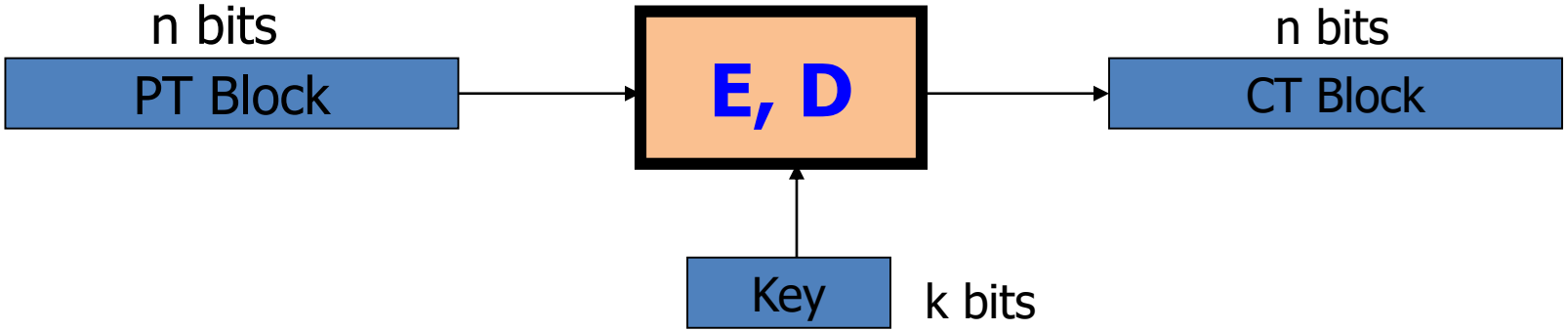

# Block ciphers


# What is a block cipher?

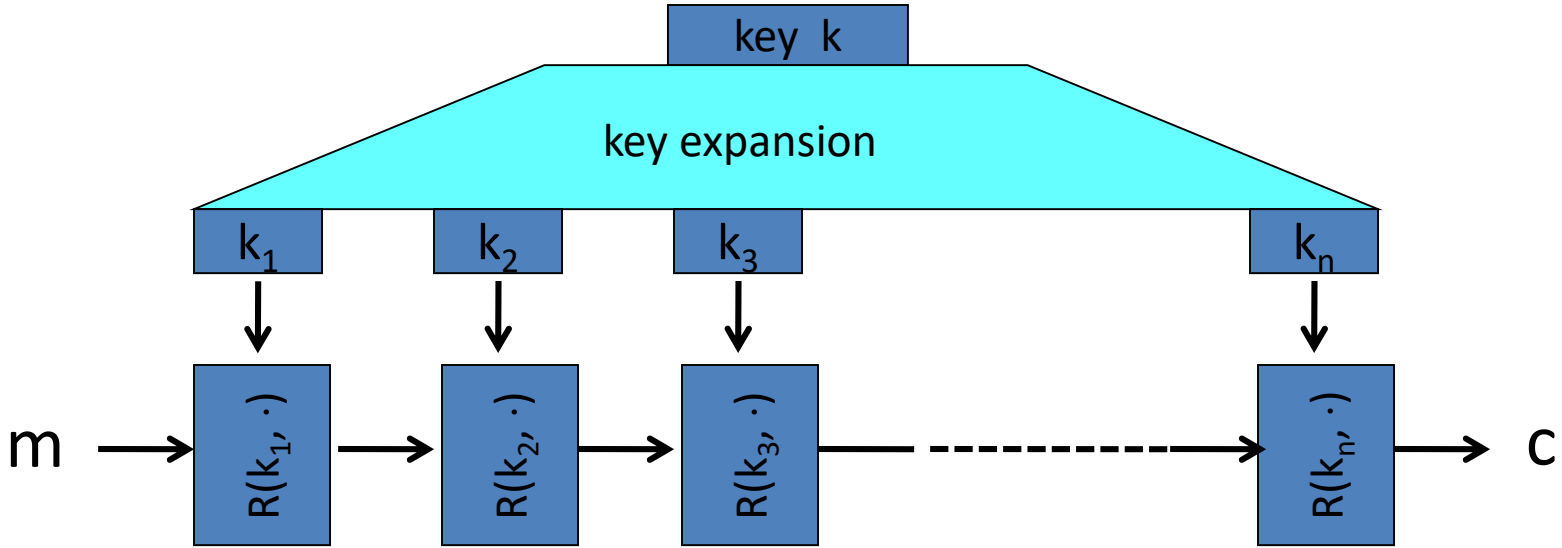
# Block ciphers: crypto work horse




Canonical examples:

1. 3DES: n= 64 bits, k = 168 bits
2. 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 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. Exists "efficient" deterministic algorithm to evaluate E(k,x)
  2. The function $E(k, \cdot)$ is one-to-one
  3. 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

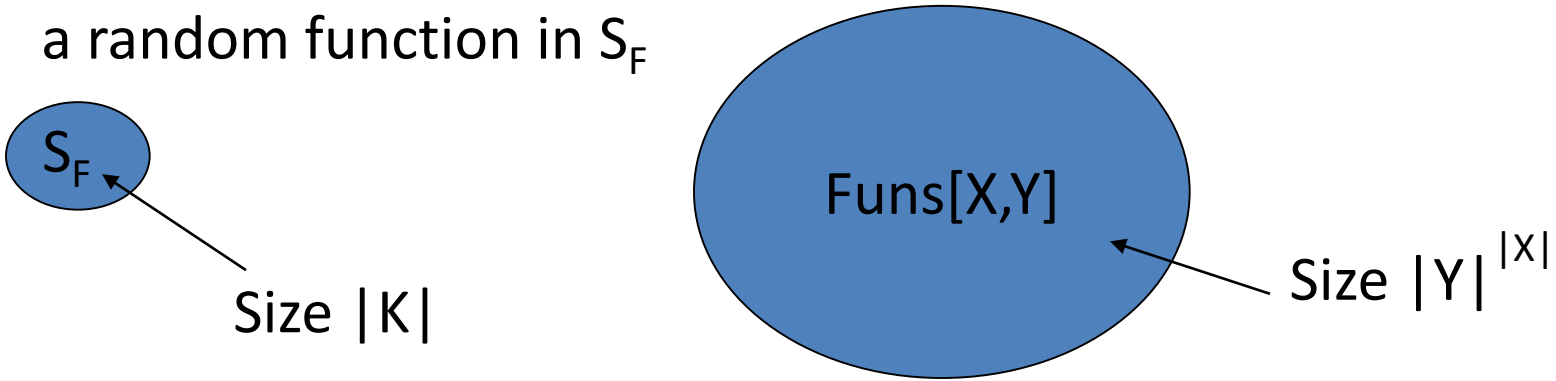
- Let $F: K \times X \rightarrow Y$ be a PRF

  Funs[X,Y]: the set of **all** functions from X to Y

  $S_F = \{ F(k,\cdot) \text{ s.t. } k \in K \} \subseteq$ Funs[X,Y]

---

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



# Secure PRFs

- Let $F: K \times X \rightarrow Y$ be a PRF

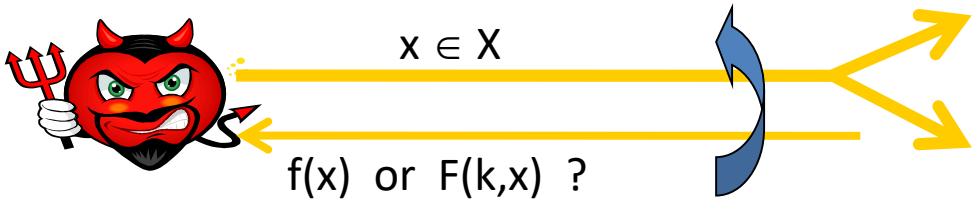
  Funs[X,Y]: the set of **all** functions from X to Y

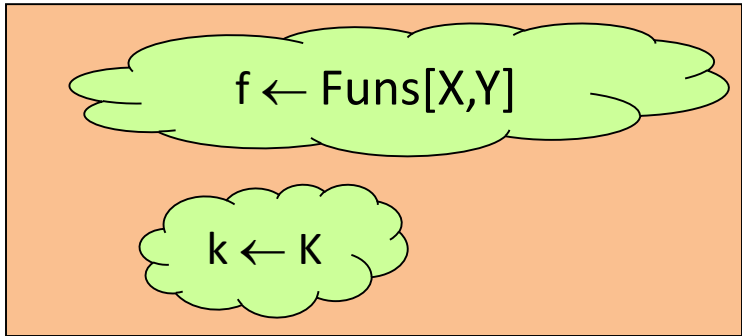
  $S_F = \{ F(k,\cdot) \text{ s.t. } k \in K \} \subseteq$ Funs[X,Y]

---

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

???



# Secure PRPs (secure block cipher)

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

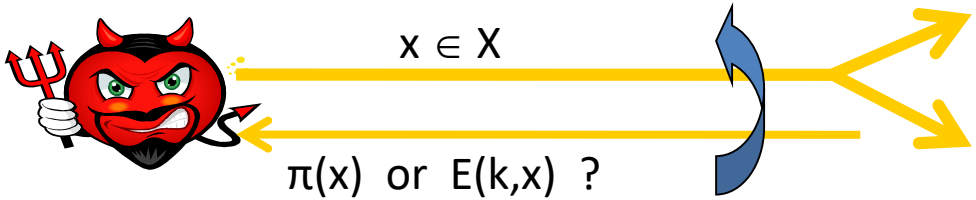
  Perms[X]: the set of all **one-to-one** functions from X to X

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

---

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

???



Let $F: K \times X \rightarrow \{0,1\}^{128}$ be a secure PRF.

Is the following G a secure PRF?

$$G(k, x) = \begin{cases} 0^{128} & \text{if } x=0 \\ F(k,x) & \text{otherwise} \end{cases}$$

- No, it is easy to distinguish G from a random function
- Yes, an attack on G would also break F
- It depends on F

# An easy application: PRF ⇒ PRG

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

Then the following $G: K \rightarrow \{0,1\}^{nt}$ is a secure PRG:

$$G(k) = F(k,0) \,||\, F(k,1) \,||\, \cdots \,||\, F(k,t-1)$$

Key property: parallelizable

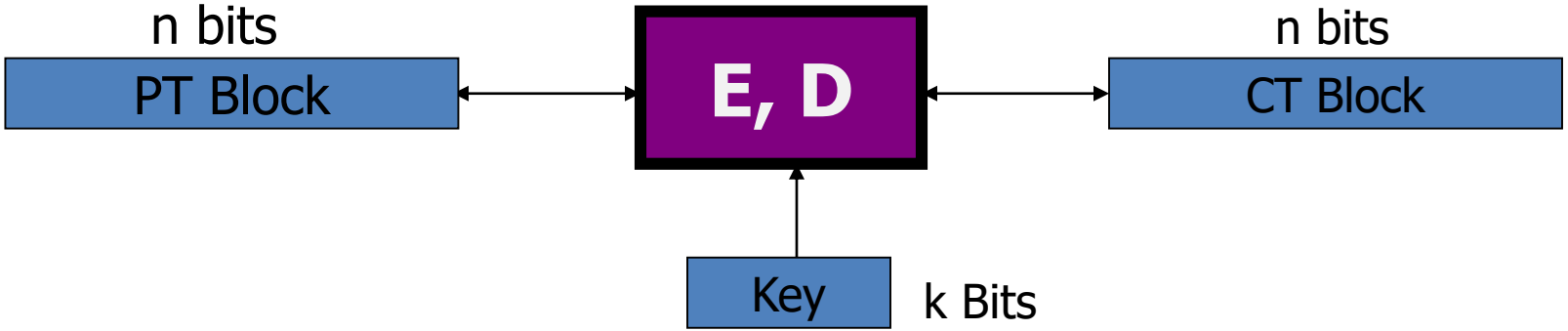
Security from PRF property: $F(k, \cdot)$ indist. from random function $f(\cdot)$

# End of Segment


# Block ciphers

## The data encryption standard (DES)

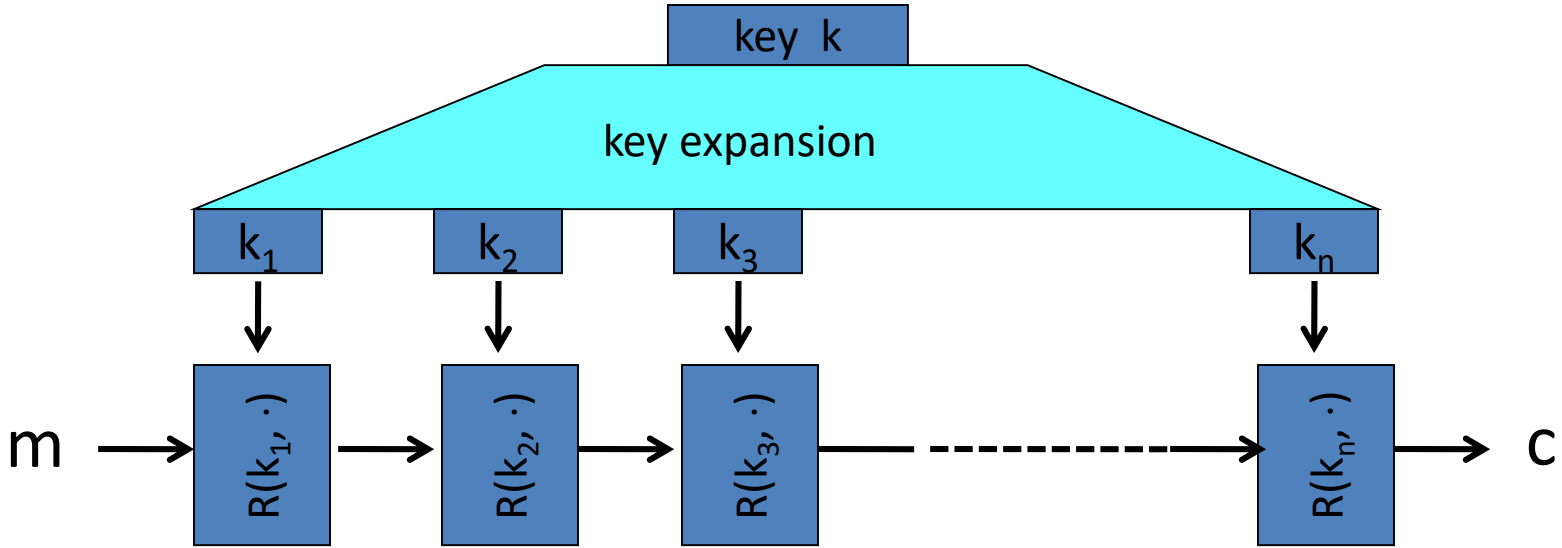
# Block ciphers: crypto work horse




Canonical examples:

1. 3DES: n= 64 bits, k = 168 bits
2. 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)

# The Data Encryption Standard (DES)

- Early 1970s: Horst Feistel designs Lucifer at IBM

  key-len = 128 bits ; block-len = 128 bits

- 1973: NBS asks for block cipher proposals.

  IBM submits variant of Lucifer.

- 1976: NBS adopts DES as a federal standard

  key-len = 56 bits ; block-len = 64 bits

- 1997: DES broken by exhaustive search

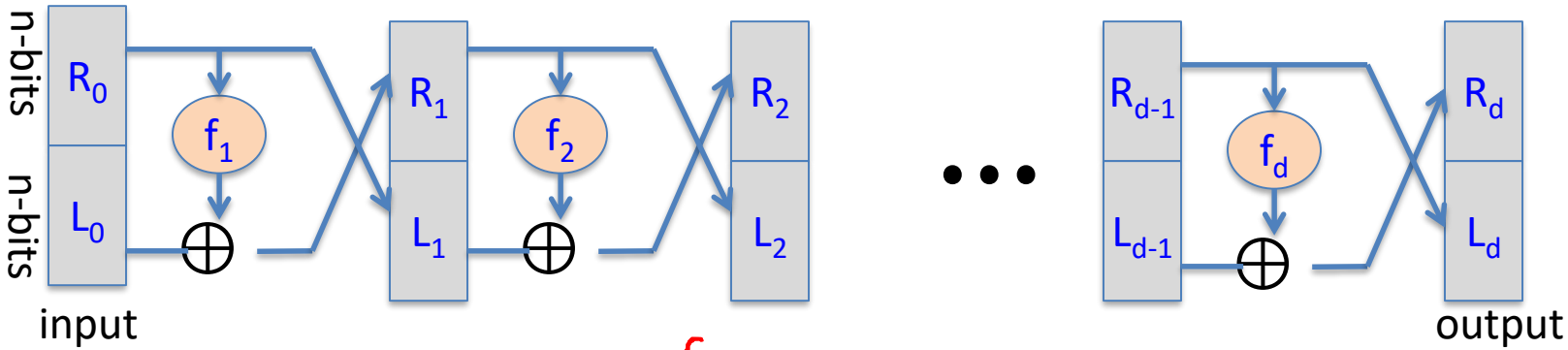
- 2000: NIST adopts Rijndael as AES to replace DES

Widely deployed in banking (ACH) and commerce

# DES: core idea – Feistel Network

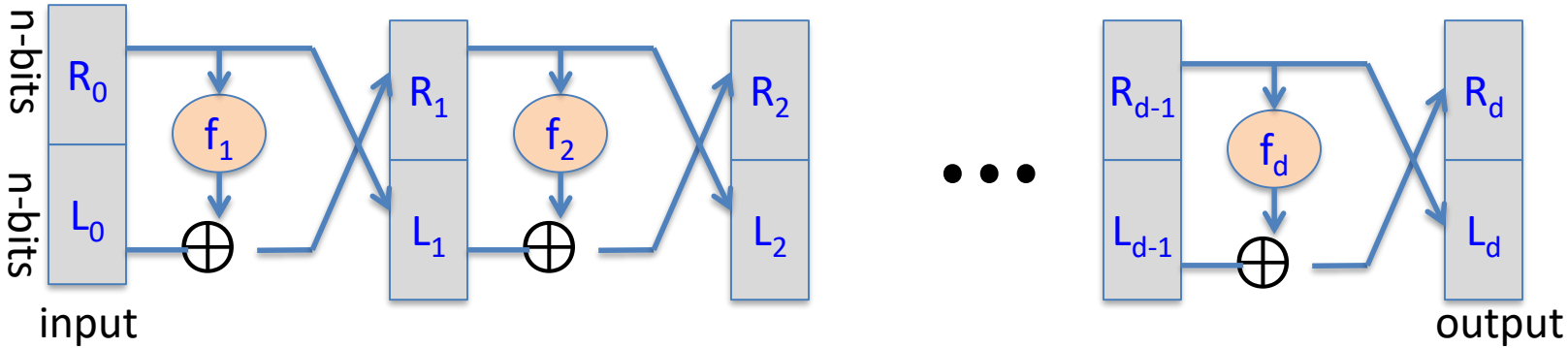
Given functions $f_1, \ldots, f_d$: $\{0,1\}^n \longrightarrow \{0,1\}^n$

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




In symbols:

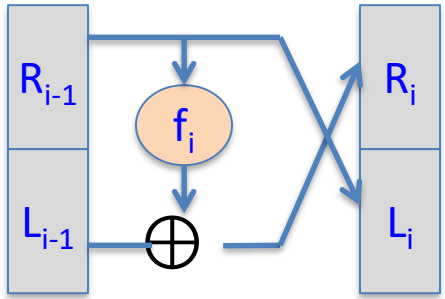
$$\begin{cases} R_i = f_i(R_{i-1}) \oplus L_{i-1} \\ L_i = R_{i-1} \end{cases}$$




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

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

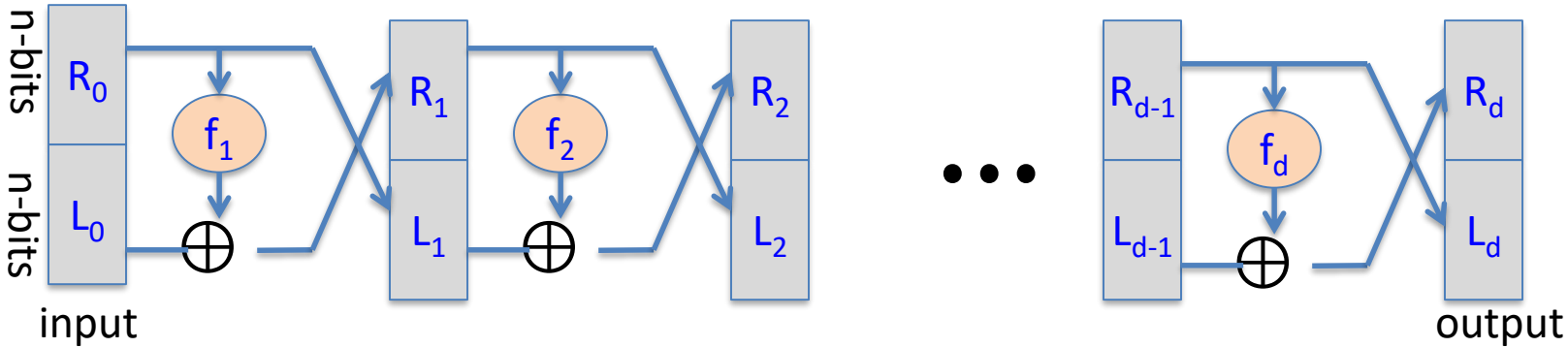
Proof: construct inverse




inverse

$R_{i-1} = L_i$

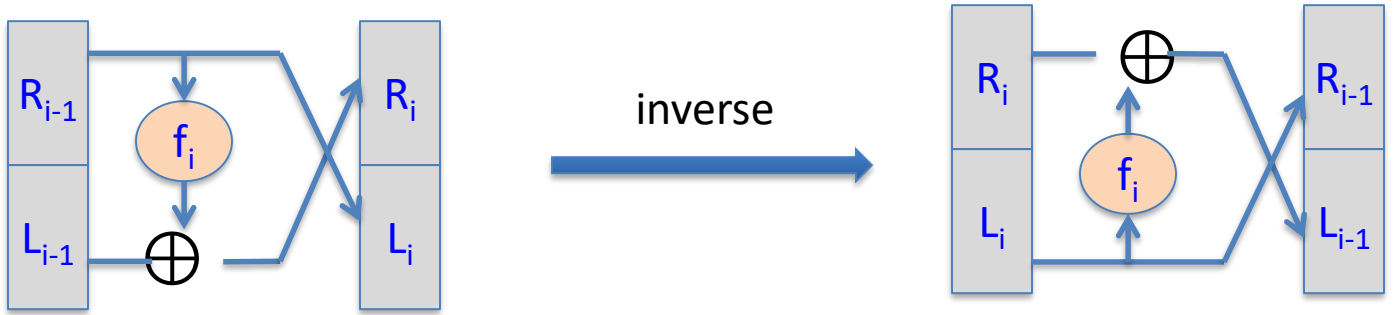
$L_{i-1} = f_i(L_i) \oplus R_i$



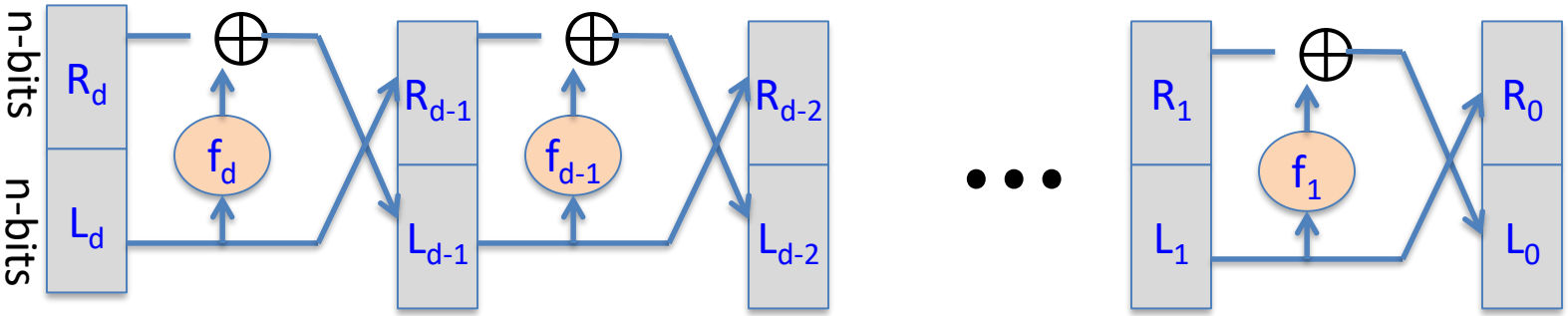

**Claim**: for all $f_1, \ldots, f_d$: $\{0,1\}^n \longrightarrow \{0,1\}^n$

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

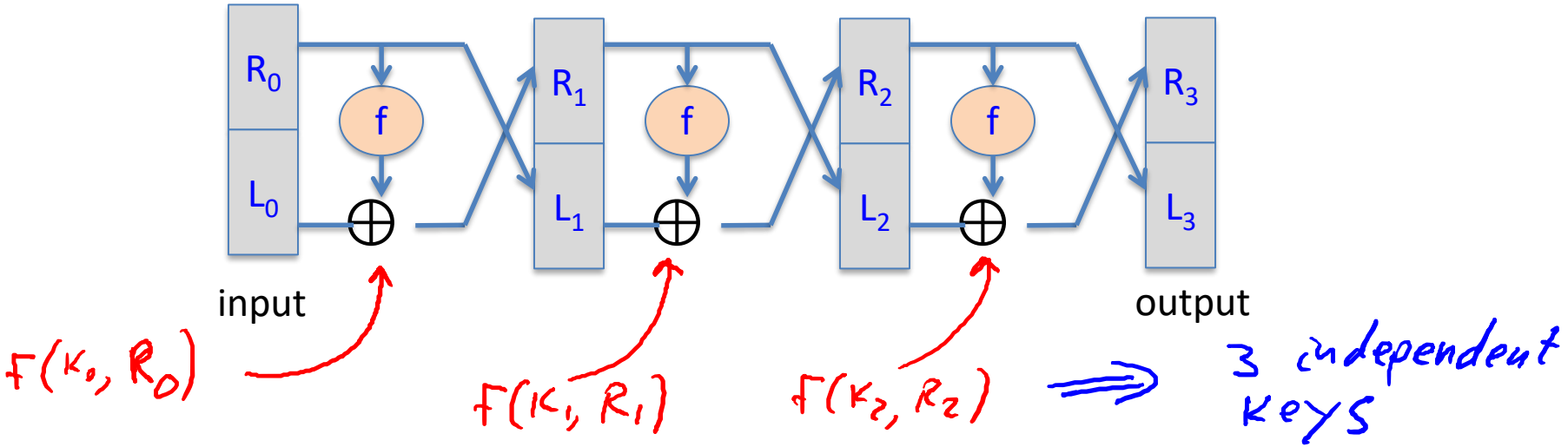
- General method for building invertible functions (block ciphers) from arbitrary functions.

- Used in many block ciphers … but not AES

"Thm:" (Luby-Rackoff '85):

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

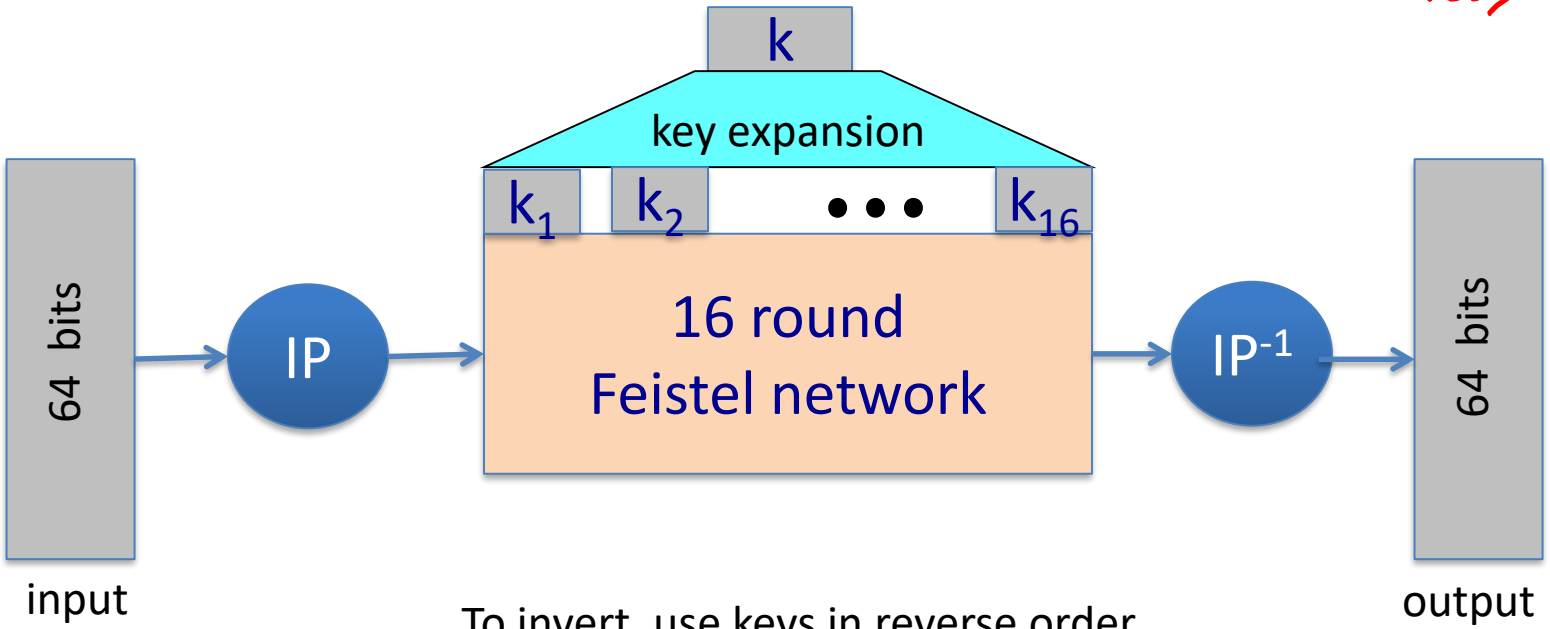
$\Rightarrow$ 3-round Feistel F: $K^3 \times \{0,1\}^{2n} \longrightarrow \{0,1\}^{2n}$ a secure PRP



# DES: 16 round Feistel network

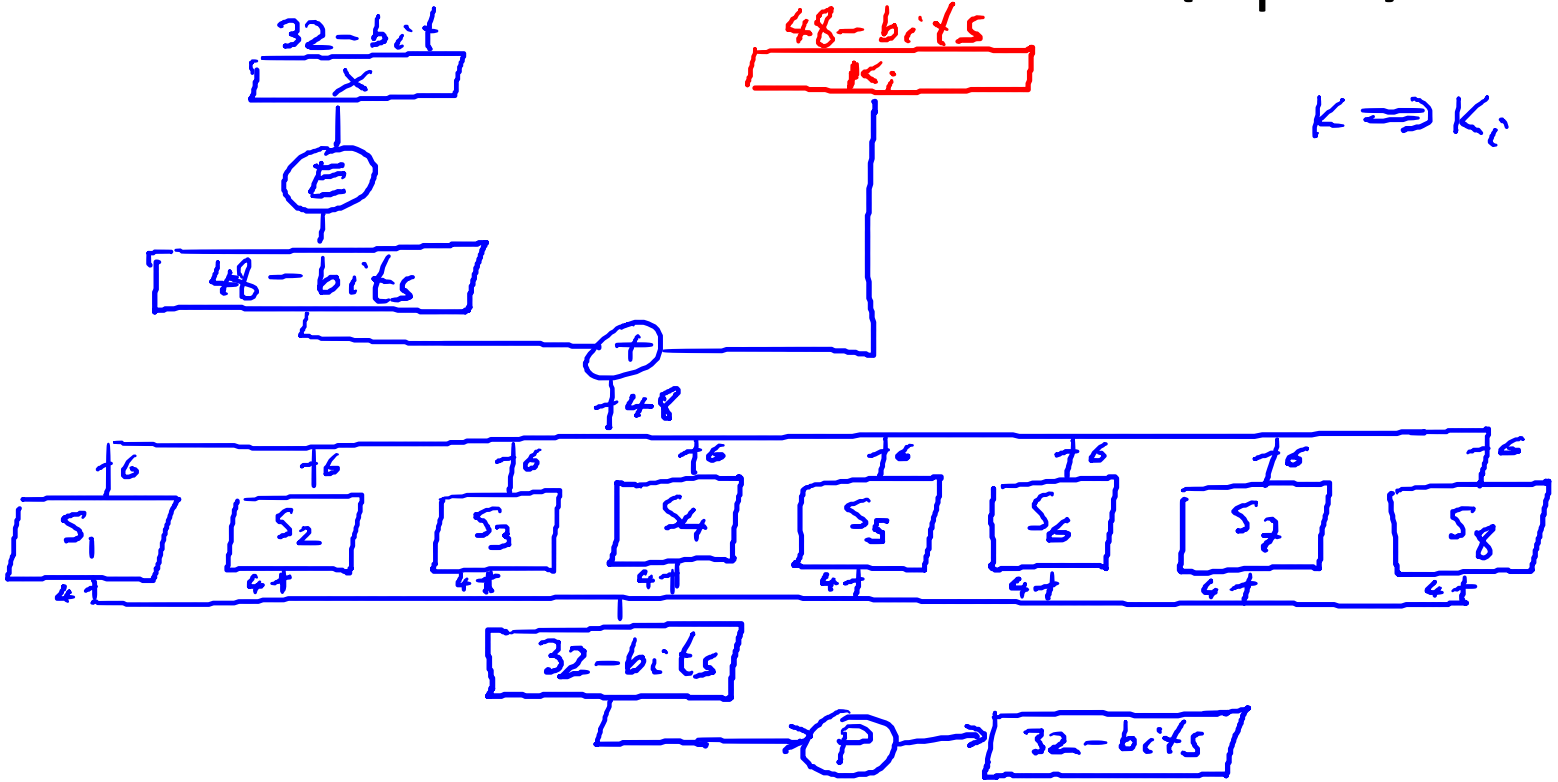
$f_1, \ldots, f_{16}: \{0,1\}^{32} \longrightarrow \{0,1\}^{32}$ , $f_i(x) = \mathbf{F}(k_i, x)$

From key K




To invert, use keys in reverse order

# The function $F(k_i, x)$




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

# The S-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 |

011011

# Example: a bad S-box choice

Suppose:

$$S_i(x_1, x_2, \ldots, x_6) = \left( x_2 \oplus x_3,\ \ x_1 \oplus x_4 \oplus x_5,\ \ x_1 \oplus x_6,\ \ x_2 \oplus x_3 \oplus x_6 \right)$$

or written equivalently: $S_i(\mathbf{x}) = A_i \cdot \mathbf{x} \pmod 2$

$$\begin{bmatrix} 0&1&1&0&0&0 \\ 1&0&0&1&1&0 \\ 1&0&0&0&0&1 \\ 0&1&1&0&0&1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} x_2 \oplus x_3 \\ x_1 \oplus x_4 \oplus x_5 \\ x_1 \oplus x_6 \\ x_2 \oplus x_3 \oplus x_6 \end{bmatrix}$$

We say that $S_i$ is a linear function.

# Example: a bad S-box choice

Then entire DES cipher would be linear: ∃fixed binary matrix B s.t.

$$\text{DES}(k,m) = \underset{64}{\overset{832}{B}} \cdot \begin{pmatrix} m \\ k_1 \\ k_2 \\ \vdots \\ k_{16} \end{pmatrix} = c \pmod 2$$

But then: $\text{DES}(k,m_1) \oplus \text{DES}(k,m_2) \oplus \text{DES}(k,m_3) = \text{DES}(k, m_1 \oplus m_2 \oplus m_3)$

$$B \begin{pmatrix} m_1 \\ k \end{pmatrix} \oplus B \begin{pmatrix} m_2 \\ k \end{pmatrix} \oplus B \begin{pmatrix} m_3 \\ k \end{pmatrix} = B \begin{pmatrix} m_1 \oplus m_2 \oplus m_3 \\ k \oplus k \oplus k \end{pmatrix}$$

$K = \begin{pmatrix} k_1 \\ \vdots \\ k_{16} \end{pmatrix}$

# 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) [BS'89]

Several rules used in choice of S and P boxes:

- No output bit should be close to a linear func. of the input bits

# End of Segment


# Block ciphers

## Exhaustive Search Attacks

# Ideal cipher model

**Def**: In the ideal cipher model, we assume the block cipher *is* a random permutation for *every* key. Furthermore, we treat these permutations as independent.

.

Example: Suppose DES is an ***ideal cipher***

It is a collection of $2^{56}$ independent random permutations, one for each key

# Exhaustive Search for block cipher key

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

Lemma: Suppose DES is an ***ideal cipher***

( $2^{56}$ random invertible functions $\pi_1, \ldots, \pi_{2^{56}} : \{0,1\}^{64} \to \{0,1\}^{64}$ )

Then ∀ m, c there is at most **one** key k s.t. c = DES(k, m)

Proof: with prob. ≥ 1 – 1/256 ≈ 99.5%

$$Pr[\exists k' \neq k : c = DES(k,m) = DES(k',m)] \leq$$

$$\leq \sum_{k' \in \{0,1\}^{56}} Pr[DES(k,m) = DES(k',m)] \leq 2^{56} \cdot \frac{1}{2^{64}} = \frac{1}{2^8}$$

# Exhaustive Search for block cipher key

For two DES pairs $(m_1, c_1=DES(k, m_1))$, $(m_2, c_2=DES(k, m_2))$

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 for DES but not AES.

# DES challenge

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

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 !! (128-bit key ⇒ $2^{72}$ days)

# Strengthening DES against ex. search

Method 1: **Triple-DES**

- Let $E : K \times M \longrightarrow M$ be a block cipher
- Define **3E**: $K^3 \times M \longrightarrow M$ as

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

$k_1 = k_2 = k_3 \Rightarrow$ single DES

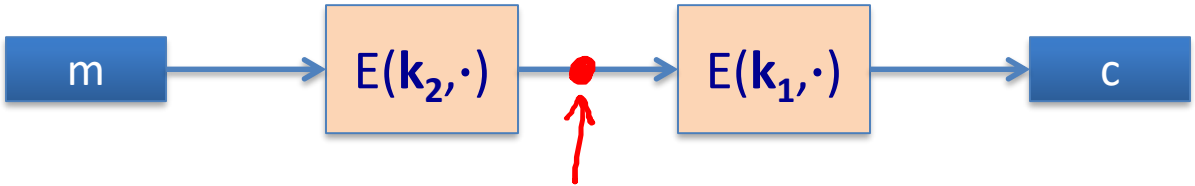
For 3DES: key-size = 3×56 = 168 bits. 3×slower than DES.

(simple attack in time $\approx 2^{118}$ )

# Why not double DES?

- Define $2E\big((k_1,k_2), m\big) = E\big(k_1, E(k_2, m)\big)$

key-len = 112 bits for DES




Find $(k_1, k_2)$ s.t.
$E(k_1, E(k_2, M)) = C$
Equivalently:
$E(k_2, M) = D(k_1, c)$

Attack: $M = (m_1,\ldots, m_{10})$ , $C = (c_1,\ldots,c_{10})$.

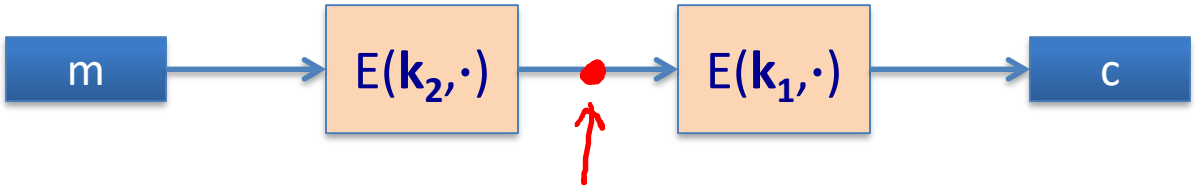
- step 1: build table.

  sort on 2nd column

| | |
|---|---|
| $k^0 = 00\ldots00$ | $E(k^0, M)$ |
| $k^1 = 00\ldots01$ | $E(k^1, M)$ |
| $k^2 = 00\ldots10$ | $E(k^2, M)$ |
| ⋮ | ⋮ |
| $k^N = 11\ldots11$ | $E(k^N, M)$ |

$2^{56}$ entries

# Meet in the middle attack




| | |
|---|---|
| $k^0 = 00\ldots00$ | $E(k^0, M)$ |
| $k^1 = 00\ldots01$ | $E(k^1, M)$ |
| $k^2 = 00\ldots10$ | $E(k^2, M)$ |
| ⋮ | ⋮ |
| $k^N = 11\ldots11$ | $E(k^N, M)$ |

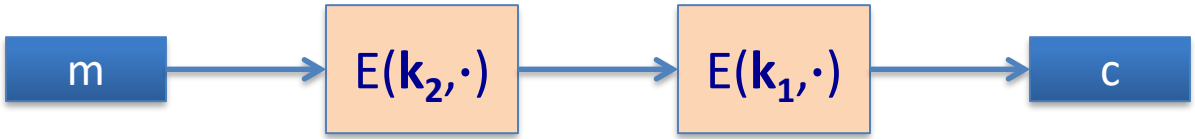
Attack: $M = (m_1,\ldots, m_{10})$ , $C = (c_1,\ldots,c_{10})$

- step 1: build table.
- Step 2: for all $k \in \{0,1\}^{56}$ do:

  test if $D(k, C)$ is in 2nd column.

  if so then $E(k^i,M) = D(k,C) \Rightarrow (k^i,k) = (k_2,k_1)$

# Meet in the middle attack

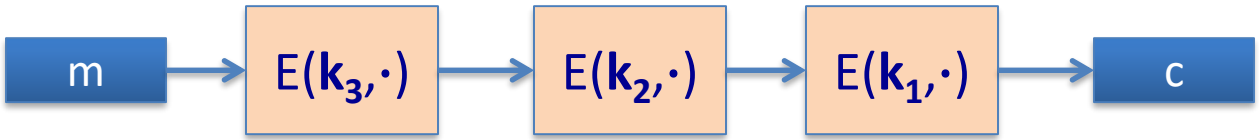



Time = $2^{56}\log(2^{56})$ + $2^{56}\log(2^{56})$ < $2^{63}$ << $2^{112}$ , space ≈ $2^{56}$

build + sort table

search in table

Same attack on 3DES: Time = $2^{118}$ , space ≈ $2^{56}$



# Method 2: DESX

$E : K \times \{0,1\}^n \longrightarrow \{0,1\}^n$ a block cipher

Define EX as $EX\big((k_1,k_2,k_3), m\big) = k_1 \oplus E(k_2, m \oplus k_3)$

For DESX: key-len = 64+56+64 = 184 bits

… but easy attack in time $2^{64+56} = 2^{120}$ (homework)

Note: $k_1 \oplus E(k_2, m)$ and $E(k_2, m \oplus k_1)$ does nothing !!

# End of Segment


# Block ciphers

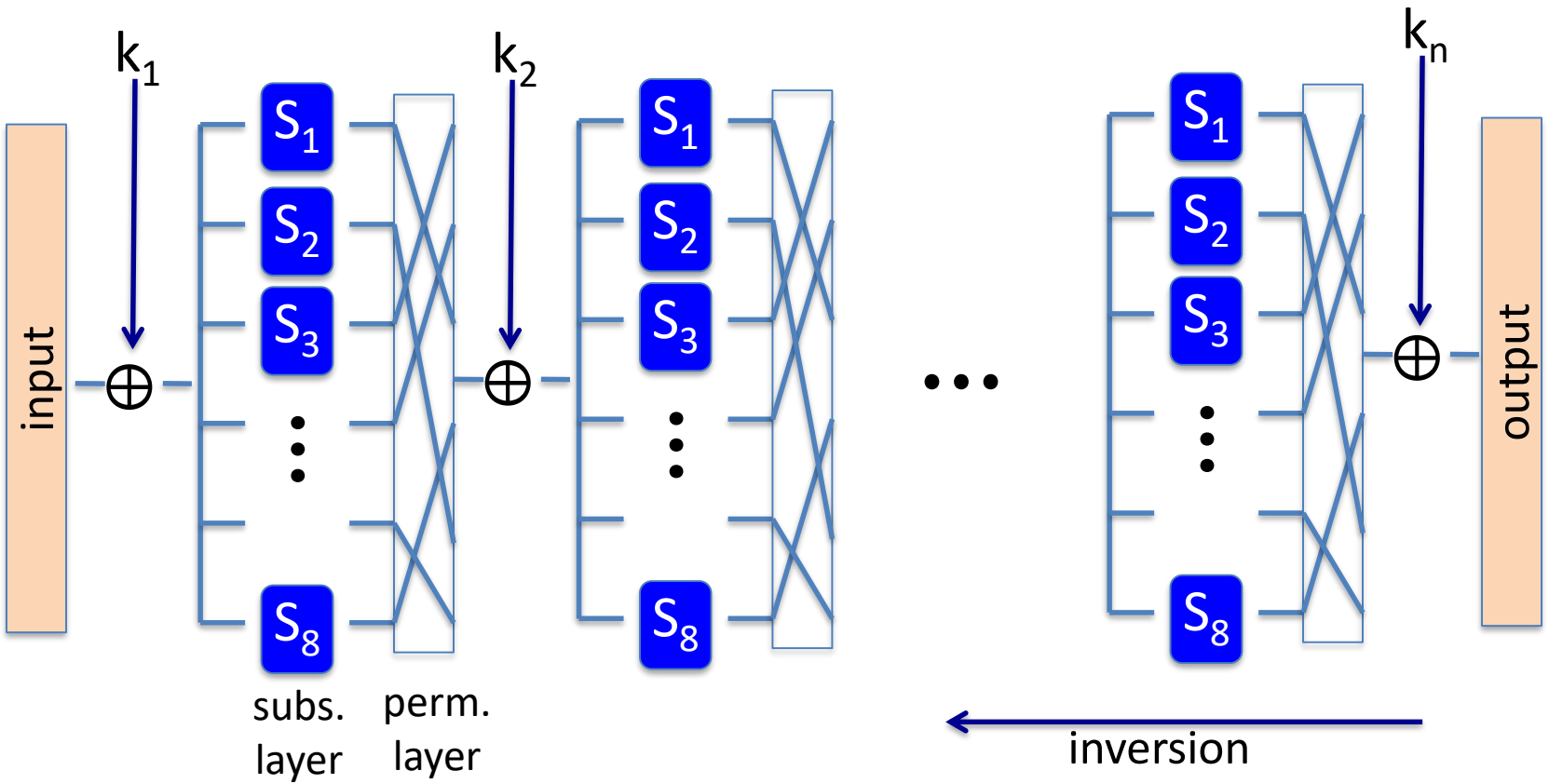
## The AES block cipher

# The AES process

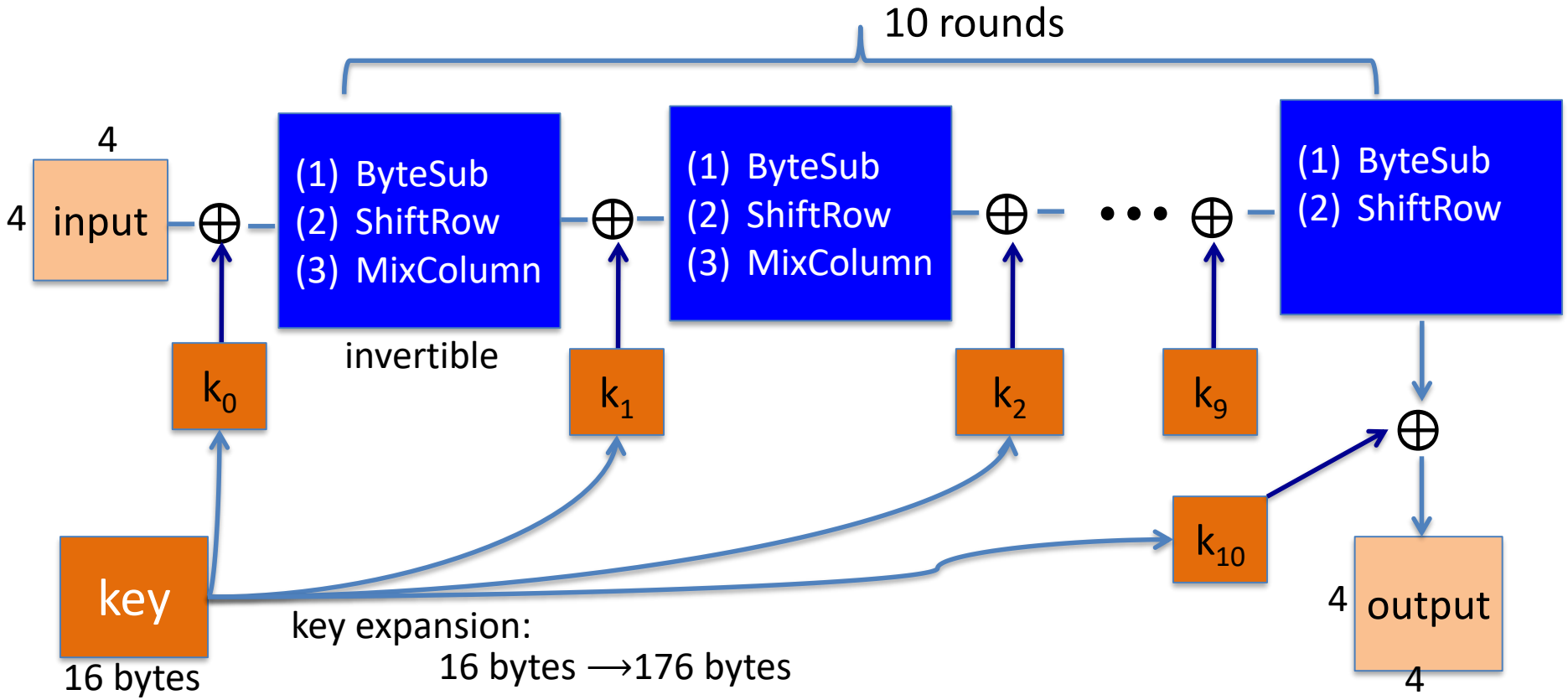
- 1997: NIST publishes request for proposal
- 1998: 15 submissions. Five claimed attacks.
- 1999: NIST chooses 5 finalists
- 2000: NIST chooses Rijndael as AES (designed in Belgium)

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

# AES is a Subs-Perm network (not Feistel)

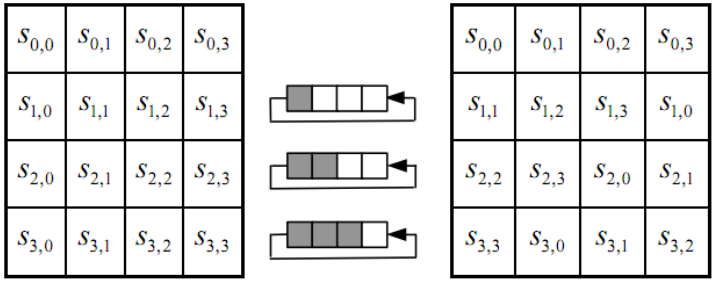


# AES-128 schematic

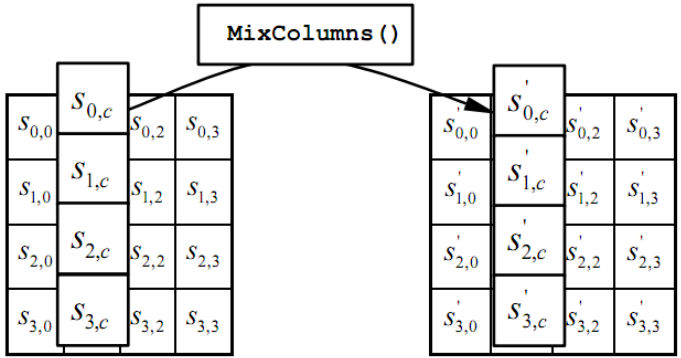


# The round function

- **ByteSub**: a 1 byte S-box. 256 byte table (easily computable)
- **ShiftRows**:




- **MixColumns**:

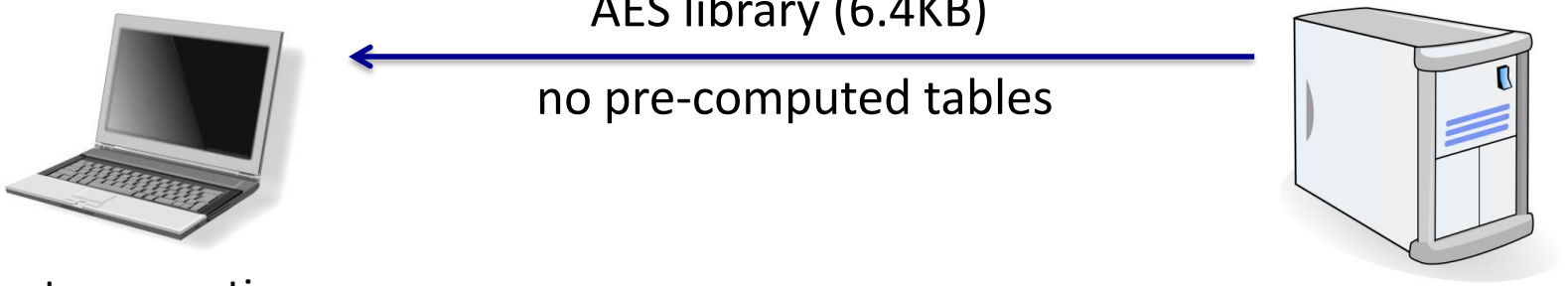


# Code size/performance tradeoff

| | Code size | Performance |
|---|---|---|
| Pre-compute round functions (24KB or 4KB) | largest | fastest: table lookups and xors |
| Pre-compute S-box only (256 bytes) | smaller | slower |
| No pre-computation | smallest | slowest |

# Example: Javascript AES

AES in the browser:




Prior to encryption:
pre-compute tables

Then encrypt using tables

http://crypto.stanford.edu/sjcl/

# 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 $\approx 2^{99}$

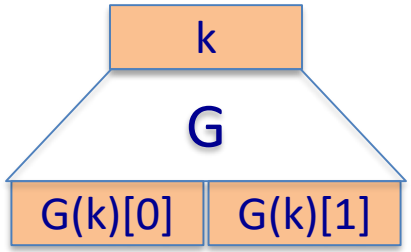
# End of Segment


# Block ciphers

## Block ciphers from PRGs

# Can we build a PRF from a PRG?

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




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

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

Thm: If G is a secure PRG then F is a secure PRF

Can we build a PRF with a larger domain?

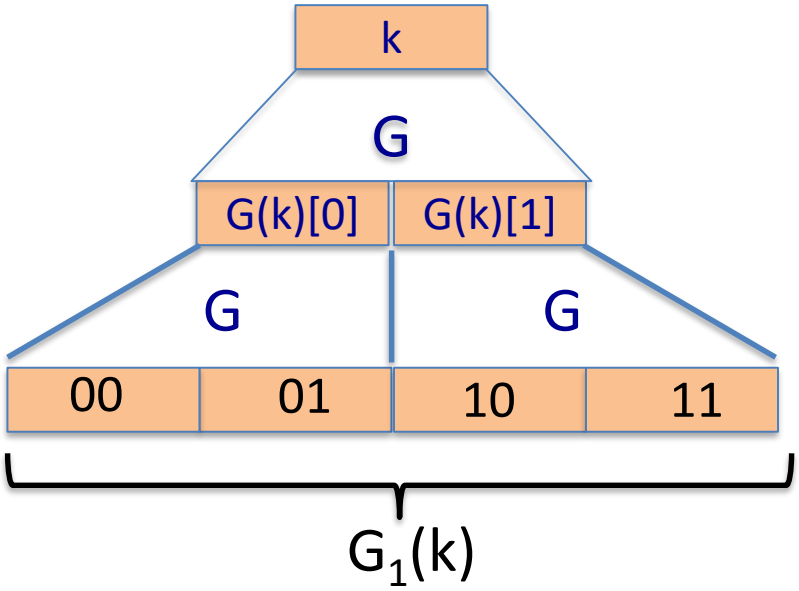
# Extending a PRG

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

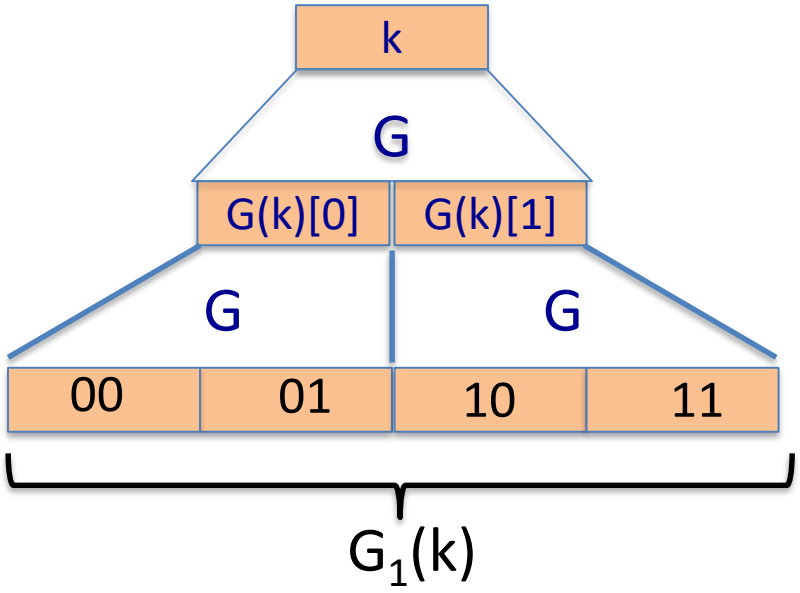
define $G_1: K \longrightarrow K^4$ as $G_1(k) = G(G(k)[0]) \,||\, G(G(k)[1])$

We get a 2-bit PRF:

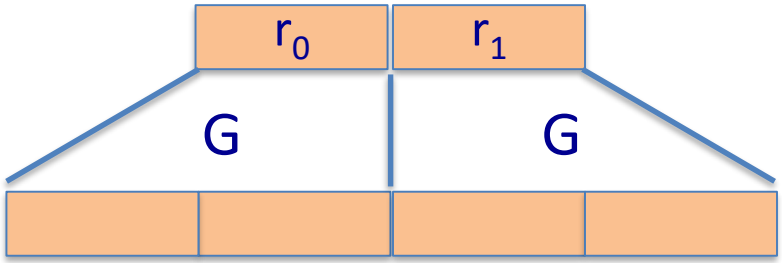
$F(k, x \in \{0,1\}^2) = G_1(k)[x]$



# $G_1$ is a secure PRG

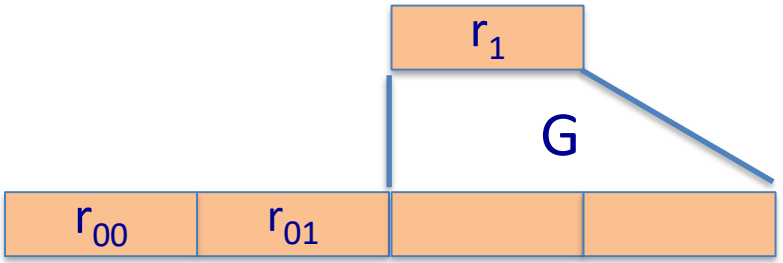



$\approx_p$




$\approx_p$


$\approx_p$

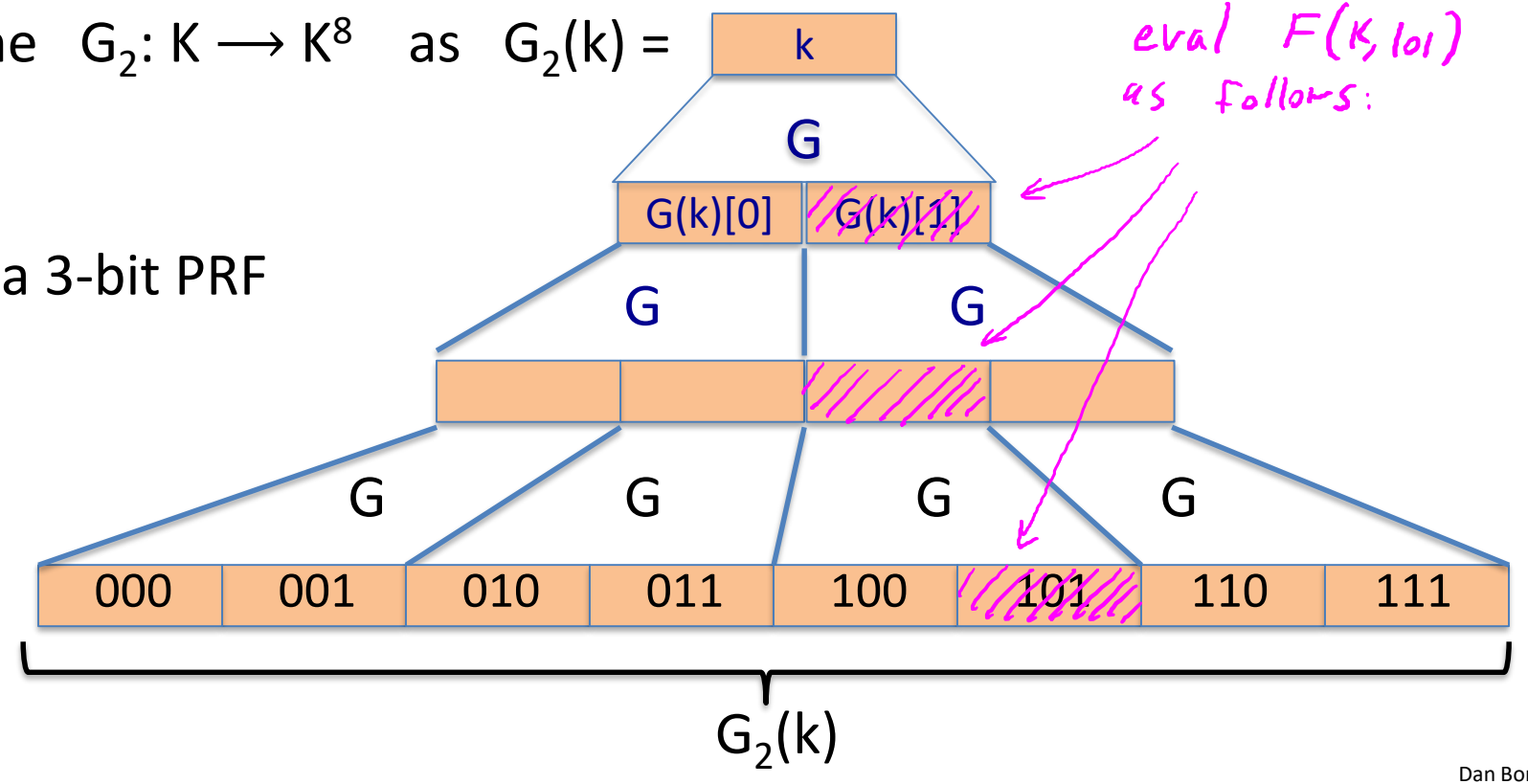


# Extending more

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

define $G_2: K \longrightarrow K^8$ as $G_2(k) =$

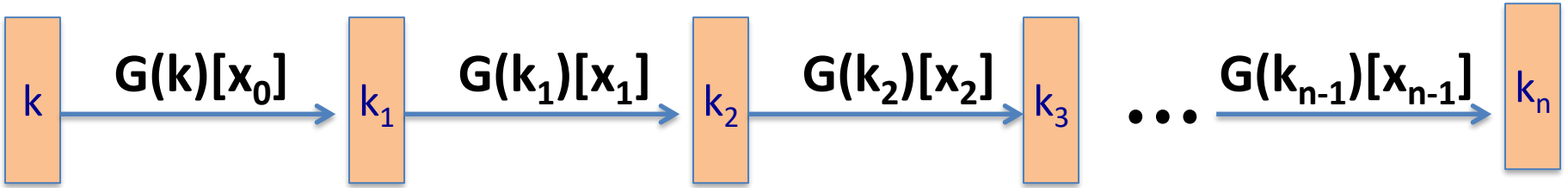
We get a 3-bit PRF



# 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
- ○

# End of Segment