non è supportato.
- **certificate_revoked:** il certificato è stato revocato dal suo firmatario.
- **certificate_expired:** il certificato è scaduto.
- **certificate_unknown:** si è verificato un errore non specificato nell'elaborazione del certificato che lo rende inaccettabile.

Il protocollo Handshake

La parte più complessa di SSL è il protocollo Handshake. Questo protocollo consente al server e al client di autenticarsi l'un l'altro e di negoziare un algoritmo di crittografia e MAC, e le chiavi crittografiche da utilizzare per proteggere i dati inviati in un record SSL. Il protocollo Handshake viene utilizzato prima della trasmissione di qualsiasi dato dell'applicazione.

Il protocollo Handshake è costituito da una serie di messaggi scambiati dal client e dal server. Questi messaggi hanno il formato rappresentato nella Figura 17.5C. Ogni messaggio contiene tre campi.

- **Type (1 byte):** indica un messaggio fra i dieci elencati nella Tabella 17.2.
- **Length (3 byte):** la lunghezza del messaggio in byte.
- **Content (≥ 0 byte):** i parametri associati a questo messaggio; anch'essi elencati nella Tabella 17.2.

Tabella 17.2 Tipi di messaggio e parametri del protocollo SSL Handshake.

Tipo messaggio	Parametri
hello_request	null
client_hello	version, random, session id, cipher suite, compression method
server_hello	version, random, session id, cipher suite, compression method
certificate	chain of X.509v3 certificates
server_key_exchange	parameters, signature
certificate_request	type, authorities
server_done	null
certificate_verify	signature
client_key_exchange	parameters, signature
finished	hash value

La Figura 17.6 mostra lo scambio iniziale necessario per stabilire una connessione logica fra il client e il server. Lo scambio si svolge in pratica in quattro fasi.

Fase 1. Inizializzazione delle funzionalità di sicurezza
Questa fase viene utilizzata per avviare una connessione logica e per stabilire le funzionalità di sicurezza che possono essere impiegate. Lo scambio viene iniziato dal client che invia il messaggio **client_hello** con i seguenti parametri.

- **Version:** versione di SSL più elevata fra quelle utilizzabili dal client.

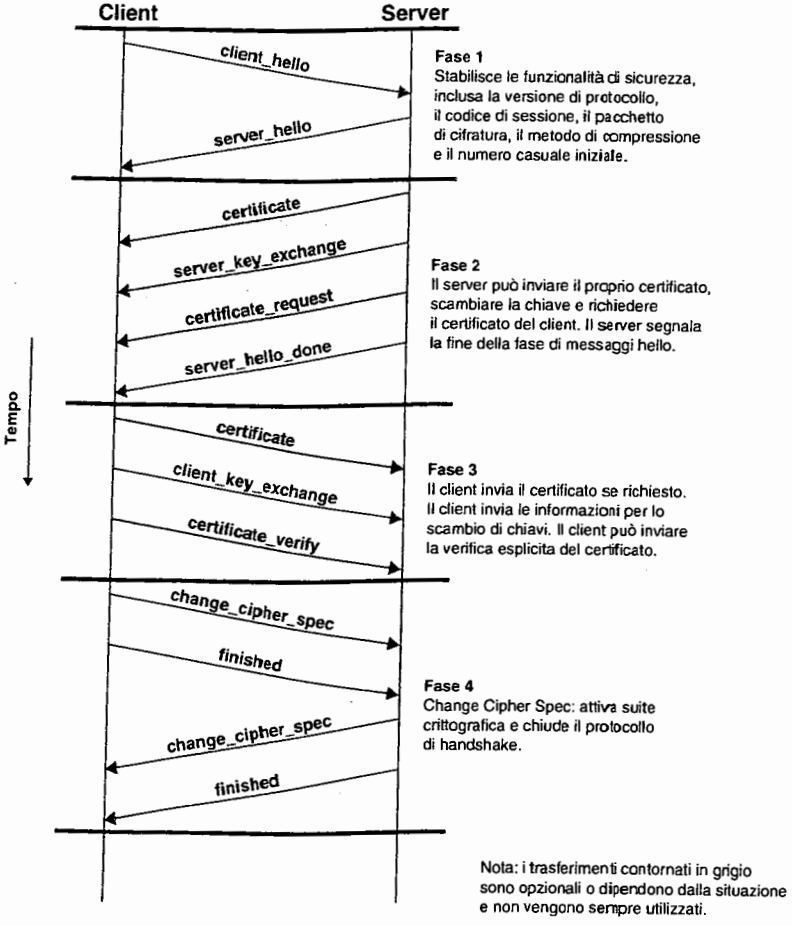


Figura 17.6 Funzionamento del protocollo Handshake.

- **Random:** struttura casuale generata dal client costituita da un timestamp di 32 bit e 28 byte prodotti da un generatore sicuro di numeri casuali. Questi valori fungono da codice nonce e vengono utilizzati durante lo scambio delle chiavi per impedire gli attacchi a replay.
- **Session ID:** un identificatore di sessione di lunghezza variabile. Un valore diverso da zero indica che il client vuole aggiornare i parametri di una connessione esistente o creare una nuova connessione in questa sessione. Il valore 0 indica che il client vuole stabilire una nuova connessione in una nuova sessione.
- **CipherSuite:** elenco che contiene gli algoritmi crittografici supportati dal client, in ordine decrescente di preferenza. Ciascun elemento della lista (ciascuna suite di crittografia) definisce sia un algoritmo di scambio delle chiavi che un CipherSpec (di cui si parlerà più avanti).
- **Compression Method:** elenco dei metodi di compressione supportati dal client.

Dopo aver inviato il messaggio client_hello, il client attende il messaggio **server_hello** che contiene gli stessi parametri del proprio messaggio client_hello. Per il messaggio server_hello si applicano le seguenti convenzioni. Il campo Version contiene la più bassa fra la versione suggerita dal client e la versione più elevata supportata dal server. Il campo Random viene generato dal server ed è indipendente dal campo Random del client. Se il campo SessionID del client è diverso da zero, il server usa lo stesso valore, altrimenti il campo SessionID del server contiene il valore per una nuova sessione. Il campo CipherSuite contiene il pacchetto di cifratura selezionato dal server fra quelli proposti dal client. Il campo Compression contiene il metodo di compressione selezionato dal server fra quelli proposti dal client.

Il primo elemento del parametro Cipher Suite è il metodo per lo scambio delle chiavi (ovvero il metodo con cui vengono scambiate le chiavi crittografiche per la crittografia convenzionale e il codice MAC). Sono supportati i seguenti metodi per lo scambio delle chiavi.

- **RSA:** la chiave segreta viene crittografata con la chiave pubblica RSA del destinatario. Si deve rendere disponibile un certificato di chiave pubblica per la chiave del destinatario.
- **Fixed Diffie-Hellman:** questo è uno scambio della chiave Diffie-Hellman in cui il certificato del server contiene i parametri pubblici Diffie-Hellman firmati dall'autorità di certificazione. In pratica il certificato di chiave pubblica contiene i parametri della chiave pubblica Diffie-Hellman. Il client fornisce i propri parametri della chiave pubblica Diffie-Hellman in un certificato (se è richiesta l'autenticazione del client) oppure in un messaggio per lo scambio delle chiavi. Questo metodo produce come risultato una chiave segreta fissa fra i due nodi basata sul calcolo Diffie-Hellman utilizzando le chiavi pubbliche fisse.
- **Ephemeral Diffie-Hellman:** questa tecnica viene utilizzata per creare chiavi segrete effimere (temporanee, monouso). In questo caso, vengono scambiate le chiavi pubbliche Diffie-Hellman, firmate utilizzando la chiave privata RSA o DSS del mittente. Il destinatario può verificare la firma utilizzando la corrispondente chiave pubblica. I certificati vengono utilizzati per autenticare le chiavi pubbliche. Questa sembra la più sicura delle tre opzioni Diffie-Hellman in quanto produce una chiave temporanea autenticata.
- **Anonymous Diffie-Hellman:** viene utilizzato l'algoritmo base Diffie-Hellman, senza autenticazione. Ovvero ciascuna parte invia all'altra parte i propri parametri pubblici

Diffie-Hellman senza autenticazione. Questo approccio è vulnerabile ad attacchi man-in-the-middle in cui un estraneo esegue uno scambio anonimo Diffie-Hellman con entrambe le parti.
- **Fortezza:** la tecnica definita per lo schema Fortezza.

Dopo la definizione del metodo per lo scambio delle chiavi, si trova CipherSpec, che include i seguenti campi.

- **CipherAlgorithm:** uno qualsiasi degli algoritmi menzionati in precedenza: RC4, RC2, DES, 3DES, DES40, IDEA, Fortezza.
- **MACAlgorithm:** MD5 o SHA-1.
- **CipherType:** Stream o Block.
- **IsExportable:** True o False
- **HashSize:** 0, 16 (per MD5) o 20 (per SHA-1) byte.
- **Key Material:** una sequenza di byte che contiene i dati utilizzati per la generazione delle chiavi di scrittura.
- **IV Size:** le dimensioni del vettore di inizializzazione per la crittografia CBC (Cipher Block Chaining).

Fase 2. Autenticazione del server e scambio delle chiavi

Il server inizia questa fase inviando il proprio certificato, sempre che debba essere autenticato; il messaggio contiene un certificato X.509 oppure una catena di certificati. Il messaggio **certificate** è obbligatorio per ogni metodo concordato per lo scambio di chiavi tranne che per lo scambio Diffie-Hellman anonimo. Si noti che se viene utilizzato lo scambio fiXed Diffie-Hellman, questo messaggio certificate funge da messaggio per lo scambio delle chiavi del server poiché contiene i parametri Diffie-Hellman pubblici del server.

Poi, se necessario, può essere inviato un messaggio **server_key_exchange**. Questo non è necessario in due casi: quando il server ha inviato un certificato con i parametri fiXed Diffie-Hellman o quando deve essere utilizzato lo scambio delle chiavi RSA. Il messaggio server_key_exchange è obbligatorio nei seguenti casi.

- **Anonymous Diffie-Hellman:** il contenuto del messaggio è costituito dai due valori globali Diffie-Hellman (un numero primo e una sua radice primitiva) più la chiave Diffie-Hellman pubblica del server (vedere la Figura 10.7).
- **Ephemeral Diffie-Hellman:** il contenuto del messaggio include i tre parametri Diffie-Hellman necessari per la forma Diffie-Hellman anonima più una firma di questi parametri.
- **Scambio delle chiavi RSA in cui il server usa RSA ma ha una chiave RSA di sola firma:** in questo caso il client non può semplicemente inviare una chiave segreta crittografata con la chiave pubblica del server. Al contrario il server deve creare una coppia temporanea di chiavi RSA pubblica/privata e utilizzare il messaggio server_key_exchange per inviare la chiave pubblica. Il contenuto del messaggio include i due parametri della chiave pubblica temporanea RSA (esponente e modulo, vedere la Figura 9.5), più una firma di questi parametri.
- **Fortezza.**

Occorre fornire ulteriori dettagli sulle firme. Come di consueto, una firma viene creata prendendo il valore hash di un messaggio e crittografandolo con la chiave privata del mittente. In questo caso il codice hash è definito nel seguente modo:

hash(ClientHello.random || ServerHello.random || ServerParams)

Dunque il codice hash non copre solo i parametri Diffie-Hellman o RSA ma anche i due nonce dei messaggi hello iniziali. Questo evita gli attacchi a replay. Nel caso di una firma DSS, il calcolo hash viene eseguito utilizzando l'algoritmo SHA-1. Nel caso di una firma RSA, vengono calcolati entrambi i valori hash MD5 e SHA-1 e il concatenamento di questi due valori hash (36 byte) viene crittografato utilizzando la chiave privata del server.

Quindi un server non anonimo (un server che non utilizza la forma Diffie-Hellman anonima) può richiedere un certificato al client. Il messaggio **certificate_request** include due parametri: certificate_type e certificate_authorities. Il certificate_type indica l'algoritmo a chiave pubblica e il suo impiego.

- RSA, solo firma.
- DSS, solo firma.
- RSA per fiXed Diffie-Hellman; in questo caso la firma viene utilizzata solo per l'autenticazione inviando un certificato firmato con RSA.
- DSS per fiXed Diffie-Hellman; usato anch'esso solo per l'autenticazione.
- RSA per Ephemeral Diffie-Hellman.
- DSS per Ephemeral Diffie-Hellman.
- Fortezza.

Il secondo parametro del messaggio certificate_request è un elenco delle autorità di certificazione accettate.

L'ultimo messaggio della Fase 2 è obbligatorio ed è il messaggio **server_done** che viene inviato dal server per indicare la fine dei messaggi di hello del server. Dopo aver inviato questo messaggio, il server attende una risposta del client. Questo messaggio non ha alcun parametro.

Fase 3. Autenticazione del client e scambio delle chiavi

Dopo aver ricevuto il messaggio server_done, il client deve verificare che il server abbia fornito un certificato valido (se richiesto) e deve controllare che i parametri di server_hello siano accettabili. Se tutto è soddisfacente, il client invia al server uno o più messaggi.

Se il server ha richiesto un certificato, il client inizia questa fase inviando un messaggio **certificate**. Se non è disponibile alcun certificato adatto, il client invia l'allarme no_certificate.

Poi viene il messaggio **client_key_exchange** che deve essere inviato in questa fase. Il contenuto del messaggio dipende dal tipo di scambio delle chiavi.

- **RSA:** il cliente genera un valore *segreto pre-master* di 48 byte e ne esegue la crittografia con la chiave pubblica tratta dal certificato del server o con la chiave RSA temporanea tratta da un messaggio server_key_exchange. Il suo uso per calcolare il codice segreto master verrà descritto più avanti.
- **Ephemeral o Anonymous Diffie-Hellman:** vengono inviati i parametri pubblici Diffie-Hellman del client.

- **Fixed Diffie-Hellman:** i parametri Diffie-Hellman pubblici del client sono stati precedentemente inviati in un messaggio certificate e dunque il contenuto di questo messaggio è nullo.
- **Fortezza:** vengono inviati i parametri Fortezza del client.

Infine, in questa fase, il client può inviare un messaggio **certificate_verify** per fornire una verifica esplicita di un certificato del client. Questo messaggio viene inviato solo dopo un certificato del client che ha funzionalità di firma (ovvero tutti i certificati tranne quelli contenenti parametri fixed Diffie-Hellman). Questo messaggio firma un codice hash sulla base dei messaggi precedenti, come indicato di seguito:

CertificateVerify.signature.md5_hash
MD5(master_secret || pad_2 || MD5(handshake_messages || master_secret || pad_1));
Certificate.signature.sha_hash
SHA(master_secret || pad_2 || SHA(handshake_messages || master_secret || pad_1));

dove pad_1 e pad_2 sono i valori definiti in precedenza per il codice MAC, handshake_messages fa riferimento a tutti i messaggi del protocollo Handshake inviati o ricevuti a partire da client_hello (escluso questo messaggio) e master_secret è il codice segreto calcolato la cui costruzione verrà descritta più avanti in questa parte del capitolo. Se la chiave privata dell'utente è DSS, viene utilizzata per crittografare il codice hash SHA-1. Se la chiave privata dell'utente è RSA, viene utilizzata per crittografare il concatenamento dei codici hash MD5 e SHA-1. In entrambi i casi lo scopo è quello di verificare la proprietà della chiave privata da parte del client. Anche se qualcuno dovesse usare fraudolentemente il certificato del client, non sarebbe in grado di inviare questo messaggio.

Fase 4. Fine

Questa fase completa l'impostazione di una connessione sicura. Il client invia un messaggio **change_cipher_spec** e copia il CipherSpec temporaneo nel CipherSpec corrente. Si noti che questo messaggio non è parte del protocollo Handshake ma viene inviato utilizzando il protocollo Change Cipher Spec. Il client invia immediatamente il messaggio **finished** con i nuovi algoritmi, chiavi e segreti. Il messaggio finished verifica che i processi di scambio delle chiavi e di autenticazione abbiano avuto successo. Il contenuto del messaggio finished è il concatenamento dei due valori hash:

MD5(master_secret || pad2 || MD5(handshake_messages || Sender || master_secret || pad1))
SHA(master_secret || pad2 || SHA(handshake_messages || Sender || master_secret || pad1))

dove Sender è un codice che identifica che il mittente è il client e handshake_messages è costituito da tutti i dati di tutti i messaggi di handshake fino a questo messaggio escluso.

In risposta a questi due messaggi, il server invia il proprio messaggio change_cipher_spec, trasferisce il CipherSpec provvisorio nel CipherSpec corrente e invia il proprio messaggio finished. A questo punto la procedura di handshake è completa e il client e il server possono iniziare a scambiarsi i dati a livello applicazione.

Calcoli crittografici

A questo punto occorre trattare altri due argomenti importanti: la creazione di un valore segreto master condiviso tramite lo scambio di chiavi e la generazione dei relativi parametri crittografici.

La creazione del valore segreto master

Il valore segreto master condiviso è un valore monouso di 48 byte (384 bit) generato per una sessione tramite scambio di chiavi sicuro. La creazione si svolge in due fasi. Innanzitutto viene scambiato il valore pre_master_secret. Poi viene calcolato il valore master_secret da entrambe le parti. Per lo scambio del valore pre_master_secret vi sono due possibilità.

- **RSA:** il client genera un valore pre_master_secret di 48 byte, crittografato con la chiave RSA pubblica del server e la invia al server. Il server esegue la decrittografia del testo cifrato utilizzando la propria chiave privata per recuperare il valore pre_master_secret.
- **Diffie-Hellman:** sia il client che il server generano una chiave pubblica Diffie-Hellman. Dopo aver scambiato queste chiavi, ognuno dei due svolge il calcolo Diffie-Hellman per creare il valore pre_master_secret.

Entrambi i lati calcolano il valore master_secret nel seguente modo:

master_secret = MD5(pre_master_secret || SHA('A' || pre_master_secret ||
ClientHello.random || ServerHello.random)) ||
MD5(pre_master_secret || SHA('BB' || pre_master_secret ||
ClientHello.random || ServerHello.random)) ||
MD5(pre_master_secret || SHA('CCC' || pre_master_secret ||
ClientHello.random || ServerHello.random))

dove ClientHello.random e ServerHello.random sono i due valori nonce scambiati nei messaggi hello iniziali.

Generazione dei parametri crittografici

CipherSpecs richiede un segreto MAC di scrittura per il client, un segreto MAC di scrittura per il server, una chiave di scrittura per il client, una chiave di scrittura per il server, un vettore di inizializzazione di scrittura per il client e un vettore di inizializzazione di scrittura per il server che vengono generati dal valore segreto master in questo ordine. Questi parametri vengono generati dal valore segreto master tramite calcoli hash che producono parametri di lunghezza appropriata.

La generazione del materiale per la chiave dal valore segreto master utilizza lo stesso formato per la generazione del valore segreto master a partire dal pre-master:

key_block = MD5(master_secret || SHA('A' || master_secret ||
ServerHello.random || ClientHello.random)) ||
MD5(master_secret || SHA('BB' || master_secret ||
ServerHello.random || ClientHello.random)) ||

MD5(master_secret || SHA('CCC' || master_secret ||
ServerHello.random || ClientHello.random)) || . . .

fino a generare un output di dimensioni sufficienti. Il risultato di questa struttura algoritmica è una funzione pseudocasuale. Si può considerare il valore master_secret come il seme pseudocasuale per questa funzione. I numeri casuali del client e del server servono per complicare l'analisi crittografica (a questo proposito si consulti il Capitolo 18).

Transport Layer Security

TLS è un'iniziativa di standardizzazione di IETF il cui obiettivo è di produrre una versione standard per Internet di SSL. TLS è stata definita come Proposed Internet Standard nel documento RFC 2246 che è molto simile a SSLv3. In questa parte del capitolo si evidenzieranno le differenze.

Numero di versione

Il valore TLS Record Format coincide con il valore SSL Record Format (Figura 17.4) e i campi dell'intestazione hanno lo stesso significato. L'unica differenza è nei valori della versione. Per la versione corrente di TLS, Major Version è 3 e Minor Version è 1.

Codice MAC (Message Authentication Code)

Vi sono due differenze fra gli schemi MAC di SSLv3 e TLS: l'algoritmo utilizzato e il tipo di calcolo MAC. TLS utilizza l'algoritmo HMAC definito nel documento RFC 2104. Come si è detto nel Capitolo 12, HMAC è definito nel seguente modo:

$$\text{HMAC}_K(M) = \text{H}[(K^+ \oplus \text{opad}) \,||\, \text{H}[(K^+ \oplus \text{ipad}) \,||\, M]]$$

dove:

- H = funzione hash incorporata (per TLS si tratta di MD5 o SHA-1)
- M = messaggio di input per HMAC.
- K^+ = chiave segreta riempita di valori "0" sulla sinistra in modo che il risultato sia uguale alla lunghezza di blocco del codice hash (per MD5 e SHA-1, la lunghezza di blocco è di 512 bit)
- ipad = 00110110 (36 in esadecimale) ripetuto per 64 volte (512 bit)
- opad = 01011100 (5C in esadecimale) ripetuto per 64 volte (512 bit)

SSLv3 utilizza lo stesso algoritmo, tranne per il fatto che i byte di riempimento sono concatenati con la chiave segreta (invece di eseguire uno XOR con la chiave segreta) e riempiti fino a raggiungere la lunghezza del blocco. Il livello di sicurezza dovrebbe essere simile in entrambi i casi.

Per TLS, il calcolo MAC prevede i campi indicati nella seguente espressione:

HMAC_hash(MAC_write_secret, seq_num || TLSCompressed.type ||
TLSCompressed.version || TLSCompressed.length || TLSCompressed.fragment))

Il calcolo MAC copre tutti i campi coperti dal calcolo SSLv3 più il campo TLSCompressed.version che è la versione del protocollo impiegato.

Funzione pseudocasuale

TLS utilizza una funzione pseudocasuale chiamata PRF che espande i valori segreti in blocchi di dati per la generazione o la convalida della chiave. L'obiettivo è quello di utilizzare un valore segreto condiviso relativamente compatto per generare lunghi blocchi di dati sicuri da ogni genere di attacco alle funzioni hash e i ai codici MAC. La funzione PRF si basa sulla seguente funzione di espansione dei dati (Figura 17.7):

P_hash(secret, seed) = HMAC_hash(secret, A(1) || seed) ||
HMAC_hash(secret, A(2) || seed) ||
HMAC_hash(secret, A(3) || seed) || . . .

dove A() è definito come:

A(0) = seme
A(i) = HMAC_hash(secret, A(i – 1))

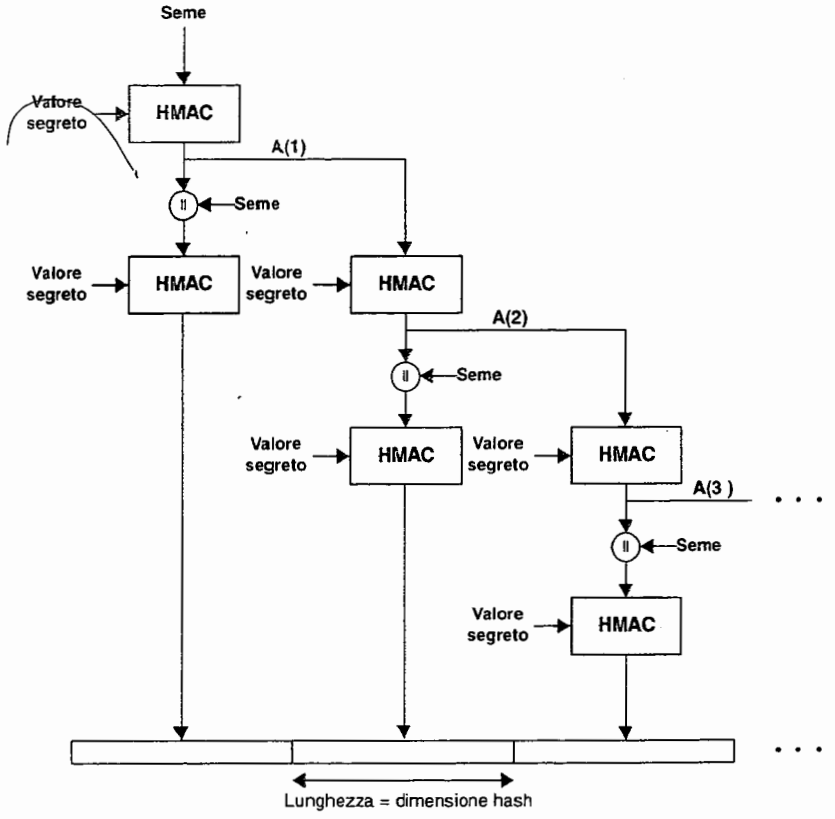


Figura 17.7 La funzione TLS P_hash (valore segreto, seme).

La funzione di espansione dei dati utilizza l'algoritmo HMAC con MD5 o SHA-1 come funzione hash di base. Come si può vedere, P_hash può essere iterata tutte le volte che è necessario per produrre la quantità desiderata di dati. Per esempio, se P_SHA-1 dovrebbe essere utilizzato per generare 64 byte di dati, dovrà essere iterato per quattro volte producendo 80 byte di dati, dei quali ne verranno eliminati 16. In questo caso, anche P_MD5 dovrà essere iterato quattro volte producendo esattamente 64 byte di dati. Si noti che ciascuna iterazione prevede due esecuzioni di HMAC, ognuna delle quali a sua volta prevede due esecuzioni dell'algoritmo hash sottostante.
Per rendere PRF più sicura possibile, la funzione utilizza due algoritmi hash in modo da garantire la sicurezza, sempre che uno dei due algoritmi rimanga sicuro. PRF è definita come:

PRF(secret, label, seed) = P_MD5(S1, label || seed) ⊕ P_SHA-1(S2, label || seed)

PRF prende come input un valore segreto, un'etichetta di identificazione e un seme e produce un output di lunghezza arbitraria. L'output viene creato suddividendo il valore segreto in due parti (S1 e S2) ed eseguendo P_hash su ciascuna metà, utilizzando MD5 su una e SHA-1 sull'altra. Ai due risultati viene applicato un OR esclusivo per produrre l'output; per questo scopo, P_MD5 dovrà generalmente essere iterato più volte rispetto a P_SHA-1 per produrre una quantità di dati uguale per l'input della funzione di OR esclusivo.

Codici di allarme

TLS supporta tutti i codici di allarme definiti in SSLv3 con l'eccezione di no_certificate, ai quali ne aggiunge alcuni altri, fra cui i seguenti che sono sempre irreversibili.

- **decryption_failed:** un testo cifrato è stato decrittografato in modo non valido; poteva non essere un multiplo della lunghezza del blocco o i suoi valori di riempimento, dopo essere stati controllati, erano errati.
- **record_overflow:** è stato ricevuto un record TLS con un payload (testo cifrato) di lunghezza superiore a $2^{14} + 2048$ o il testo cifrato è stato decrittografato a una lunghezza maggiore di $2^{14} + 1024$ byte.
- **unknown_ca:** è stata ricevuta una catena di certificati valida ma un certificato non è stato accettato poiché l'autorità di certificazione non è stata trovata o non coincideva con alcune delle autorità di certificazione note e fidate.
- **access_denied:** è stato ricevuto un certificato valido ma quando è stato richiesto o il controllo degli accessi, il mittente ha deciso di non procedere con la negoziazione.
- **decode_error:** non si è potuto decodificare un messaggio poiché un campo fuoriusciva dall'intervallo specificato o la lunghezza del messaggio era errata.
- **export_restriction:** è stata rilevata una negoziazione non coerente con le restrizioni di esportazione relative alla lunghezza delle chiavi.
- **protocol_version:** la versione di protocollo che il client ha tentato di negoziare viene riconosciuta ma non è supportata.
- **insufficient_security:** viene restituito al posto di handshake_failure quando una negoziazione non è riuscita poiché il server richiede tecniche di cifratura più sicure di quelle supportate dal client.
- **internal_error:** un errore interno indipendente dal nodo peer, oppure dal protocollo, rende impossibile continuare.

Ecco i nuovi allarmi rimanenti.

- **decrypt_error:** un'operazione crittografica di handshake è fallita, per esempio non è stata in grado di verificare una firma, decrittografare uno scambio di chiavi o convalidare un messaggio terminato.
- **user_canceled:** questo handshake è stato annullato per motivi differenti da un errore di protocollo.
- **no_renegotiation:** inviato da un client in risposta a una richiesta hello o dal server in risposta a un hello del client dopo l'handshake iniziale. Entrambi questi messaggi produrrebbero normalmente una rinegoziazione ma questo allarme indica che il mittente non è stato in grado di rinegoziare. Questo messaggio è sempre un warning.

Suite crittografiche

Vi sono alcune differenze fra le suite crittografiche disponibili sotto SSLv3 e TLS.

- **Scambio delle chiavi:** TLS supporta tutte le tecniche di scambio delle chiavi di SSLv3 ad eccezione di Fortezza.
- **Algoritmi di crittografia simmetrici:** TLS include tutti gli algoritmi di crittografia simmetrica presenti in SSLv3, ad eccezione di Fortezza.

Tipi di certificati del client

TLS definisce i seguenti tipi di certificati che possono essere richiesti in un messaggio certificate_request: rsa_sign, dss_sign, rsa_fixed_dh e dss_fixed_dh. Questi sono definiti anche in SSLv3. Inoltre, SSLv3 include rsa_ephemeral_dh, dss_ephemeral_dh e fortezza_kea. Ephemeral Diffie-Hellman richiede la firma dei parametri Diffie-Hellman con RSA o DSS; per TLS, vengono utilizzati i tipi rsa_sign e dss_sign; non è necessario un tipo distinto per firmare i parametri Diffie-Hellman. TLS non include lo schema Fortezza.

Messaggi certificate_verify e finished

Nel messaggio TLS certificate_verify, i codici hash MD5 e SHA-1 vengono calcolati solo su handshake_messages. In SSLv3, i calcoli hash includono anche il valore segreto master e i bit di riempimento. Si è ritenuto che questi campi aggiuntivi non migliorassero la sicurezza. Come in SSLv3, il messaggio finished di TLS è un codice hash basato sul valore master_secret condiviso, i messaggi precedenti di handshake e un'etichetta che identifica il client o il server. Il calcolo è però leggermente differente. In TLS si ha:

PRF(master_secret, finished_label, MD5(handshake_messages) ||
SHA-1(handshake_messages))

dove finished_label è la stringa "client finished" per il client e "server finished" per il server.

Calcoli crittografici

Il valore pre_master_secret di TLS viene calcolato nello stesso modo di SSLv3. Come in SSLv3, il valore master_secret in TLS viene calcolato come funzione hash del valore pre_master_secret e dei due numeri casuali di hello. La forma del calcolo TLS è differente da quella di SSLv3 ed è definita nel seguente modo:

master_secret =
PRF(pre_master_secret, "master secret", ClientHello.random || ServerHello.random)

L'algoritmo viene eseguito finché non vengono prodotti 48 byte di output pseudocasuale. Il calcolo del key-block (chiavi segrete MAC, chiavi di crittografia della sessione e vettori di inizializzazione) è definito come segue:

key_block =
PRF(master_secret, "key expansion",
SecurityParameters.server_random || SecurityParameters.client_random)

fino a produrre un output di lunghezza sufficiente. Come in SSLv3, key_block è funzione di master_secret e dei numeri casuali del client e del server ma in TLS l'algoritmo impiegato è differente.

Riempimento

In SSL, i byte di riempimento aggiunti prima della crittografia dei dati utente sono la quantità minima necessaria per fare in modo che la dimensione dei dati da crittografare sia multipla della lunghezza del blocco dell'algoritmo di crittografia. In TLS i byte di riempimento possono essere di qualsiasi dimensione che produca un totale multiplo della lunghezza del blocco dell'algoritmo di crittografia, fino a un massimo di 255 byte. Per esempio, se il testo in chiaro (o il testo compresso nel caso venga impiegata la compressione) più il codice MAC, più il byte padding.length fosse di 79 byte, allora i byte di riempimento in byte potrebbero essere 1, 9, 17 e così via fino a 249. Questa variabilità viene utilizzata per vanificare gli attacchi basati sull'analisi della lunghezza dei messaggi scambiati.

17.3 SET (Secure Electronic Transaction)

SET è una specifica di crittografia aperta e di sicurezza progettata per proteggere le transazioni Internet effettuate tramite carta di credito. La versione corrente, SETv1, è emersa da una gara per standard di sicurezza lanciata da MasterCard e Visa nel febbraio del 1996. Lo sviluppo delle specifiche iniziali ha coinvolto varie società fra cui IBM, Microsoft, Netscape, RSA, Terisa e Verisign. A partire dal 1996, vi sono stati numerosi test di questo meccanismo e nel 1998 si sono resi disponibili i primi prodotti compatibili SET.

SET non è un sistema di pagamento ma piuttosto un insieme di protocolli di sicurezza e di formati che consentono agli utenti di impiegare in modo sicuro l'infrastruttura di pagamento tramite carta di credito esistente su una rete aperta come Internet. In pratica SET fornisce tre servizi.

- Fornisce un canale di comunicazione sicuro fra le parti coinvolte nella transazione.
- Garantisce la sicurezza tramite l'utilizzo di certificati digitali X.509v3.
- Garantisce la privacy poiché le informazioni sono disponibili alle parti coinvolte nella transazione solo quando e dove è necessario.

SET è una specifica complessa definita in tre volumi pubblicati nel maggio del 1997:

- **Volume 1:** Business Description (80 pagine)

- **Volume 2:** Programmer's Guide (629 pagine)
- **Volume 3:** Formal Protocol Definition (262 pagine)

Si tratta di un totale di 971 pagine di specifiche. Al contrario le specifiche di SSLv3 occupavano solo 63 pagine e le specifiche di TLS 80 pagine. In questa parte del capitolo verrà fornita una panoramica di queste complesse specifiche.

Panoramica su SET

Un buon modo per iniziare la discussione su SET consiste nell'analizzare i suoi requisiti commerciali, le sue funzionalità principali e gli attori coinvolti in una transazione.

Requisiti

Il Volume 1 delle specifiche di SET elenca i seguenti requisiti commerciali per l'elaborazione sicura dei pagamenti con carta di credito via Internet o tramite altre reti.

- **Segretezza del pagamento e delle informazioni sugli ordini:** è necessario garantire al possessore della carta di credito che queste informazioni siano sicure e accessibili solo al destinatario. La segretezza riduce anche il rischio di frodi da entrambe le parti della transazione o da parte di terzi. SET utilizza la crittografia per garantire la segretezza.
- **Garanzia dell'integrità di tutti i dati trasmessi:** occorre garantire che durante la trasmissione dei messaggi SET non intervenga alcuna modifica nei contenuti. Per garantire l'integrità vengono utilizzate le firme digitali.
- **Garantire che il possessore della carta di credito sia l'utente legittimo di un conto corrispondente:** un meccanismo che collega il possessore della carta di credito a un determinato numero di conto riduce le possibilità di frodi e il costo globale di elaborazione dei pagamenti. Per verificare che il titolare della carta di credito sia un utente legittimo di un conto vengono utilizzate firme e certificati digitali.
- **Garantire che un venditore possa accettare le transazioni con carte di credito tramite la sua relazione con un'istituzione finanziaria:** questo requisito è complementare al precedente. Il titolare deve essere in grado di identificare i venditori con cui condurre transazioni sicure. Anche in questo caso vengono utilizzate firme e certificati digitali.
- **Garantire l'uso delle migliori pratiche di sicurezza e tecniche di progettazione dei sistemi per proteggere tutte le parti legittime coinvolte nelle transazioni di commercio elettronico:** SET è una specifica ben collaudata basata su algoritmi e protocolli crittografici di elevata sicurezza.
- **Creare un protocollo che non dipende dai meccanismi di sicurezza di livello trasporto e non ne impedisca l'uso:** SET può operare con sicurezza su uno stack TCP/IP "grezzo". Tuttavia SET non interferisce con l'uso di altri meccanismi di sicurezza come IPSec e SSL/TLS.
- **Facilitare e incoraggiare l'interoperabilità fra vari software e provider di rete:** i protocolli e i formati SET sono indipendenti dalla piattaforma hardware, dal sistema operativo e dal software Web.

Le funzionalità principali di SET

Per soddisfare ai requisiti appena elencati, SET include le seguenti funzionalità.

- **Segretezza delle informazioni:** le informazioni sul conto del titolare e sul pagamento sono sicure lungo tutto il transito attraverso la rete. Una caratteristica importante di SET è il fatto che impedisce al destinatario di conoscere il numero di carta di credito del titolare: tale codice verrà fornito solo alla banca. Per garantire la segretezza viene utilizzata la crittografia convenzionale DES.
- **Integrità dei dati:** le informazioni sul pagamento inviate dal titolare al venditore includono informazioni sull'ordine, i dati personali e le istruzioni di pagamento. SET garantisce che questi contenuti non vengano alterati durante il transito. L'integrità dei messaggi è garantita da firme digitali RSA che utilizzano codici hash SHA-1. Determinati messaggi sono anche protetti da HMAC con SHA-1.
- **Autenticazione del conto del titolare:** SET consente al venditore di verificare che un titolare sia un utente legittimo di una carta valida. Per questo scopo SET utilizza certificati digitali con X.509v3 con firme RSA.
- **Autenticazione del venditore:** SET consente al titolare di verificare che un venditore abbia una relazione con un'istituzione finanziaria che gli consenta di accettare il pagamento tramite carta di credito. Per questo scopo, SET utilizza certificati digitali X.509v3 con firme RSA.

Si noti che a differenza di IPSec e SSL/TLS, SET fornisce una sola scelta per ciascun algoritmo di crittografia. Questo ha senso poiché SET è un'unica applicazione con un unico insieme di requisiti mentre IPSec e SSL/TLS hanno lo scopo di supportare varie applicazioni.

Gli attori di SET

La Figura 17.8 indica gli attori in un sistema SET.

- **Titolare (della carta di credito o di pagamento):** in un ambiente elettronico, i consumatori e gli acquirenti aziendali interagiscono con i venditori tramite personal computer via Internet. Un titolare è un possessore legittimo di una carta di pagamento (per esempio MasterCard o Visa).
- **Venditore:** persona o azienda dotata di beni o servizi da vendere al titolare della carta. Normalmente questi beni e servizi vengono offerti tramite un sito Web o tramite posta elettronica. Un venditore che accetta carte di credito deve avere una relazione con un acquisitore.
- **Emettitore:** istituzione finanziaria, per esempio una banca, che assegna la carta di pagamento al titolare. Alla fine è l'emettitore il responsabile del pagamento o del debito del titolare.
- **Acquisitore:** istituzione finanziaria che crea un conto per il venditore ed elabora le autorizzazioni delle carte di pagamento e i pagamenti stessi. I venditori possono normalmente accettare più carte di credito ma senza dover gestire direttamente i vari emettitori. L'acquisitore fornisce la garanzia al venditore che una determinata carta di credito è attiva e che l'acquisto in corso non supera il limite di credito. L'acquisitore effettua anche il trasferimento elettronico del pagamento sul conto del venditore. Successivamente l'acquisitore viene rimborsato dall'emettitore tramite una qualche forma di rete di pagamento per il trasferimento elettronico di denaro.
- **Gateway di pagamento:** funzione svolta dall'acquisitore o da chi elabora i messaggi di pagamento per il venditore. Il gateway di pagamento si interfaccia fra SET e le reti di pagamento tramite carta bancaria esistenti, per svolgere le funzioni di autorizzazione e pagamento. Il venditore scambia messaggi SET con il gateway di pagamento via Internet mentre il gateway di pagamento ha una connessione diretta o di rete con il sistema di elaborazione finanziaria dell'acquisitore.
- **Autorità di certificazione (CA):** entità autorizzata a emettere certificati di chiave pubblica X.509v3 per i titolari di carte, i venditori e i gateway di pagamento. Il successo di SET dipenderà dall'esistenza di un'infrastruttura di autorità di certificazione disponibile per questo scopo. Come si è detto nei capitoli precedenti, viene utilizzata una gerarchia di autorità certificazione in modo che i partecipanti non debbano essere certificati direttamente dall'unica autorità principale.

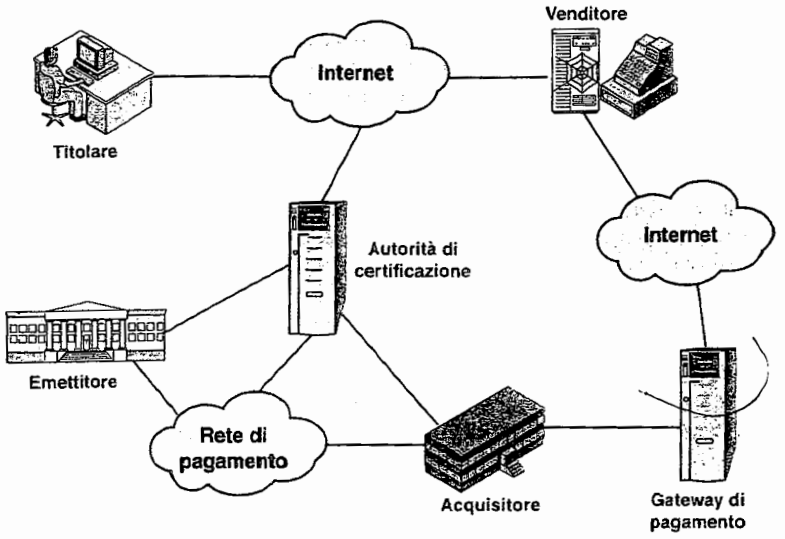


Figura 17.8 I componenti coinvolti nelle transazioni sicure.

Ora si discuterà brevemente la sequenza di eventi necessari per effettuare una transazione. Quindi si parlerà di alcuni dettagli di crittografia.

1. **Il cliente apre un conto.** Il cliente ottiene una carta di credito (per esempio MasterCard o Visa) tramite una banca che supporta pagamenti elettronici e SET.
2. **Il cliente riceve un certificato.** Dopo che ne è stata verificata l'identità, il cliente riceve un certificato digitale X.509v3 firmato dalla banca. Il certificato verifica la chiave

pubblica RSA del cliente e la sua data di scadenza. Inoltre stabilisce una relazione, garantita dalla banca, fra la coppia di chiavi del cliente e la sua carta di credito.

3. **I venditori hanno i propri certificati.** Un venditore che accetta un determinato tipo di carta di credito deve essere in possesso di due certificati per due chiavi pubbliche in suo possesso: uno per firmare i messaggi e uno per lo scambio delle chiavi. Il venditore ha bisogno anche di una copia del certificato di chiave pubblica del gateway di pagamento.
4. **Il cliente effettua un ordine.** Si tratta di un'operazione che il cliente svolge accedendo al sito Web del venditore per scegliere degli articoli e determinarne il prezzo. A questo punto il cliente invia al venditore l'elenco degli articoli da acquistare; il venditore restituisce un modulo di ordine contenente l'elenco degli articoli, il loro prezzo, la somma totale e il numero d'ordine.
5. **Il venditore viene verificato.** Oltre al modulo d'ordine, il venditore invia una copia del proprio certificato in modo che il cliente possa verificare che sta trattando con un punto vendita autorizzato.
6. **L'ordine e il pagamento vengono inviati.** Il cliente invia al venditore l'ordine e le informazioni di pagamento insieme al proprio certificato. L'ordine conferma l'acquisto degli articoli contenuti nel modulo d'ordine. Il pagamento contiene i dettagli della carta di credito. Le informazioni di pagamento vengono crittografate in modo che non possano essere lette dal venditore. Il certificato del cliente consente al venditore di verificare il cliente.
7. **Il venditore richiede l'autorizzazione del pagamento.** Il venditore invia le informazioni sul pagamento al gateway di pagamento chiedendo di verificare che il credito disponibile per il cliente sia sufficiente per questo acquisto.
8. **Il venditore conferma l'ordine.** Il venditore invia al cliente la conferma dell'ordine.
9. **Il venditore fornisce i beni o il servizio.** Il venditore invia i beni o fornisce il servizio al cliente.
10. **Il venditore richiede il pagamento.** Questa richiesta viene inviata al gateway di pagamento che gestisce tutta l'elaborazione del pagamento.

Doppia firma

Prima di approfondire la descrizione del protocollo SET occorre discutere un'innovazione interessante introdotta in SET: la doppia firma. Lo scopo della doppia firma è di collegare due messaggi destinati a due diversi destinatari. In questo caso, il cliente vuole inviare al venditore le informazioni sull'ordine (OI) e alla banca e le informazioni sul pagamento (PI). Il venditore non deve conoscere il numero di carta di credito del cliente e la banca non deve conoscere i dettagli dell'ordine del cliente. Il cliente ha così un livello di protezione superiore in termini di privacy mantenendo distinte le due informazioni. Tuttavia i due elementi devono essere collegati in modo da poter essere utilizzati per risolvere eventuali dispute. Il collegamento è necessario per fare in modo che il cliente possa dimostrare che questo pagamento era relativo a questo ordine e non a un altro bene o servizio.

Per comprendere la necessità di questo collegamento, si supponga che i clienti inviino al venditore due messaggi: un messaggio con l'ordine (OI) firmato e un messaggio di pagamento (PI) firmato e che il venditore inoltri le informazioni di pagamento PI alla

banca. Se il venditore cattura un altro ordine da questo cliente, potrà sostenere che le precedenti informazioni di pagamento sono relative a quest'ultimo ordine e non a quello originario. Il collegamento fra queste due informazioni evita questa possibilità.

La Figura 17.9 mostra l'uso della doppia firma per rispondere ai requisiti del paragrafo precedente. Il cliente prende il codice hash (utilizzando SHA-1) delle informazioni di pagamento e il codice hash delle informazioni sull'ordine. Questi due valori hash vengono poi concatenati e viene eseguita un'ulteriore operazione hash sul risultato. Infine il cliente esegue la crittografia del codice hash finale con la propria chiave di firma privata creando una doppia firma. L'operazione può essere riassunta nel seguente modo:

$$DS = E(PR_C, [H(H(PI) \| H(OI)])$$

dove PR_c è la chiave di firma privata del cliente. Ora si supponga che il venditore sia in possesso della doppia firma (DS), delle informazioni d'ordine OI e del codice message digest per il PI (PIMD). Il venditore ha anche la chiave pubblica del cliente, tratta dal suo certificato. Quindi il venditore può calcolare le due quantità seguenti:

$$H(PIMD \| H[OI]) ; D(PU_C, DS)$$

dove PU_c è la chiave di firma pubblica del cliente. Se queste due quantità sono uguali, allora il venditore ha verificato la firma. Analogamente, se la banca è in possesso della doppia firma DS, dell'informazione di pagamento PI, del codice message digest per OI (OIMD) e della chiave pubblica del cliente, allora potrà calcolare il seguente valore:

$$H(H(OI) \| OIMD); D(PU_C, DS)$$

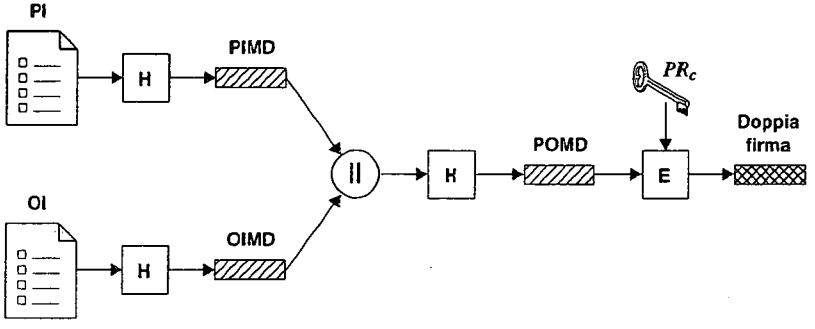


Figura 17.9 Costruzione della doppia firma.

Anche in questo caso, se queste due quantità sono uguali, allora la banca ha verificato la firma. In breve ecco cosa accade.

1. Il venditore ha ricevuto le informazioni d'ordine OI e ha verificato la firma.
2. La banca ha ricevuto le informazioni di pagamento e ha verificato la firma.
3. Il cliente ha collegato le informazioni di pagamento OI con le informazioni sull'ordine PI e può dimostrare il loro collegamento.

Per esempio, si supponga che il venditore voglia sostituire un'altra informazione d'ordine in questa transazione, a proprio vantaggio: dovrà pertanto trovare un'altra informazione d'ordine OI il cui valore hash corrisponda al valore OIMD esistente. Con SHA-1 l'operazione è sostanzialmente impossibile. Pertanto il venditore non potrà collegare un'altra informazione d'ordine OI con questa informazione di pagamento PI.

Elaborazione dei pagamenti

La Tabella 17.3 elenca i tipi di transazioni supportati da SET. Di seguito si parlerà in dettaglio delle seguenti transazioni:

- richiesta di acquisto;
- autorizzazione di pagamento;
- cattura del pagamento.

Richiesta di acquisto

Prima dell'inizio dello scambio di richiesta di acquisto, il titolare ha completato la navigazione, la scelta e l'ordine. La fine di questa fase preliminare si verifica quando il venditore invia al cliente un modulo d'ordine completato. Tutto ciò si svolge senza impiegare SET.

Lo scambio della richiesta di acquisto è costituito da quattro messaggi: Initiate Request, Initiate Response, Purchase Request e Purchase Response.

Per inviare i messaggi SET al venditore, il titolare deve avere una copia dei certificati del venditore e del gateway di pagamento. Il cliente richiede i certificati nel messaggio **Initiate Request** inviato al venditore. Questo messaggio include il tipo di carta di credito utilizzato dal cliente. Il messaggio include anche un identificatore ID assegnato dal cliente alla coppia richiesta/risposta e un valore nonce utilizzato per impedire gli attacchi a replay.

Il venditore genera una risposta e la firma con la propria chiave di firma privata. La risposta include il valore nonce inviato dal cliente, un altro valore nonce inviato al cliente che questi deve restituire nel messaggio successivo e un identificatore per questa transazione di acquisto. Oltre alla risposta firmata, il messaggio **Initiate Response** include il certificato di firma del venditore e il certificato di scambio delle chiavi del gateway di pagamento.

Il titolare verifica i certificati del venditore e del gateway tramite le rispettive firme certificate dall'autorità di certificazione e quindi crea le informazioni d'ordine e le informazioni di pagamento. L'identificatore di transazione assegnato dal venditore viene inserito sia nelle informazioni d'ordine che nelle informazioni di pagamento. Le informazioni d'ordine non contengono dati espliciti sull'ordine (per esempio il numero e il prezzo degli articoli). Piuttosto contiene un riferimento generato nello scambio fra il venditore e il cliente durante la fase di acquisto prima del primo messaggio SET. Poi il titolare della carta di

Tabella 17.3 Tipi di transazione SET.

Registrazione del titolare	I titolari di carte di credito devono registrarsi con un'autorità di certificazione prima di poter inviare i propri messaggi SET ai venditori.
Registrazione del venditore	I venditori devono registrarsi con un'autorità di certificazione prima di poter scambiare messaggi SET con i clienti e i gateway di pagamento.
Richiesta di acquisto	Messaggio inviato dal cliente al venditore contenente le informazioni d'ordine per il venditore e le informazioni di pagamento per la banca.
Autorizzazione del pagamento	Scambio fra il venditore e il gateway di pagamento per autorizzare una data somma per un acquisto con una determinata carta di credito.
Cattura del pagamento	Consente al venditore di richiedere il pagamento al gateway di pagamento.
Interrogazione sullo stato del certificato	Se l'autorità di certificazione non è in grado di completare l'elaborazione di una richiesta di certificato con rapidità, invia una risposta al titolare o al venditore indicando che il richiedente dovrà riverificare in seguito. Il titolare o il venditore invia un messaggio di interrogazione per determinare lo stato della richiesta del certificato o per ricevere il certificato nel caso in cui la richiesta sia stata approvata.
Interrogazione di acquisto	Consente al titolare di verificare lo stato di elaborazione di un ordine dopo aver ricevuto una risposta sull'acquisto. Si noti che questo messaggio non include le informazioni quali lo stato dei beni ma indica lo stato dell'autorizzazione, della cattura e dell'elaborazione del credito.
Revoca dell'autorizzazione	Consente a un venditore di correggere le precedenti richieste di autorizzazione. Se l'ordine non viene completato, il venditore revoca l'intera autorizzazione. Se una parte dell'ordine non viene completata (per esempio quando i beni vengono resi), il venditore revoca una parte della somma relativa all'autorizzazione.
Correzione della cattura	Consente a un venditore di correggere gli errori di cattura come per esempio le somme sulle transazioni che sono state introdotte in modo errato da un addetto.
Credito	Consente a un venditore di accreditare il conto del titolare, per esempio quando i beni vengono resi o quando si sono danneggiati durante il trasporto. Si noti che il messaggio SET *Credit* viene sempre attivato dal venditore e mai dal titolare. Tutte le comunicazioni fra il titolare o il venditore che richiedono l'elaborazione del credito si verificano all'esterno di SET.
Revoca del credito	Consente a un venditore di correggere una richiesta di credito precedentemente emessa.
Richiesta del certificato del gateway di pagamento	Consente a un venditore di interrogare il gateway di pagamento o ricevere una copia dei certificati di scambio della chiave e di firma in possesso del gateway.
Amministrazione	Consente a un venditore di comunicare informazioni al gateway di pagamento relative alle proprie vendite.
Messaggio d'errore	Indica che qualcuno ha rifiutato un messaggio poiché non è valido il formato o non ha passato i test di verifica dei contenuti.

credito prepara il messaggio **Purchase Request** (Figura 17.10). A questo fine genera una chiave di crittografia simmetrica monouso, K_s. Il messaggio include i seguenti elementi.

1. **Informazioni d'acquisto.** Queste informazioni verranno inoltrate al gateway di pagamento dal venditore e sono costituite dai seguenti elementi.
   - Le informazioni di pagamento PI.
   - La doppia firma, calcolata sulle informazioni di pagamento PI e sulle informazioni dell'ordine OI, effettuata con la chiave di firma privata del cliente.
   - Il codice OI Message Digest (OIMD)

   Il codice OIMD è necessario, come si è detto in precedenza, affinché il gateway di pagamento possa verificare la doppia firma. Tutti questi elementi sono crittografati con K_s. L'ultimo elemento è il seguente.
   - La busta digitale. Questa viene costituita crittografando K_s con la chiave di scambio pubblica del gateway di pagamento. Viene chiamata busta digitale poiché deve essere aperta (decrittografata) prima che possano essere letti gli altri articoli elencati in precedenza.

   Il valore di K_s non viene rivelato al venditore. Pertanto il venditore non potrà leggere alcuna informazione relativa al pagamento.
2. **Informazioni relative all'ordine.** Queste informazioni sono necessarie al venditore e sono costituite dai seguenti elementi.
   - Le informazioni sull'ordine OI.
   - La doppia firma, calcolata sulle informazioni di pagamento e sulle informazioni relative all'ordine, firmate con la chiave di firma privata del cliente.
   - Il codice PI Message Digest (PIMD).

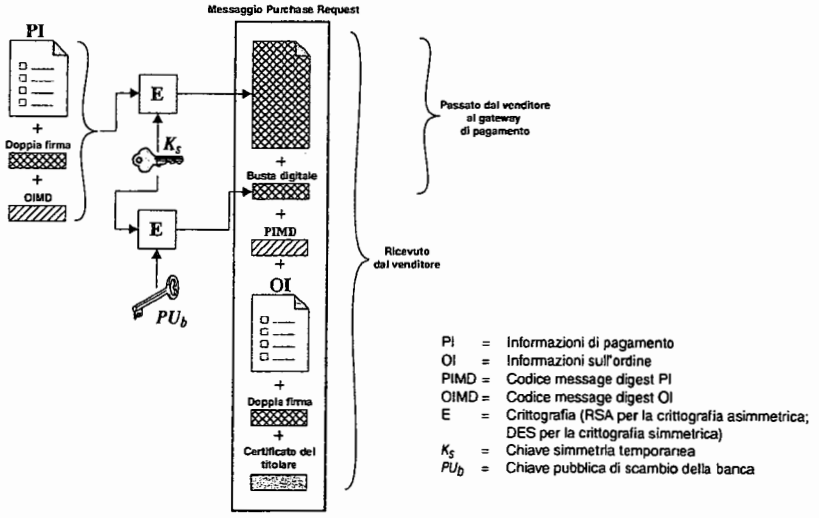


Figura 17.10 Il titolare invia la richiesta di acquisto (messaggio Purchase Request).

Il codice PIMD è necessario perché il venditore possa verificare la doppia firma. Si noti che le informazioni sull'ordine vengono inviate in chiaro.

3. **Certificato del titolare.** È costituito dalla chiave di firma pubblica del titolare. È richiesta dal venditore e dai gateway di pagamento.

Quando il venditore riceve il messaggio Purchase Request, svolge le seguenti operazioni (vedere la Figura 17.11).

1. Verifica i certificati del titolare tramite le firme dell'autorità di certificazione.
2. Verifica la doppia firma utilizzando la chiave di firma pubblica del cliente. Questo garantisce che l'ordine non sia stato falsificato durante il transito e che sia stato firmato utilizzando la chiave di firma privata del titolare.
3. Elabora l'ordine e inoltra le informazioni di pagamento al gateway di pagamento per l'autorizzazione (se ne parlerà più avanti).
4. Invia un messaggio Purchase Response al titolare.

Il messaggio **Purchase Response** include un blocco di risposta che conferma l'ordine e fa riferimento al numero di transazione corrispondente. Questo blocco è firmato dal venditore utilizzando la sua chiave di firma privata. Il blocco e la sua firma vengono inviati al cliente insieme al certificato di firma del venditore.

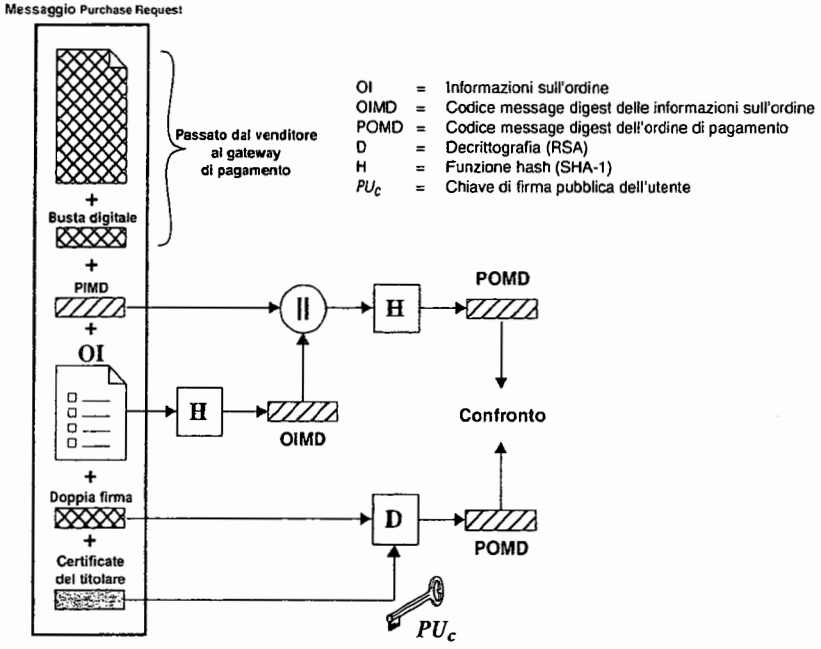


Figura 17.11 Il venditore verifica la richiesta di acquisto del cliente.

Quando il software del titolare riceve il messaggio Purchase Response, verifica il certificato del venditore e quindi verifica la firma sul blocco di risposta. Infine svolge qualche azione basata sulla risposta come per esempio la visualizzazione di un messaggio per l'utente o l'aggiornamento di un database con lo stato dell'ordine.

Autorizzazione del pagamento

Durante l'elaborazione di un ordine per un titolare, il venditore autorizza la transazione con il gateway di pagamento. L'autorizzazione di pagamento garantisce che la transazione sia stata approvata dall'ente emettitore e che il venditore riceverà il pagamento: il venditore può pertanto offrire al cliente il servizio o i beni. Lo scambio di autorizzazione del pagamento è costituito da due messaggi: Authorization Request e Authorization response.

Il venditore invia un messaggio **Authorization Request** al gateway di pagamento costituito dai seguenti elementi.

1. **Informazioni relative all'acquisto.** Queste informazioni sono state ottenute dal cliente e sono costituite dai seguenti elementi:
   - Le informazioni di pagamento PI.
   - La doppia firma, calcolata sulle informazioni di pagamento PI e le informazioni dell'ordine OI, firmate con la chiave di firma privata del cliente.
   - Il codice OI Message Digest (OIMD)
   - La busta digitale.
2. **Informazioni di autorizzazione.** Queste informazioni vengono generate dal venditore e sono costituite dai seguenti elementi.
   - Un blocco di autorizzazione che include l'identificatore della transazione firmato con la chiave di firma privata del venditore e crittografato con una chiave simmetrica monouso generata dal venditore.
   - Una busta digitale costituita crittografando la chiave monouso con la chiave di scambio pubblica del gateway di pagamento.
3. **Certificati.** Il venditore include il certificato della chiave di firma del titolare (utilizzato per verificare la doppia firma), il certificato della chiave di firma del venditore (utilizzato per verificare la firma del venditore) e il certificato per lo scambio di chiavi del venditore (necessario nella risposta del gateway di pagamento).

Il gateway di pagamento svolge le seguenti operazioni.

1. Verifica tutti i certificati.
2. Esegue la decrittografia della busta digitale del blocco di autorizzazione per ottenere la chiave simmetrica e poi esegue la decrittografia del blocco di autorizzazione.
3. Verifica la firma del venditore sul blocco di autorizzazione.
4. Esegue la decrittografia della busta digitale del blocco di pagamento per ottenere la chiave simmetrica e quindi decrittografare il blocco di pagamento.
5. Verifica la doppia firma sul blocco di pagamento.
6. Verifica che l'identificatore della transazione ricevuto dal venditore corrisponda a quello delle informazioni di pagamento ricevute (indirettamente) dal cliente.
7. Richiede e riceve un'autorizzazione dall'emettitore.

Dopo aver ottenuto l'autorizzazione dell'emettitore, il gateway di pagamento restituisce al venditore un messaggio **Authorization Response** che include i seguenti elementi.

1. **Informazioni relative all'autorizzazione.** Include un blocco di autorizzazione firmato con la chiave di firma privata del gateway e crittografato con una chiave simmetrica monouso generata dal gateway. Inoltre include una busta digitale che contiene la chiave monouso crittografata con la chiave di scambio pubblico del venditore.
2. **Informazioni sul token di cattura.** Queste informazioni verranno utilizzate per effettuare il pagamento. Questo blocco ha lo stesso formato del precedente, ovvero un token firmato e crittografato insieme a una busta digitale. Questo token non viene elaborato dal venditore ma viene restituito così com'è con una richiesta di pagamento.
3. **Certificato.** Il certificato della chiave di firma del gateway.

Con l'autorizzazione dal gateway, il venditore può fornire al cliente i beni o i servizi.

Cattura del pagamento

Per ottenere il pagamento, il venditore effettua con il gateway di pagamento una transazione di cattura del pagamento costituita da una richiesta di cattura e da una risposta di cattura. Per il messaggio **Capture Request**, il venditore genera, firma e crittografa un blocco di richiesta della cattura che include la somma del pagamento e l'identificatore della transazione. Il messaggio include anche il token di cattura crittografato ricevuto in precedenza (in Authorization Response) per questa transazione e i certificati della chiave di firma e la chiave di scambio del venditore.

Quando il gateway di pagamento riceve il messaggio di richiesta della cattura, esegue la decrittografia e verifica del blocco di richiesta della cattura ed esegue la decrittografia e la verifica del blocco del token di cattura. Quindi controlla la coerenza fra la richiesta di cattura e il token di cattura. Poi crea una richiesta di liquidazione che viene inviata all'emettitore tramite la rete di pagamento privata. Questa richiesta provoca il trasferimento dei fondi sul conto del venditore.

Ora il gateway notifica al venditore l'effettuazione del pagamento nel messaggio **Capture Response**. Il messaggio include un blocco di risposta della cattura che il gateway firma e codifica con la crittografia. Il messaggio include anche il certificato della chiave di firma del gateway. Il software del venditore memorizza la risposta di cattura da utilizzare per documentare il pagamento ricevuto dall'acquirente.

17.4 Letture e siti Web consigliati

[RESC01] è una trattazione dettagliata di SSL e TLS.

La panoramica migliore di SET si trova nel Volume 1 delle specifiche, disponibile presso il sito Web SET di MasterCad. Un'altra eccellente panoramica è [MACG97]. Un'altra buona fonte è [DREW99].

DREW99 G. Drew. *Using SET for Secure Electronic Commerce*. Upper Saddle River, NJ: Prentice Hall, 1999.

MACG97 R, Macgregor, C. Ezvan, L. Liguori e J. Han. *Secure Electronic Transactions: Credit Card Payment on the Web in Theory and Practice*. IBM RedBook SG24-4978-00, 1997. Disponibile presso il sito `www.redbooks.ibm.com`.

RESC01 E. Rescorla. *SSL and TLS: Designing and Building Secure Systems*. Reading, MA: Addison-Wesley, 2001.

Siti Web consigliati

- **SSL Page di Netscape:** contiene le specifiche di SSL.
- **Transport Layer Security Charter:** gli ultimi documenti RFC e le bozze Internet su TLS.
- **OpenSSL Project:** progetto relativo allo sviluppo di software SSL e TLS open-source. Il sito contiene documenti e collegamenti ad altri siti web.

17.5 Termini chiave, domande di ripasso e problemi

Termini chiave

Acquirente
Autorità di certificazione (CA)
Doppia firma
Emettitore
Gateway di pagamento
SET (Secure Electronic Transaction)
SSL (Secure Socket Layer)
Titolare
TLS (Transport Layer Security)
Venditore

Domande di ripasso

17.1 Quali sono i vantaggi di ciascuno dei tre approcci rappresentati nella Figura 17.1?
17.2 Quali protocolli comprende SSL?
17.3 Qual è la differenza fra una connessione e una sessione SSL?
17.4 Elencare e definire brevemente i parametri che definiscono lo stato di una sessione SSL.
17.5 Elencare e definire brevemente i parametri che definiscono una connessione di una sessione SSL.
17.6 Quali servizi fornisce il protocollo SSL Record?
17.7 Quali passi comporta il protocollo SSL Record?
17.8 Elencare e definire brevemente le categorie principali degli attori SET.
17.9 Che cos'è una doppia firma è qual è il suo scopo?

Problemi

17.1 Perché SSL e TLS prevedono un protocollo Change Cipher Spec distinto invece di includere un messaggio change_cipher_spec nel protocollo Handshake?
17.2 Considerare le seguenti minacce alla sicurezza Web e descrivere il modo in cui vengono sventate da una determinata funzionalità di SSL.
   A. Attacco ad analisi crittografica a forza bruta: ricerca esaustiva dello spazio delle chiavi di un algoritmo di crittografia convenzionale.
   B. Attacco a dizionario con testo in chiaro noto: molti messaggi sono costituiti da testo in chiaro prevedibile come per esempio il comando HTTP GET. Un hacker può costruire un dizionario contenente ogni possibile crittografia del messaggio di testo in chiaro noto. Quando viene intercettato un messaggio crittografato, l'hacker prende la porzione contenente il testo in chiaro noto e ricerca il testo cifrato nel dizionario. Il testo cifrato corrisponderà a una voce che è stata crittografata con la stessa chiave segreta. Se vi sono più corrispondenze, ciascuna di esse può essere tentata con l'intero testo cifrato per determinare il valore corretto. Questo attacco è particolarmente efficace contro chiavi di piccole dimensioni (per esempio chiavi di 40 bit).
   C. Attacco a replay: ripetizione di messaggi SSL precedenti.
   D. Attacco Man-in-the-Middle: un hacker si interpone durante lo scambio delle chiavi fungendo da client per il server e da server per il client.
   E. Sniffing della password: intercettazione delle password in HTTP o nel traffico di altre applicazioni.
   F. Spoofing IP: utilizza indirizzi IP falsificati per ingannare un host e indurlo ad accettare dati fasulli.
   G. Dirottamento IP: una connessione attiva e autenticata fra due host viene dirottata e l'hacker prende il posto di uno dei due host.
   H. SYN Flooding: l'hacker invia dei messaggi TCP SYN per richiedere una connessione ma non risponde al messaggio finale per completare la connessione. Il modulo TCP attaccato lascia quindi una connessione semiaperta per qualche minuto. Una raffica di messaggi SYN può bloccare il modulo TCP.
17.3 Sulla base di ciò che si è appreso in questo capitolo, è possibile in SSL che il ricevitore riordini i blocchi di record SSL che arrivano disordinati? In caso affermativo, spiegare come accade. In caso negativo, spiegarne il motivo.

Parte quarta

Sicurezza di sistema

La Parte quarta tratta i problemi di sicurezza al livello dei sistemi, fra cui le minacce rappresentate dalle intrusioni e dai virus e le relative contromisure, e l'impiego di firewall e sistemi fidati.

Capitolo 18: Intrusioni

Il Capitolo 18 esamina varie minacce rappresentate dagli hacker che sfruttano i punti deboli dei sistemi connessi in rete. Il capitolo si apre con una discussione riguardante i vari tipi di attacchi che possono essere sferrati da utenti non autorizzati e analizza vari approcci di prevenzione e rilevamento. Inoltre questo capitolo tratta l'argomento correlato della gestione delle password.

Capitolo 19: Software doloso

Il Capitolo 19 esamina le minacce al sistema rappresentate dal software doloso enfatizzando virus e worm. Il capitolo si apre con una rassegna riguardante i vari tipi di software doloso, in particolare i virus e i worm. Successivamente tratta le contromisure. Nella parte finale tratta gli attacchi DoS distribuiti.

Capitolo 20: I firewall

Il tipico approccio per la protezione dei computer locali dalle minacce provenienti dall'esterno prevede l'impiego di un firewall. Il Capitolo 20 tratta i principi di funzionamento dei firewall esaminando anche alcune tecniche di difesa specifiche. Questo capitolo descrive anche l'argomento correlato dei sistemi fidati.

Capitolo 18

Intrusioni

Concetti essenziali

- Le **intrusioni** non autorizzate nelle reti e nei sistemi costituiscono una delle minacce più gravi alla sicurezza dei computer.
- Sono stati sviluppati dei **sistemi di rilevazione delle intrusioni** capaci di segnalare tempestivamente eventuali intrusioni, in modo da poter prendere le azioni difensive opportune per prevenire o minimizzare i danni.
- La rilevazione delle intrusioni comporta l'identificazione dei comportamenti anomali o notoriamente associati alle intrusioni.
- Un elemento fondamentale nella prevenzione delle intrusioni è la gestione delle **password**, che ha l'obiettivo di impedire agli utenti non autorizzati di ottenere l'accesso alle password altrui.

Un importante problema di sicurezza dei sistemi connessi in rete riguarda le intrusioni di utenti o di software ostili o quanto meno indesiderati. Le violazioni effettuate dagli utenti possono assumere la forma di un login non autorizzato a una macchina o, nel caso di un utente autorizzato, l'acquisizione di privilegi superiori senza autorizzazione. Le violazioni del software possono consistere in virus, worm o cavalli di Troia.

Tutti questi attacchi riguardano la sicurezza della rete, poiché l'accesso al sistema può essere ottenuto tramite una rete. Tuttavia, questi attacchi non si limitano a intrusioni tramite le reti. Un utente dotato di un accesso da un terminale locale può tentare di violare un sistema senza utilizzare una rete intermedia. Un virus o un cavallo di Troia può essere introdotto in un sistema tramite un dischetto. Solo quello dei worm è un fenomeno esclusivamente di rete. Pertanto la violazione di sistemi è un'area in cui i problemi della sicurezza della rete e dei computer si sovrappongono.

Poiché l'enfasi di questo libro è sulla sicurezza delle reti, non si tenterà un'analisi approfondita degli attacchi o delle contromisure relative alla violazione diretta dei sistemi. In questa parte si presenterà però una panoramica generale di questi problemi.

Questo capitolo tratta l'argomento delle intrusioni. Innanzitutto verrà esaminata la natura dell'attacco poi si parlerà delle strategie di prevenzione e di rilevamento. Infine verrà esaminato l'argomento correlato della gestione delle password.

18.1 Intrusioni

Una delle due minacce alla sicurezza più pubblicizzate è rappresentata dalle intrusioni (l'altra è rappresentata dai virus), generalmente effettuate da personaggi chiamati hacker. In un importante studio sulle intrusioni, Anderson [ANDE80] ha identificato tre classi di intrusi o hacker.

- **Utente sotto mentite spoglie:** personaggio che non è autorizzato a utilizzare il computer ma che elude i controlli di accesso di un sistema per sfruttare l'account di un utente legittimo.
- **Utente legittimo disonesto:** un utente legittimo che accede ai dati, ai programmi o alle risorse cui non ha normalmente accesso o comunque che sfrutta scorrettamente i propri privilegi.
- **Utente clandestino:** personaggio che acquisisce il controllo di un sistema come supervisore e utilizza questo controllo per sfuggire all'auditing e ai metodi di controllo degli accessi o per sopprimere la raccolta delle informazioni di auditing.

Il primo è in genere un personaggio esterno; il secondo è in genere un interno e l'utente clandestino può essere interno o esterno.

Gli attacchi degli hacker possono variare da benigni fino a particolarmente maligni. A un estremo vi sono coloro che vogliono semplicemente esplorare le reti internet spinti dalla curiosità. All'altro estremo vi sono personaggi che tentano di leggere dati riservati, modificare senza autorizzazione tali dati o sovvertire il sistema.

La minaccia degli hacker è stata molto pubblicizzata, specialmente dopo l'incidente "Wily Hacker" del 1986-1987 documentato da Stoll [STOL88, 89]. Nel 1990 venne sferrato un importante attacco contro gli hacker illeciti, con arresti, pene, un drammatico processo, numerose condanne e la confisca di enormi quantità di dati e computer [STER92]. Molti ritennero quindi che il problema fosse finalmente sotto controllo.

In realtà le cose non stanno così. Per citare un esempio, un gruppo dei Bell Labs [BELL92, BELL93] ha documentato persistenti e frequenti attacchi al suo complesso di computer via Internet per un lungo arco di tempo e da varie fonti. Ecco gli attacchi subiti dal gruppo Bell.

- Tentativi di copiare il file delle password (se ne parlerà più avanti): più di una volta al giorno.
- Richieste RPC (RPC - Remote Procedure Call) sospette: più di una volta alla settimana.
- Tentativi di connettersi a macchine "esca" inesistenti: più di una volta ogni due settimane.

Gli hacker benigni possono anche essere tollerati, anche se consumano le risorse di sistema e possono rallentare le prestazioni per gli utenti legittimi. Tuttavia non esiste un modo per stabilire a priori se un hacker è benigno o maligno. Di conseguenza, anche per i sistemi che non contengono risorse particolarmente riservate, è opportuno cercare di controllare questo problema.

Un esempio che illustra drammaticamente la minaccia rappresentata dagli hacker si è verificato all'università A&M del Texas [SAFF93]. Nell'agosto del 1992, il centro di calcolo fu avvertito del fatto che una delle sue macchine era stata utilizzata per attaccare i computer di un sito via Internet. Il personale del centro di calcolo individuò vari hacker esterni coinvolti nell'operazione, che avevano utilizzato routine di violazione delle password su vari computer (su un totale di 12 000 macchine interconnesse). Il centro scollegò le macchine interessate, risolse i problemi di sicurezza noti e riprese le normali attività. Qualche giorno dopo, uno dei manager dei sistemi locali rilevò che l'attacco degli hacker era ripreso. Si scoprì che l'attacco era molto più sofisticato di quanto si fosse ritenuto inizialmente. Vennero trovati file contenenti centinaia di password catturate, comprese alcune dei server principali ritenuti sicuri. Addirittura una macchina locale era stata configurata come bulletin board e veniva utilizzata dagli hacker per contattarsi e per discutere tecniche e progressi.

Un'analisi di questo attacco ha rivelato che in realtà esistono due livelli di hacker. Il livello più elevato è costituito da utenti dalle sofisticate conoscenze tecniche e con una profonda conoscenza della tecnologia; il livello più basso è costituito dai "soldati semplici" che non fanno altro che utilizzare i programmi di cui entrano in possesso, senza conoscerne il funzionamento. Questi team di lavoro combinati rappresentano le armi più pericolose dell'esercito degli hacker: conoscenze sofisticate sul modo in cui effettuare le intrusioni e la volontà di dedicare innumerevoli ore a "girare la maniglia" delle porte per trovare un punto debole.

Uno dei risultati della crescente consapevolezza del problema degli hacker è stata l'attivazione di numerosi team di risposta alle emergenze relative ai computer (CERT - Computer Emergency Response Teams). Questi team raccolgono informazioni sui punti deboli dei sistemi e le distribuiscono ai responsabili. Sfortunatamente anche gli hacker hanno accesso ai report del CERT. Nell'incidente alla A&M del Texas, le analisi successive hanno dimostrato che gli hacker avevano sviluppato programmi per verificare sulle macchine attaccate la presenza di ogni punto debole individuato dal CERT. Se anche una sola delle macchine non aveva prontamente reagito alle indicazioni del CERT, rimaneva totalmente scoperta a tali attacchi.

Oltre ai programmi per la violazione delle password, gli hacker modificarono il software di login per catturare le password degli utenti che si connettevano al sistema. Questo permise loro di raccogliere un'enorme quantità di password che venivano poi rese disponibili tramite il bulletin board impiantato su una delle macchine cadute sotto l'attacco.

In questa parte del capitolo si parlerà delle tecniche utilizzate per le intrusioni. Poi si parlerà dei metodi di rilevamento delle intrusioni. Infine si parlerà degli approcci a password per la prevenzione dell'intrusioni.

Tecniche di intrusione

L'obiettivo dell'hacker è quello di acquisire l'accesso a un sistema o ampliare i propri privilegi di accesso a un sistema. In generale questo richiede che l'hacker acquisisca infor-

mazioni che dovrebbero essere protette. Nella maggior parte dei casi, si tratta delle password degli utenti. Conoscendo la password di qualche altro utente, l'hacker può connettersi al sistema ed esercitare i privilegi assegnati all'utente legittimo.

In genere un sistema deve contenere un file che associa una password a ogni utente autorizzato. Se tale file non viene protetto, sarà facilissimo accedervi per estrarre le password. Il file delle password può essere protetto in due modi.

- **Funzione monodirezionale:** il sistema memorizza solo il valore di una funzione della password dell'utente. Quando l'utente presenta la password, il sistema la trasforma e confronta il risultato con il valore memorizzato. In pratica, il sistema svolge normalmente una trasformazione monodirezionale (irreversibile) in cui la password viene utilizzata per generare una chiave per la funzione monodirezionale che produce un output di lunghezza fissa. Per questo scopo possono essere utilmente impiegate le funzioni hash.
- **Controllo degli accessi:** il sistema limita l'accesso al file delle password a pochissimi account.

Se si attiva una o entrambe queste contromisure, un potenziale hacker avrà bisogno di più tempo per carpire le password. Sulla base di un'indagine documentata e di interviste a vari hacker, [ALVA90] elenca le seguenti tecniche impiegate per la violazione delle password.

1. Tentare le password standard utilizzate dagli account di base forniti con il sistema. Molti amministratori non si preoccupano di cambiare queste impostazioni standard.
2. Provare in modo esaustivo tutte le password più brevi (quelle costituite da 1-3 caratteri).
3. Tentare le parole di un dizionario disponibile online o un elenco di password probabili. Si trovano vari esempi di queste nei servizi di informazioni per hacker.
4. Raccogliere informazioni sugli utenti: il nome completo, il nome della moglie o del marito e dei figli, i quadri presenti in ufficio, i libri di hobbistica in ufficio e così via.
5. Provare a utilizzare i numeri di telefono, il codice fiscale o il numero della stanza d'albergo.
6. Provare il numero di targa.
7. Usare un cavallo di Troia (se ne parlerà nel Paragrafo 18.2) per aggirare le restrizioni di accesso.
8. Intercettare la linea fra l'utente remoto e il sistema host.

I primi sei sono semplicemente dei metodi per indovinare una password. Se un hacker deve verificare ogni password tentando di effettuare il login, l'attacco diventa lungo, noioso e facile da rilevare. Per esempio un sistema può semplicemente rifiutare ogni ulteriore tentativo di login dopo i primi tre tentativi, costringendo l'hacker a disconnettersi e riconnettersi al sistema. In questi casi, è difficile tentare un gran numero di password. Tuttavia è poco probabile che l'hacker tenti metodi così rozzi. Per esempio, se un hacker può ottenere un accesso con un basso livello di privilegi a un file delle password crittografato, allora la strategia adottata consisterà nel catturare tale file e poi utilizzare il meccanismo di crittografia del sistema fino a scoprire una password valida che fornisca privilegi più elevati.

Gli attacchi a tentativi sono fattibili e in realtà molto efficaci quando è possibile effettuare un gran numero di tentativi in modo automatico, senza che questa operazione venga rilevata. Più avanti in questo capitolo si avrà modo di parlare dei metodi impiegabili per sventare gli attacchi di questo tipo.

Il settimo metodo di attacco, quello dei cavalli di Troia, può essere particolarmente difficile da sventare. Un esempio di programma che elude i controlli di accesso è stato citato in [ALVA90]. Un utente a bassi privilegi ha prodotto un gioco e ha invitato l'operatore del sistema a utilizzarlo nel tempo libero. Il programma in effetti era un gioco ma conteneva anche del codice che in background copiava il file delle password, che non era crittografato ma era leggibile solo per l'amministratore del sistema, in un file di servizio dell'utente. Poiché il programma del gioco era stato lanciato dall'operatore, ne ereditava i privilegi e pertanto aveva accesso al file delle password.

L'ottavo attacco elencato, quello dell'intercettazione della linea, è un problema di sicurezza fisica. Può essere sventato adottando tecniche di crittografia di canale, di cui si è parlato nel Paragrafo 7.1.

Altre tecniche di intrusione non richiedono il furto della password. Gli hacker possono ottenere l'accesso al sistema sfruttando per esempio tecniche di *buffer overflow* ("tracimazione di buffer") su un programma che viene eseguito con particolari privilegi. Anche l'acquisizione illegittima di privilegi può essere condotta con modalità analoghe.

Ora si rivolgerà l'attenzione alle due importanti contromisure fondamentali: il rilevamento e la prevenzione. Il rilevamento riguarda la conoscenza di un attacco, prima o dopo che abbia avuto successo. La prevenzione è un obiettivo di sicurezza importante e rappresenta una battaglia costante. La difficoltà deriva dal fatto che la difesa deve sventare tutti gli attacchi possibili mentre l'hacker è libero di cercare l'anello debole nella catena della difesa e attaccare solo tale punto.

18.2 Rilevamento delle intrusioni

Inevitabilmente, anche il miglior sistema di prevenzione delle intrusioni non può aver sempre successo. La seconda linea di difesa è rappresentata dal rilevamento delle intrusioni ed è stata l'argomento di molte ricerche negli ultimi anni. Questo interesse è motivato da varie considerazioni fra cui le seguenti.

1. Se un'intrusione viene rilevata con una certa rapidità, l'hacker può essere identificato ed espulso dal sistema prima che possa produrre dei danni e violare dei dati. Anche se il rilevamento non è sufficientemente rapido per impedire l'accesso all'hacker, prima viene rilevata l'intrusione, minori saranno i danni e più rapidamente sarà possibile ripristinare il sistema.
2. Un efficace sistema di rilevamento può fungere da deterrente per impedire le intrusioni.
3. Il rilevamento delle intrusioni consente di raccogliere informazioni sulle tecniche di intrusione adottate, in modo da rafforzare il sistema di prevenzione delle intrusioni.

Il rilevamento delle intrusioni si basa sulla supposizione che il comportamento dell'hacker differisca da quello dell'utente legittimo in modi ben determinati. Naturalmente non ci si può attendere una distinzione netta fra le attività di attacco di un hacker e il normale uso delle risorse da parte di un utente autorizzato: ci si deve attendere una certa sovrapposizione fra i due comportamenti.

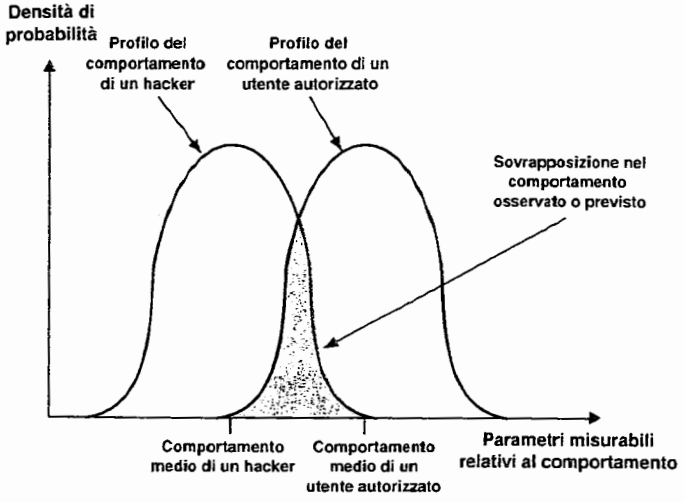


Figura 18.1 I profili di comportamento degli hacker e degli utenti autorizzati.

La Figura 18.1 suggerisce, in termini molto astratti, la natura del problema che il progettista di un sistema di rilevamento delle intrusioni deve affrontare. Sebbene il tipico comportamento di un hacker differisca dal tipico comportamento di un utente autorizzato, esiste un'area di sovrapposizione. Pertanto un'interpretazione più "allargata" dei comportamenti dell'hacker, in grado di individuare la maggior parte delle loro attività, condurrà a un certo numero di "falsi positivi" ovvero a utenti autorizzati che vengono identificati come hacker. Al lato opposto, se si cerca di limitare il problema dei falsi positivi restringendo le attività che vengono interpretate come quelle di un hacker, si cade nel problema opposto, ovvero gli attacchi più subdoli non vengono identificati. Pertanto la pratica del rilevamento delle intrusioni prevede l'adozione di compromessi.

Nello studio di Anderson [ANDE80] si supponeva che fosse possibile discriminare con una certa sicurezza un hacker da un utente legittimo. Gli schemi di comportamento degli utenti legittimi possono essere definiti osservando il passato e questo consente di individuare le deviazioni più significative rispetto a questi schemi. Anderson suggeriva che l'operazione di rilevamento di un utente legittimo che si comporta in modo non autorizzato è più difficile in quanto la distinzione fra comportamento anomalo e normale può essere più flebile. Anderson concludeva che tali violazioni non risulterebbero rilevabili se si utilizzasse esclusivamente la ricerca dei comportamenti anomali. Tuttavia il comportamento non legittimo di un utente interno può essere rilevabile definendo in modo intelligente le classi di condizioni che suggeriscono gli utilizzi non autorizzati. Infine Anderson riteneva che il rilevamento di un utente clandestino non rientrasse nelle possibilità delle tecniche automatiche. Queste osservazioni, effettuate nel 1980, rimangono tuttora valide.

[PORR92] identifica i seguenti approcci al rilevamento delle intrusioni.

1. **Rilevamento statistico delle anomalie:** prevede la raccolta di dati relativi al comportamento degli utenti legittimi lungo un determinato arco di tempo. Questi test statistici vengono applicati ai comportamenti osservati per determinare con un elevato livello di sicurezza se siano legittimi o meno.
   A. Rilevamento a soglie: questo approccio prevede la definizione di soglie, indipendenti dall'utente, relative alla frequenza con cui si verificano determinati eventi.
   B. Rilevamento a profilo: viene sviluppato un profilo delle attività di ciascun utente che viene poi impiegato per rilevare i cambiamenti di comportamento dei singoli account.
2. **Rilevamento a regole:** prevede la definizione di un insieme di regole che possono essere utilizzate per decidere se un determinato comportamento sia di un hacker.
   A. Rilevamento delle anomalie: vengono definite delle regole per rilevare le deviazioni rispetto agli schemi d'uso precedenti.
   B. Rilevamento degli attacchi: un approccio a sistemi esperti che ricerca i comportamenti sospetti.

In pratica, gli approcci statistici tentano di definire il comportamento normale, previsto, mentre gli approcci a regole tendono a definire il comportamento sospetto.

In termini di tipi di hacker elencati in precedenza, il rilevamento statistico delle anomalie è efficace contro gli hacker esterni che difficilmente riproducono gli schemi di comportamento degli account di cui si appropriano. Tuttavia tali tecniche possono essere inefficaci contro gli utenti interni. Per questi attacchi, gli approcci a regole possono essere in grado di riconoscere eventi e sequenze che, nel loro contesto, rivelano un attacco. In pratica un sistema può utilizzare una combinazione di entrambi gli approcci per essere efficace contro un'ampia gamma di attacchi.

Record di auditing

Uno strumento fondamentale per il rilevamento delle intrusioni è rappresentato dai record di auditing. Alcune registrazioni delle attività effettuate dagli utenti sono utili come input al sistema di rilevamento delle intrusioni. Fondamentalmente vengono utilizzate due strategie.

- **Registrazioni di auditing native:** praticamente tutti i sistemi operativi multiutente includono del software che raccoglie informazioni sulle attività degli utenti. Il vantaggio di questo approccio consiste nel fatto che evita di impiegare ulteriore software per la raccolta delle informazioni. Lo svantaggio è che i record nativi possono non contenere tutte le informazioni di cui si ha bisogno o possono contenerle in una forma inappropriata.
- **Registrazioni di auditing specifiche per il rilevamento:** si può implementare un sistema di raccolta delle informazioni che genera dei record di auditing contenenti esclusivamente le informazioni necessarie per il sistema di rilevamento delle intrusioni. Un vantaggio di questo approccio è il fatto che può essere impiegato indipendentemente dal sistema e quindi su sistemi differenti. Lo svantaggio è l'ulteriore sovraccarico dovuto al fatto che, in pratica, sulla macchina operano due pacchetti di registrazione delle attività.

Un buon esempio di registrazioni di auditing specifiche per il rilevamento è quello sviluppato da Dorothy Denning [DENN87]. Ciascun record di auditing contiene i seguenti campi.

- **Soggetto:** l'esecutore delle azioni. Un soggetto è normalmente un utente o comunque un processo che agisce per conto di un utente o di un gruppo di utenti. Tutte le attività derivano dai comandi emessi dai soggetti. I soggetti possono essere raggruppati in varie classi di accesso che possono anche sovrapporsi.
- **Azione:** l'operazione svolta dal soggetto su un determinato oggetto; per esempio può trattarsi di un'azione di login, lettura, I/O o esecuzione.
- **Oggetto:** ciò su cui opera l'azione. Può trattarsi di file, programmi, messaggi, record, terminali, stampanti e strutture create dal programma o dall'utente. Quando un soggetto è destinatario di un'azione, come nel caso della posta elettronica, tale soggetto può anche essere considerato un oggetto. Gli oggetti possono essere raggruppati in base al tipo. La granularità degli oggetti può variare in base al tipo dell'oggetto e all'ambiente. Per esempio le azioni eseguite su un database possono essere registrate a livello dell'intero database o dei record che lo compongono.
- **Condizioni di eccezione:** indicano l'eventuale eccezione sollevata.
- **Uso delle risorse:** un elenco di elementi con un'indicazione del livello di utilizzo della risorsa (per esempio il numero di righe stampate o visualizzate, il numero di record letti o scritti, il tempo di CPU impiegato, le unità di I/O utilizzate, il tempo di sessione trascorso).
- **Time-stamp:** un identificatore temporale che indica il momento in cui si è svolta l'azione.

La maggior parte delle operazioni svolte dagli utenti è costituita da una sequenza di azioni elementari. Per esempio, l'operazione di copia di un file prevede l'esecuzione del comando utente che include la convalida dell'accesso e dell'esecuzione della copia, più la lettura da un file, più la scrittura su un altro file. Si consideri il seguente comando:

```
COPY GAME.EXE TO <Library>GAME.EXE
```

emesso dall'utente Smith per copiare il file eseguibile GAME dalla directory corrente alla directory <Library>. Possono essere generati i seguenti record di auditing:

Smith	execute	<Library> COPY.EXE	0	CPU = 00002	11058721678
Smith	read	<Smith> GAME.EXE	0	RECORDS = 0	11058721679
Smith	write	<Library> COPY.EXE	write-viol	RECORDS = 0	11058721680

In questo caso, la copia viene annullata poiché Smith non ha il permesso di scrittura su <Library>. La decomposizione delle operazioni svolte da un utente in azioni elementari presenta tre vantaggi.

1. Poiché gli oggetti sono entità di un sistema che possono essere protette, l'utilizzo di azioni elementari consente di controllare tutti i comportamenti relativi a un oggetto. Pertanto il sistema potrà rilevare tutti i tentativi di aggirare i controlli di accesso (prendendo nota delle anomalie nel numero di condizioni di eccezione sollevate) e può rilevare gli attacchi svolti con successo notando anomalie nell'insieme degli oggetti accessibili al soggetto.
2. Le registrazioni di auditing riguardanti un solo oggetto e un'unica azione semplificano il modello e l'implementazione.
3. Data la struttura semplice e uniforme dei record di auditing per il rilevamento, può essere relativamente facile ottenere queste informazioni o quanto meno una loro parte con un'associazione diretta fra i record di auditing nativi e i record di auditing specifici per il rilevamento.

Rilevamento statistico delle anomalie

Come si è detto, le tecniche di rilevamento statistico delle anomalie rientrano a grandi linee in due categorie: rilevamento a soglia e a profilo. Il rilevamento a soglia prevede il conteggio di un tipo ben determinato di eventi in un determinato intervallo di tempo. Se il conteggio supera ciò che può essere considerato un numero ragionevole e che può verificarsi normalmente, si presuppone che sia in corso un'intrusione.

L'analisi delle soglie, da sola, è un sistema di rilevamento rozzo e inefficace anche per gli attacchi meno sofisticati. Occorre determinare sia le soglie che l'intervallo temporale. Data la variabilità di comportamento degli utenti, tali soglie genereranno con ogni probabilità una grande quantità di falsi positivi o, al contrario, di falsi negativi. Tuttavia il rilevamento delle soglie può essere utile se affiancato da tecniche più sofisticate.

Il rilevamento delle anomalie a profilo si concentra invece sulla definizione del comportamento passato dei singoli utenti o dei gruppi di utenti individuando le deviazioni significative. Un profilo può essere costituito da un insieme di parametri, in modo che la deviazione su un unico parametro non sia sufficiente per generare un allarme.

La base di questo approccio è costituita dall'analisi dei record di auditing. I record di auditing forniscono l'input per la funzione di rilevamento delle intrusioni in due modi. Innanzitutto, il progettista deve decidere vari metodi quantitativi che possono essere utilizzati per valutare il comportamento degli utenti. Può essere utilizzata un'analisi dei record di auditing lungo un determinato arco di tempo, in modo da determinare il profilo delle attività dell'utente medio. Pertanto i record di auditing serviranno per definire il comportamento tipico. In secondo luogo, i record di auditing correnti rappresentano l'input utilizzato per rilevare l'intrusione. Il modello a rilevamento delle intrusioni analizza i record di auditing in input per determinare le deviazioni rispetto al comportamento medio.

Fra gli esempi di valutazioni che possono essere utili per il rilevamento delle intrusioni a profilo vi sono i seguenti.

- **Contatore:** un intero non negativo che può essere incrementato ma non decrementato o che può essere azzerato da un'azione amministrativa. In genere il conteggio di determinati tipi di eventi viene considerato in un determinato arco di tempo. Fra gli esempi vi sono il numero di login effettuati da un utente in un'ora, il numero di volte che viene

eseguito un determinato comando durante una singola sessione utente e il numero di errori nelle password commessi in un minuto.

- **Sonda:** intero non negativo che può essere incrementato o decrementato. In genere una sonda viene utilizzata per misurare il valore corrente di una determinata entità. Fra gli esempi vi sono il numero di connessioni logiche assegnate a un'applicazione utente e il numero di messaggi in uscita accodati per un processo utente.
- **Timer degli intervalli:** tempo trascorso fra due eventi correlati. Per esempio il tempo trascorso fra due successivi login di un account.
- **Utilizzo delle risorse:** quantità di risorse consumate in un determinato arco di tempo. Per esempio il numero di pagine stampate in una sessione utente e il tempo totale consumato dall'esecuzione di un programma.

Date queste metriche generali, è possibile impiegare vari test per determinare se l'attività corrente rientra in limiti accettabili. [DENN87] elenca i seguenti approcci.

- Media e deviazione standard.
- Multivariazione.
- Processi di Markov.
- Serie temporali.
- Valutazione operativa.

Il test statistico più semplice consiste nel misurare la **media e la deviazione standard** di un parametro in un determinato arco di tempo. Questo dà un'idea del comportamento medio e della sua variabilità. L'uso della media e della deviazione standard è applicabile a un'ampia varietà di contatori, timer e misuratori di risorse. Ma queste misure, da sole, sono in generale troppo rozze per il rilevamento delle intrusioni.

Il modello a **multivariazione** si basa sulla correlazione fra due o più variabili. Il comportamento degli hacker può essere caratterizzato con maggiore sicurezza considerando tali correlazioni (per esempio il tempo del microprocessore e le risorse utilizzate oppure la frequenza di login e il tempo di sessione).

Il modello a **processi di Markov** viene utilizzato per definire le probabilità delle transizioni fra vari stati. Per esempio, questo modello può essere utilizzato per ricercare le transizioni fra determinati comandi.

Il modello a **serie temporali** si concentra sugli intervalli temporali, ricercando sequenze di eventi che si verificano troppo velocemente o troppo lentamente. Possono essere adottati vari test statistici in grado di caratterizzare sequenze temporali anomale.

Infine il **modello operativo** si basa sul giudizio di ciò che è considerato anomalo piuttosto che su un'analisi automatizzata dei record di auditing storici. In genere vengono definiti dei limiti fissi e si sospetta un'intrusione ogni volta che vengono superati questi limiti. Questo tipo di approccio funziona particolarmente bene quando il comportamento dell'hacker può essere dedotto da determinati tipi di attività. Per esempio, una grande quantità di tentativi di login in un breve arco di tempo suggerisce un tentativo di attacco.

Come esempio d'uso di queste varie valutazioni e modelli, la Tabella 18.1 mostra varie misure considerate o verificate dal sistema di rilevamento delle intrusioni IDES (Intrusion Detection System) dello SRI (Stanford Research Institute) [DENN87, JAVI91, LUNT88].

Tabella 18.1 Misure utilizzabili per il rilevamento dell'intrusione.

Misura	Modello	Tipo di intrusione rilevata
	Attività di login e di sessione	
Frequenza e orario di login	Media e deviazione standard	In genere gli hacker tentano di effettuare il login al di fuori dagli orari di lavoro.
Frequenza di login da punti diversi	Media e deviazione standard	Gli hacker possono connettersi da un punto che un determinato utente utilizza raramente o non utilizza mai.
Tempo trascorso dall'ultimo login	Operativo	Violazione di un account non più in uso.
Tempo trascorso per sessione	Medio e deviazione standard	Significative deviazioni possono indicare le attività di un hacker esterno.
	Attività di login e di sessione	
Quantità di output verso l'esterno	Media e deviazione standard	Un volume eccessivo di dati trasmessi verso posizioni remote possono indicare la fuga di dati riservati.
Utilizzo delle risorse per sessione	Media e deviazione standard	Livelli di utilizzo insoliti della CPU o delle attività di I/O possono segnalare la presenza di un hacker.
Errori nell'introduzione della password	Operativo	Tentativi di indovinare la password.
Errori di login a determinati terminali	Operativo	Tentativo di violazione delle password
	Attività di esecuzione di comandi o di programmi	
Frequenza di esecuzione	Media e deviazione standard	Può indicare l'attività di un hacker che probabilmente userà comandi differenti oppure di un utente legittimo che ha acquisito l'accesso a comandi privilegiati.
Utilizzo delle risorse da parte dei programmi	Media e deviazione standard	Un valore anomalo può suggerire l'infezione da virus o da cavallo di Troia, che produce effetti collaterali che aumentano le attività di I/O o l'utilizzo del microprocessore.
Richieste di esecuzione negate	Modello operativo	Può indicare tentativi di attacco da parte di singoli utenti che cercano di acquisire privilegi più elevati.

(segue)

Tabella 18.1 Misure utilizzabili per il rilevamento dell'intrusione. *(continua)*

Misura	Modello	Tipo di intrusione rilevata
	Attività di accesso ai file	
Frequenza di lettura, scrittura, creazione o cancellazione	Media e deviazione standard	Le anormalità per gli accessi in lettura e scrittura dei singoli utenti possono indicare delle attività di attacco.
Record letti o scritti	Media e deviazione standard	Le anormalità possono indicare un tentativo di ottenere dati riservati tramite inferenze e aggregazioni.
Numero di fallimenti per attività di lettura, scrittura, creazione, cancellazione	Operativo	Può indicare tentativi persistenti di accedere a file senza autorizzazione.

Il vantaggio principale dei profili statistici è che non è necessaria alcuna conoscenza precedente dei problemi di sicurezza: il programma di rilevamento apprende il comportamento "normale" e poi ricerca le deviazioni. Questo approccio non si basa su caratteristiche e punti deboli specifici del sistema, pertanto dovrebbe risultare portabile con facilità su sistemi differenti.

Rilevamento delle intrusioni in base a regole

Le tecniche di rilevamento delle intrusioni basate su regole osservano gli eventi in corso nel sistema e applicano un insieme di regole che consentono di decidere se una determinata sequenza di attività è sospetta o meno. In termini molto generali, si possono caratterizzare gli approcci in due classi: basate sul rilevamento delle anomalie o sull'identificazione degli attacchi, anche se vi sono delle sovrapposizioni.

Il **rilevamento delle anomalie a regole** è simile, in termini di approccio e potenza, al rilevamento statistico delle anomalie. Con l'approccio a regole, vengono analizzati dei record di auditing storici per identificare gli schemi d'uso e generare automaticamente delle regole che descrivano tali schemi. Le regole possono rappresentare i comportamenti passati degli utenti, i programmi, i privilegi, le sequenze temporali, i terminali e così via. Viene poi osservato il comportamento corrente e ciascuna transazione viene confrontata con l'insieme di regole per determinare se è conforme allo schema di comportamento osservato storicamente.

Come per il rilevamento statistico, il rilevamento delle anomalie basato su regole non richiede la conoscenza dei punti deboli del sistema. Piuttosto lo schema osserva il comportamento passato e, in pratica, presuppone che il futuro sarà come il passato. Perché questo approccio sia efficace, è necessario adottare un database di regole di dimensioni considerevoli. Per esempio, uno schema descritto in [VACC89] contiene all'incirca 10^4-10^6 regole.

L'**identificazione degli attacchi basata su regole** sfrutta un approccio molto differente per il rilevamento delle intrusioni che si basa sull'adozione di sistemi esperti. La caratteristica principale di tali sistemi è l'uso di regole per l'identificazione degli attacchi noti o degli attacchi che sfruttano punti deboli noti. Possono anche essere definite delle regole che identificano i comportamenti sospetti, anche se tali comportamenti rientrano nei limiti degli schemi d'uso precedenti. In genere le regole utilizzate in questi sistemi sono specifiche della macchina e del sistema operativo. Inoltre tali regole vengono generate da esperti e non per mezzo di analisi automatizzate dei record di auditing. La procedura normale consiste nell'interrogare gli amministratori di sistema e gli analisti della sicurezza per raccogliere un insieme di situazioni di attacco note e di eventi chiave che minacciano la sicurezza del sistema.[1] Pertanto la resistenza di questo approccio dipende dall'abilità di coloro che sono coinvolti nella definizione delle regole.

Un semplice esempio del tipo di regole che possono essere utilizzate si trova in NIDX, uno dei primi sistemi che utilizzava regole euristiche che consentivano di assegnare un grado di sospetto alle attività [BAUE88]. Fra gli esempi di regole euristiche vi sono i seguenti.

1. Gli utenti non dovrebbero leggere i file contenuti nelle directory personali di altri utenti.
2. Gli utenti non dovrebbero scrivere su file di altri utenti.
3. Gli utenti che si connettono spesso accedono agli stessi file utilizzati in precedenza.
4. Gli utenti in genere non accedono direttamente ai dischi ma utilizzano i servizi di alto livello offerti dal sistema operativo.
5. Gli utenti non dovrebbero trovarsi connessi più volte allo stesso sistema.
6. Gli utenti non eseguono copie dei programmi di sistema.

Lo schema di identificazione dell'attacco utilizzato in IDES è rappresentativo della strategia seguita. I record di auditing vengono esaminati subito dopo essere stati generati e vengono confrontati con la base di regole. Se viene trovata una corrispondenza, il livello di sospetto dell'utente cresce. Se viene individuata una corrispondenza con un numero sufficiente di regole, la valutazione supera una certa soglia e provoca la generazione di un'anomalia.

L'approccio IDES si basa sull'esame dei record di auditing. Un punto debole di questo meccanismo è la sua mancanza di flessibilità. Per una determinata situazione di attacco, vi possono essere varie sequenze di record di auditing alternative con variazioni lievi o subdole. Può essere difficile specificare tutte le varianti tramite regole esplicite. Un altro metodo consiste nello sviluppare un modello di più alto livello indipendente dai record di auditing specifici. Un esempio è il modello a transizione di stati chiamato USTAT [ILGU93]. USTAT si occupa di azioni generali piuttosto che di specifiche azioni dettagliate e registrate dal meccanismo di auditing di Unix. USTAT è implementato su un sistema SunOS che genera record di auditing su 239 eventi. Di questi, solo 28 vengono utilizzati da un preprocessore che li associa a dieci azioni generali (vedere la Tabella 18.2). Utilizzando solo queste azioni e i parametri associati, viene sviluppato un diagramma delle transizioni di stato che consente di individuare le attività sospette.

[1] Tali interviste possono essere estese agli hacker (pentiti o meno) che intendono condividere la loro esperienza a pagamento [FREE93].

Poiché un gran numero di eventi soggetti ad auditing viene associato a un numero più ridotto di azioni, il processo di creazione delle regole è più semplice. Inoltre il modello a transizioni di stato può essere modificato con facilità per considerare nuovi metodi di attacco.

Base rate fallacy: la stima dei falsi allarmi

Per poter essere di qualche utilità pratica, un sistema di rilevamento delle intrusioni dovrebbe rilevare una grande percentuale delle intrusioni mantenendo nel contempo i falsi allarmi a un livello accettabile. Se rileva solo una percentuale modesta delle intrusioni effettivamente subite, il sistema fornisce un falso senso di sicurezza. Se al contrario il sistema fa scattare troppi falsi allarmi quando in realtà non è in corso alcuna intrusione, gli amministratori inizieranno a ignorare gli allarmi o al contrario dedicheranno troppo tempo all'analisi dei falsi allarmi.

Sfortunatamente, data la natura delle probabilità coinvolte, è molto difficile riuscire a ottenere un elevato tasso di rilevamento con un basso tasso di falsi allarmi. In generale, se il numero effettivo delle intrusioni è basso rispetto al numero di utenti legittimi di un sistema, il numero di falsi allarmi sarà elevato, a meno che i controlli siano estremamente precisi. Uno studio dei sistemi di rilevamento delle intrusioni esistenti presentato in [AXEL00] mostra che i sistemi attualmente in uso non hanno ancora risolto il problema dell'elevato tasso di falsi allarmi. Per informazioni di base sulla matematica di questo problema, consultare l'Appendice 18.A.

Tabella 18.2 Azioni USTAT e tipi di eventi SunOS.

Azione USTAT	Tipo di evento SunOS
Read	open_r, open_rc, open_rtc, open_rwc, open_rwtc, open_rt, open_rw, open_rwt
Write	truncate, ftruncate, creat, open_rtc, open_rwc, open_rwtc, open_rt, open_rw, open_rwt, open_w, open_wt, open_wc, open_wct
Create	mkdir, creat, open_rc, open_rtc, open_rwc, open_rwtc, open_wc, open_wtc, mknod
Delete	rmdir, unlink
Execute	exec, execve
Exit	exit
Modify_Owner	chown, fchown
Modify_Perm	chmod, fchmod
Rename	rename
Hardlink	link

Sistemi distribuiti di rilevamento delle intrusioni

Fino a poco tempo fa, il lavoro sui sistemi di rilevamento delle intrusioni si è concentrato su sistemi indipendenti. Ma in genere un'azienda deve difendere un insieme distribuito di host connessi da una rete locale o da una interconnessione fra reti. Sebbene sia possibile installare una difesa utilizzando sistemi di rilevamento delle intrusioni indipendenti su ciascun host, la difesa più efficace può essere ottenuta coordinando i vari sistemi di rilevamento delle intrusioni presenti nella rete.

Porras evidenzia i seguenti principali problemi nella progettazione dei sistemi distribuiti di rilevamento delle intrusioni [PORR92].

- Un sistema distribuito di rilevamento delle intrusioni può dover gestire vari formati di record di auditing. In un ambiente eterogeneo, sistemi differenti impiegheranno metodi differenti di raccolta delle informazioni di auditing e, se si utilizza il rilevamento delle intrusioni, potranno essere impiegati anche formati differenti per i record di auditing per la sicurezza.
- Uno o più nodi della rete fungono da punti di raccolta e analisi dei dati di tutti i sistemi della rete. Dunque occorre trasmettere attraverso la rete i dati grezzi di auditing o un loro riepilogo. Pertanto vi è la necessità di garantire l'integrità e la segretezza di questi dati. L'integrità è necessaria per impedire che un hacker possa nascondere le proprie attività alterando le informazioni di auditing trasmesse. La segretezza è necessaria poiché le informazioni di auditing trasmesse possono essere preziose.
- Si può utilizzare un'architettura centralizzata o decentrata. Con un'architettura centralizzata, vi è un unico punto centrale di raccolta e analisi di tutti i dati di auditing. Questo facilita la correlazione dei report in arrivo ma crea un potenziale collo di bottiglia e un unico punto critico (relativamente a guasti potenziali). Con un'architettura decentrata, vi sono più centri di analisi ma questi devono coordinare le proprie attività e scambiarsi informazioni.

Un buon esempio di sistema distribuito per il rilevamento delle intrusioni è quello sviluppato dall'università di California [HEBE92, SNAP91]. La Figura 18.2 mostra l'architettura globale del sistema che è costituito da tre componenti principali.

- **Agente di monitoring degli host:** modulo per la raccolta delle informazioni di auditing che opera come un processo in background sul sistema monitorato. Il suo scopo è quello di raccogliere dall'host i dati sugli eventi relativi alla sicurezza e trasmetterli al sistema centrale.
- **Agente di monitoring della rete locale:** si comporta come l'agente per gli host ma analizza il traffico della rete locale e invia i risultati al sistema centrale.
- **Modulo centrale di gestione:** riceve i report dagli agenti di monitoring della rete locale e degli host ed effettua tutte le elaborazioni e le correlazioni per consentire il rilevamento delle intrusioni.

Questo meccanismo è progettato per essere indipendente dal sistema operativo o dall'implementazione di auditing del sistema. La Figura 18.3 [SNAP91] mostra l'approccio generale utilizzato. L'agente cattura ciascun record di auditing prodotto dal sistema nativo di

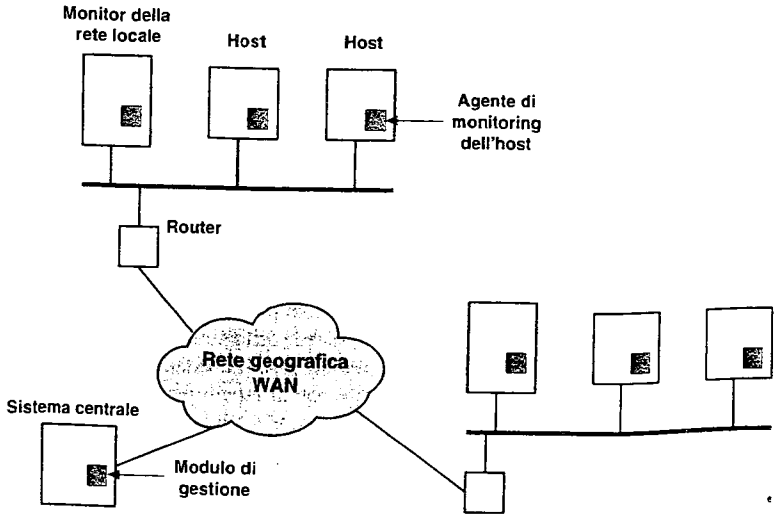


Figura 18.2 L'architettura di un sistema distribuito di rilevamento delle intrusioni.

raccolta delle informazioni di auditing. Viene applicato un filtro che conserva solo i record rilevanti alla sicurezza. Questi record vengono poi riformulati in un formato standard chiamato HAR (Host Audit Record). Poi un modulo logico a modelli analizza i record alla ricerca di attività sospette. Al livello più basso, l'agente esegue la scansione degli eventi degni di nota e che sono interessanti indipendentemente dagli eventi passati. Fra gli esempi vi sono gli errori di accesso ai file, l'accesso a file di sistema e il cambiamento del controllo degli accessi a un file. Al livello superiore, l'agente individua le sequenze di eventi, per esempio relative ai metodi di attacco noti (signature o firme). Infine l'agente ricerca i comportamenti anomali dei singoli utenti sulla base del loro profilo storico, per esempio il numero di programmi eseguiti, il numero di file consultati e così via.

Quando viene rilevata un'attività sospetta, viene inviato un allarme al sistema centrale di gestione. Questo sistema include un sistema esperto che è in grado di trarre inferenze dai dati ricevuti. Il sistema centrale può anche interrogare i singoli sistemi richiedendo copie dei record HAR per correlarle con quelle fornite da altri agenti.

L'agente di monitoring della rete locale fornisce anch'esso informazioni al manager centrale. Il monitor della rete locale rileva le connessioni fra gli host, i servizi utilizzati e il volume del traffico. Ricerca eventi significativi quali improvvisi cambiamenti nel carico della rete, l'uso di servizi di sicurezza e attività di rete come *rlogin*.

L'architettura rappresentata nelle Figure 18.2 e 18.3 è piuttosto generale e flessibile. Offre una struttura di base per un approccio indipendente dalla macchina che può espandersi da un sistema di rilevamento delle intrusioni indipendente fino a diventare un sistema in grado di correlare le attività svolte su più siti e reti in modo da individuare le attività sospette che altrimenti non verrebbero rilevate.

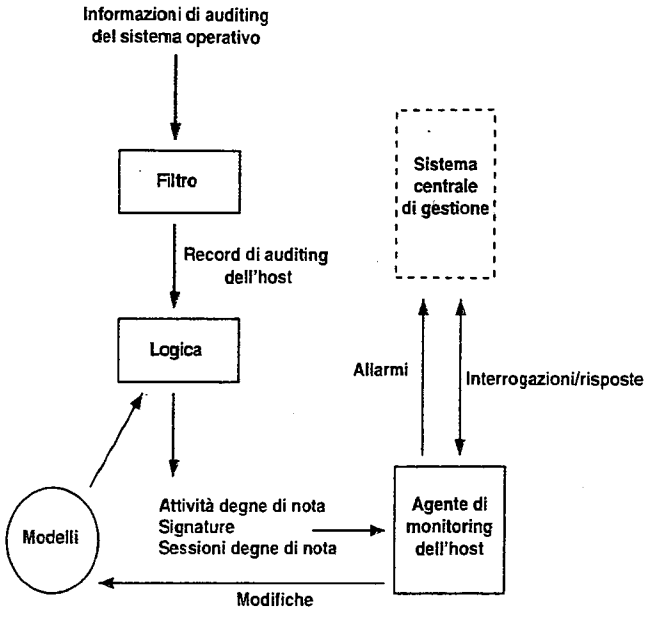


Figura 18.3 L'architettura dell'agente.

Honeypot

Un'innovazione relativamente recente nella tecnologia di rilevamento delle intrusioni è costituita dagli honeypot, letteralmente "vasi di miele". Gli honeypot sono sistemi fittizi progettati per tener lontano un potenziale hacker dai sistemi critici. I sistemi honeypot hanno i seguenti scopi:

- distrarre un hacker dall'accesso ai sistemi critici;
- raccogliere informazioni sulle attività degli hacker;
- incoraggiare l'hacker a rimanere nel sistema per un tempo sufficiente per consentire agli amministratori di rispondere all'attacco.

Questi sistemi sono pieni di informazioni fasulle realizzate appositamente per sembrare preziose ma che non interesserebbero a un utente legittimo del sistema. Pertanto, ogni accesso all'honeypot è sospetto. Il sistema è dotato di monitor molto sensibili, in grado di rilevare questi accessi e di raccogliere informazioni sulle attività dell'hacker. Poiché ogni attacco contro un honeypot sembrerà avere successo, gli amministratori avranno tutto il tempo per attivarsi e registrare le attività dell'hacker senza esporre in alcun modo i sistemi veri e propri.

Gli sforzi iniziali consistevano nel creare un singolo computer honeypot con indirizzi IP progettati appositamente per attrarre gli hacker. Le ricerche più recenti si sono concentrate sulla realizzazione di intere reti honeypot che emulano la rete aziendale, dotate perfino di traffico effettivo o simulato e dati. Una volta che gli hacker entrano nella rete, gli amministratori possono osservare il loro comportamento in dettaglio e stabilire le difese.

Il formato di scambio delle informazioni tra sistemi di rilevamento delle intrusioni

Per facilitare lo sviluppo di sistemi distribuiti di rilevamento delle intrusioni in grado di operare su un'ampia gamma di piattaforme e ambienti, è necessario sviluppare degli standard di interoperabilità. Tali standard sono sviluppati dal gruppo di lavoro IETF "Intrusion Detection Working Group". Lo scopo di questo gruppo di lavoro è quello di definire i formati dei dati e le procedure di scambio per la condivisione delle informazioni utili ai sistemi di rilevamento delle intrusioni e di difesa e ai sistemi di gestione che possono dover interagire con essi. Il gruppo di lavoro produrrà i seguenti risultati.

1. Un documento dei requisiti, che descrive, motivandoli, i requisiti funzionali di alto livello per le comunicazioni fra i sistemi di rilevamento delle intrusioni e i requisiti per le comunicazioni fra i sistemi di rilevamento delle intrusioni e i sistemi di gestione. Per illustrare i requisiti verranno anche definite specifiche situazioni.
2. Una specifica del linguaggio comune per il rilevamento delle intrusioni, che descrive il formato dei dati che soddisfano i requisiti.
3. Un documento di base che identifica i protocolli esistenti più utili per le comunicazioni fra i sistemi di rilevamento delle intrusioni e che descrive i formati utilizzati per i dati.

Al momento attuale, tutti questi documenti sono in fase di bozza Internet.

18.3 Gestione delle password

Protezione delle password

La prima linea di difesa contro gli hacker è rappresentata dal sistema delle password. Praticamente tutti i sistemi multiutente richiedono che l'utente introduca il proprio nome o un identificatore e una password. La password consente di autenticare il codice utente di colui che si sta collegando al sistema. A sua volta il codice utente supporta la sicurezza nei modi seguenti.

- Determina se l'utente è autorizzato ad accedere a un sistema. In alcuni sistemi, solo gli utenti registrati.
- Determina i privilegi assegnati all'utente. Alcuni utenti possono essere supervisori ("superuser") ovvero possono svolgere funzioni e leggere file particolarmente protetti dal sistema operativo. Alcuni sistemi sono dotati di account guest o anonimo che hanno privilegi più limitati rispetto ai normali utenti.

- Viene utilizzato per un controllo discrezionale degli accessi. Per esempio un utente, elencando i codici di altri utenti, può consentire loro di leggere i propri file.

La vulnerabilità delle password

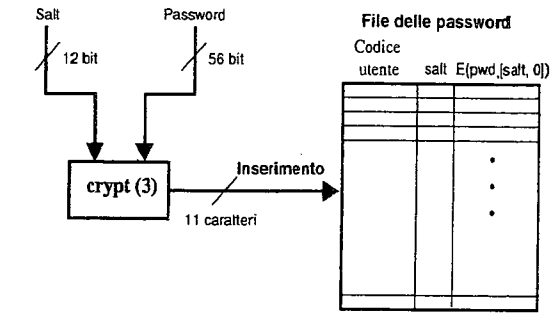
Per comprendere la natura delle minacce contro i sistemi a password, si consideri il meccanismo ampiamente utilizzato in Unix in cui le password non vengono mai conservate in chiaro. Viene invece impiegata la seguente procedura (rappresentata nella Figura 18.4A). Ciascun utente seleziona una password con una lunghezza minima di 8 caratteri. La password viene convertita in un valore di 56 bit (utilizzando la codifica ASCII a 7 bit) che funge da chiave di input per una routine di crittografia. La routine di crittografia, chiamata crypt (3) si basa sull'algoritmo DES. L'algoritmo DES viene modificato utilizzando un valore "salt" di 12 bit. In genere questo valore è in relazione con l'istante in cui è stata assegnata la password all'utente. L'algoritmo DES modificato agisce su un input costituito da un blocco di 64 bit di valori "0". L'output dell'algoritmo funge poi da input per una seconda crittografia. Questa operazione viene ripetuta per un totale di 25 crittografie. Il valore di output risultante a 64 bit viene infine tradotto in una sequenza di 11 caratteri. Il codice hash così ottenuto dalla password viene quindi memorizzato, insieme a una copia in chiaro del "salt", nel file delle password con riferimento al codice utente corrispondente. Questo metodo si è rivelato sicuro contro numerosi attacchi crittanalitici.

Il valore "salt" ha tre scopi.

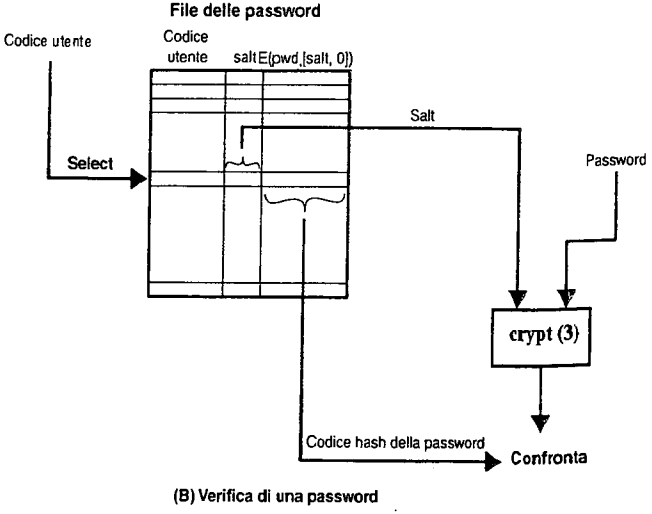
- Evita la possibilità che nel file delle password due password uguali risultino visibili. Anche se due utenti scegliessero la stessa password, tali password verrebbero assegnate in istanti differenti e pertanto il codice memorizzato per la password dei due utenti risulterebbe differente.
- Aumenta la lunghezza della password senza che l'utente debba ricordare due ulteriori caratteri. Il numero di password possibili aumenta di un fattore 4096, aumentando pertanto la difficoltà di indovinare la password.
- Impedisce l'uso di implementazioni hardware di DES che faciliterebbero un attacco alle password tramite metodi a forza bruta.

Quando un utente tenta di connettersi a un sistema Unix, deve fornire il proprio codice utente e la relativa password. Il sistema operativo usa il codice utente per eseguire la ricerca all'interno del file delle password e prelevare il valore salt in chiaro e la password crittografata. Il valore salt e la password fornita dall'utente vengono utilizzati come input per la routine di crittografia. Se il risultato coincide con il valore memorizzato della password, questa viene accettata.

La routine di crittografia è progettata in modo da scoraggiare gli attacchi per tentativi. Le implementazioni software di DES sono lente rispetto alle versioni hardware e l'utilizzo di 25 iterazioni moltiplica per 25 il tempo richiesto. Tuttavia, dopo la progettazione iniziale di questo algoritmo, sono avvenuti due cambiamenti. Innanzitutto le implementazioni più recenti dell'algoritmo sono molto più veloci. Per esempio, il worm Internet descritto nel Capitolo 19 è stato in grado di indovinare online qualche centinaio di password in un tempo ragionevolmente breve utilizzando un algoritmo di crittografia più efficiente rispet-



(A) Caricamento di una nuova password



(B) Verifica di una password

Figura 18.4 Il meccanismo di gestione delle password in Unix.

to a quello standard contenuto nei sistemi Unix attaccati. In secondo luogo, le prestazioni dell'hardware continuano ad aumentare e dunque qualsiasi algoritmo software viene eseguito con maggiore rapidità.

Pertanto si devono considerare due minacce al meccanismo delle password di Unix. Innanzitutto un utente può acquisire l'accesso a una macchina utilizzando un account guest e poi eseguire su tale macchina un programma per l'individuazione delle password (password

cracker). L'hacker potrebbe essere in grado di acquisire centinaia e anche migliaia di password con un consumo limitato di risorse del sistema. Inoltre, se l'hacker riuscisse a ottenere una copia del file delle password, potrebbe applicarvi il programma cracker sulla propria macchina a suo piacimento. Questo consentirebbe all'hacker di tentare molte migliaia di password in un arco di tempo ragionevole.

Come esempio, si citerà un programma password cracker individuato in Internet nell'agosto del 1993 [MADS93]. Utilizzando un computer parallelo Thinking Machines Corporation si poterono ottenere prestazioni di 1560 crittografie al secondo per unità vettoriale. Con quattro unità vettoriali per ogni nodo di elaborazione (una configurazione standard), si ottengono 800 000 crittografie al secondo su una macchina di 128 nodi (di dimensioni modeste) e 6,4 milioni di crittografie al secondo su una macchina di 1024 nodi.

Ma anche questa potenza di calcolo non consente ancora a un hacker di utilizzare una tecnica cieca a forza bruta e di tentare tutte le possibili combinazioni di caratteri per scoprire la password. Al contrario, i password cracker contano sul fatto che molti utenti scelgono password facilmente individuabili.

Alcuni utenti, quando viene loro offerta la possibilità di scegliere la propria password, ne introducono una assurdamente breve. La Tabella 18.3 mostra i risultati di uno studio della Purdue University. Lo studio ha analizzato il cambio di password su 54 macchine per un totale di circa settemila account utente. Circa il 3% delle password aveva una lunghezza di tre o meno caratteri. Un hacker potrebbe iniziare l'attacco ricercando esaustivamente tutte le password di lunghezza minore o uguale a tre caratteri. Una semplice soluzione consiste nel fare in modo che il sistema rifiuti una password più corta di, per esempio, 6 caratteri o che richieda addirittura che tutte le password abbiano una lunghezza esattamente di 8 caratteri. La maggior parte degli utenti non si lamenterebbe di queste restrizioni.

Tabella 18.3 Lunghezza osservata delle password (SPAF92a).

Lunghezza	Numero	Frazione del totale
1	55	0,004
2	87	0,006
3	212	0,02
4	449	0,03
5	1260	0,09
6	3035	0,22
7	2917	0,21
8	5772	0,42
Totale	**13787**	**1,0**

La lunghezza della password rappresenta solo una parte del problema. Molte persone, quando possono scegliere la propria password, ne scelgono una troppo facilmente individuabile: per esempio il proprio nome, il nome della via in cui abitano, una parola presente nel dizionario e così via. Questo semplifica moltissimo l'individuazione della password: l'hacker non dovrà fare altro che confrontare il file delle password con un elenco delle password più probabili. Poiché molti utenti usano password facilmente individuabili, una strategia di questo tipo avrebbe successo praticamente su qualsiasi sistema.

Una dimostrazione dell'efficacia di questa tecnica si trova in [KLEI90]. L'autore ha raccolto da varie fonti dei file di password Unix contenenti circa 14000 password crittografate. Il risultato, che l'autore ha definito agghiacciante, è indicato nella Tabella 18.4. È stato individuato circa un quarto delle password utilizzando la seguente strategia.

1. Utilizzare il nome dell'utente, le iniziali, il nome dell'account e altre informazioni personali. In totale sono state tentate 130 diverse permutazioni per ciascun utente.
2. Utilizzare parole tratte da vari dizionari. L'autore aveva compilato un primo dizionario di più di 60000 parole, ottenute dal dizionario online del sistema stesso e da vari altri elenchi.
3. Utilizzare varie permutazioni delle parole del passo 2. Per esempio provare la prima lettera maiuscola o introdurre un carattere di controllo per trasformare l'intera parola in maiuscolo, invertire la parola, trasformare la lettera "o" nella cifra "0" e così via. Queste permutazioni hanno aggiunto all'elenco un altro milione di parole.
4. Tentare varie permutazioni fra lettere maiuscole e minuscole delle parole del passo 2 che non erano state considerate nel passo 3. Questo ha aggiunto all'elenco altri due milioni di parole.

Pertanto il test considerava circa tre milioni di parole. Utilizzando l'implementazione Thinking Machines più veloce descritta in precedenza, il tempo necessario per crittografare tutte queste parole e tutti i possibili valori salt è di circa un'ora. Si deve considerare che una ricerca così ampia produrrebbe un tasso di successo di circa il 25% ma che anche un unico tentativo potrebbe essere sufficiente per acquisire un'ampia gamma di privilegi su un sistema.

Tabella 18.4 Password violate su un campione di 13 797 account (KLEI90).

Tipo di password	Dimensioni ricerca	Numero corrispondenze	Percentuale di password individuate	Rapporto costi/ benefici[A]
Nome utente/account	130	368	2,7%	2,830
Sequenze di caratteri	866	22	0,2%	0,025
Numeri	427	9	0,1%	0,021
Nomi cinesi	392	56	0,4%	0,143
Nomi di luoghi	628	82	0,6%	0,131

(segue)

Tabella 18.4 Password violate su un campione di 13 797 account (KLEI90). *(continua)*

Tipo di password	Dimensioni ricerca	Numero corrispondenze	Percentuale di password individuate	Rapporto costi/ benefici[A]
Nomi comuni	2239	548	4,0%	0,245
Nomi femminili	4280	161	1,2%	0,038
Nomi maschili	2866	140	1,0%	0,049
Nomi non comuni	4955	130	0,9%	0,026
Miti e leggende	1246	66	0,5%	0,053
Shakespeare	473	11	0,1%	0,023
Termini sportivi	238	32	0,2%	0,134
Fantascienza	691	59	0,4%	0,085
Film e attori	99	12	0,1%	0,121
Cartoni animati	92	9	0,1%	0,098
Personaggi famosi	290	55	0,4%	0,190
Frasi e modi di dire	933	253	1,8%	0,271
Soprannomi	33	9	0,1%	0,273
Biologia	58	1	0,0%	0,017
Dizionario di sistema	19683	1027	7,4%	0,052
Nomi di macchine	9018	132	1,0%	0,015
Codici mnemonici	14	2	0,0%	0,143
Bibbia	7525	83	0,6%	0,011
Parole varie	3212	54	0,4%	0,017
Parole Yiddish	56	0	0,0%	0,000
Asteroidi	2407	19	0,1%	0,007
Totale	62727	3340	24,2%	0,053

[A] Calcolato come il numero di corrispondenze individuate diviso per le dimensioni della ricerca. Più parole devono essere verificate per trovare di una corrispondenza e minore sarà il rapporto costi/ benefici.

Controllo degli accessi

Un modo per sventare un attacco alle password consiste nell'impedire all'hacker di accedere al file delle password. Se la porzione crittografata del file delle password è accessibile solo agli utenti privilegiati, l'hacker non potrà leggere il file se prima non ottiene la password di un utente privilegiato. [SPAF92a] evidenzia alcuni difetti di questa strategia.

- Molti sistemi, fra cui la maggior parte dei sistemi Unix, sono suscettibili a violazioni impreviste. Una volta che un hacker ha acquisito l'accesso in qualche modo, potrebbe voler ottenere una serie di password per poter utilizzare account differenti in sessioni di login differenti, in modo da ridurre i rischi di rilevamento. In alternativa un utente con un account potrebbe voler sfruttare gli accessi privilegiati di un altro account o sabotare il sistema.
- Un incidente nel meccanismo di protezione potrebbe rendere leggibile il file delle password, compromettendo pertanto tutti gli account.
- Alcuni degli utenti hanno degli account su altre macchine in altri domini di protezione per i quali usano la stessa password. Pertanto, l'identificazione delle password su una macchina potrebbe provocare una violazione anche delle altre macchine.

Occorre quindi una strategia più efficace che costringa gli utenti a scegliere delle password difficili da indovinare.

Strategie per la scelta della password

La lezione che si può trarre dai due esperimenti appena descritti (Tabelle 18.3 e 18.4) è che molti utenti, se lasciati senza controllo, scelgono password troppo brevi o troppo facili da indovinare. All'estremità opposta, se agli utenti vengono assegnate delle password costituite da 8 caratteri stampabili scelti casualmente, risulterà praticamente impossibile violare le password. Ma sarà anche praticamente impossibile per molti utenti ricordare la propria password. Fortunatamente, anche limitando l'universo delle password a stringhe di caratteri ragionevolmente facili da ricordare, le dimensioni dell'universo delle password risultano troppo ampie per poter violare queste ultime. L'obiettivo è quello di eliminare le password troppo facili da indovinare e consentire all'utente di scegliere una password facile da ricordare. Si possono adottate quattro tecniche.

- Istruzioni all'utente.
- Password generate dal computer.
- Controllo delle password reattivo.
- Controllo delle password proattivo

Si può comunicare agli utenti la necessità di utilizzare password difficili da indovinare e si possono fornire delle direttive per la scelta di password più robuste. Questa **strategia di istruzione** degli utenti ha scarse probabilità di successo nella maggior parte delle installazioni, specialmente quando vi è una ampia base di utenti e un notevole turnover. Molti utenti ignoreranno semplicemente le indicazioni. Altri potrebbero avere idee errate su cosa sia una "password robusta". Per esempio, molti utenti ritengono (erroneamente) che basti

invertire una parola o scegliere l'ultima lettera maiuscola per renderla difficile da indovinare.

Password generate dal computer: anche queste password presentano dei problemi. Se le password hanno una natura piuttosto casuale, gli utenti non saranno in grado di ricordarle. Anche nel caso in cui la password fosse pronunciabile, l'utente potrebbe avere difficoltà nel ricordarla ed essere tentato di trascriverla. In generale, i meccanismi di generazione automatica delle password sono male accettati dagli utenti. Il documento FIPS PUB 181 definisce uno dei migliori generatori di password automatici. Lo standard include non solo una descrizione dell'approccio ma anche un listato completo del codice sorgente C dell'algoritmo. L'algoritmo genera parole costituite da sillabe pronunciabili e le concatena per formare una parola. Un generatore di numeri casuali produce un flusso casuale di caratteri che viene impiegato per costruire le sillabe delle parole.

La **strategia di controllo reattivo** delle password prevede che il sistema esegua periodicamente il proprio password cracker per individuare le password facili da indovinare. Il sistema annulla tutte le password che riesce a indovinare e avverte l'utente. Questa tecnica presenta però alcuni difetti. Innanzitutto, se si vuole applicare questa tecnica in modo efficace, si dovranno eseguire operazioni intensive a livello delle risorse. Dato che un hacker che sia in grado di sottrarre il file delle password può dedicare tutto il tempo di CPU all'individuazione delle password per ore o anche per giorni, impiegare un controllo delle password reattivo efficace rappresenta un notevole svantaggio. Inoltre, ogni password esistente rimane vulnerabile finché non viene individuata dal controllo reattivo.

L'approccio più promettente per migliorare la sicurezza delle password è il **controllo proattivo**. Con questo meccanismo, un utente può selezionare una password a proprio piacimento. Poi il sistema controlla se la password è accettabile e, in caso contrario, la rifiuta. Tali sistemi si basano sulla filosofia che, con indicazioni sufficienti da parte del sistema, gli utenti sono in grado di selezionare password facili da ricordare ma difficili da indovinare con un attacco a dizionario.

In realtà la verifica proattiva delle password cerca di trovare un compromesso fra accettabilità da parte dell'utente e robustezza. Se il sistema rifiuta troppe password, gli utenti si lamenteranno del fatto che è troppo difficile scegliere una password. Ma se il sistema usa un algoritmo troppo semplice per definire cosa è accettabile, i cracker otterranno indicazioni per raffinare la tecnica di ricerca. In questa parte del capitolo si vedranno i vari approcci al controllo proattivo delle password.

Il primo approccio è un semplice sistema per far rispettare alcune regole. Per esempio, possono essere attivate le seguenti regole.

- Tutte le password devono avere una lunghezza minima di 8 caratteri.
- Nei primi 8 caratteri, le password devono includere quanto meno una lettera maiuscola, una lettera minuscola, una cifra e un segno di punteggiatura.

Queste regole possono essere affiancate da note per l'utente. Sebbene questo approccio sia superiore alla semplice istruzione degli utenti, può non essere sufficiente per sventare gli attacchi dei programmi password cracker. Questo schema avverte i cracker di quali password non provare e potrebbe non impedire la violazione delle password.

Un'altra procedura possibile consiste semplicemente nel compilare un dizionario di tutte le password "inaccettabili". Quando un utente seleziona una password, il sistema controlla se tale password è presente in tale elenco. Questo approccio presenta due problemi.

- **Spazio:** per poter essere efficace il dizionario deve essere molto esteso. Per esempio, il dizionario utilizzato nello studio Purdue [SPAF92a] occupa più di 30 MB di memoria.
- **Tempo:** può essere necessario molto tempo per eseguire una ricerca in un dizionario molto esteso. Inoltre, per controllare tutte le probabili permutazioni delle parole del dizionario, o si includono tali parole nel dizionario stesso, rendendolo decisamente enorme, oppure occorre introdurre una certa dose di elaborazione in tutte le ricerche.

Esistono due tecniche per sviluppare un sistema di controllo proattivo delle password efficace ed efficiente basato sul rifiuto delle parole presenti in un elenco. Uno di questi sviluppa un modello di Markov per la generazione delle password facili da indovinare [DAVI93]. La Figura 18.5 mostra una versione semplificata di questo modello con un linguaggio costituito da un alfabeto di tre caratteri. Lo stato del sistema corrisponde alla lettera più recente. Il valore della transizione da uno stato a un altro rappresenta la probabilità che una lettera segua un'altra. Pertanto la probabilità che la lettera successiva sia "b" considerando che la lettera corrente è "a" sarà pari a 0,5.

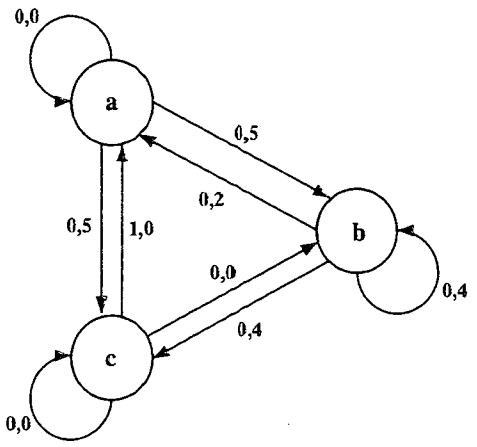
In generale un modello di Markov è una quadrupla $[m, A, \mathbf{T}, k]$, dove m è il numero di stati del modello, A è lo spazio degli stati, $\mathbf{T}$ è la matrice delle probabilità delle transizioni e k è l'ordine del modello. Per un modello di ordine k, la probabilità di effettuare una transizione a una determinata lettera dipende dalle precedenti k lettere che sono state generate. La Figura 18.5 mostra un semplice modello di ordine 1.

Gli autori discutono lo sviluppo e l'impiego di un modello di ordine 2. Innanzitutto viene costruito un dizionario delle password facili da indovinare. Poi la matrice delle transizioni viene calcolata nel modo seguente.

1. Determinare la matrice delle frequenze $\mathbf{f}$ dove $\mathbf{f}(i, j, k)$ è il numero delle presenze del trigramma costituito dai caratteri i, j e k. Per esempio, la password *parsnips* fornisce i trigrammi par, ars, rsn, sni, nip e ips.
2. Per ogni bigramma ij, calcolare $\mathbf{f}(i, j, \infty)$ come il numero totale di trigrammi che iniziano con ij. Per esempio, f(a, b, ∞) sarà il numero totale di trigrammi nella forma aba, abb, abc e così via.
3. Calcolare gli elementi di $\mathbf{T}$ nel seguente modo:

$$\mathbf{T}(i,j,k) = \frac{\mathbf{f}(i,j,k)}{\mathbf{f}(i,j,\infty)}$$

Il risultato è un modello che riflette la struttura delle parole contenute nel dizionario. Con questo modello, la domanda "La password è inaccettabile?" si trasforma nella domanda "La stringa (password) è generabile da questo modello di Markov?". Per una determinata password, si possono ricercare le probabilità delle transizioni di tutti i suoi trigrammi. Per determinare se la password è probabile o improbabile per questo modello possono essere utilizzati semplici test statistici. Le password che possono essere generate con probabilità elevata da questo modello vengono rifiutate. Gli autori dichiarano di aver ottenuto buoni risultati per un modello di ordine 2. Il loro sistema individua praticamente tutte le password



M = {3, {a, b, c} , T, i } dove

$$\mathbf{T} = \begin{bmatrix} 0{,}0 & 0{,}5 & 0{,}5 \\ 0{,}2 & 0{,}4 & 0{,}4 \\ 1{,}0 & 0{,}0 & 0{,}0 \end{bmatrix}$$

Esempio di stringa probabilmente appartenente al linguaggio: abbcacaba
Esempio di stringa probabilmente non appartenente a questo linguaggio: aacccbaaa

Figura 18.5 Un esempio di modello di Markov.

contenute nel dizionario e non esclude troppe password potenzialmente "buone" da creare disagio agli utenti.

Spafford [SPAF92a, SPAF92b] presenta un approccio piuttosto differente che si basa sull'uso di un filtro di Bloom [BLOO70]. Per iniziare si spiegherà il funzionamento del filtro di Bloom. Un filtro di Bloom di ordine k è costituito da un insieme di funzioni hash indipendenti $H_1(x)$, $H_2(x)$, . . . , $H_k(x)$ dove ciascuna funzione mappa una password in un valore hash nell'intervallo compreso fra 0 e $N - 1$. In pratica:

$$H_i(X_j) = y \qquad 1 \le i \le k \qquad 1 \le j \le D \qquad 0 \le y \le N-1$$

dove:

X_j = j-esima parola nel dizionario delle password
D = numero di parole contenute nel dizionario delle password.

Quindi al dizionario viene applicata la seguente procedura.

1. Viene definita una tabella hash di N bit che inizialmente sono impostati tutti a "0".
2. Per ciascuna password, vengono calcolati i suoi k valori hash e i bit corrispondenti nella tabella hash vengono impostati a 1. Pertanto, se $H_i(X_j) = 67$ per una coppia (i, j), il sessantasettesimo bit della tabella hash viene impostato a 1; se il bit contiene già il valore 1, rimane tale.

Quando al controllo viene presentata una nuova password, vengono calcolati i suoi k valori hash. Se tutti i bit corrispondenti della tabella hash sono uguali a 1, la password viene rifiutata. In questo modo tutte le password del dizionario verranno rifiutate. Ma vi saranno anche alcuni "falsi positivi" ovvero password che non sono contenute nel dizionario ma che producono una corrispondenza nella tabella hash. Per capire come ciò accada, si consideri un meccanismo con due funzioni hash. Si supponga che le password *undertaker* e *hulkhogan* siano contenute nel dizionario mentre *xG%#jj98* no. Inoltre si supponga che:

$$H_1(\text{undertaker}) = 25 \qquad H_1(\text{hulkhogan}) = 83 \qquad H_1(\text{xG\%\#jj98=}) = 655$$
$$H_2(\text{undertaker}) = 998 \qquad H_2(\text{hulkhogan}) = 665 \qquad H_1(\text{xG\%\#jj98=}) = 998$$

Se la password xG%#jj98 viene presentata al sistema, verrà rifiutata sebbene non sia presente nel dizionario. Se vi sono troppi falsi positivi, può essere difficile per gli utenti selezionare una password. Pertanto si deve progettare un meccanismo hash che riduca i falsi positivi. Si può dimostrare che la probabilità di un falso positivo può essere approssimata da:

$$P \approx (1 - e^{kD/N})^k = (1 - e^{k/R})^k$$

o, in modo equivalente:

$$R \approx \frac{-k}{\ln(1 - P^{1/k})}$$

dove:

k = numero delle funzioni hash
N = numero dei bit della tabella hash
D = numero delle parole contenute nel dizionario
$R = N/D$, rapporto fra le dimensioni della tabella hash (in bit) e le dimensioni del dizionario (in parole)

La Figura 18.6 traccia P in funzione di R per vari valori di k. Si supponga di avere un dizionario di 1 milione di parole e di voler avere una probabilità di 0,01 di rifiuto di una password non presente nel dizionario. Se si scelgono sei funzioni hash, il rapporto richiesto sarà $R = 9{,}6$. Pertanto, occorre una tabella hash di $9{,}6 \times 10^6$ bit o circa 1,2 MB. Al contrario la memorizzazione dell'intero dizionario richiederebbe circa 8 MB di spazio. Pertanto si ottiene una compressione di un fattore più o meno uguale a 7. Inoltre il controllo delle password prevede il semplice calcolo di sei funzioni hash ed è indipendente dalle dimensioni del dizionario mentre usando l'intero dizionario occorrerebbe effettuare una ricerca completa.[2]

[2]Sia il modello di Markov che il filtro di Bloom prevedono l'impiego di tecniche probabilistiche. Nel caso del modello di Markov, vi è una piccola probabilità che qualche password del dizionario non venga individuata e una piccola probabilità che qualche password non presente nel dizionario venga rifiutata. Nel caso del filtro di Bloom vi è una piccola probabilità che qualche password non presente nel dizionario venga rifiutata. Anche in questo caso si può notare che l'impiego di una tecnica probabilistica semplifica la soluzione (vedere la Nota 1 nel Capitolo 15).

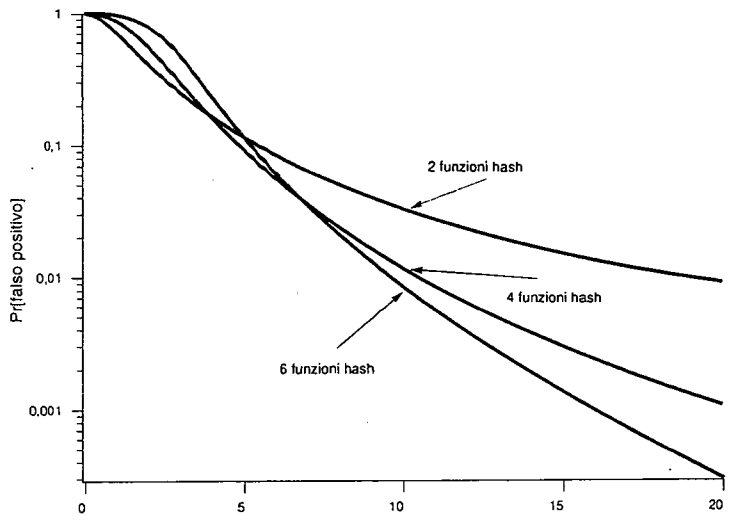


Figura 18.6 Prestazioni del filtro di Bloom.

18.4 Letture e siti Web consigliati

Due trattazioni dettagliate del rilevamento delle intrusioni sono [BACE00] e [PROC01]. Un testo più conciso ma molto interessante è [BACE01]. Due articoli brevi ma utili sull'argomento sono [KENT00] e [MCHU00]. [NING04] è una rassegna delle tecniche di intrusione più recenti. [HONE01] è una trattazione completa sui sistemi honeypot che fornisce un'analisi dettagliata degli strumenti e dei metodi impiegati dagli hacker.

BACE00 R. Bace. *Intrusion Detection*. Indianapolis, IN: Macmillan Technical Publishing, 2000.

BACE01 R. Bace e P. Mell. *Intrusion Detection Systems*. NIST Special Publication SP 800-31, Novembre 2000.

HONE01 The Honeynet Project. *Know Your Enemy: Revealing the Security Tools, Tactics, and Motives of the Blackhat Community*. Reading, MA: Addison-Wesley, 2001.

KENT00 S. Kent. "On the Trail of Intrusions into Information Systems". *IEEE Spectrum*, Dicembre 2000.

MCHU00 J. McHugh, A. Christie e J. Allen. "The Role of Intrusion Detection Systems". *IEEE Software*, Settembre/Ottobre 2000.

NING04 P. Ning *et al.* "Techniques and Tools for Analyzing Intrusion Alerts". *ACM Transactions on Information and System Security*, Maggio 2004.

PROC01 P. Proctor. *The Practical Intrusion Detection Handbook*. Upper Saddle River, NJ: Prentice Hall, 2001.

Siti Web consigliati

- **CERT Coordination Center:** è l'organizzazione sviluppatasi dal Computer Emergency Response Team costituito dall'agenzia Defense Advanced Research Projects. Il sito fornisce informazioni sui problemi di sicurezza in Internet, i punti deboli e le statistiche degli attacchi.
- **Honeypot Project:** un progetto di ricerca che studia le tecniche degli hacker e lo sviluppo di prodotti honeypot.
- **Honeypots:** un'ottima raccolta di articoli scientifici e tecnici.
- **Intrusion Detection Working Group:** include tutti i documenti generati da questo gruppo.

18.5 Termini chiave, domande di ripasso e problemi

Termini chiave

Formato di scambio per il rilevamento delle intrusioni
Hacker
Honeypot
Password
Record di auditing
Rilevamento delle intrusioni
Rilevamento delle intrusioni a regole
Rilevamento statistico delle anomalie
Tasso di errore nel rilevamento delle intrusioni
Teorema di Bayes
Valore salt

Domande di ripasso

18.1 Elencare e definire brevemente le tre classi di intrusi o hacker.
18.2 Quali sono le due tecniche più comunemente utilizzate per proteggere il file delle password?
18.3 Quali sono i tre vantaggi di un sistema di rilevamento delle intrusioni?
18.4 Qual è la differenza fra rilevamento statistico delle anomalie e rilevamento delle intrusioni basato su regole?

18.5 Quali valutazioni metriche sono utili per il rilevamento delle intrusioni a profilo?
18.6 Qual è la differenza fra rilevamento delle anomalie a regole e rilevamento delle violazioni a regole?
18.7 Cosa si intende con honeypot?
18.8 Che cos'è il valore salt nel contesto della gestione delle password in Unix?
18.9 Elencare e definire brevemente quattro tecniche utilizzate per evitare l'impiego di password facili da indovinare.

Problemi

18.1 Un taxi causa un incidente notturno e si dilegua. Nella città operano due società di taxi: Green e Blue. Si conoscono i seguenti fatti.
- L'85% dei taxi presenti in città appartiene a Green e il 15% appartiene a Blue.
- Un testimone ha identificato il taxi come appartenente a Blue.

La corte ha controllato l'affidabilità del testimone nelle stesse circostanze che si sono verificate la notte dell'incidente è ha concluso che il testimone è riuscito a identificare correttamente il colore del taxi l'80% delle volte. Qual è la probabilità che il taxi coinvolto nell'incidente sia di Blue invece che di Green?

18.2 Si supponga che le password vengano selezionate da combinazioni di 4 caratteri scelti fra i 26 caratteri alfabetici. Si supponga che un hacker sia in grado di provare le password a una velocità di una al secondo.
A. Supponendo che non esista alcun feedback per l'hacker fino al completamento di ciascun tentativo, qual è il tempo previsto per scoprire la password corretta?
B. Supponendo che esista un sistema di feedback che avverta dell'errore ogni volta che viene introdotto un carattere errato, qual è il tempo previsto per scoprire la password corretta?

18.3 Si supponga che gli elementi di origine di lunghezza k vengano mappati in modo uniforme in elementi di destinazione di lunghezza p. Se ciascuna cifra può assumere un valore su r, il numero di elementi di origine è r^k e il numero di elementi di destinazione è il numero (più piccolo) r^p. Un determinato elemento di origine x_i viene mappato su un determinato elemento di destinazione y_j.
A. Qual è la probabilità che un avversario possa selezionare in un tentativo l'elemento di origine corretto?
B. Qual è la probabilità che un avversario sia in grado di selezionare un elemento di origine differente x_k ($x_i \neq x_k$) che produce lo stesso elemento di destinazione y_j?
C. Qual è la probabilità che un avversario possa produrre l'elemento di destinazione corretto in un tentativo?

18.4 Un generatore di password fonetico seleziona casualmente due segmenti per ciascuna password di sei lettere. La forma di ciascun segmento è CVC (consonante, vocale, consonante) dove V = <a,e,i,o,u> e $C = \overline{V}$.
A. Qual è la popolazione totale delle password?
B. Qual è la probabilità che un avversario possa indovinare correttamente una password?

18.5 Si supponga che le password siano limitate ai 95 caratteri ASCII stampabili e che tutte le password abbiano una lunghezza di 10 caratteri. Si supponga che un password cracker abbia una velocità di 6,4 milioni di crittografie al secondo. Quanto tempo sarà necessario per verificare in modo esaustivo tutte le possibili password su un sistema Unix?

18.6 Dati i rischi noti del sistema di password Unix, la documentazione SunOS-4.0 consiglia di eliminare il file delle password e sostituirlo con un file pubblico chiamato /etc/publickey. La voce dell'utente A contenuta in questo file sarà costituita dall'identificatore dell'utente ID_A, dalla chiave pubblica dell'utente PU_a e dalla corrispondente chiave privata PR_a. Questa chiave privata viene crittografata utilizzando DES con una chiave che deriva dalla password di login dell'utente P_a. Quando A si collega, il sistema esegue la decrittografia di $E[P_a, PR_a]$ per ottenere PR_a.

A. Il sistema quindi verifica che sia stato fornito correttamente P_a. In quale modo?

B. In quale modo un avversario può attaccare questo sistema?

18.7 Lo schema di crittografia utilizzato per le password Unix è monodirezionale, ovvero non è possibile invertirlo. Sarebbe quindi corretto dire che in realtà si tratta di un codice hash e non di una crittografia della password?

18.8 Si è detto che l'inclusione del codice salt nello schema di password di Unix aumenta la difficoltà di indovinare la password di un fattore 4096. Ma il valore salt è conservato in chiaro nella stessa voce della password cifrata corrispondente. Pertanto questi due caratteri sono noti all'hacker e non devono essere indovinati. Perché allora si è detto che il valore salt aumenta la sicurezza?

18.9 Supponendo di avere risposto con successo al problema precedente e di aver capito quindi l'importanza del valore salt, ecco un'altra domanda. Sarebbe possibile sventare completamente qualsiasi attacco da parte dei password cracker aumentando drasticamente le dimensioni del valore salt, per esempio a 24 o 48 bit?

18.10 Si consideri il filtro di Bloom trattato nel Paragrafo 18.3. Si definisca k = numero delle funzioni hash, N = numero dei bit della tabella hash e D = numero delle parole contenute nel dizionario.

A. Dimostrare che il numero previsto di bit uguali a zero nella tabella hash può essere espresso come:

$$\phi = \left(1 - \frac{k}{N}\right)^D$$

B. Dimostrare che la probabilità che una parola di input, non presente nel dizionario, venga erroneamente accettata come appartenente al dizionario è:

$$P = (1 - \phi)^k$$

C. Dimostrare che l'espressione precedente può essere approssimata come:

$$P \approx (1 - e^{kD/N})^k$$

18.11 Progettare un sistema di accesso ai file per consentire selettivamente l'accesso in lettura e/o in scrittura a determinati utenti, a seconda delle autorizzazioni configurate nel sistema. Le istruzioni possono essere del formato seguente:

READ (F, utente A): richiesta di lettura del file F da parte dell'utente A.
WRITE (F, Utente A): richiesta di scrittura da parte dell'utente A di una copia, eventualmente modificata, del file F.

Esiste un record header per ciascun file, contenente le informazioni di autorizzazione, ovvero l'elenco degli utenti che possono leggere e/o scrivere. Il file deve essere crittografato con una chiave sconosciuta agli utenti ma conosciuta dal sistema.

Appendice 18.A Base rate fallacy: la stima dei falsi allarmi

Si partirà con un ripasso di alcuni importanti risultati della teoria delle probabilità per affrontare poi la problematica della stima dei falsi allarmi.

Probabilità condizionale e indipendenza

Spesso si vuole conoscere una probabilità condizionata da un particolare evento. L'effetto della condizione è quello di rimuovere alcune delle possibilità dallo spazio del campione. Per esempio, qual è la probabilità di ottenere una somma pari a 8 lanciando due dadi se si sa che uno dei due dadi ha prodotto un numero pari? Si può ragionare nel seguente modo. Poiché un dado è pari e la somma è pari, il secondo dado deve essere necessariamente un numero pari. Pertanto i risultati possono essere solo tre: (2, 6), (4, 4) e (6, 2), rispetto all'insieme totale delle possibilità ovvero [36 - (numero di eventi con entrambe le facce dispari)]. La probabilità risultante è 3 / 27 = 1 / 9.

Formalmente, la **probabilità condizionale** di un evento A supponendo che si sia verificato l'evento B, rappresentata come $\Pr[A|B]$, è definita come il rapporto:

$$\Pr[A \mid B] = \frac{\Pr[AB]}{\Pr[B]}$$

dove si suppone che Pr[B] sia diversa da 0.

Nell'esempio, A = {somma è 8} e B = {almeno un dado pari}. La quantità $\Pr[AB]$ considera tutti i risultati in cui la somma è 8 e almeno un dado è pari. Come si è visto, si possono prevedere tre risultati. Pertanto, $\Pr[AB] = 3 / 36 = 1 / 12$. Basta poco per convincersi che $\Pr[B] = 3 / 4$. Ora si può calcolare:

$$\Pr[A \mid B] = \frac{1/12}{3/4} = \frac{1}{9}$$

Questo risultato concorda con il ragionamento precedente.

Due eventi A e B sono detti **indipendenti** se $\Pr[AB] = \Pr[A]\Pr[B]$. Si può vedere con facilità che se A e B sono indipendenti, $\Pr[A|B] = \Pr[A]$ e $\Pr[B|A] = \Pr[B]$.

Il teorema di Bayes

Il teorema di Bayes è uno dei risultati più importanti della teoria delle probabilità. Innanzitutto occorre definire la formula della probabilità totale. Dato un insieme di eventi reciprocamente esclusivi $E_1, E_2, \ldots E_n$, tali che l'unione di questi eventi copra tutti i possibili risultati e dato un evento arbitrario A si può dimostrare che:

$$\Pr[A] = \sum_{i=1}^{n} \Pr[A \mid E_i]\Pr[E_i] \tag{18.1}$$

Il teorema di Bayes può essere formulato nel seguente modo:

$$\Pr[E_i \mid A] = \frac{\Pr[A \mid E_i]P[E_i]}{\Pr[A]} = \frac{\Pr[A \mid E_i]P[E_i]}{\sum_{j=1}^{n} \Pr[A \mid E_j]\Pr[E_j]} \tag{18.2}$$

La Figura 18.7A illustra i concetti della probabilità totale e del teorema di Bayes.

Il teorema di Bayes viene utilizzato per calcolare la probabilità che si sia verificato veramente un determinato fatto date delle prove a favore. Per esempio, si supponga di dover trasmettere una sequenza di "0" e "1" su una linea di trasmissione rumorosa. S0 e S1 sono rispettivamente i casi che sia stato inviato uno 0 o un 1 in un determinato momento e R0 e R1 sono rispettivamente i casi che venga ricevuto uno 0 o un 1. Si supponga di conoscere le probabilità della sorgente, ovvero $\Pr[S1] = p$ e $\Pr[S0] = 1 - p$. Ora viene osservata la linea di comunicazione per determinare con quale frequenza si verifica un errore quando viene inviato un 1 e quando viene inviato uno 0 e vengono calcolate le seguenti probabilità: $\Pr[R0|S1] = p_a$ e $\Pr[R1|S0] = p_b$. Se viene ricevuto il valore "0", si può utilizzare il teorema di Bayes per calcolare la probabilità condizionale di un errore, ovvero

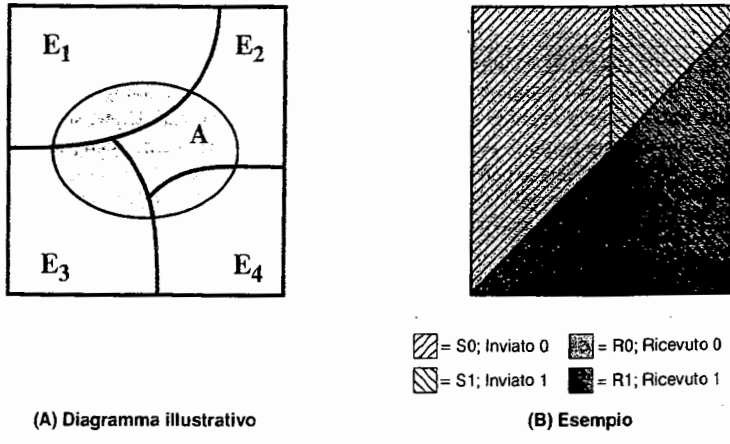


Figura 18.7 Rappresentazione della probabilità totale e del teorema di Bayes.

la probabilità condizionale che sia stato inviato un "1" supponendo che sia stato ricevuto uno "0".

$$\Pr[S1 \mid R0] = \frac{\Pr[R0 \mid S1]\Pr[S1]}{\Pr[R0 \mid S1]\Pr[S1] + \Pr[R0 \mid S0]\Pr[S0]} = \frac{p_a p}{p_a p + (1 - p_b)(1 - p)}$$

La Figura 18.7B illustra l'equazione precedente. Nella figura, lo spazio dei campioni è rappresentato da un quadrato unitario. Metà del quadrato corrisponde a S0 e metà a S1 e dunque $\Pr[S0] = \Pr[S1] = 0{,}5$. Analogamente, metà del quadrato corrisponde a R0 e metà a R1 e dunque $\Pr[R0] = \Pr[R1] = 0{,}5$. All'interno dell'area che rappresenta S0, un quarto dell'area corrisponde a R1 e dunque $\Pr[R1|S0] = 0{,}25$. Le altre probabilità condizionali sono analoghe.

La base rate fallacy

Si consideri la seguente situazione. Un paziente effettua un'analisi clinica per una determinata malattia, che dà esito positivo (ovvero indica che il paziente è affetto dalla malattia). Vengono comunicate le seguenti informazioni.

- La precisione del test è dell'87% (ovvero se un paziente è malato, nell'87% dei casi l'analisi fornisce il risultato corretto e se il paziente è sano, nell'87% dei casi l'analisi fornisce il risultato corretto).
- L'incidenza della malattia nella popolazione è dell'1%.

Posto che il test sia positivo, qual è la probabilità che il paziente non sia affetto dalla malattia? In pratica qual è la probabilità che questo sia un falso allarme? Per ottenere la risposta corretta occorre ricorrere al teorema di Bayes:

$$\Pr[\text{sano} \mid \text{positivo}] = \frac{\Pr[\text{positivo} \mid \text{sano}]\Pr[\text{sano}]}{\Pr[\text{positivo} \mid \text{malato}]\Pr[\text{malato}] + \Pr[\text{positivo} \mid \text{sano}]\Pr[\text{sano}]}$$
$$= \frac{(0{,}13)(0{,}99)}{(0{,}87)(0{,}01) + (0{,}13)(0{,}99)} = 0{,}937$$

Pertanto, nella maggior parte dei casi, quando il test è positivo, si tratta di un falso allarme. Questo problema, utilizzato in uno studio [PIAT91], venne sottoposto a un certo numero di persone. La maggior parte dei soggetti diede la risposta 13%. La grande maggioranza, fra cui molti medici, diede risposte inferiori al 50%. Molti medici che avevano dato una risposta scorretta, si lamentarono che se così stessero le cose, eseguire analisi cliniche non sarebbe di alcuna utilità. La ragione per cui la maggior parte delle persone dà una risposta scorretta è che non tiene in considerazione il tasso di incidenza nella popolazione e risolve il problema in modo intuitivo. Si tratta quindi di un *errore di valutazione* noto come "base-rate fallacy".

Come può essere risolto questo problema? Si supponga di poter ottenere valutazioni corrette nel 99,9% dei casi. Ovvero si supponga di avere $\Pr[\text{positivo|malato}] = 0{,}999$ e $\Pr[\text{negativo|sano}] = 0{,}999$. Inserendo questi valori nell'Equazione 18.2, si ottiene $\Pr[\text{sano|positivo}] = 0{,}09$. Pertanto, se si può rilevare con precisione la presenza e l'assenza

della malattia a livello del 99,9%, il livello dei falsi allarmi sarà solo dell'9%. La situazione è migliorata, ma non è ancora ottimale. Inoltre si supponga ora che l'incidenza della malattia nella popolazione sia di solo 1 / 10 000 = 0,0001, sempre ipotizzando un'accuratezza del 99,9%. Questo porta la percentuale dei falsi allarmi al 91%. Analizzando situazioni realistiche, [AXEL00] ha concluso che le probabilità assoicate ai sistemi di rilevamento delle intrusioni erano tali da rendere il tasso dei falsi allarmi insoddisfacente.

Capitolo 19

Software doloso

Concetti essenziali

- Per **software doloso** si intende un software introdotto intenzionalmente in un sistema con l'obiettivo di creare danni.
- Un **virus** è un software che può £contagiare" altri programmi modificandoli. La modifica include una copia del virus stesso, che può quindi procedere nel contagiare altri programmi.
- Un **worm** è un programma che si può replicare inviando copie di se stesso da un computer all'altro tramite le connessioni di rete. Al suo arrivo, il worm può essere attivato per replicarsi e propagarsi ulteriormente. Il worm, oltre a propagarsi, esegue solitamente operazioni indesiderate.
- Un attacco **DoS** (Denial of Service) è un tentativo di impedire ai legittimi utenti di utilizzare un servizio.
- Un attacco **DDoS** (DoS distribuito) è un attacco DoS sferrato da più sorgenti opportunamente coordinate.

Questo capitolo affronta il problema del software doloso, in particolare i virus e i worm.

19.1 I virus e altre minacce correlate

Le minacce più sofisticate contro i computer si presentano sotto forma di programmi che sfruttano i punti deboli dei sistemi. In questo contesto si considerano sia i programmi applicativi che i programmi di servizio come per esempio gli editor e i compilatori.

Per iniziare verrà offerta una panoramica di queste minacce software. La parte rimanente di questo paragrafo è dedicata ai virus e ai worm.

Programmi dolosi

La terminologia relativa a questi argomenti non è sempre coerente, perché non esiste ancora un accordo universale sui vocaboli e perché vi sono sovrapposizioni fra le varie categorie. La Tabella 19.1, sostanzialmente basata su [SZOR05], costituisce un'utile guida. Il software doloso può essere suddiviso in due categorie: quello che richiede la presenza di un programma ospite e quello che opera in modo indipendente. Il primo tipo è costituito fondamentalmente da frammenti di codice che non potrebbero funzionare se non in un programma applicativo, un programma di servizio o un programma di sistema. Per esempio virus, bombe logiche e backdoor. Il secondo consiste in programmi completi che vengono eseguiti dal sistema operativo. Per esempio worm e zombie.

Si può distinguere anche fra software che si replicano e quelli che non si replicano. I primi sono costituiti da frammenti di programmi o programmi indipendenti che vengono

Tabella 19.1 Terminologia relativa al software doloso.

Nome	Descrizione
Virus	Contagia un programma e si propaga ad altri programmi.
Verme (worm)	Programma che si propaga ad altri computer.
Bomba logica	Scatena particolari azioni al verificarsi di determinate condizioni.
Cavallo di Troia	Programma che contiene funzionalità nascoste.
Backdoor (o trapdoor)	Modifica o un programma per consentire accessi non autorizzati.
Exploit	Codice specifico a una porticolare lacuna di sicurezza o o un insieme di lacune.
Downloader	Programma che installa software doloso nella macchina attaccota. Viene solitamente inviato per posto elettronico.
Auto-rooter	Strumento utilizzato dagli hacker per ottenere illecitamente l'accesso remoto ad altre macchine.
Kit (generatore di virus)	Insieme di strumenti per la generazione automotico di nuovi virus.
Programma spammer	Utilizzato per inviare ingenti volumi di messaggi di posta elettronica non sollecitati.
Flooder	Utilizzato per inondare computer in rete con enormi volumi di traffico in modo da impedire loro lo svolgimento dei compiti previsti (attacchi DoS).
Keylogger	Catturano la sequenza di carotteri introdotti da tastiera nei sistemi contagiati.
Rootkit	Insieme di strumenti per hacker da utilizzare dopo essersi illecitomente introdotti in un computer con i privilegi di root.
Zombie	Programma attivato su una macchino contagiota, per sferrare attocchi ad altre macchine.

attivati da una condizione. I secondi sono frammenti di codice che, quando eseguiti, possono produrre più copie di se stessi da attivare successivamente, sullo stesso sistema o in altri sistemi. Di seguito si descriveranno brevemente alcune delle principali categorie di software doloso eccetto i virus e i worm che verranno trattati in dettaglio più avanti.

Backdoor

Una backdoor (porta di servizio), detta anche trapdoor, è un punto di accesso segreto in un programma, sfruttabile da chi lo conosca per acquisire l'accesso al sistema evitando le normali procedure di sicurezza di accesso. Le backdoor sono utilizzate legittimamente da molti anni dai programmatori per eseguire il debug e la verifica dei programmi. Ciò avviene normalmente quando i programmatori sviluppano un'applicazione che contiene complesse procedure di autenticazione o una lunga fase di configurazione che richiede l'introduzione di una grande quantità di valori. Per eseguire il debug del programma, lo sviluppatore potrebbe voler acquisire particolari privilegi o evitare le noiose operazioni di configurazione e autenticazione necessarie. Il programmatore potrebbe anche volersi assicurare che esista un metodo per controllare il programma nel caso ci fossero problemi nella procedura di autenticazione contenuta nell'applicazione. La backdoor riconosce una determinata sequenza di input o viene attivata quando eseguita da un particolare codice utente o solo con una sequenza molto improbabile di eventi.

Le backdoor diventano un problema quando vengono utilizzate per acquisire un accesso non autorizzato. La backdoor era il punto debole del computer di cui si parla nel film *War Games*. Ecco un altro esempio: durante lo sviluppo di Multics, vennero condotti dei test di penetrazione da parte di un "tiger team" dell'USAF che simulava gli avversari. Una tattica impiegata consistette nell'inviare un aggiornamento fasullo al sistema operativo dal sito sul quale era in esecuzione Multics. L'aggiornamento conteneva un cavallo di Troia (se ne parlerà più avanti) che poteva essere attivato da una backdoor e che consentiva al tiger team di acquisire l'accesso al sistema. L'attacco fu così ben congegnato che gli sviluppatori di Multics non riuscirono a trovarlo, neppure dopo essere stati informati della sua presenza [ENGE80].

È difficile implementare i controlli del sistema operativo per le backdoor. Le misure di sicurezza devono concentrarsi sulle attività di sviluppo del programma e sugli aggiornamenti del software.

Bombe logiche

Una delle minacce meno recenti, rispetto ai virus e ai worm, è quella delle bombe logiche. La bomba logica è costituita da codice incluso in un programma legittimo; la bomba è configurata per "esplodere" quando si verificano determinate condizioni. Per esempio la bomba logica può scattare in presenza (o assenza) di determinati file, in un particolare giorno della settimana o in una certa data o quando un determinato utente esegue l'applicazione. Una volta scattata, la bomba logica può modificare o cancellare i dati o interi file, provocare il blocco della macchina o svolgere altre operazioni dannose. Un esempio sor-

prendente dell'impiego delle bombe logiche è il caso di Tim Lloyd che venne condannato per aver preparato una bomba logica che costò alla sua azienda, Omega Engineering, più di 10 milioni di dollari, ne pregiudicò la strategia di crescita e portò al licenziamento di 80 lavoratori [GAUD00]. Alla fine, Lloyd venne condannato a 41 mesi di prigione e a un risarcimento di 2 milioni di dollari.

Cavalli di Troia

Un cavallo di Troia è un programma o una procedura utile o apparentemente utile contenente codice nascosto che, quando viene richiamato, svolge alcune operazioni indesiderate o dannose.

I cavalli di Troia possono essere utilizzati per svolgere indirettamente delle funzioni che un utente non autorizzato non potrebbe svolgere direttamente. Per esempio, per acquisire l'accesso ai file di un altro utente di un sistema condiviso, un utente potrebbe creare un cavallo di Troia che cambi i permessi di accesso ai file dell'utente che lo esegue, consentendone la lettura a ogni utente. L'autore potrebbe indurre gli utenti a eseguire il programma inserendolo in una directory comune e assegnandogli un nome tale da farlo sembrare un programma utility. Dopo che un altro utente avrà eseguito il programma, l'autore potrà accedere alle informazioni contenute nei suoi file. Un esempio di cavallo di Troia difficile da rilevare è un compilatore modificato in modo da inserire del codice di attacco nei programmi al momento della compilazione, per esempio un programma di login al sistema [THOM84]. Il codice crea una backdoor nel programma di login che consente all'autore di connettersi al sistema utilizzando una password speciale. Questo cavallo di Troia non potrebbe essere individuabile semplicemente leggendo il codice sorgente del programma di login.

Un'altra motivazione comune che spinge alla realizzazione di cavalli di Troia è la distruzione dei dati. Il programma sembra svolgere una funzione utile (per esempio potrebbe trattarsi di una calcolatrice) mentre, dietro le quinte, cancella i file dell'utente. Per esempio, un dirigente di CBS venne colpito da un cavallo di Troia che distrusse tutte le informazioni contenute nel suo computer [TIME90]. Il cavallo di Troia faceva parte di una routine grafica prelevata da un servizio BBS.

Zombie

Uno zombie è un programma che assume segretamente il controllo di un altro computer connesso a Internet utilizzandolo poi per sferrare attacchi che non consentano di risalire al vero responsabile. Gli zombie vengono utilizzati per svolgere attacchi denial-of-service, normalmente contro determinati siti Web. Gli zombie vengono impiantati in centinaia di computer che appartengono a utenti ignari e poi vengono utilizzati per sommergere i siti Web con un'enorme quantità di traffico Internet. Nel paragrafo 19.3 si discutono gli zombie nel contesto degli attacchi DoS.

La natura dei virus

Un virus è un programma che può infettare altri programmi modificandoli; tra le modifiche vi è la copia dello stesso virus, che può pertanto procedere a infettare altri programmi.

I virus biologici sono piccoli frammenti di codice genetico, DNA o RNA, che possono assumere il controllo delle cellule viventi e indurle a creare migliaia di copie del virus. Analogamente al virus biologico, un virus per computer trasporta del codice necessario per eseguire delle copie di se stesso. Il virus contagia solitamente un programma di un computer. Appena il computer infetto entra in contatto con un software non infetto, il virus infetta questo nuovo programma. Pertanto l'infezione può diffondersi da computer a computer grazie al fatto che gli utenti ignari si scambiano dischi o programmi attraverso la rete. In un ambiente di rete, la possibilità di accedere ad applicazioni presenti in altri computer rappresenta il "terreno di coltura" ideale per la diffusione di virus.

Un virus può fare qualsiasi cosa. L'unica differenza rispetto ai normali programmi è il fatto che infetta gli altri programmi e che viene eseguito segretamente con il programma che lo ospita. Quando il virus è in esecuzione può svolgere qualsiasi funzione, anche cancellare file e programmi.

Il tipico ciclo di vita di un virus prevede quattro fasi.

- **Fase dormiente:** il virus è inattivo. Il virus verrà attivato da un evento, per esempio una data, la presenza di un altro programma o file o il superamento di una soglia di capacità del disco. Non tutti i virus hanno questa fase.
- **Fase di propagazione:** il virus inserisce una copia di se stesso in altri programmi o in determinate aree di sistema sui dischi. Ogni programma infetto conterrà pertanto un clone del virus che a sua volta entrerà in fase di propagazione.
- **Fase di attivazione:** il virus viene attivato per svolgere la funzione prestabilita. Come nella fase dormiente, la fase di attivazione può essere scatenata da vari eventi, per esempio dopo che il virus ha eseguito un determinato numero di copie di se stesso.
- **Fase di esecuzione:** viene eseguita la funzione del virus. Questa può essere innocua, per esempio la visualizzazione di un messaggio sullo schermo, o provocare danni, come la distruzione di programmi e file di dati.

La maggior parte dei virus svolge il proprio lavoro su un determinato sistema operativo e, in alcuni casi, su una determinata piattaforma hardware. Pertanto i virus sono progettati per sfruttare i punti deboli di sistemi specifici.

La struttura dei virus

Un virus può essere inserito in qualsiasi punto di un programma eseguibile. Questa operazione fa in modo che il programma, una volta richiamato, esegua innanzitutto il codice del virus e poi il codice originario del programma.

La Figura 19.1 (basata su [COHE94]) rappresenta la struttura generale di un virus. In questo caso, il codice virale V viene fatto precedere al programma infettato e si presuppone che il punto di ingresso del programma sia la prima riga.

Un programma infetto inizia con il codice virale e funziona nel seguente modo. La prima riga di codice è un salto al programma principale del virus. La seconda riga è un contrasse-

```
program V :=

{goto main;
   1234567;

        subroutine infetta-eseguibile :=
              {loop:
              file := scegli-file-eseguibile-a-caso;
              if (prima-riga-del-file = 1234567)
                    then goto loop
                    else aggiungi-V-all'inizio-del-file; }

        subroutine danneggia :=
              {esegui-il-danno}

        subroutine controlla-condizione-trigger :=
              {restituisci-true-se-vale-la-condizione}

main:         programma-principale :=
              {infetta-eseguibile;
              if controlla-condizione-trigger then danneggia;
              goto next;}
next:
}
```

Figura 19.1 Un semplice virus.

gno particolare che viene utilizzato dal virus per determinare se la potenziale vittima (un programma) è già stata infettata o meno dal virus. Quando viene richiamato il programma, il controllo passa immediatamente al virus. Questo controlla innanzitutto se vi sono file eseguibili non infetti, nel qual caso li infetta. Poi il virus può svolgere alcune operazioni che normalmente danneggiano il sistema. Queste azioni potrebbero essere esguite ogni volta che viene richiamato il programma o potrebbero essere scatenate solo in determinate condizioni. Infine il virus trasferisce il controllo al programma originario. Se la fase di infezione del programma è ragionevolmente rapida, un utente non noterà alcuna differenza con l'esecuzione del programma non infetto.

Un virus come quello appena descritto può essere rilevato con facilità poiché la versione infetta di un programma è più lunga della corrispondente versione non infetta. Un modo per aggirare questa semplice tecnica per individuare il virus consiste nel comprimere il file eseguibile in modo che la versione infettata abbia le stesse dimensioni della versione originaria. La Figura 19.2, tratta da [COHE94], mostra in termini generali le operazioni svolte. Le righe principali del codice di questo virus sono numerate e la Figura 19.3, sempre tratta da [COHE94], ne illustra il funzionamento. Si suppone che il programma P_1 sia infettato dal virus CV. Quando P_1 viene richiamato, il controllo passa al virus che svolge le seguenti operazioni.


```
program CV :=

{goto main;
   01234567;

        subroutine infetta-eseguibile :=
                    {loop:
                          file := scegli-file-eseguibile-a-caso;
                    if (prima-riga-del-file = 01234567) then goto loop;
              (1) comprimi-file;
              (2) aggiungi-CV-all'inizio-del-file;
              }

main: programma-principale :=
              {if chiedi-permesso then infetta-eseguibile;
              (3) espandi-il-resto-del-file;
              (4) esegui-file-non-compresso;}
              }
```

Figura 19.2 Logica di funzionamento di un virus a compressione.

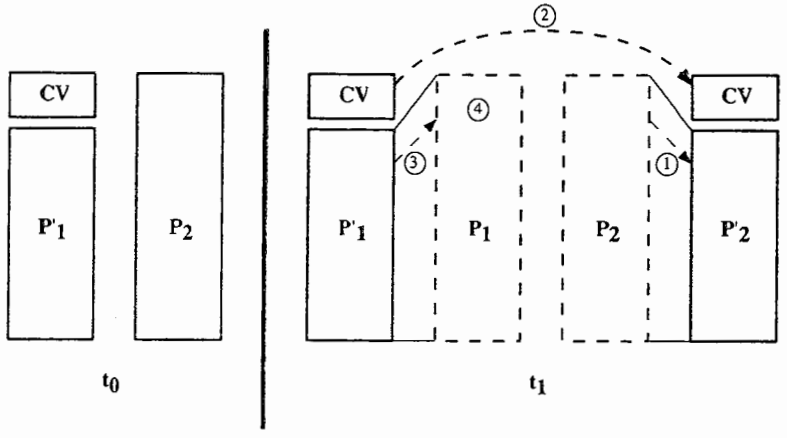


Figura 19.3 Un virus a compressione.

1. Per ogni file non infetto P_2 trovato, il virus esegue innanzitutto la compressione del file per produrre P'_2 che è più breve rispetto al programma originario esattamente delle dimensioni del virus.
2. Al programma compresso viene fatta precedere una copia del virus.
3. La versione compressa del programma originale infettato P'_1 viene espansa.

4. Viene eseguito il programma originale espanso.

In questo esempio, il virus non fa altro che propagarsi. Come nell'esempio precedente, il virus può includere una bomba logica.

Infezione iniziale

Una volta acquisito l'accesso a un sistema infettando un programma, un virus può infettare alcuni o addirittura tutti i file eseguibili di tale sistema ogni volta che viene eseguito il programma infetto. Pertanto l'infezione virale può essere impedita evitando che il virus possa accedere al sistema. Sfortunatamente la prevenzione è molto difficile poiché un virus può essere presente in qualsiasi programma proveniente dall'esterno del sistema. Pertanto, ogni volta che si acquisisce nuovo software si corre un rischio di infezione.

I vari tipi di virus

Da quando sono comparsi i virus, è stata una continua e aspra battaglia fra i realizzatori di virus e i realizzatori di software antivirus. A mano a mano che venivano sviluppate contromisure contro i tipi di virus esistenti, venivano sviluppati nuovi tipi di virus. [STEP93] suggerisce le seguenti categorie per definire i vari tipi di virus.

- **Virus parassiti:** il tipo tradizionale e tuttora più comune di virus. Un virus parassita si allega ai file eseguibili e si replica quando viene eseguito il programma infetto, trovando altri file eseguibili da infettare.
- **Virus residenti in memoria:** si localizzano nella memoria principale nell'ambito di un programma residente. Da questo momento in poi, il virus infetterà ogni programma che verrà eseguito.
- **Virus per il settore di boot:** infettano il record di avvio principale del sistema e dunque vengono eseguiti ogni volta che il sistema viene avviato da un disco contenente il virus.
- **Virus stealth (invisibili):** una forma di virus progettata appositamente per nascondersi, in modo da non poter essere rilevato dal software anti-virus.
- **Virus polimorfici:** virus che mutano a ogni infezione, complicando l'individuazione della loro "signature".
- **Virus metamorfici:** come i virus polimorfici, i virus metamorfici mutano a ogni infezione. Questi ultimi, tuttavia, si riscrivono completamente ad ogni iterazione, aumentando la difficoltà di identificazione. I virus metamorfici possono cambiare il loro comportamento oltre che il loro aspetto.

Un esempio di **virus invisibile** è quello descritto in precedenza: un virus che utilizza la compressione in modo che il programma infettato abbia esattamente le stesse dimensioni della versione non infettata. Naturalmente è possibile utilizzare tecniche molto più sofisticate. Per esempio un virus può inserire una logica di intercettazione nelle routine di I/O su disco in modo che quando vi sarà un tentativo di leggere le porzioni sospette del disco, il virus presenterà la versione originale, non infetta, del programma. Pertanto il virus non è veramente "*invisibile*" ma piuttosto adotta tecniche che gli consentono di sfuggire al rilevamento.

Un **virus polimorfico** si replica generando copie funzionalmente equivalenti ma che presentano sequenze di bit significativamente differenti. Come nel caso dei virus invisibili, lo scopo è quello di evitare la rilevazione da parte dei programmi antivirus. In questo caso la "signature" del virus cambia a ogni copia. Per ottenere questa variazione, i virus possono inserire casualmente delle istruzioni superflue o scambiare l'ordine di istruzioni indipendenti. Un approccio più efficace prevede l'impiego della crittografia. Una parte del virus, generalmente chiamata *motore di mutazione*, crea una chiave di crittografia casuale per crittografare la parte rimanente del virus. La chiave è conservata nel virus e lo stesso motore di mutazione viene modificato. Quando viene richiamato il programma infetto, il virus utilizza la chiave di crittografia per decrittografare la parte di virus crittografato. Quando il virus si replica, sceglie una chiave casuale differente.

Un altro strumento a disposizione dei realizzatori di virus è il toolkit di creazione di virus. Tale toolkit consente a un programmatore relativamente inesperto di creare con rapidità vari virus differenti. Anche se i virus creati con questi toolkit sono in genere poco sofisticati, il gran numero di nuovi virus generati crea problemi per i meccanismi antivirus.

Virus a macro

Nella metà degli anni Novanta, i virus a macro sono diventati di gran lunga i virus più diffusi. I virus a macro sono particolarmente pericolosi per tre motivi.

1. Un virus a macro è indipendente dalla piattaforma. Praticamente tutti i virus a macro infettano i documenti Word. Questo significa che può essere infettata qualsiasi piattaforma hardware e qualsiasi sistema operativo che supporti il programma Microsoft Word.
2. I virus a macro infettano i documenti e non le porzioni di codice eseguibile. La maggior parte delle informazioni introdotte nei computer è costituita da documenti e non da programmi.
3. I virus a macro si diffondono con facilità, per esempio tramite messaggi di posta elettronica.

I virus a macro sfruttano il meccanismo delle macro di Microsoft Word e di altre applicazioni come Microsoft Excel. In pratica una macro è un programma eseguibile incluso in un documento. In genere gli utenti impiegano le macro per svolgere operazioni ripetitive senza troppa fatica. Il linguaggio per macro è in genere una versione del linguaggio di programmazione Basic. È possibile definire una combinazione di tasti in modo da richiamare una macro e associare alla macro specifici comandi.

Le versioni più recenti di Word offrono una maggiore protezione contro i virus a macro. Per esempio, Microsoft offre lo strumento opzionale Macro Virus Protection in grado di rilevare i file Word sospetti e avvertire gli utenti dei potenziali rischi derivanti dall'apertura di un file contenente macro. Vari produttori di software antivirus hanno sviluppato degli strumenti in grado di rilevare ed eliminare i virus a macro. Come per altri tipi di virus, vi è una lotta serrata fra gli sviluppatori di virus a macro e quelli di antivirus; questi virus non costituiscono comunque più la minaccia predominante.

Virus di posta elettronica

Uno sviluppo più recente è costituito dai virus di posta elettronica. I primi virus di posta elettronica a rapida diffusione, come Melissa, facevano uso di una macro di Word incorporata in un allegato. Se il destinatario apriva l'allegato di posta elettronica, veniva attivata la macro di Word che svolgeva le seguenti operazioni.

1. Inviava il virus di posta elettronica a tutti gli indirizzi presenti nella rubrica dell'utente attaccato.
2. Eseguiva dei danni locali.

Alla fine del 1999, è comparsa una versione più potente del virus di posta elettronica. Questa versione può essere attivata semplicemente aprendo un messaggio di posta elettronica che contiene il virus e senza aprire l'allegato. Il virus utilizza il linguaggio Visual Basic, supportato dai pacchetti di posta elettronica.

Pertanto vi è tutta una nuova generazione di software doloso in grado di diffondersi tramite posta elettronica e di replicarsi via Internet utilizzando le funzionalità del pacchetto di posta elettronica. Il virus si propaga non appena viene attivato (tramite l'apertura dell'allegato di posta elettronica o l'apertura del messaggio) e viene inviato a tutti gli indirizzi di posta elettronica noti sull'host infetto. Pertanto, mentre i normali virus impiegavano mesi o anni per propagarsi, ai virus di posta elettronica bastano poche ore. Questo rende più complicata la reazione da parte del software antivirus prima che i danni subiti siano già gravissimi. Alla fine, per sventare la crescente minaccia, l'unica possibilità consiste nel migliorare la sicurezza dei programmi di servizio e del software applicativo per Internet.

I worm

Un worm è un programma che si può replicare da un computer all'altro tramite le connessioni di rete. All'arrivo, il worm può essere attivato per replicarsi e propagarsi ulteriormente. Il worm, oltre a propagarsi, esegue solitamente operazioni indesiderate. Un virus di posta elettronica ha alcune delle caratteristiche di un worm in quanto si propaga da sistema a sistema. Tuttavia viene classificato come virus poiché richiede un intervento umano per propagarsi. Un worm invece ricerca continuamente nuove macchine da infettare e ogni nuova macchina infettata funge da trampolino di lancio automatico per attaccare altre macchine. I worm di rete si diffondono da sistema a sistema utilizzando le connessioni di rete. Una volta attivo in un sistema, un worm di rete può comportarsi come un virus o un batterio o impiantare dei cavalli di Troia o svolgere varie altre attività dannose o distruttive.

Per replicarsi, un worm di rete utilizza un meccanismo di rete. Ecco alcuni esempi.

- **Sistema di posta elettronica:** il worm invia per posta elettronica una copia di se stesso ad altri sistemi.
- **Funzionalità di esecuzione remota:** il worm esegue una copia di se stesso su un altro sistema.
- **Funzionalità di login remoto:** il worm si connette come utente remoto di un sistema per poi copiarsi da un sistema all'altro tramite specifici comandi.

La nuova copia del worm viene quindi eseguita sul sistema remoto da cui può propagarsi ad altri sistemi.

Un worm di rete esibisce le stesse caratteristiche di un virus: una fase dormiente, una fase di propagazione, una fase di attivazione e una fase di esecuzione. Nella fase di propagazione svolge solamente le seguenti operazioni.

1. Ricerca altri sistemi da infettare esaminando le tabelle degli host o altri indirizzi di sistemi remoti.
2. Attiva una connessione con un sistema remoto.
3. Copia se stesso sul sistema remoto per diffondere l'epidemia.

Il worm di rete può anche tentare di determinare se un sistema è già stato infettato prima di copiarsi su tale sistema. Nei sistemi multitasking può anche celare la sua presenza assumendo il nome di un processo di sistema o comunque utilizzando un nome insospettabile per l'operatore di sistema. I worm di rete, come i virus, sono difficili da sconfiggere.

Il worm di Morris

Fino alla recente generazione di worm, il worm più noto era quello rilasciato in Internet da Robert Morris nel 1998. Il worm di Morris era stato progettato per diffondersi su sistemi Unix, utilizzando varie tecniche di propagazione. Quando il worm entrava in esecuzione, cercava innanzitutto di scoprire altri host accessibili in cui propagarsi. Svolgeva questa operazione esaminando vari elenchi e tabelle, fra cui le tabelle di sistema che dichiaravano quali altre macchine erano fidate per questo host, i file di inoltro della posta elettronica degli utenti e le tabelle dei permessi per l'accesso agli account remoti, utilizzando un programma che indicava lo stato delle connessioni di rete. Per ciascun host scoperto, il worm tentava vari metodi di accesso.

1. Tentava di connettersi con un host remoto come utente legittimo. Per questo fatto il worm tentava innanzitutto di violare il file delle password locali e poi usava le password scoperte e i corrispondenti codici utente, supponendo che un utente avrebbe utilizzato la stessa password su più sistemi. Per ottenere le password, il worm impiegava un programma di violazione delle password che effettuava i seguenti tentativi.
   A. Il nome di account di ciascun utente e alcune permutazioni del nome.
   B. Un elenco di 432 password che Morris riteneva candidati probabili.
   C. Tutte le parole contenute nella directory di sistema locale.
2. Sfruttava un bug contenuto nel protocollo finger che riporta informazioni relative a ciascun utente.
3. Sfruttava una trap door nell'opzione di debug del processo remoto che riceve e invia la posta elettronica.

Se uno di questi attacchi aveva successo, il worm otteneva un modo per comunicare con l'interprete dei comandi del sistema operativo. A questo punto inviava all'interprete un semplice programma di bootstrap, emetteva un comando per l'esecuzione di questo programma e poi si disconnetteva. Il programma di bootstrap richiamava il programma genitore e prelevava la parte rimanente del worm. A questo punto la nuova copia del worm poteva andare in esecuzione.

Recenti attacchi tramite worm

L'era contemporanea dei worm si è aperta con il worm Code Red nel luglio del 2001. Il worm Code Red sfrutta un problema di sicurezza del server Microsoft IIS (Internet Information Server) per accedere ai sistemi e diffondersi. Inoltre disattiva il controllo sui file di sistema di Windows. Il worm controlla alcuni indirizzi IP casuali per diffondersi in altri sistemi. Per un determinato periodo di tempo, il worm non fa altro che diffondersi. Poi sferra un attacco denial-of-service contro un sito Web del governo inondandolo di pacchetti provenienti da un'enorme quantità di host. A questo punto il worm sospende le proprie attività e si riattiva periodicamente. Nella seconda ondata di attacchi, il worm Code Red ha infettato circa 360 000 server in 14 ore. Oltre ai problemi provocati al server colpito, il worm Code Red può consumare enormi quantità di capacità di trasmissione Internet, disturbandone il funzionamento.

Il worm Code Red II è una variante che colpisce i server Microsoft IIS. Inoltre questo nuovo worm installa una back-door che consente a un hacker di controllare le attività sui computer colpiti.

Alla fine del 2001 è comparso un worm più versatile, chiamato Nimda, che si diffonde utilizzando vari meccanismi.

- Da client a client tramite messaggi di posta elettronica.
- Da client a client tramite condivisioni di rete aperte.
- Da server Web a client tramite la navigazione di siti Web violati.
- Da client a server Web tramite la scansione attiva e lo sfruttamento di vari punti deboli del server Microsoft IIS 4.0/5.0.
- Da client a server Web tramite la scansione delle back-door lasciate dai worm "Code Red II".

Il worm modifica i documenti Web (i file di tipo .htm, .html e .asp) e determinati file eseguibili presenti nei sistemi infettati e crea più copie di se stesso cambiando continuamente nome di file.

Agli inizi del 2003 apparve il worm SQL Slammer. Questi sfruttava una lacuna di Microsoft SQL Server nella gestione della condizione di buffer overflow. Il worm, estremamente compatto, si diffuse rapidamente contagiando il 90% dei sistemi vulnerabili nel giro di dieci minuti. Alla fine del 2003 apparve il worm Sobig.f, che sfruttava i server proxy non protetti per convertire le macchine contagiate in motori di spamming. Al picco della sua diffusione, Sobig.f era responsabile globalmente di un messaggio ogni 17 e riuscì a replicarsi più di un milione di volte nelle prime 24 ore.

Mydoom è un worm di posta elettronica apparso nel 2004. Fece seguito a un crescendo di computer contagiati da backdoor, che consentì agli hacker di ottenere l'accesso remoto a dati sensibili quali password e numeri di carte di credito. Mydoom, che si replicava fino a 1000 volte al minuto, inondò Internet con 1000 milioni di messaggi in sole 36 ore.

Stato della tecnologia dei Worm

Lo stato dell'arte relativo alla tecnologia dei Worm comprende gli aspetti seguenti:

- **Multipiattaforma:** i worm più recenti non sono confinati a Windows ma possono attaccare una varietà di piattaforme, in particolare le varietà di UNIX più diffuse.
- **Mutliexploit:** i worm più recenti penetrano nei sistemi con molteplici modalità, sfruttando lacune di sicurezza in server Web, browser, servizi di posta, servizi di condivisione di file e altre applicazioni di rete.
- **Diffusione rapidissima:** una tecnica per aumentare la velocità di diffusione di un worm consiste nella scansione preventiva di Internet per identificare gli indirizzi di rete dei computer vulnerabili.
- **Polimorfismo:** i worm, per evitare l'identificazione, raggirare i filtri e confondere le analisi in tempo reale, adottano la tecnica polimorfica dei virus. Ciascuna copia di un particolare worm utilizza nuovo codice generato dinamicamente, che utilizza istruzioni funzionalmente equivalenti, e tecniche di crittografia.
- **Metamorfismo:** oltre a cambiare il proprio aspetto, i worm metamorfici utilizzano schemi di comportamento differenti a seconda dello stadio di propagazione.
- **Mezzi di trasporto:** i worm, a motivo della possibilità di attaccare rapidamente un elevatissimo numero di macchine, sono ideali per diffondere altri strumenti di attacco distribuito, per esempio gli zombie che effettuano attacchi DoS distribuiti.
- **Exploit "giorno zero"**: per ottenere la massima sorpresa e diffusione, i worm cercano di sfruttare lacune di sicurezza sconosciute, che vengono scoperte dalla comunità della rete solo al lancio del worm.

19.2 Contromisure contro i virus

Strategie antivirus

La soluzione ideale alla minaccia rappresentata dai virus è la prevenzione: non consentire ai virus di raggiungere il sistema. Questo obiettivo è, in generale, impossibile da ottenere sebbene la prevenzione possa ridurre il numero di attacchi virali svolti con successo. La seconda migliore strategia è essere in grado di svolgere le seguenti attività.

- **Rilevamento:** una volta che si è verificata l'infezione, determinare l'accaduto e localizzare il virus.
- **Identificazione:** una volta rilevato, identificare il virus che ha infettato il programma.
- **Rimozione:** una volta identificato, rimuoverne tutte le tracce dal programma infetto e ripristinarne lo stato originario. Rimuovere il virus da tutti i sistemi infetti in modo da non diffondere il contagio.

Se i sistemi di rilevamento hanno successo ma l'identificazione o la rimozione non sono possibili, l'alternativa è quella di cancellare completamente il programma infetto e ricaricare una versione di backup "pulita".

I miglioramenti nella tecnologia dei virus e degli anti-virus procedono di pari passo. I primi virus erano frammenti di codice relativamente semplice che potevano essere identificati ed eliminati con un pacchetto software antivirus poco sofisticato. Ma i virus si sono evoluti e sia i virus che, di conseguenza, gli anti-virus, sono diventati sempre più complessi e sofisticati. [STEP93] identifica quattro generazioni di software anti-virus:

- prima generazione: semplici scanner;
- seconda generazione: scanner euristici;
- terza generazione: individuazione delle attività;
- quarta generazione: protezione completa.

Uno scanner di **prima generazione** identifica un virus in base alla sua "signature". Il virus può contenere parti variabili ma ha fondamentalmente la stessa struttura e configurazione di bit in tutte le copie. Questi scanner si limitano al rilevamento dei virus noti. Un altro tipo di scanner di prima generazione memorizza la lunghezza dei programmi e controlla le eventuali variazioni.

Uno scanner di **seconda generazione** non conta solo su una determinata signature ma utilizza regole euristiche per ricercare i segni di una probabile infezione. Una classe di questi scanner ricerca i frammenti di codice frequentemente associati ai virus. Per esempio, uno scanner può ricercare l'inizio di un ciclo di crittografia utilizzato in un virus polimorfico e scoprire la chiave di crittografia. Una volta scoperta la chiave, lo scanner potrà decrittografare il virus per identificarlo, rimuovere l'infezione e riportare in servizio il programma.

Un altro approccio di seconda generazione è il controllo dell'integrità. A ciascun programma può essere aggiunto un codice checksum. Se un virus infetta il programma senza cambiare il valore checksum, allora un controllo dell'integrità sarà in grado di individuare la modifica. Per sconfiggere un virus abbastanza sofisticato da cambiare il valore checksum quando infetta un programma, può essere utilizzata una funzione hash crittografata. La chiave di crittografia viene conservata separatamente rispetto al programma, in modo che il virus non possa generare un nuovo codice hash e poi crittografarlo. Utilizzando una funzione hash invece che un semplice valore checksum, il virus non potrà modificare il programma in modo da produrre lo stesso codice hash.

I programmi di **terza generazione** sono programmi residenti in memoria che identificano un virus in base alle sue azioni invece che alla struttura del programma infetto. Tali programmi presentano il vantaggio di non richiedere la definizione di un elenco di signature e di ricerche euristiche per un'ampia varietà di virus. Piuttosto, è sufficiente identificare l'insieme limitato di azioni che indicano un'infezione e quindi intervenire.

I prodotti di **quarta generazione** sono pacchetti costituiti da varie tecniche antivirus combinate fra loro. Per esempio vi sono le componenti scanner e di rilevamento delle attività. Inoltre, un pacchetto di questo tipo include funzionalità di controllo degli accessi che limita la capacità dei virus di penetrare in un sistema e di modificare i file per diffondere l'infezione. Lo scontro continua. I pacchetti di quarta generazione mettono in campo una strategia di difesa più ampia, allargando le funzionalità di difesa a misure di sicurezza più generali.

Tecniche antivirus avanzate

Continuano a essere sviluppati nuovi prodotti e strategie antivirus. In questa parte del capitolo verranno affrontate due delle più importanti.

La tecnologia GD

La tecnologia GD (Generic Decryption) consente al programma anti-virus di rilevare con facilità anche i più complessi virus polimorfici pur conservando una notevole velocità di scansione [NACH97]. Quando viene eseguito un file contenente un virus polimorfico, il virus deve decrittografarsi per attivarsi. Per rilevare questa struttura, i file eseguibili vengono esaminati da uno scanner GD che contiene i seguenti elementi.

- **Emulatore di CPU:** computer virtuale realizzato esclusivamente tramite software. Le istruzioni contenute nel file eseguibile vengono interpretate dall'emulatore invece di essere eseguite dal microprocessore. L'emulatore include una versione software di tutti i registri e dell'hardware del microprocessore in modo che il processore sottostante non venga interessato dai programmi interpretati dall'emulatore.
- **Scanner della signature dei virus:** modulo che esegue la scansione del codice alla ricerca delle signature dei virus noti.
- **Modulo di controllo dell'emulazione:** controlla l'esecuzione del codice.

All'inizio di ogni simulazione, l'emulatore inizia a interpretare le istruzioni contenute nel codice, una alla volta. Pertanto, se il codice include una routine di decrittografia che espande e quindi rende visibile il virus, questo codice verrà interpretato. In pratica il virus svolge il lavoro per conto del programma anti-virus auto-esponendosi. Periodicamente, il modulo di controllo interrompe l'interpretazione alla ricerca delle signature dei virus.

Durante l'interpretazione, il codice non può provocare alcun danno al computer in quanto viene interpretato in un ambiente completamente controllato.

Il problema più difficile del progetto di uno scanner GD è quello di determinare la durata di ciascun ciclo di interpretazione. In genere gli elementi di un virus vengono attivati non appena il programma inizia l'esecuzione ma non necessariamente le cose funzionano in questo modo. Più a lungo uno scanner emula un determinato programma, maggiori saranno le probabilità di individuare i virus nascosti. Tuttavia il programma anti-virus ha a disposizione un tempo limitato prima che l'utente inizi a lamentarsi.

Sistema immunitario digitale

Il sistema immunitario digitale è una strategia generale di protezione anti-virus sviluppata da IBM [KEPH97a], [KEPH97b]. La motivazione di questo sviluppo è stata la crescente minaccia di propagazione di virus via Internet. Si descrive innanzitutto questa minaccia per poi riassumere l'approccio adottato da IBM.

Nel passato, la minaccia di virus era caratterizzata da una diffusione relativamente lenta di nuovi virus e nuove mutazioni. Il software antivirus veniva tipicamente aggiornato mensilmente e questo era sufficiente per controllare il problema. Inoltre Internet giocava un ruolo relativamente marginale nella diffusione dei virus. Ma, come sostiene [CHES97], due delle principali tendenze nelle tecnologie di Internet hanno avuto negli ultimi anni un impatto crescente nella diffusione dei virus.

- **Sistemi di posta elettronica integrati:** sistemi come Lotus Notes e Microsoft Outlook, semplificano notevolmente l'invio e la ricezione di qualunque tipo di messaggio e l'utilizzo degli oggetti ricevuti.

- **"Mobilità" di programmi:** funzionalità come Java e ActiveX consentono ai programmi di trasferirsi da un sistema a un altro.

In risposta alla minaccia rappresentata da queste nuove funzionalità Internet, IBM ha sviluppato un prototipo di sistema immunitario digitale. Questo sistema espande l'utilizzo dell'emulazione di programmi trattata in precedenza e fornisce un'emulazione generale e un sistema di rilevamento dei virus. L'obiettivo di questo sistema è quello di garantire tempi di risposta rapidi in modo che i virus possano essere individuati non appena vengono introdotti. Quando un nuovo virus entra in un'azienda, il sistema immunitario digitale lo cattura automaticamente, lo analizza, aggiunge il codice di rilevamento e di protezione, rimuove il virus e poi passa le informazioni relative a tale virus ai sistemi sui quali è in esecuzione il programma IBM AntiVirus, in modo che possa essere rilevato prima che riesca a infettare altri sistemi.

La Figura 19.4 illustra le tipiche operazioni svolte dal sistema immunitario digitale.

1. Un programma di monitoring su ciascun PC usa una varietà di tecniche euristiche basate sul comportamento del sistema, sui cambiamenti sospetti ai programmi sulle signature per determinare la possibile presenza di un virus. Il programma di monitoring inoltra una copia di qualsiasi programma sospetto a una macchina amministrativa dell'azienda.
2. La macchina amministrativa esegue la crittografia del campione e la invia a una macchina centrale di analisi dei virus.
3. Questa macchina crea un ambiente in cui il programma infetto possa essere eseguito con sicurezza per essere analizzato. Le tecniche utilizzate per questo scopo comprendono l'emulazione o la creazione di un ambiente protetto all'interno del quale eseguire e monitorare il programma sospetto. La macchina di analisi dei virus produce una prescrizione per identificare e rimuovere il virus.

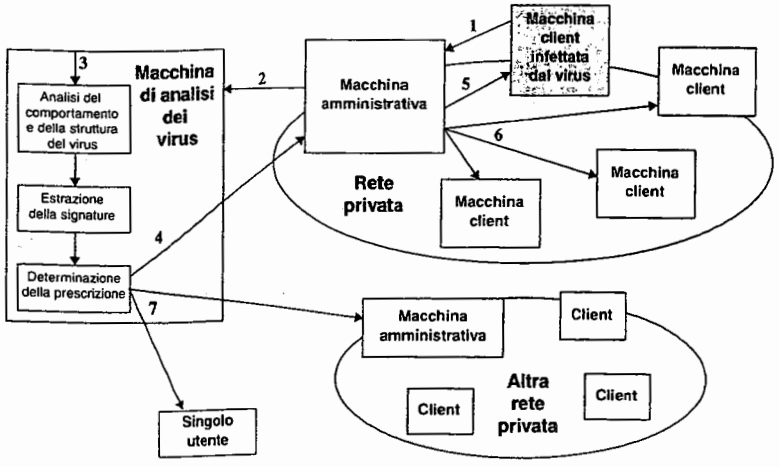


Figura 19.4 Sistema immunitario digitale.

4. La prescrizione viene inviata alla macchina amministrativa.
5. La macchina amministrativa inoltra la prescrizione al client infetto.
6. La prescrizione viene inoltrata anche agli altri client dell'azienda.
7. Gli abbonati al sistema in tutto il mondo ricevono regolari aggiornamenti anti-virus che li proteggono dal nuovo virus.

Il successo del sistema immunitario digitale dipende dalla capacità della macchina di analisi dei virus di rilevare nuovi e innovativi comportamenti dei virus. Analizzando e monitorando costantemente i virus esistenti, dovrebbe essere possibile continuare ad aggiornare il software di immunizzazione digitale per mantenerlo al passo delle minacce.

Software di bloccaggio del comportamento

A differenza degli scanner euristici o a impronte digitali, il software di bloccaggio del comportamento si integra con il sistema operativo del computer e controlla il comportamento dei programmi in tempo reale alla ricerca di azioni pericolose. Il software di bloccaggio del comportamento blocca le azioni potenzialmente pericolose prima che abbiano la possibilità di interessare il sistema. Ecco i comportamenti che possono essere monitorati

- I tentativi di aprire, visualizzare, cancellare e/o modificare i file.
- I tentativi di formattare le unità dischi e altre operazioni su disco irrecuperabili.
- Le modifiche alla logica dei file eseguibili o agli script delle macro.
- Le modifiche delle impostazioni critiche del sistema, per esempio la configurazione di avvio.
- Le richieste ai client di posta elettronica e di messaggi istantanei di inviare contenuti eseguibili.
- L'avvio di comunicazioni di rete.

Se il sistema di blocco del comportamento rileva che un programma sta avviando comportamenti ritenuti pericolosi, può bloccare questi comportamenti in tempo reale e/o chiudere il software in questione. Questo dà al sistema un vantaggio fondamentale rispetto alle tecniche di rilevamento antivirus quali i sistemi a impronte digitali o euristici. Anche se esistono miliardi di modi per offuscare e disporre le istruzioni di un virus o di un worm, molte delle quali in grado di evadere il rilevamento da parte di uno scanner della signature o euristico, alla fine il codice deve effettuare richieste ben determinate al sistema operativo. Dato che il sistema di blocco del comportamento può intercettare tutte queste richieste, può identificare e bloccare le azioni pericolose indipendentemente dal modo in cui vengano nascoste dalla logica del programma.

La capacità di osservare il software in tempo reale conferisce chiaramente un grande vantaggio, tuttavia vi sono anche dei difetti. Poiché il codice pericoloso viene in effetti eseguito sulla macchina di destinazione prima che possano essere identificati tutti i suoi comportamenti, potrebbe provocare gravi danni al sistema prima che il virus possa essere rilevato e bloccato dal sistema di blocco del comportamento. Per esempio, un nuovo virus potrebbe agire su vari file apparentemente poco importanti del disco fisso prima di infettare, un file ed essere bloccato. Anche se l'infezione venisse bloccata, l'utente potrebbe non

essere più in grado di trovare i suoi file, provocando una perdita di produttività o danni anche peggiori.

19.3 Gli attacchi DoS distribuiti

Gli attacchi DDoS (*distributed denial of service*) costituiscono per le aziende una minaccia significativa e apparentemente in continua espansione [VIJA02]. Uno studio relativo a un periodo di tre settimane nel 2001, ha consentito di rileavare più di 12000 attacchi contro oltre 5000 obiettivi, ai danni di notissime aziende di commercio elettronico come Amazon e Hotmail fino a minuscoli ISP stranieri e connessioni telefoniche [MOOR01]. Gli attacchi DDoS impediscono il normale funzionamento delle reti di computer inondando i server, le reti o addirittura le macchine degli utenti con traffico inutile, in modo che gli utenti legittimi non possano più ottenere l'accesso a tali risorse. In un tipico attacco DDoS, un elevato numero di computer viene compromesso in modo tale che questi inviino pacchetti inutili. Negli ultimi anni i metodi di attacco e gli strumenti sono divenuti più complessi, efficaci e rendono sempre più difficile l'identificazione dei veri responsabili, mentre le tecnologie di difesa non sono riuscite a far fronte ad attacchi su larga scala [CHAN02].

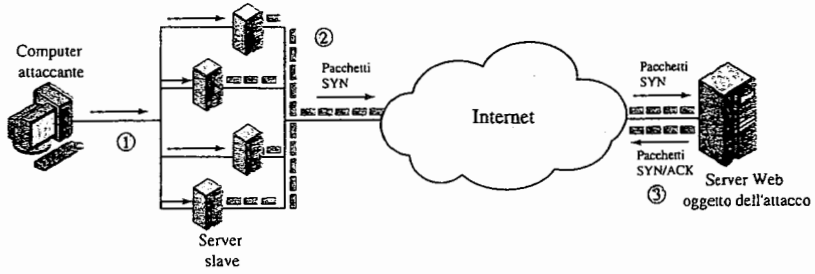
Un attacco DoS è un tentativo di impedire ai legittimi utenti di utilizzare un servizio. Quando tale attacco proviene da un singolo computer o nodo di rete, viene chiamato DoS. Un attacco DDoS costituisce una minaccia più grave: in questo caso l'hacker acquisisce preventivamente il controllo di numerosi computer in Internet per poi sferrare un attacco coordinato sull'obiettivo. Questo paragrafo analizza questo tipo di attacchi. Per prima cosa si considerano la natura e la tipologia di attacchi. Successivamente si esaminano gli strumenti con i quali l'hacker è in grado di ottenere il controllo del gruppo di computer da utilizzare nell'attacco. Infine si considerano le contromisure attualmente disponibili.

Descrizione degli attacchi DDoS

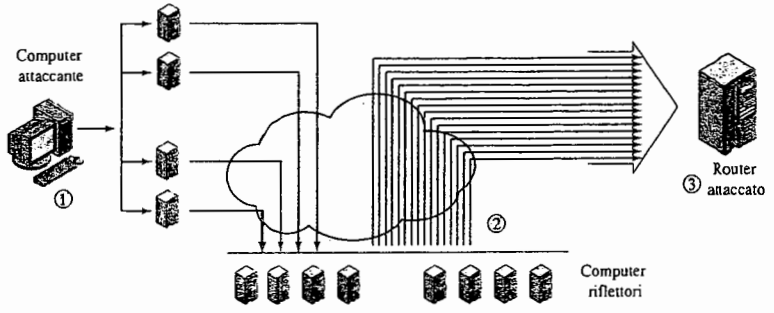
Un attacco DDoS cerca di consumare le risorse dell'obiettivo in modo che questi non sia più in grado di offrire il servizio previsto ai legittimi utenti. È possibile classificare gli attacchi DDoS in funzione del tipo di risorsa che viene consumata. Generalmente, questa consiste o in un risorsa interna di un host sul sistema attaccato, o nella capacità trasmissiva della rete locale alla quale questi è connesso.

Un semplice esempio di attacco ad una risorsa interna è l'attacco SYN flood, illustrato nella Figura 19.5.

1. L'hacker prende il controllo di vari host su Internet, programmandoli per contattare il server Web oggetto dell'attacco.
2. Questi host (*slave*, o schiavi) inviano al server pacchetti TCP/IP SYN (sincronizzazione/inizializzazione) con indirizzo IP di ritorno errato.
3. I pacchetti SYN consistono in richieste di apertura di connessioni TCP. Il server Web risponde, per ciascuno di tali pacchetti, con un corrispondente pacchetto SYN/ACK (riscontro di sincronizzazione), nel tentativo di stabilire una connessione TCP con



(a) Attacco distribuito SYN flood



(b) Attacco distribuito ICMP

Figura 19.5 Esempi di semplici attacchi DDoS.

un'entità TCP ad un indirizzo fasullo. Il server Web mantiene una struttura dati per ciascuna richiesta SYN in attesa di una risposta, e rimane quindi bloccato quando il numero di tali richieste supera le sue capacità. Come risultato, mentre la vittima è in attesa di completare l'apertura delle connessioni fasulle, le connessioni legittime non possono essere servite.

La struttura dati relativa allo stato delle connessioni rappresenta un bersaglio molto diffuso, ma non è l'unico. Gli esempi seguenti sono riportati in [CERT01]:

1. In numerosi sistemi vi è un limite al numero di strutture dati disponibili per contenere le informazioni relative ai processi (identificatori di processo, record nella tabella dei processi ecc.). L'hacker potrebbe esaurire queste strutture dati scrivendo un semplice programma o script che non fa altro che creare continuamente copie di se stesso.
2. L'hacker potrebbe anche tentare di esaurire lo spazio su disco in altri modi, tra i quali:
   - generare un numero elevatissimo di messaggi di posta elettronica;
   - generare intenzionalmente errori che devono essere tracciati nei file di log.
   - introdurre file nelle aree di ftp anonimo sulle risorse di rete condivise.

La Figura 19.5b illustra un esempio di un attacco che consuma le risorse di capacità trasmissiva.

La procedura è la seguente:

1. L'hacker ottiene il controllo di più host su Internet e li programma per inviare pacchetti ICMP ECHO[1] con l'indirizzo IP falsificato della macchina da attaccare a un gruppo di host che agiscono da riflettori, come successivamente descritto.
2. I nodi riflettori ricevono le numerose richieste con IP falsificato e reagiscono inviando pacchetti ECHO REPLY al sito da attaccare.
3. Il router del sistema attaccato viene così inondato dai riflettori con tali pacchetti, che gli impediscono di gestire il traffico legittimo.

Gli attacchi DDoS possono anche essere classificati come DDoS diretti o riflettori. In un attacco DDoS *diretto* (Figura 19.6a) l'hacker installa software zombie su un certo numero di siti Internet. Spesso l'attacco coinvolge due livelli di macchine zombie: master e slave. Gli host di entrambi sono state contagiate con software doloso; l'hacker coordina e scatena gli zombie master, i quali a loro volta coordinano e scatenano gli zombie slave. L'impiego di due livelli di zombie rende più difficile identificare la sorgente dell'attacco e costituisce una rete di attaccanti più robusta.

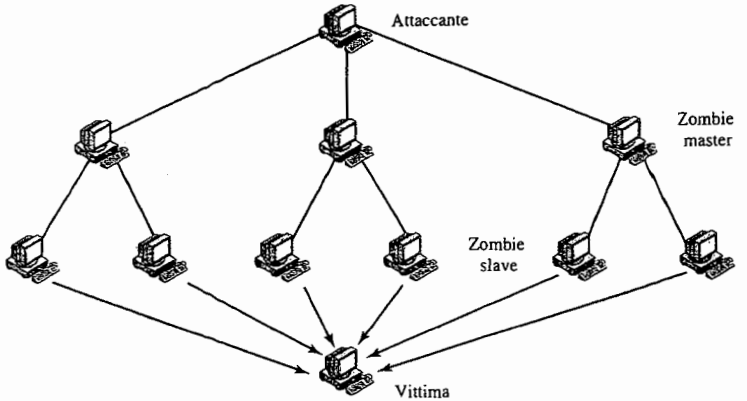
L'attacco DDoS *riflettore* aggiunge un ulteriore livello di macchine (Figura 19.6b). In questo tipo di attacco gli zombie slave creano dei pacchetti che richiedono una risposta, contenenti l'indirizzo della macchina da attaccare come indirizzo IP sorgente nell'header del pacchetto IP. Questi pacchetti vengono inviati a macchine non contagiate, chiamate riflettori. Queste rispondono con pacchetti diretti alla macchina attaccata. Un attacco DDoS riflettore può facilmente coinvolgere più macchine e generare più traffico rispetto a un attacco DDoS diretto, e può quindi risultare più nocivo. Inoltre, risalire alla sorgente dell'attacco o filtrare i pacchetti di attacco è più difficile perché questi provengono da macchine non infette disperse sulla rete.

Messa in opera della rete di attacco

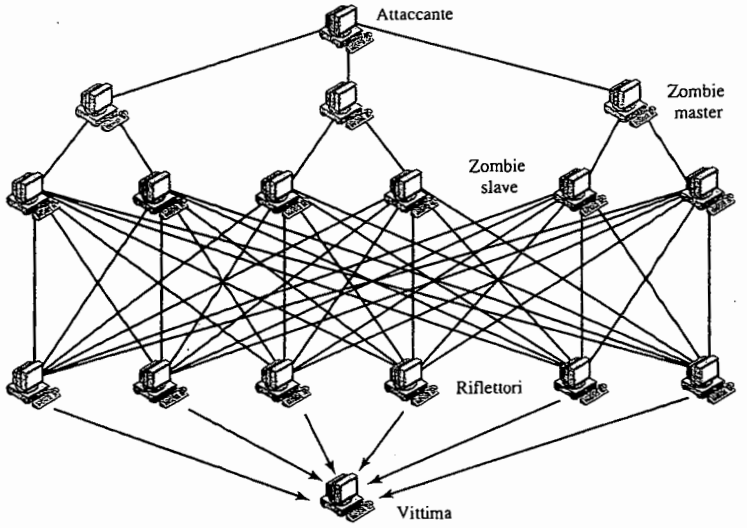
L'hacker, come primo passo di un attacco DDoS, contagia con software zombie un certo numero di macchine che verranno utilizzate per sferrare l'attacco. Le componenti essenziali in questa fase dell'attacco sono:

1. Il software che può sferrare l'attacco DDoS. Tale software deve poter essere eseguito su un grande numero di macchine, deve essere in grado di nascondersi, deve essere in grado di comunicare con l'hacker o utilizzare qualche forma di meccanismo scatenante a tempo e deve poter lanciare l'attacco contro l'obiettivo.
2. Una lacuna di sicurezza in un elevato numero di sistemi. L'hacker deve conoscere una lacuna di sicurezza che numerosi utenti e amministratori di sistema non hanno corretto e che consente all'hacker di installare software zombie.
3. Una strategia per localizzare le macchine vulnerabili, processo noto come scansione.

1 Il protocollo ICMP (Internet Control Message Protocol) è un protocollo di livello IP per lo scambio di pacchetti di controllo fra router e host o fra host. Il pacchetto ECHO richiede al ricevente di rispondere con un pacchetto ECHO REPLY per verificare la connettività fra le due macchine interessate.



(a) Attacco DDoS diretto



(b) Attacco DDoS riflettore

Figura 19.6 Tipi di attacchi DDoS basati su flooding.

Nel processo di scansione, l'hacker ricerca innanzitutto un insieme di macchine vulnerabili e le contagia. Successivamente, in genere, il software zombie installato su queste macchine ripete il medesimo processo di scansione fino a creare un'enorme rete di macchine infette. Le seguenti strategie di scansione sono riportate in [MIRK04]:

- **Casuale:** le macchine contagiate scandiscono indirizzi IP generati casualmente, utilizzando un seme differente. Questa tecnica produce un elevato traffico Internet che può arrecare seri danni ancor prima che venga lanciato l'attacco finale.
- **Hit-list** (elenco di successi): l'hacker compila innanzitutto un elenco di macchine potenzialmente vulnerabili. Questo processo può essere molto lento per evitare l'identificazione del tentativo di attacco. Una volta ottenuto l'elenco, l'hacker inizia a contagiare alcune delle macchine. Ogni macchina contagiata si occupa poi di scandire una parte delle macchine nella lista. Questo processo comporta un brevissimo tempo di scansione, che rende difficile identificare l'attacco.
- **Topologico:** questo metodo utilizza le informazioni contenute nelle macchine contagiate per identificare altre macchine da scandire.
- **Sottorete locale:** se si riesce a contagiare un host oltre il firewall, l'host può poi ricercare altre vittime nella propria rete locale. Tale host utilizza la struttura dell'indirizzo di sottorete per identificare altre macchine che sarebbero altrimenti protette dal firewall.

Contromisure agli attacchi DDoS

In generale, vi sono quattro linee di difesa contro gli attacchi DDoS [CHAN02]:

- **Prevenzione dell'attacco e prelazione (prima dell'attacco)**: questi meccanismi permettono alla vittima di sopportare un tentativo di attacco senza rifiutare il servizio agli utenti legittimi. Fra le tecniche possibili vi sono l'imposizione di limiti al consumo delle risorse e la messa a disposizione di risorse addizionali di backup sulla base delle necessità. I meccanismi di prevenzione modificano inoltre i sistemi e i protocolli Internet per ridurre le possibilità di attacchi DDoS.
- **Rilevazione dell'attacco e filtraggio (durante l'attacco)**: questa strategia cerca di rilevare gli attacchi il più presto possibile per poter reagire immediatamente, in modo da limitarne l'impatto. La rilevazione comporta la ricerca di comportamenti sospetti. La reazione consiste nell'escludere tramite filtraggio i pacchetti che possono far parte dell'attacco.
- **Tracciamento e identificazione della sorgente dell'attacco (durante e dopo l'attacco)**: in questo caso si tenta in primo luogo di identificare la sorgente dell'attacco come primo passo per prevenire altri attacchi successivi. Tuttavia, questo metodo non è solitamente in grado di produrre risultati in modo sufficientemente rapido, quando possibile, per ridurre l'impatto di un attacco in corso.

La difficoltà nel reagire agli attacchi DDoS è l'enorme numero di modi con cui possono essere perpetrati: le contromisure devono quindi evolvere con le minacce.

19.4 Letture e siti Web consigliati

Per una comprensione approfondita dei virus, si consiglia di leggere [SZR05]. Un'altra trattazione eccellente è [HARL01]. Ottimi articoli generali sui virus e i worm sono

[CASS01], [FORR97], [KEPH97] e [NACH97]. [MEIN01] fornisce una buona trattazione del worm Code Red.

[PATR04] è un'interessante rassegna di attacchi DDoS. [MIRK04] è una descrizione rigorosa dei possibili attacchi DDoS e delle relative contromisure. [CHAN02] è una buona analisi delle strategie di difesa dagli attacchi DDoS.

CASS01 S. Cass. "Anatomy of Malice". *IEEE Spectrum*, Novembre 2001.

CHAN02 R. Chang. "Defending Against Flooding-Based Distributed Denial-of-Service Attacks: A Tutorial". *IEEE Communications Magazine*, Ottobre 2002.

FORR97 S. Forrest, S Hofmeyr e A. Somayaji. "Computer Immunology". *Communications of the ACM*, Ottobre 1997.

HARL01 D. Harley, R. Slade e U. Gattiker. *Viruses Revealed*. New York: Osborne/McGraw-Hill, 2001.

KEPH97 J. Kephart, G. Sorkin, D. Chess e S. White. "Fighting Computer Viruses". *Scientific American*, Novembre 1997.

MEIN01 C. Meinel. "Code Red for the Web". *Scientific American*, Ottobre 2001.

MIRK04 J. Mirkovic e P. Relher. "A Taxonomy of DDoS Attack and DDoS Defense Mechanisms". *ACM SIGCOMM Computer Communications Review*, Aprile 2004.

NACH97 C. Nachenberg. "Computer Virus-Antivirus Coevolution". *Communications of the ACM*, Gennaio 1997.

PATR04 C. Patrikakis, M. Masikos e O. Zouraraki. "Distributed Denial of Service Attacks". *The Internet Protocol Journal*, Dicembre 2004.

SZOR05 P. Szor. *The Art of Computer Virus Research and Defense*. Reading, MA: Addison Wesley, 2005.

Sito Web consigliato

- **AntiVirus On-line:** il sito IBM di informazioni sui virus.
- **Vmyths:** dedicato allo smascheramento di credenze e concetti fasulli relativi ai veri virus.
- **DDos Attacks/Tools:** ricco elenco di documenti e collegamenti ed altri siti Web.

19.5 Termini chiave, domande di ripasso e problemi

Termini chiave

Attacchi DDoS riflettore
Auto-rooter
Backdoor
Bomba logica
Cavallo di Troia
Distributed denial of service (DDoS)
Downloader
Exploit
Flooder
Keylogger

Kit	Virus
Rootkit	Virus a macro
Sistema immunitario digitale	Virus invisibile
Software doloso	Virus polimorfico
Spammer	Virus di posta elettronica
Trap door	Worm
	Zombie

Domande di ripasso

19.1 Qual è il ruolo della compressione nel funzionamento di un virus?
19.2 Qual è il ruolo della crittografia nel funzionamento di un virus?
19.3 Quali sono le tipiche fasi di funzionamento di un virus o di un worm?
19.4 In termini generali, come si propaga un worm?
19.5 Che cos'è un sistema immunitario digitale?
19.6 Come funziona il software di bloccaggio del comportamento?

Problemi

19.1 Vi è un difetto nel virus rappresentato nella Figura 19.1. Quale?
19.2 Ci si chiede se sia possibile sviluppare un programma che possa analizzare un frammento di software per determinare se si tratti di un virus. Si supponga di avere un programma D in grado di eseguire questa funzione. Ovvero, dato un qualsiasi programma P, il risultato dell'esecuzione di D(P) risulta TRUE quando P è un virus, e FALSE in caso contrario. Ora si consideri il seguente programma:

```
Program CV :=
   { ...
   main-program :=
      {if D(CV) then goto next:
         else infect-executable;
      }
next:
   }
```

Nel programma precedente, infect-executable è un modulo che esegue la scansione della memoria alla ricerca di programmi eseguibili e che si replica in questi programmi. Determinare se D è in grado di decidere correttamente se CV è un virus.

Capitolo 20

I firewall

Concetti essenziali

- Il **firewall** costituisce una barriera attraverso la quale deve passare tutto il traffico diretto in qualsiasi direzione. Le policy di sicurezza del firewall determinano quale traffico è autorizzato a essere trasmesso.
- Un firewall può essere progettato per operare come filtro di pacchetti a livello IP, oppure per operare con protocolli di livello superiore
- Un **sistema fidato** è costituito da un computer e dal relativo sistema operativo che implementano, in modo verificabile, una particolare policy di sicurezza. Solitamente l'obiettivo principale di un sistema fidato è il controllo degli accessi. Una policy viene implementata in modo tale da definire le associazioni tra i soggetti e gli oggetti ai quali i primi possono accedere.
- I cosiddetti **CC**, o "criteri comuni per la valutazione della sicurezza delle tecnologie dell'informazione", scaturiscono da un'iniziativa degli organi di standardizzazione di varie nazioni per sviluppare un insieme comune di requisiti di sicurezza e un modo sistematico per valutare i prodotti informatici rispetto a tali requisiti.

I firewall possono costituire un mezzo efficace per proteggere un sistema locale o una rete dagli attacchi alla sicurezza provenienti dalla rete, consentendo nel frattempo l'accesso al mondo esterno tramite reti geografiche e Internet.

Questo capitolo si apre con una panoramica riguardante le funzionalità e i principi di funzionamento dei firewall. Quindi si affronterà l'argomento della sicurezza del firewall stesso e, in particolare, i concetti di sistema fidato e di sistema operativo fidato.

20.1 I principi progettuali dei firewall

I sistemi informativi delle aziende, degli enti governativi e delle altre organizzazioni hanno subito un'evoluzione continua.

- Sistemi di elaborazione dei dati centralizzati costituiti da un mainframe centrale che supporta dei terminali direttamente connessi.
- Reti locali (LAN – Local Area Networks) che interconnettono i PC e i terminali sia fra di loro che con il mainframe.
- Reti interne costituite da una serie di reti locali che interconnettono i PC, i server e talvolta uno o due mainframe.
- Reti a livello aziendale costituite da più reti interne distribuite geograficamente e interconnesse tramite una rete geografica.
- Connettività a Internet in cui tutte le varie reti interne sono allacciate a Internet e possono essere connesse anche da una rete geografica privata.

La connettività a Internet non è più una questione di scelta per le aziende. Le informazioni e i servizi disponibili in Internet sono ormai fondamentali. Inoltre i singoli utenti dell'azienda hanno necessità di connessione a Internet e se questa funzionalità non venisse offerta tramite la rete locale, impiegherebbero delle connessioni telefoniche fra il PC e un provider Internet. Tuttavia, se da un lato l'accesso a Internet garantisce dei vantaggi per l'azienda, consente anche al mondo esterno di raggiungere i sistemi presenti nella rete locale interna, fatto che comporta dei rischi. Sebbene sia possibile dotare ciascun server e ciascuna workstation della rete di funzionalità di sicurezza adeguate, come la protezione contro le intrusioni, non si tratta di un approccio molto pratico. Si consideri una rete formata da centinaia o anche migliaia di sistemi che utilizzano varie versioni di Unix e Windows. Se dovesse essere scoperta una lacuna di sicurezza, sarebbe necessario aggiornare ogni singolo sistema potenzialmente interessato. L'alternativa, sempre più diffusa, è rappresentata dal firewall. Il firewall si frappone fra la rete interna e Internet per stabilire un collegamento controllato ed erigere una barriera di sicurezza con l'esterno. Lo scopo di questa difesa perimetrale è quello di proteggere i beni dell'azienda dagli attacchi provenienti da Internet e di fornire un unico punto di accesso in cui è possibile imporre la sicurezza e la registrazione degli auditing. Il firewall può essere un unico sistema computerizzato o può essere costituito da due o più sistemi che cooperano per svolgere la funzione di firewall.

In questa parte del capitolo si parlerà innanzitutto delle caratteristiche generali dei firewall. Poi si descriveranno i vari tipi di firewall attualmente in uso. Infine verranno esaminate le configurazioni più comuni dei firewall.

Le caratteristiche dei firewall

[BELL94] elenca i seguenti obiettivi progettuali di un firewall.

1. Tutto il traffico proveniente dall'interno e diretto verso l'esterno e viceversa deve passare attraverso il firewall. Ciò viene ottenuto bloccando fisicamente tutti gli accessi alla rete locale che non attraversano il firewall. È possibile utilizzare varie configurazioni, come si vedrà più avanti.
2. Solo il traffico autorizzato, in base alla politica di sicurezza locale, potrà attraversare il firewall. Come si vedrà più avanti si possono utilizzare vari tipi di firewall che implementano altrettanti tipi di politiche di sicurezza.

3. Il firewall stesso deve essere immune agli attacchi. Questo implica l'impiego di un sistema fidato e di un sistema operativo sicuro. Questo argomento verrà trattato nel Paragrafo 20.2.

[SMIT97] elenca quattro tecniche generali utilizzabili dai firewall per controllare l'accesso e per far rispettare la politica di sicurezza del sito. Originariamente i firewall si concentravano principalmente sul controllo dei servizi ma in seguito si sono evoluti per fornire tutte e quattro queste funzionalità.

- **Controllo dei servizi:** determina i tipi di servizi Internet che possono attraversare il firewall per giungere all'interno o per uscire dalla rete. Il firewall può filtrare il traffico sulla base dell'indirizzo IP e del numero di porta TCP, può fornire del software proxy (intermediario) che riceve e interpreta ciascuna richiesta di servizi prima di farla procedere oppure può ospitare il software del server stesso, per esempio un servizio Web o di posta elettronica.
- **Controllo della direzione:** determina la direzione consentita per determinate richieste di servizio permettendone il passaggio attraverso il firewall.
- **Controllo degli utenti:** controlla l'accesso a un servizio in base all'utente che lo richiede. Questa caratteristica viene tipicamente applicata agli utenti interni rispetto al perimetro del firewall (gli utenti locali). Può essere applicata anche al traffico in arrivo dagli utenti esterni. Quest'ultima possibilità richiede una tecnologia di autenticazione sicura come quella fornita da IPSec (Capitolo 16).
- **Controllo del comportamento:** controlla il modo in cui vengono utilizzati determinati servizi. Per esempio, il firewall può filtrare la posta elettronica per eliminare i messaggi spam o può consentire l'accesso dall'esterno solo a una parte delle informazioni contenute in un server Web locale.

Prima di procedere nei dettagli dei tipi e delle configurazioni dei firewall, è opportuno riepilogare ciò che si ci si può aspettare da un firewall. Ecco le funzionalità offerte.

1. Un firewall definisce un unico punto di accesso che non consente l'accesso alla rete protetta da parte di utenti non autorizzati, proibisce l'ingresso o l'uscita dalla rete di servizi potenzialmente vulnerabili e offre la protezione da vari tipi di attacchi al protocollo IP. L'uso di un unico punto di accesso semplifica la gestione della sicurezza in quanto tutte le funzionalità di sicurezza vengono raggruppate in un unico sistema o gruppo di sistemi.
2. Un firewall costituisce un punto di monitoring degli eventi relativi alla sicurezza. Sul sistema firewall possono essere implementati gli auditing e gli allarmi.
3. Un firewall è una comoda piattaforma per varie funzioni Internet non legate alla sicurezza. Fra queste vi possono essere un traduttore di indirizzi di rete che associa gli indirizzi locali agli indirizzi Internet e una funzione di gestione della rete che registra informazioni di auditing relative all'utilizzo di Internet.
4. Un firewall può fungere da piattaforma per IPSec. Utilizzando la modalità tunnel descritta nel Capitolo 16, il firewall può essere utilizzato per implementare delle reti private virtuali.

Ma i firewall hanno anche dei limiti, fra cui i seguenti.

1. Il firewall non può proteggere da attacchi che, per definizione, sono in grado di oltrepassarlo. I sistemi interni potrebbero essere dotati di funzionalità di connessione diretta a un provider Internet tramite linea telefonica. Una rete locale interna potrebbe essere dotata di una batteria di modem che offre la connessione diretta ai dipendenti che operano da sedi remote.
2. Il firewall non protegge dalle minacce provenienti dall'interno, come per esempio un dipendente insoddisfatto o un dipendente che coopera inconsapevolmente con un hacker esterno.
3. Il firewall non può proteggere dal trasferimento di programmi o file infettati da virus. Data la varietà di sistemi operativi e di applicazioni presenti nel perimetro, sarebbe poco pratico e praticamente impossibile per il firewall eseguire la scansione di tutti i file, i messaggi di posta elettronica e altri messaggi in ingresso alla ricerca dei virus.

I vari tipi di firewall

La Figura 20.1 illustra i tre tipi più comuni di firewall: i filtri di pacchetti, i gateway di livello applicativo e i Gateway a livello del circuito.

Router a filtraggio di pacchetti

Un router a filtraggio di pacchetti applica un insieme di regole a ogni pacchetto IP in ingresso e per decidere se inoltrarlo o eliminarlo. Il router è normalmente configurato per filtrare i pacchetti che procedono in entrambe le direzioni (da o verso la rete interna). Le regole di filtraggio si basano su informazioni contenute nei pacchetti di reti.

- **Indirizzo IP di sorgente:** l'indirizzo IP del sistema in cui ha avuto origine il pacchetto IP (per esempio 192.168.1.1).
- **Indirizzo IP di destinazione:** l'indirizzo IP del sistema che il pacchetto IP sta tentando di raggiungere (per esempio 192.168.1.2).
- **Indirizzi di sorgente e di destinazione a livello di trasporto:** il numero di porta a livello di trasporto (ovvero TCP o UDP) che definisce le applicazioni, per esempio SNMP o Telnet.
- **Campo protocollo del pacchetto IP:** definisce il protocollo di trasporto.
- **Interfaccia:** per un router con tre o più porte, l'interfaccia del router da cui proviene il pacchetto o a cui è destinato il pacchetto.

Il filtro di pacchetti viene normalmente configurato con un elenco di regole che si basano sulle corrispondenze con i campi dell'intestazione IP o TCP. Quando vi è una corrispondenza con una delle regole, questa viene richiamata per determinare se inoltrare o eliminare il pacchetto. Se non esiste alcuna corrispondenza con nessuna delle regole, viene eseguita un'azione standard, di default. Ecco le due possibili politiche di default.

- **Default = discard:** ciò che non viene espressamente permesso è proibito.
- **Default = forward:** ciò che non viene espressamente proibito è permesso.

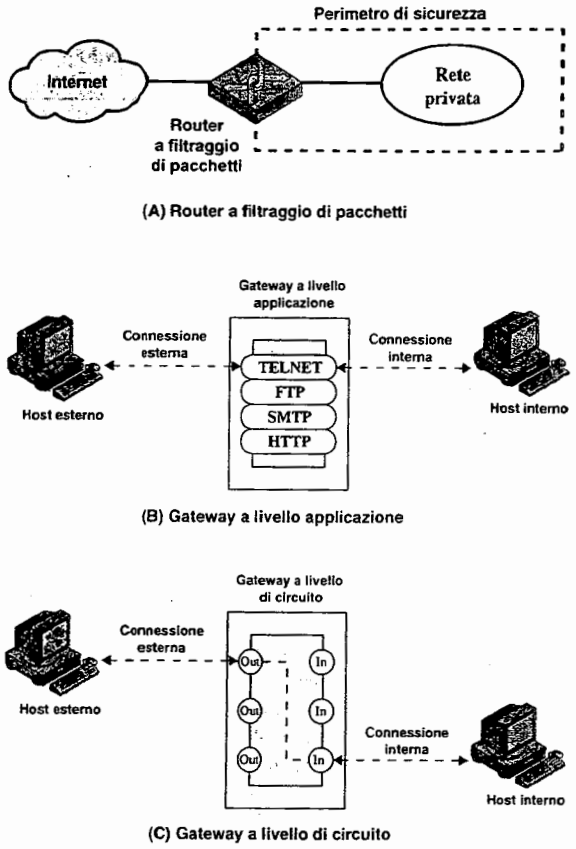


Figura 20.1 I vari tipi di firewall.

La prima politica è la più conservativa. Inizialmente viene bloccato tutto e i servizi devono essere aggiunti caso per caso. Questa politica è più visibile agli utenti che probabilmente vedranno il firewall come un ostacolo. La politica di default di inoltro (la seconda) aumenta la facilità d'uso per gli utenti finali ma riduce la sicurezza; l'amministratore della sicurezza deve, in pratica, reagire a ogni minaccia non appena viene resa nota.

La Tabella 20.1 tratta da [BELL94b] fornisce alcuni esempi di insiemi di regole a filtraggio di pacchetti. In ciascun insieme, le regole vengono applicate dall'alto verso il basso. La presenza di un asterisco in un campo indica un "carattere jolly" che considera ogni valore. Si presuppone l'impiego della politica default = discard.

Tabella 20.1 Esempi di filtraggio di pacchetti.

	Azione	Host interno	Porta	Host esterno	Porta	Commento
A	block	*	*	SPIGOT	*	Non ci possiamo fidare
	allow	OUR-GW	25	*	*	Connessione alla nostra porta SMTP
B	block	*	*	*	*	default
C	allow	*	*	*	25	Connessione alla loro porta SMTP

	Azione	Sorgente	Porta	Destinazione	Porta	Flag	Commento
D	allow	{host interni}	*	*	25		I nostri pacchetti verso la loro porta SMTP
	allow	*	25	*	*	ACK	Le loro risposte
E	allow	{host interni}	*	*	*		Le nostre chiamate in uscita
	allow	*	*	*	*	ACK	Le risposte alle nostre chiamate
	allow	*	*	*	>1024		Traffico verso i non-server

A. La posta elettronica in ingresso è consentita (la porta 25 è relativa al protocollo SMTP), ma solo verso un host gateway. Tuttavia i pacchetti provenienti da un determinato host esterno, SPIGOT, vengono bloccati in quanto tale host ha inviato nel passato enormi file allegati nei messaggi di posta elettronica.
B. Questa è un'affermazione esplicita della politica di default. Tutti gli insiemi di regole includono implicitamente questa regola in ultima posizione.
C. Questo insieme di regole ha lo scopo di specificare che ogni host interno può inviare messaggi di posta elettronica verso l'esterno. Un pacchetto TCP la cui destinazione è la porta 25 viene indirizzato al server SMTP sulla macchina di destinazione. Il problema di questa regola è che l'uso della porta 25 per la ricezione SMTP è solo una convenzione: una macchina esterna potrebbe essere configurata in modo da associare altre applicazioni alla porta 25. Per come è stata definita questa regola, un hacker potrebbe acquisire l'accesso alle macchine interne inviando dei pacchetti dalla porta TCP sorgente 25.
D. Questo insieme di regole consente di ottenere i risultati non ottenuti in C. Le regole sfruttano una caratteristica delle connessioni TCP. Una volta che è stata configurata una connessione, il flag ACK di un segmento TCP viene impostato in modo da eseguire l'acknowledgement dei segmenti inviati dall'altro capo. Pertanto questo insieme di regole consente il passaggio dei pacchetti IP dove l'indirizzo IP sorgente è uno degli host interni indicati nell'elenco e la porta di destinazione TCP è la numero 25. Inoltre consente l'ingresso dei pacchetti con il numero di porta sorgente 25 che includono il flag ACK nel segmento TCP. Si noti che per definire esplicitamente queste regole si designano esplicitamente i sistemi sorgente e di destinazione.
E. Questo insieme di regole è un metodo per la gestione delle connessioni FTP. Con FTP, vengono utilizzate due connessioni TCP: una di controllo per configurare il trasferimento di file e una dati per il trasferimento effettivo del file. La connessione dati usa una porta differente che viene assegnata dinamicamente per il trasferimento. La maggior parte dei server, e pertanto la maggior parte dei bersagli degli attacchi, si trova su porte con un numero basso; la maggior parte delle chiamate verso l'uscita tende a utilizzare porte con una numerazione elevata, in genere superiore a 1023. Pertanto questo insieme di regole consente il passaggio dei seguenti elementi.

- Pacchetti con origine interna.
- Pacchetti di risposta per una connessione attivata da una macchina interna.
- Pacchetti destinati a una porta alta di una macchina interna.

Questo meccanismo richiede che i sistemi siano configurati in modo che vengano utilizzati solo i numeri di porta appropriati.

L'insieme di regole E evidenzia la difficoltà di gestione delle applicazioni a livello di filtraggio dei pacchetti. Un altro modo per gestire FTP e applicazioni analoghe prevede l'impiego di un gateway a livello applicazione, di cui si parlerà più avanti.

Un vantaggio del router a filtraggio di pacchetti è la sua semplicità. Inoltre i filtri sui pacchetti sono generalmente trasparenti agli utenti e molto veloci. [WACK02] elenca i seguenti punti deboli dei firewall a filtraggio di pacchetti.

- Poiché i firewall a filtraggio di pacchetti non esaminano i dati dei livelli superiori, non possono prevenire attacchi che sfruttano i punti deboli specifici delle applicazioni. Per esempio, un firewall a filtraggio di pacchetti non può bloccare specifici comandi per le applicazioni; se un firewall a filtraggio di pacchetti consente una determinata applicazione, saranno consentite tutte le funzioni disponibili all'interno di tale applicazione.
- Dato che il firewall ha a disposizione informazioni limitate, la funzionalità di logging presente nei firewall a filtraggio di pacchetti è molto limitata. Normalmente i log di filtraggio dei pacchetti contengono le stesse informazioni utilizzate per prendere la decisione di controllo di accesso (indirizzo di sorgente, indirizzo di destinazione e tipo di traffico).
- La maggior parte dei firewall a filtraggio di pacchetti non supporta i meccanismi più avanzati di autenticazione degli utenti. Ancora una volta questo limite è dovuto principalmente alla mancanza di funzionalità di alto livello nel firewall.
- In generale i firewall sono vulnerabili agli attacchi che sfruttano i problemi delle specifiche e dello stack di protocolli TCP/IP, come per esempio lo *spoofing* dell'indirizzo di rete. Molti firewall a filtraggio di pacchetti non sono in grado di rilevare anomalie in un pacchetto di rete all'interno del quale le informazioni di indirizzamento del livello 3 del modello OSI siano state modificate. Gli attacchi a spoofing consentono agli hacker di oltrepassare i controlli di sicurezza implementati da una piattaforma a firewall.
- Infine, dato il numero ridotto di variabili utilizzate nelle decisioni di controllo di accesso, i firewall a filtraggio di pacchetti sono sensibili alle violazioni alla sicurezza dovute a un'errata configurazione. In altre parole, è facile configurare accidentalmente un firewall a filtraggio di pacchetti per consentire tipi di traffico, sorgenti e destinazioni che dovrebbero essere proibite in base alle politiche di sicurezza delle informazioni stabilite dall'azienda.

Ecco alcuni degli attacchi che possono essere sferrati contro i router a filtraggio di pacchetti e le relative contromisure.

- **Spoofing dell'indirizzo IP:** l'hacker trasmette i pacchetti dall'esterno specificando un indirizzo IP sorgente appartenente a un host interno. L'hacker spera che l'indirizzo fasullo venga accettato in quanto appartiene a un host interno fidato. La contromisura consiste nell'eliminare tutti i pacchetti provenienti dall'esterno che contengono come indirizzo sorgente quello di una macchina interna.
- **Attacchi a source routing:** la stazione emittente specifica il percorso di un pacchetto attraverso Internet nella speranza che questo oltrepassi le misure di sicurezza che non analizzano le informazioni di source routing. La contromisura consiste nell'eliminare tutti i pacchetti che usano questa opzione.
- **Attacchi a frammentazione:** l'hacker utilizza l'opzione di frammentazione IP per creare frammenti talmente piccoli da costringere a disporre le informazioni di intestazione TCP in un frammento distinto del pacchetto. Questo attacco ha lo scopo di aggirare le regole di filtraggio che dipendono dalle informazioni contenute nell'intestazione TCP. Solitamente, un filtro di pacchetti prende decisioni sulla base del primo frammento del pacchetto. Tutti i frammenti successivi del medesimo pacchetto vengono scartati o meno a seconda che sia stato scartato il primo frammento del pacchetto. L'hacker spera che il router a filtraggio esamini solo il primo frammento e che i frammenti rimanenti vengono lasciati passare. Questo attacco può essere sconfitto imponendo la regola che il primo frammento di un pacchetto deve contenere almeno una parte predefinita dell'header di trasporto. Se il primo frammento viene rifiutato, il filtro dovrà poi scartare tutti i frammenti successivi del medesimo pacchetto.

Firewall di ispezione a stati

Un filtro a pacchetti tradizionale prende decisioni di filtraggio considerando singolarmente i pacchetti senza tenere in considerazione il contesto rappresentato dai livelli sovrastanti. Per comprendere il significato di contesto e il motivo per cui un filtro di pacchetti tradizionali è limitato rispetto al contesto, è necessario fornire qualche informazione di base. La maggior parte delle applicazioni standardizzate che opera su TCP adotta un modello client/server. Per esempio, il protocollo SMTP (Simple Mail Transfer Protocol) consente di trasmettere i messaggi di posta elettronica da un sistema client a un sistema server. Il sistema client genera nuovi messaggi di posta elettronica in base all'input degli utenti. Il sistema server accetta i messaggi di posta elettronica in arrivo e li inserisce nelle caselle degli utenti appropriati. SMTP configura una connessione TCP fra il client e il server, in cui il numero di porta TCP che identifica il server SMTP è 25. Il numero di porta TCP per il client SMTP è un numero compreso fra 1024 e 65 535, generato dal client SMTP stesso.

In generale, quando un'applicazione che utilizza TCP crea una sessione con un host remoto, crea una connessione TCP in cui il numero di porta TCP per l'applicazione server remota è un numero minore di 1024 e il numero di porta TCP per l'applicazione locale client è un numero compreso fra 1024 e 65 535. I numeri minori di 1024 riguardano porte "note" che vengono assegnate in modo permanente a determinate applicazioni (per esempio la porta 25 è assegnata al server SMTP). I numeri compresi fra 1024 e 65 535 vengono generati dinamicamente e hanno un significato temporaneo per la sola durata di una connessione TCP.

Un semplice firewall a filtraggio di pacchetti deve consentire il traffico di rete in ingresso su tutte queste porte con un numero elevato, in modo da consentire il passaggio del traffico TCP. Questo crea un punto debole che può essere sfruttato da utenti non autorizzati.

Un filtro di ispezione dei pacchetti a stati consente di rendere più efficaci le regole per il traffico TCP creando un elenco di connessioni TCP in uscita come quello indicato nella Tabella 20.2. Vi è una voce per ogni connessione attualmente attiva. Il filtro di pacchetti accetterà l'ingresso del traffico diretto a un numero di porta elevato solo per quei pacchetti che rispondono al profilo di una delle voci contenute in questo elenco.

Tabella 20.2 Esempio di tabella di un firewall a stati (WACK02).

Indirizzo sorgente	Porta sorgente	Indirizzo di destinazione	Porta di destinazione	Stato della connessione
192.168.1.100	1030	210.9.88.29	80	Established
192.168.1.102	1031	216.32.42.123	80	Established
192.168.1.101	1033	173.66.32.122	25	Established
192.168.1.106	1035	177.231.32.12	79	Established
223.43.21.231	1990	192.168.1.6	80	Established
219.22.123.32	2112	192.168.1.6	80	Established
210.99.212.18	3321	192.168.1.6	80	Established
24.102.32.23	1025	192.168.1.6	80	Established
223.212.212	1046	192.168.1.6	80	Established

Gateway a livello applicazione

Un gateway a livello applicazione, chiamato anche proxy server (letteralmente "server intermediario"), si comporta come un ripetitore di traffico a livello delle applicazioni (Figura 20.1B). L'utente contatta il gateway utilizzando un'applicazione TCP come Telnet o FTP e il gateway chiede all'utente il nome dell'host remoto cui accedere. Quando l'utente risponde e fornisce un codice utente e informazioni di autenticazione validi, il gateway contatta l'applicazione sull'host remoto e rinvia i segmenti TCP contenenti i dati dell'applicazione fra i due punti terminali. Se il gateway non implementa il codice proxy per una determinata applicazione, il servizio non sarà supportato e non potrà essere inoltrato tramite il firewall. Inoltre, il gateway può essere configurato per supportare solo determinate funzionalità di un'applicazione che l'amministratore di rete considera accettabili, impedendo tutte le altre funzionalità.

I gateway a livello applicazione sono più sicuri dei filtri di pacchetti. Invece di tentare di gestire tutte le possibili combinazioni consentite e proibite a livello TCP e IP, i gateway a livello applicazione devono solo esaminare alcune applicazioni consentite. Inoltre è facile registrare ed effettuare l'auditing di tutto il traffico in ingresso a livello delle applicazioni.

Un grave svantaggio di questo tipo di gateway è il livello di elaborazione richiesto per ogni connessione. In pratica la connessione fra gli utenti finali viene interrotta al livello del gateway e il gateway deve esaminare e inoltrare tutto il traffico in entrambe le direzioni.

Gateway a livello di circuito

Un terzo tipo di firewall è rappresentato dal gateway a livello di circuito (Figura 20.1C). Può trattarsi di un sistema indipendente o di una funzione specializzata svolta da un gateway solo per determinate applicazioni. Un gateway a livello di circuito non consente una connessione TCP end-to-end ma configura due connessioni TCP, una fra se stesso e l'utente TCP sull'host interno e una fra se stesso e l'utente TCP sull'host esterno. Una volta attivate le due connessioni, il gateway rinvia i segmenti TCP da una connessione all'altra senza esaminarne il contenuto. La funzione di sicurezza è costituita dalla scelta delle connessioni consentite.

Un utilizzo tipico dei gateway a livello di circuito è la situazione nella quale l'amministratore di sistema si fida degli utenti interni. Il gateway può essere configurato per supportare un filtraggio a livello applicazione o un servizio proxy sulle connessioni provenienti dall'esterno e delle funzioni a livello di circuito per le connessioni provenienti dall'interno. In questa configurazione, il gateway incorre nel sovraccarico elaborativo dovuto all'esame dei dati delle applicazioni in ingresso alla ricerca di funzioni proibite ma non deve subire il sovraccarico sui dati in uscita.

Un esempio di implementazione di gateway a livello dei circuiti è rappresentato dal pacchetto SOCKS [KOBL92]; la versione 5 di SOCKS è definita dal documento RFC 1928 nel seguente modo:

"Il protocollo è progettato per fornire una struttura di connessione affinché applicazioni client-server nei domini TCP e UDP possano utilizzare in modo comodo e sicuro i servizi di un firewall di rete. Il protocollo concettualmente si frappone fra il livello applicazione e il livello trasporto e per questo non fornisce i servizi gateway di livello rete come per esempio l'inoltro dei messaggi ICMP".

SOCKS è costituito dai seguenti componenti.

- Il server SOCKS operante su un firewall Unix.
- Il client SOCKS eseguito sugli host interni protetti dal firewall.
- Versioni SOCKS di vari programmi client standard come FTP e Telnet. L'implementazione del protocollo SOCKS comporta normalmente la ricompilazione (o il ri-collegamento) delle applicazioni client TCP per utilizzare le routine di incapsulazione appropriate contenute nella libreria SOCKS.

Quando un client TCP desidera stabilire una connessione con un oggetto raggiungibile solo tramite un firewall (tale vincolo imposto tramite la particolare implementazione), deve aprire una connessione TCP sulla porta SOCKS appropriata del server SOCKS. Il servizio SOCKS è situato sulla porta TCP 1080. Se la richiesta di connessione ha successo, il client entra in una fase di negoziazione del metodo di autenticazione da impiegare, autentica con il metodo scelto e quindi invia una richiesta di invio. Il server SOCKS valuta la richiesta e consente o proibisce la connessione. Gli scambi UDP vengono gestiti in modo analogo. In pratica viene aperta una connessione TCP per autenticare un utente per l'invio e la ricezione di segmenti UDP, segmenti che vengono inoltrati fintantoché la connessione TCP rimane aperta.

Host bastione

Un host bastione è un sistema che l'amministratore del firewall elegge quale punto di forza critico per la sicurezza della rete. In genere l'host bastione funge da piattaforma per un gateway a livello applicazione o di circuito. Ecco le caratteristiche principali di un host bastione.

- La piattaforma hardware dell'host bastione adotta una versione sicura del sistema operativo e dunque è un sistema fidato.
- Sull'host bastione sono installati unicamente i servizi che l'amministratore della rete considera essenziali. Fra questi vi sono applicazioni proxy come Telnet, DNS, FTP SMTP e l'autenticazione degli utenti.
- L'host bastione può richiedere ulteriori autenticazioni prima di consentire a un utente l'accesso ai servizi proxy. Inoltre ciascun servizio proxy può richiedere una propria autenticazione prima di consentire l'accesso agli utenti.
- Ciascun proxy è configurato per supportare solo un sottoinsieme dei comandi dell'applicazione.
- Ciascun proxy è configurato per consentire l'accesso solo a determinati sistemi host. Questo significa che il sottoinsieme dei comandi consentiti potrebbe essere applicato solo a un sottoinsieme dei sistemi della rete protetta.
- Ciascun proxy gestisce informazioni di auditing dettagliate registrando tutto il traffico, ogni connessione e la relativa durata. Il log di auditing è uno strumento essenziale per scoprire e bloccare gli attacchi degli hacker.
- Ogni modulo del proxy è un pacchetto software estremamente compatto progettato in modo specifico per la sicurezza della rete. Data la sua relativa semplicità, risulta più facile identificare le sue eventuali lacune di sicurezza. Per esempio, una tipica applicazione di posta elettronica Unix può essere costituita da oltre 20 000 righe di codice mentre un proxy di posta elettronica può contenerne meno di 1000.
- Ciascun proxy è indipendente dagli altri proxy dell'host bastione. Se vi è un problema nel funzionamento di un proxy o se viene individuata una lacuna di sicurezza, il proxy potrà essere disinstallato senza che ciò pregiudichi il funzionamento delle altre applicazioni proxy. Inoltre quando la popolazione di utenti richiede un nuovo servizio, l'amministratore di rete può facilmente installare il relativo proxy sull'host bastione.
- Un proxy generalmente non esegue alcun accesso a disco se non per la lettura iniziale del file di configurazione. Questo complica l'installazione di un cavallo di Troia, di uno sniffer o di altri software pericolosi sull'host bastione da parte degli hacker.
- Ogni proxy è eseguito come utente non privilegiato in una directory privata e sicura dell'host bastione.

Configurazioni firewall

Oltre a utilizzare una semplice configurazione costituita da un unico sistema, che può essere un router a filtraggio di pacchetti o un gateway (Figura 20.1), sono possibili configurazioni più articolate. La Figura 20.2 illustra tre tipiche configurazioni firewall.

Nella configurazione a firewall con host schermato, **single-homed bastion** (Figura 20.2A) il firewall è costituito da due sistemi: un router a filtraggio di pacchetti e un host bastione. In genere il router è configurato in modo che:

1. Per il traffico proveniente da Internet, sia consentito l'ingresso solo ai pacchetti IP destinati all'host bastione.
2. Per il traffico proveniente dalla rete interna, sia consentita l'uscita ai soli pacchetti provenienti dall'host bastione.

L'host bastione svolge le funzioni di autenticazione e proxy. Questa configurazione offre una maggiore sicurezza rispetto al semplice router a filtraggio di pacchetti o al gateway a livello applicazione per due motivi. Innanzitutto viene implementato il filtraggio sia a livello di rete che a livello applicazione consentendo maggiore flessibilità nella definizione delle politiche di sicurezza. In secondo luogo, un hacker deve violare due diversi sistemi prima di poter compromettere la sicurezza della rete interna.

Questa configurazione offre anche una maggiore flessibilità nel fornire l'accesso diretto a Internet. Per esempio, la rete interna potrebbe includere un server di informazioni pubblico come un server Web, per il quale non è richiesto un elevato livello di sicurezza. In tal caso, il router può essere configurato per consentire il traffico diretto fra il server di informazioni e Internet.

Nella configurazione single-homed appena descritta, se il router a filtraggio di pacchetti venisse completamente violato, il traffico potrebbe scorrere direttamente attraverso il router fra Internet e gli altri host della rete privata. Il firewall a host schermato in configurazione **dual-homed bastion** impedisce fisicamente tale violazione alla sicurezza (vedere la Figura 20.2B). Esiste anche in questo caso il vantaggio dei due livelli di sicurezza della configurazione precedente. Anche in questo caso un server di informazioni o altri host potrebbero avere una comunicazione diretta con il router se ciò fosse accettabile in base alla politica di sicurezza.

La configurazione firewall a sottorete schermata rappresentata nella Figura 20.2C è la più sicura fra quelle appena considerate. In questa configurazione vengono utilizzati due router a filtraggio di pacchetti, uno fra l'host bastione e Internet e uno fra l'host bastione e la rete interna. Questa configurazione crea una sottorete isolata che può essere costituita dal semplice host bastione ma che può anche includere uno o più server di informazioni e modem per connessioni telefoniche. In genere sia Internet che la rete interna hanno accesso agli host sulla sottorete schermata ma il traffico che attraversa questa sottorete viene bloccato. Questa configurazione presenta vari vantaggi.

- Ora vi sono tre livelli di difesa contro gli hacker.
- Il router esterno mostra a Internet solo l'esistenza della sottorete schermata; pertanto la rete interna risulta invisibile da Internet.
- Analogamente, il router interno mostra alla rete interna solo l'esistenza della sottorete schermata, pertanto i sistemi della rete interna non possono realizzare percorsi diretti verso Internet.

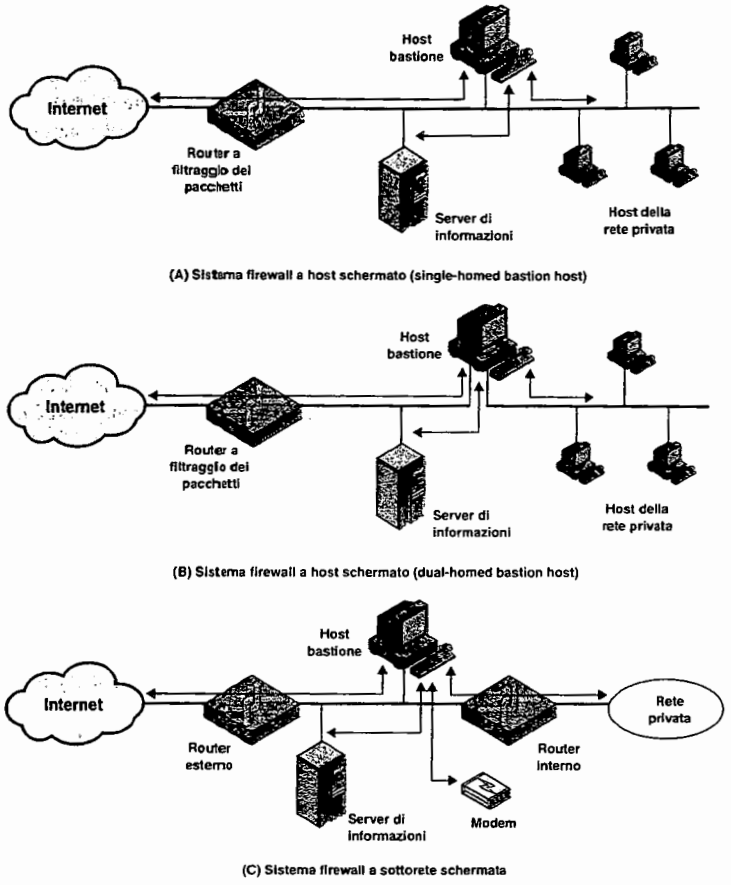


Figura 20.2 Tre configurazioni firewall.

20.2 Sistemi fidati

Un modo per migliorare la capacità di difesa di un sistema dagli hacker e dai programmi dolosi consiste nell'implementare una tecnologia a sistema fidato. Questa parte del capitolo presenta una breve panoramica su questo argomento. Si inizierà descrivendo alcuni dei concetti di base relativi al controllo dell'accesso ai dati.

Il controllo dell'accesso ai dati

Dopo avere eseguito con successo un'operazione di login, all'utente viene assegnato l'accesso a un insieme di host e applicazioni. Questo in genere non è sufficiente per un sistema che contiene dati riservati nel proprio database. Tramite la procedura di controllo degli accessi degli utenti, un utente può essere identificato sul sistema. A ciascun utente è associato un profilo che specifica quali operazioni e quali accessi ai file sono consentiti. Il sistema operativo può pertanto attivare delle regole sulla base del profilo dell'utente. Ma il sistema di gestione del database deve controllare l'accesso a specifici record e perfino a determinate parti dei record. Per esempio, nel reparto amministrazione di un'azienda tutti devono potere ottenere un elenco del personale ma solo determinate persone potranno avere accesso alle informazioni sugli stipendi. Questo problema non è certo un dettaglio. Mentre il sistema operativo può consentire all'utente l'accesso a un file o a un'applicazione, oltre a questo livello non esiste alcun ulteriore controllo di sicurezza. Il sistema di gestione del database invece deve prendere una decisione per ogni singola richiesta di accesso. Tale decisione non dipende solo dall'identità dell'utente ma anche dai dati consultati e perfino dalle informazioni già divulgate all'utente.

Un modello generale di controllo degli accessi utilizzato nei sistemi di gestione dei database o dei file è quello chiamato a matrice di accesso (Figura 20.3A). Ecco i principali elementi di questo modello.

- **Soggetto:** entità in grado di accedere agli oggetti. In generale un soggetto è un processo. Ogni utente o applicazione infatti acquisisce l'accesso a un oggetto tramite un processo che li rappresenta.
- **Oggetto:** qualsiasi elemento per il quale deve essere controllato l'accesso. Fra gli esempi vi sono i file, le parti di file, i programmi e i segmenti di memoria.
- **Diritti di accesso:** modalità con la quale un soggetto può accedere a un oggetto. Esempi di operazioni eseguibili sono la lettura, la scrittura e l'esecuzione.

Un asse della matrice è rappresentato dai soggetti identificati che richiedono l'accesso ai dati. In genere questo elenco è costituito da singoli utenti o gruppi di utenti sebbene l'accesso possa essere controllato per terminali, host o applicazioni in sostituzione o in aggiunta al controllo sugli utenti. L'altro asse elenca gli oggetti cui è possibile accedere. Al livello di dettaglio più elevato, gli oggetti possono essere i singoli campi di dati. Gli oggetti della matrice possono essere rappresentati da raggruppamenti più complessi come record, file o perfino interi database. Ciascun elemento della matrice indica i diritti di accesso del soggetto su tale oggetto.

In pratica una matrice degli accessi è normalmente una matrice sparsa implementata per decomposizione in due diversi modi. La matrice può essere decomposta in base alle colonne, fornendo una **lista di controllo degli accessi** (Figura 20.3B). Pertanto, per ciascun oggetto, una lista di controllo degli accessi elenca gli utenti e i relativi diritti di accesso. La lista di controllo degli accessi può contenere una voce di default o pubblica che consente di assegnare un determinato insieme di diritti agli utenti che non hanno specifici permessi di accesso. Gli elementi di questo elenco possono includere singoli utenti o anche gruppi di utenti.

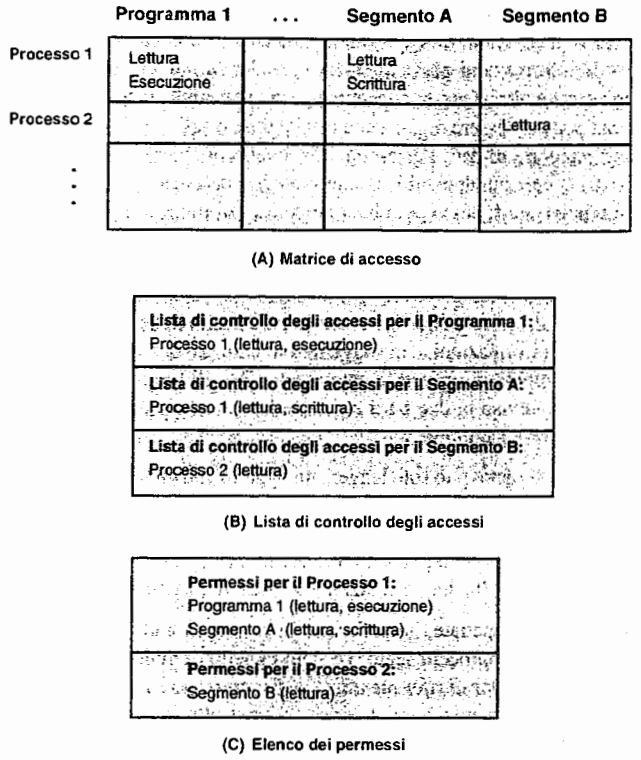


Figura 20.3 Struttura di controllo degli accessi.

La decomposizione in base alle righe fornisce un **elenco dei permessi** (Figura 20.3C) che specifica gli oggetti e le operazioni disponibili per un utente. Ciascun utente avrà un certo numero di permessi e può esser autorizzato ad assegnare tali permessi ad altri. Poiché questi permessi possono essere dispersi nel sistema, rappresentano un grave problema di sicurezza rispetto alle liste di controllo degli accessi. In particolare, il permesso di accesso non deve essere falsificabile. Una tecnina per ottenere ciò consiste nel fare in modo che il sistema operativo conservi tutti i permessi per conto dell'utente. Questi permessi dovranno essere conservati in un'area di memoria inaccessibile agli utenti.

Il concetto di sistema fidato

Ciò che si detto finora riguardava la protezione di un determinato messaggio o elemento contro attacchi attivi o passivi da parte di un determinato utente. Un requisito per certi versi

differente ma ampiamente applicabile è quello di proteggere i dati o le risorse sulla base dei cosiddetti livelli di sicurezza. Questo requisito è normalmente previsto negli ambienti militari, dove le informazioni possono essere considerate non classificate, confidenziali, segrete o top-secret e oltre. Questo concetto è però applicabile anche ad altri settori, dove le informazioni possono essere organizzate in categorie e agli utenti è consentito l'accesso a determinate categorie di dati. Per esempio, il livello di sicurezza più elevato potrebbe essere quello dei documenti di pianificazione strategica dell'azienda, accessibili solo ai top manager e al relativo staff; poi vi potrebbero essere i dati finanziari più delicati e i dati sul personale, accessibili solo da parte del personale di amministrazione, ai top manager e così via.

Quando esistono più categorie o livelli di dati, si parla di **sicurezza multilivello**. La formulazione generale del requisito di sicurezza multilivello è che un soggetto di alto livello non deve fornire informazioni a un soggetto di livello più basso a meno che questo non rifletta la precisa volontà di un utente autorizzato. Per gli scopi implementativi, questo requisito può essere diviso in due parti e riformulato in modo più semplice. Un sistema sicuro multilivello deve garantire:

- **No read up:** un soggetto può solo leggere un oggetto con un livello di sicurezza uguale o inferiore. Nella documentazione tecnica viene chiamato **proprietà di sicurezza semplice**.
- **No write down:** un soggetto può solo scrivere in un oggetto con un livello di sicurezza uguale o superiore. Nella documentazione tecnica viene chiamato "*-property".

Queste due regole, se rispettate, garantiscono la sicurezza multilivello. Per i sistemi di elaborazione dei dati, l'approccio seguito è stato oggetto di molte ricerche e sviluppi e si basa sul concetto di *monitor di riferimento*. Questo approccio è rappresentato nella Figura 20.4. Il monitor di riferimento è un elemento di controllo presente nell'hardware e nel sistema operativo di un computer che regola l'accesso dei soggetti agli oggetti sulla base dei loro parametri di sicurezza. Il monitor di riferimento ha accesso a un file, il database centrale della sicurezza, che elenca i privilegi di accesso (security clearance) di ciascun soggetto e gli attributi di protezione (classification level) di ciascun oggetto. Il monitor di riferimento garantisce il rispetto delle due regole di sicurezza elencate in precedenza e ha le seguenti proprietà.

- **Mediazione completa:** le regole di sicurezza vengono applicate a ogni accesso e non, per esempio, solo quando viene aperto un file.
- **Isolamento:** il monitor di riferimento e il database sono protetti da qualsiasi modifica non autorizzata.
- **Verificabilità:** la correttezza del monitor di riferimento deve essere dimostrabile. Ovvero deve essere possibile dimostrare matematicamente che il monitor di riferimento garantisce il rispetto delle regole di sicurezza, una mediazione e un isolamento completi.

Si tratta di requisiti molto rigidi. Il requisito di mediazione completa significa che ogni accesso ai dati contenuti nella memoria principale e nei dischi e su nastro deve essere mediato. Un'implementazione puramente software imporrebbe un aggravio prestazionale troppo elevato; la soluzione deve essere almeno parzialmente hardware. Il requisito di isolamento significa che non deve essere possibile per un estraneo, indipendentemente dalla sua abilità, alterare la logica di funzionamento del monitor di riferimento o il contenuto del database centrale di sicurezza. Infine il requisito di prova matematica è formidabile per un sistema così complesso come un computer. Un sistema in grado di soddisfare questi requisiti è chiamato **sistema fidato**.

L'ultimo elemento rappresentato nella Figura 20.4 è il file di auditing. Gli eventi più importanti della sicurezza, come il rilevamento delle violazioni alla sicurezza e le modifiche autorizzate al database centrale della sicurezza vengono conservate nel file di auditing.

Nel tentativo di rispondere alle sue stesse esigenze e come servizio pubblico, il Dipartimento della Difesa degli Stati Uniti ha fondato nel 1981 il Computer Security Center nell'ambito della NSA (National Security Agency) con l'obiettivo di incoraggiare lo sviluppo di sistemi fidati. Questo obiettivo viene perseguito tramite il programma Commercial Product Evaluation Program del centro. In pratica, si tenta di valutare come i prodotti disponibili commercialmente soddisfino i requisiti di sicurezza appena descritti. Il centro classifica i prodotti valutati in base alla gamma di funzionalità. Queste valutazioni sono necessarie per gli acquisti del Dipartimento della Difesa ma vengono rese pubblicamente accessibili: pertanto possono essere utili anche alle aziende che acquistano apparecchiature disponibili commercialmente.

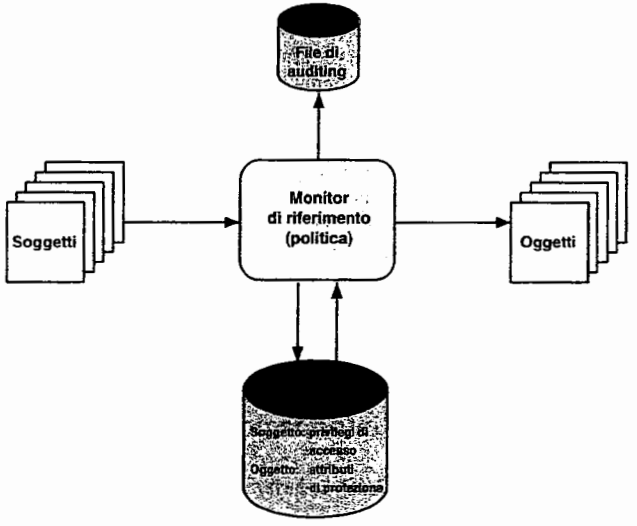


Figura 20.4 Concetto del monitor di riferimento.

Difesa contro i cavalli di Troia

Un modo per difendersi dagli attacchi a cavallo di Troia consiste nell'uso di un sistema operativo sicuro e fidato.

La Figura 20.5 illustra un esempio. In questo caso viene utilizzato un cavallo di Troia per aggirare il meccanismo di sicurezza standard utilizzato dalla maggior parte dei sistemi di gestione dei file e dei sistemi operativi: la lista di controllo degli accessi. In questo esempio l'utente Bob interagisce tramite un programma con un file di dati contenente la stringa di caratteri segreta "CPE170KS". L'utente Bob ha assegnato i permessi di lettura/scrittura al file solo per i programmi operanti per suo conto; ovvero solo i processi di proprietà di Bob possono accedere al file.

L'attacco a cavallo di Troia inizia quando un utente ostile, Alice, acquisisce legittimamente l'accesso al sistema e installa un cavallo di Troia e un file di servizio che verrà utilizzato durante l'attacco. Alice assegna a se stessa il permesso di lettura e scrittura su questo file e assegna a Bob il solo permesso di scrittura (Figura 20.5A). Ora Alice induce Bob a richiamare il programma contenente il cavallo di Troia, sostenendo che si tratta di un programma della massima utilità. Quando il programma scopre di essere eseguito da Bob, legge la stringa di caratteri segreta dal file di Bob e la copia nel file di servizio di Alice (Figura 20.5B). Entrambe le operazioni di lettura e scrittura soddisfano i vincoli imposti dalla lista di controllo degli accessi. Per conoscere il valore della stringa, ad Alice non rimane che accedere al proprio file.

Ora si vedrà come si comporta un sistema operativo sicuro in questa stessa situazione (Figura 20.5C). I livelli di sicurezza vengono assegnati ai soggetti al momento del login sulla base di criteri quali il terminale utilizzato per l'accesso e l'utente interessato, identificato tramite codice utente e password. In questo esempio vi sono due livelli di sicurezza, riservato e pubblico, ordinati in modo che riservato sia superiore a pubblico. I processi e i file di dati di Bob ricevono il livello di sicurezza riservato. I file e i processi di Alice ricevono invece il livello di sicurezza pubblico. Se Bob richiama il cavallo di Troia (Figura 20.5D), tale programma acquisisce il livello di sicurezza di Bob. Il programma sarà pertanto in grado, rispettando la proprietà di sicurezza semplice, di leggere la stringa di caratteri segreta. Quando però il programma tenta di memorizzare la stringa in un file pubblico (il file di servizio di Alice), viene violata la proprietà "*-property" e il tentativo viene bloccato dal monitor di riferimento. Pertanto il tentativo di scrivere nel file di servizio viene impedito anche se la lista di controllo degli accessi lo consentirebbe: la politica di sicurezza ha la precedenza sul meccanismo di controllo degli accessi.

20.3 Criteri comuni per la valutazione della sicurezza delle tecnologie dell'informazione

Lo sforzo compiuto dalla NSA (National Security Agency) e altre agenzie governative negli Stati Uniti per definire requisiti e criteri di valutazione per i sistemi fidati è riflesso in sforzi analoghi di altre nazioni. I cosiddetti CC, o *Common Criteria for Information Technology and Security Evaluation*, scaturiscono da un'iniziativa degli organi di standardizzazione di varie nazioni per sviluppare standard internazionali di specifica dei requisiti di sicurezza e dei criteri di valutazione.

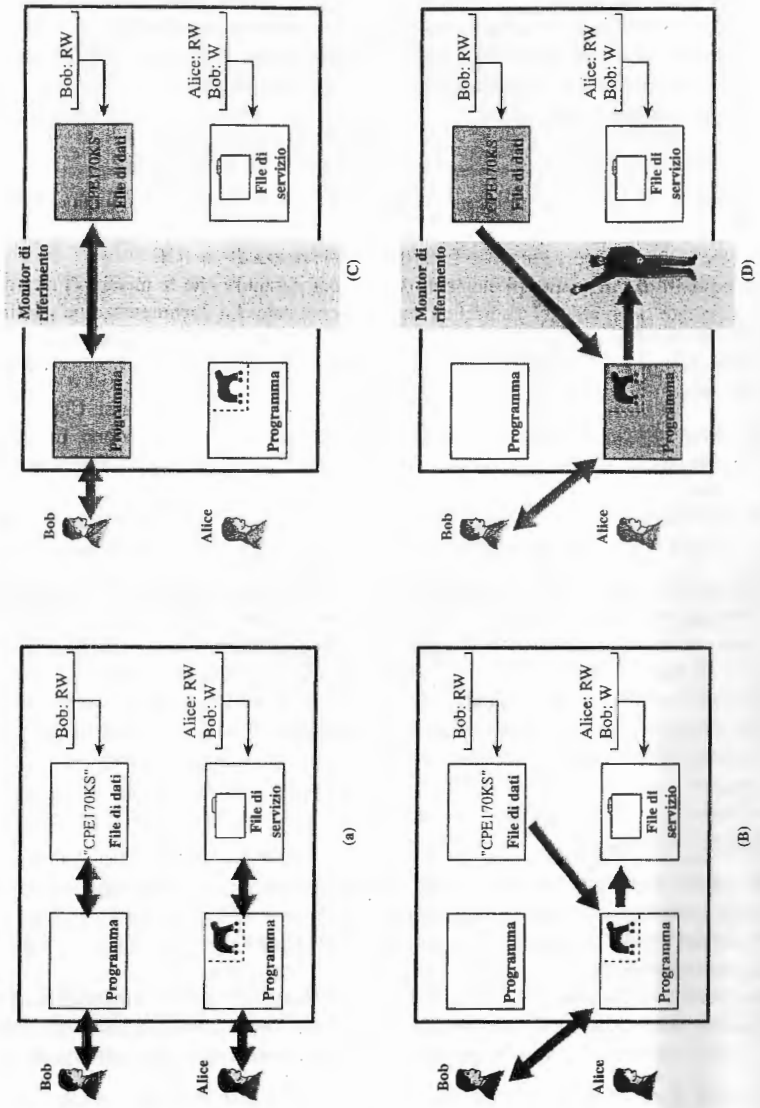


Figura 20.5 Cavalli di Troia su sistemi operativi sicuri.

Requisiti

I CC definiscono un insieme comune di potenziali requisiti di sicurezza da utilizzare nella valutazione. L'acronimo TOE (Target Of Evaluation, o *obiettivo della valutazione*) si riferisce a quella parte di applicazione o sistema oggetto di valutazione. I requisiti si raggruppano in due categorie.

- **Requisiti funzionali:** definiscono il comportamento di sicurezza desiderato. La documentazione stabilisce un insieme di componenti che forniscono una modalità standardizzata per esprimere i requisiti funzionali di sicurezza di un TOE.
- **Requisiti di garanzia:** gli elementi di base per garantire che le misure di sicurezza impostate siano efficaci e correttamente implementate. La documentazione stabilisce un insieme di componenti per esprimere in forma standardizzata i requisiti di garanzia relativi a un TOE.

Sia i requisiti funzionali sia i requisiti di garanzia sono organizzati in classi. Una *classe* è un insieme di requisiti che condividono il medesimo obiettivo o intenzione. Le Tabelle 20.3 e 20.4 definiscono sinteticamente le classi dei requisiti funzionali e di garanzia. Ciascuna di queste classi contiene un insieme di famiglie. I requisiti in ciascuna famiglia condividono gli obiettivi di sicurezza ma differiscono nell'enfasi o nel rigore. Per esempio la classe *audit* contiene sei famiglie relative a vari aspetti di auditing (tra le quali la generazione dei dati di audit, l'analisi di audit e la memorizzazione degli eventi di audit). Ciascuna famiglia, a sua volta, contiene una o più componenti. Una *componente* descrive un insieme specifico di requisiti di sicurezza e costituisce il più piccolo insieme selezionabile per far parte delle strutture definite nei CC.

Tabella 20.3 Requisiti funzionali di sicurezza CC. *(continua)*

Classe	Descrizione
Audit	Riguarda il riconoscimento, la registrazione, la memorizzazione e l'analisi delle informazioni relative alle attività di sicurezza. Queste attività generano i record di audit, che possono essere esaminati per determinarne l'impatto sulla sicurezza.
Supporto crittografico	Utilizzata nel caso in cui il TOE implementa funzioni di crittografia. Queste possono essere impiegate, per esempio, per supportare la comunicazione, l'identificazione e autenticazione, o la separazione dei dati.
Comunicazioni	Fornisce due famiglie relative alla non-ripudiazione da parte del mittente e del destinatario dei dati.
Protezione dei dati utente	Specifica i requisiti relativi alla protezione dei dati utente nel TOE durante l'importazione, l'esportazione e la memorizzazione, oltre agli attributi di sicurezza relativi ai dati utente.
Identificazione e autenticazione	Garantisce l'identificazione non ambiguo degli utenti autorizzati e la corretta associazione degli attributi di sicurezza agli utenti e ai soggetti.

(segue)

Tabella 20.3 Requisiti funzionali di sicurezza CC.

Classe	Descrizione
Gestione della sicurezza	Specifica la gestione degli attributi, dei dati e delle funzioni di sicurezza.
Privacy	Garantisce all'utente la protezione dalla rilevazione e dall'utilizzo improprio della propria identità da parte di altri utenti.
Protezione delle funzioni di sicurezza del TOE	Riguarda la protezione dei dati relativi alle funzioni di sicurezza del TOE (TSF), anziché dei dati utente. La classe concerne l'integrità e la gestione dei meccanismi e dati TSF.
Utilizzazione delle risorse	Riguarda la disponibilità delle risorse richieste, quali le capacità di elaborazione e di memorizzazione. Include i requisiti relativi a: tolleranza ai guasti, priorità dei servizi e allocazione delle risorse.
Accesso al TOE	Specifica i requisiti funzionali, oltre a quelli specificati per l'identificazione e l'autenticazione, per controllare l'apertura di una sessione utente. I requisiti di accesso al TOE determinano per esempio i limiti nel numero e nella portata delle sessioni utente, riportando informazioni storiche relative agli accessi e alle modifiche dei parametri di accesso.
Canali/percorsi fidati	Riguarda i percorsi di comunicazione fidati fra gli utenti e TSF, e fra TSF.

Tabella 20.4 Requisiti di garanzia di sicurezza CC.

Classe	Descrizione
Gestione della configurazione	Richiede la protezione adeguata dell'integrità del TOE. Più in particolare, la gestione della configurazione garantisce che il TOE e la documentazione utilizzata per la valutazione siano quelle preparate per la distribuzione.
Consegna e impiego operativo	Riguarda le misure, procedure e standard per la sicurezza delle fasi di consegna, installazione e uso operativo del TOE. Ha lo scopo di garantire che la protezione di sicurezza offerta dal TOE non venga compromessa durante queste fasi.
Sviluppo	Concerne il raffinamento delle TFS dalle specifiche definite negli ST all'implementazione, oltre al mapping dai requisiti di sicurezza al più basso livello di rappresentazione.
Documenti di supporto	Riguarda l'impiego operativo sicuro del TOE da parte di utenti e amministratori.
Supporto al ciclo di sviluppo	Inerente al ciclo di vita del TOE, comprende definizione, strumenti e tecniche del ciclo di vita, sicurezza dell'ambiente di sviluppo e correzione degli errori identificati dagli utenti.
Verifiche	Riguarda la dimostrazione che il TOE soddisfa i propri requisiti funzionali. Le famiglie di questa classe sono attinenti all'ampiezza e alla profondità delle verifiche dello sviluppatore e ai requisiti per le verifiche indipendenti.

(segue)

Tabella 20.4 Requisiti di garanzia di sicurezza CC. *(continua)*

Classe	Descrizione
Valutazione della vulnerabilità	Definisce i requisiti per l'identificazione di eventuali lacune di sicurezza che potrebbero essere introdotte per costruzione, impiego, utilizzo improprio o configurazione scorretta del TOE. Le famiglie di questa classe sono attinenti all'identificazione delle vulnerabilità tramite analisi di canale nascosto, analisi della configurazione del TOE, esame della robustezza dei meccanismi delle funzioni di sicurezza e identificazione degli errori introdotti nella fase di sviluppo del TOE. La seconda famiglia riguarda la classificazione della sicurezza delle componenti del TOE. La terza e la quarta famiglia riguardano l'analisi dell'impatto delle modifiche sulla sicurezza e la certificazione di conformità alle procedure. Questa classe fornisce gli elementi di base per stabilire gli schemi di mantenimento delle garanzie di sicurezza.
Mantenimento delle garanzie di sicurezza	Fornisce i requisiti che si devono applicare dopo che il TOE è stato certificato rispetto ai CC. Questi requisiti hanno lo scopo di garantire che il TOE continuerà a soddisfare i propri obiettivi di sicurezza, nonostante le modifiche al TOE stesso o al suo ambiente.

Per esempio, la classe di requisiti funzionali per il supporto crittografico comprende due famiglie: gestione della chiave di crittografia ed elaborazione crittografica. Vi sono quattro componenti nella famiglia gestione della chiave di crittografia, utilizzate per specificare: l'algoritmo di generazione e la dimensione, il metodo di distribuzione, il metodo di accesso e il metodo di distruzione della chiave. Per ciascuna componente è possibile fare riferimento a uno standard per definire il requisito. Vi è una sola componente nella famiglia elaborazione crittografica; essa specifica un algoritmo e la dimensione della chiave sulla base del particolare standard assegnato.

Gli insiemi di componenti funzionali e di garanzia possono essere raggruppati in package riutilizzabili, che si sono rivelati utili per soddisfare gli obiettivi identificati. Un esempio di tale package sono le componenti funzionali richieste per il controllo di accesso.

Profili e obiettivi

I CC definiscono anche due tipi di documenti che possono essere generati utilizzando i requisiti definiti tramite CC.

- **Profili di protezione (PP – Protection Profiles):** definiscono un insieme di requisiti e obiettivi di sicurezza indipendenti dall'implementazione, per una categoria di prodotti o sistemi che soddisfano esigenze di sicurezza dei clienti similari. Un PP deve essere riutilizzabile e deve definire requisiti di provata utilità ed efficacia nel soddisfare gli obiettivi identificati. Il concetto di PP è stato sviluppato per supportare la definizione di standard funzionali e come ausilio per la formulazione delle specifiche per gli acquisti.
- **Obiettivi di sicurezza (ST – Security Targets):** contengono gli obiettivi e i requisiti di sicurezza di un particolare TOE e definiscono le misure funzionali e di garanzia offerte da tale TOE per soddisfare i requisiti definiti. L'ST può dichiarare la conformità con uno o più PP e costituisce la base per una valutazione. L'ST è fornito dal produttore o dallo sviluppatore.

La Figura 20.6 illustra la relazione fra requisiti da un lato e profili e obiettivi dall'altro. Per definire i requisiti del prodotto, l'utente può scegliere un insieme di componenti come PP. L'utente può anche fare riferimento a package predefiniti che raccolgono vari requisiti solitamente raggruppati all'interno di un documento di requisiti del prodotto. Analogamente il produttore o il progettista possono scegliere varie componenti e package per definire un ST.

La Figura 20.7 illustra ciò che nei documenti CC viene indicato come paradigma dei requisiti funzionali di sicurezza. Questa illustrazione è sostanzialmente basata sul concetto di monitor di riferimento ma adotta la terminologia e filosofia di progettazione dei CC.

20.4 Letture e siti Web consigliati

Un trattato classico sui firewall è [CHAP00]. Un altro classico, recentemente aggiornato, è [CHES00]. [LODI98], [OPPL97] e [BELL94] sono ottimi articoli generali su questo argomento. [WACK02] è un'eccellente panoramica sulle tecnologie e le politiche dei firewall. [AUDI04] e [WILS05] contengono utili informazioni sui firewall.

[GASS88] è un'ampia trattazione dei sistemi fidati. [PFLE03] e [GOLL99] sono anch'essi ottime trattazioni. [FELT03] e [OPPL05] forniscono utili informazioni sui sistemi fidati.

AUDI04 G. Audin. "Next-Gen Firewalls: What to Expect". *Business Communications Review*, Giugno 2004.

BELL94b S. Bellovin e W. Cheswick. "Network Firewalls". *IEEE Communications Magazine*, Settembre 1994.

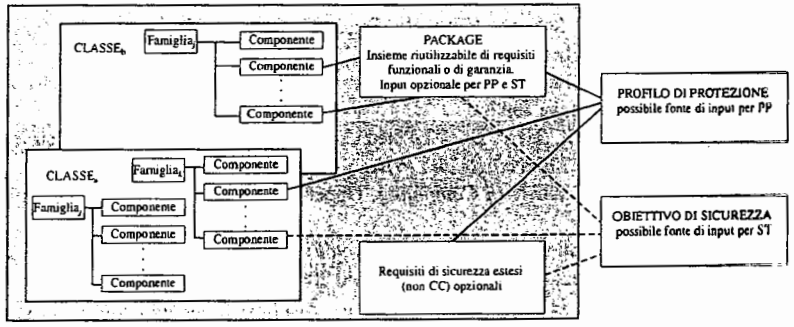


Figura 20.6 Organizzazione e definizione dei requisiti CC.

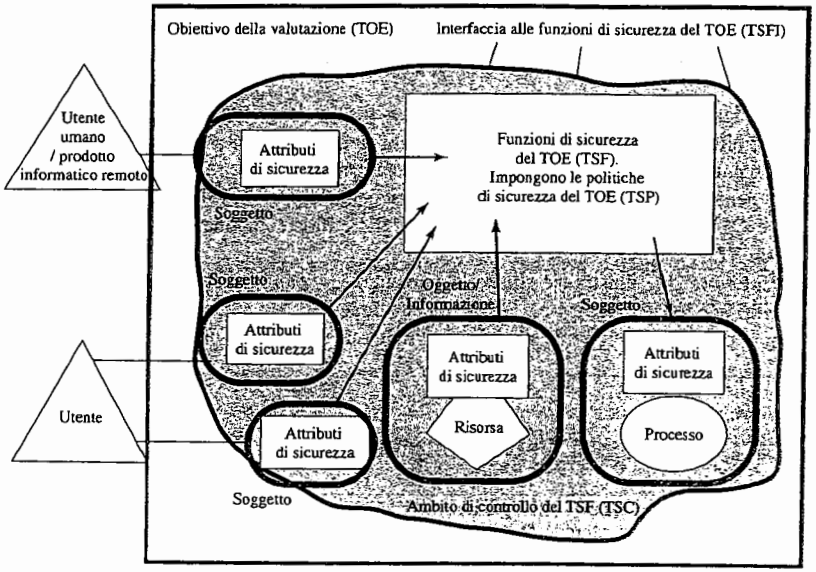


Figura 20.7 Paradigma dei requisiti funzionali di sicurezza.

CHAP00 D. Chapman e E. Zwicky. *Building Internet Firewalls*. Sebastopol, CA: O'Reilly, 2000.

CHES03 W. Cheswick e S. Bellovin. *Firewalls and Internet Security: Repelling the Wily Hacker*. Reading, MA: Addison-Wesley, 2003.

FELT03 E. Felten. "Understanding Trusted Computing: Will Its Benefits Outweigh Its Drawbacks?" *IEEE Security and Privacy*, Maggio/Giugno 2003.

GASS88 M. Gasser. *Building a Secure Computer System*. New York: Van Nostrand Reinhold, 1988.

GOLL99 D. Gollmann. *Computer Security*. New York: Wiley, 1999.

LODI98 S. Lodin e C. Schuba. "Firewalls Fend Off Invasions from the Net." *IEEE Spectrum*, Febbraio 1998.

OPPL97 R. Oppliger. "Internet Security: Firewalls and Beyond". *Communications of the ACM*, Maggio 1997.

OPPL05 R. Oppliger e R. Rytz. "Does Trusted Computing Remedy Computer Security Problems?" *IEEE Security and Privacy*, Marzo/Aprile 2005.

PFLE03 C. Pfleeger. *Security in Computing*. Upper Saddle River, NJ: Prentice Hall, 2003.

WACK02 J. Wack, K. Cutler e J Pole. *Guidelines on Firewalls and Firewall Policy*. NIST Special Publication SP 800-41, Gennaio 2002.

WILS05 J. Wilson. " The Future of the Firewall". *Business Communications Review*, Maggio 2005.

Siti Web consigliati

- **Firewall.com:** vari link a risorse software e riferimenti relativi ai firewall.
- **Trusted Computing Group:** Gruppo di produttori coinvolti nello sviluppo e nella diffusione di standard per sistemi fidati. Il sito comprende documenti di riferimento, specifiche e collegamenti ai siti dei produttori.
- **Common Criteria Portal:** Sito Web ufficiale del progetto Common Criteria.

20.5 Termini chiave, domande di ripasso e problemi

Termini chiave

Common Criteria (CC)
Diritti di accesso
Firewall
Firewall di ispezione a stati
Funzionalità consentite
Gateway a livello di circuito
Gateway a livello applicazione
Host bastione
Lista di controllo degli accessi (ACL)
Matrice di accesso
Monitor di riferimento
Oggetto
Router a filtraggio di pacchetti
Sicurezza multilivello
Sistema fidato
Soggetto

Domande di ripasso

20.1 Elencare tre obiettivi progettuali di un firewall.
20.2 Elencare quattro tecniche utilizzate dai firewall per controllare gli accessi e garantire il rispetto della politica di sicurezza.
20.3 Quali informazioni vengono utilizzate da un tipico router a filtraggio di pacchetti?
20.4 Quali sono i punti deboli dei router a filtraggio di pacchetti?
20.5 Qual è la differenza fra un router a filtraggio di pacchetti e un firewall di ispezione a stati?
20.6 Che cos'è un gateway a livello applicazione?
20.7 Che cos'è un gateway a livello di circuito?
20.8 Quali sono le differenze fra le tre configurazioni rappresentate nella Figura 20.2?
20.9 Nel contesto del controllo degli accessi, qual è la differenza fra un soggetto e un oggetto?

20.10 Qual è la differenza fra una lista di controllo degli accessi e un elenco dei permessi?
20.11 Quali sono le due regole applicate da un monitor di riferimento?
20.12 Quali proprietà deve avere per un monitor di riferimento?
20.13 Cosa sono i Common Criteria?

Problemi

20.1 Come indicato nel Paragrafo 20.1, una strategia per sconfiggere gli attacchi a frammentazione consiste nell'imporre che il primo frammento di un pacchetto IP contenga una parte dell'header di livello trasporto di dimensioni superiori a un limite minimo fissato. Se il primo pacchetto viene rifiutato, tutti i frammenti successivi possono essere rifiutati. Tuttavia, considerate le caratteristiche di IP, è possibile che i pacchetti arrivino disordinati. Di conseguenza, è possibile che un frammento intermedio passi attraverso il filtro prima che il frammento iniziale venga rifiutato. Come è possibile gestire questa situazione?

20.2 La dimensione in byte del payload nel primo frammento di un pacchetto IPv4 è: Lunghezza Totale – (4 x IHL). Se questo valore è inferiore al minimo richiesto (8 byte per TCP), questo frammento e l'intero pacchetto vengono rifiutati. Suggerire un metodo alternativo per ottenere il medesimo risultato utilizzando solamente il campo Fragment Offset.

20.3 La RFC 791, che specifica il protocollo IPv4, descrive un algoritmo di riassemblaggio che prevede che i vecchi frammenti possano essere sovrascritti da nuovi frammenti parzialmente sovrapposti. Con questo tipo di implementazione l'hacker potrebbe costruire una serie di pacchetti nel quale il primo frammento (con offset nullo) contiene informazioni innocue (e viene quindi accettato dai filtri) e qualche frammento successivo con offset non nullo riscrive e modifica parte del primo frammento, in particolare, per esempio, la porta di destinazione nell'header TCP. Questo secondo frammento verrebbe accettato dalla maggior parte delle implementazioni dei filtri perché non ha offset nullo. Suggerire un metodo che potrebbe essere utilizzato da un filtro di pacchetti per sconfiggere questo attacco.

20.4 La necessità della prima regola di un sistema sicuro multilivello è piuttosto ovvia. Qual è l'importanza della seconda regola?

20.5 Nella Figura 20.5, un anello della catena "copia e osserva in seguito" del cavallo di Troia è rotto. Vi sono due altre possibili strategie di attacco: Alice si connette e tenta di leggere direttamente la stringa oppure assegna un livello di sicurezza più elevato al file di servizio. Il monitor di riferimento è in grado di bloccare questi due attacchi?

Bibliografia

Abbreviazioni
ACM Association for Computing Machinery
IEEE Institute of Electrical and Electronics Engineers
NIST National Institute of Standards and Technology

ADAM90 Adams, C., and Tavares, S. "Generating and Counting Binary Bent Sequences." *IEEE Transactions on Information Theory*, 1990.

ADAM94 Adams, C. "Simple and Effective Key Scheduling for Symmetric Ciphers." *Proceedings, Workshop in Selected Areas of Cryptography, SAC '94*. 1994.

AKL83 Akl, S. "Digital Signatures: A Tutorial Survey." *Computer*, February 1983.

ALVA90 Alvare, A. "How Crackers Crack Passwords or What Passwords to Avoid." *Proceedings, UNIX Security Workshop II*, August 1990.

ANDE80 Anderson, J. *Computer Security Threat Monitoring and Surveillance.* Fort Washington, PA: James P. Anderson Co., April 1980.

AXEL00 Axelsson, S. "The Base-Rate Fallacy and the Difficulty of Intrusion Detection." *ACM Transactions and Information and System Security*, August 2000.

BACE00 Bace, R. *Intrusion Detection.* Indianapolis, IN: Macmillan Technical Publishing, 2000.

BACE01 Bace, R., and Mell, P. *Intrusion Detection Systems.* NIST Special Publication SP 800-31, November 2000.

BARK91 Barker, W. *Introduction to the Analysis of the Data Encryption Standard (DES).* Laguna Hills, CA: Aegean Park Press, 1991.

BAUE88 Bauer, D., and Koblentz, M. "NIDX—An Expert System for Real-Time Network Intrusion Detection." *Proceedings, Computer Networking Symposium*, April 1988.

BELL90 Bellovin, S., and Merritt, M. "Limitations of the Kerberos Authentication System." *Computer Communications Review*, October 1990.

BELL92 Bellovin, S. "There Be Dragons." *Proceedings, UNIX Security Symposium III*, September 1992.

BELL93 Bellovin, S. "Packets Found on an Internet." *Computer Communications Review*, July 1993.

BELL94 Bellovin, S., and Cheswick, W. "Network Firewalls." *IEEE Communications Magazine*, September 1994.

BELL96a Bellare, M., Canetti, R., and Krawczyk, H. "Keying Hash Functions for Message Authentication." *Proceedings, CRYPTO '96*, August 1996; New York: Springer-Verlag. An expanded version is available at http://www.cse.ucsd.edu/users/mihir.

BELL96b Bellare, M., Canetti, R., and Krawczyk, H. "The HMAC Construction." *CryptoBytes*, Spring 1996.

BELL97 Bellare, M., and Rogaway, P. "Collision-Resistant Hashing: Towards Making UOWHF's Practical." *Proceedings, CRYPTO '97*, 1997; New York: Springer-Verlag.

BERL84 Berlekamp, E. *Algebraic Coding Theory.* Laguna Hills, CA: Aegean Park Press, 1984.

BERS92 Berson, T. "Differential Cryptanalysis Mod 2^{32} with Applications to MD5." *Proceedings, EUROCRYPT '92*, May 1992; New York: Springer-Verlag.

BETH91 Beth, T., Frisch, M., and Simmons, G. eds. *Public-Key Cryptography: State of the Art and Future Directions.* New York: Springer-Verlag, 1991.

BIHA93 Biham, E., and Shamir, A. *Differential Cryptanalysis of the Data Encryption Standard.* New York: Springer-Verlag, 1993.

BIHA00 Biham, E., and Shamir, A. "Power Analysis of the Key Scheduling of the AES Candidates" Proceedings, Second AES Candidate Conference, 24 October 2000. http://csrc.nist.gov/encryption/aes/round1/conf2/aes2conf.htm.

BLAK99 Blake, I., Seroussi, G., and Smart, N. *Elliptic Curves in Cryptography.* Cambridge: Cambridge University Press, 1999.

BLOO70 Bloom, B. "Space/time Trade-offs in Hash Coding with Allowable Errors." *Communications of the ACM*, July 1970.

BLUM86 Blum, L., Blum, M., and Shub, M. "A Simple Unpredictable Pseudo-Random Number Generator." *SIAM Journal on Computing*, No. 2, 1986.

BOER93 Boer, B., and Bosselaers, A. "Collisions for the Compression Function of MD5." *Proceedings, EUROCRYPT '93*, 1993; New York: Springer-Verlag.

BOSS96 Bosselaers, A., Govaerts, R., and Vandewlle, J. "Fast Hashing on the Pentium." *Proceedings, Crypto '96*, August 1996; New York: Springer-Verlag.

BOSS97 Bosselaers, A., Dobbertin, H., and Preneel, B. "The RIPEMD-160 Cryptographic Hash Function." *Dr. Dobb's Journal*, January 1997.

BRIG79 Bright, H., and Enison, R. "Quasi-Random Number Sequences from Long-Period TLP Generator with Remarks on Application to Cryptography." *Computing Surveys*, December 1979.

BRYA88 Bryant, W. *Designing an Authentication System: A Dialogue in Four Scenes.* Project Athena document, February 1988. Available at http://web.mit.edu/kerberos/www/dialogue.html.

BURN97 Burn, R. *A Pathway to Number Theory.* Cambridge, England: Cambridge University Press, 1997.

CAMP92 Campbell, K., and Wiener, M. "Proof that DES is not a Group." *Proceedings, Crypto '92*, 1992; New York: Springer-Verlag.

CASS01 Cass, S. "Anatomy of Malice." *IEEE Spectrum*, November 2001.

CHAP95 Chapman, D., and Zwicky, E. *Building Internet Firewalls.* Sebastopol, CA: O'Reilly, 1995.

CHEN98 Cheng, P., et al. "A Security Architecture for the Internet Protocol." *IBM Systems Journal*, Number 1, 1998.

CHES97 Chess, D. "The Future of Viruses on the Internet." *Proceedings, Virus Bulletin International Conference*, October 1997.

CHES00 Cheswick, W., and Bellovin, S. *Firewalls and Internet Security: Repelling the Wily Hacker.* Reading, MA: Addison-Wesley, 2000.

COCK73 Cocks, C. *A Note on Non-Secret Encryption.* CESG Report, November 1973.

COHE94 Cohen, F. *A Short Course on Computer Viruses.* New York: Wiley, 1994.

COME00 Comer, D. *Internetworking with TCP/IP, Volume I: Principles, Protocols and Architecture.* Upper Saddle River, NJ: Prentice Hall, 2000.

COPP94 Coppersmith, D. "The Data Encryption Standard (DES) and Its Strength Against Attacks." *IBM Journal of Research and Development*, May 1994.

CRAN01 Crandall, R., and Pomerance, C. *Prime Numbers: A Computational Perspective.* New York: Springer-Verlag, 2001.

DAEM99 Daemen, J., and Rijmen, V. *AES Proposal: Rijndael, Version 2.* Submission to NIST, March 1999. http://csrc.nist.gov/encryption/aes.

DAEM01 Daemen, J., and Rijmen, V. "Rijndael: The Advanced Encryption Standard." *Dr. Dobb's Journal*, March 2001.

DAEM02 Daemen, J., and Rijmen, V. *The Design of Rijndael: The Wide Trail Strategy Explained.* New York, Springer-Verlag, 2000.

DAMG89 Damgard, I. "A Design Principle for Hash Functions." *Proceedings, CRYPTO '89*, 1989; New York: Springer-Verlag.

DAVI89 Davies, D., and Price, W. *Security for Computer Networks.* New York: Wiley, 1989.

DAVI93 Davies, C., and Ganesan, R. "BApasswd: A New Proactive Password Checker." *Proceedings, 16th National Computer Security Conference*, September 1993.

DAWS96 Dawson, E., and Nielsen, L. "Automated Cryptoanalysis of XOR Plaintext Strings." *Cryptologia*, April 1996.

DENN81 Denning, D. "Timestamps in Key Distribution Protocols." *Communications of the ACM*, August 1981.

DENN82 Denning, D. *Cryptography and Data Security.* Reading, MA: Addison-Wesley, 1982.

DENN83 Denning, D. "Protecting Public Keys and Signature Keys." *Computer*, February 1983.

DENN87 Denning, D. "An Intrusion-Detection Model." *IEEE Transactions on Software Engineering*, February 1987.

DESK92 Deskins, W. *Abstract Algebra.* New York: Dover, 1992.

DIFF76a Diffie, W., and Hellman, M. "New Directions in Cryptography." *Proceedings of the AFIPS National Computer Conference*, June 1976.

DIFF76b Diffie, W., and Hellman, M. "Multiuser Cryptographic Techniques." *IEEE Transactions on Information Theory*, November 1976.

DIFF77 Diffie, W., and Hellman, M. "Exhaustive Cryptanalysis of the NBS Data Encryption Standard." *Computer*, June 1977.

DIFF79 Diffie, W., and Hellman, M. "Privacy and Authentication: An Introduction to Cryptography." *Proceedings of the IEEE*, March 1979.

DIFF88 Diffie, W. "The First Ten Years of Public-Key Cryptography." *Proceedings of the IEEE*, May 1988. Reprinted in [SIMM92].

DOBB96a Dobbertin, H. "The Status of MD5 After a Recent Attack." *CryptoBytes*, Summer 1996.

DOBB96b Dobbertin, H., Bosselaers, A., and Preneel, B. "RIPEMD-160: A Strengthened Version of RIPEMD." *Proceedings, Third International Workshop on Fast Software Encryption*, 1996; New York: Springer-Verlag.

DORA99 Doraswamy, N., and Harkins, D. *IPSec*. Upper Saddle River, NJ: Prentice Hall, 1999.

DREW99 Drew, G. *Using SET for Secure Electronic Commerce*. Upper Saddle River, NJ: Prentice Hall, 1999.

EFF98 Electronic Frontier Foundation. *Cracking DES: Secrets of Encryption Research, Wiretap Politics, and Chip Design*. Sebastopol, CA: O'Reilly, 1998.

ELGA85 ElGamal, T. "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms." *IEEE Transactions on Information Theory*, July 1985.

ELLI70 Ellis, J. *The Possibility of Secure Non-Secret Digital Encryption*. CESG Report, January 1970.

ELLI99 Ellis, J. "The History of Non-Secret Encryption." *Cryptologia*, July 1999.

ENGE80 Enger, N., and Howerton, P. *Computer Security*. New York: Amacom, 1980.

ENGE99 Enge, A. *Elliptic Curves and Their Applications to Cryptography*. Norwell, MA: Kluwer Academic Publishers, 1999.

FEIS73 Feistel, H. "Cryptography and Computer Privacy." *Scientific American*, May 1973.

FEIS75 Feistel, H., Notz, W., and Smith, J. "Some Cryptographic Techniques for Machine-to-Machine Data Communications." *Proceedings of the IEEE*, November 1975.

FERN99 Fernandes, A. "Elliptic Curve Cryptography." *Dr. Dobb's Journal*, December 1999.

FLUH00 Fluhrer, S., and McGrew, D. "Statistical Analysis of the Alleged RC4 Key Stream Generator." *Proceedings, Fast Software Encryption 2000*, 2000.

FLUH01 Fluhrer, S., Mantin, I., and Shamir, A. "Weakness in the Key Scheduling Algorithm of RC4." *Proceedings, Workshop in Selected Areas of Cryptography*, 2001.

FORD95 Ford, W. "Advances in Public-Key Certificate Standards." *ACM SIGSAC Review*, July 1995.

FORR97 Forrest, S., Hofmeyr, S., and Somayaji, A. "Computer Immunology." *Communications of the ACM*, October 1997.

FRAN01 Frankel, S. *Demystifying the IPSec Puzzle*. Boston: Artech House, 2001.

FREE93 Freedman, D. "The Goods on Hacker Hoods." *Forbes ASAP*, 13 September 1993.

FUMY93 Fumy, S., and Landrock, P. "Principles of Key Management." *IEEE Journal on Selected Areas in Communications*, June 1993.

GARD72 Gardner, M. *Codes, Ciphers, and Secret Writing*. New York: Dover, 1972.

GARD77 Gardner, M. "A New Kind of Cipher That Would Take Millions of Years to Break." *Scientific American*, August 1977.

GARF97 Garfinkel, S., and Spafford, G. *Web Security & Commerce*. Cambridge, MA: O'Reilly and Associates, 1997.

GARR01 Garrett, P. *Making, Breaking Codes: An Introduction to Cryptology*. Upper Saddle River, NJ: Prentice Hall, 2001.

GASS88 Gasser, M. *Building a Secure Computer System*. New York: Van Nostrand Reinhold, 1988.

GAUD00 Gaudin, S. "The Omega Files." *Network World*, June 26, 2000.

GOLL99 Gollmann, D. *Computer Security*. New York: Wiley, 1999.

GONG92 Gong, L. "A Security Risk of Depending on Synchronized Clocks." *Operating Systems Review*, January 1992.

GONG93 Gong, L. "Variations on the Themes of Message Freshness and Replay." *Proceedings, IEEE Computer Security Foundations Workshop*, June 1993.

GRAH94 Graham, R., Knuth, D., and Patashnik, O. *Concrete Mathematics: A Foundation for Computer Science*. Reading, MA: Addison-Wesley, 1994.

HAMM91 Hamming, R. *The Art of Probability for Scientists and Engineers*. Reading, MA: Addison-Wesley, 1991.

HARL01 Harley, D., Slade, R., and Gattiker, U. *Viruses Revealed*. New York: Osborne/McGraw-Hill, 2001.

HEBE92 Heberlein, L., Mukherjee, B., and Levitt, K. "Internetwork Security Monitor: An Intrusion-Detection System for Large-Scale Networks." *Proceedings, 15th National Computer Security Conference*, October 1992.

HELD96 Held, G. *Data and Image Compression: Tools and Techniques*. New York: Wiley, 1996.

HERS75 Herstein, I. *Topics in Algebra*. New York: Wiley, 1975.

HEVI99 Hevia, A., and Kiwi, M. "Strength of Two Data Encryption Standard Implementations Under Timing Attacks." *ACM Transactions on Information and System Security*, November 1999.

HEYS95 Heys, H., and Tavares, S. "Avalanche Characteristics of Substitution-Permutation Encryption Networks." *IEEE Transactions on Computers*, September 1995.

HONE01 The Honeynet Project. *Know Your Enemy: Revealing the Security Tools, Tactics, and Motives of the Blackhat Community*. Reading, MA: Addison-Wesley, 2001.

HUIT98 Huitema, C. *IPv6: The New Internet Protocol*. Upper Saddle River, NJ: Prentice Hall, 1998.

IANS90 I'Anson, C., and Mitchell, C. "Security Defects in CCITT Recommendation X.509—The Directory Authentication Framework." *Computer Communications Review*, April 1990.

ILGU93 Ilgun, K. "USTAT: A Real-Time Intrusion Detection System for UNIX." *Proceedings, 1993 IEEE Computer Society Symposium on Research in Security and Privacy*, May 1993.

JAIN91 Jain, R. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. New York: Wiley, 1991.

JAVI91 Javitz, H., and Valdes, A. "The SRI IDES Statistical Anomaly Detector." *Proceedings, 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, May 1991.

JONE82 Jones, R. "Some Techniques for Handling Encipherment Keys." *ICL Technical Journal*, November 1982.

JUEN85 Jueneman, R., Matyas, S., and Meyer, C. "Message Authentication." *IEEE Communications Magazine*, September 1988.

JUEN87 Jueneman, R. "Electronic Document Authentication." *IEEE Network Magazine*, April 1987.

JURI97 Jurisic, A., and Menezes, A. "Elliptic Curves and Cryptography." *Dr. Dobb's Journal*, April 1997.

KAHN96 Kahn, D. *The Codebreakers: The Story of Secret Writing*. New York: Scribner, 1996.

KALI95 Kaliski, B., and Robshaw, M. "The Secure Use of RSA." *CryptoBytes*, Autumn 1995.

KALI96a Kaliski, B., and Robshaw, M. "Multiple Encryption: Weighing Security and Performance." *Dr. Dobb's Journal*, January 1996.

KALI96b Kaliski, B. "Timing Attacks on Cryptosystems." *RSA Laboratories Bulletin*, January 1996. http://www.rsasecurity.com/rsalabs.

KATZ00 Katzenbeisser, S., ed. *Information Hiding Techniques for Steganography and Digital Watermarking*. Boston: Artech House, 2000.

KEHN92 Kehne, A., Schonwalder, J., and Langendorfer, H. "A Nonce-Based Protocol for Multiple Authentications" *Operating Systems Review*, October 1992.

KELS98 Kelsey, J., Schneier, B., and Hall, C. "Cryptanalytic Attacks on Pseudorandom Number Generators." *Proceedings, Fast Software Encryption*, 1998. http://www.counterpane.com/pseudorandom_number.html.

KENT00 Kent, S. "On the Trail of Intrusions into Information Systems." *IEEE Spectrum*, December 2000.

KEPH97a Kephart, J., Sorkin, G., Chess, D., and White, S. "Fighting Computer Viruses." *Scientific American*, November 1997.

KEPH97b Kephart, J., Sorkin, G., Swimmer, B., and White, S. "Blueprint for a Computer Immune System." *Proceedings, Virus Bulletin International Conference*, October 1997.

KLEI90 Klein, D. "Foiling the Cracker: A Survey of, and Improvements to, Password Security." *Proceedings, UNIX Security Workshop II*, August 1990.

KNUD98 Knudsen, L., et al. "Analysis Method for Alledged RC4." *Proceedings, ASIACRYPT '98*, 1998.

KNUT97 Knuth, D. *The Art of Computer Programming, Volume 1: Fundamental Algorithms*. Reading, MA: Addison-Wesley, 1997.

KNUT98 Knuth, D. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Reading, MA: Addison-Wesley, 1998.

KOBL92 Koblas, D., and Koblas, M. "SOCKS." *Proceedings, UNIX Security Symposium III*, September 1992.

KOBL94 Koblitz, N. *A Course in Number Theory and Cryptography*.

KOCH96 Kocher, P. "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems." *Proceedings, Crypto '96*, August 1996.

KOCH98 Kocher, P., Jaffe, J., and Jun, B. Introduction to Differential Power Analysis and Related Attacks." http://www.cryptography.com/dpa/technical/ index.html.

KOHN78 Kohnfelder, L. *Towards a Practical Public-Key Cryptosystem*. Bachelor's Thesis, M.I.T., May 1978.

KOHL89 Kohl, J. "The Use of Encryption in Kerberos for Network Authentication." *Proceedings, Crypto '89*, 1989; New York: Springer-Verlag.

KOHL94 Kohl, J., Neuman, B., and Ts'o, T. "The Evolution of the Kerberos Authentication Service." in Brazier, F., and Johansen, D. *Distributed Open Systems*. Los Alamitos, CA: IEEE Computer Society Press, 1994. Available at http://web.mit.edu/kerberos/www/papers.html.

KONH81 Konheim, A. *Cryptography: A Primer*. New York: Wiley, 1981.

KORN96 Komer, T. *The Pleasures of Counting*. Cambridge, England: Cambridge University Press, 1996.

KUMA97 Kumar, I. *Cryptology*. Laguna Hills, CA: Aegean Park Press, 1997.

KUMA98 Kumanduri, R., and Romero, C. *Number Theory with Computer Applications*. Upper Saddle River, NJ: Prentice Hall, 1998.

KUMA98 Kumanduri, R., and Romero, C. *Number Theory with Computer Applications*. Upper Saddle River, NJ: Prentice Hall, 1998.

LAM92a Lam, K., and Gollmann, D. "Freshness Assurance of Authentication Protocols" *Proceedings, ESORICS '92*, 1992; New York: Springer-Verlag.

LAM92b Lam, K., and Beth, T. "Timely Authentication in Distributed Systems." *Proceedings, ESORICS '92*, 1992; New York: Springer-Verlag.

LE93 Le, A., Matyas. S., Johnson, D., and Wilkins, J. "A Public Key Extension to the Common Cryptographic Architecture." *IBM Systems Journal*, No. 3, 1993.

LEHM51 Lehmer, D. "Mathematical Methods in Large-Scale Computing." *Proceedings, 2nd Symposium on Large-Scale Digital Calculating Machinery*, Cambridge: Harvard University Press, 1951.

LEVE90 Leveque, W. *Elementary Theory of Numbers*. New York: Dover, 1990.

LEWA00 Lewand, R. *Cryptological Mathematics*. Washington, DC: Mathematical Association of America, 2000.

LEWI69 Lewis, P., Goodman, A., and Miller, J. "A Pseudo-Random Number Generator for the System/360." *IBM Systems Journal*, No. 2, 1969.

LIDL94 Lidl, R., and Niederreiter, H. *Introduction to Finite Fields and Their Applications*. Cambridge: Cambridge University Press, 1994.

LIPM00 Lipmaa, H., Rogaway, P., and Wagner, D. "CTR Mode Encryption." *NIST First Modes of Operation Workshop*, October 2000. http://csrc.nist.gov/ encryption/modes.

LODI98 Lodin, S., and Schuba, C. "Firewalls Fend Off Invasions from the Net." *IEEE Spectrum*, February 1998.

LUNT88 Lunt, T., and Jagannathan, R. "A Prototype Real-Time Intrusion-Detection Expert System." *Proceedings, 1988 IEEE Computer Society Symposium on Research in Security and Privacy*, April 1988.

MACG97 Macgregor, R., Ezvan, C., Liguori, L., and Han, J. *Secure Electronic Transactions: Credit Card Payment on the Web in Theory and Practice*. IBM RedBook SG24-4978-00, 1997. Available at www.redbooks.ibm.com.

MADS93 Madsen, J. "World Record in Password Checking." *Usenet, comp.security.misc newsgroup*, August 18, 1993.

MANT01 Mantin, I., Shamir, A. "A Practical Attack on Broadcast RC4." *Proceedings, Fast Software Encryption*, 2001.

MARK97 Markham, T. "Internet Security Protocol." *Dr. Dobb's Journal*, June 1997.

MATS93 Matsui, M. "Linear Cryptanalysis Method for DES Cipher." *Proceedings, EUROCRYPT '93*, 1993; New York: Springer-Verlag.

MATY91a Matyas, S. "Key Handling with Control Vectors." *IBM Systems Journal*, No. 2, 1991.

MATY91b Matyas, S., Le, A., and Abraham, D. "A Key-Management Scheme Based on Control Vectors." *IBM Systems Journal*, No. 2, 1991.

MCHU00 McHugh, J., Christie, A., and Allen, J. "The Role of Intrusion Detection Systems." *IEEE Software*, September/October 2000.

MEIN01 Meinel, C. "Code Red for the Web." *Scientific American*, October 2001.

MENE97 Menezes, A., Oorshcot, P., and Vanstone, S. *Handbook of Applied Cryptography.* Boca Raton, FL: CRC Press, 1997.

MERK79 Merkle, R. *Secrecy, Authentication, and Public Key Systems.* Ph.D. Thesis, Stanford University, June 1979.

MERK81 Merkle, R., and Hellman, M. "On the Security of Multiple Encryption." *Communications of the ACM*, July 1981.

MERK89 Merkle, R. "One Way Hash Functions and DES." *Proceedings, CRYPTO '89*, 1989; New York: Springer-Verlag.

MEYE82 Meyer, C., and Matyas, S. *Cryptography: A New Dimension in Computer Data Security.* New York: Wiley, 1982.

MEYE88 Meyer, C., and Schilling, M. "Secure Program Load with Modification Detection Code." *Proceedings, SECURICOM 88*, 1988.

MILL75 Miller, G. "Riemann's Hypothesis and Tests for Primality." *Proceedings of the Seventh Annual ACM Symposium on the Theory of Computing*, May 1975.

MILL88 Miller, S., Neuman, B., Schiller, J., and Saltzer, J. "Kerberos Authentication and Authorization System." *Section E.2.1, Project Athena Technical Plan*, M.I.T. Project Athena, Cambridge, MA. 27 October 1988.

MILL98 Miller, S. *IPv6: The New Internet Protocol.* Upper Saddle River, NJ: Prentice Hall, 1998.

MIST96 Mister, S., and Adams, C. "Practical S-Box Design." *Proceedings, Workshop in Selected Areas of Cryptography, SAC '96.* 1996.

MIST98 Mister, S., and Tavares, S. "Cryptanalysis of RC4-Like Ciphers." *Proceedings, Workshop in Selected Areas of Cryptography, SAC '98.* 1998.

MITC90 Mitchell, C., Walker, M., and Rush, D. "CCITT/ISO Standards for Secure Message Handling." *IEEE Journal on Selected Areas in Communications*, May 1989.

MITC92 Mitchell, C., Piper, F., and Wild, P. "Digital Signatures." In [SIMM92].

MIYA90 Miyaguchi, S., Ohta, K., and Iwata, M. "Confirmation that Some Hash Functions Are Not Collision Free." *Proceedings, EUROCRYPT '90*, 1990; New York: Springer-Verlag.

MUFT89 Muftic, S. *Security Mechanisms for Computer Networks.* New York: Ellis Horwood, 1989.

MURH98 Murhammer, M., et al. *TCP/IP: Tutorial and Technical Overview.* Upper Saddle River: NJ: Prentice Hall, 1998.

MURP90 Murphy, S. "The Cryptanalysis of FEAL-4 with 20 Chosen Plaintexts." *Journal of Cryptology*, No. 3, 1990.

MYER91 Myers, L. *Spycomm: Covert Communication Techniques of the Underground.* Boulder, CO: Paladin Press, 1991.

NACH97 Nachenberg, C. "Computer Virus-Antivirus Coevolution." *Communications of the ACM*, January 1997.

NECH92 Nechvatal, J. "Public Key Cryptography." In [SIMM92].

NECH00 Nechvatal, J., et al. *Report on the Development of the Advanced Encryption Standard.* National Institute of Standards and Technology. October 2, 2000.

NEED78 Needham, R., and Schroeder, M. "Using Encryption for Authenti-cation in Large Networks of Computers." *Communications of the ACM*, December 1978.

NEUM90 Neumann, P. "Flawed Computer Chip Sold for Years." *RISKS-FORUM Digest*, Vol. 10, No. 54, October 18, 1990.

NEUM93a Neuman, B., and Stubblebine, S. "A Note on the Use of Timestamps as Nonces." *Operating Systems Review*, April 1993.

NEUM93b Neuman, B. "Proxy-Based Authorization and Accounting for Distributed Systems." *Proceedings of the 13th International Conference on Distributed Computing Systems*, May 1993.

NICH96 Nichols, R. *Classical Cryptography Course.* Laguna Hills, CA: Aegean Park Press, 1996.

NICH99 Nichols, R. ed. *ICSA Guide to Cryptography.* New York: McGraw-Hill, 1999.

NIST97 National Institute of Standards and Technology. "Request for Candidate Algorithm Nominations for the Advanced Encryption Standard." Federal Register, September 12, 1997.

ODLY95 Odlyzko, A. "The Future of Integer Factorization." *CryptoBytes*, Summer 1995.

OORS90 Oorschot, P., and Wiener, M. "A Known-Plaintext Attack on Two-Key Triple Encryption." *Proceedings, EUROCRYPT '90*, 1990; New York: Springer-Verlag.

OORS94 Oorschot, P., and Wiener, M. "Parallel Collision Search with Application to Hash Functions and Discrete Logarithms." *Proceedings, Second ACM Conference on Computer and Communications Security*, 1994.

OPPL97 Oppliger, R. "Internet Security: Firewalls and Beyond." *Communications of the ACM*, May 1997.

ORE67 Ore, O. *Invitation to Number Theory.* Washington, DC: The Mathematical Association of America, 1967,

PARK88 Park, S., and Miller, K. "Random Number Generators: Good Ones Are Hard to Find." *Communications of the ACM*, October 1988.

PFLE97 Pfleeger, C. *Security in Computing.* Upper Saddle River, NJ: Prentice Hall, 1997.

PIAT91 Piattelli-Palmarini, M. "Probability: Neither Rational nor Capricious." *Bostonia*, March 1991.

POHL81 Pohl, I., and Shaw, A. *The Nature of Computation: An Introduction to Computer Science.* Rockville, MD: Computer Science Press, 1981.

POPE79 Popek, G., and Kline, C. "Encryption and Secure Computer Networks." *ACM Computing Surveys*, December 1979.

PORR92 Porras, P. *STAT: A State Transition Analysis Tool for Intrusion Detection.* Master's Thesis, University of California at Santa Barbara, July 1992.

PREN96 Preneel, B., and Oorschot, P. "On the Security of Two MAC Algorithms." *Lecture Notes in Computer Science 1561; Lectures on Data Security*, 1999; New York: Springer-Verlag.

PREN97 Preneel, B., Bosselaers, A., and Dobbertin, H. "The Cryptographic Hash Function RIPEMD-160." *CryptoBytes*, Autumn 1997.

PREN99 Preneel, B. "The State of Cryptographic Hash Functions." *Proceedings, EUROCRYPT '96*, 1996; New York: Springer-Verlag.

PROC01 Proctor, P. *The Practical Intrusion Detection Handbook.* Upper Saddle River, NJ: Prentice Hall, 2001.

RABI78 Rabin, M. "Digitalized Signatures." In *Foundations of Secure Computation*, DeMillo, R., Dobkin, D., Jones, A., and Lipton, R., eds. New York: Academic Press, 1978.

RABI80 Rabin, M. "Probabilistic Algorithms for Primality Testing." *Journal of Number Theory*, December 1980.

RAND55 Rand Corporation. *A Million Random Digits.* New York: The Free Press, 1955. http://www.rand.org/publications/classics/randomdigits.

RESC01 Rescorla, E. *SSL and TLS: Designing and Building Secure Systems.* Reading, MA: Addison-Wesley, 2001.

RIBE96 Ribenboim, P. *The New Book of Prime Number Records.* New York: Springer-Verlag, 1996.

RIVE78 Rivest, R., Shamir, A., and Adleman, L. "A Method for Obtaining Digital Signatures and Public Key Cryptosystems." *Communications of the ACM*, February 1978.

RIVE90 Rivest, R. "The MD4 Message Digest Algorithm." *Proceedings, Crypto '90*, August 1990; New York: Springer-Verlag.

RIVE94 Rivest, R. "The RC5 Encryption Algorithm." *Proceedings, Second International Workshop on Fast Software Encryption*, December 1994; New York: Springer-Verlag.

RIVE95 Rivest, R. "The RC5 Encryption Algorithm." *Dr. Dobb's Journal*, January 1995.

ROBS95a Robshaw, M. *Stream Ciphers.* RSA Laboratories Technical Report TR-701, July 1995. http://www.rsasecurity.com/rsalabs/index.html.

ROBS95b Robshaw, M. *Block Ciphers.* RSA Laboratories Technical Report TR-601, August 1995. http://www.rsasecurity.com/rsalabs/index.html.

ROBS95c Robshaw, M. *MD2, MD4, MD5, SHA and Other Hash Functions.* RSA Laboratories Technical Report TR-101, July 1995. http://www.rsasecurity.com/ rsalabs/index.html.

ROSE00 Rosen, K. *Elementary Number Theory and its Applications.* Reading, MA: Addison-Wesley, 2000.

ROSI99 Rosing, M. *Implementing Ellipt Curve Cryptography.* Greeenwich, CT: Manning Publications, 1999.

RUBI97 Rubin, A. "An Experience Teaching a Graduate Course in Cryptography." *Cryptologia*, April 1997.

RUEP92 Rueppel, T. "Stream Ciphers." In [SIMM92].

SAFF93 Safford, D., Schales, D., and Hess, D. "The TAMU Security Package: An Ongoing Response to Internet Intruders in an Academic Environment." *Proceedings, UNIX Security Symposium IV*, October 1993.

SAUE81 Sauer, C., and Chandy, K. *Computer Systems Performance Modeling.* Englewood Cliffs, NJ: Prentice Hall, 1981.

SCHA96 Schaefer, E. "A Simplified Data Encryption Standard Algorithm." *Cryptologia*, January 1996.

SCHN91 Schnorr, C. "Efficient Signatures for Smart Card." *Journal of Cryptology*, No. 3, 1991.

SCHN93 Schneier, B. "Description of a New Variable-Length Key, 64-bit Block Cipher (Blowfish)." *Proceedings, Workshop on Fast Software Encryption*, December 1993; New York: Springer-Verlag.

SCHN94 Schneier, R. "The Blowfish Encryption Algorithm." *Dr. Dobb's Journal*, April 1994.

SCHN96 Schneier, B. *Applied Cryptography.* New York: Wiley, 1996.

SCHN00 Schneier, B. *Secrets and Lies: Digital Security in a Networked World.* New York: Wiley 2000.

SHAN49 Shannon, C. "Communication Theory of Secrecy Systems." *Bell Systems Technical Journal*, No. 4, 1949.

SIMM92 Simmons, G., ed. *Contemporary Cryptology: The Science of Information Integrity.* Piscataway, NJ: IEEE Press, 1992.

SIMM93 Simmons, G. "Cryptology." *Encyclopaedia Britannica,* 15th ed., 1993.

SIMO95 Simovits, M. *The DES: An Extensive Documentation and Evaluation.* Laguna Hills, CA: Aegean Park Press, 1995.

SING99 Singh, S. *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography.* New York: Anchor Books, 1999.

SINK66 Sinkov, A. *Elementary Cryptanalysis: A Mathematical Approach.* Washington, DC: The Mathematical Association of America, 1966.

SMIT97 Smith, R. *Internet Cryptography.* Reading, MA: Addison-Wesley, 1997.

SNAP91 Snapp, S., et al. "A System for Distributed Intrusion Detection." *Proceedings, COMPCON Spring '91*, 1991.

SPAF92a Spafford, E. "Observing Reusable Password Choices." *Proceedings, UNIX Security Symposium III*, September 1992.

SPAF92b Spafford, E. "OPUS: Preventing Weak Password Choices." *Computers and Security*, No. 3, 1992.

STAL00 Stallings, W. *Data and Computer Communications. Sixth Edition.* Upper Saddle River, NJ: Prentice Hall, 2000.

STAL02 Stallings, W. "The Advanced Encryption Standard." *Cryptologia*, to appear.

STEI88 Steiner, J., Neuman, C., and Schiller, J. "Kerberos: An Authentication Service for Open Networked Systems." *Proceedings of the Winter 1988 USENIX Conference*, February 1988.

STEP93 Stephenson, P. "Preventive Medicine." *LAN Magazine*, November 1993.

STER92 Sterling, B. *The Hacker Crackdown: Law and Disorder on the Electronic Frontier.* New York: Bantam, 1992.

STEV94 Stevens, W. *TCP/IP Illustrated, Volume 1: The Protocols.* Reading, MA: Addison-Wesley, 1994.

STIN02 Stinson, D. *Cryptography: Theory and Practice.* Boca Raton, FL: CRC Press, 2002.

STOL88 Stoll, C. "Stalking the Wily Hacker." *Communications of the ACM*, May 1988.

STOL89 Stoll, C. *The Cuckoo's Egg.* New York: Doubleday, 1989.

THOM84 Thompson, K. "Reflections on Trusting Trust (Deliberate Software Bugs)." *Communications of the ACM*, August 1984.

TIME90 Time, Inc. *Computer Security, Understanding Computers Series.* Alexandria, VA: Time-Life Books, 1990.

TIPP27 Tippett, L. *Random Sampling Numbers.* Cambridge, England: Cambridge University Press, 1927.

TOUC95 Touch, J. "Performance Analysis of MD5." *Proceedings, SIGCOMM '95*, October 1995.

TSUD92 Tsudik, G. "Message Authentication with One-Way Hash Functions." *Proceedings, INFOCOM '92*, May 1992.

TUCH79 Tuchman, W. "Hellman Presents No Shortcut Solutions to DES." *IEEE Spectrum*, July 1979.

TUNG99 Tung, B. *Kerberos: A Network Authentication System.* Reading, MA: Addison-Wesley, 1999.

VACC89 Vaccaro, H., and Liepins, G. "Detection of Anomalous Computer Session Activity." *Proceedings of the IEEE Symposium on Research in Security and Privacy*, May 1989.

VOYD83 Voydock, V., and Kent, S. "Security Mechanisms in High-Level Network Protocols." *Computing Surveys*, June 1983.

WACK02 Wack, J., Cutler, K., and Pole, J. *Guidelines on Firewalls and Firewall Policy.* NIST Special Publication SP 800-41, January 2002.

WAYN93 Wayner, P. "Should Encryption Be Regulated?" *Byte*, May 1993.

WAYN96 Wayner, P. *Disappearing Cryptography.* Boston: AP Professional Books, 1996.

WEBS86 Webster, A., and Tavares, S. "On the Design of S-Boxes." *Proceedings, Crypto '85*, 1985; New York: Springer-Verlag.

WIEN90 Wiener, M. "Cryptanalysis of Short RSA Secret Exponents." *IEEE Transactions on Information Theory*, vol. IT-36, 1990.

WEIS93 Weiss, J., and Schremp, D. "Putting Data on a Diet." *IEEE Spectrum*, August 1993.

WILL76 Williamson, M. *Thoughts on Cheaper Non-Secret Encryption.* CESG Report, August 1976.

WOO92a Woo, T., and Lam, S. "Authentication for Distributed Systems." *Computer*, January 1992.

WOO92b Woo, T., and Lam, S. " 'Authentication' Revisited." *Computer*, April 1992.

YIN97 Yin, Y. "The RC5 Encryption Algorithm: Two Years On." *CryptoBytes*, Winter 1997.

YUVA79 Yuval, G. "How to Swindle Rabin." *Cryptologia*, July 1979.

ZENG91 Zeng. K., Yang, C., Wei, D., and Rao, T. "Pseudorandom Bit Generators in Stream-Cipher Cryptography." *Computer*, February 1991.

ZIV77 Ziv, J., and Lempel, A. "A Universal Algorithm for Sequential Data Compression." *IEEE Transactions on Information Theory*, May 1977.

Indice analitico

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

X

Z

Finito di stampare
nel mese di ottobre 2006

W. Stallings
CRITTOGRAFIA
E SICUREZZA DELLE RETI
Seconda edizione
© McGraw-Hill Companies, S.r.l.
88 386 6377-7