SYMMETRIC CRYPTOGRAPHIC ALGORITHMS

SIMPLIFIED DATA ENCRYPTION STANDARD (S-DES):

The overall structure of the simplified DES

The S-DES encryption algorithm takes an 8-bit of plaintext (example: 01110010) and a 10-bit key as input and produces an 8-bit of ciphertext as output. The S-DES decryption algorithm takes an 8-bit of ciphertext and the same 10-bit key used to produce that ciphertext as input and produces the original 8-bit block of plaintext.

The encryption algorithm involves five functions:

- An initial permutation (IP)
- First round function labeled fk, which involves both permutation and substitution operations and depends on a key input
- A simple permutation function that switches/swap (SW) the two halves of the data
- Second round of function fk
- A permutation function that is the inverse of the initial permutation

The decryption algorithm is the inverse of the encryption algorithm

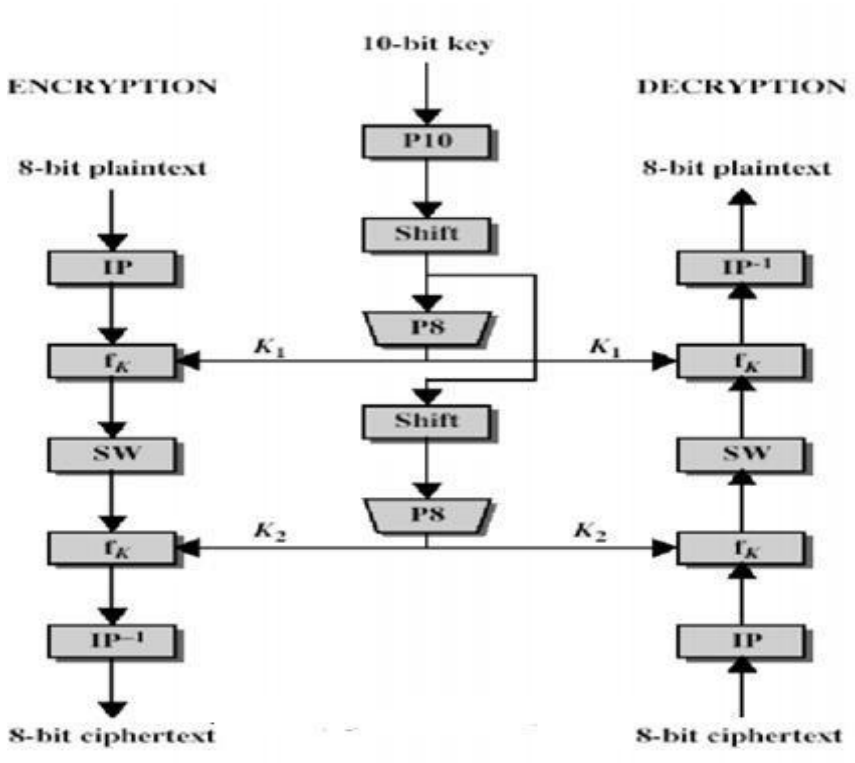


Fig. S-DES algorithm

We can express the encryption algorithm as

$$\textbf{CIPHER TEXT}=IP^{-1}(f_{K2}(SW(f_{K1}(IP(\text{plaintext})))))$$

Where

$$K_1=P8(\text{Shift}(P10(\text{Key})))$$

$$K_2=P8(\text{Shift}(\text{Shift}(P10(\text{Key}))))$$

Decryption algorithm is expressed as

$$\textbf{CIPHER TEXT}=IP^{-1}(f_{K1}(SW(f_{K2}(IP(\text{plaintext})))))$$

S-DES Key Generation

Key generation in S-DES takes 10 bits of initial key as an input and produces two 8 bit sub keys K1 and K2 which are required for two rounds of function fk.

The key generation involves following functions:

- It takes 10-bit key as input
- An permutation (P10) on 10 bit key
- Circular left shift by one bit(LS-1)
- An permutation (P8) which produces 8 bit sub key K1 from initial 10 bit key
- Circular left shift by 2 bits(LS-2) on the output of LS-1
- An permutation (P8) which produces 8 bit sub key K2

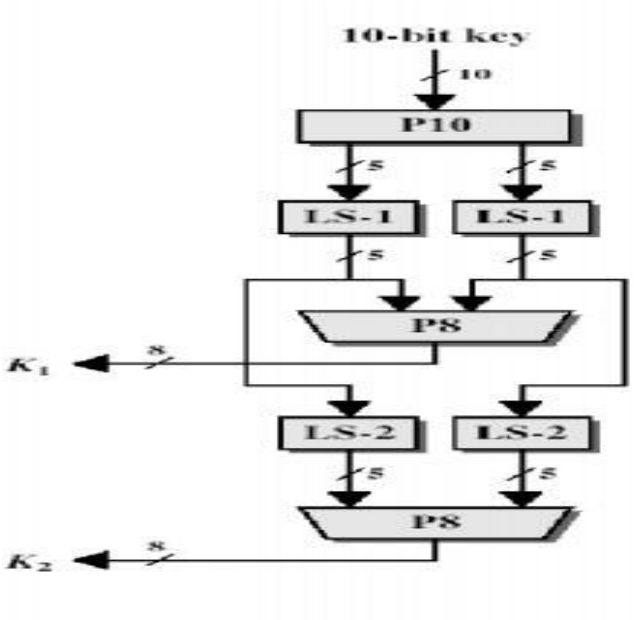


Fig – S-DES key generation

Let us produce two 8 bit sub keys by using a 10 bit key: **1010000010**

1 0 1 0 0 0 0 0 1 0

Let the 10 bit key be designated as (K_1 K_2 K_3 K_4 K_5 K_6 K_7 K_8 K_9 K_{10}) then permutation P10 is defined as

P10 (K_1 K_2 K_3 K_4 K_5 K_6 K_7 K_8 K_9 K_{10}) = (K_3 K_5 K_2 K_7 K_4 K_{10} K_1 K_9 K_8 K_6)

(Permutation is rearranging of bit positions)

P10= 3 5 2 7 4 10 1 9 8 6

So the 1st output bit is bit of 3 of the input, 2nd output bit is bit of 5of the input and so on. Above 10 bit key is permuted (P10) to

P10 = 1 0 0 0 0 0 1 1 0 0

Next step is to perform a circular left shift/rotation by one bit (LS-1) separately on the 1st five bits and second five bits

P10 = 1 0 0 0 0 0 1 1 0 0

1st 5 bits **2nd 5 bits**

LS-1 **LS -1**

0 0 0 0 1 1 1 0 0 0

After circular left shift we will get output as LS-1 = 0 0 0 0 1 1 1 0 0 0

1 2 3 4 5 6 7 8 9 10 (Bit positions)

Next we apply P8 which picks out and permute 8 bits out of 10 bits according to the following rule

P8= 6 3 7 4 8 5 10 9

So the 1st output bit is bit of 6 of the LS-1, 2nd output bit is bit of 3 of the LS-1 and so on

P8= 1 0 1 0 0 1 0 0----------K1

After P8 permutation we will get **1st 8 bit sub key K1 which is 1 0 1 0 0 1 0 0**

Then to produce second sub key K2 let us go back to the pair of 5 bit strings produced by the two LS-1 function and perform a circular left shift of 2 bit positions on each string.

LS- 1= 0 0 0 0 1 1 1 0 0 0

LS-2 LS-2 (left shift by 2 bits)

0 0 1 0 0 0 0 0 1 1

Finally P8 is applied again on the output of LS_2 to produce K2

LS_2 = 0 0 1 0 0 0 0 0 1 1

P8= 6 3 7 4 8 5 10 9 (standard)

Final output is P8= 0 1 0 0 0 0 1 1---------- K2

2nd 8 bit sub key K2 is 0 1 0 0 0 0 1 1

S-DES Encryption:

Fig below shows the S-DES encryption algorithm which involves the sequential application of five functions.

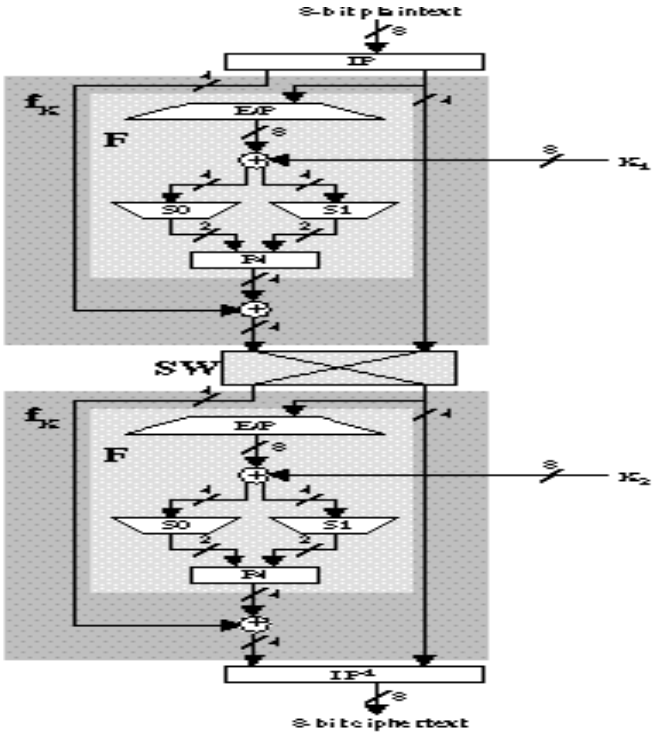


Figure 3.3 Simplified DES Encryption Detail

1st step: Initial permutations (IP)

The input to the algorithm is an 8 bit block of plaintext which we 1st permute using the IP function:

IP= 2 6 3 1 4 8 5 7

Description	Marks

Let plaintext = 0 1 1 1 0 0 1 0 (1 2 3 4 5 6 7 8)

Ip = 2 6 3 1 4 8 5 7

Ip = 1 0 1 0 1 0 0 1

Now Split Ip into two 4 bit halves

Ip = 1 0 1 0 (L - 4 bit) 1 0 0 1 (R 4 bit) → E/P

Take right 4 bit and perform Expansion and permutation

E/P = 4 1 2 3 2 3 4 1 (1 2 3 4)

o/p = 1 0 0 1

E/P = 1 1 0 0 0 0 1 1 - 8bit

perfor Xor Operation of E/P with 8 bit Sub Key K_1 which is already generated

8 bit Sub key K_1 from previous section

$K_1 = 10100100$

$E/p \oplus K_1$

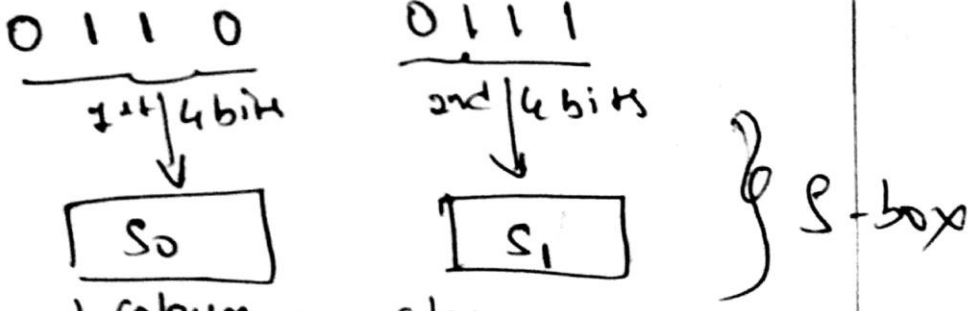
$E/p = 11000011$

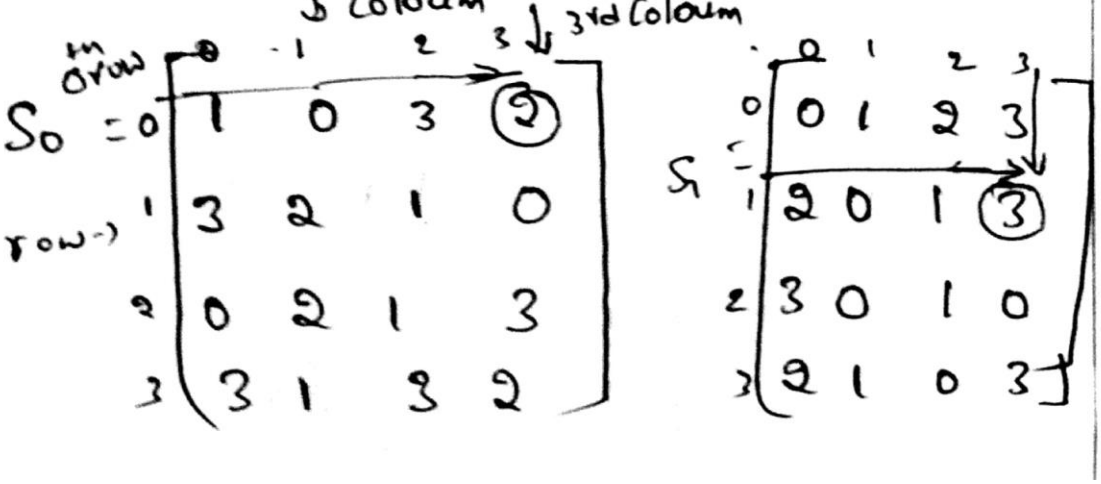
$\oplus$

$K_1 = 10100100$

01100111

Split Xor output into two halves





Q no.	Description	Marks

The S box operates as follows.

- The 1st & Fourth (4th) input bits are treated as 2 bit number that specify a row of the S-box.
- 2nd & 3rd input bits specify a column of the S-box.

1 & 4th bit – row
2 & 3rd bit – column of sbox

Now input for S_0 is $\overset{1}{0}\ \overset{2}{1}\ \overset{3}{1}\ \overset{4}{0}$

1st & 4th bits are : 0 0 = 0 – row

2nd & 3rd bits are = 1 1 = 3 – column

in an Sbox (S_0) 0th row & 3rd column value is $2 = 10_{=}$

So S_0 o/p is $= 10_{=}$ $S_0 = 10_{=}$

Similarly for S_1 input is 0111

- 1st & 4th bits are 01 = 1 – row
  2nd & 3rd bits are 11 = 3 – column

in an Sbox S_1 1st row & 3rd column value is $3 = 11_{=}$

So $S_1 = 11$

Now after Sbox we will get 2 bits from S_0 & 2 bit from S_1.

Then 4 bit Sbox output is given to permutation P_4

$P_4 = 2\ 4\ 3\ 1$

input is = $\overset{1}{1}\ \overset{2}{0}\ \overset{3}{1}\ \overset{4}{1}$

S_0 S_1

$P_4 = 0111$

finally output of P_4 is Xored with initial left 4 bits.

$P_4 = 0\ 1\ 1\ 1$

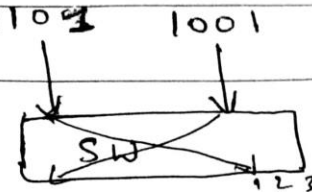
(+)

L-4bits = 1 0 1 0

1 1 0 1 :-

The Switch function interchanges the left and right 4 bits so that second instance round of fk operates on a different 4 bits.

Q. No.	Description	Marks Distribution



E/P = 4 1 2 3 2 3 4 1

= 1 1 1 0 . 1 0 1 1

Now output of E/P is xored with second sub key K_2

E/P = 1110 1011

⊕

K_2 = 0100 0011

1010 1000

1 & 4th bit - row
2nd & 3rd bit - Column

S_0 S_1

S_0 =

	0	1	2	3
0	1	0	3	2
1	3	2	1	0
2	0	(2)	1	3
3	3	1	3	2

S_1 =

	0	1	2	3
0	0	1	2	3
1	2	0	1	3
2	(3)	0	1	0
3	2	1	0	3

S_0 = 10 S_1 = 11

$S = 1011$ (1 2 3 4)

$P_4 = 2\ 4\ 3\ 1$

$= 0\ 1\ 1\ 1$

$_2$

Description	Marks

P_4 ⊕ with L4 bits

P_4 = 0 1 1 1

⊕

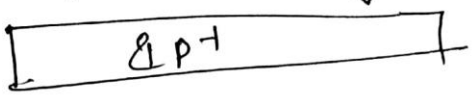
L-4bit = 1 0 0 1

(round 2) 1 1 1 0

1 2 3 4

1 0 1

5 6 7 8 R-4bit of (round 2



$8p^{-1}$ = 4 1 3 5 7 2 8 6

$4p^{-1}$ = 0 1 1 1 0 1 1 1

8 bit Cipher text is 0111 0111

Feistel cipher structure

- The input to the encryption algorithm are a plaintext block of length 2w bits and a key K. the plaintext block is divided into two halves L0 and R0. The two halves of the data pass through„n" rounds of processing and then combine to produce the ciphertext block.
- Each round „i" has inputs L_{i-1} and R_{i-1}, derived from the previous round, as well as the subkey K_i, derived from the overall key K. in general, the subkeys K_i are different from K and from each other.
- All rounds have the same structure. A substitution is performed on the left half of the data (as similar to S-DES). This is done by applying a round function F to the right half of the data and then taking the XOR of the output of that function and the left half of the data.
- The round function has the same general structure for each round but is parameterized by the round sub key k_i. Following this substitution, a permutation is performed that consists of the interchange of the two halves of the data. This structure is a particular form of the substitution-permutation network.
- The exact realization of a Feistel network depends on the choice of the following parameters and design

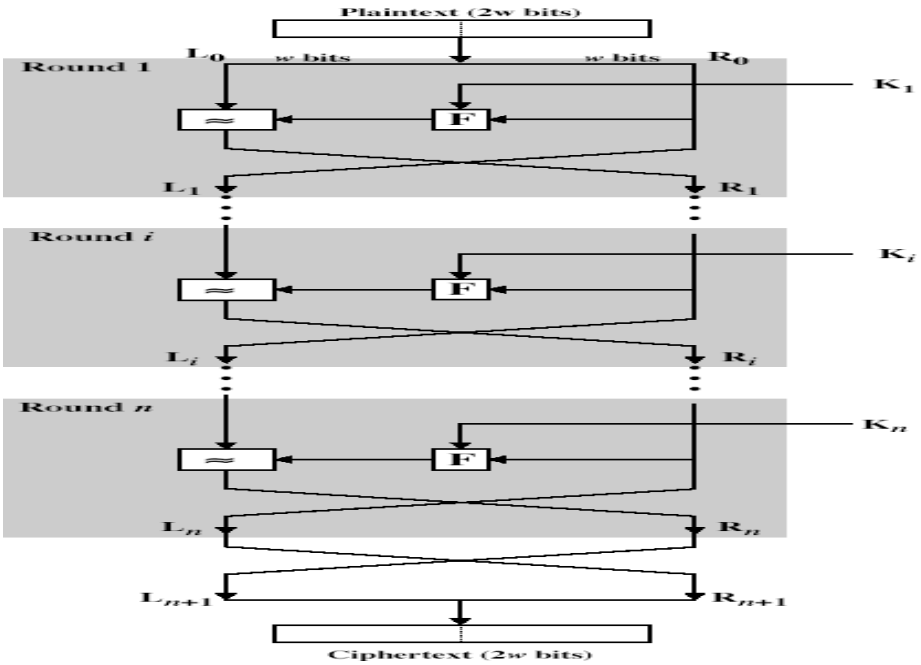


Fig: Classical Feistel Network

Features:

- **Block size** - Increasing size improves security, but slows cipher
- **Key size** - Increasing size improves security, makes exhaustive key searching harder, but may slow cipher
- **Number of rounds** - Increasing number improves security, but slows cipher
- **Subkey generation** - Greater complexity can make analysis harder, but slows cipher
- **Round function** - Greater complexity can make analysis harder, but slows cipher
- **Fast software en/decryption & ease of analysis** - are more recent concerns for practical use and testing

Feistel encryption and decryption:

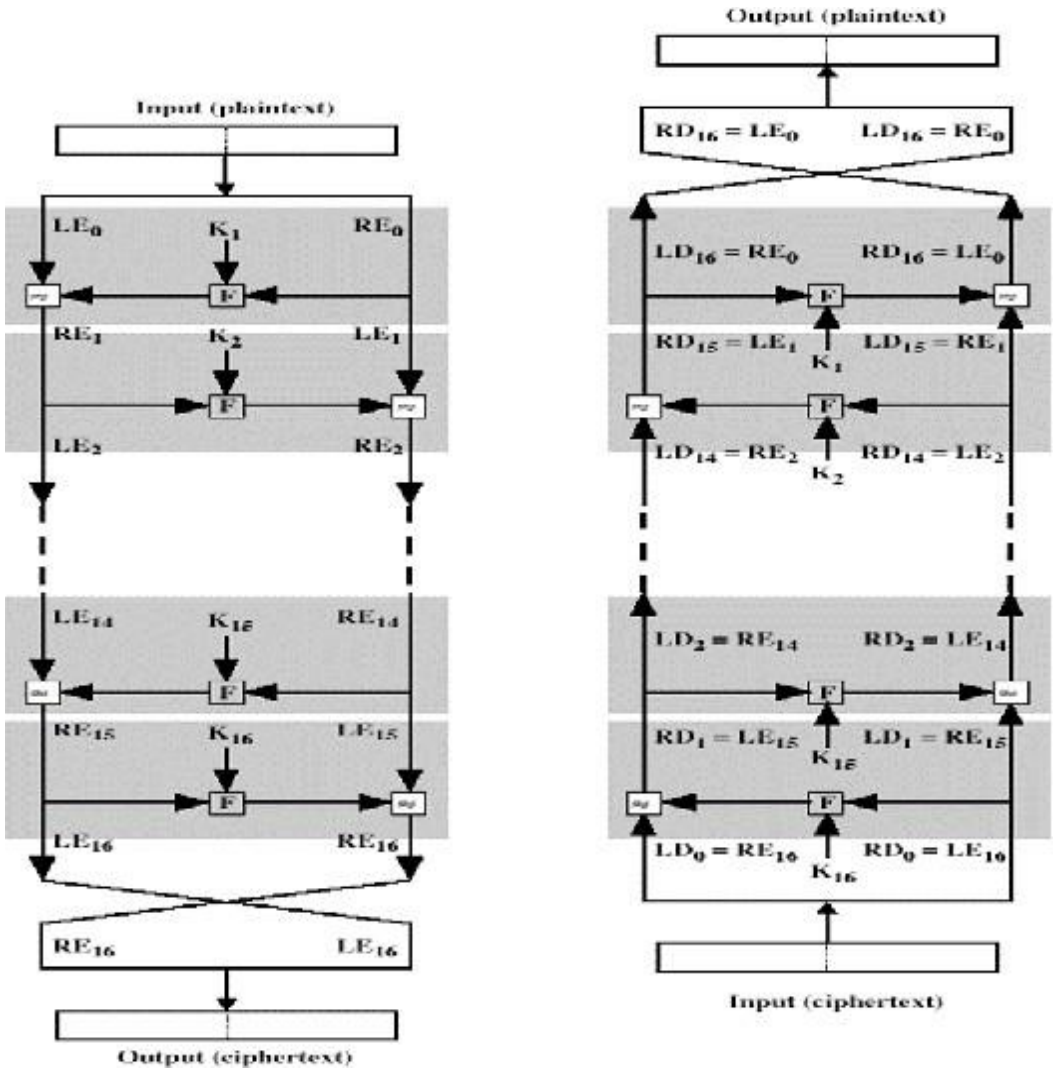


Fig: Feistel encryption and decryption

- The process of decryption is essentially the same as the encryption process. The rule is as follows: use the cipher text as input to the algorithm, but use the subkey k_i in reverse order. i.e., k_n in the first round, k_{n-1} in second round and so on.
- We use the notation LE_i and RE_i for data traveling through the decryption algorithm. The diagram above indicates that, at each round, the intermediate value of the decryption process is same (equal) to the corresponding value of the encryption process with two halves of the value swapped.

$$\text{i.e., } RE_i \,||\, LE_i \text{ (or) equivalently } RD_{16-i} \,||\, LD_{16-i}$$

- After the last iteration of the encryption process, the two halves of the output are swapped, so that the cipher text is $RE_{16} \,||\, LE_{16}$.
- The output of that round is the cipher text. Now take the cipher text and use it as input to the same algorithm. The input to the first round is $RE_{16} || LE_{16}$, which is equal to the 32-bit swap of the output of the sixteenth round of the encryption process.
- Now we will see how the output of the first round of the decryption process is equal to a 32-bit swap of the input to the sixteenth round of the encryption process. First consider the encryption process,

$$LE_{16} = RE_{15}$$

$$RE_{16} = LE_{15} \oplus F(RE_{15}, K_{16}) \text{ on the decryption side,}$$

$$LD_1 = RD_0 = LE_{16} = RE_{15}$$

$$RD_1 = LD_0 \oplus F(RD_0, K_{16})$$

$$= RE_{16} \oplus F(RE_{15}, K_{16})$$

$$= [LE_{15} \oplus F(RE_{15}, K_{16})] \oplus F(RE_{15}, K_{16})$$

$$RD_1 = LE_{15}$$

Therefore, $LD_1 = RE_{15}$ and $RD_1 = LE_{15}$

In general, for the i^{th} iteration of the encryption algorithm,

$$LE_i = RE_{i-1}$$
$$RE_i = LE_{i-1} \oplus F(RE_{i-1}, K_i)$$

Finally, the output of the last round of the decryption process is $RE_0 \,||\, LE_0$. A 32-bit swap recovers the original plaintext.

BLOCK CIPHER PRINCIPLES

Virtually, all symmetric block encryption algorithms in current use are based on a structure referred to as Fiestel block cipher. For that reason, it is important to examine the design principles of the Fiestel cipher.

We begin with a **comparison of stream cipher with block cipher.**

- **A stream cipher** is one that encrypts a digital data stream one bit or one byte at a time. E.g, vigenere cipher.
- **A block cipher** is one in which a block of plaintext is treated as a whole and used to produce a cipher text block of equal length. Typically a block size of 64 or 128 bits is used.

DATA ENCRYPTION STANDARD (DES)

In May 1973, and again in Aug 1974 the NBS (now NIST) called for possible encryption algorithms for use in unclassified government applications and the response was mostly disappointing, however IBM submitted their Lucifer design following a period of redesign and comment it became the Data Encryption Standard (DES)

It was adopted as a (US) federal standard in Nov 76, published by NBS as a hardware only scheme in Jan 77 and by ANSI for both hardware and software standards in ANSI X3.92-1981 (also X3.106-1983 modes of use) subsequently it has been widely adopted and is now published in many standards around the world of Australian Standard AS2805.5-1985

One of the largest users of the DES is the banking industry

- Rapid advances in computing speed though have rendered the 56 bit key susceptible to exhaustive key search, as predicted by Diffie & Hellman
- The DES has also been theoretically broken using a method called Differential Cryptanalysis, however in practice this is unlikely to be a problem

Overview of the DES Encryption Algorithm

- The basic process in enciphering a 64-bit data block using the DES consists of:
- Initial permutation (IP)
- 16 rounds of a complex key dependent calculation F(round function)

- A final permutation, being the inverse of IP

In more detail the 16 rounds of F can be described functionally as

$$L(i) = R(i-1)$$

$$R(i) = L(i-1) (+) P(S(E(R(i-1)) (+) K(i)))$$

The sub keys used by the 16 rounds are formed by the **key schedule** which consists of an initial permutation of the key (PC1) which selects 56-bits in two 28-bit halves and each half is circular left shifted either 1 or 2 places depending on the **key rotation schedule** KS. Then after shift operation output is again permuting by PC2 for use in function f, this can be described functionally as:

$$K(i) = PC2(KS(PC1(K), i))$$

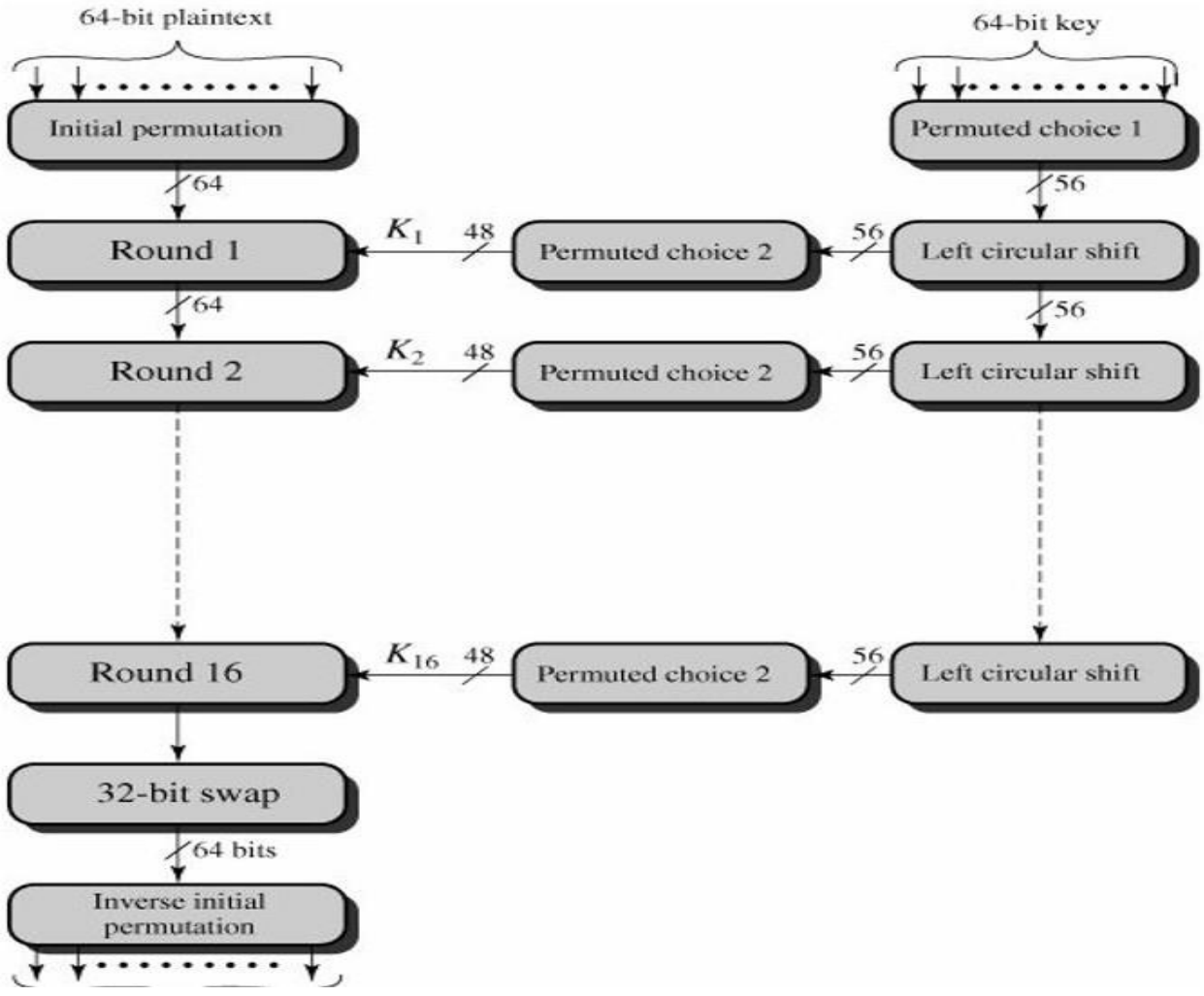
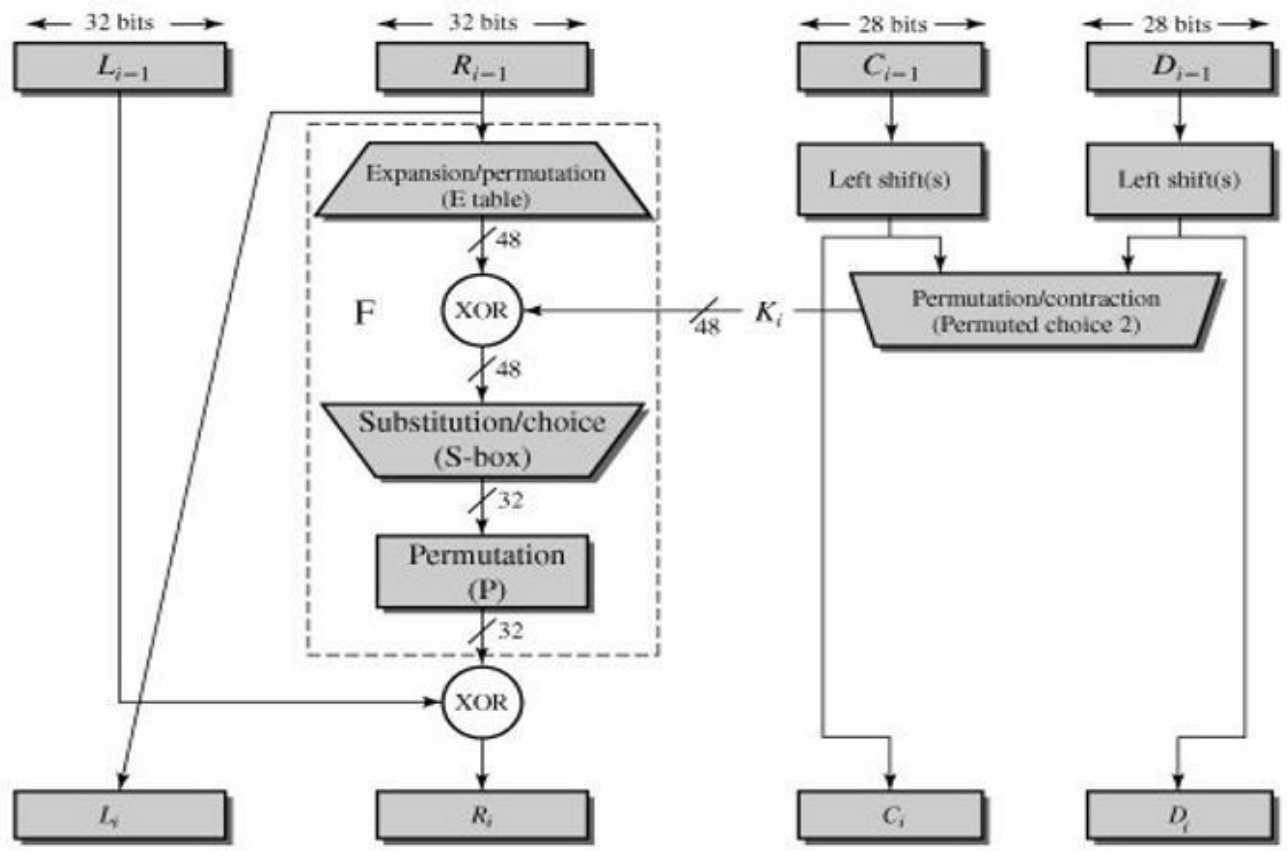


Fig .General depiction of DES encryption algorithm

Single Round of DES Algorithm:

Fig 2 below shows the internal structure of a single round DES algorithm. In 1st step 64 bit input is spilt into the left and right halves as separate 32- bit quantities, labeled L (left) and R (right).



- The round key Ki is 48 bits and round input is 32 bits
- After initial permutation IP the 64 bit output is split into the two equal halves that are right 32 bits and left 32 bits.
- This Right input is first expanded to 48 bits by using a table that defines a permutation plus an expansion.
- The resulting 48 bits are XORed with Ki (sub key of size 48 bits)
- This 48-bit result passes through a substitution function (S-BOX) that produces a 32-bit output,
- The output of S-BOX is permuted again
- 32 bit permuted output is finally XORed with Left 32 bits.
- Now the XORed output and right 32 bits acts as an input for the next round.

1. INITIAL PERMUTATION

The initial permutation and its inverse are defined by tables, as shown in Tables 1a and 1b, respectively. The tables are to be interpreted as follows. The input to a table consists of 64 bits numbered 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.

(a) Initial Permutation (IP)							
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 .1a

Inverse of initial permutation

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.1b

2. The permuted input block split into two halves each is 32 bits. The first 32 bits are called L and the last 32 bits are called R.
Now, The F function will start the rest of all the steps.

3. Expansion Permutation:

It will expand a right 32 bits given as input to 48 bits by using a table that defines a permutation plus an expansion.

Expansion Permutation (E)							
	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	

4. Preform Exclusive-OR between the sub-key and Expansion Permutation (E) on R.
5. The result is passed through a substitution function and produce 32 bits output.

6. SUBSTITUTION FUNCTION

- The role of the S-boxes in the function F is illustrated in Figure below.
- The substitution consists of a set of eight S-boxes, each of which accepts 6 bits as input and produces 4 bits as output.
- These transformations are defined in Table below, which is interpreted as follows:
- The first and last bits of the input to box Si form a 2-bit binary number to select rows in the table for Si.
- The middle four bits select one of the sixteen columns.
- The decimal value in the cell selected by the row and column is then converted to its 4-bit representation to produce the output.
- For example, in S1 for input 011001, the row is 01 (row 1) and the column is 1100 (column 12). The value in row 1, column 12 is 9, so the output is 1001.

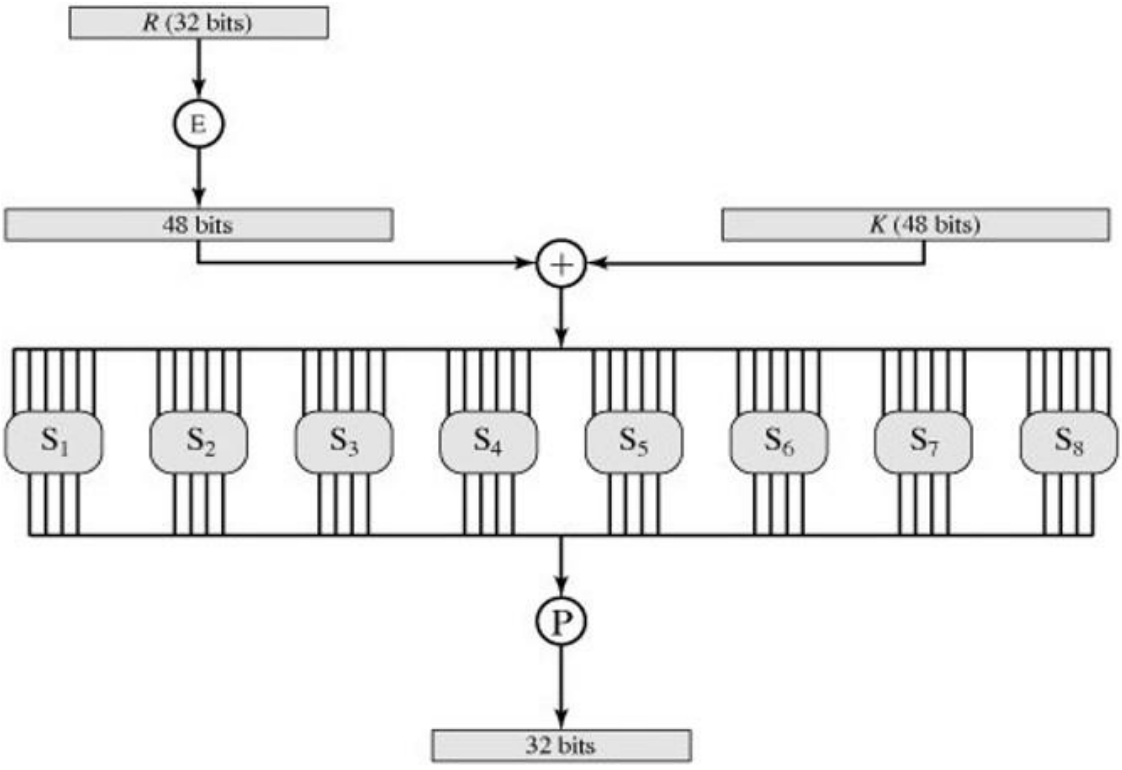


Table . Definition of DES S-Boxes

S_1

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

S_2

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

S_3

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

S_4

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

S_5

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

S_6

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

S_7

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

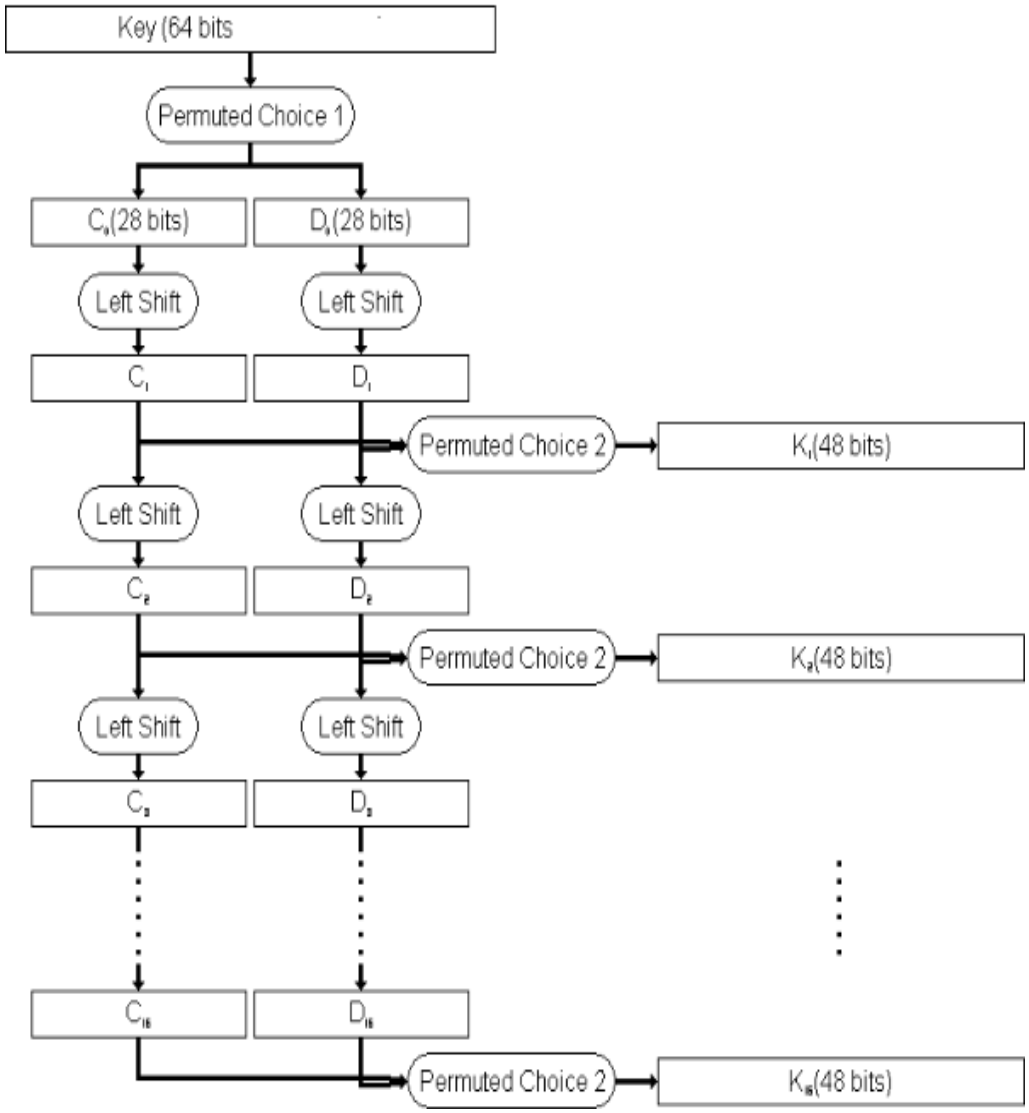
S_8

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

7. **Permutation Function (P):** The result of the substitution operation (output of S-Boxes) passes through a Permutation Function (P).

Permutation Function (P)							
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

Key Generation



- The key generator algorithm takes 64 bits key as input.
- The key (64 bits) is first subjected to a permutation governed by a table labeled Permuted Choice One. Every eighth bit is ignored the resulting 56-bit key.

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

- It is then treated as two 28-bit quantities, labeled C0 and D0.
- At each round, Ci-1 and Di-1 are separately subjected to a circular left shift, or rotation, of 1 or 2 bits, as governed by Table below. These shifted values serve as input to the next round.

Round number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

- They also serve as input to Permuted Choice Two, which produces a 48-bit output that serves as input to the round function F.

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

The Avalanche Effect

A desirable property of any encryption algorithm is that a small change in either the plaintext or the key should produce a significant change in the cipher text. In particular, a change in one bit of the plaintext or one bit of the key should produce a change in many bits of the cipher text it is known as avalanche effect. If the change were small, this might provide a way to reduce the size of the plaintext or key space to be searched

Avalanche Effect in DES

(a) Change in Plaintext		(b) Change in Key	
Round	Number of bits that differ	Round	Number of bits that differ
0	1	0	0
1	6	1	2
2	21	2	14
3	35	3	28
4	39	4	32
5	34	5	30
6	32	6	32
7	31	7	35
8	29	8	34
9	42	9	40

The Strength of DES:

Since its adoption as a federal standard, there have been lingering concerns about the level of security provided by DES. These concerns, by and large, fall into two areas:

1. Key size and
2. The nature of the algorithm.

The Use of 56-Bit Keys:

- With a key length of 56 bits, there are 2^{56} possible keys, which is approximately 7.2×10^{16}. Thus a brute-force attack appears impractical. Assuming that, on average, half the key space has to be searched, a single machine performing one DES encryption per microsecond would take more than a thousand years to break the cipher
- DES finally and definitively proved insecure in July 1998, when the Electronic Frontier Foundation (EFF) announced that it had broken a DES encryption using a special-purpose "DES cracker". The attack took less than three days.

The Nature of the DES Algorithm

Another concern is the possibility that cryptanalysis is possible by exploiting the characteristics of the DES algorithm. The focus of concern has been on the eight substitution tables, or S-boxes, that are used in each iteration. Because the design criteria for these boxes, and indeed for the entire algorithm, were not made public, there is a suspicion that the boxes were constructed in such a way that cryptanalysis is possible for an opponent who knows the weaknesses in the S-boxes. This assertion is tantalizing, and over the years a number of regularities and unexpected behaviors of the S-boxes have been discovered. Despite this, no one has so far succeeded in discovering the supposed fatal weaknesses in the S-boxes.

Block Cipher Design Principles:

- Number of Rounds
- Design of Function

  S-Box Design
- Key Schedule Algorithm

BLOCK CIPHER MODES OF OPERATION

- DES encrypts 64-bit blocks of data, using a 56-bit key
- We need some way of specifying how to use it in practice, given that we usually have an arbitrary amount of information to encrypt
- The way we use a block cipher is called its **Mode of operation** and five have been defined for the DES.

Block Modes

Splits messages in blocks (ECB, CBC)

1. Electronic Codebook Book (ECB)

- Where the message is broken into independent 64-bit blocks which are encrypted.

2. Cipher Block Chaining (CBC)

- Again the message is broken into 64-bit blocks, but they are linked together in the encryption operation with an IV.

Stream Modes

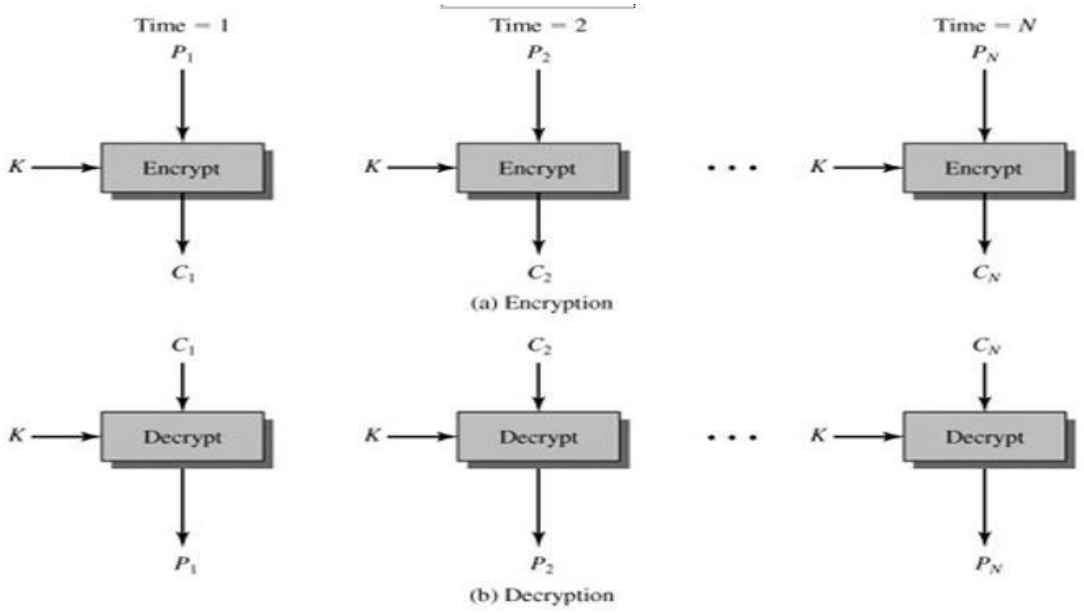
On bit stream messages (CFB, OFB)

3. **Cipher Feedback (CFB)**
   - Where the message is treated as a stream of bits, added to the output of the DES, with the result being feedback for the next stage
4. **Output Feedback (OFB)**
   - Where the message is treated as a stream of bits, added to the message, but with the feedback being independent of the message
5. **Counter mode (CTR):** It uses the counters
   - Each mode has its advantages and disadvantages

Electronic codebook (ECB)

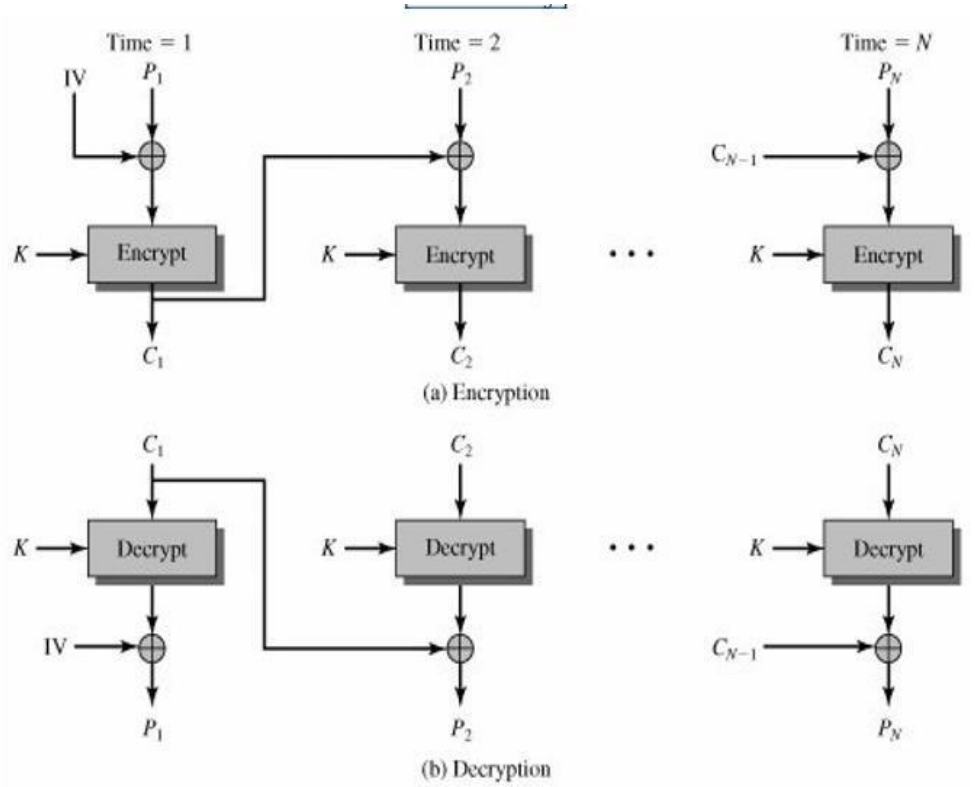
The simplest mode is the electronic codebook (ECB) mode; in which plaintext is handled one block at a time and each block of plaintext is encrypted using the same key. The term codebook is used because, for a given key, there is a unique cipher text for every b-bit block of plaintext. Therefore, we can imagine a gigantic codebook in which there is an entry for every possible b-bit plaintext pattern showing its corresponding cipher text.

The disadvantage of ECB is that the same b-bit block of plaintext, if it appears more than once in the message, always produces the same cipher text



Cipher Block Chaining (CBC)

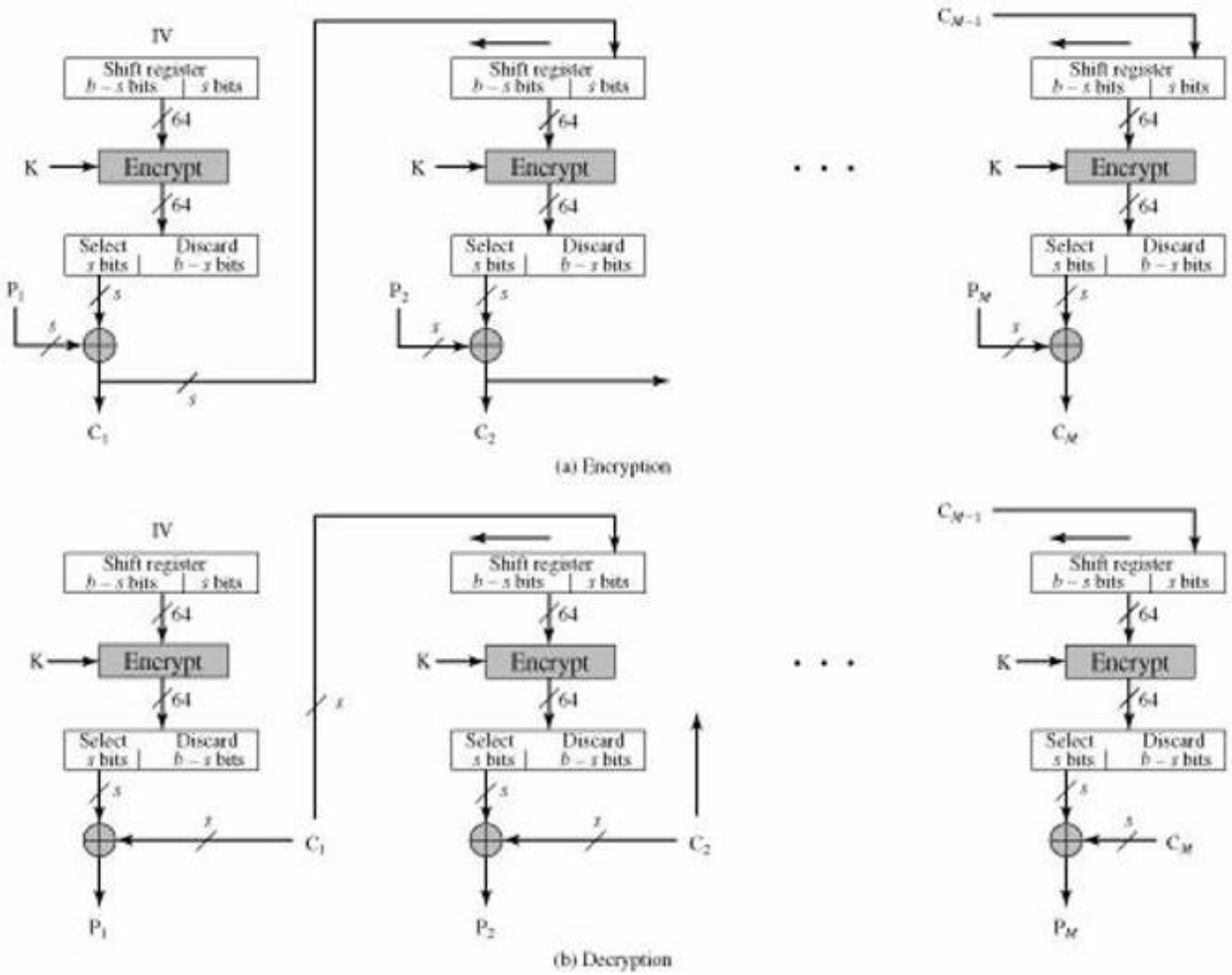
- To overcome the security deficiencies of ECB, we would like a technique in which the same plaintext block, if repeated, produces different cipher text blocks.
- A simple way to satisfy this requirement is the cipher block chaining (CBC) mode.
- In this scheme, the input to the encryption algorithm is the XOR of the current plaintext block and the preceding cipher text block.
- The same key is used for each block.
- In effect, we have chained together the processing of the sequence of plaintext blocks.



Cipher Feedback Mode (CFB)

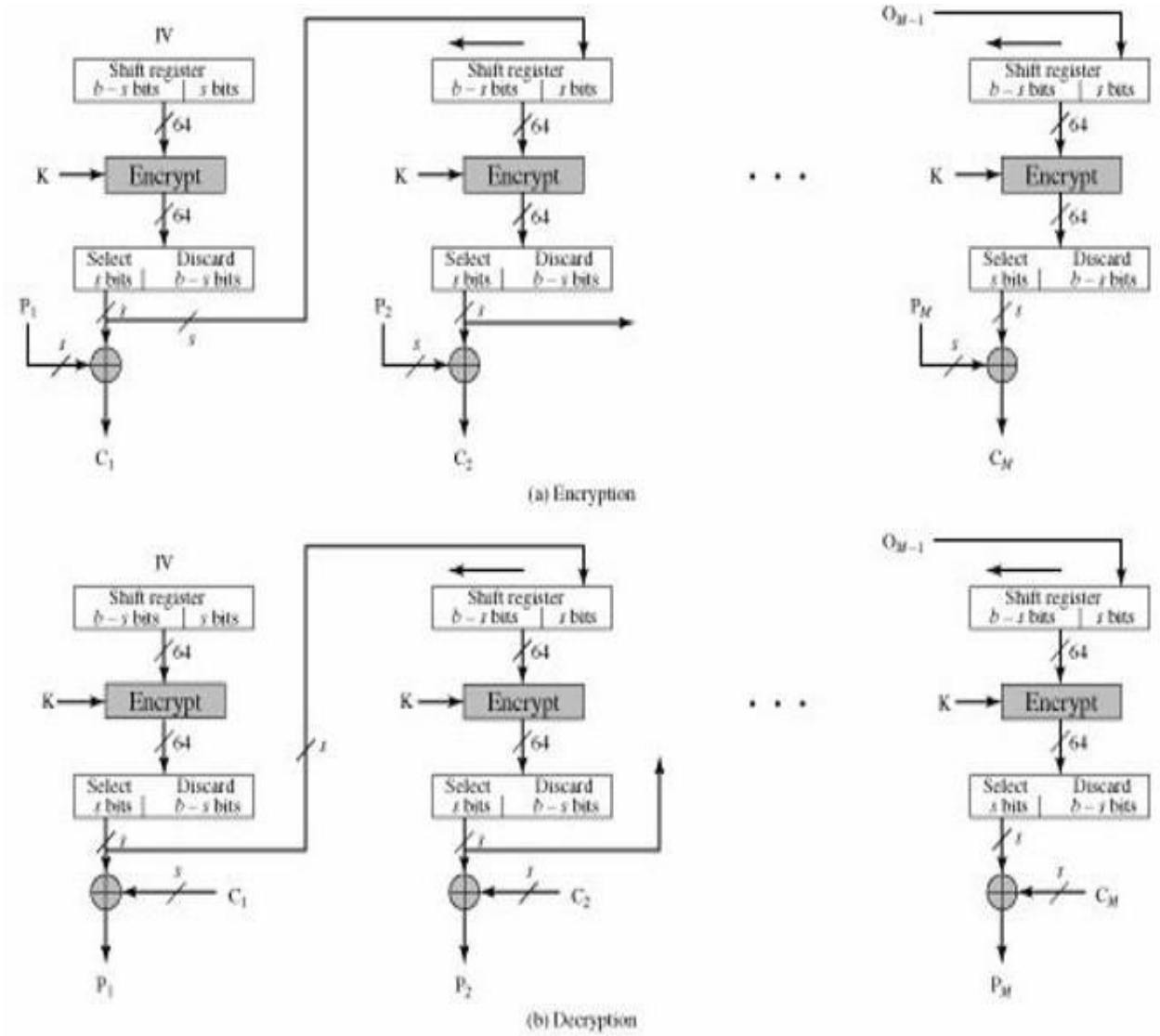
- The DES scheme is essentially a block cipher technique that uses b-bit blocks. However, it is possible to convert DES into a stream cipher, using either the cipher feedback (CFB) or the output feedback mode.
- One desirable property of a stream cipher is that the cipher text be of the same length as the plaintext. Thus, if 8-bit characters are being transmitted, each character should be encrypted to produce a cipher text output of 8 bits.
- Figure below depicts the CFB scheme. In the figure, it is assumed that the unit of transmission is s bits; a common value is s = 8.

- As with CBC, the units of plaintext are chained together, so that the cipher text of any plaintext unit is a function of all the preceding plaintext. In this case, rather than units of b bits, the plaintext is divided into segments of s bits.
- The input to the encryption function is a b-bit shift register that is initially set to some initialization vector (IV). The leftmost (most significant) s bits of the output of the encryption function are XORed with the first segment of plaintext P1 to produce the first unit of cipher text C1, which is then transmitted. In addition, the contents of the shift register are shifted left by s bits and C1 is placed in the rightmost (least significant) s bits of the shift register. This process continues until all plaintext units have been encrypted.
- For decryption, the same scheme is used, except that the received cipher text unit is XORed with the output of the encryption function to produce the plaintext unit. Note that it is the encryption function that is used, not the decryption function.



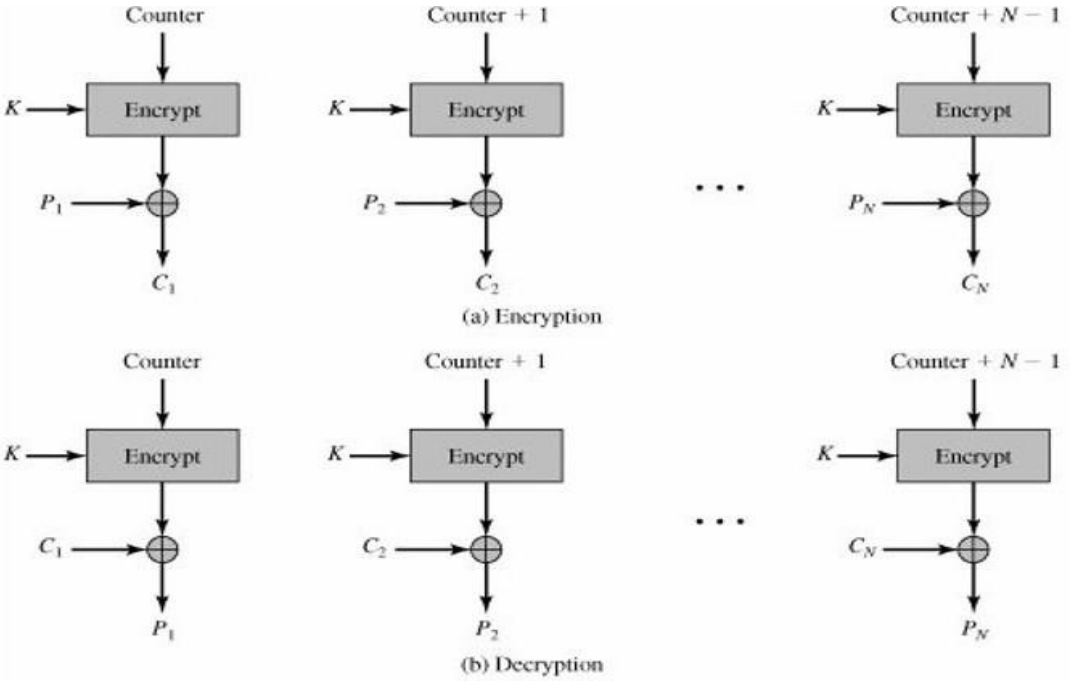
Output Feedback Mode (OFB)

- The output feedback (OFB) mode is similar in structure to that of CFB, as can be seen, it is the output of the encryption function that is fed back to the shift register in OFB, whereas in CFB the cipher text unit is fed back to the shift register.
- One advantage of the OFB method is that bit errors in transmission do not propagate
- The disadvantage of OFB is that it is more vulnerable to a message stream modification attack than is CFB



Counter Mode

- Figure depicts the CTR mode.
- A counter, equal to the plaintext block size is used.
- The only requirement is that the counter value must be different for each plaintext block that is encrypted.
- Typically, the counter is initialized to some value and then incremented by 1 for each subsequent block
- For encryption, the counter is encrypted and then XORed with the plaintext block to produce the cipher text block there is no chaining.
- For decryption, the same sequence of counter values is used, with each encrypted counter XORed with a cipher text block to recover the corresponding plaintext block.



Advanced Encryption Standard (AES)

- The Rijndael proposal for AES defined a cipher in which the block length and the key length can be independently specified to be **128, 192, or 256 bits.**
- The AES specification uses the same three key size alternatives but limits the **block length to 128 bits.**
- Fig below shows the overall structure of AES. The input to the encryption and decryption algorithms is a single 128-bit block.
- The key that is provided as input is expanded into an array of forty-four 32-bit words, w[i]. **Four distinct words (128 bits) serve as a round key for each round**

- Four different stages are used,
- one of permutation and three of substitution:
- Substitute bytes: Uses an S-box to perform a byte-by-byte substitution of the block
- Shift Rows: A simple permutation
- Mix Columns: A substitution that makes use of arithmetic over GF.
- Add Round Key: A simple bitwise XOR of the current block with a portion of the expanded key
- The structure is quite simple. For both encryption and decryption, the cipher begins with an Add Round Key stage, followed by nine rounds that each includes all four stages, followed by a tenth round of three stages.
- Each stage is easily reversible. For the Substitute Byte, Shift Rows, and Mix Columns stages, an inverse function is used in the decryption algorithm