

# INSTITUTE OF INFORMATION TECHNOLOGY



**Jahangirnagar University**

জাহাঙ্গীরনগর বিশ্ববিদ্যালয়

## **IT-4259: Computer Network Security**

for

**4th Year 2nd Semester of B.Sc (Honors) in IT (5th Batch)**

### **Lecture: 09**

**Modern Block Cipher: Data Encryption Standard (DES)**

**Prepared by:**

**K M Akkas Ali**

[akkas\\_khan@yahoo.com](mailto:akkas_khan@yahoo.com), [akkas@juniv.edu](mailto:akkas@juniv.edu)

**Associate Professor**

**Institute of Information Technology (IIT)**

**Jahangirnagar University, Dhaka-1342**

## Objectives of this Lecture:

- ❖ To introduce modern block ciphers and their characteristics.
- ❖ To introduce the components of modern block ciphers.
- ❖ To discuss product ciphers and discuss between Feistel and non-Feistel ciphers.
- ❖ To review a short history of DES.
- ❖ To define the basic structure of DES.
- ❖ To describe the details of building elements of DES.
- ❖ To describe the round keys generation process.
- ❖ To analyze DES.

# Modern Symmetric-key Ciphers:

## Character-oriented Vs. Bit-oriented Ciphers:

- The traditional symmetric-key ciphers are character-oriented ciphers.
- Now-a-days, the information to be encrypted is not just text; it can also consist of numbers, graphics, audio, and video data. It is convenient to convert these types of data into a stream of bits, to encrypt the stream, and then to send the encrypted stream.
- So, we need bit-oriented ciphers.
  - ❖ When data is treated as the collection of bits, it becomes larger. Mixing a larger number of symbols increases security.

## Kinds of Modern Symmetric-key Ciphers:

Modern symmetric-key ciphers can be divided into two broad categories:

- Stream ciphers
- Block ciphers

# Stream Ciphers:

- Stream cipher encrypts a single character or bit of plaintext at a time. It also decrypts a single character or bit of ciphertext at a time.
- Call the plaintext stream  $P$ , the ciphertext stream  $C$ , and the key stream  $K$ .

$$P = P_1P_2P_3, \dots$$

$$C = C_1C_2C_3, \dots$$

$$K = (k_1, k_2, k_3, \dots)$$

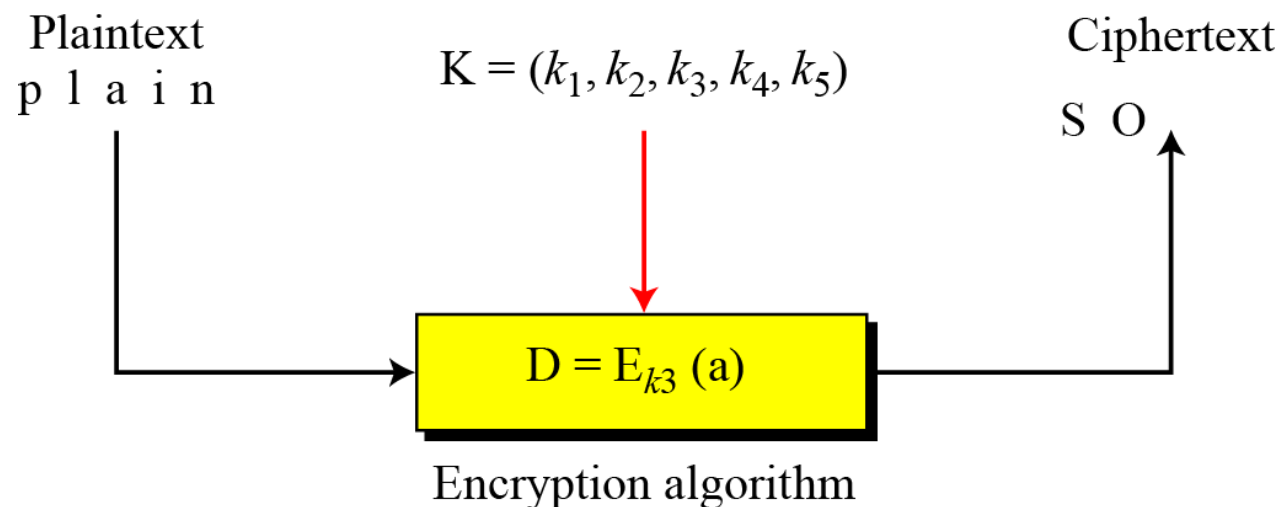
$$C_1 = E_{k_1}(P_1)$$

$$C_2 = E_{k_2}(P_2)$$

$$C_3 = E_{k_3}(P_3) \dots$$

- Figure below shows the idea behind stream cipher.

- ❑ Characters in the plaintext are fed into encryption algorithm, one at a time.
- ❑ The ciphertext characters are also created one at a time.



**Figure: Idea behind stream cipher**

# Stream Ciphers (continued...)

## Example:

- Given plaintext: 10011011110100001  
Let the keystream be a stream of 1s and 0s.
- If we use an exclusive or (XOR) with the keystream and plaintext, we get ciphertext.
- This keystream is called periodic, since the sequence '10' repeats over and over.

Plaintext	:	10011011110100001	
Keystream	:	10101010101010101	
Ciphertext	:	00110001011110100	(by XORing each plaintext bit with corresponding keystream bit)

- To decrypt this ciphertext, all we need to do is again XOR the ciphertext with the keystream:

Ciphertext	:	00110001011110100
Keystream	:	10101010101010101
Plaintext (XOR)	:	10011011110100001

# Block Ciphers:

- A symmetric-key modern block cipher encrypts an  $n$ -bit block of plaintext or decrypts an  $n$ -bit block of ciphertext together.
- The decryption algorithm must be the inverse of the encryption algorithm.
- Both operations must use the same secret key of  $k$ -bit length so that Bob can retrieve the message sent by Alice.
  - ❖ If the message has the fewer than  $n$  bits, padding must be added to make it an  $n$ -bit block.
  - ❖ If the message has more than  $n$  bits, it should be divided into  $n$ -bit blocks and the appropriate padding must be added to the last block if necessary.
  - ❖ The common values of  $n$  are 64, 128, 256, or 512 bits.

## Example

Plaintext : The only thing we have to fear is fear itself

Modified plaintext : Theonlythingwehavetofearisfearitself

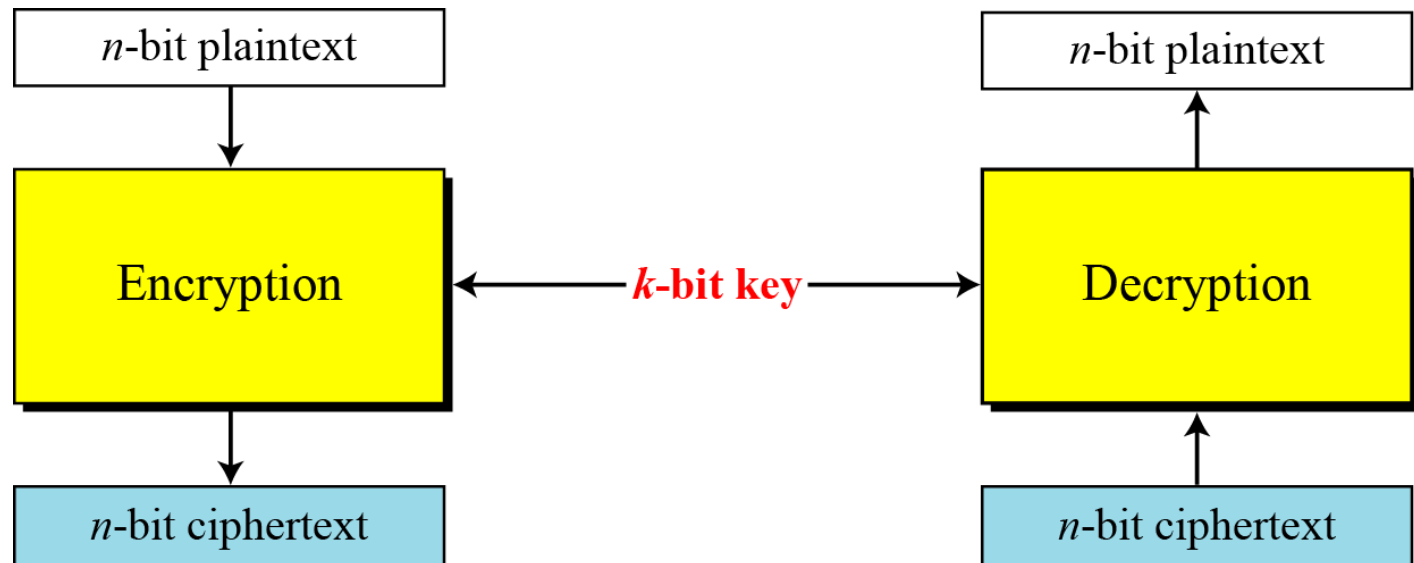
Plaintext blocks : Theonlyt hingweha vetofear isfearit selfXend (break the plaintext into 8-character block)

Ciphertext blocks : tylnoehT ahewgnih raefotev tiraefsi dneXfles (just reverse each plaintext block)

Ciphertext : tylnoehTahewgnihraefotevtiraefsidneXfles

# Block Ciphers (continued...):

- Block ciphers are currently better analyzed, and seem to have a broader range of applications. Many current ciphers are block ciphers. Example: **Data Encryption Standard (DES)**, **Advanced Encryption Standard (AES)**, and **International Data Encryption Algorithm (IDEA)** all are block algorithms.
- Figure below shows the general idea of encryption and decryption in a modern block cipher.



**Figure:** A modern block cipher

# Block Cipher: Substitution Vs. Transposition

- A modern block cipher can be designed to act as a substitution cipher or a transposition cipher.
  - ❑ In substitution block cipher, a 1-bit or a 0-bit in the plaintext can be replaced by either a 0 or a 1. This means that the plaintext and the ciphertext can have a different number of 1's. A 64-bit plaintext block of 12 0's and 52 1's can be encrypted to a ciphertext of 34 0's and 30 1's.
  - ❑ If the cipher is designed as a transposition cipher, the bits are only reordered (transposed); there is the same number of 1's (or 0's) in the plaintext and in the ciphertext.
- In either case, the number of n-bit possible plaintexts or ciphertexts is  $2^n$ .
- Modern block ciphers are designed as substitution ciphers **because** the inherent characteristics of transposition cipher (preserving the number of 1's or 0's) make the cipher vulnerable to exhaustive-search attacks, as the next example shows.



# Block Cipher: Substitution Vs. Transposition

## Example:

- Suppose that we have a block cipher where  $n = 64$ . If there are 10 1's in the ciphertext, how many trial-and-error tests does Eve need to do to recover the plaintext from the intercepted ciphertext in each of the following cases?
  - a. The cipher is designed as a substitution cipher.
  - b. The cipher is designed as a transposition cipher.

## Solution:

- a. In the first case (substitution), Eve has no idea how many 1's are in the plaintext. She needs to try all possible  $2^{64}$  64-bit blocks (18,446,744,073,709,551,616) to find one that makes sense. If Eve could try 1 billion blocks per second, it would still take hundreds of years, on average, before she could be successful.
  - b. In the second case (transposition), Eve knows that there are exactly 10 1's in the plaintext, because transposition does not change the number of 1's (or 0's) in the ciphertext. Eve can launch an exhaustive-search attack using only those 64-bit blocks that have exactly 10 1's. There are only  $(64!)/[(10!)(54!)] = 151,473,214,816$  words out of  $2^{64}$  64-bit words that have exactly 10 1's. Eve can test all of them in less than 3 minutes if she can do 1 billion tests per second.
- **Therefore, to be resistant to exhaustive-search attack, a modern block cipher needs to be designed as a substitution cipher.**

# Components of a Modern Block Ciphers:

- Modern block ciphers normally are keyed substitution ciphers in which the key allows only partial mappings from the possible inputs to the possible outputs.
- However, modern block ciphers normally are not designed as a single unit.
- To provide the required properties of a modern block cipher, such as **diffusion** and **confusion**, a modern block cipher is made of a combination of several units:
  - ❑ Transposition units (called P-boxes)
  - ❑ Substitution units (called S-boxes)
  - ❑ Some other units

# Components of a Modern Block Ciphers (continued...):

## P-Boxes:

- A P-box (permutation box) is a component in a modern block cipher that transposes bits.

## Types of P-Boxes:

- Three types of P-boxes are used in modern block ciphers:
  - (1) Straight P-Boxes
  - (2) Expansion P-Boxes
  - (3) Compression P-Boxes

# Components of a Modern Block Ciphers (continued...):

## Straight P-Boxes:

- A straight P-Box is a permutation which has ***n*** inputs and ***n*** outputs.
- There are ***n!*** possible mappings.
- Figure below shows a 5 x 5 straight P-box.

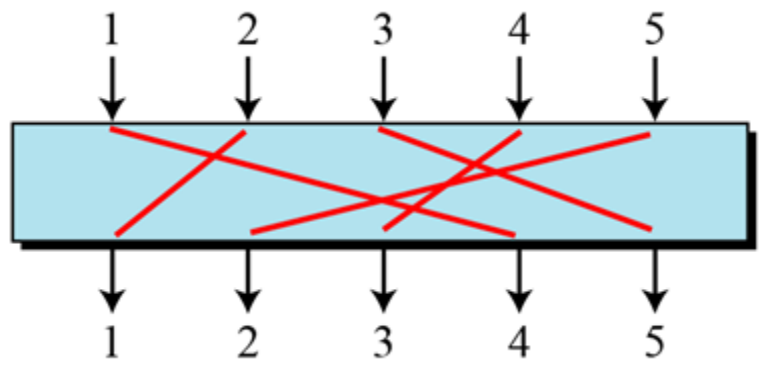


Figure: A 5x5 straight P-box

## Example of all mappings for Straight P-Boxes:

- Figure below shows a 3x3 straight P-box with all 6 (3!) possible mappings.

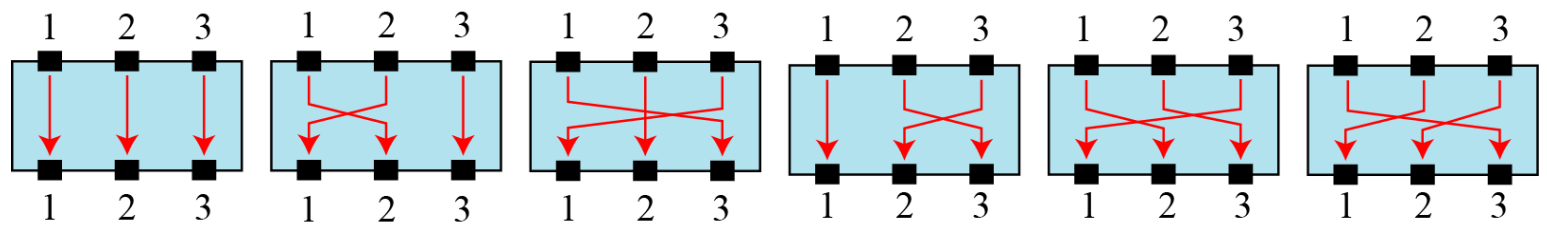


Figure: The possible mappings of a 3 × 3 Straight P-box

# Components of a Modern Block Ciphers (continued...):

## Permutation Table of a P-Box:

- A P-box can use a key to define one of the possible  $n!$  mappings.
- A P-box can be implemented either in hardware or software.
- If it is implemented in software, a permutation table is used to show the rules for transposing bits.
- The entries in the permutation table are the inputs and the positions of the entries are the outputs.
- Table below shows an example of a permutation table for a 64 x 64 straight P-box when n is 64.

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

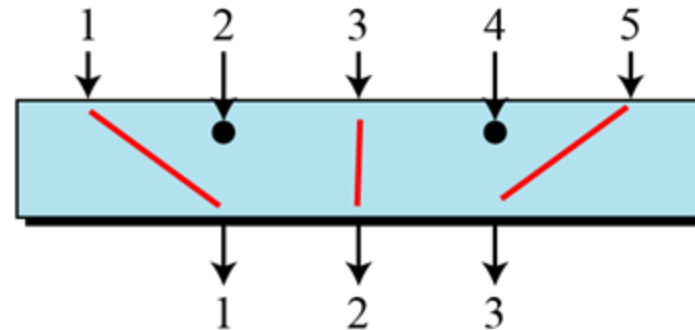
**Table:** Example of a permutation table for a 64 x 64 straight P-box

- ❖ The table has 64 entries, corresponding to the 64 inputs.
- ❖ The position (index) of the entry corresponds to the output.
- ❖ The first entry of the table is 58 which has the first index. Therefore, the first output comes from the 58<sup>th</sup> input.
- ❖ Similarly, the last entry of the table contains 7 which has the 64<sup>th</sup> index. Therefore, the last output comes from the 7<sup>th</sup> input.

# Components of a Modern Block Ciphers (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.
- Figure below shows a 5 x 3 compression P-box.



**Figure:** A 5x3 Compression P-box

- The compression P-boxes used in modern block ciphers are keyless normally, where a permutation table shows the rules for transposing bits.
- Compression P-boxes are used when we need to permute bits and the same time decrease the number of bits for the next stage of encryption/decryption.

# Components of a Modern Block Ciphers (continued...):

## Permutation Table of a Compression P-Box:

- A permutation table for a compression P-box has ***m*** entries, but the content of each entry is from ***1*** to ***n*** with some missing values (those inputs that are blocked).
- Table below shows an example of a permutation table for a 32 x 24 compression P-box when ***n*** is 32 and ***m*** is 24.
- Note that the inputs 7, 8, 9, 15, 16, 23, 24, and 25 are blocked.

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

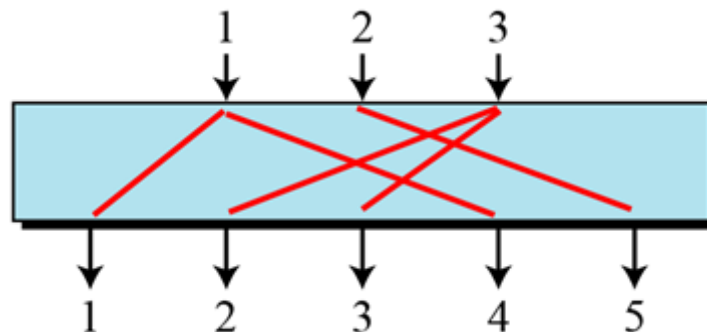
**Table:** Example of a permutation table for a 32 x 24 compression P-box

- ❖ The above permutation table has 24 entries with each value ranges from 1 to 32 with missing inputs 7, 8, 9, 15, 16, 23, 24, and 25 that are blocked.
- ❖ The position (index) of the entry corresponds to the output.
- ❖ The first entry of the table is 01 which has the first index. Therefore, the first output comes from the 1<sup>st</sup> input.
- ❖ Similarly, the fourth entry of the table contains 21 which has the 4<sup>th</sup> index. Therefore, the 4<sup>th</sup> output comes from the 21<sup>st</sup> input.

# Components of a Modern Block Ciphers (continued...):

## Expansion P-Boxes:

- A 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.
- Figure below shows a 3 x 5 expansion P-box.



**Figure:** A 3 x 5 Expansion P-box

- The expansion P-boxes used in modern block ciphers normally are keyless, where a permutation table shows the rules for transposing bits.
- Expansion P-boxes are used when we need to permute bits and the same time increase the number of bits for the next stage of encryption/decryption.



# Components of a Modern Block Ciphers (continued...):

## Permutation Table of an Expansion P-Box:

- A permutation table for an expansion P-box has  $m$  entries, but the content of each entry is from  $1$  to  $n$  with  $m-n$  of the entries are repeated (those inputs that are mapped to more than one output).
- Table below shows an example of a permutation table for a  $12 \times 16$  expansion P-box when  $n$  is 12 and  $m$  is 16.
- Note that each of the inputs 1, 3, 9, and 12 is mapped to two outputs.

01	09	10	11	12	01	02	03	03	04	05	06	07	08	09	12
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

**Table:** Example of a permutation table for a  $12 \times 16$  expansion P-box

- ❖ The above permutation table has 16 entries with each value ranges from 1 to 12 with each of the inputs 1, 3, 9, and 12 is mapped to two outputs.
- ❖ The position (index) of the entry corresponds to the output.
- ❖ The first entry of the table is 01 which has the first index. Therefore, the first output comes from the 1<sup>st</sup> input.
- ❖ Similarly, the fourth entry of the table contains 11 which has the 4<sup>th</sup> index. Therefore, the 4<sup>th</sup> output comes from the 11<sup>th</sup> input.

# Components of a Modern Block Ciphers (continued...):

## Invertibility of a P-Boxes:

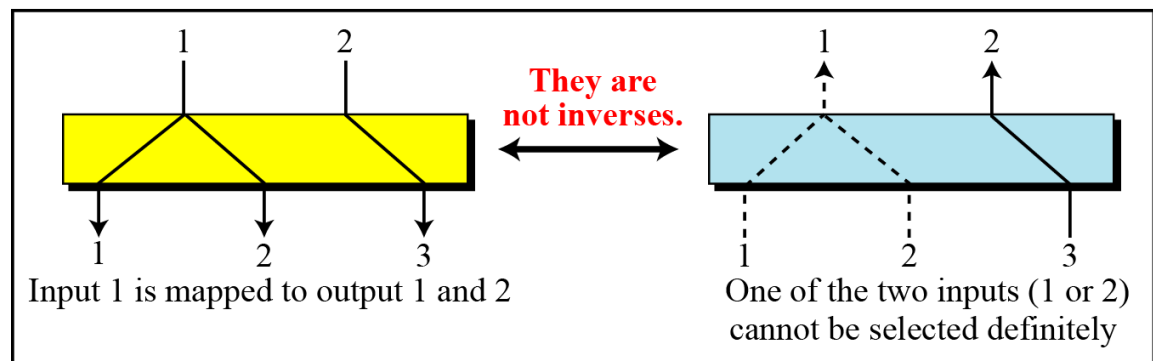
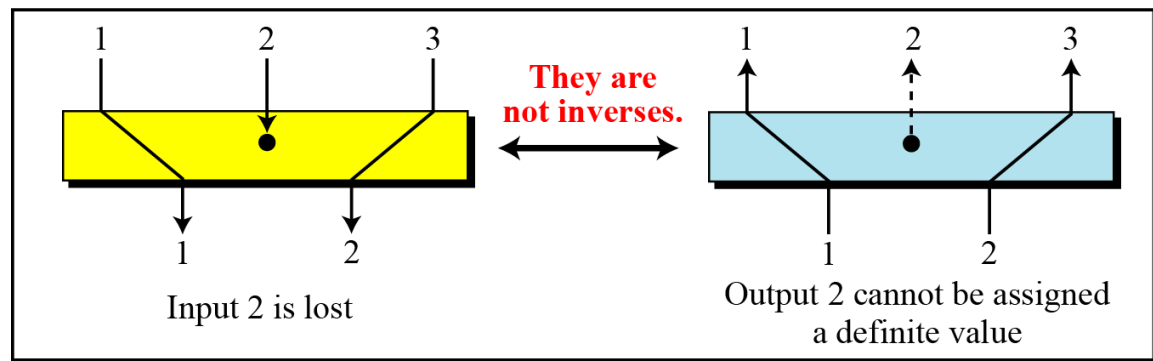
- If a P-box is used in the encryption cipher and its inverse is used in the decryption cipher, then the P-box is called invertible P-box.
- The permutation tables of an invertible P-box need to be the inverses of each other.
- A straight P-box is invertible, but compression and expansion P-boxes are not.

❑ In a compression P-box, an input can be dropped during encryption; the decryption algorithm does not have a clue how to replace the dropped bit (a choice between a 0 or a 1-bit).

❑ In an 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.

❑ Figure below demonstrates both cases.

Compression P-box



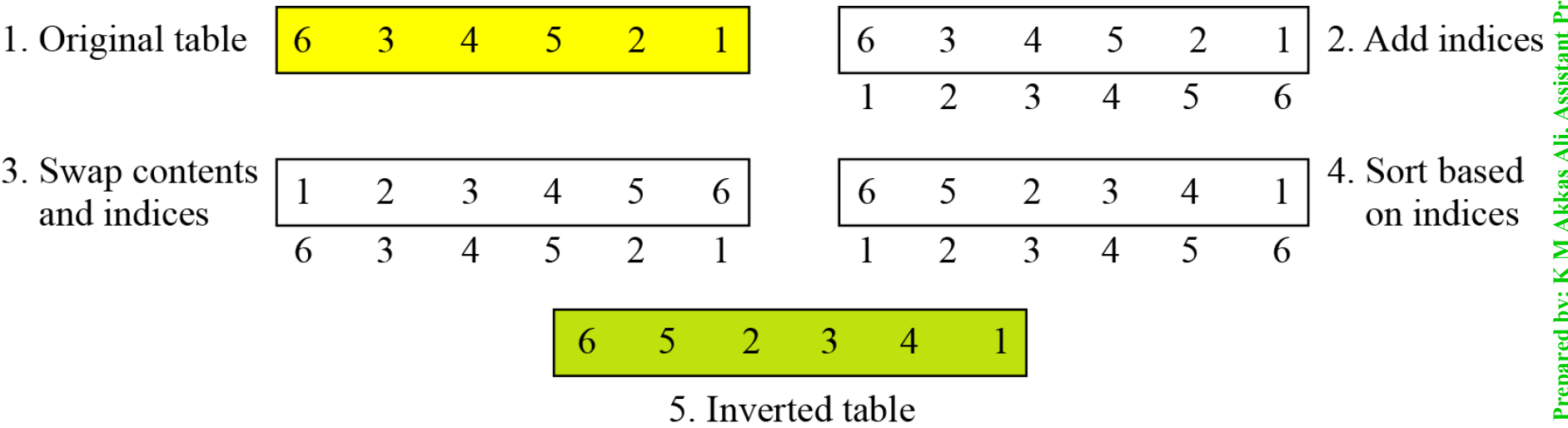
Expansion P-box

**Figure:** Compression and Expansion P-boxes as non-invertible components

# Components of a Modern Block Ciphers (continued...):

## Inverting a Permutation Table of a P-Box:

- How can the inverse of a permutation table be created if the initial or original permutation table is given, or vice versa?
- The process can be done manually in a few steps.
- Figure below shows how to invert a permutation table represented as a one-dimensional table.



**Figure:** Inverting a permutation table

# Components of a Modern Block Ciphers (continued...):

## S-Boxes:

- An S-box (substitution box) can be thought of as a miniature substitution cipher.
- However, an S-box can have a different number of inputs and outputs. In other words, the input to an S-box could be an  $n$ -bit word, but the output can be an  $m$ -bit word, where  $m$  and  $n$  are not necessarily the same.
- Although an S-box can be keyed or keyless, modern block ciphers normally use keyless S-boxes, where the mapping from the inputs to the outputs is predetermined.

# Components of a Modern Block Ciphers (continued...):

## Input-Output Relationship for a 3x2 S-Box by Table:

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

**Table:** Input-Output relationship for a 3 x 2 S-box

- Based on the above S-box table, an input of **010** yields the output 01. An input of **101** yields the output of 00.

# Components of a Modern Block Ciphers (continued...):

## Invertibility of an S-Boxes:


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

**Example:** Figure below shows an example of an invertible S-box.

- ❑ One of the tables is used in the encryption algorithm; the other table is used in the decryption algorithm.
- ❑ In each table, the leftmost bit of the input defines the row; the next two bits define the column. The output is the value where the input row and column meet.


- ❖ **For example**, if the input to the left box is 001, the output is 101.
- ❖ Similarly, if the input to the right box is 101, the output is 001.
- ❖ The example shows that the two tables are inverses of each other.

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

# Components of a Modern Block Ciphers (continued...):

## Exclusive-Or:

- An important component in most block ciphers is the exclusive-or operation.
- Addition and subtraction operations are performed by a single operation called exclusive-or (XOR).

## Properties of XOR Operation:

The following five properties of the exclusive-or operation makes this operation a very interesting component for use in a block cipher:

### 1. Closure:

This property guarantees that the result of exclusive-oring two n-bit words is another n-bit word.

### 2. Associativity:

This property allows us to use more than one exclusive-or operator in any order:

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

### 3. Commutativity:

This property allows us to swap the inputs without affecting the output:  $x \oplus y \leftrightarrow y \oplus x$

### 4. Existence of identity:

The identity element for the exclusive-or operation is an n-bit word that consists of all 0's. This property implies that exclusive-oring of a word with the identity element does not change that word:  $x \oplus (000...0) = x$ . This property is used in the Feistel cipher.

### 5. Existence of inverse:

This property implies that exclusive-oring of a word with itself yields the identity element:  $x \oplus x = (000...0)$ . This property is used in the Feistel cipher.

# Components of a Modern Block Ciphers (continued...):

## Complement:

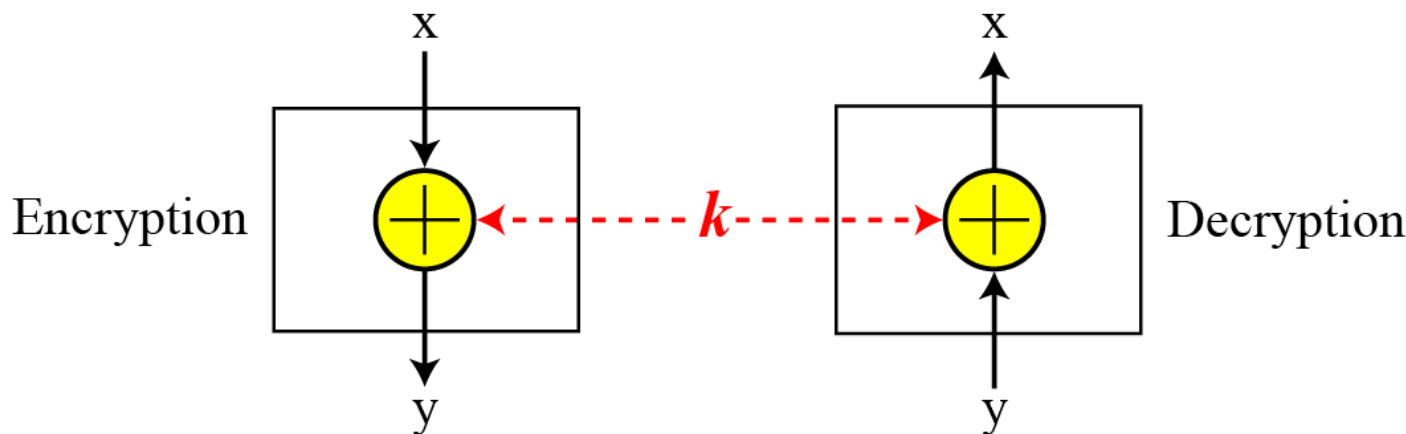
- The complement operation is a unary operation (one input and one output) that flips each bit in a word. A 0-bit is changed to a 1-bit; a 1-bit is changed to a 0-bit.
- We are interested in the complement operation in relation to the exclusive-or operation.
- If  $x^{\sim}$  is the complement of  $x$ , then the following two relations hold:
  - $x \oplus x^{\sim} = (111...1)$
  - $x \oplus (111...1) = x^{\sim}$
- We will use the above relations while discussing the security of some ciphers.



# Components of a Modern Block Ciphers (continued...):

## Inverse:

- The inverse of a component in a cipher makes sense if the component represents a unary operation (one input and one output).
- For example, a keyless P-box or a keyless S-box can be made invertible because they have one input and one output.
- An exclusive OR operation is a binary operation. The inverse of an exclusive-or operation can make sense only if one of the inputs is fixed (is the same in encryption and decryption). For example, if one of the inputs is the key, which normally is the same in encryption and decryption, then an exclusive-or operation is self-invertible, which is shown in the figure below.
- We will use this property when we discuss the structure of block ciphers.

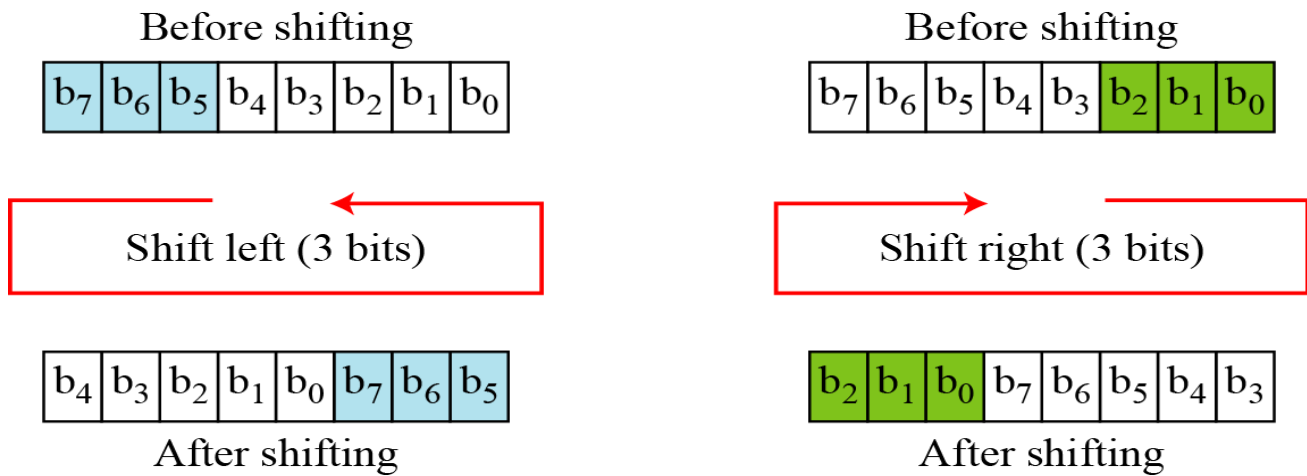


**Figure:** Invertibility of the exclusive-or operation

# Components of a Modern Block Ciphers (continued...):

## Circular Shift:

- Another component found in some modern block ciphers is the circular shift operation which mixes the bits in a word and helps hide the patterns in the original word.
- Shifting can be to the left or to the right:
  - ❑ The **circular left-shift** operation shifts each bit in an  $n$ -bit word  $k$  positions to the left; the leftmost  $k$  bits are removed from the left and become the rightmost bits.
  - ❑ The **circular right-shift** operation shifts each bit in an  $n$ -bit word  $k$  positions to the right; the rightmost  $k$  bits are removed from the right and become the leftmost bits.
- Figure below shows both left and right operations in the case where  $n = 8$  and  $k = 3$ .



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

# Components of a Modern Block Ciphers (continued...):

## Properties of Circular Shift Operation:

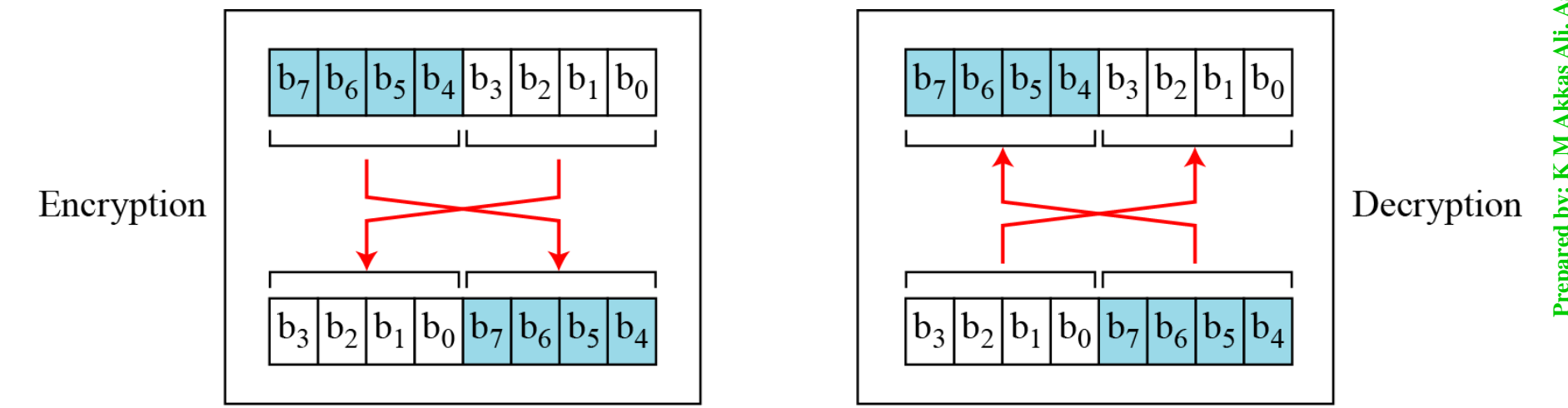
The circular shift operation has the following properties:

- ❖ A circular left-shift operation is the inverse of the circular right-shift operation. If one is used in the encryption cipher, the other can be used in the decryption cipher.
- ❖ In circular shift operation, the shifting is modulo  $n$ . In other words, if  $k = 0$  or  $k = n$ , there is no shifting. If  $k$  is larger than  $n$ , then the input is shifted  $k \bmod n$  bits.
- ❖ In circular shift operation, shifting a word more than once is the same as shifting it only once.

# Components of a Modern Block Ciphers (continued...):

## Swap:

- The swap operation is a special case of the circular shift operation where  $k = n/2$  where  $n$  is the block size in bits and  $k$  is the position to be shifted.
- Swap operation is valid only if  $n$  is an even number.
- Because left-shifting  $n/2$  bits is the same as right-shifting  $n/2$ , this component is self-invertible.
- A swap operation in the encryption cipher can be totally canceled by a swap operation in the decryption cipher.
- Figure below shows the swapping operation for an 8-bit word.

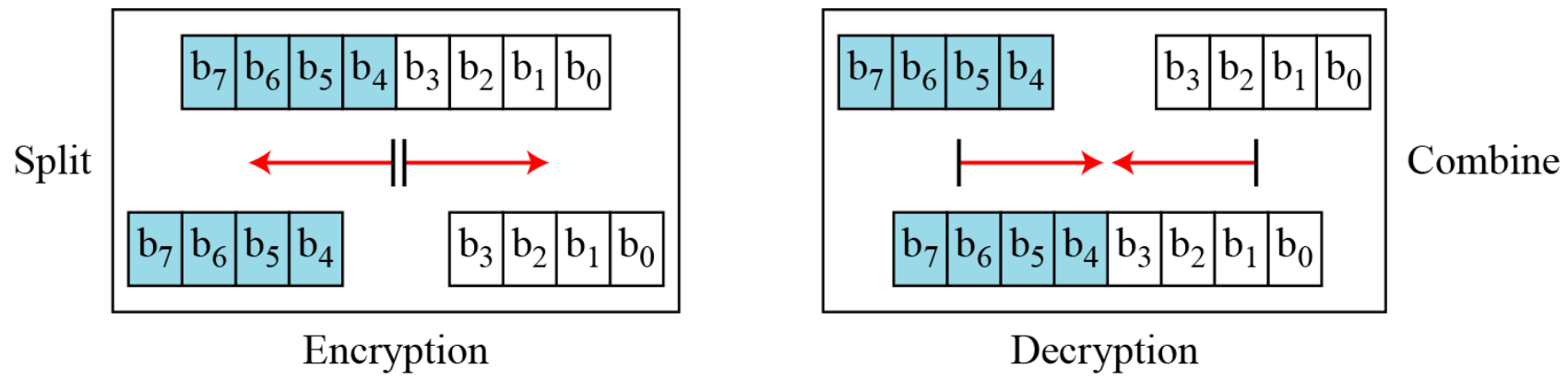


**Figure :** Swap operation on an 8-bit word

# Components of a Modern Block Ciphers (continued...):

## Split and Combine:

- The split operation normally splits an n-bit word in the middle, creating two equal-length words.
- The combine operation normally concatenates two equal-length words to create an n-bit word.
- These two operations are inverses of each other and can be used as a pair to cancel each other out. If one is used in the encryption cipher, the other is used in the decryption cipher.
- Figure below shows the two operations in the case where  $n = 8$ .

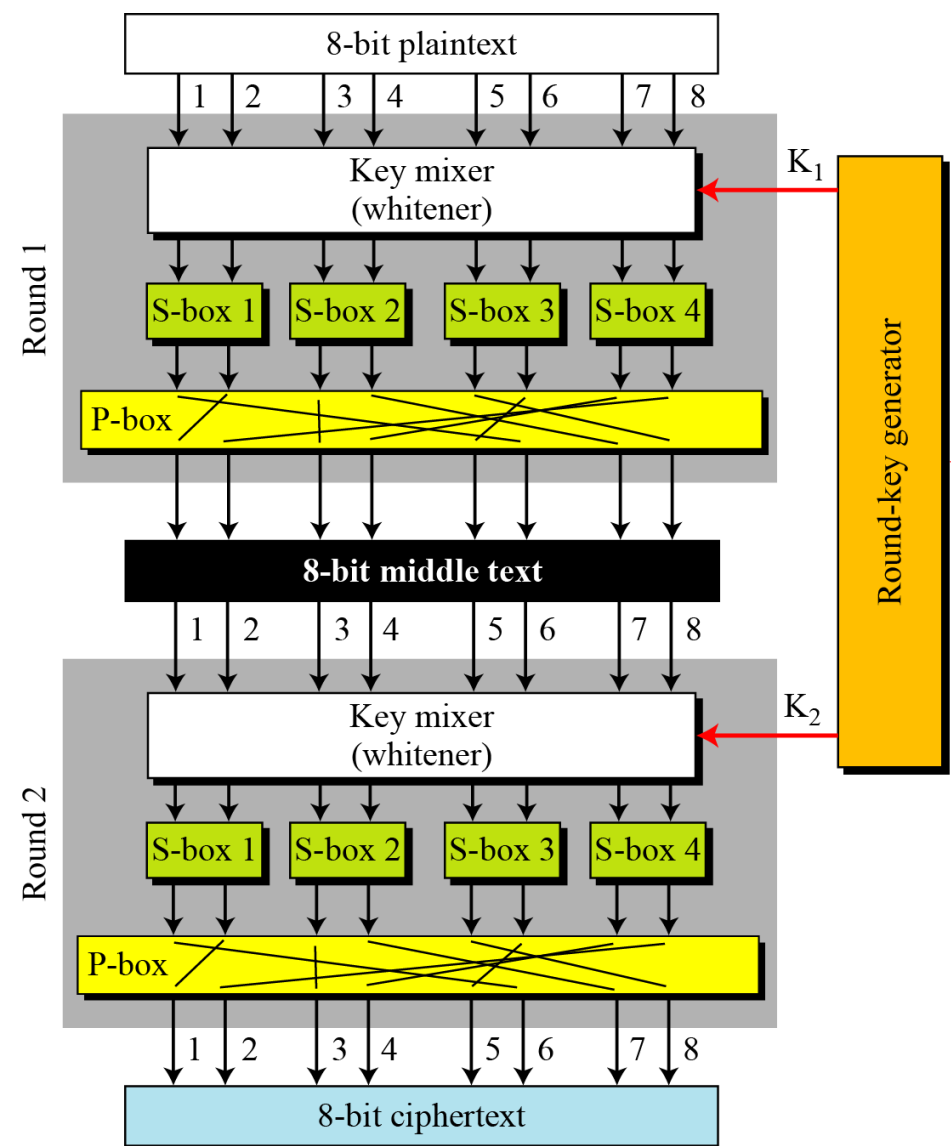


**Figure :** Split and combine operations on an 8-bit word

# Components of a Modern Block Ciphers (continued...):

## Product Cipher:

- A product cipher is a complex cipher combining substitution, permutation, and other components.
- **Shannon** introduced the concept of a product cipher.
- Modern block ciphers are all product ciphers.
- Figure below shows a simple product cipher with two rounds.
- Three transformations happen at each round:
  - a) The 8-bit text is mixed with the key to whiten the text (hide the bits using the key). This is normally done by exclusive-oring the 8-bit word with the 8-bit key.
  - b) The outputs of the whitener are organized into four 2-bit groups and are fed into four S-boxes. The values of bits are changed based on the structure of the S-boxes in this transformation.
  - c) The outputs of S-boxes are passed through a P-box to permute the bits so that in the next round each box receives different inputs.



Prepared by: K M Akkas Ali, Assistant Professor, IIT, JU

**Figure :** A simple product cipher with two rounds

# Components of a Modern Block Ciphers (continued...):

## Kinds of Product Ciphers:

- Modern block ciphers are all product ciphers, but they are divided into two classes:
  - ❑ **Feistel ciphers**
  - ❑ **Non-Feistel ciphers**
- ❑ **Feistel ciphers:**
  - ❖ In 1973, **Feistel** designed a very intelligent and interesting cipher that has been used for decades. Several block ciphers are based on the Feistel structure.
    - ❖ This type of ciphers use both invertible and noninvertible components.
    - ❖ A Feistel cipher can have three types of components: self-invertible, invertible, and noninvertible.
    - ❖ A Feistel cipher combines all noninvertible elements in a unit (called mixer) and uses the same unit in the encryption and decryption algorithms.
    - ❖ The block cipher DES, IDEA, RC5 (Rivest's Cipher) are good examples of a Feistel cipher. But Feistel design is not used in AES.

# Components of a Modern Block Ciphers (continued...):

## ❑ **Non-Feistel ciphers:**

- This type of ciphers use only invertible components.
- A component in the encryption cipher has the corresponding component in the decryption cipher.
- For example, S-boxes need to have an equal number of inputs and outputs to be compatible. No compression or expansion P-boxes are allowed, because they are not invertible.
  - ❖ In a non-Feistel cipher, there is no need to divide the plaintext into two halves as we saw in the Feistel ciphers.
  - ❖ The block cipher AES is a good example of a non-Feistel cipher.



# Confusion and Diffusion:

- The terms diffusion and confusion were introduced by Claude Shannon to capture the two basic building blocks for product cipher.
- Every block cipher involves a transformation of a block of plaintext into a block of ciphertext, where the transformation depends on the key.
- Hence, the block cipher needs to completely obscure statistical properties of original message.
- Shannon suggested combining S & P elements to obtain diffusion and confusion.

## ❑ Diffusion:

- ❖ The idea of diffusion is to hide the relationship between the ciphertext and the plaintext. That is, the statistical relationship between the plaintext and ciphertext is made as complex as possible in order to thwart attempts to deduce the key. This will frustrate the adversary who uses ciphertext statistics to find the plaintext.
- ❖ Diffusion implies that each symbol (bit) in the ciphertext is dependent on some or all symbols in the plaintext. In other words, if a single symbol in the plaintext is changed, several or all symbols in the ciphertext will also be changed.

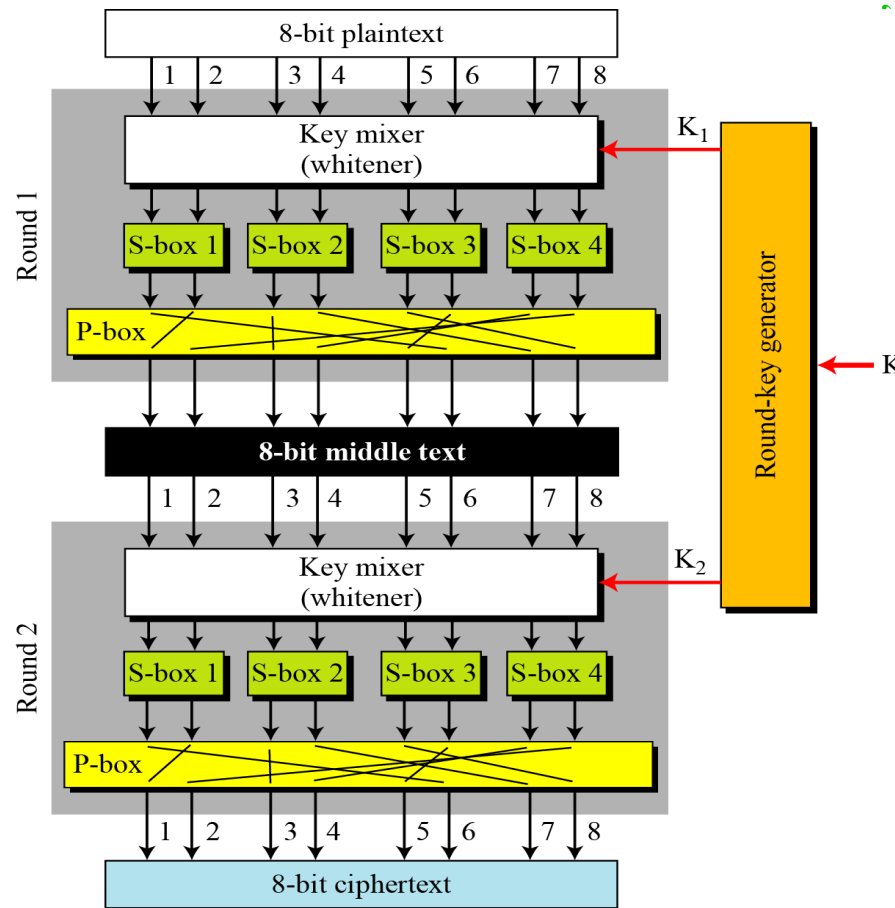
## ❑ Confusion:

- ❖ The idea of confusion is to hide the relationship between the ciphertext and the key. That is, the relationship between the ciphertext and the key is made as complex as possible in order to thwart attempts to discover the key. This will frustrate the adversary who tries to use the ciphertext to find the key.
- ❖ In other words, if a single bit in the key is changed, most or all bits in the ciphertext will also be changed.

# Components of a Modern Block Ciphers (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. Each iteration is referred to as a round.
- The block cipher uses a key schedule or key generator that creates different keys for each round from the cipher key.
- In an N-round cipher, the plaintext is encrypted N times to create the ciphertext; the ciphertext is decrypted N times to create the plaintext.
- We refer to the text created at the intermediate levels (between two rounds) as the **middle text**.
- Figure below shows a simple product cipher with two rounds. In practice, product ciphers have more than two rounds.



**Figure :** Two rounds in a product cipher IIT, JU

# Brief History of Data Encryption Standard (DES)

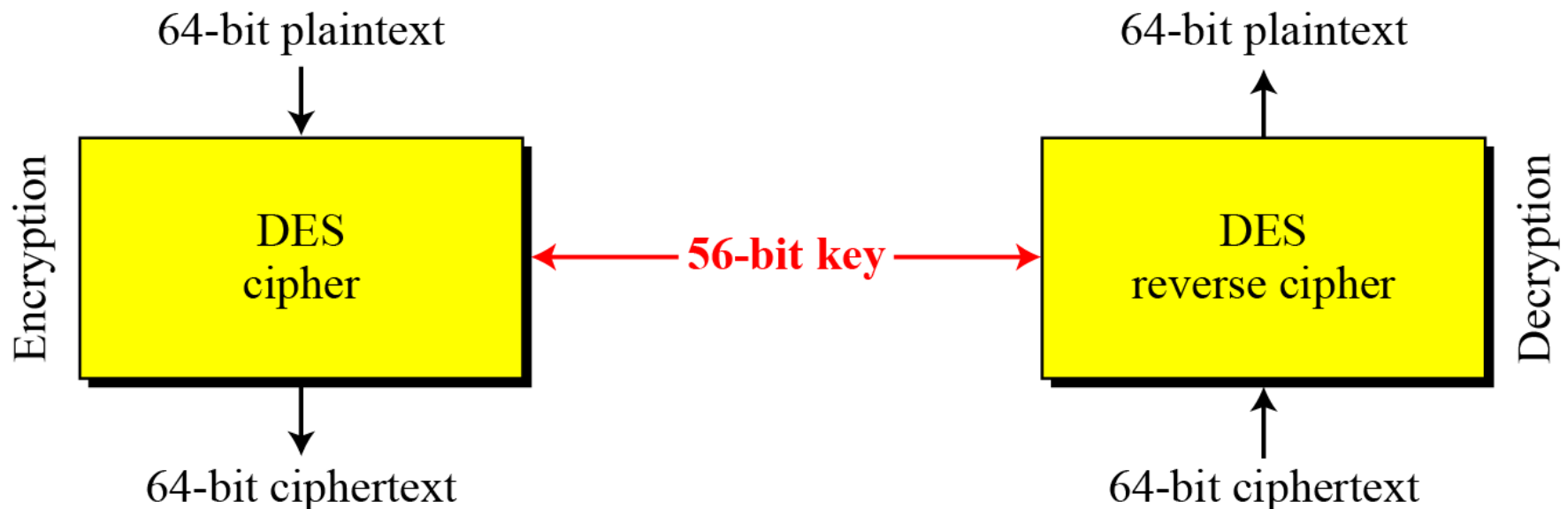
- The Data Encryption Standard (DES) is a **symmetric-key block cipher published by** the National Institute of Standards and Technology (NIST).
  - ❑ In 1973, NIST published a request for proposals for a national symmetric-key cryptosystem.
  - ❑ A proposal from IBM, a modification of a research project 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).
  - ❑ After the publication, IBM sought technical advice from the National Security Agency (NSA) for the modification of Lucifer.
- The modified version of LUCIFER was put forward as a proposal for the new national encryption standard requested by the National Bureau of Standards (NBS, now known as the National Institute of Standards and Technology, NIST). It was finally adopted in 1977 as the Data Encryption Standard -DES (FIPS PUB 46).
- Some of the changes made to LUCIFER have been the subject of much controversy even to the present day **for two reasons**:
  - ❑ First, the critics questioned the small key length (only 56 bits) which could make the cipher vulnerable to brute-force attack. Even though DES actually accepts a 64 bit key as input, the remaining eight bits are used for parity checking and have no effect on DES's security.
  - ❑ Second, critics were concerned about some hidden design behind the internal structure of DES. They were suspicious that some part of the structure (e.g. the S-boxes) may have some hidden trapdoor that would allow the NSA to decrypt the message without the need for the key.

# Overview of DES

- DES is a 64 bit block cipher **with key length 56 bits**.
- In DES, the plaintext input bit string is divided into 64-bit blocks and each block is encrypted using the same 56-bit key. The same key is used for decryption. Hence, DES is a symmetric block cipher.
- It was designed by IBM in 1976 for the National Bureau of Standards (NBS), with approval from the National Security Agency (NSA).
- It had been used as a standard method of encryption until 2000, but with increase in speed in computers, it is no more considered secure as a cryptanalyst can break the code by exhaustively searching for all the keys using a fast computer.
- However, a modification of DES, called triple DES (or 3 DES), is now used which is more secure and is difficult to break.
- From 2001, DES has been replaced by a new standard known as the Advanced Encryption Standard (AES).
- After 25 years of analysis, the only security problem with DES found is that its key length is too short.
- Although it's wide spread use came to an end in 2000, its design idea is still used in most block ciphers.

# Overview of Data Encryption Standard (DES)

- The block diagram of encryption and decryption process in DES is shown in the figure below.
- At the encryption site, DES takes a 64-bit plaintext and creates a 64-bit ciphertext.
- At the decryption site, it takes a 64-bit ciphertext and creates a 64-bit block of plaintext.
- The same 56-bit cipher key is used for both encryption and decryption.



**Figure:** Encryption and decryption with DES

# DES Algorithm/DES Structure/ Encryption of the DES:

- The actual DES encryption algorithm is quite complex.
- Plaintext is broken into blocks of length 64 bits. Each 64-bit block of plaintext is encrypted using a 56-bit key.
- A 56-bit key  $k$  is fed into a subkey generating algorithm to produce 16 round subkeys  $k_1, k_2, k_3, \dots, k_{16}$  of length 48 bits each.
  - ❑ At first, an initial permutation (IP) is performed on the 64-bit block of plaintext. (The initial permutation rearranges the bits of the plaintext to form the “permuted input” based on the IP table shown in Table-5).
  - ❑ After initial permutation, the 64-bit permuted block is divided into two 32-bit sub-blocks represented by  $L_0$  and  $R_0$  as the left and right sub-block respectively.
  - ❑ The encryption then proceeds through 16 rounds of identical operations using a different sub-key of length 48-bit in each round on the left and right halves of the block. (As shown in the figure, the inputs to each round consist of the  $L_i, R_i$  pair and a 48 bit subkey which is a shifted and contracted version of the original 56 bit key).
    - ❖ The 48-bit subkey  $k_i$  for round  $i$  (where  $i=1, 2, 3, 4, \dots, 16$ ) is generated from the original 56-bit key.
    - ❖ To do so, it uses a subkey function SK which is the permutation of 56 bits (i.e. choosing any combination of 56-bit key from  $2^{56}$  combinations of keys) and then dropping bits so that its length remains 48 bits.

# DES Algorithm/DES Structure/ Encryption of the DES (continued...):

- ❑ The output found using key  $k_i$  after  $i_{th}$  round is represented by  $L_i$  and  $R_i$  respectively where  $i=1, 2, 3, \dots, 16$ . Round  $i$  has input  $L_{i-1} || R_{i-1}$  and output  $L_i || R_i$  where

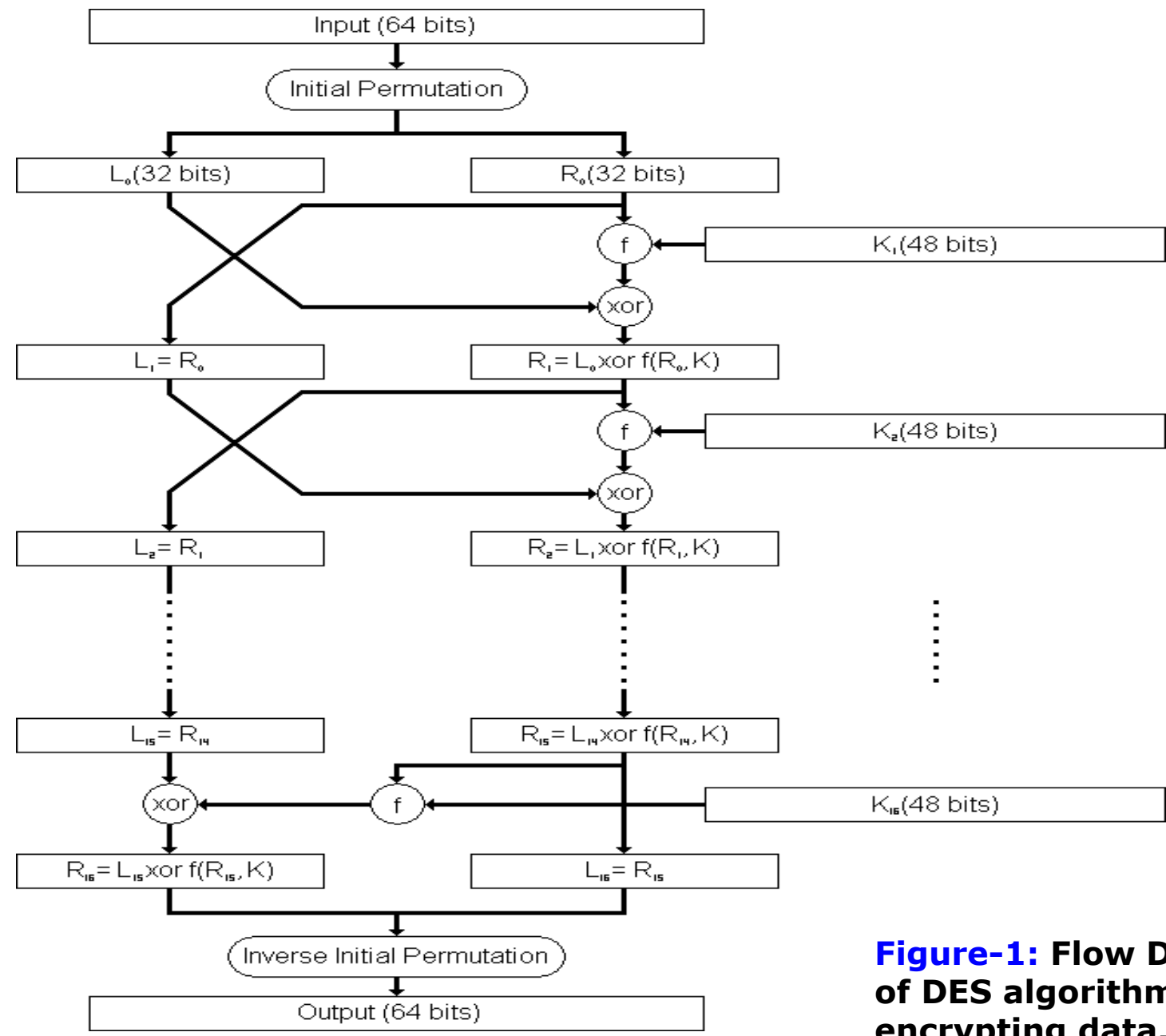
$$\diamond L_i = R_{i-1}$$

$$\diamond R_i = L_{i-1} \oplus f(R_{i-1}, k_i)$$

- ❑ In the final round, the left (L) and right (R) halves are swapped, so that the decryption algorithm has the same structure as the encryption algorithm.
- ❑ After the final round (16<sup>th</sup> round), the right and left halves are joined or concatenated.
- ❑ Then, a final permutation  $IP^{-1}$  (which is the inverse of the initial permutation defined in **Table-5**), is applied to the 64-bit joining block. The output of this final permutation is the 64 bit encrypted output (ciphertext).

- **Figure-1** below illustrates how the algorithm works.
- Decryption is identical to encryption, except that the subkeys are used in the opposite order. That is, subkey 16 is used in round 1, subkey 15 is used in round 2, etc., ending with subkey 1 being used in round 16.
- Each 64-bit block of plaintext is encrypted using a 56-bit key.

**DES Algorithm/DES Structure/ Encryption of the DES (continued...):**



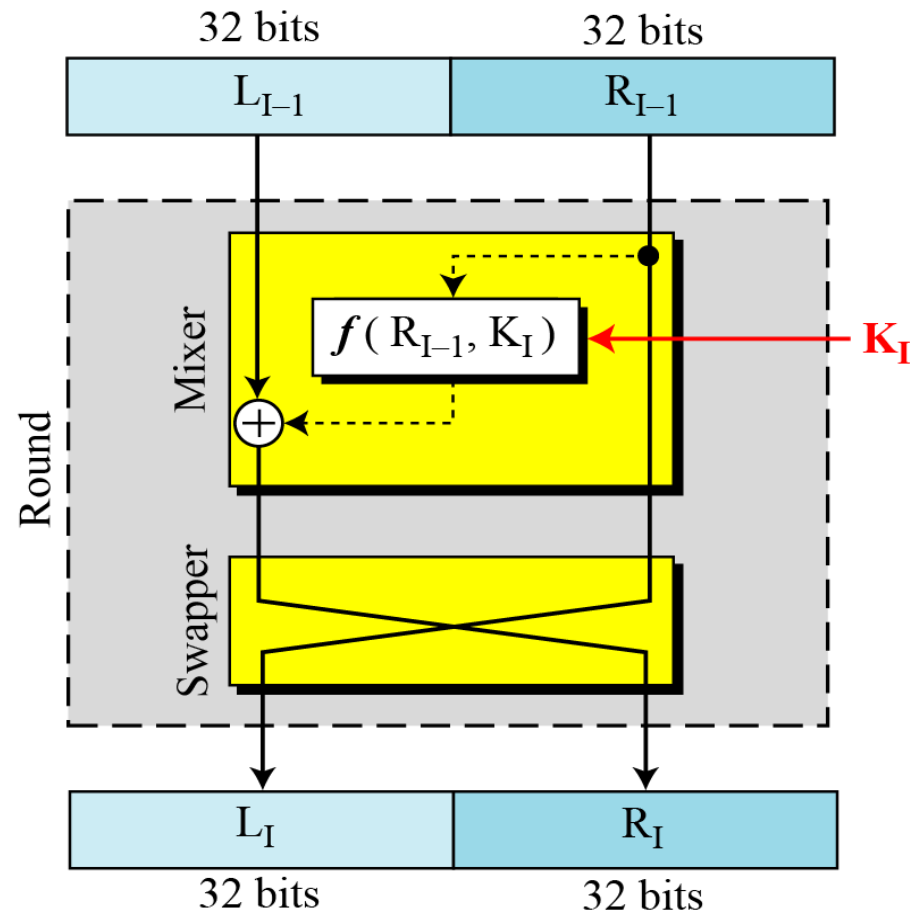
**Figure-1: Flow Diagram of DES algorithm for encrypting data.**

Prepared by: K M Akkas Ali, Assistant Professor, IIT, JU



# DES Rounds:

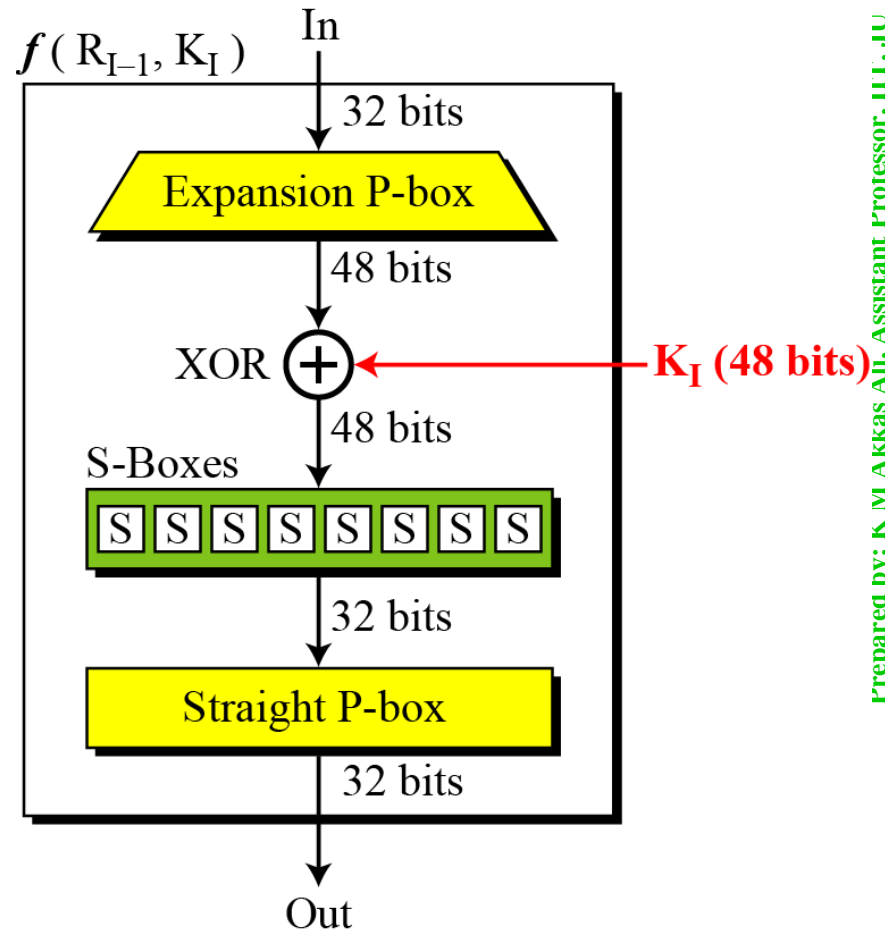
- In DES, substitution and permutation are used a number of times in iterations called rounds. Generally, **the more rounds there are, the more secure the algorithm is.**
- **DES uses 16 rounds.** Each round of DES is a Feistel cipher.
  - The round takes  $L_{i-1}$  and  $R_{i-1}$  from previous round (or the initial permutation box) and creates  $L_i$  and  $R_i$ , which go to the next round (or final permutation box).
  - **Each round has two cipher elements:** mixer and swapper. Each of these elements is invertible.
    - ❖ The swapper swaps the left half of the text with the right half. The mixer performs XOR operation.



**Figure:** A round in DES (encryption site)

# DES Round Function $f(R_{i-1}, K_i)$ :

- The heart of DES is DES round function.
- The round function mixes the bits of the right (R) portion using the subkey for the current round.
- It applies a 48-bit key to the rightmost 32 bits ( $R_{i-1}$ ) to produce a 32-bit output.
- All noninvertible elements in DES are collected inside the round function  $f(R_{i-1}, K_i)$
- This function is the main part of every round and consists of four sections:
  1. An expansion P-box (E-box, for 32 bit to 48 bit conversion)
  2. A whitener (Exclusive-or that adds key)
  3. A group of S-boxes (for 48 bit to 32 bit conversion)
  4. A straight permutation P-box



**Figure: DES function**

# DES Round Function $f(R_{i-1}, K_i)$ :

## 1. The E-box expansion permutation:

- 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.
- To do this, the 32-bit R value is expanded to 48 bits using an expansion P-box permutation table E (shown in Table-1).
- The expansion table defines a permutation plus an expansion.

32	1	2	3	4	5
4	5	6	7	8	9
8	9	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	30	31	32	1

Table-1: Expansion P-box table

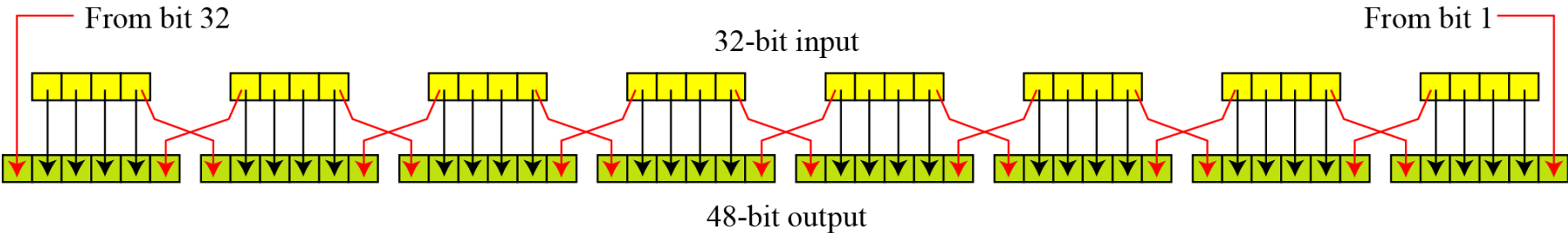
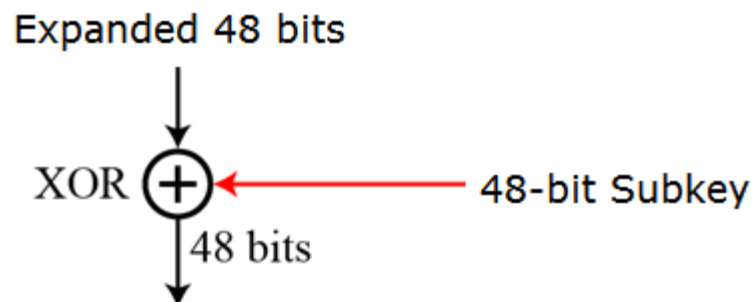


Figure: Expansion permutation

# DES Round Function $f(R_{i-1}, K_i)$ :

## 2. Whitener (Exclusive-or):

- 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.
- That expanded value is then exclusive-or'ed with the 48-bit subkey.



**Figure: Whitener**

# DES Round Function $f(R_{i-1}, K_i)$ :

## 3. The S-boxes (substitute 48 bits to 32 bits):

- In DES, a non-linearity is introduced into the encryption so that decryption will be computationally infeasible without the secret key. This is achieved with the use of S-boxes . which are basically non-linear substitution tables where either the output is smaller than the input or vice versa.
  - ❖ The S-boxes are the only non-linear operation in DES that do the real mixing (confusion).
- DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output, that is it accepts a 48-bit input and produces 32-bit number as output (defined in table-2).
  - ❖ The resulting 48 bits from whitener operation are divided into eight 6-bit chunks, each of which is fed into an S-Box that mixes the bits and produces a 4-bit output (The 8 S-boxes are shown in table-3). Those 4-bit outputs are combined into a 32-bit value.
  - ❖ The first and last bits of the 6-bit input of each S-box determine which column permutation is used. It provides non-linearity (confusion).

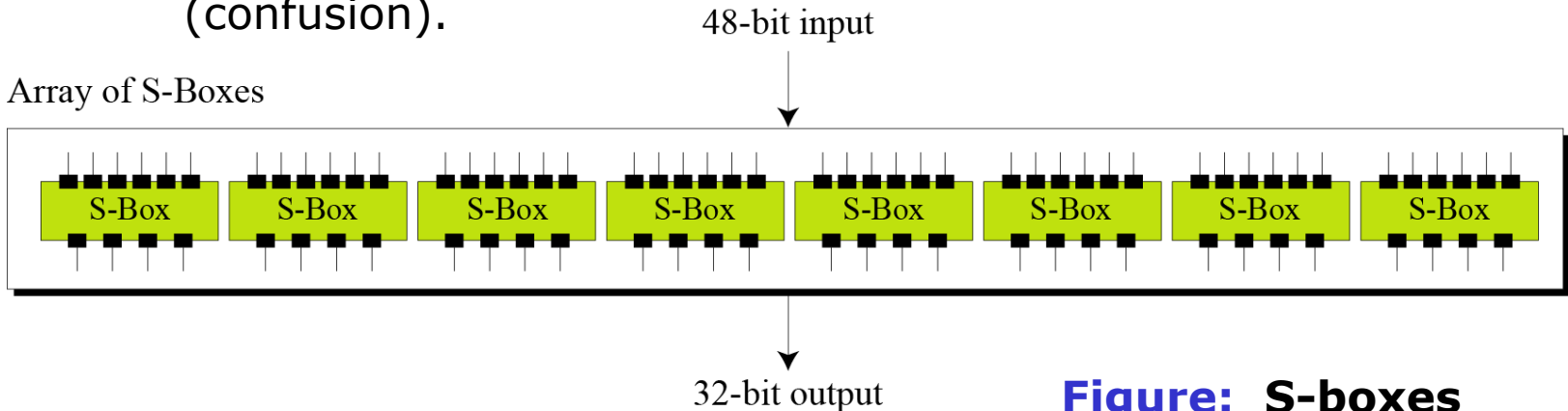


Figure: S-boxes

# DES Round Function $f(R_{i-1}, K_i)$ :

## S-box table:

- Table below shows the permutation for S-box 1. For the rest of the boxes see the textbook.

	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

Table-2: S-box 1

# DES Round Function $f(R_{i-1}, K_i)$ :

## Example: S-Box

- S-Box 1 is given below. The input to S-box 1 is  $B=101011$ . What is the output?

## Solution:

- Given input  $B=101011$
- Therefore,  $b_1b_6=11$ , which is 3 in decimal. So, row  $r=3$
- $b_2b_3b_4b_5=0101$ , which is 5 in decimal. So, column  $c=5$
- Now we look for the value of output in row 3, column 5 in the S-box 1. The result is 09 in decimal, which in binary is 1001. So, the input 101011 yields the output 1001.

	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

# DES Round Function $f(R_{i-1}, K_i)$ :

## S-Boxes:

➤ Tables below show the 8 S-boxes used in DES.

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

S-box 5

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

S-box 2

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

S-box 6

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

S-box 3

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

S-box 7

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

S-box 4

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

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	08	13	15	12	09	09	03	05	06	11

Table-3: S-Boxes (S-box 1 to S-box 8)



# DES Round Function $f(R_{i-1}, K_i)$ :

## 4. Straight Permutation (P-box):

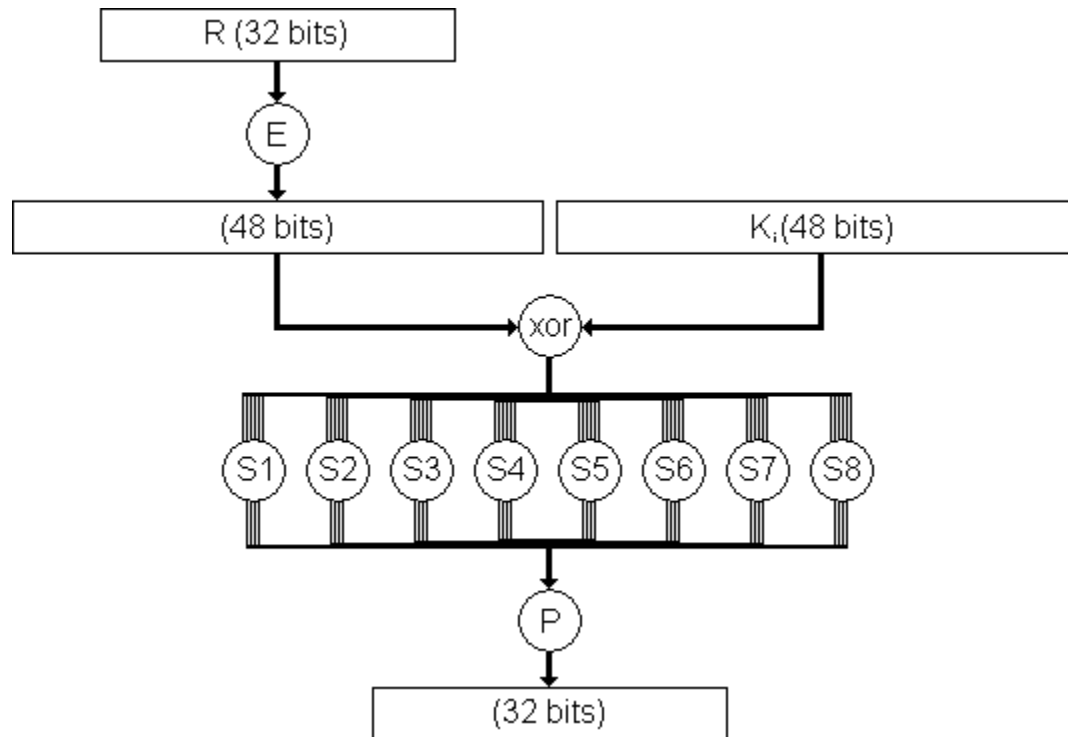
- The combined 32 bits from the previous step are permuted once again to produce the 32 bits output of the f-function using expansion P-box table (Shown in Table-4).

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

**Table-4: Straight permutation table**

# DES Round Function $f(R_{i-1}, K_i)$ :

- Figure below shows the DES round function  $f(x, k)$ .



**Figure-:** The complex  $f(x, k)$  function of the DES algorithm.

## Note:

- The s-boxes provide the “confusion” of data and key values, whilst the permutation  $P$  then spreads this as widely as possible, so each S-box output affects as many S-box inputs in the next round as possible, giving “diffusion”.

Initial Permutation (IP) and Final Permutation IP<sup>-1</sup>:

- The first step of the data computation in DES algorithm is initial permutation (IP) on {1, 2, 3, ....., 64}.
- Permutation reorders the input data bits.
- The initial and final permutations are straight P-boxes that are inverses of each other. They have no cryptography significance in DES.
- The initial permutation (IP) and its inverse (IP<sup>-1</sup>) are defined by tables, as shown in table-5 below.

IP

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

IP<sup>-1</sup>

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

Table-5: Initial and Final Permutation Tables

## Initial Permutation (IP) and Final Permutation IP<sup>-1</sup>: continue...

- The tables are to be interpreted as follows.
  - ❑ The input to a table consists of 64 bits numbered left to right from 1 to 64.
  - ❑ The 64 entries in the permutation table contain a permutation of the numbers from 1 to 64.
  - ❑ Each entry in the permutation table indicates the position of a numbered input bit in the output, which also consists of 64 bits.
  - ❑ The values in each matrix identify where each bit of the input message is mapped to in the output message. For example, the matrix for IP shows that the 58th bit from the input gets mapped to the first bit of the output; the 50th of the input maps to the second of the output, and so on.
  - ❑ Note that examples are specified using hexadecimal. Here a 64-bit plaintext value of "675a6967 5e5a6b5a" (written in left & right halves) after permuting with IP becomes "fffb2194d 004df6fb".

## Expansion Permutation Table (E)/Expansion P-box table:

- This table expands the 32 bit data to 48 bits
- ❑  $\text{Result}(i) = \text{input}(\text{array}(i))$
- ❑ The expansion table defines a permutation plus an expansion that involves duplication of 16 of the bits.

32	1	2	3	4	5
4	5	6	7	8	9
8	9	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	30	31	32	1

**Table-6: Expansion P-box table**

## S-Boxes:

- In DES, a non-linearity is introduced into the encryption so that decryption will be computationally infeasible without the secret key. This is achieved with the use of S-boxes which are basically non-linear substitution tables where either the output is smaller than the input or vice versa.
- In DES, the resulting 48 bits from expansion box are XORed with key.
- This 48 bit result passes through a substitution function comprising 8 S-boxes each of which maps 6 input bits to 4 output bits.
  - ❑ An S-box is a 4 by 16 matrix containing values from 00 to 15.
  - ❑ DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output.
  - ❑ Each S-box has its own table. So, for 64 bits input block of DES, we need a total of eight S-box tables.
  - ❑ Each of the unique selection functions  $S_1, S_2, \dots, S_8$ , takes a 6-bit block as input and yields a 4-bit block as output.

Example: S-Box

- S-Box 1 is given below. The input to S-box 1 is B=101011. What is the output?

Solution:

- Given input B=101011
- Therefore,  $b_1b_6=11$ , which is 3 in decimal. So, row  $r=3$
- $b_2b_3b_4b_5=0101$ , which is 5 in decimal. So, column  $c=5$
- Now we look for the value of output in row 3, column 5 in the S-box 1. The result is 09 in decimal, which in binary is 1001. So, the input 101011 yields the output 1001.

	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

# DES Elements/ DES Building Blocks (continued...):

## S-Boxes:

➤ Tables below show the 8 S-boxes.

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

S-box 5

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

S-box 2

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

S-box 6

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

S-box 3

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

S-box 7

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

S-box 4

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

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

Table-7: S-Boxes (S-box 1 to S-box 8)



# Various Permutation Tables used in DES:

## Various Permutation Tables:

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

**Table:** Initial Permutation Table, IP

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

**Table:** Inverse Initial Permutation Table, IP<sup>-1</sup>

32	1	2	3	4	5
4	5	6	7	8	9
8	9	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	30	31	32	1

**Table:** Expansion Permutation Table, E

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

**Table:** Permutation Function Table

# DES Decryption:

- Decryption is identical to encryption, except that the subkeys are used in the opposite order. The ciphertext is used as the input to the DES algorithm, but the keys  $K_i$  is used in reverse order. That is, subkey 16 is used in round 1, subkey 15 is used in round 2, etc., ending with subkey 1 being used in round 16.
- In encryption, round  $i$  has input  $L_{i-1} || R_{i-1}$  and output  $L_i || R_i$  where
  - ❑  $L_i = R_{i-1}$
  - ❑  $R_i = L_{i-1} \oplus f(R_{i-1}, k_i)$
  - ❑  $k_i$  is the subkey for the  $i$ th round
- In decryption, round  $i$  has input  $L_i || R_i$  and output  $L_{i-1} || R_{i-1}$  where
  - ❑  $R_{i-1} = L_i$
  - ❑  $L_{i-1} = R_i \oplus f(L_i, k_i)$
  - ❑  $k_i$  is the subkey for the  $i$ th round

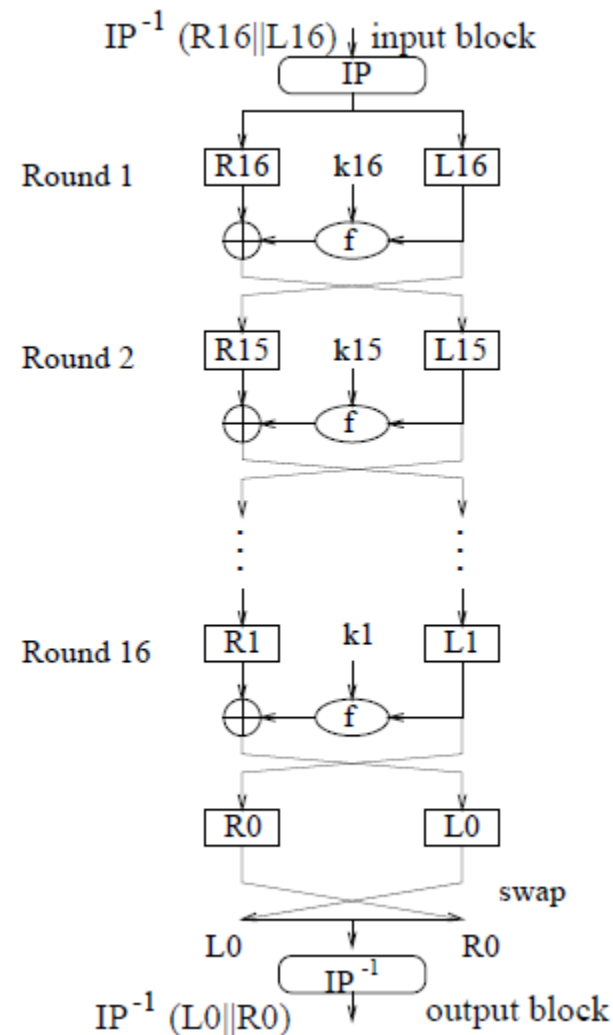
$$\begin{array}{lll} R_{15} & = & L_{16}, & L_{15} & = & R_{16} & \oplus & f(L_{16}, k_{16}) \\ R_{14} & = & L_{15}, & L_{14} & = & R_{15} & \oplus & f(L_{15}, k_{15}) \\ R_{13} & = & L_{14}, & L_{13} & = & R_{14} & \oplus & f(L_{14}, k_{14}) \\ & \vdots & & & \vdots & & & \\ R_2 & = & L_3, & L_2 & = & R_3 & \oplus & f(L_3, k_3) \\ R_1 & = & L_2, & L_1 & = & R_2 & \oplus & f(L_2, k_2) \end{array}$$

# DES Decryption (continue...):

- DES decryption is illustrated in the figure below:

## Remarks:

- The encryption and decryption process work, independent of how  $f(x,k)$  is designed! So different designs of the building block  $f(x,k)$  give different block ciphers.
- Security of DES depends on the design of round function  $f(x,k)$  and the key scheduling algorithm for producing the round subkeys.



**Figure: Decryption of DES**

# Modes of Operation in DES:

- The DES algorithm is a basic building block for providing data security.
- To apply DES in a variety of applications, five modes of operation have been defined which cover virtually all variation of use of the algorithm and these are shown in table-8 below.

Mode	Description	Typical Application
Electronic Codebook (ECB)	Each block of 64 plaintext bits is encoded independently using the same key.	•Secure transmission of single values (e.g., an encryption key)
Cipher Block Chaining (CBC)	The input to the encryption algorithm is the XOR of the next 64 bits of plaintext and the preceding 64 bits of ciphertext.	•General-purpose block-oriented transmission •Authentication
Cipher Feedback (CFB)	Input is processed J bits at a time. Preceding ciphertext is used as input to the encryption algorithm to produce pseudorandom output, which is XORed with plaintext to produce next unit of ciphertext.	•General-purpose stream-oriented transmission •Authentication
Output Feedback (OFB)	Similar to CFB, except that the input to the encryption algorithm is the preceding DES output.	•Stream-oriented transmission over noisy channel (e.g., satellite communication)
Counter (CTR)	Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block.	•General-purpose block-oriented transmission •Useful for high-speed requirements

Table-8: Modes of operation in DES

Prepared by: **K M Akkas Ali, Assistant Professor, IIT, JU**

## Example-1:

0x0002 0000 0000 0001

## Solution:

- The input in binary is:

[illegible]

- We see that only bit 15 and bit 64 are 1s; the other sixty-two bits are 0s. Therefore, the output must also have only two 1s and sixty-two 0s.
- Using initial permutation table (Table-1), we can find the output related to these two bits.
- According to the IP table, bit 15 in the input becomes bit 63 in the output. Bit 64 in the input becomes bit 25 in the output. The result in binary is:

**0000000000000000000000000000100000000000000000000000000**

- Which in hexadecimal is:

0x0000 0080 0000 0002

