CS716/IS703
Data Security/Information System Security

Week 3:
Block Cipher

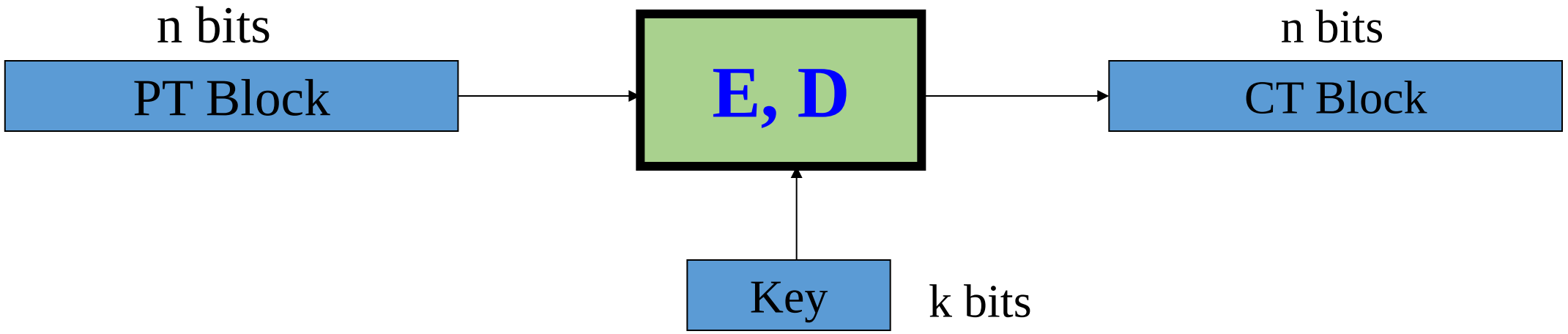
Fall 2025

Assoc. Prof. Hisham Dahshan
hishamdahshan@aast.edu

Our Goal Today

- Block Cipher Algorithms:
  - Data Encryption Standard (DES)
  - Advanced Encryption Standard (AES)

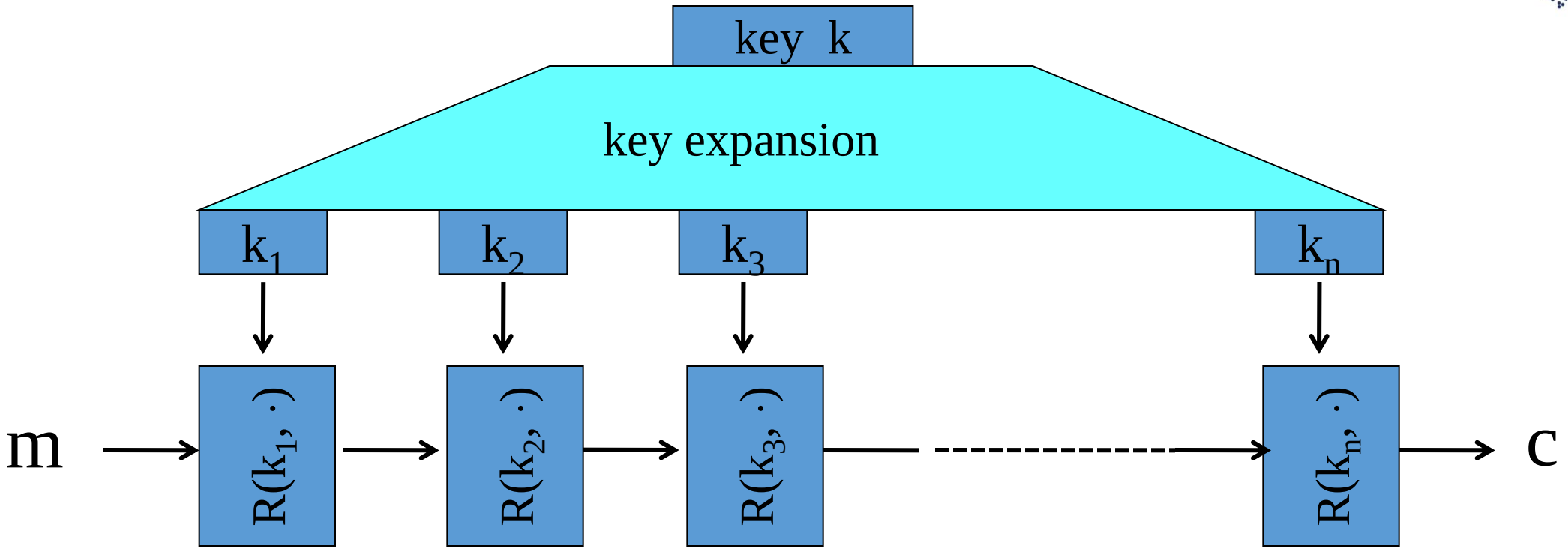
Block ciphers: crypto work horse



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 (# rounds =48), for AES-128 ((# rounds =10)

Block ciphers

The data encryption standard (DES)

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 and commerce

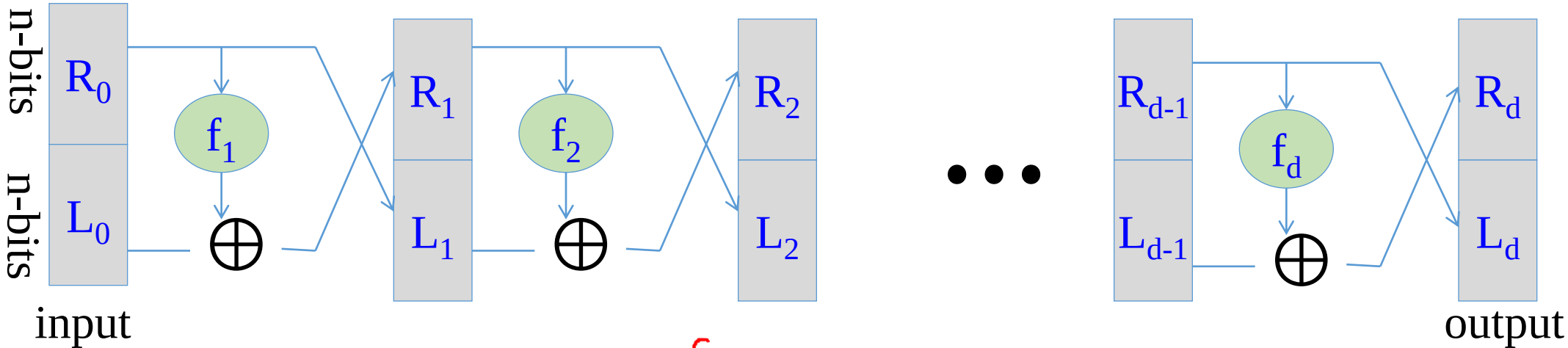
DES: core idea – Feistel Network

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

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

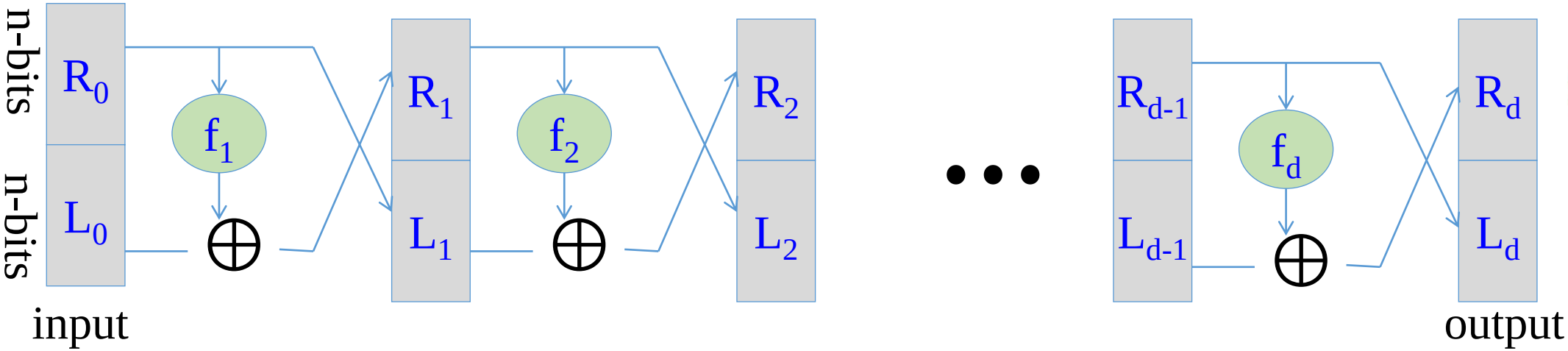
R: right half block

L: lift half block



In symbols:

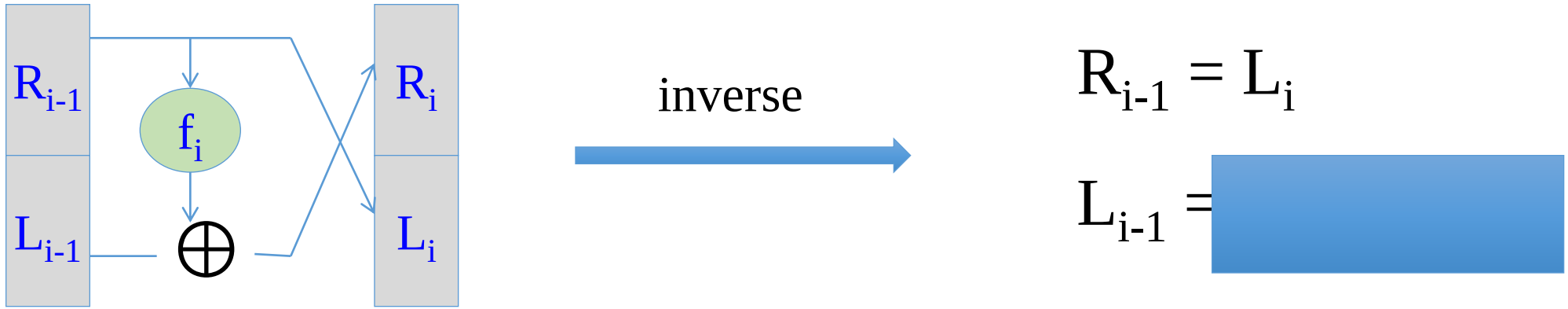
$$\begin{cases} R_i = f_i(R_{i-1}) \oplus L_{i-1} \\ L_i = R_{i-1} \end{cases}$$

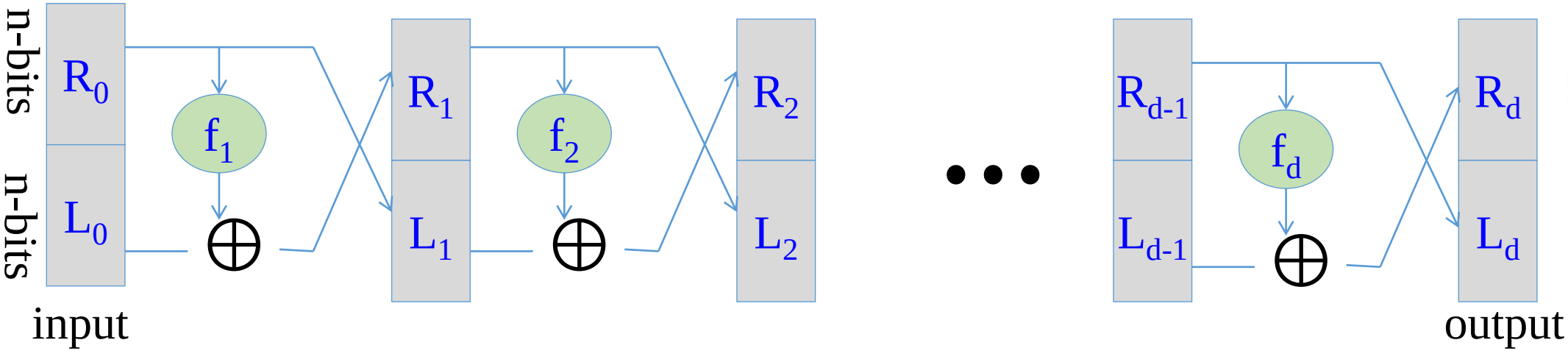


Claim: for all $f_1, \ldots, f_d$: $\{0,1\}^n \longrightarrow \{0,1\}^n$

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

Proof: construct inverse

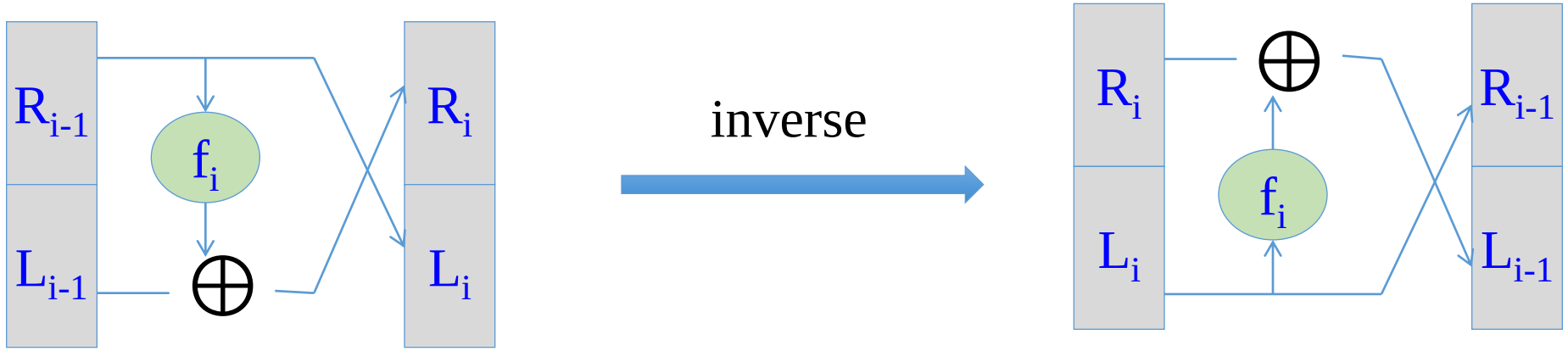




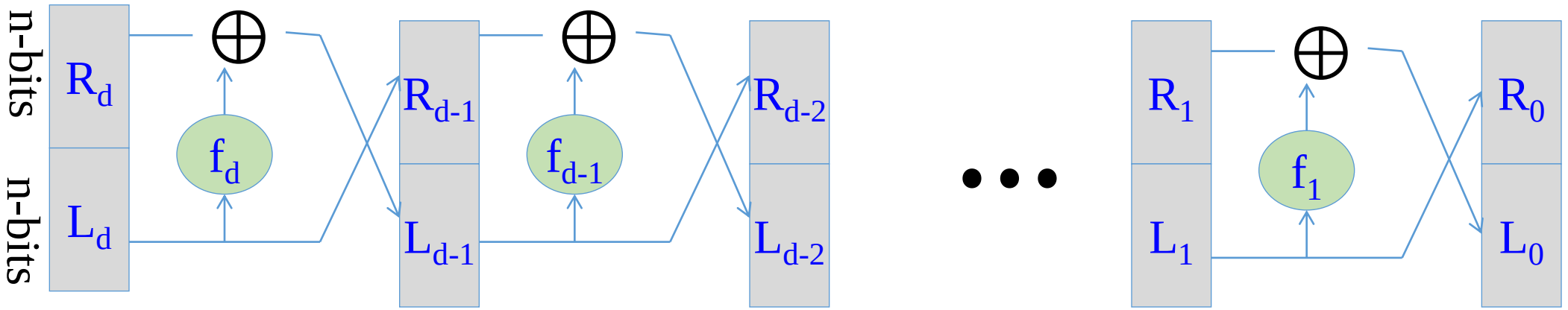
Claim: for all $f_1, \ldots, f_d$: $\{0,1\}^n \longrightarrow \{0,1\}^n$

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

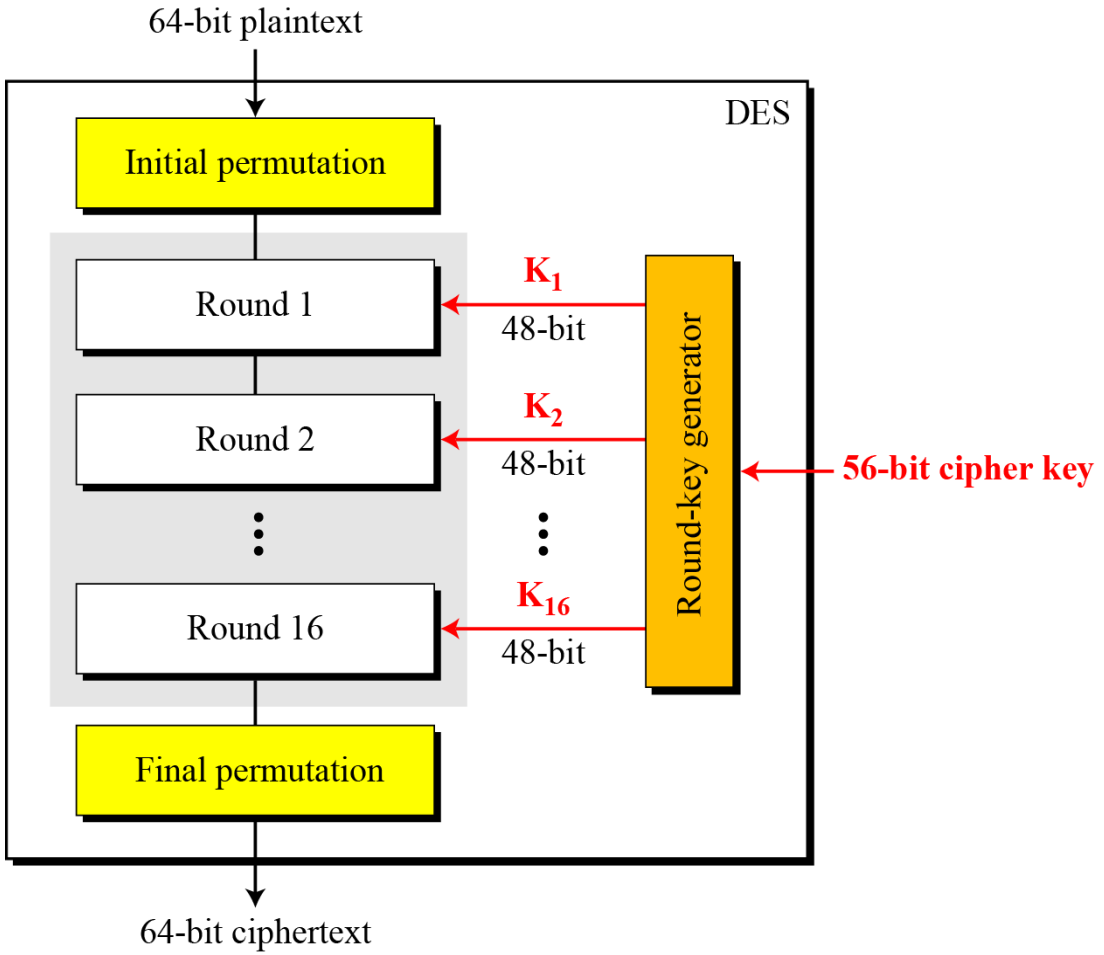
- General method for building invertible functions (block ciphers) from arbitrary functions.

- Used in many block ciphers … but not AES

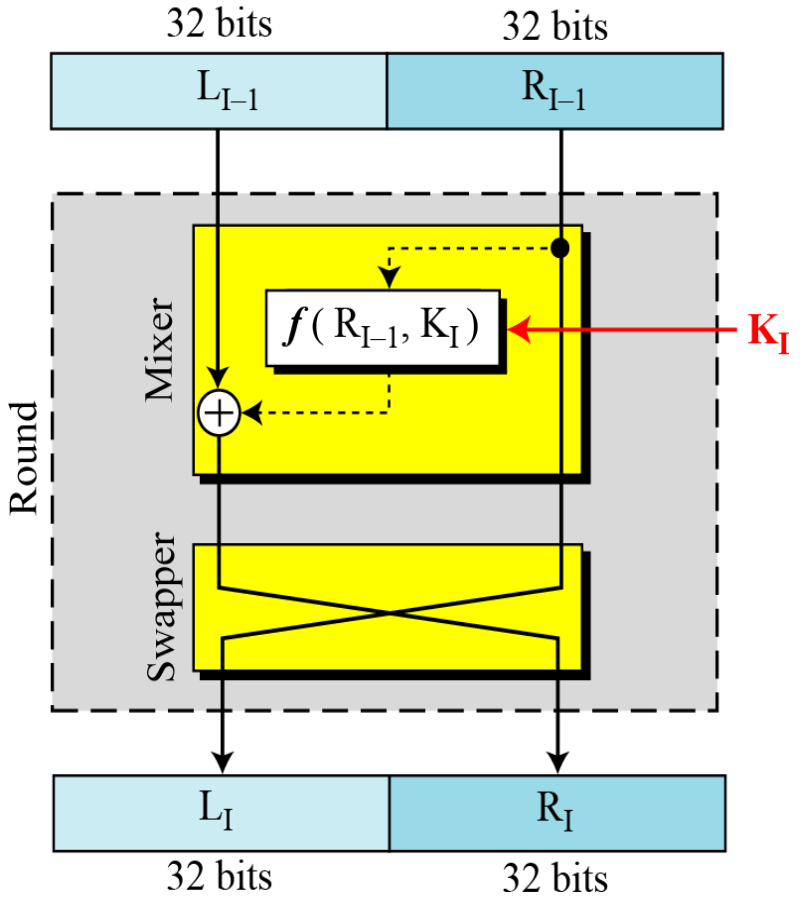
DES Encryption Overview

- 16 rounds using 64-bit block and 48-bit subkey

General structure of DES

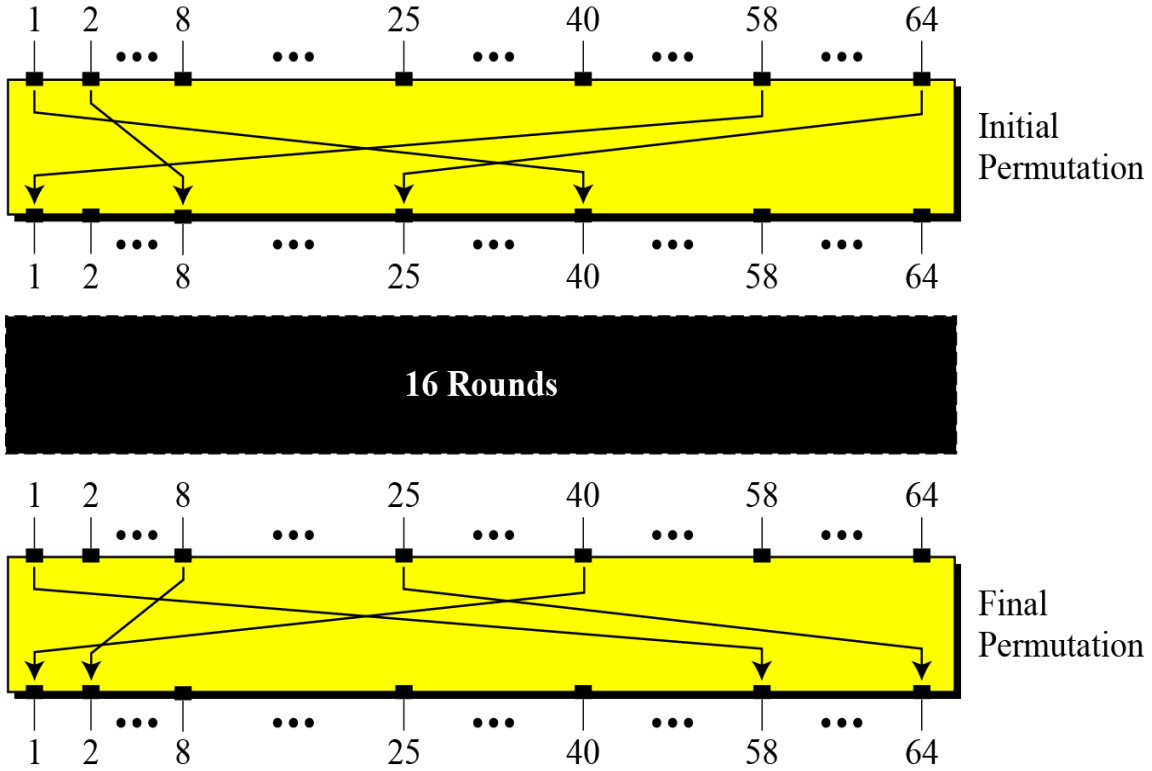


A round in DES (encryption site)



Initial and Final Permutations

Initial and final permutation steps in DES

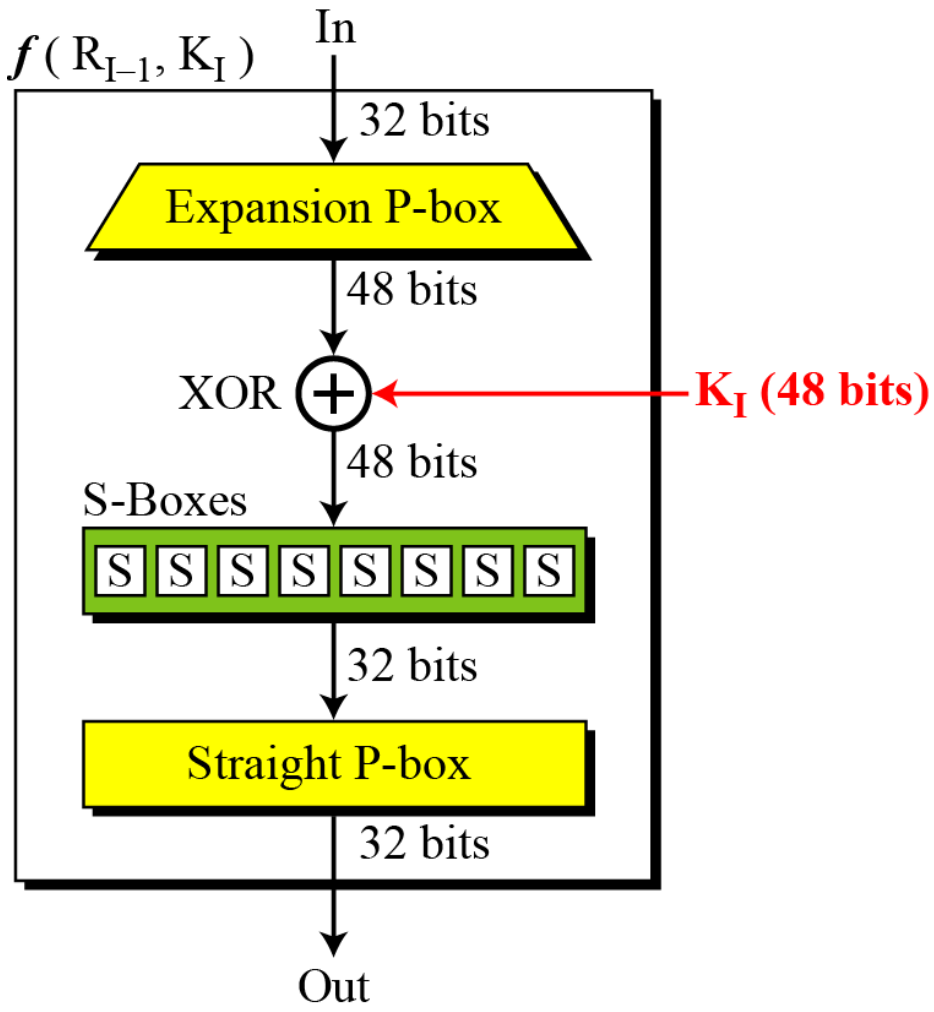


The initial and final permutations are straight P-boxes that are inverses of each other.

DES Function

The heart of DES is the DES function. The DES function applies a 48-bit key to the rightmost 32 bits to produce a 32-bit output.

DES function



The S-boxes

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

For example, an input "**0**11011**1**" has outer bits "**01**" and inner bits "1101"; the corresponding output would be "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

Example: a bad S-box choice

Suppose:

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

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.

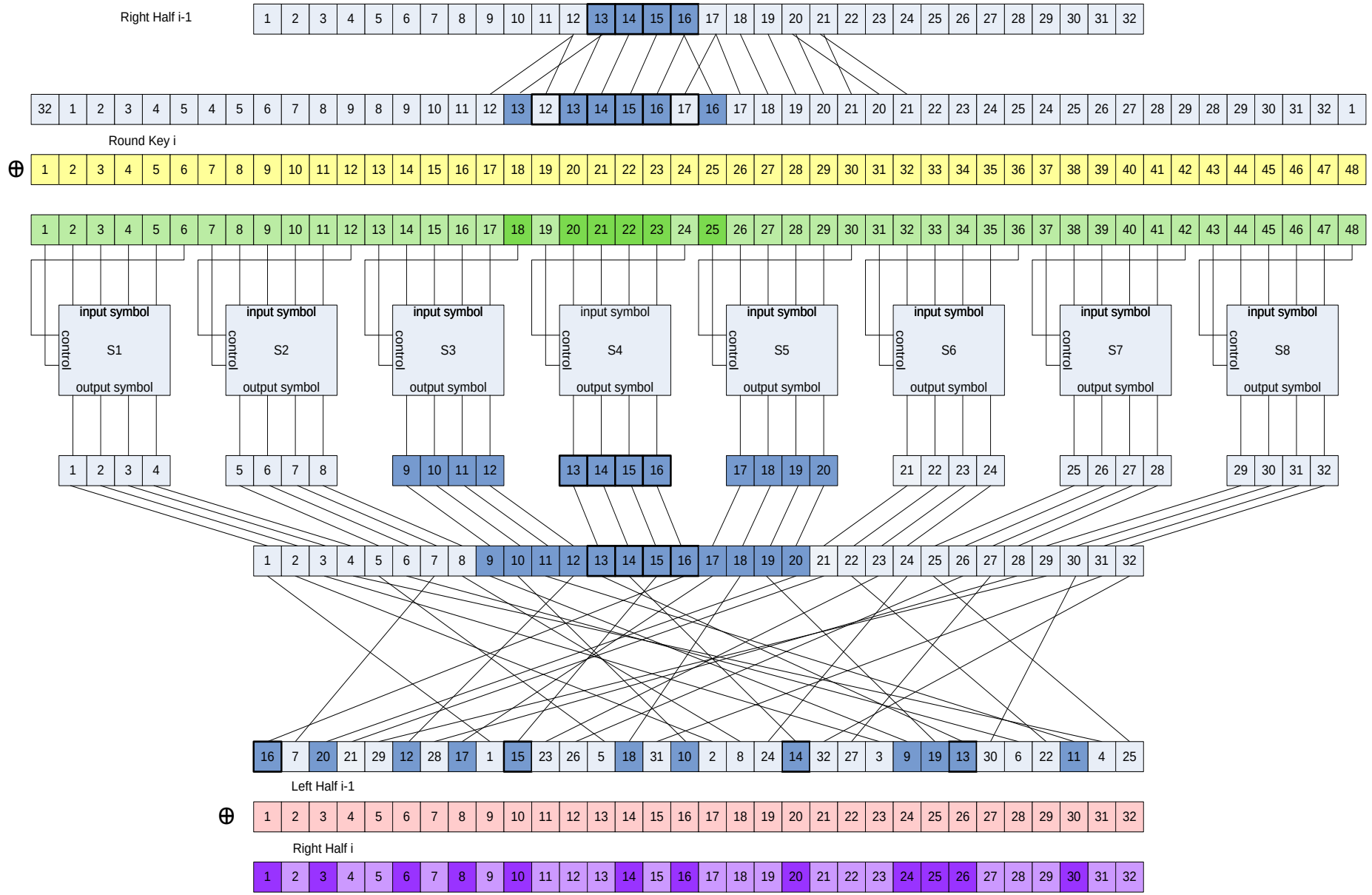
Choosing the S-boxes and P-box

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
  - ⋮

DES Round in Full



Block ciphers

Exhaustive Search Attacks

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

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

$k_1 = k_2 = k_3 \Rightarrow$ single DES

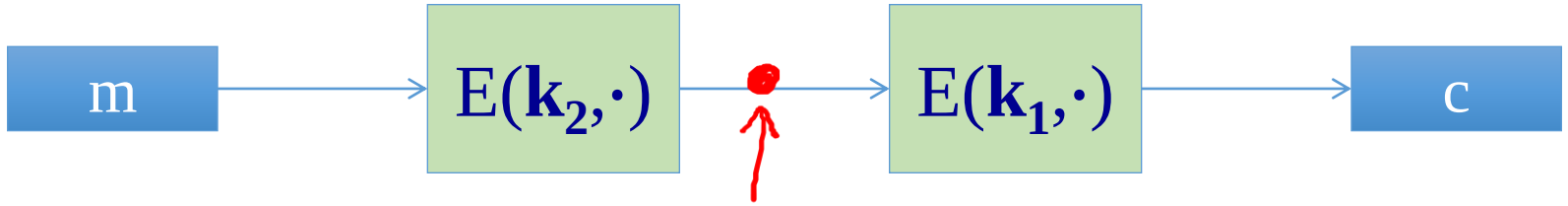
For 3DES: key-size = 3×56 = 168 bits. 3×slower than DES.

(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:
$E(K_2, M) = D(K_1, C)$

Assume that you are a cryptanalyst who has access to the plaintext (m) and encrypted text (c). Your aim is to recover the secret key.

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

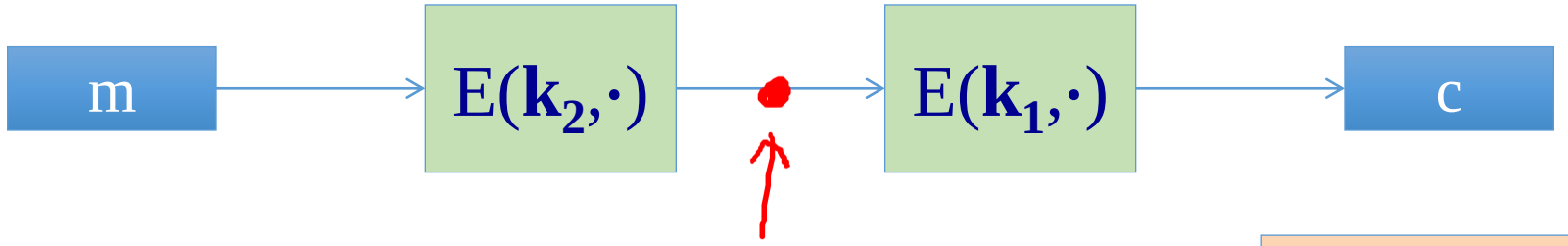
- step 1: build table.

  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



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

- step 1: build table.

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

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

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

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

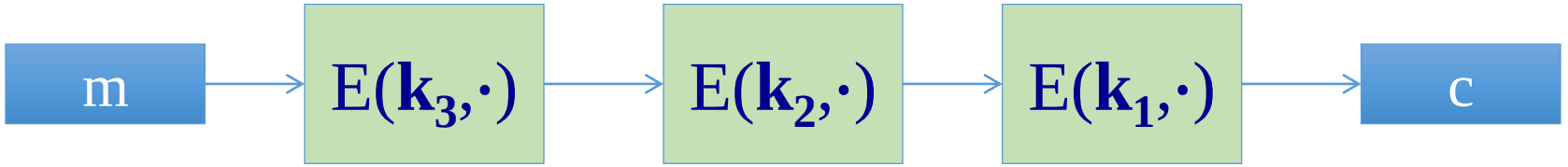
Meet in the middle attack

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

table

search in table

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



Block ciphers

More attacks on block ciphers

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

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

Block ciphers

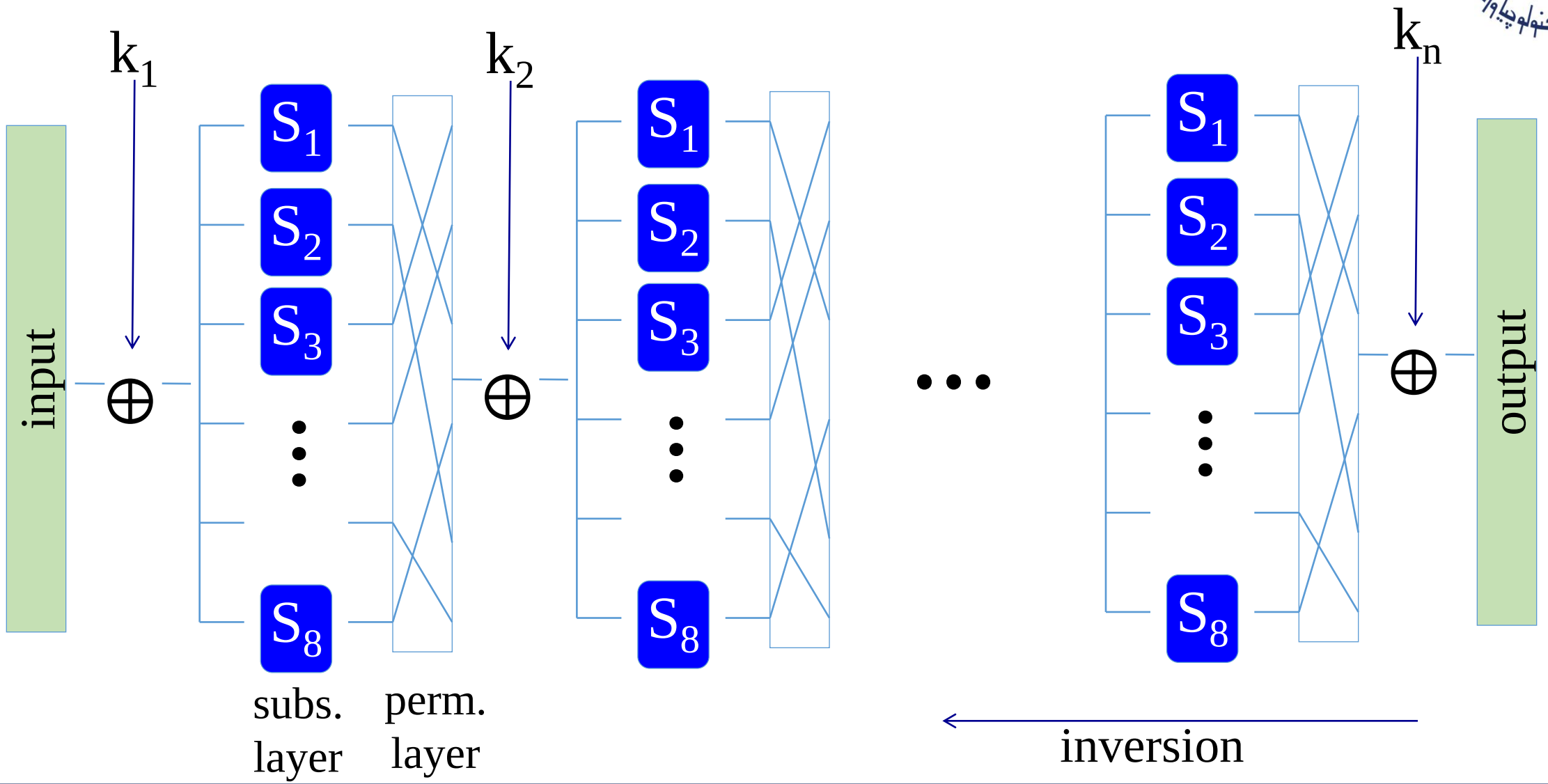
The AES block cipher

The AES process

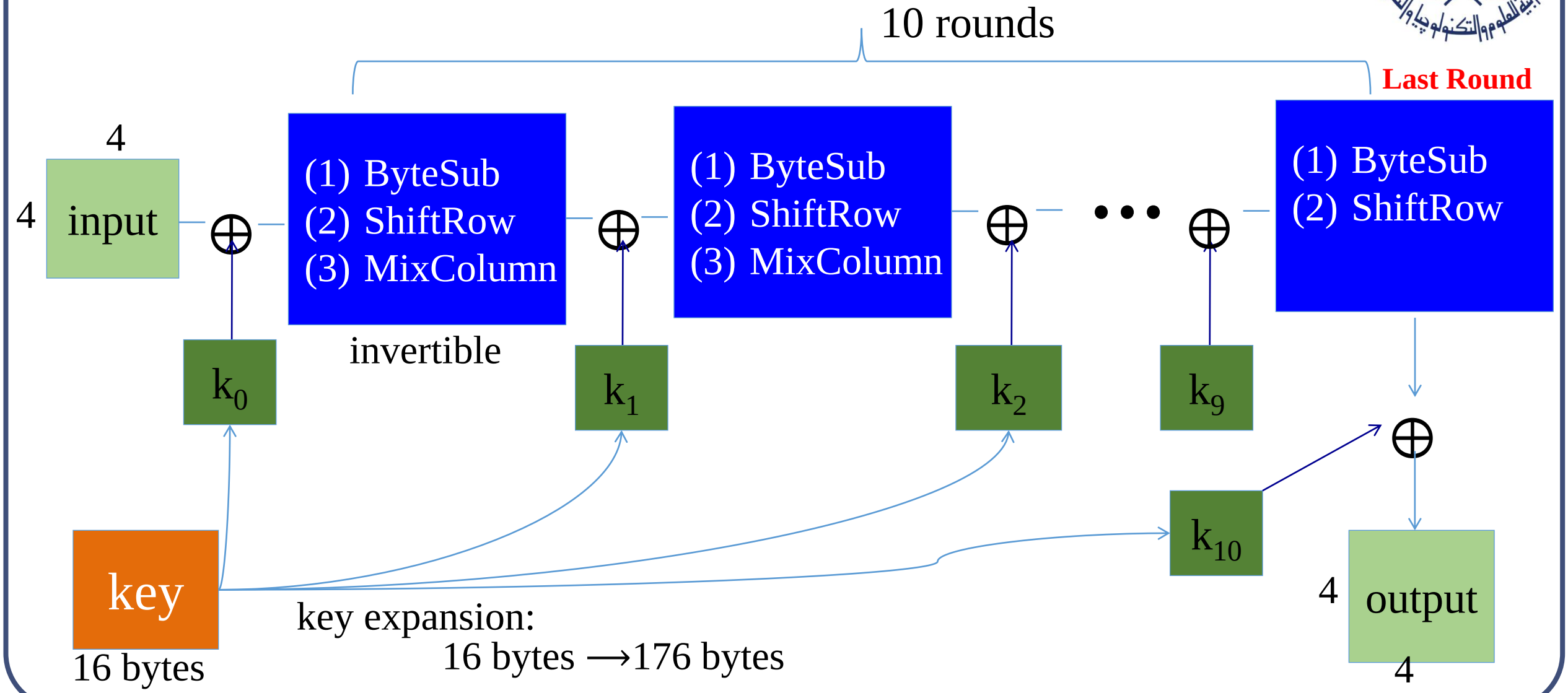
- 1997: NIST publishes request for proposal
- 1998: 15 submissions. Five claimed attacks.
- 1999: NIST chooses 5 finalists
- 2000: NIST chooses Rijndael as AES (designed in Belgium)

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

AES is a Subs-Perm network (not Feistel)

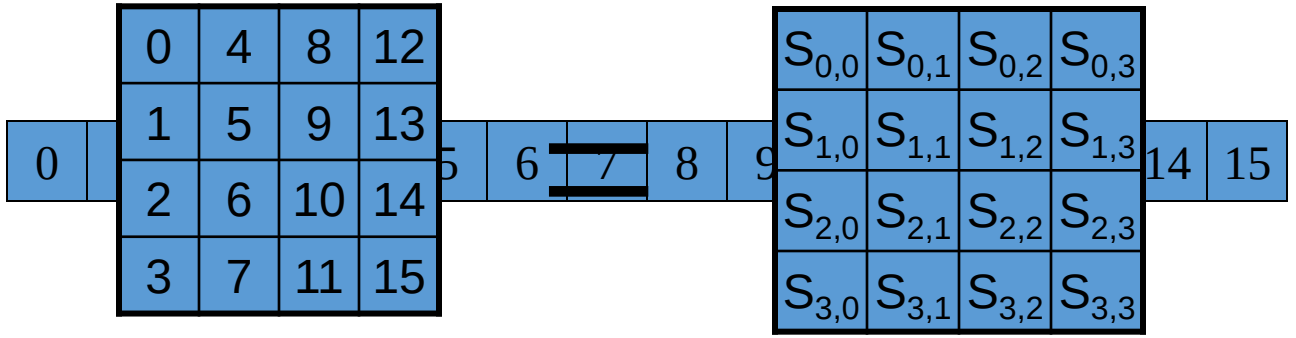


AES-128 schematic



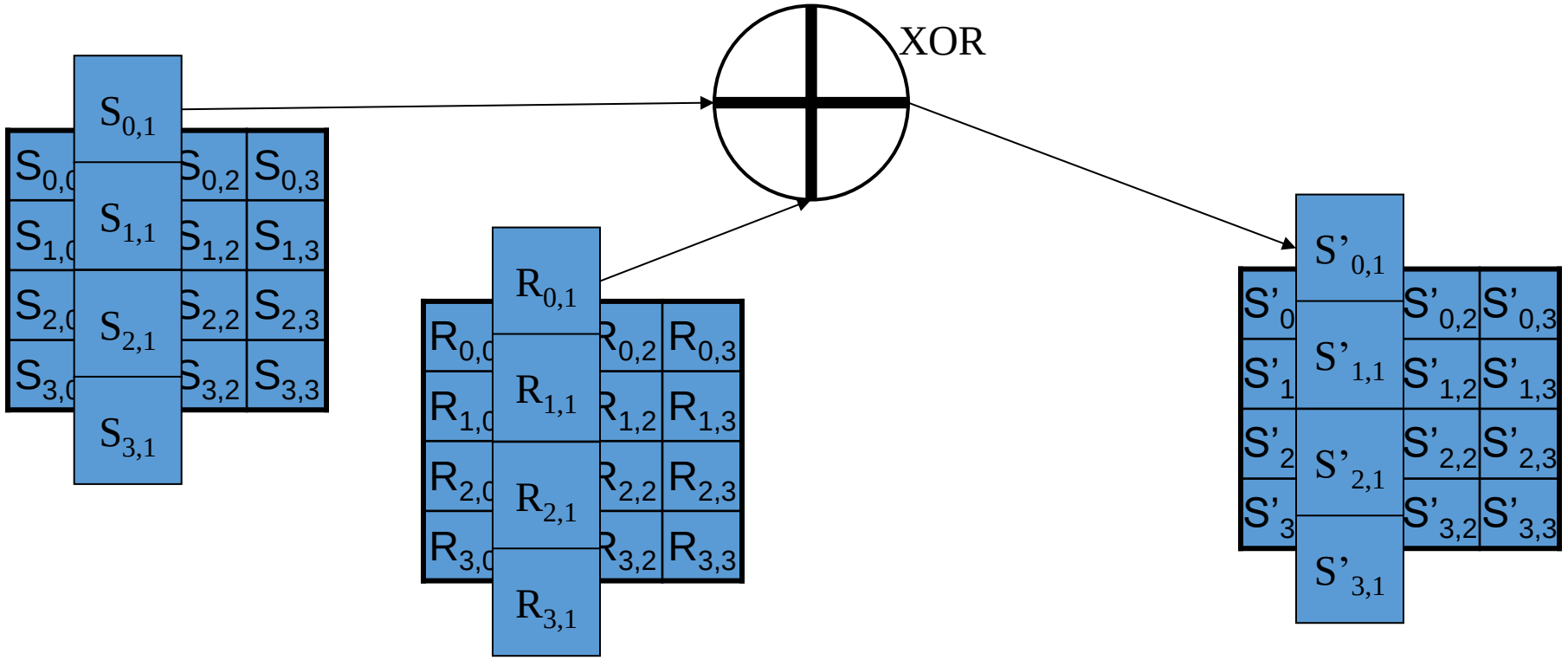
Convert to State Array

Input block:



AddRoundKey

- XOR each byte of the round key with its corresponding byte in the state array



SubBytes

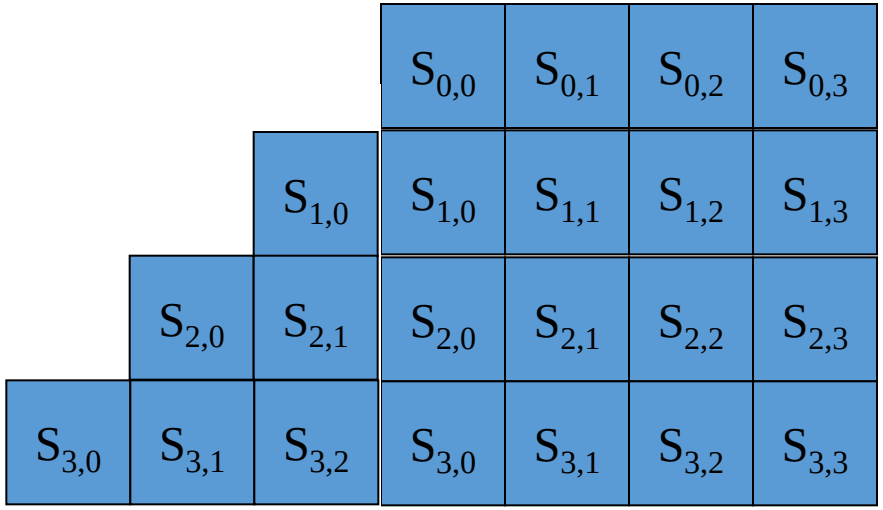
- Replace each byte in the state array with its corresponding value from the S-Box

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Byte (55) will be substituted by Byte (fc)

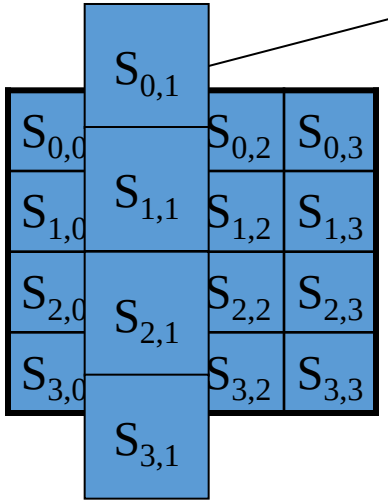
ShiftRows

- Last three rows are cyclically shifted



MixColumns

- Apply MixColumn transformation to each column



$$S'_{0,c} = (\{02\} \bullet S_{0,c}) \oplus (\{03\} \bullet S_{1,c}) \oplus S_{2,c} \oplus S_{3,c}$$

$$S'_{1,c} = S_{0,c} \oplus (\{02\} \bullet S_{1,c}) \oplus (\{03\} \bullet S_{2,c}) \oplus S_{3,c}$$

$$S'_{2,c} = S_{0,c} \oplus S_{1,c} \oplus (\{02\} \bullet S_{2,c}) \oplus (\{03\} \bullet S_{3,c})$$

$$S'_{3,c} = (\{03\} \bullet S_{0,c}) \oplus S_{1,c} \oplus S_{2,c} \oplus (\{02\} \bullet S_{3,c}$$

AES Diffusion: Single Byte

Round 1

s00	s01	s02	s03
s10	s11	s12	s13
s20	s21	s22	s23
s30	s31	s32	s33

Input

s00	s01	s02	s03
s11	s12	s13	s10
s22	s23	s20	s21
s33	s30	s31	s32

After ShiftRows

s'00	s'01	s'02	s'03
s'11	s'12	s'13	s'10
s'22	s'23	s'20	s'21
s'33	s'30	s'31	s'32

After MixColumns

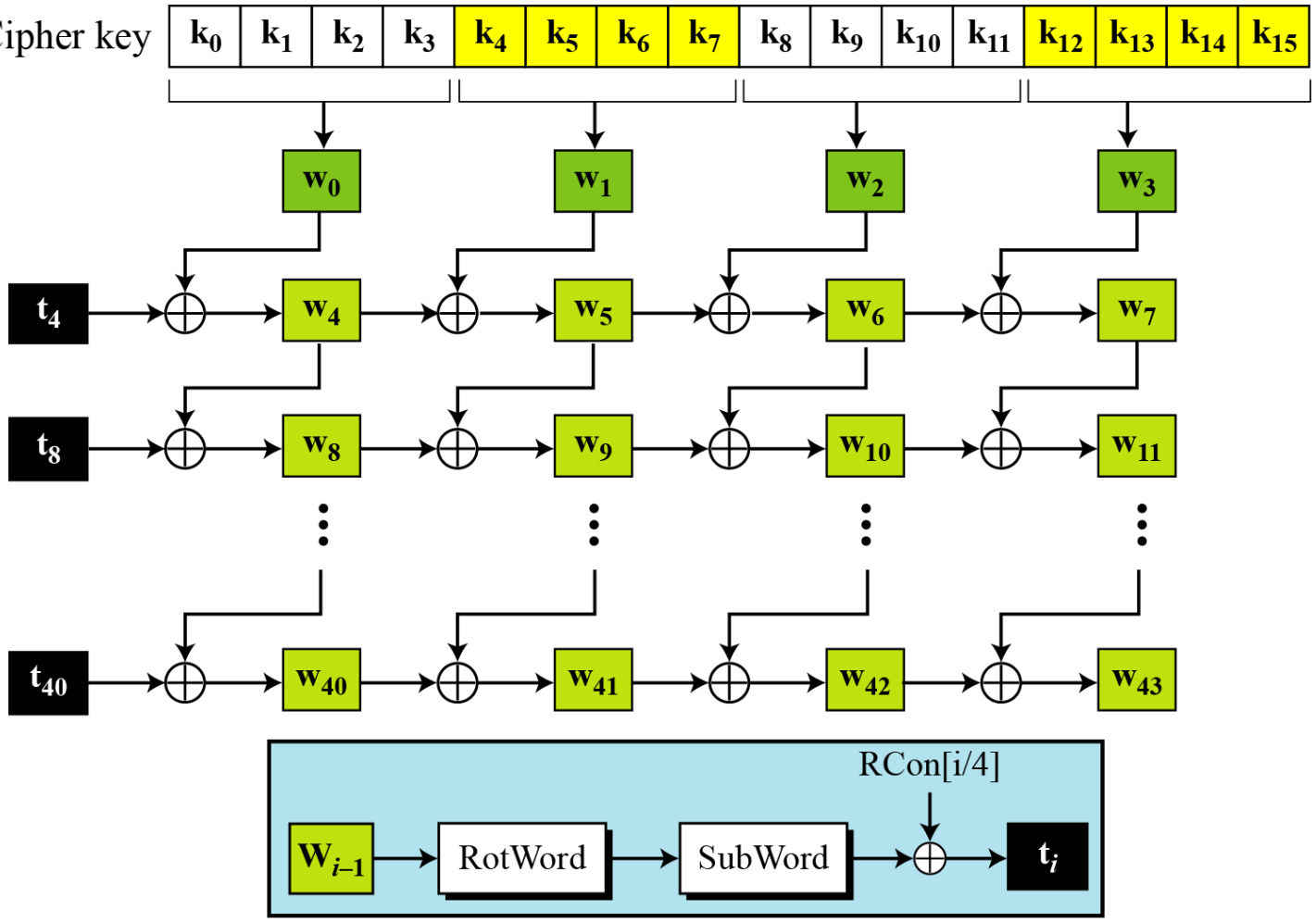
Note: AddRoundKey has no impact on diffusion

Round 2

s'00	s'01	s'02	s'03
s'12	s'13	s'10	s'11
s'20	s'21	s'22	s'23
s'32	s'33	s'30	s'31

s''00	S''01	s''02	s''03
s''12	s''13	s''10	s''11
s''20	s''21	s''22	s''23
s''32	s''33	s''30	s''31

Key Expansion in AES-128



Making of t_i (temporary) words $i = 4\ N_r$

Round	*Constant (RCon)*	*Round*	*Constant (RCon)*
1	$(\underline{\mathbf{01}}\ 00\ 00\ 00)_{16}$	6	$(\underline{\mathbf{20}}\ 00\ 00\ 00)_{16}$
2	$(\underline{\mathbf{02}}\ 00\ 00\ 00)_{16}$	7	$(\underline{\mathbf{40}}\ 00\ 00\ 00)_{16}$
3	$(\underline{\mathbf{04}}\ 00\ 00\ 00)_{16}$	8	$(\underline{\mathbf{80}}\ 00\ 00\ 00)_{16}$
4	$(\underline{\mathbf{08}}\ 00\ 00\ 00)_{16}$	9	$(\underline{\mathbf{1B}}\ 00\ 00\ 00)_{16}$
5	$(\underline{\mathbf{10}}\ 00\ 00\ 00)_{16}$	10	$(\underline{\mathbf{36}}\ 00\ 00\ 00)_{16}$

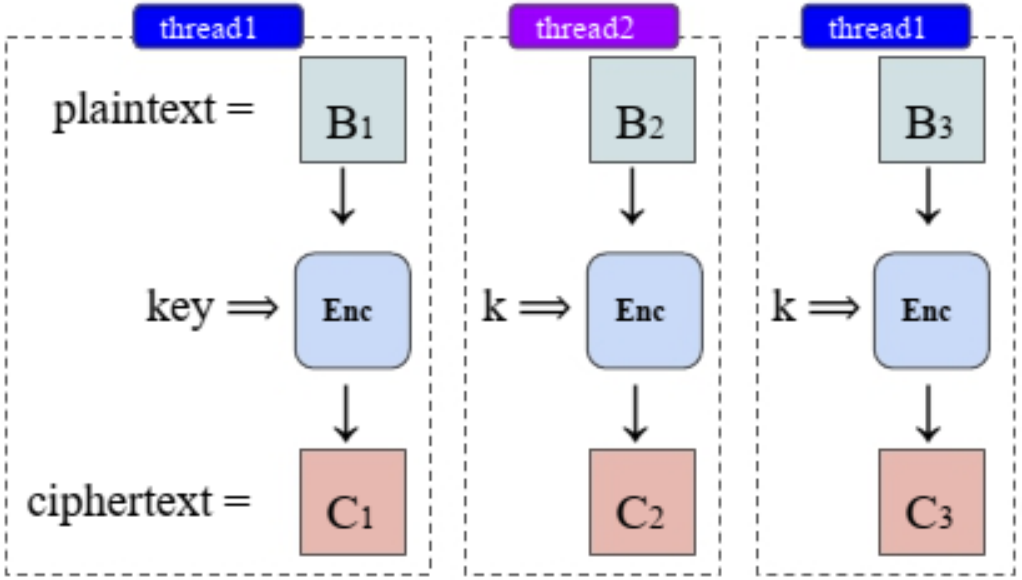
Modes of Operation

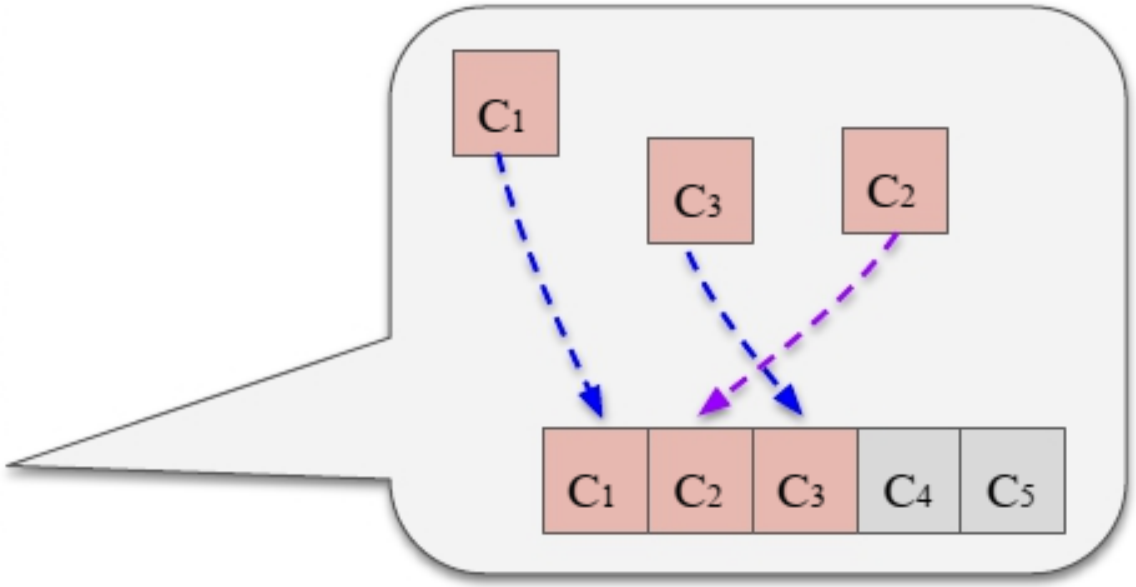
- What do I do if I want to encrypt more that one block of data with a block cipher?
- Simple A: Divide the to-be-encrypted plaintext into block-size chunks, and then apply the cipher to each block.
- Real A: Divide the to-be-encrypted plaintext into block-size chunks, and then apply the cipher to the sequence of blocks using a *mode of operation*.

Mode #1: Electronic Codebook (ECB)

- Electronic Codebook (**ECB**)
  - Message divided into code blocks
  - Each block encrypted **separately**; decrypted separately too

- **ECB Strengths:**
  - Construction is **un-chained**
    - Message can be split up and processed in parallel—**fast!**
    - No need to wait on previous block's encryption

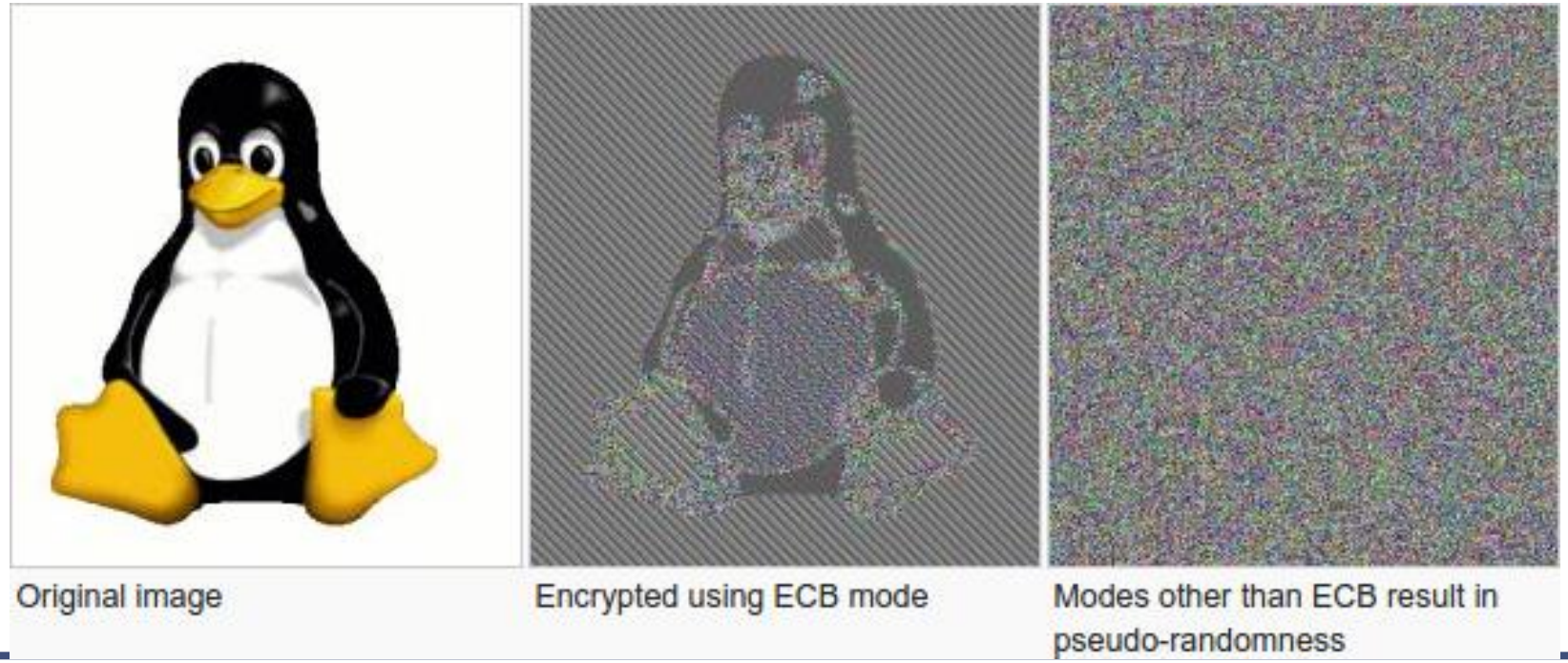




The Penguin Principle

- **Identical plaintext blocks produce identical ciphertext block:** pattern detection
- Patterns not likely in normal text – newspaper, book – due to need to align on block boundary
- Patterns likely in structured text – log files

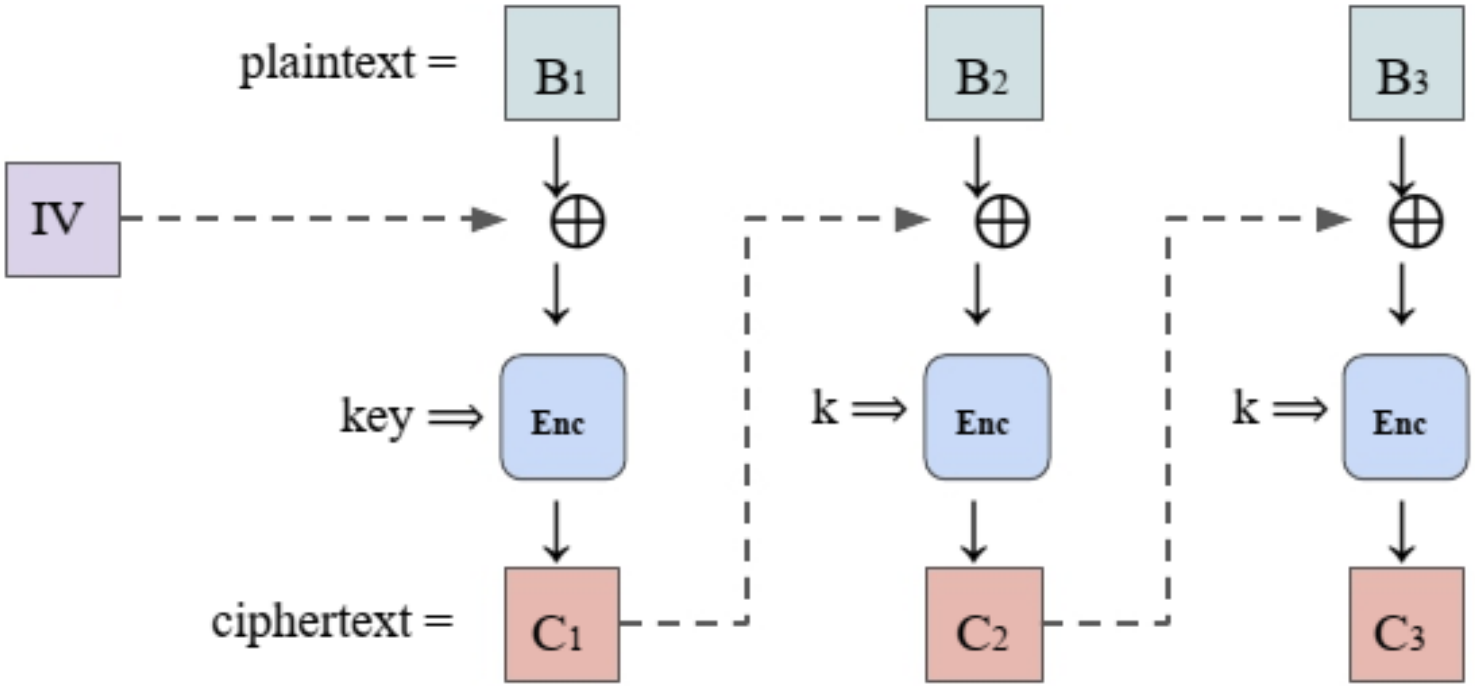
If you can still see the penguin after "encrypting" the image something is very wrong with the encryption scheme.



Original image | Encrypted using ECB mode | Modes other than ECB result in pseudo-randomness

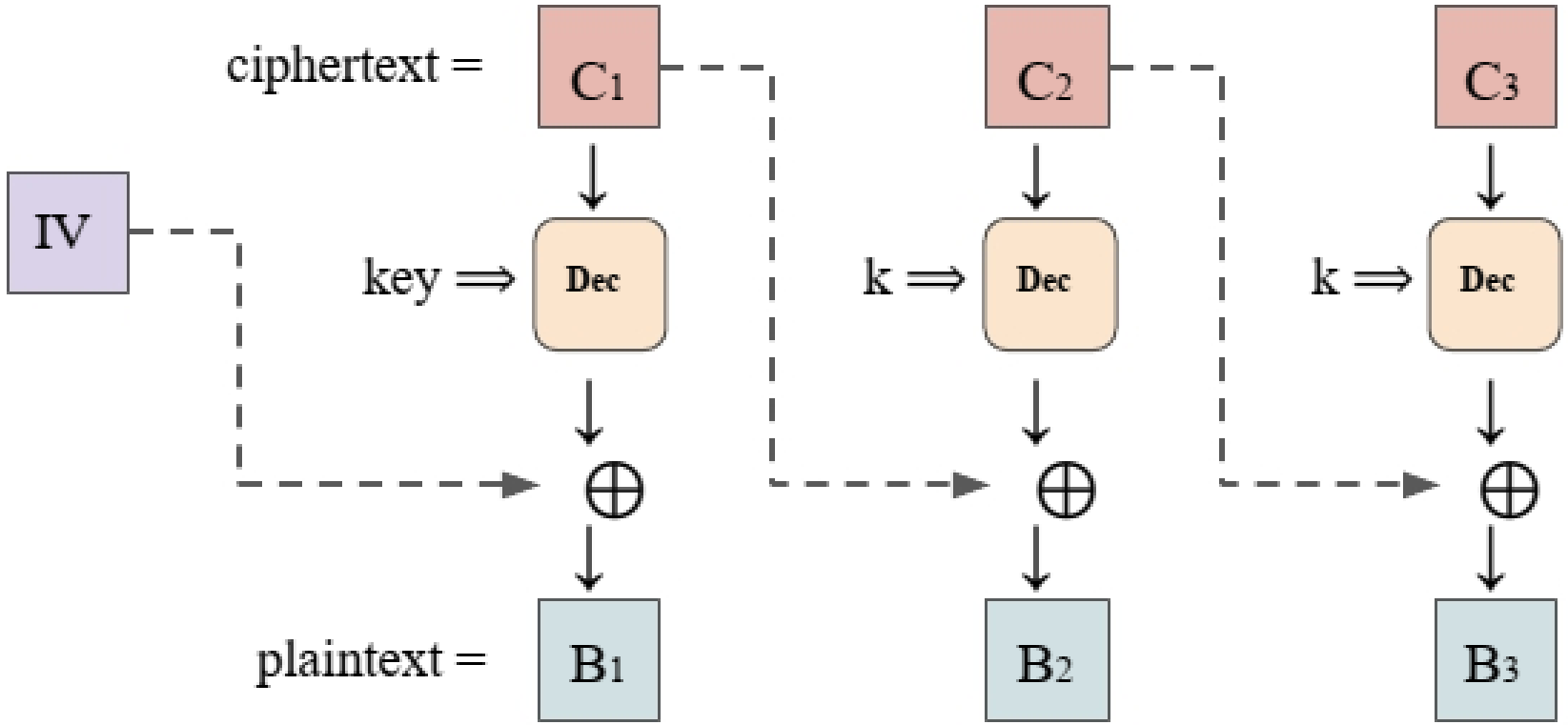
Mode #2: Cipher Block Chaining (CBC)

- **Key idea:** seed current block with **ciphertext from the previous block**
  - Since first block has no "previous" cipher, seed it with a 64-bit initialization vector (I.V.)
    - A **random or pseudo-random** block that's unpredictable



Mode #2: Cipher Block Chaining (CBC)

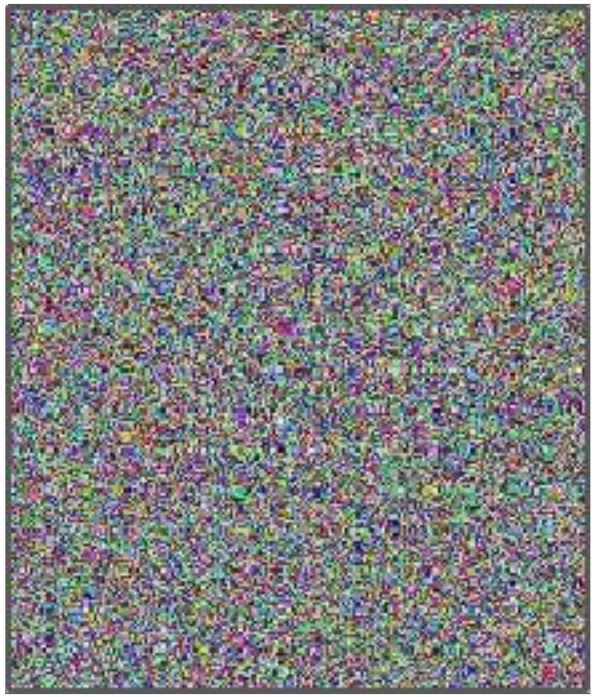
- **Decryption** operates similarly:



Mode #2: Cipher Block Chaining (CBC)

- **CBC Strengths:**
  - Chained construction far stronger than ECB
    - **More diffusion!**
    - Negates ECB's need for super-large blocks

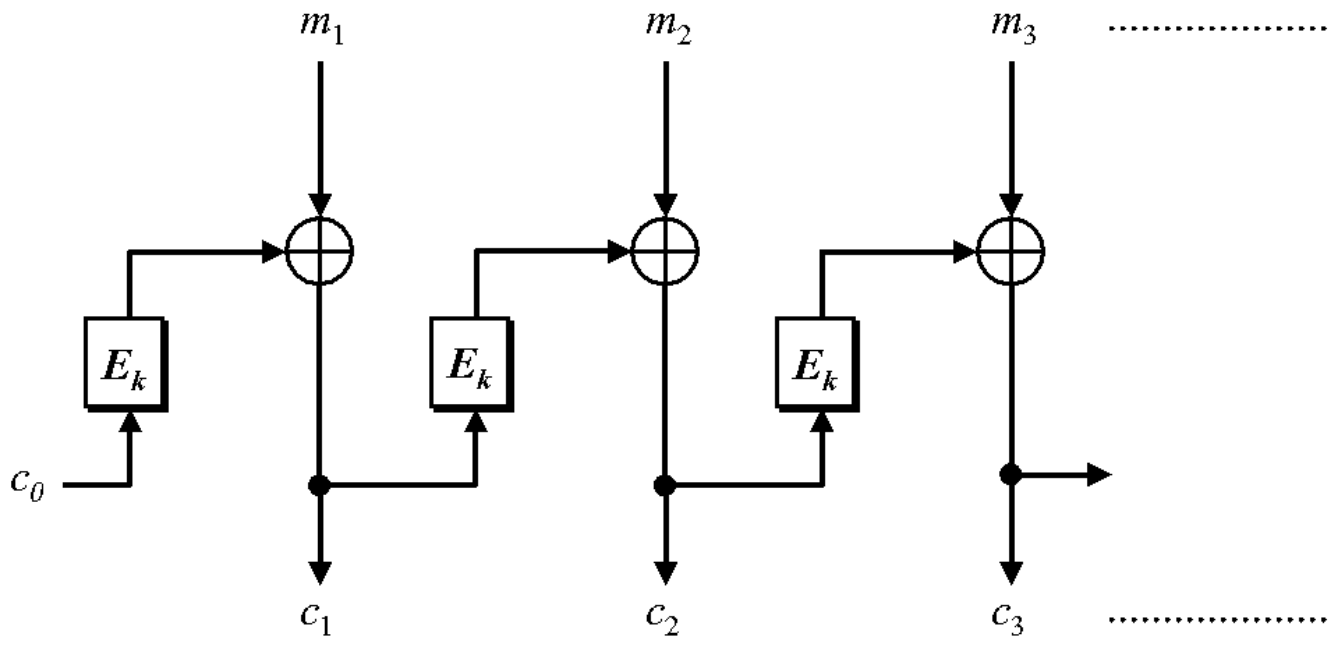
- **CBC Drawbacks:**
  - Completely sequential
    - **Cannot be parallelized!**
    - No leveraging advances in multi-threading etc.



Mode #3: Cipher Feedback Mode (CFB)

- In CFB mode, the previous ciphertext block is encrypted and the output produced is combined with the plaintext block using XOR to produce the current ciphertext block.
- An initialization vector c_0 is used as a *seed* for the process.

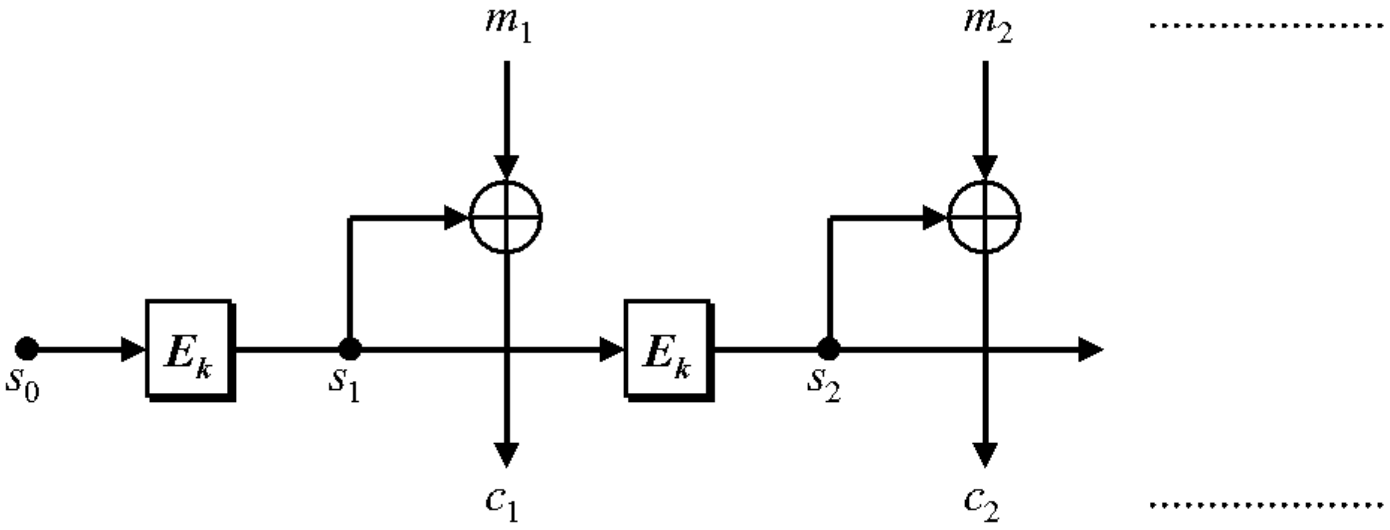
$$C_i = P_i \oplus E_K(C_{i-1})$$



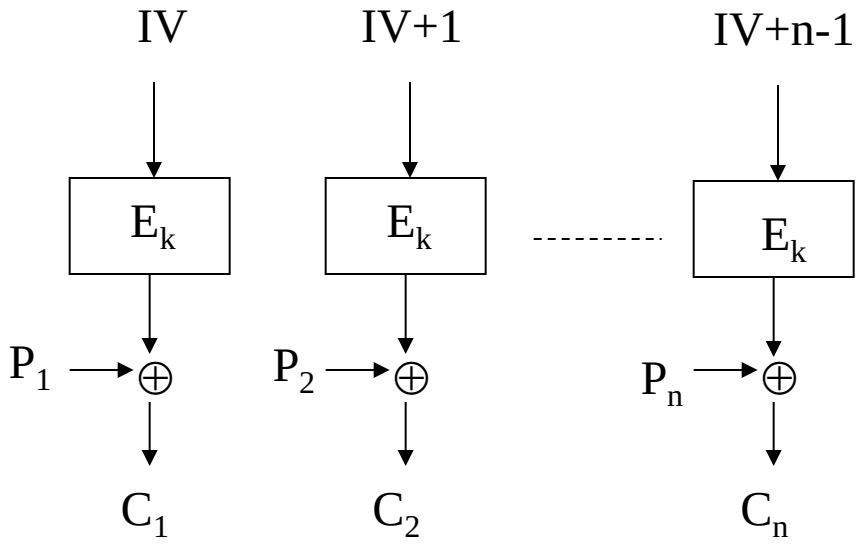
Mode #4: Output Feedback Mode (OFB)

- OFB mode is similar to CFB mode except that the quantity XORed with each plaintext block is generated independently of both the plaintext and ciphertext.
- An initialization vector s_0 is used as a *seed* for a sequence of data blocks s_i, and each data block s_i is derived from the encryption of the previous data block s_{i-1}.

$$C_i = P_i \oplus S_i$$
$$S_i = E_K(S_{i-1})$$



CTR Mode



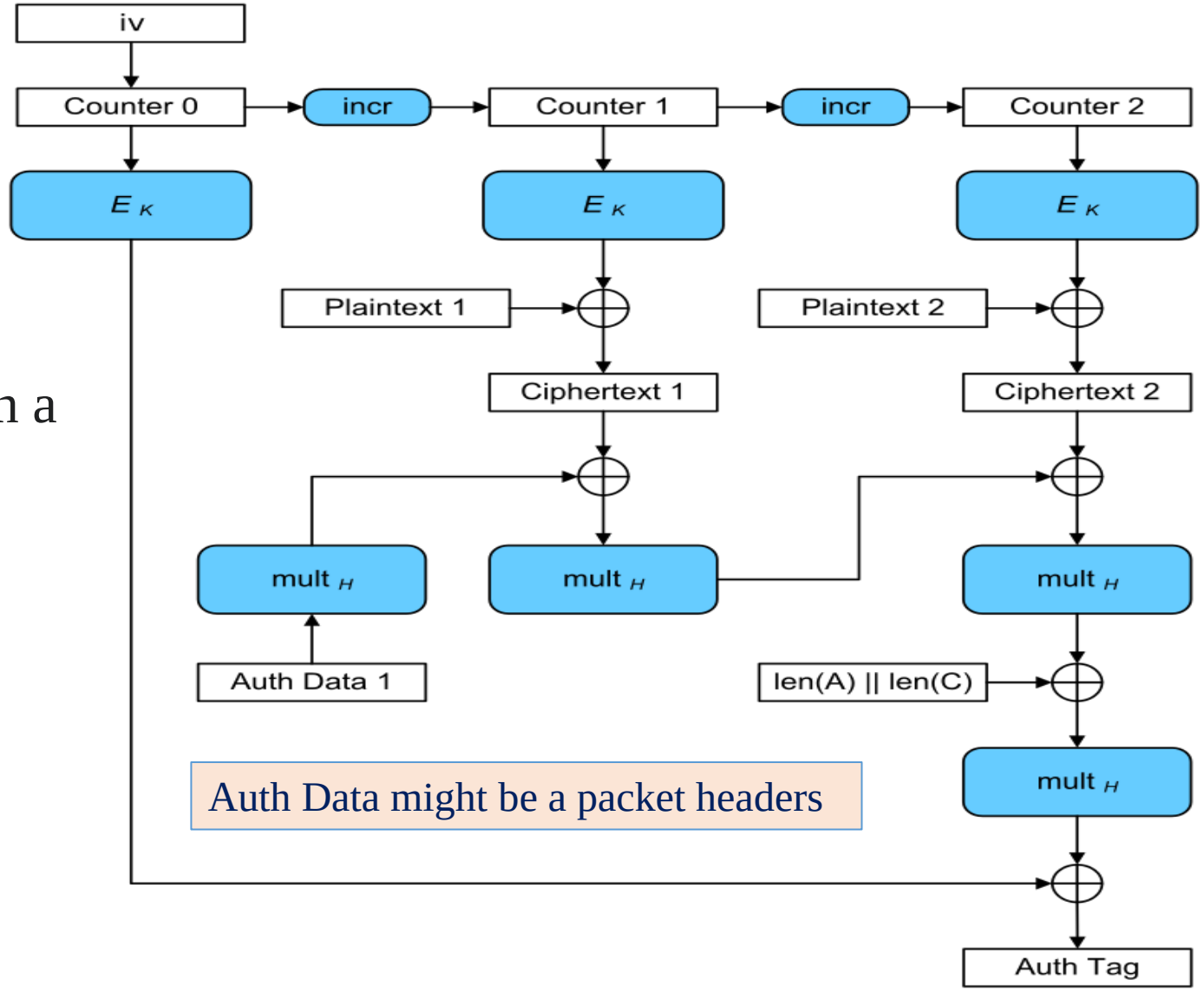
Creates key stream and XORs with plaintext
Need to avoid reusing key and IV+i value combination

Galois Counter Mode

- Authentication Encryption with Associated Data
  - Ensure integrity of ciphertext
  - Ensures attacker cannot tamper with associated packet data
    - Source IP, Destination IP, Why can’t these values be encrypted?
- Encryption is largely parallelizable!
- Examples of industrial standards using GCM
  - IEEE 802.1AE (MACsec), WPA3-Enterprise
  - TLS 1.2 and 1.3.
  - IPSEC, SSH
  - Data storage and transmission: IEEE P1619.1
  - Satellite communication: Industrial applications use AES-GCM for secure data transmission between ground stations and satellites.

Galois Counter Mode (GCM)

$mult_H$ is a multiplication with a key-dependent constant H



Auth Data might be a packet headers

Questions?