



# CRYPTOGRAPHY

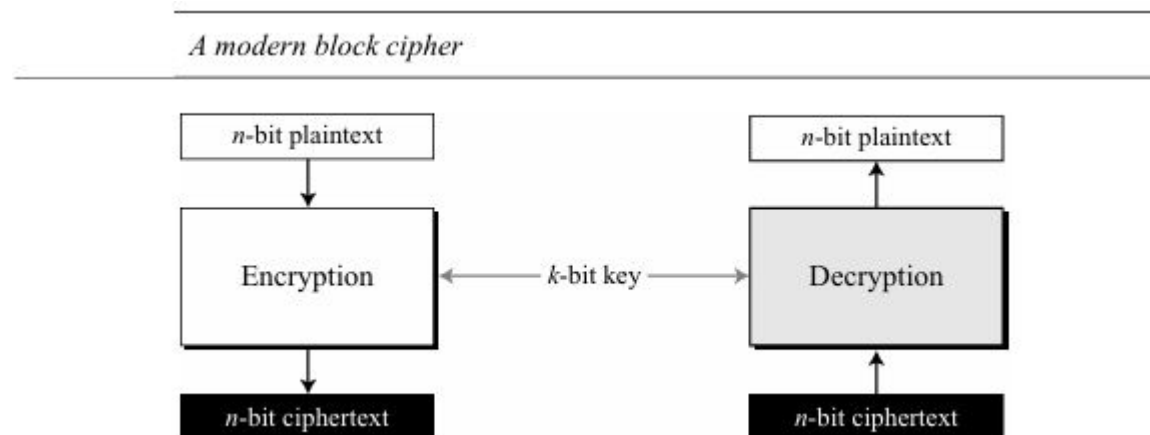
---

**Archana M**

Department of Computer Applications

# 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.
- The decryption algorithm must be the inverse of the encryption algorithm, and both operations must use the same secret key so that Bob can retrieve the message sent by Alice



- If the message has 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 for  $n$  are 64, 128, 256, or 512 bits.

## Example

---

How many padding bits must be added to a message of 100 characters if 8-bit ASCII is used for encoding and the block cipher accepts blocks of 64 bits?

***Solution:*** Encoding 100 characters using 8-bit ASCII results in an 800-bit message. The plaintext must be divisible by 64.

If  $|M|$  and  $|Pad|$  are the length of the message and the length of the padding

$$|M| + |Pad| = 0 \bmod 64 \quad \rightarrow \quad |Pad| = -800 \bmod 64 \quad \rightarrow \quad 32 \bmod 64$$

This means that 32 bits of padding (for example, 0's) need to be added to the message. The plain text then consists of 832 bits or thirteen 64-bit blocks

Note that only the last block contains padding. The cipher uses the encryption algorithm thirteen times to create thirteen ciphertext blocks.

# Substitution or Transposition

---

- A modern block cipher can be designed to act as a substitution cipher or a transposition cipher.
- If the cipher is designed as a substitution 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 in the plaintext and in the ciphertext.
- In either case, the number of n-bit possible plaintexts or ciphertexts is  $2^n$  because each of the n bits in the block can have one of the two values, 0 or 1

## 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?

- The cipher is designed as a substitution cipher.
- The cipher is designed as a transposition cipher.

# Substitution or Transposition

---

## Solution:

In the first case (substitution), Eve has no idea how many 1's are in the plaintext. Eve needs to try all possible  $2^{64}$ -bit blocks 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.*

In the second case (transposition), Eve knows that there are exactly 10 1's in the plain text, 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  $(64!) / [(10!) (54!)] = 151,473,214,816$  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.*

# Components of a Modern Block Cipher

---

- 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 transposition units (called P-boxes), substitution units (called S-boxes).

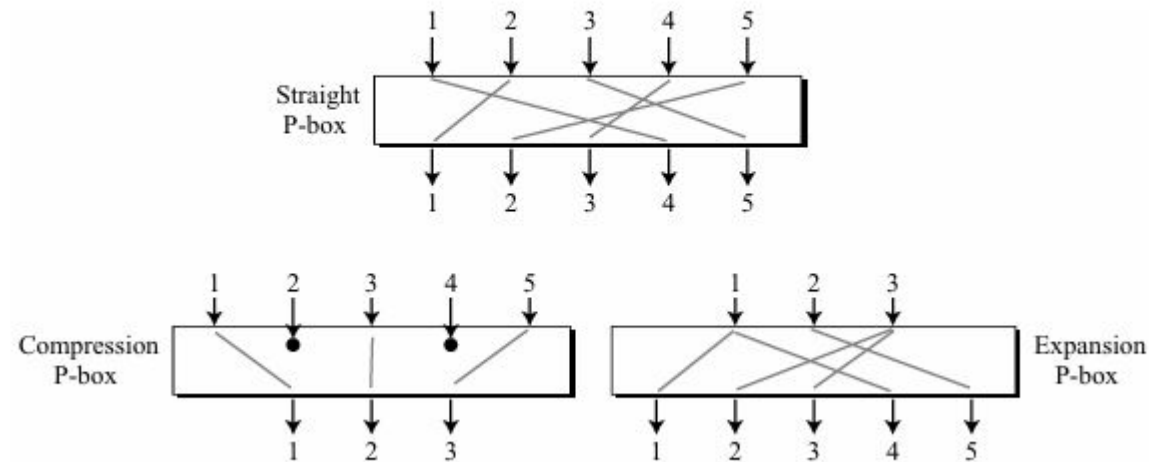
## *P-Boxes*

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

We can find three types of P-boxes in modern block ciphers:

- straight P-boxes, expansion P-boxes, and compression P-boxes

*Three types of P-boxes*



**Straight P-Boxes** A straight P-Box with  $n$  inputs and  $n$  outputs is a permutation. There are  $n!$  possible mappings

## Straight P-Boxes

Although a P-box can use a key to define one of the  $n!$  mappings, P-boxes are normally keyless, which means that the mapping is predetermined.

- If the P-box is implemented in hardware, it is prewired;
- if it is implemented in software, a permutation table shows the rule of mapping

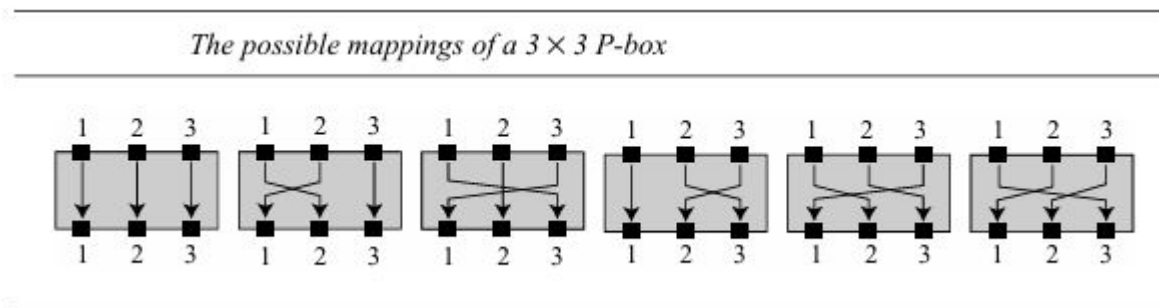


Figure shows all 6 possible mappings of a  $3 \times 3$  P-box.

- In the second case, the entries in the table are the inputs and the positions of the entries are the outputs
- Table 5.1 shows an example of a straight permutation table when  $n$  is 64.(corresponding to the 64 inputs)
- The position (index) of the entry corresponds to the output. Because the first entry contains the number 58, we know that the first output comes from the 58th input.
- Because the last entry is 7, we know that the 64th output comes from the 7th input, and so on

**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

## *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.

The compression P-boxes used in modern block ciphers normally are keyless with a permutation table showing the rule for transposing bits

Table 5.2 shows an example of a permutation table for a  $32 \times 24$  compression P-box. Note that inputs 7, 8, 9, 15, 16, 23, 24, and 25 are blocked.

**Table 5.2** *Example of a  $32 \times 24$  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 input
  - The expansion P-boxes used in modern block ciphers normally are keyless, where a permutation table shows the rule for transposing bit.
  - We need to know that a permutation table for an expansion P-box has  $m$  entries, but  $m - n$  of the entries are repeated (those inputs mapped to more than one output).
- 
- Table 5.3 shows an example of a permutation table for a  $12 \times 16$  expansion P-box. Note that each of the inputs 1, 3, 9, and 12 is mapped to two outputs.

**Table 5.3** *Example of a  $12 \times 16$  permutation table*

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

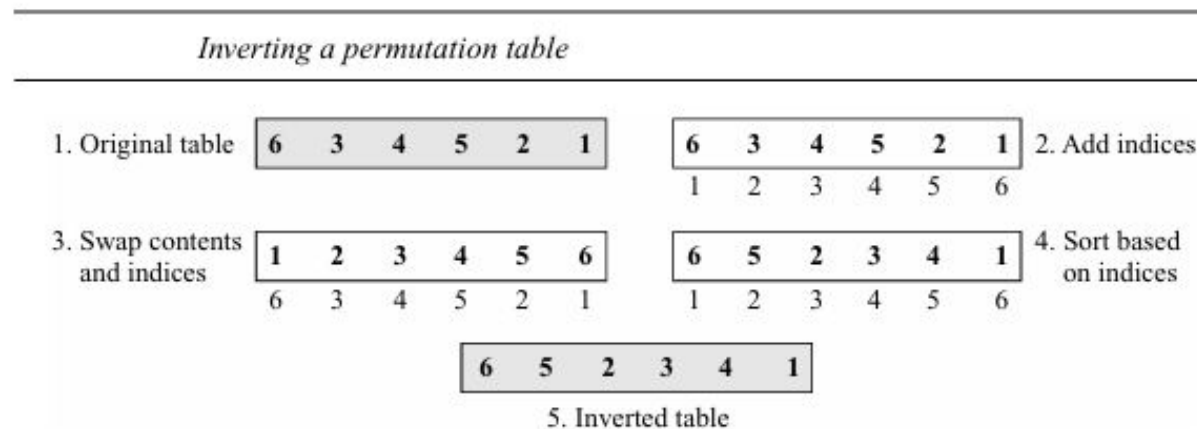
## Invertibility

A straight P-box is invertible.

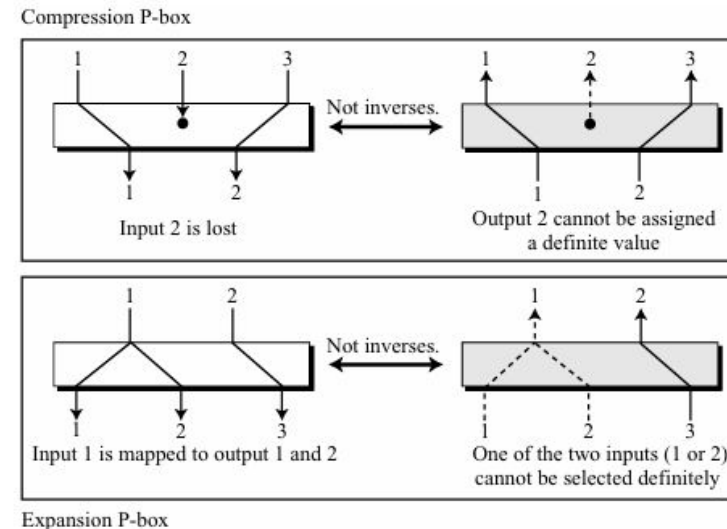
This means that we can use a straight P-box in the encryption cipher and its inverse in the decryption cipher.

The permutation tables, however, need to be the inverses of each other.

Figure 5.6 shows how to invert a permutation table represented as a one-dimensional table.



- Compression and expansion P-boxes have no inverses.
- 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
- In an expansion P-box, an input may be mapped to more than one output during encryption;



# S-Boxes

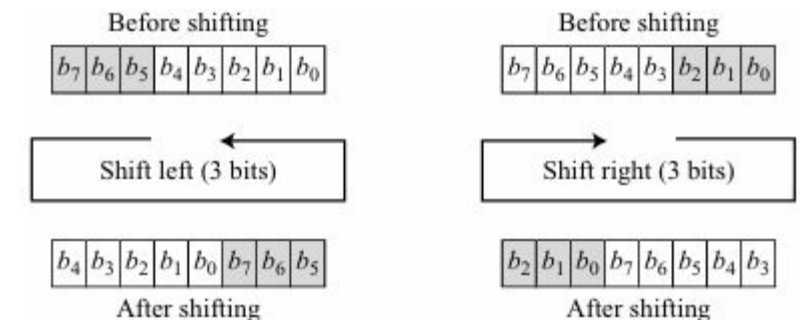
---

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

# Circular Shift

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

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



# Product Ciphers

---

- Shannon introduced the concept of a product cipher
- A product cipher is a complex cipher combining substitution, permutation
- Shannon's idea in introducing the product cipher was to enable the block ciphers to have two important properties: *diffusion and confusion*

# Diffusion

---

- The idea of diffusion is to hide the relationship between the ciphertext and the plaintext
- This will frustrate the adversary who uses ciphertext statistics to find the plaintext.
- Diffusion implies that each symbol (character or 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.

*Diffusion hides the relationship between the ciphertext and the plaintext.*

# Confusion

---

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

*Confusion hides the relationship between the ciphertext and the key*

# Two Classes of Product Ciphers

---

- Modern block ciphers are all product ciphers, but they are divided into two classes.
  - The ciphers in the first class use both invertible and noninvertible components
  - The ciphers in this class are normally referred to as ***Feistel ciphers***
  - ***DES*** is a good example of a Feistel cipher

- 
- The ciphers in the second class use only invertible components.
  - We refer to ciphers in this class as non-Feistel ciphers
  - The block cipher AES is a good example of a non-Feistel cipher

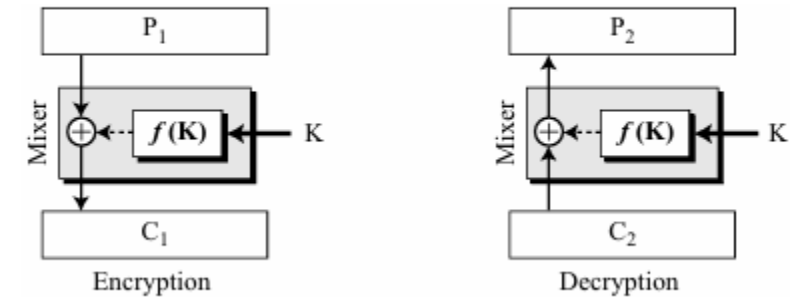
# Feistel Ciphers

---

- 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.
- A Feistel cipher combines all noninvertible elements in a unit and uses the same unit in the encryption and decryption algorithms
- The question is how the encryption and decryption algorithms are inverses of each other if each has a non invertible unit.
- Feistel showed that they can be canceled out.

- In the encryption, a noninvertible function,  $f(K)$ , accepts the key as the input.
- The output of this component is exclusive-ored with the plaintext.
- The result becomes the ciphertext.
- We call the combination of the function and the exclusive-or operation the mixer (for lack of another name).
- The mixer plays an important role in the later development of the Feistel cipher.

*The first thought in Feistel cipher design*



- Because the key is the same in encryption and decryption, we can prove that the two algorithms are inverses of each other.
- In other words, If  $C_2=C_1$  (no change in the ciphertext during transmission) then  $P_2=P_1$

**Encryption:**  $C_1 = P_1 \oplus f(K)$

**Decryption:**  $P_2 = C_2 \oplus f(K) = C_1 \oplus f(K) = P_1 \oplus f(K) \oplus f(K) = P_1 \oplus (00\dots 0) = P_1$

- The above argument proves that, although the mixer has a noninvertible element, the mixer itself is self-invertible.

# 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

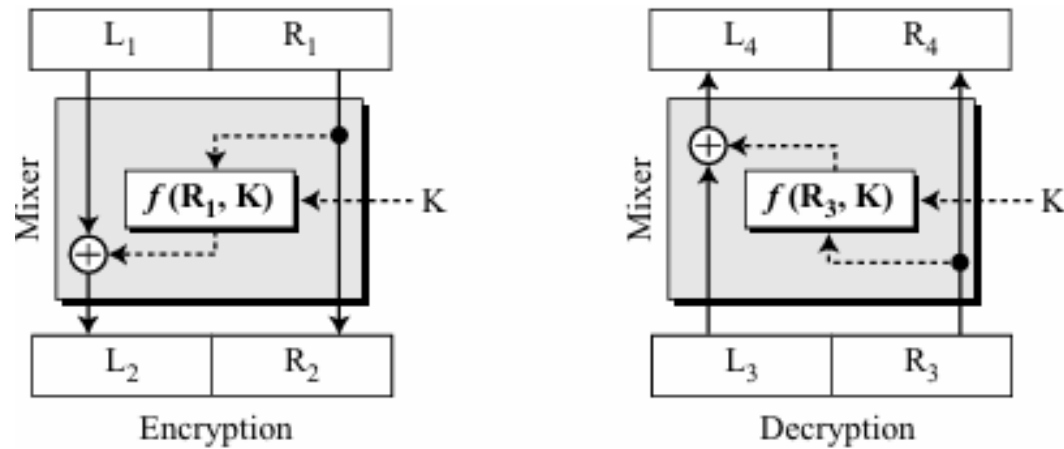
- The function extracts the first and second bits to get 11 in binary or 3 in decimal. The result of squaring is 9, which is 1001 in binary.

**Encryption:**  $C = P \oplus f(K) = 0111 \oplus 1001 = 1110$

**Decryption:**  $P = C \oplus f(K) = 1110 \oplus 1001 = 0111$     Same as the original P

- The function  $f(101) = 1001$  is noninvertible, but the exclusive-or operation allows us to use the function in both encryption and decryption algorithms.
- In other words, the function is non-invertible, but the mixer is self-invertible.

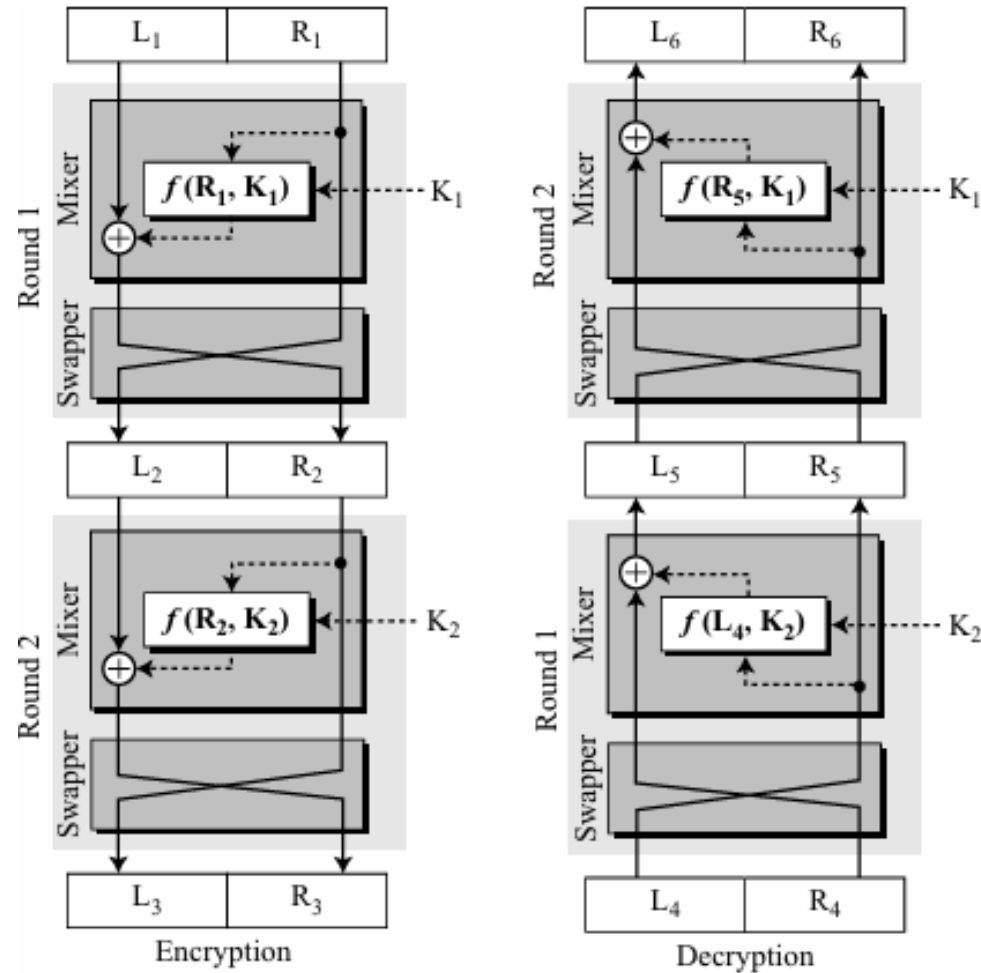
*Improvement of the previous Feistel design*



- The plaintext used in the encryption algorithm is correctly regenerated by the decryption algorithm.

# Final Design

*Final design of a Feistel cipher with two rounds*



# 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.
- 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.

# *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).

## History

- In 1973, NIST published a request for proposals for a national symmetric-key crypto system.
- 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)

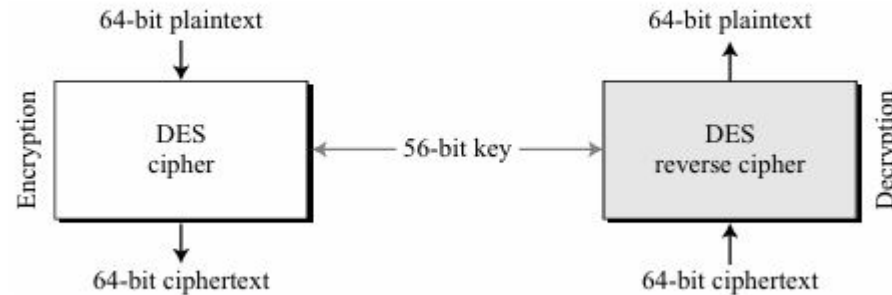
- 
- After the publication, the draft was criticized severely for two reasons. First, critics questioned the small key length (only 56 bits), which could make the cipher vulnerable to brute-force attack.
  - Second, critics were concerned about some hidden design behind the internal structure of DES.
  - They were suspicious that some part of the structure (the S-boxes) may have some hidden trapdoor that would allow the National Security Agency (NSA) to decrypt the messages without the need for the key
  - Later IBM designers mentioned that the internal structure was designed to prevent differential cryptanalysis.

- DES was finally published as FIPS 46 in the Federal Register in January 1977.
- NIST, however, defines DES as the standard for use in unclassified applications.
- DES has been the most widely used symmetric-key block cipher since its publication
- NIST later issued a new standard (FIPS 46-3) that recommends the use of triple DES (repeated DES cipher three times) for future applications

# Overview

- DES is a block cipher, as shown in Figure

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

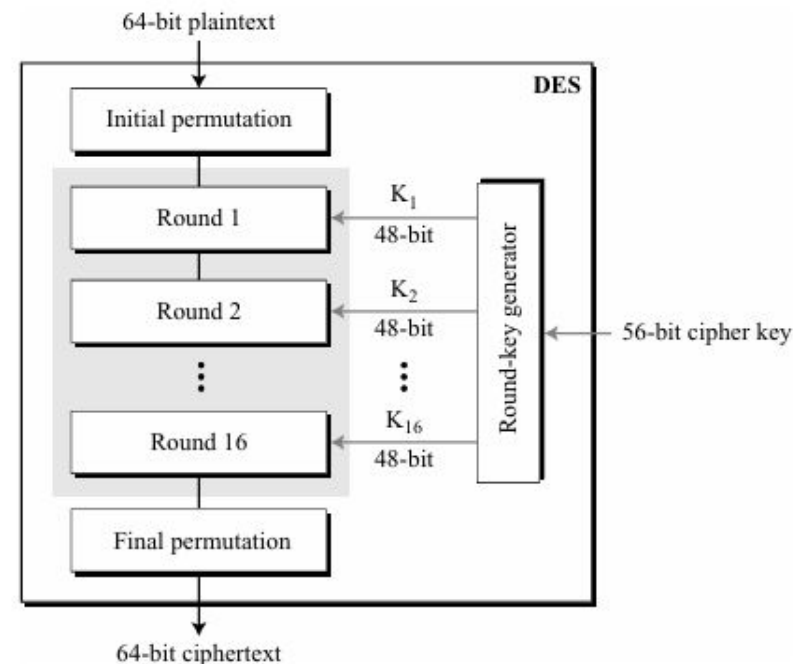


- At the encryption site, DES takes a 64-bit plaintext and creates a 64-bit ciphertext;
- At the decryption site, DES takes a 64-bit ciphertext and creates a 64-bit block of plain text.
- The same 56-bit cipher key is used for both encryption and decryption.

# 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 shows the elements of DES cipher at the encryption site.

**Figure 6.2** General structure of DES

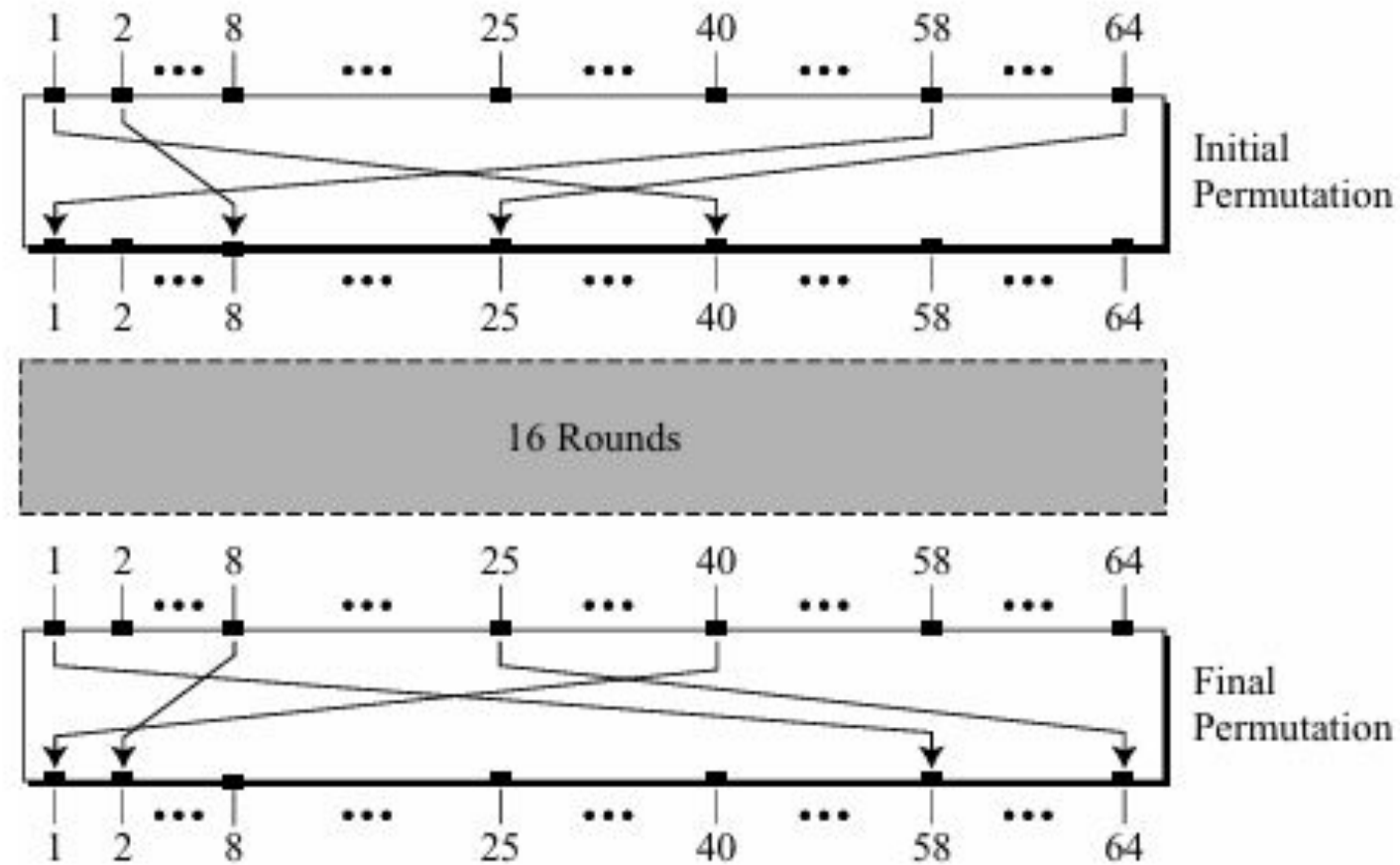


# Initial and Final Permutations

---

- The figure 6.3 shows the initial and final permutations (P-boxes).
- Each of these permutations takes a 64-bit input and permutes them according to a predefined rule.
- These permutations are keyless straight permutations that are the inverse of each other.
- For example, in the initial permutation, the 58th bit in the input becomes the first bit in the output.
- Similarly in the final permutation, the first bit in the input becomes the 58th bit in the output.

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



- The permutation rules for these P-boxes are shown in Table 6.1.
- Each side of the table can be thought of as a 64-element array.

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

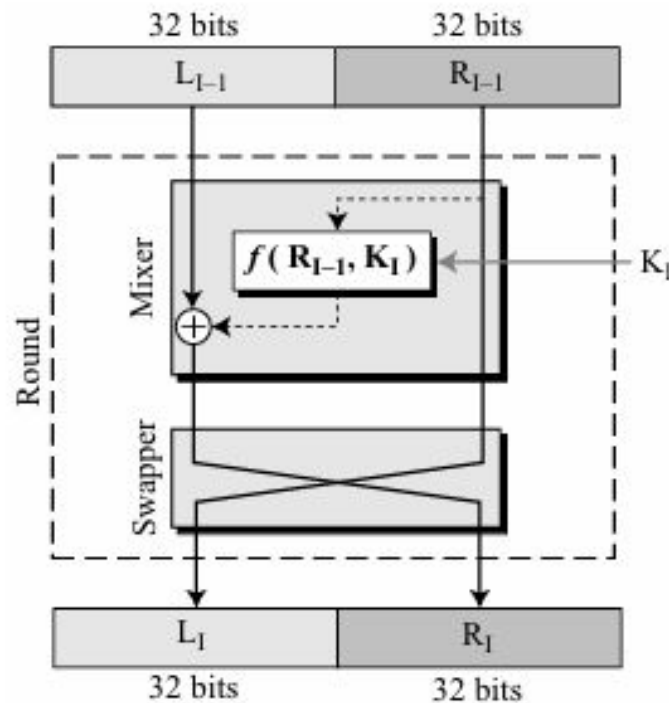
<i>Initial Permutation</i>	<i>Final Permutation</i>
58 50 42 34 26 18 10 02	40 08 48 16 56 24 64 32
60 52 44 36 28 20 12 04	39 07 47 15 55 23 63 31
62 54 46 38 30 22 14 06	38 06 46 14 54 22 62 30
64 56 48 40 32 24 16 08	37 05 45 13 53 21 61 29
57 49 41 33 25 17 09 01	36 04 44 12 52 20 60 28
59 51 43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

- *These two permutations have no cryptography significance in DES. Both permutations are keyless and predetermined. The reason they are included in DES is not clear and has not been revealed by the DES designers*

# Rounds

- DES uses 16 rounds. Each round of DES is a Feistel cipher, as shown in Figure 6.4

**Figure 6.4** *A round in DES (encryption site)*

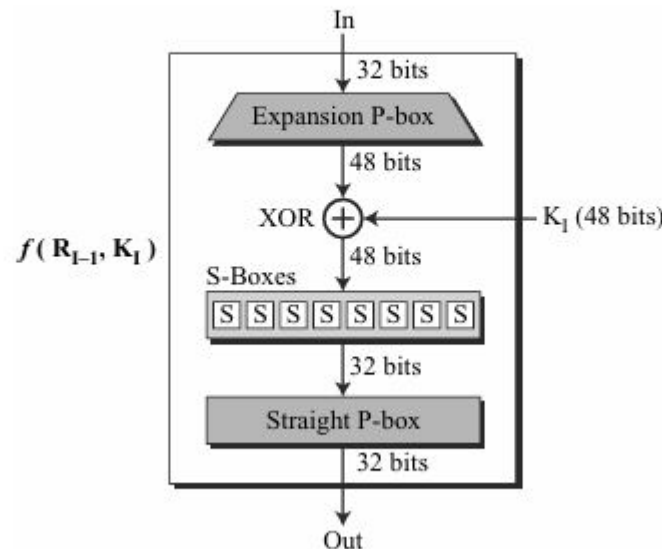


- The round takes  $L_{I-1}$  and  $R_{I-1}$  from previous round (or the initial permutation box) and creates  $L_I$  and  $R_I$ , which go to the next round (or final permutation box).
- we can assume that each round has two cipher elements (mixer and swapper).
- Each of these elements is invertible. The swapper is obviously invertible.
- It swaps the left half of the text with the right half.
- The mixer is invertible because of the XOR operation.
- All noninvertible elements are collected inside the function  $f(R_{I-1}, K_I)$

# DES Function

- The heart of DES is the DES function. The DES function applies a 48-bit key to the rightmost 32 bits  $R_{I-1}$  to produce a 32-bit output.
- This function is made up of four sections: an expansion P-box, a whitener (that adds key), a group of S-boxes, and a straight P-box as shown in Figure 6.5.

Figure 6.5 DES function

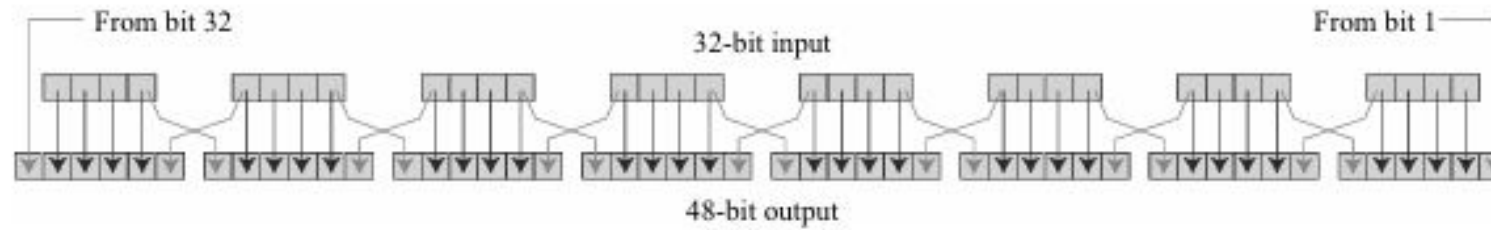


# Expansion P-box

---

- Since  $\mathbf{R}_{I-1}$  is a 32-bit input and  $\mathbf{K}_I$  is a 48 bit key we first need to expand  $\mathbf{R}_{I-1}$  to 48 bits.  $\mathbf{R}_{I-1}$  is divided into 8 4-bit sections.
- Each 4-bit section is then expanded to 6 bits.
- This expansion permutation follows a predetermined rule.
- For each section, input bits 1, 2, 3, and 4 are copied to output bits 2, 3, 4, and 5, respectively.
- Output bit 1 comes from bit 4 of the previous section; output bit 6 comes from bit 1 of the next section
- If sections 1 and 8 can be considered adjacent sections, the same rule applies to bits 1 and 32
- Figure 6.6 shows the input and output in the expansion permutation

**Figure 6.6** *Expansion permutation*



**Table 6.2** *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	30	31	32	01

- 
- Although the relationship between the input and output can be defined mathematically, DES uses Table 6.2 to define this P-box.
  - Note that the number of output ports is 48, but the value range is only 1 to 32. Some of the inputs go to more than one output.
  - For example, the value of input bit 5 becomes the value of output bits 6 and 8

# Whitener (XOR)

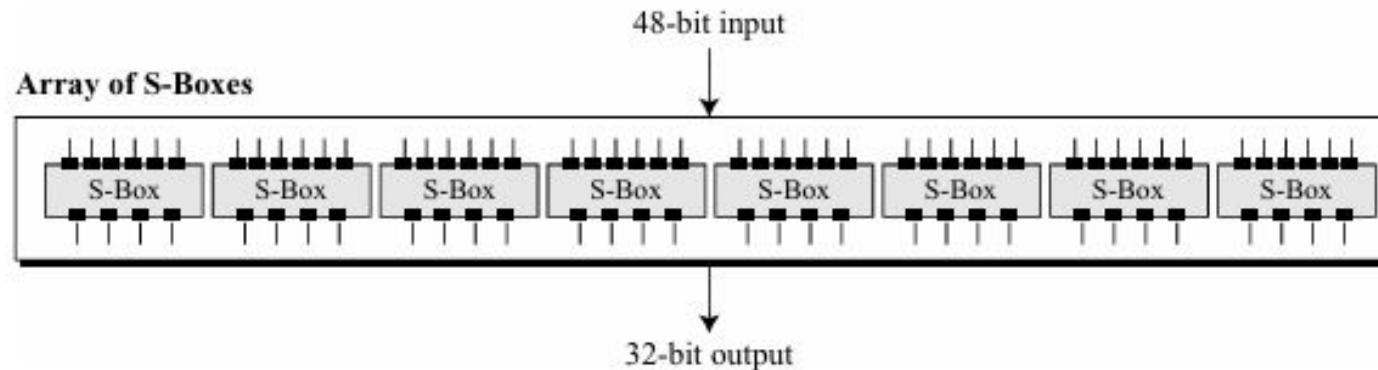
---

- 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

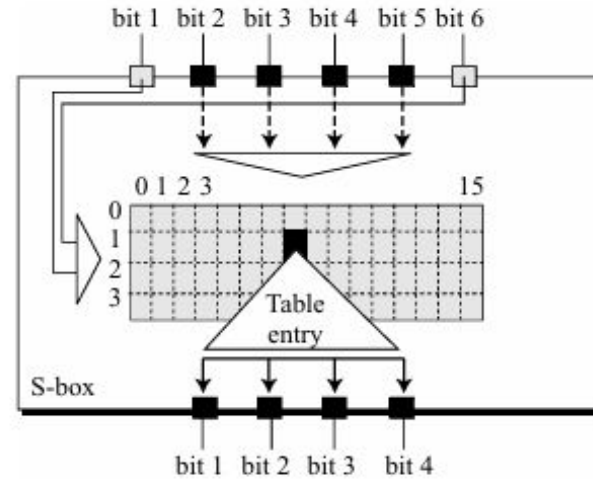
- The S-boxes do the real mixing (confusion).
- DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output. See Figure 6.7

**Figure 6.7** *S-boxes*



- 
- The 48-bit data from the second operation is divided into eight 6-bit chunks, and each chunk is fed into a box.
  - The result of each box is a 4-bit chunk;
  - when these are combined the result is a 32-bit text
  - The substitution in each box follows a pre-determined rule based on a 4-row by 16-column table.
  - The combination of bits 1 and 6 of the input defines one of four rows;
  - the combination of bits 2 through 5 defines one of the sixteen columns

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



- Because each S-box has its own table, we need eight tables, as shown in Tables 6.3 to 6.10, to define the output of these boxes.
- The values of the inputs (row number and column number) and the values of the outputs are given as decimal numbers 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

**Table 6.4** *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

**Table 6.5** *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

**Table 6.6** *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

**Table 6.7** *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

**Table 6.8** *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

**Table 6.9** *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

**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

# Example

---

1. The input to S-box 1 is 1 00011 . What is the output?

Solution : the input 100011 yields the output 1100

2. The input to S-box 8 is 00000 . What is the output?

Solution : the input 000000 yields the output 1101.

# Cipher and Reverse Cipher

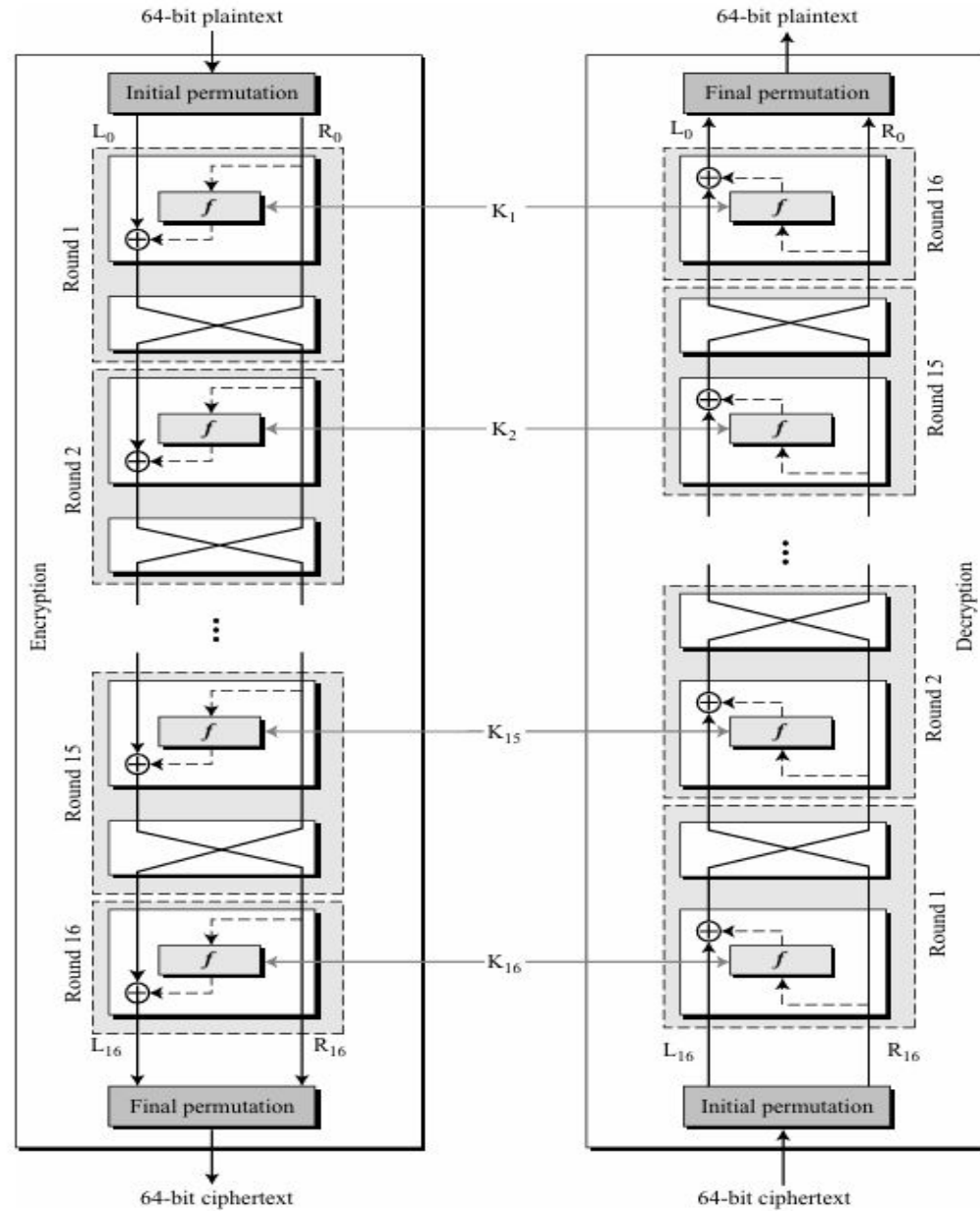
---

- Using mixers and swappers, we can create the cipher and reverse cipher, each having 16 rounds.
- The cipher is used at the encryption site;
- the reverse cipher is used at the decryption site.
- The whole idea is to make the cipher and the reverse cipher algorithms similar.

## **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.

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



# Algorithm

- Algorithm 6.1 gives the pseudocode for the cipher and four corresponding routines in the first approach.

**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)
}

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

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

```
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)
}

substitute (inBlock[32], outBlock[48], SubstitutionTables[8, 4, 16])
{
    for (i = 1 to 8)
    {
        row  $\leftarrow 2 \times \text{inBlock}[i \times 6 + 1] + \text{inBlock}[i \times 6 + 6]$ 
        col  $\leftarrow 8 \times \text{inBlock}[i \times 6 + 2] + 4 \times \text{inBlock}[i \times 6 + 3] +$ 
             $2 \times \text{inBlock}[i \times 6 + 4] + \text{inBlock}[i \times 6 + 5]$ 

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

        outBlock[[i × 4 + 1]  $\leftarrow$  value / 8;           value  $\leftarrow$  value mod 8
        outBlock[[i × 4 + 2]  $\leftarrow$  value / 4;           value  $\leftarrow$  value mod 4
        outBlock[[i × 4 + 3]  $\leftarrow$  value / 2;           value  $\leftarrow$  value mod 2
        outBlock[[i × 4 + 4]  $\leftarrow$  value

    }
}
```

# DES ANALYSIS

---

Properties - Two desired properties of a block cipher are the avalanche effect and the completeness.

## Avalanche Effect

- Avalanche effect means a small change in the plaintext (or key) should create a significant change in the ciphertext.
- DES has been proved to be strong with regard to this property.

### *Example 6.7*

To check the avalanche effect in DES, let us encrypt two plaintext blocks (with the same key) that differ only in one bit and observe the differences in the number of bits in each round.

Plaintext: 0000000000000000  
Ciphertext: 4789FD476E82A5F1

Key: 22234512987ABB23

Plaintext: 00000000000000001  
Ciphertext: 0A4ED5C15A63FEA3

Key: 22234512987ABB23

---

## Completeness effect

- Completeness effect means that each bit of the ciphertext needs to depend on many bits on the plaintext.
- The diffusion and confusion produced by P-boxes and S-boxes in DES, show a very strong completeness effect

# DES Weaknesses

---

## Weaknesses in Cipher Design

S-boxes: At least three weaknesses are mentioned in the literature for S-boxes.

1. In S-box 4, the last three output bits can be derived in the same way as the first output bit by complementing some of the input bits.
2. Two specifically chosen inputs to an S-box array can create the same output.
3. It is possible to obtain the same output in a single round by changing bits in only three neighboring S-boxes

---

P-boxes One mystery and one weakness were found in the design of P-boxes:

1. It is not clear why the designers of DES used the initial and final permutations; these have no security benefits.
2. In the expansion permutation (inside the function), the first and fourth bits of every 4-bit series are repeated.

# Weakness in the Cipher Key

---

**Key Size:** Critics believe that the most serious weakness of DES is in its key size (56 bits).

To do a brute-force attack on a given ciphertext block, the adversary needs to check  $2^{56}$  keys:

- With available technology, it is possible to check one million keys per second. This means that we need more than two thousand years to do brute-force attacks on DES using only a computer with one processor.
- Computer networks can simulate parallel processing. In 1977 a team of researchers used 3500 computers attached to the Internet to find a key challenged by RSA Laboratories in 120 days. The key domain was divided among all of these computers, and each computer was responsible to check the part of the domain.

- 
- If we can make a computer with one million chips (parallel processing), then we can test the whole key domain in approximately 20 hours.
  - When DES was introduced, the cost of such a computer was over several million dollars, but the cost has dropped rapidly. A special computer was built in 1998 that found the key in 112 hours.
  - If 3500 networked computers can find the key in 120 days, a secret society with 42,000 members can find the key in 10 days.

# STREAM AND BLOCK CIPHERS

## Stream Ciphers

- In a stream cipher, encryption and decryption are done one symbol (such as a character or a bit) at a time.
- We have a plaintext stream, a ciphertext stream, and a key stream.
- Call the plaintext stream  $P$ , the ciphertext stream  $C$ , and the key stream  $K$ .

$$P = P_1 P_2 P_3, \dots$$

$$C = C_1 C_2 C_3, \dots$$

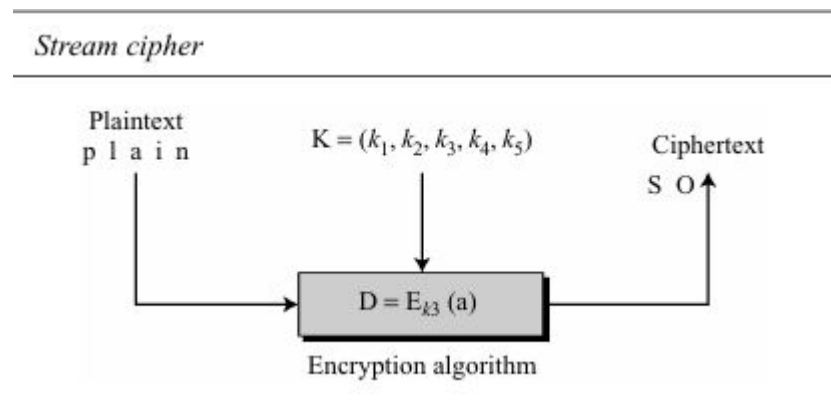
$$K = (k_1, k_2, k_3, \dots)$$

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

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

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

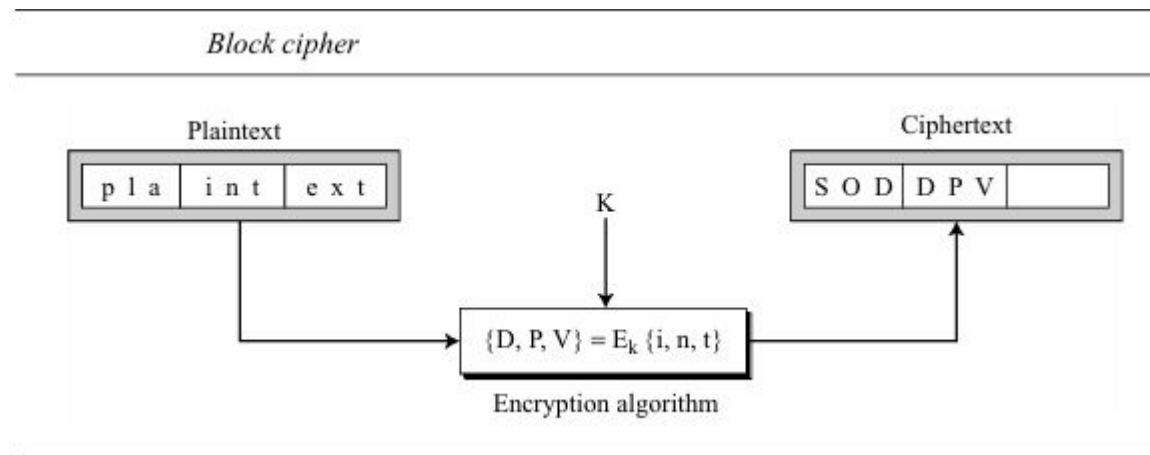
- Characters in the plaintext are fed into the encryption algorithm, one at a time; the ciphertext characters are also created one at a time.
- The key stream, can be created in many ways. It may be a stream of predetermined values;
- It may be created one value at a time using an algorithm.
- The values may depend on the plaintext or ciphertext characters. The values may also depend on the previous key values.



## Block Ciphers

In a block cipher, a group of plaintext symbols of size  $m$  ( $m > 1$ ) are encrypted together creating a group of ciphertext of the same size.

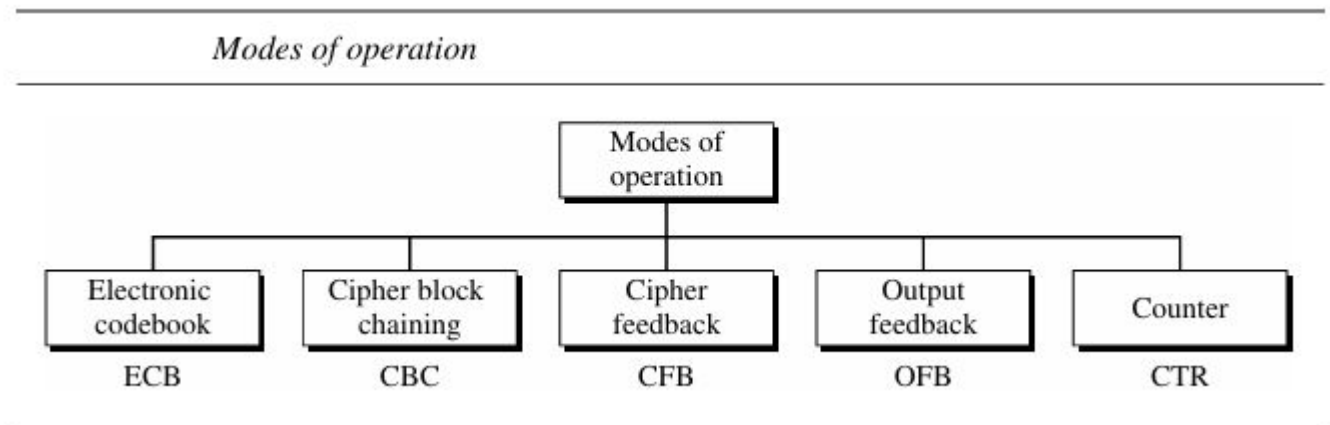
Based on the definition, in a block cipher, a single key is used to encrypt the whole block even if the key is made of multiple values.



In a block cipher, a ciphertext block depends on the whole plaintext block.

# USE OF MODERN BLOCK CIPHERS

- Symmetric-key encipherment can be done using modern block ciphers.
- DES encrypts and decrypts a block of 64 bits; AES encrypts and decrypts a block of 128 bits
- In real life applications, the text to be enciphered is of variable size and normally much larger than 64 or 128 bits
- ***Modes of operation*** have been devised to encipher text of any size employing either DES or AES.



# Electronic Codebook (ECB) Mode

---

- The simplest mode of operation is called the electronic codebook (ECB) mode.
- The plaintext is divided into  $N$  blocks.
- The block size is  $n$  bits. If the plaintext size is not a multiple of the block size, the text is padded to make the last block the same size as the other blocks.
- The same key is used to encrypt and decrypt each block

**Figure 8.2** *Electronic codebook (ECB) mode*

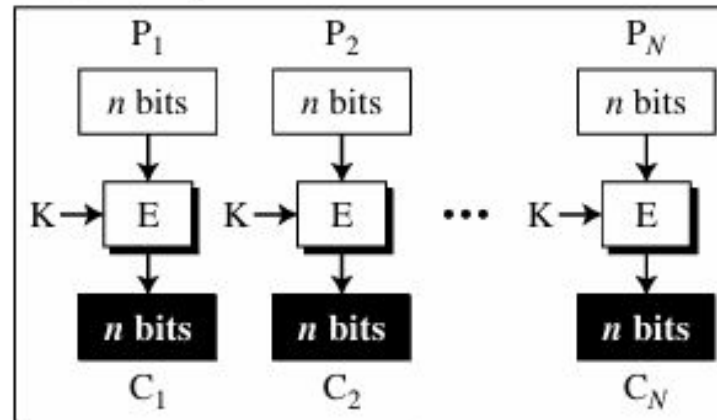
E: Encryption

D: Decryption

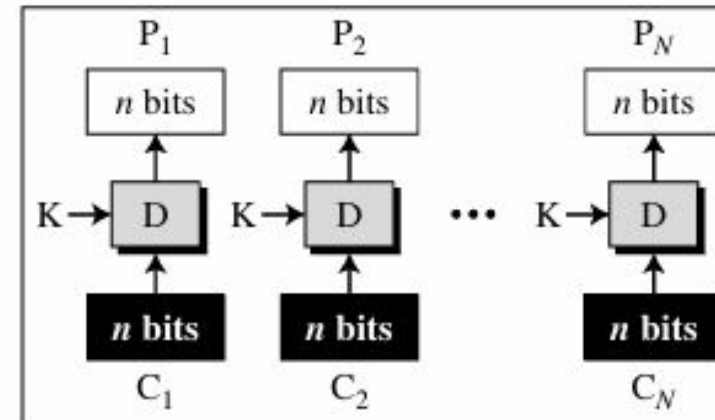
$P_i$ : Plaintext block  $i$

$C_i$ : Ciphertext block  $i$

K: Secret key



Encryption



Decryption

The relation between plaintext and ciphertext block is shown below:

Encryption:  $C_i = E_K (P_i)$

Decryption:  $P_i = D_K (C_i)$

# Cipher Block Chaining (CBC) Mode

---

- In CBC mode, each plaintext block is exclusive-ored with the previous ciphertext block before being encrypted.
- When a block is enciphered, the block is sent, but a copy of it is kept in memory to be used in the encryption of the next block.
- The reader may wonder about the initial block. There is no ciphertext block before the first block.
- In this case, a phony block called the initialization vector (IV) is used ,  
The sender and receiver agree upon a specific predetermined IV

**Figure 8.3** Cipher block chaining (CBC) mode

E: Encryption

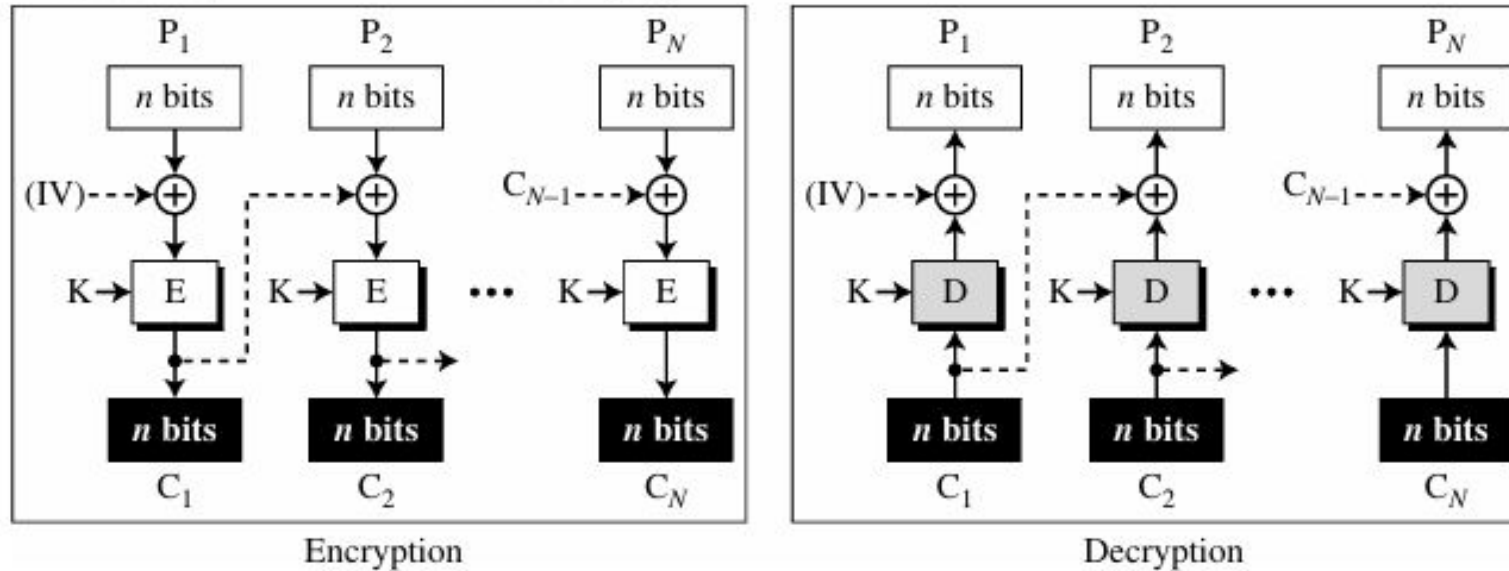
D : Decryption

$P_i$ : Plaintext block  $i$

$C_i$ : Ciphertext block  $i$

K: Secret key

IV: Initial vector ( $C_0$ )



The relation between plaintext and ciphertext blocks is shown below:

**Encryption:**

$$C_0 = IV$$

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

**Decryption:**

$$C_0 = IV$$

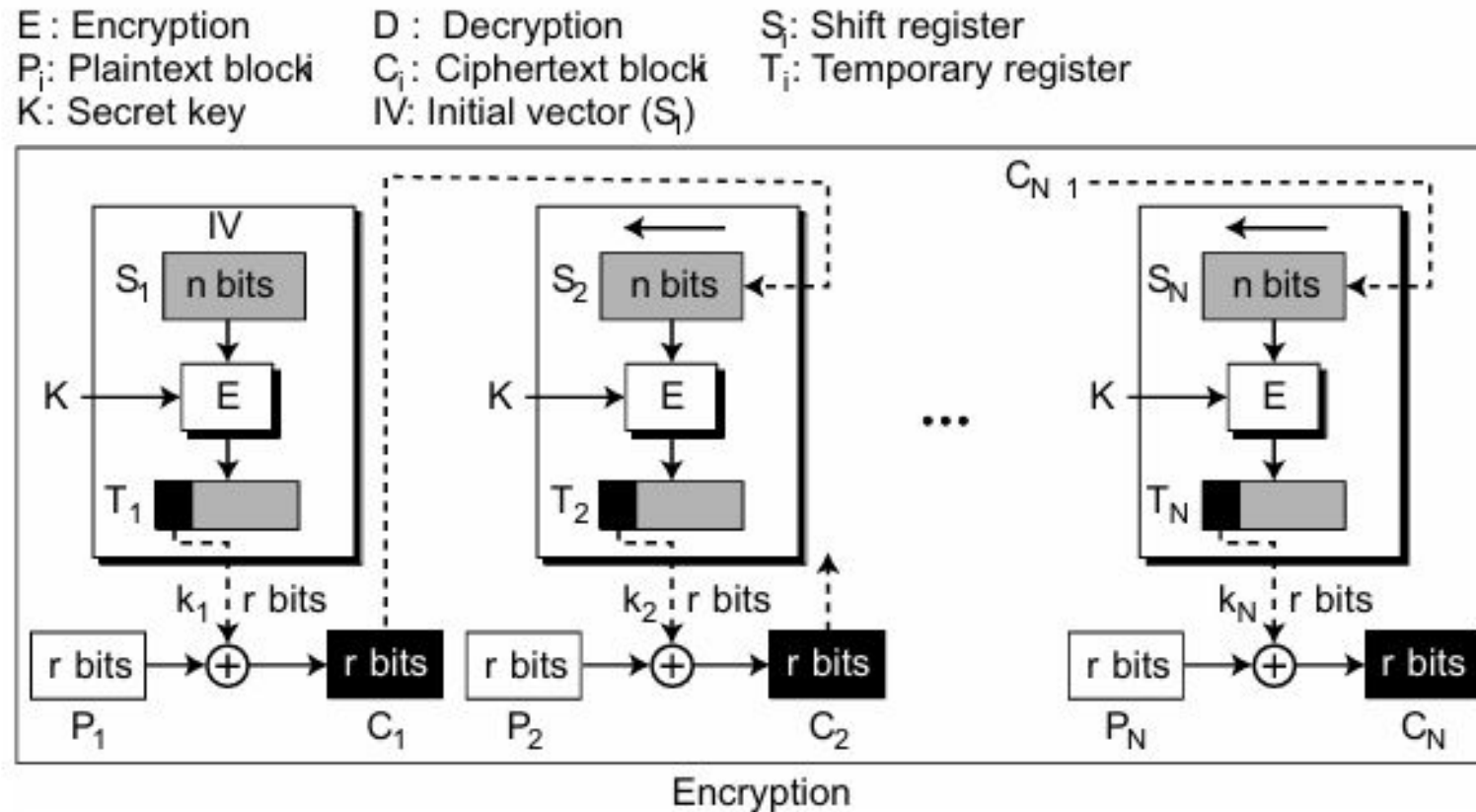
$$P_i = D_K (C_i) \oplus C_{i-1}$$

# Cipher Feedback (CFB) Mode

---

- ECB and CBC modes encrypt and decrypt blocks of the message.
- The block size,  $n$ , is predetermined by the underlying cipher;
- for example,  $n = 64$  for DES and  $n = 128$  for AES.
- In some situations, we need to use DES or AES as secure ciphers, but the plain text or ciphertext block sizes are to be smaller.
- For example, to encrypt and decrypt ASCII 8-bit characters, you would not want to use one of the traditional ciphers
- The solution is to use DES or AES in cipher feedback (CFB) mode.
- In this mode the size of the block used in DES or AES is  $n$ , but the size of the plaintext or ciphertext block is  $r$ , where  $r \leq n$

Figure 8.4 Encryption in cipher feedback (CFB) mode

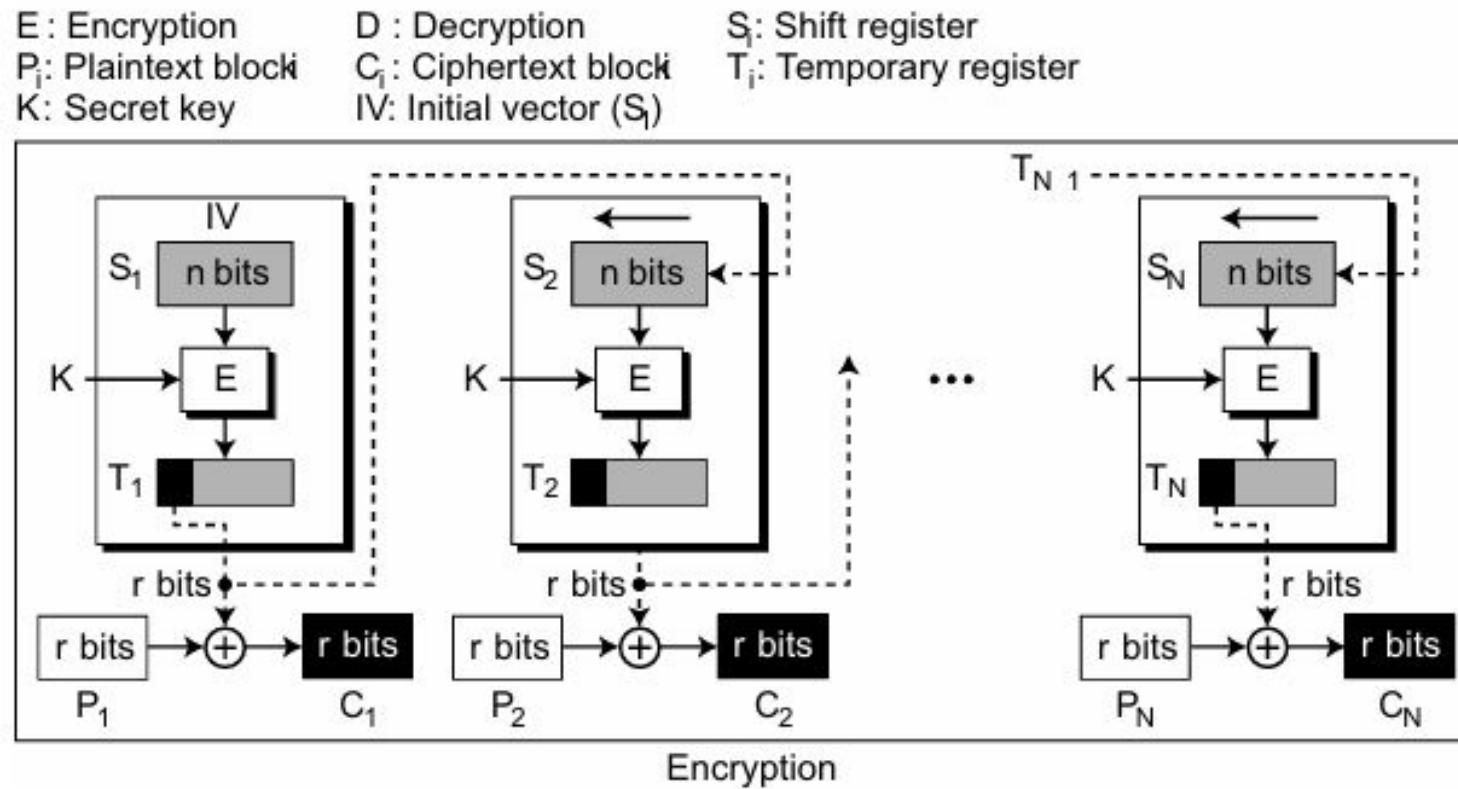


# Output Feedback (OFB) Mode

---

- Output feedback (OFB) mode is very similar to CFB mode, with one difference:
- each bit in the ciphertext is independent of the previous bit or bits. This avoids error propagation.
- If an error occurs in transmission, it does not affect the bits that follow.
- Note that, like CFB, both the sender and the receiver use the encryption algorithm

Figure 8.6 Encryption in output feedback (OFB) mode

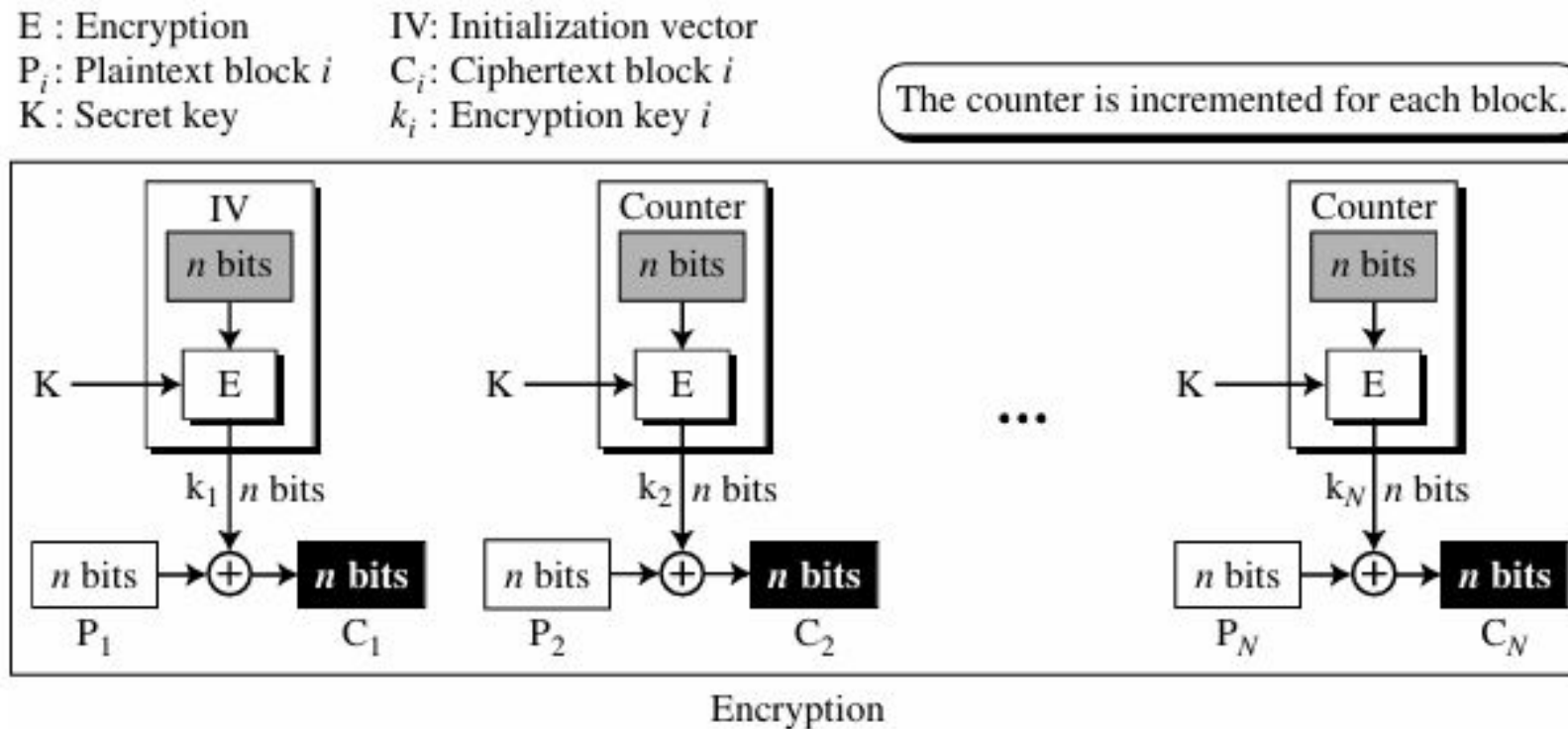


# Counter (CTR) Mode

---

- In the counter (CTR) mode, there is no feedback.
- The pseudo randomness in the key stream is achieved using a counter.
- An n-bit counter is initialized to a pre-determined value (IV) and incremented based on a predefined rule ( $\text{mod } 2^n$ )
- The plaintext and ciphertext block have the same block size as the underlying cipher (e.g., DEA or AES).
- Plaintext blocks of size n are encrypted to create ciphertext blocks of size n

**Figure 8.8** *Encryption in counter (CTR) mode*



# USE OF STREAM CIPHERS

---

- Although the five modes of operations enable the use of block ciphers for encipherment of messages or files in large units (ECB, CBC, and CTR) and small units (CFB and OFB),
- sometimes pure stream are needed for enciphering small units of data such as characters or bits
- Stream ciphers are more efficient for real-time processing.

# RC4

---

- RC4 is a stream cipher that was designed in 1984 by Ronald Rivest for RSA Data Security.
- RC4 is used in many data communication and networking protocols, including SSL/TLS .
- RC4 is a byte-oriented stream cipher in which a byte (8 bits) of a plaintext is exclusive-ored with a byte of key to produce a byte of a ciphertext.
- The secret key, from which the one-byte keys in the key stream are generated, can contain anywhere from 1 to 256 bytes

# State

---

- RC4 is based on the concept of a state.
- At each moment, a state of 256 bytes is active, from which one of the bytes is randomly selected to serve as the key for encryption.
- The idea can be shown as an array of bytes:

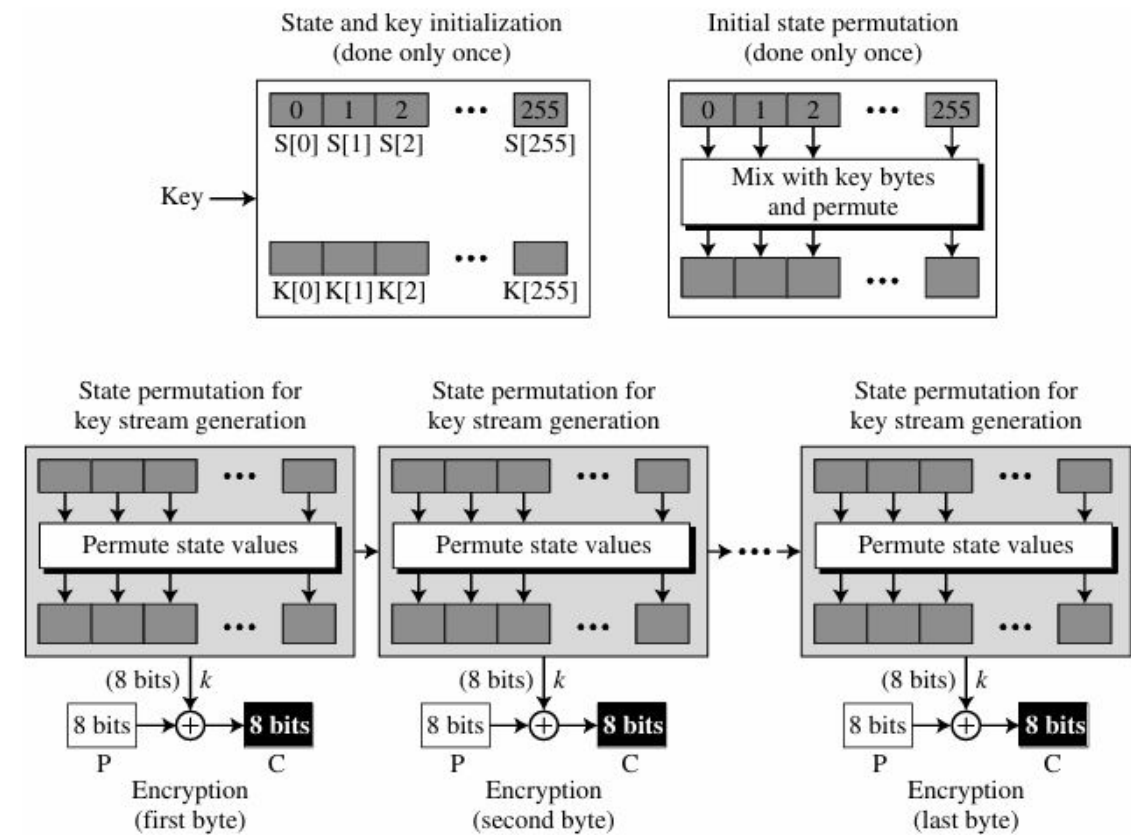
S[0] S[1] S[2] ... S[255]

- Note that the indices of the elements range between 0 and 255.
- The contents of each element is also a byte (8 bits) that can be interpreted as an integer between 0 to 255.

# The Idea

- Figure 8.10 shows the whole idea of RC4.
- The first two boxes are performed only once (initializing);
- the permutation for creating stream key is repeated as long as there are plaintext bytes to encrypt

**Figure 8.10** The idea of RC4 stream cipher



# Initialization

---

Initialization is done in two steps:

- In the first step, the state is initialized to values 0, 1, ..., 255. A key array,  $K[0]$ ,  $K[1]$ , ...,  $K[255]$  is also created.
- If the secret key has exactly 256 bytes, the bytes are copied to the  $K$  array;
- otherwise, the bytes are repeated until the  $K$  array is filled.

```
for ( $i = 0$  to 255)
{
     $S[i] \leftarrow i$ 
     $K[i] \leftarrow \text{Key}[i \bmod \text{KeyLength}]$ 
}
```

- In the second step, the initialized state goes through a permutation (swapping the elements) based on the value of the bytes in  $K[i]$ .
- The key byte is used only in this step to define which elements are to be swapped.
- After this step, the state bytes are completely shuffled

```
j ← 0
for (i = 0 to 255)
{
    j ← (j + S[i] + K[i]) mod 256
    swap (S[i] , S[j])
}
```

# Key Stream Generation

- The keys in the key stream, the  $k$ 's, are generated, one by one.
- First, the state is permuted based on the values of state elements and the values of two individual variables,  $i$  and  $j$ .
- Second, the values of two state elements in positions  $i$  and  $j$  are used to define the index of the state element that serves as  $k$ .
- The following code is repeated for each byte of the plaintext to create a new key element in the key stream.
- The variables  $i$  and  $j$  are initialized to 0 before the first iteration, but the values are copied from one iteration to the next

```
 $i \leftarrow (i + 1) \bmod 256$   
 $j \leftarrow (j + S[i]) \bmod 256$   
swap ( $S[i]$ ,  $S[j]$ )  
 $k \leftarrow S[(S[i] + S[j]) \bmod 256]$ 
```

# Security Issues

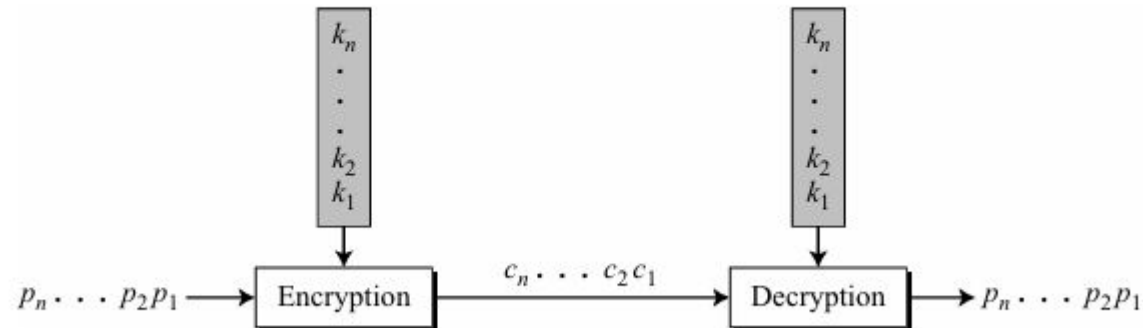
---

- It is believed that the cipher is secure if the key size is at least 128 bits (16 bytes).
- There are some reported attacks for smaller key sizes (less than 5 bytes),
- but the protocols that use RC4 today all use key sizes that make RC4 secure
- However, like many other ciphers, it is recommended the different keys be used for different sessions. This prevents Eve from using differential cryptanalysis on the cipher.

# One-Time Pad

- Stream ciphers are faster than block ciphers.
- The hardware implementation of a stream cipher is also easier.
- When we need to encrypt binary streams and transmit them at a constant rate, a stream cipher
- Stream ciphers are also more immune to the corruption of bits during transmission. is the better choice to use

**Figure 5.21** *Stream cipher*



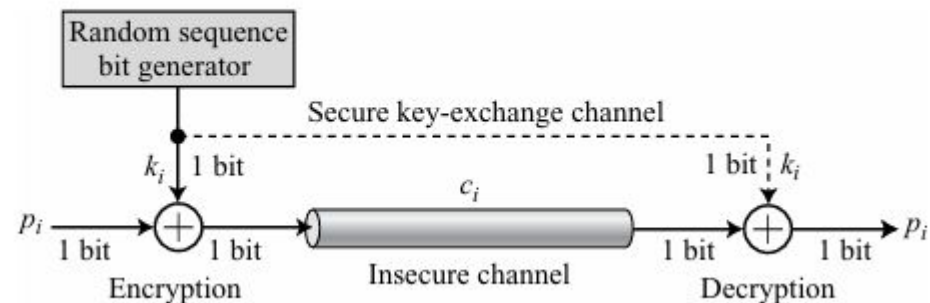
- Looking at Figure 5.21, one can suggest that the main issue in modern stream ciphers is how to generate the key stream  $K = k_n \dots k_2 k_1$ .
- Modern stream ciphers are divided into two broad categories: synchronous and nonsynchronous
- In a synchronous stream cipher, the key stream is independent of the plaintext or ciphertext stream.
  - The key stream is generated and used with no relationship between key bits and the plaintext or ciphertext bits

## One-Time Pad

The simplest and the most secure type of synchronous stream cipher is called the one time pad, which was invented and patented by Gilbert Vernam.

- A one-time pad cipher uses a key stream that is randomly chosen for each encipherment.
- The encryption and decryption algorithms each use a single exclusive-or operation.
- Based on properties of the exclusive-or operation discussed earlier, the encryption and decryption algorithms are inverses of each other.
- It is important to note that in this cipher the exclusive-or operation is used one bit at a time.

**Figure 5.22** *One-time pad*



- 
- The one-time pad is an ideal cipher. It is perfect.
  - There is no way that an adversary can guess the key or the plaintext and ciphertext statistics.
  - There is no relationship between the plaintext and ciphertext, either
  - In other words, the ciphertext is a true random stream of bits even if the plaintext contains some patterns.
  - Eve cannot break the cipher unless she tries all possible random key streams, which would be  $2^n$  if the size of the plaintext is  $n$  bits.
  - Anyway this perfect and ideal cipher is very difficult to achieve.



# THANK YOU

---

**Archana M**

Department of Computer Applications

**[archanam@pes.edu](mailto:archanam@pes.edu)**