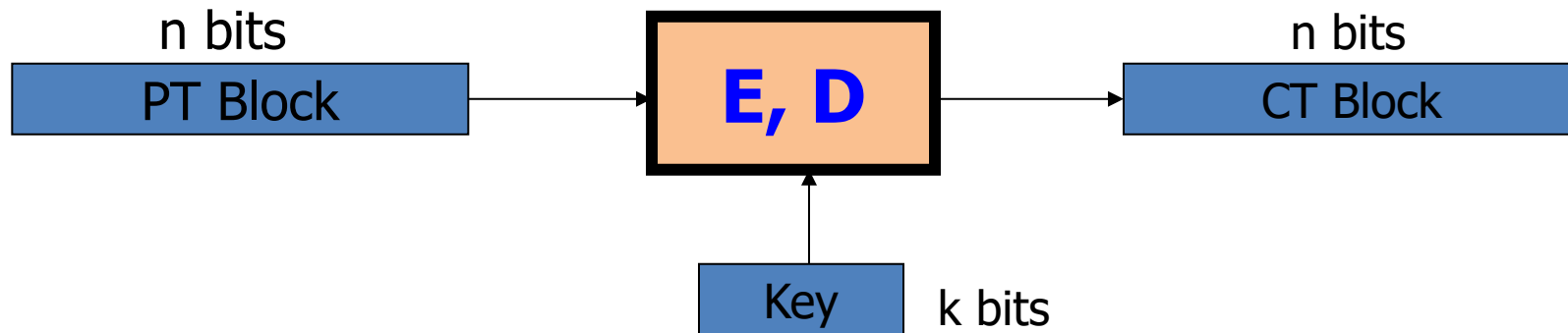


Block ciphers

This slide is made based the online course of Cryptography by Dan Boneh

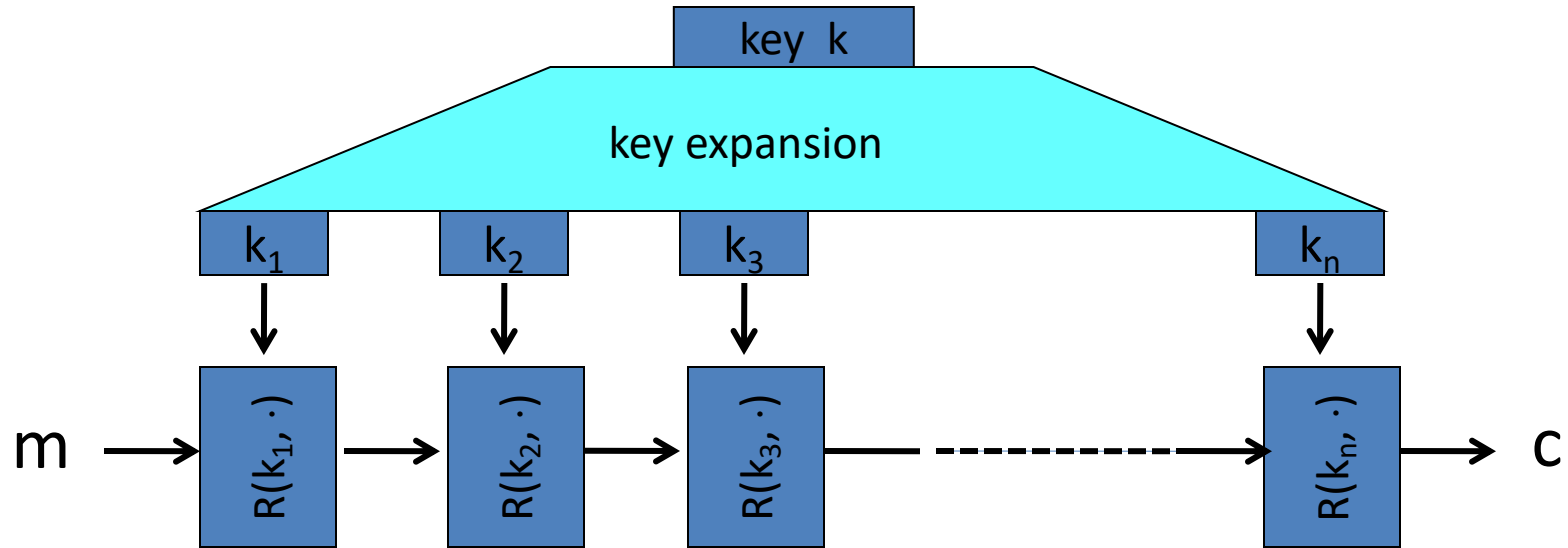
Block ciphers: crypto work horse



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)

	<u>Cipher</u>	<u>Block/key size</u>	<u>Speed (MB/sec)</u>
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 $X = \{0,1\}^{64}$, $K = \{0,1\}^{168}$

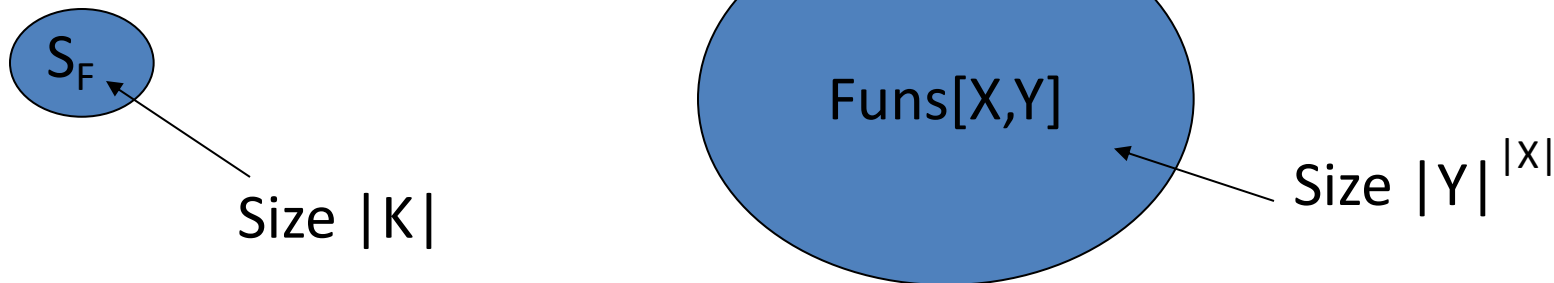
- Functionally, any PRP is also a PRF.
 - A PRP is a PRF where $X=Y$ and is efficiently invertible.

Secure PRFs

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

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

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

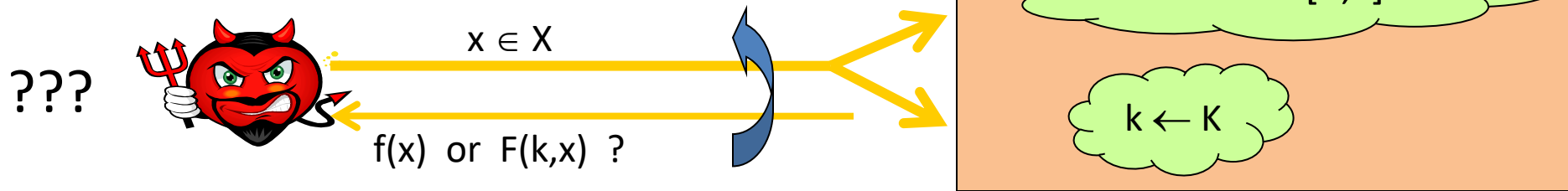


Secure PRFs

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

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

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

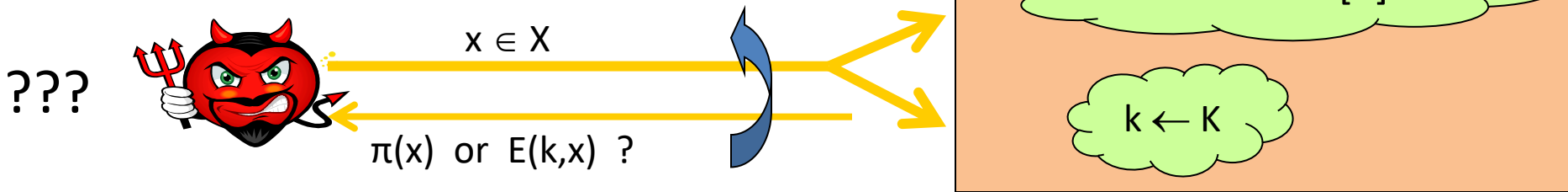


Secure PRPs (secure block cipher)

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

$$\left\{ \begin{array}{l} \text{Perms}[X]: \text{ the set of all } \underline{\text{one-to-one}} \text{ functions from } X \text{ to } Y \\ S_F = \{ E(k, \cdot) \text{ s.t. } k \in K \} \subseteq \text{Perms}[X, Y] \end{array} \right.$$


- Intuition: a PRP is **secure** if
a random function in $\text{Perms}[X]$ 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: $\text{PRF} \Rightarrow \text{PRG}$

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:

$$G(k) = F(k,0) \parallel F(k,1) \parallel \dots \parallel F(k,t-1)$$

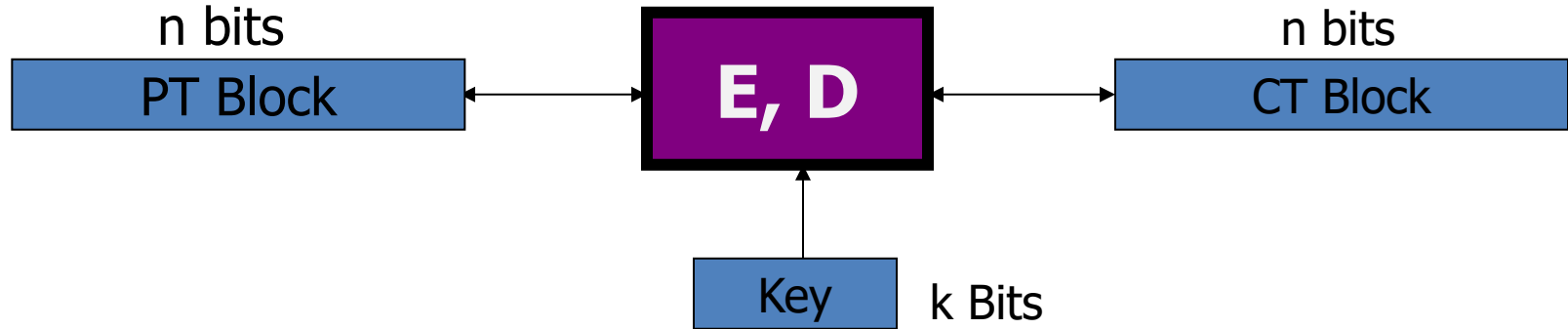
Key property: parallelizable

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

End of Segment

DES

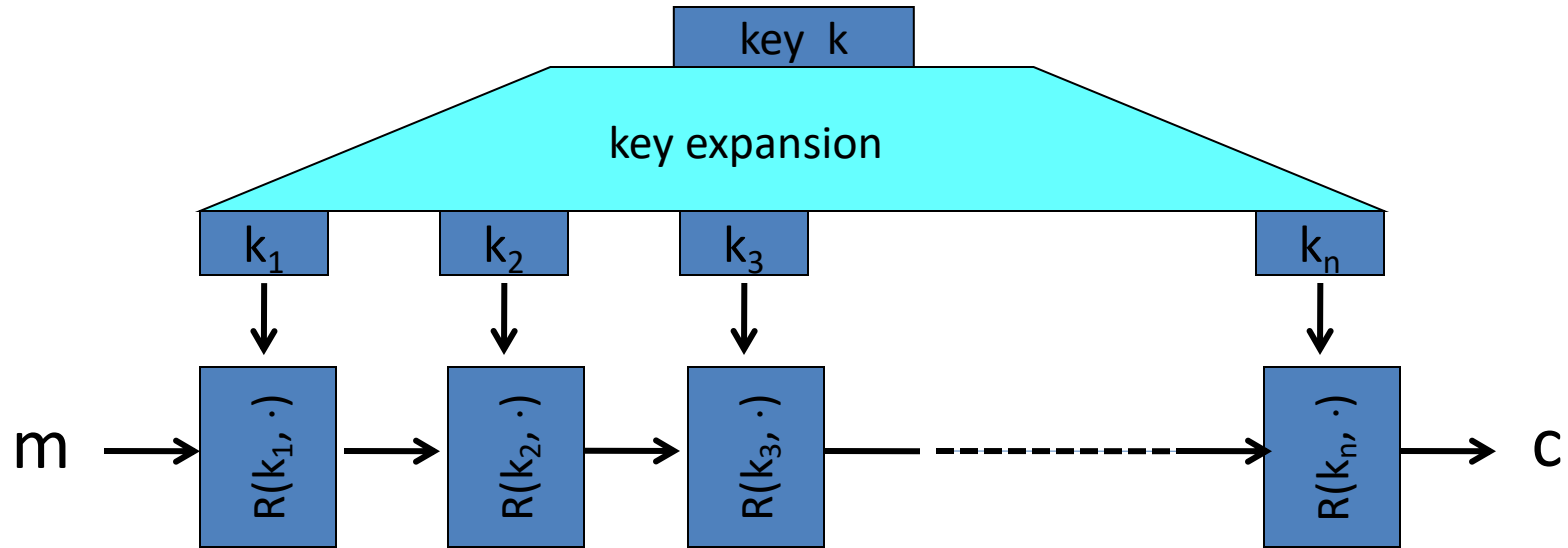
Block ciphers: crypto work horse



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)

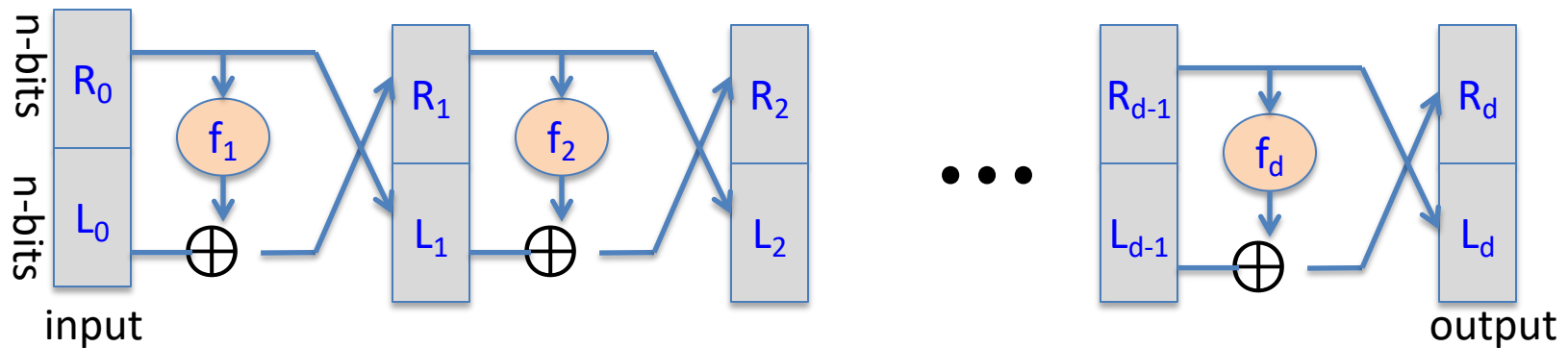
- Early 1970s: Horst Feistel designs Lucifer at IBM
key-len = 128 bits ; block-len = 128 bits
- 1973: NBS asks for block cipher proposals.
IBM submits variant of Lucifer.
- 1976: NBS adopts DES as a federal standard
key-len = 56 bits ; block-len = 64 bits
- 1997: DES broken by exhaustive search
- 2000: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

Widely deployed in banking (ACH) and commerce

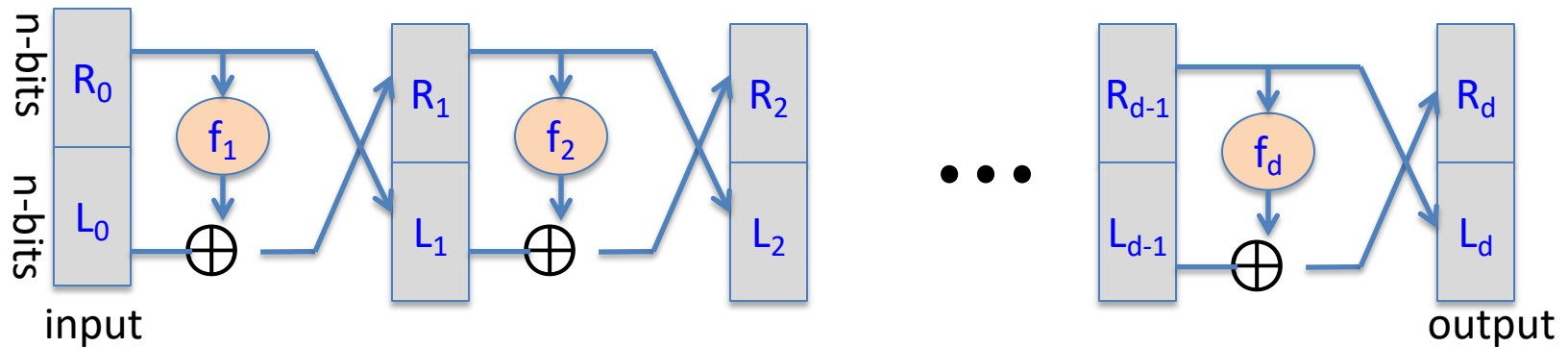
DES: core idea – Feistel Network

Given functions $f_1, \dots, f_d: \{0,1\}^n \rightarrow \{0,1\}^n$

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



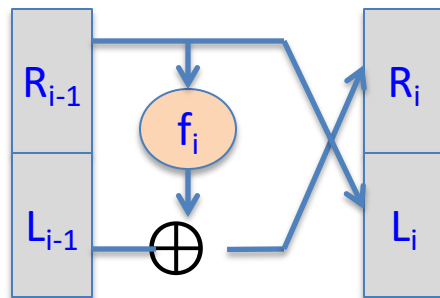
In symbols:
$$\begin{cases} R_i = f_i(R_{i-1}) \oplus L_{i-1} \\ L_i = R_{i-1} \end{cases}$$



Claim: for all $f_1, \dots, f_d: \{0,1\}^n \rightarrow \{0,1\}^n$

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

Proof: construct inverse

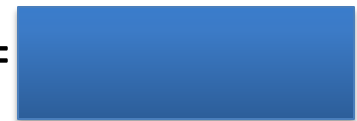


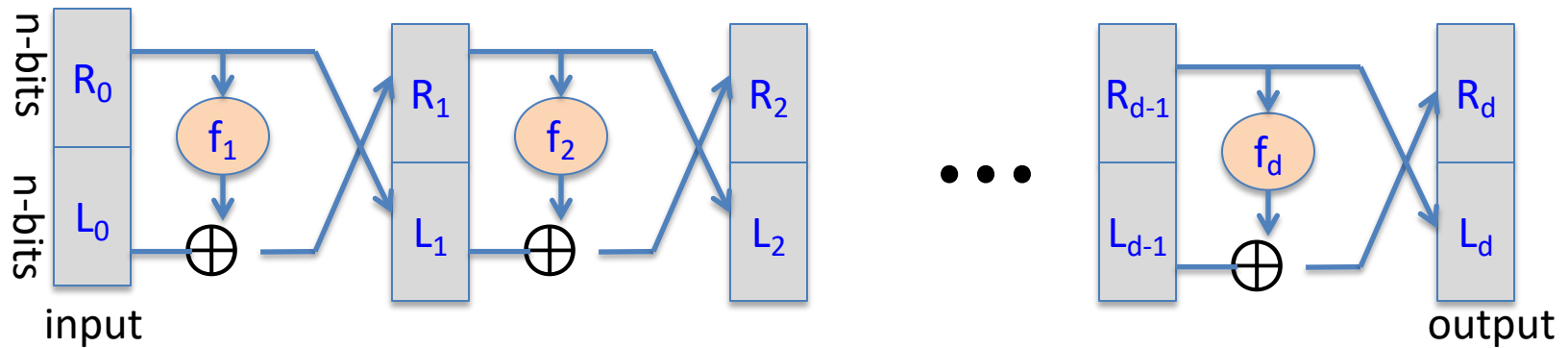
inverse



$$R_{i-1} = L_i$$

$$L_{i-1} =$$

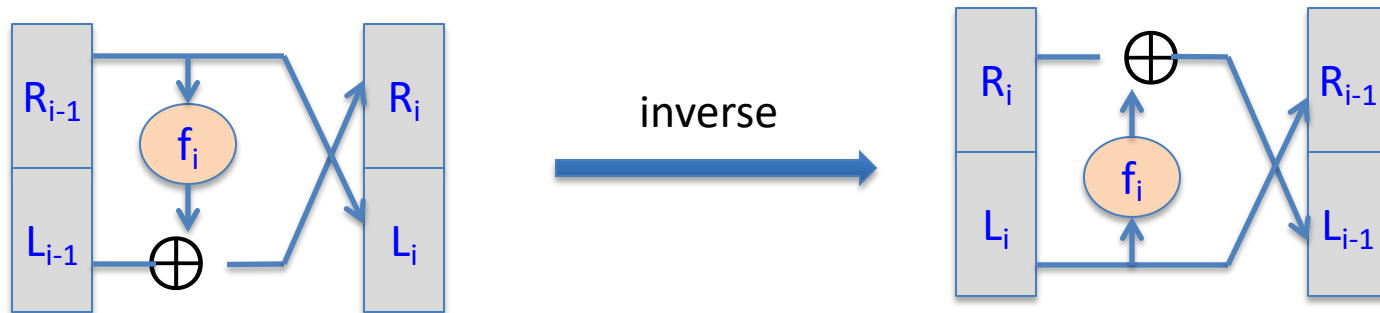




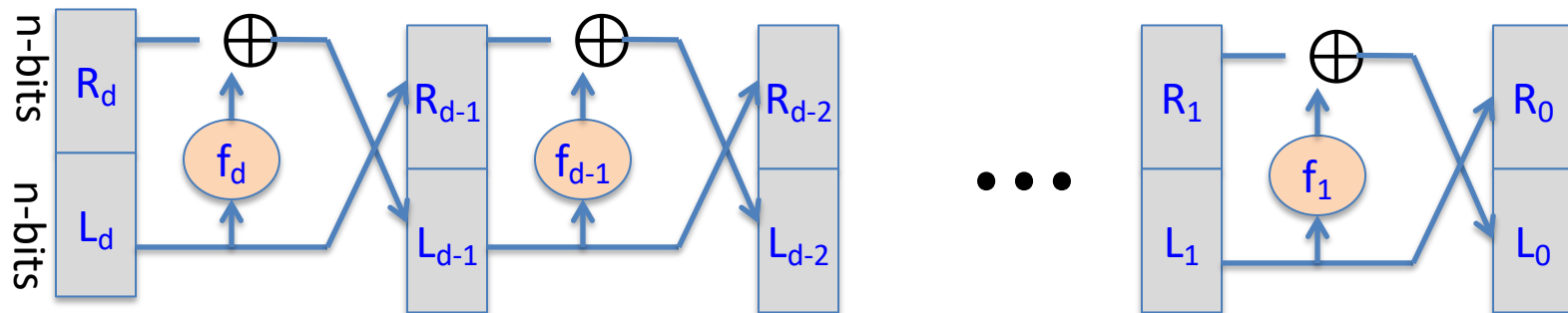
Claim: for all $f_1, \dots, f_d: \{0,1\}^n \rightarrow \{0,1\}^n$

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

Proof: construct inverse



Decryption circuit

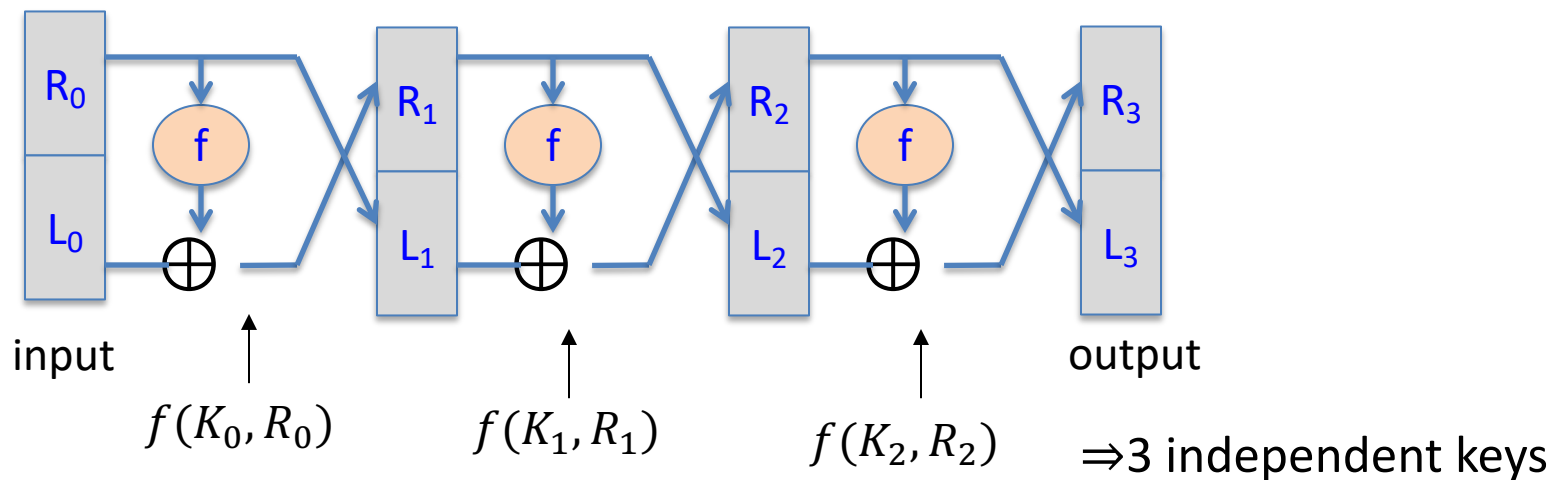


- Inversion is basically the same circuit, with f_1, \dots, f_d applied in reverse order
- General method for building invertible functions (block ciphers) from arbitrary functions.
- Used in many block ciphers ... but not AES

“Thm:” (Luby-Rackoff ‘85):

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

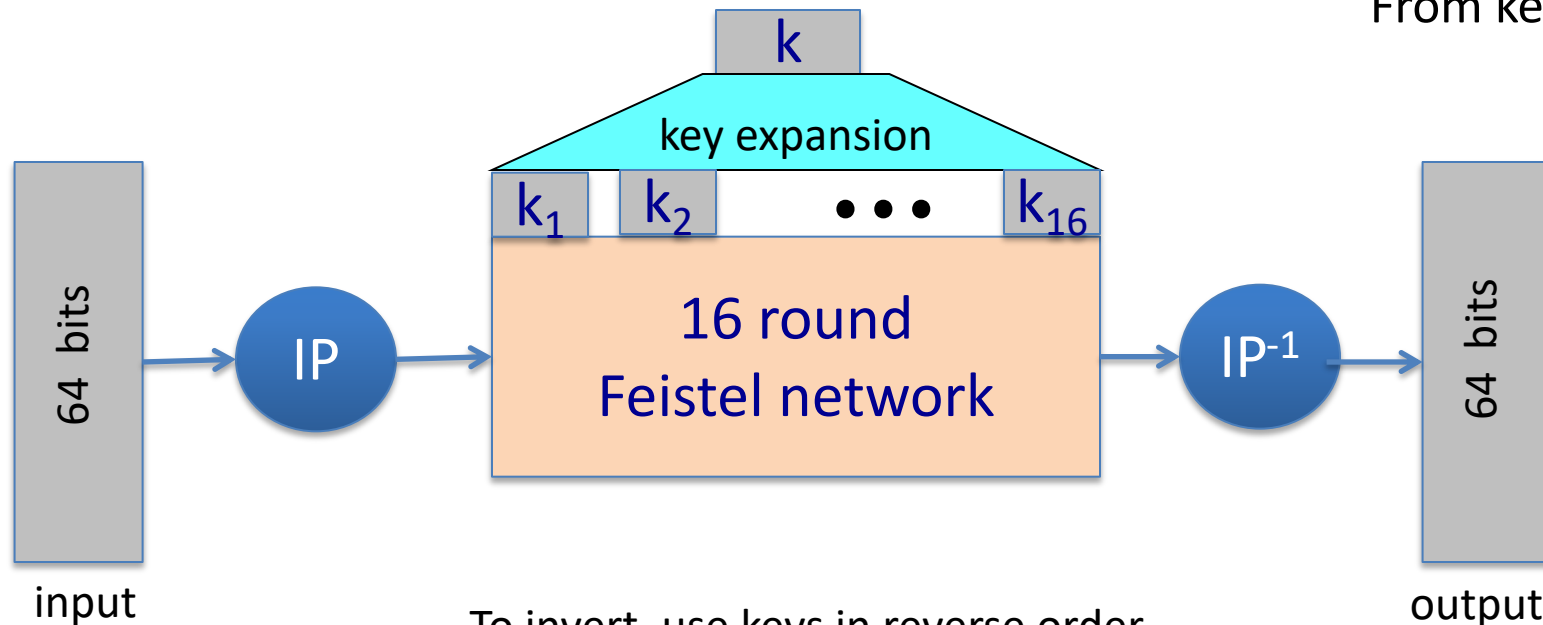
\Rightarrow 3-round Feistel $F: K^3 \times \{0,1\}^{2n} \rightarrow \{0,1\}^{2n}$ a secure PRP



DES: 16 round Feistel network

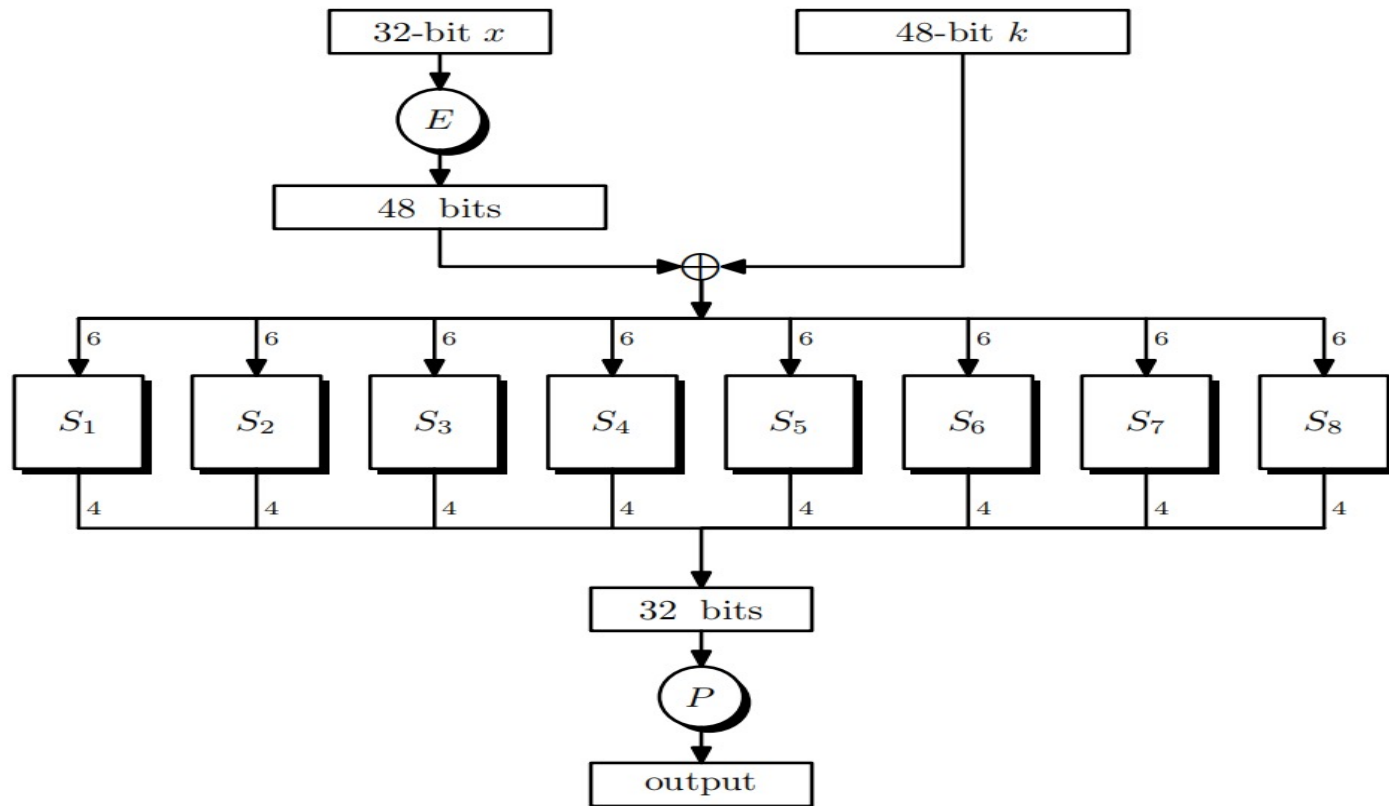
$$f_1, \dots, f_{16}: \{0,1\}^{32} \rightarrow \{0,1\}^{32} \quad , \quad f_i(x) = \mathbf{F}(k_i, x)$$

From key k



To invert, use keys in reverse order

The function $F(k_i, x)$



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

The S-boxes

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

S₅		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

Example: a bad S-box choice

Suppose:

$$S_i(x_1, x_2, \dots, x_6) = (x_2 \oplus x_3, x_1 \oplus x_4 \oplus x_5, x_1 \oplus x_6, x_2 \oplus x_3 \oplus x_6)$$

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

The diagram illustrates the linear function S_i as a matrix multiplication. On the left, a 4x6 matrix A_i is shown in a light orange box with blue text. The rows of the matrix are: $0\ 1\ 1\ 0\ 0\ 0$, $1\ 0\ 0\ 1\ 1\ 0$, $1\ 0\ 0\ 0\ 0\ 1$, and $0\ 1\ 1\ 0\ 0\ 1$. To the right of the matrix is a vertical vector \mathbf{x} in a light orange box with blue text, containing the elements $x_1, x_2, x_3, x_4, x_5, x_6$. A dot operator \cdot is placed between the matrix and the vector. To the right of the dot is an equals sign $=$. On the far right, a light orange box with blue text contains the resulting vector of four expressions: $x_2 \oplus x_3$, $x_1 \oplus x_4 \oplus x_5$, $x_1 \oplus x_6$, and $x_2 \oplus x_3 \oplus x_6$.

We say that S_i is a linear function.

Example: a bad S-box choice

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

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

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

$$k = \begin{pmatrix} k_1 \\ \vdots \\ k_{16} \end{pmatrix} \quad \boxed{B} \begin{matrix} m_1 \\ k \end{matrix} \oplus \boxed{B} \begin{matrix} m_2 \\ k \end{matrix} \oplus \boxed{B} \begin{matrix} m_3 \\ k \end{matrix} = \boxed{B} \begin{matrix} m_1 \oplus m_2 \oplus m_3 \\ k \oplus k \oplus k \end{matrix}$$

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
- S-boxes are 4-to-1 maps
-
-
-

Exhaustive Search Attacks

Exhaustive Search for block cipher key

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

Lemma: Suppose DES is an *ideal cipher*

(2^{56} random invertible functions $\pi_1, \pi_2, \dots, \pi_{2^{56}}: \{0,1\}^{64} \rightarrow \{0,1\}^{64}$)

Then $\forall m, c$ there is at most one key k s.t. $c = \text{DES}(k, m)$

Proof:

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

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

Exhaustive Search for block cipher key

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

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. $\text{DES}(k, m_i) = c_i$ 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 \$)

\Rightarrow 56-bit ciphers should not be used !! (128-bit key $\Rightarrow 2^{72}$ days)

Strengthening DES against ex. search

Method 1: Triple-DES

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

$$3E((k_1, k_2, k_3), m) = E(k_1, D(k_2, E(k_3, m)))$$
$$K_1 = K_2 = K_3 \Rightarrow \text{single DES}$$

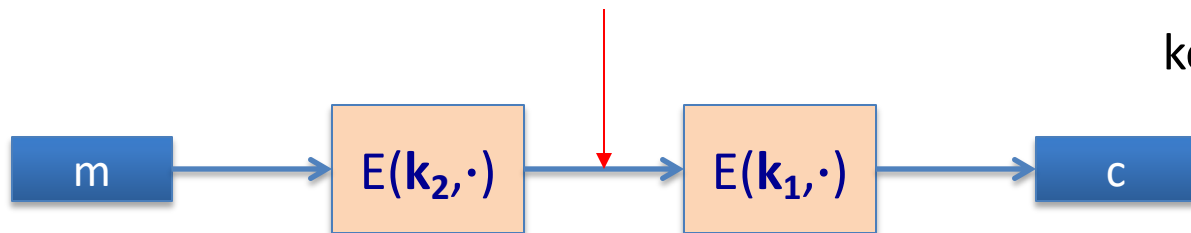
For 3DES: key-size = $3 \times 56 = 168$ bits. 3x slower than DES.

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

Why not double DES?

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

key-len = 112 bits for DES



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

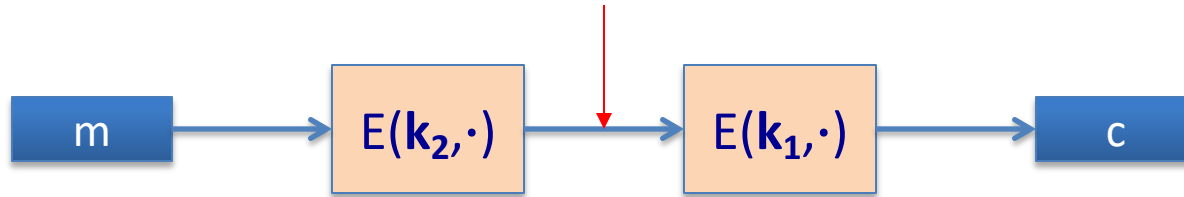
Attack: $M = (m_1, \dots, m_{10})$, $C = (c_1, \dots, c_{10})$.

step 1: build table.

sort on 2nd column

$k^0 = 00\dots00$	$E(k^0, M)$	} 2^{56} entries
$k^1 = 00\dots01$	$E(k^1, M)$	
$k^2 = 00\dots10$	$E(k^2, M)$	
\vdots	\vdots	
$k^N = 11\dots11$	$E(k^N, M)$	

Meet in the middle attack



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

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

$k^0 = 00 \dots 00$	$E(k^0, M)$
$k^1 = 00 \dots 01$	$E(k^1, M)$
$k^2 = 00 \dots 10$	$E(k^2, M)$
\vdots	\vdots
$k^N = 11 \dots 11$	$E(k^N, M)$

Meet in the middle attack

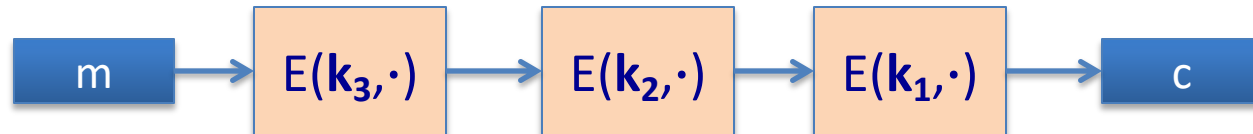


$$\text{Time} = 2^{56} \log(2^{56}) + 2^{56} \log(2^{56}) < 2^{63} \ll 2^{112}, \quad \text{space} \approx 2^{56}$$

Build+sort table

Search in table

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



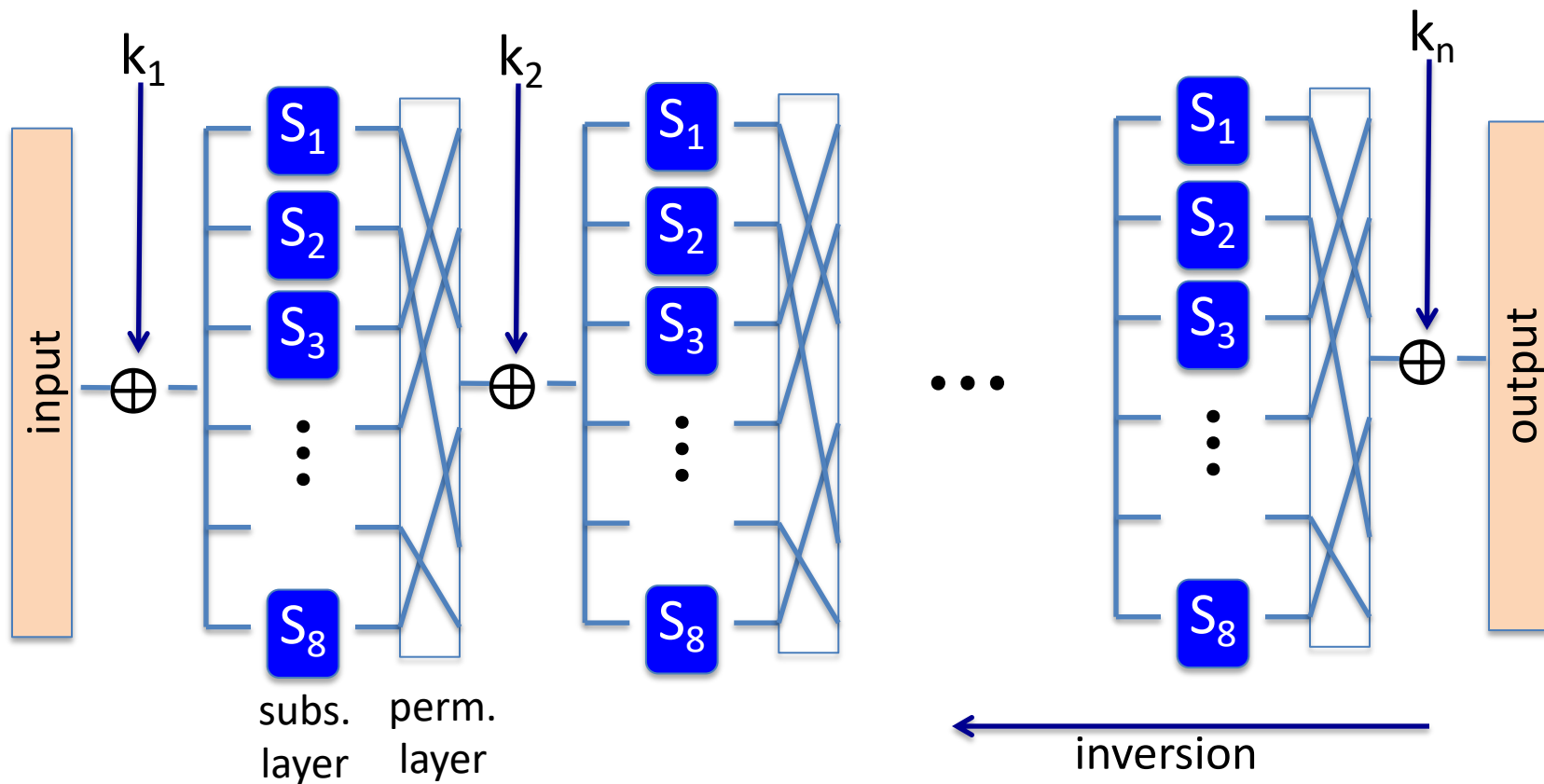
AES

The AES process

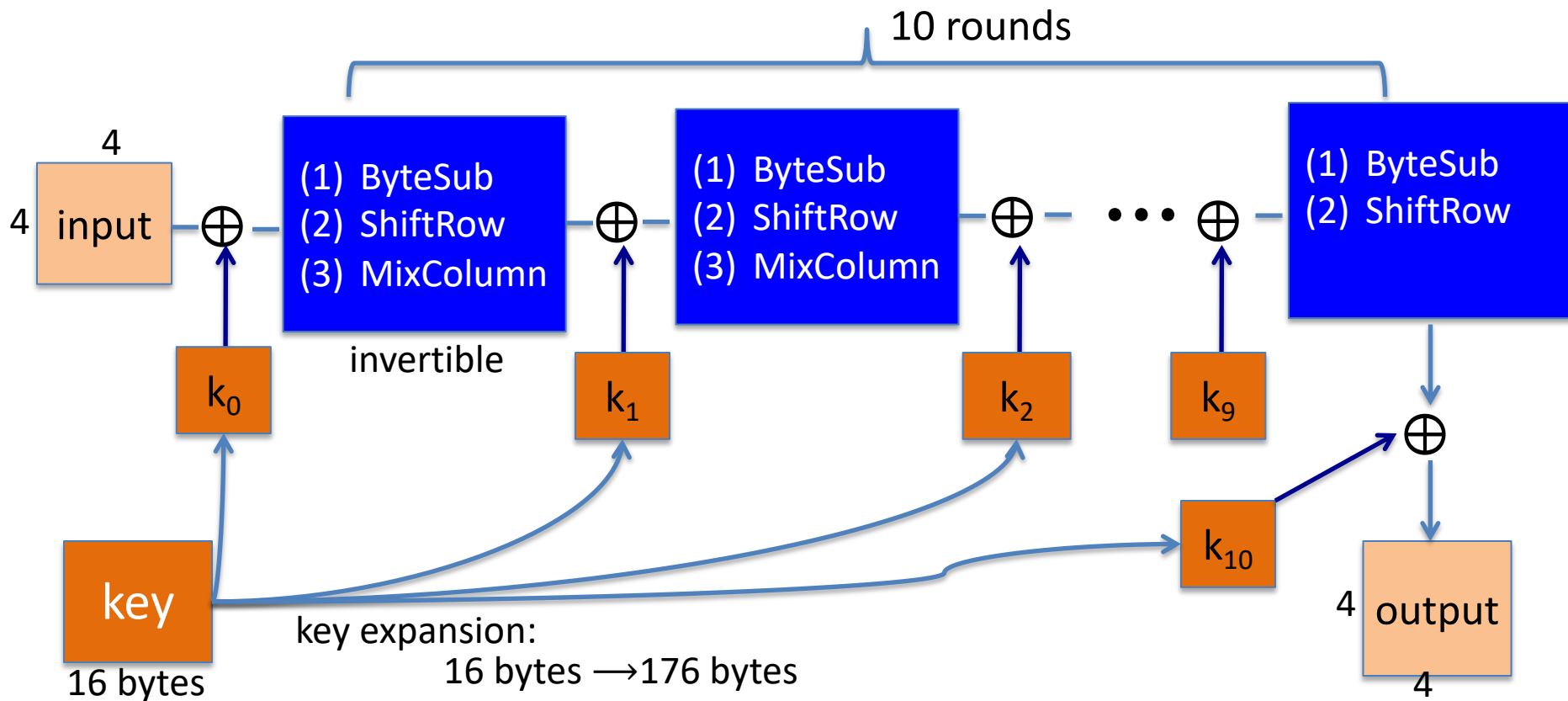
- 1997: NIST publishes request for proposal
- 1998: 15 submissions. Five claimed attacks.
- 1999: NIST chooses 5 finalists
- 2000: NIST chooses Rijndael as AES (designed in Belgium)

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

AES is a Subs-Perm network (not Feistel)



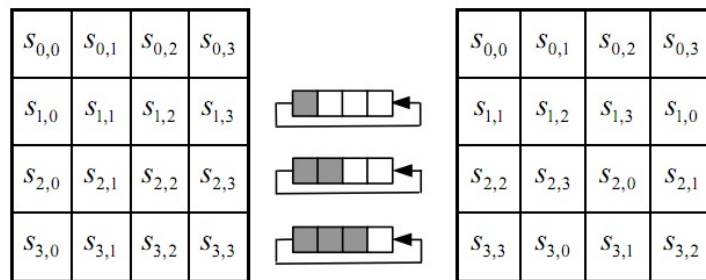
AES-128 schematic



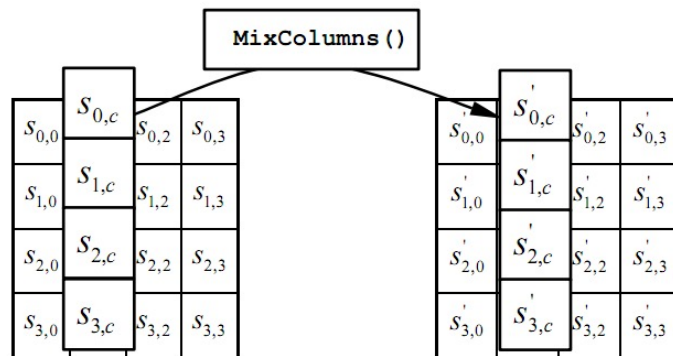
The round function

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

- **ShiftRows:**



- **MixColumns:**



Code size/performance tradeoff

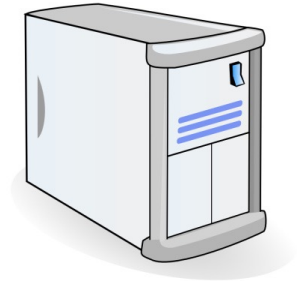
	Code size	Performance
Pre-compute round functions (24KB or 4KB)	largest	fastest: table lookups and xors
Pre-compute S-box only (256 bytes)	smaller	slower
No pre-computation	smallest	slowest

Example: Javascript AES

AES in the browser:



AES library (6.4KB)
no pre-computed tables



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 $\approx 2^{99}$

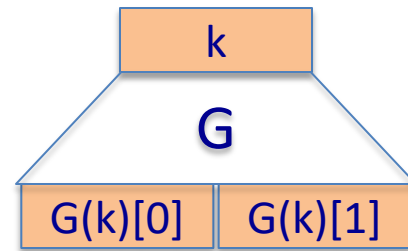
Block ciphers from PRGs

Can we build a PRF from a PRG?

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

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

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



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

Proof pp.146 - 149

Can we build a PRF with a larger domain?

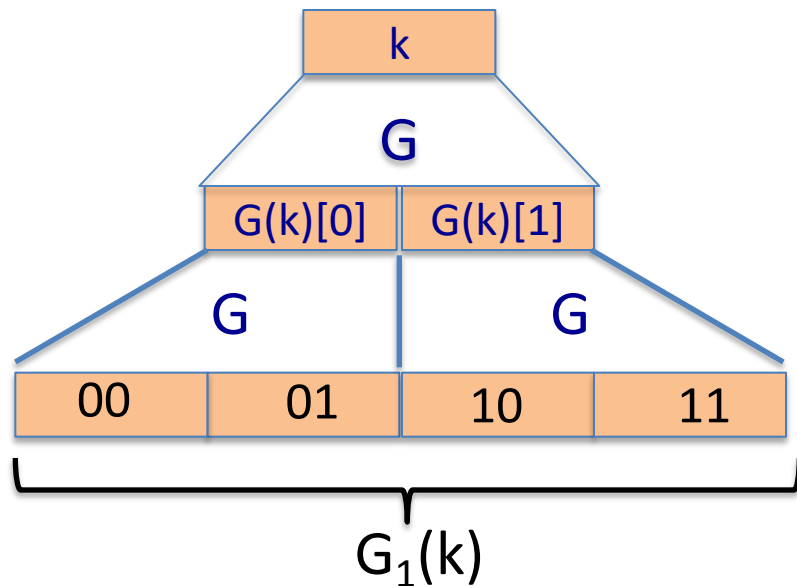
Extending a PRG

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

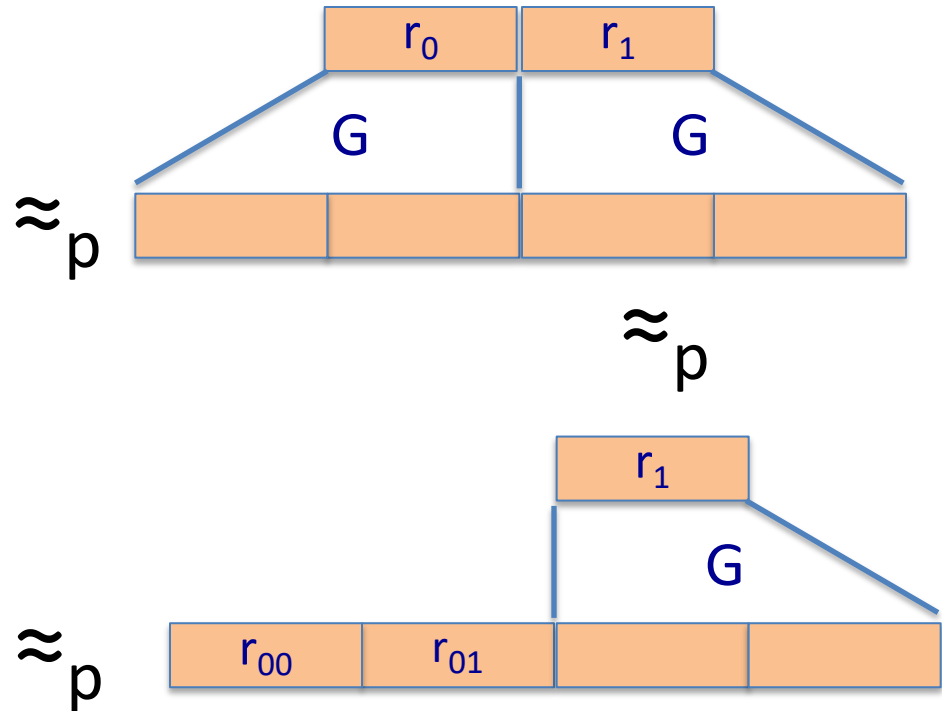
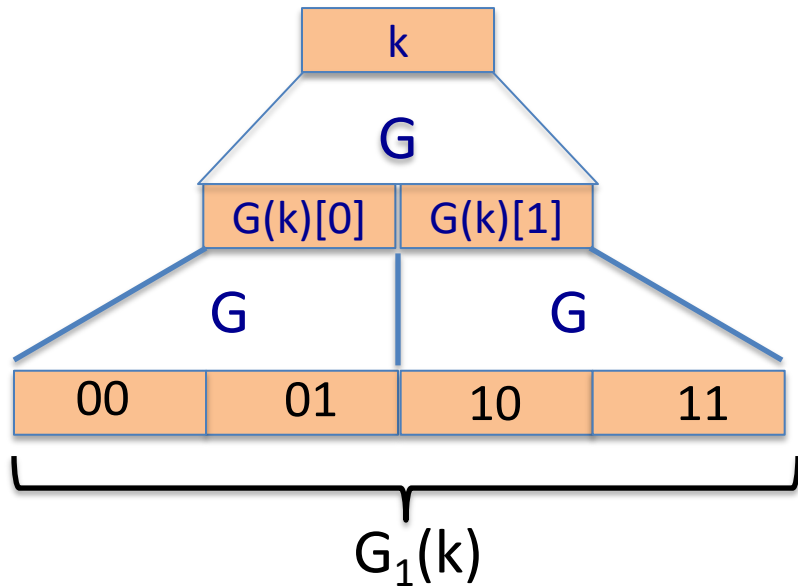
define $G_1: K \rightarrow K^4$ as $G_1(k) = G(G(k)[0]) \parallel G(G(k)[1])$

We get a 2-bit PRF:

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



G_1 is a secure PRG

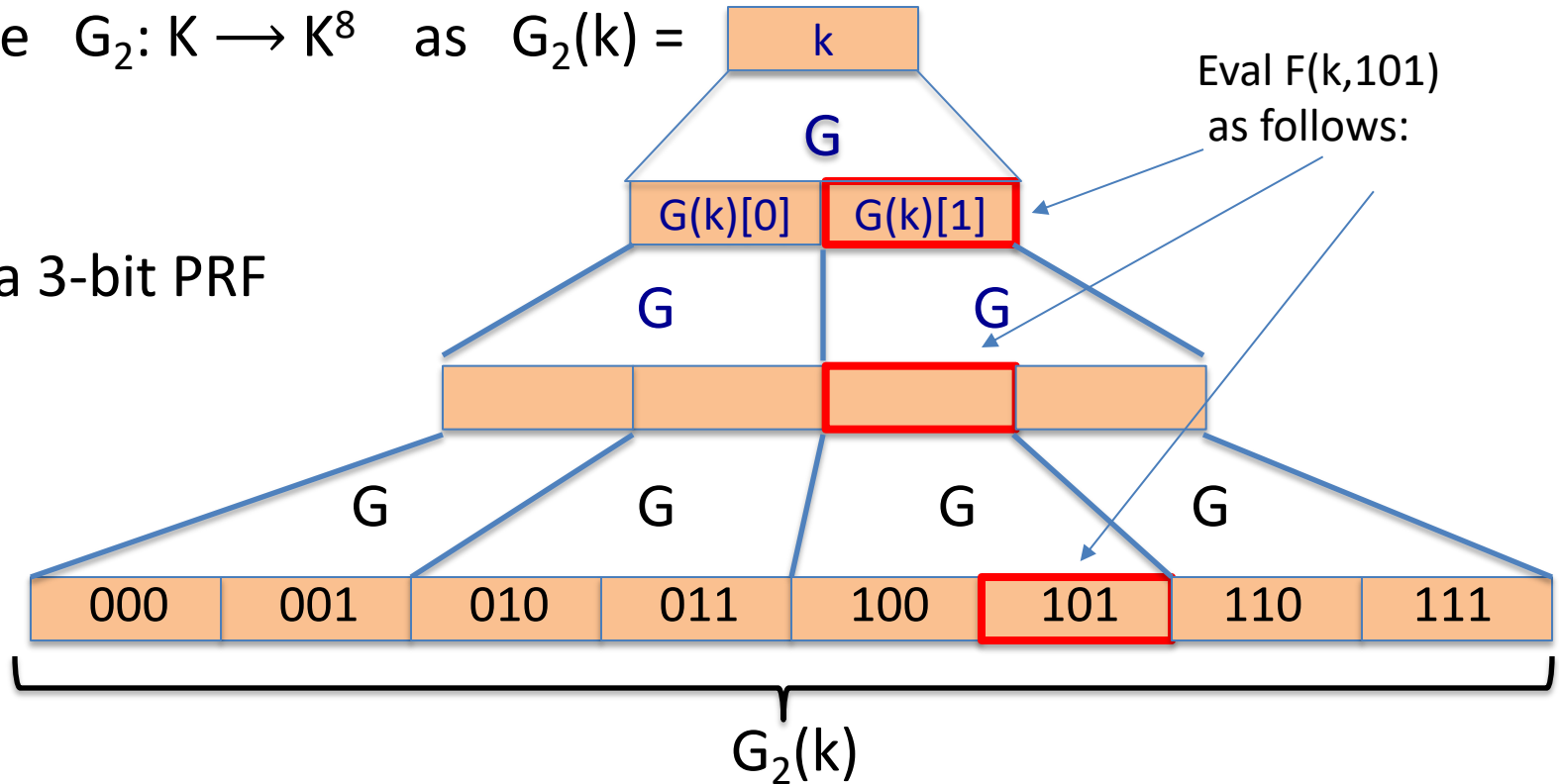


Extending more

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

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

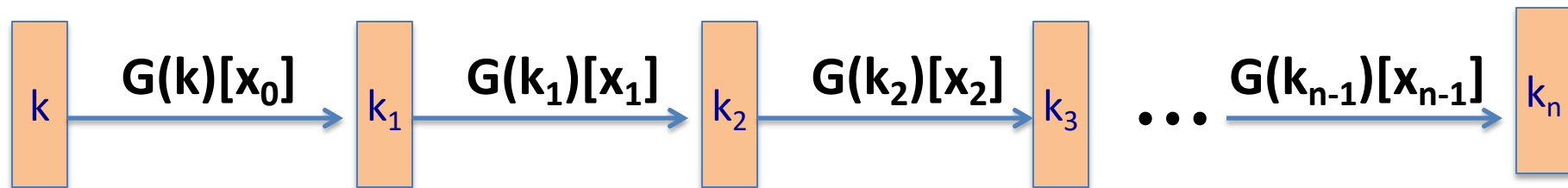
We get a 3-bit PRF



Extending even more: the GGM PRF

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

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



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

Not used in practice due to slow performance.

End of Segment

Bit Commitment Scheme

Problem Description

Alice and Bob are going out on a date. Alice wants to see one movie and Bob wants to see another. They decide to flip a random coin to choose the movie. If the coin comes up “heads” they will go to Alice’s choice; otherwise, they will go to Bob’s choice. When Alice and Bob are in close proximity this is easy: one of them, say Bob, flips a coin and they both verify the result. When they are far apart and are speaking on the phone this is harder. Bob can flip a coin on his side and tell Alice the result, but Alice has no reason to believe the outcome. Bob could simply claim that the coin came up “tails” and Alice would have no way to verify this. Not a good way to start a date.

Solution

1. Bob commit to a bit $b \in \{0, 1\}$
2. Later, Bob can open the commitment and convince Alice that b was the value he committed to
3. Committing to a bit b results in a commitment string c , that Bob sends to Alice, and an opening string s that Bob uses for opening the commitment later.

Properties for Commitment Scheme

- **Hiding:** The commitment string c reveals no information about the committed bit b . More precisely, the distribution on c when committing to the bit 0 is indistinguishable from the distribution on c when committing to the bit 1. In the bit commitment scheme we present, the hiding property depends on the security of a certain PRG G .
- **Binding:** Let c be a commitment string output by Bob. If Bob can open the commitment as some $b \in \{0, 1\}$ then he cannot open it as \bar{b} . This ensures that once Bob commits to a bit b he can open it as b and nothing else. In the commitment scheme we present the binding property holds unconditionally.

Protocol

$G : \mathcal{S} \rightarrow \mathcal{R}$ be a secure PRG where $|\mathcal{R}| \geq |\mathcal{S}|^3$

Bob commits to bit $b_0 \in \{0, 1\}$:

Step 1: Alice chooses a random $r \in \mathcal{R}$ and sends r to Bob.

Step 2: Bob chooses a random $s \in \mathcal{S}$ and computes $c \leftarrow \text{com}(s, r, b_0)$
where $\text{com}(s, r, b_0)$ is the following function:

$$c = \text{com}(s, r, b_0) := \begin{cases} G(s) & \text{if } b_0 = 0, \\ G(s) \oplus r & \text{if } b_0 = 1. \end{cases}$$

Bob outputs c as the commitment string and uses s as the opening string.

Alice accepts the opening if $c = \text{com}(s, r, b_0)$ and rejects otherwise.