Key Exchange

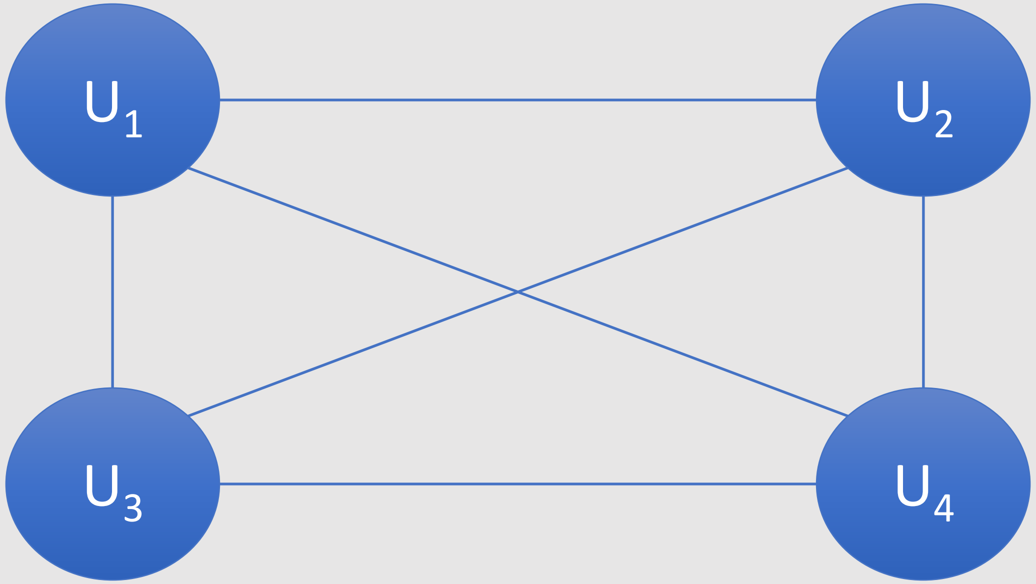
Outline

- Trusted 3rd Parties
- Merkle Puzzles
- The Diffie-Hellman Protocol

Trusted 3rd Parties

Key Management

Problem: **n** users. Storing mutual secret keys is difficult

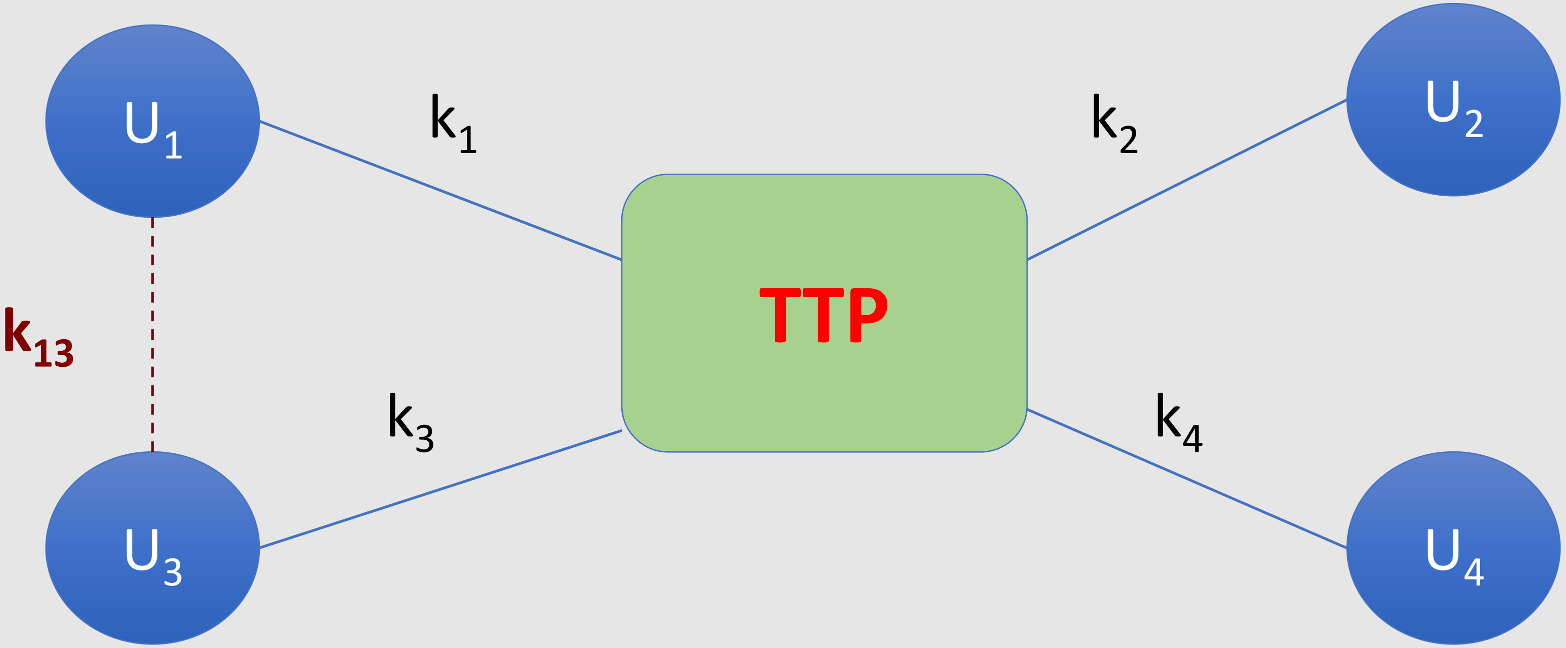


O(n) keys per user

O(n²) keys in total

A Better Solution

Online **Trusted 3rd Party** (TTP)



Every user only remembers **ONE key**

Generating keys: A toy protocol

Alice wants a shared key with Bob. **Eavesdropping** security **only**.

Bob **(k_B)** | **Alice** **(k_A)** | **TTP**

"Alice wants key with Bob"

choose random k_{AB}

$E(k_A, \text{"Alice, Bob"} \,\|\, k_{AB})$;

ticket = $E(k_B, \text{"Alice, Bob"} \,\|\, k_{AB})$

ticket

k_{AB} k_{AB}

(E,D) a CPA-secure cipher

Generating keys: A toy protocol

Alice wants a shared key with Bob. Eavesdropping security only.

Eavesdropper sees: $E(k_A,$ "A, B" ll $k_{AB})$; $E(k_B,$ "A, B" ll $k_{AB})$

(E,D) is CPA-secure $\Rightarrow$ eavesdropper learns nothing about k_{AB}

Note: **TTP needed for every key exchange, knows all session keys.**

(basis of Kerberos system)

Key Question

Can we generate shared keys **without** an **online** trusted 3rd party?

Answer: **yes!**

Starting point of public-key cryptography:

- Merkle (1974),
- Diffie-Hellman (1976),
- RSA (1977)
- …

Merkle Puzzles

Key exchange without an online TTP?

- Goal: Alice and Bob want a shared key, unknown to eavesdropper
- Security against **eavesdropping only** (**no tampering**)

- **Can this be done using generic symmetric crypto?**

Merkle Puzzles (1974)

Answer: yes, but **very inefficient**

Main tool: **"puzzles"**

- Puzzles: Problems that can be solved with "some effort"
- Example:
  - $E(k,m)$ a symmetric cipher with $k \in \{0,1\}^{128}$
  - **puzzle = E(P, "message")** where $\mathbf{P = 0^{96} \,\|\, b_1 \ldots b_{32}}$
  - To "solve" a puzzle, find **P** by trying all $\mathbf{2^{32}}$ possibilities

Merkle Puzzles

<u>Alice</u>:

- Prepare **2^{32} puzzles:**
  - For i = 1, …, 2^{32} choose random $P_i \in \{0,1\}^{32}$ and random $x_i, k_i \in \{0,1\}^{128}$ $x_i \neq x_j$

    Set $\textbf{puzzle}_i \longleftarrow E(\,0^{96} \,\|\, P_i,\ \text{“Puzzle \# ”} \,\|\, x_i \,\|\, k_i\,)$
  - Send $\textbf{puzzle}_1, \ldots, \textbf{puzzle}_{2^{32}}$ to Bob.

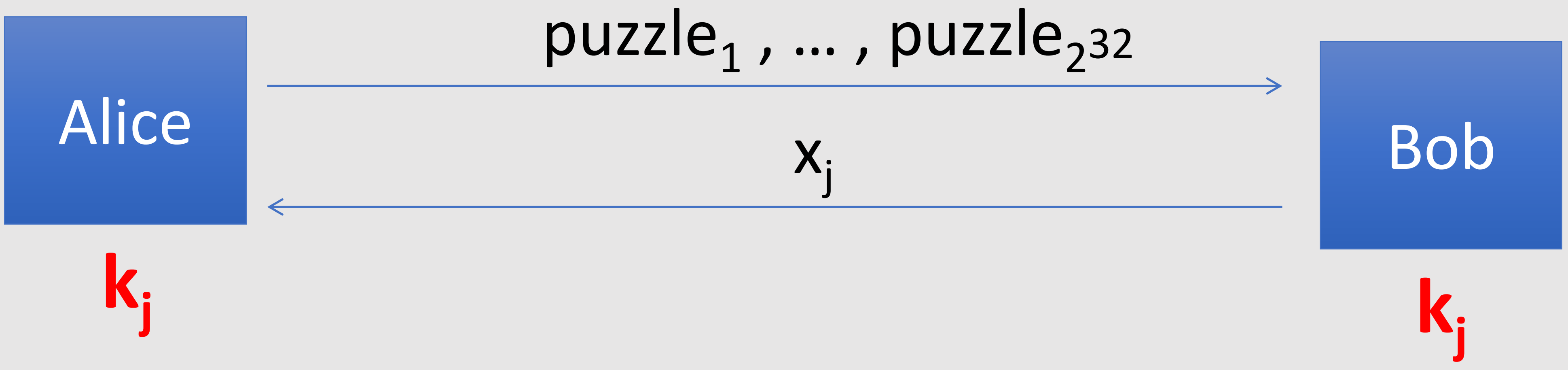
<u>Bob</u>:

- Choose a **random puzzle**$_j$ and **solve** it. Obtain (x_j, k_j) and use k_j as shared secret.
- Send x_j to Alice.

<u>Alice</u>:

- Lookup puzzle with number x_j .
- Use k_j as shared secret.

In a figure



Alice's work: **O(2^{32})** (prepare 2^{32} puzzles) in general **O(n)**

Bob's work: **O(2^{32})** (solve **one** puzzle) in general **O(n)**

Eavesdropper's work: **O(2^{64})** (solve **2^{32}** puzzles) in general **O(n^2)**

Impossibility Result

Can we achieve a better gap using a general symmetric cipher?

Answer: **unknown**

The Diffie-Hellman Protocol

Key exchange without an online TTP?

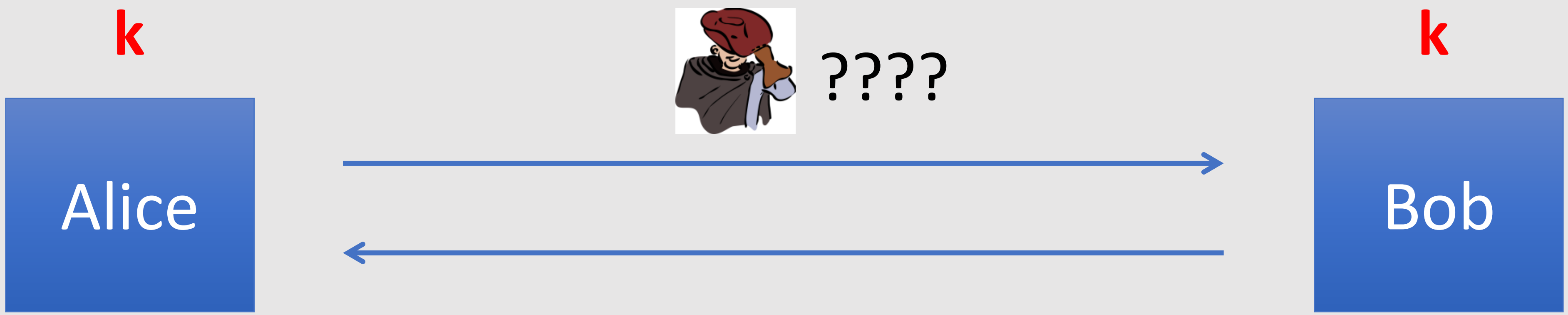
- Goal: Alice and Bob want a shared key, unknown to eavesdropper
- Security against **eavesdropping only** (**no tampering**)

- **Can this be done with an exponential gap?**

The Diffie-Hellman Protocol

High-level idea:

- Alice and Bob **do NOT share any secret information beforehand**
- Alice and Bob exchange messages
- After that, Alice and Bob have agreed on a shared secret key **k**
- **k** unknown to eavesdropper



The Diffie-Hellman Protocol

(Security) Based on the **Discrete Logarithm** Problem:

Given

- **g**
- **p**
- **$g^k \bmod p$**

Find k

The Diffie-Hellman Protocol

Fix a large prime **p** (e.g., 600 digits)

Fix an integer **g** in {2, …, p-2}

Alice

Choose random **a** in {1,…,p-2}

Bob

Choose random **b** in {1,…,p-2}

$\mathbf{g^a \pmod{p}}$ →

← $\mathbf{g^b \pmod{p}}$

Alice computes

Bob computes

$$(g^b)^a \pmod{p} \quad = \quad g^{ab} \pmod{p} \quad = \quad (g^a)^b \pmod{p}$$

SECRET KEY

Security

Eavesdropper sees: **p**, **g**, **g^a (mod p)**, and **g^b (mod p)**

Can she compute **g^{ab} (mod p)** ??

How hard is the DH function mod p?

Suppose prime **p** is **n** bits long.

Best known algorithm (GNFS): run time exp($\tilde{O}(\sqrt[3]{n})$)

Insecure against man-in-the-middle

As described, the protocol is insecure against **active** attacks

Alice	**MiTM**	**Bob**
$A = g^a \pmod p$ $\longrightarrow$	a'	$A' = g^{a'} \pmod p$ $\longrightarrow$
$\longleftarrow$ $B' = g^{b'} \pmod p$	b'	$\longleftarrow$ $B = g^b \pmod p$
$g^{ab'} \pmod p$	$g^{ab'}, g^{a'b} \pmod p$	$g^{a'b} \pmod p$