CSE467: Computer Security

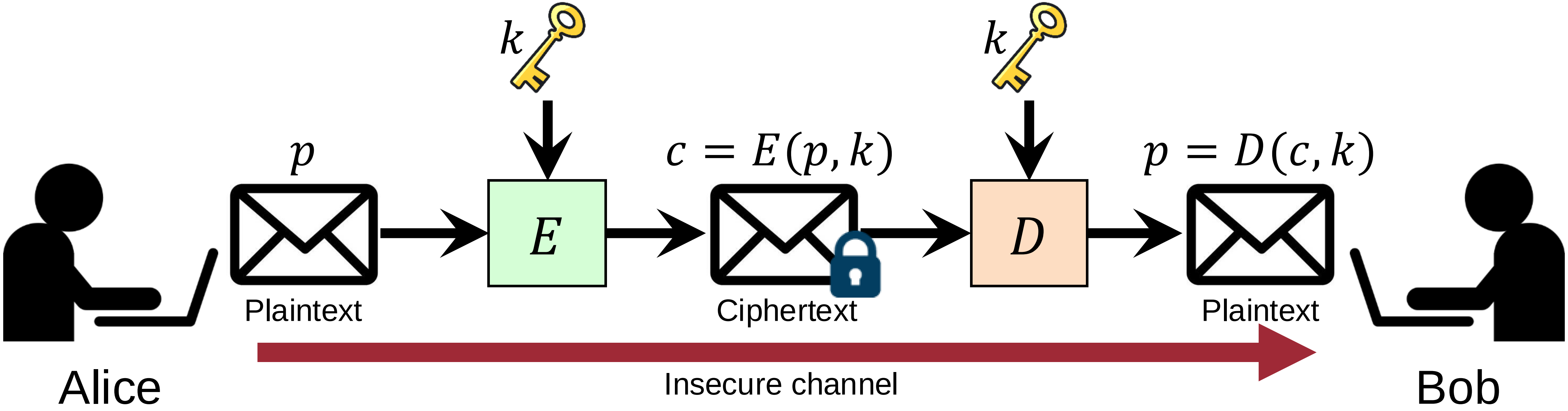
4. Symmetric-key Encryption (1)

Seongil Wi

Department of Computer Science and Engineering

Recap: Cryptography

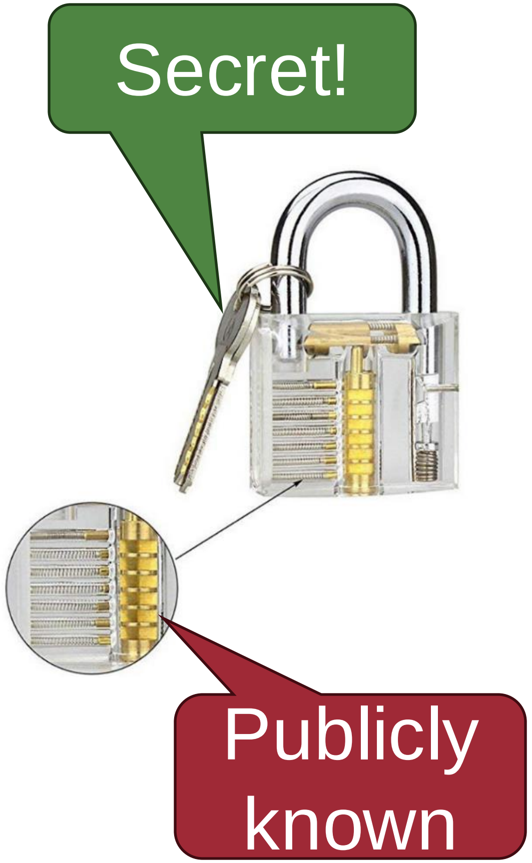
- "Secret writing" in Greek
- Goal: protect your (sensitive) messages/data from eavesdropping
- The most basic building block of computer security
- Two functions: encryption (E) and decryption (D) parameterized by a plaintext (p), ciphertext (c), and cryptographic key (k)



Recap: Kerckhoff’s Principle

You should always assume that the ***adversary knows the encryption/decryption algorithm***!

\- Auguste Kerckhoffs



The resistance of the cipher must be based only on the ***secrecy of the key*** 

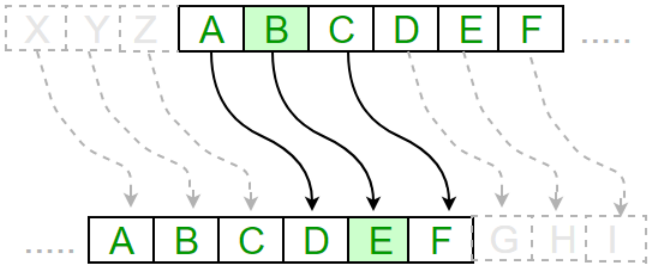
Recap: Caesar Cipher

- Encryption: shift each plaintext character k places forward

E $E(p, k) = (p + k)\ mod\ 26$

D $D(c, k) = (c - k)\ mod\ 26$

Q. How many other keys could be chosen?



Q. Robust enough?

Recap: Substitution Cipher

- One-to-one mapping (bijection)
- Example:
  - Plaintext: `eungyeongbaek`
  - Key: Substitution mapping table

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
q	w	e	r	t	y	u	i	o	p	a	s	d	f	g	h	j	k	l	z	x	c	v	b	n	m

  - Ciphertext: `txfuntgfuwqta`

- Key space?
  - $26! \approx 2^{88} \approx 4 \times 10^{26}$

- Q. Robust enough?

Recap: Vigenere Cipher

- Encryption: poly-alphabetic **shift**

E $$E(p,k) = (p_i + k_i)\ mod\ 26$$

D $$D(c,k) = (c_i - k_i)\ mod\ 26$$

Invented in 16th century and had been unbreakable for hundreds of years!

- Example
  - Plaintext: `tellhimaboutme`
  - Key (repeated): `cafecafecafeca`
  - Ciphertext: `veqpjiredozxoe`

Problems?

- ***Letters are mapped to different ciphertexts***: smooth out the frequency distribution in ciphertext

Recap: Adversary Assumptions

- What are the adversary capabilities?
- Attacker capabilities (in order of increasing attack power)
  - **Ciphertext-only attack**: most basic attack
  - **Known-plaintext attack**: attacker obtains certain plaintext/ciphertext pairs
  - **Chosen-plaintext attack**: attacker obtains plaintext/ciphertext pairs for plaintext of its choice
  - **Chosen-ciphertext attack**: attacker obtains plaintext/ciphertext pairs for ciphertext of its choice

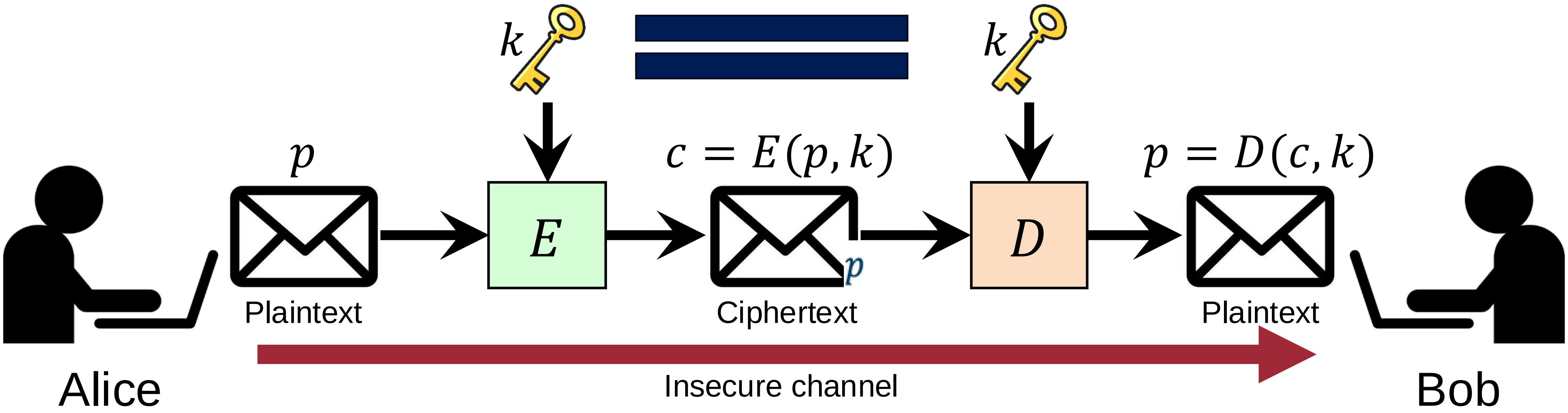
Recap: Brute Force Search

- Always possible to simply try every key!
- Therefore, the key should be secure against **exhaustive key search**!

Key size (Bits)	# of alternative keys	Time required at 1 decryption/μs	Time required at 10^6 decryption/μs
32	$2^{32} = 4.3 \times 10^9$	2^{31}μs = 5.8 minutes	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	2^{55}μs = 1,142 years	10.01 hours
128	$2^{128} = 3.4 \times 10^{38}$	2^{127}μs = 5.4 $\times$ 10^{24} years	5.4 $\times$ 10^{18} years
168	$2^{168} = 3.7 \times 10^{50}$	2^{167}μs = 5.9 $\times$ 10^{36} years	5.9 $\times$ 10^{30} years

Today's Topic: Symmetric-key Encryption

- **Symmetric**: the encryption and decryption keys *are the same*
- Plaintexts and ciphertexts are all bit vectors from now on (for simplicity!)



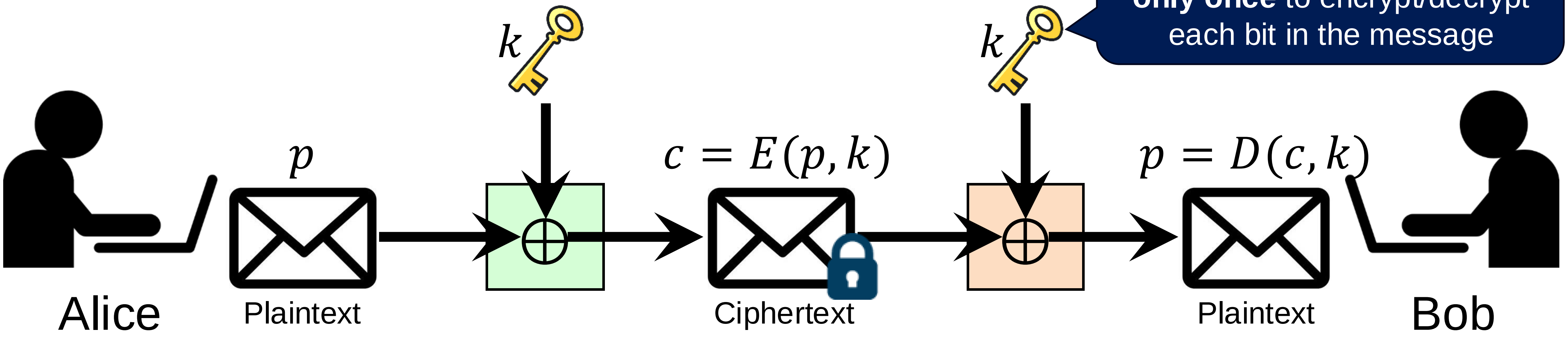
Stream Ciphers vs. Block Ciphers

- How the data gets encrypted?
  - **Stream ciphers** encrypt bits individually
    - E.g., One-time Pad, RC4, …
  - **Block ciphers** encrypt fixed-length groups of bits, *called blocks*, at a time, applying the algorithm to the each block
    - Used in most modern algorithms
    - E.g., DES, Triple-DES, AES, …

Stream Ciphers (Example): One-time Pad

- a.k.a Vernam cipher (≠ one-time password)
- Let a message p be of length n bits
- Assume that we have a key k that is a **completely random sequence** of bits of length n
- Then we can use $E(p,k) = p \oplus k$ and $D(c,k) = c \oplus k$ for encryption and decryption, respectively

Every key stream bit k_i is used **only once** to encrypt/decrypt each bit in the message



Stream Ciphers (Example): One-time Pad

- a.k.a Vernam cipher ($\neq$ one-time password)
- Let a message p be of length n bits
- Assume that we have a key k that is a **completely random sequence** of bits of length n
- Then we can use $E(p,k) = p \oplus k$ and $D(c,k) = c \oplus k$ for encryption and decryption, respectively

k 

k 

Every key stream bit k_i is used **only once** to encrypt/decrypt each bit in the message

The One-time Pad (OTP) is *unconditionally secure*
(i.e., cannot be broken even with infinite computational resources)
=> Then, what is the main drawback of OTP?

Limitations of One Time Pad

- The OTP (i.e., key) should be truly random
- The OTP should be at least as long as the message
- Both copies of the OTPs are destroyed immediately after use

Towards Practical Encryption Schemes

Block Ciphers

Towards Practical Encryption Schemes

- The OTP (i.e., key) should be truly random
- The OTP should be at least as long as the message
- Both copies of the OTPs are destroyed immediately after use

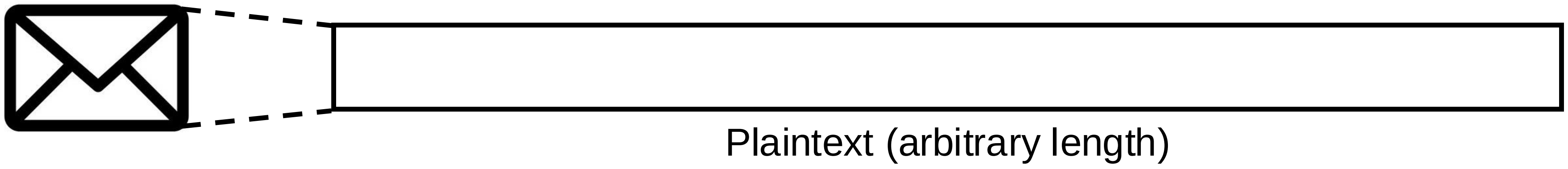
Fixed-length key, plaintext, ciphertext

Reusing a fixed-length key

Block Ciphers!

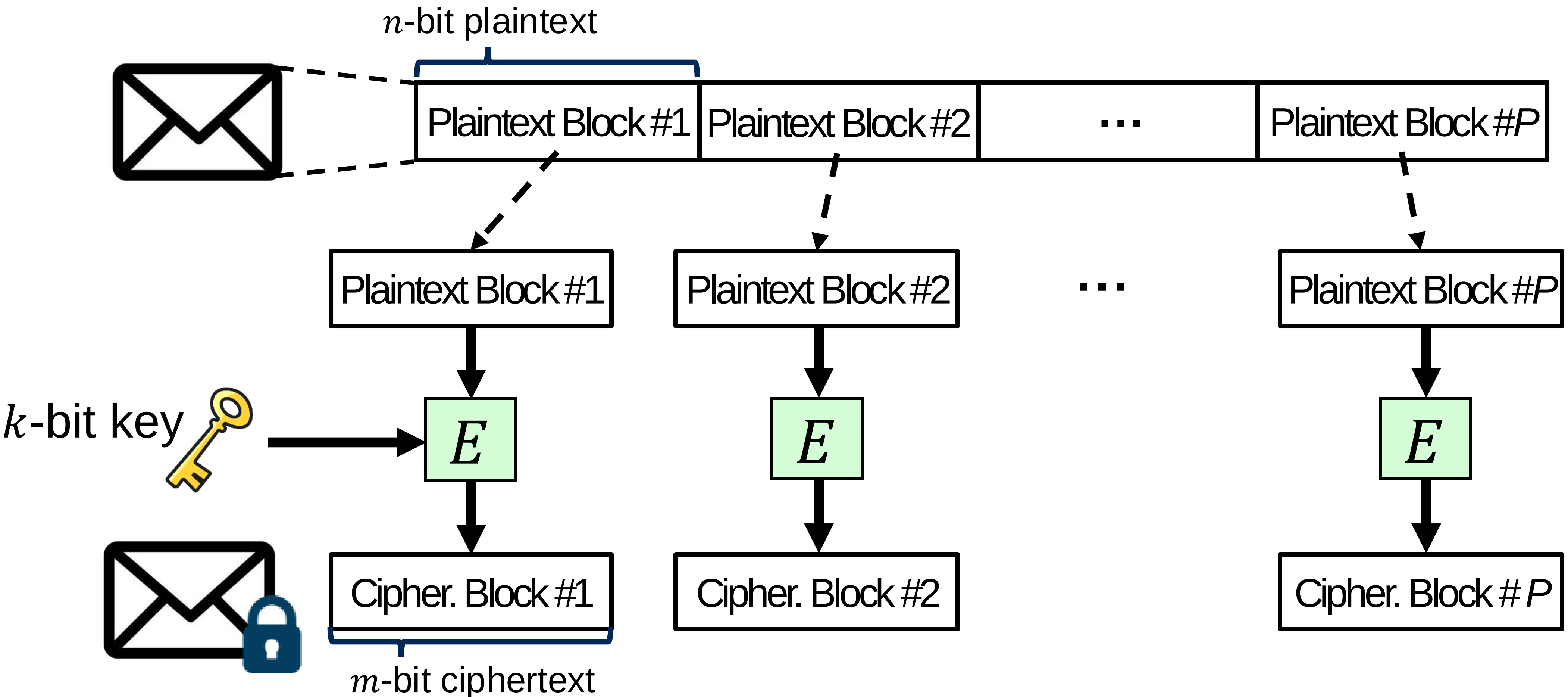
Block Ciphers

- Encrypt/decrypt data in blocks of fixed lengths (n-bit block)
- Split a message into blocks and encrypt them individually



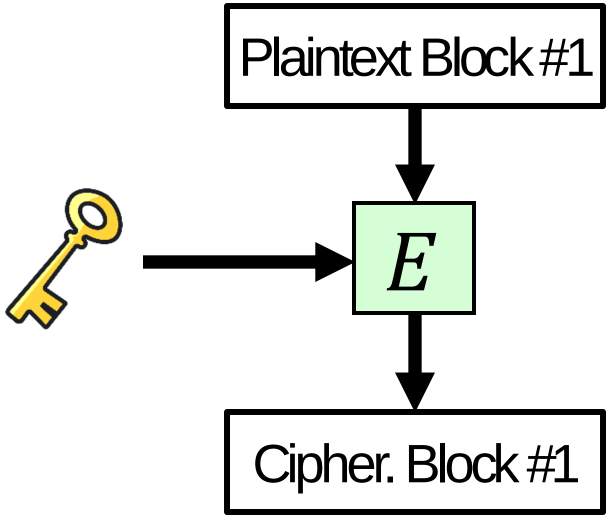
Block Ciphers

- Encrypt/decrypt data in blocks of fixed lengths (n-bit block)
- Split a message into blocks and encrypt them individually



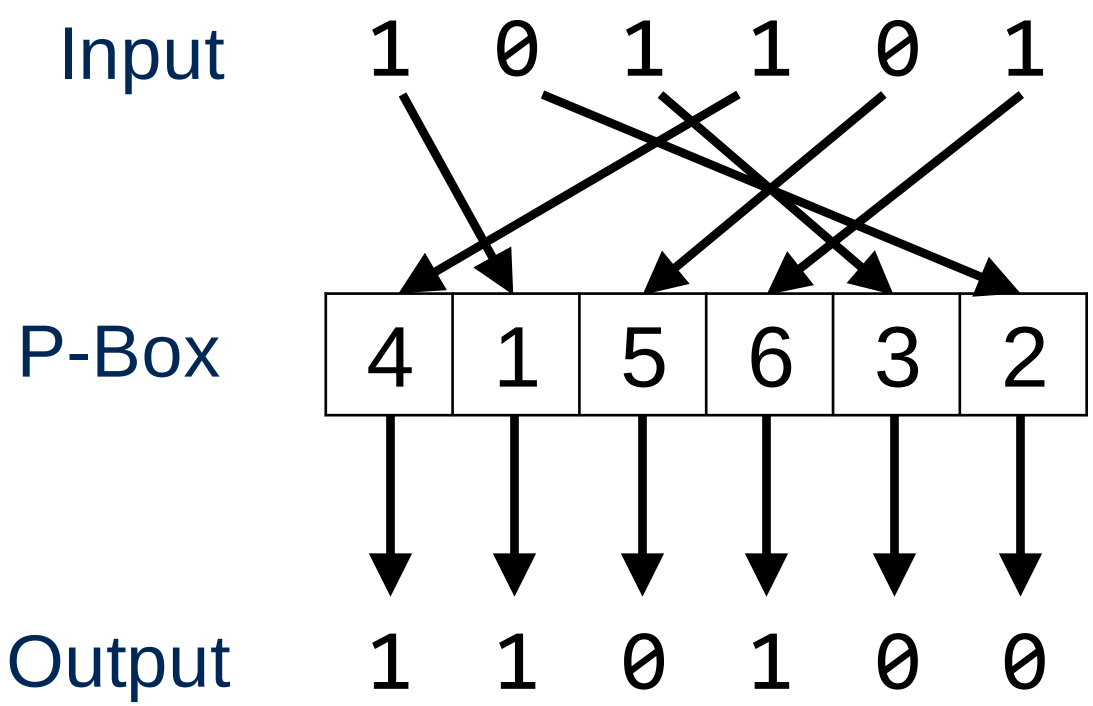
Components of a Modern Block Cipher

- A modern block ciphers are usually made of combination of
  - **Transposition cipher (called P-box)**
  - **Substitution cipher (called S-box),**
  - **Some other units (e.g., XOR, circular shift, swap, split, and combine)**



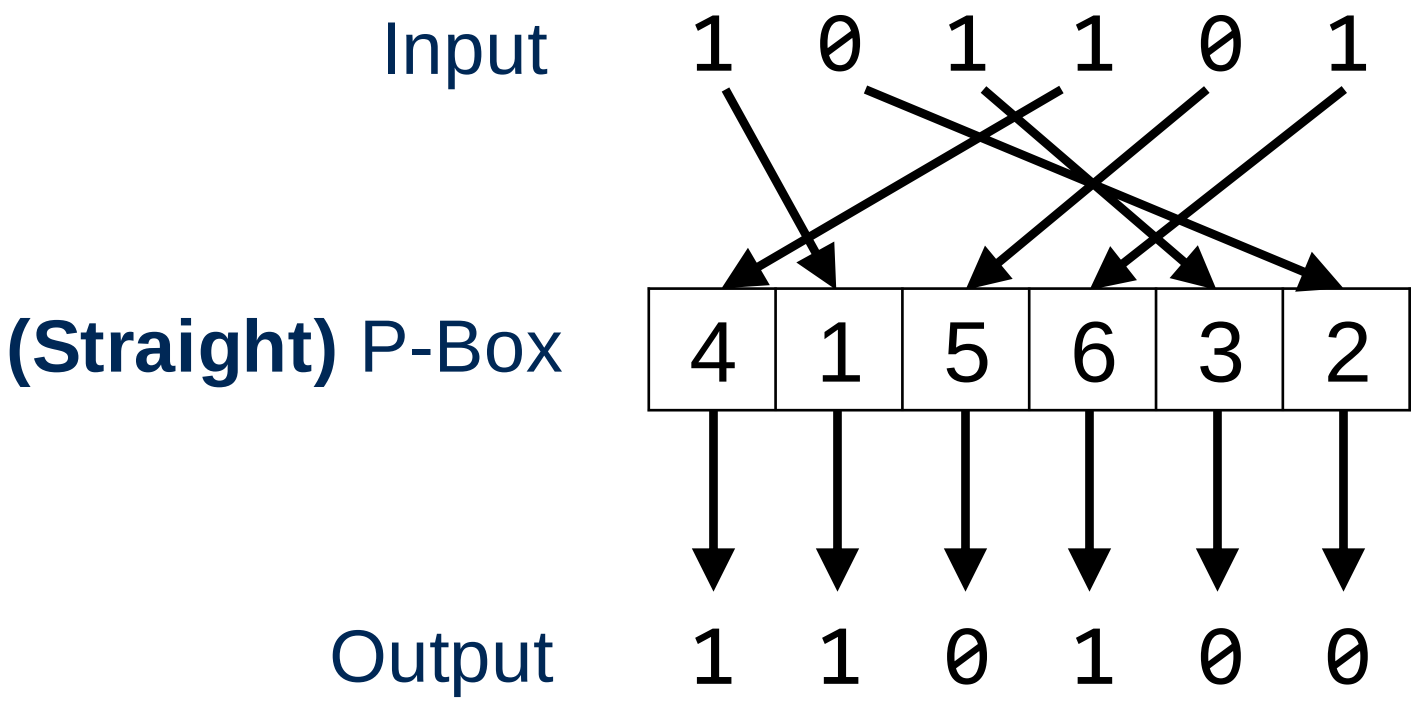
Transposition Cipher (P-Box)

- A P-box (permutation box) parallels the traditional transposition cipher. It transposes bits.



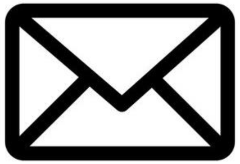
Three Types of P-Boxes

- Straight P-box
- Compression P-box
- Expansion P-box



Exercise: Invertibility

- Invert a P-box (used for decryption)



1 0 1 1 0 1

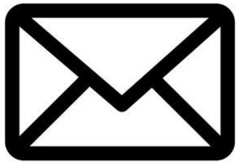
Original P-Box
(used for encryption)

6	3	4	5	2	1

1 1 1 0 0 1

Inverted P-Box
(used for decryption)

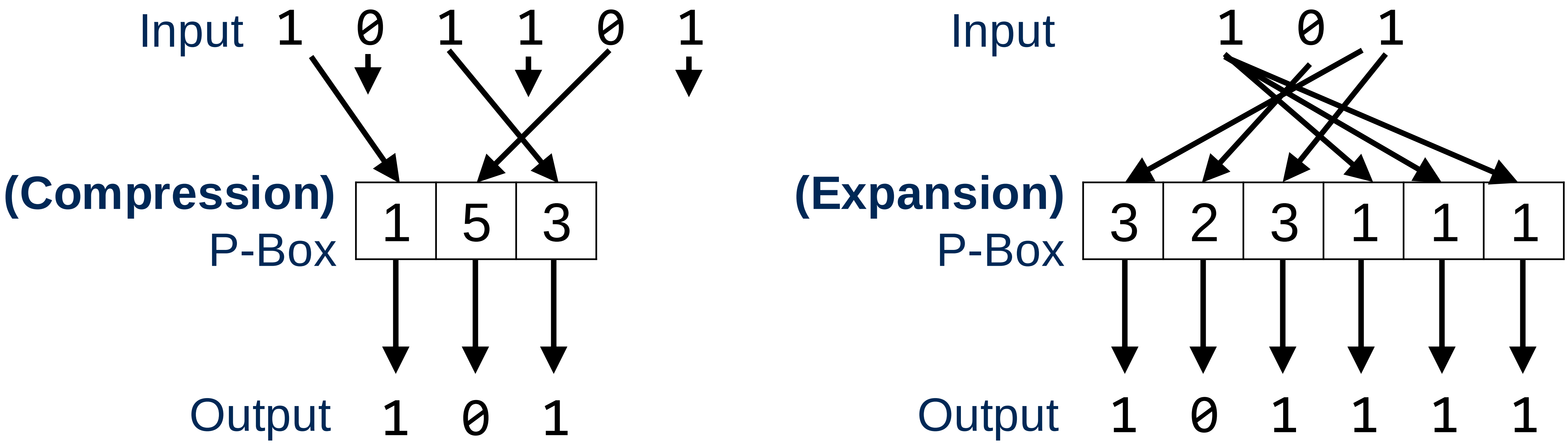
6	5	2	3	4	1



1 0 1 1 0 1

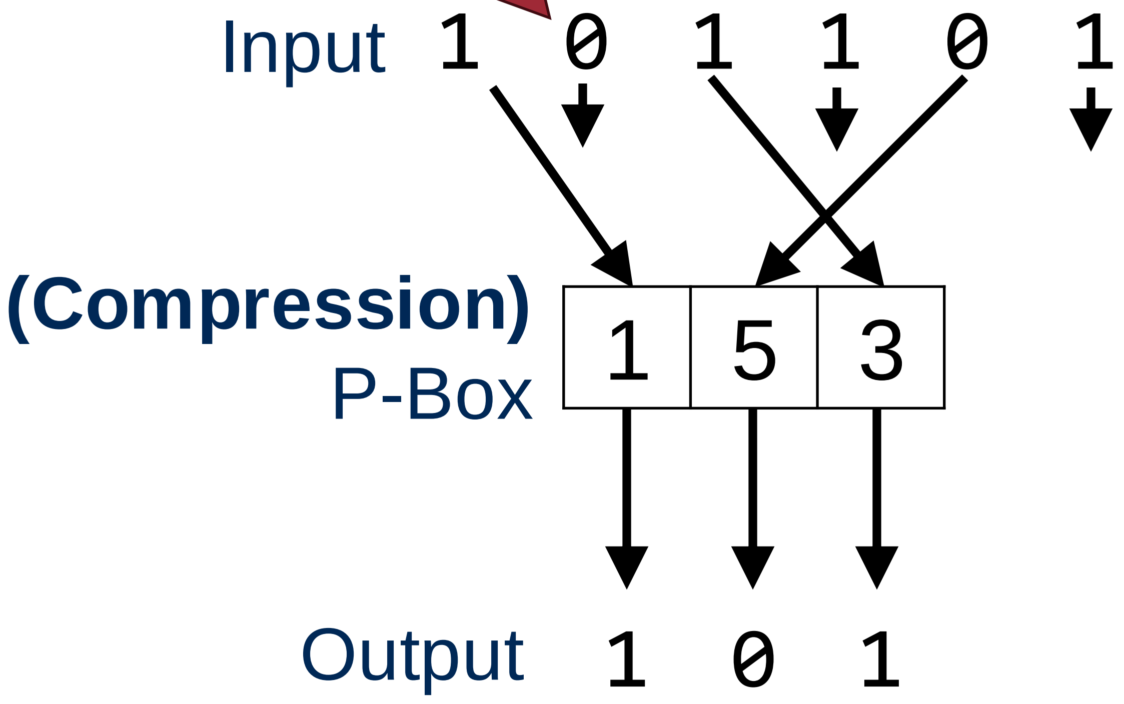
Three Types of P-Boxes

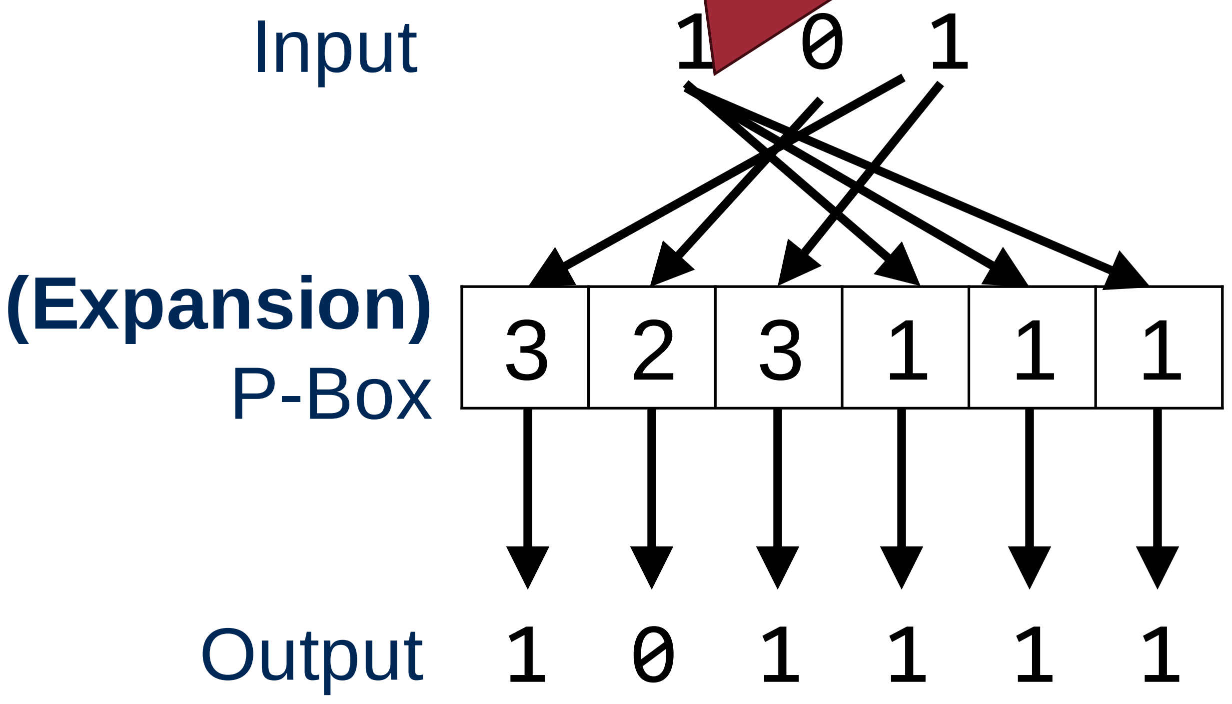
- Straight P-box
- Compression P-box: n inputs and m outputs where $m < n$
- Expansion P-box: n inputs and m outputs where $m > n$

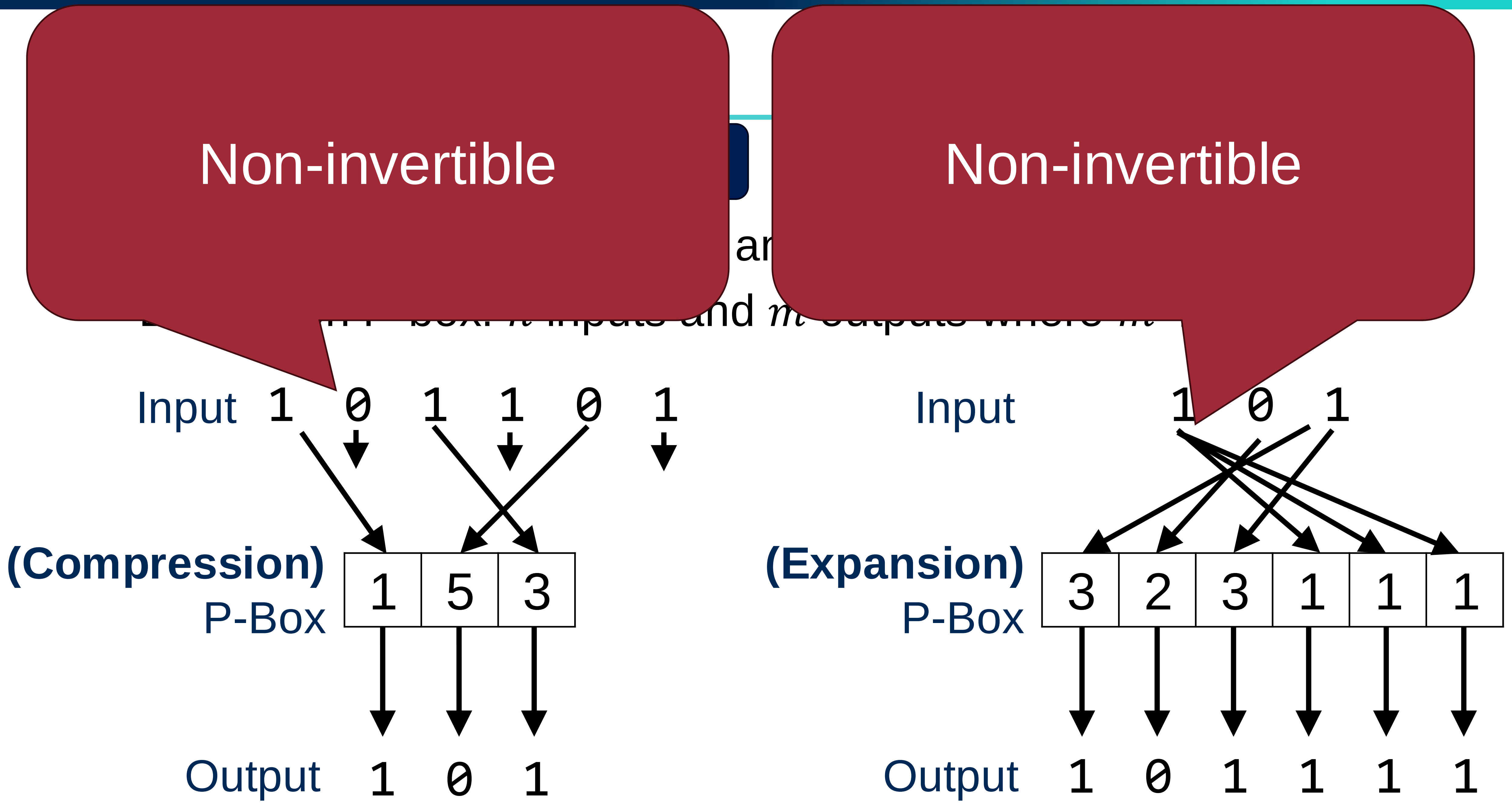


An input can be ***dropped***!
=> The decryption algorithm does not have a clue *how to replace the dropped bit*

An input can be mapped to ***more than one output***
=> The decryption algorithm does not have a clue *which of the several inputs are mapped to an output*






Non-invertible
Non-invertible
Input 1 0 1 1 0 1
(Compression)
P-Box
1 5 3
Output 1 0 1
Input 1 0 1
(Expansion)
P-Box
3 2 3 1 1 1
Output 1 0 1 1 1 1

Invertibility of P-Boxes

- Straight P-box
- Compression P-box: n inputs and m outputs where $m < n$
- Expansion P-box: n inputs and m outputs where $m > n$

Invertible

Non-invertible

Non-invertible

However, there are ciphers that use compression or expansion P-boxes

😀

Substitution Cipher (S-Box)

- An S-box (substitution box) can be thought of as a miniature substitution cipher
- $m \times n$ substitution units, where m and n are not necessarily the same

Substitution Cipher (S-Box)

- An S-box (substitution box) can be thought of as a miniature substitution cipher
- $m \times n$ substitution units, where m and n are not necessarily the same

Input `1 0 1`

S-box	00	01	10	11
0	00	10	01	11
1	10	$\boxed{00}$	11	01

Leftmost bit

Rightmost bits

Output `0 0`

Q: Is S-Box invertible?

Invertibility of S-Boxes

- An S-box *may* or *my not* be invertible
- In an invertible S-box, the number of input bits should be the same as the number of output bits

- E.g., invertible S-Boxes

3 bits 0 0 1

Encryption

S-box	00	01	10	11
0	011	101	111	100
1	000	010	001	110

3 bits 1 0 1

3 bits 1 0 1

Decryption

S-box	00	01	10	11
0	100	110	101	000
1	011	001	111	010

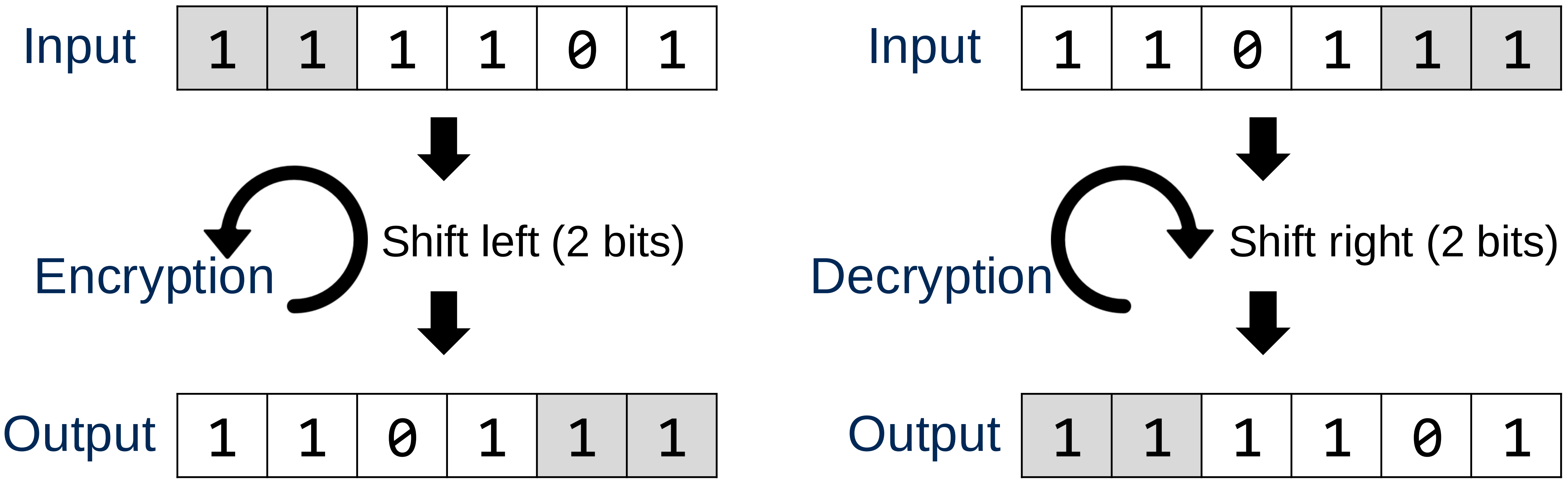
3 bits 0 0 1

Exclusive-Or (XOR)

- An important component in most block ciphers
- Invertible operation

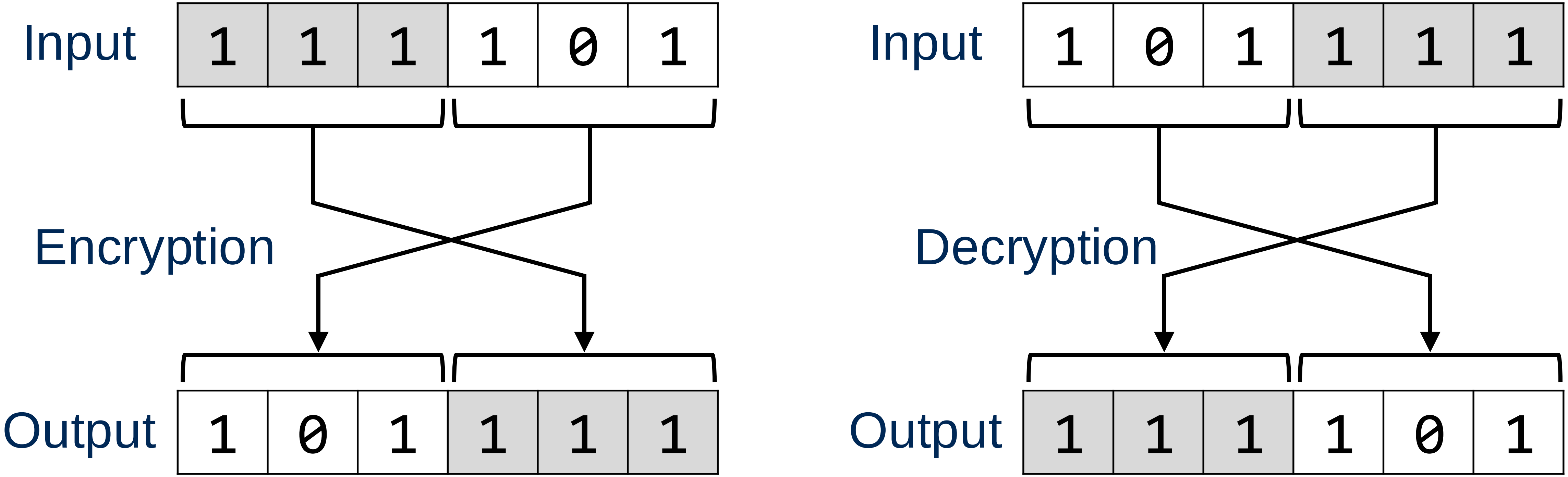
Circular Shift

- Recall the Caesar Cipher
- Invertible operation



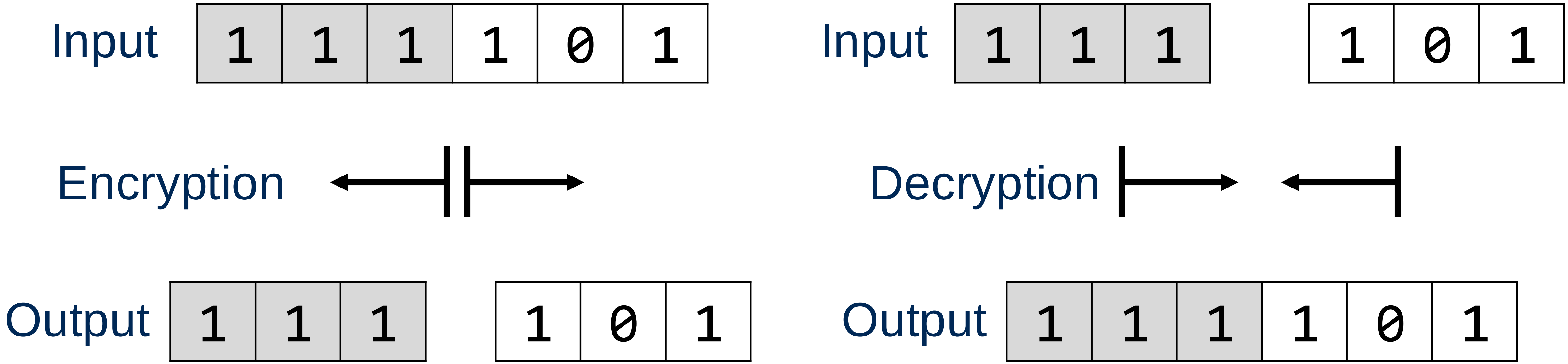
Swap

- A special case of the circular shift operation where $k = n/2$
- Invertible operation



Split and Combine

- Invertible operations



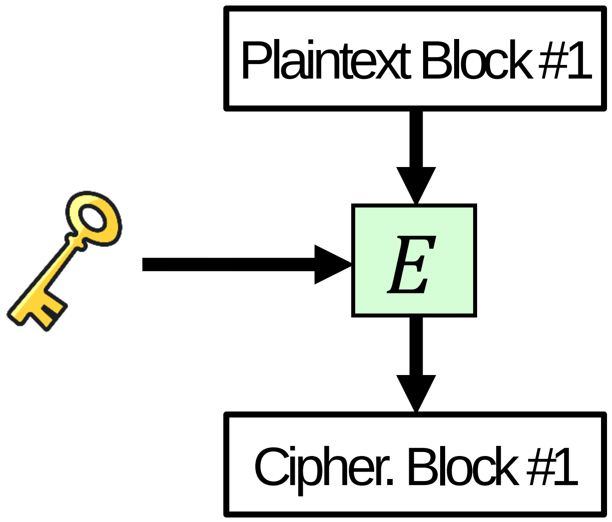
Design Principles for Block Ciphers

- A modern block ciphers are usually made of combination of
  - Transposition cipher
  - Substitution cipher
  - Some other units

Property #1: Diffusion

Property #2: Confusion

Property #3: Rounds



Property #1: Diffusion

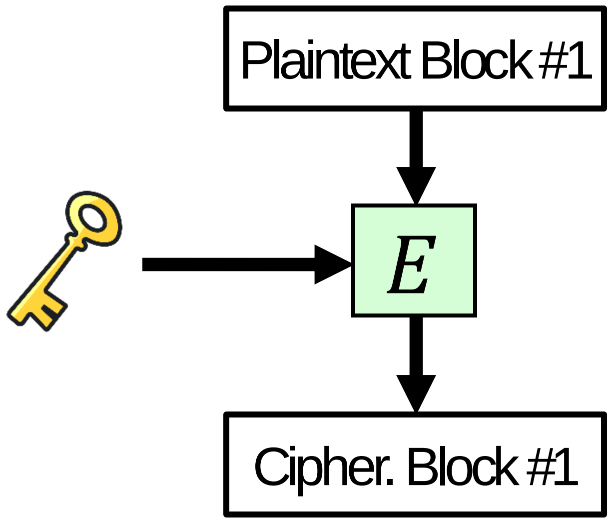
- A modern block ciphers are usually made of combination of
  er

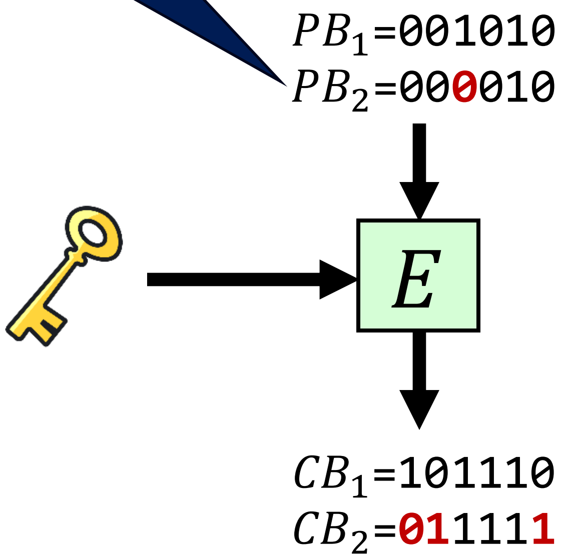
The influence of **one plaintext bit** is spread ***over many ciphertext bits***

Property #1: Diffusion

Property #2: Confusion

Property #3: Rounds





Recall: Chosen-Plaintext Attack (CPA)

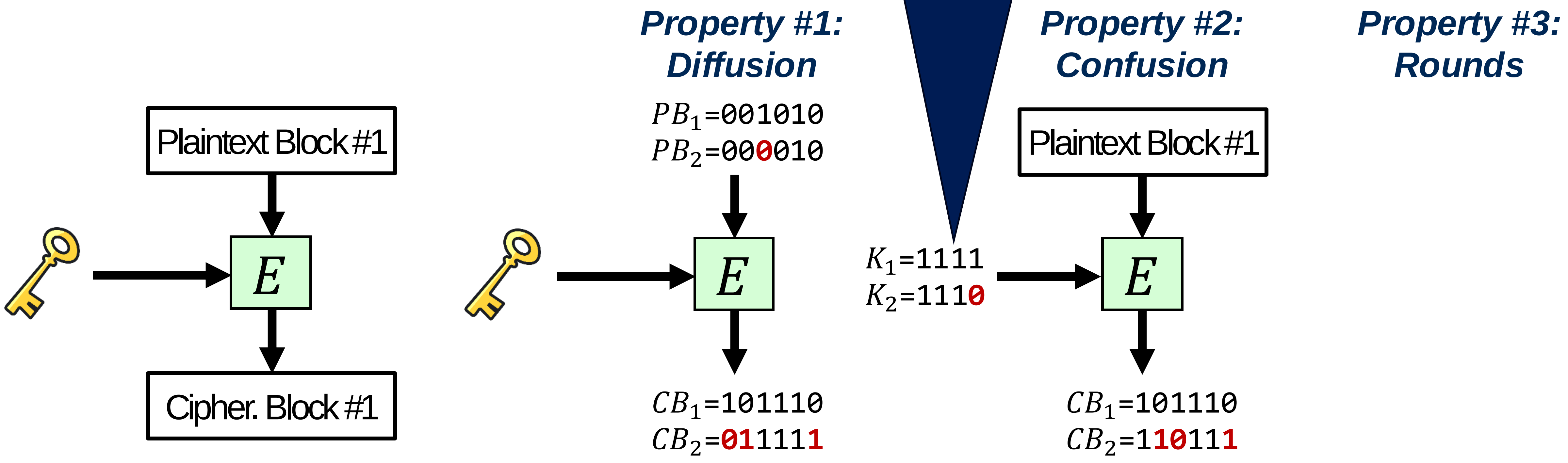
- ✓ The bit we vary is consistently negated
- ✓ As one bit varies, the remaining ones are left unchanged

	Plaintext	Ciphertext
Try #1	11111	01001
Try #2	11110	01000
Try #3	11101	01011
Try #4	11011	01101
Try #5	10111	00001
Try #6	01111	11001

Property #2: Confusion

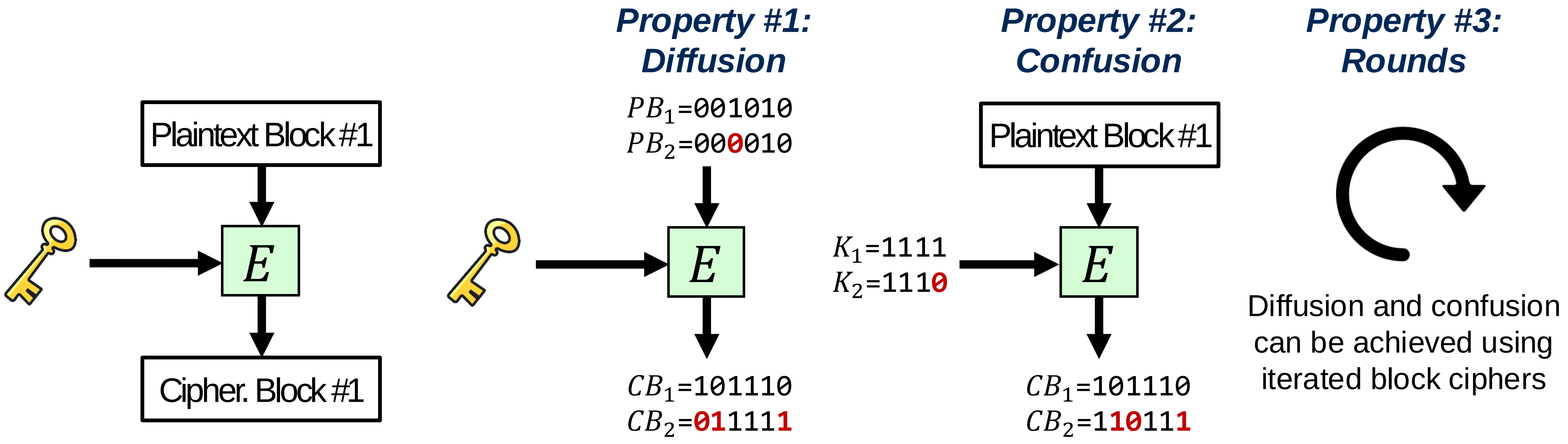
- A modern block ciphers are usually made of combination of
  - Transposition cipher
  - Substitution cipher
  - Some other units

The influence of **one key bit** is spread ***over many ciphertext bits***

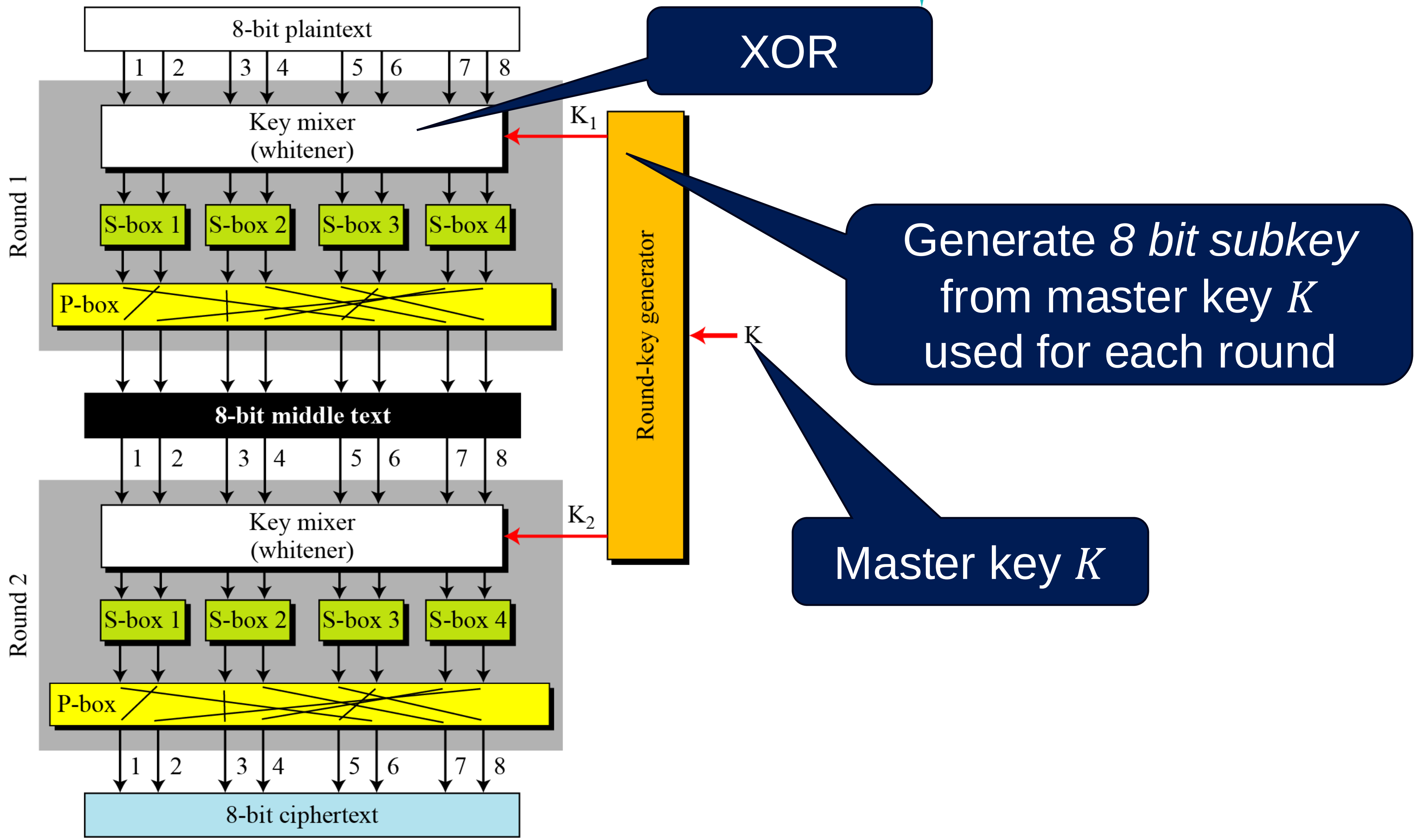


Property #3: Rounds

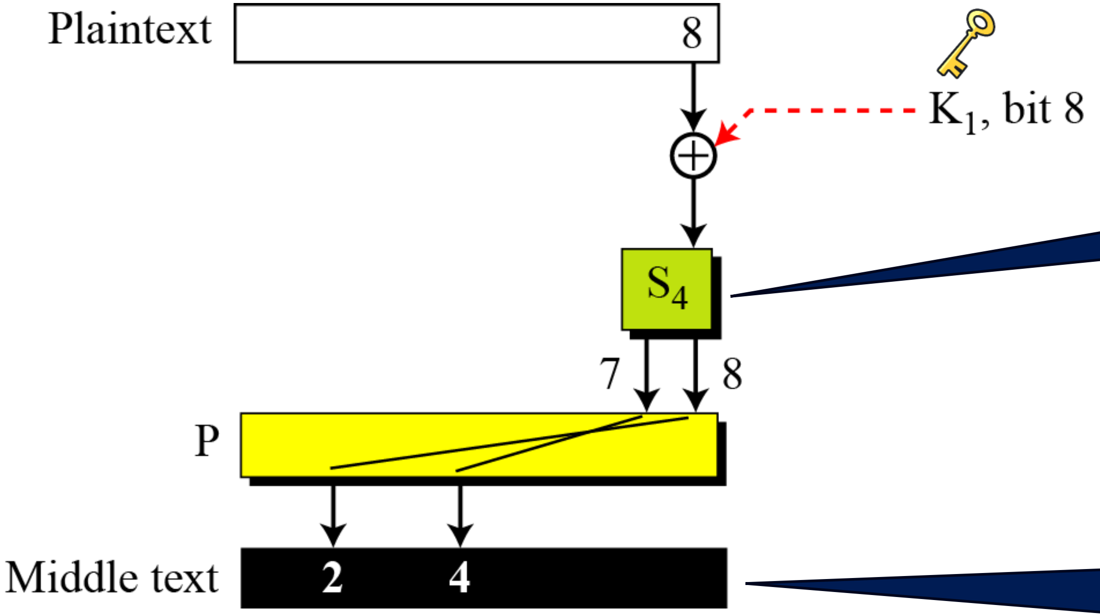
- A modern block ciphers are usually made of combination of
  - Transposition cipher
  - Substitution cipher
  - Some other units



Example: A Block Cipher made of Two Rounds



Example: A Block Cipher made of Two Rounds

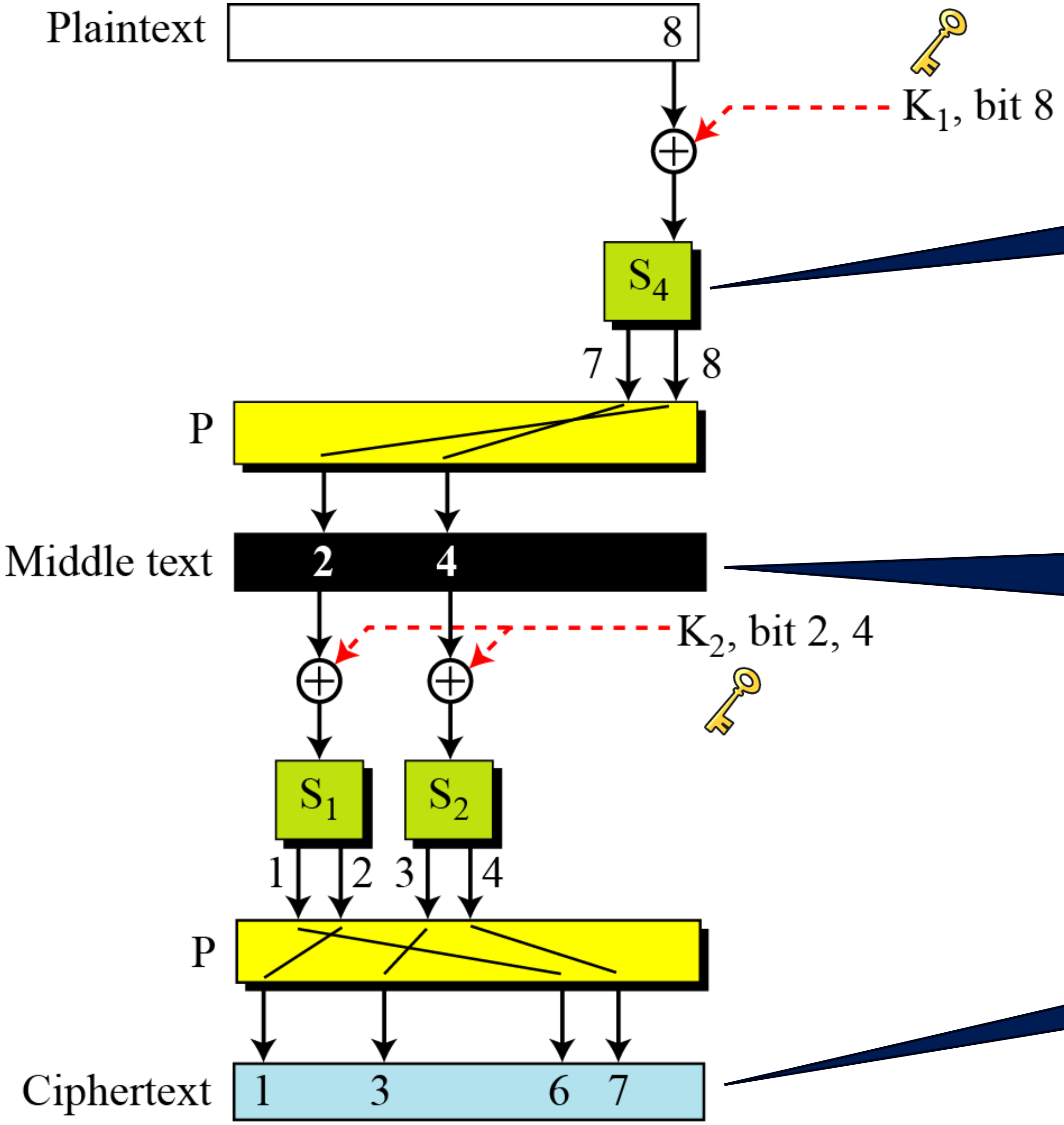


1 × 2 S-box

Diffusion

- P_8 affects 2th and 4th bits after ***round 1***

Example: A Block Cipher made of Two Rounds

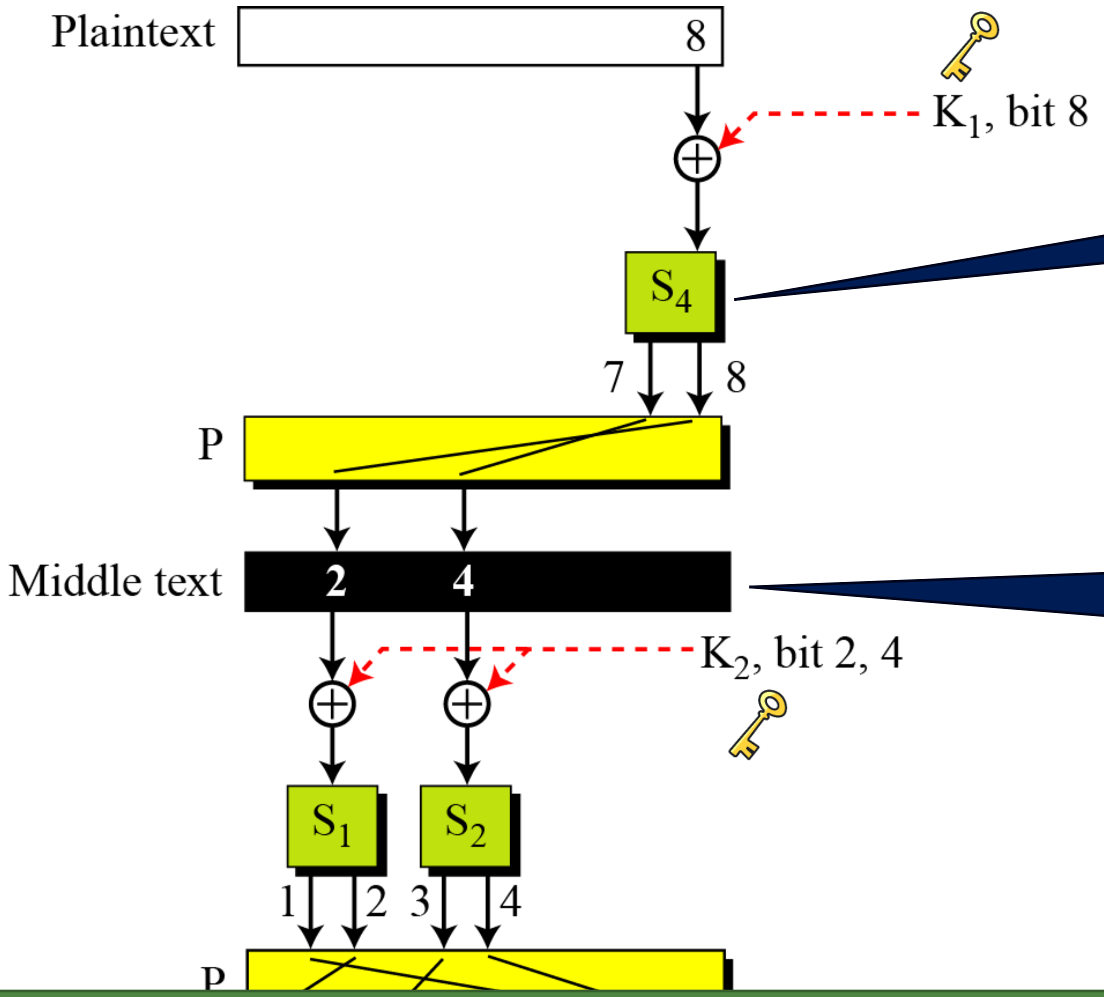


1 × 2 S-box

Diffusion

- P_8 affects 2th and 4th bits after ***round 1***
- P_8 affects 1st, 3rd, 6th, and 7th bits after ***round 2***

Example: A Block Cipher made of Two Rounds



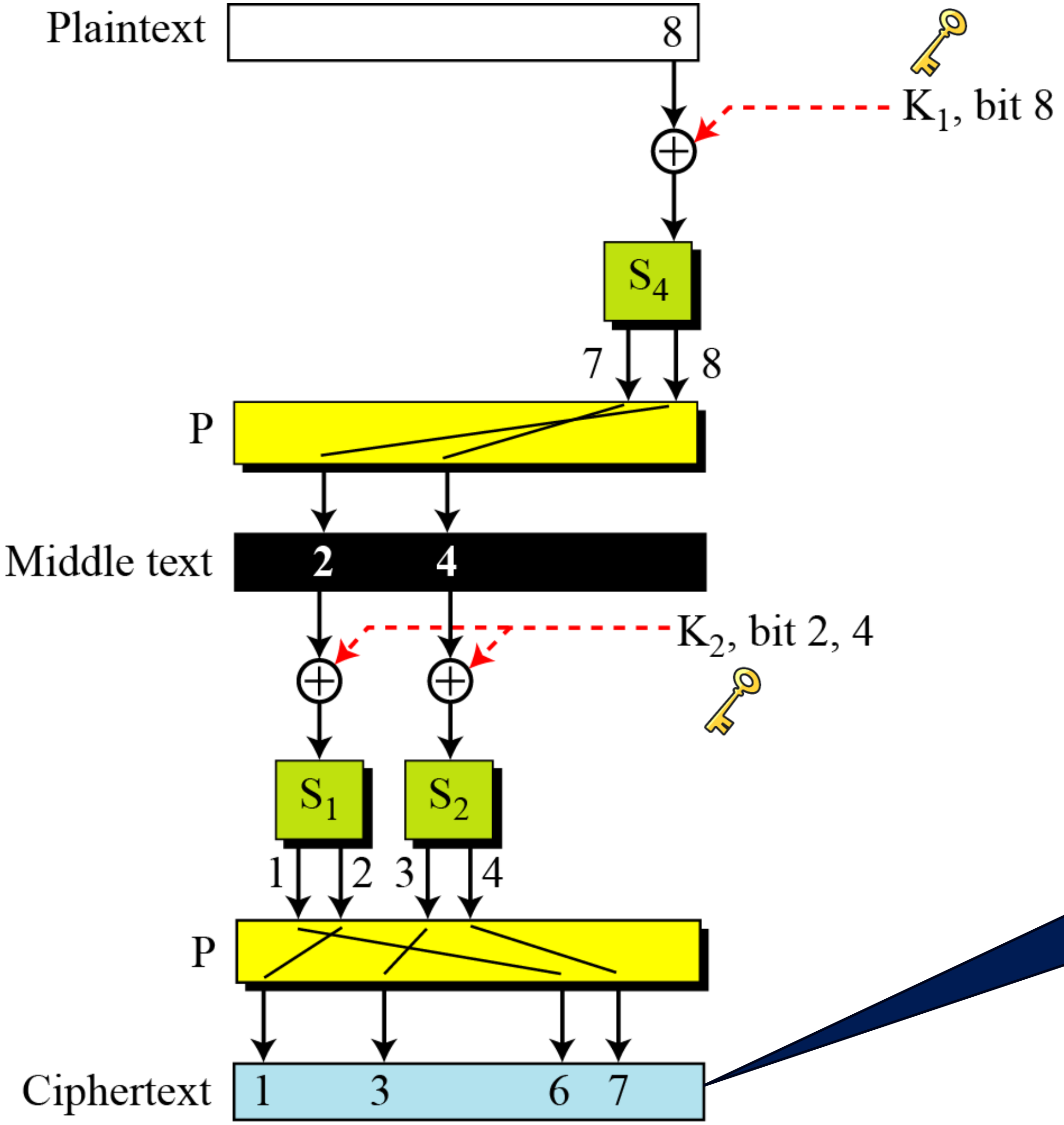
1 × 2 S-box

Diffusion

- P_8 affects 2th and 4th bits after ***round 1***
- P_8 affects 1st, 3rd, 6th, and 7th bits after ***round 2***

If the round length increases, a single bit of plaintext affects all ciphertext

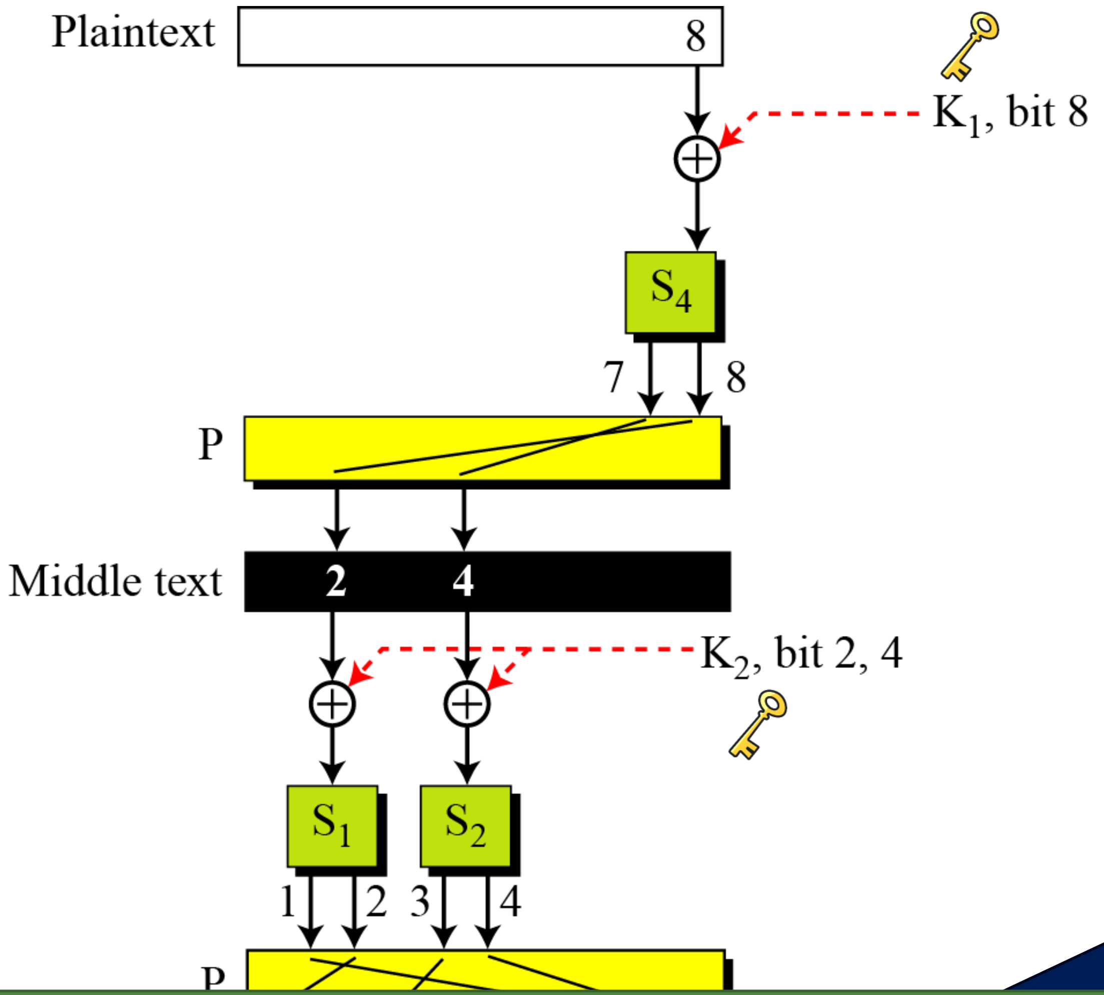
Example: A Block Cipher made of Two Rounds



Confusion

- The 8th bit of K_1 and 2nd and 4th bits of K_2 affect the four bits of the ciphertext

Example: A Block Cipher made of Two Rounds



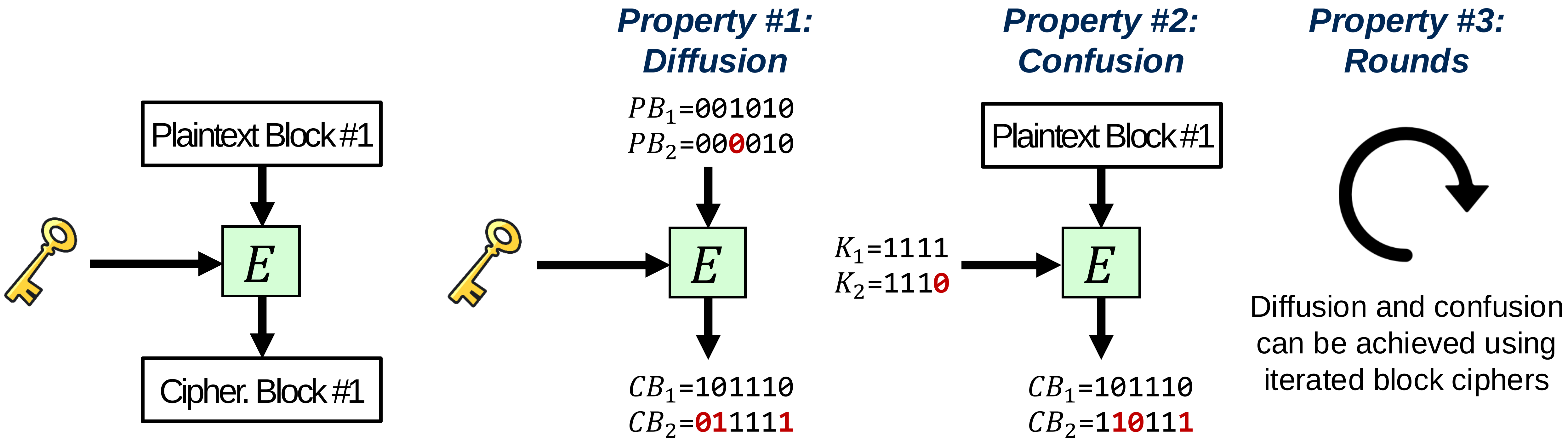
Confusion

- The 8th bit of K_1 and 2nd and 4th bits of K_2 affect the four bits of the ciphertext

If the round length increases, a single bit of key affects all ciphertext

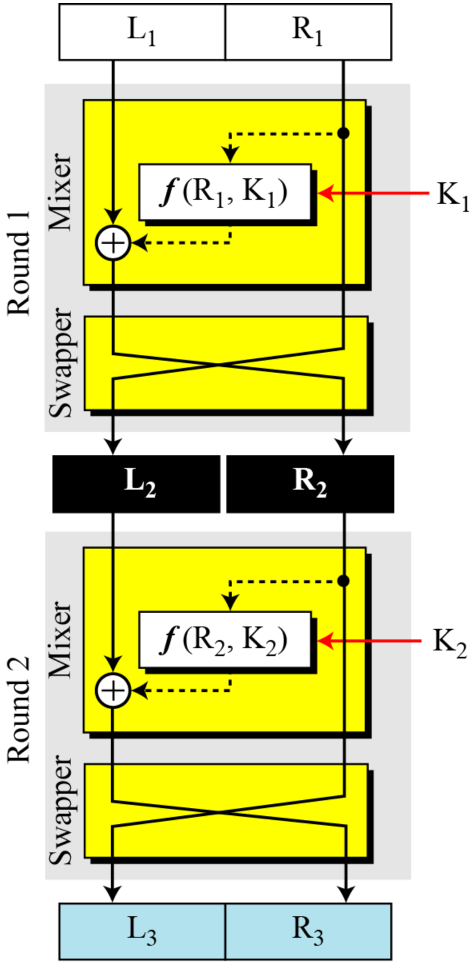
Recap: Design Principles for Block Ciphers

- A modern block ciphers are usually made of combination of
  - Transposition cipher
  - Substitution cipher
  - Some other units

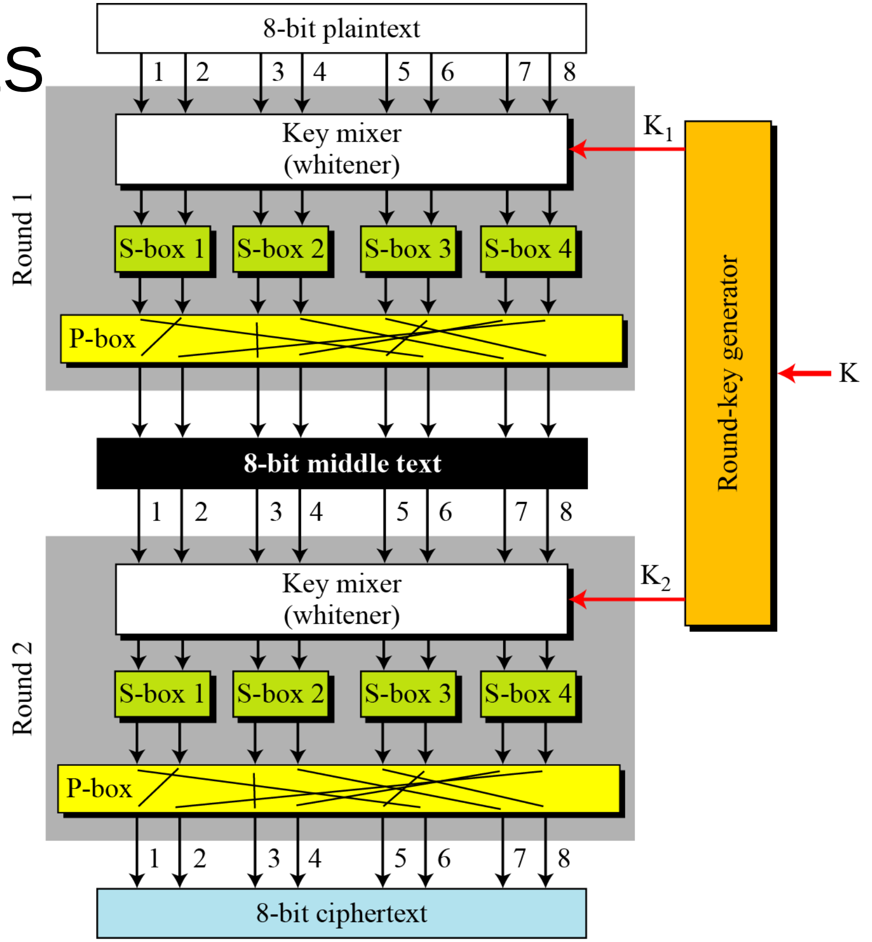


Two Classes of Block Ciphers

- **Feistel ciphers**
  - E.g., DES

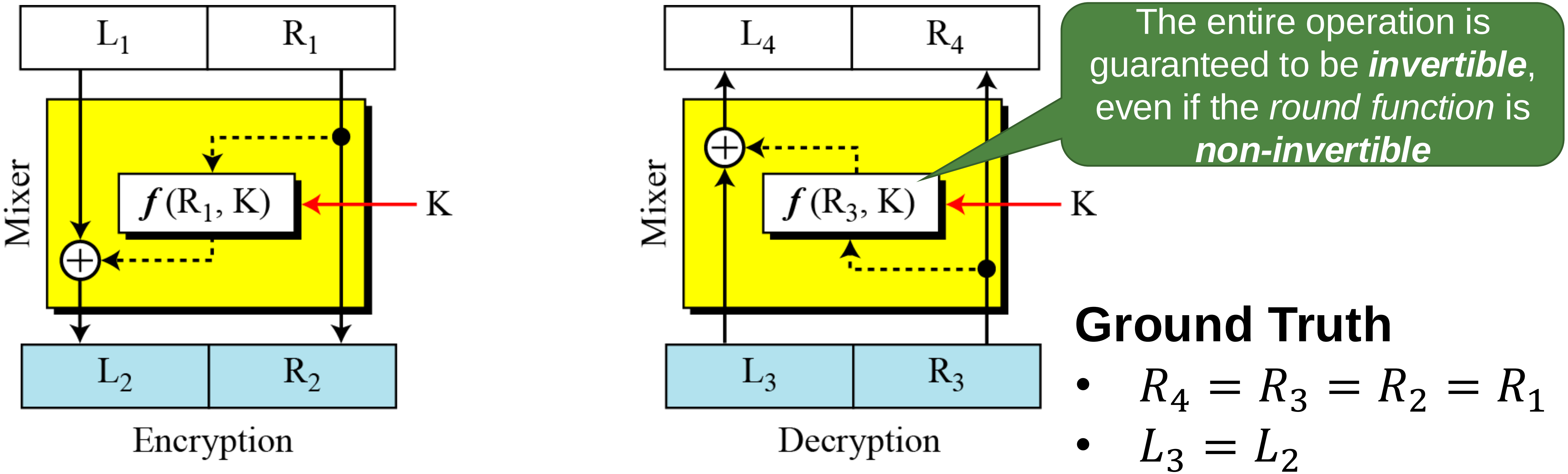


- **Substitution-permutation (SP) ciphers**
  - E.g., AES



Feistel Cipher – Basic Components

- Use the **same algorithm** for both *encryption* and *decryption*



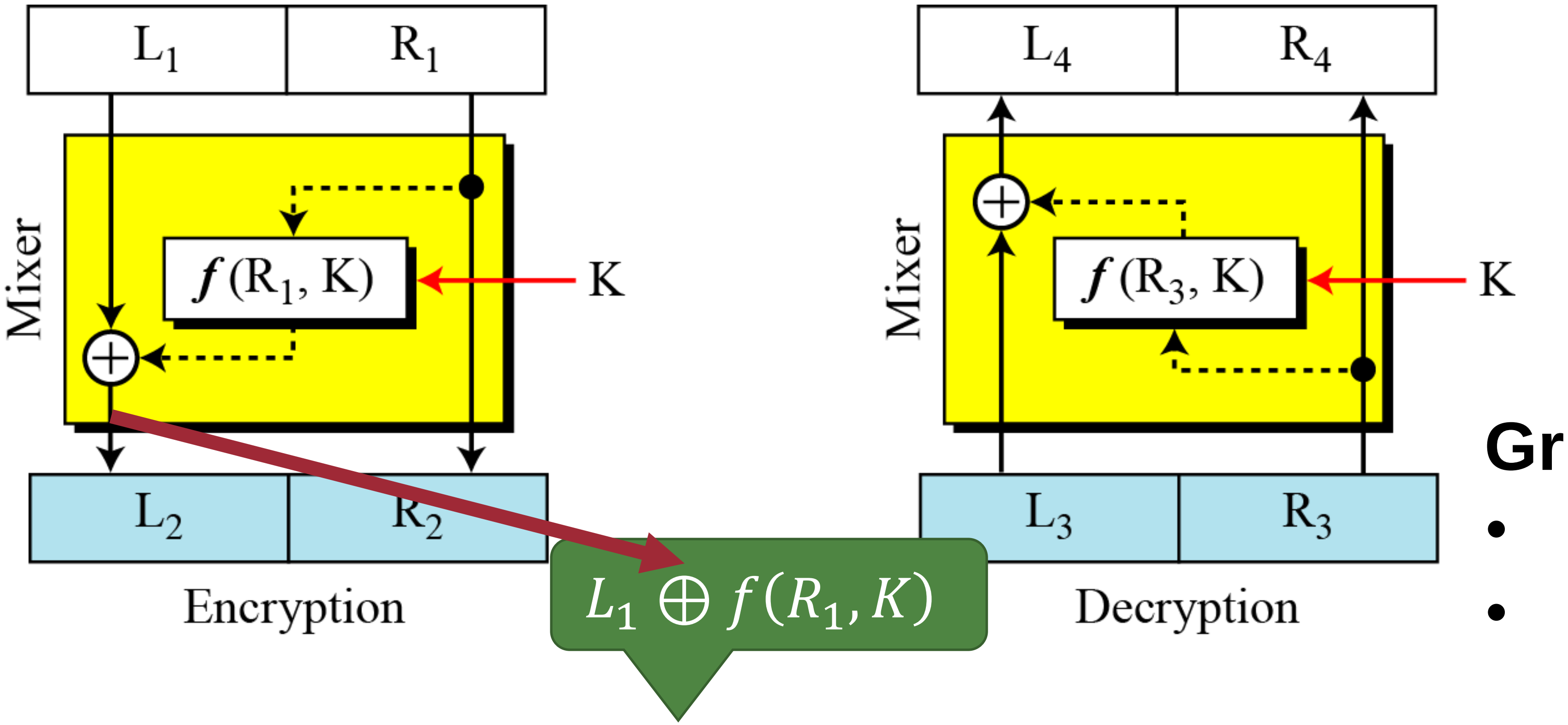
Ground Truth

- $R_4 = R_3 = R_2 = R_1$
- $L_3 = L_2$

$$L_4 = L_3 \oplus f(R_3, K)$$

Feistel Cipher – Basic Components

- Use the **same algorithm** for both *encryption* and *decryption*



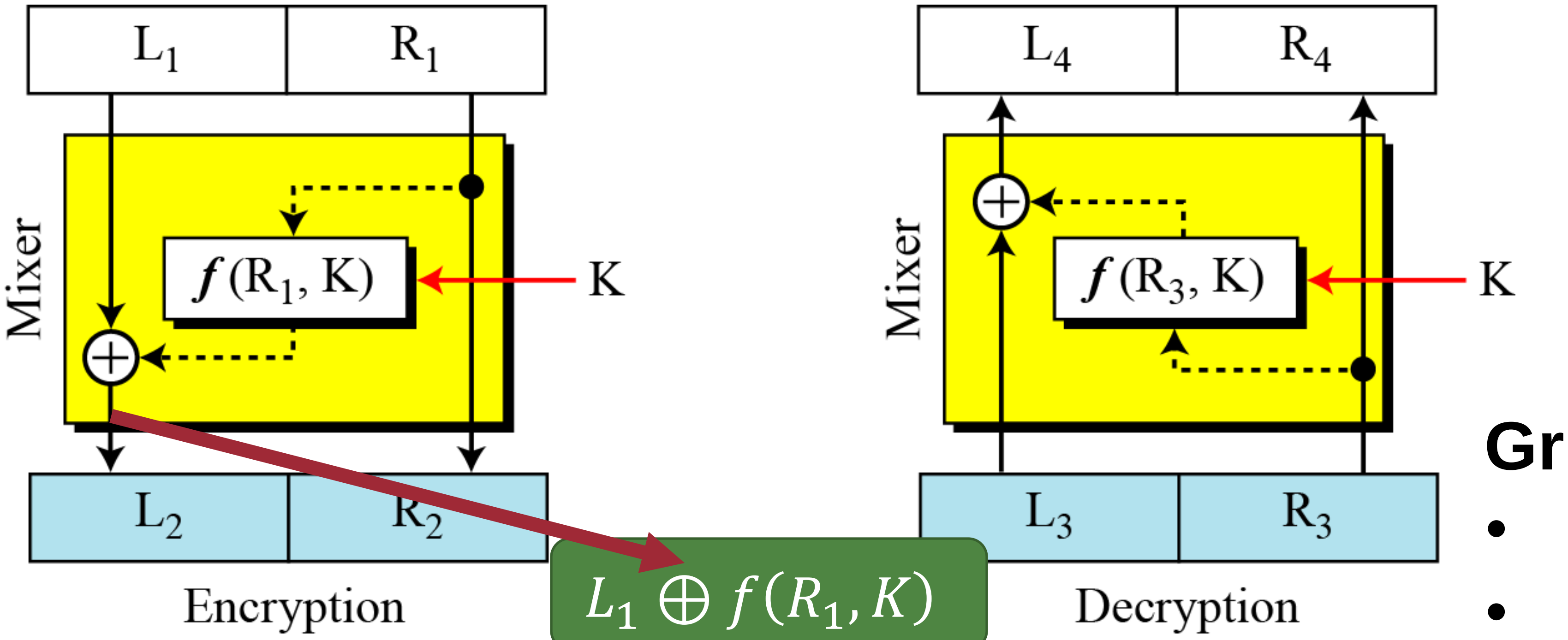
$$L_4 = L_3 \oplus f(R_3, K) = L_2 \oplus f(R_1, K)$$

Ground Truth

- $R_4 = R_3 = R_2 = R_1$
- $L_3 = L_2$

Feistel Cipher – Basic Components

- Use the **same algorithm** for both *encryption* and *decryption*



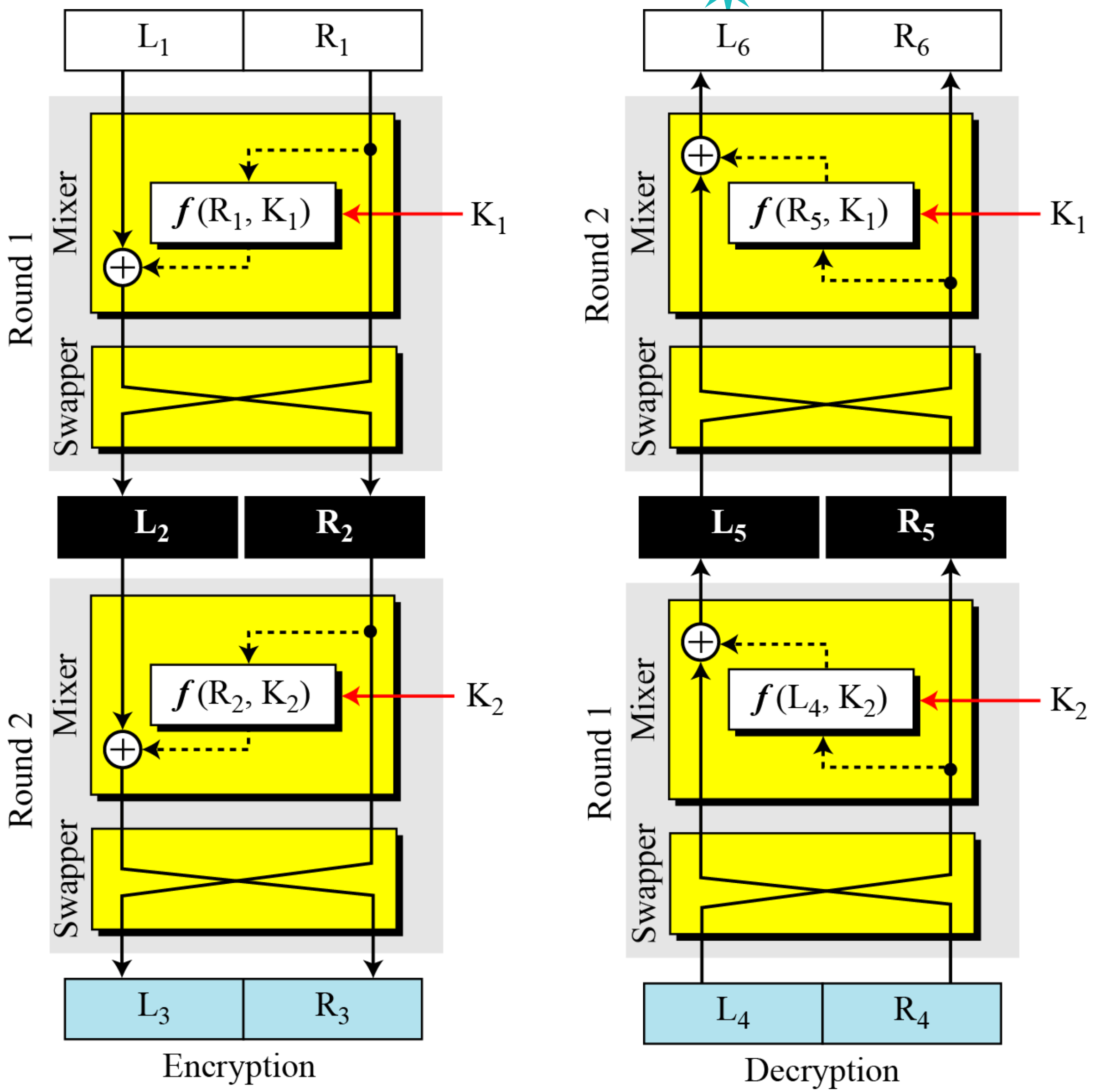
$L_1 \oplus f(R_1, K)$

Ground Truth

- $R_4 = R_3 = R_2 = R_1$
- $L_3 = L_2$

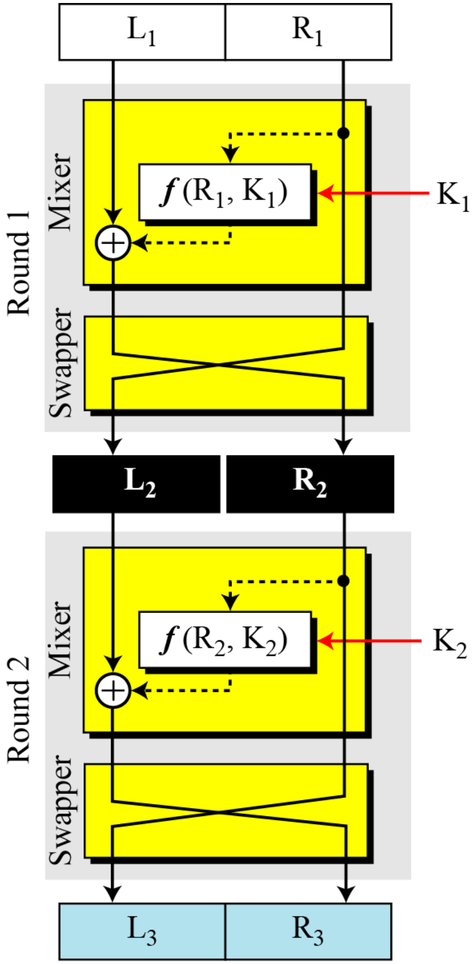
$$L_4 = L_3 \oplus f(R_3, K) = L_2 \oplus f(R_1, K)$$
$$= L_1 \oplus f(R_1, K) \oplus f(R_1, K) = L_1 \oplus 0 = L_1$$

A Feistel Cipher with Two Rounds

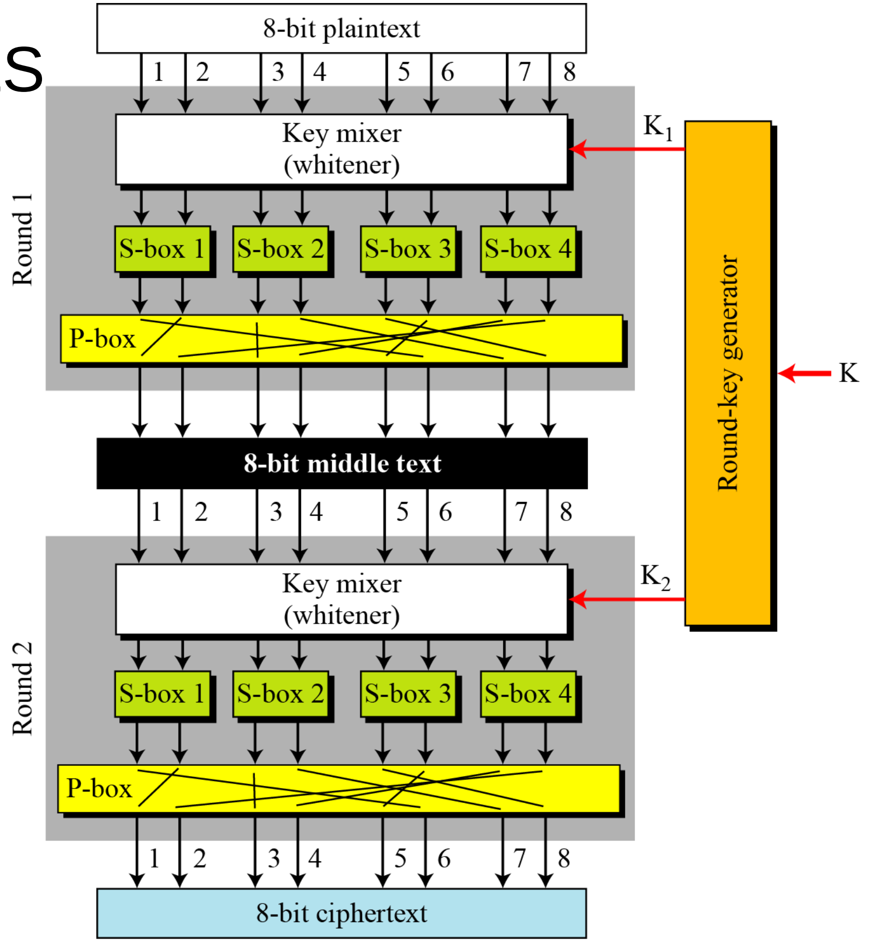


Pros and Cons?

- **Feistel ciphers**
  - E.g., DES



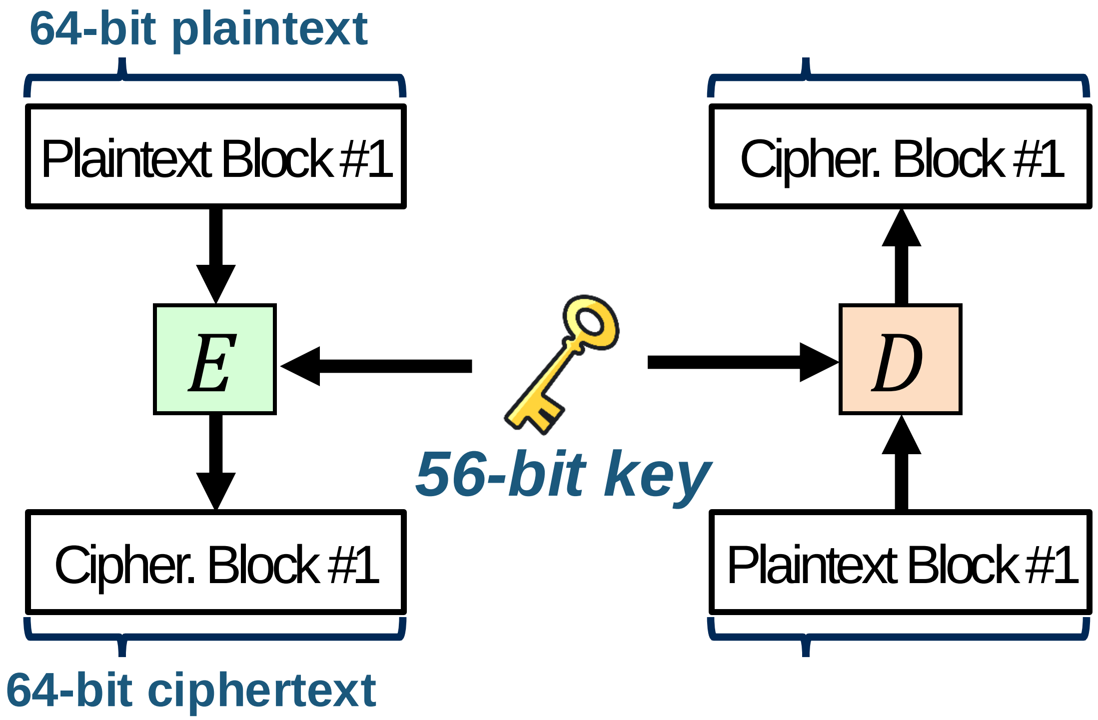
- **Substitution-permutation (SP) ciphers**
  - E.g., AES



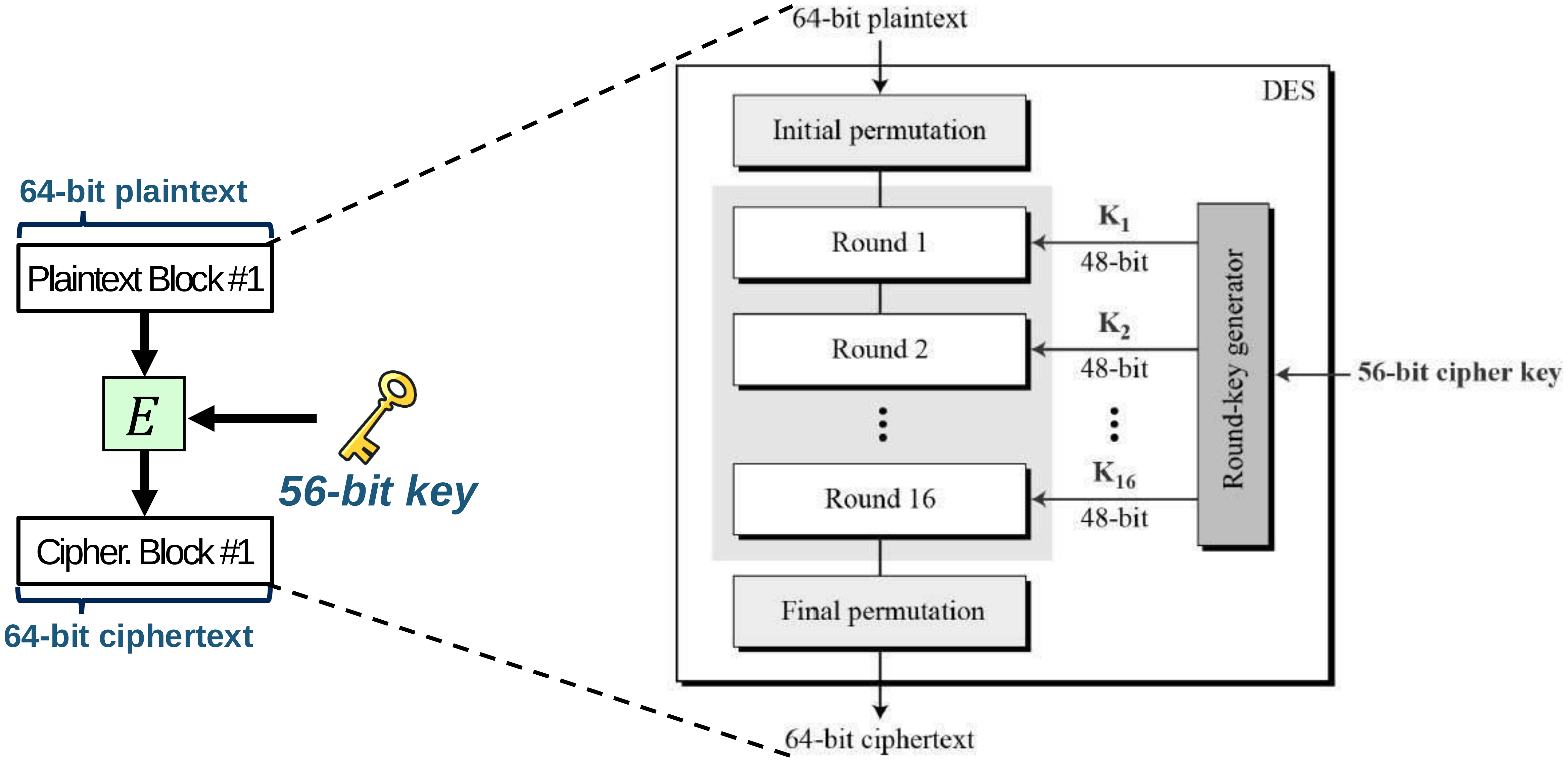
Data Encryption Standard (DES)

Data Encryption Standard (DES)

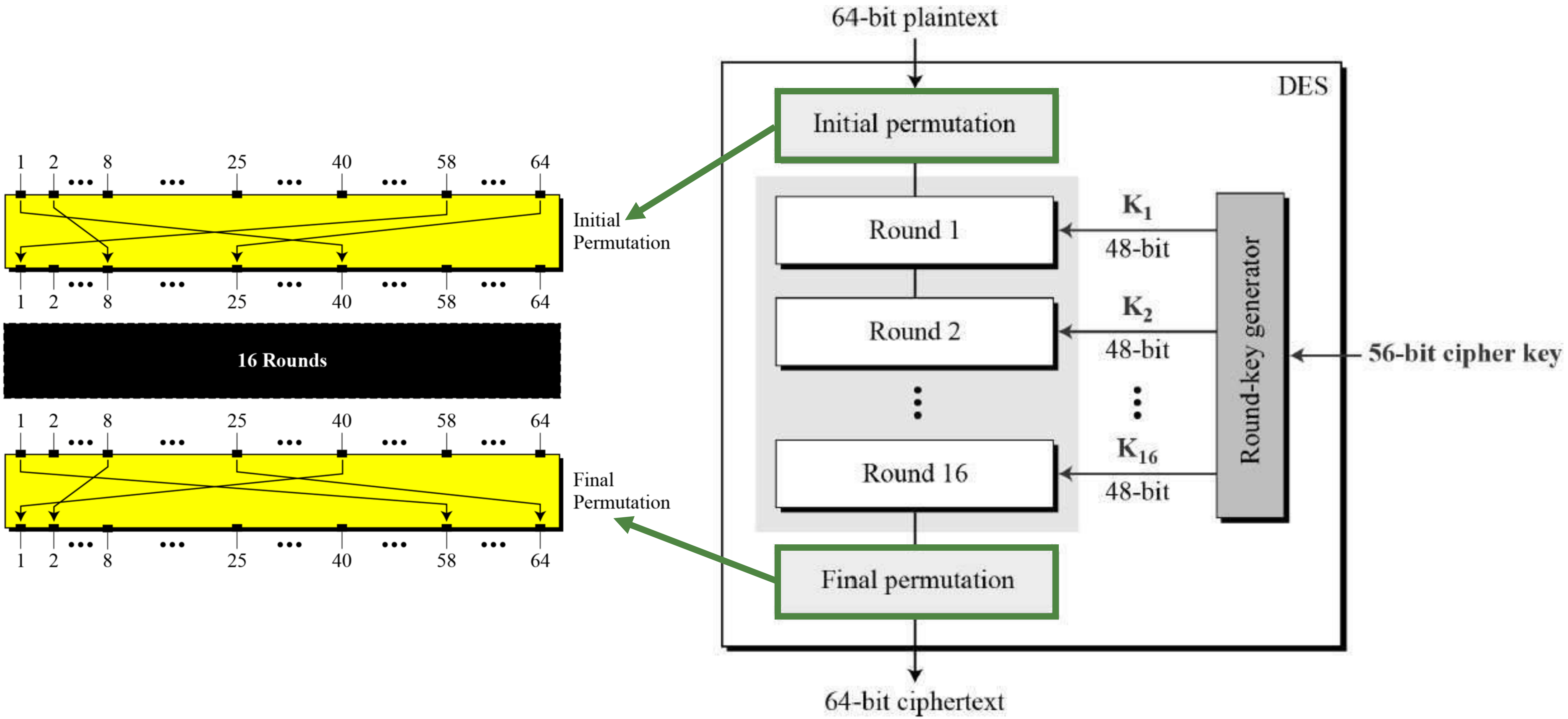
- Adopted in 1977 by NBS (now NIST)
- Encrypts 64 bit data using 56 bit key
- Has widespread use
- **Due to its short key length**, it is used until 1999, and replaced by AES



General Structure of DES



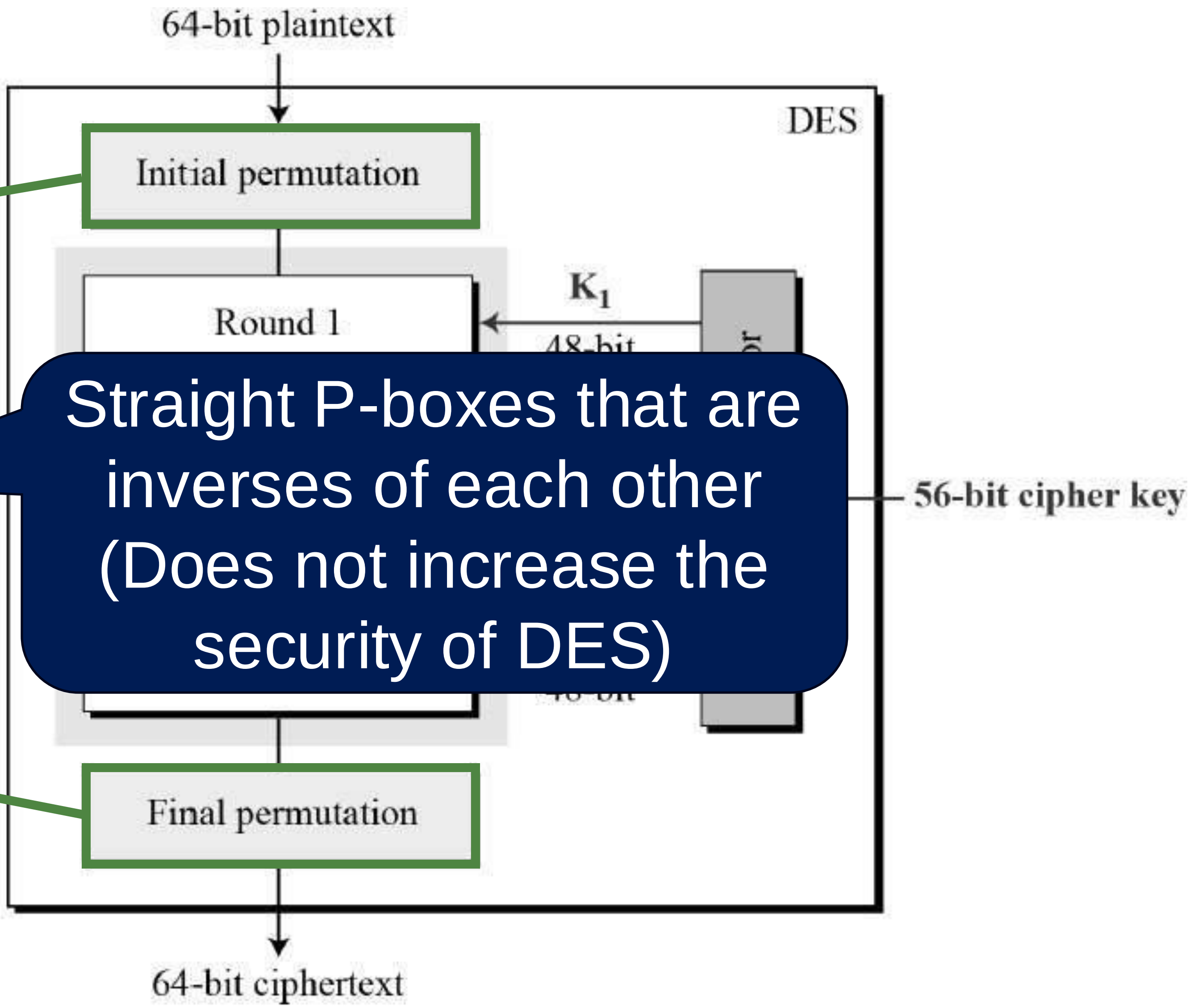
Initial and Final Permutations (P-Boxes)



Initial and Final Permutations (P-Boxes)

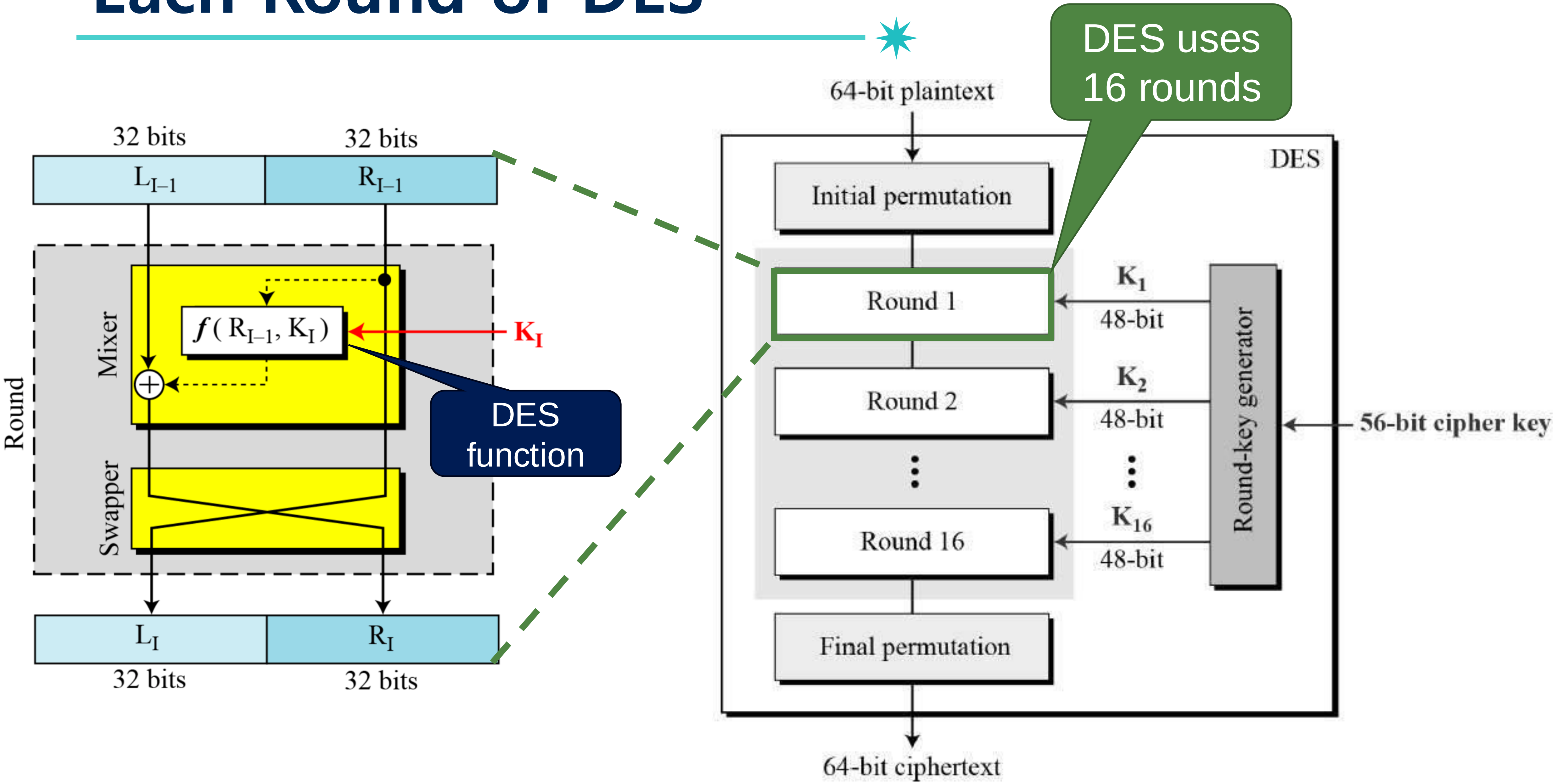
Initial Permutation							
58	50	42	34	26	18	10	02
60	52	44	36	28	20	12	04
62	54	46	38	30	22	14	06
64	56	48	40	32	24	16	08
57	49	41	33	25	17	09	01
59	51	43	35	27	19	11	03
61	53	45	37	29	21	13	05
63	55	47	39	31	23	15	07

Final Permutation							
40	08	48	16	56	24	64	32
39	07	47	15	55	23	63	31
38	06	46	14	54	22	62	30
37	05	45	13	53	21	61	29
36	04	44	12	52	20	60	28
35	03	43	11	51	19	59	27
34	02	42	10	50	18	58	26
33	01	41	09	49	17	57	25

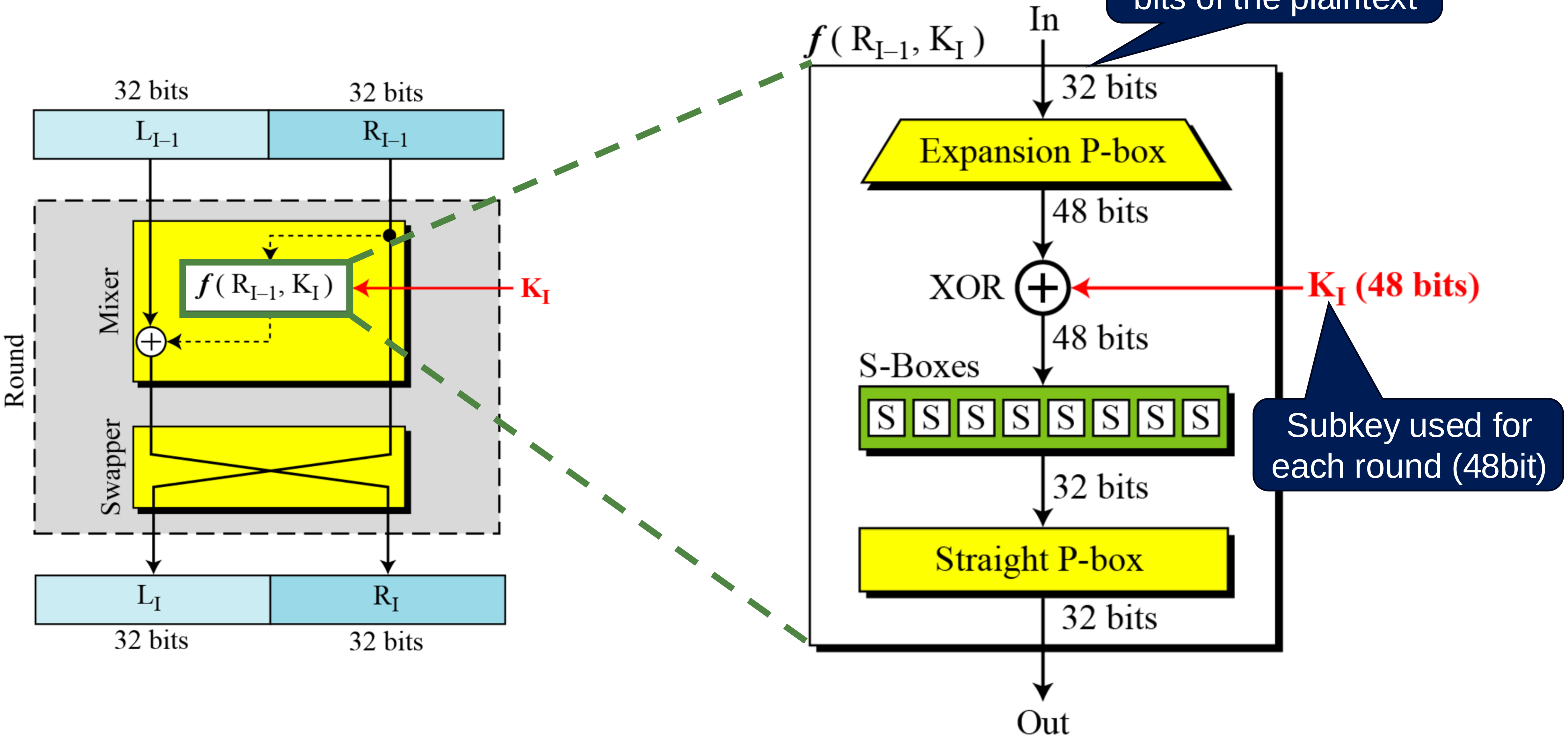


Straight P-boxes that are inverses of each other (Does not increase the security of DES)

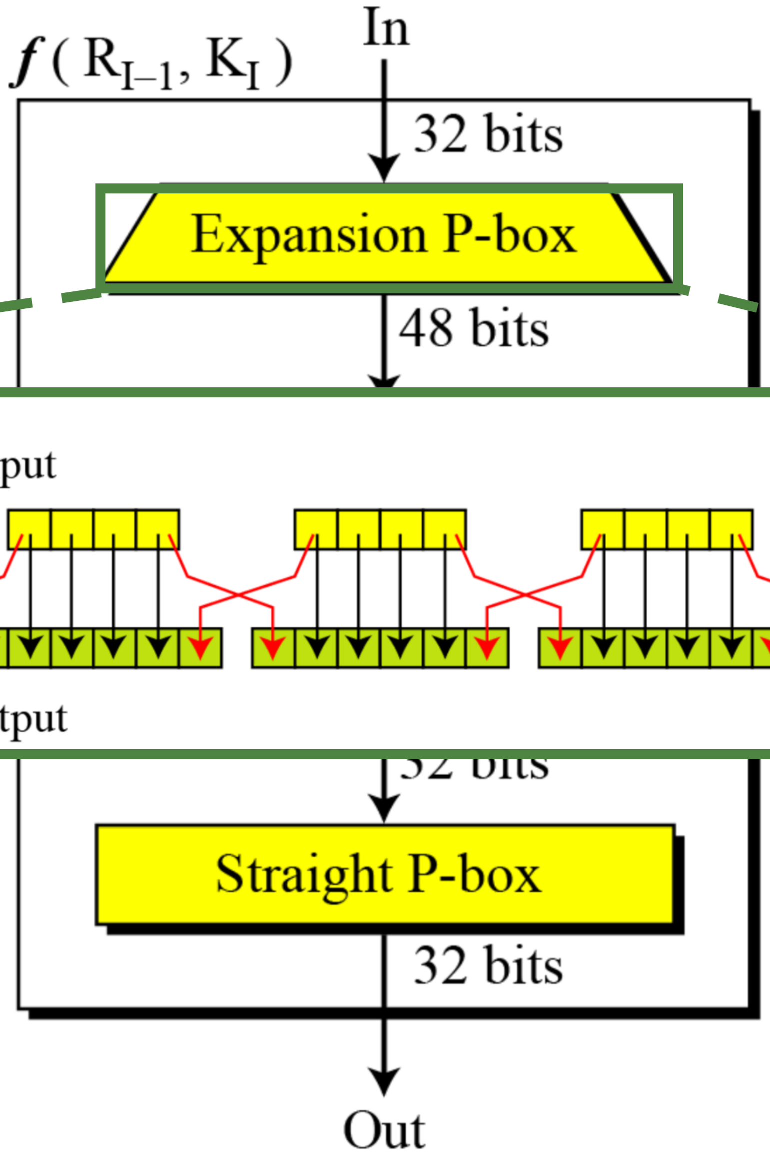
Each Round of DES



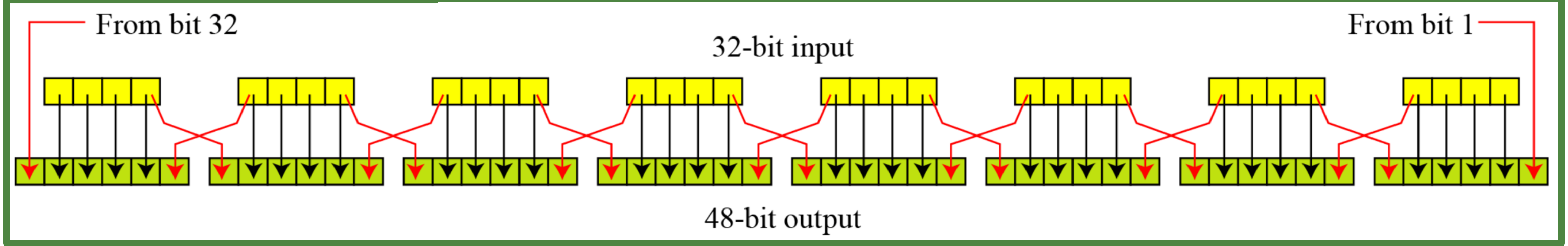
DES Function



DES Function - Expansion P-box

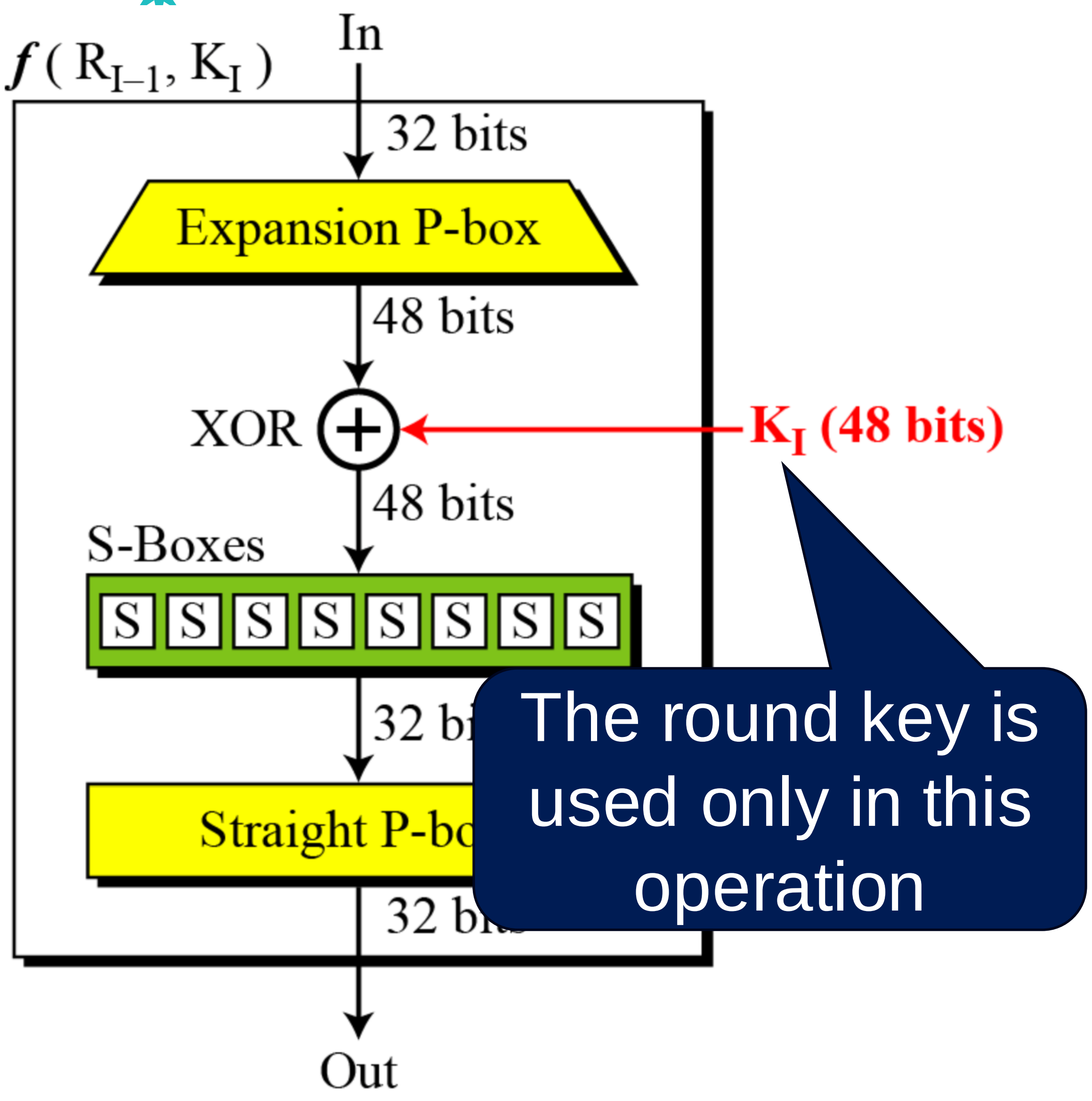


32 X 48 Expansion P-box

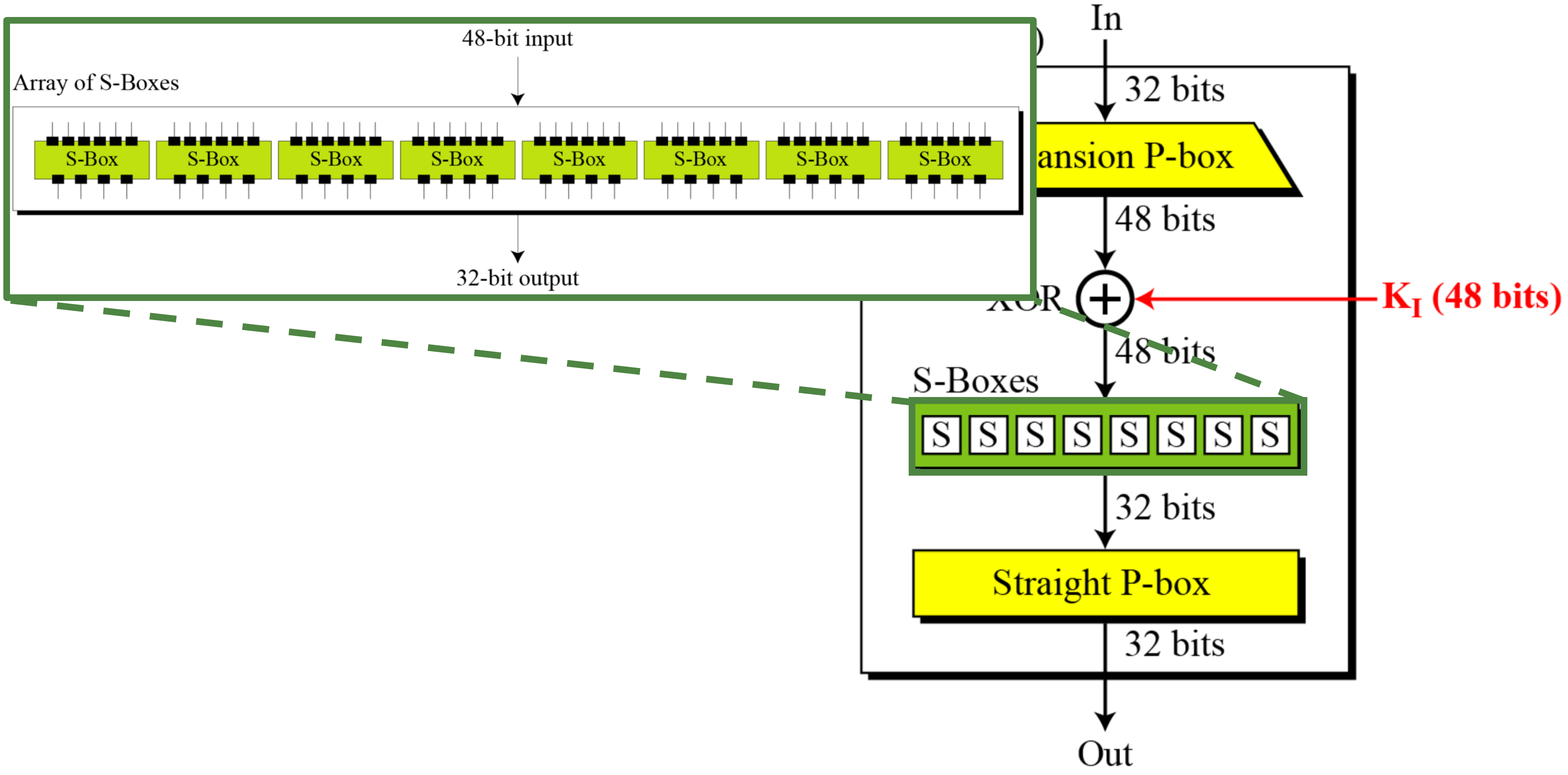


32	01	02	03	04	05
04	05	06	07	08	09
08	09	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	31	31	32	01

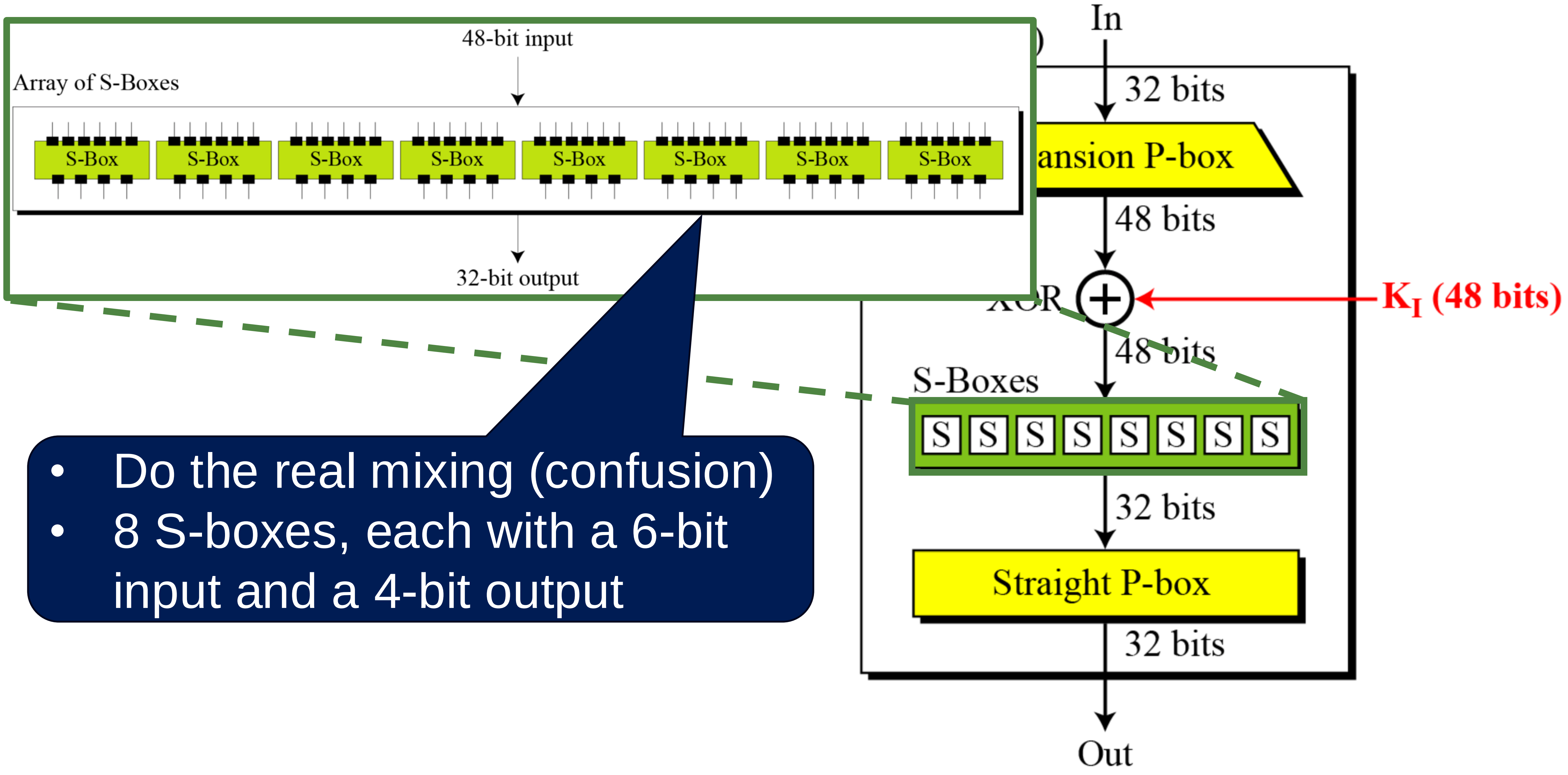
DES Function - XOR



DES Function - S-Boxes



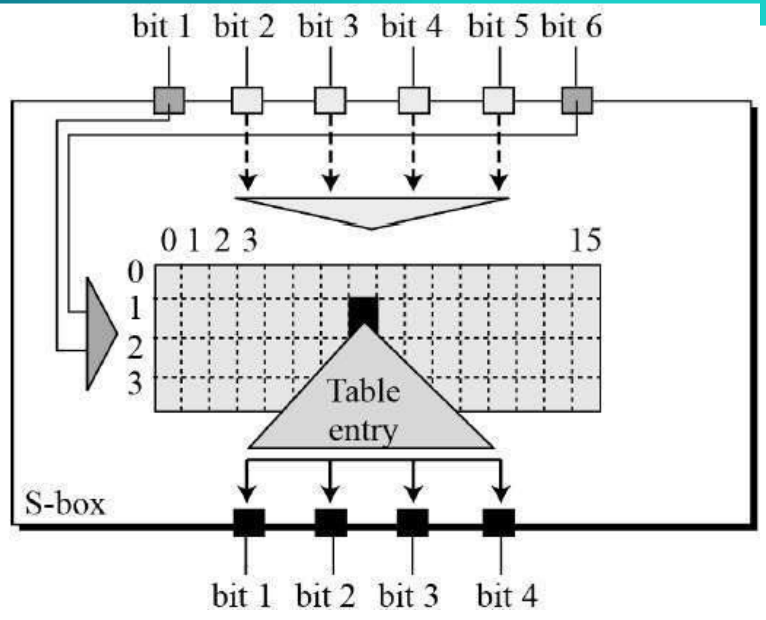
DES Function - S-Boxes



- Do the real mixing (confusion)
- 8 S-boxes, each with a 6-bit input and a 4-bit output

DES Function - S-Boxes

- Non-invertible S-Boxes
- For the rest of the boxes, see the next slides



100011

S-box 1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	10	03	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

1100

S_1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S_5	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

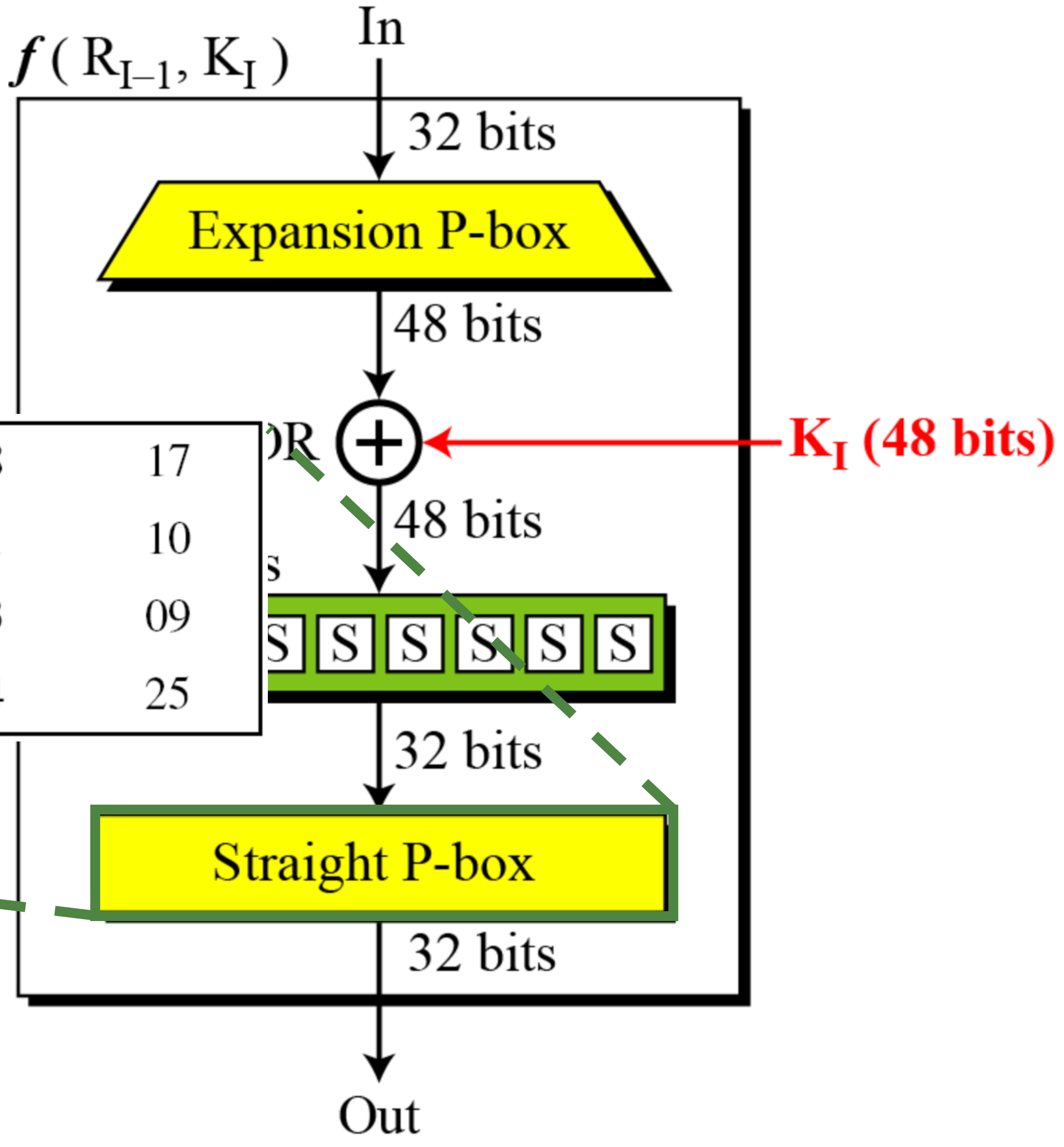
S_6	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S_7	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

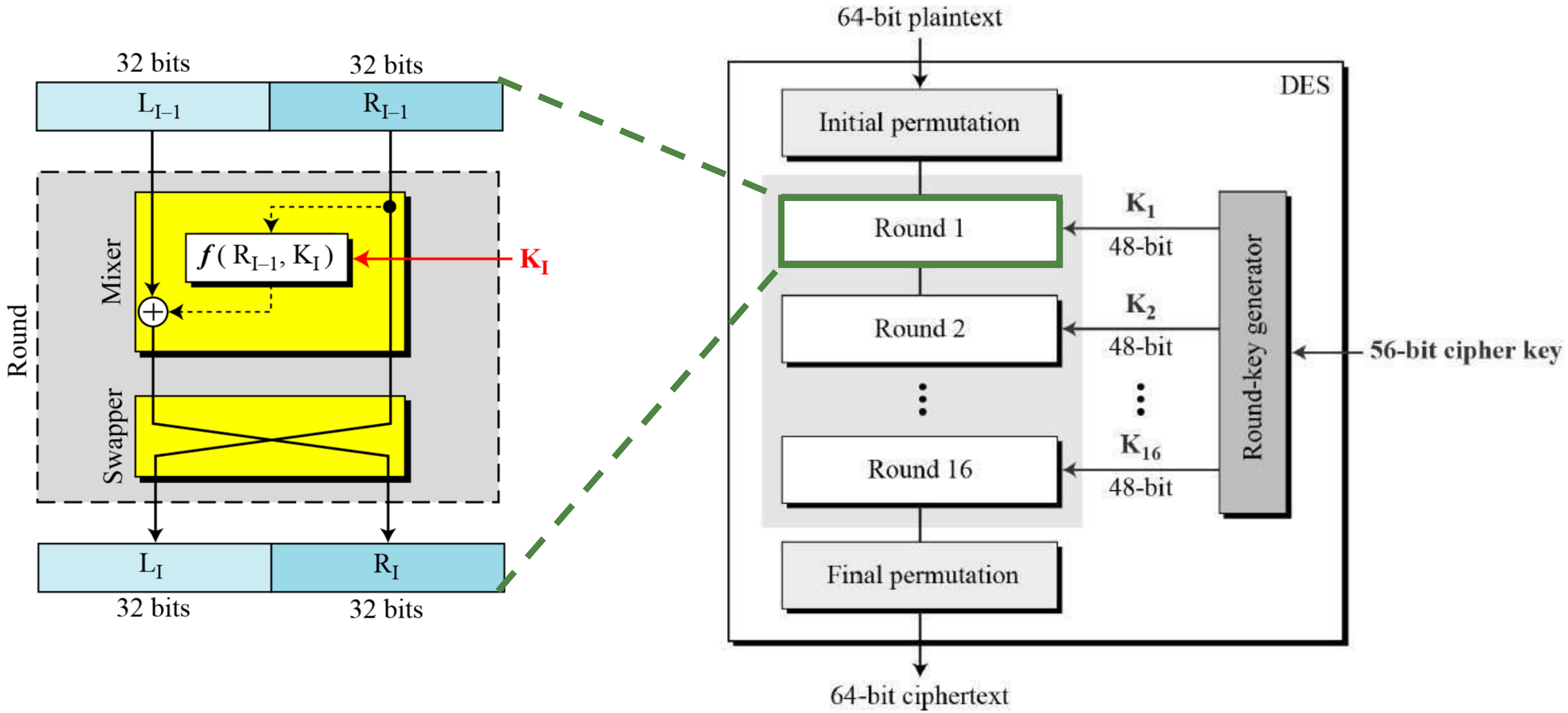
S_8	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

DES Function - Straight P-Box

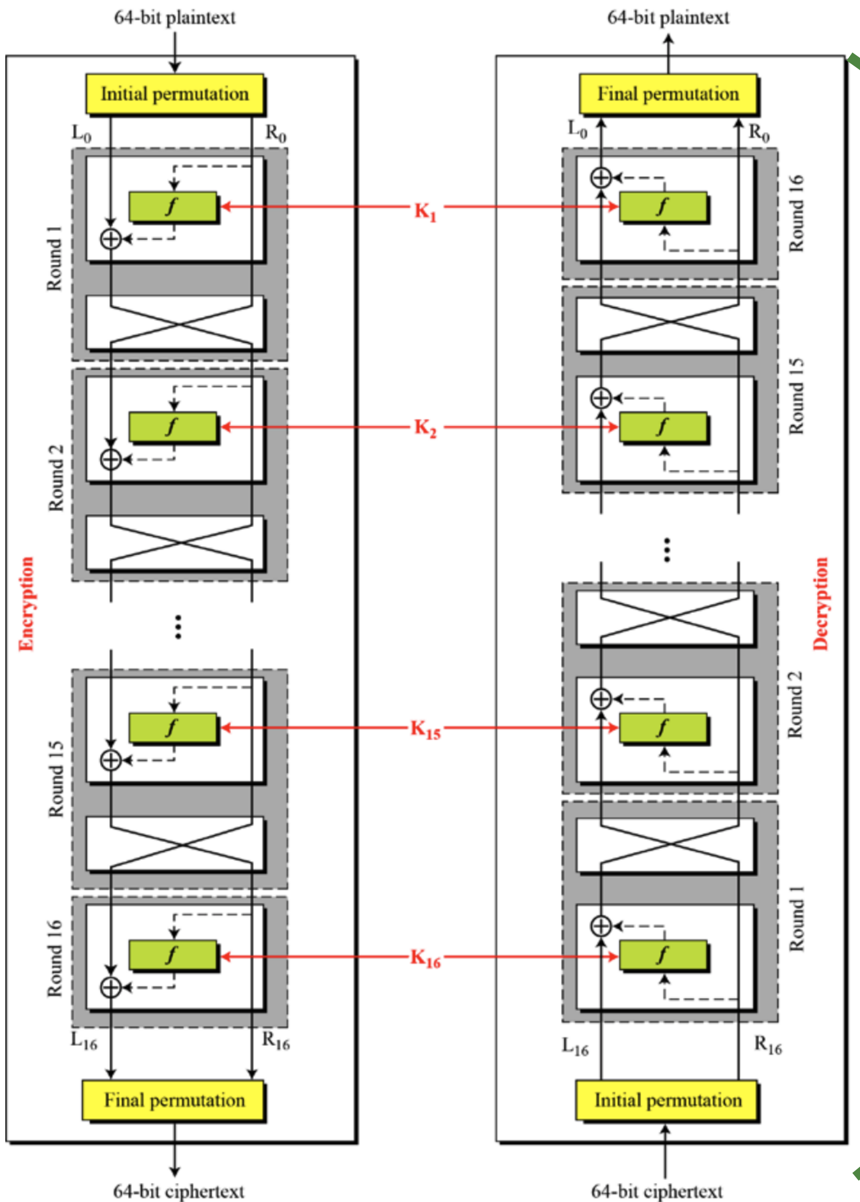
16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

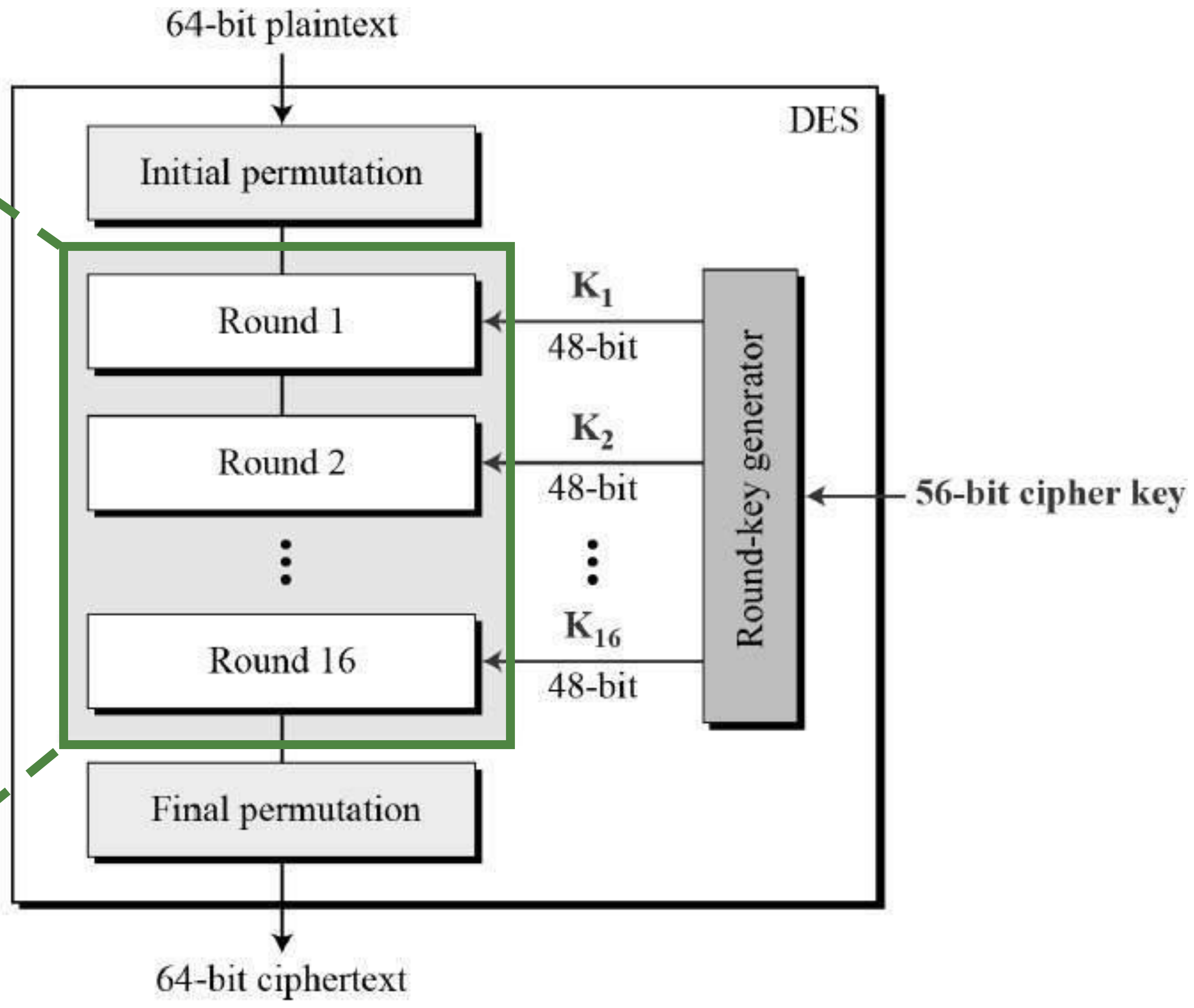


Round of DES

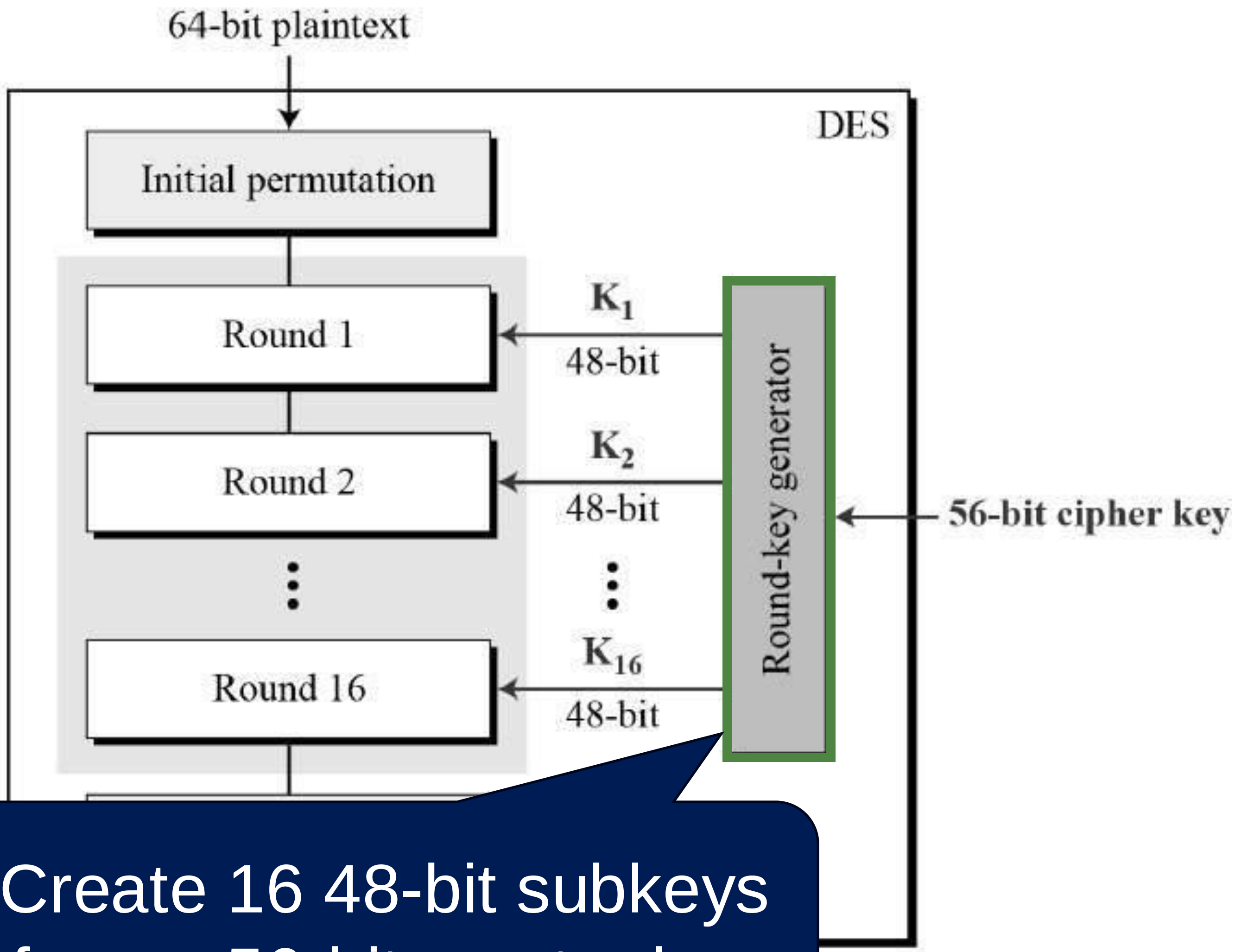

32 bits
32 bits
L_{I-1}
R_{I-1}
Round
Mixer
$f\,(R_{I-1}, K_I)$
K_I
Swapper
L_I
R_I
32 bits
32 bits
64-bit plaintext
DES
Initial permutation
Round 1
K_1
48-bit
Round 2
K_2
48-bit
Round 16
K_{16}
48-bit
Round-key generator
56-bit cipher key
Final permutation
64-bit ciphertext

Round Structure of DES



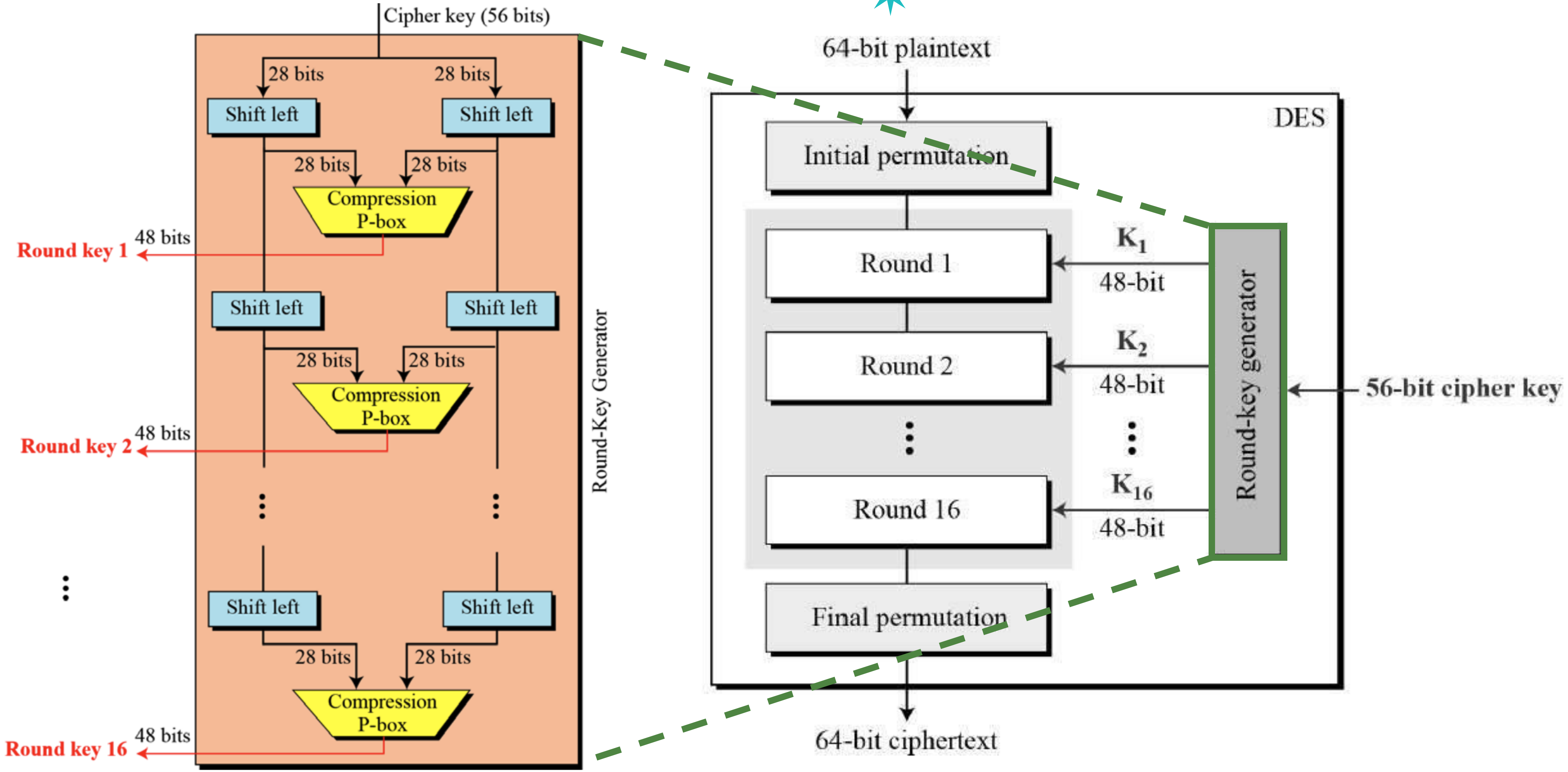


Key Generation in DES

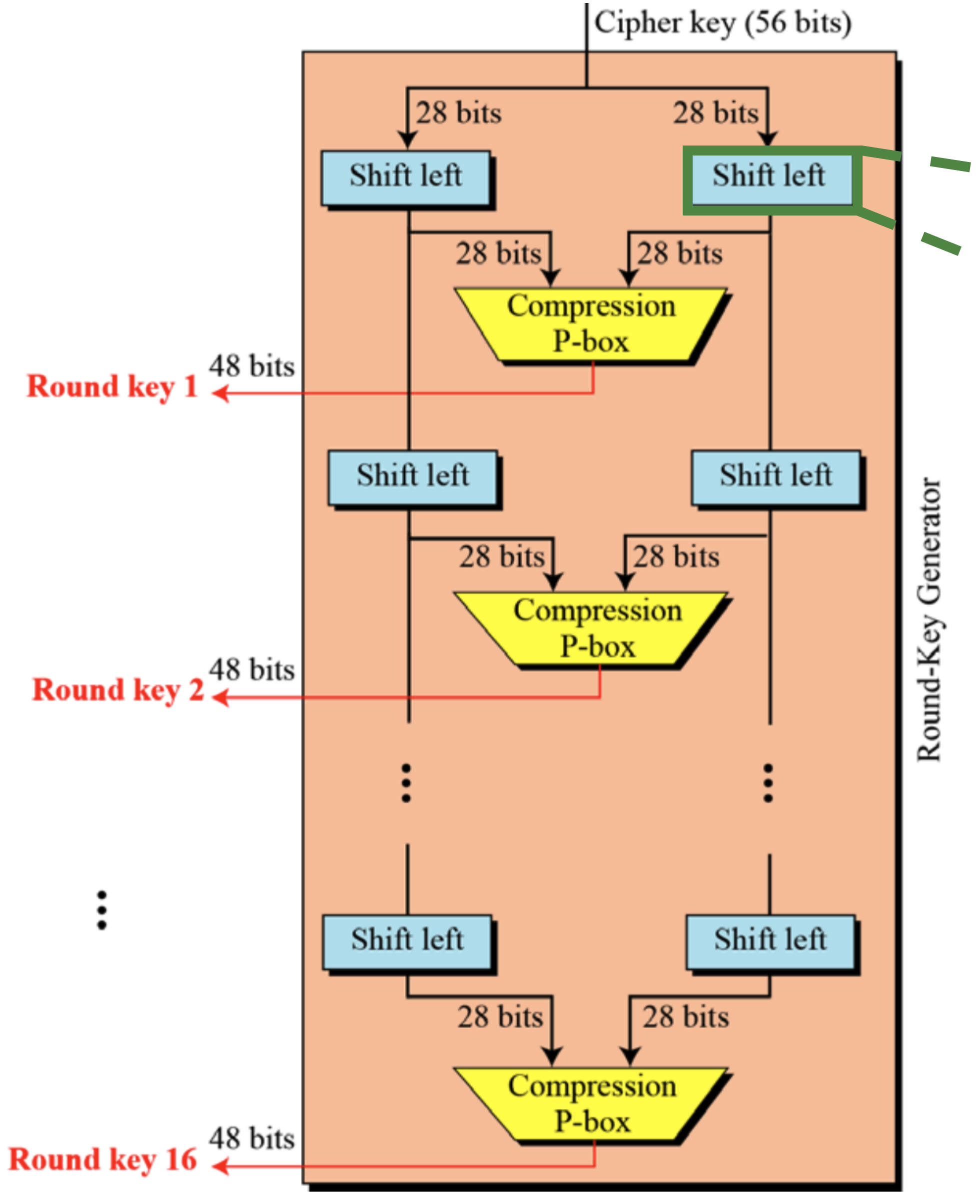


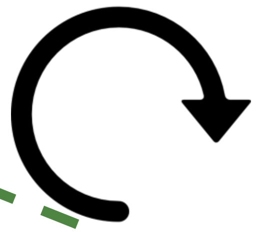
Create 16 48-bit subkeys from a 56-bit master key

Key Generation in DES



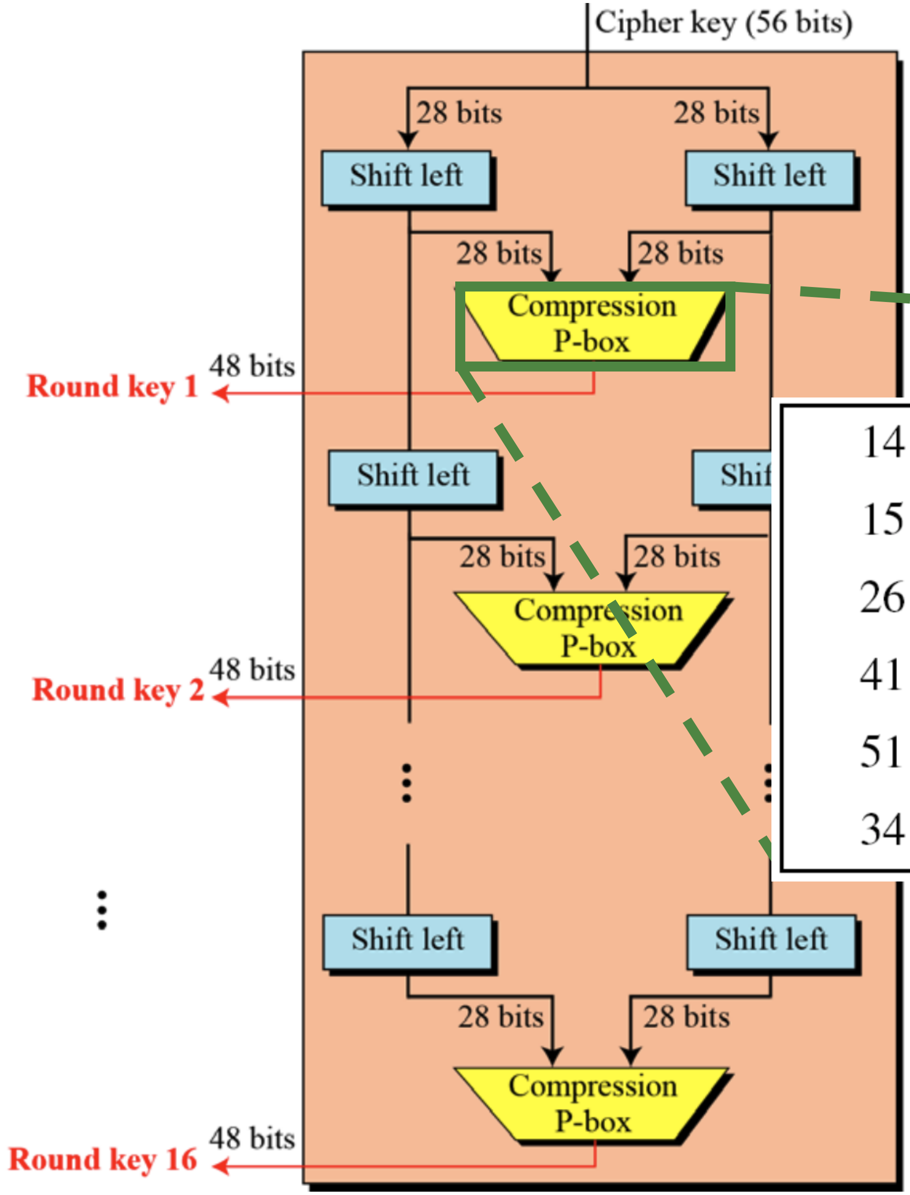
Key Generation - Shift Left





- **In rounds 1, 2, 9, 16**: shift left by one bit
- **The other rounds**: shift left by two bits

Key Generation - Compression P-box



14	17	11	24	01	05	03	28
15	06	21	10	23	19	12	04
26	08	16	07	27	20	13	02
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

Example of DES

- Determine what the ciphertext block would be (all in hexadecimal)
- Plaintext: 123456ABCD132536
- Key: AABB09182736CCDD

Plaintext: 123456ABCD132536			
After initial permutation: 14A7D67818CA18AD After splitting: L_0=14A7D678 R_0=18CA18AD			
Round	*Left*	*Right*	*Round Key*
Round 1	18CA18AD	5A78E394	194CD072DE8C
Round 2	5A78E394	4A1210F6	4568581ABCCE
Round 3	4A1210F6	B8089591	06EDA4ACF5B5
Round 4	B8089591	236779C2	DA2D032B6EE3

Example of DES

Round 5	236779C2	A15A4B87	69A629FEC913
Round 6	A15A4B87	2E8F9C65	C1948E87475E
Round 7	2E8F9C65	A9FC20A3	708AD2DDB3C0
Round 8	A9FC20A3	308BEE97	34F822F0C66D
Round 9	308BEE97	10AF9D37	84BB4473DCCC
Round 10	10AF9D37	6CA6CB20	02765708B5BF
Round 11	6CA6CB20	FF3C485F	6D5560AF7CA5
Round 12	FF3C485F	22A5963B	C2C1E96A4BF3
Round 13	22A5963B	387CCDAA	99C31397C91F
Round 14	387CCDAA	BD2DD2AB	251B8BC717D0
Round 15	BD2DD2AB	CF26B472	3330C5D9A36D
Round 16	19BA9212	CF26B472	181C5D75C66D
After combination: 19BA9212CF26B472			
Ciphertext: C0B7A8D05F3A829C			*(after final permutation)*

Example of DES

- See how Bob can decipher the ciphertext using the same key

Ciphertext: C0B7A8D05F3A829C			
After initial permutation: 19BA9212CF26B472 After splitting: L_0=19BA9212 R_0=CF26B472			
Round	*Left*	*Right*	*Round Key*
Round 1	CF26B472	BD2DD2AB	181C5D75C66D
Round 2	BD2DD2AB	387CCDAA	3330C5D9A36D
. . .	. . .	. . .	. . .
Round 15	5A78E394	18CA18AD	4568581ABCCE
Round 16	14A7D678	18CA18AD	194CD072DE8C
After combination: 14A7D67818CA18AD			
Plaintext:123456ABCD132536 (after final permutation)			

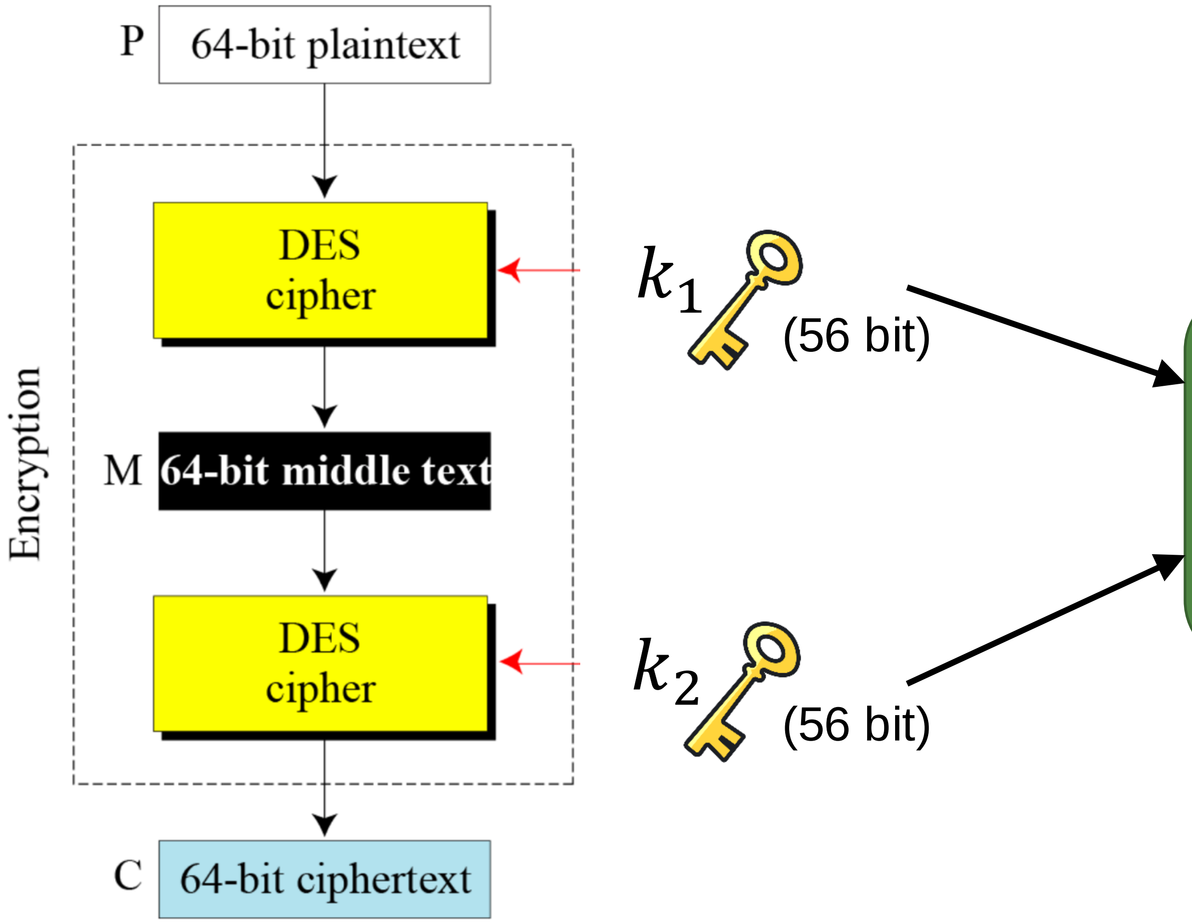
DES Weakness

- The most serious weakness of DES is in its key size
- The adversary needs to check 2^{56} keys
  - If we make one million processor computer, it will take about 20 hours
  - A special computer was built in 1998 that found the key in 112 hours

How do we make it robust?
=> *Let's use double-DES on each block*

Double-DES?

- $C = E_{K2}(E_{K1}(PB))$

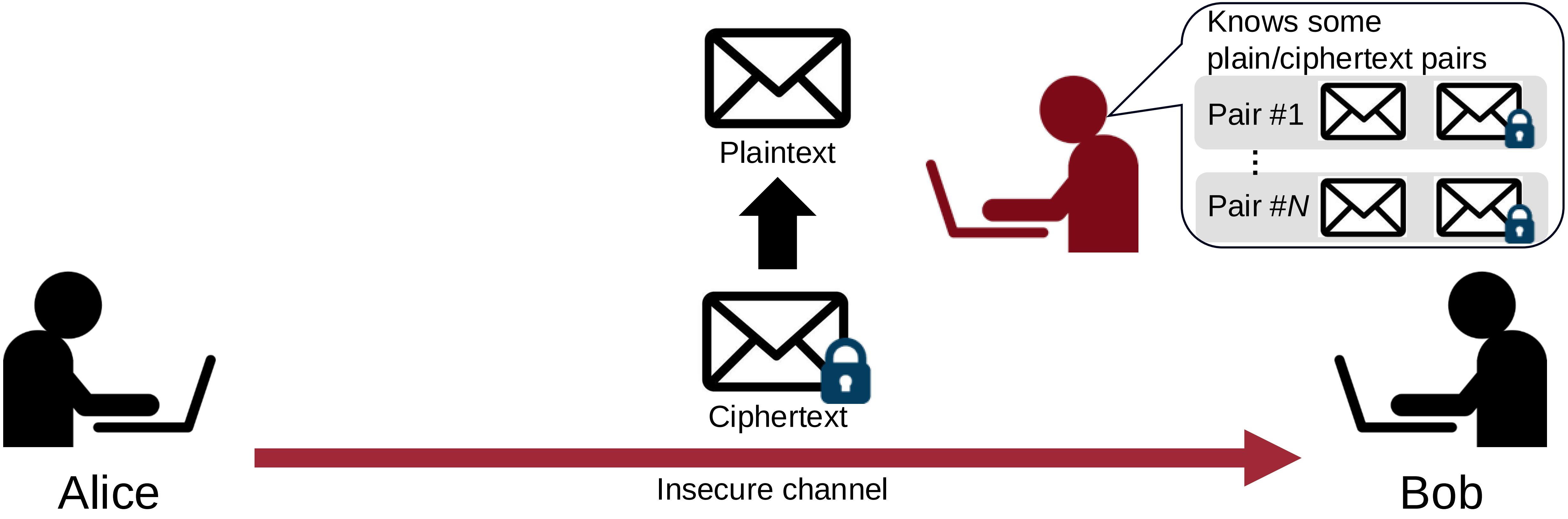


The adversary needs to check 2^{112} keys! What is the problem?

Let's assume that the adversary conduct Known-Plaintext Attack (KPA)

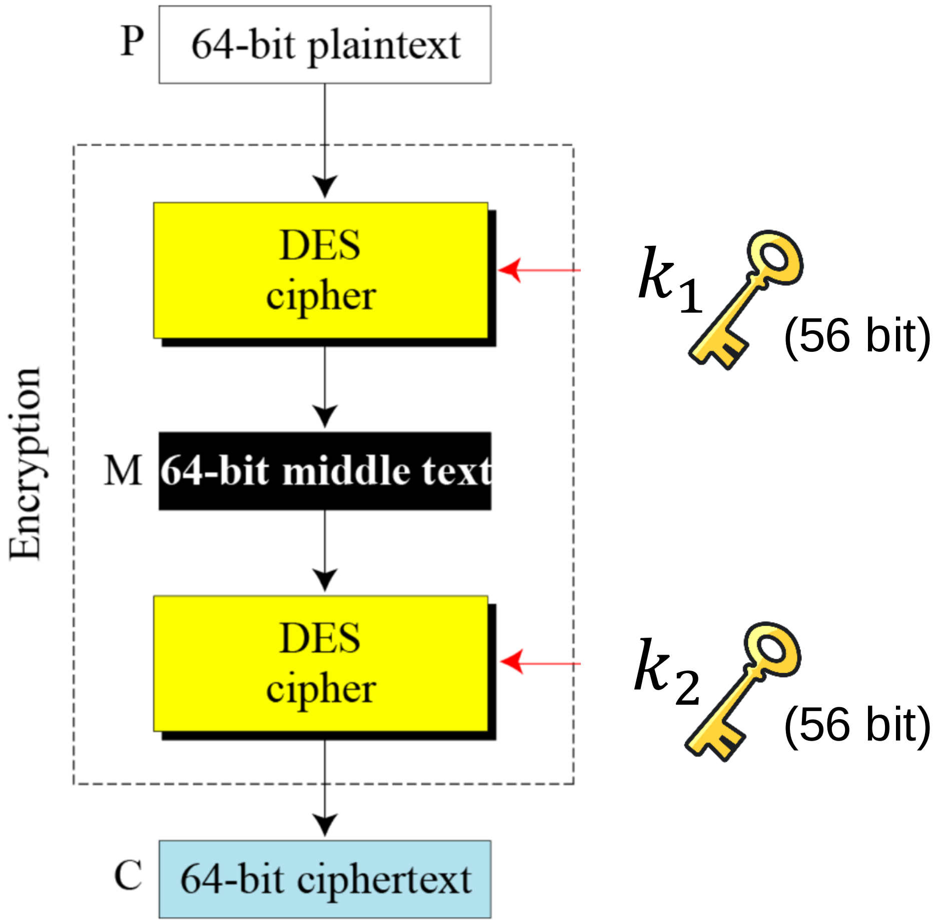
Recap: Known-Plaintext Attack (KPA)

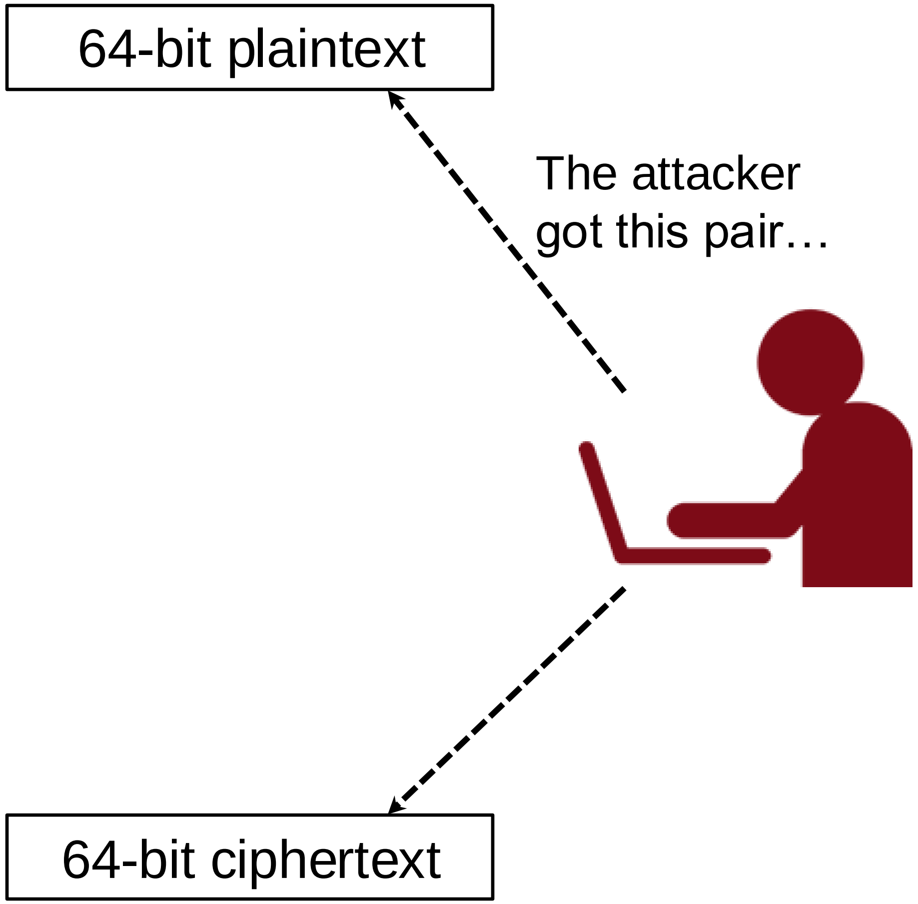
- The attacker is assumed to have access to **multiple plaintexts and their corresponding ciphertexts**
- Can the attacker compute the key from the ciphertext?



Meet-in-the-Middle-Attack for Double DES

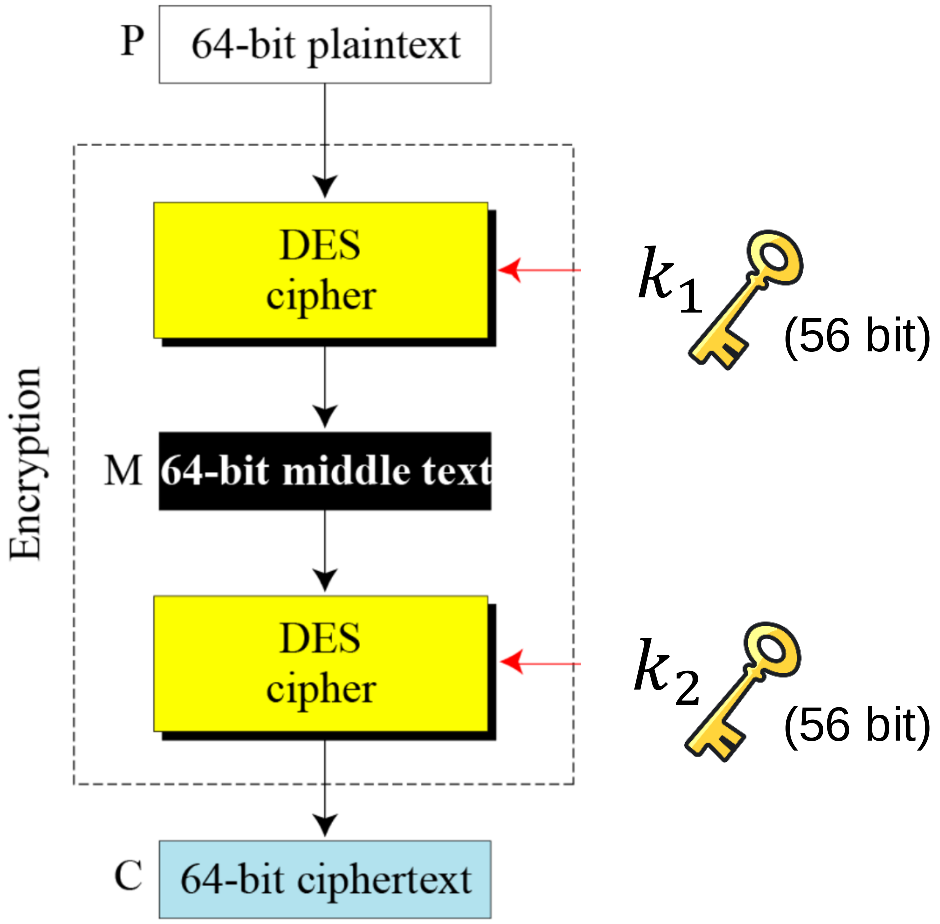
- $C = E_{K2}(E_{K1}(PB))$
- Vulnerable to "meet-in-the-middle" attack

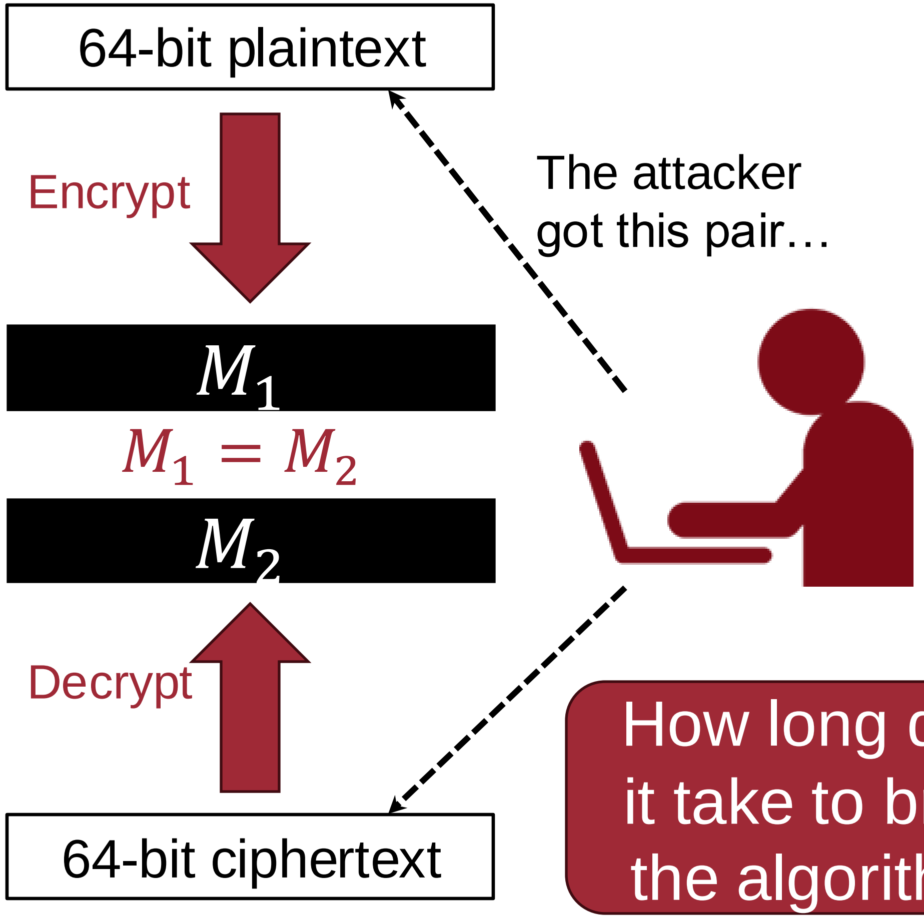




Meet-in-the-Middle-Attack for Double DES

- $C = E_{K2}(E_{K1}(PB))$
- Vulnerable to “meet-in-the-middle” attack

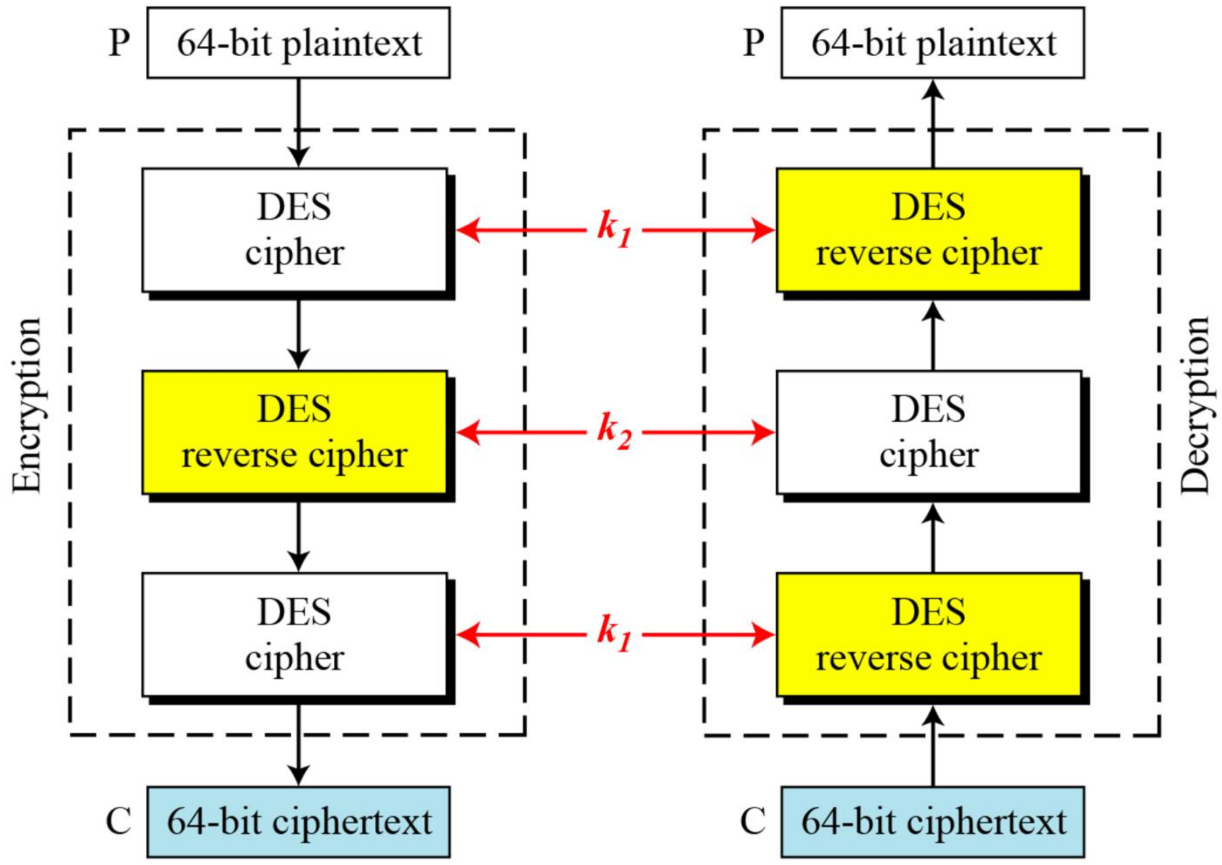




How long does it take to break the algorithm?

Triple-DES with Two-Keys

- Use three encryptions
- But we can use two keys with E-D-E sequence
  - $C = E_{K1}(D_{K2}(E_{K1}(PB)))$



Triple-DES with Two-Keys

- Use three encryptions
- But we can use two keys with E-D-E sequence
  - $C = E_{K1}(D_{K2}(E_{K1}(PB)))$
- Standardized in ANSI X9.17 & ISO8732
- Has been adopted by some Internet applications, e.g., PGP, S/MIME
- Deprecated after 2023
  - Vulnerable to Sweet32 attack

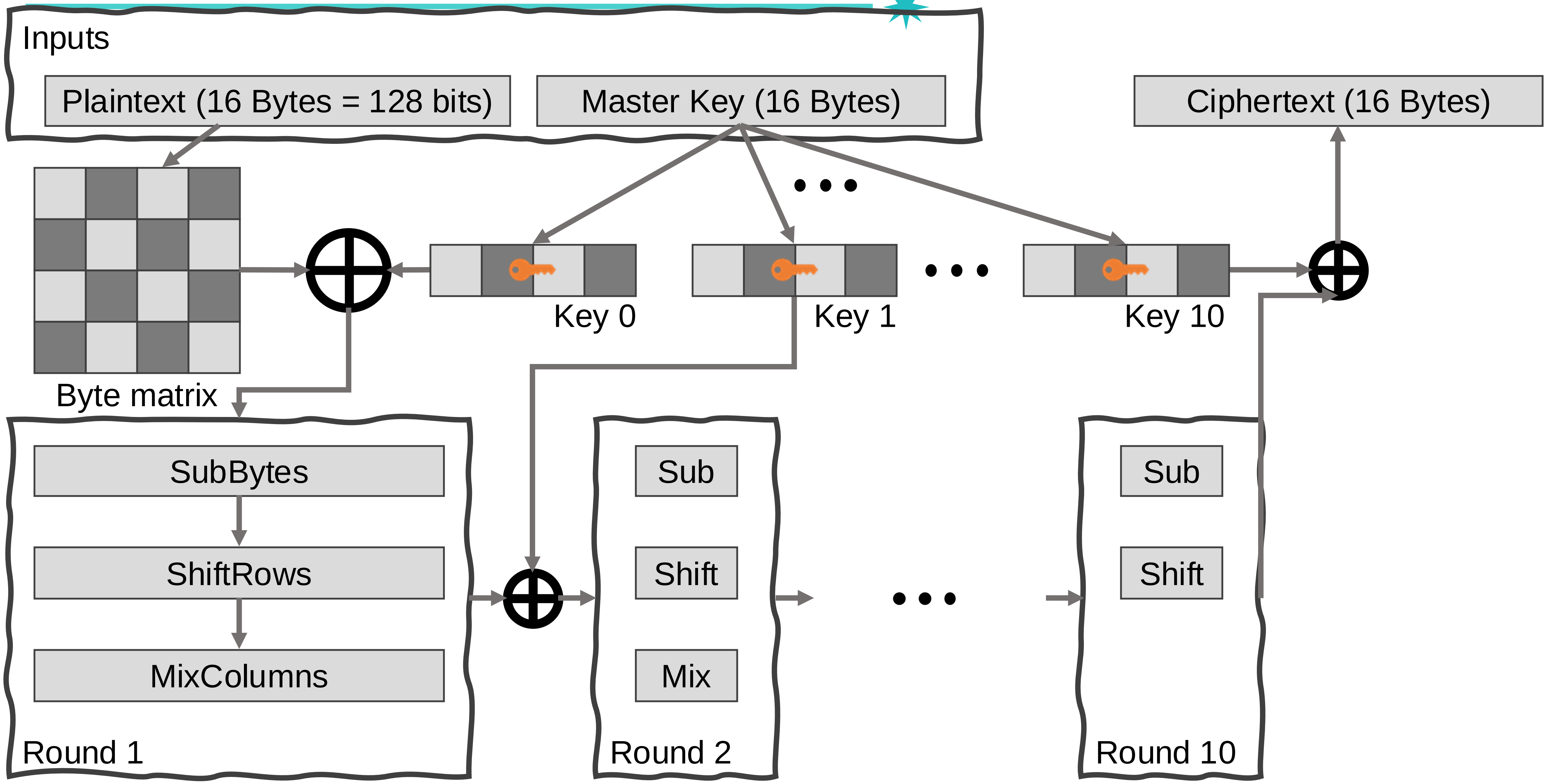
Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES)

- Established by the U.S. National Institute of Standards and Technology (NIST) in 2001
- Selected by an open competition
  - Skipping long history
- A.k.a., Rijndael Cipher

- Triple-DES is slow in software, has small blocks
- A *de-facto* standard symmetric key scheme
  - Substitution-permutation (SP) ciphers
  - Key length: 128 or 192 or 256 bits
  - Block length: 128 bits

Advanced Encryption Standard (AES)



States

- Consider the minimum case of 128-bit key
- Input and output: 4 x 4 matrix of bytes
- (Intermediate) State: 4 x 4 matrix of bytes

input bytes

in_0	in_4	in_8	in_{12}
in_1	in_5	in_9	in_{13}
in_2	in_6	in_{10}	in_{14}
in_3	in_7	in_{11}	in_{15}

→

State array

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

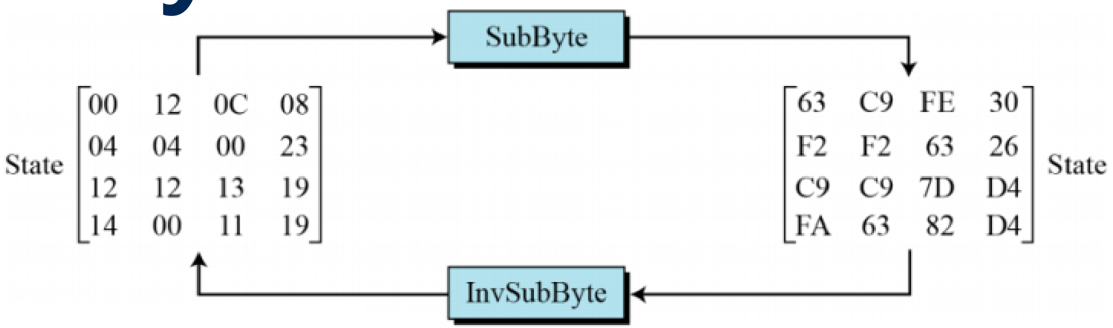
→

output bytes

out_0	out_4	out_8	out_{12}
out_1	out_5	out_9	out_{13}
out_2	out_6	out_{10}	out_{14}
out_3	out_7	out_{11}	out_{15}

AES in a Nutshell (1): SubBytes

- Non-linear byte substitution
- Example: if $S_{1,1} = 53\ then\ S'_{1,1} = ed$



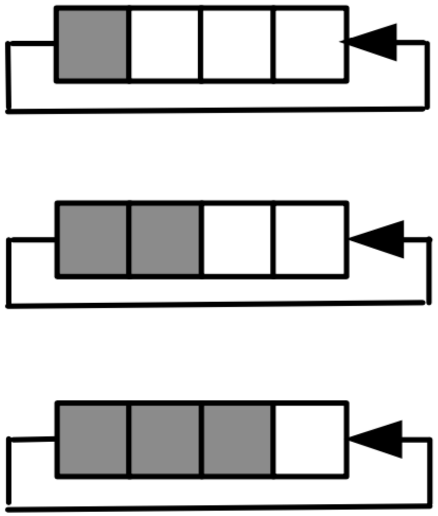
		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	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

AES in a Nutshell (2): ShiftRows

- Circular shift over different numbers of bytes

S

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



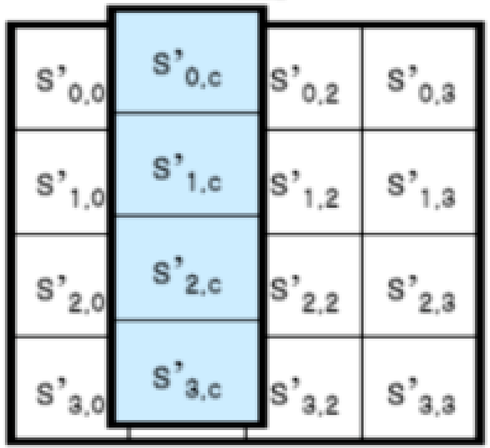
S ’

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

AES in a Nutshell (3): MixColumns

- Matrix multiplication on each column
  - Multiplied by a fixed array

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$



AES in a Nutshell (4): Put it All Together

- Encryption
  - For each round: AddRoundKey ∘ MixColumns ∘ ShiftRows ∘ SubBytes

- Decryption: the inverse of the encryption
  - For each round: $\text{SubBytes}^{-1} \circ \text{ShiftRows}^{-1} \circ \text{MixColumns}^{-1} \circ \text{AddRoundKey}^{-1}$

It is highly recommended to simulate the AES algorithm![1)]

[1)] https://formaestudio.com/rijndaelinspector/archivos/Rijndael_Animation_v4_eng-html5.html

AES Design Considerations

- In a Feistel cipher, half the bits are moved, but not changed during each round.
  - AES treats all bits uniformly, making the effect of diffusing the input bits faster
- Until recently, no known attacks that are better than exhaustive key search up to six rounds
  - Four extra rounds provide a large enough security margin of safety

Conclusion

- **Symmetric**: the encryption and decryption keys *are the same*

- **Stream ciphers** encrypt bits individually

- **Block ciphers** encrypt fixed-length groups of bits
  - Design principles for block ciphers
    - Diffusion
    - Confusion
    - Rounds
  - DES (Feistel ciphers)
  - AES (Substitution-permutation ciphers)

Question?