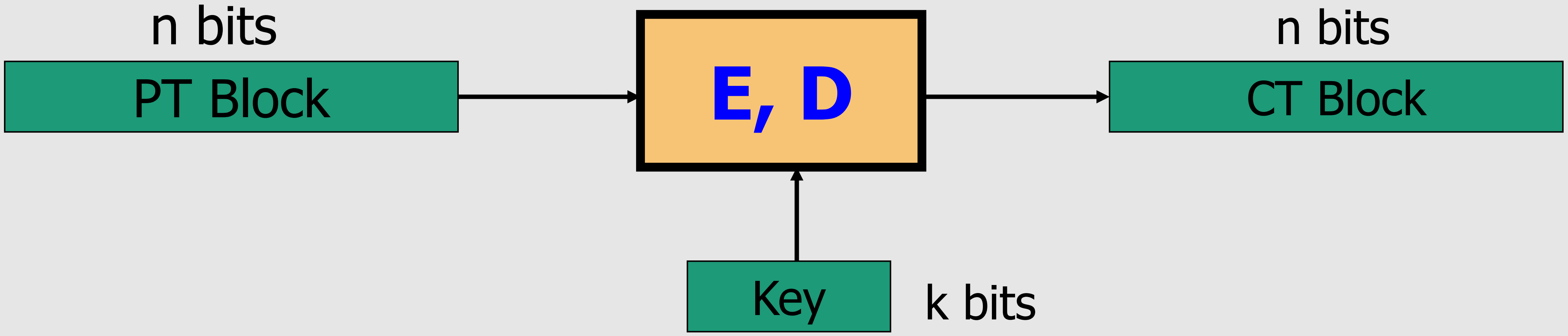
# Block Ciphers

# Outline

- Block Ciphers
- Pseudo Random Functions (PRFs)
- Pseudo Random Permutations (PRPs)
- DES – Data Encryption Standard
- AES – Advanced Encryption Standard
- PRF $\Rightarrow$ PRG
- PRG $\Rightarrow$ PRF

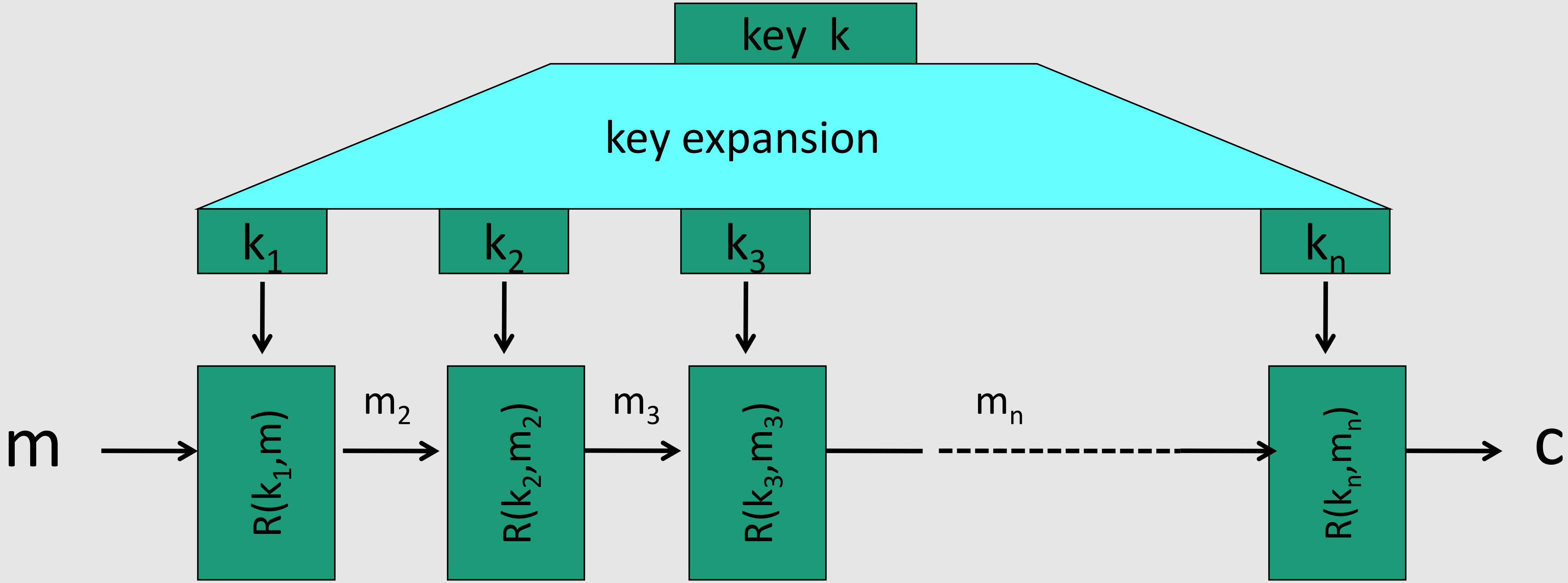
# Block Ciphers: crypto work horse




Canonical examples:

- **DES**: n= 64 bits, k = 56 bits
- **3DES**: n= 64 bits, k = 168 bits
- **AES**: n=128 bits, k = 128, 192, 256 bits

# Block Ciphers Built by Iteration




R(k,m) is called a **round function**

**for 3DES (n=48), for AES-128 (n=10)**

# Performance: Crypto++ 5.6.0 [ Wei Dai ]

AMD Opteron, 2.2 GHz ( Linux)

| | Cipher | Block/key size | Speed (MB/sec) |
|---|---|---|---|
| stream | RC4 | | 126 |
| | Salsa20/12 | | 643 |
| | Sosemanuk | | 727 |
| block | 3DES | 64/168 | 13 |
| | AES-128 | 128/128 | 109 |

# Abstractly: PRPs and PRFs

- **Pseudo Random Function (PRF)** defined over (K,X,Y):

  $$F: K \times X \rightarrow Y$$

  such that there exists **"efficient" algorithm** to evaluate F(k,x)

- **Pseudo Random Permutation (PRP)** defined over (K,X):

  $$E: K \times X \rightarrow X$$

  such that:

  1. There exists **"efficient"** **<u>deterministic</u>** algorithm to evaluate E(k,x)
  2. The function $E(k, \cdot)$ is **one-to-one** (for every k)
  3. There exists "efficient" **inversion algorithm** D(k,y)

# Running example

- Example PRPs: 3DES, AES, …

  AES: $K \times X \rightarrow X$ where $K = X = \{0,1\}^{128}$

  3DES: $K \times X \rightarrow X$ where $X = \{0,1\}^{64}$ , $K = \{0,1\}^{168}$

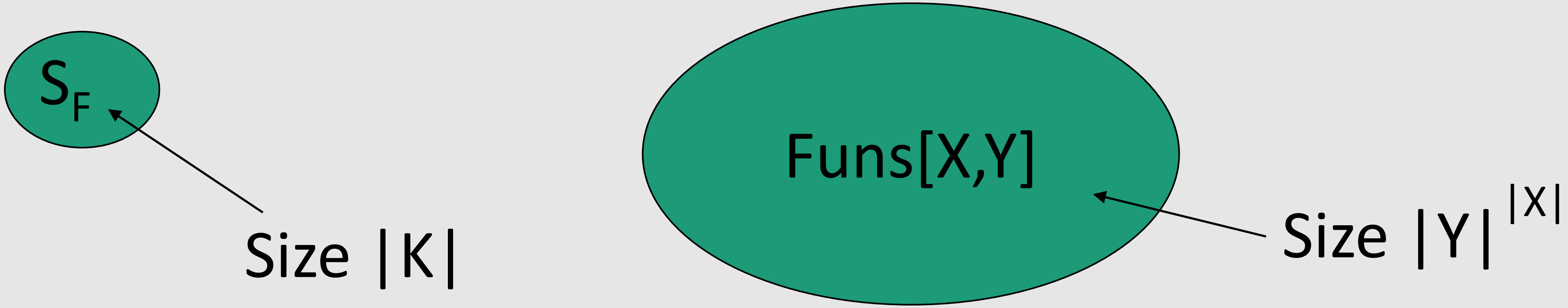
- Functionally, any **PRP is also a PRF.**
  - A PRP is a PRF where X=Y and is efficiently invertible.

# Secure PRFs

- Let $F: K \times X \rightarrow Y$ be a PRF. Set some notation:

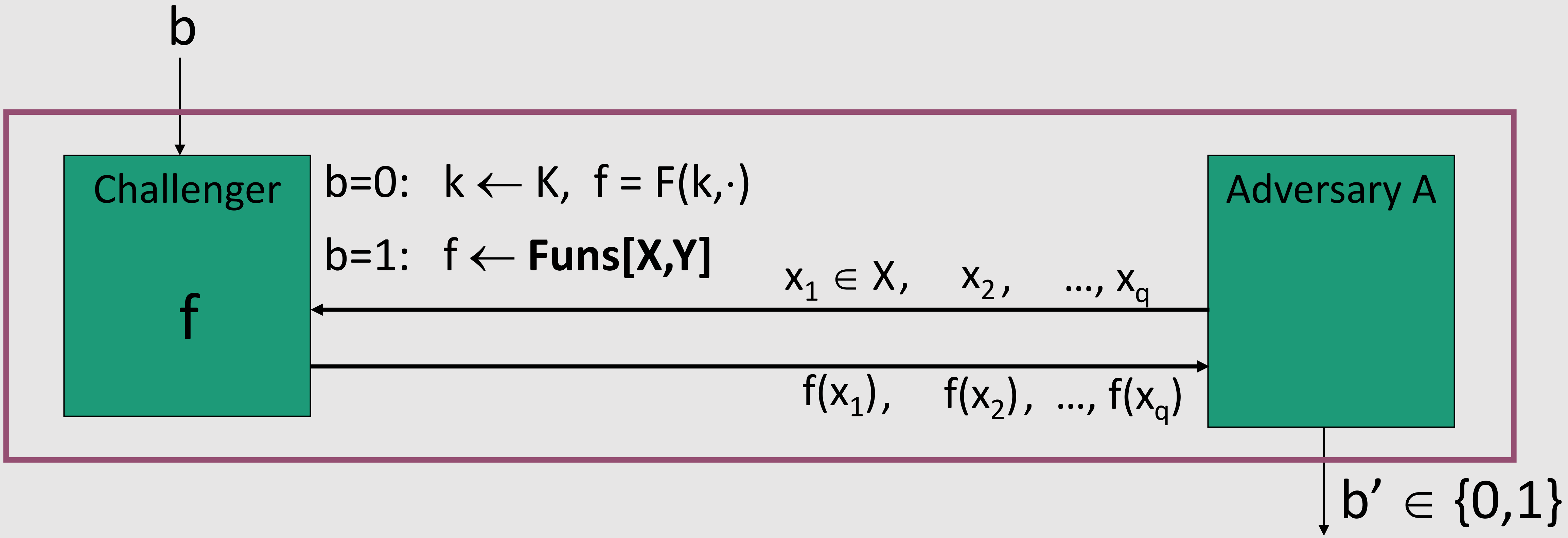
  $$\begin{cases} \text{Funs}[X,Y]: \text{ the set of } \textbf{all} \text{ functions from } X \text{ to } Y \\ S_F = \{ F(k,\cdot) \text{ s.t. } k \in K \} \subseteq \text{Funs}[X,Y] \end{cases}$$

- **Intuition:** a PRF is **secure** if a random function in Funs[X,Y] is "indistinguishable" from a random function in $S_F$



# Secure PRF: definition

- Consider a PRF $\mathbf{F} : \mathbf{K} \times \mathbf{X} \rightarrow \mathbf{Y}$. For b=0,1 define experiment EXP(b) as:




**Definition:** **F** is a **secure PRF** if for all "efficient" adversary A:

$$\mathbf{Adv_{PRF}[A,F]} := \left| \Pr[\mathrm{EXP}(0)=1] - \Pr[\mathrm{EXP}(1)=1] \right|$$ is "negligible".

# Secure PRPs (secure block cipher)

- Let E: $K \times X \rightarrow X$ be a PRP

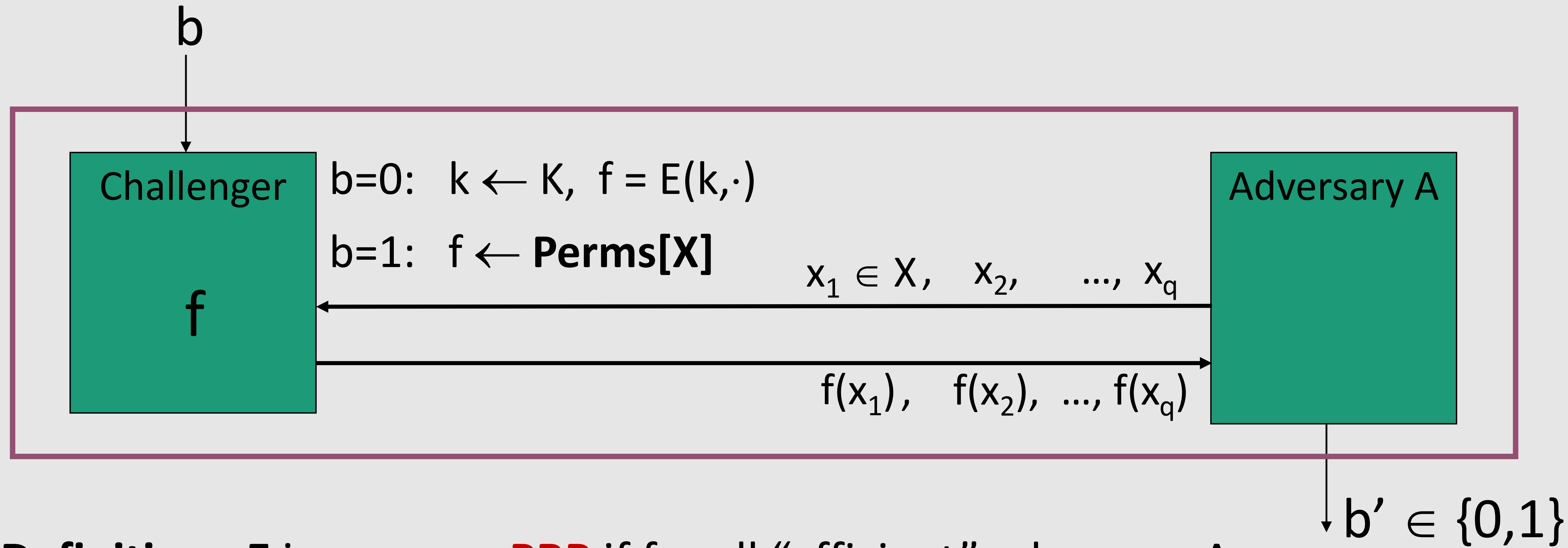
  Perms[X]: the set of **all one-to-one** functions from X to X (i.e., **permutations**)

  $S_E = \{ E(k,\cdot) \text{ s.t. } k \in K \} \subseteq \text{Perms}[X]$

- **Intuition:** a PRP is **secure** if a random function in Perms[X] is "indistinguishable" from a random function in $S_E$

# Secure PRP (secure block cipher)

- Consider a PRP $\mathbf{E} : \mathbf{K} \times \mathbf{X} \rightarrow \mathbf{X}$. For b=0,1 define experiment EXP(b) as:




**Definition.** **E** is a **secure PRP** if for all "efficient" adversary A:

$$\mathbf{Adv_{PRP}[A,E]} = \left| \Pr[\mathrm{EXP}(0)=1] - \Pr[\mathrm{EXP}(1)=1] \right| \text{ is "negligible".}$$

# Data Encryption Standard (DES)

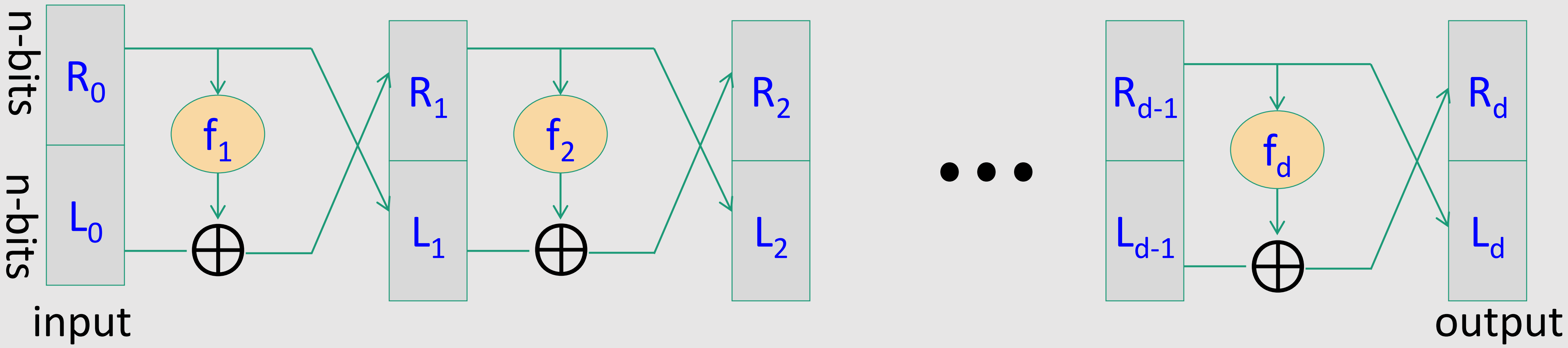
# The Data Encryption Standard (DES)

- Early 1970s: Horst Feistel designs Lucifer at IBM
  key-length = 128 bits ; block-length = 128 bits
- 1973: NBS (nowadays called NIST) asks for block cipher proposals.
  IBM submits variant of Lucifer.
- 1976: NBS adopts DES as a federal standard
  key-length = 56 bits ; block-length = 64 bits
- 1997: DES broken by exhaustive search
- 2000: NIST adopts Rijndael as AES to replace DES

# DES: core idea – Feistel Network

Given functions $f_1, \ldots, f_d$: $\{0,1\}^n \longrightarrow \{0,1\}^n$ (not necessarily invertible)

Goal: build **invertible** function F: $\{0,1\}^{2n} \longrightarrow \{0,1\}^{2n}$




In symbols: $R_i = f_i(R_{i-1}) \oplus L_{i-1}$

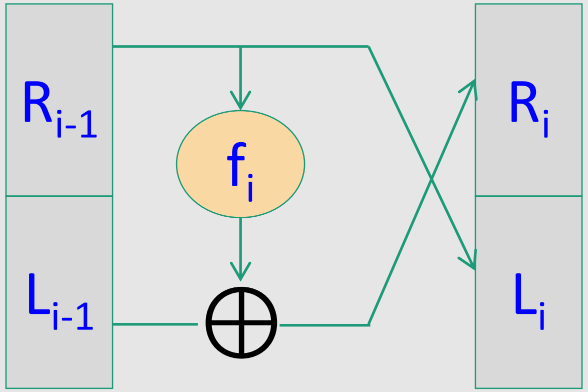
$L_i = R_{i-1}$

# Feistel network is invertible

**Claim**: for all (**arbitrary**) $f_1, \dots, f_d$: $\{0,1\}^n \longrightarrow \{0,1\}^n$

Feistel network $F: \{0,1\}^{2n} \longrightarrow \{0,1\}^{2n}$ is **invertible**

Proof: construct inverse




inverse

$R_{i-1} = L_i$

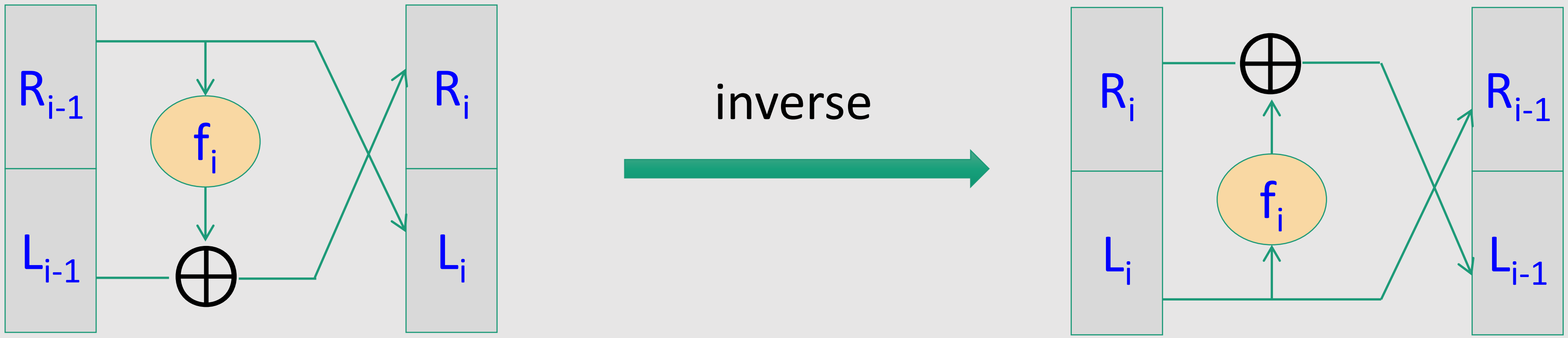
$L_{i-1} =$

# Feistel network is invertible

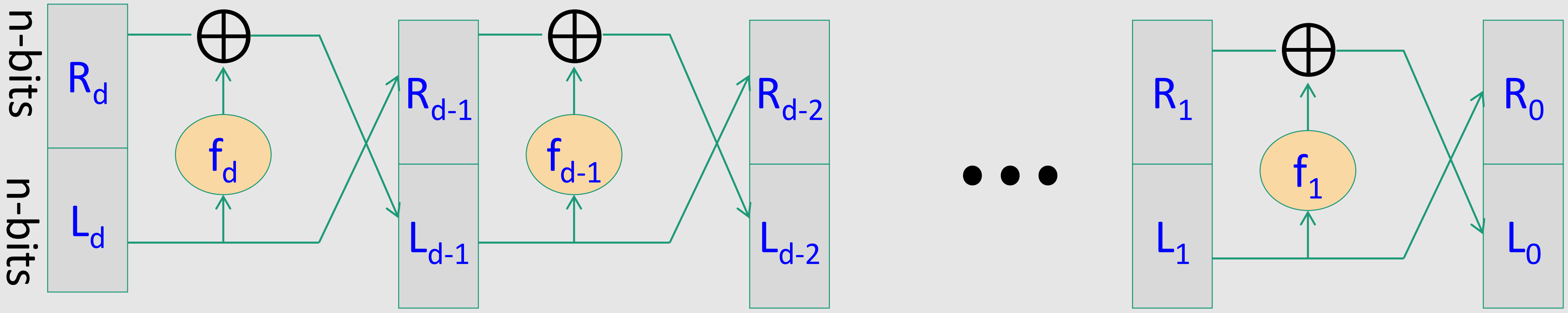
**Claim**: for all (**arbitrary**) $f_1, \ldots, f_d$: $\{0,1\}^n \longrightarrow \{0,1\}^n$

Feistel network $F$: $\{0,1\}^{2n} \longrightarrow \{0,1\}^{2n}$ is **invertible**

Proof: construct inverse



# Decryption circuit



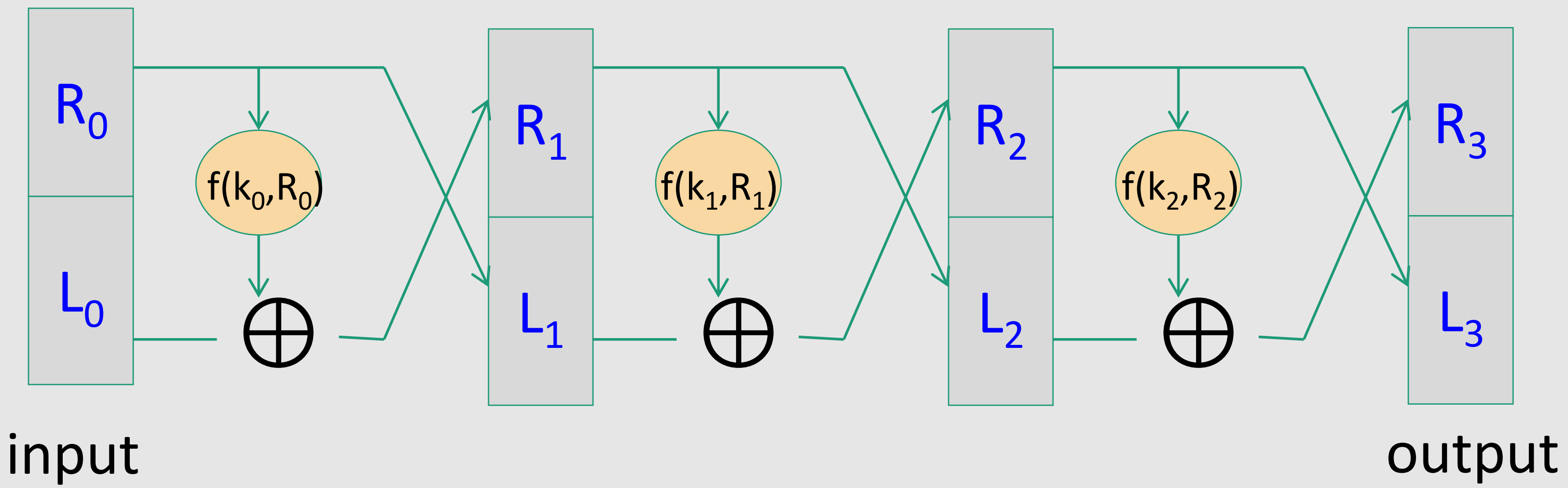

- Inversion is basically the same circuit, with $f_1, \ldots, f_d$ applied in reverse order
- General method for building invertible functions (block ciphers) from arbitrary functions.
- Used in many block ciphers … but not AES

**Theorem** (Luby-Rackoff '85):

f: $K \times \{0,1\}^n \longrightarrow \{0,1\}^n$ a **secure PRF**

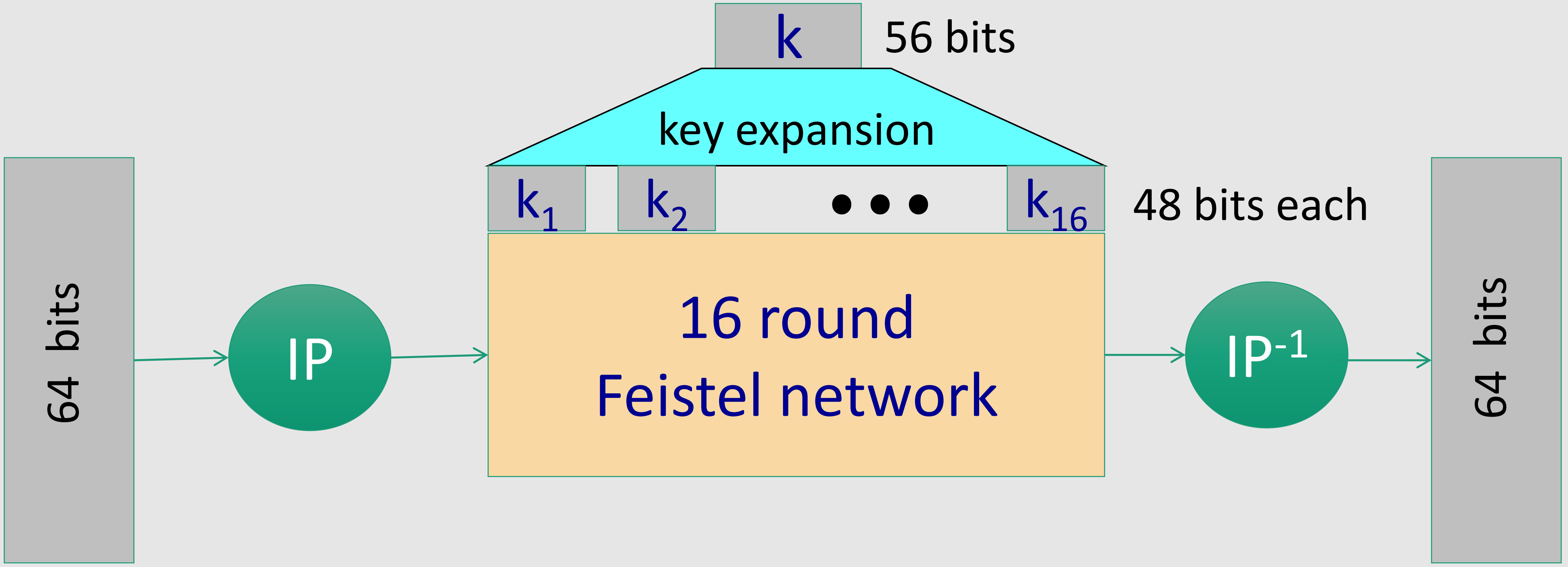
$\Rightarrow$ 3-round Feistel F: $K^3 \times \{0,1\}^{2n} \longrightarrow \{0,1\}^{2n}$ is a **secure PRP**

($k_0$, $k_1$, $k_2$ three **independent** keys)



# DES: 16 round Feistel network

$f_1, \ldots, f_{16}: \{0,1\}^{32} \longrightarrow \{0,1\}^{32}$ , $f_i(x) = F(k_i, x)$

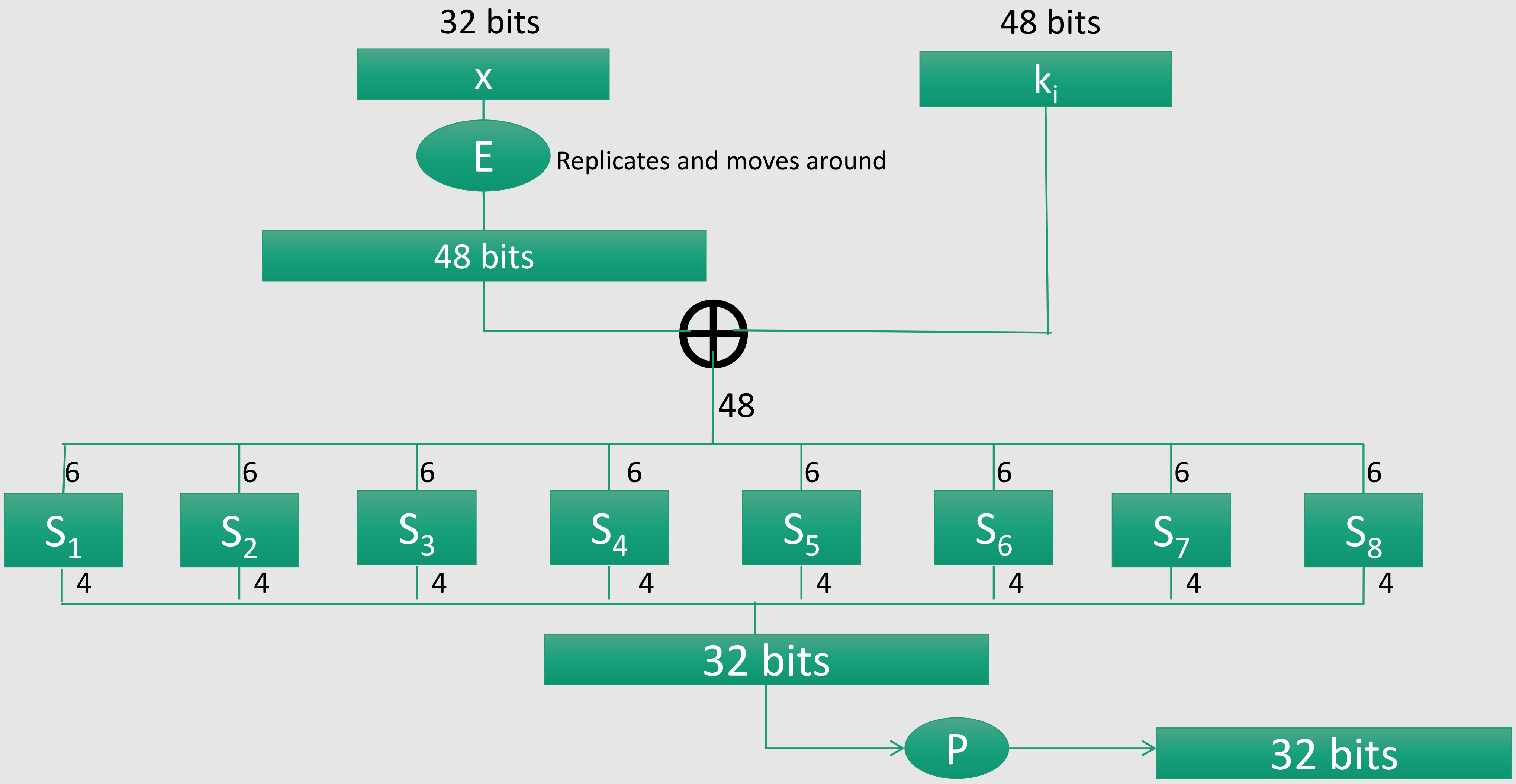



input

To invert, use keys in reverse order

output

# The function $F(k_i, x)$




S-box: function $\{0,1\}^6 \longrightarrow \{0,1\}^4$ , implemented as look-up table.

# The S-boxes (substitution boxes)

$$S_i: \{0,1\}^6 \longrightarrow \{0,1\}^4$$

| $S_5$ | | Middle 4 bits of input | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **0000** | **0001** | **0010** | **0011** | **0100** | **0101** | **0110** | **0111** | **1000** | **1001** | **1010** | **1011** | **1100** | **1101** | **1110** | **1111** |
| Outer bits | **00** | 0010 | 1100 | 0100 | 0001 | 0111 | 1010 | 1011 | 0110 | 1000 | 0101 | 0011 | 1111 | 1101 | 0000 | 1110 | 1001 |
| | **01** | 1110 | 1011 | 0010 | 1100 | 0100 | 0111 | 1101 | 0001 | 0101 | 0000 | 1111 | 1010 | 0011 | 1001 | 1000 | 0110 |
| | **10** | 0100 | 0010 | 0001 | 1011 | 1010 | 1101 | 0111 | 1000 | 1111 | 1001 | 1100 | 0101 | 0110 | 0011 | 0000 | 1110 |
| | **11** | 1011 | 1000 | 1100 | 0111 | 0001 | 1110 | 0010 | 1101 | 0110 | 1111 | 0000 | 1001 | 1010 | 0100 | 0101 | 0011 |

$$S_5(011011) \longrightarrow 1001$$

# Choosing the S-boxes and P-box

- Choosing the S-boxes and P-box at random would result in an insecure block cipher (key recovery after $\approx 2^{24}$ outputs)
- Several rules used in choice of S and P boxes:
  - No output bit should be close to a linear func. of the input bits
  - S-boxes are 4-to-1 maps (4 pre-images for each output)
  - …

# Exhaustive Search for block cipher key

**Goal**: given a few input output pairs $\left(m_i, c_i = E(k, m_i)\right)$ i=1,..,3
find key k.

# Exhaustive Search for block cipher key

**Goal**: given a few input output pairs $\left(m_i,\ c_i = E(k, m_i)\right)$ i=1,..,3
find key k.

Lemma: Suppose DES is an ***ideal cipher***
( $2^{56}$ random invertible functions $\Pi_1, \ldots, \Pi_{2^{\wedge}56} : \{0,1\}^{64} \to \{0,1\}^{64}$)

Then $\forall$ m, c there is at most <u>**one**</u> key k s.t. c = DES(k, m)

with prob. $\geq 1 - 1/256 \approx 99.5\%$

Proof:

$$\Pr[\exists k' \neq k: c=DES(k,m)=DES(k',m)] \leq \sum_{k' \in \{0,1\}^{56}} \Pr[DES(k,m) = DES(k',m)] \leq 2^{56} \times 1/(2^{64}) =$$
$$= 1/(2^{8}) = 1/256$$

# Exhaustive Search for block cipher key

For two DES pairs $\big(m_1, c_1{=}\text{DES}(k, m_1)\big), \ \big(m_2, c_2{=}\text{DES}(k, m_2)\big)$

unicity prob. $\approx 1 - 1/2^{71}$

For AES-128: given two inp/out pairs, unicity prob. $\approx 1 - 1/2^{128}$

$\Rightarrow$ two input/output pairs are enough for exhaustive key search.

# Exhaustive Search Attacks

# DES challenge

8 byte blocks, encrypted with the same key

$m_1$ $m_2$ $m_3$

msg = "The unknown messages is: XXXX … "

CT = $c_1$ $c_2$ $c_3$ $c_4$

**Goal**: find $k \in \{0,1\}^{56}$ s.t. $DES(k, m_i) = c_i$ for $i=1,2,3$ and decrypt $c_4, c_5 \ldots$

1997: Internet search -- **3 months**

1998: EFF machine (deep crack) -- **3 days** (250K $)

1999: combined search -- **22 hours**

2006: COPACOBANA (120 FPGAs) **--** **7 days** (10K $)

⇒ 56-bit ciphers should not be used !!

# Strengthening DES against exhaustive search

- Method 1: **Triple-DES**
- Method 2: **DESX**
- General construction that can be applied to other block ciphers as well.

# Triple DES

- Consider a **block cipher**

  **E** : $K \times M \longrightarrow M$

  **D** : $K \times M \longrightarrow M$

- Define **3E: $K^3 \times M \longrightarrow M$** as

  $$\mathbf{3E\,(k_1,k_2,k_3,\,m) = E(k_1,\,D(k_2,\,E(k_3,m)))}$$

- For **3DES** (or **Triple DES**)
  - **key lenght** = 3×56 = **168 bits**.
  - **3×slower** than DES.
  - **$k_1=k_2=k_3 \Rightarrow$ single DES**
  - **simple attack in time $\approx 2^{118}$** (more on this later …)

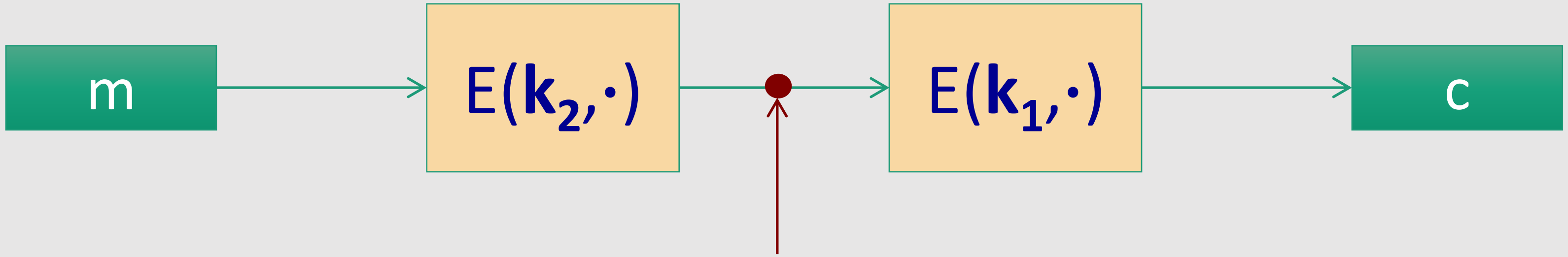
# Why not double DES?

- Given a block cipher **E**, define $\mathbf{2E}(k_1, k_2, m) = \mathbf{E}(k_1, \mathbf{E}(k_2, m))$
- **Double DES:** $\mathbf{2DES}(k_1, k_2, m) = \mathbf{E}(k_1, \mathbf{E}(k_2, m))$

  key-length = 112 bits for 2DES
- **Attack:** Given **m** and **c** the goal is to

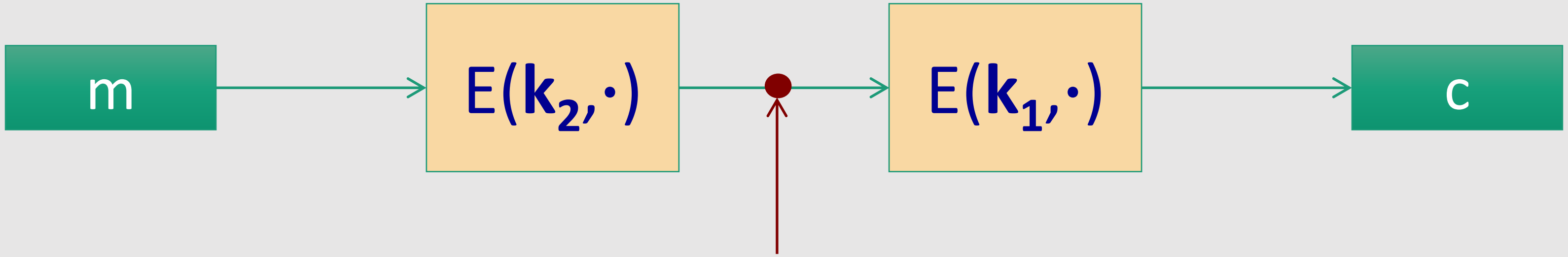
  find $(k_1,k_2)$ s.t. $\mathbf{E}(k_1, \mathbf{E}(k_2,m)) = c$ **or equivalently**

  find $(k_1,k_2)$ s.t. $\mathbf{E}(k_2,m) = \mathbf{D}(k_1,c)$



# Meet in the middle attack

- **Attack:** Given **m** and **c** the goal is to
  find $(k_1,k_2)$ s.t. $\mathbf{E(k_1, E(k_2,m)) = c}$ **or equivalently**
  find $(k_1,k_2)$ s.t. $\mathbf{E(k_2,m) = D(k_1,c)}$




- **Attack involves TWO STEPS**

# Meet in the middle attack

**Step 1:**

- build table.
- sort on $2^{nd}$ column

| | |
|---|---|
| $k^0 = 00…00$ | $E(k^0, m)$ |
| $k^1 = 00…01$ | $E(k^1, m)$ |
| $k^2 = 00…10$ | $E(k^2, m)$ |
| ⋮ | ⋮ |
| $k^N = 11…11$ | $E(k^N, m)$ |

} $2^{56}$ entries

# Meet in the middle attack

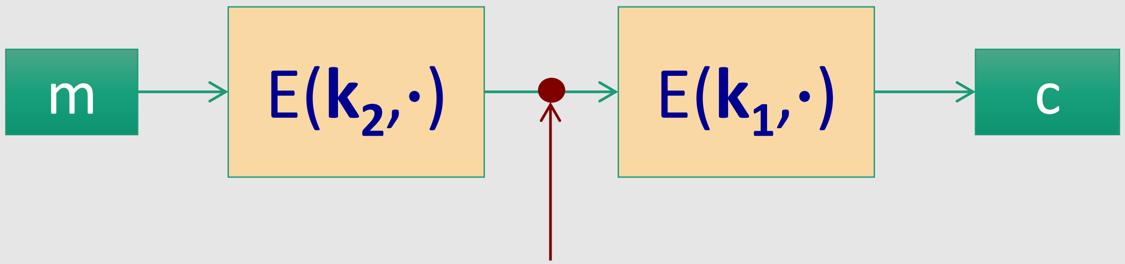
**Step 2:**

- for each $k \in \{0,1\}^{56}$ do:

  test if $D(k, c)$ is in the $2^{nd}$ column of the table

  If so, then $\mathbf{E(k^i,m) = D(k,c)} \quad \Rightarrow \quad \mathbf{(k^i,k) = (k_2,k_1)}$

| | |
|---|---|
| $k^0 = 00…00$ | $E(k^0, m)$ |
| $k^1 = 00…01$ | $E(k^1, m)$ |
| ⋮ | ⋮ |
| $k^i = 00…….$ | $E(k^i, m)$ |
| ⋮ | ⋮ |
| $k^N = 11…11$ | $E(k^N, m)$ |



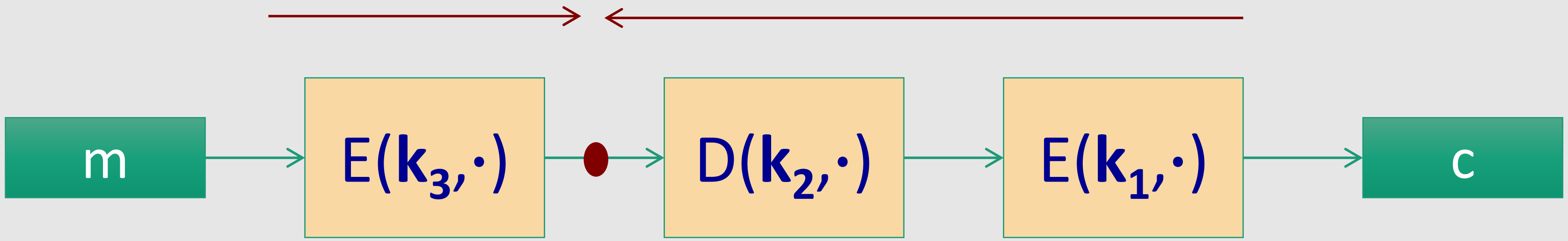
# Meet in the middle attack

$$\text{Time} = \underbrace{2^{56}\log(2^{56})}_{\text{build + sort table}} + \underbrace{2^{56}\log(2^{56})}_{\text{search in table}} < 2^{63} \ll 2^{112} \;,$$

$$\text{Space} \approx 2^{56}$$

# Meet in the middle attack

**Same attack on 3DES:**




Time = $2^{118}$ , space ≈ $2^{56}$

Time = $\underbrace{2^{56}\log(2^{56})}_{\text{build + sort table}} + \underbrace{2^{112}\log(2^{56})}_{\text{search in table}} < 2^{118}$

# DESX

- Consider a **block cipher**

  $\mathbf{E} : K \times M \longrightarrow M$

  $\mathbf{D} : K \times M \longrightarrow M$

- Define **EX** as

$$\mathbf{EX}(k_1, k_2, k_3, m) = k_1 \oplus \mathbf{E}(k_2, m \oplus k_3)$$

- For **DESX**
  - key-len = **64**+**56**+**64** = 184 bits $\qquad k_1 \oplus \mathbf{E}(k_2, m \oplus k_3)$
  - … but easy attack in time $2^{64+56} = 2^{120}$

- Note: $k_1 \oplus E(k_2, m)$ and $E(k_2, m \oplus k_1)$ insecure !!

  (XOR outside) or (XOR inside) ⇒ As weak as E w.r.t. exhaustive search

# Few others attacks on block ciphers

# Linear attacks on DES

A tiny bit of linearly in $S_5$ lead to a $2^{43}$ time attack.

Total attack time $\approx 2^{43}$ ( $\ll 2^{56}$ ) with $2^{42}$ random inp/out pairs

# Quantum attacks

Generic search problem:

Let $f: X \longrightarrow \{0,1\}$ be a function.

Goal: find $x^* \in X$ s.t. $f(x^*)=1$.

Classical computer: best generic algorithm **time = O( |X| )**

Quantum computer [Grover '96] : **time = O( $|X|^{1/2}$ )**

# Quantum exhaustive search

Given **m** and **c = E(k,m)** define

$$\text{For } k \in K,\ f(k) = \begin{cases} 1 & \text{if } E(k,m) = c \\ 0 & \text{otherwise} \end{cases}$$

Grover ⇒ quantum computer can find k in time $O(\ |K|^{1/2}\ )$

DES: time $\approx 2^{28}$ , AES-128: time $\approx 2^{64}$

Quantum computer ⇒ 256-bits key ciphers (e.g., AES-256)

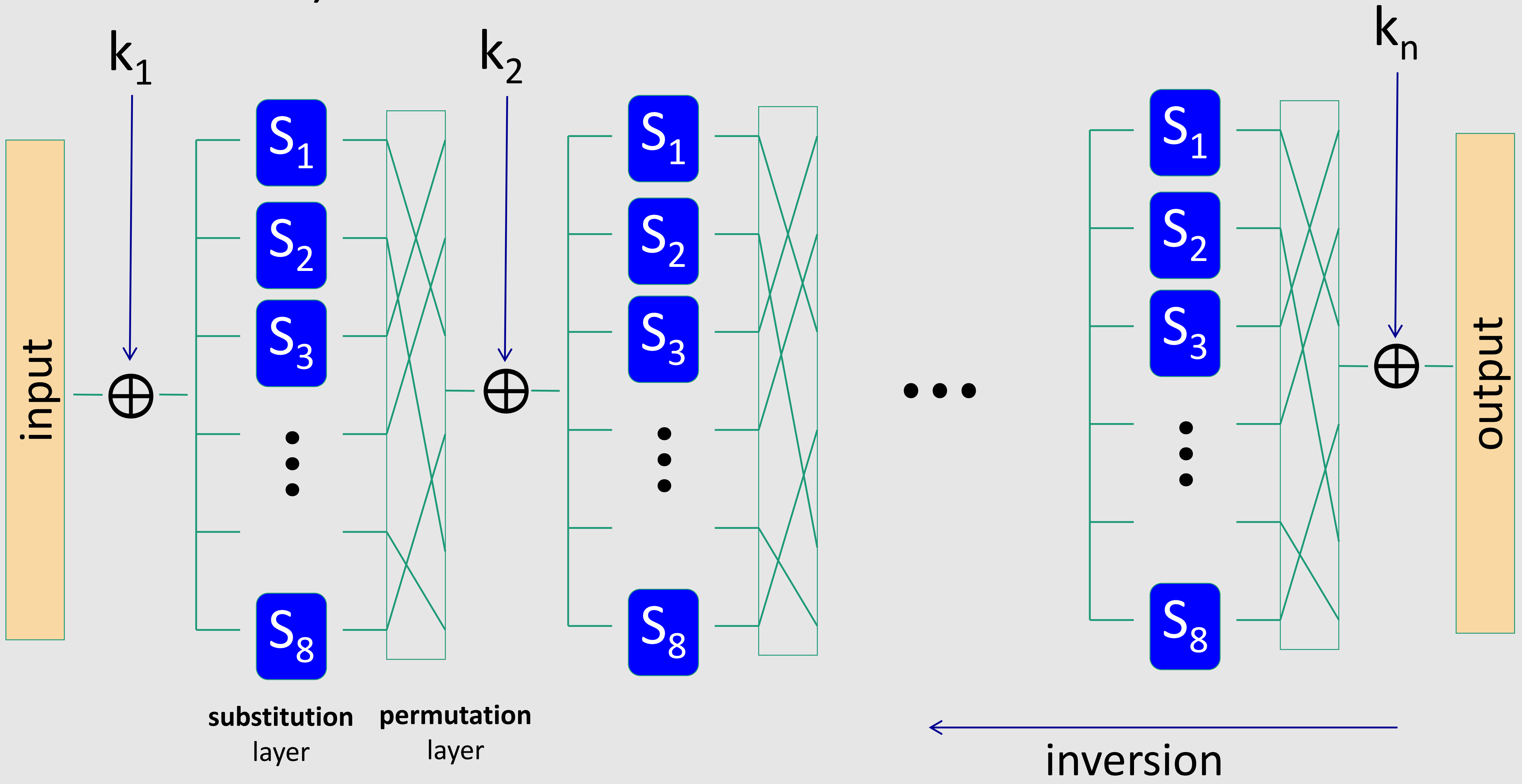
# Advanced Encryption Standard (AES)

# The AES process

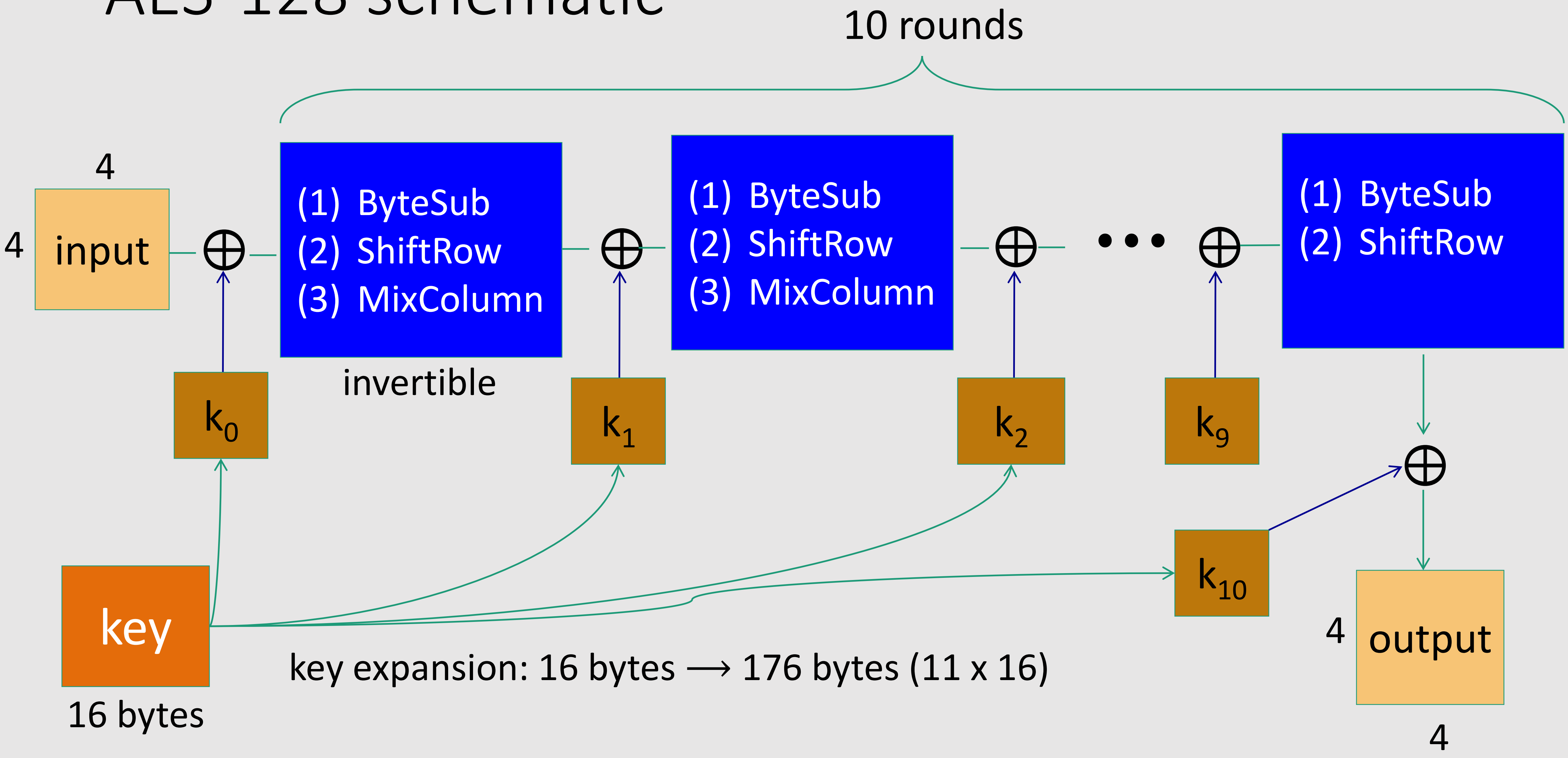
- 1997: NIST publishes request for proposal
- 1998: 15 submissions. Five claimed attacks.
- 1999: NIST chooses 5 finalists
- 2000: NIST chooses Rijndael as AES (designed in Belgium)

Key sizes: 128, 192, 256 bits. Block size: 128 bits

# AES is a Substitution–permutation Network (not Feistel)



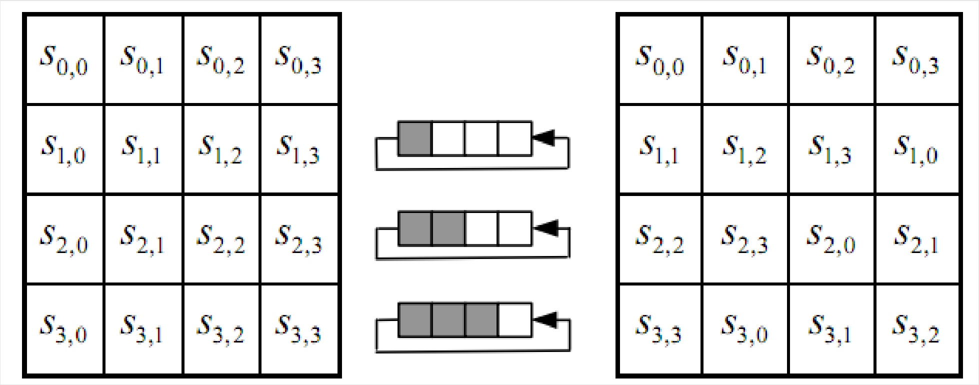
# AES-128 schematic



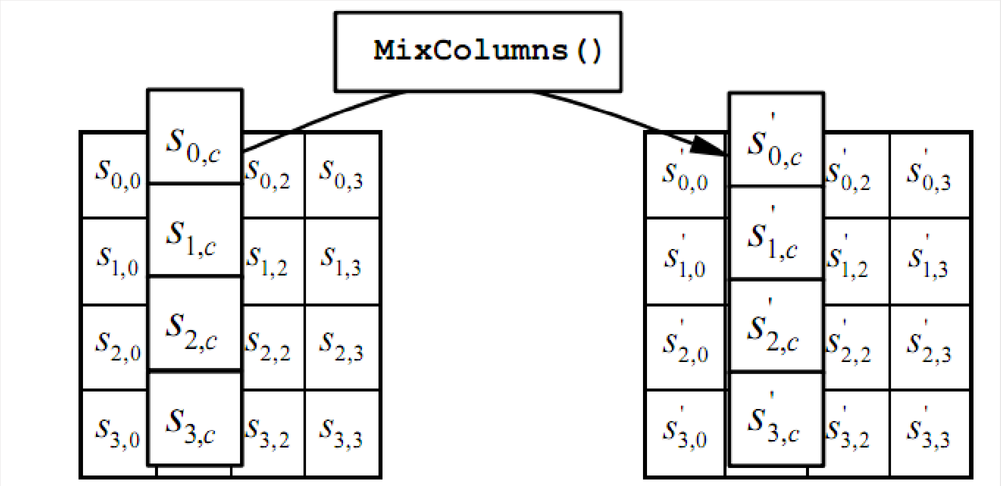
# The round function

- **ByteSub**: a 1 byte S-box. 256 byte table (easily computable)
  - Apply S-box to each byte of the 4x4 input A, i.e., A[i,j] = S[A[i,j]], for 1 ≤i,j≤4

- **ShiftRows**:




- **MixColumns**:



# AES in hardware

AES instructions in Intel Westmere:

- **aesenc, aesenclast**: do one round of AES

  128-bit registers: xmm1=state, xmm2=round key

  **aesenc xmm1, xmm2** ; puts result in xmm1

- **aeskeygenassist**: performs AES key expansion
- Claim 14 x speed-up over OpenSSL on same hardware

Similar instructions on AMD Bulldozer

# Attacks

- Best key recovery attack:

  four times better than ex. search [BKR'11]

- Related key attack on AES-256: [BK'09]

  Given $2^{99}$ inp/out pairs from **four related keys** in AES-256

  can recover keys in time $\approx 2^{99}$

$$\text{PRF} \Rightarrow \text{PRG}$$

$$\text{PRG} \Rightarrow \text{PRF}$$

# An easy application: PRF $\Rightarrow$ PRG (counter mode)

- Let **F: K $\times$ {0,1}$^n$ $\rightarrow$ {0,1}$^n$** be a **PRF.**

- We define the **PRG** **G: K $\rightarrow$ {0,1}$^{nt}$** as follows:

  (**t** is a parameter that we can choose)

$$G(k) = F(k, \langle 0 \rangle n) \;||\; F(k, \langle 1 \rangle n) \;||\; \cdots \;||\; F(k, \langle t\text{-}1 \rangle n)$$

- **Properties:**

  - **Theorem:** If **F** is a **secure PRF** then **G** is a **secure PRG**

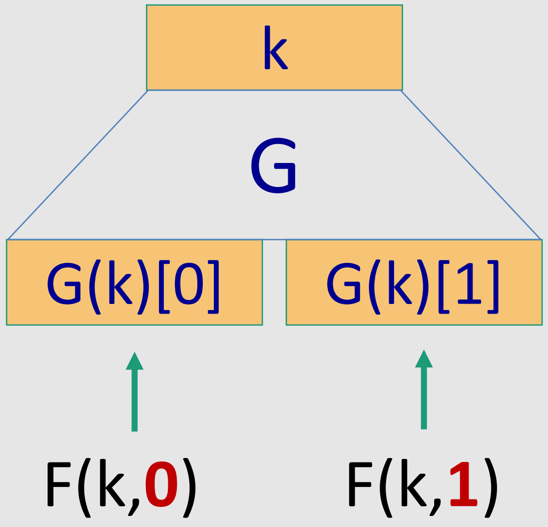
  - Key property: **parallelizable**

# Can we build a PRF from a PRG?

Let $G: K \longrightarrow K^2$ be a PRG

Define a 1-bit PRF $F: K \times \{0,1\} \longrightarrow K$ as

$$F(k, x \in \{0,1\}) = G(k)[x]$$




**Theorem.** If **G** is a **secure PRG** then **F** is a **secure PRF**

Can we build a PRF with a larger domain? (e.g., 128 bits)

# Extending a PRG

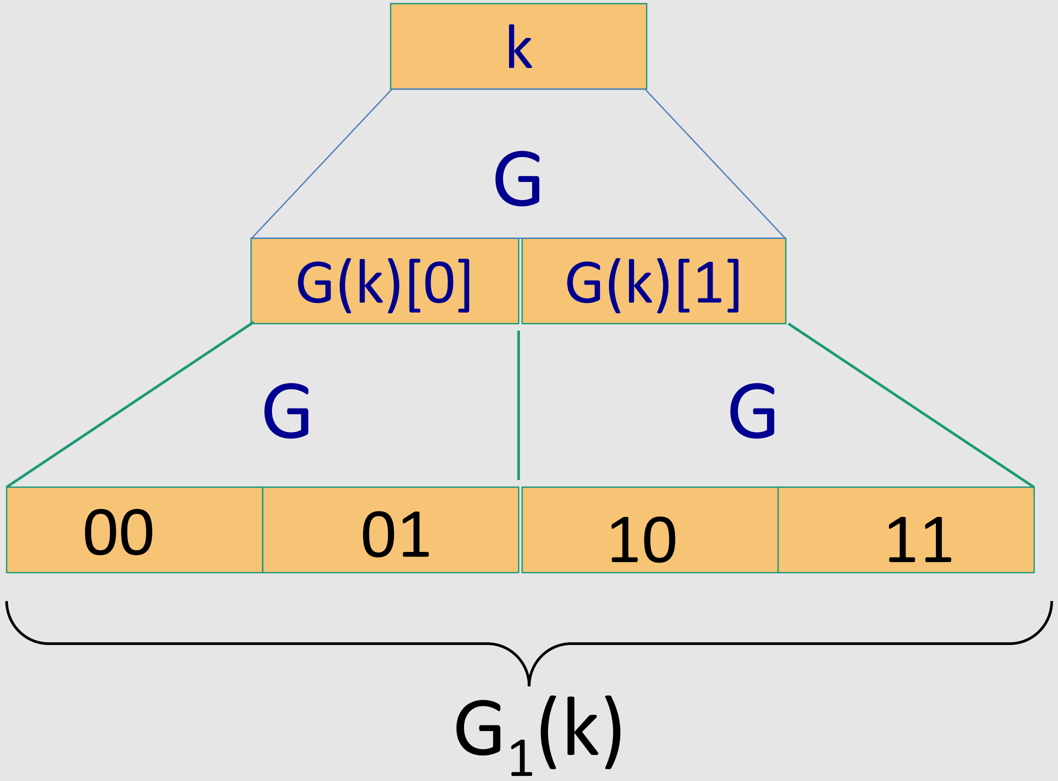
Let $G: K \longrightarrow K^2$ be a PRG

Define $G_1: K \longrightarrow K^4$ as

$$G_1(k) = G(G(k)[0]) \parallel G(G(k)[1])$$

Then define a 2-bit PRF $F: K \times \{0,1\}^2 \longrightarrow K$ as

$$F(k, x \in \{0,1\}^2) = G_1(k)[x]$$



# Extending more

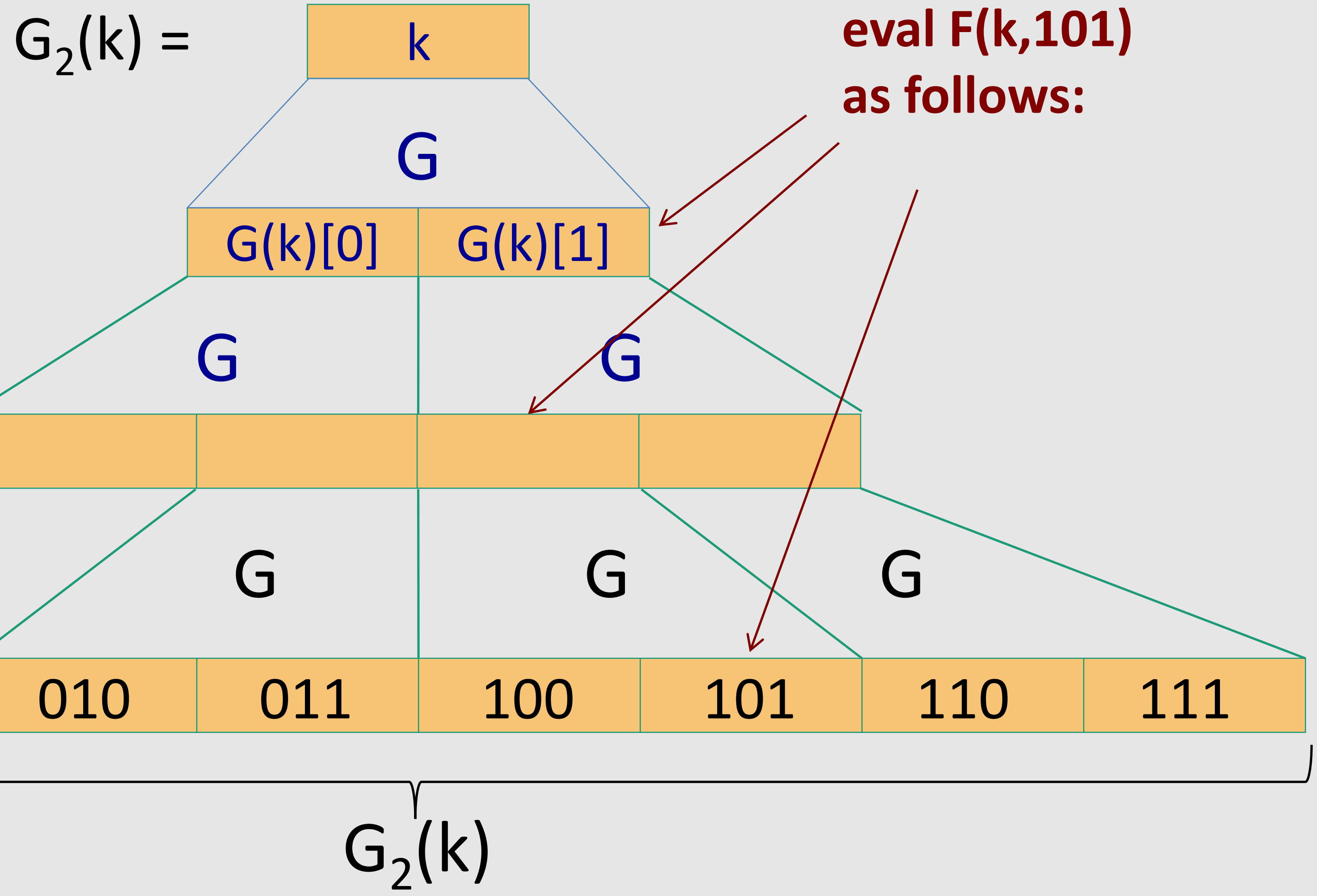
Let $G: K \longrightarrow K^2$ .

Define $G_2: K \longrightarrow K^8$ as

Then define a 3-bit PRF

$F: K \times \{0,1\}^3 \longrightarrow K$ as

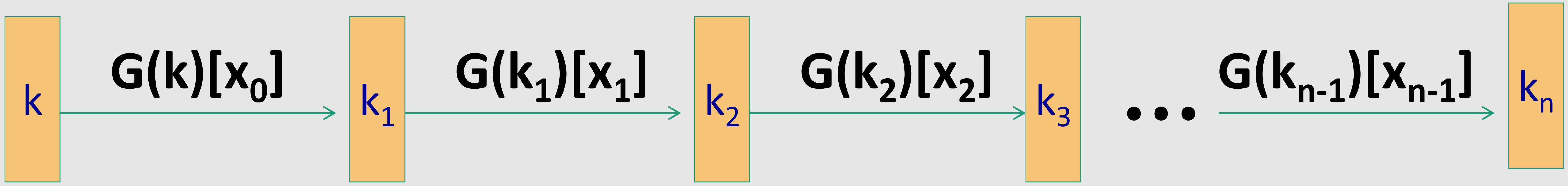
$F(k, x\in\{0,1\}^3) = G_2(k)[x]$



# Extending even more: the GGM PRF

Let $G: K \longrightarrow K^2$ . define PRF $F: K \times \{0,1\}^n \longrightarrow K$ as

For input $x = x_0\, x_1 \ldots x_{n-1} \in \{0,1\}^n$ do:




Security: **G** a **secure PRG** $\Rightarrow$ **F** is a **secure PRF** on $\{0,1\}^n$ .

**Not used in practice due to slow performance.**

# Secure block cipher from a PRG?

Can we build a secure PRP from a secure PRG?

- No, it cannot be done
- Yes, just plug the GGM PRF into the Luby-Rackoff theorem ←
- It depends on the underlying PRG

**Theorem** (Luby-Rackoff '85):

f: $K \times \{0,1\}^n \longrightarrow \{0,1\}^n$ a **secure PRF**

$\Rightarrow$ 3-round Feistel F: $K^3 \times \{0,1\}^{2n} \longrightarrow \{0,1\}^{2n}$ is a **secure PRP**

($k_0$, $k_1$, $k_2$ three **independent** keys)