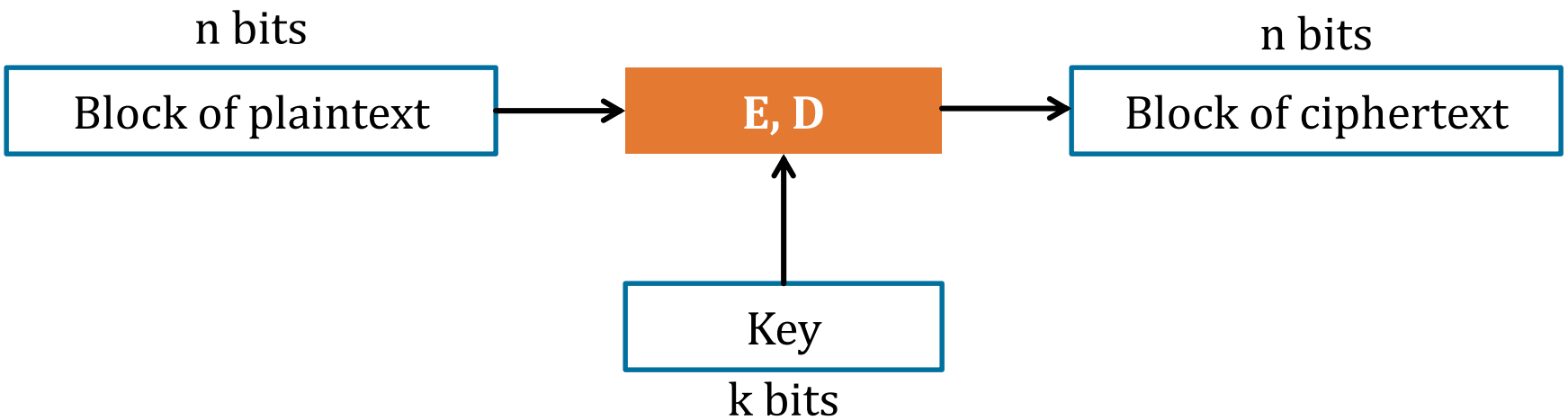
Cryptography: Block Ciphers

Edward J. Schwartz
Carnegie Mellon University

Credits:
Slides originally designed by David Brumley. *Many* other slides are from Dan Boneh's June 2012 Coursera crypto class.

What is a block cipher?

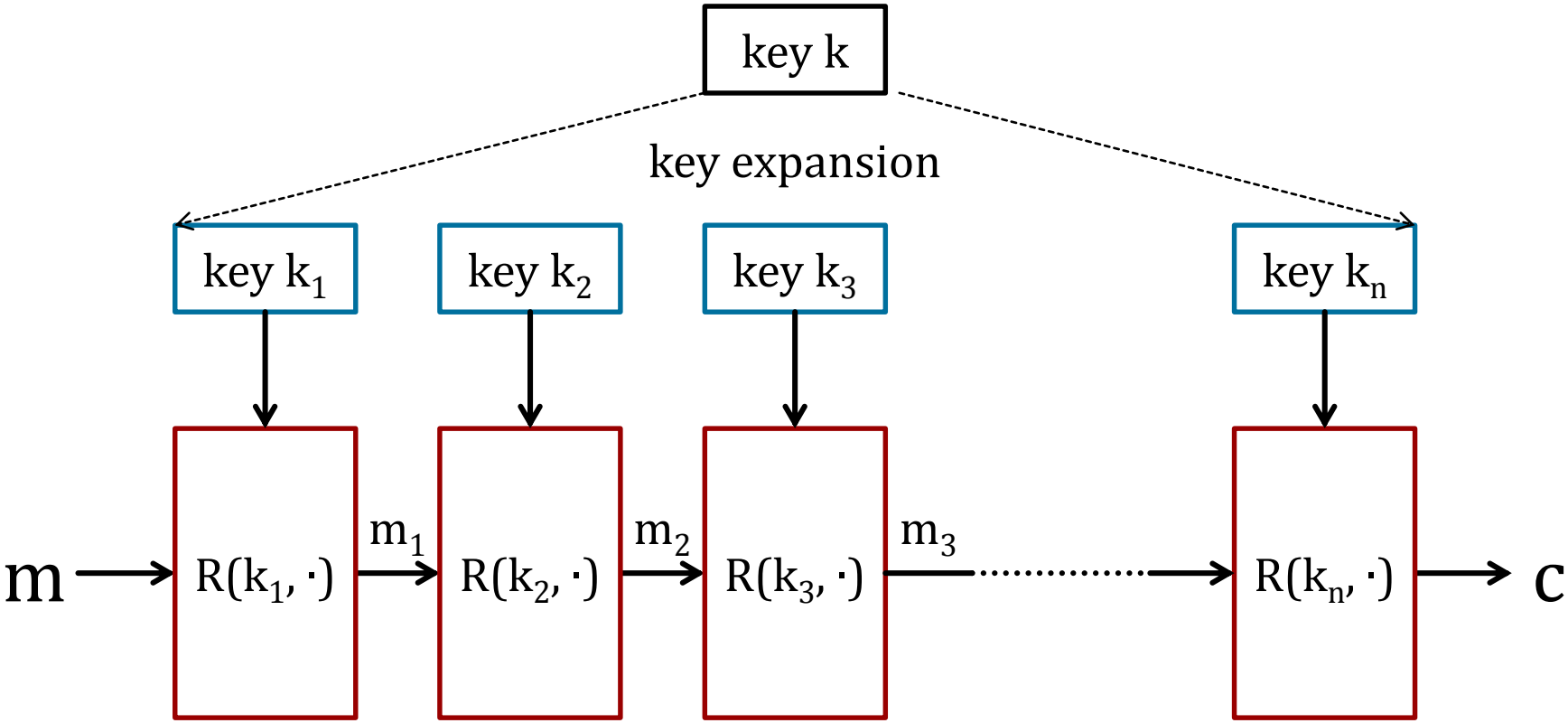
Block ciphers are the crypto work horse



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*

Ex: 3DES (n=48), AES128 (n=10)

Performance: Stream vs. block ciphers

Crypto++ 5.6.0 [Wei Dai]

AMD Opteron, 2.2 GHz (Linux)

	Cipher	Block/key size	Throughput [MB/s]
Stream	RC4		126
	Salsa20/12		643
	Sosemanuk		727
Block	3DES	64/168	13
	AES128	128/128	109

Block ciphers

The Data Encryption Standard (DES)

History of DES

- **1970s**: Horst Feistel designs Lucifer at IBM
  key = 128 bits, block = 128 bits

- **1973:** NBS asks for block cipher proposals.
  IBM submits variant of Lucifer.

- **1976:** NBS adopts DES as federal standard
  key = 56 bits, block = 64 bits

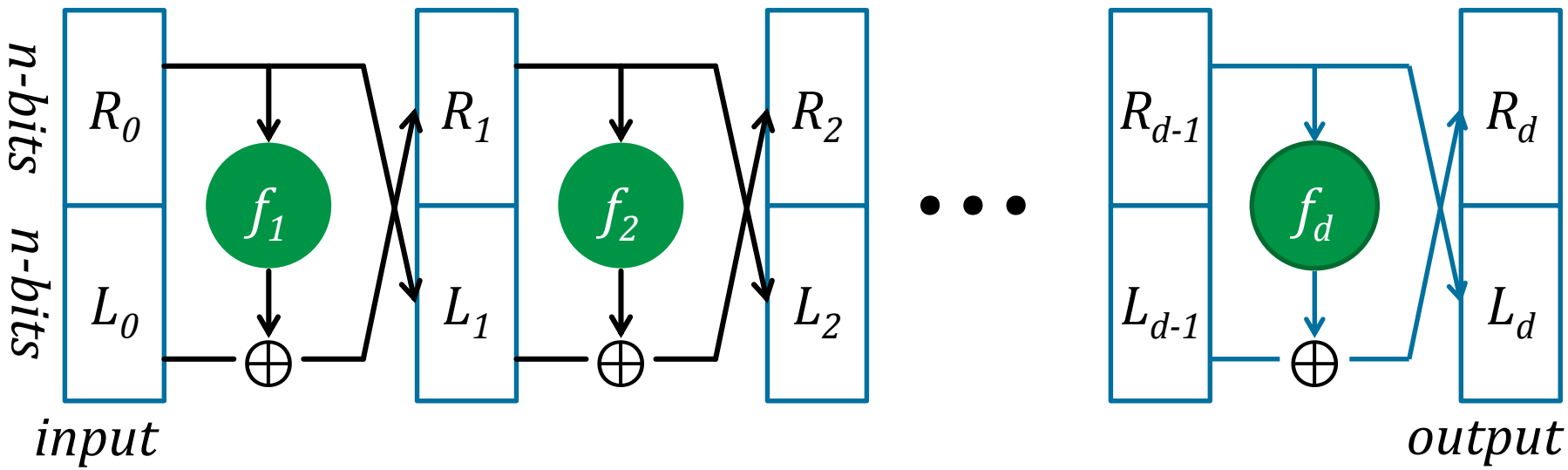
- **1997:** DES broken by exhaustive search

- **2000:** NIST adopts Rijndael as AES to replace DES. AES currently widely deployed in banking, commerce and Web

DES: core idea – Feistel network

Given <u>one-way</u> functions $f_1, \ldots, f_d : \{0,1\}^n \to \{0,1\}^n$

Goal: build <u>invertible</u> function $F : \{0,1\}^{2n} \to \{0,1\}^{2n}$



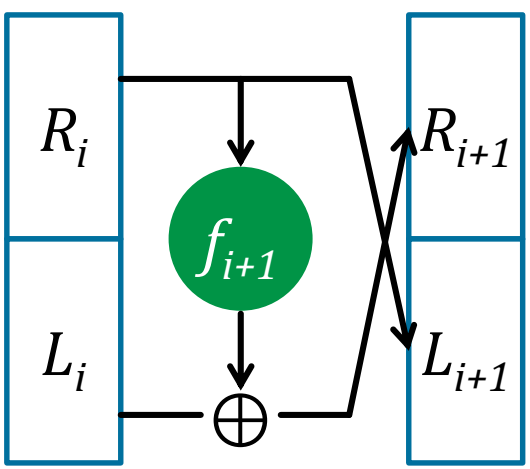
In symbols: $\begin{cases} R_i = f_i(R_{i-1}) \oplus L_{i-1} \\ L_i = R_{i-1} \end{cases}$

Feistel network - inverse

Claim: $f_1, \ldots, f_d : \{0,1\}^n \to \{0,1\}^n$

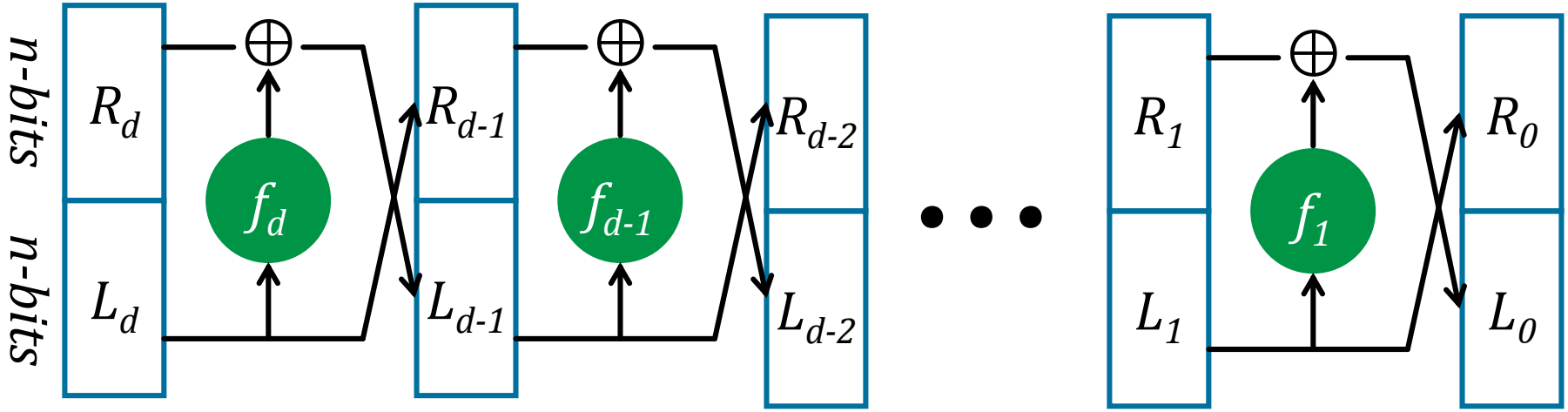
Feistel function F is invertible $F : \{0,1\}^{2n} \to \{0,1\}^{2n}$

Proof: construct inverse $\begin{cases} R_i = \\ L_i = \end{cases}$



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

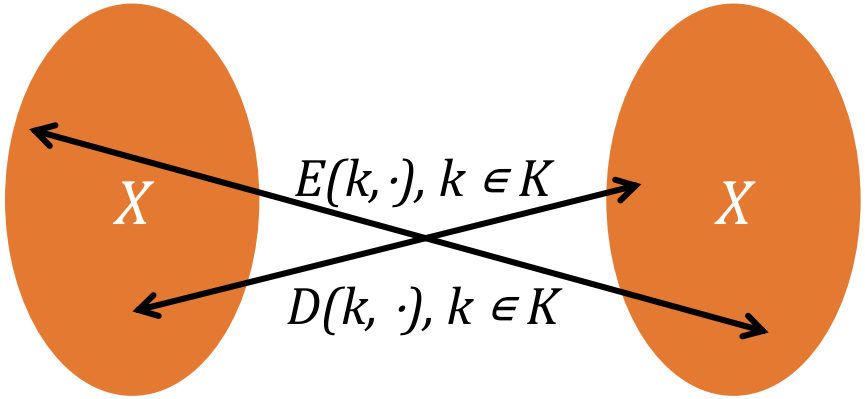
Recall from Last Time: Block Ciphers are (Modeled As) PRPs

Pseudo Random <u>Permutation</u> (PRP) defined over (K,X)

$$E : K \times X \to X$$

such that:

1. Exists "efficient" deterministic algorithm to evaluate *E(k,x)*
2. The function *E(k, ·)* is one-to-one
3. Exists "efficient" inversion algorithm *D(k,y)*

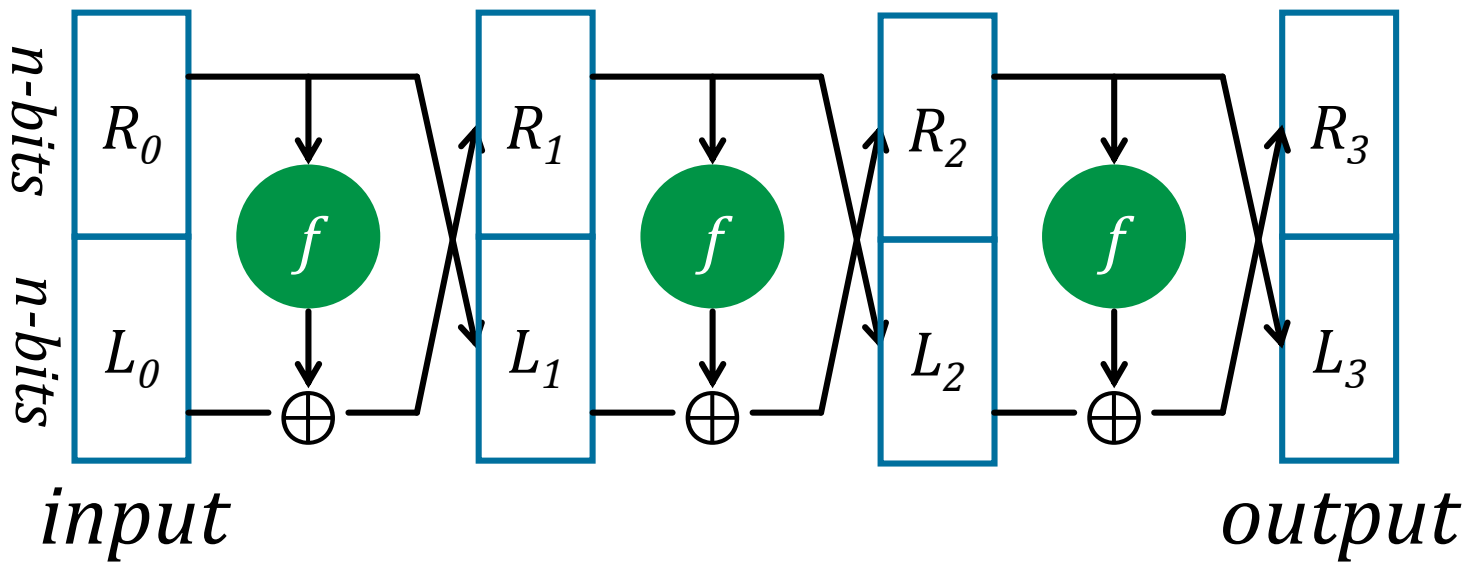


Luby-Rackoff Theorem (1985)

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

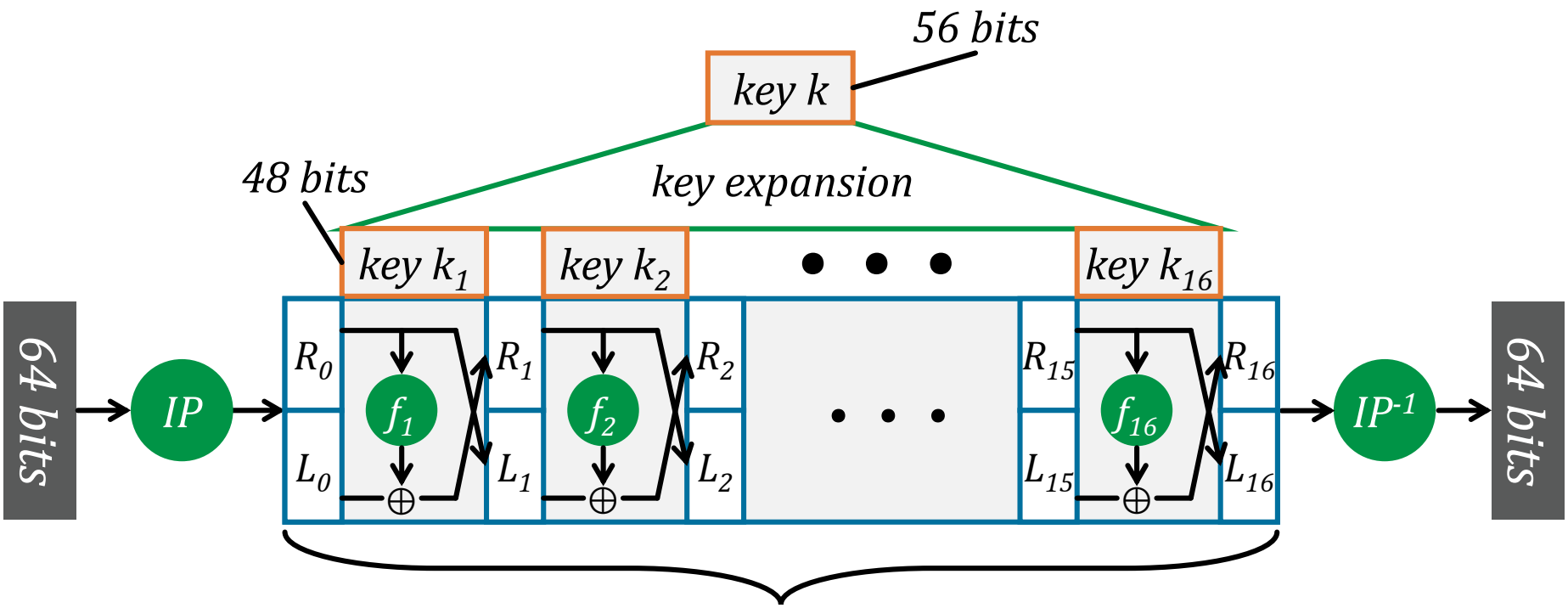
$\Rightarrow$ 3-round Feistel $F : K^3 \times \{0,1\}^{2n} \to \{0,1\}^{2n}$

is a secure PRP



DES: 16 round Feistel network

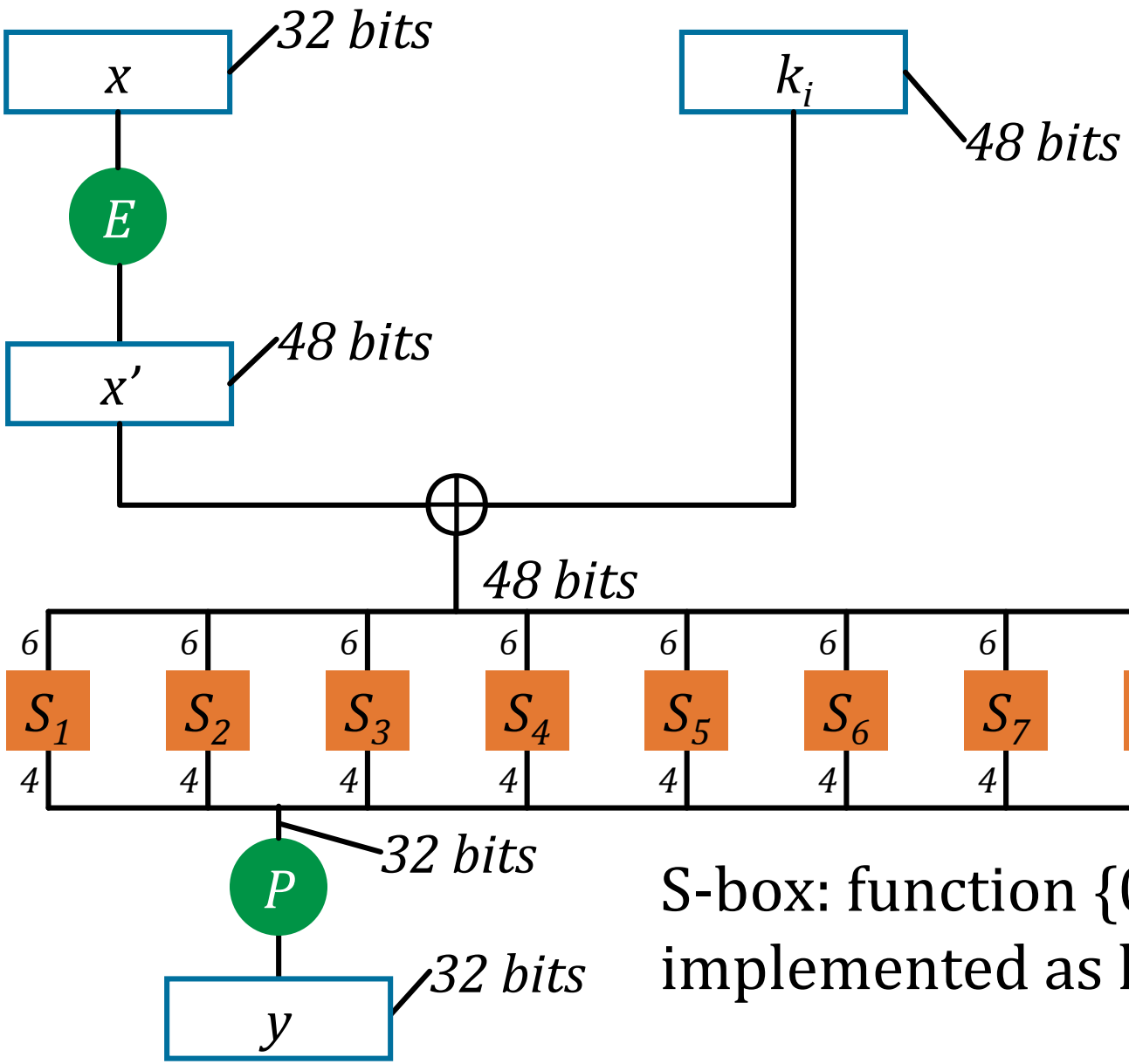
$$f_1, \ldots, f_{16} : \{0,1\}^{32} \to \{0,1\}^{32} \text{ and } f_i(x) = \mathbf{F}(k_i, x)$$



16 round Feistel network

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 lookup table.

The S-boxes

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

e.g., $011011 \longrightarrow 1001$

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

The S-boxes

- Alan Konheim (one of the designers of DES) commented, "We sent the S-boxes off to Washington. They came back and were all different.“
- 1990: (Re-)Discovery of differential cryptanalysis
  - DES S-boxes resistant to differential cryptanalysis!
  - Both IBM and NSA knew of attacks, but they were classified

Block cipher attacks

Exhaustive Search for block cipher key

Goal: given a few input output pairs

$(m_i, c_i = E(k, m_i))$ $i=1,..,n$ find key k.

Attack: Brute force to find the key k.

Homework: What is the probability that the key *k* found with one <m,c> pair is correct? For two pairs?

DES challenge

msg = "The unknown messages is: XXXXXXXX..."

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 $i=1,2,3$

Proof: Reveal $DES^{-1}(k, c_4)$

1976	DES adopted as federal standard		
1997	Distributed search	3 months	
1998	EFF deep crack	3 days	$250,000
1999	Distributed search	22 hours	
2006	COPACOBANA (120 FPGAs)	7 days	$10,000

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

Strengthening DES

Method 1: **Triple-DES**

Let $E : K \times M \longrightarrow M$ be a block cipher

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

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

3DES

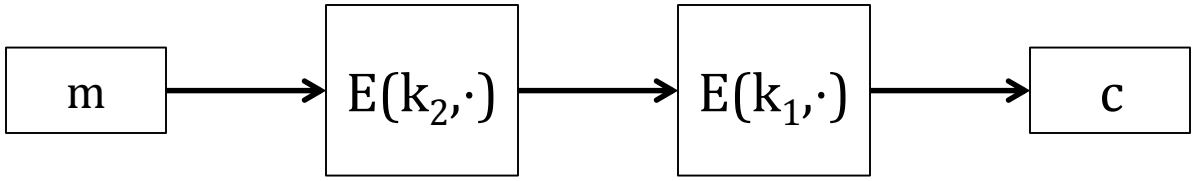
- Key-size: 3×56 = 168 bits
- 3×slower than DES
- Simple attack in time: $\approx 2^{118}$

$k_1 = k_2 = k_3 \Rightarrow$ DES

Why not 2DES?

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

key-len = 112 bits for 2DES

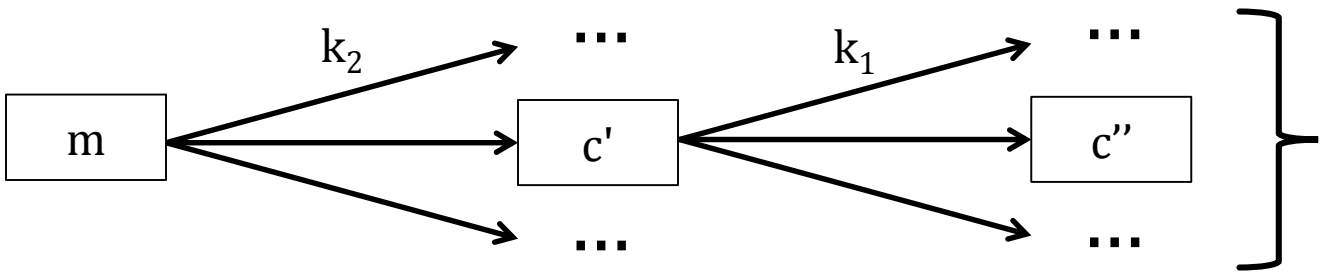


Naïve Attack: $M = (m_1,\ldots, m_{10})$, $C = (c_1,\ldots,c_{10})$.

For each $k_2 \in \{0,1\}^{56}$:

For each $k_1 \in \{0,1\}^{56}$:

if $E(k_2, E(k_1, m_i)) = c_i$ then (k_2, k_1)



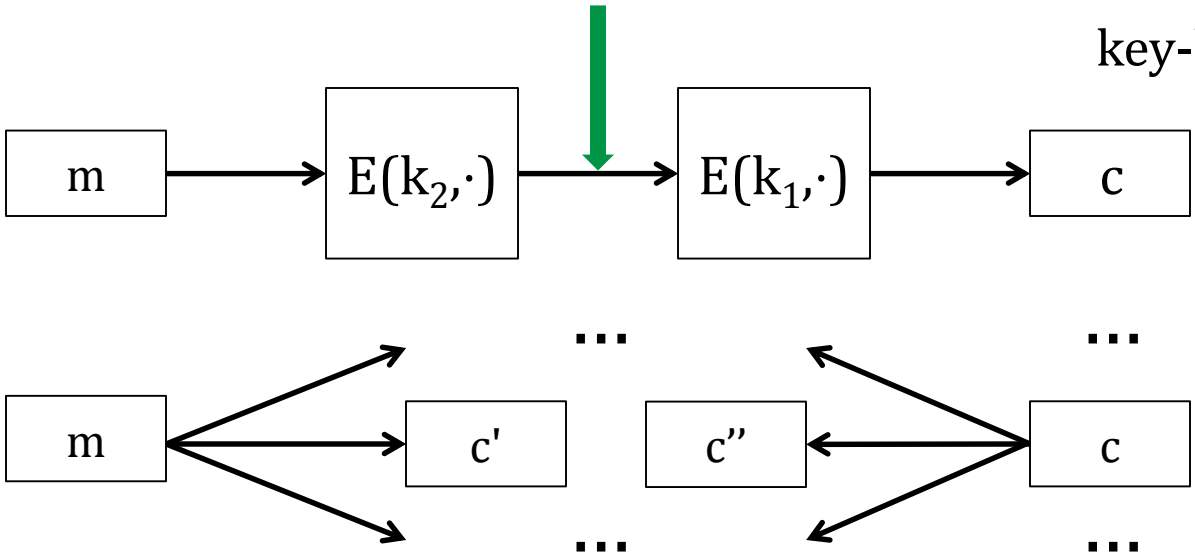
2^{112} checks

$c'' = c$?

Meet in the middle attack

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

key-len = 112 bits for 2DES

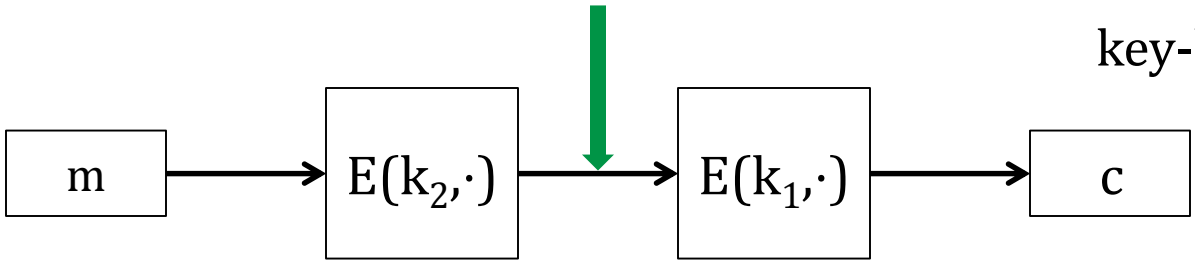


Idea: key found when c' = c'': $E(k_i, m) = D(k_j, c)$

Meet in the middle attack

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

key-len = 112 bits for 2DES



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

- step 1: build table.

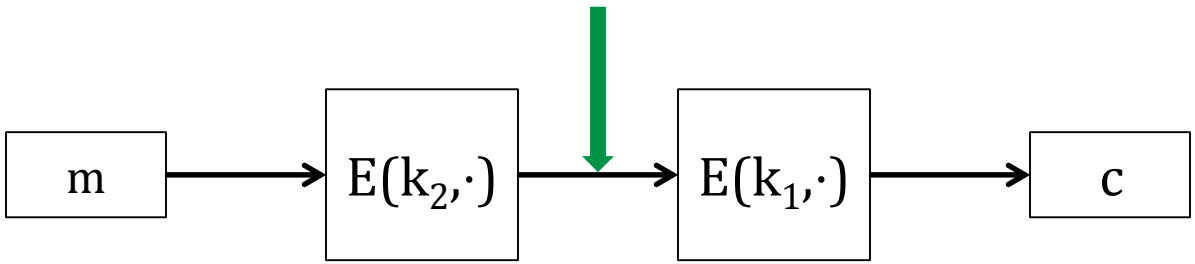
  sort on 2nd column

  maps c' to k_2

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

2^{56} entries

Meet in the middle attack



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

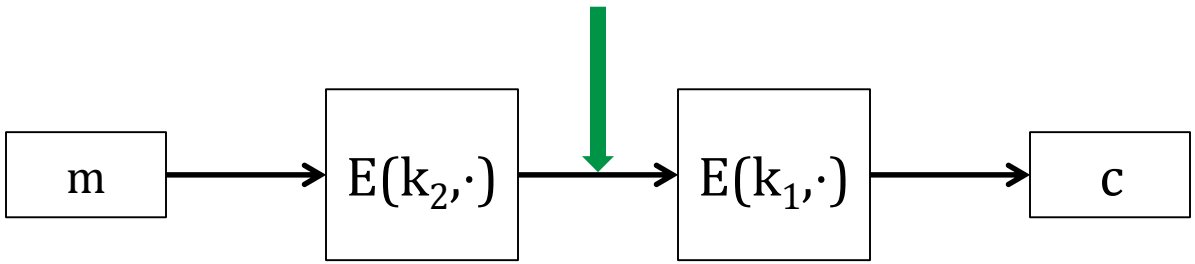
$k^0 = 00\ldots00$	$E(k^0, M)$
$k^1 = 00\ldots01$	$E(k^1, M)$
$k^2 = 00\ldots10$	$E(k^2, M)$
$\vdots$	$\vdots$
$k^N = 11\ldots11$	$E(k^N, M)$

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

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

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

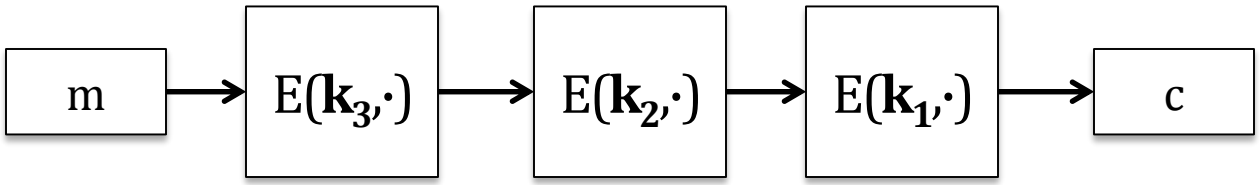
Meet in the middle attack



$$\text{Time} = \underbrace{2^{56}\log(2^{56})}_{\text{[Build \& Sort Table]}} + \underbrace{2^{56}\log(2^{56})}_{\text{[Search Entries]}} < 2^{63} \quad << \quad 2^{112}$$

Space $\approx 2^{56}$ [Table Size]

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



Method 2: DESX

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

Define EX as

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

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

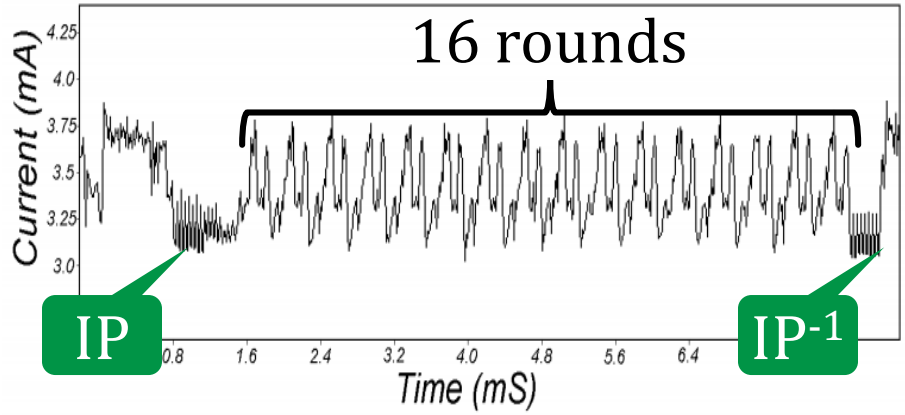
… but easy attack in time $2^{64+56} = 2^{120}$

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

Attacks on the implementation

1. Side channel attacks:
   - Measure **time** to do enc/dec, measure **power** for enc/dec

smartcard



[Kocher, Jaffe, Jun, 1998]

Card is doing DES

2. Fault attacks:
   - Computing errors in the last round expose the secret key k

⇒ never implement crypto primitives yourself …

Block ciphers

AES – Advanced encryption standard

The AES process

- 1997: DES broken by exhaustive search
- 1997: NIST publishes request for proposal
- 1998: 15 submissions
- 1999: NIST chooses 5 finalists
- 2000: NIST chooses Rijndael as AES
  (developed by Daemen and Rijmen at K.U. Leuven, Belgium)

Key sizes: 128, 192, 256 bits

Block size: 128 bits

AES core idea: Subs-Perm network

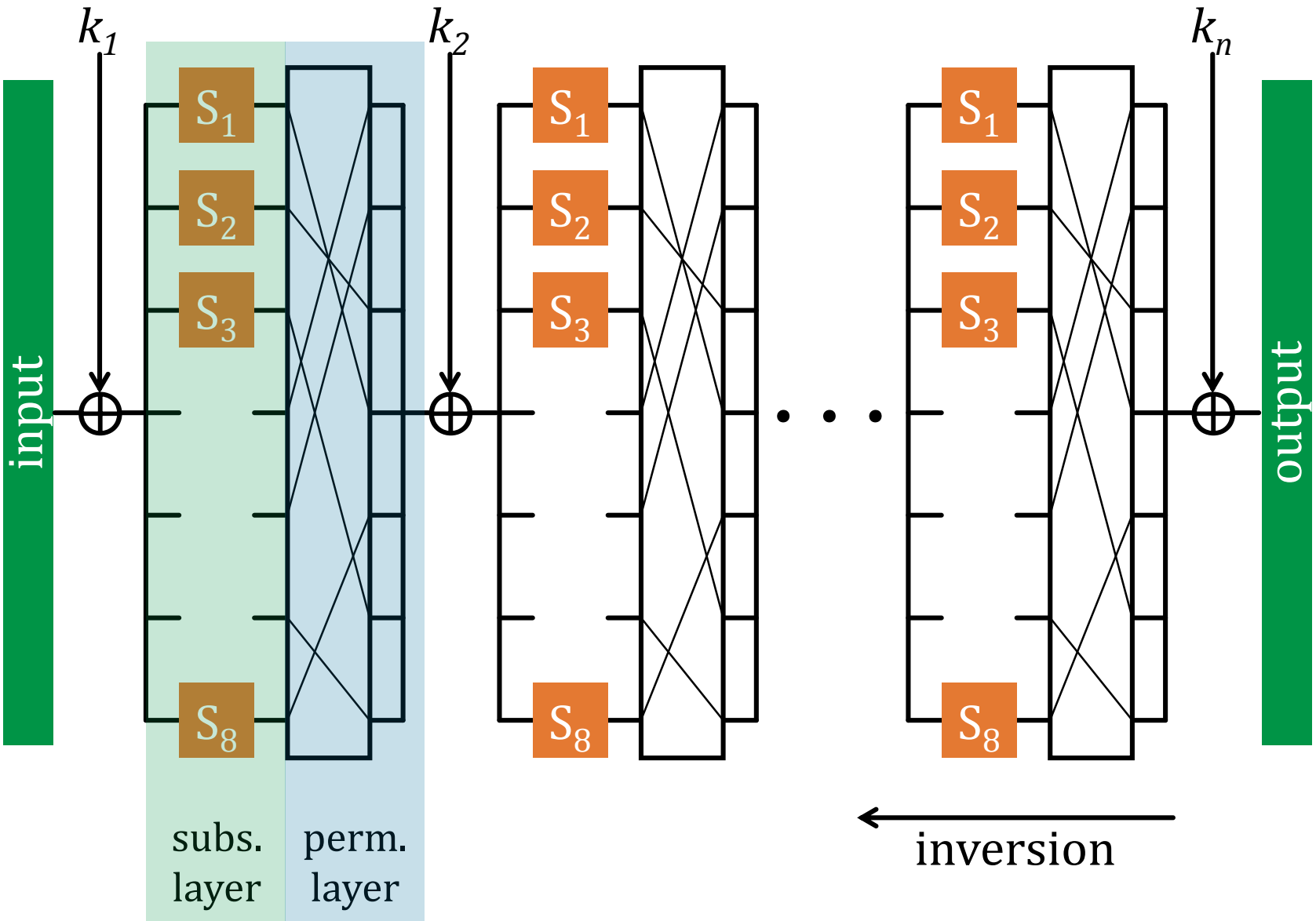
DES is based on Feistel networks

AES is based on the idea of

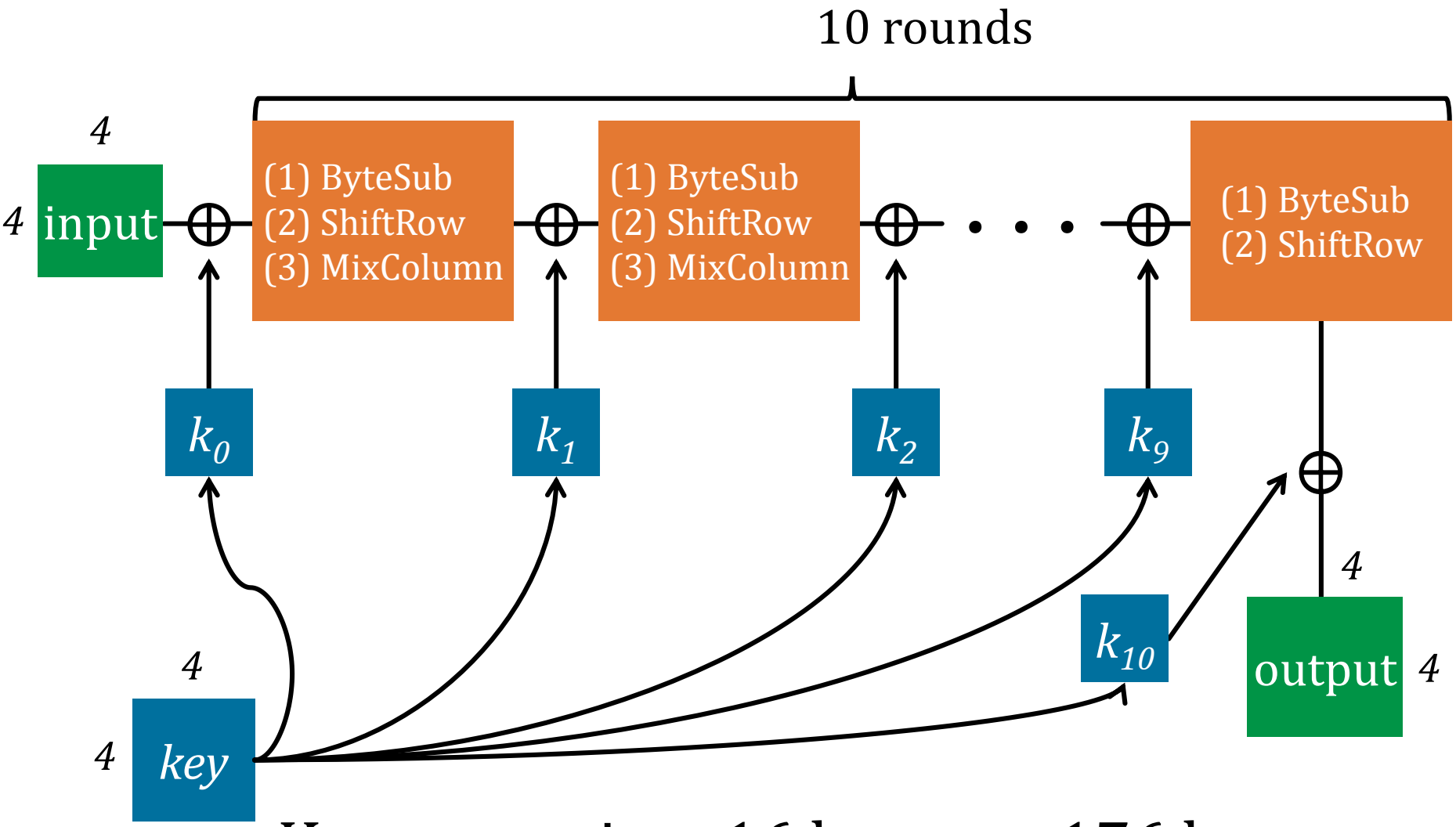
substitution-permutation networks

That is, alternating steps of substitution and permutation operations

AES: Subs-Perm network



AES128 schematic

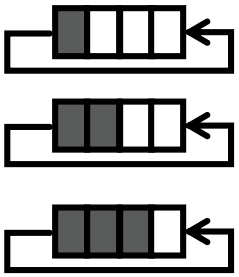


Key expansion: 16 bytes ⟶176 bytes

The round function

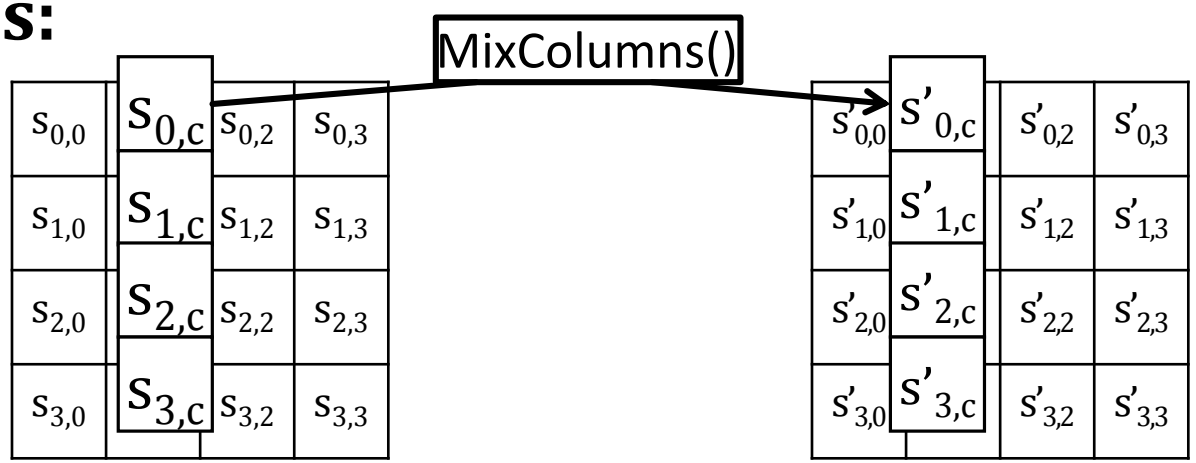
- **ByteSub:** a 1 byte S-box. 256 byte table (easily computable)
- **ShiftRows:**

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$



$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,1}$	$s_{1,2}$	$s_{1,3}$	$s_{1,0}$
$s_{2,2}$	$s_{2,3}$	$s_{2,0}$	$s_{2,1}$
$s_{3,3}$	$s_{3,0}$	$s_{3,1}$	$s_{3,2}$

- **MixColumns:**

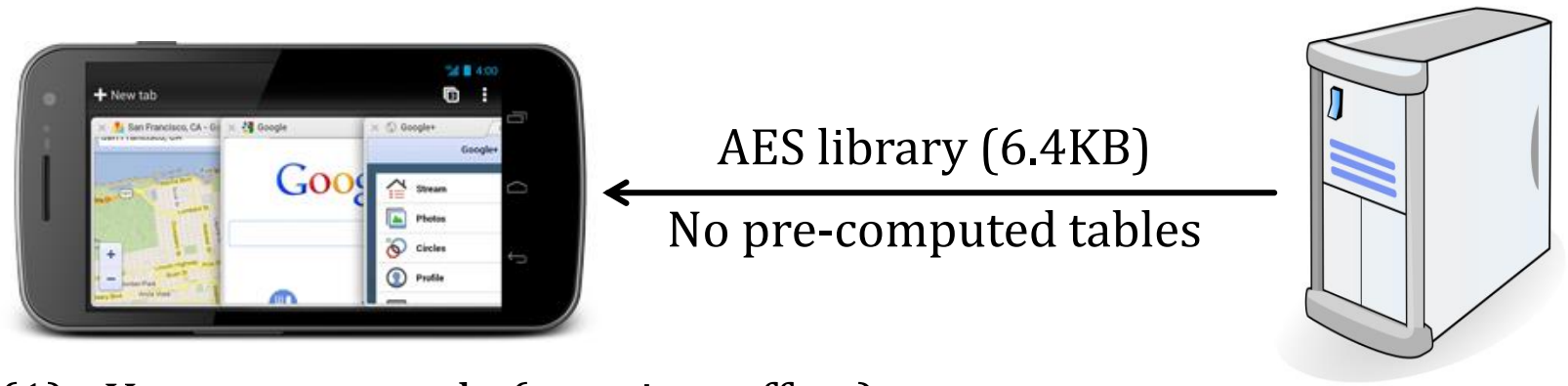


Code size/performance tradeoff

	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

Size + performance (Javascript AES)

AES in the browser



(1) Uncompress code (one time effort)

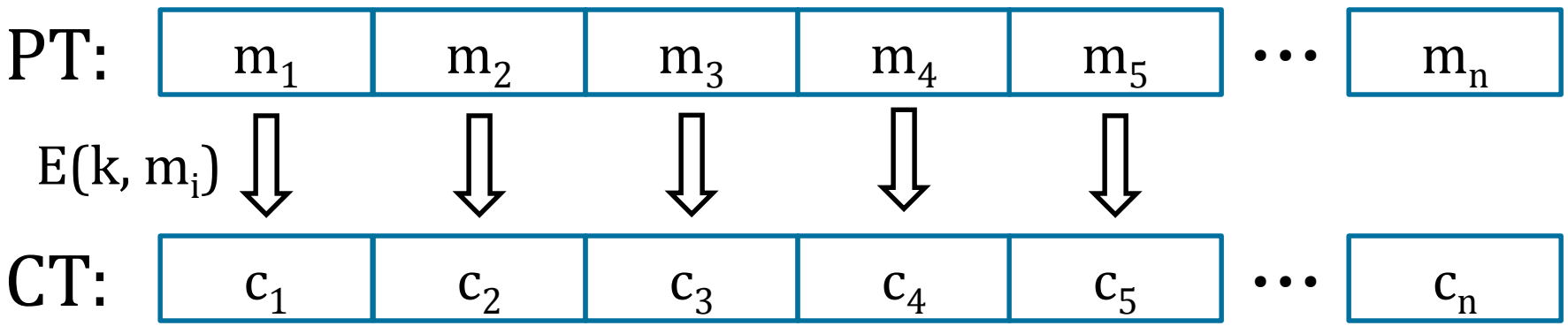
(2) Pre-compute tables (one time effort)

(3) Perform encryption using tables

Implementation: Stanford Javascript Crypto Library
https://crypto.stanford.edu/sjcl/

Modes of operation

Electronic Code Book (ECB) Mode



Problem:

$$m_1 = m_2 \longrightarrow c_1 = c_2$$

What can possibly go wrong?

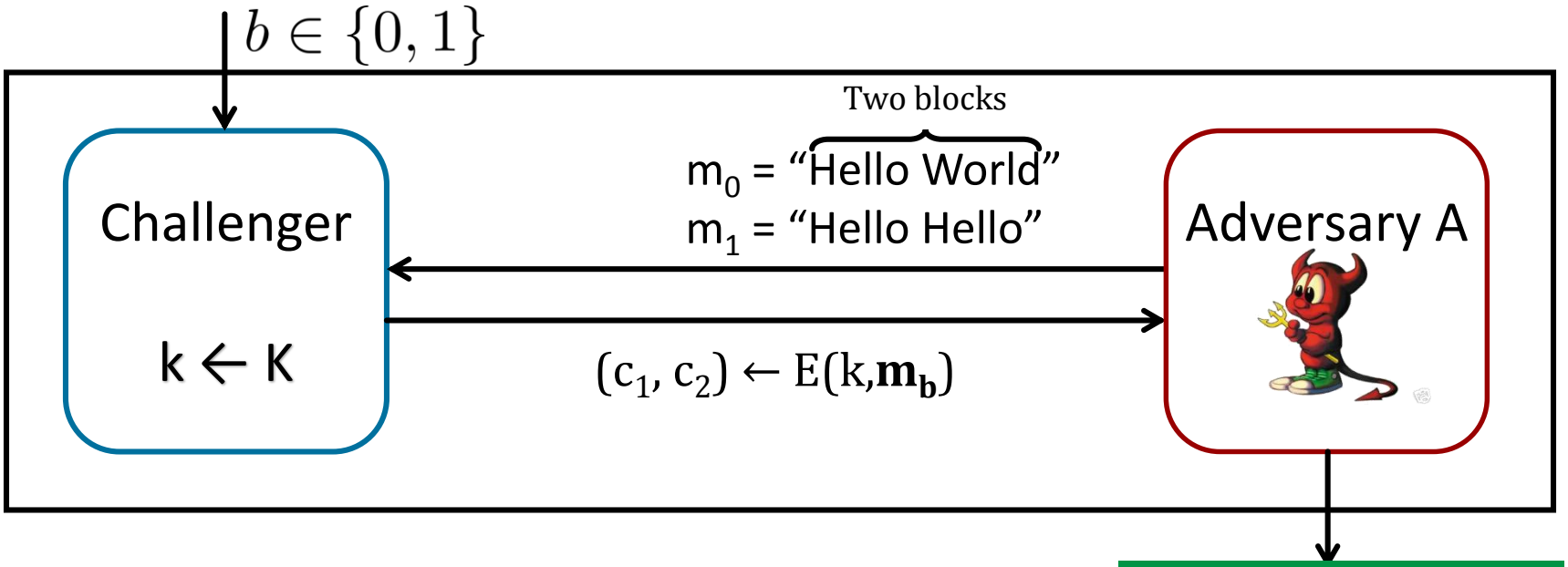
Plaintext

Ciphertext

Images from Wikipedia

Semantic security for ECB mode

ECB is not semantically secure for messages that contain more than one block

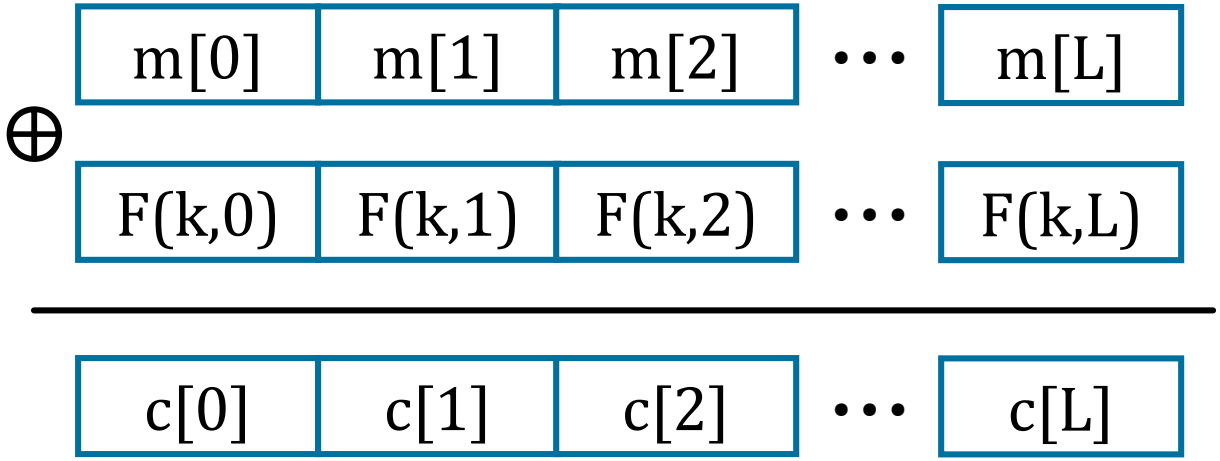


$\mathrm{Adv}_{SS}[A,ECB] =$

Deterministic counter mode

from a PRF F:

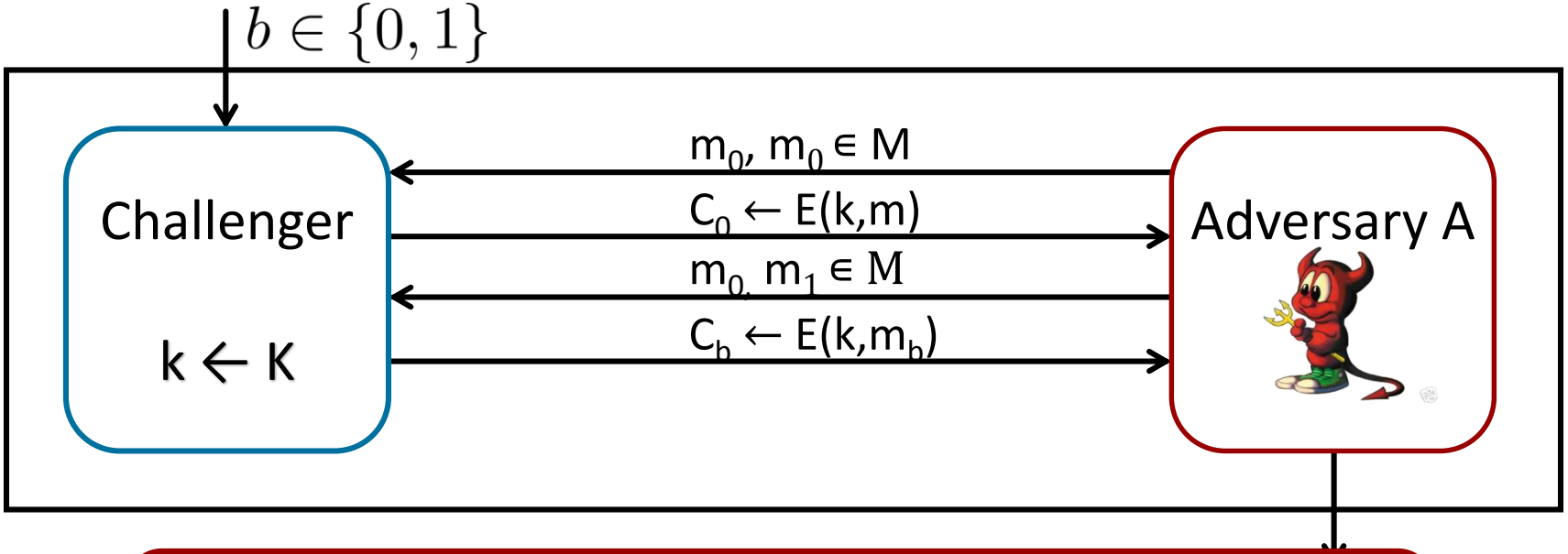
$E_{DETCTR}(k,m) =$



Stream cipher built from a PRF (e.g. AES, 3DES)
Better than ECB but only works as long as the key is only used once (one-time-key)

Semantic security under CPA

Modes that return the same ciphertext (e.g., ECB, CTR) for the same plaintext are not semantically secure under a chosen plaintext attack (CPA) (many-time-key)



Encryption modes must be randomized or use a nonce (or are vulnerable to CPA)

Semantic security under CPA

Modes that return the same ciphertext (e.g., ECB, CTR) for the same plaintext are not semantically secure under a chosen plaintext attack (CPA) (many-time-key)

Two solutions:

1. Randomized encryption

   Encrypting the same msg twice gives different ciphertexts (w.h.p.)

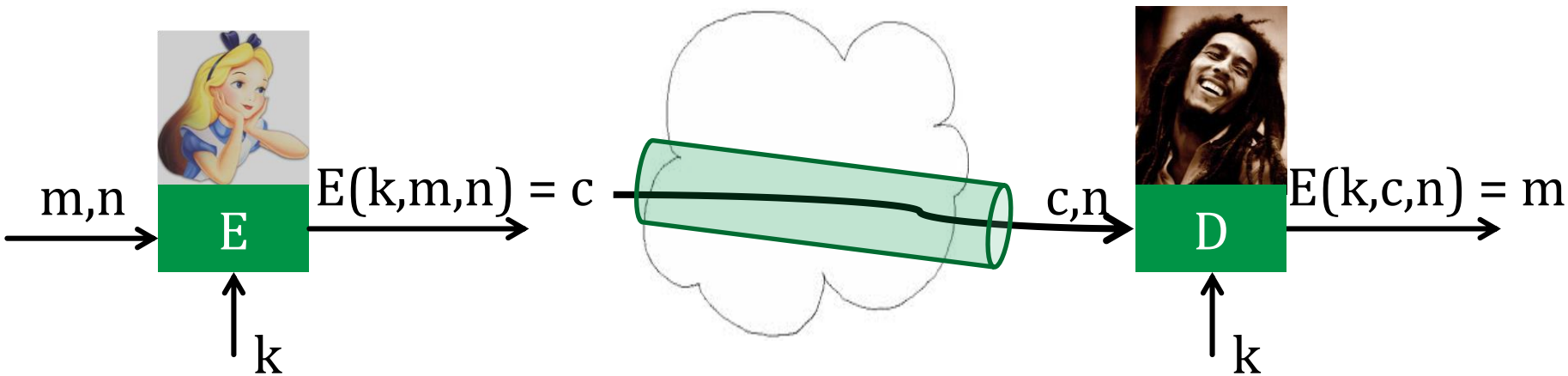
   Ciphertext must be longer than plaintext

2. Nonce-based encryption

Nonce-based encryption

Nonce n: a value that changes for each msg.

E(k,m,n) / D(k,c,n)



(k,n) pair never used more than once

Nonce-based encryption

Method 1: Nonce is a counter

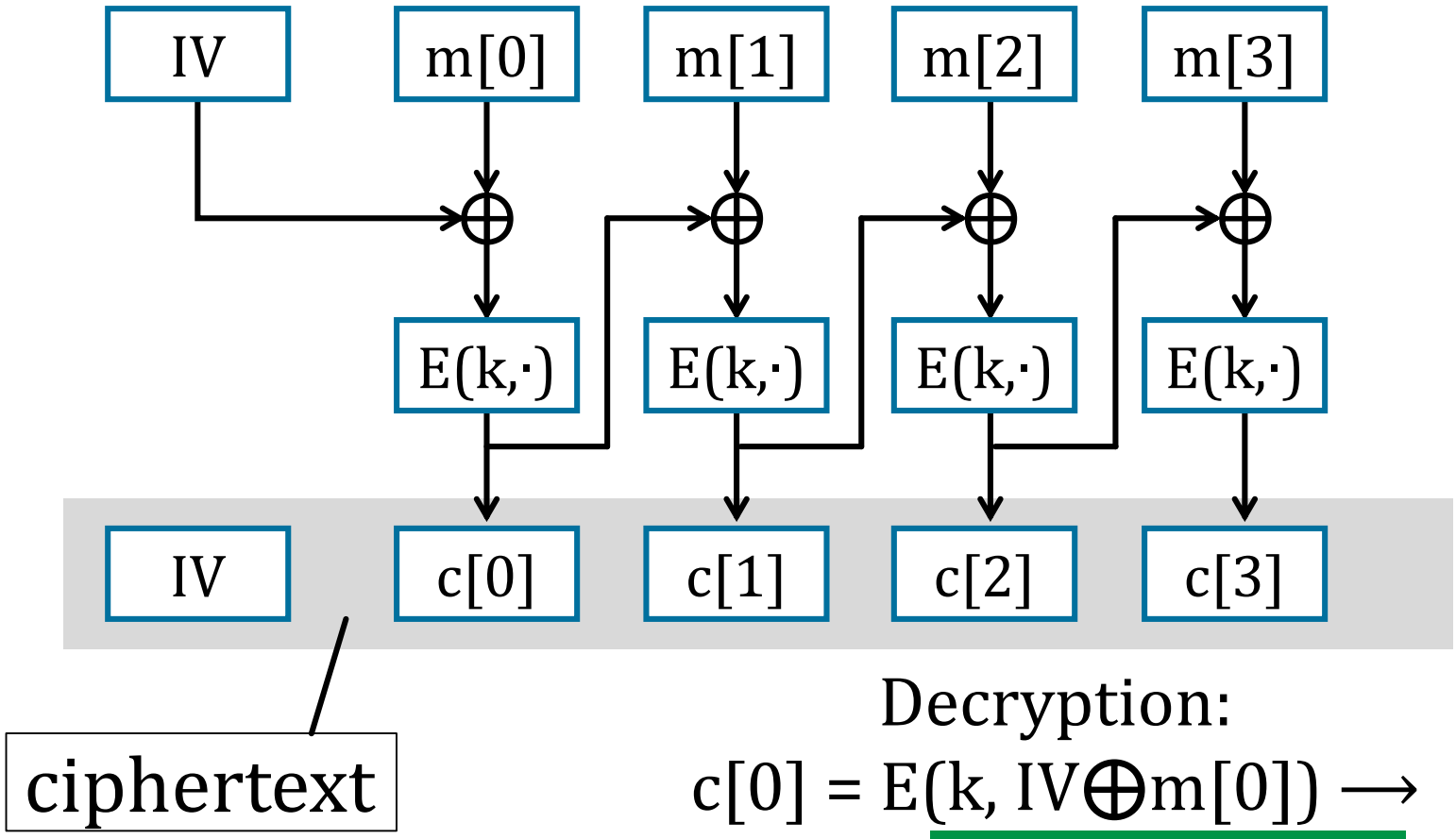
- Used when encryptor keeps state from msg to msg
- If decryptor has same state, nonce need not be transmitted (i.e., len(PT) = len(CT))

Method 2: Sender chooses a random nonce

- No state required but nonce has to be transmitted with CT

Cipher block chaining mode (CBC)

Let(E,D) be a PRP. $E_{CBC}(k,m)$: chose **<u>random</u>** IV ∈ X and do:



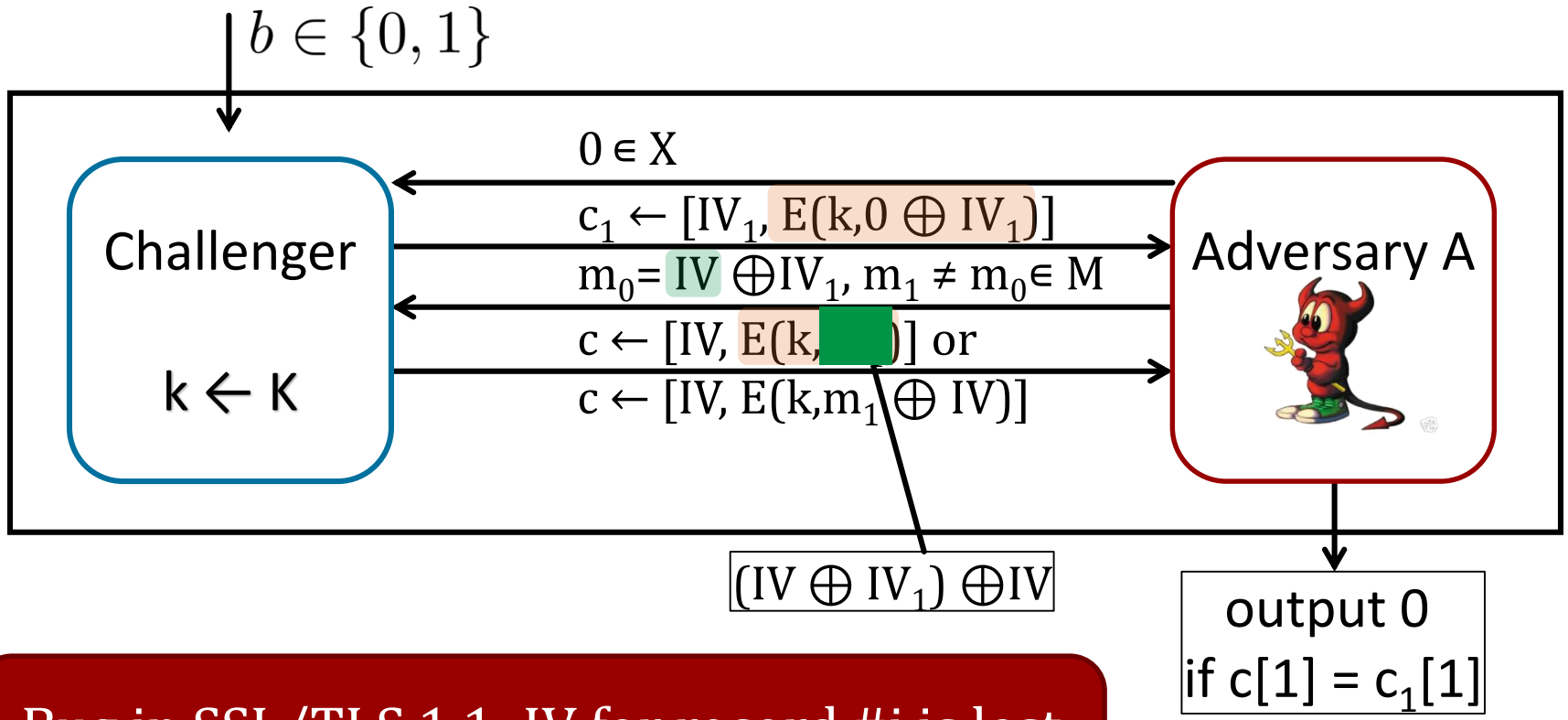
Decryption:

$c[0] = E(k, IV \oplus m[0]) \longrightarrow$

$m[0] =$

Attack on CBC with Predictable IV

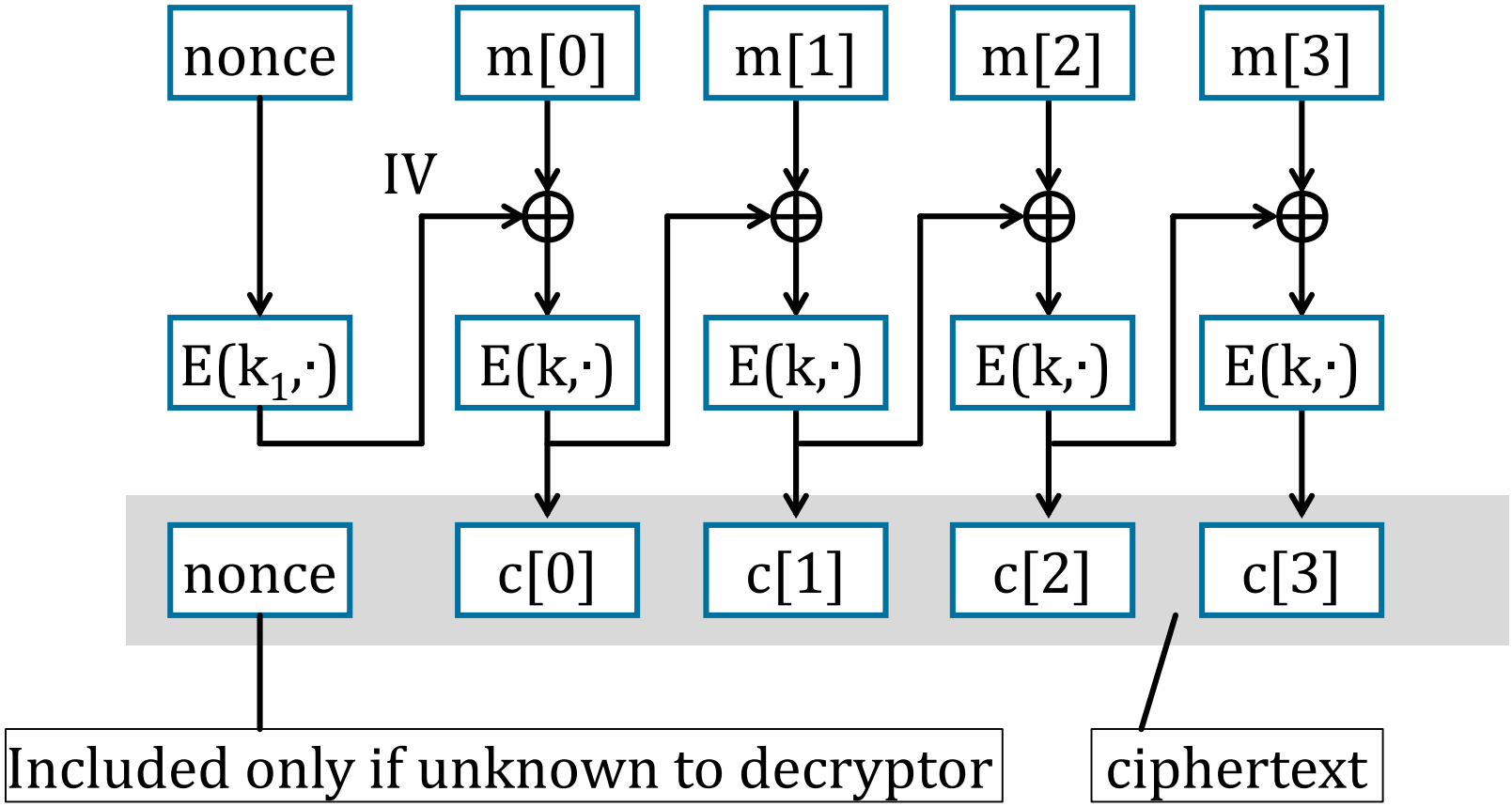
Suppose given $c \leftarrow E_{CBC}(k,m)$ Adv. can predict IV for next msg.



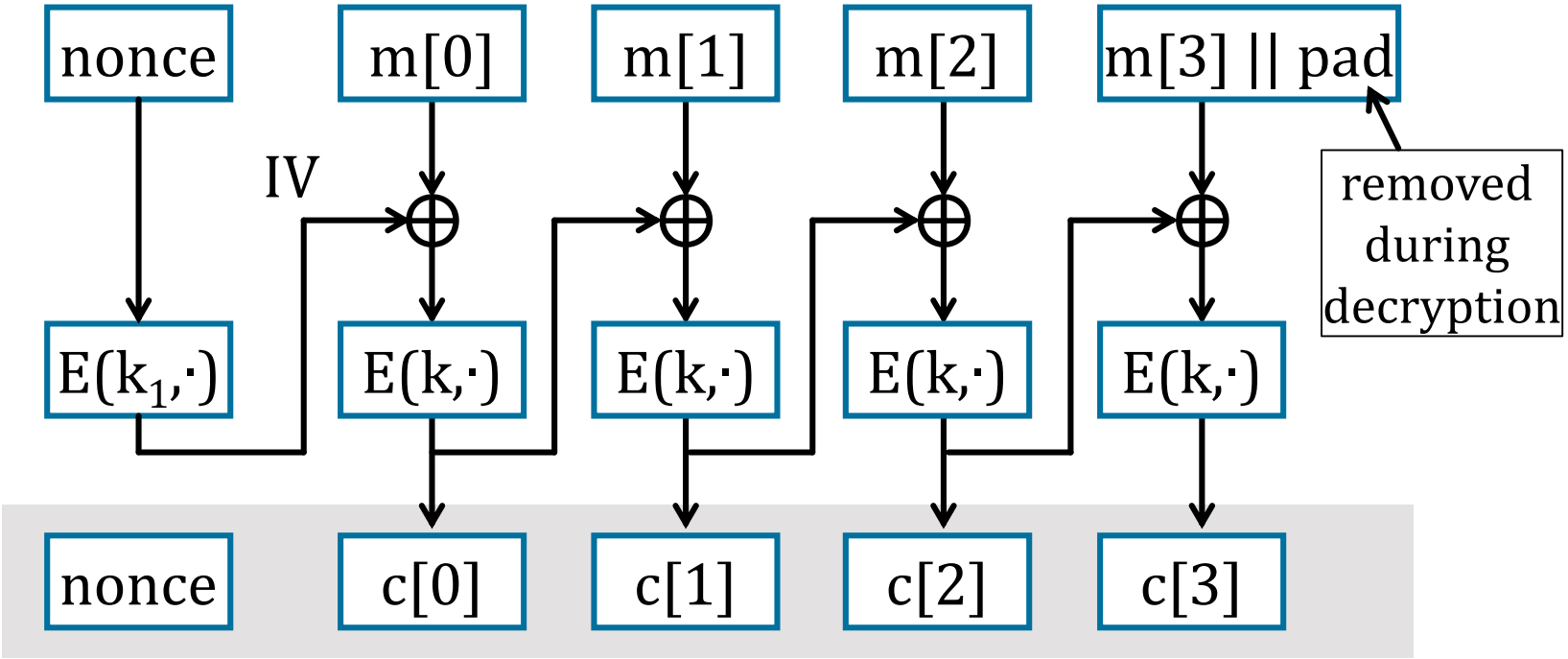
Bug in SSL/TLS 1.1: IV for record #i is last CT block of record #(i-1)

Nonce-based CBC

CBC with unique nonce: key = (k, k_1) two independent keys
unique nonce means: (key,n) pair is used for only one msg.



CBC: padding



TLS: for n > 0 n byte pad is: | n | n | … | n |

If no pad needed, add a dummy block: | 16 | 16 | … | 16 |

Padding oracle side channel attacks

Cipher block chaining mode (CBC)

Example applications:

1. File system encryption:

   use the same AES key to encrypt all files (e.g., loopaes)

2. IPsec:

   use the same AES key to encrypt multiple packets

Problem:

If attacker can <u>predict</u> IV, CBC is not CPA-secure

Summary

Block ciphers

- Map fixed length input blocks to same length output blocks
- Canonical block ciphers: 3DES, AES
- PRPs are effectively block ciphers
- PRPs can be created from arbitrary functions through Feistel networks
  - 3DES based on Feistel networks
  - AES based on substitution-permutation networks


Questions?

END

Linear and differential attacks [BS'89,M'93]

Given *many* inp/out pairs, can recover key in time less than 2^{56} .

<u>Linear cryptanalysis</u> (overview) :

let c = DES(k, m)

Suppose for random k,m :

$$\Pr\left[\; m[i_1]\oplus\cdots\oplus m[i_r] \;\oplus\; c[j_j]\oplus\cdots\oplus c[j_v] \;=\; k[l_1]\oplus\cdots\oplus k[l_u] \;\right] = \tfrac{1}{2} + \varepsilon$$

For some ε. For DES, this exists with

$\varepsilon = 1/2^{21} \approx 0.0000000477$

Linear attacks

$$\Pr\left[\; m[i_1]\oplus\cdots\oplus m[i_r] \;\oplus\; c[j_j]\oplus\cdots\oplus c[j_v] \;=\; k[l_1]\oplus\cdots\oplus k[l_u] \;\right] = ½ + \varepsilon$$

Thm: given $1/\varepsilon^2$ random $(m, c=DES(k, m))$ pairs then

$$k[l_1,\ldots,l_u] \;=\; \mathrm{MAJ}\left[\; m[i_1,\ldots,i_r] \;\oplus\; c[j_j,\ldots,j_v] \;\right]$$

with prob. ≥ 97.7%

⇒ with $1/\varepsilon^2$ inp/out pairs can find $k[l_1,\ldots,l_u]$ in time $\approx 1/\varepsilon^2$.

Linear attacks

For DES, $\varepsilon = 1/2^{21} \Rightarrow$

with 2^{42} inp/out pairs can find $k[l_1,\ldots,l_u]$ in time 2^{42}

Roughly speaking: can find 14 key "bits" this way in time 2^{42}

Brute force remaining $56-14=42$ bits in time 2^{42}

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

Lesson

A tiny bit of linearity in S_5 lead to a 2^{42} time attack.

$\Rightarrow$ don't design ciphers yourself !!

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 $= O(|X|^{1/2})$

Can quantum computers be built: unknown

Quantum exhaustive search

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

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

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)