

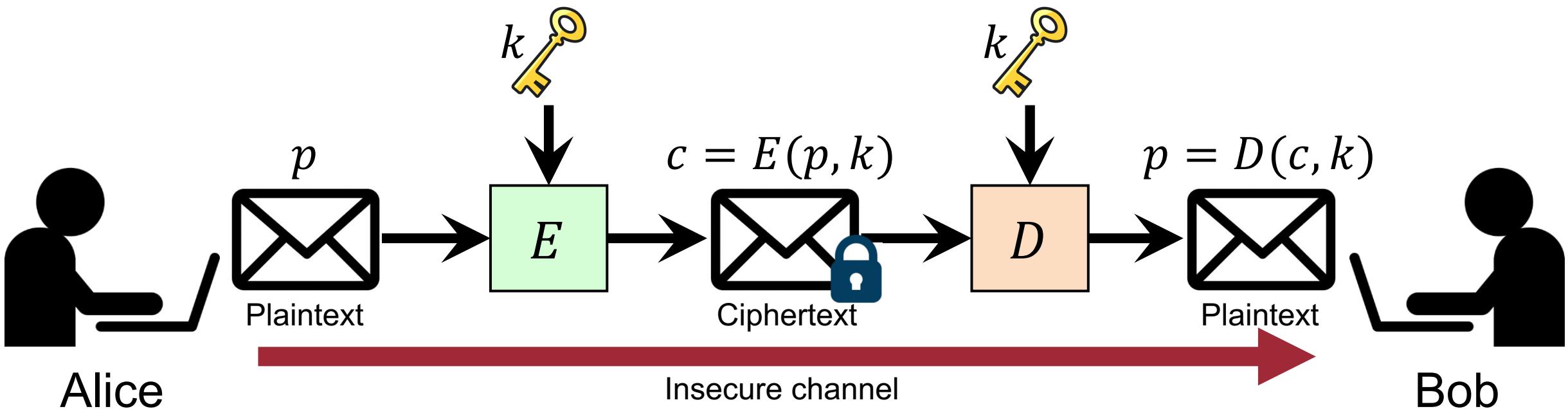
CSE467: Computer Security

4. Symmetric-key Encryption (1)

Seongil Wi

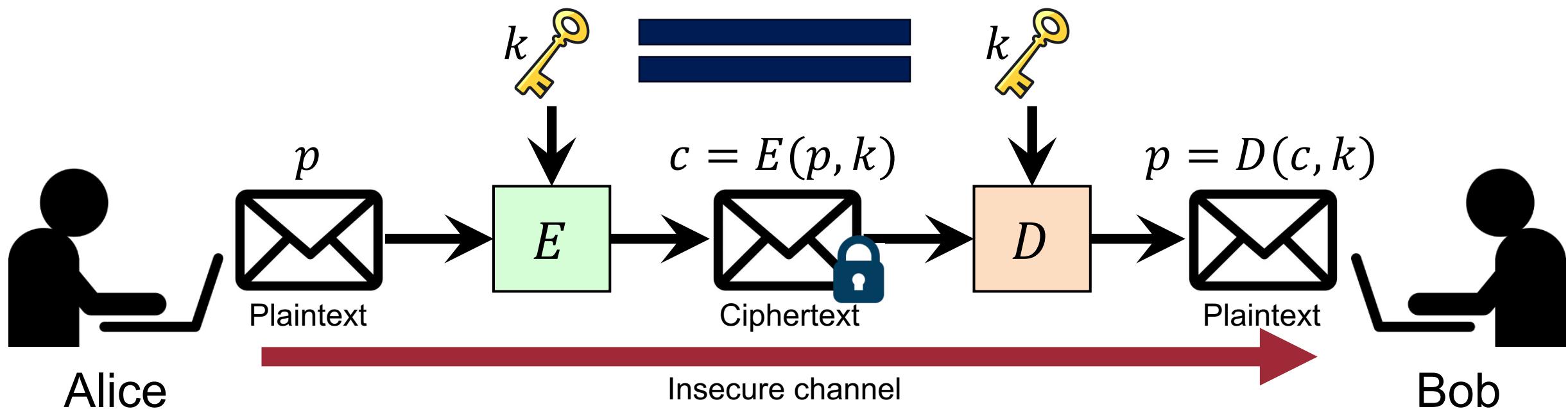
Cryptography

- “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)



Symmetric-key Encryption

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



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*



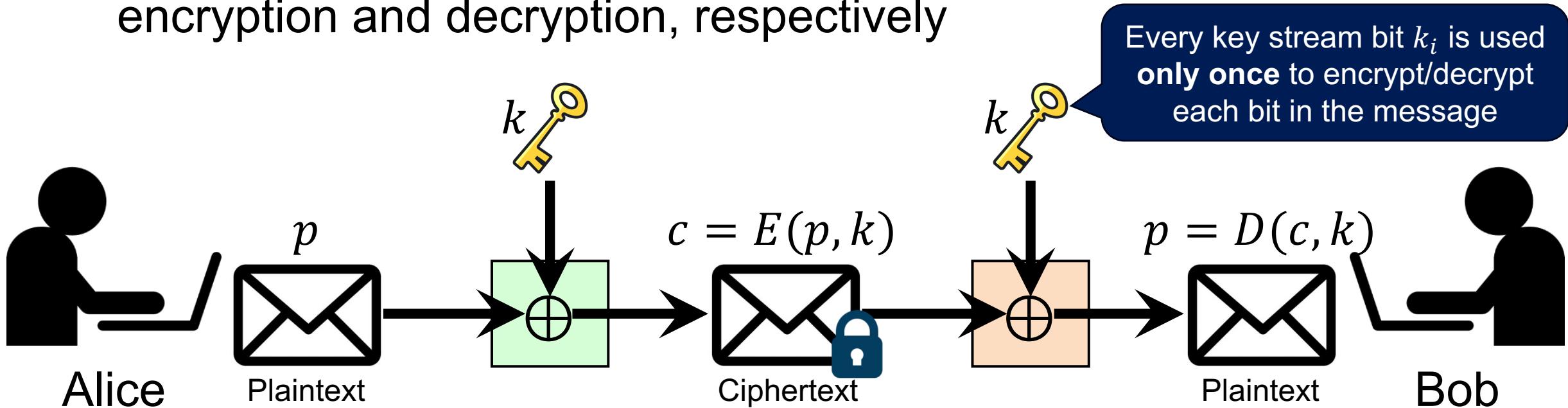
Stream Ciphers vs. Block Ciphers

- 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

6

- 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



Stream Ciphers (Example): One-time Pad

7

- 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



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 should be at least as long as the message
- Both copies of the OTPs are destroyed immediately after use



KGB (USSR)

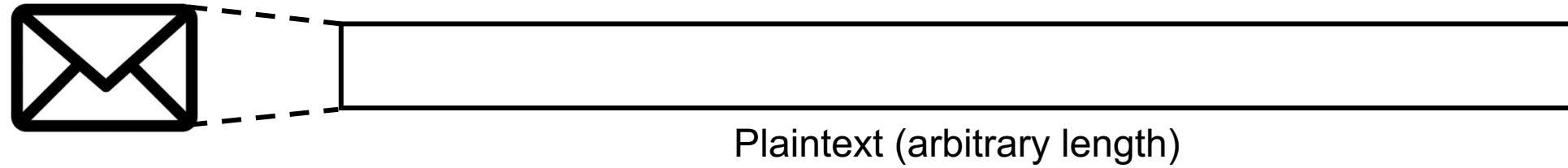


DDR (East-Germany)

Block Cipher

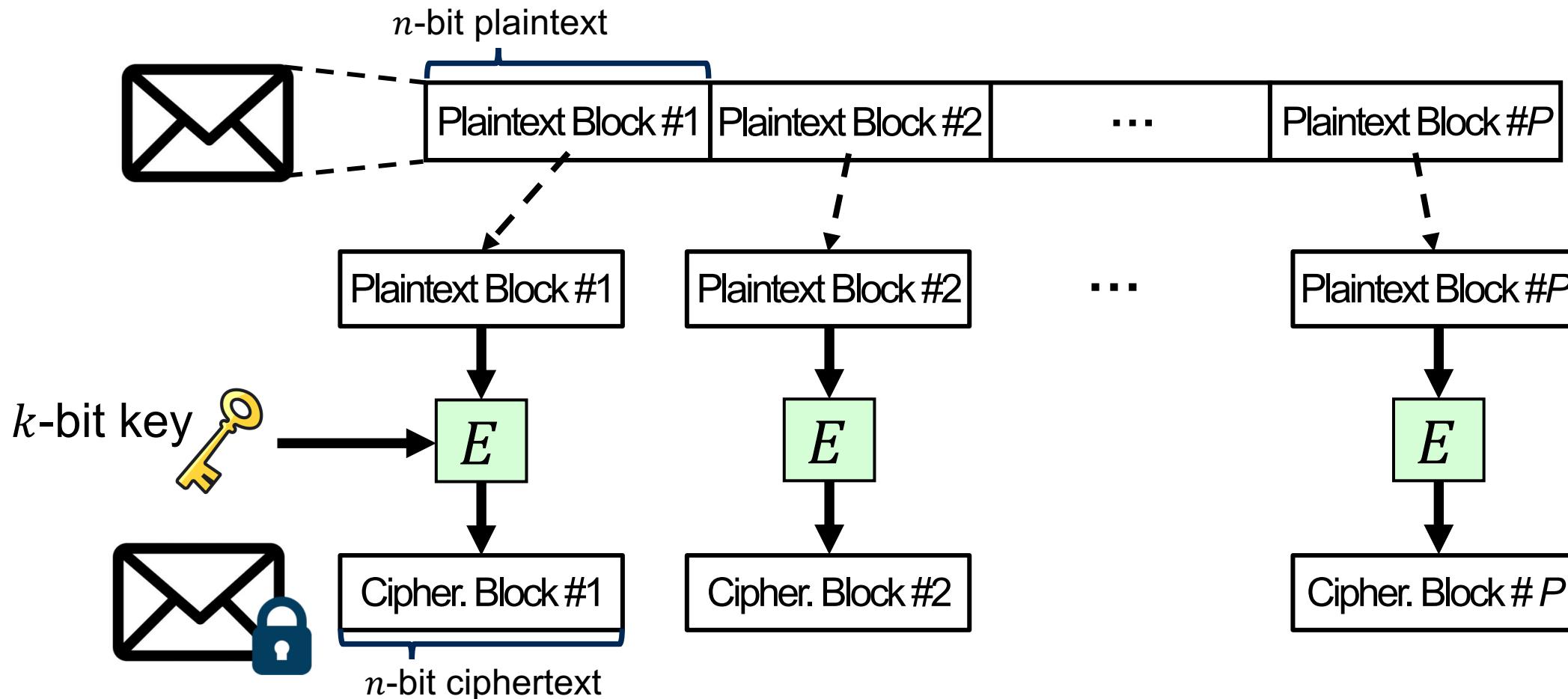


- Encrypt/decrypt data in blocks of fixed lengths (n -bit block)
- Split a message into blocks and encrypt them individually



Block Cipher

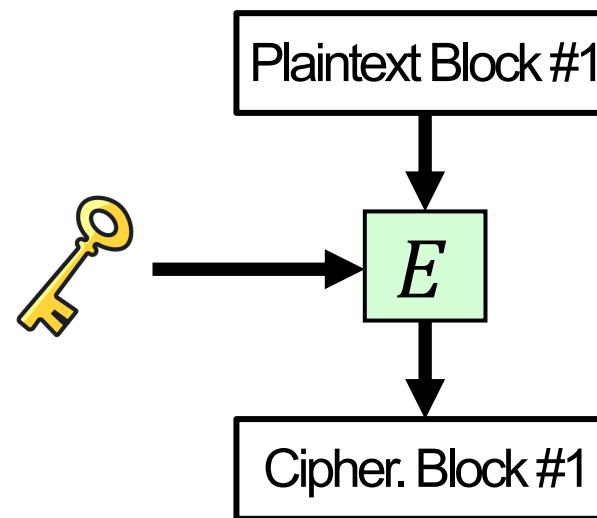
- Encrypt/decrypt data in blocks of fixed lengths (n -bit block)
- Split a message into blocks and encrypt them individually



Components of a Modern Block Cipher

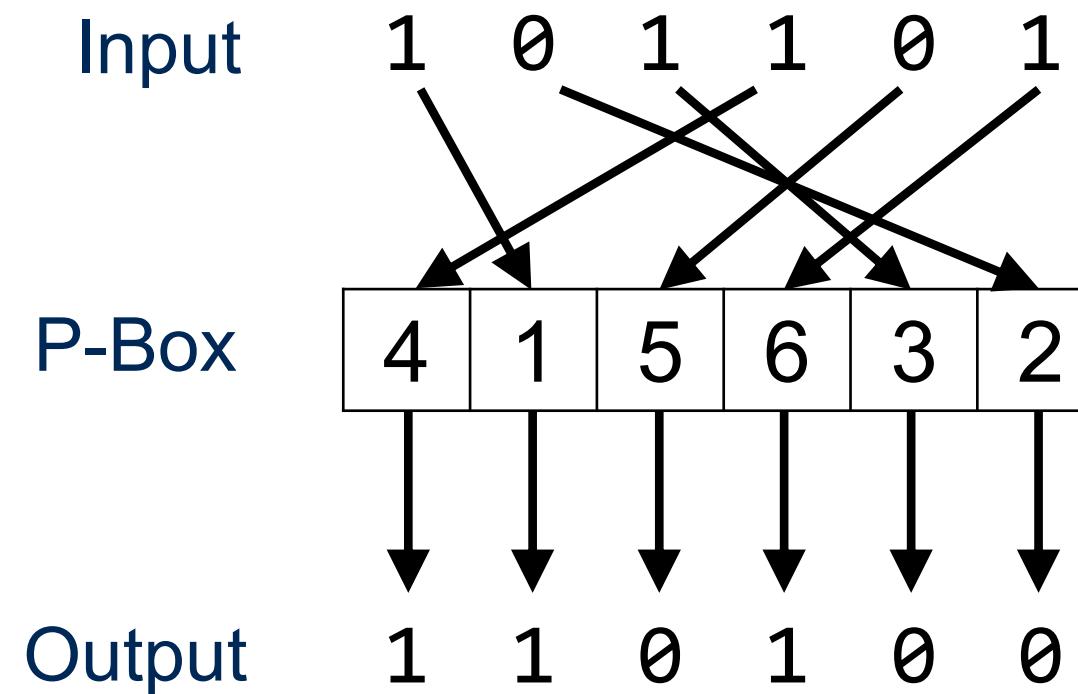
11

- 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)**



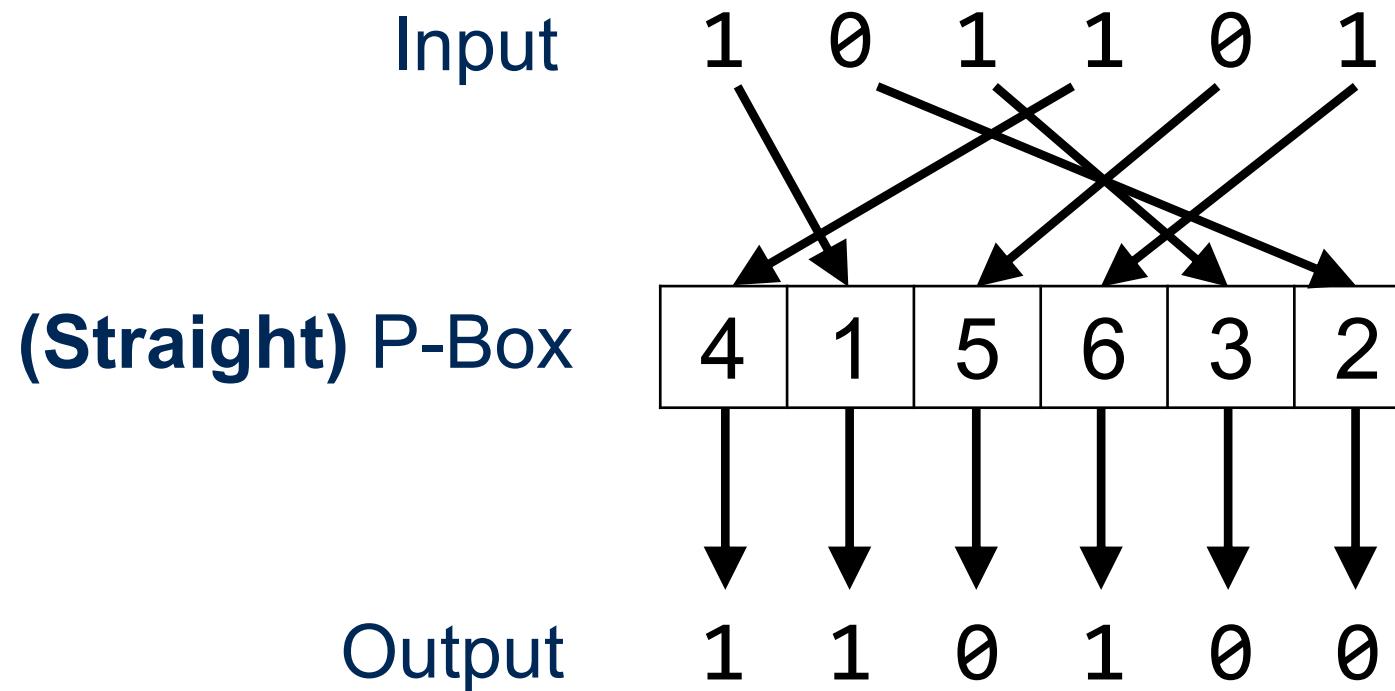
Transposition Cipher (P-Box)

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



Three Types of P-Boxes

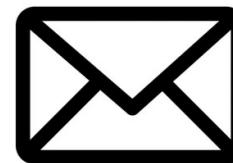
- Straight P-box
- Compression P-box
- Expansion P-box



Exercise: Invertibility

14

- Invert a P-box (used for decryption)



1 2 3 4 5 6

Original P-Box
(used for encryption)

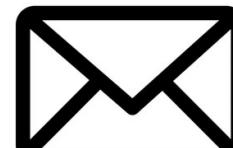
6	3	4	5	2	1
---	---	---	---	---	---



6 3 4 5 2 1

Inverted P-Box
(used for decryption)

6	5	2	3	4	1
---	---	---	---	---	---

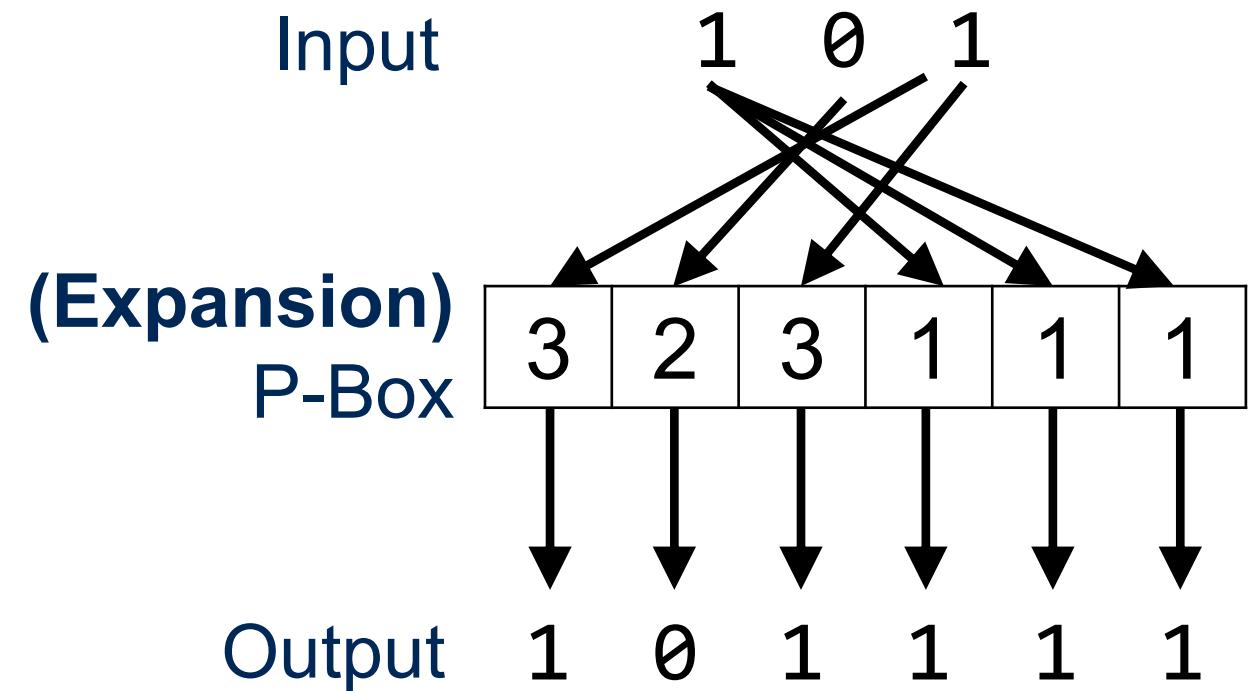
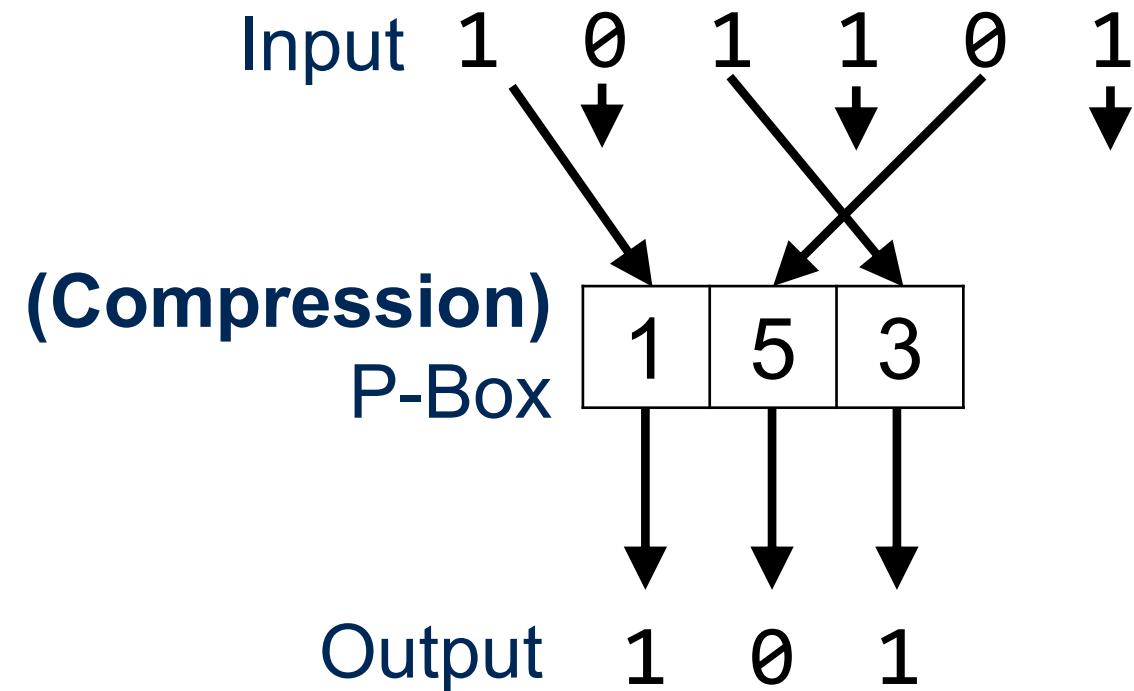


1 2 3 4 5 6

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 $m > n$

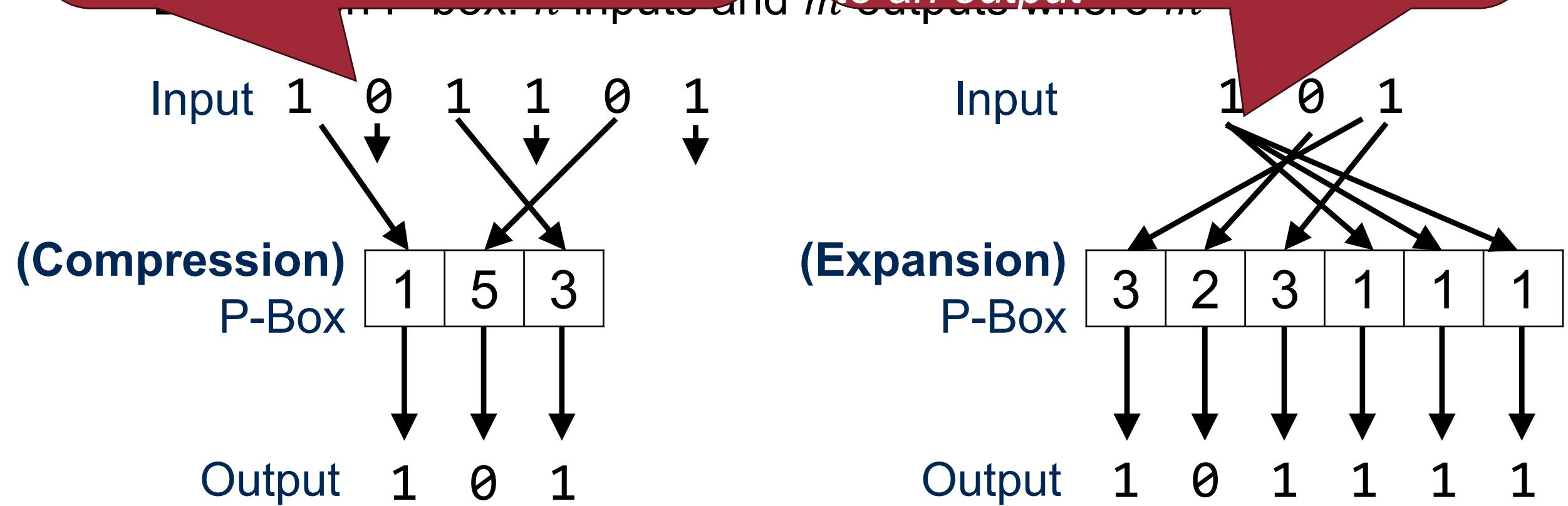


An input can be *dropped*!

=> The decryption algorithm does not have a clue *how to replace the dropped bit*

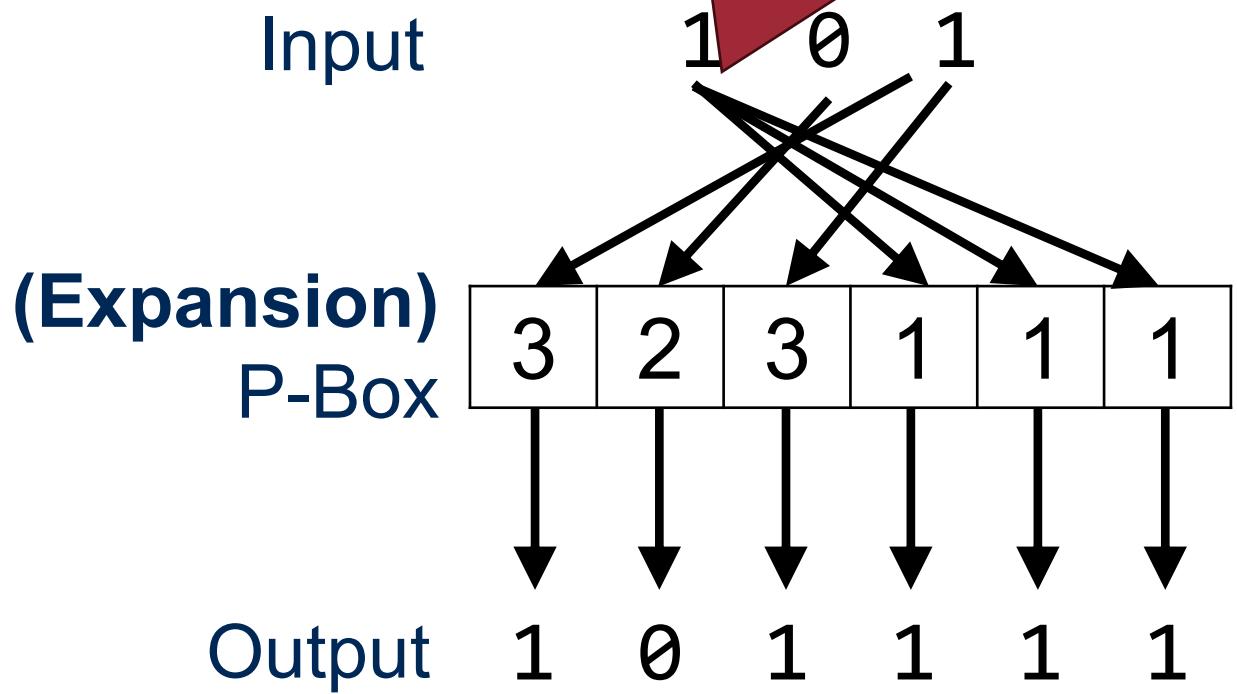
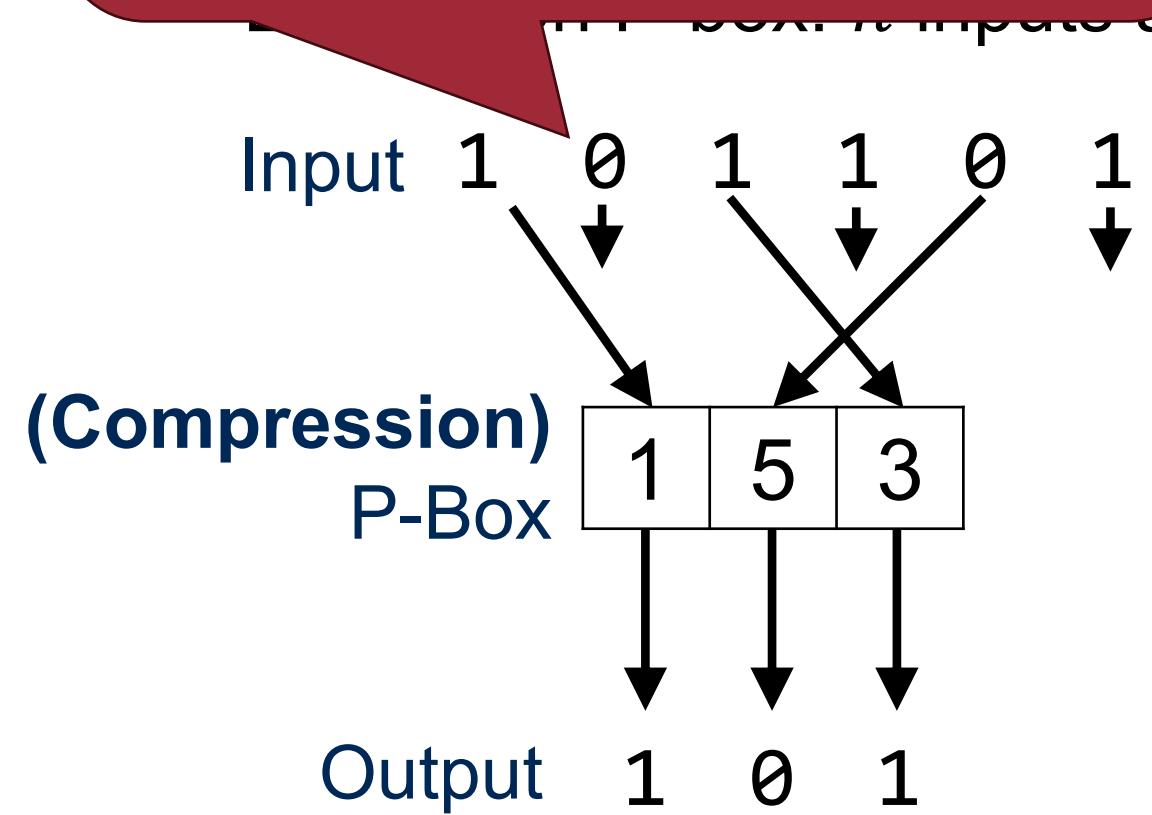
An input can be mapped to *more than one output*

=> The decryption algorithm does not have a clue *which of the several inputs are mapped to an output*



Non-invertible

Non-invertible



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

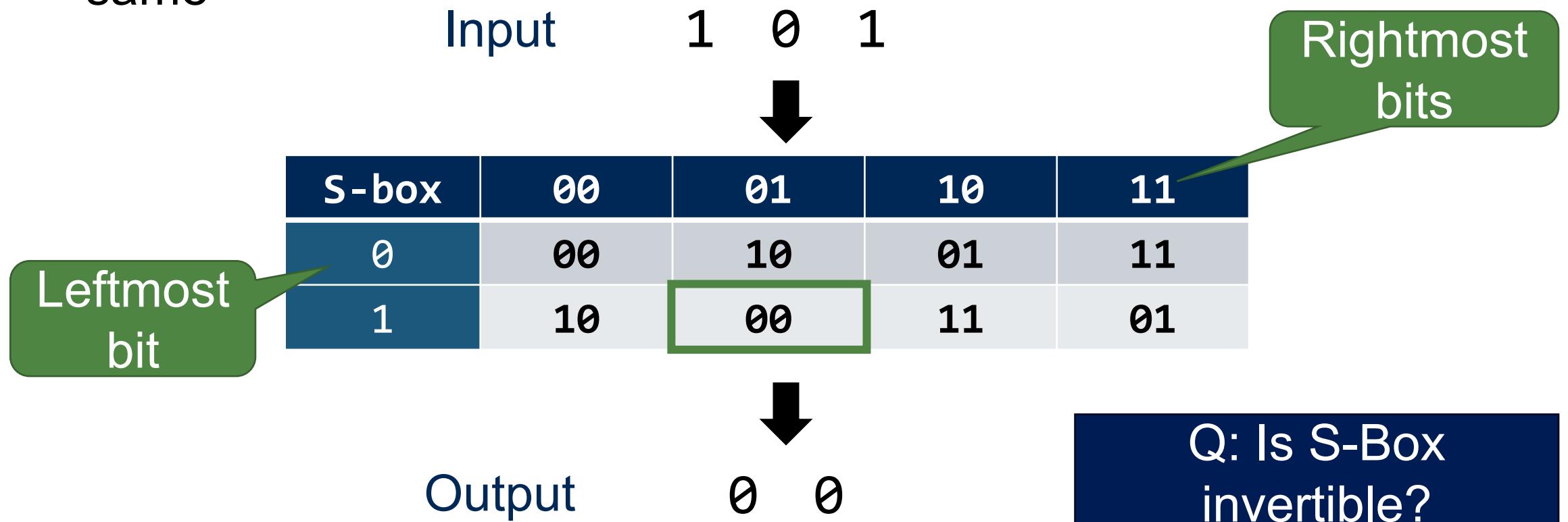
Non-invertible

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



Invertibility of S-Boxes



- An S-box *may or may 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

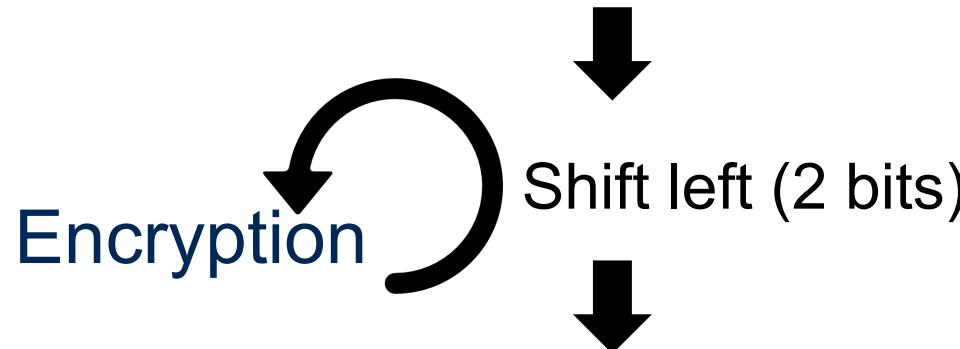
Circular Shift

22

- Recall the Caesar Cipher
- Invertible operation

Input

1	1	1	1	0	1
---	---	---	---	---	---

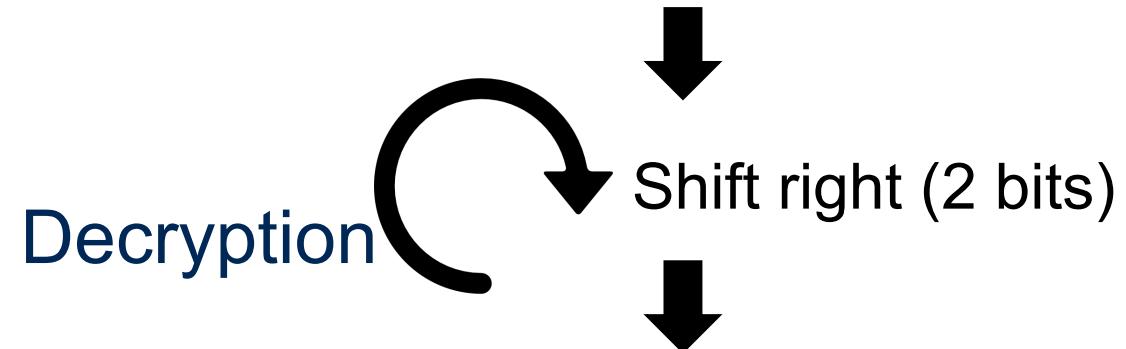


Output

1	1	0	1	1	1
---	---	---	---	---	---

Input

1	1	0	1	1	1
---	---	---	---	---	---



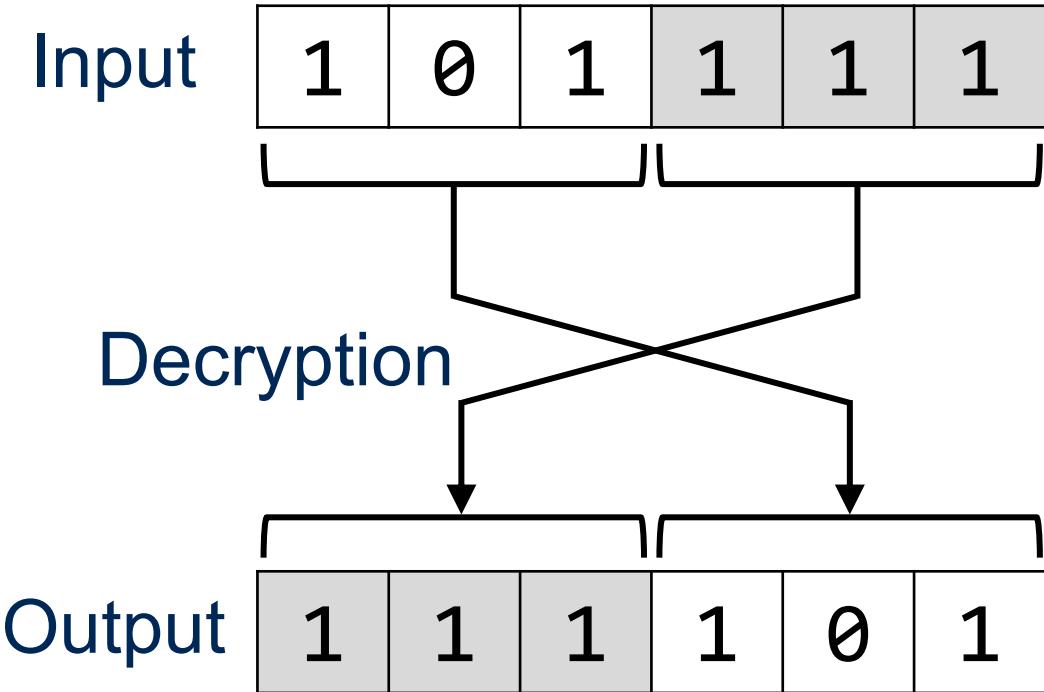
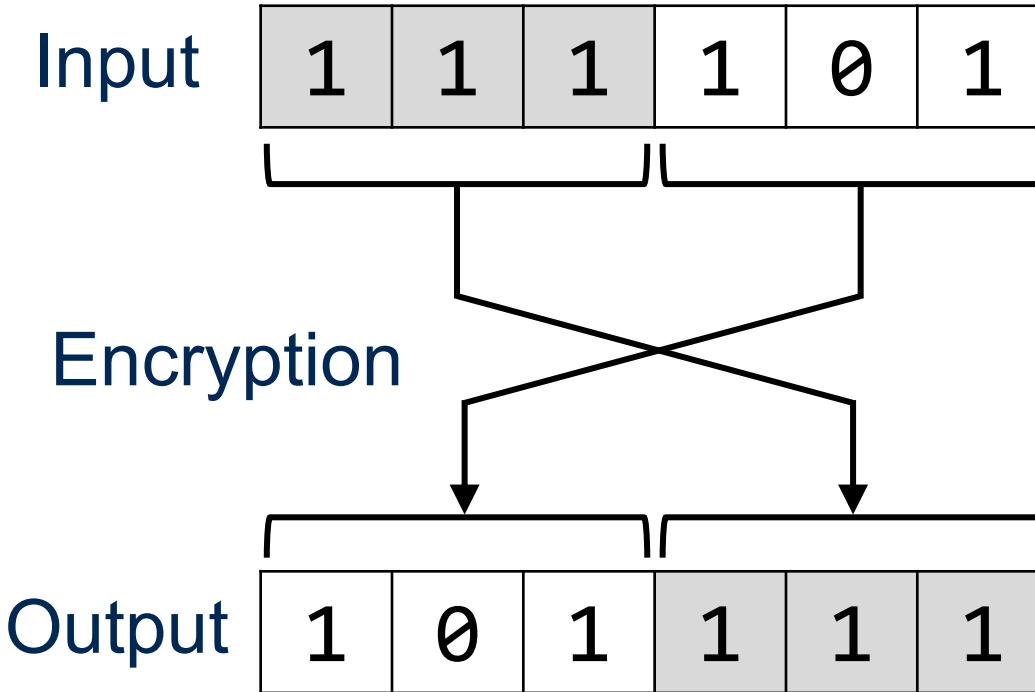
Output

1	1	1	1	0	1
---	---	---	---	---	---

Swap



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



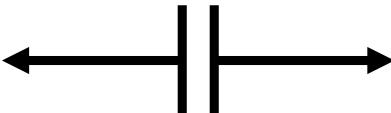
Split and Combine

24

- Invertible operations

Input

1	1	1	1	0	1
---	---	---	---	---	---

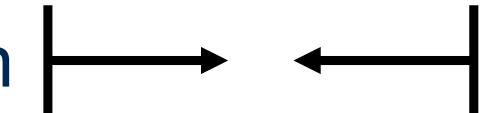
Encryption 

Output

1	1	1
1	0	1

Input

1	1	1
1	0	1

Decryption 

Output

1	1	1	1	0	1
---	---	---	---	---	---

Design Principles for Block Ciphers

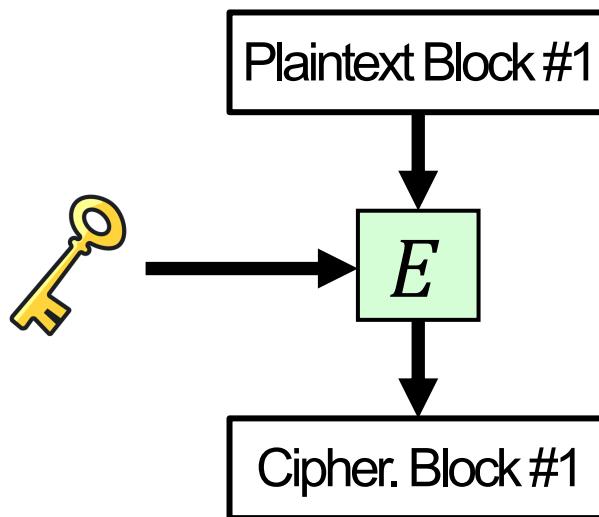
25

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

*Property #1:
Diffusion*

*Property #2:
Confusion*

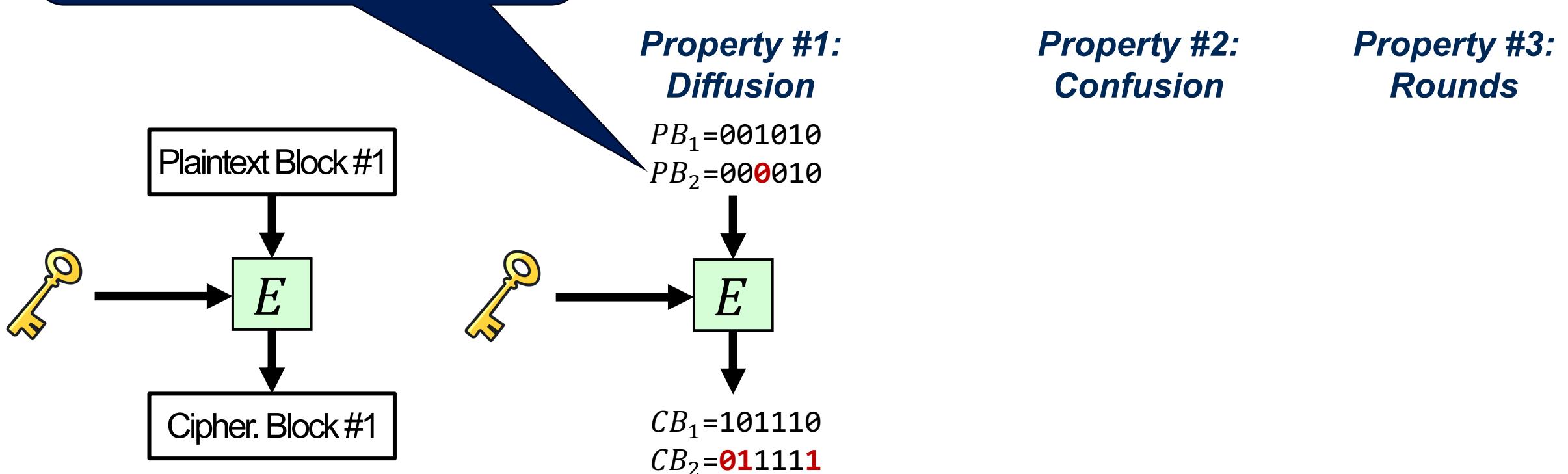
*Property #3:
Rounds*



Property #1: Diffusion

- A modern block ciphers are usually made of combination of

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



Recall: Chosen-Plaintext Attack (CPA)



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

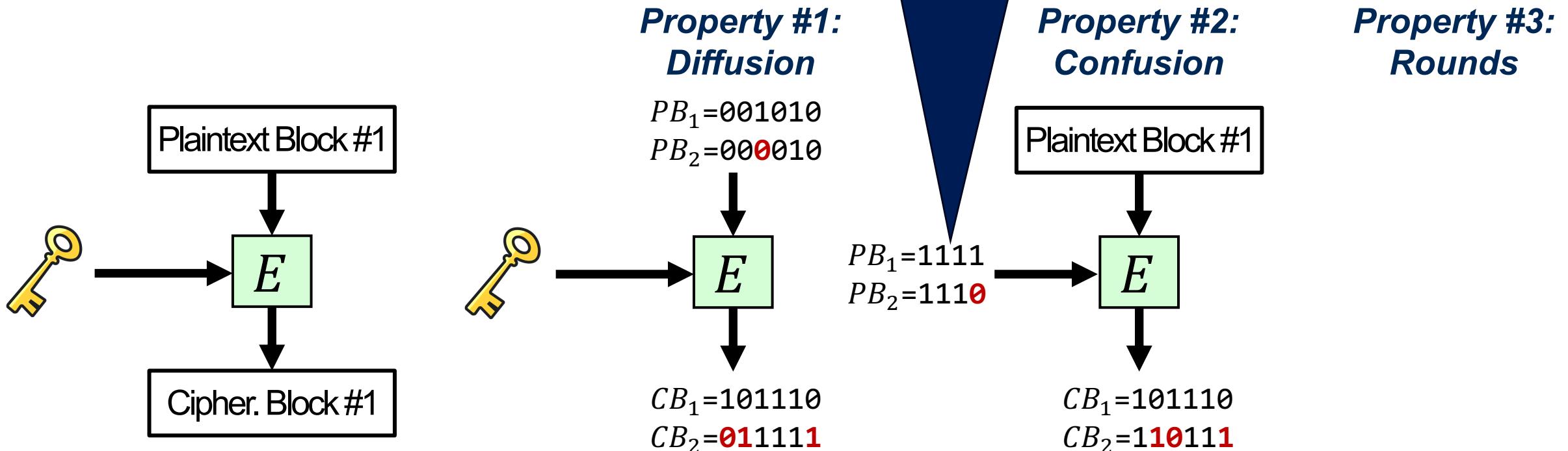


	Plaintext	Ciphertext
Try #1	11111	01001
Try #2	11110	01000
Try #3	11101	01011
Try #4	11011	01101
Try #5	10111	00001
Try #6	01111	11001

Property #2: Confusion

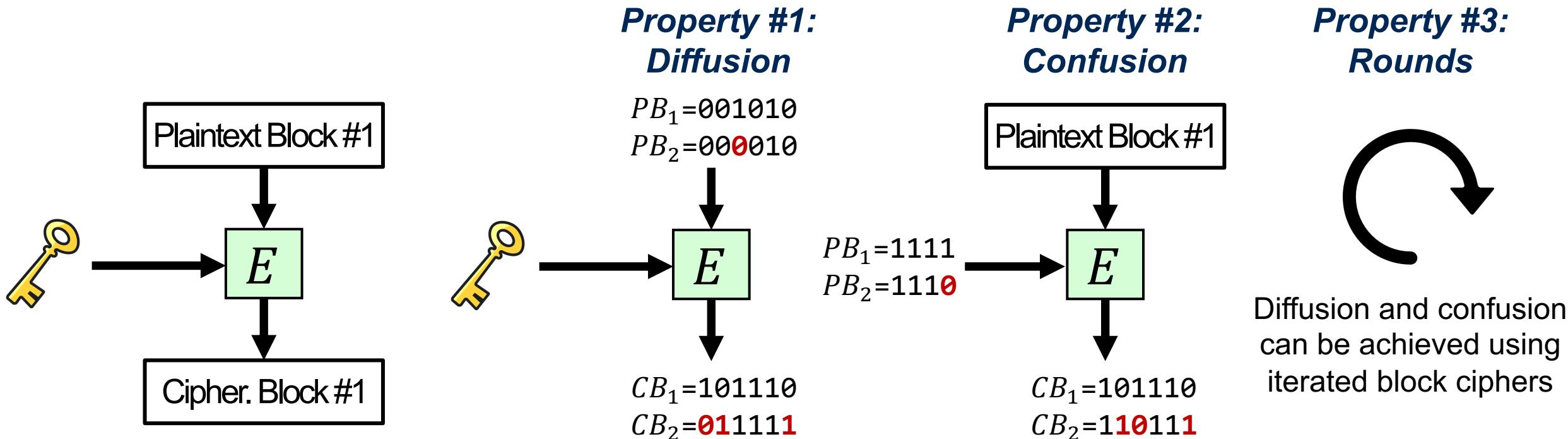
- A modern block ciphers are usually made of combination of
 - Transposition cipher
 - Substitution cipher
 - Some other units

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

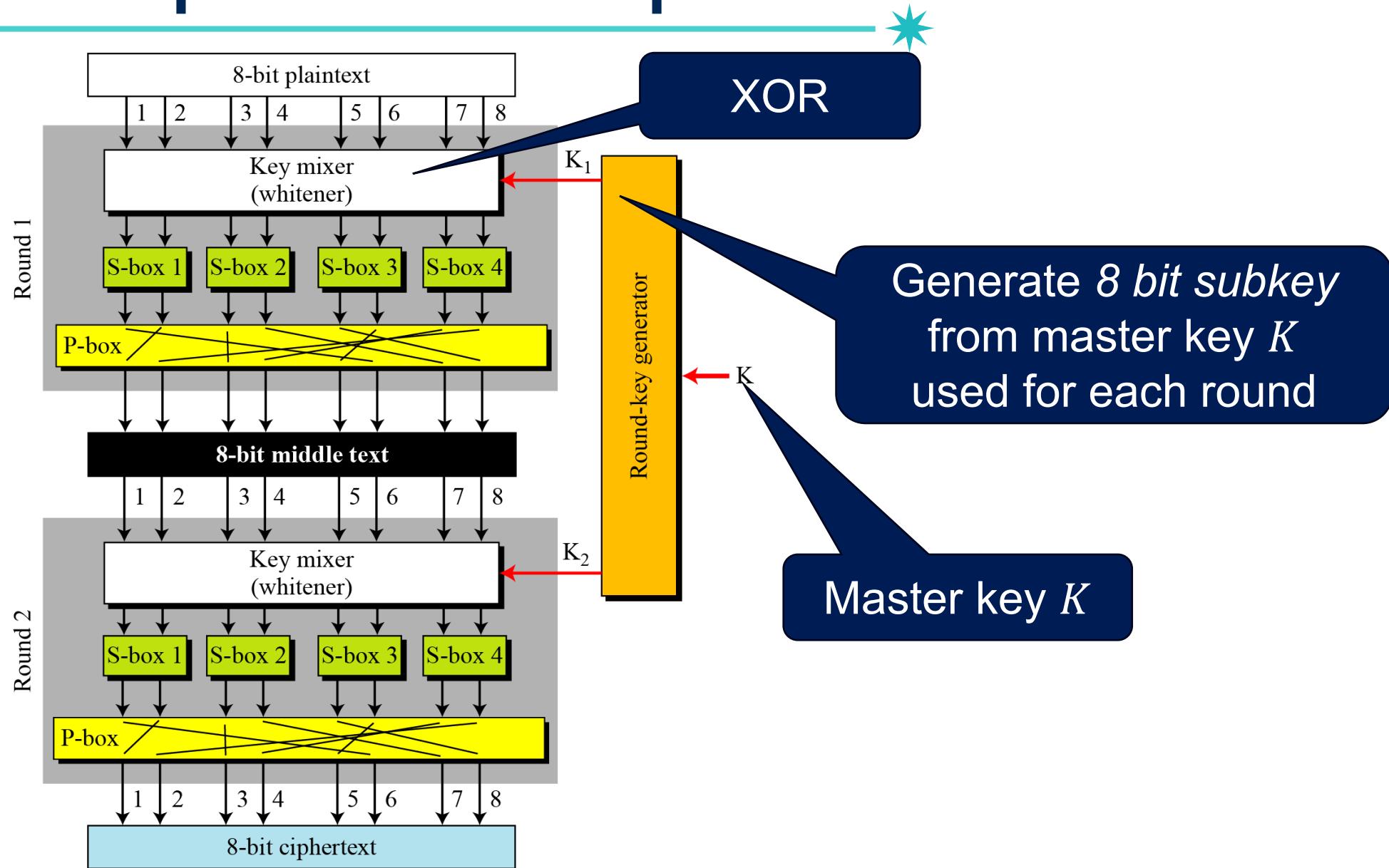


Property #3: Rounds

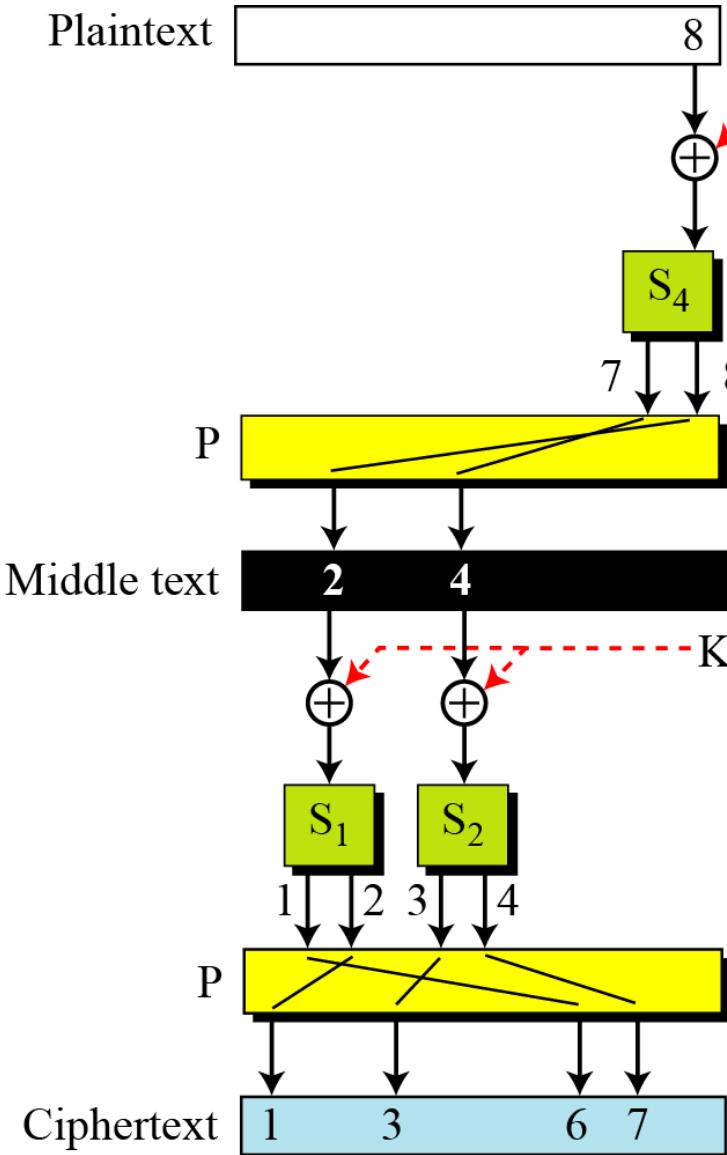
- A modern block ciphers are usually made of combination of
 - Transposition cipher
 - Substitution cipher
 - Some other units



Example: A Block Cipher made of Two Rounds³⁰



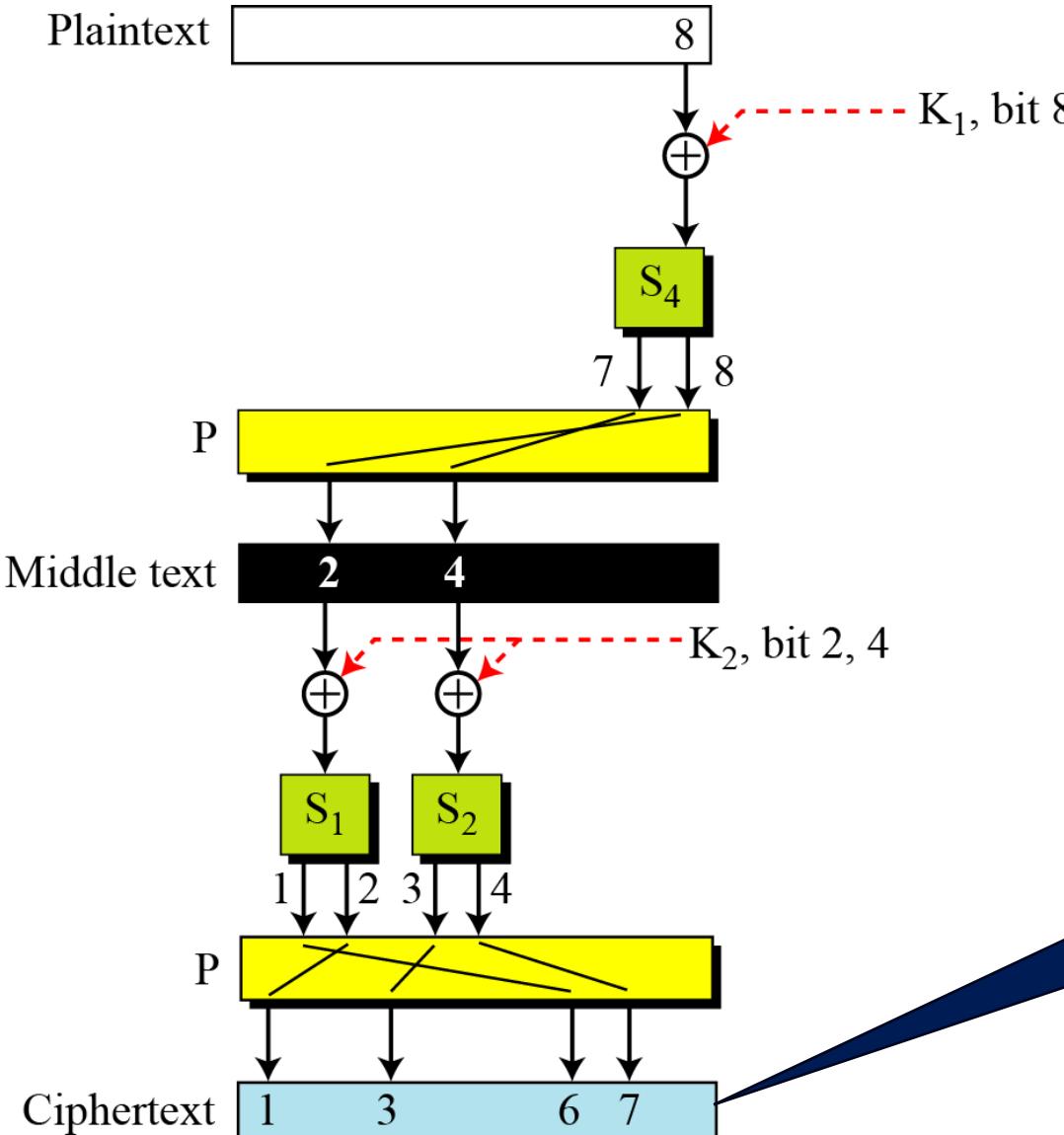
Example: A Block Cipher made of Two Rounds³¹



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 Rounds³²



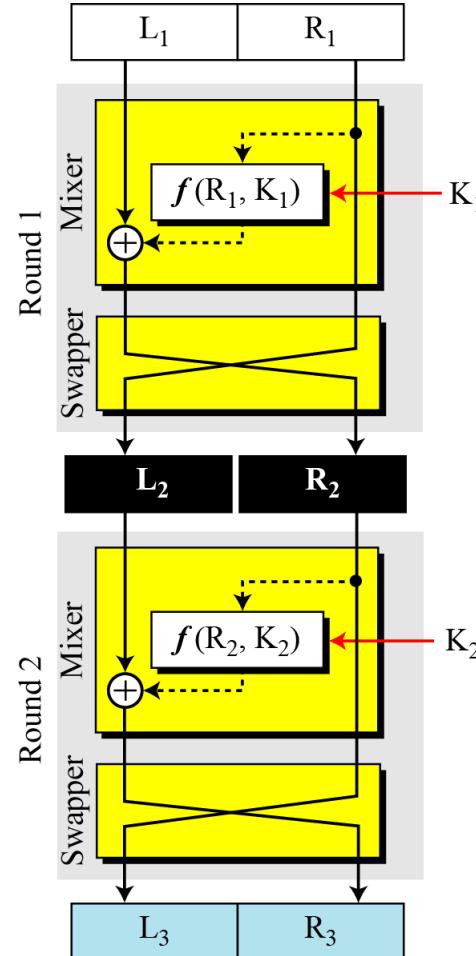
Confusion

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

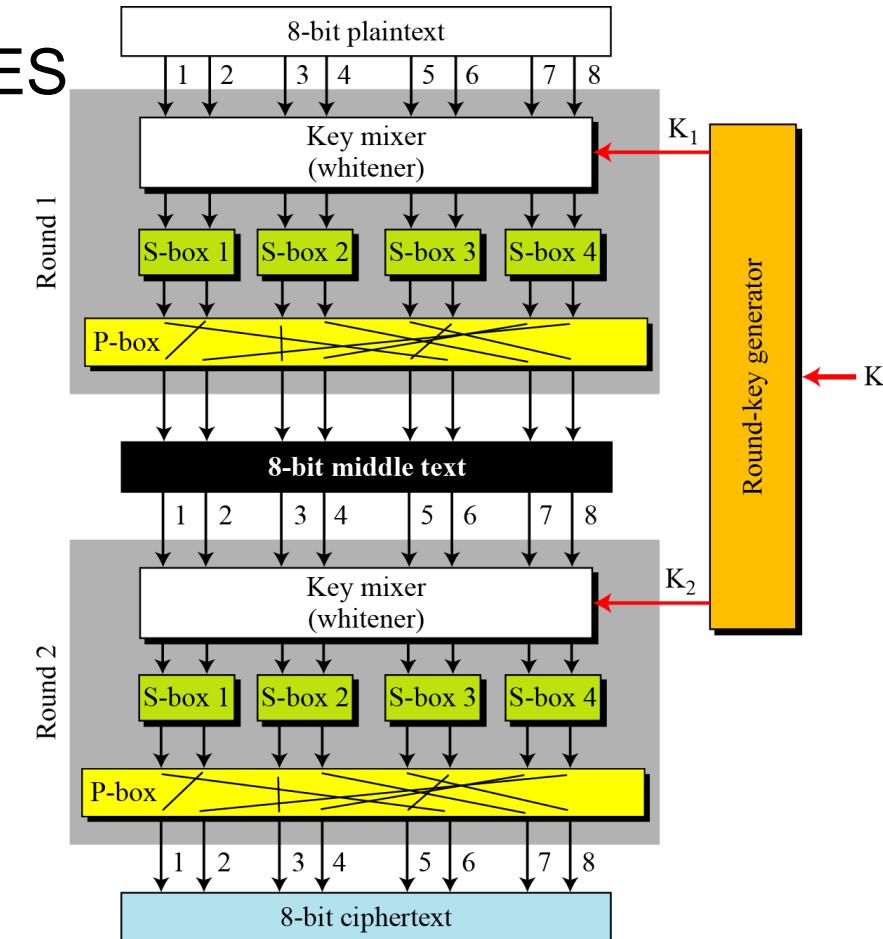
Two Classes of Block Ciphers

33

- Feistel ciphers
 - E.g., DES



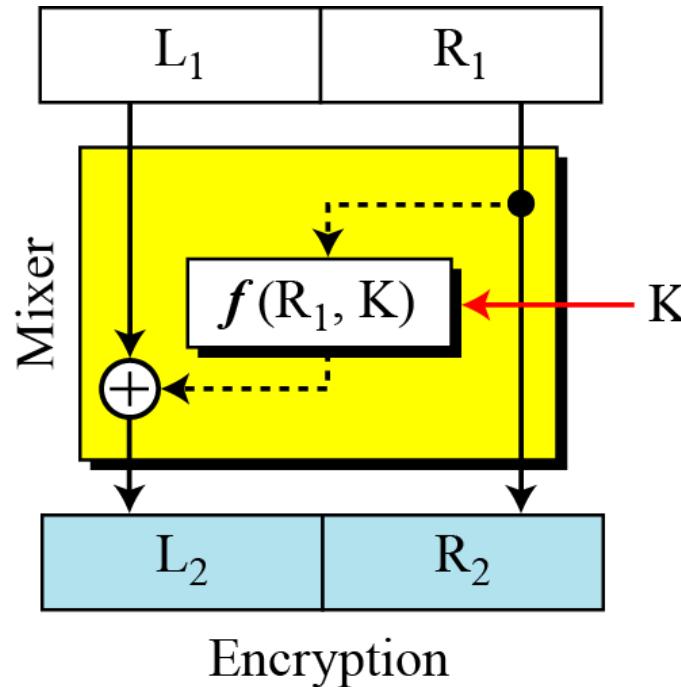
- Substitution-permutation (SP) ciphers
 - E.g., AES



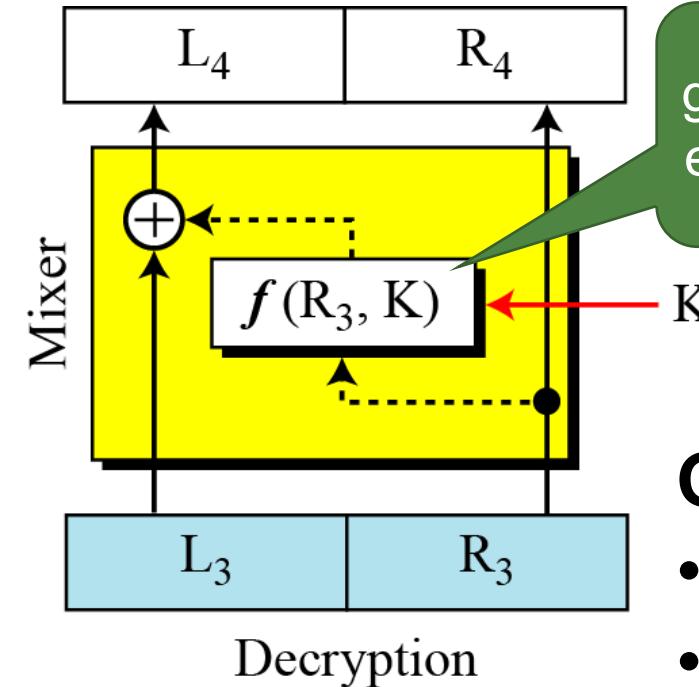
Feistel Cipher – Basic Components

34

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



$$L_2 = L_1 \oplus f(R_1, K)$$



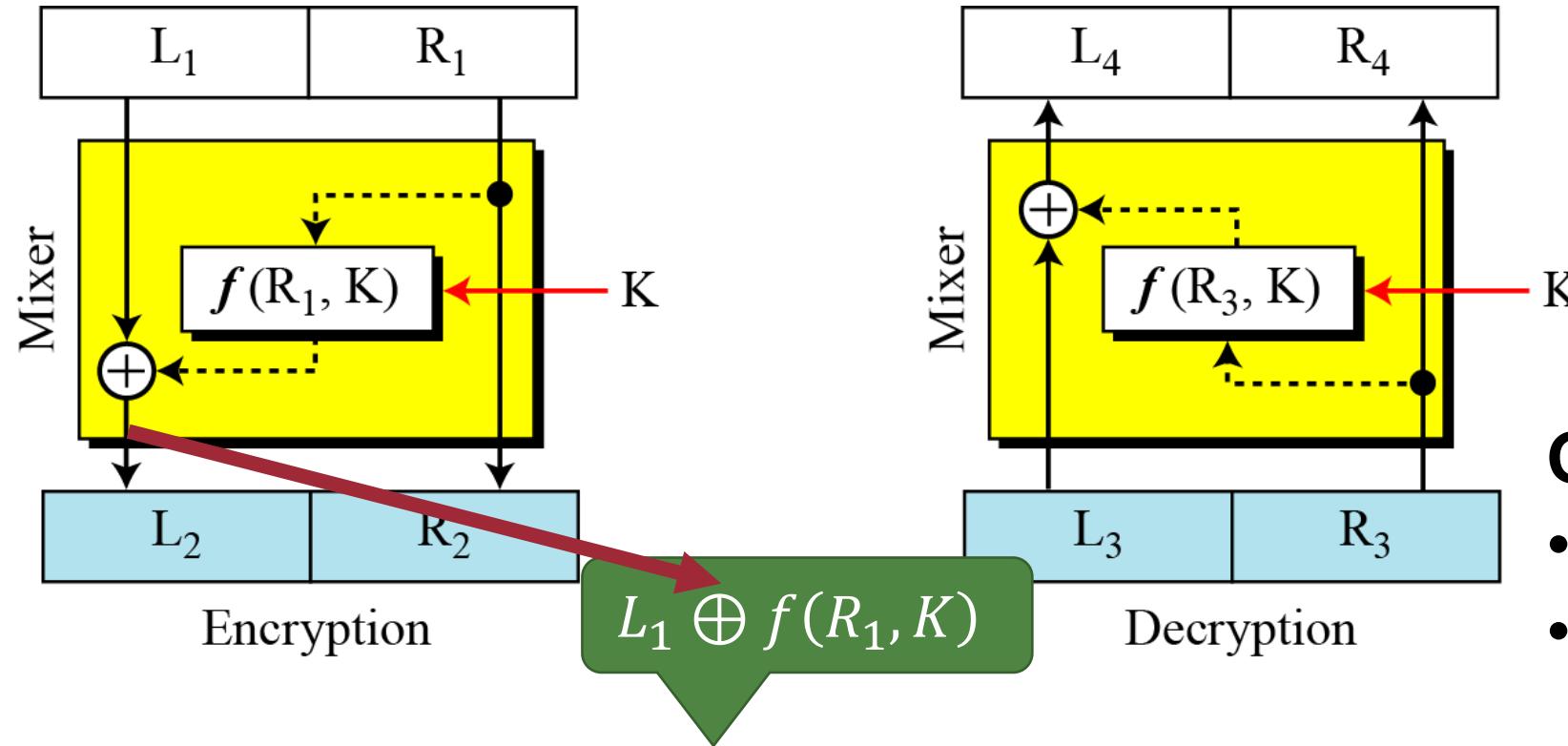
Ground Truth

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

Feistel Cipher – Basic Components

35

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



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

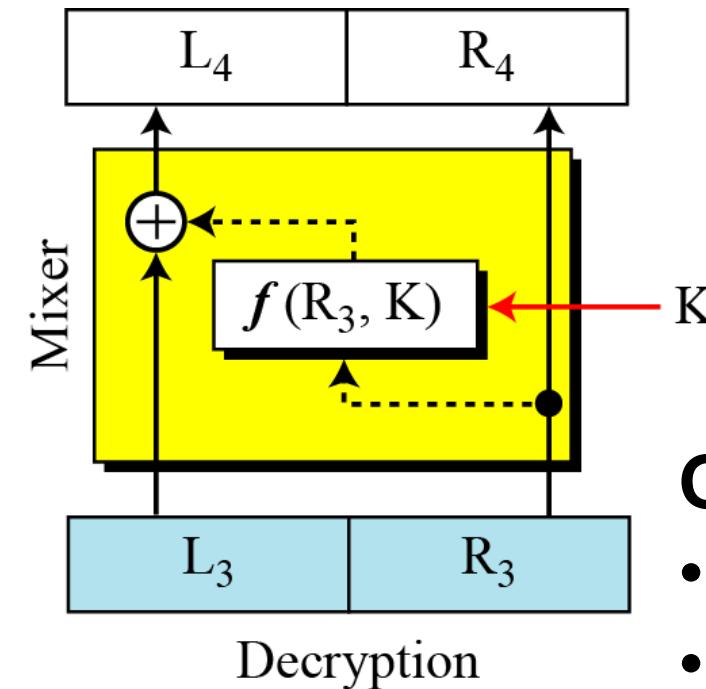
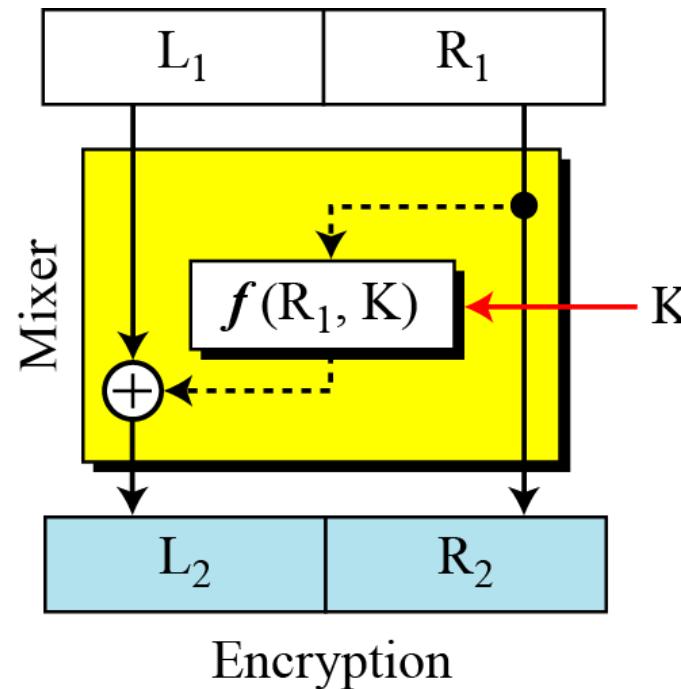
Ground Truth

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

Feistel Cipher – Basic Components

36

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



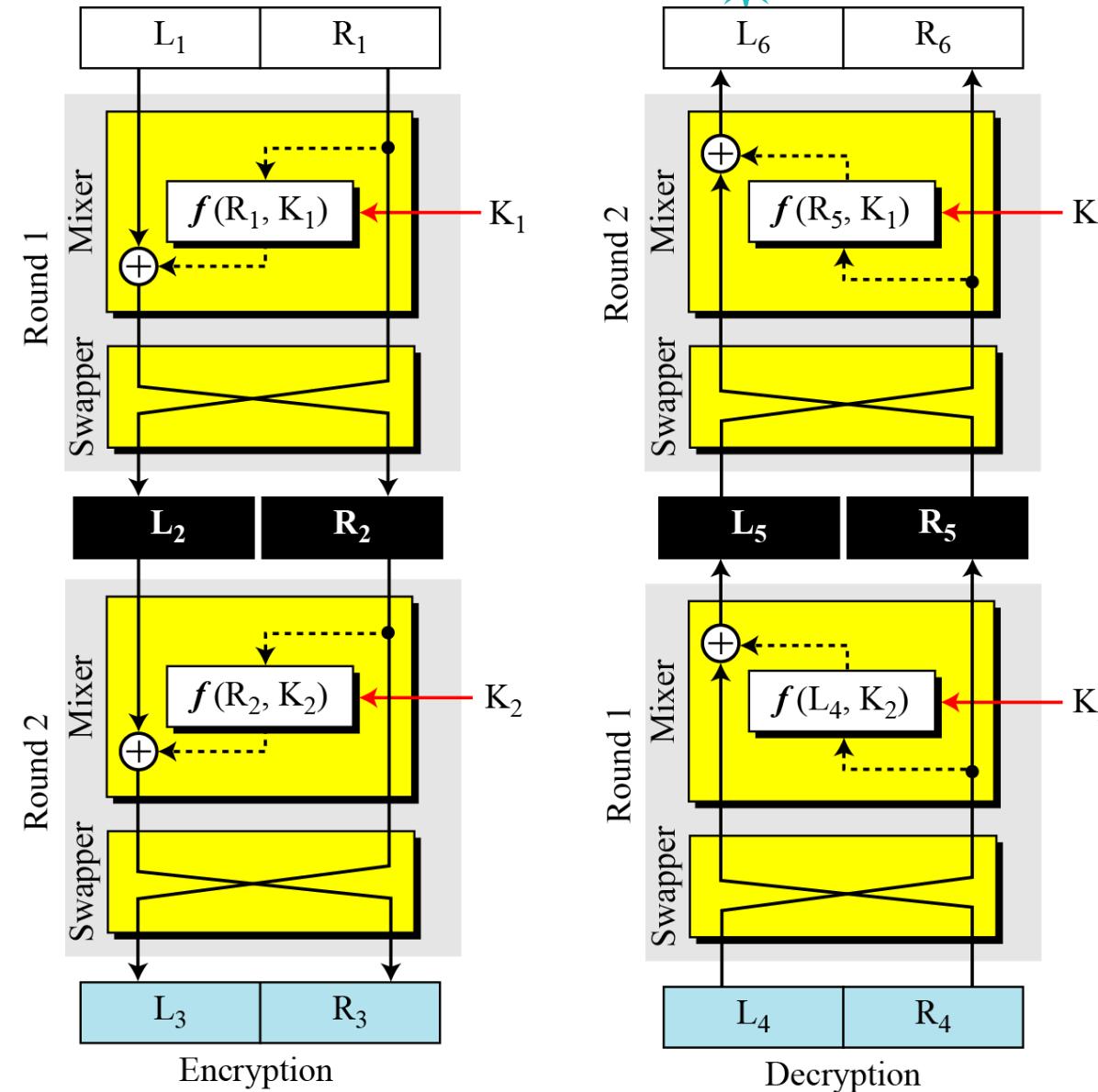
Ground Truth

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

$$\begin{aligned}L_4 &= L_3 \oplus f(R_3, K) = L_2 \oplus f(R_1, K) \\&= L_1 \oplus f(R_1, K) \oplus f(R_1, K) = L_1 \oplus 0 = L_1\end{aligned}$$

A Feistel Cipher with Two Rounds

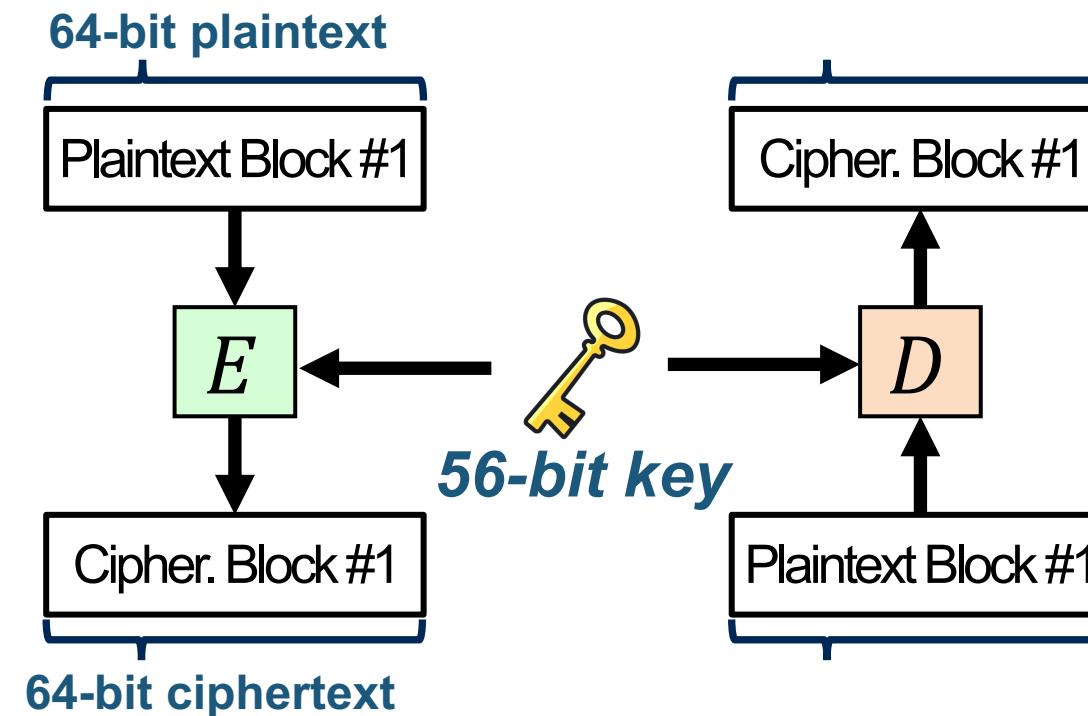
37



Data Encryption Standard (DES)

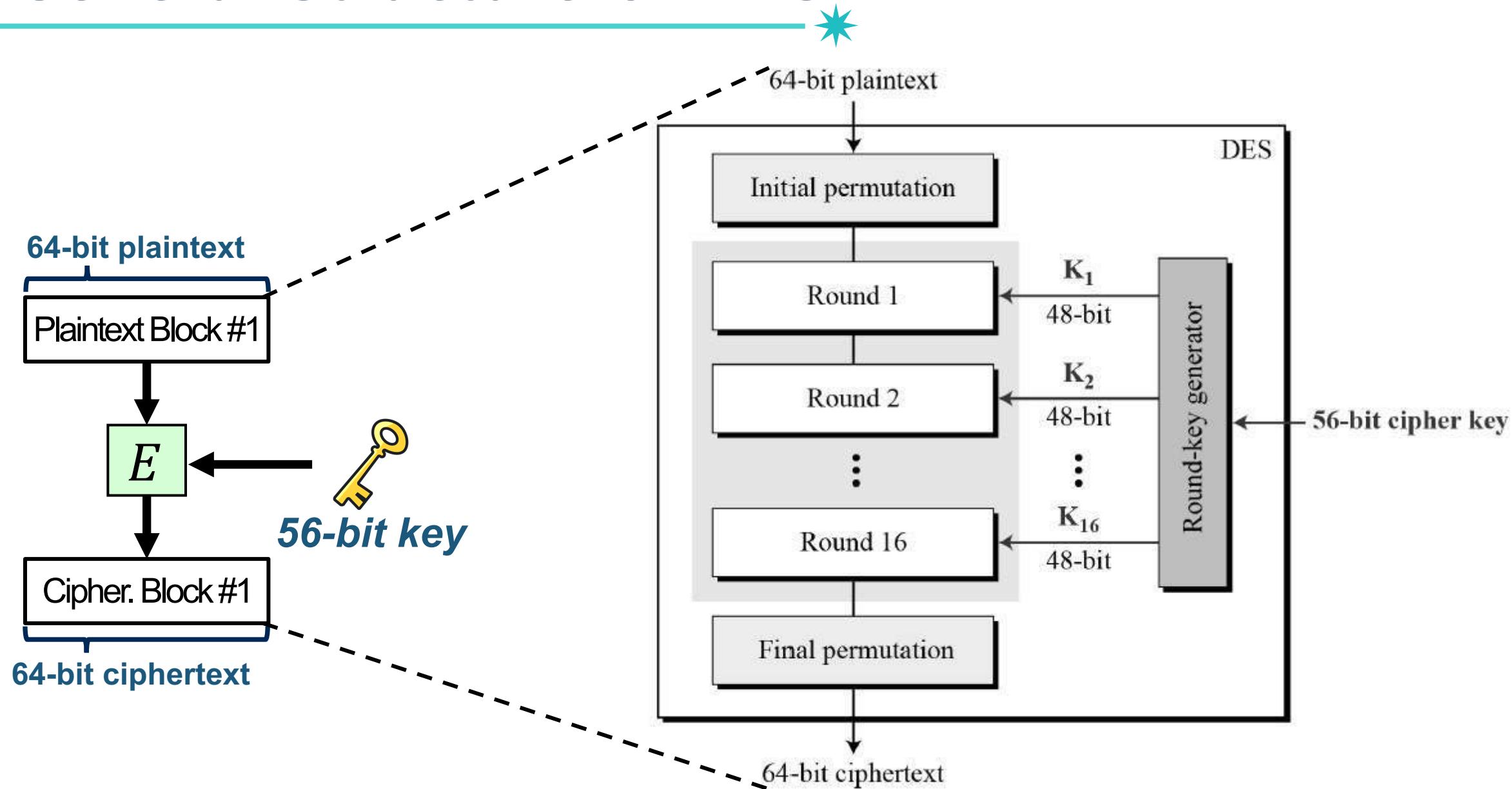
38

- 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



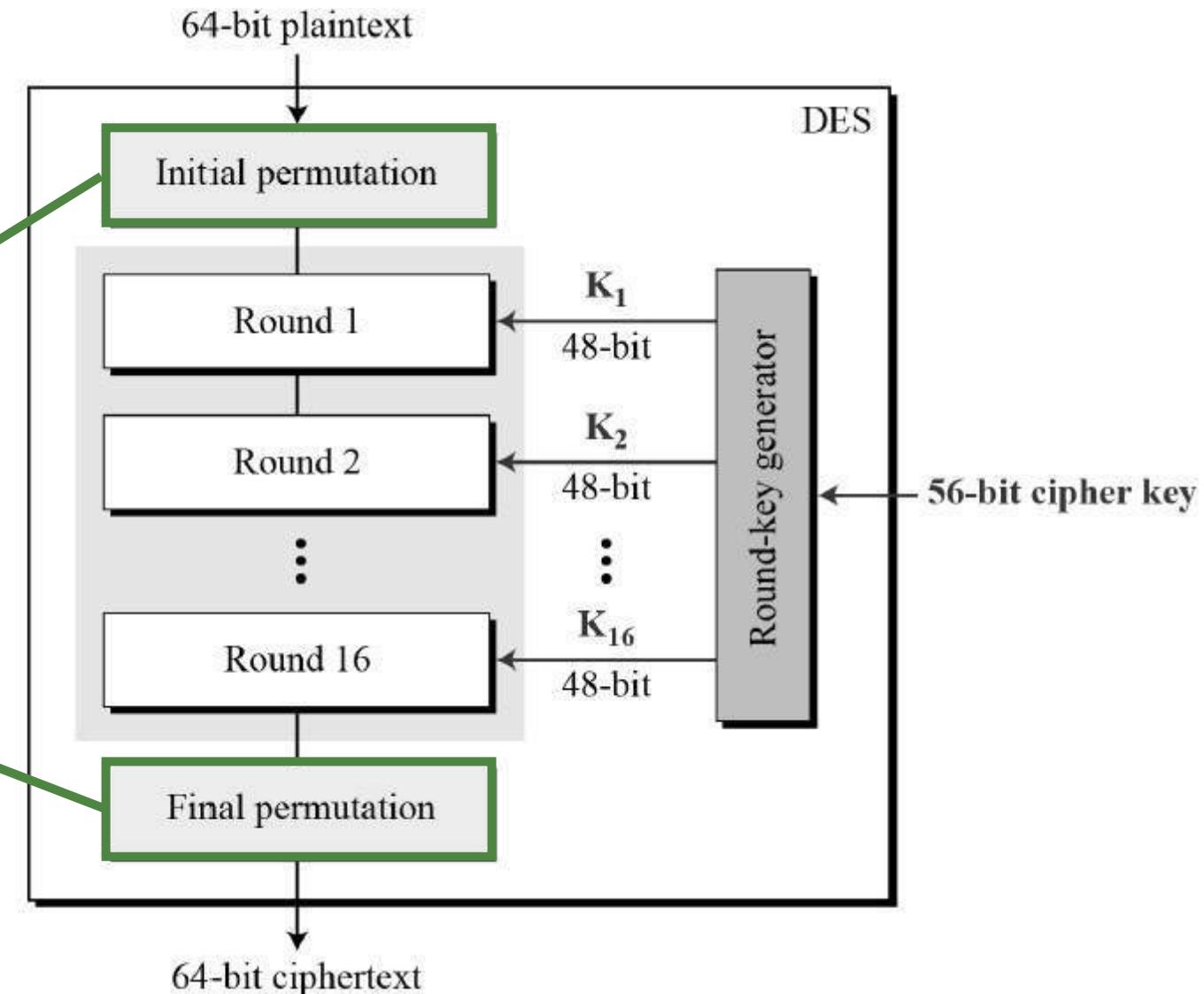
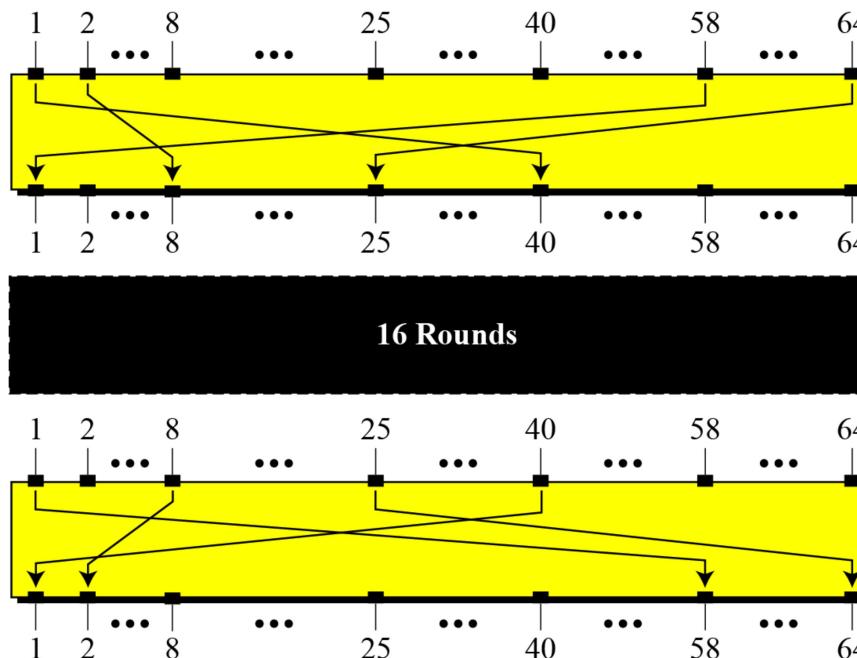
General Structure of DES

39



Initial and Final Permutations (P-Boxes)

40

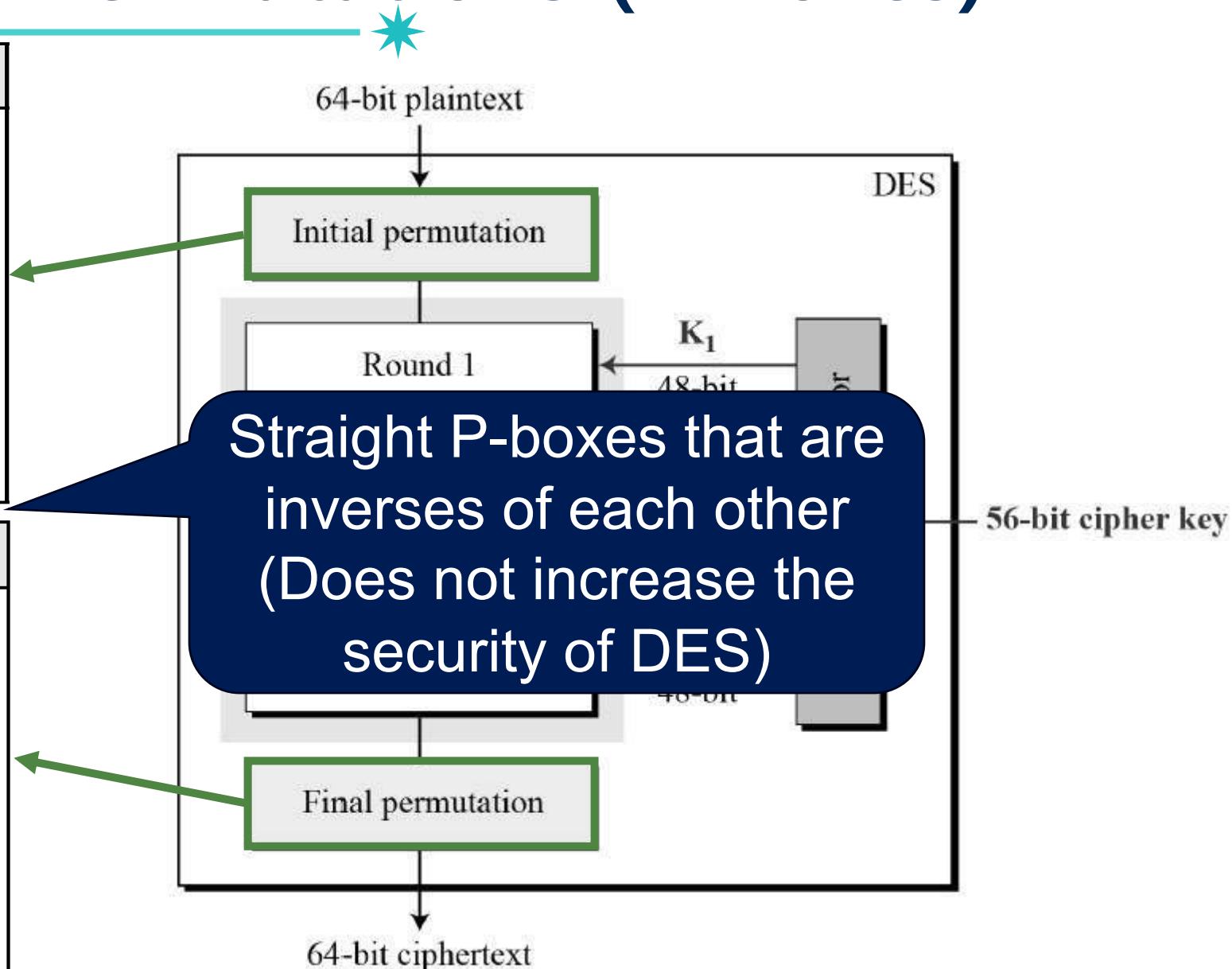


Initial and Final Permutations (P-Boxes)

41

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



Example



- Find the output of the final permutation box when the input is given in hexadecimal as:

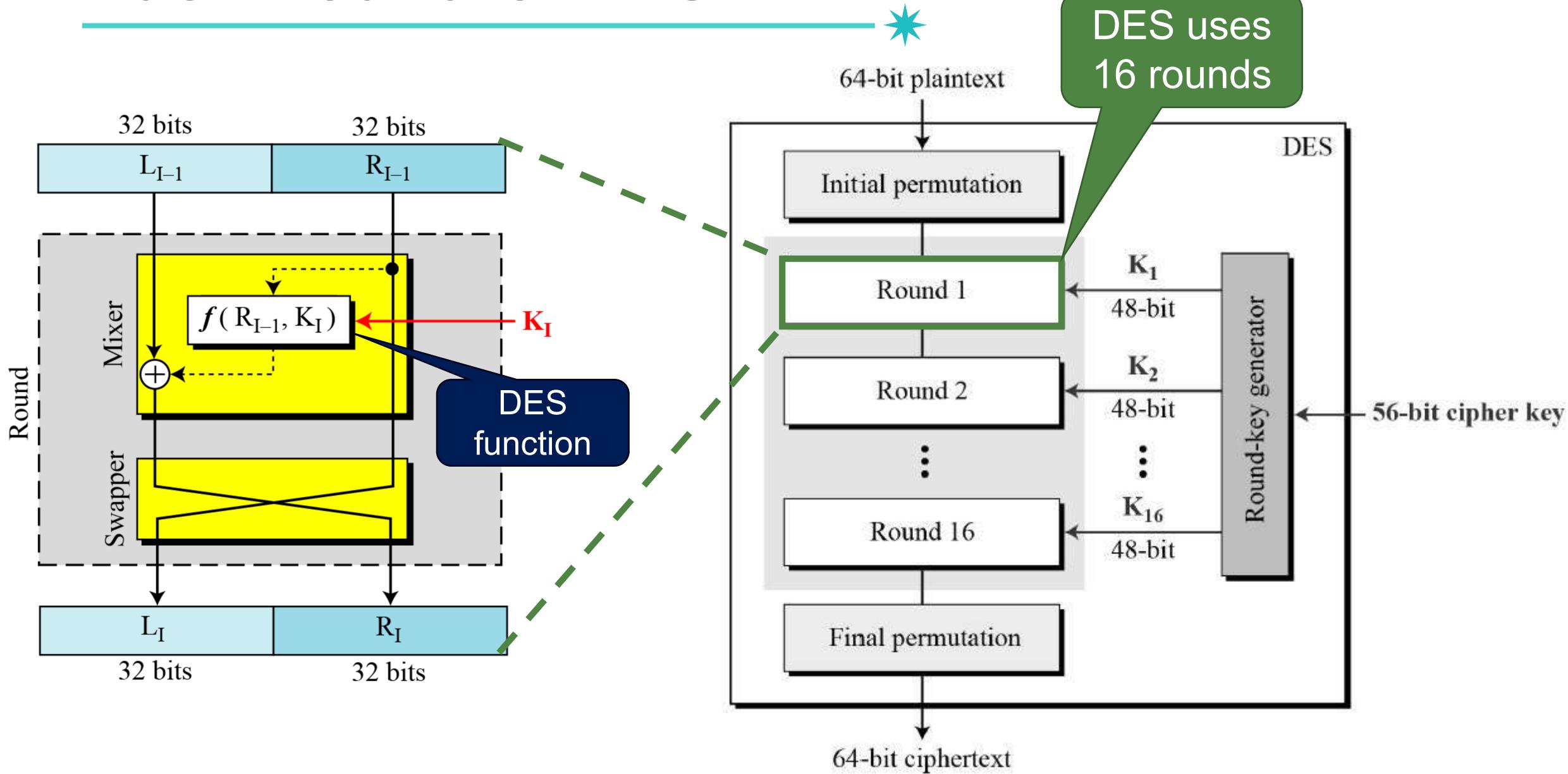
0x0000 0080 0000 0002

- **Solution:**
 - Only bit 25 and bit 63 are 1s; the other bits are 0s. In the final permutation, bit 25 becomes bit 64 and bit 63 becomes bit 15. The result is:

0x0002 0000 0000 0001

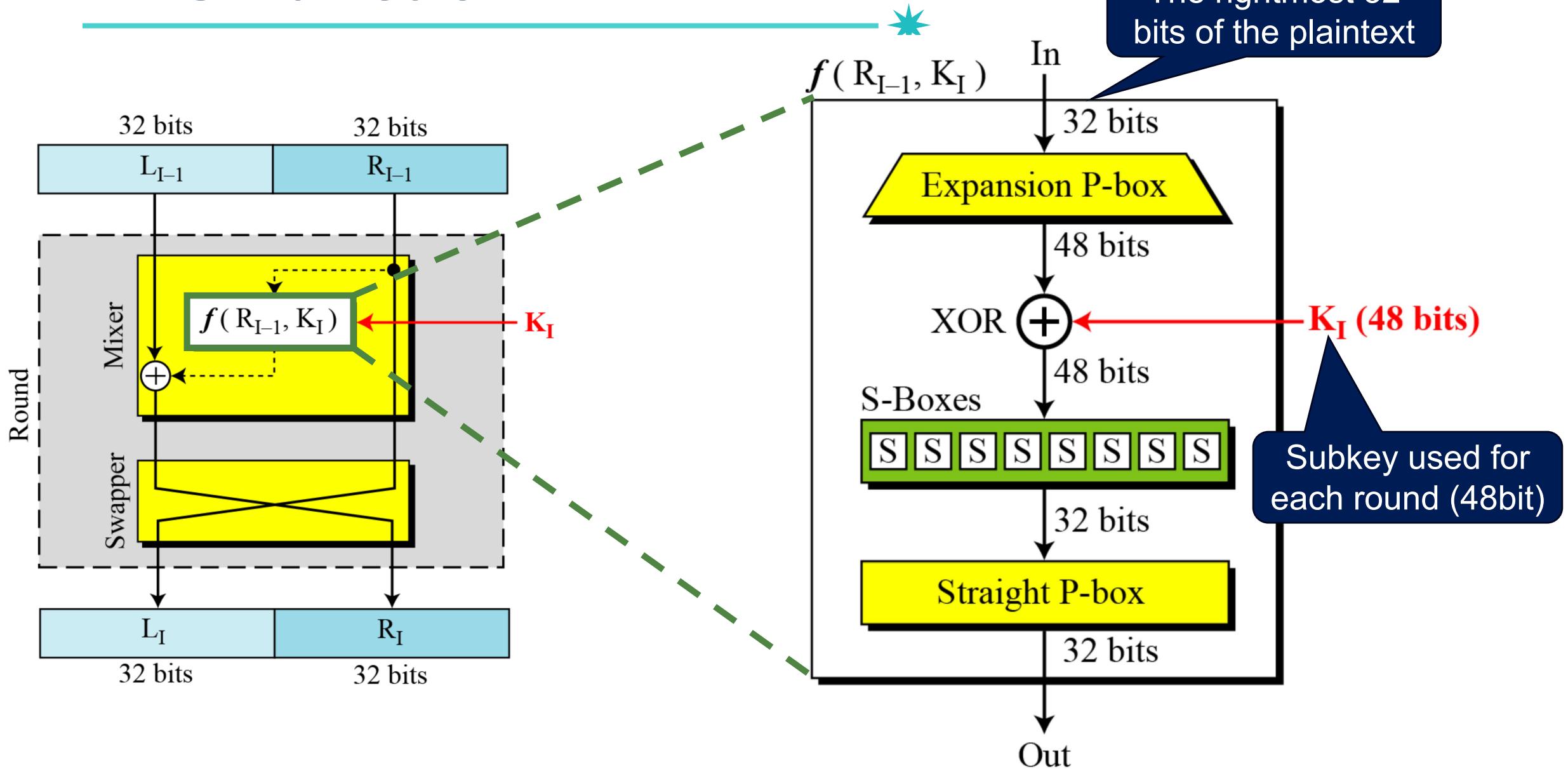
Each Round of DES

43



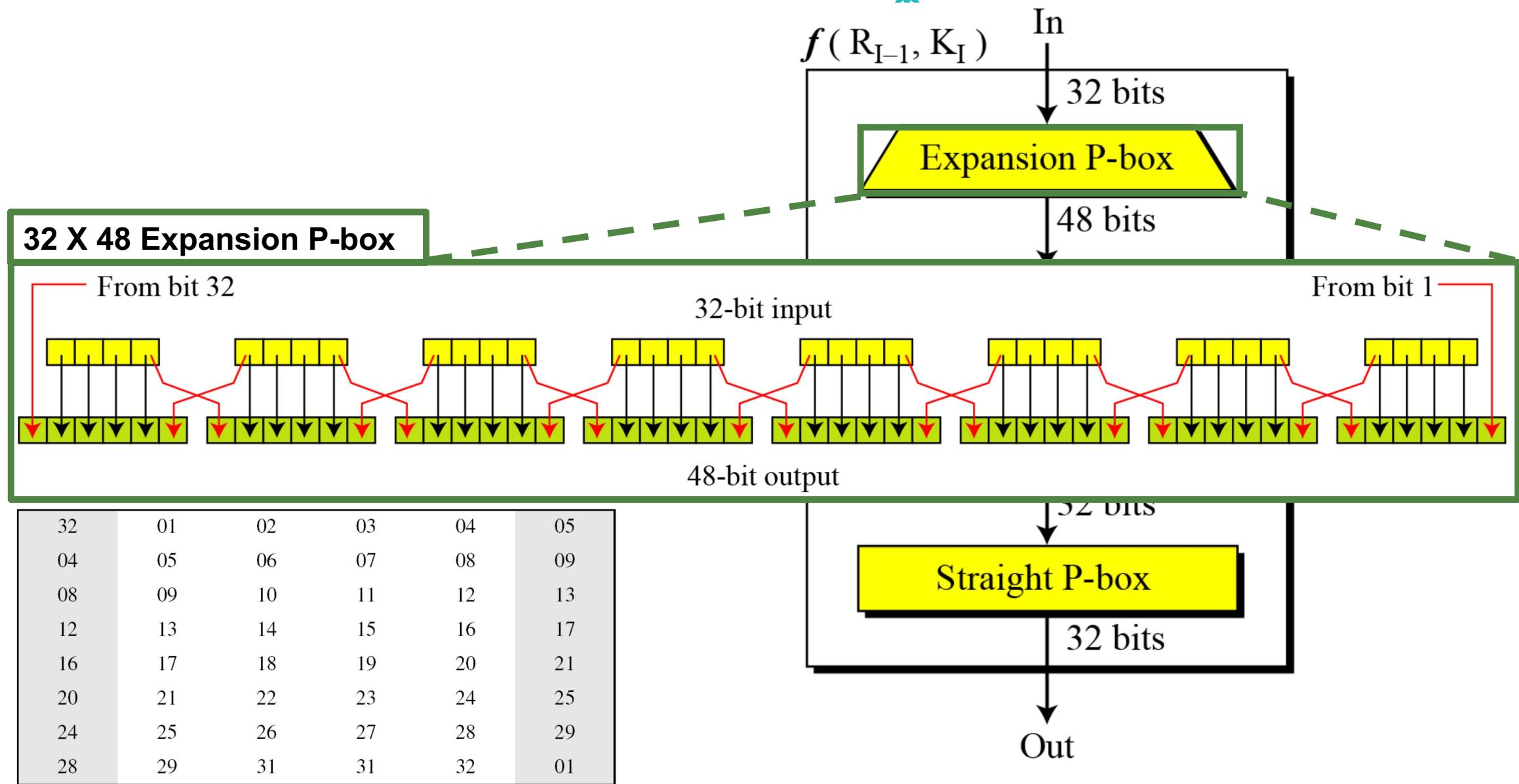
DES Function

44



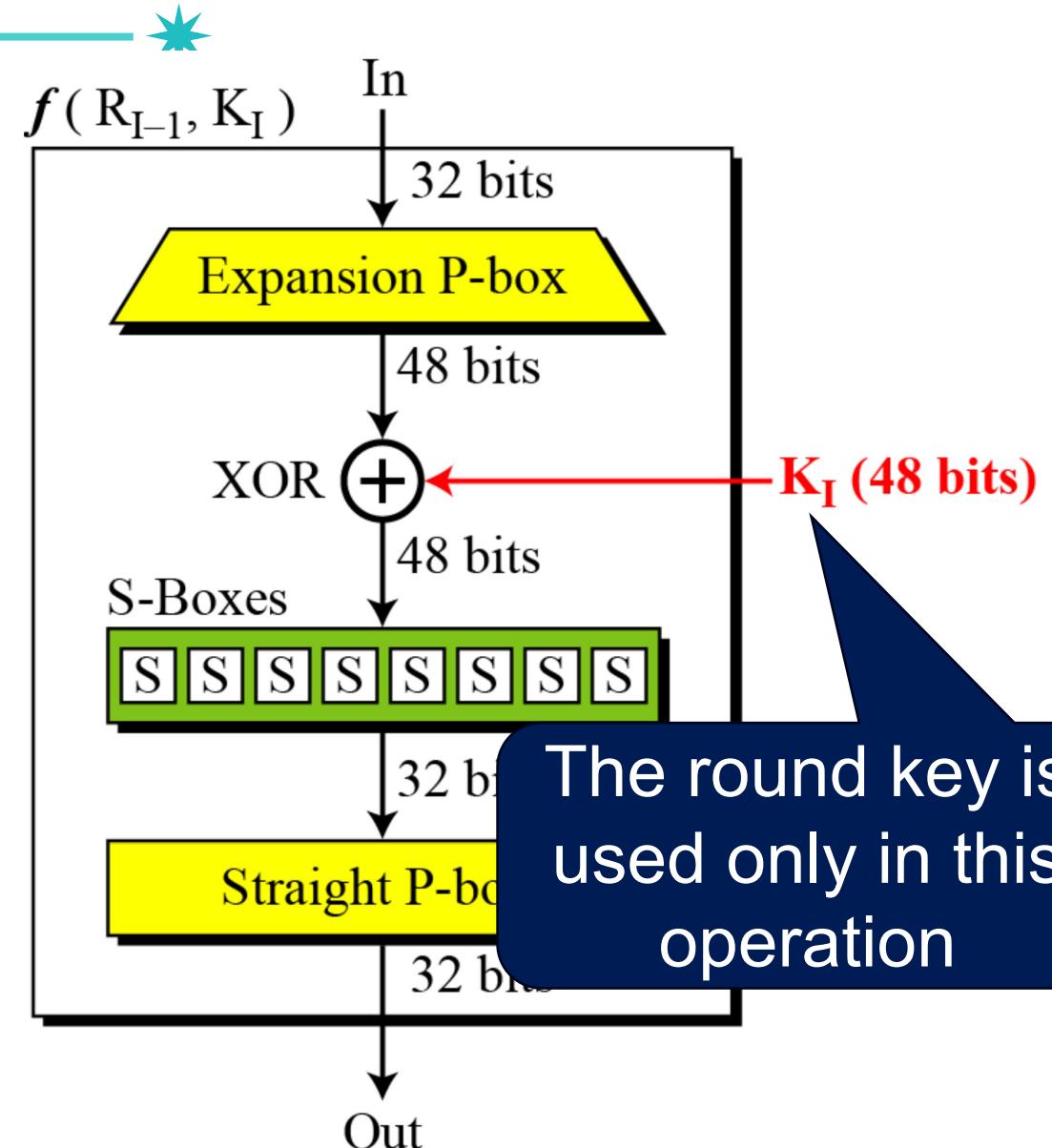
DES Function - Expansion P-box

45



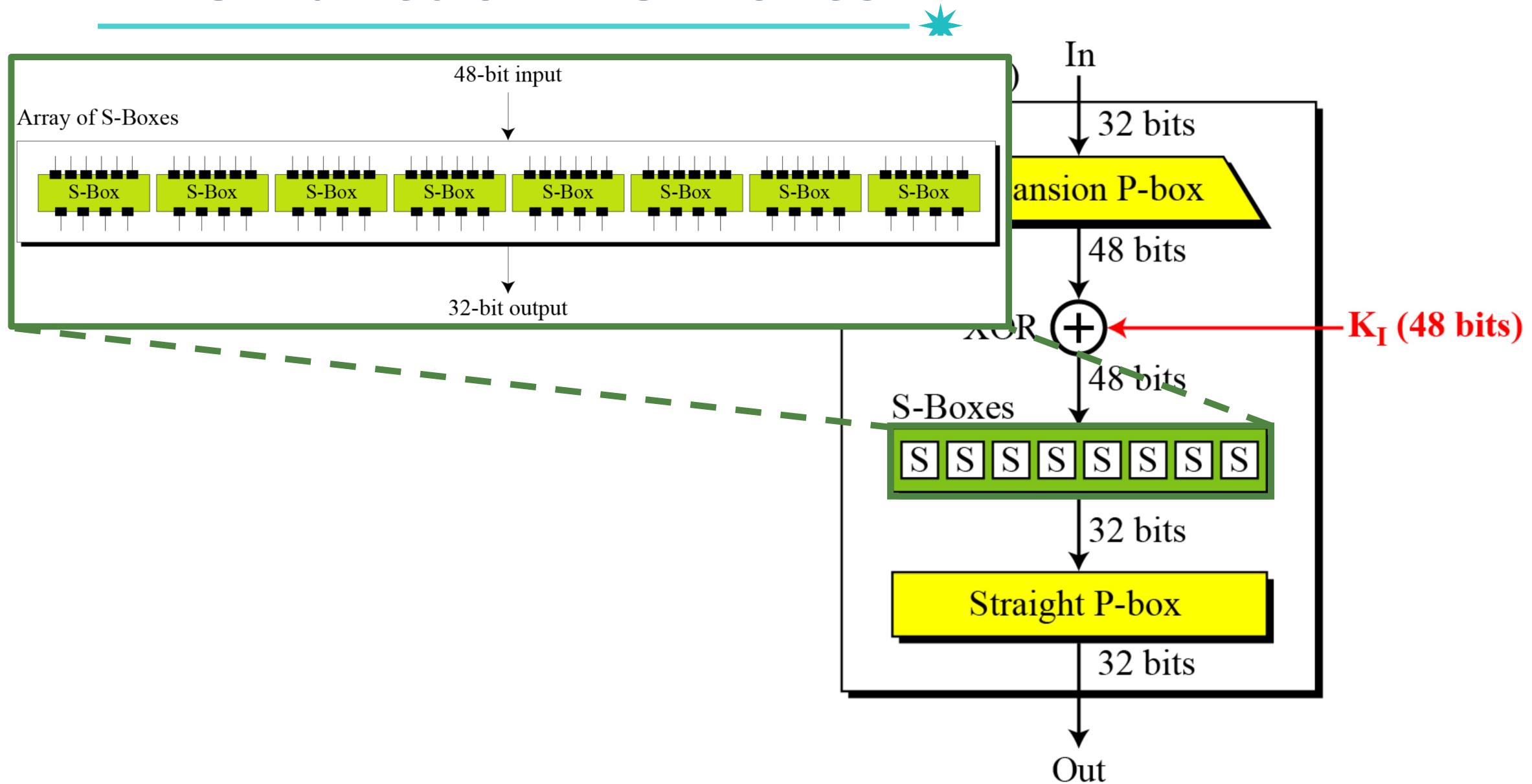
DES Function - XOR

46



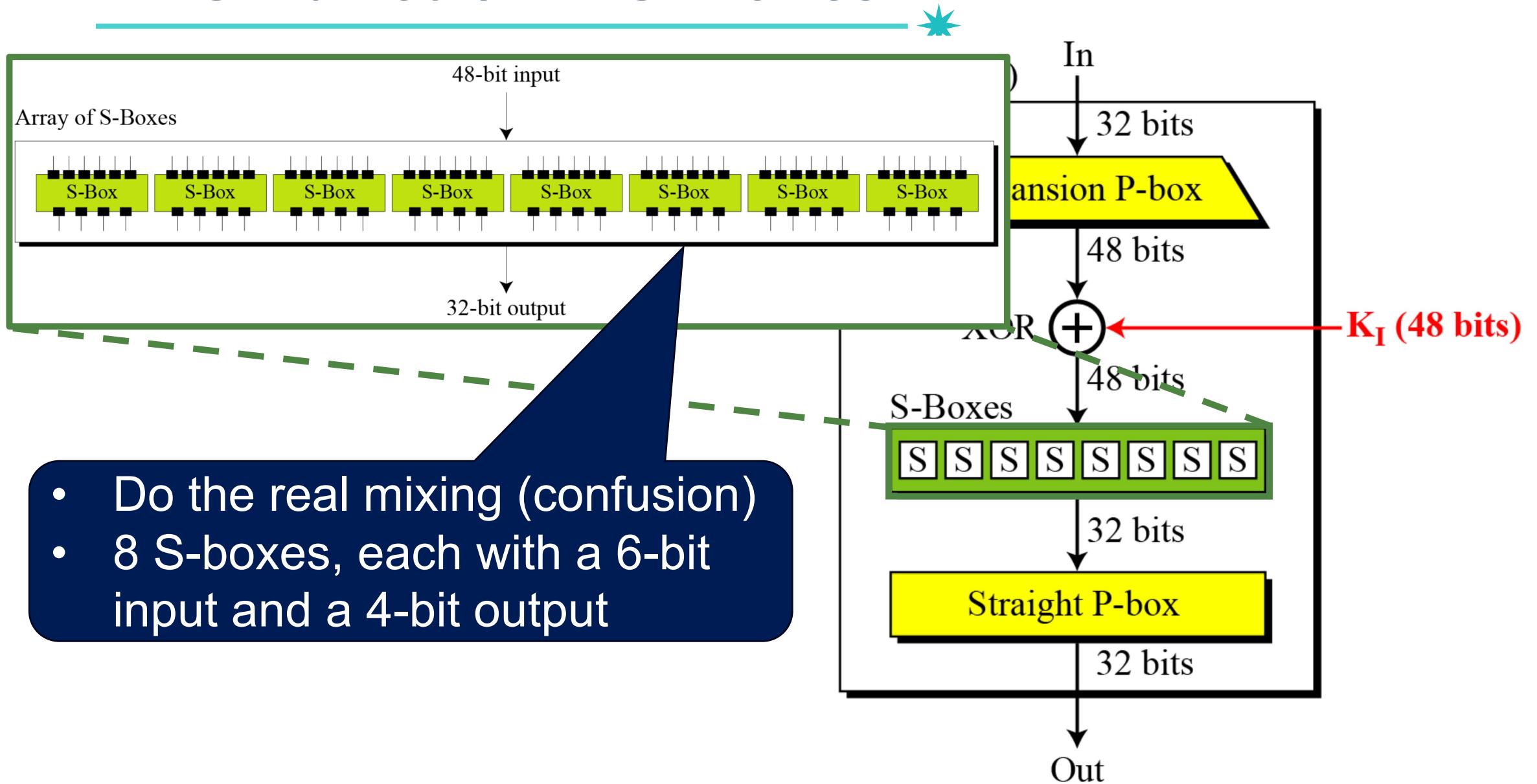
DES Function - S-Boxes

47



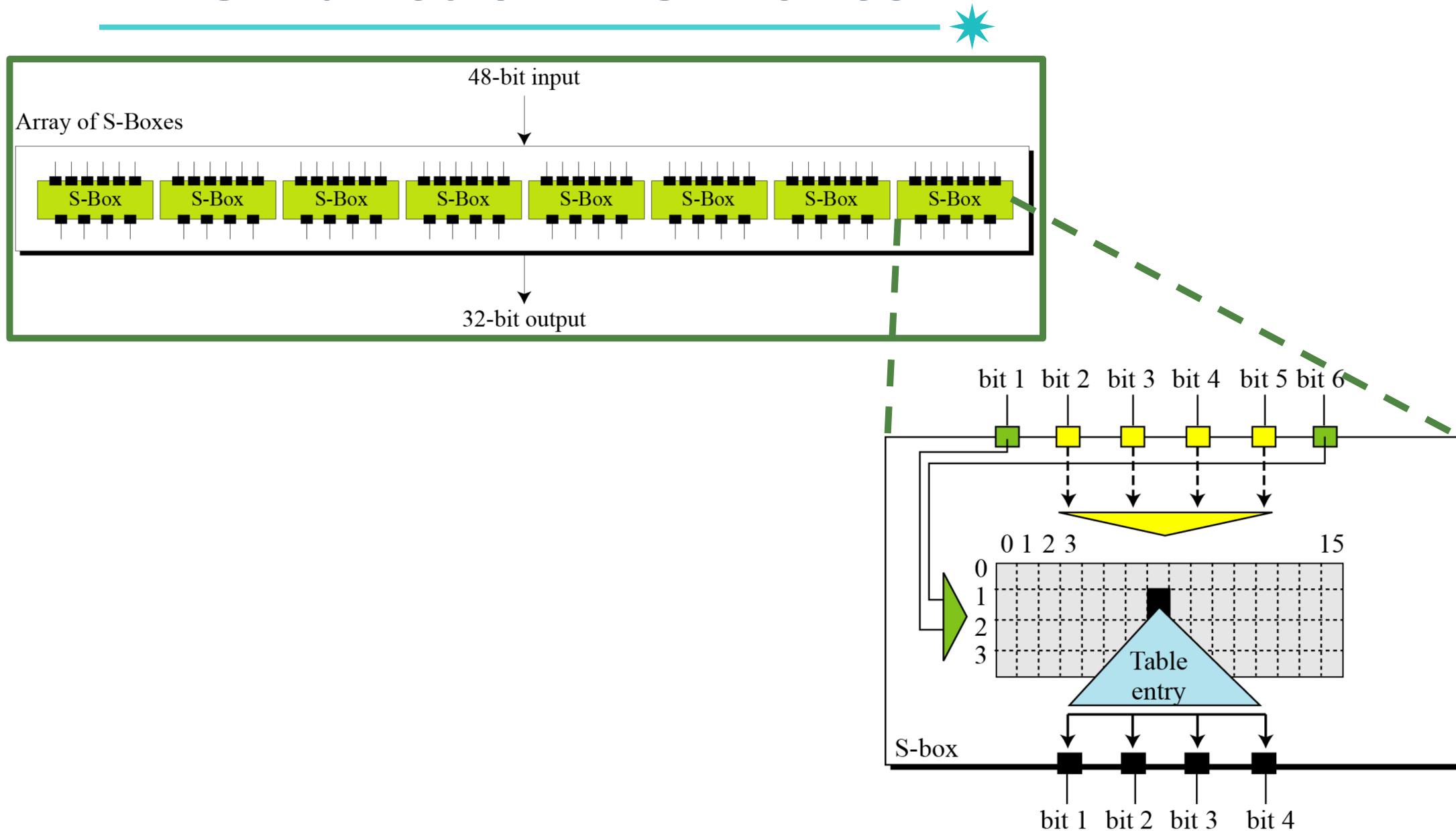
DES Function - S-Boxes

48



DES Function - S-Boxes

49

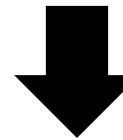


DES Function - S-Boxes

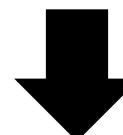
50

- 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

4

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

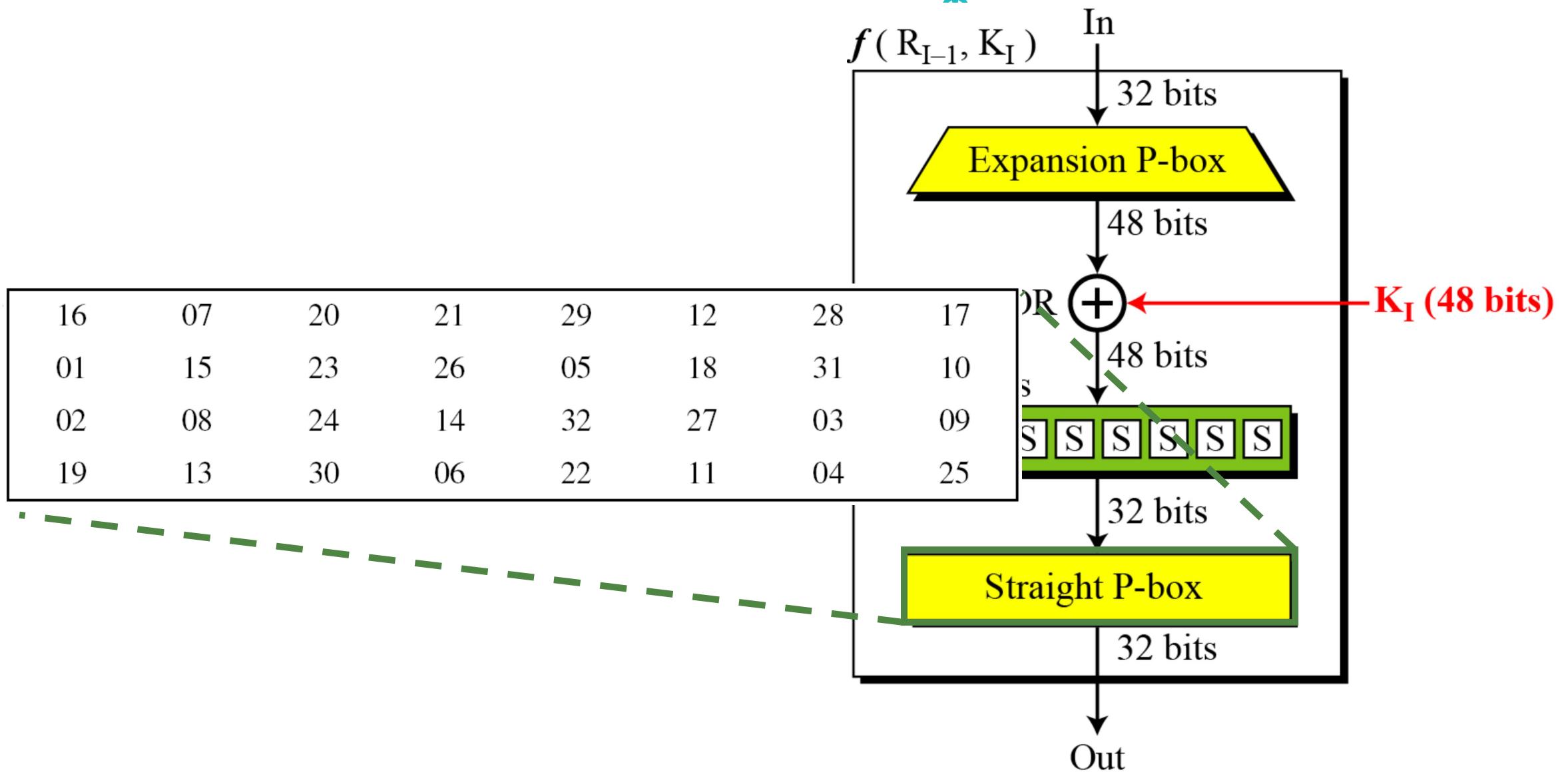
s_6	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

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

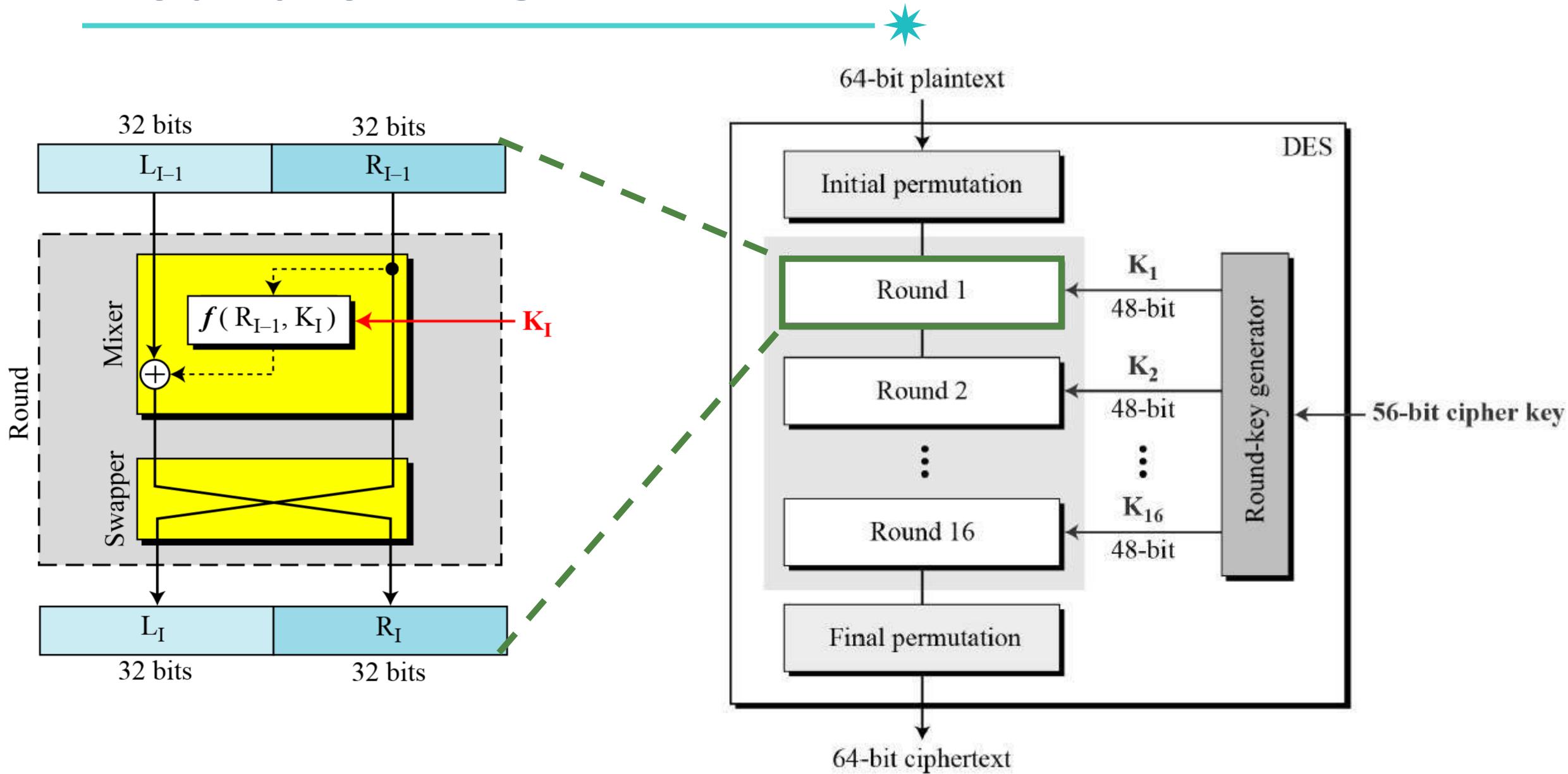
DES Function - Straight P-Box

53



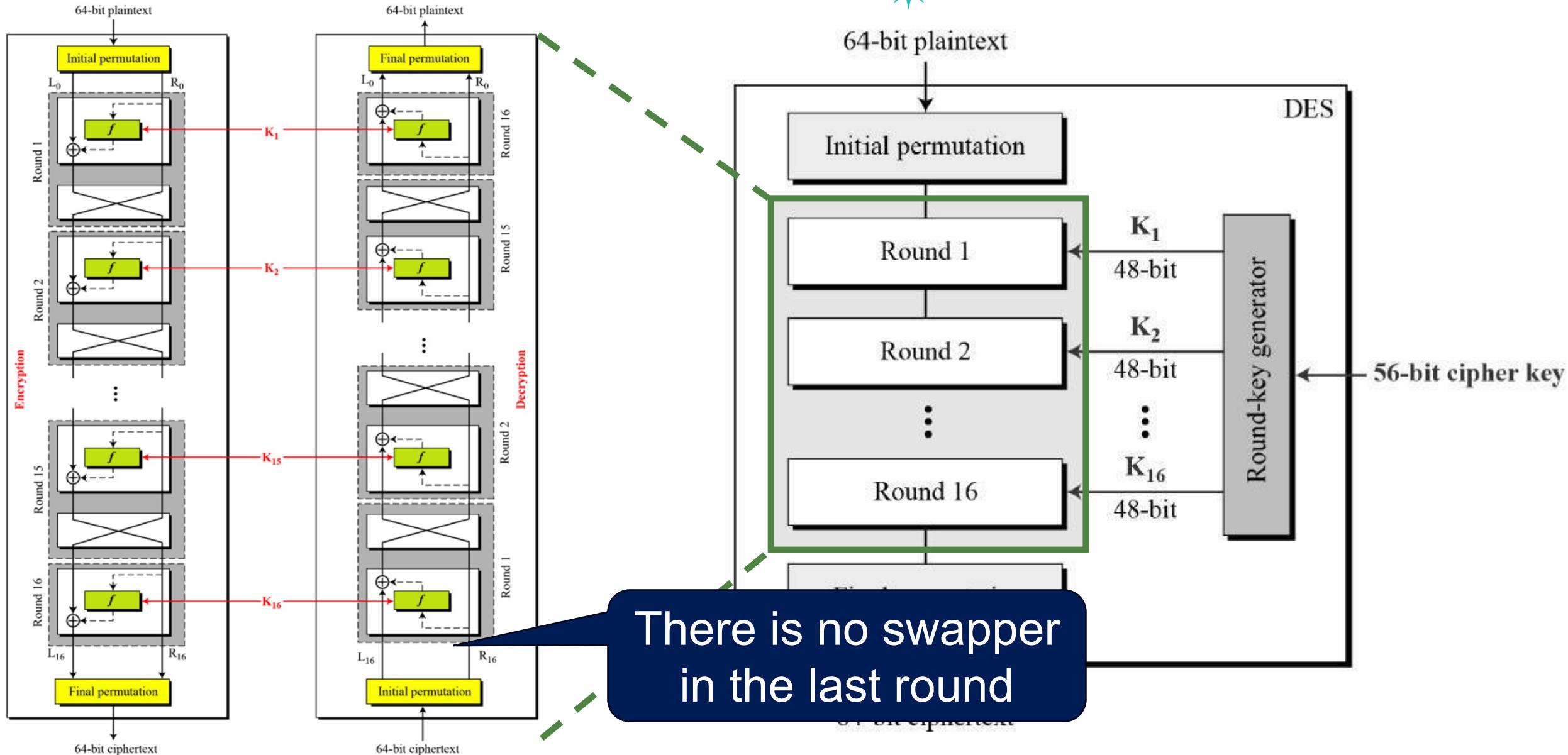
Round of DES

54

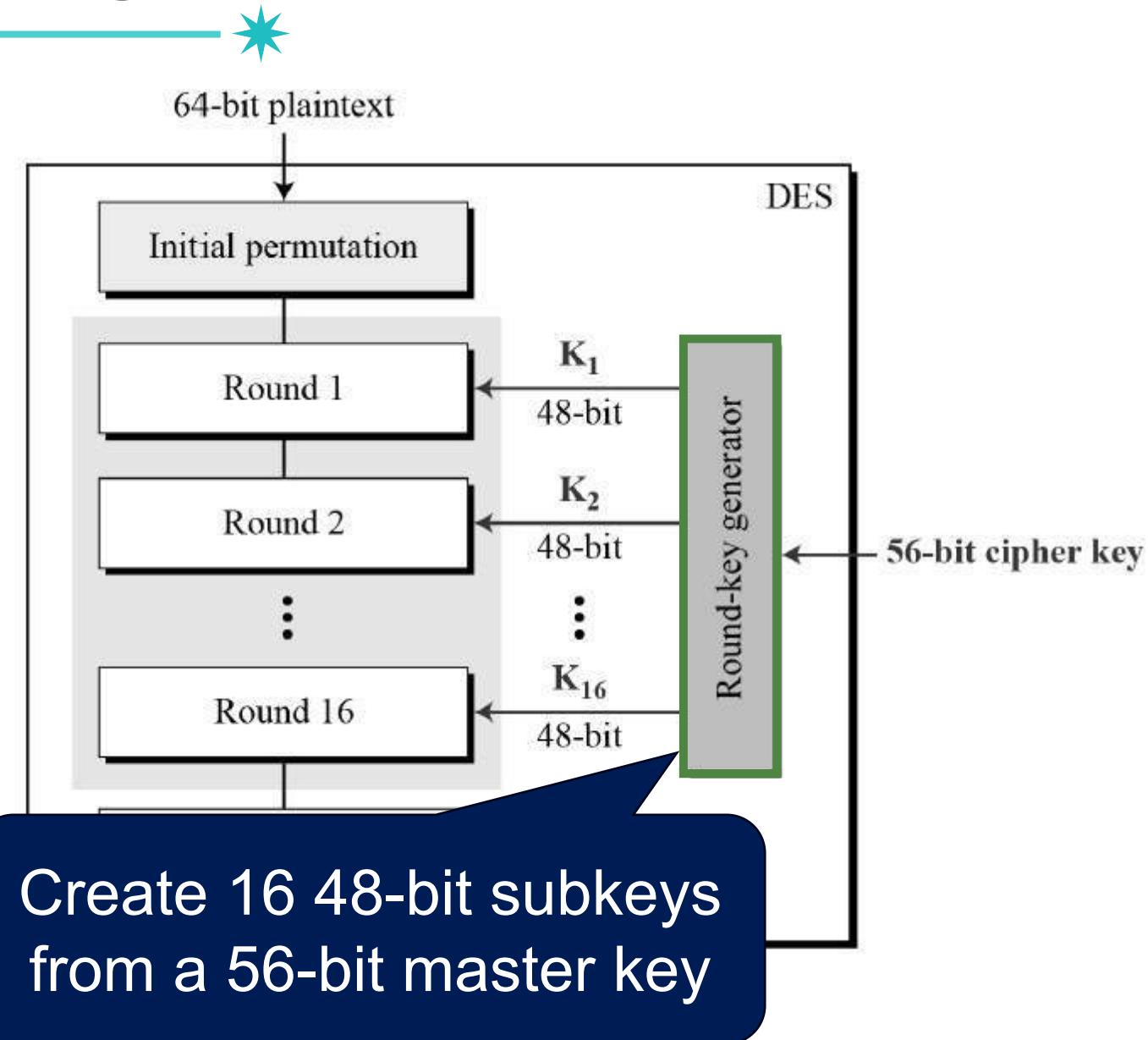


Round Structure of DES

55

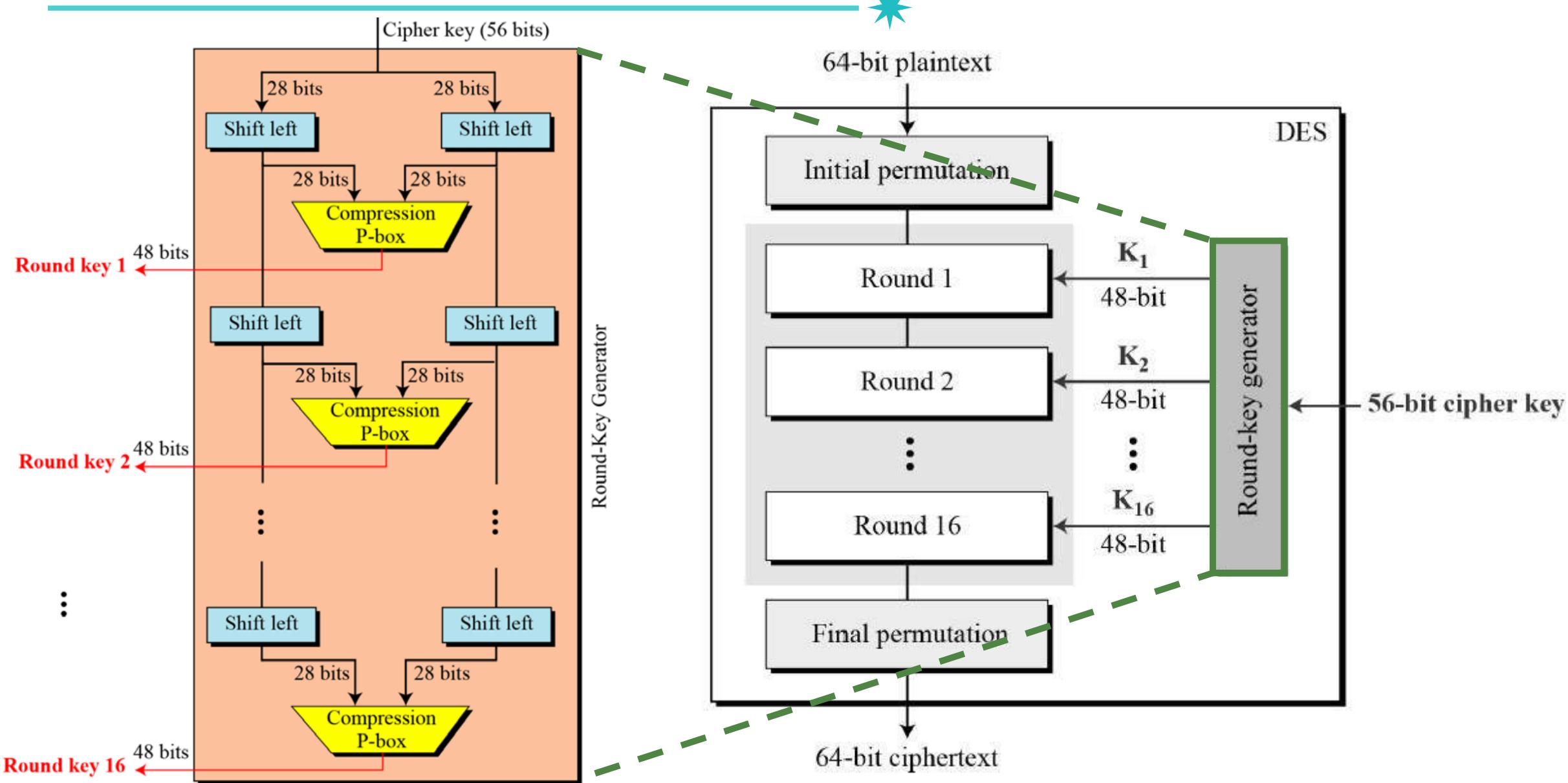


Key Generation in DES



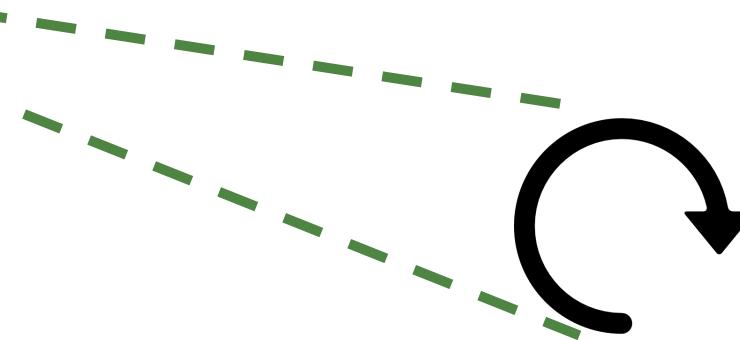
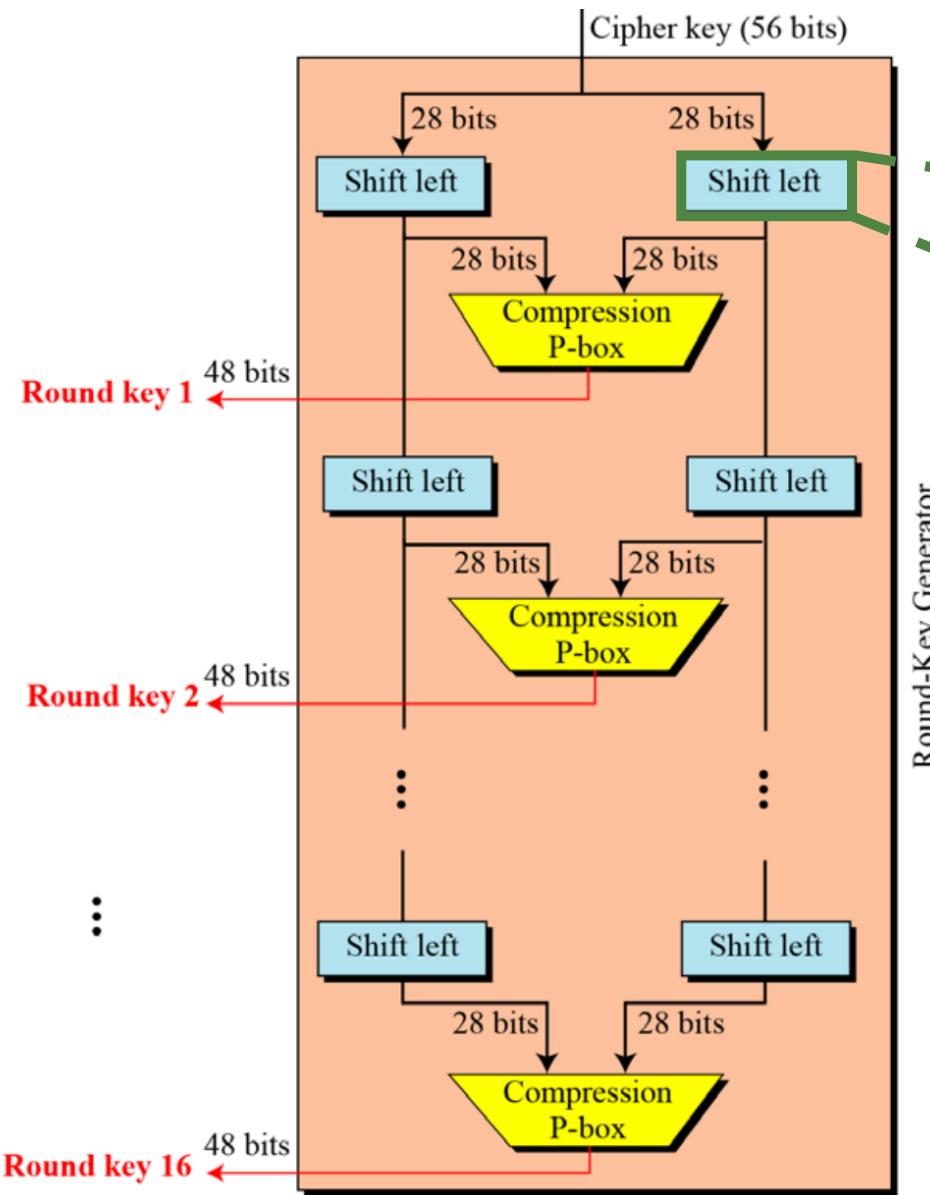
Key Generation in DES

57



Key Generation - Shift Left

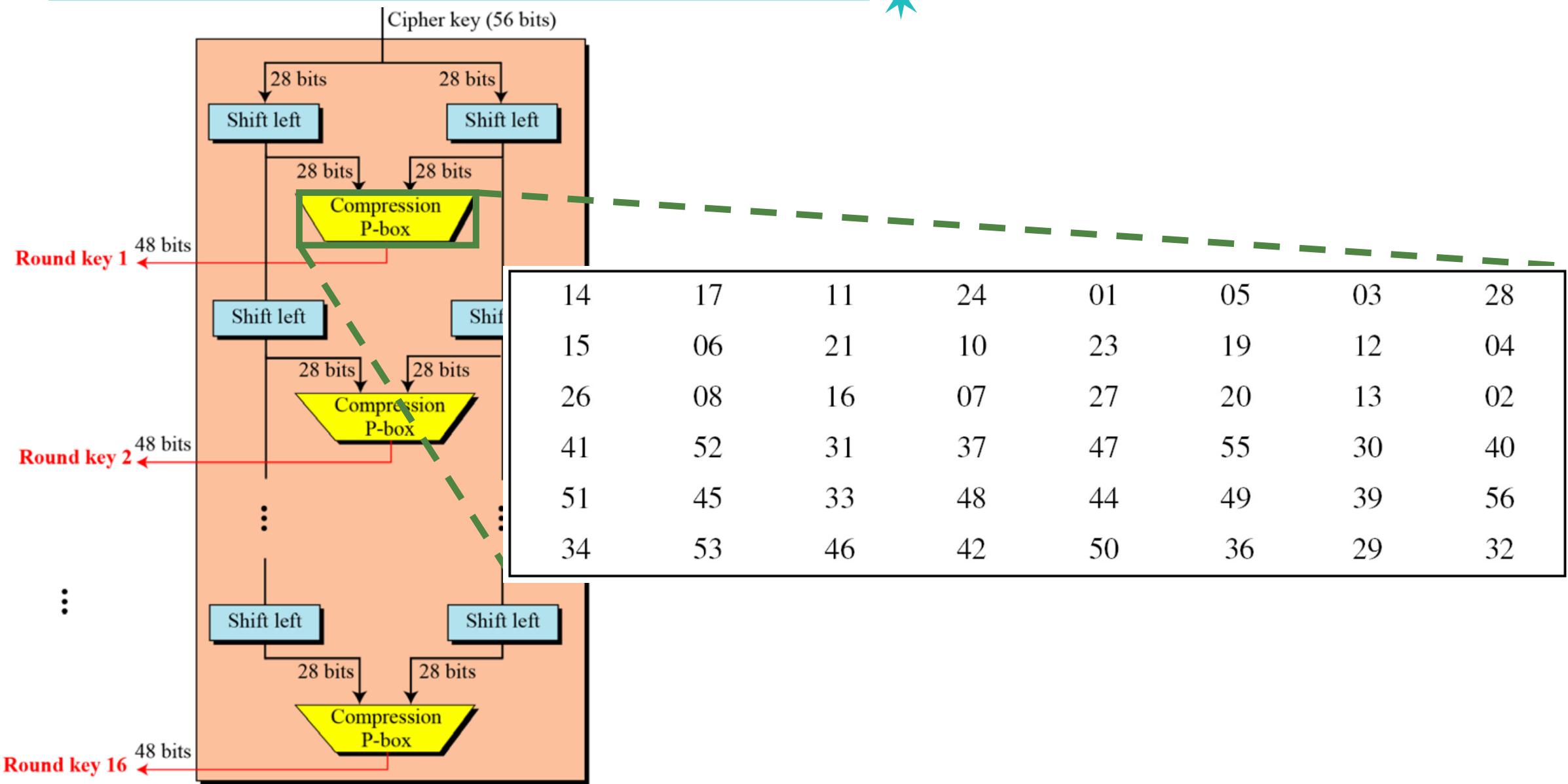
58



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

Key Generation - Compression P-box

59



Example of DES



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

<i>Plaintext:</i> 123456ABCD132536			
<i>After initial permutation:</i> 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

<i>Round 5</i>	236779C2	A15A4B87	69A629FEC913
<i>Round 6</i>	A15A4B87	2E8F9C65	C1948E87475E
<i>Round 7</i>	2E8F9C65	A9FC20A3	708AD2DDB3C0
<i>Round 8</i>	A9FC20A3	308BEE97	34F822F0C66D
<i>Round 9</i>	308BEE97	10AF9D37	84BB4473DCCC
<i>Round 10</i>	10AF9D37	6CA6CB20	02765708B5BF
<i>Round 11</i>	6CA6CB20	FF3C485F	6D5560AF7CA5
<i>Round 12</i>	FF3C485F	22A5963B	C2C1E96A4BF3
<i>Round 13</i>	22A5963B	387CCDAA	99C31397C91F
<i>Round 14</i>	387CCDAA	BD2DD2AB	251B8BC717D0
<i>Round 15</i>	BD2DD2AB	CF26B472	3330C5D9A36D
<i>Round 16</i>	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

<i>Ciphertext:</i> C0B7A8D05F3A829C			
<i>After initial permutation:</i> 19BA9212CF26B472			
<i>After splitting:</i> L ₀ =19BA9212 R ₀ =CF26B472			
<i>Round</i>	<i>Left</i>	<i>Right</i>	<i>Round Key</i>
<i>Round 1</i>	CF26B472	BD2DD2AB	181C5D75C66D
<i>Round 2</i>	BD2DD2AB	387CCDAA	3330C5D9A36D
...
<i>Round 15</i>	5A78E394	18CA18AD	4568581ABCCE
<i>Round 16</i>	14A7D678	18CA18AD	194CD072DE8C
<i>After combination:</i> 14A7D67818CA18AD			
<i>Plaintext:</i> 123456ABCD132536	<i>(after final permutation)</i>		

DES Weakness



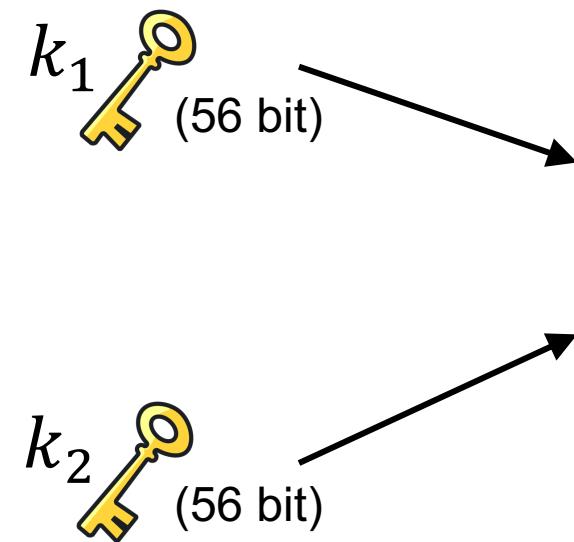
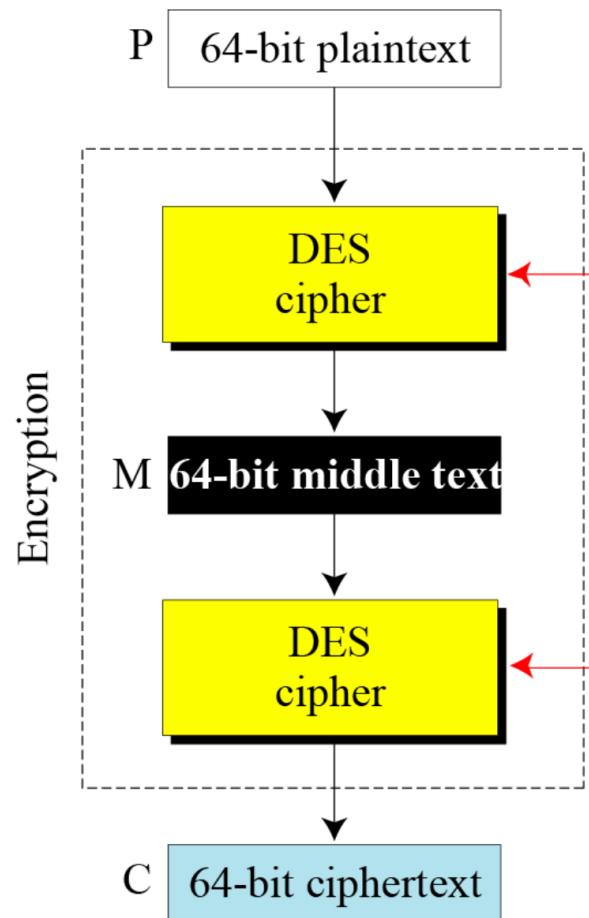
- The most serious weakness of DES is in its key size
- The adversary needs to check 2^{56} keys
 - If we make one million processor computer, it will take about 20 hours
 - A special computer was built in 1998 that found the key in 112 hours

How do we make it robust?

=> *Let's use double-DES on each block*

Double-DES?

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

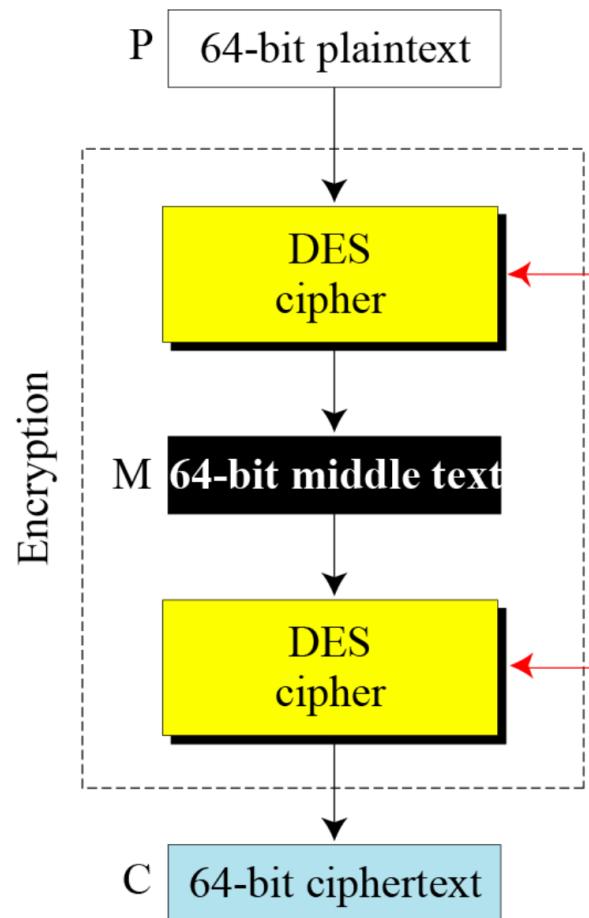


The adversary needs
to check 2^{112} keys!
What is the problem?

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

Meet-in-the-Middle-Attack for Double DES⁶⁵

- $C = E_{K_2}(E_{K_1}(PB))$
- Vulnerable to “meet-in-the-middle” attack



k_1 (56 bit)
 k_2 (56 bit)

64-bit plaintext

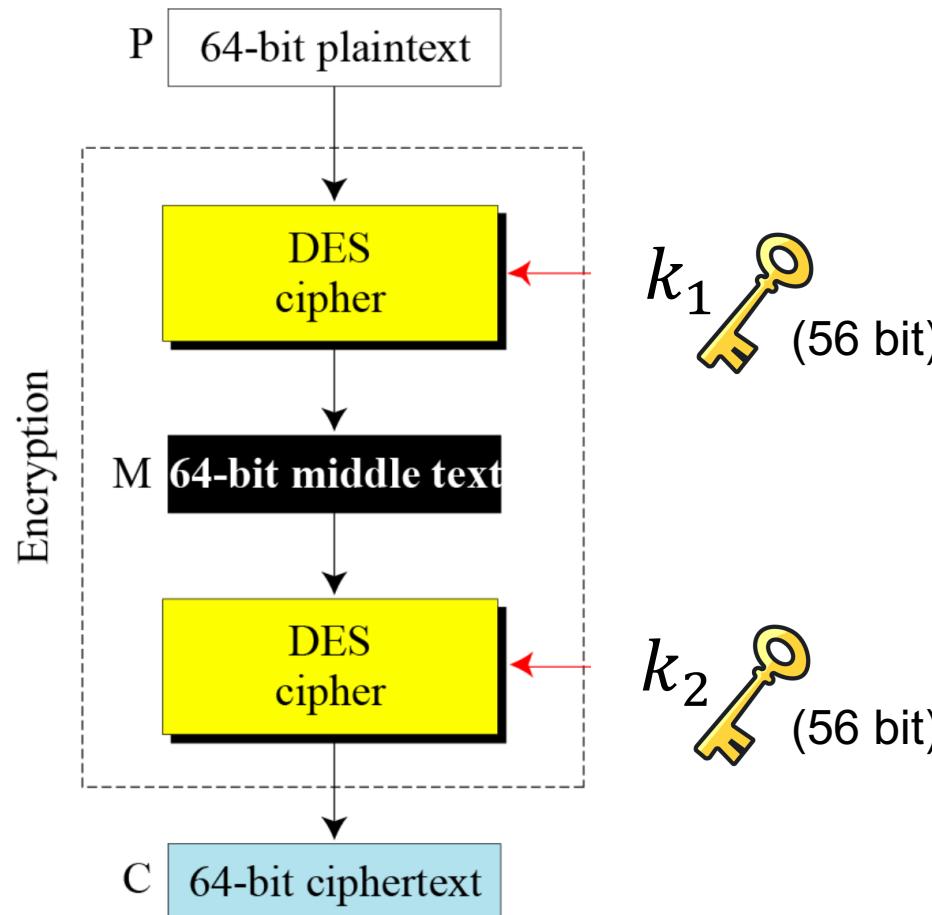
64-bit ciphertext

The attacker
got this pair...

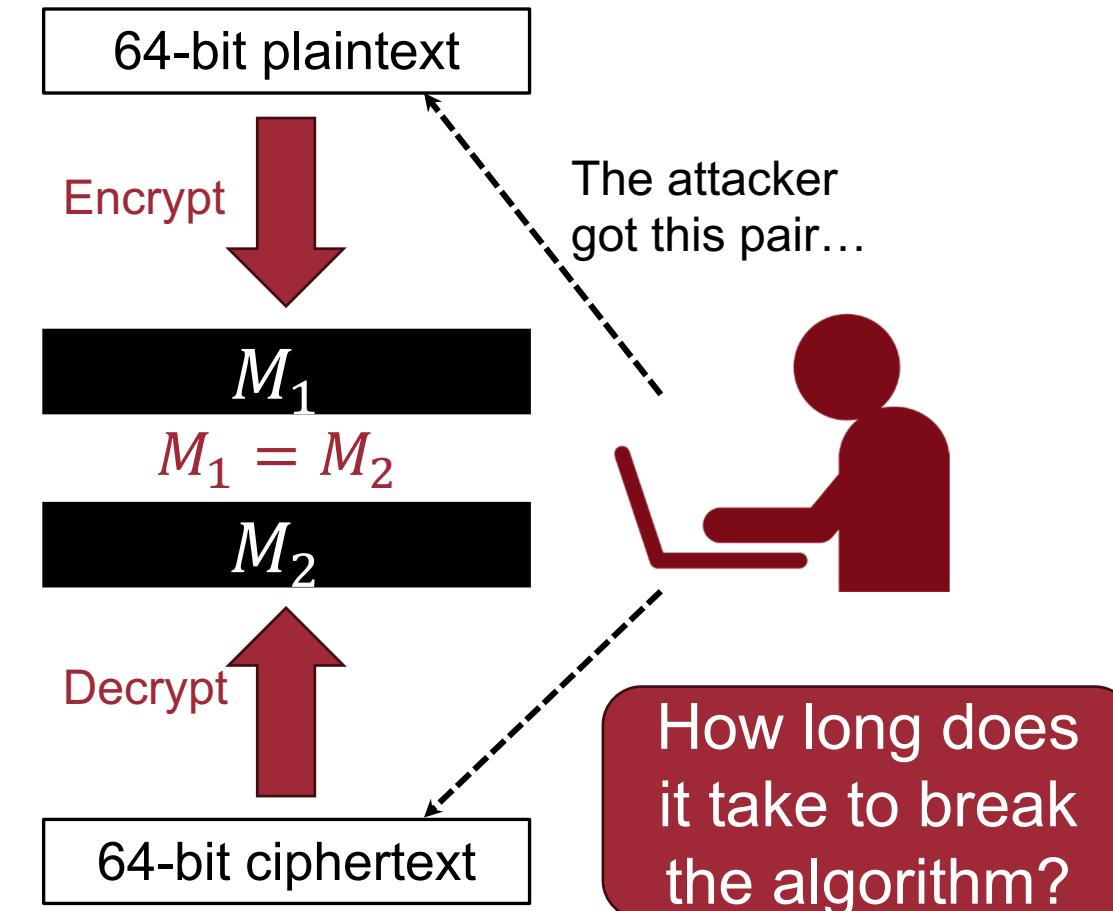


Meet-in-the-Middle-Attack for Double DES⁶⁶

- $C = E_{K_2}(E_{K_1}(PB))$
- Vulnerable to “meet-in-the-middle” attack

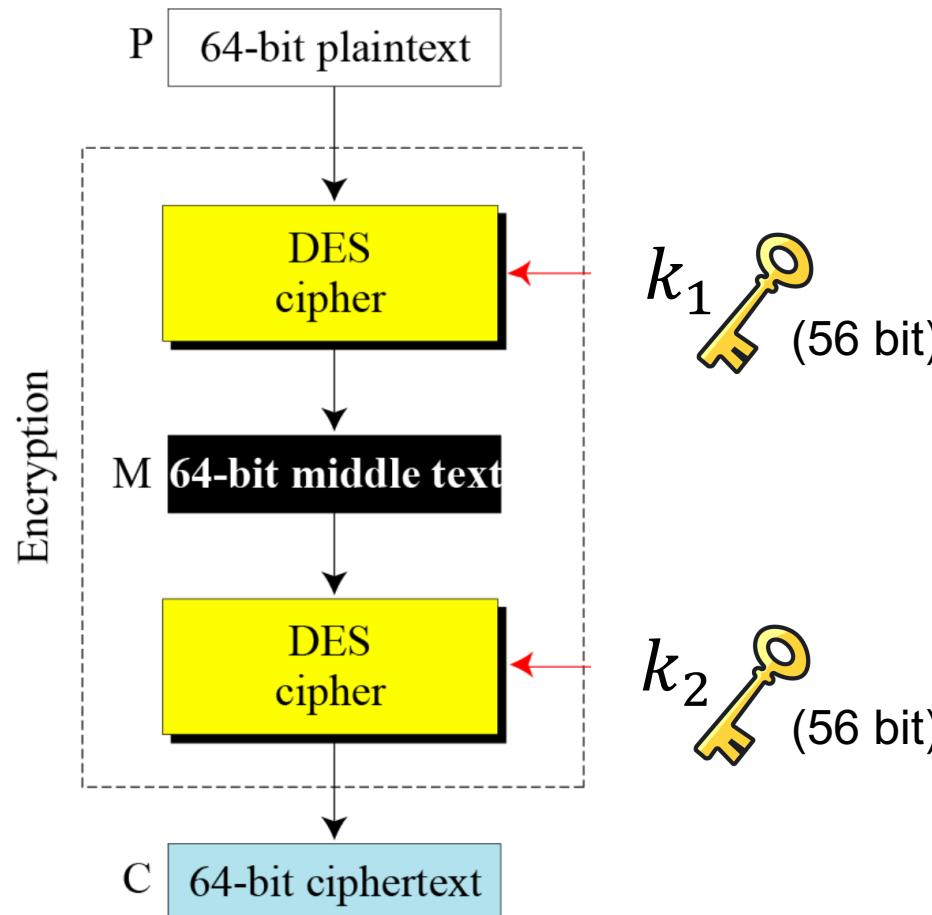


k_1 (56 bit)
 k_2 (56 bit)

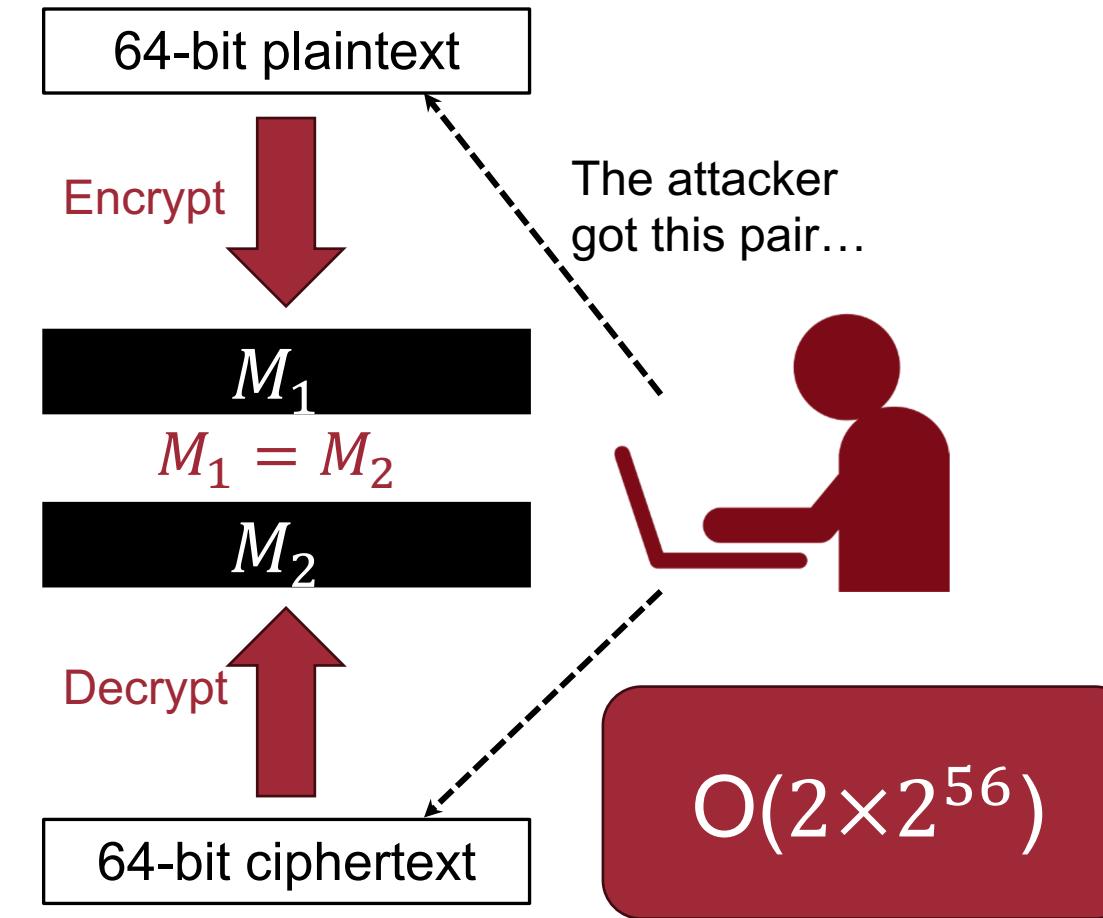


Meet-in-the-Middle-Attack for Double DES⁶⁷

- $C = E_{K_2}(E_{K_1}(PB))$
- Vulnerable to “meet-in-the-middle” attack



k_1 (56 bit)
 k_2 (56 bit)

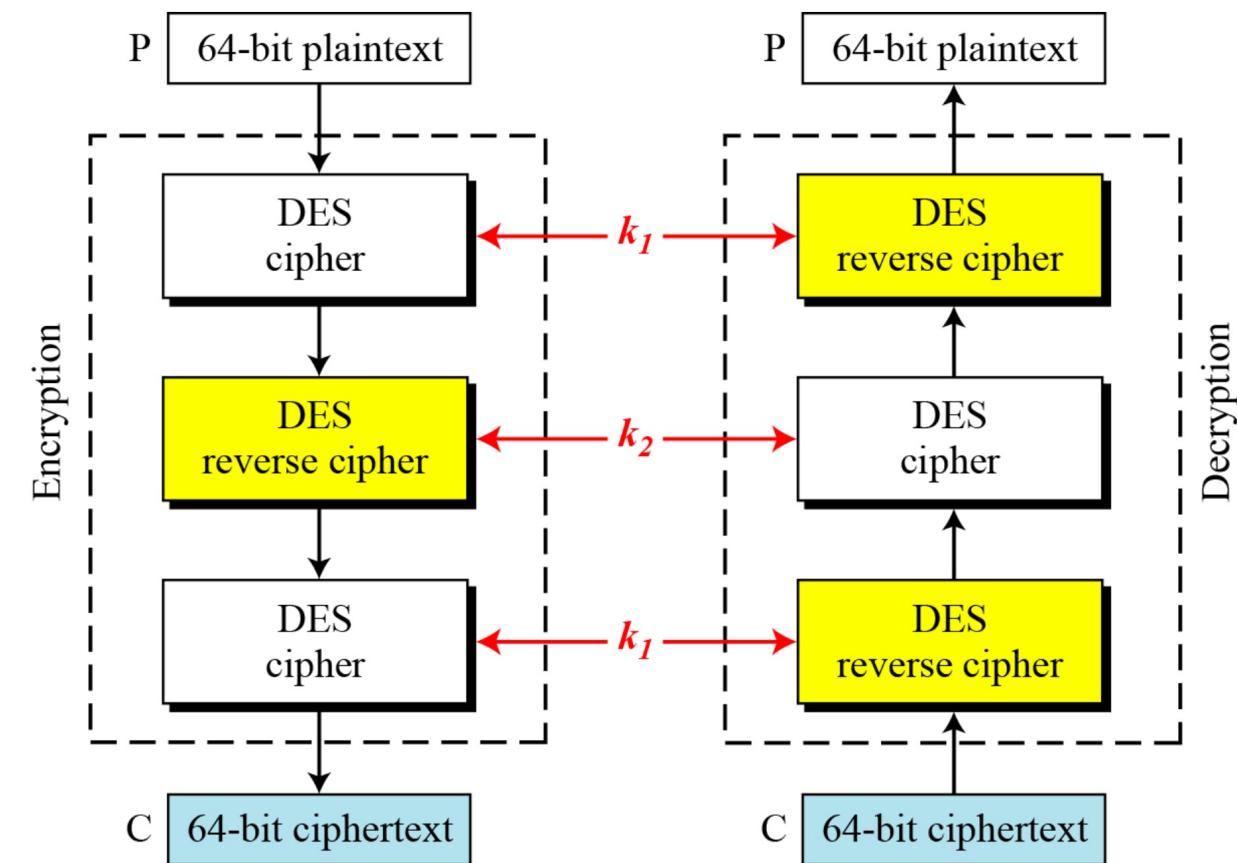


Triple-DES with Two-Keys

68

- Use three encryptions
- But we can use two keys with E-D-E sequence

$$- C = E_{K_1}(D_{K_2}(E_{K_1}(PB)))$$



Triple-DES with Two-Keys

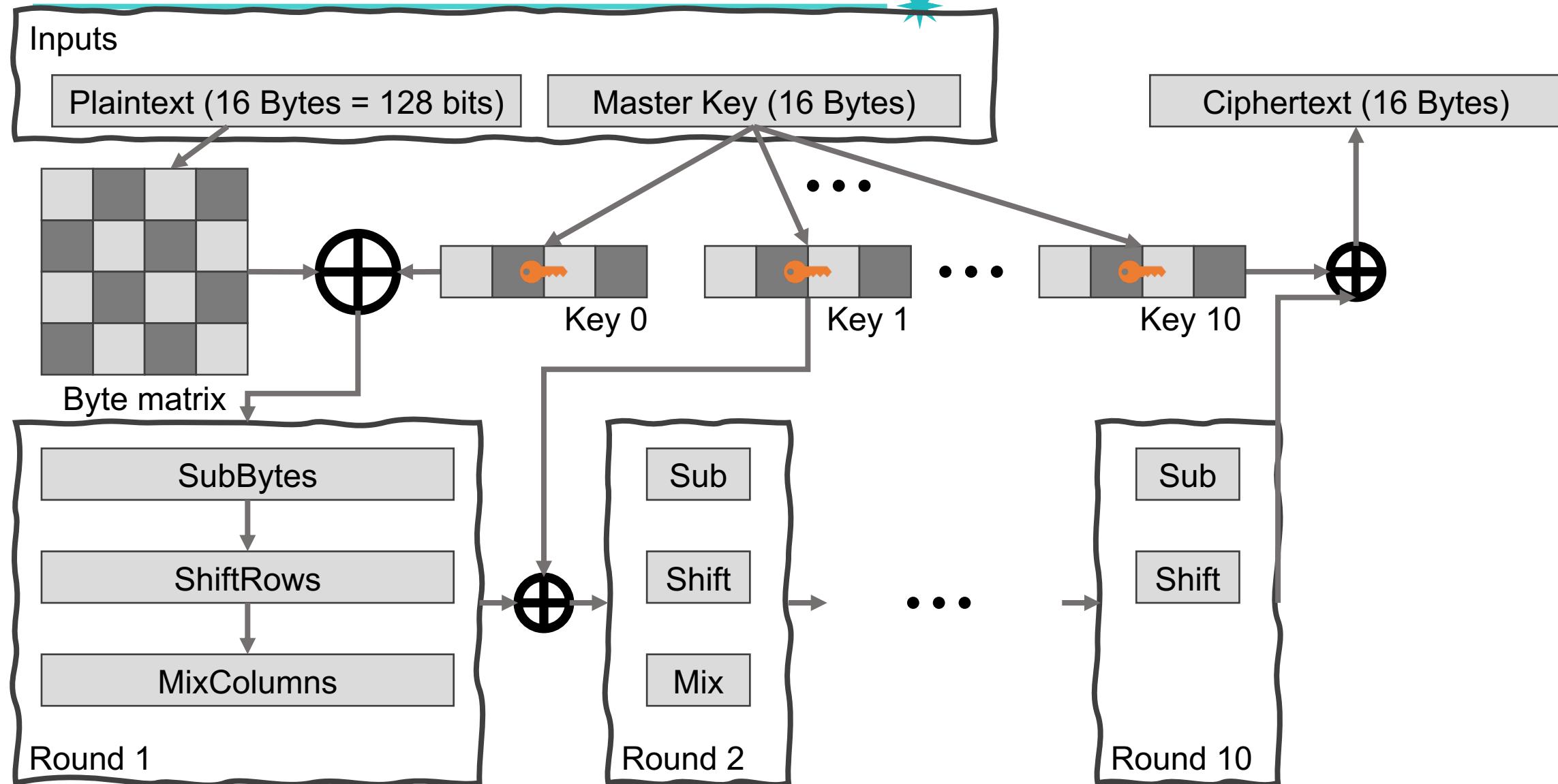
- Use three encryptions
- But we can use two keys with E-D-E sequence
 - $C = E_{K_1}(D_{K_2}(E_{K_1}(PB)))$
- Standardized in ANSI X9.17 & ISO8732
- No current known practical attacks
- Has been adopted by some Internet applications, e.g., PGP, S/MIME

Advanced Encryption Standard (AES)

- Triple-DES is slow in software, has small blocks
- Selected by an open competition
 - Skipping long history
- A *de-facto* standard symmetric key scheme

Advanced Encryption Standard (AES)

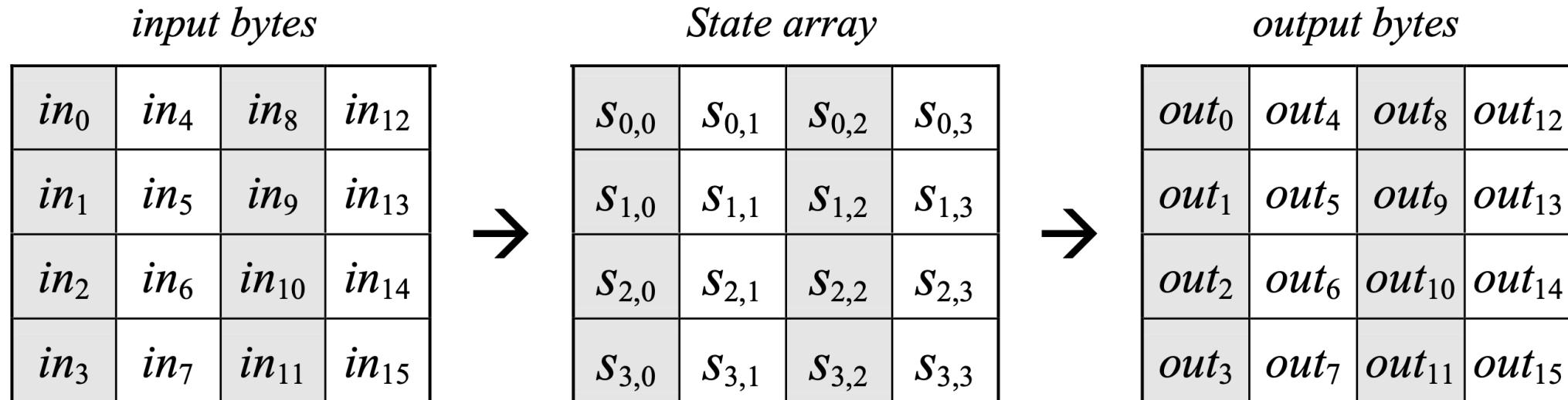
71



AES in a Nutshell (1): States



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



AES in a Nutshell (2): 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	
		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 (3): ShiftRows

74

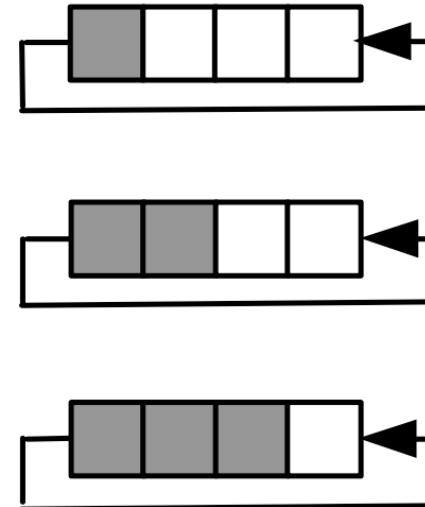
- Circular shift over different numbers of bytes

 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'

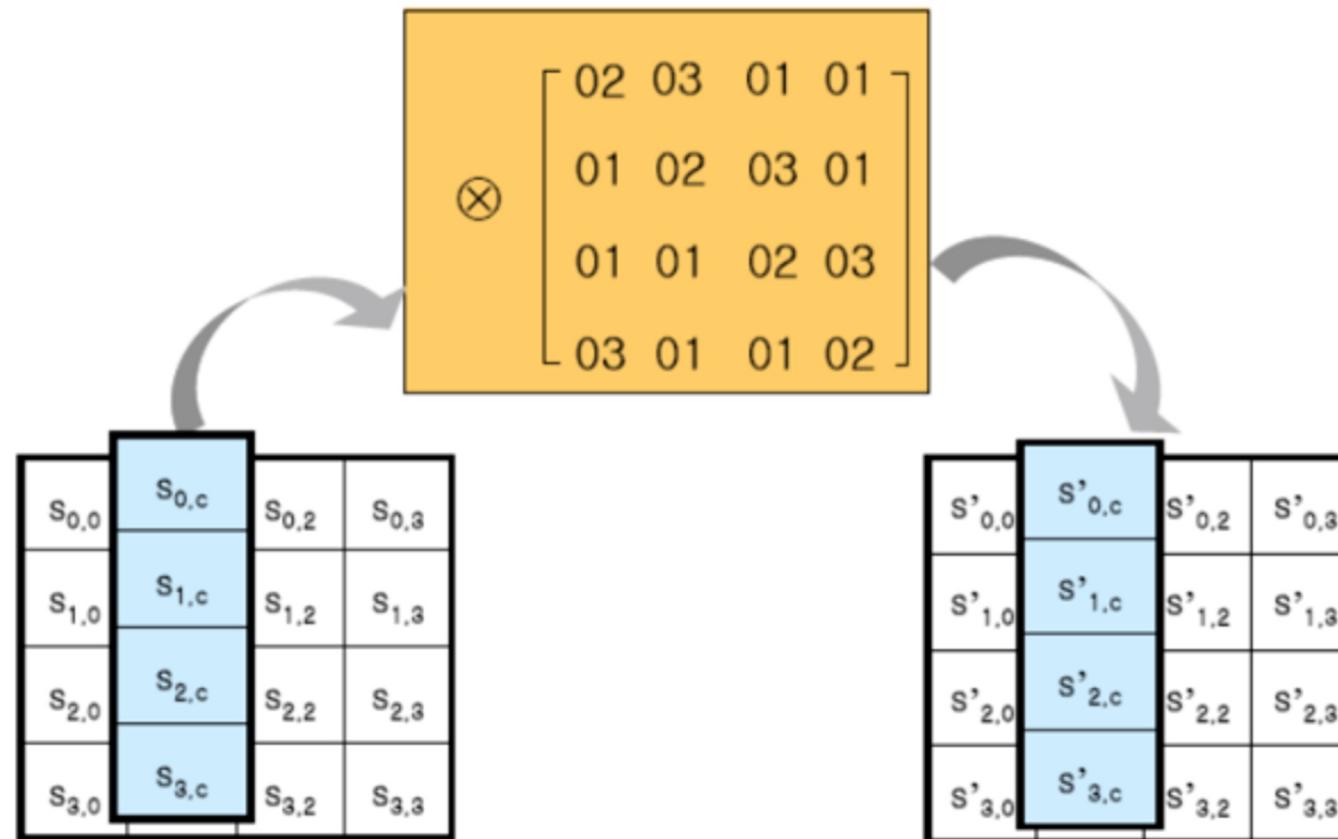
$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,1}$	$s_{1,2}$	$s_{1,3}$	$s_{1,0}$
$s_{2,2}$	$s_{2,3}$	$s_{2,0}$	$s_{2,1}$
$s_{3,3}$	$s_{3,0}$	$s_{3,1}$	$s_{3,2}$



AES in a Nutshell (4): MixColumns

75

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



AES in a Nutshell (5): Put it All Together

76

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

Question?