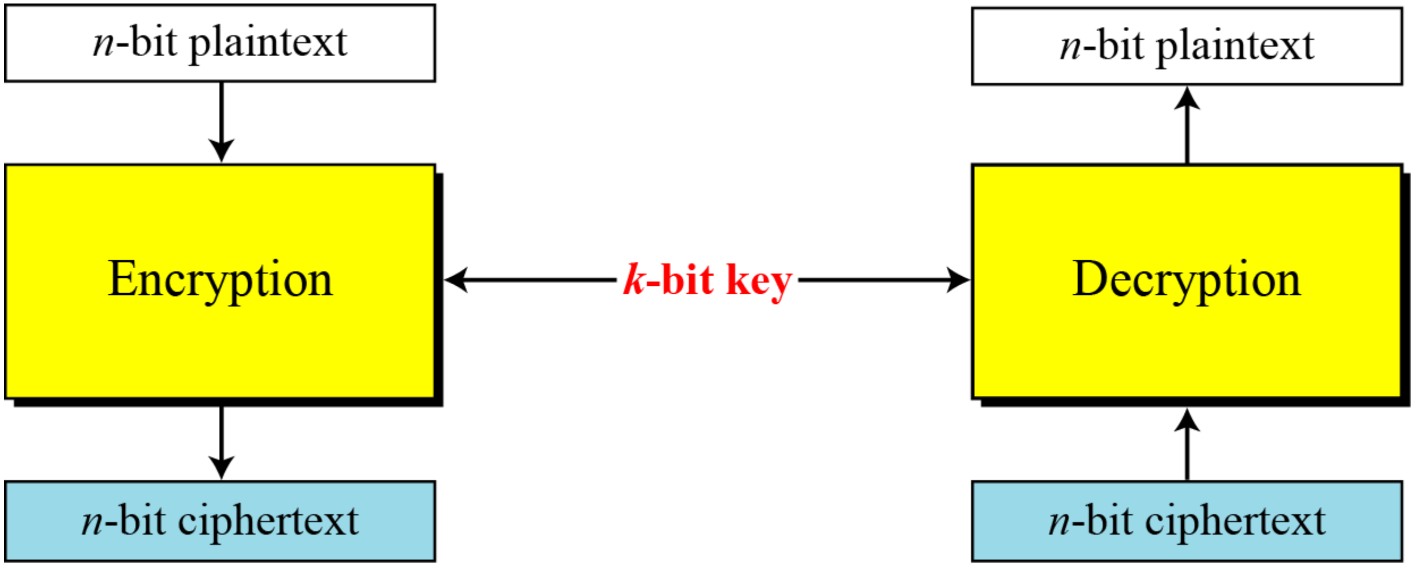
Mordern Block Ciphers

MODERN BLOCK CIPHERS

A symmetric-key modern block cipher encrypts an n-bit block of plaintext or decrypts an n-bit block of ciphertext. The encryption or decryption algorithm uses a k-bit key.

5.1 Continued

Figure 5.1 ***A modern block cipher***



5.1.1 Substitution or Transposition

A modern block cipher can be designed to act as a substitution cipher or a transposition cipher.

Note

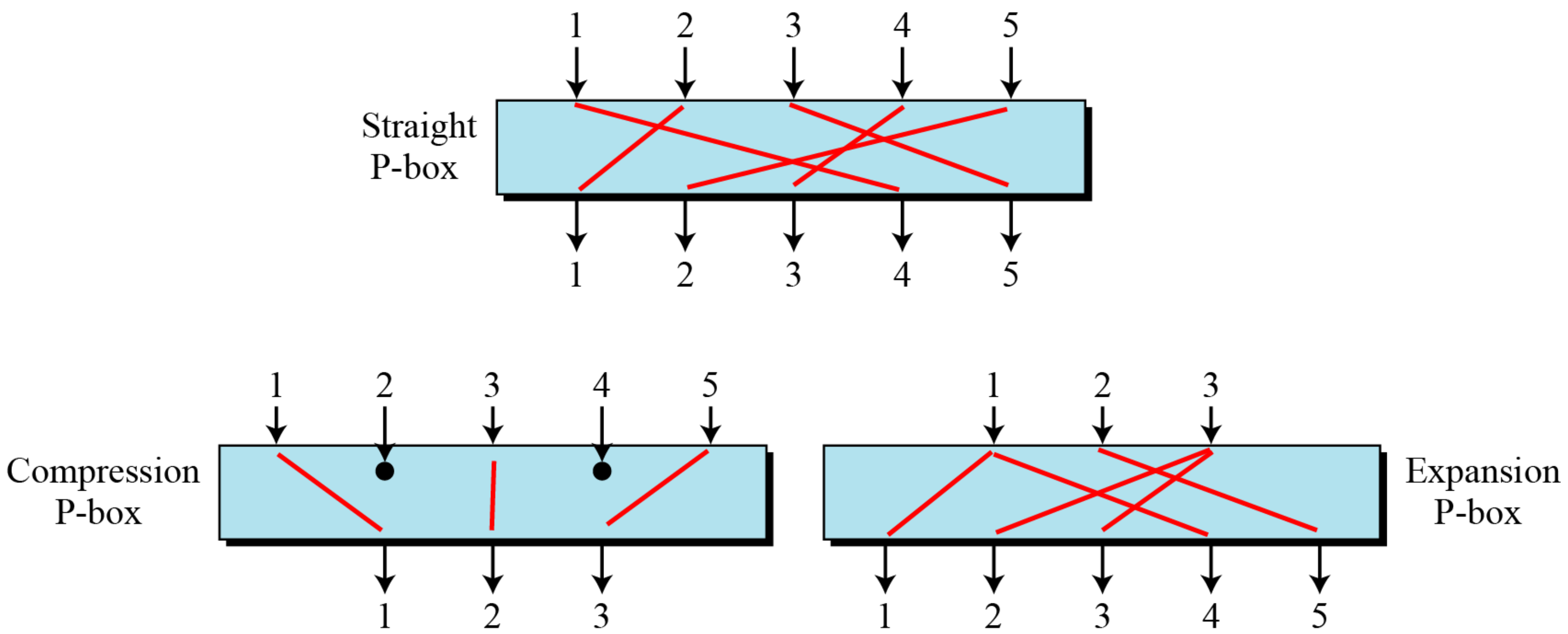
To be resistant to exhaustive-search attack, a modern block cipher needs to be designed as a substitution cipher.

5.1.2 P Box

- **Permutation Box parallels the traditional transposition cipher for characters**
- **It transposes bits**

5.1.3 Continued

Figure 5.4 ***Three types of P-boxes***



5.1.3 Continued

Straight P-Boxes

- *A straight P-box is a P-box with n inputs and n outputs*
- *n! possible mappings*
- *P Box is normally Keyless*
- *Mapping is predetermined*
- *If implemented in Hardware, then Prewired*
- *IF implemented in software, Permutation table shows rule of mapping*

Straight P-Boxes

Table 5.1 ***Example of a permutation table for a straight P-box***

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

5.1.3 Continued

Straight P-Boxes

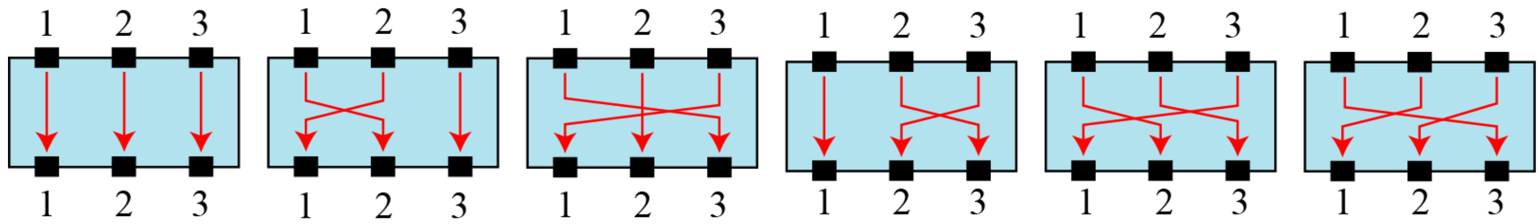
64 Inputs
64 Outputs
The index of the entry corresponds with the output
First entry is 58=>First Output comes from 58th Input
Last Entry is 07=>64th Output comes from 7th Input

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

Example 5.5

Figure 5.5 shows all 6 possible mappings of a 3 × 3 P-box.

Figure 5.5 ***The possible mappings of a 3 × 3 P-box***



Example 5.6

Design an 8 × 8 permutation table for a straight P-box that moves the two middle bits (bits 4 and 5) in the input word to the two ends (bits 1 and 8) in the output words. Relative positions of other bits should not be changed.

Solution

Example 5.6

Design an 8 × 8 permutation table for a straight P-box that moves the two middle bits (bits 4 and 5) in the input word to the two ends (bits 1 and 8) in the output words. Relative positions of other bits should not be changed.

Solution

We need a straight P-box with the table [4 1 2 3 6 7 8 5]. The relative positions of input bits 1, 2, 3, 6, 7, and 8 have not been changed, but the first output takes the fourth input and the eighth output takes the fifth input.

5.1.3 Continued

Compression P-Boxes

A compression P-box is a P-box with n inputs and m outputs where m < n.
Some of the inputs are blocked and do not reach the output.

Table 5.2 *Example of a 32 × 24 permutation table*

01	02	03	21	22	26	27	28	29	13	14	17
18	19	20	04	05	06	10	11	12	30	31	32

5.1.3 Continued

Compression P-Boxes

Normally Keyless with permutation table showing the rule for transposing bits

Table 5.2 ***Example of a 32 × 24 permutation table***

01	02	03	21	22	26	27	28	29	13	14	17
18	19	20	04	05	06	10	11	12	30	31	32

5.1.3 Continued

Compression P-Boxes

32 X 24 Compression P Box
Inputs 7,8,9,15,16,23,24 and 25 are blocked

Used when we need to permute bits and the same time decrease the number of bits for next stage

Table 5.2 *Example of a 32 × 24 permutation table*

01	02	03	21	22	26	27	28	29	13	14	17
18	19	20	04	05	06	10	11	12	30	31	32

Expansion P-Boxes

An expansion P-box is a P-box with n inputs and m outputs where m > n.

Some of the inputs are connected to more than one output

Table 5.3 *Example of a 12 × 16 permutation table*

5.1.3 Continued

Expansion P-Boxes

Normally Keyless with permutation table showing the rule for transposing bits

Each of the inputs 1,3,9 and 12 are mapped to two outputs

Used when we need to permute bits and the same time increase the number of bits for next stage

Table 5.3 ***Example of a 12 × 16 permutation table***

5.1.3 Continued

P-Boxes: Invertibility

Note

A straight P-box is invertible, but compression and expansion P-boxes are not.

5.1.3 Continued

P-Boxes: Invertibility

Note

A straight P-box is invertible=>We can use Straight box in the encryption cipher and its inverse in the decryption cipher

Permutation tables need to be inverses of each other

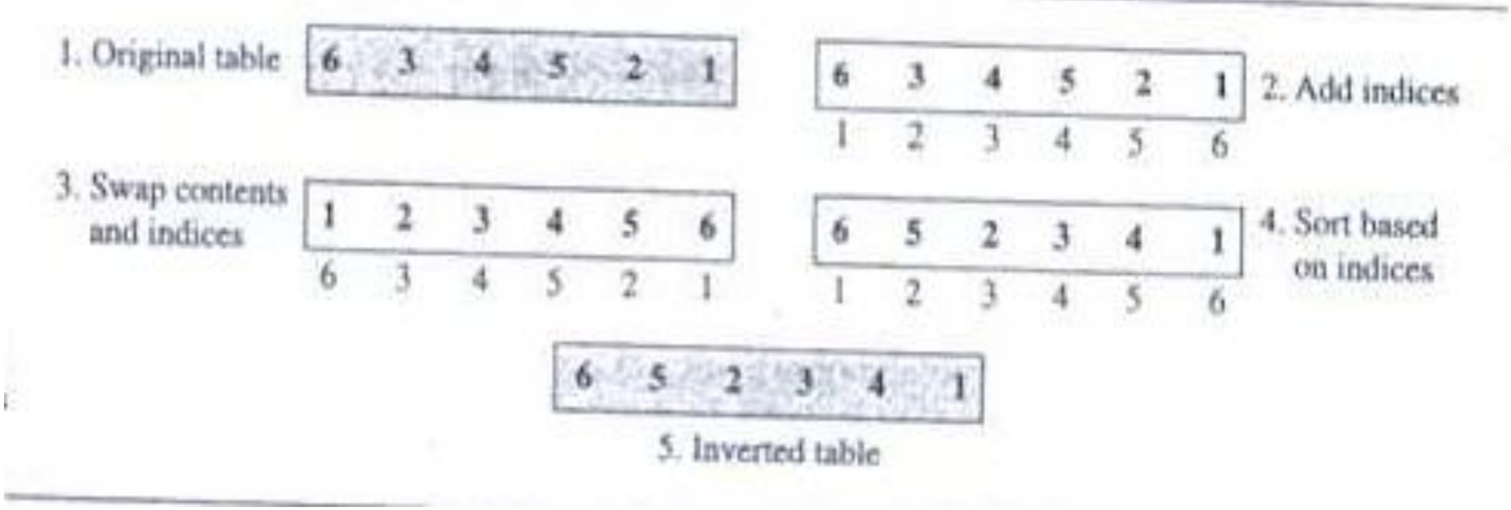
5.1.3 Continued

P-Boxes: Invertibility

Note

Permutation tables need to be inverses of each other

Figure 5.6 *Inverting a permutation table*



5.1.3 Continued

P-Boxes: Invertibility

Note

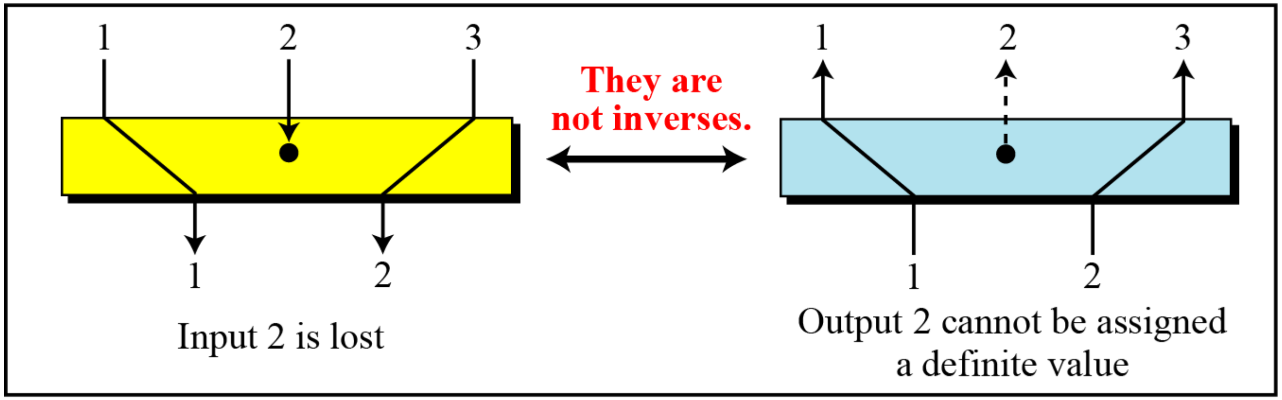
Compression and expansion P-boxes are not invertible.

P-Boxes: Invertibility

Note

In Compression P Box-
An input can be dropped During Encryption .
The Decryption Algorithm doesn't have a clue how to replace the dropped bit.

Compression P-box

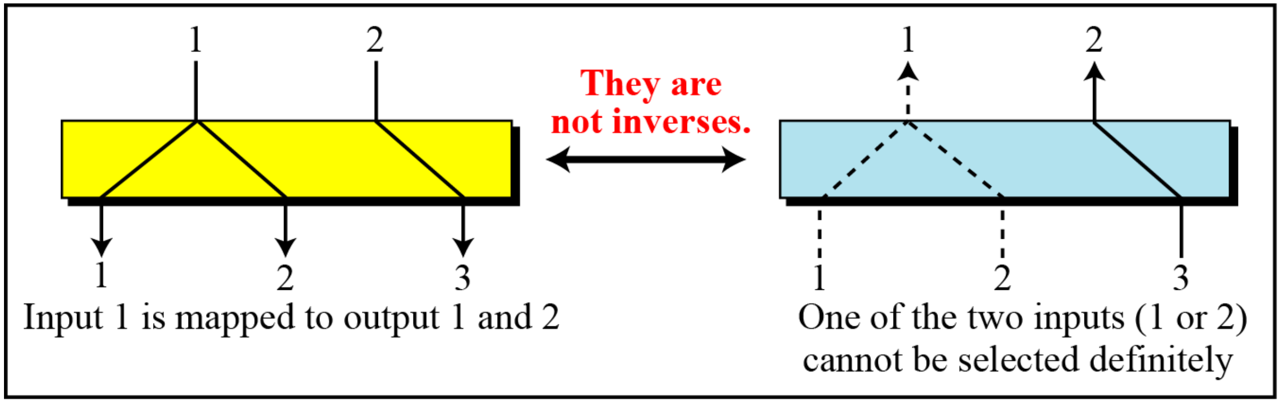


5.1.3 Continued

P-Boxes: Invertibility

Note

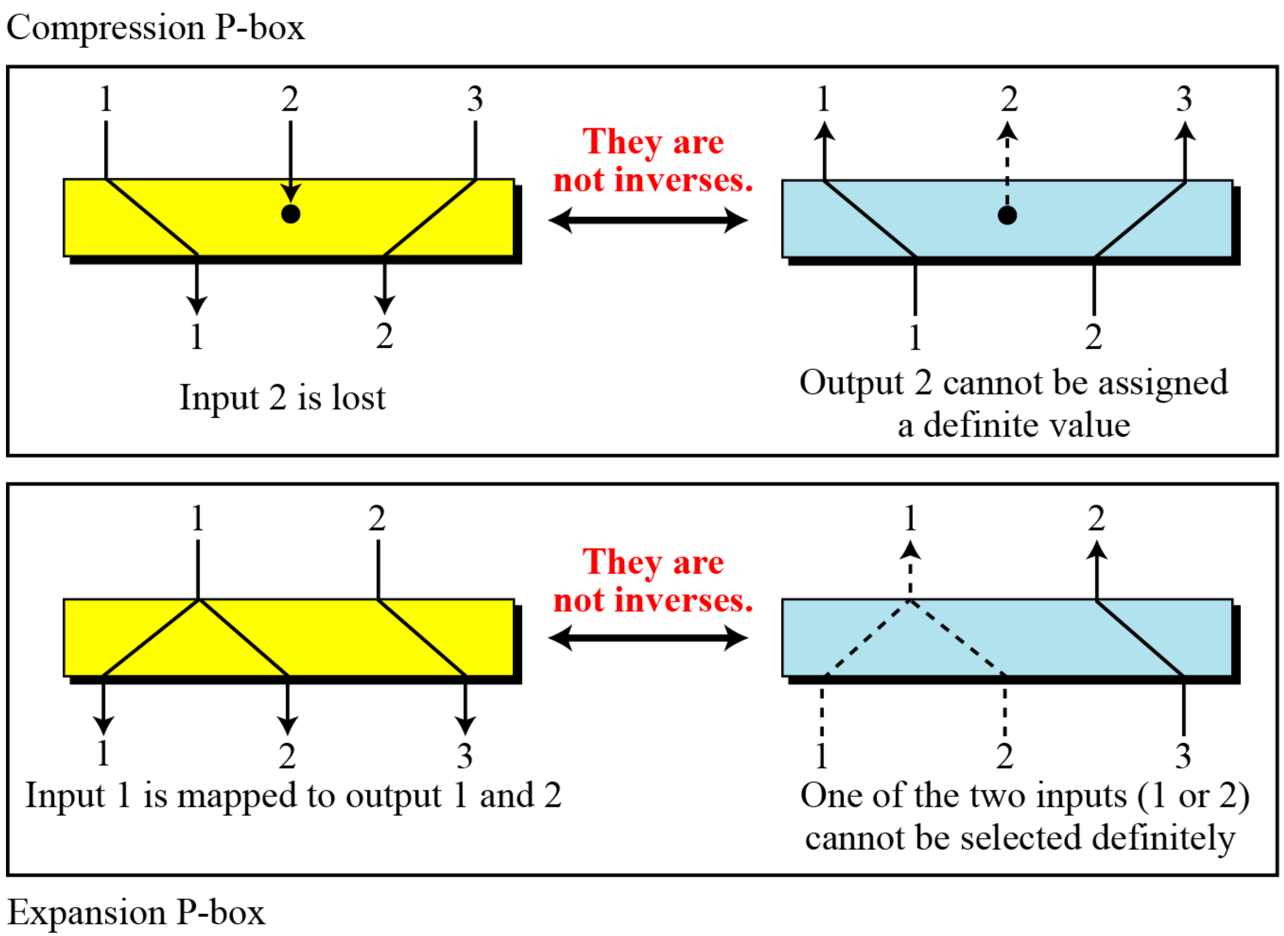
In Expansion P Box-
An input may be mapped to more than one output during Encryption. The Decryption algorithm does not have a clue which of the several inputs are mapped to an output.



Expansion P-box

5.1.3 Continued

Figure 5.7 *Compression and expansion P-boxes are non-invertible*



5.1.3 Continued

P-Boxes: Invertibility

Note

Compression P-box is not the inverse of expansion P-box or vice versa

5.1.3 Continued

S-Box

An S-box (substitution box) can be thought of as a miniature substitution cipher.

Note

An S-box is an $m \times n$ substitution unit, where m and n are not necessarily the same.

5.1.3 Continued

S-Box

- ***S Box can be keyed or keyless***

- ***Modern Block ciphers normally use keyless S Box where the mapping from inputs to outputs is predetermined***

5.1.3 Continued

S-Box

- ***S Box are substitution ciphers in which relationship between input and output is defined by***
    - ***a table or***
    - ***mathematical relation***

5.1.3 Continued

Linear vs Non-Linear S-Box

S Box with n inputs and m outputs

Inputs:x1,x2,…...xn

Outputs:y1,y2,……...ym

Relationship between inputs and outputs can be expressed as

$$y_1 = f_1(x_1, x_2, \ldots, x_n)$$
$$y_2 = f_2(x_1, x_2, \ldots, x_n)$$
$$\ldots$$
$$y_m = f_m(x_1, x_2, \ldots, x_n)$$

Linear S-Box

The relations can be expressed as:

$$
\begin{aligned}
y_1 &= a_{1,1}x_1 \oplus a_{1,2}x_1 \oplus \cdots \oplus a_{1,n}x_n \\
y_2 &= a_{2,1}x_1 \oplus a_{2,2}x_1 \oplus \cdots \oplus a_{2,n}x_n \\
&\cdots \\
y_m &= a_{m,1}x_1 \oplus a_{m,2}x_1 \oplus \cdots \oplus a_{m,n}x_n
\end{aligned}
$$

Non Linear S-Box

We cannot have the above relations for every output

5.1.3 Continued

Example 5.9

Linear Equations-
It **has only one degree**. Or we can also define it as an equation having the maximum degree 1.

Non Linear Equations-
A nonlinear equation **has the degree as 2 or more than 2**, but not less than 2.

5.1.3 Continued

Example 5.8

In an S-box with three inputs and two outputs, we have

$$y_1 = x_1 \oplus x_2 \oplus x_3 \qquad y_2 = x_1$$

- The S-box is linear as we have equation for every output
- Because $a_{1,1} = a_{1,2} = a_{1,3} = a_{2,1} = 1$ and $a_{2,2} = a_{2,3} = 0$. The relationship can be represented by matrices, as shown below:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

5.1.3 Continued

Example 5.8

In an S-box with three inputs and two outputs, we have

$$y_1 = x_1 \oplus x_2 \oplus x_3 \qquad y_2 = x_1$$

If the input is 110 then the output y1= 0 and y2= 1
If the input is 001 then the output y1= 1 and y2= 0.

5.1.3 Continued

Example 5.9

In an S-box with three inputs and two outputs, we have

$$y_1 = (x_1)^3 + x_2 \qquad y_2 = (x_1)^2 + x_1 x_2 + x_3$$

The S-box is nonlinear because there is no linear relationship between the inputs and the outputs.

Example 5.10

The following table defines the input/output relationship for an S-box of size 3 × 2.

- The leftmost bit of the input defines the row;
- The two rightmost bits of the input define the column.
- The two output bits are values on the cross section of the selected row and column.

Leftmost bit

Rightmost bits

	00	01	10	11
0	00	10	01	11
1	10	00	11	01

Output bits

Based on the table, an input of 010 yields the output 01. An input of 101 yields the output of 00.

5.1.3 Continued

S-Boxes: Invertibility

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.

5.1.3 Continued

Example 5.11

Figure 5.8 shows an example of an invertible S-box.
For example, if the input to the left box is 001, the output is 101.
The input 101 in the right table creates the output 001, which shows that the two tables are inverses of each other.

Figure 5.8 ***S-box tables for Example 5.11***

3 bits

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

Table used for encryption

3 bits

3 bits

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

Table used for decryption

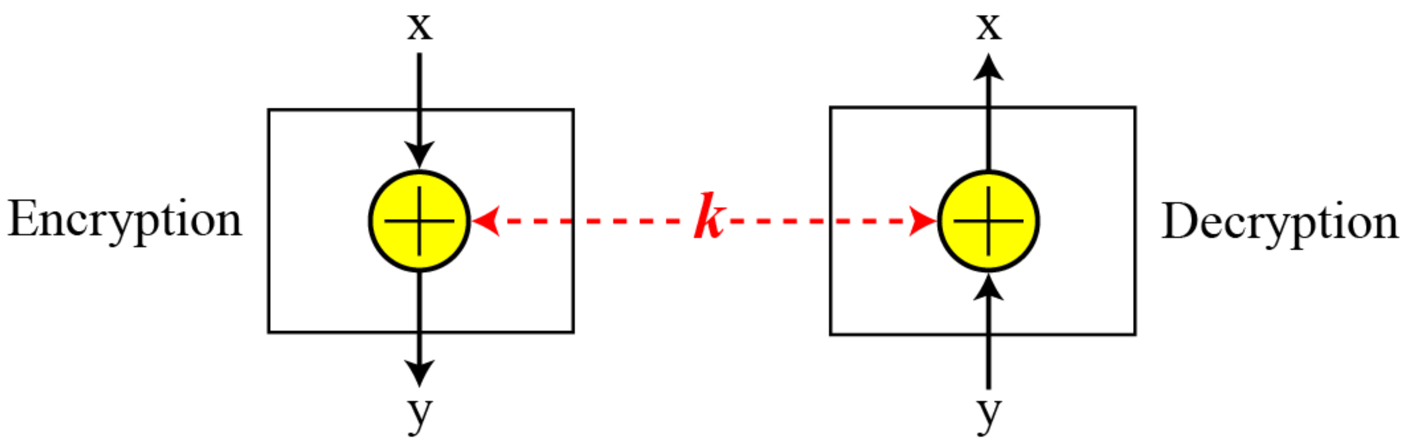
3 bits

5.1.3 Continued

Exclusive-Or

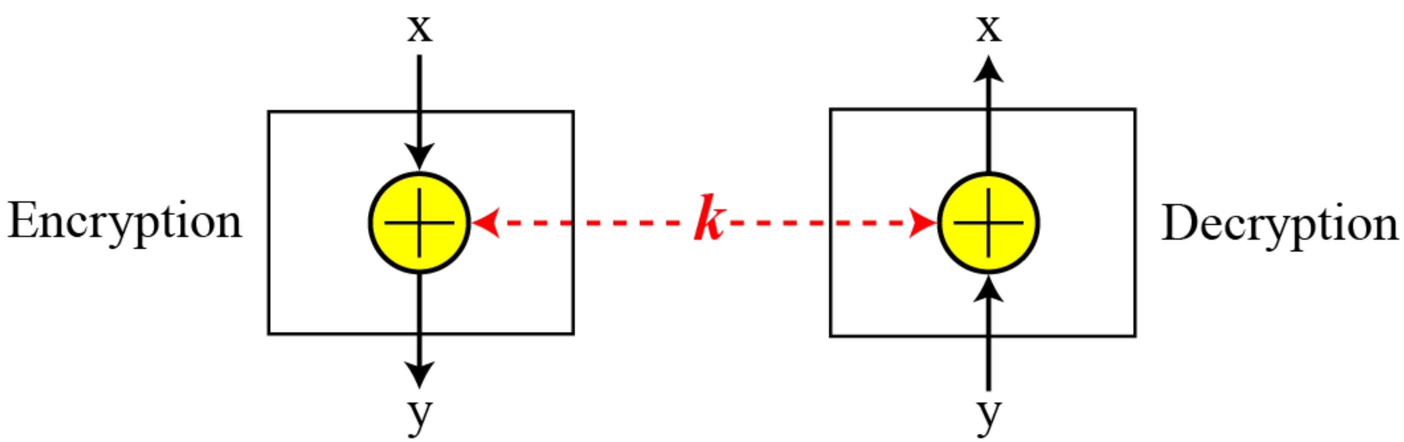
An important component in most block ciphers is the exclusive-or operation.

Figure 5.9 ***Invertibility of the exclusive-or operation***



5.1.1 Continued

Figure 5.9 *Invertibility of the exclusive-or operation*



5.1.1 Continued

Figure 5.9 ***Invertibility of the exclusive-or operation***

5 Properties of EXOR Operation which makes this operation important

1) ***Closure***
2) ***Associativity***
3) ***Commutativity***
4) ***Existence of identity***
5) ***Existence of inverse***

5.1.1 Continued

Figure 5.9 *Invertibility of the exclusive-or operation*

Closure-Result of EXORing two n bit words is another n bit word

Associativity-Allows to use EXOR operator in any order

$$x \oplus (y \oplus z) \leftrightarrow (x \oplus y) \oplus z$$

Commutativity-Allows to swap the inputs without affecting the output

$$x \oplus y \leftrightarrow y \oplus x$$

5.1.1 Continued

Figure 5.9 *Invertibility of the exclusive-or operation*

Existence of identity-
N bit word containing all O's
Exclusive ORing of a word with the identity element does not change that word

$$x \oplus (00\ldots0) = x$$

Existence of inverse-
Each word is additive inverse of itself.
EXORing a word with itself yeilds the identity element

$$x \oplus x = (00\ldots0)$$

5.1.3 Continued

Circular Shift

Another component found in some modern block ciphers is the circular shift operation.

Mixes bits in a word and helps hide the patterns in the original word

Circular Shift

Circular left shift operation shifts each bit in a n bit word ,k positions left.

The leftmost k bits are removed from the left and become the rightmost bits

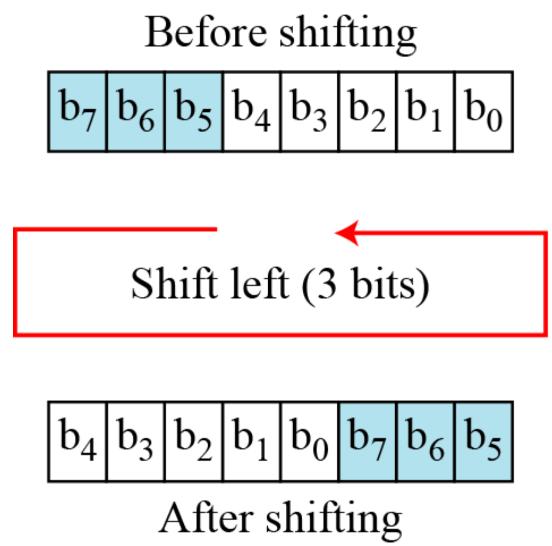


Figure 5.10 ***Circular shifting an 8-bit word to the left or right***

Circular Shift

Circular right shift operation shifts each bit in a n bit word ,k positions right.
The rightmost k bits are removed from the right and become the leftmost bits

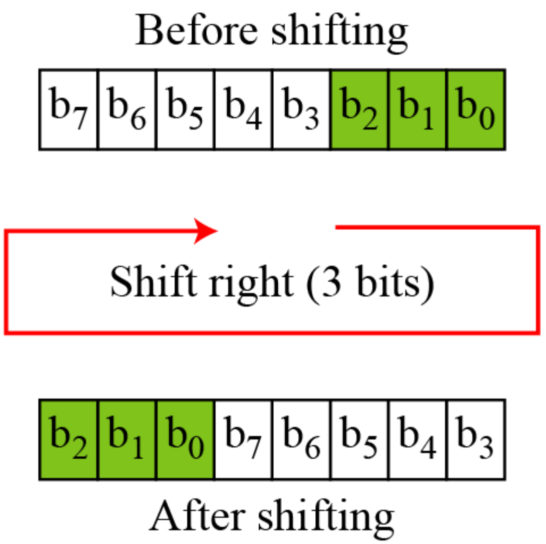


Figure 5.10 *Circular shifting an 8-bit word to the left or right*

5.1.3 Continued

Circular Shift

Invertibility- A circular left shift operation is the inverse of the circular right shift operation

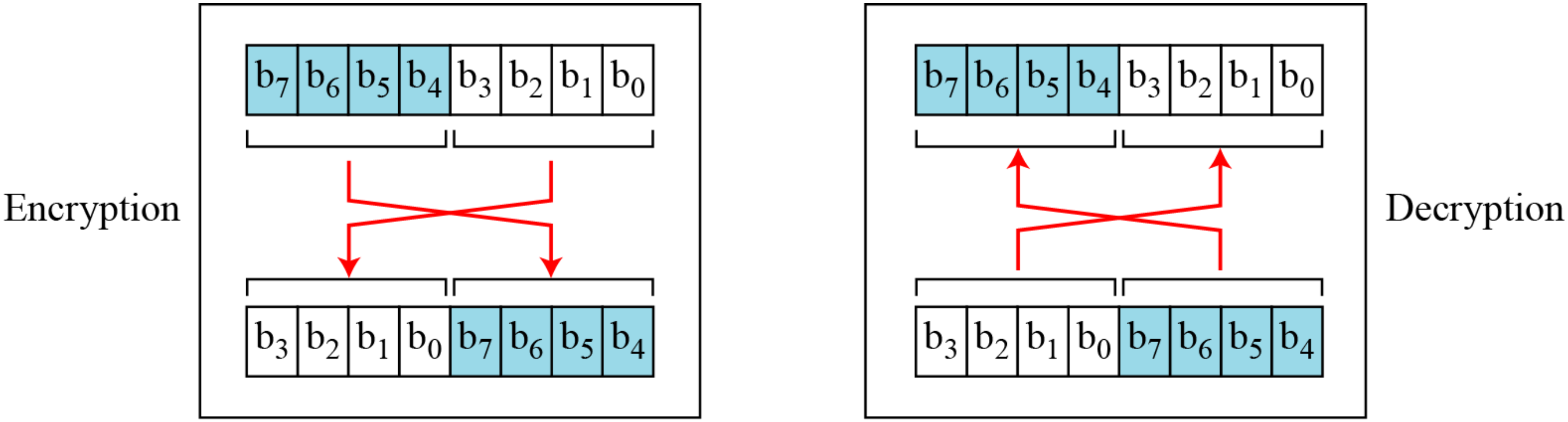
If one is used in encryption cipher, The other can be used in the Decryption cipher

5.1.3 Continued

Swap

The swap operation is a special case of the circular shift operation where k = n/2.

Figure 5.11 ***Swap operation on an 8-bit word***

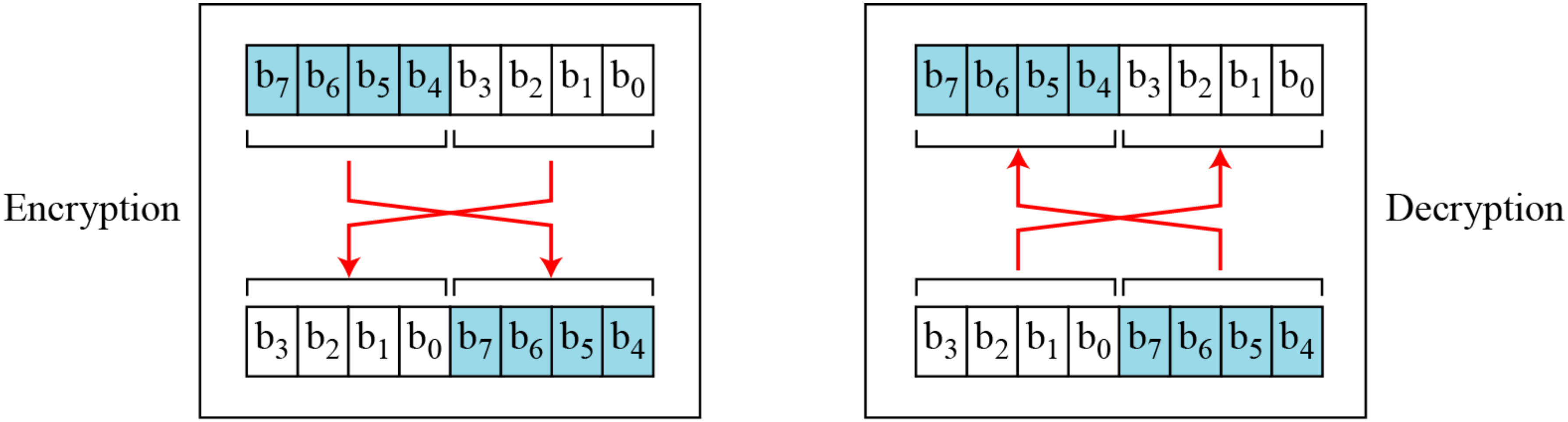


5.1.3 Continued

Swap

Operation is valid only if n is an even number
Self Invertible-A swap operation in Encryption cipher can be totally cancelled by a swap operation in the decryption cipher

Figure 5.11 ***Swap operation on an 8-bit word***

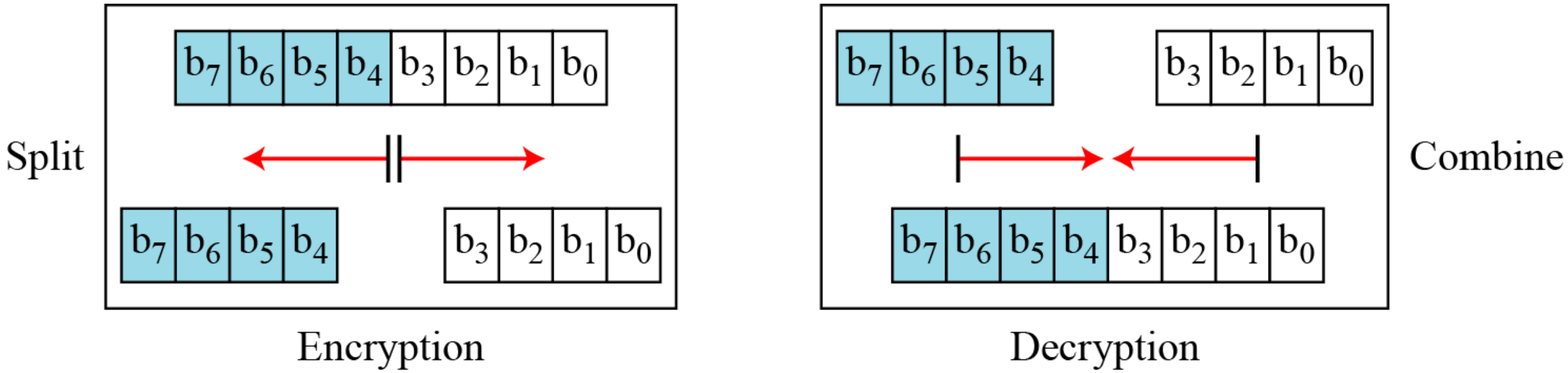


5.1.3 Continued

Split and Combine

Two other operations found in some block ciphers are split and combine.

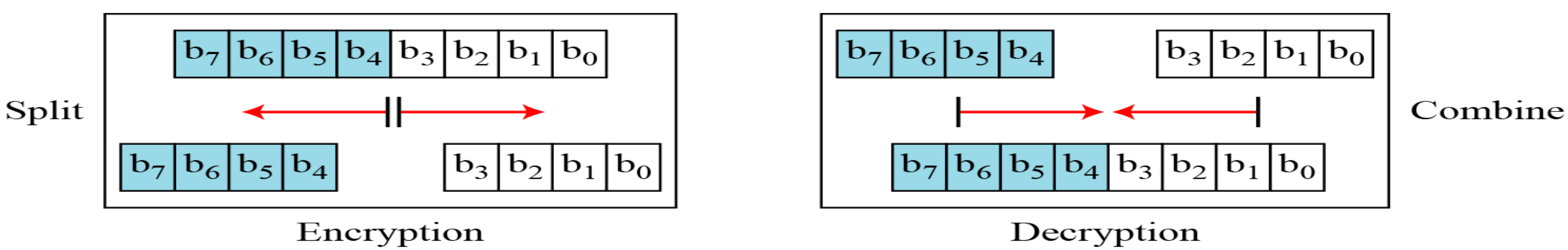
Figure 5.12 ***Split and combine operations on an 8-bit word***



5.1.3 Continued

Figure 5.12 ***Split and combine operations on an 8-bit word***

- ***Split operation splits an n-bit word in the middle creating two equal length words***
- ***Combine operation normally concatenates two equal length words to create an n bit word.***
- ***Inverse of each other***
- ***Used as a pair to cancel each other out.***



5.1.4 Product Ciphers

Shannon introduced the concept of a product cipher. A product cipher is a complex cipher combining substitution, permutation, and other components discussed in previous sections.

5.1.4 Continued

Diffusion

- ***The idea of diffusion is to hide the relationship between the ciphertext and the plaintext.***

Note

Diffusion hides the relationship between the ciphertext and the plaintext.

- ***Frustate the adversary who uses ciphertext statistics to find plaintext***
- ***If a single symbol in plaintext is changes, several or all symbols in the ciphertext will also be changed***

5.1.4 Continued

Confusion

- ***The idea of confusion is to hide the relationship between the ciphertext and the key.***

Note

Confusion hides the relationship between the ciphertext and the key.

- ***Frustate the adversary who uses ciphertext statistics to find key***
- ***If a single bit in the key is changed, most or all bits in the ciphertext will also be changed***

5.1.4 Continued

Rounds

Diffusion and confusion can be achieved using iterated product ciphers where each iteration is a combination of S-boxes, P-boxes, and other components.

5.1.4 Continued

Rounds

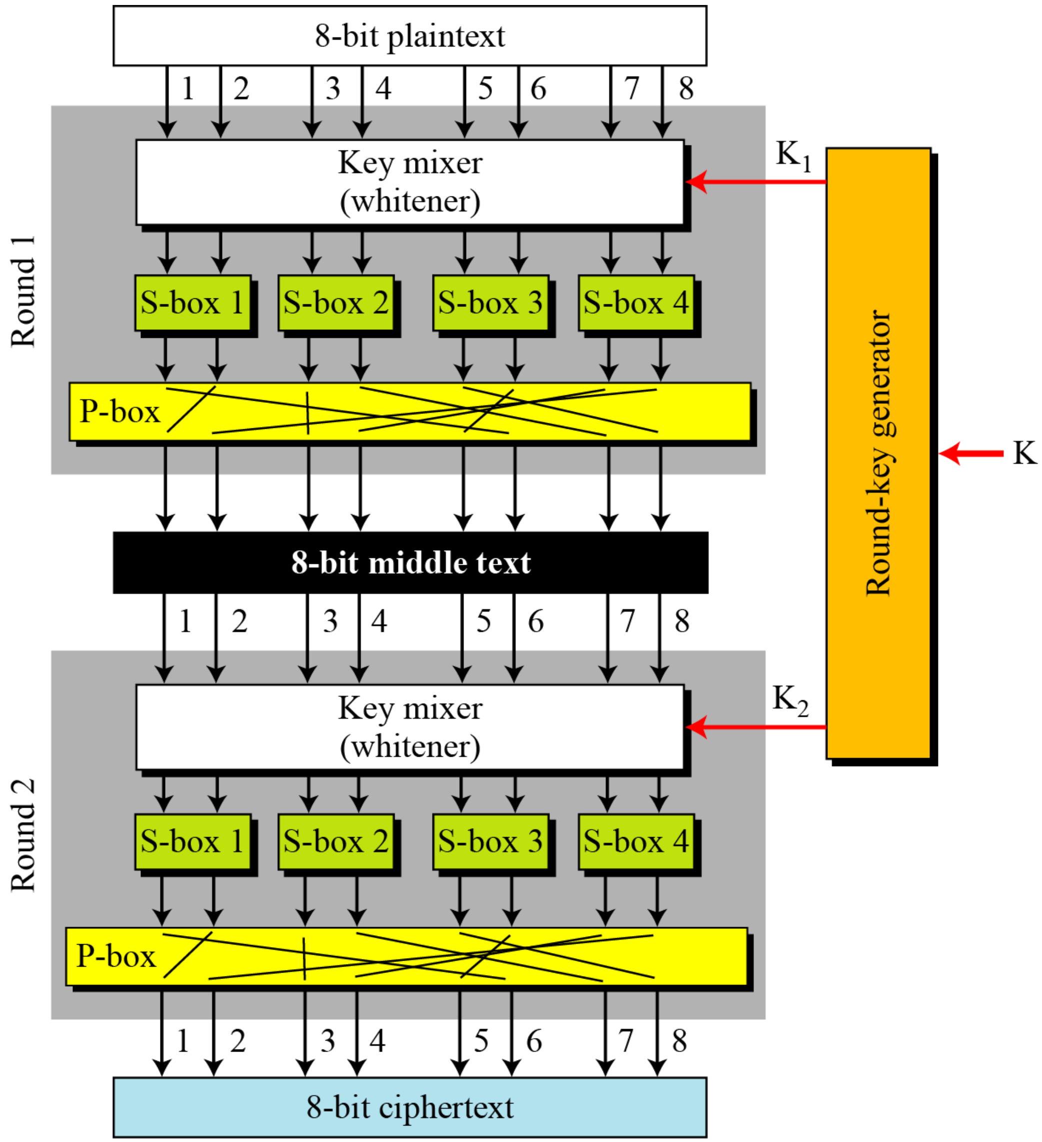
The block cipher uses a Key schedule or Key generator that creates different keys for each round.

In a N-round cipher, the plaintext is encrypted N times to create the ciphertext.

The ciphertext is decrypted N times to create the plaintext

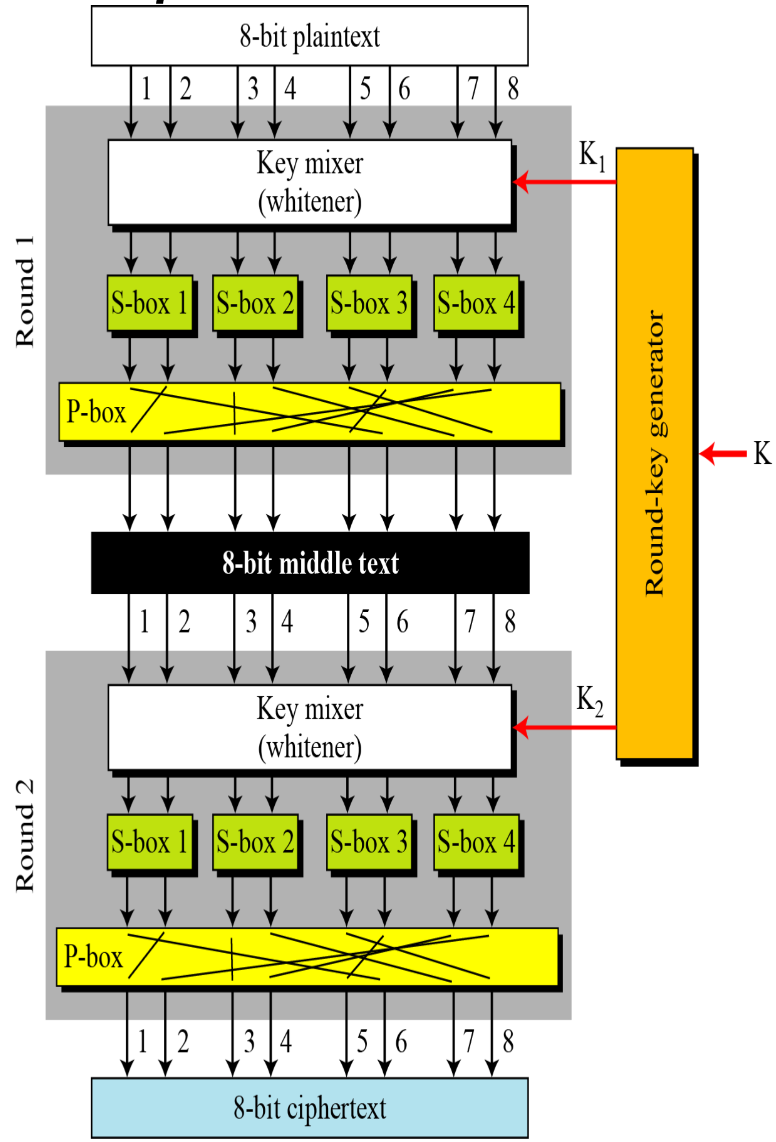
5.1.4 Continued

Figure 5.13 *A product cipher made of two rounds*



5.1.4 Continued

Figure 5.13 *A product cipher made of two rounds*



Rounds

Key Mixer-

- ***8 bit text is mixed with Key to whiten the text***
- ***Done by EXORing 8 bit word with 8 bit key***

Sbox-

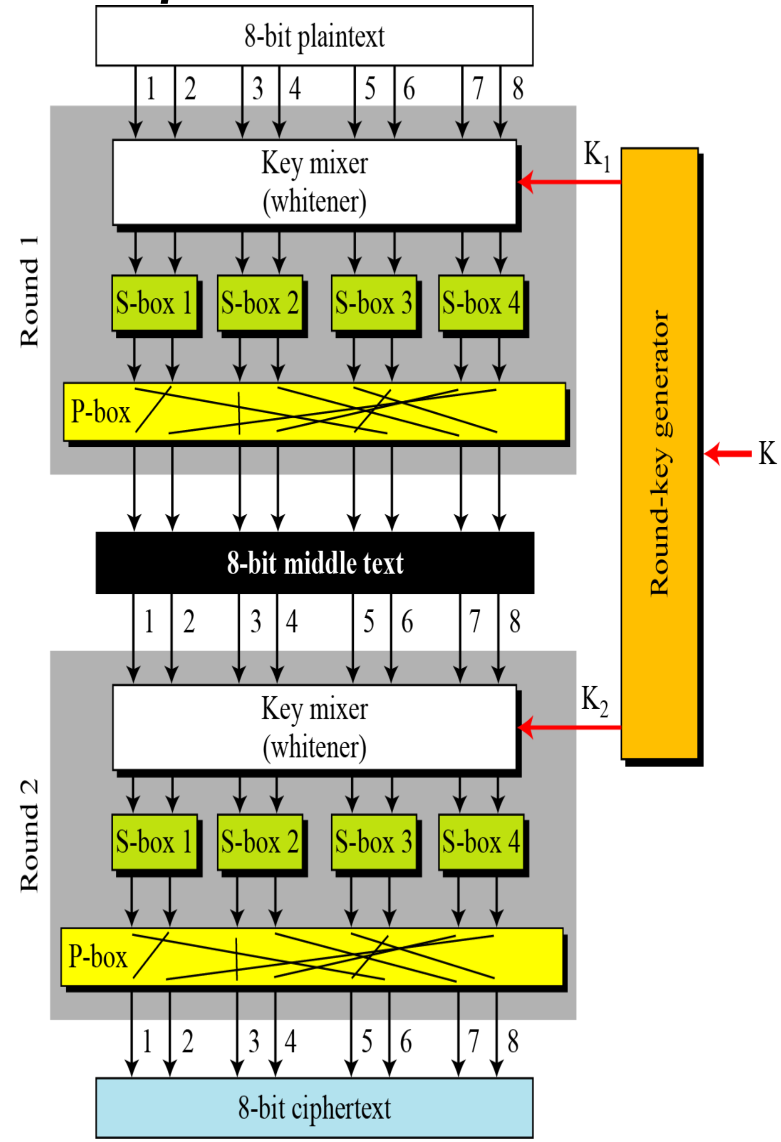
- ***Output organized as 4 groups of 2 bits, Fed into 4 S box***
- ***Values of Bits change based on Structure of S Box***

Figure 5.13 *A product cipher made of two rounds*

Rounds

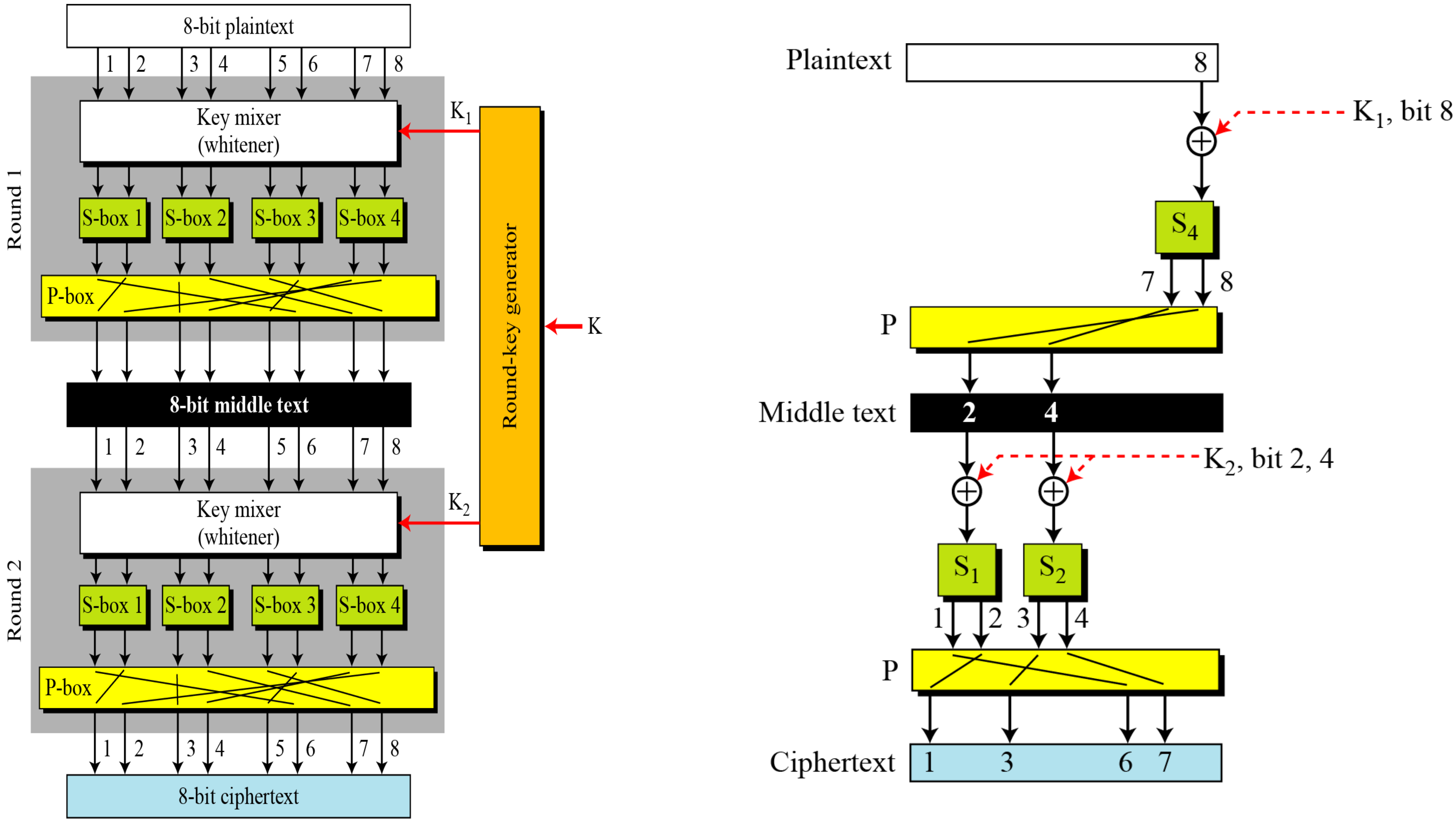
Pbox-

- ***O/P of S Box sent to P Box to permute the bits***



5.1.4 Continued

Figure 5.14 *Diffusion and confusion in a block cipher*



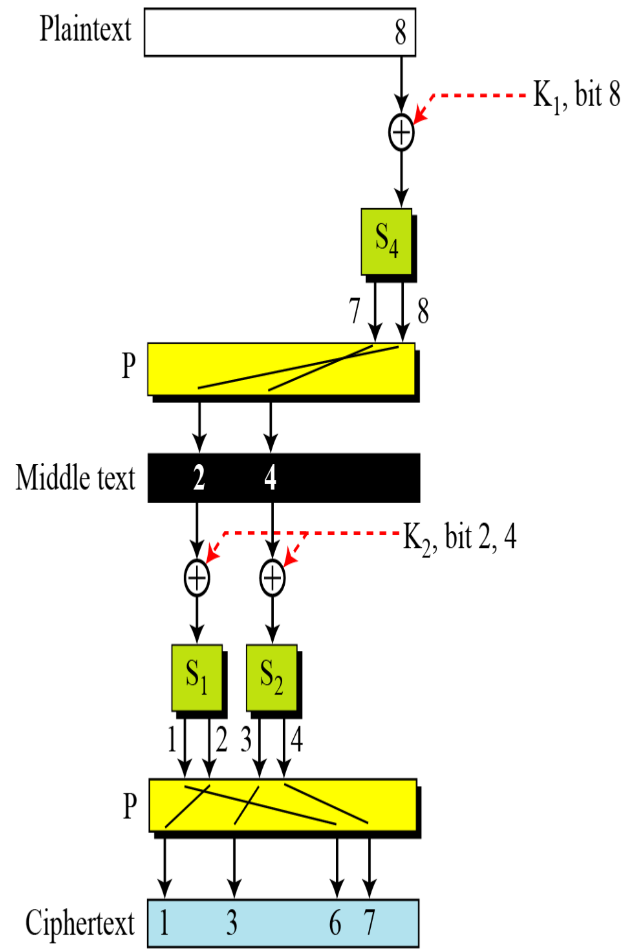
How Changing a single bit in the plain text affects many bits in the ciphertext

5.1.4 Continued

Diffusion:

1st Round-

- ***Bit 8 after EXORing with corresponding bit of K1 affects 2 bits i.e. Bit7 and Bit 8 through S Box***
- ***After permutation-Bit 7 becomes Bit 2, Bit 8 becomes Bit 4***
- ***After 1st round, Bit 8 has affected Bits 2 and 4***



5.1.4 Continued

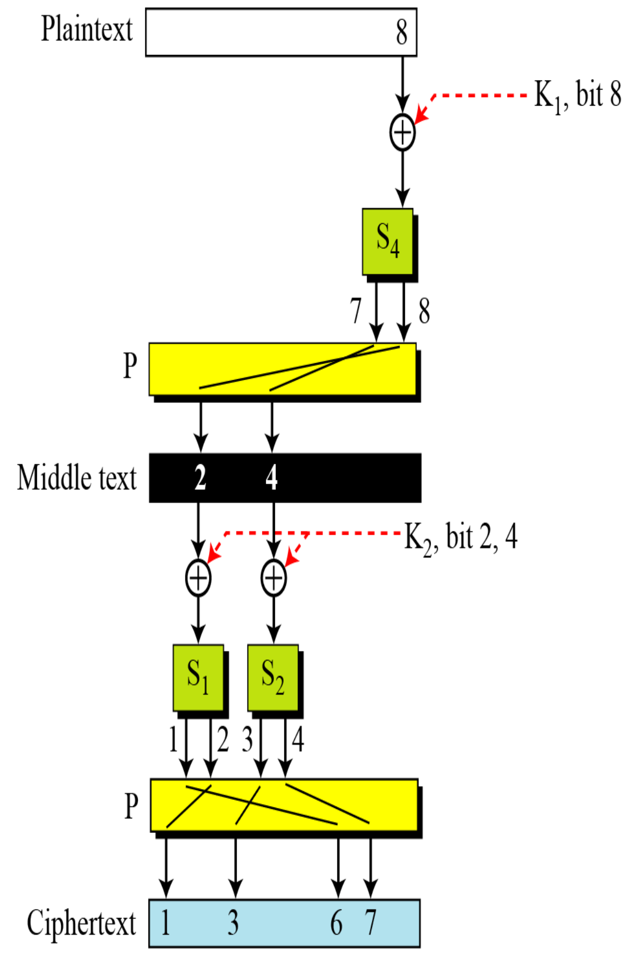
2^{nd} Round-

- ***Bit 2 and Bit 4 are EXORed with corresponding bits of Key K2***
- ***Bit 2 affects Bits 1,2 through S Box 1***
- ***Bits 4 affects Bits 3,4 through S Box2***

- ***After Permutation, Bit 1 becomes Bit 6, Bit 2 becomes Bit 1.***

- ***Bit 4 becomes Bit 7, Bit 3 remains same***

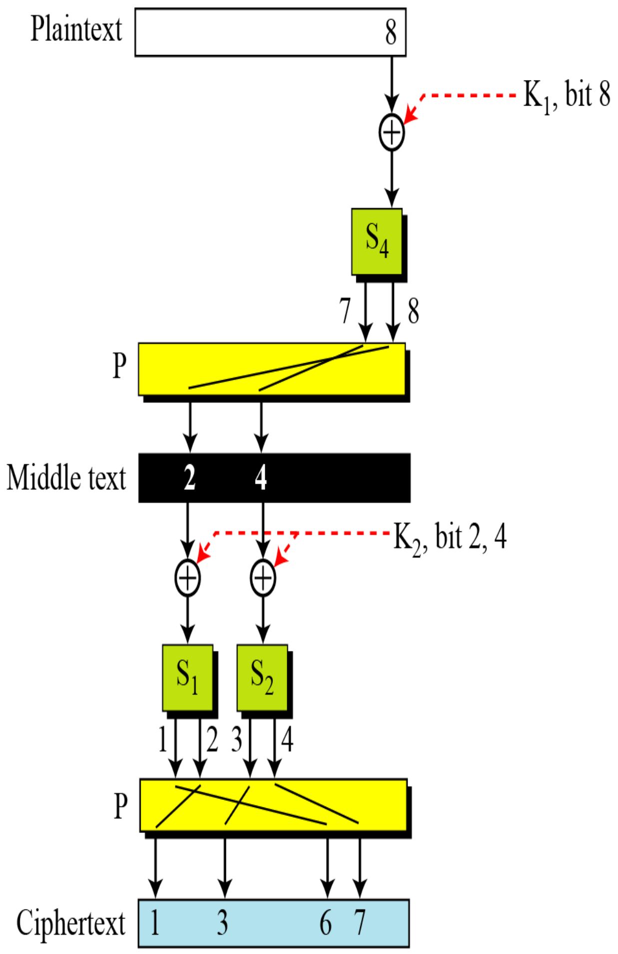
- ***After 2^{nd} round, Bit 8 has affected Bits 1,3,6,7***



5.1.4 Continued

Diffusion:

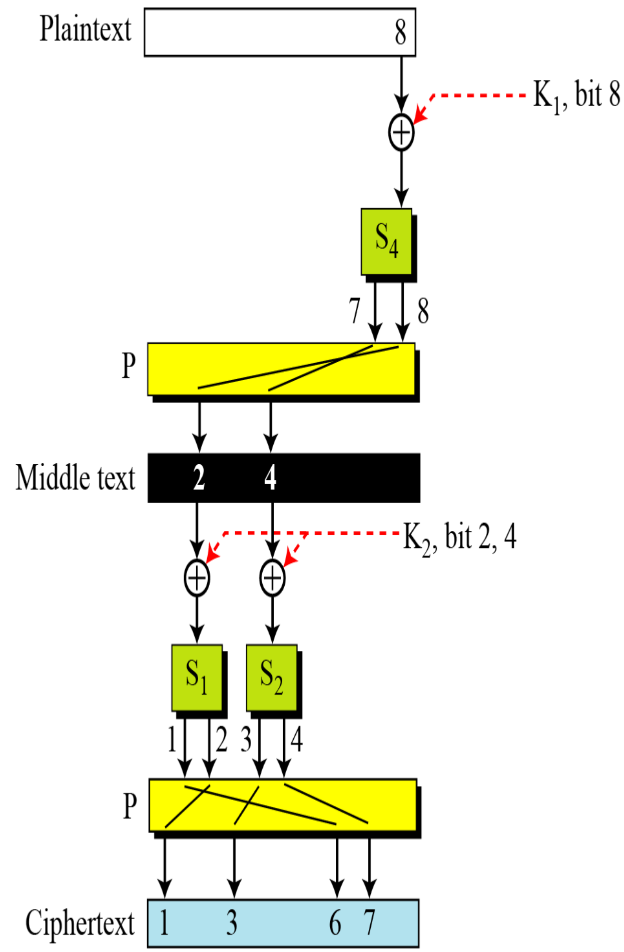
- ***One bit in plain text has affected several bits in ciphertext***
- ***Bit 8 has affected Bits 1,3,6,7***



5.1.4 Continued

Confusion:

- ***4 bits of ciphertext bits 1,3,6,7 are affected by 3 bits in the Key***
- ***i.e. Bit 8 in Key1, Bits 2 and 4 in K2***
- ***Each bit in each round key affects several bits in the ciphertext.***
- ***The Relationship between ciphertext and key is obscured***



5.1.5 Two Classes of Product Ciphers

Modern block ciphers are all product ciphers, but they are divided into two classes.

1. Feistel ciphers

2. Non-Feistel ciphers

5.1.5 Two Classes of Product Ciphers

1. Feistel ciphers- Uses both invertible and non-invertible components

2. Non-Feistel ciphers-Uses only invertible components

5.1.5 Continued

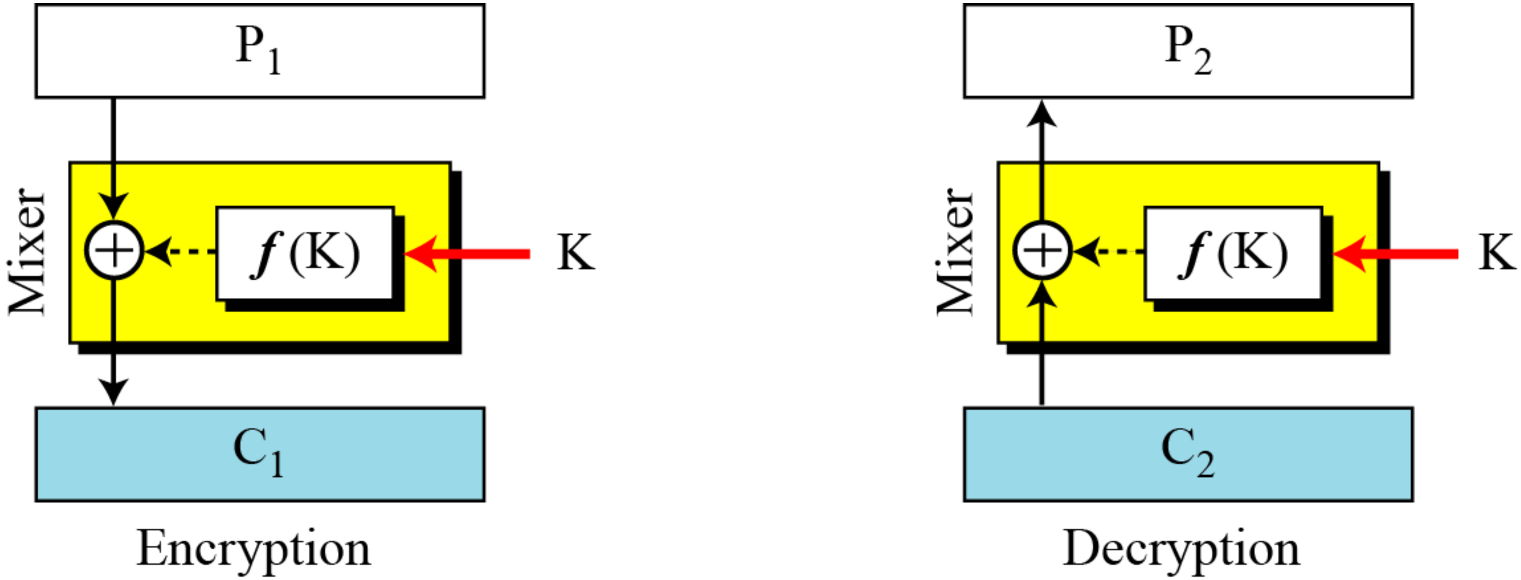
Feistel Ciphers

Feistel designed a very intelligent and interesting cipher that has been used for decades.

A Feistel cipher can have three types of components: self-invertible, invertible, and noninvertible.

5.1.5 *Continued*

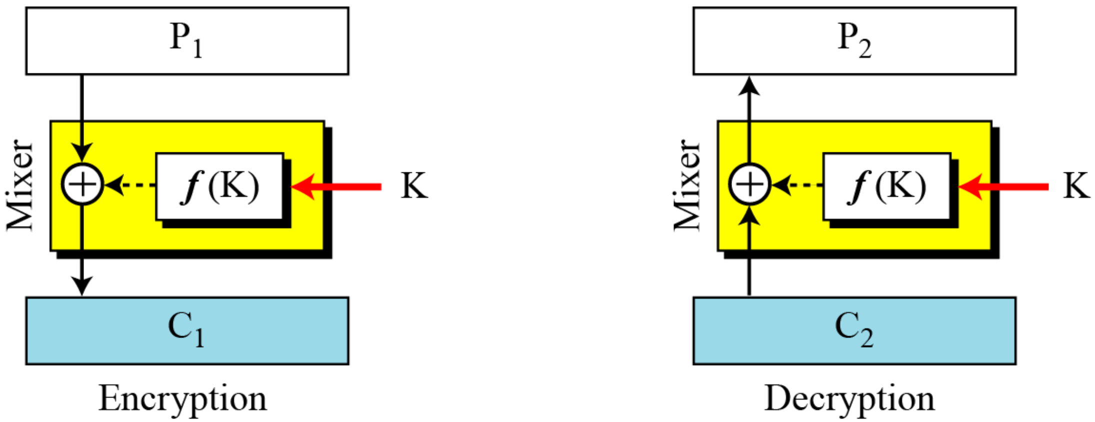
Figure 5.15 ***The first thought in Feistel cipher design***



5.1.5 Continued

Encryption-

- ***A non invertible function f(K) accepts the key as the input***
- ***The Output is EXORed with the plain text to give Ciphertext***
- ***Combination of Function and EXOR =MIXER***

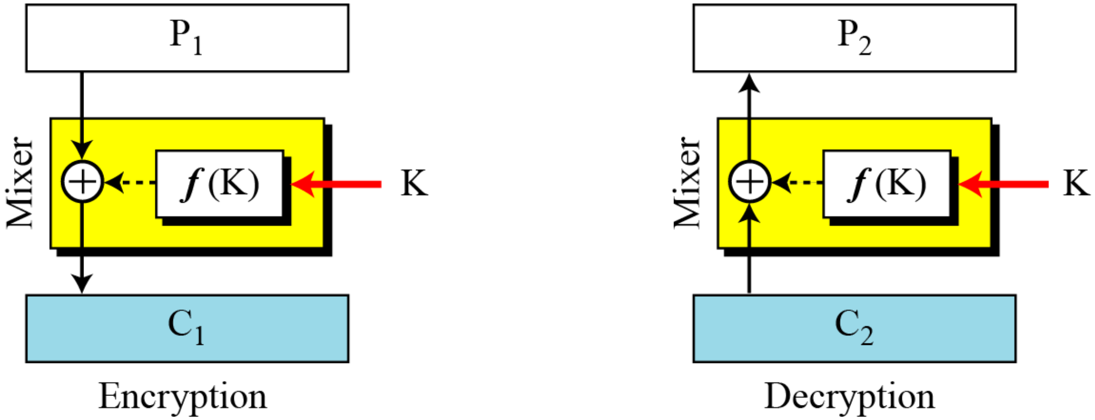


5.1.5 Continued

Encryption-

- ***Because the Key is the same in Encryption and Decryption***
- ***The Two algorithms are inverses of each other***
- ***If C2=C1, No change in the ciphertext then P2=P1***

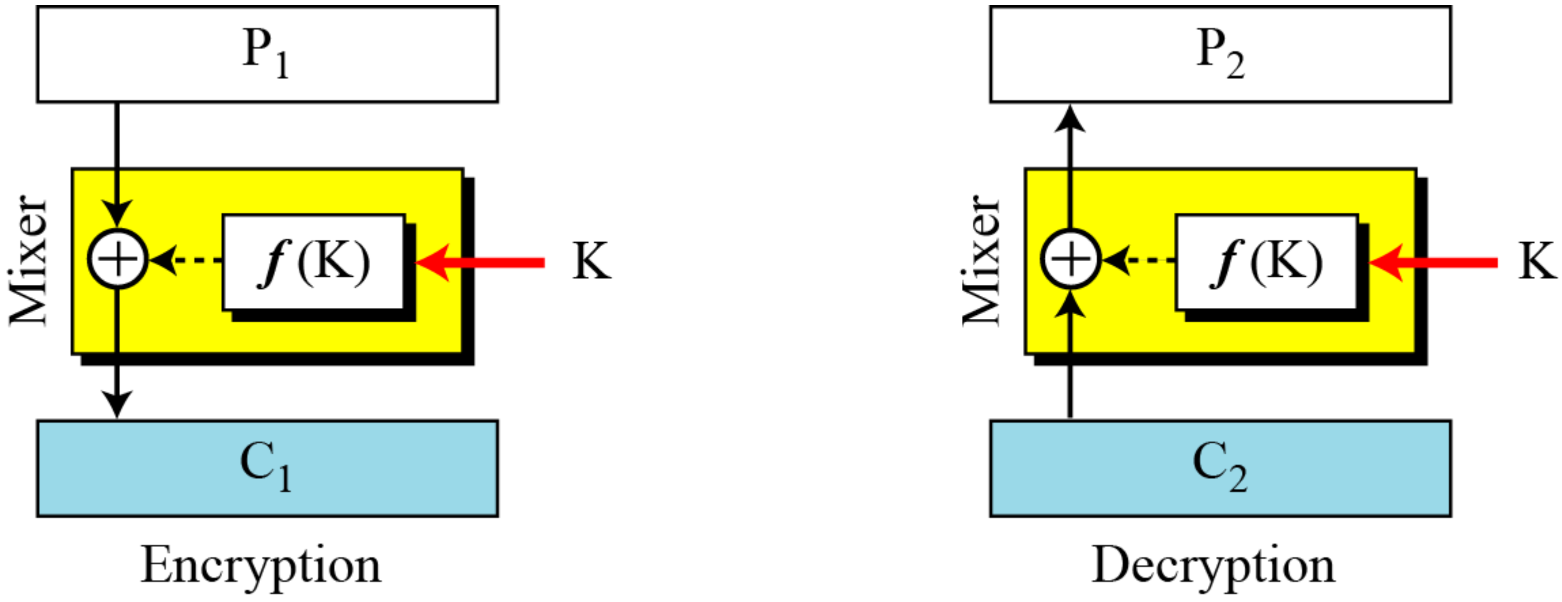
- ***Encryption:*** $C1=P1 \oplus f(K)$
- ***Decryption:*** $P2=C2 \oplus f(K)=C1 \oplus f(K)$
- $=P1 \oplus f(K) \oplus f(K)$
- $=P1$



5.1.5 Continued

- ***Although the Mixer has a non-invertible element f(K), The Mixer is self-invertible***

- ***The Mixer is Feistel Design is Self Invertible***

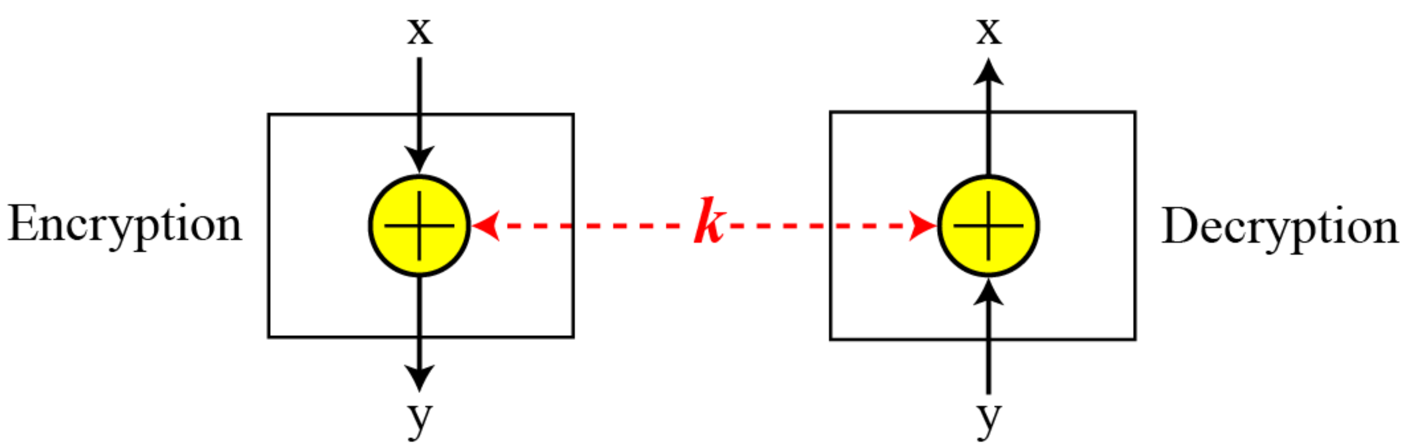


Inverse property of EXOR

The inverse of EXOR makes sense only if one of the inputs is fixed,

Example- If one of the input is key, which is normally same in Encryption and Decryption, EXOR is self invertible

y=x EXOR k
x=y EXOR k



Example 5.12

The plaintext and ciphertext are each 4 bits long and the key is 3 bits long. Assume that the function takes the first and third bits of the key, interprets these two bits as a decimal number, squares the number, and interprets the result as a 4-bit binary pattern. Show the results of encryption and decryption if the original plaintext is 0111 and the key is 101.

Example 5.12

This is a trivial example. The plaintext and ciphertext are each 4 bits long and the key is 3 bits long. Assume that the function takes the first and third bits of the key, interprets these two bits as a decimal number, squares the number, and interprets the result as a 4-bit binary pattern. Show the results of encryption and decryption if the original plaintext is 0111 and the key is 101.

Solution

Key=101

Encryption-

The function extracts the first and second bits to get 11 in binary or 3 in decimal. The result of squaring is 9, which is 1001 in binary.

f(K)=1001

B3	B2	B1
1	0	1

Decryption-

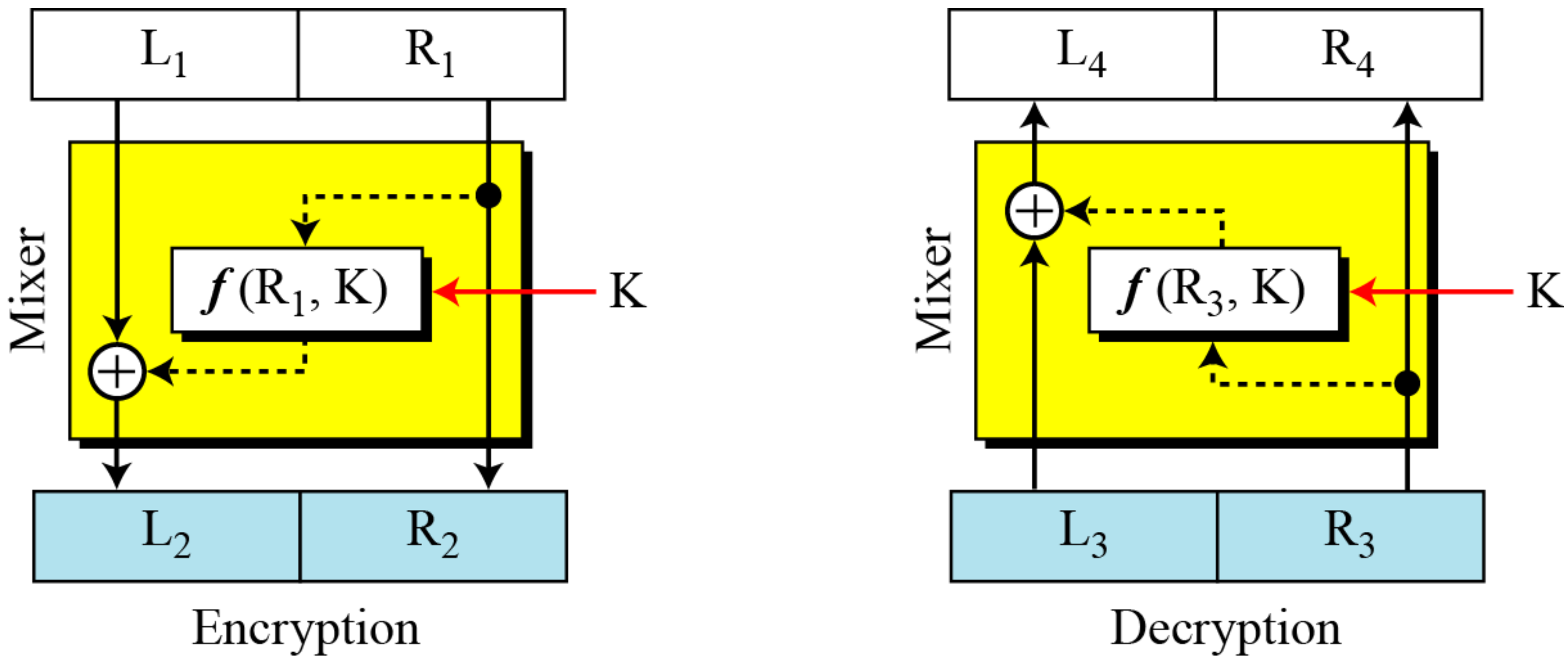
The same function can be used

$$\textbf{Encryption: } C = P \oplus f\,(K) = 0111 \oplus 1001 = 1110$$

$$\textbf{Decryption: } P = C \oplus f\,(K) = 1110 \oplus 1001 = 0111$$

5.1.5 Continued

Improvement of the previous Feistel design

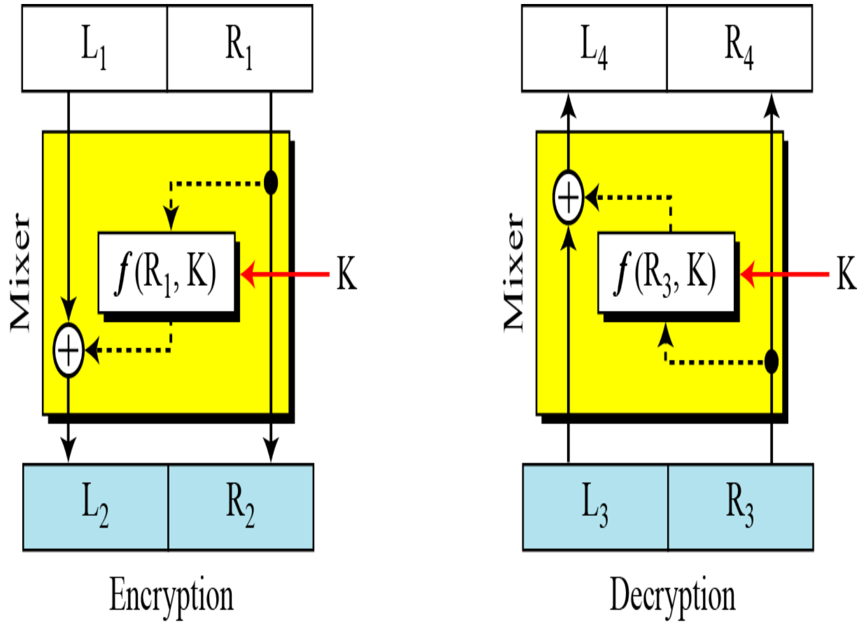


5.1.5 Continued

Improvement of the previous Feistel design

Encryption-

- ***Input to the non-invertible element i.e. function should be same***
- ***In addition to Key***
- ***Input of fn to also be part of the plaintext in the encryption and part of ciphertext in the decryption***

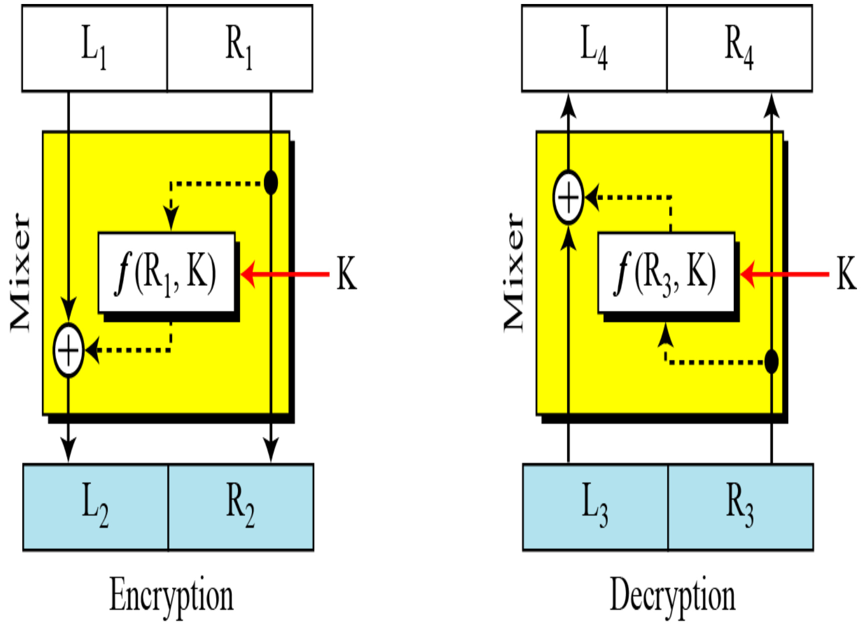


5.1.5 Continued

Improvement of the previous Feistel design

Encryption-

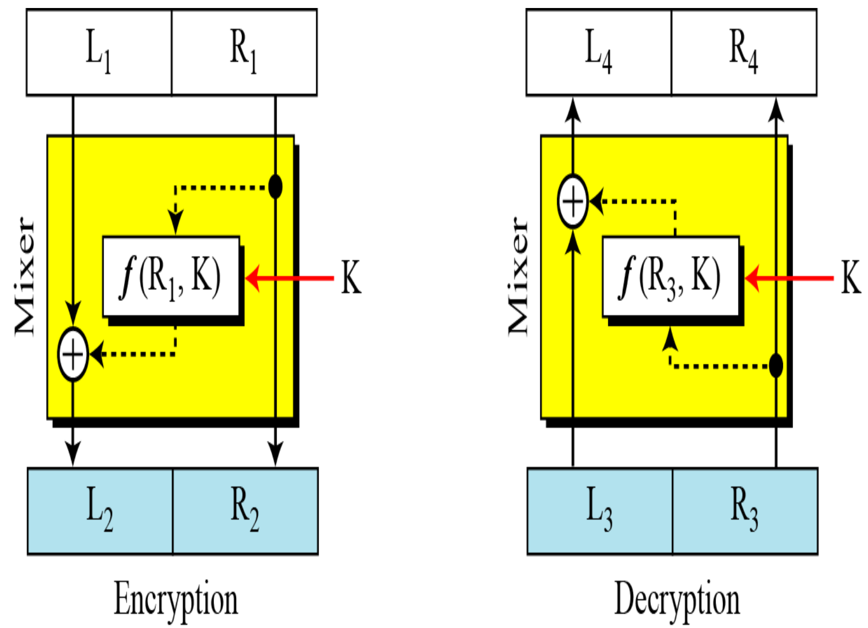
- ***Divide plain text into two equal length blocks,***
- ***Left Block and Right Block***
- ***Right Block be the input to the function***
- ***Left block be EXOred with the function ouput***



5.1.5 Continued

Improvement of the previous Feistel design

- ***Input to the function must be exactly the same in Encryption and Decryption***
- ***Right section of the plain text in Encryption and right section of the ciphertext in decryption must be same.***
- ***Right section must go into and out of the encryption and decryption process unchanged***



5.1.5 Continued

Improvement of the previous Feistel design

Assume

L3=L2 and R3=R2

No change in the ciphertext during transmission

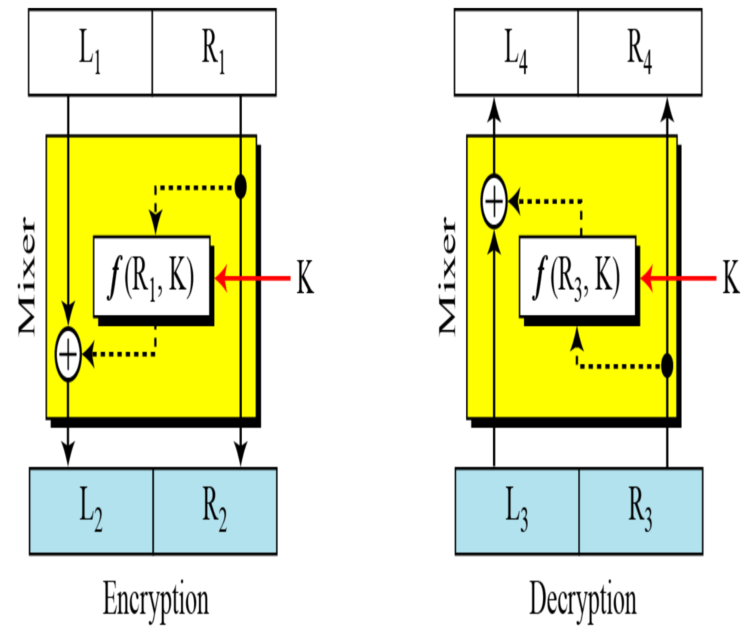
R4=R3=R2=R1

$L4 = L3 \oplus f(R3,K)$

$\quad = L2 \oplus f(R2,K)$

$\quad = L1 \oplus f(R1,K) \oplus f(R1,K)$

$\quad = L1$

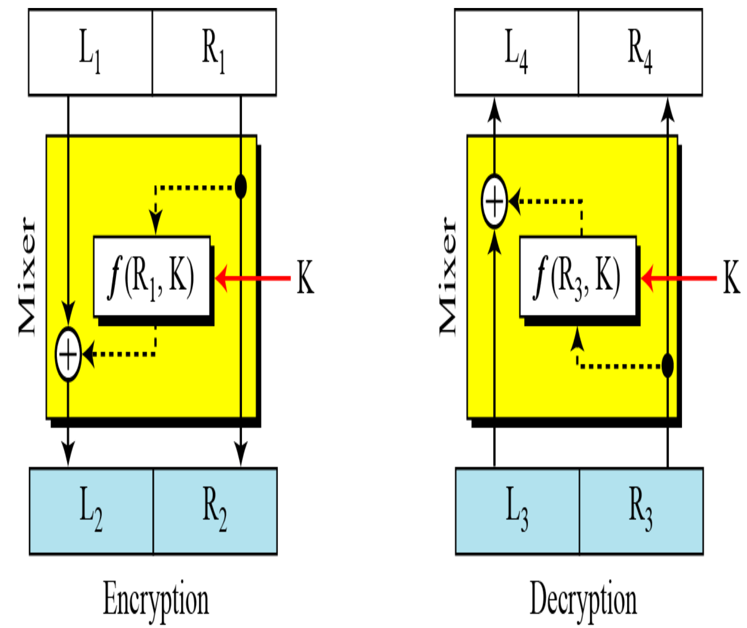


5.1.5 Continued

Improvement of the previous Feistel design

- ***Plaintext used in encryption is correctly regenerated***

- ***Encryption and Decryption are Inverses of each other***



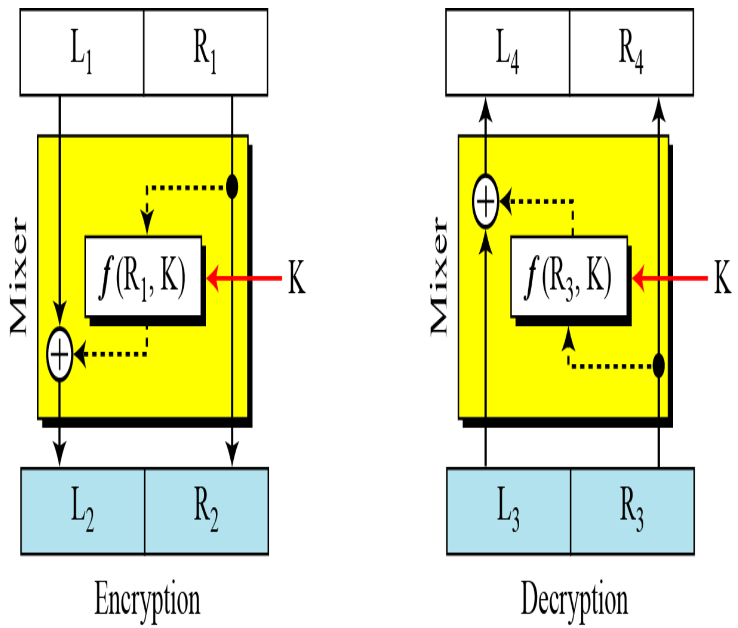
5.1.5 Continued

Improvement of the previous Feistel design

Drawback-

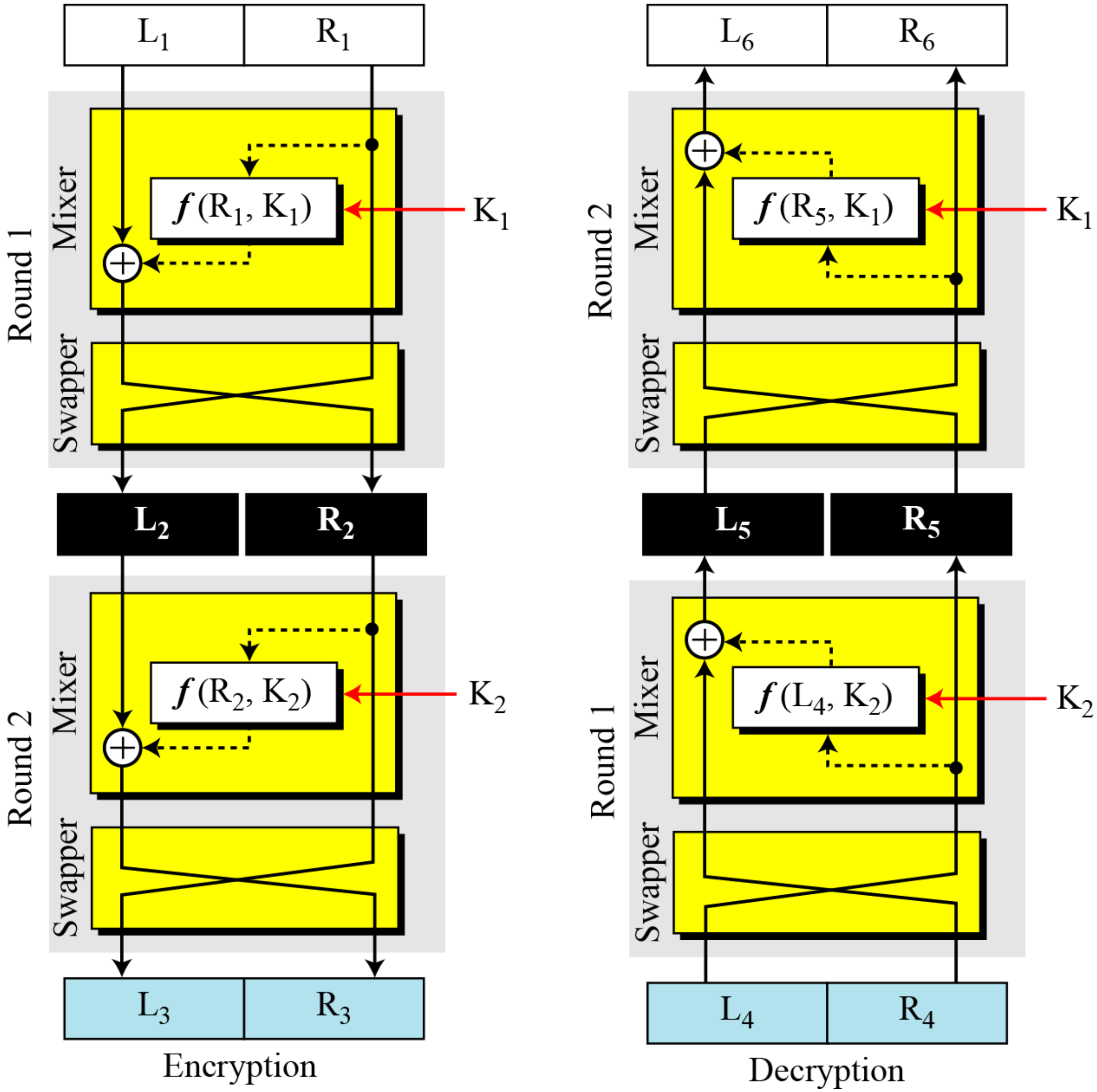
- ***Right half of the plaintext never changes***

- ***Cryptanalyst can find the Right half of the plaintext by intercepting the cipher text and extracting the right half of it.***



5.1.5 Continued

Final design of a Feistel cipher with two rounds

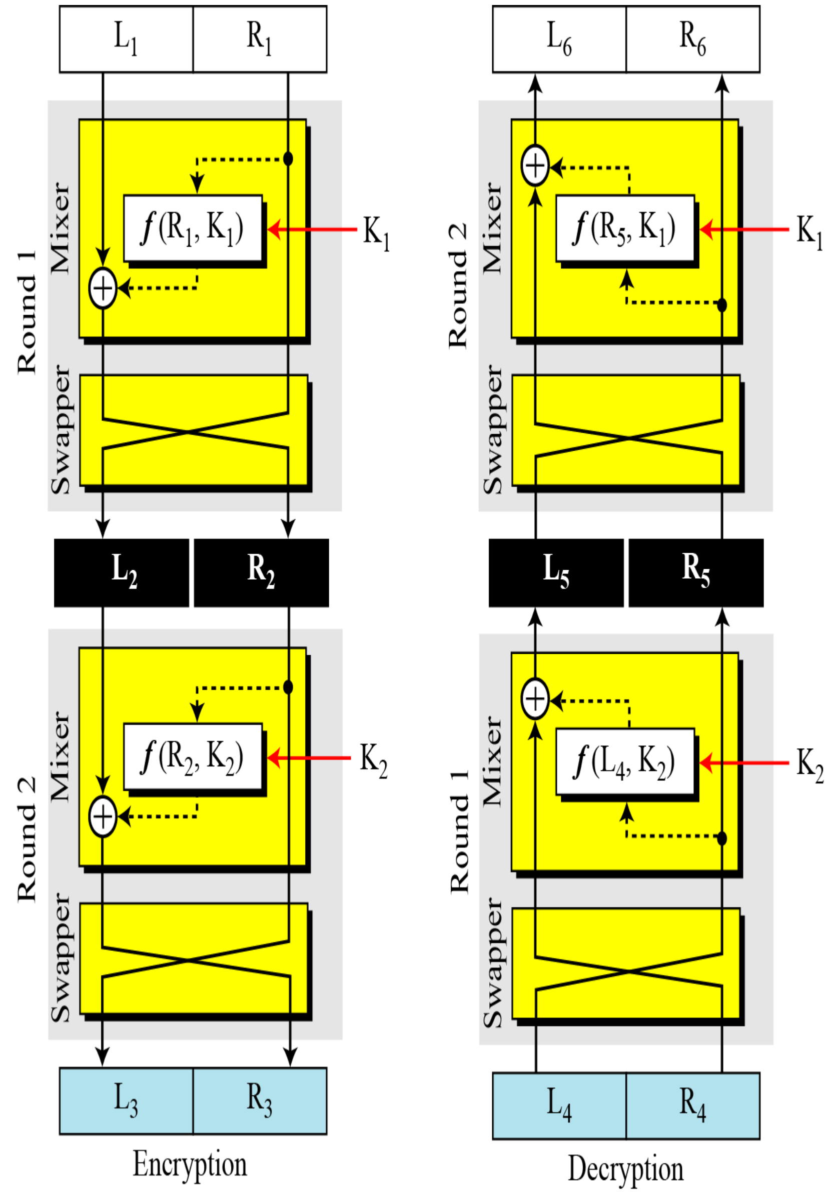


5.1.5 Continued

Final design of a Feistel cipher with two rounds

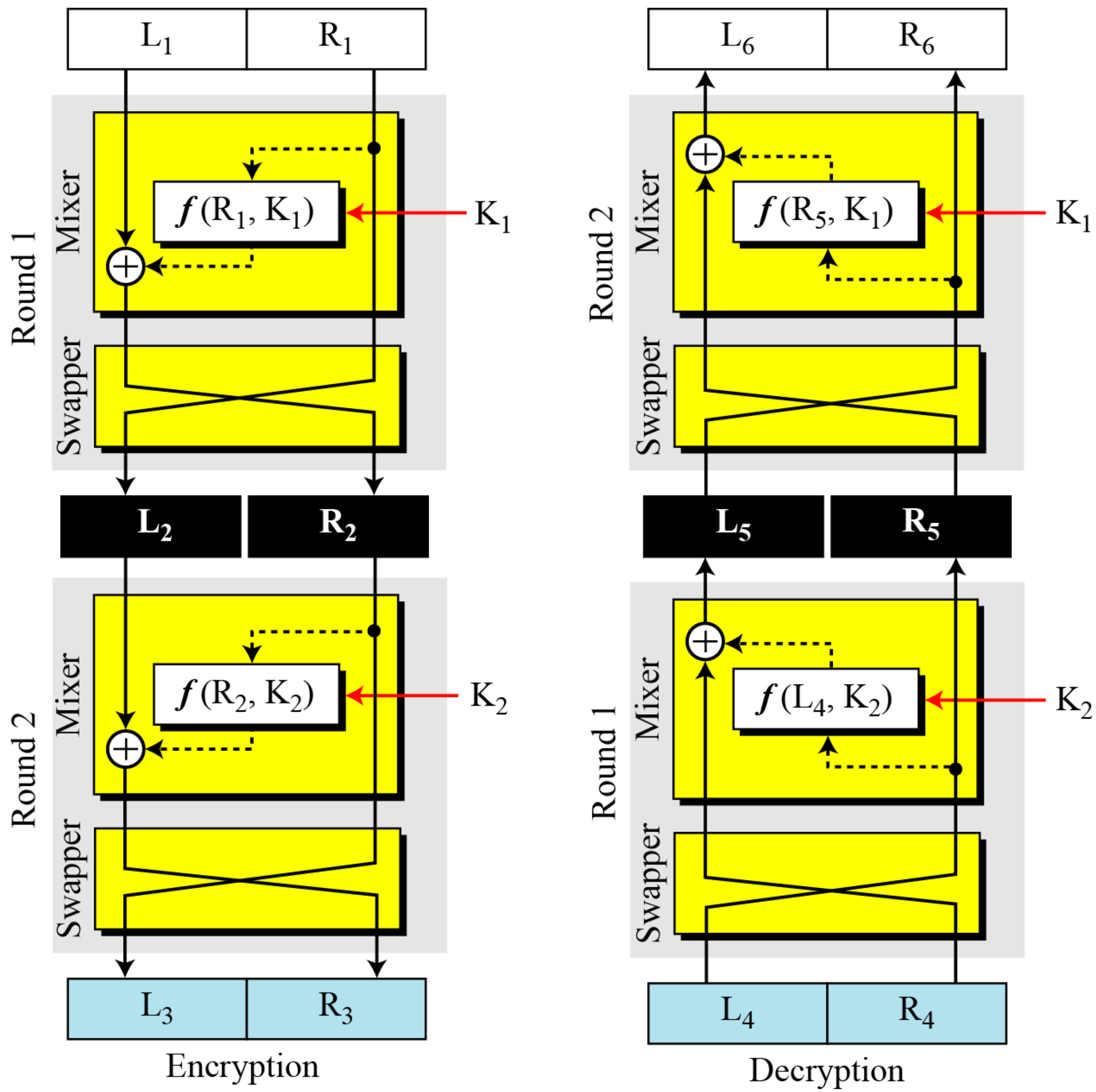
Improvement done-

- ***Increase the Number of Rounds***
- ***Add Swapper to each Round***
- ***Effect of swapper in encryption is cancelled by the effect of the swapper in decryption round***
- ***Two keys K1 and K2***
- ***Used in Reverse order in the encryption and decryption***



5.1.5 Continued

Figure 5.17 *Final design of a Feistel cipher with two rounds*



5.1.5 Continued

Non-Feistel Ciphers

A non-Feistel cipher uses only invertible components. A component in the encryption cipher has the corresponding component in the decryption cipher.

5.1.5 Continued

Non-Feistel Ciphers

Example-

S Boxes need to have equal number of inputs and outputs

No Compression or Expansion P Box allowed as they are not invertible

A 2 X 2 S Box can be designed to be invertible

A straight P Box can be designed to be invertible by using the appropriate permutation table.

The Data Encryption Standard (DES)

6-1 INTRODUCTION

The Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST).

6.1.1 History

In 1973, NIST published a request for proposals for a national symmetric-key cryptosystem. A proposal from IBM, a modification of a project called Lucifer, was accepted as DES. DES was published in the Federal Register in March 1975 as a draft of the Federal Information Processing Standard (FIPS).

6.1.1 History

- ***Finally published as FIPS 46 in the Federal Register in January 1977***

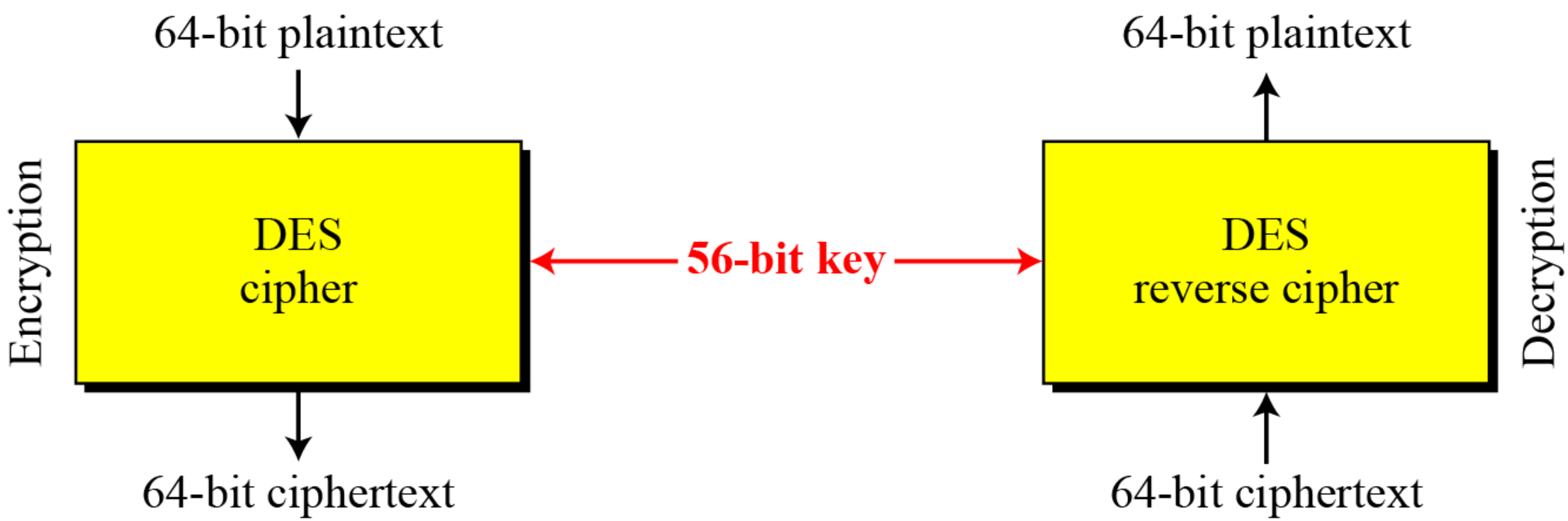
- ***A new standard FIPS 46-3 recommended the use of Triple DES- repeated DES cipher three times***

- ***AES, The recent standard is supposed to replace DES in the long run***

6.1.2 Overview

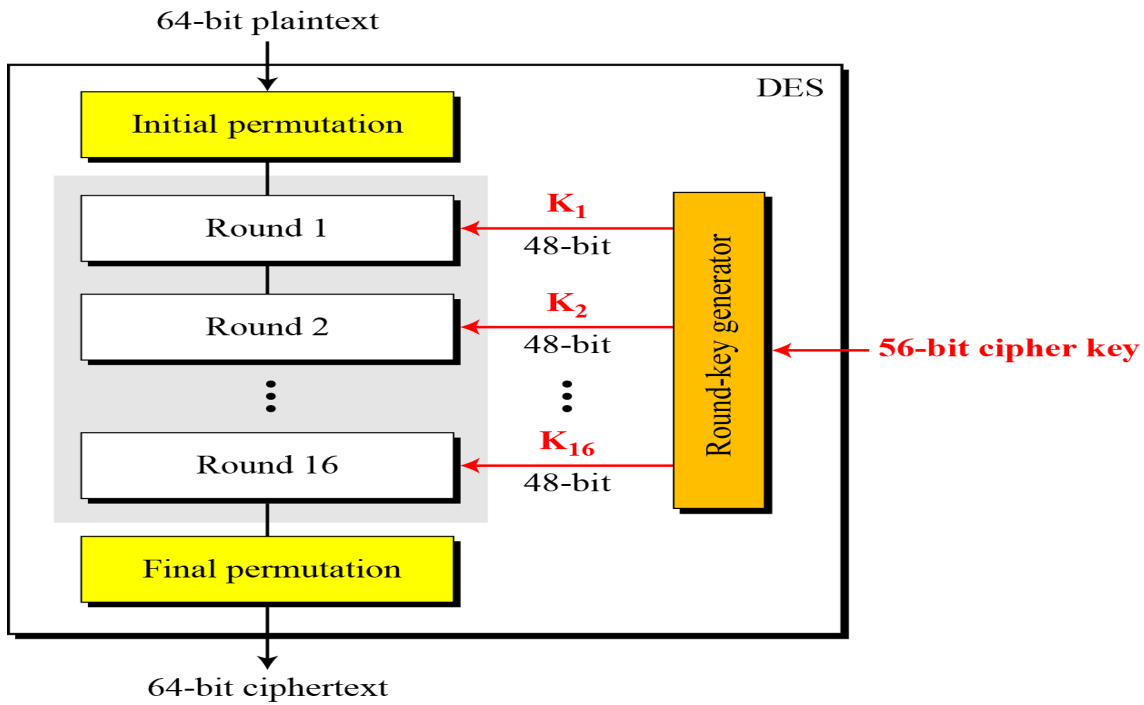
DES is a block cipher, as shown in Figure 6.1.

Figure 6.1 ***Encryption and decryption with DES***



6-2 DES STRUCTURE

The encryption process is made of two permutations (P-boxes), which we call initial and final permutations, and sixteen Feistel rounds.



Each round uses a different 48 bit round key generated from the cipher key according to a predefined algorithm

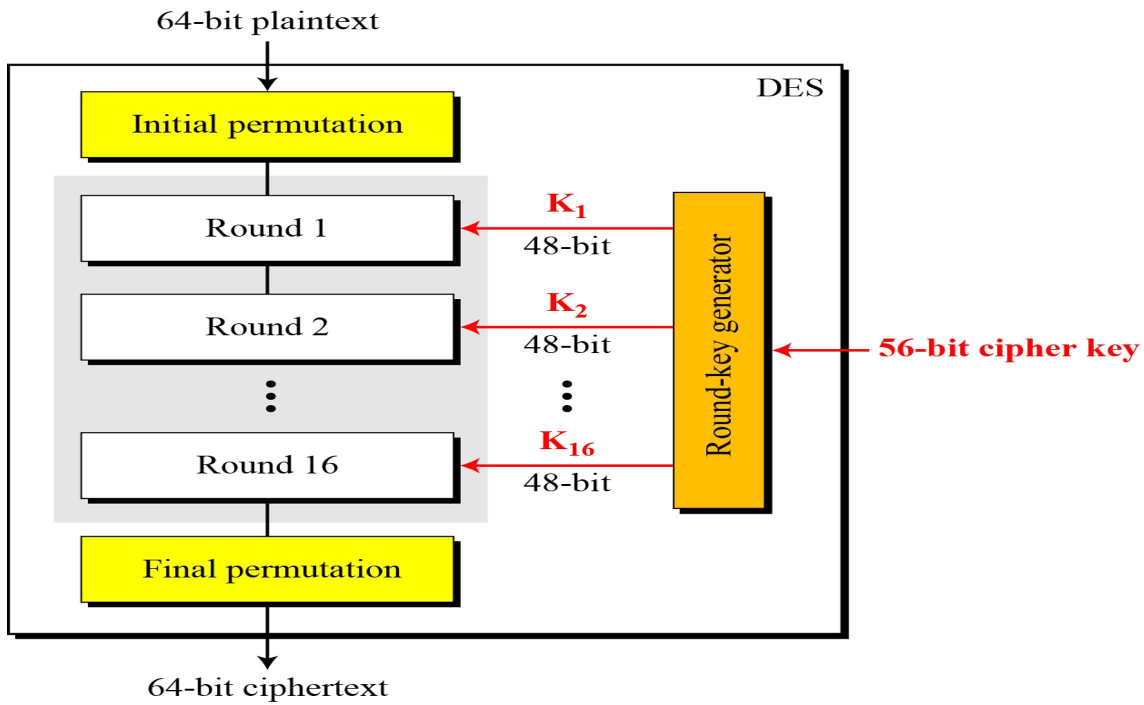
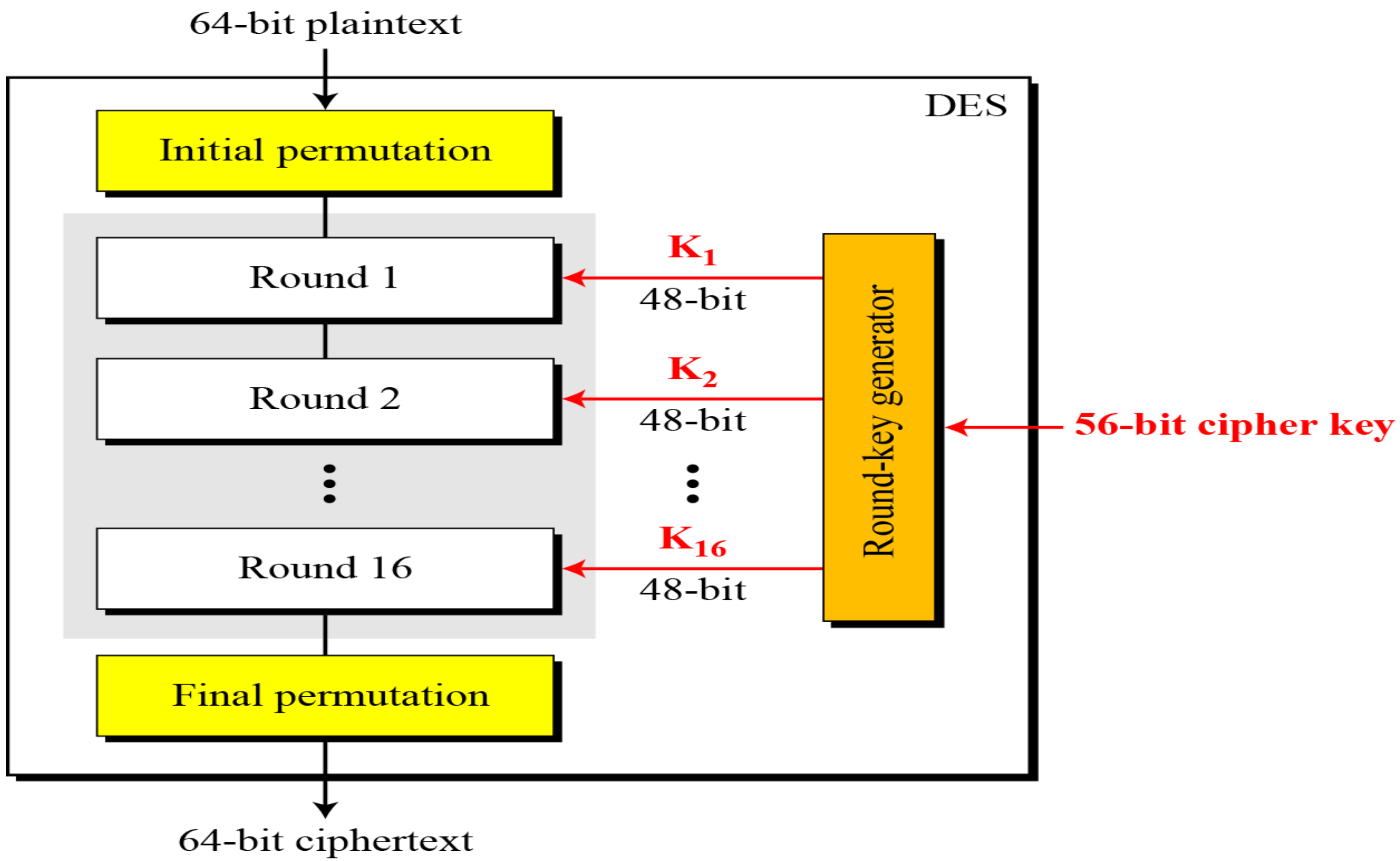


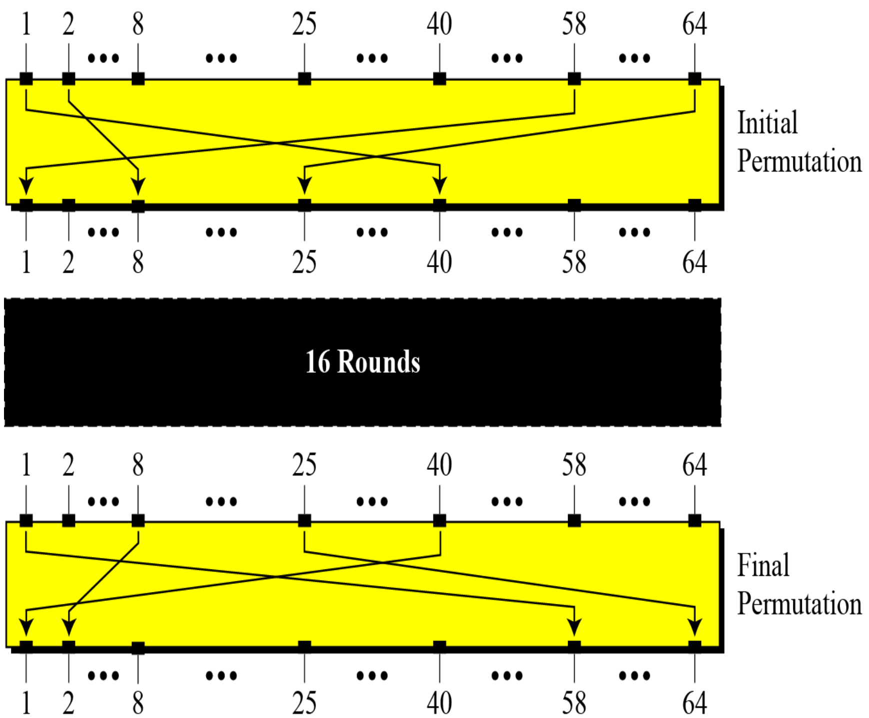
Figure 6.2 ***General structure of DES***



6.2.1 Initial and Final Permutations

Figure 6.3 ***Initial and final permutation steps in DES***

- *Takes 64 bit input*
- *Permutes them according to a predefined rule*
- *If the rounds between these two permutations do not exist, the 58th bit entering the initial permutation is the same as 58th bit leaving the final permutation*



6.2.1 Continue

Table 6.1 ***Initial and final permutation tables***

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

Permutation Rules for these P Boxes

6.2.1 Continued

Note

The initial and final permutations are keyless straight P-boxes that are inverses of each other. They have no cryptography significance in DES.

6.2.1 Continued

Example 6.1

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

0x0002 0000 0000 0001

6.2.1 Continued

Example 6.1

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

Solution

0x0002 0000 0000 0001

Bit 15 Bit 64

The input has only two 1's (Bit 15 and 64), The output may also have only two 1s (Straight Permutation)

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

Bit 15 in input becomes Bit 63 in output
Bit 64 in Input Becomes Bit 25 in output
The output has only two 1's in Bit 25 and 63
Result-

0x0000 0080 0000 0002

6.2.1 Continued

Example 6.1

0x0002 0000 0000 0001

0000 0000 0000 0010   0000 0000 0000 0000   0000 0000 0000 0000   0000 0000 0000 0001

The input has only two 1’s (Bit 15 and 64), The output may also have only two 1s (Straight Permutation)

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

Bit 15 in input becomes Bit 63 in output

Bit 64 in Input Becomes Bit 25 in output

The output has only two 1’s in Bit 25 and 63

Result-

0000 0000 0000 0010   0000 0000 1000 0000   0000 0000 0000 0000   0000 0000 0000 0010

0x0000 0080 0000 0002

6.2.1 *Continued*

Example 6.2

Prove that the initial and final permutations are the inverse of each other by finding the output of the final permutation if the input is

0x0000 0080 0000 0002

6.2.1 Continued

Example 6.2

Prove that the initial and final permutations are the inverse of each other by finding the output of the final permutation if the input is

0x0000 0080 0000 0002

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

Solution

Only Bit 25 and Bit 63 are 1s
In the final permutation ,
Bit 25 becomes Bit 64 and Bit 63 becomes Bit 15
The result in hexadecimal is

0x0002 0000 0000 0001

6.2.2 Rounds

- ***DES uses 16 rounds.***
- ***Each round of DES is a Feistel cipher.***
- ***(I-1)th Input from previous round, Creates L_I,R_I to go to next round***

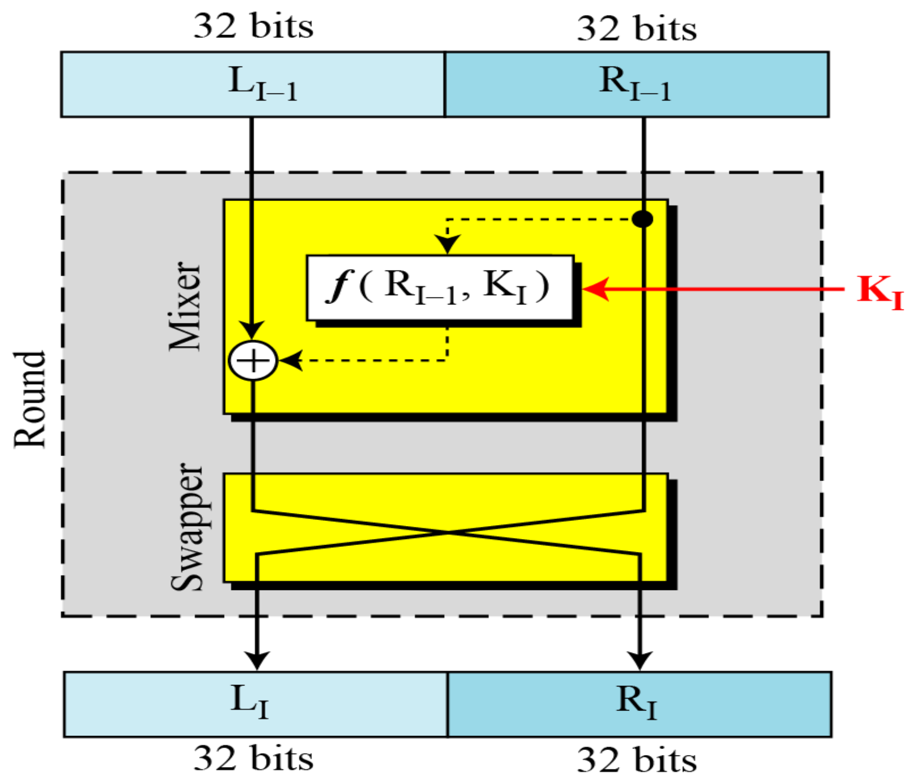


Figure 6.4
A round in DES (encryption site)

6.2.2 Rounds

- *Swapper is invertible*
- *Mixer is invertible because of EXOR operation*
- *All non invertible elements are collected inside the function f*

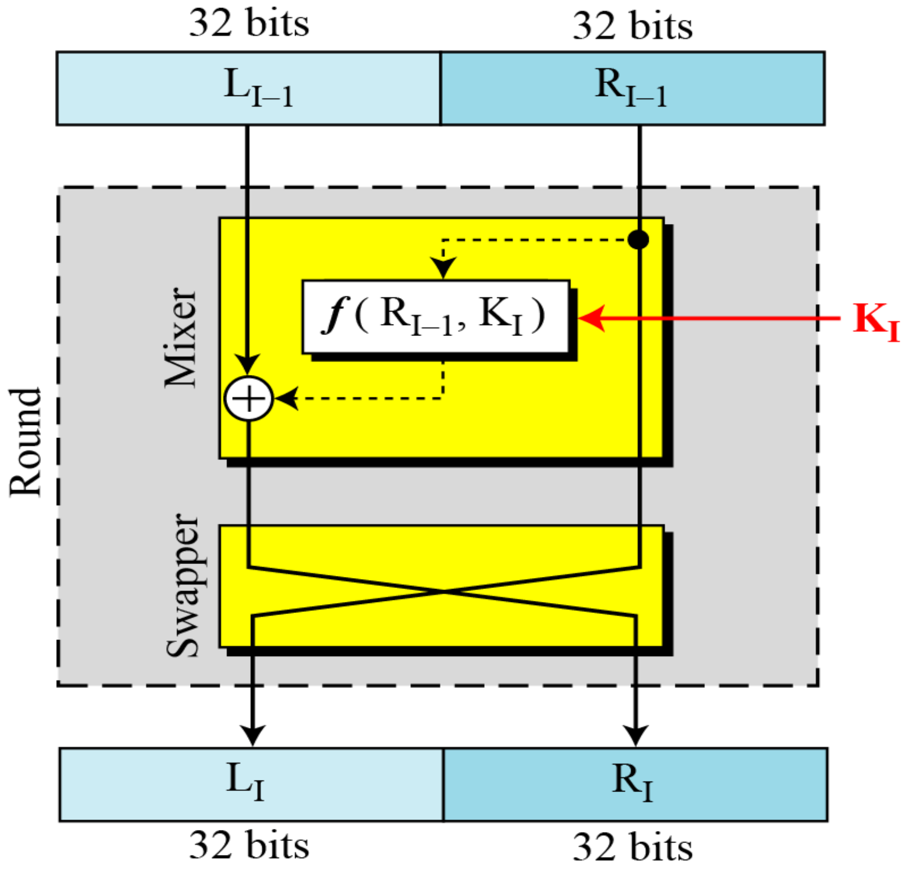
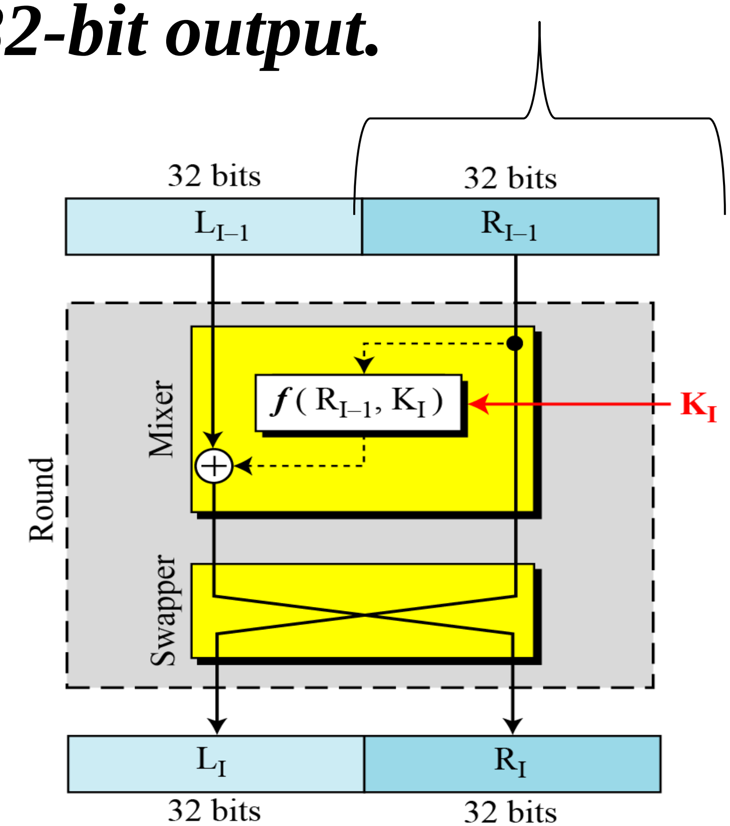


Figure 6.4
A round in DES (encryption site)

6.2.2 Continued

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.



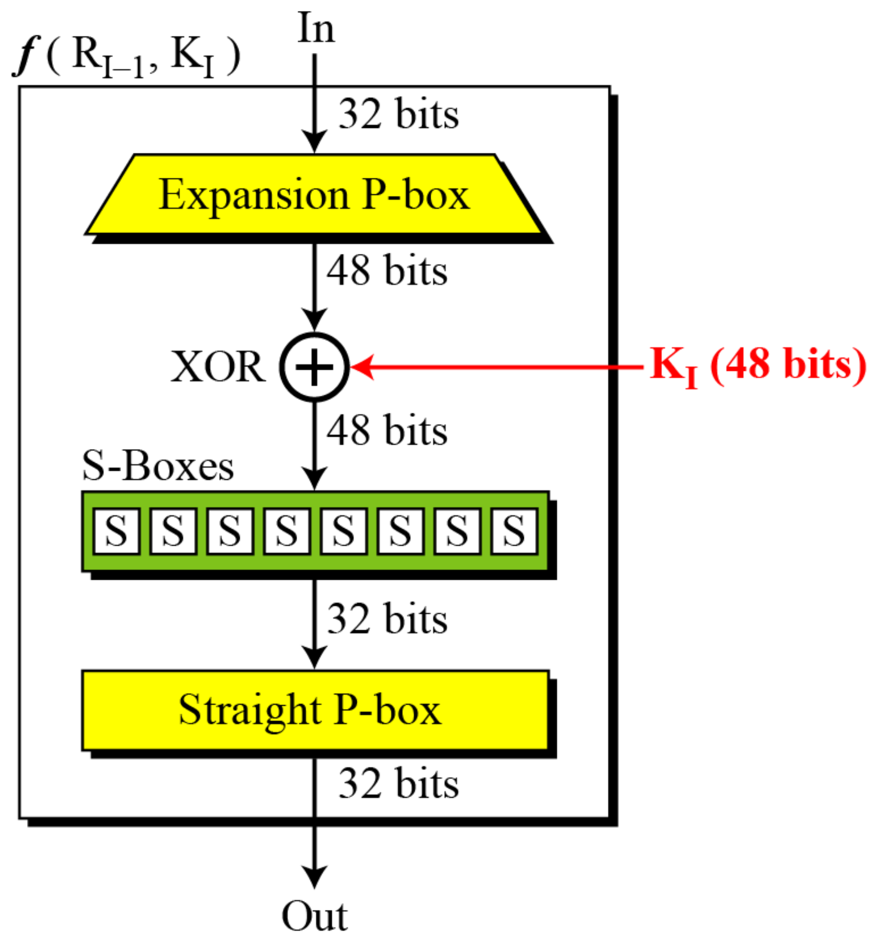


Figure 6.5
DES function

6.2.2 Continued

DES Function

Made of four sections:

- ***Expansion P Box***
- ***A whitener***
- ***Group of S Boxes***
- ***Straight P Box***

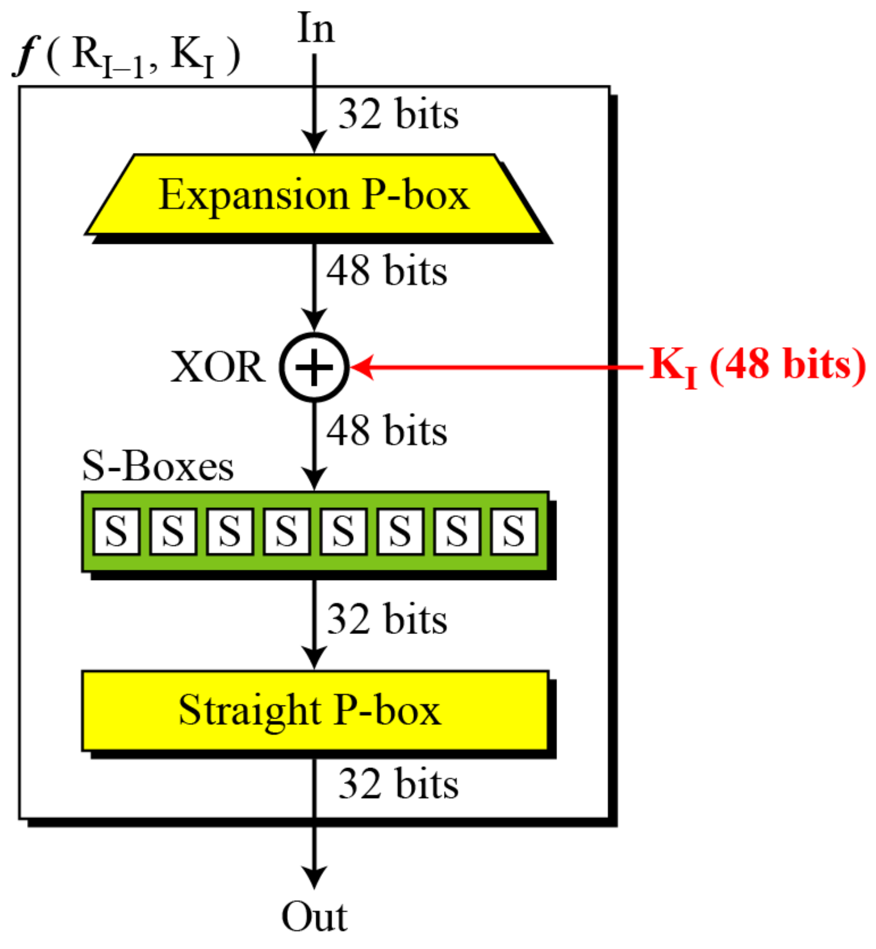


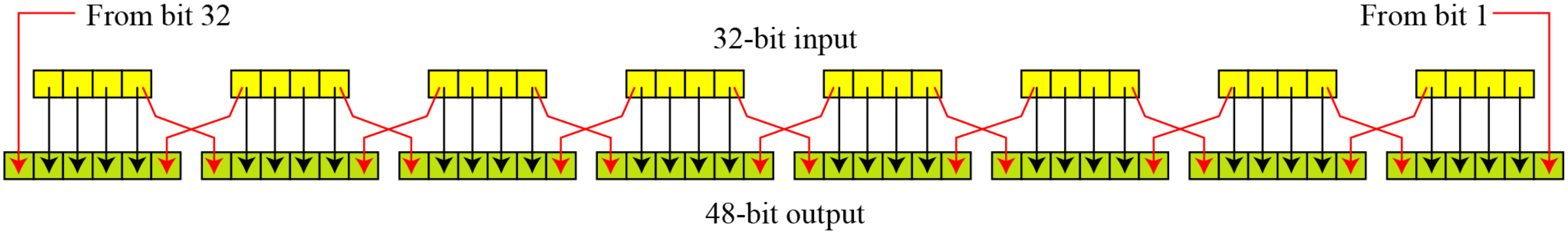
Figure 6.5
DES function

6.2.2 Continue

Expansion P-box

Since R_{I-1} is a 32-bit input and K_I is a 48-bit key, we first need to expand R_{I-1} to 48 bits.

Figure 6.6 ***Expansion permutation***

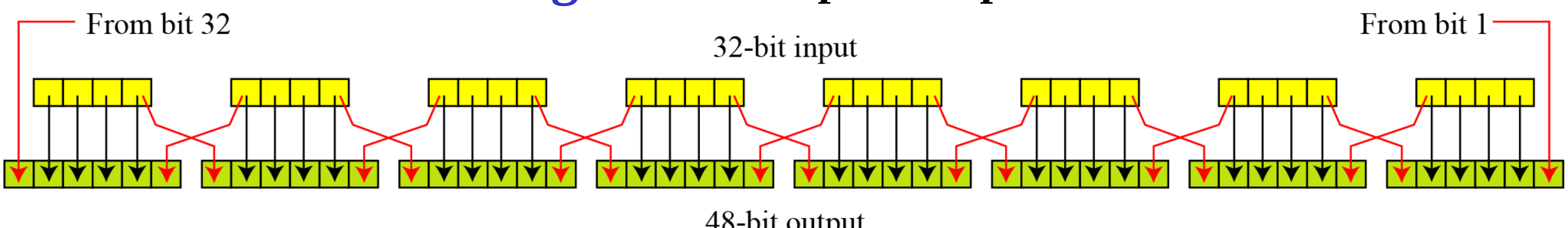


6.2.2 Continue

Expansion P-box

- *Each R_{I-1} is divided into eight 4-Bit sections*
- *Each 4 Bit section is then expanded to 6 bits*
- *Predetermined Rule used*

Figure 6.6 *Expansion permutation*



Expansion P-box

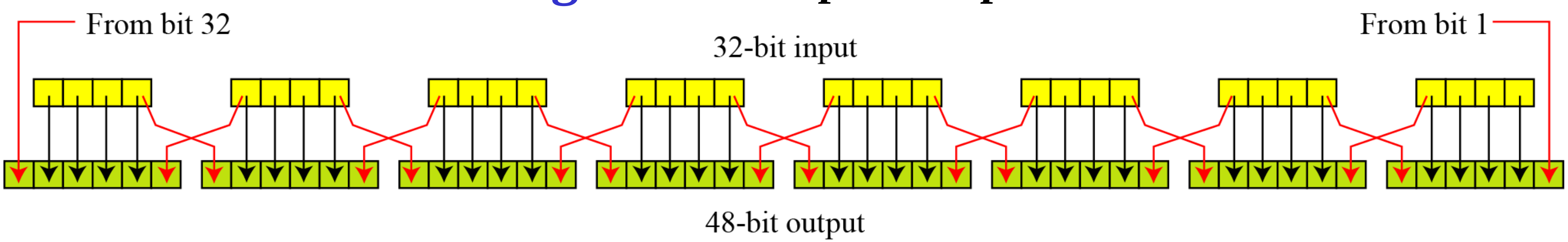
For each section, I/P bits 1,2,3 and 4 are copied to O/P bits 2,3,4 and 5 respectively.

Output Bit 1 comes from bit 4 of previous section

Output bit 6 comes from bit 4 of next section

Sections 1 and 8 are considered adjacent, Same rule applies to bits 1 and 32

Figure 6.6 ***Expansion permutation***



6.2.2 Continue

Although the relationship between the input and output can be defined mathematically, DES uses Table 6.2 to define this P-box.

A 6 X 8 Table

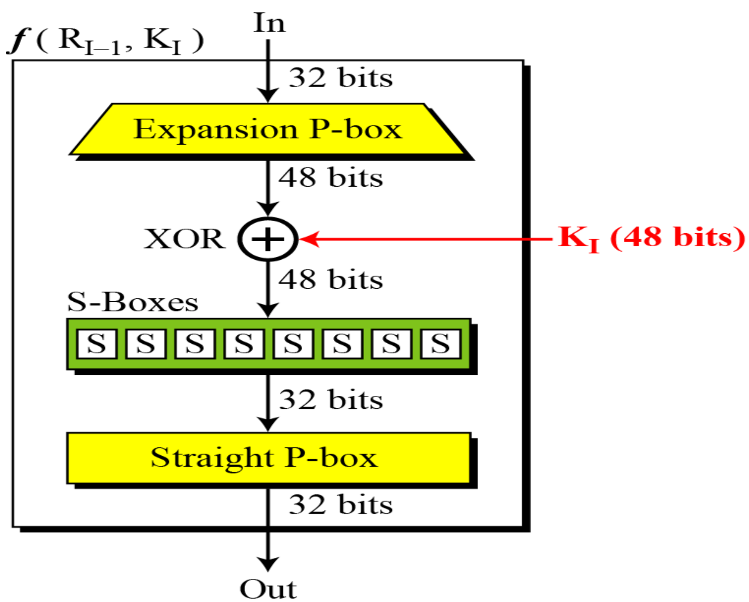
Table 6.6 *Expansion P-box table*

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

6.2.2 Continue

Whitener (XOR)

- *After the expansion permutation, DES uses the XOR operation on the expanded right section and the round key.*
- *Note that both the right section and the key are 48-bits in length.*
- *Also note that the round key is used only in this operation.*



S-Boxes

- The S-boxes do the real mixing (confusion).
- 48 bit output from Whitener is divided into Eight 6-bit chunks
- DES uses 8 S-boxes,
- Each 6-bit chunk as input and a 4-bit output.

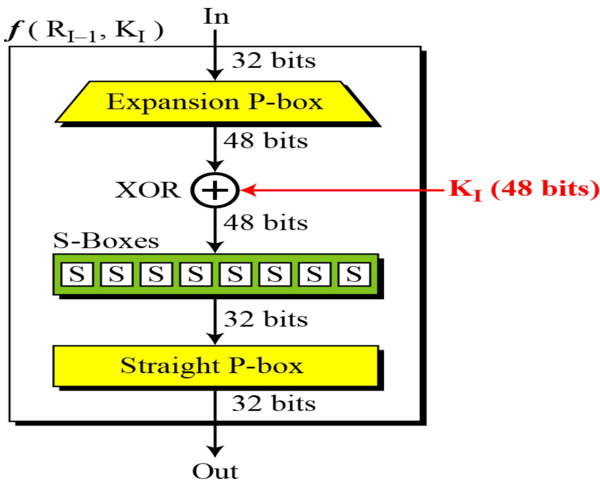
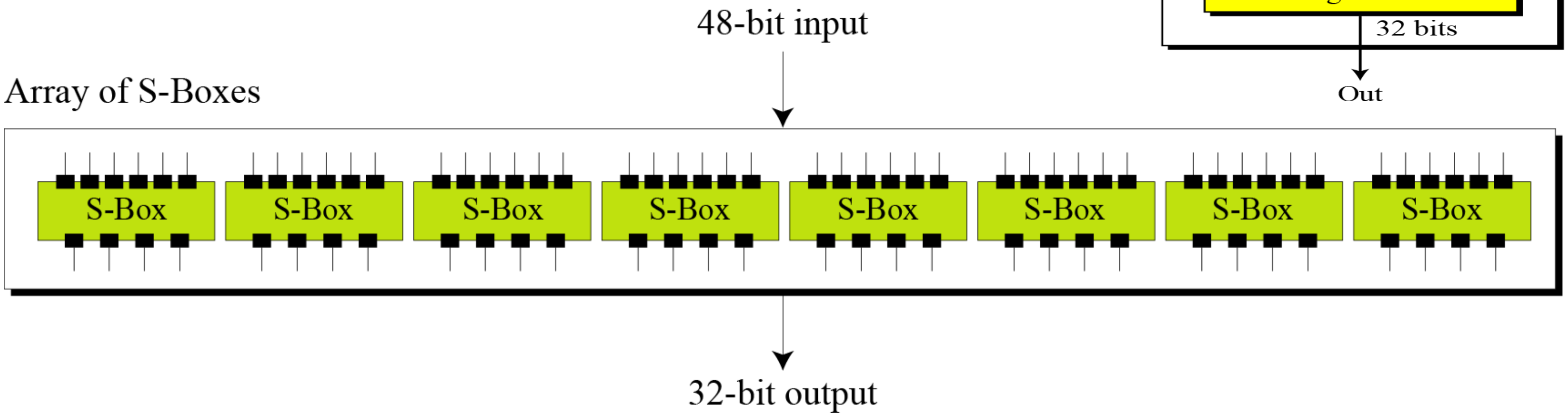


Figure 6.7 *S-boxes*



6.2.2 Continue

Figure 6.8 ***S-box rule***

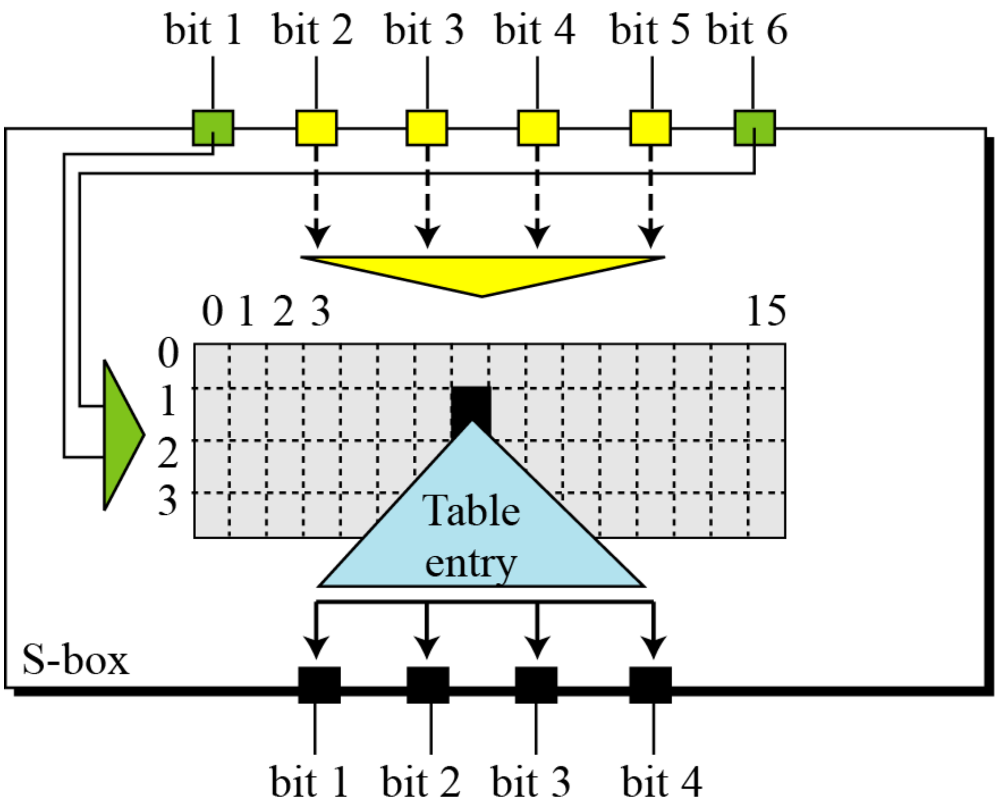


Figure 6.8 ***S-box rule***

S-Boxes

- ***The substitution in each box follows a predetermined rule***
- ***Based on a 4 row by 16 column table***
- ***Combination of bits 1 and 6 of the input defines one of the rows***
- ***Combination of bits 2 through 5 defines one of the sixteen columns***

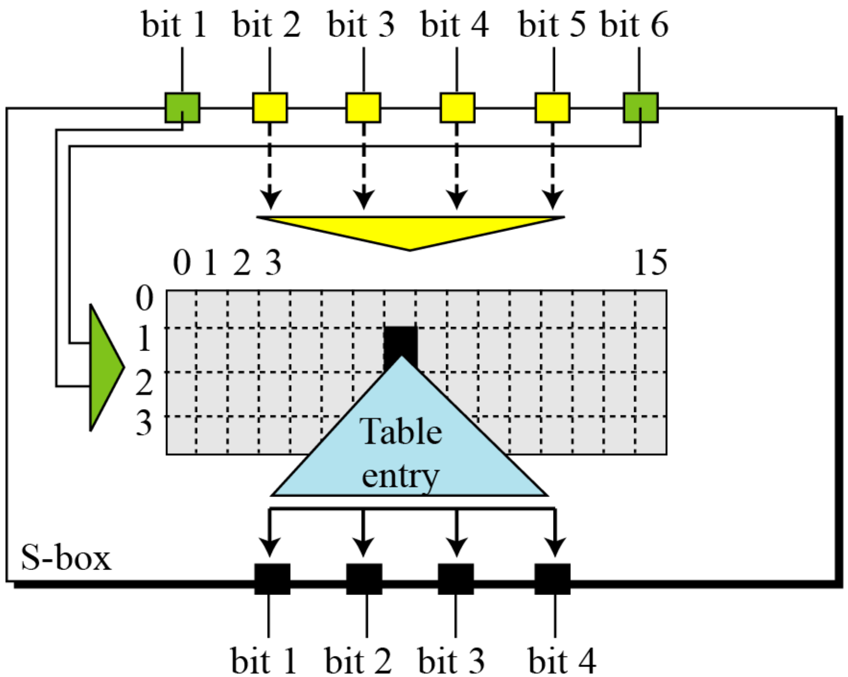
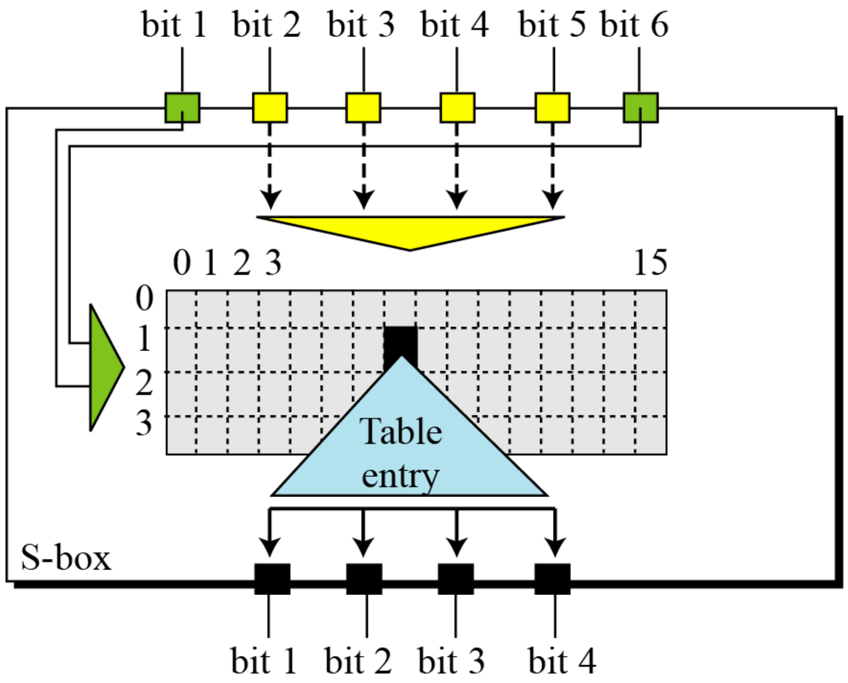


Figure 6.8 *S-box rule*

S-Boxes

- ***Each S Box has its own table***
- ***We need 8 tables***



6.2.2 Continue

- *Table 6.3 shows the permutation for S-box 1.*
- *For the rest of the boxes see the textbook.*
- *The row number and column no, output are given as decimal to save space*
- *These need to be changed to binary*

Table 6.3 *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

6.2.2 Continued

Example 6.3

The input to S-box 1 is 100011. What is the output?

	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

Solution

- If we write the first and the sixth bits together, we get 11 in binary, which is 3 in decimal.
- The remaining bits are 0001 in binary, which is 1 in decimal.
- We look for the value in row 3, column 1, in Table (S-box 1).
- The result is 12 in decimal, which in binary is 1100. So the input 100011 yields the output 1100.

6.2.2 Continued

Example 6.4

The input to S-box 8 is 000000. What is the output?

Table 6.10 *S-box 8*

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

Solution

If we write the first and the sixth bits together, we get 00 in binary, which is 0 in decimal. The remaining bits are 0000 in binary, which is 0 in decimal. We look for the value in row 0, column 0, in Table 6.10 (S-box 8). The result is 13 in decimal, which is 1101 in binary. So the input 000000 yields the output 1101.

6.2.2 Continue

Straight Permutation

Table 6.11 ***Straight permutation table***

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

6.2.2 Continue

Straight Permutation

- *Last operation in DES function*
- *Permutation with 32 bit input and 32 bit output*

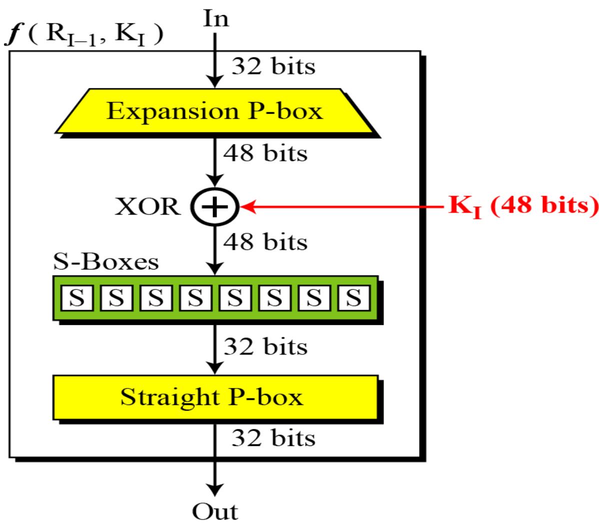


Table 6.11 *Straight permutation table*

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

6.2.3 Cipher and Reverse Cipher

Using mixers and swappers, we can create the cipher and reverse cipher, each having 16 rounds.

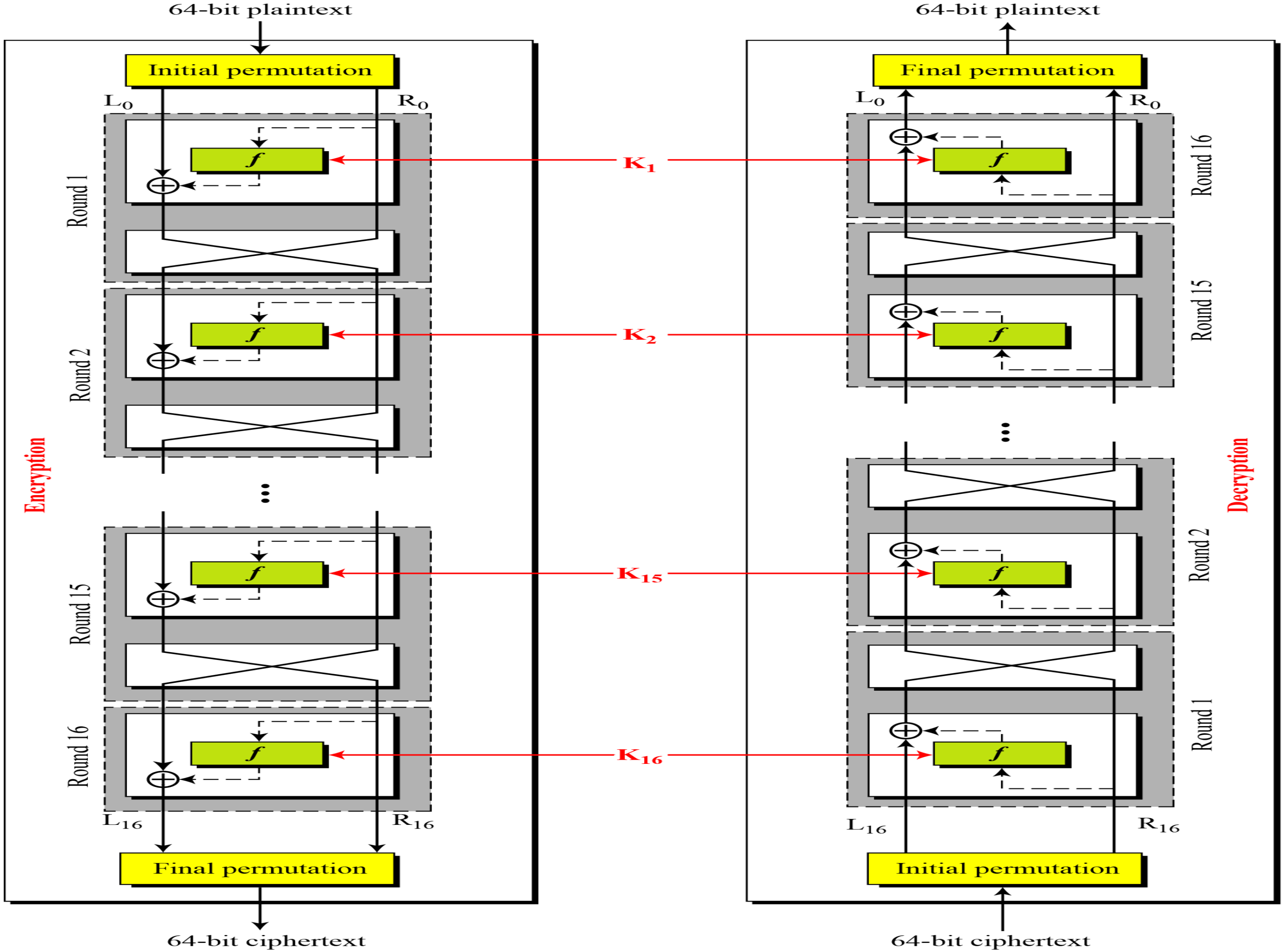
First Approach

To achieve this goal, one approach is to make the last round (round 16) different from the others; it has only a mixer and no swapper.

Note

In the first approach, there is no swapper in the last round.

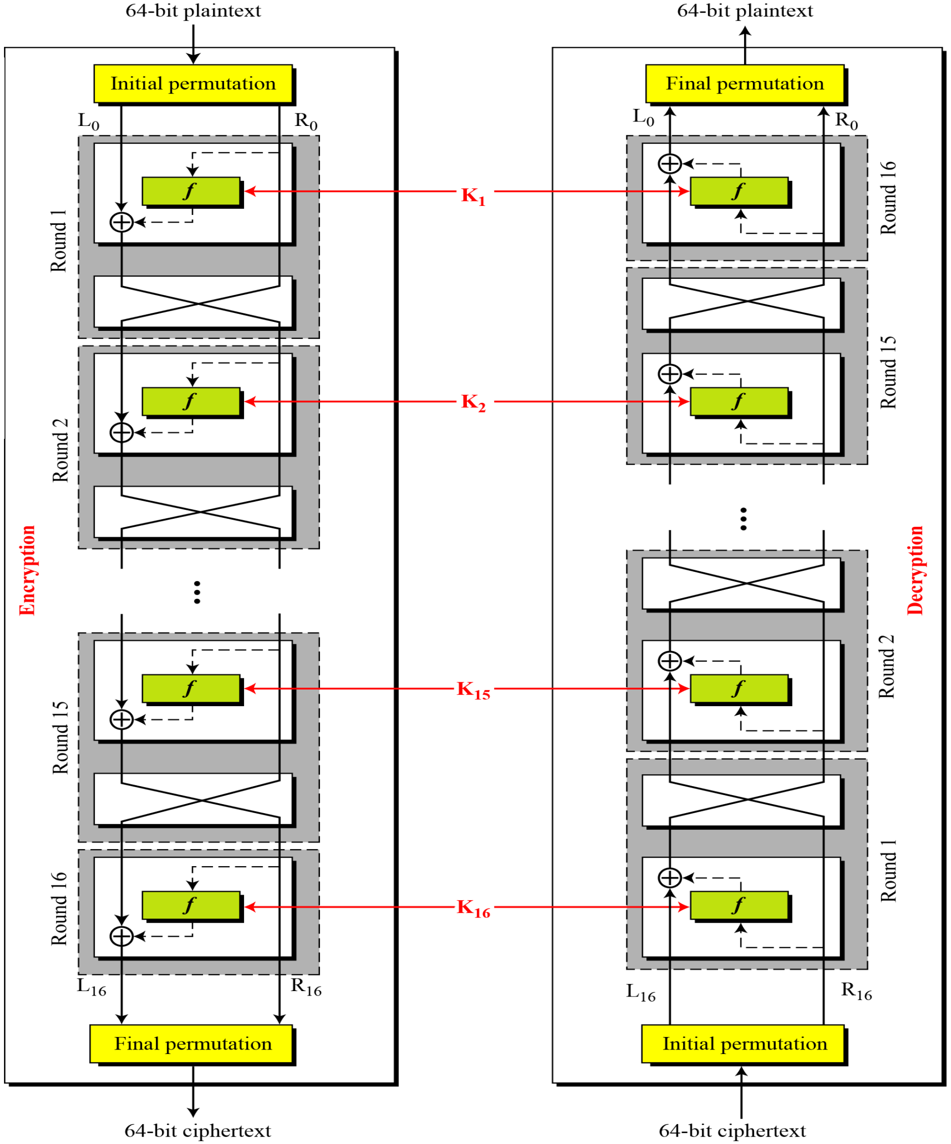
Figure 6.9 *DES cipher and reverse cipher for the first approach*



6.2.3 Continued

Figure 6.9 *DES cipher and reverse cipher for the first approach*

Round Keys(K1 to K16) should be applied in the reverse order in Decryption

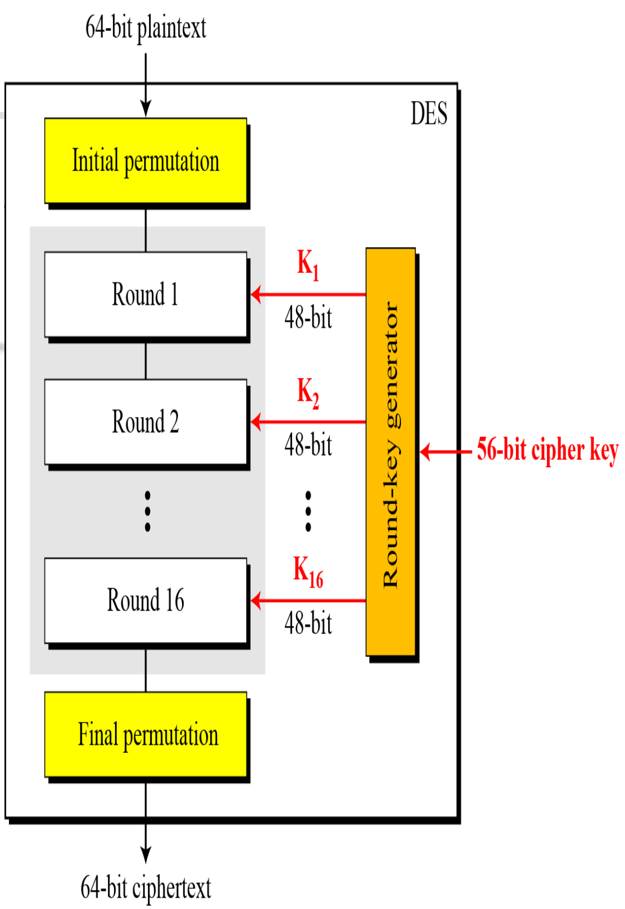


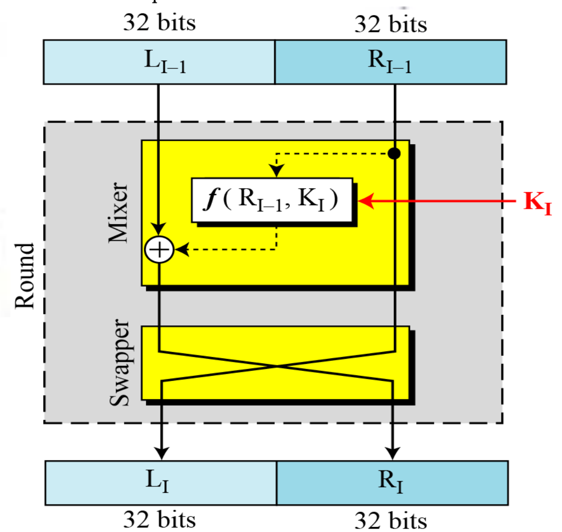
6.2.3 Continued

Algorithm 6.1 *Pseudocode for DES*

Algorithm 6.1 *Pseudocode for DES cipher*

```
Cipher (plainBlock[64], RoundKeys[16, 48], cipherBlock[64])
{
    permute (64, 64, plainBlock, inBlock, InitialPermutationTable)
    split (64, 32, inBlock, leftBlock, rightBlock)
    for (round = 1 to 16)
    {
        mixer (leftBlock, rightBlock, RoundKeys[round])
        if (round!=16) swapper (leftBlock, rightBlock)
    }
    combine (32, 64, leftBlock, rightBlock, outBlock)
    permute (64, 64, outBlock, cipherBlock, FinalPermutationTable)
}
```





6.2.3 Continued

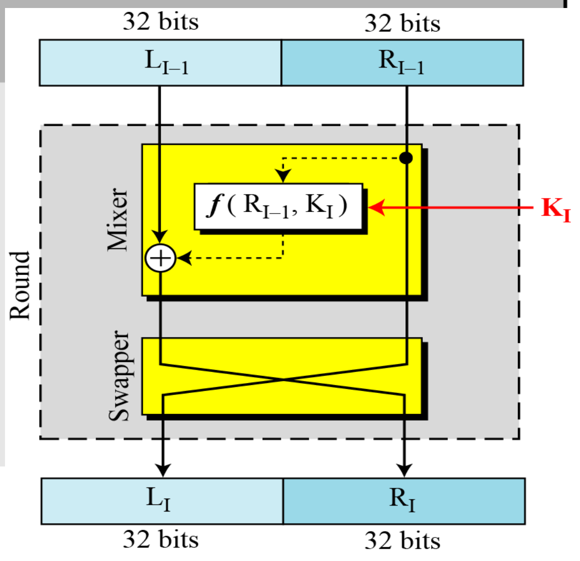
Algorithm 6.1 ***Pseudocode for DES cipher*** **(Continued)**

```
mixer (leftBlock[48], rightBlock[48], RoundKey[48])
{
    copy (32, rightBlock, T1)
    function (T1, RoundKey, T2)
    exclusiveOr (32, leftBlock, T2, T3)
    copy (32, T3, rightBlock)
}
```

- **Copy rightblock in T1**
- **Apply fn to T1,RoundKey, Store Output in T2**
- **Exor of leftblock and T2,O/P=T3**

```
swapper (leftBlock[32], rigthBlock[32])
{
    copy (32, leftBlock, T)
    copy (32, rightBlock, leftBlock)
    copy (32, T, rightBlock)
}
```

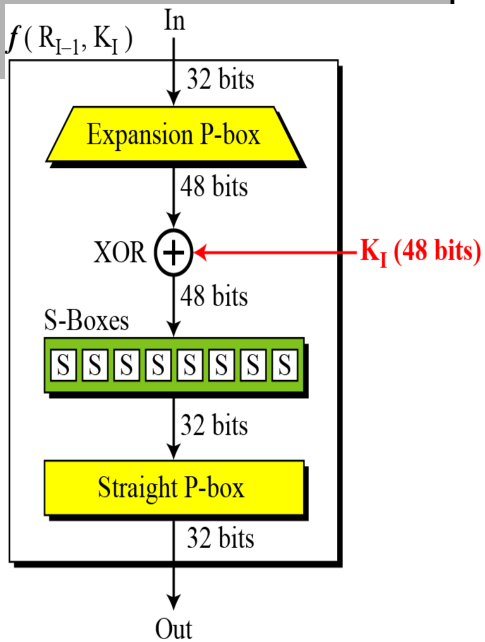
- **Copy leftblock in T**
- **Copy rightblock in leftblock**
- **Copy T in rightblock**



6.2.3 Continued

Algorithm 6.1 ***Pseudocode for DES cipher*** **(Continued)**

```
function (inBlock[32], RoundKey[48], outBlock[32])
{
    permute (32, 48, inBlock, T1, ExpansionPermutationTable)
    exclusiveOr (48, T1, RoundKey, T2)
    substitute (T2, T3, SubstituteTables)
    permute (32, 32, T3, outBlock, StraightPermutationTable)
}
```



Algorithm 6.1 *Pseudocode for DES cipher* (Continued)

```
substitute (inBlock[32], outBlock[48], SubstitutionTables[8, 4, 16])
{
    for (i = 1 to 8)
    {
        row ← 2 × inBlock[i × 6 + 1] + inBlock [i × 6 + 6]
        col ← 8 × inBlock[i × 6 + 2] + 4 × inBlock[i × 6 + 3] +
                2 × inBlock[i × 6 + 4] + inBlock[i × 6 + 5]

        value = SubstitutionTables [i][row][col]


        outBlock[[i × 4 + 1] ← value / 8;          value ← value mod 8
        outBlock[[i × 4 + 2] ← value / 4;          value ← value mod 4
        outBlock[[i × 4 + 3] ← value / 2;          value ← value mod 2
        outBlock[[i × 4 + 4] ← value
    }
}
```

6.2.3 Continued

Alternative Approach

We can make all 16 rounds the same by including one swapper to the 16th round and add an extra swapper after that (two swappers cancel the effect of each other).

Key Generation

The round-key generator creates sixteen 48-bit keys out of a 56-bit cipher key.

6.2.3 Continued

Shifting

Rounds	Shift
1, 2, 9, 16	one bit
Others	two bits

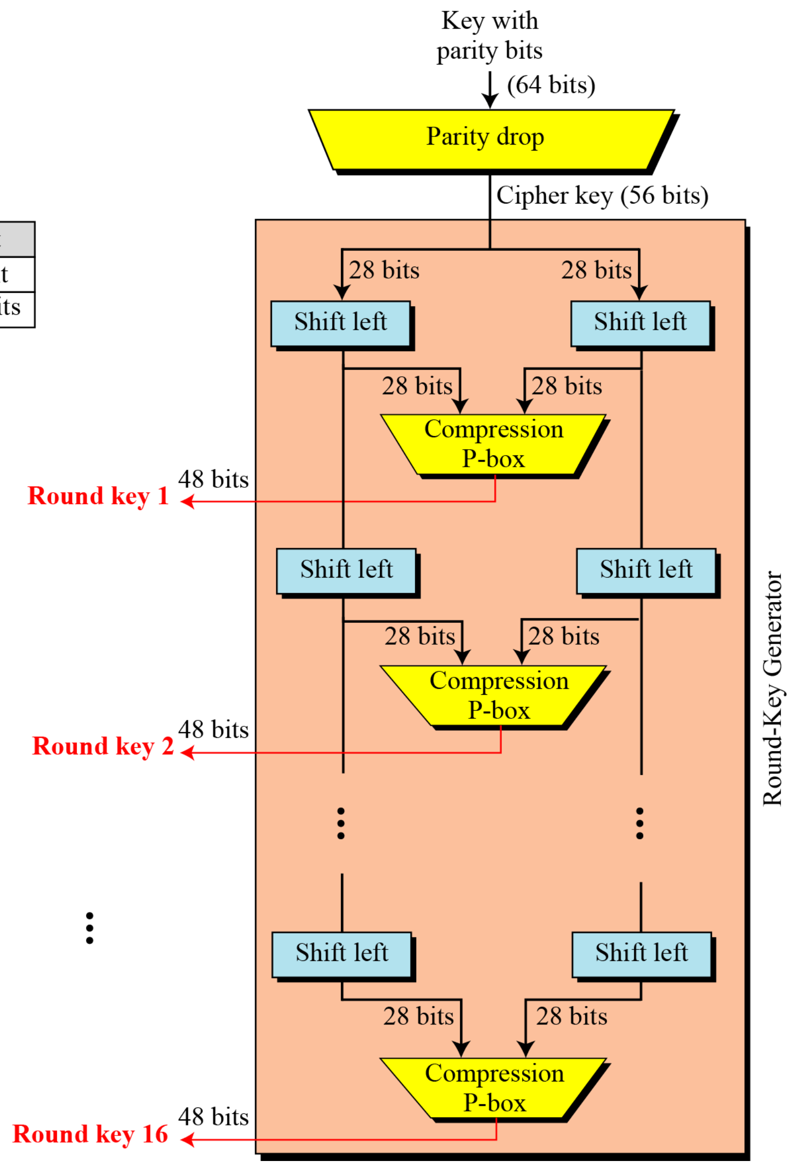


Figure 6.10

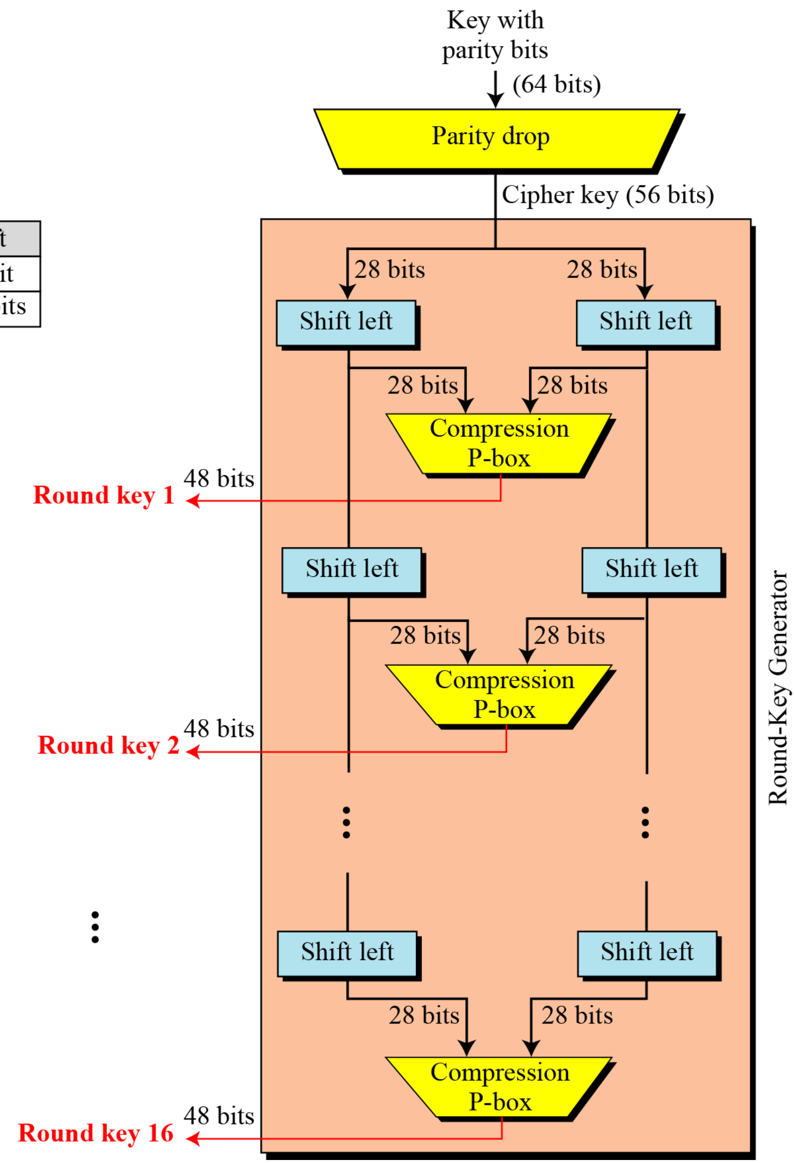
Key generation

Key Generation

Cipher key is normally given as a 64bit key in which 8 extra bits are parity bits, which are dropped of before the actual key generation process.

Shifting

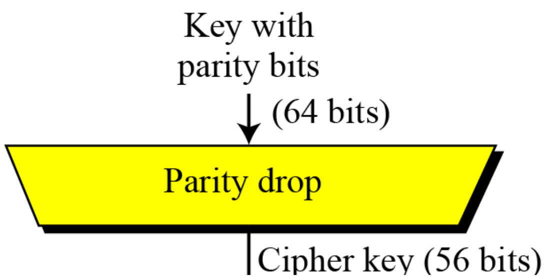
Rounds	Shift
1, 2, 9, 16	one bit
Others	two bits



6.2.3 Continued

Parity Drop

- ***Drops the parity bits***
- ***Bits 8,16,24,32,……,64 from the 64 bit key***
- ***Permutes the remaining bits according to table***
- ***The remaining 56 bit value is the actual cipher key which is used to generate round keys.***



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

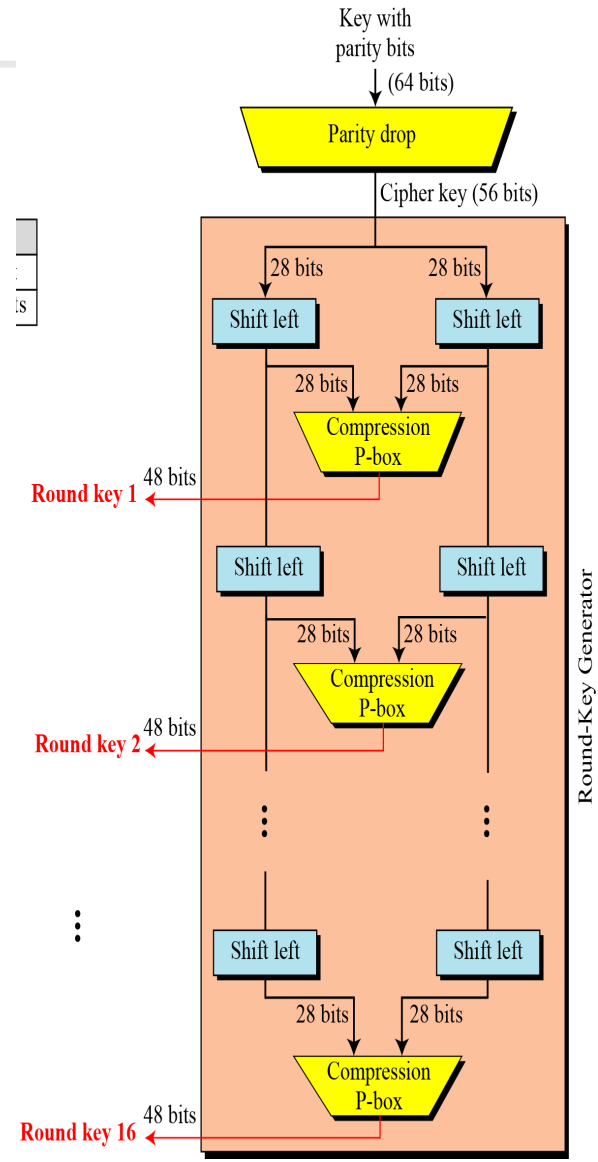
6.2.3 Continued

Shift Left

- *After permutation , Key is divided into two 28 bit parts*
- *Each part is shifted left one or two bits*

- *In round 1,2,9 and 16 shifting is One bit*
- *In Other rounds, its Two bits*
- *Two parts are then combined to form a 56 bit part*

- *No of bit shifts is shown in table below:*



Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bit shifts	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

6.2.3 Continued

Compression P Box

- *Changes 58 bits to 48 bits which are used as a key for a round*

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

Table 6.14 *Key-compression table*

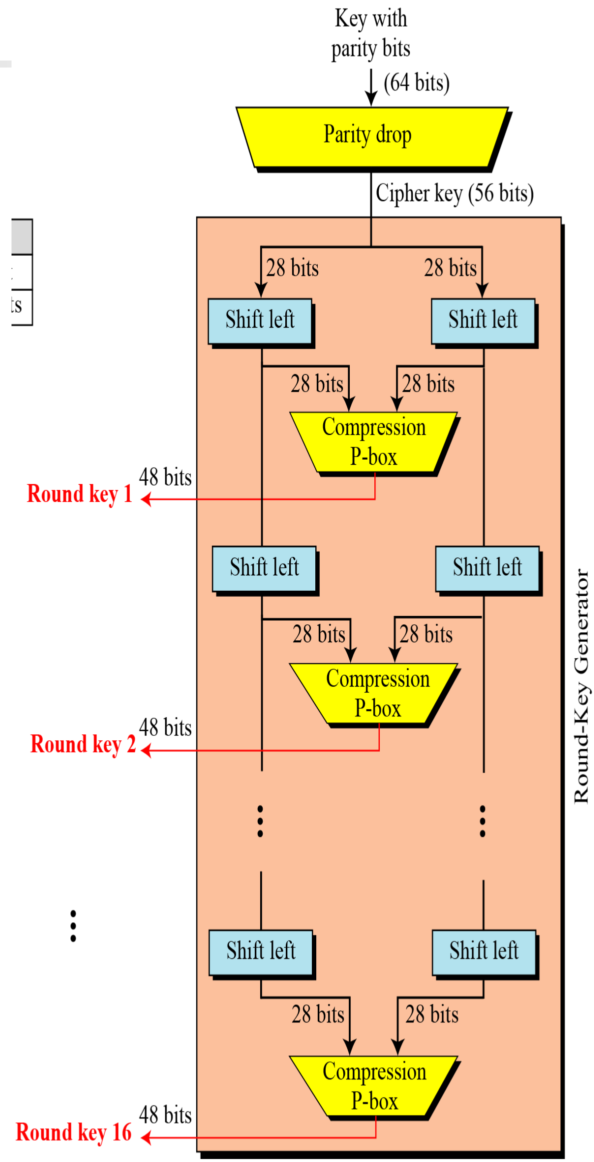
6.2.3 Continued

Algorithm 6.2 *Algorithm for round-key generation*

```
Key_Generator (keyWithParities[64], RoundKeys[16, 48], ShiftTable[16])
{
     permute (64, 56, keyWithParities, cipherKey, ParityDropTable)
     split (56, 28, cipherKey, leftKey, rightKey)
     for (round = 1 to 16)
     {
          shiftLeft (leftKey, ShiftTable[round])
          shiftLeft (rightKey, ShiftTable[round])
          combine (28, 56, leftKey, rightKey, preRoundKey)
          permute (56, 48, preRoundKey, RoundKeys[round], KeyCompressionTable)
     }
}
```

Shifting

Rounds	Shift
1, 2, 9, 16	one bit
Others	two bits



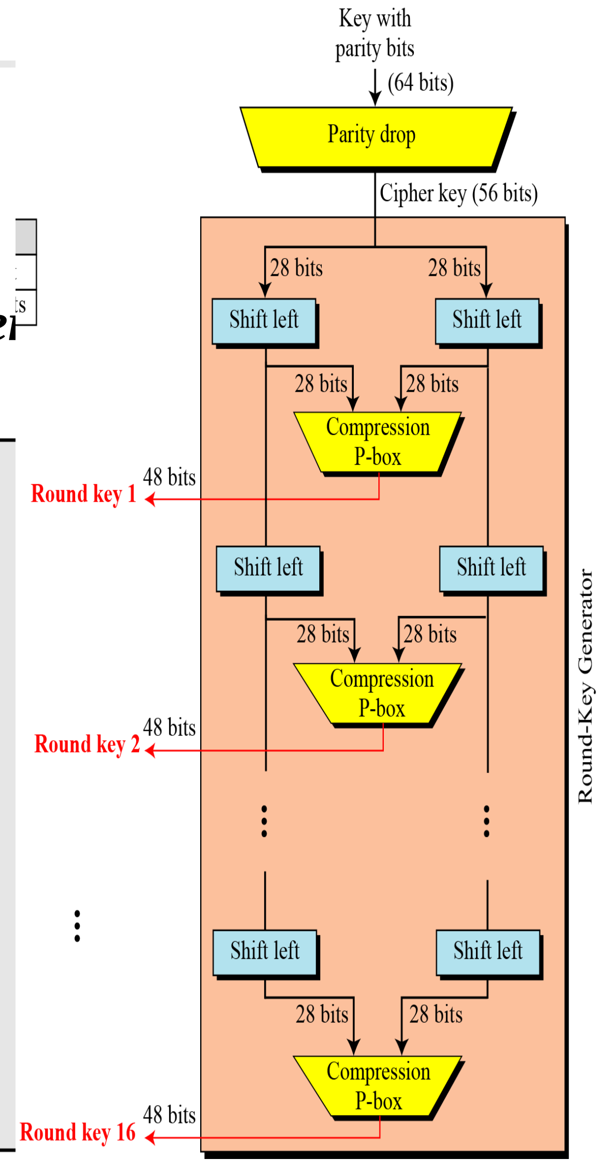
6.2.3 Continued

Algorithm 6.2 *Algorithm for round-key gener*

```
shiftLeft (block[28], numOfShifts)
{
    for (i = 1 to numOfShifts)
    {
        T ← block[1]
        for (j = 2 to 28)
        {
            block [j−1] ← block [j]
        }
        block[28] ← T
    }
}
```

Shifting

Rounds	Shift
1, 2, 9, 16	one bit
Others	two bits

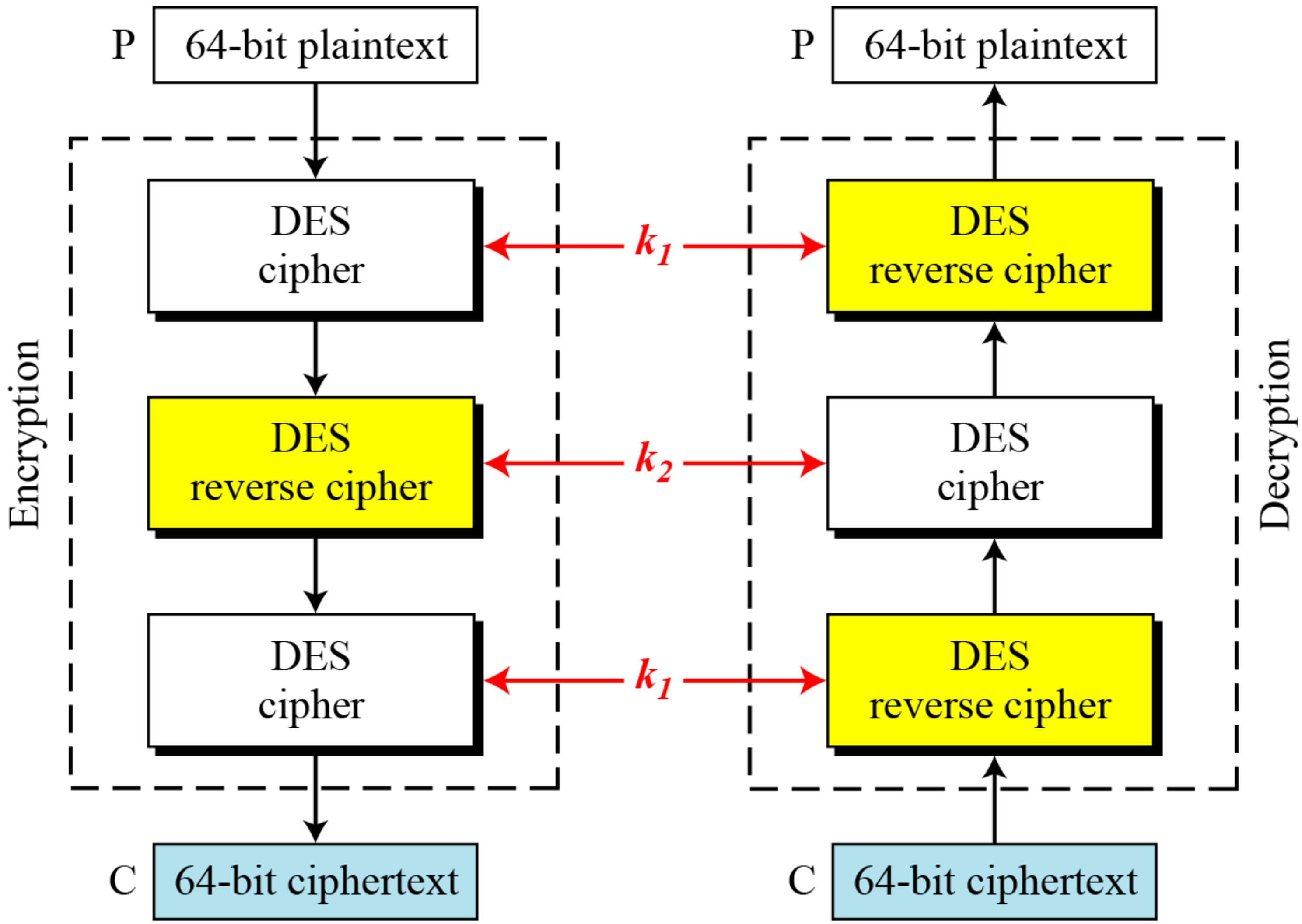


6-4 Multiple DES

The major criticism of DES regards its key length. Fortunately DES is not a group. This means that we can use double or triple DES to increase the key size.

6.4.2 Triple DES

Figure 6.16 ***Triple DES with two keys***



6.4.2 Continuous

Triple DES with Three Keys

The possibility of known-plaintext attacks on triple DES with two keys has enticed some applications to use triple DES with three keys. Triple DES with three keys is used by many applications such as PGP