UNIT-2

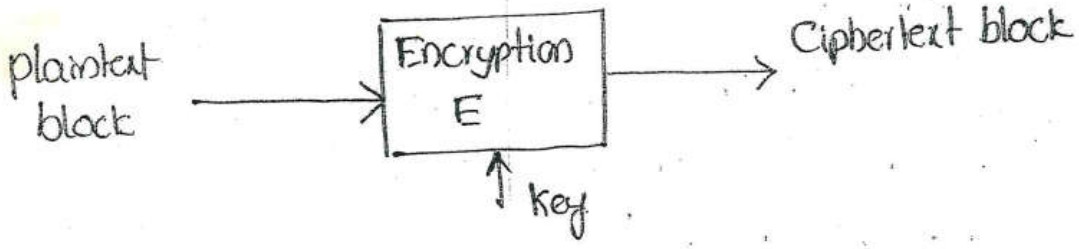
BLOCK CIPHERS

Block Cipher: It is a type of symmetric encryption which operates on blocks of data.

Ex: DES, IDEA, AES

→ The block cipher breaks message into fixed sized blocks.

→ Block cipher takes one block of data at a time and transform it to another block of same length using the user provided secret key.



Properties of Good Ciphers:

Claude shannon gave two properties for a good cryptosystem:

1. **Confusion:** Confusion means that the key should not relate in a simple way to the ciphertext.

   * Substitution is the mechanism for confusion

2. **Diffusion:** Diffusion means that if we change a character of plain the several characters of ciphertext should change and similarly if we change a character in ciphertext then several characters of plaintext should change.

   * Transposition or permutation is a mechanism for diffus

(1) Traditional Block Cipher Structure:

(Feistel Cipher Structure).

→ Feistel proposed a cryptographically stronger cipher that alternates substitutions and permutations.)*

the Fiestal cipher model is based on block cipher where group of characters converting from plain text to cipher text

Encryption:

1. partition input block into two halves.
2. Process the two halves through multiple rounds which perform a substitution on left half based on a round function of right half and subkey
3. A permutation is performed that consists of interchange of two halves of the data

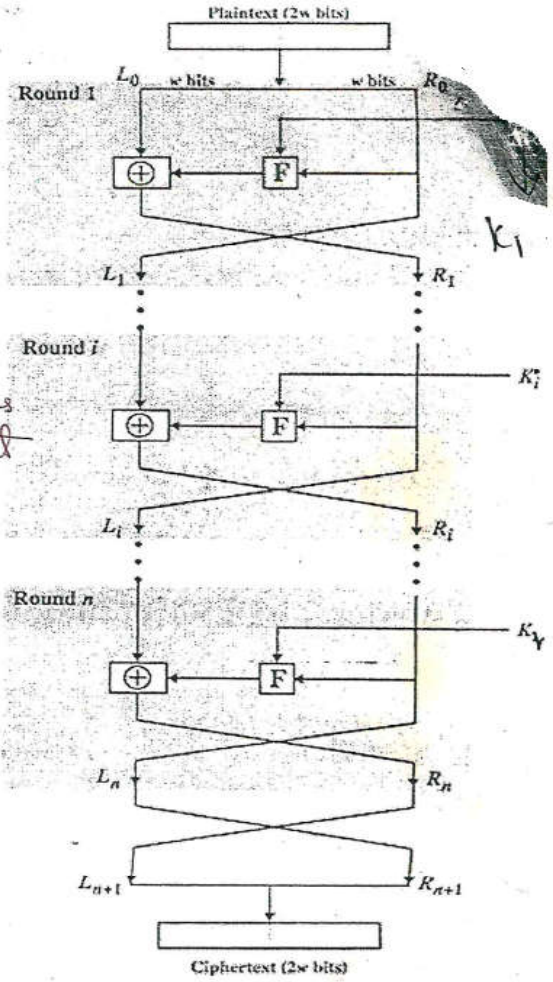


Figure: Feistel Cipher Structure

→ Mathematic Representation of Encryption:

Each round i has L_{i-1} and R_{i-1} and a subkey K_i

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

$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

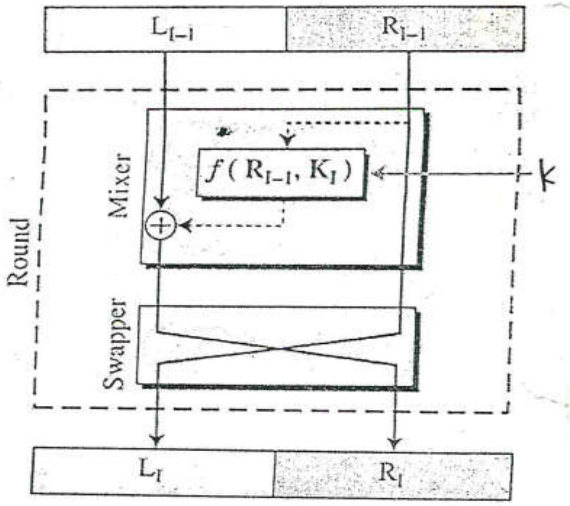


Figure: Feistel Cipher Round Structure

→ The Feistel Structure uses Substitution-Permutation Network (S-P net) proposed by Shannon. S-P Network provides Confusion and Diffusion of a message.

Decryption:

→ The process of decryption is essentially the same as encryption process

→ The decryption uses the ciphertext as input to the ~~alg~~ algorithm, but use the subkeys K_i in the reverse order. That is, K_n is used in first round, K_{n-1} is used in second round and so on K_1 will be used in last round.

→ The Decryption process mathematically represented as

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

$$L_{i-1} = R_i \oplus F(R_{i-1}, K_i)$$

$$= R_i \oplus F(L_i, K_i)$$

Block Cipher Design Principles:—

The Feistel network depends on the choice of the following parameters and design features:

1. Block Size: Larger the block size means greater the security. but reduced speed for encryption/ decryption.

2. key size: Larger the key size means greater the security but decrease speed for encryption/ decryption.

   * The greater security is acheived by greater resistance to brute-force attacks and greater confusion.

3. No of Rounds: Multiple rounds offer increased security. A typical size is 16 rounds.

4. Round Function: Greater complexity means greater resistance to cryptanalysis.

5. Subkey Generation Algorithm: Greater complexity means greater resistance to cryptanalysis.

There are two more consideration in the design of Feistel Cipher:

1. Fast software encryption/ decryption: The speed of execution of the algorithm becomes a concern.

2. Ease of analysis: If the algorithm concisely and clearly explai it is easier to analyze the algorithm for cryptana vulnerabilities.

(9) DES (Data Encryption Standard):

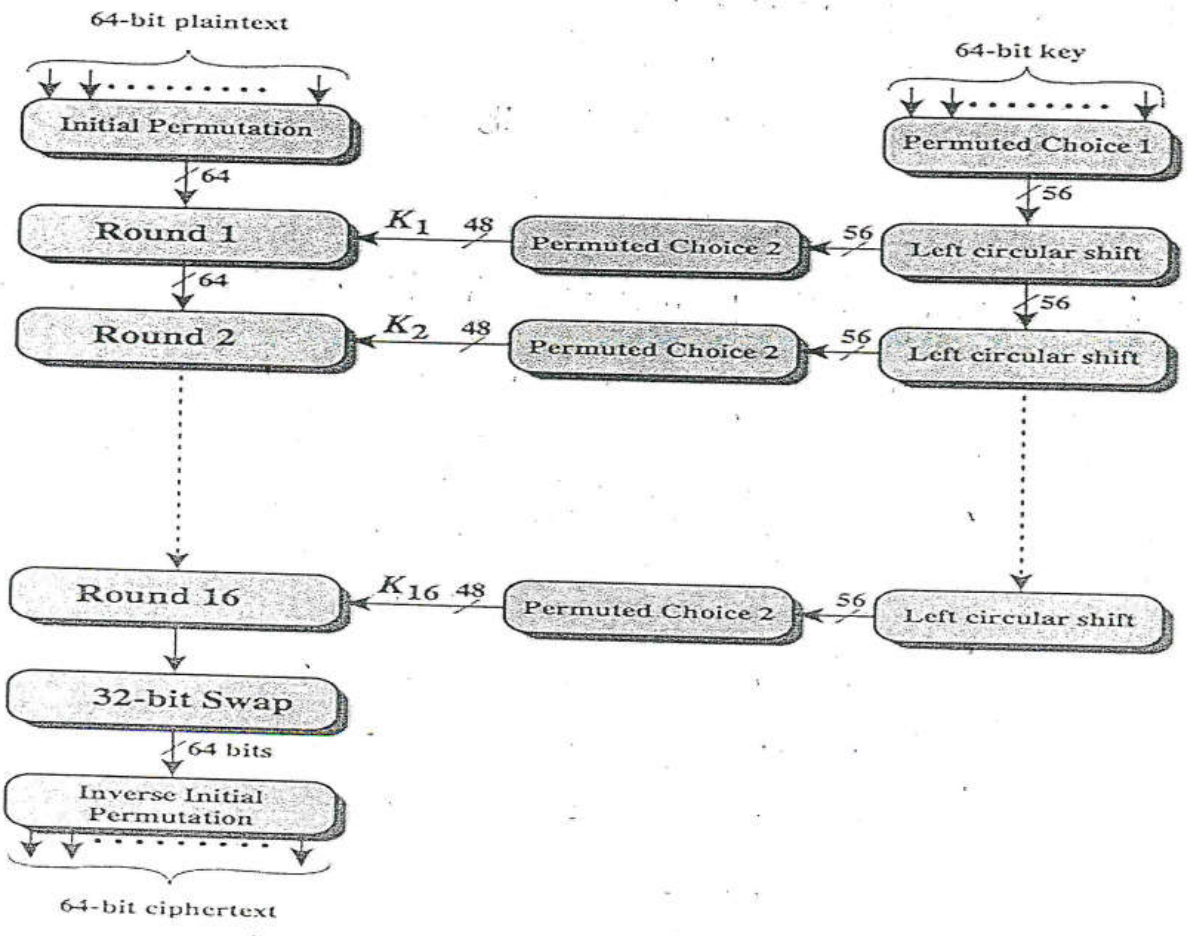
→ DES is also called "Data Encryption Algorithm (DEA)". It is most widely used block cipher.

→ Features:

Plaintext block size : 64 bits
Keysize : 56 bits (In reality 64 bits, but 8 bits are used for parity
Block size : 32 bits
No. of rounds : 16
Subkey size : 48 bits
Ciphertext : 64 bits.

→ DES uses both transposition and substitution, so it is called "produc cipher.

a) DES Encryption:



Encryption Process:-

i) Initial Permutation (IP)

ii) Generation of round keys

iii) Encryption in 16 identical rounds

a) Substitution

b) Permutation

Additionally swap left and right halves

iv) Inverse Initial Permutation (IP^{-1})

i) Initial Permutation (IP):

* IP reorders the input data bits. The even bits are moved to left half and odd bits are moved to right half.
* Consider the plaintext bits arranged as

1 2 3 4 5 6 7 8
9 10 11 12 13 14 15 16
⋮
57 58 59 60 61 62 63 64

* After applying initial permutation the bit positions are changed to

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

(ii) Details of single round:

→ The Left and Right halves of each 64-bit intermediate value are treated as separate 32 bit quantities labeled L (Left) and R (Right).

→ The overall processing at each round can be summarized as:

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

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

→ Round key k_i is 48 bits

→ The Plaintext blocks Left and Right are 32 bits.

F(R,K): DES have 8 S-boxes which map 6 bits to 4 bits.

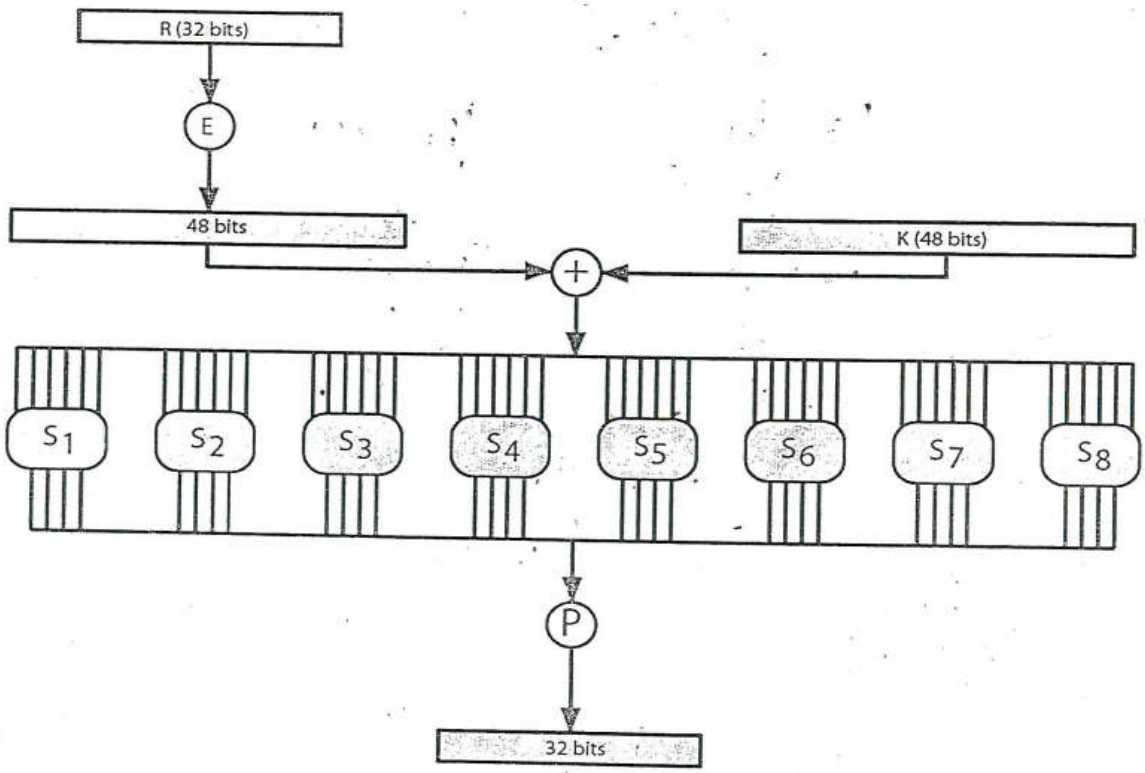


Fig: F(R,K).

(ii) Generation of Round key:

* DES actually uses 64 bits key. The initial 64 bit key is transformed 56 bit key by discarding 8th bit of initial key.
* 56 bit key is divided into two halves (C, D) of 28-bits.
* At each round, C_{i-1} and D_{i-1} are subjected to a circular ~~rotation~~ shift ~~using~~ or rotation of 1 or 2 bits depending on key schedul
* Apply permuted Choice 2 for the shifted values to produce 48 bit round key. (K_i)

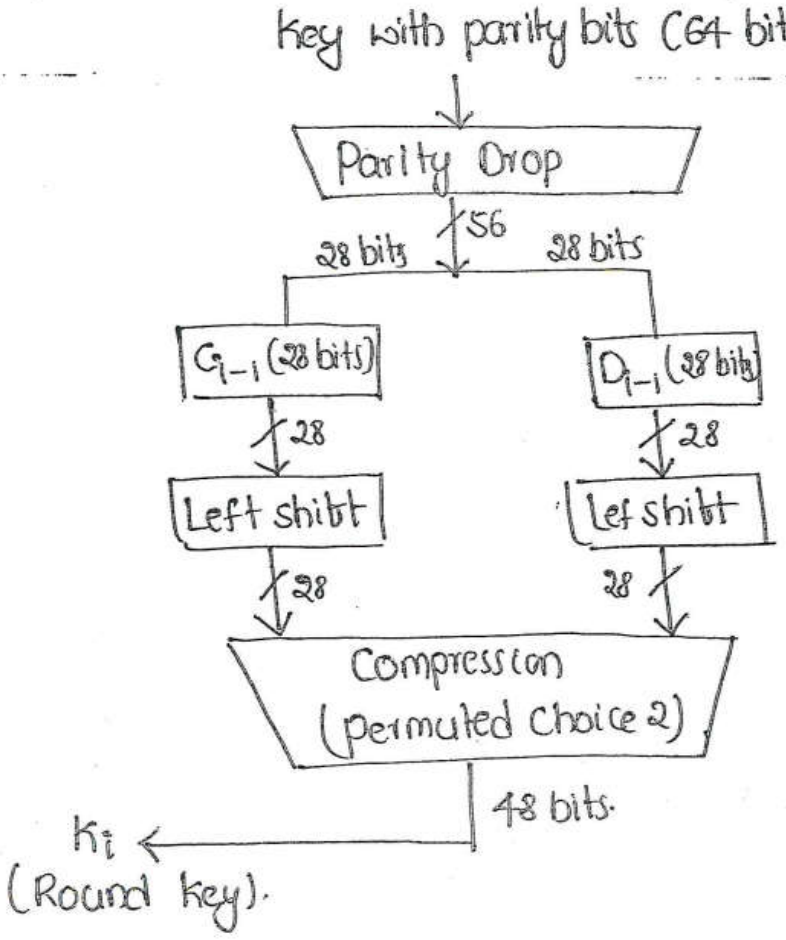


Fig: Round key generation

Round Function:

1. R input is expanded to 48 bits by using Expansion Permutation
2. The resulting 48 bits are XORed with K_i.
3. The output is passed through S-boxes which accepts 6-bits of input and produce 4 bits as output. It results in 32 bit output.

Expansion Permutation:

→ The Right half is given as input for Expansion permutation

→ The Right half is expanded from 32-bits to 48 bits by using a table that defines a permutation plus an expansion that involve duplication of 16 of the Right half bits

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

5
9
13
17
21
25
29
1

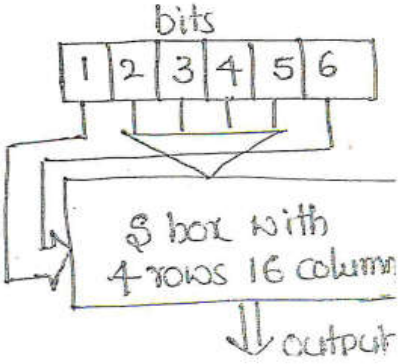
S-Boxes:

→ DES have 8 S-boxes which can map 6 bit input to 4 bit outpu

→ Each S-box contain 64 entries and each row contain numbers from 0-

→ For each S-box input,

1. Outer bits 1&6 (row bits) select one row
2. Inner bits 2-5 (column bits) select one column
3. Then value in the intersection is substituted.



S-Boxes (S1-S8)

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

(v) Inverse Initial Permutation (IP^{-1}):

→ IP^{-1} is the inverse of Initial Permutation (IP). It is also called "Final Permutation".

→ The Final permutation is as follows.

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

DES Decryption:

* The DES Decryption algorithm is identical to the encryption algorithm step by step in the same order only with the subkeys applied in the reverse order.

Strength of DES:-

a. The Use of 56-bit key: 56-bit key is used in encryption. There are 256 possible keys.

* The brute force attack on such no. of keys is impractical.

b. The Nature of Algorithm: Cryptanalyst can perform cryptanalysis by exploiting the characteristic of DES algorithm but no one has succeeded in finding out the weakness.

c. Timing Attacks: A timing attack exploits the fact that encryption decryption algorithm takes slightly different amount of time on different inputs.

d. Avalanche Effect: It means that very small change in input will lead to very big change in output.

Weaknesses of DES:-

a. Two chosen inputs to an S-box can create the same output.

b. The purpose of initial and final permutation is not clear.

Double DES:-

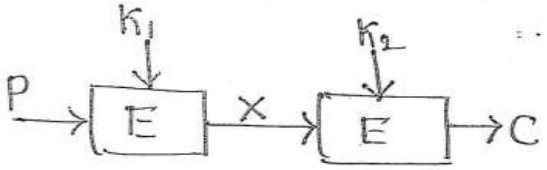
→ Double DES uses two instances of DES ciphers for encryption and two instances of reverse ciphers for decryption.

→ Each instance use a different key. The size of key is doubled.

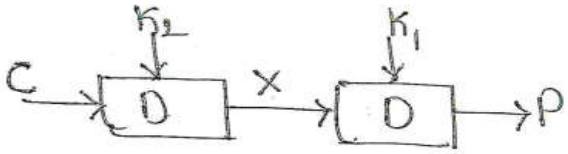
→ Double DES is vulnerable to meet-in-the-middle attack

Encryption: $C = E(K_2, E(K_1, P))$

Decryption: $P = D(K_1, D(K_2, C))$



Encryption



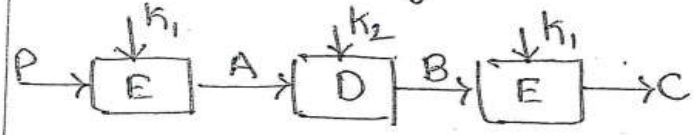
Decryption

Triple DES with 2 keys:

→ Use three stages of DES for encryption and decryption.

→ 1st, 3rd stage uses K_1 and 2nd stage use K_2 key.

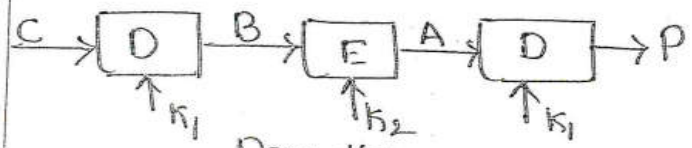
→ It is much stronger than Double DES.



Encryption.

Encryption:

$C = E(K_1, D(K_2, E(K_1, P)))$



Decryption

Decryption

$P = D(K_1, E(K_2, D(K_1, C)))$

~~Dec~~ Triple DES with 3 keys:

→ It uses 3 keys for 3 stages of DES.

Encryption: $C = E(K_3, D(K_2, E(K_1, P)))$

Decryption: $P = D(K_1, E(K_2, D(K_3, C)))$

(3) AES (Advanced Encryption Standard):-

→ AES was published by the National Institute of Standards and Technology in 2001.

→ AES is a symmetric block cipher that is intended to replace DES

→ Features:-

Plaintext block size : 128 bits

Keysize :

128	192	256	bits
10	12	14	

No. of Rounds :

No. of rounds in AES depends on keysize.

→ AES operates on 4×4 column-major matrix called as "state"

→ AES operates on 4 transformations:

1. SUB BYTES: Non-linear substitution step, where each byte is replaced with another according to S-box.

2. SHIFT ROWS: This is transformation step, where last three rows of the state are shifted cyclically by a certain no. of rotations.

1st row ← 0 rotations 2nd row ← 1 rotation

3rd row ← 2 rotations 4th row ← 3 rotations

3. MIX COLUMNS: Mixing operations operates on the columns of the state combining four bytes in each column.

4. ADD ROUND KEY: Each byte of the state is combined with a block round key using XOR.

AES Structure:-

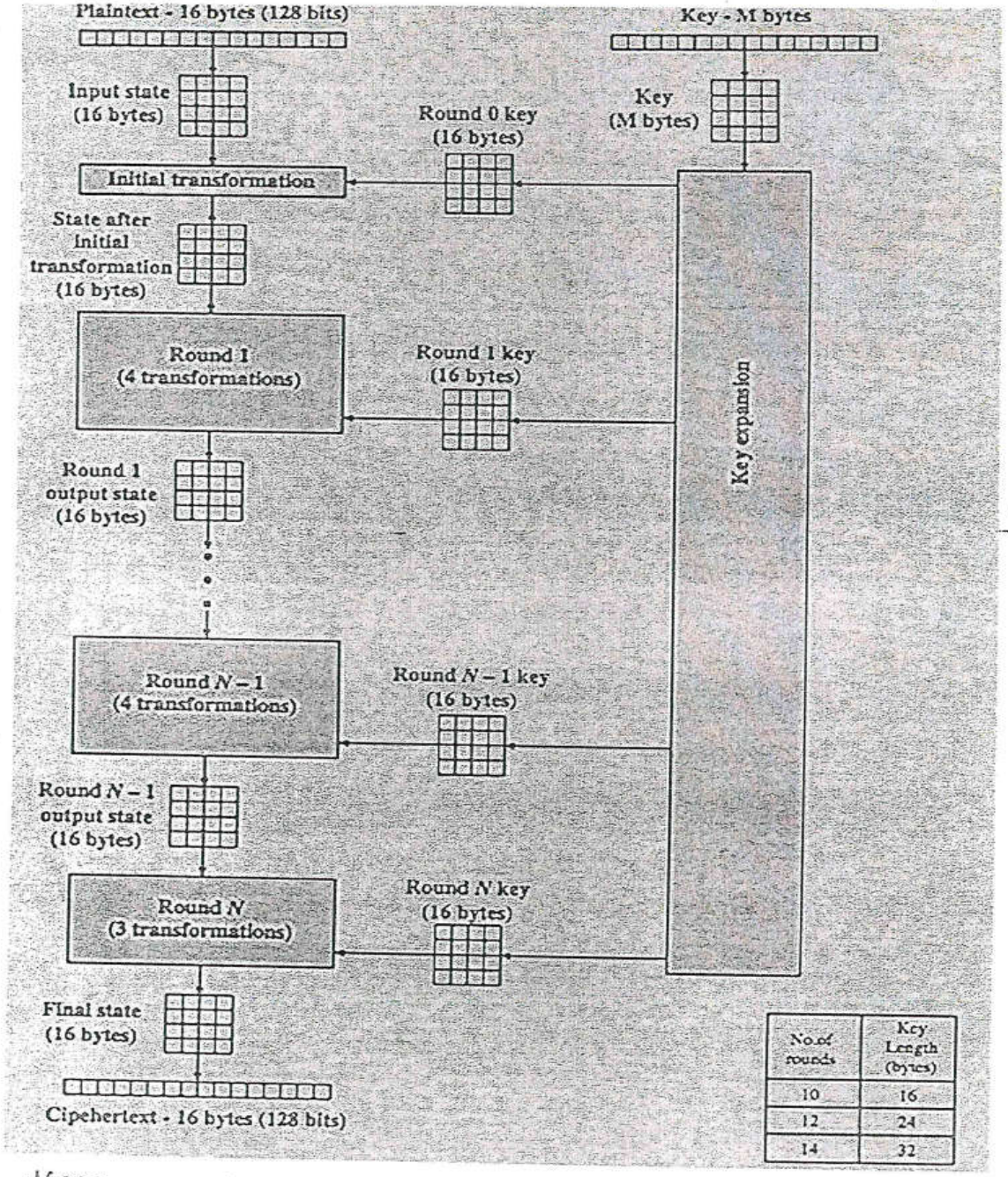


Fig: AES Structure

Encryption:

1. plaintext is divided into 16 bytes
2. The plaintext will be converted to 4x4 matrix called "Input state".
3. The round 0, XORs the input state with the Round key.
4. Rounds 1 to N-1 uses 4 transformations.
5. Round N uses only 3 transformations (except MIXCOLUMNS)
6. Convert 4x4 matrix to 16 bytes to get the ciphertext.

(4) AES Transformation Functions:

1. SUB BYTES (Substitute Bytes):

→ Each byte of the state is replaced by another byte using S-box

→ procedure:

* Leftmost 4 bits are used as a row value
* Rightmost 4 bits are used as a column value
* The row and column values serve as indexes into S-box to select a unique 8-bit output value.

Ex: The hexadecimal value 95

references row 9

column 5

replaces the value 2A.

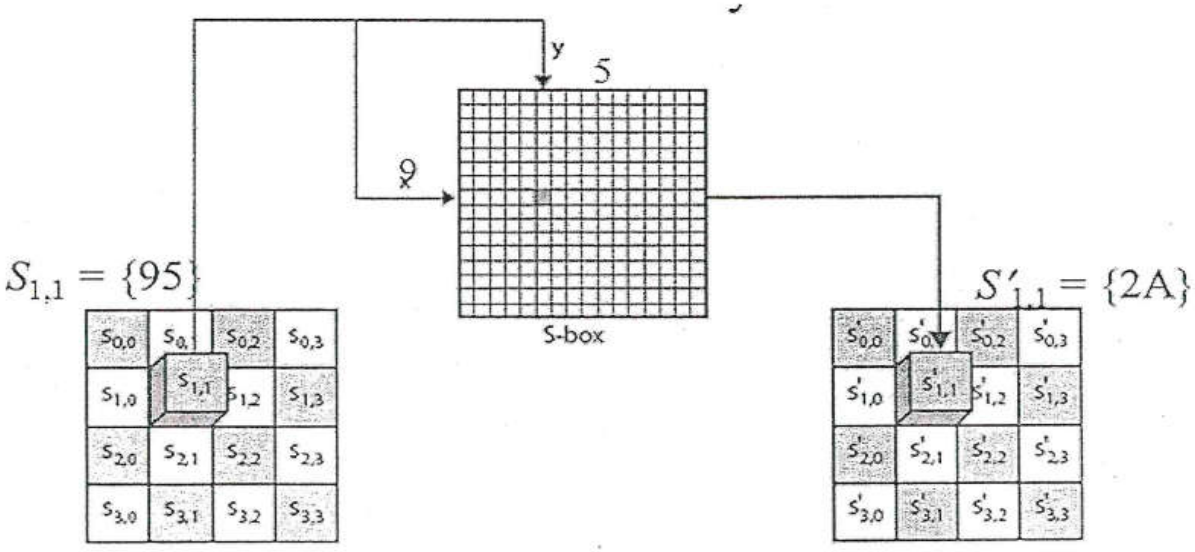


Figure: Substitution of Bytes Transformation Function

2. SHIFT ROWS: The rows of the state matrix are rotated by certain no. of times.

First row is unchanged.

Second row is rotated 1 time left

Third row is rotated 2 times left

Fourth row is rotated 3 times left

→ Decryption uses right rotation.

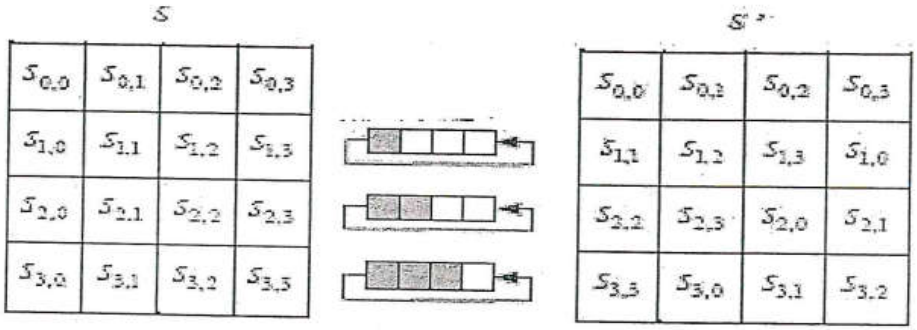


Figure: Shift Rows Transformation Function

3. MIX COLUMNS:

→ In this step, the four bytes of each column of the state are combined using an invertible linear transformation.

→ The MixColumns function takes four bytes as input and outputs four bytes.

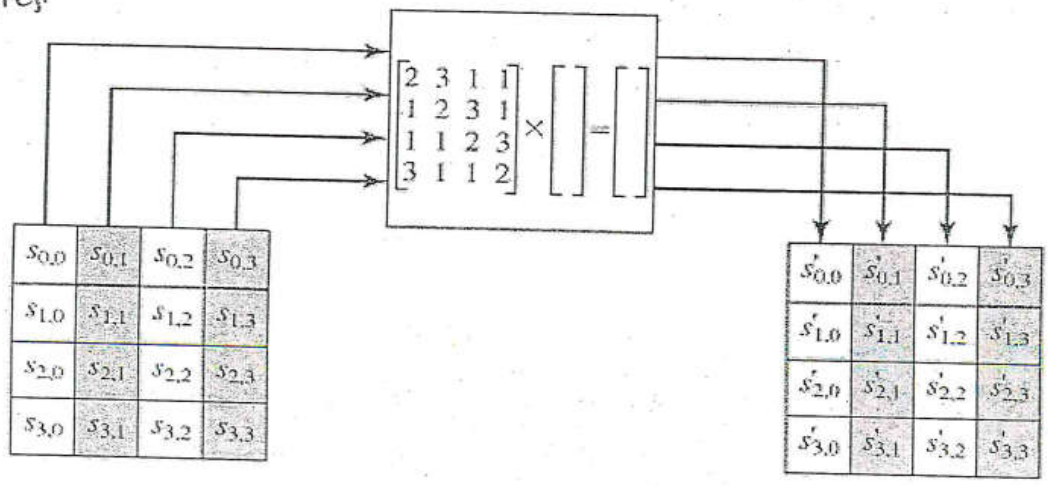


Figure: Mix Columns Transformation Function

→ Each column is transformed using a fixed matrix $\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$

→ Every column in input state is multiplied by the matrix

multiplication by 1 means – no change

multiplication by 2 means – shifting to the left

multiplication by 3 means – shifting to the left and XOR with initial unshifted value.

4. ADDROUNDKEY:

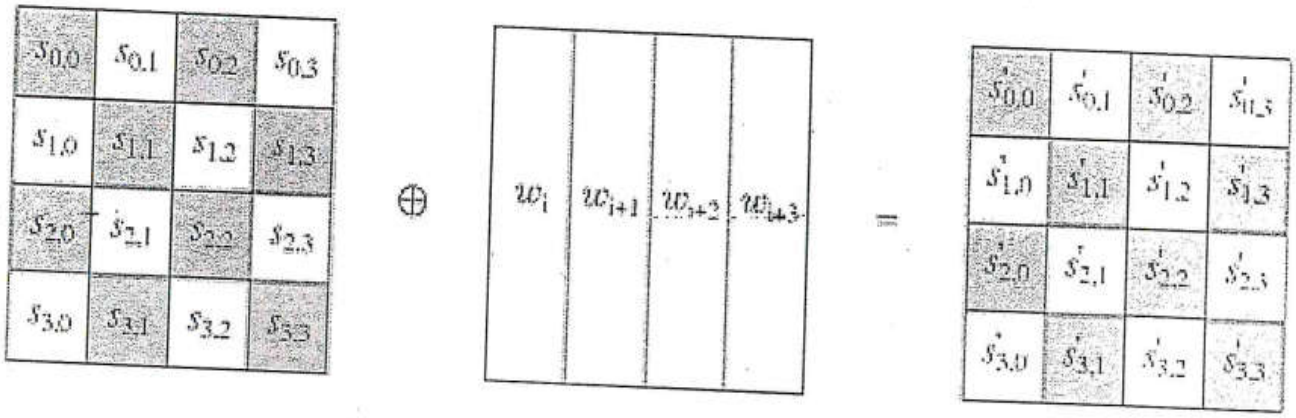


Figure: Add Round Key Transformation Function

→ In this operation, a round key is applied to the state by simply applying bitwise XOR.

→ The Round key is derived from cipher key by the means of key schedule.

→ The Round key length is equal to block key length.

→ In this, the roundkey is combined with the state.

(5) Key Expansion :

→ Key expansion algorithm takes as input key of 128 bits and produces a linear array of 44 words.

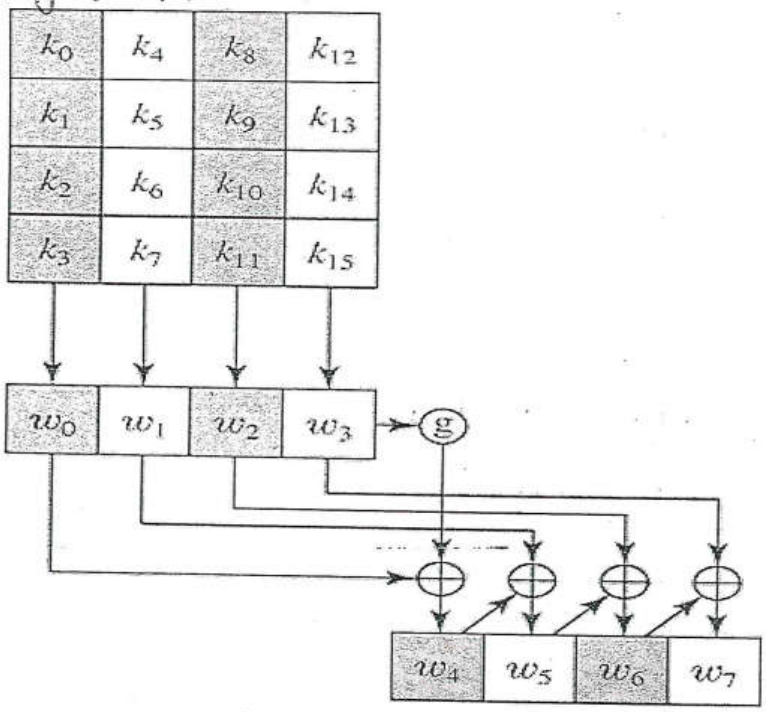


Figure: AES Key Expansion

→ The key is copied into the first four words of the expanded key. The remainder of the expanded key is filled in four words at a time.

→ Each word w_i depends on the preceding word w_{i-1} and the word four positions back w_{i-4}.

$$w_i = w_{i-1} \oplus w_{i-4}.$$

→ First subkey (w_3, w_2, w_1, w_0) = Cipher key.

$w_4 = w_0 \oplus g_1$
$w_5 = w_4 \oplus w_1$
$w_6 = w_5 \oplus w_2$
$w_7 = w_6 \oplus w_3$

$w_8 = w_4 \oplus g_2$
$w_9 = w_8 \oplus w_5$
$w_{10} = w_9 \oplus w_6$
$w_{11} = w_{10} \oplus w_7$

The procedure is repeated until all subkeys are generated.

(6) BlowFish Algorithm:-

→ BlowFish was designed in 1993 by Bruce Schneier as a fast, free alternative to existing encryption algorithms.

→ Features:

plaintext block size : 64 bits
keysize : Varies from 32-448 bits. By default it is 128 bits
No of rounds : 16

→ Blowfish algorithm is divided into two parts:

a. key-Expansion
b. Data Encryption

a. Key Expansion:

→ Key Expansion will convert a key of atmost 448 bits into several subkeys totalling 4168 bytes.
→ P-array is used to store the subkeys. P array contain 18 32-bit subk
→ key expansion is done using 4 Four 32-bit sboxes. Each S-box contain 256 entries.

Generating the subkeys:

① Initialize P-array and then four S-boxes with a fixed string.
② XOR P1 with first 32-bits of the key, XOR P2 with second 32-bits of the key and so on for all bits of the key. Repeatedly cycle thro the keybits until entire P-array has been XORed with key bits.
③ Encrypt all zero-strings with the blowfish algorithm using subkeys described in steps ① and ②.
④ Replace P1 and P2 with the output step ③

(c) Continue the process, replacing all entries of the array P, and then all four S-boxes in order with the output of continuously changing blowfish algorithm.

→ 521 iterations required to generate all required subkeys.

b) Data Encryption:

→ It is having a function to iterate 16 times of network.

→ Each round consists of key dependent permutation and a key and data dependent substitution.

→ All operations are XORs and additions on 32-bit words.

→ The only additional operations are four indexed array data lookup tables for each round.

Encryption Algorithm:

① Divide plaintext into two halves xL and xR.

② For i = 1 to 16

$xL = xL \oplus P_i$

$xR = F(xL) \oplus xR$

Swap xL and xR

③ $xR = xR \oplus P_{17}$

④ $xL = xL \oplus P_{18}$

Recombine xL and xR

Function F:

The Function F takes 32 bit input and Function divides 32-bit input to 4 eight bit parts.

The eight bits are given as input for 4 S-boxes.

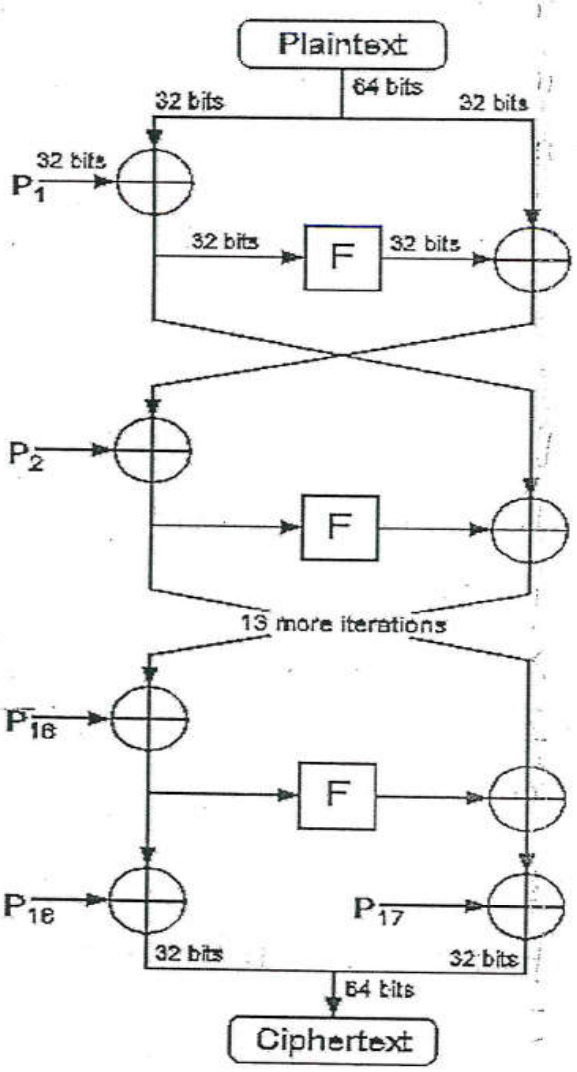


Figure: BlowFish Encryption

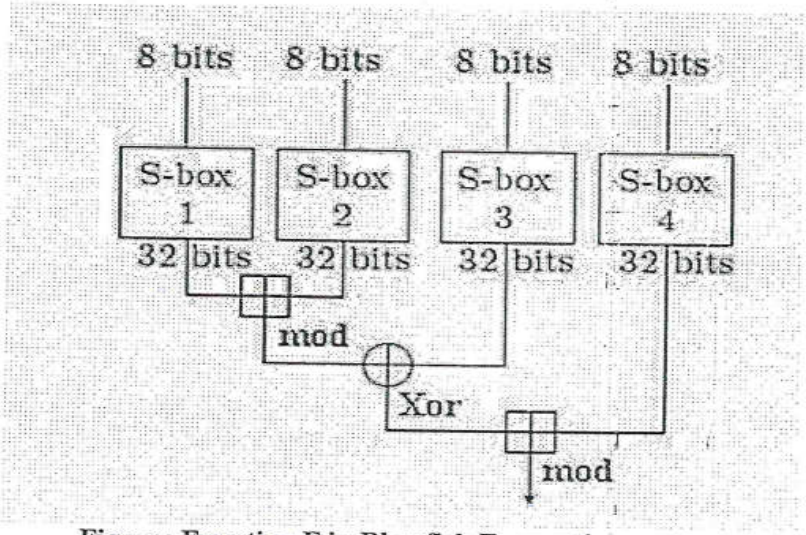


Figure: Function F in Blowfish Encryption

CAST-128:-

→ CAST-128 is a design procedure for symmetric encryption algorithm developed by Carlisle Adams and Stabford Tavares.

→ CAST 128 follows Feister Cipher structure.

→ Features:

plaintext block size	: 64 bits
Keysize	: Varies from 40 bits to 128 bits
No. of rounds	: 16, if the keysize greater than & 12, if the keysize < 80 bits. (keysizes are in 8-bit increments).

→ CAST-128 uses two keys for each round,

1. Masking key : 32 bits (K_{m_i}) 2. Rotation key (K_{r_i}) : 5 bits

→ CAST-128 uses four different operations,

+ – Modified addition modulo 2^{32}

– – Modified subtraction modulo 2^{32}

XOR – Bitwise XOR

<<< – Circular left Rotate

Encryption Algorithm:

① Plaintext is divided into two halves L_0 and R_0

② For each round i,

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus F_i(R_{i-1}, K_{m_i}, K_{r_i})$$

③ The procedure is repeated for 16 rounds to get ciphertext

Function F: Function F is designed to have good confusion and diffusion.

→ It uses s-box substitutions, modulo 2^{32} addition and subtraction, XOR operations and key dependent ~~transformations~~ rotations.

Definition of F

Rounds 1,4 7,10,13,16	$I=(Km_i + R_{i-1})<<<Kr_i)$ $F=((S1[I_a] \oplus S2[I_b]) - S3[I_c]) + S4[I_d]$
Rounds 2,5 8,11,14	$I=(Km_i \oplus R_{i-1})<<<Kr_i)$ $F=((S1[I_a] - S2[I_b]) + S3[I_c]) \oplus S4[I_d]$
Rounds 3,6 9,12,15	$I=(Km_i - R_{i-1})<<<Kr_i)$ $F=((S1[I_a] + S2[I_b]) \oplus S3[I_c]) - S4[I_d]$

*Strength of Function F depends on strength of S-boxes.

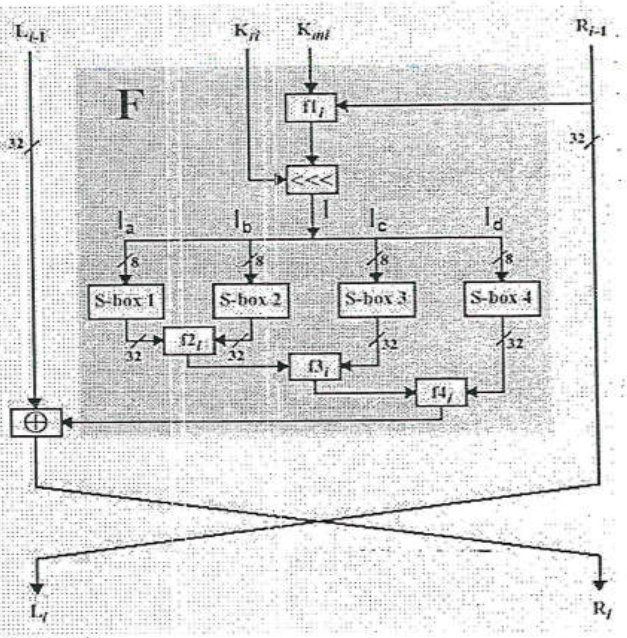


Figure: Single Round of CAST 128

→ S-Boxes:

CAST-128 uses 8 substitution boxes,

S1-S4 : Round Function S-boxes

S5-S8 : Key schedule S-boxes.

Each s-box has 256 entries numbered from 0 to 255

Decryption:

Decryption is same as encryption with round keys in reverse order.

Subkey Generation: Subkey generation is complex process.

Label the bytes of 128 bit key as x0 x1 x2 x3 x4 x5 x6 x7 x8 x9 xA xB xC xD xE xF.

Km_1 to Km_{16} — Masking subkeys

Kr_1 to Kr_{16} — Rotation subkeys

z_0 to z_F — Intermediate Byte

k_1 to k_{32} — Intermediate words

k_1 to k_{32} are calculated using S-boxes (S5-S8)

Then the subkeys are defined

```
for i=1 to 16 do
    Km_i = k_i ;
    Kr_i = k_16 + i ;
```

IDEA (International Data Encryption Algorithm):

→ IDEA is a symmetric block cipher proposed to replace DES.

→ IDEA was included in PGP

→ It is a variant of Feistel Cipher.

→ Features:

plaintext block size	:	64 bits
keysize	:	128 bits
No. of rounds	:	8 + Final Transformation.

→ Basic Operations:

IDEA uses 3 basic operations. Each operation is performed on two 16 bit inputs.

1. Bitwise XOR
2. Modified addition modulo 2^{16}.
3. Modified multiplication modulo $2^{16}+1$.

→ In IDEA, confusion is achieved by using 3 seperate operations in combination.

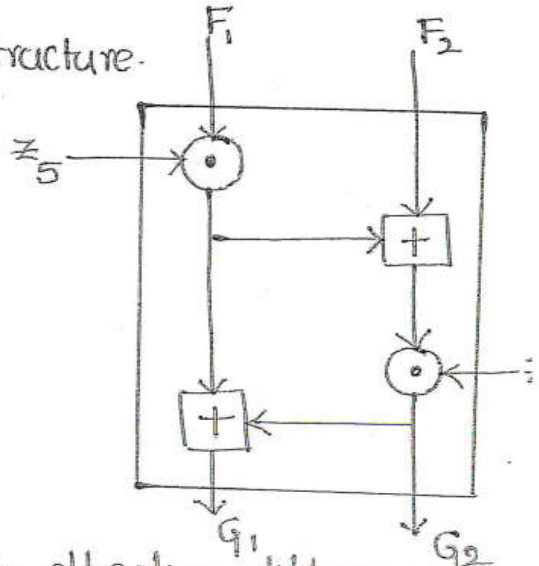
→ In IDEA, Diffusion is achieved by MA structure.

Multiplication/Addition structure:

F1, F2 : Two 16-bit values derived from plaintext

Z5,Z6 : Two 16-bit subkeys derived from the key.

G1, G2 : Two 16-bit outputs.



MA structure is repeated 8 times to provide effective diffusion.

IDEA Encryption Structure:-

① Divide the 64-bit input block into 4 16-bits data blocks.

② Four data blocks are then processed through eight rounds and each round uses 6, 16-bit subkeys generated from original key.

③ In each round,

The data blocks are transformed by applying 3 operations with the subkeys. The whole encryption uses a total of 52 subkeys.

④ Finally, the output transformation is applied to produce four 16-blocks of ciphertext which are then concatenated to form the final 64 bit ciphertext block.

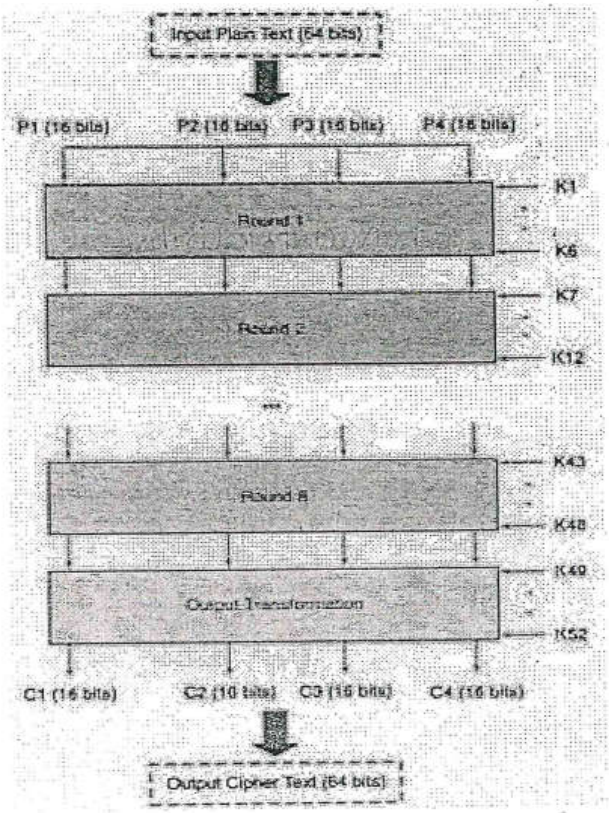


Figure: IDEA Encryption Process

Round in IDEA:-

→ IDEA have eight rounds and each round involves a series of operations on the 4 data blocks with 6 keys.

→ Steps in each round:

1. Multiply* P1 and K1
2. Add* P2 and K2.
3. Add* P3 and K3
4. Multiply* P4 and K4
5. XOR results of step 1 and step 3
6. XOR results of step 2 and step 4
7. Multiply* results of step 5 with K5
8. Add* results of step 6 and step 7
9. Multiply* results of step 8 with K_6.
10. Add* results of step 7 and step 9.
11. XOR results of step 1 and step 9
12. XOR results of step 3 and step 9
13. XOR results of step 2 and step 10
14. XOR results of step 4 and step 10

Multiply* — Modified multiplication modulo $2^{16}+1$

Add* — Modified addition modulo 2^{16}.

Modular arithmetic is required to ensure that even if the results of addition and multiplication of two 16-bit numbers contains more than 16 bits, we bring back to 16-bits.

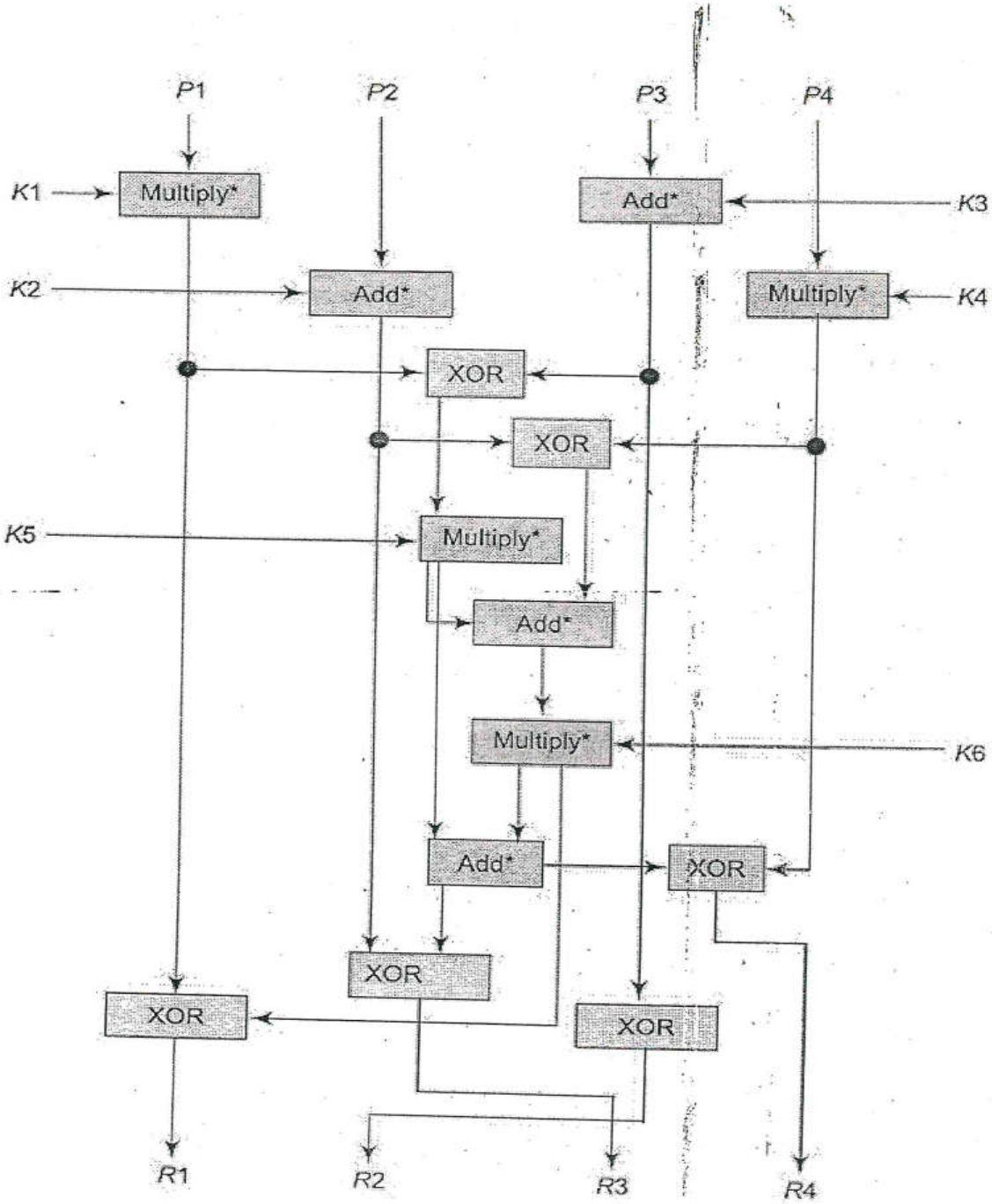


Figure: Single Round in IDEA

Output Transformation :-

→ The output Transformation is one time process. It takes place at the end of 8th round.

→ Steps in output transformation:

1. Multiply* R1, K1
2. Add* R2, K2
3. Add* R3, K3
4. Multiply R4, K4

R1, R2, R3, R4 — Four subblocks after Round 8

K1, K2, K3, K4 — The 16-bit keys available for output transformation.

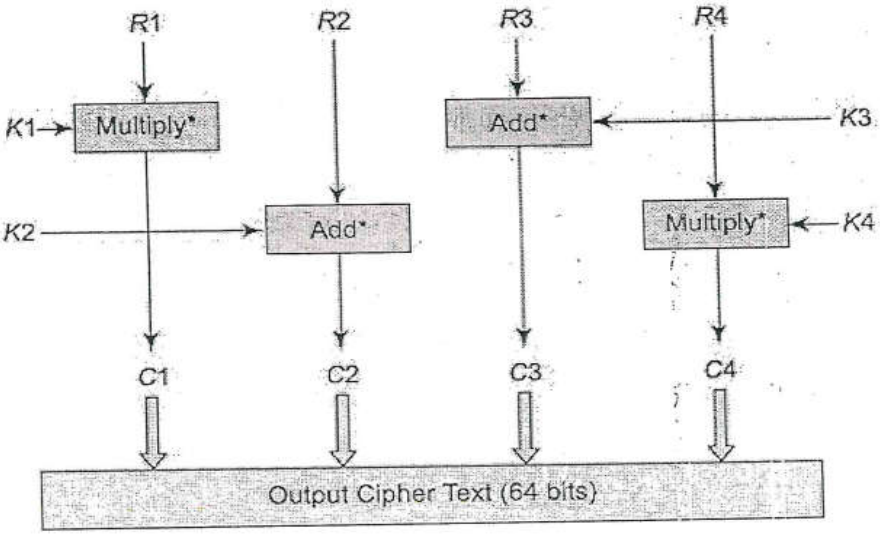


Figure: OUTPUT Transformation

Subkey Generation:

→ In IDEA,

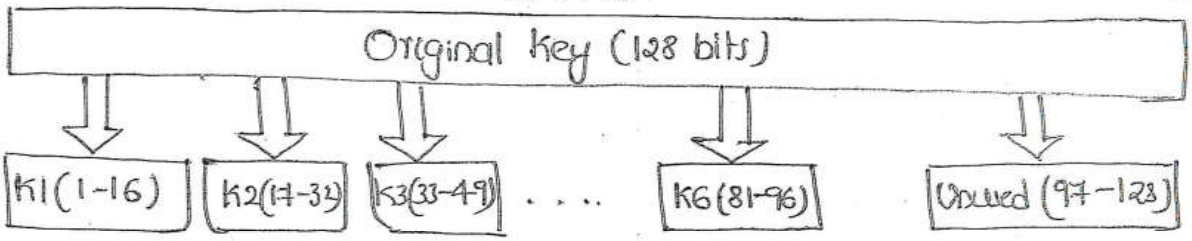
each of eight rounds make use of six subkeys ($8 \times 6 = 48$) and output transformation uses 4 subkeys ($1 \times 4 = 4$)

Total 52 subkeys are generated from 128 bit key

a) First Round: Initial key consists of 128 bits, from which 6 subkeys K1 to K6 are generated for the first round.

→ Out of 128 bits, the first 96 bits are used for the first round.

→ 97-128 bits are unused.

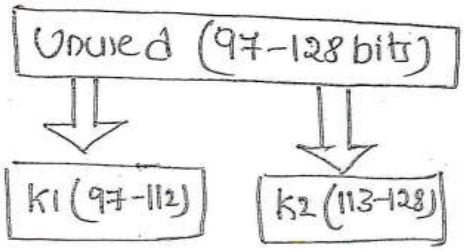


b) Second Round:

→ IDEA employs a technique of "key shifting".

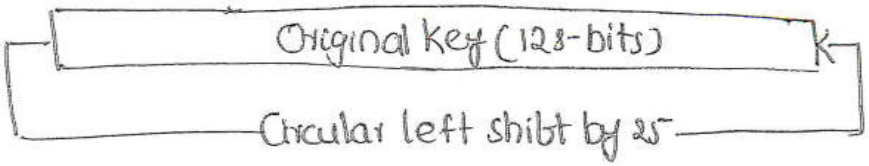
→ The original key is shifted left circularly by 25 bits.

1.

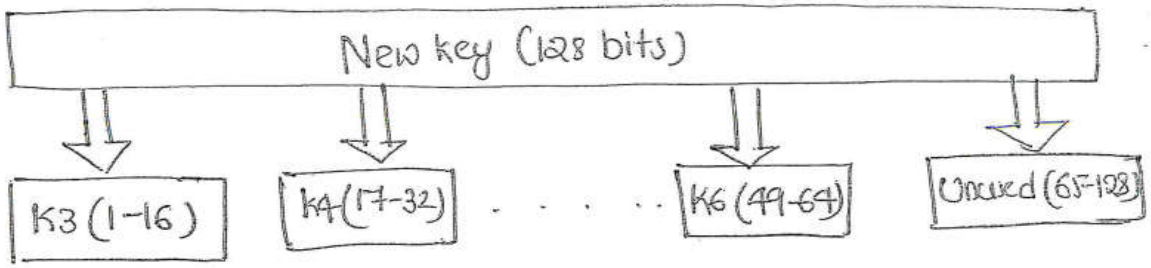


Subkeys K1, K2 for Round 2.

2.



3. Start allocation of subkey K3 to K6 for Round 2



The same process is repeated upto the 8th round.

c) Output Transformation:

At the end of ~~eight~~ eighth round, the key was exhausted, Hence the key is again shifted by 25 bits.

The first 64 bits are takes as the 4 subkeys for output transformation

IDEA Decryption:

The Decryption is same as encryption but the keys are inverse of encryption subkeys.

Strength of IDEA:

IDEA uses 128 bit key, which is double than keysize of DES, thus to break IDEA, 2^{128} encryption operations required

Block Cipher Modes of Operation:-

a) Electronic Code Book Mode (ECB):

→ It is simplest mode of operation.

→ The plaintext message is divided into blocks of 64 bits each.

→ Each block is encrypted independently of other blocks.

→ For all the blocks in a message, the same key is used for encryption

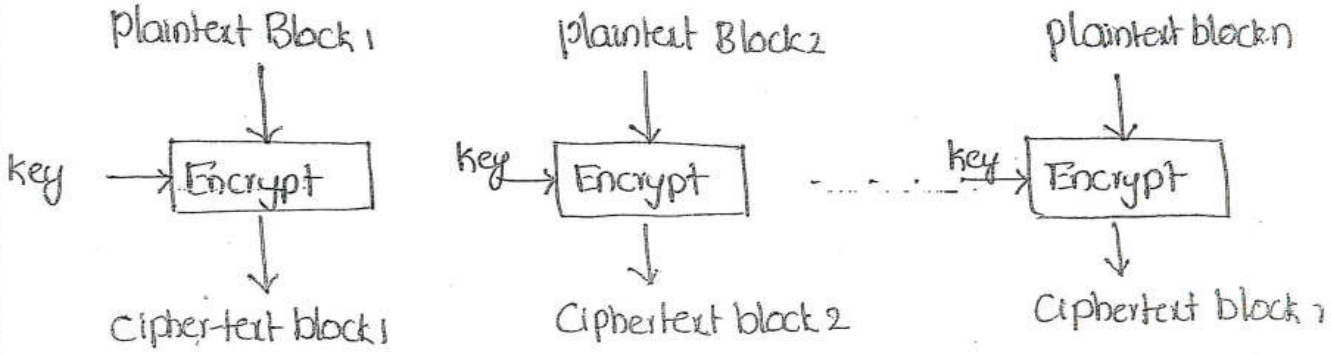


Fig: Encryption Process

→ Decryption is also performed one block at a time using the same key used in encryption

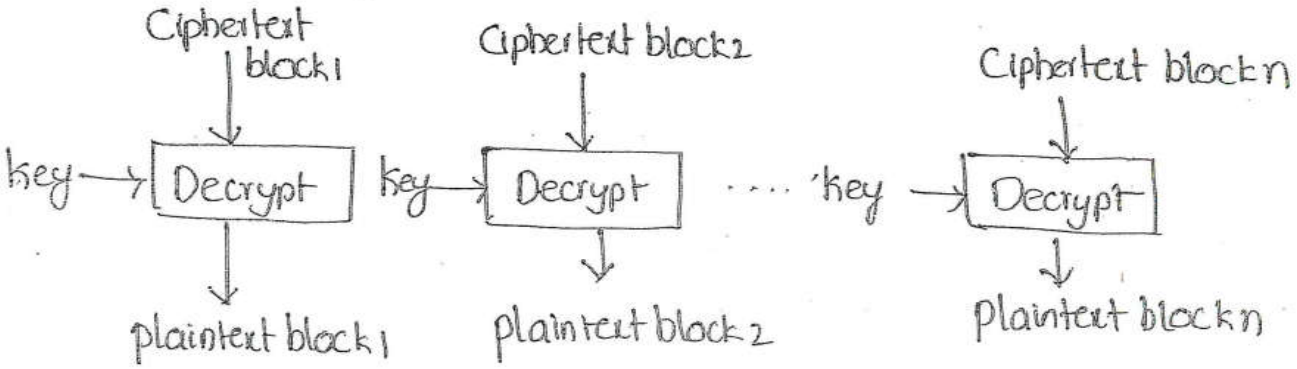


Figure: Decryption Process

Limitation:

* ECB mode is ideal for short messages and may not be secure for lengthy messages.

b) Cipher Block Chaining Mode (CBC):

→ Chaing adds Feedback mechanism to a block cipher.

→ In CBC, the plaintext is XORed with previous ciphertext block bef it is encrypted.

Encryption process:

1. XOR, Initial Vector (IV) and plaintext block1. then encrypt the resul

   Initial vector is randomly generated to make each message unique

2. XOR, plaintext block-2 with ciphertext block1, then it is encrypte using same key.

3. The process continues for all the remaining plaintext blocks of the

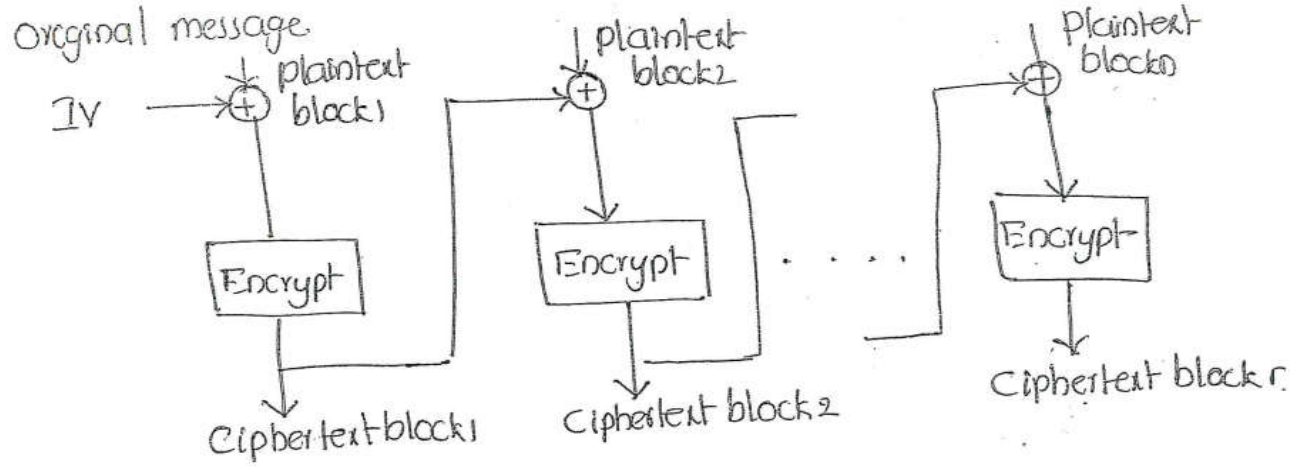


Figure: Encryption.

Decryption Process:

1. Ciphertext block1 is passed through the decryption algorithm using the same key used in encryption for all plaintext blocks. The output of this step is XORed with IV. This process produces Plaintext block1
2. The ciphertext block2 is decrypted as its output is XORed with Ciphertext block1, which can produce plaintext block 2
3. Process continues for all the ciphertext blocks in the encrypted message.

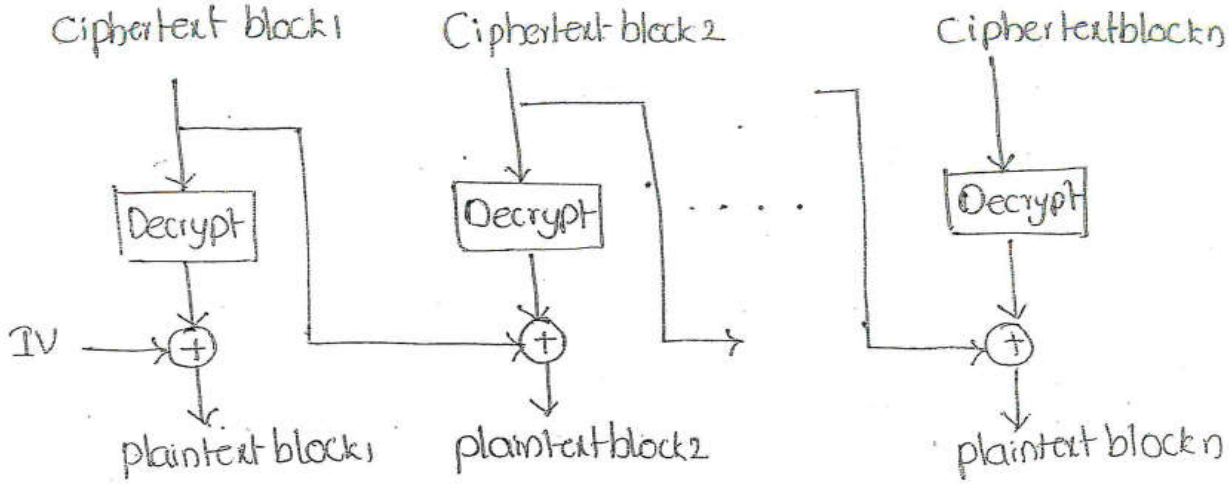


Fig: Decryption Process.

Limitations:

1. Message length must be aligned to the cipher block size.
2. Encryption is sequential.

c) Cipher Feedback Mode (CFB):

→ CFB can encrypt plaintext units that are smaller than the defined block size.

→ In this mode, the cipher text block is encrypted and the output is XORed with current plaintext block to create the current ciphertext block.

Encryption:

1. Input to the encryption function is a b-bit shift register that is initially set to IV (Initialization Vector).
2. The leftmost s-bits of output of encryption function are XORed with first segment of plaintext P1 to produce the Ciphertext bloc
3. The contents of the shift register are shifted left by s bits, and cipher text block 1 (C1) is placed in the rightmost s bits of shift register.

4. Continue the process until all plaintext units ~~are~~ have been encrypt

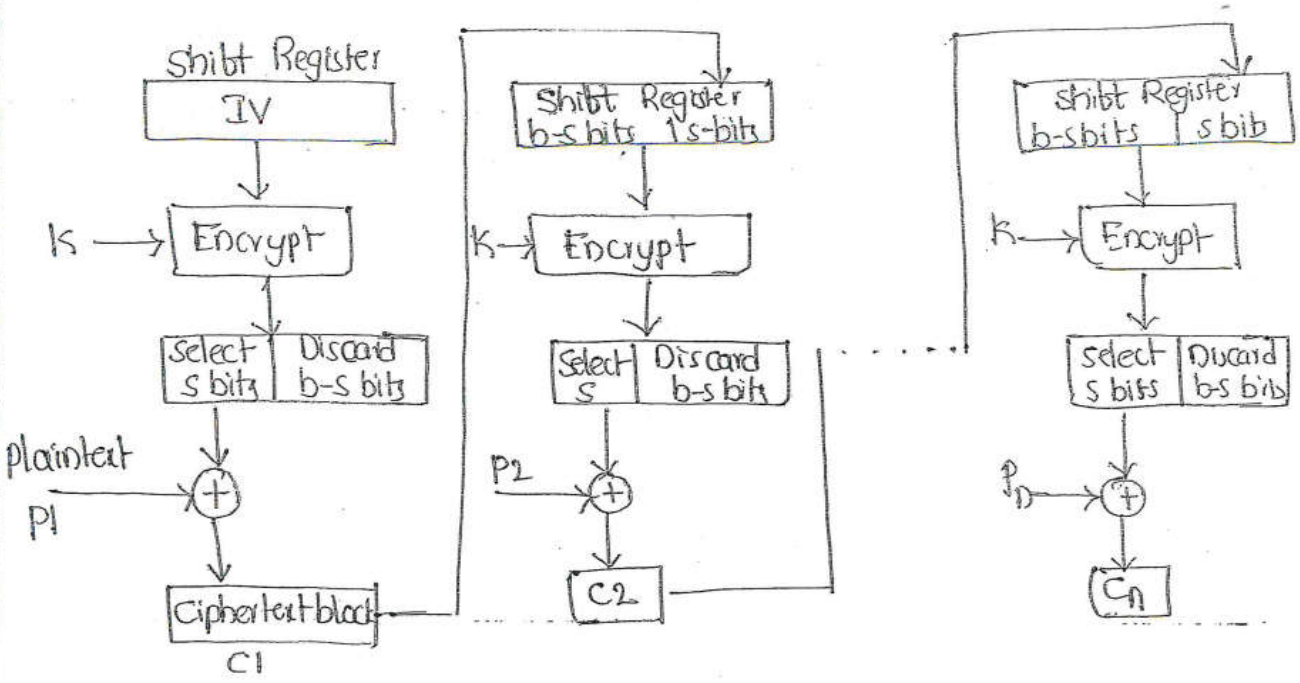


Fig: Encryption

Decryption:

Decryption uses same scheme, except the received ciphertext unit is XORed with output of encryption function to produce plaintext unit

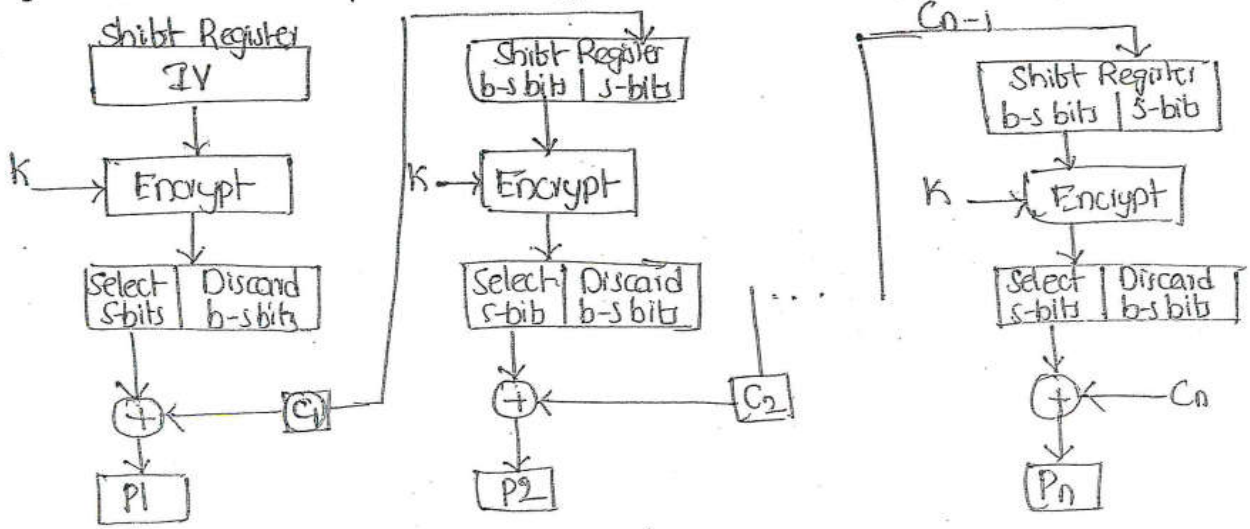


Fig: Decryption.

Limitations:

1. Error propogates for several blocks after the error.
2. Need to stall while block is encrypted after every n-bits.

c) Output Feedback Mode: (OFB)

→ OFB is similar in structure to that of CFB.

→ For OFB, output of encryption is fedback to become input for encrypting the next block of plaintext.

→ OFB operates on full blocks of plaintext and ciphertext

→ OFB requires an Initialization Vector (IV). The IV must be a nonce. That is, IV must be unique to each execution of encryption

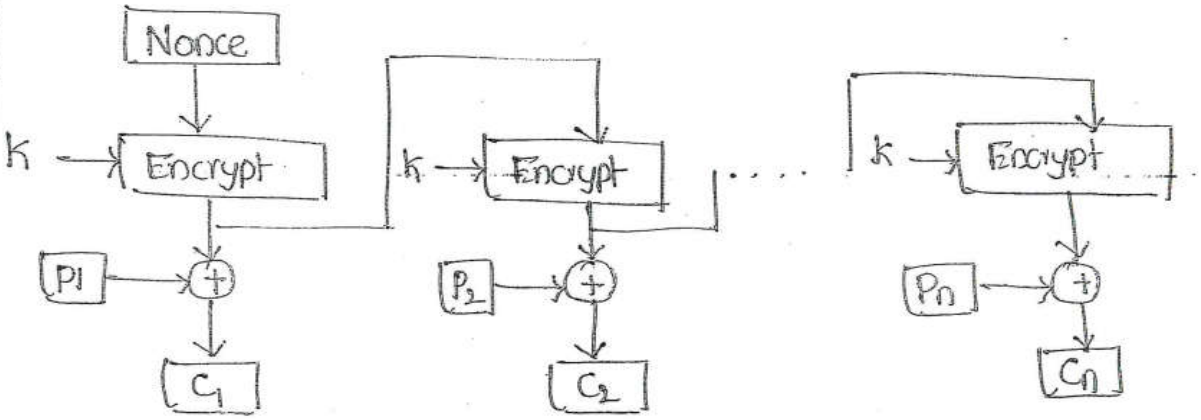


Fig: Encryption

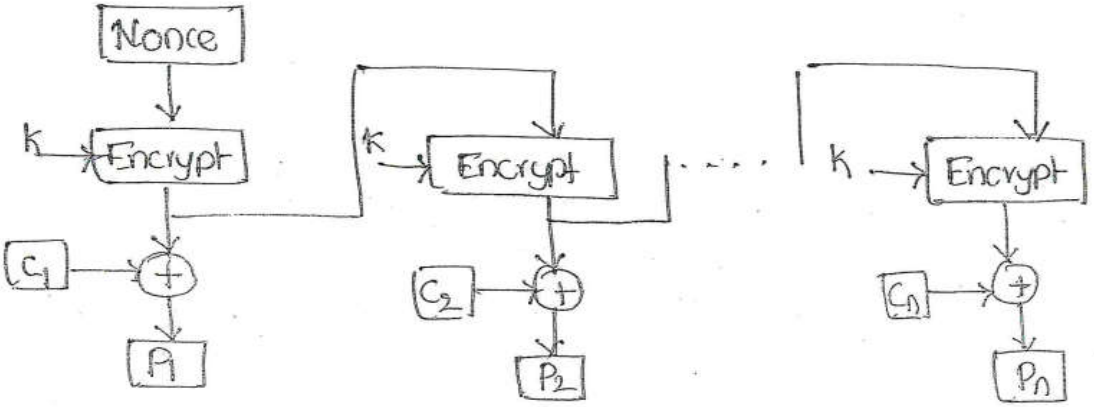


Fig: Decryption

Disadvantages:

1. OFB is more vulnerable to message stream modification attack.

e) Counter Mode: (CTR)

→ CTR mode uses a counter equal to the plaintext block size. It is initialized to some value and incremented by 1 for each subsequent block.

Encryption:

The counter is encrypted and XORed with plaintext block to produ ciphertext block.

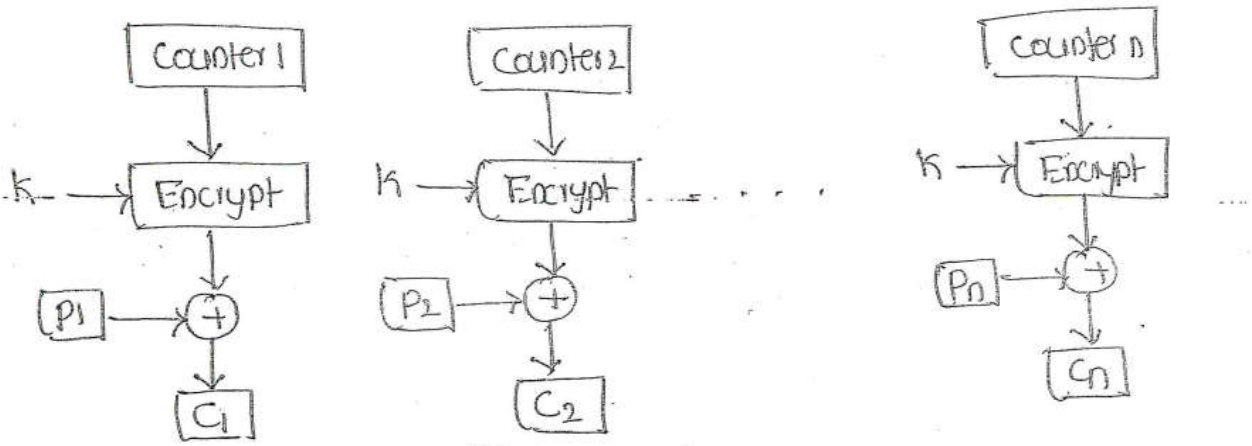


Fig: Encryption

Decryption:

Each encrypted counter is XORed with ciphertext to produce plaintext.

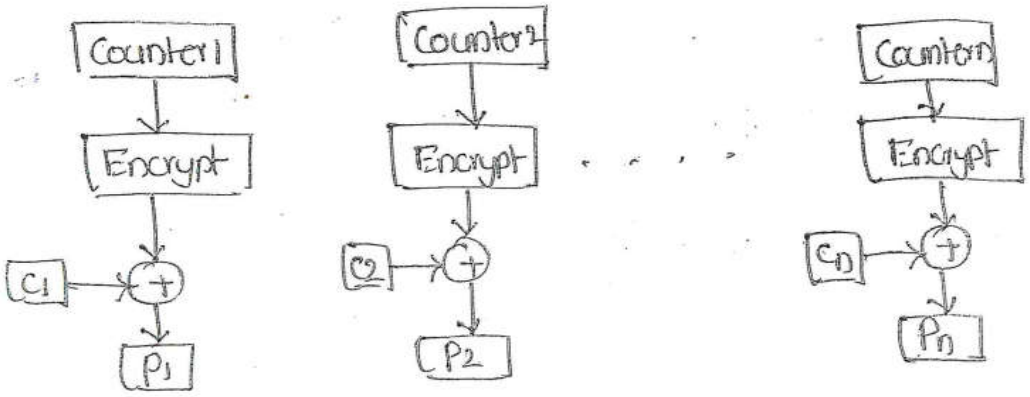


Fig: Decryption

Advantages:

1. Hardware efficiency
2. No error propagation
3. More secure
4. Simple to implement
5. Random Access.