UNIT-II

Symmetric key ciphers:-

The Encryption process can be done in 2 ways

① ~~Stream~~

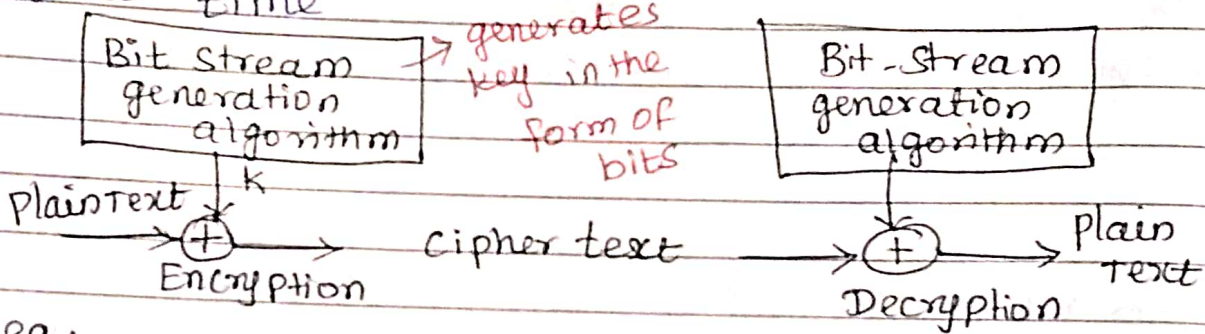
① Stream cipher

② Block cipher.

Stream cipher and Block cipher.

Stream cipher

A stream cipher is one that encrypts a digital data stream one bit or one byte at a time



eg:-

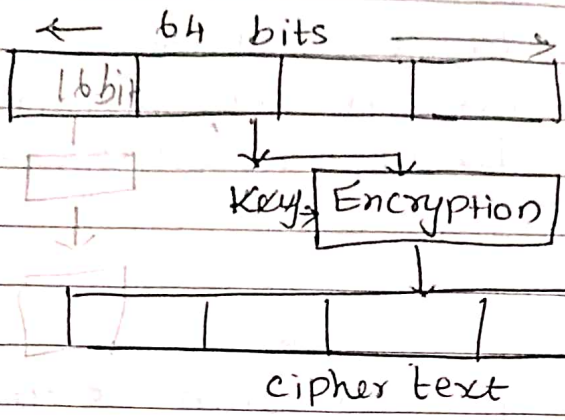
```
1 0 1 1 0 1 1 0   ← P.T
0 1 0 1 0 1 0 1   ← key
-----------------
1 1 1 0 0 0 1 1   ← C.T

1 1 1 0 0 0 1 1   ← C.T
0 1 0 1 0 1 0 1   ← key
-----------------
1 0 1 1 0 1 1 0   ← P.T
```

Block cipher:-

A Block cipher is one in which a block of plain Text is treated as a whole and used to produce a cipher text block of equal length

- Typically, a block size of 64 and 128 bits is used.



$\frac{16 \times 4}{64} = 2$

Stream Cipher	Block Cipher
(i) Type of symmetric key cipher that convert the plain text → cipher text by converting one bit / one byte of plain text at a time	(i) Type of symmetric key cipher that converts the plain text to cipher text by converting plain text Block wise at a time
ii) uses 8 bits at a time	ii) uses 64 bit or more at a time
iii) faster	iii) slower
iv) more complex	iv) less complex
v) example RC4	v) example DES, RC5, Blowfish

Diffusion and confusion.

The terms confusion and diffusion were introduced by claude Shannon.

* Shannon's concern was to prevent Crypt analysis, based on Statistical analysis.

Diffusion

The idea of diffusion is to hide the relationship between the cipher text and plain text

Diffusion means that if we change a single bit of the plaintext, then Statiscally half of the bits in the cipher text Should change.

→ Transposition

Confusion

It hides the relationship between cipher text and the key.

If a single bit in the key is changed then most / all bits of the cipher text will also be changed.

each bit of cipher text should depend on key.

→ Substitution

FEISTEL CIPHER STRUCTURE

* Most of the Block cipher technique follows this structure
* The plain text is divided in to 2 equal halves Lo and Ro
* The 2 halves of the data pass through 'N' Rounds of processing and these Combine to produce to Cipher text Block.
* On the right half we apply an function and in the function we will used a Sub key generated from the master key
* The o/p of this is XORed with the left half and then their o/p will be Swapped

This is one Single round

* We will have 'N' Rounds → depends on algorithm

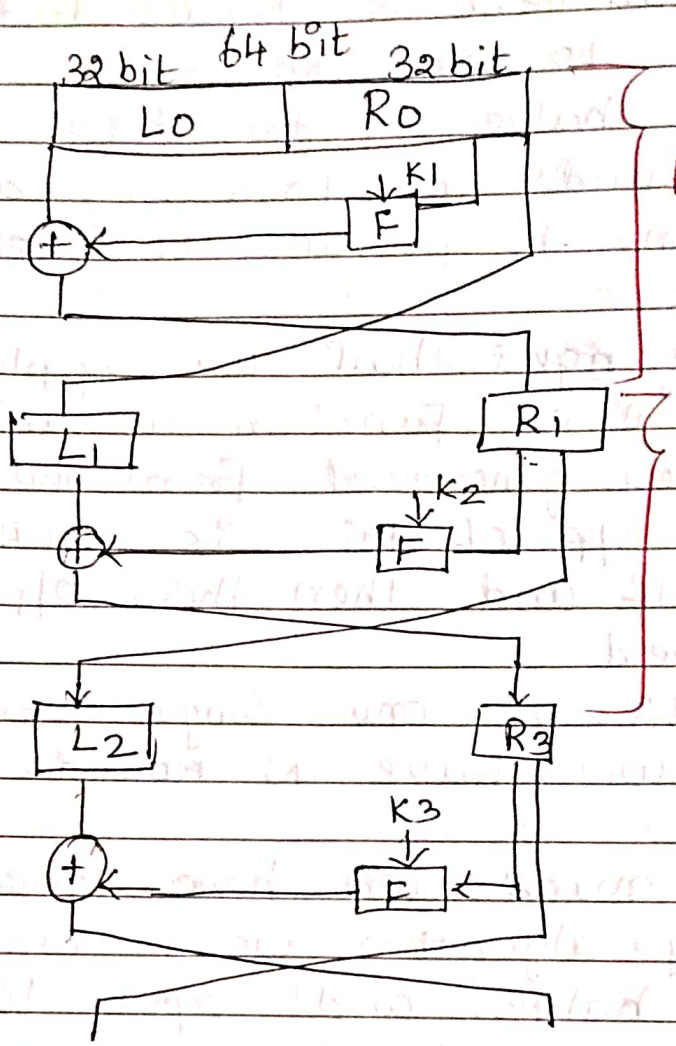
All rounds will have Same Structure.

If any algorithm we divide the plain text in 2 halves and apply the function on Right hand side and XOR with left halve and the o/p is swapped then the algorithm follows fiestel Structure.

Parameters

1. Block size → Larger block size, more security
2. Key size → Larger Key size means more Security but may decrease the Speed of encryption & Decryption
3. no of Rounds → depends on algorithm

Subkey generation algorithm → more complex
Round fn → more complex function
harder for cryptanalyst to attack.



DES Data Encryption Standard Algorithm

i) Same Algorithm and key are used for encryption & decryption

ii) The key length is 56 bits. The key originally consists of 64 bit, only 56 bit of these are actually used by algorithm.

iii) Each round performs the steps of substitution and transposition.

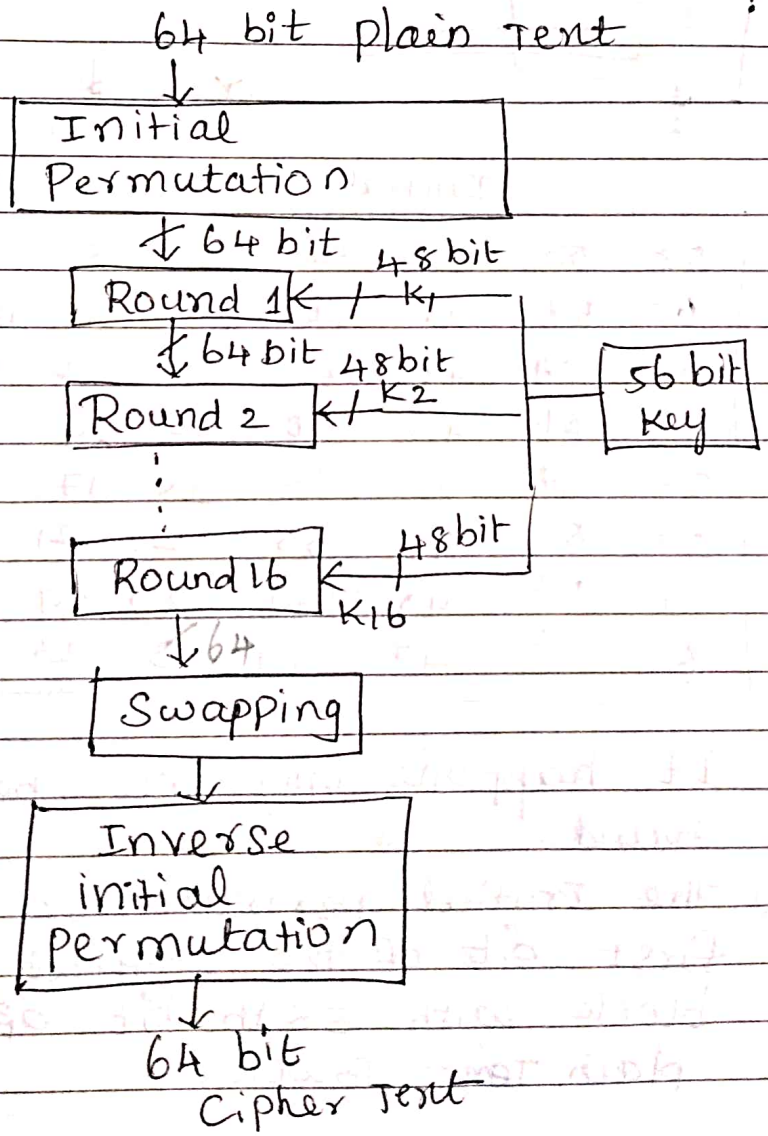
iv) 2 attributes 1. Substitution - also called confusion
2. Transposition - " diffusion

DES Data Encryption Standard

(i) Block cipher
ii) Symmetric cipher (same key for encryption & Decryption)
iii) 64 bit block (plain Text)
iv) 16 Rounds → each round is a Fiestel round

Steps
(i) Initial permutation
ii) 16 Fiestel rounds
iii) Swapping (Left & Right Swap)
iv) Final Permutation / Inverse inital Permutation

Basic structure



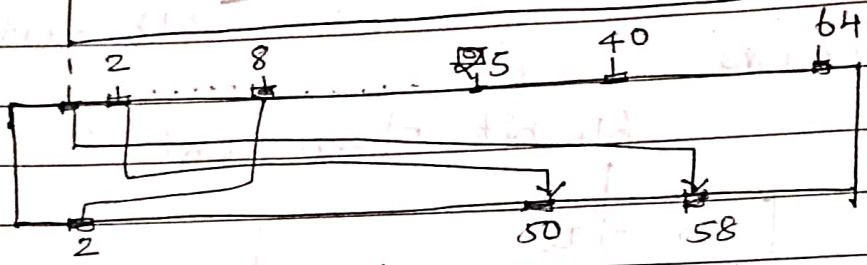
Initial Permutation (P-Boxes)

Re order the bit stream

eg: 1st bit of i/p stream is moved to 58th bit of o/p stream

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

64 bit plain Text



Initial Permutation

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

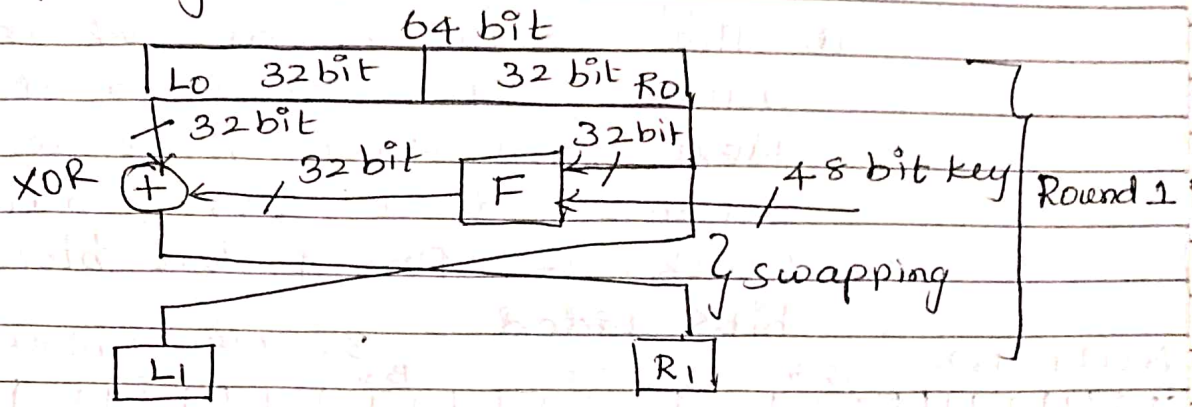
rearranged

(i) It happens only once, before the first round.

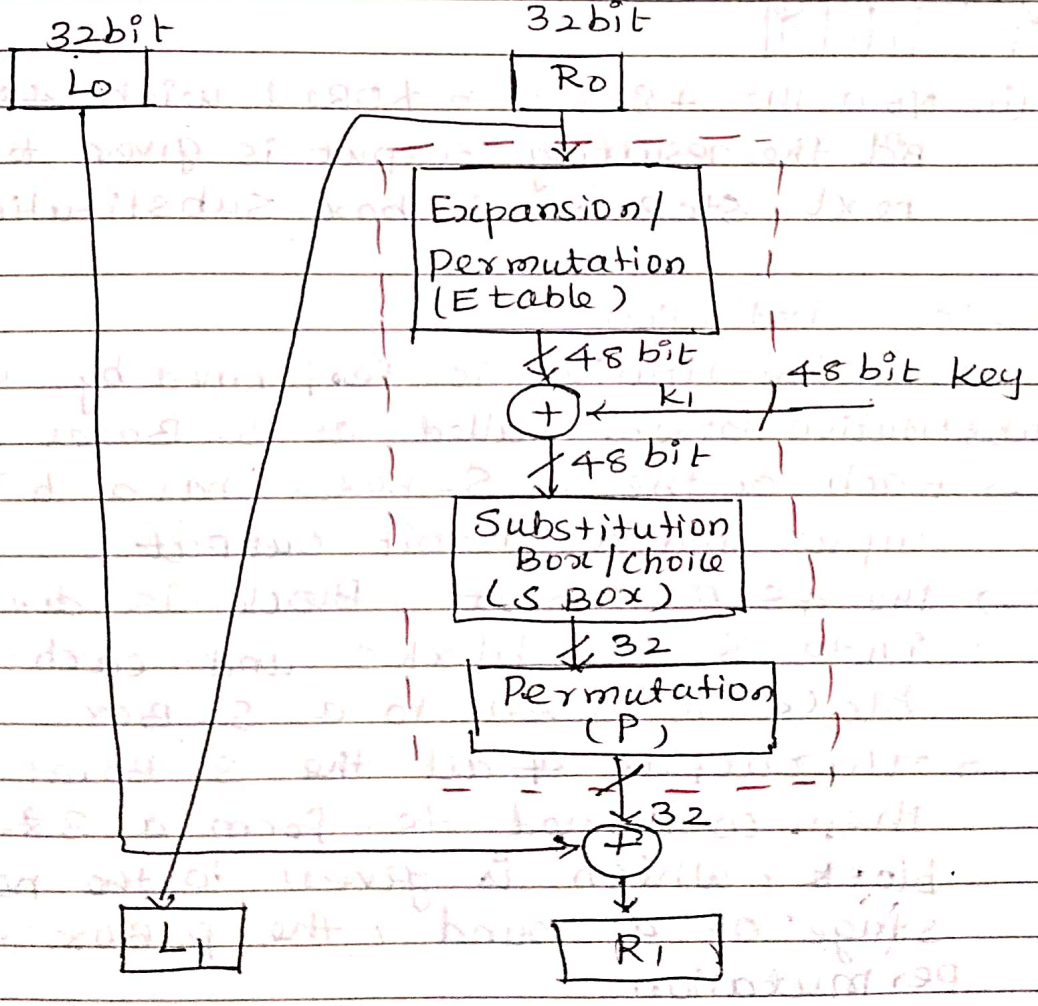
ii) The Initial permutation replaces the first bit of the original plain text block with 58th bit of the original plain Text Block.

iii) After Initial permutation is done the resulting 64 bit permuted block is divided into two 32 bits blocks.

Details of Single Round



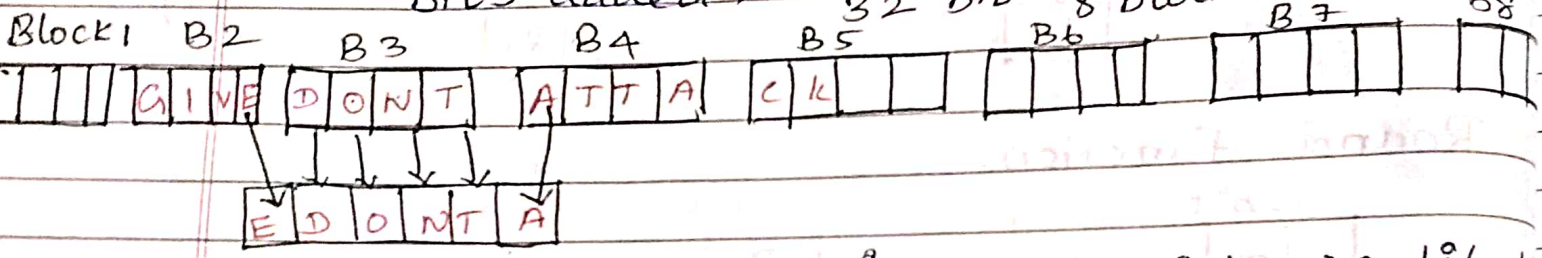
Round Function



Expansion permutation

(i) During Expansion permutation, The Ro input is 32 bits. This Ro input is first expanded to 48 bits. This happens as follows

(a) The 32 bit is divided in to 8 blocks, each consisting of 4 bits Next, each 4 bit block is then expanded to a corresponding 6-bit block. ie Per 4 bit block, 2 more bits added.



(ii) Now the 48 bit is XORed with 48-bit k and the resulting output is given to the next step ie S-box substitution

S-Box Substitution

The Substitution is performed by 8 Substitution Boxes called as S-Boxes

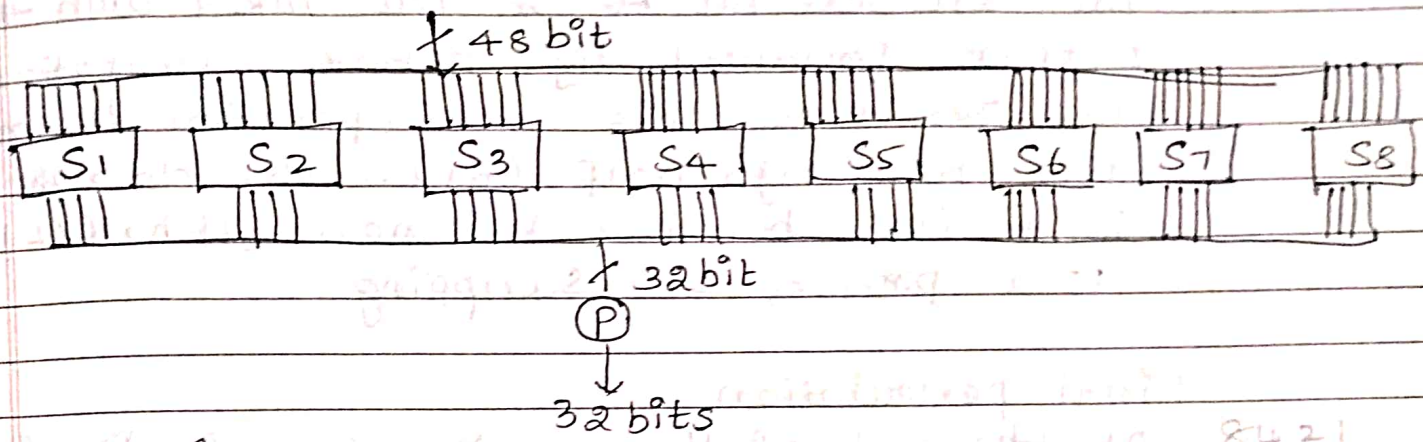
→ Each of the 8 S-Boxes has a 6 bit input and a 4-bit output

→ The 48 bit input Block is divided in to 8 sub blocks and each such block is given to a S-Box.

→ The output of all the S-Boxes are then combined to form a 32-bit block, which is given to the next stage of a round, the P-Box permutation

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



example

let 6 bit i/p ⇒ 001011

First and Last bit } 01 ⇒ 1 (Row)

middle bit } 0101 = 5 (col)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S_1 0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

First row and 5 col intersection = 2

0010//.

P- Box permutation

The output of S-Box consists of 32 bits, these are permuted using a P-Box

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

The left side Bit Lo is now XORed with the output produced by P-Box permutation. The result of this XOR operation becomes the new right half (R1). The old right half (R0) becomes the new left half (L1) is a process of swapping.

Final permutation

→ At the end of the 16 rounds, the Final permutation is performed (only once)

→ This is a simple Transposition.

→ the o/p of final Permutation is the 64 bit cipher text

Key generation

→ Each round, a 56 bit is available

→ The key originally consists of 64 bits, however only 56 of these are actually used by the algorithm. Eight bits are used solely for checking parity and are discarded. Hence the effective key is 56 bits.

→ From this 56 bit key, a different 48 bit subkey is generated during each round
→ This 56 bit key is divided into 2 halves each of 28 bits.
→ These halves are circularly shifted left by one or two positions depending on the round

input key

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

Permuted choice one (PC-1)

57	49	41	33	25	17	9	C0
1	58	50	42	34	26	18	
10	2	59	60	43	35	27	
19	11	3	51	52	44	36	
63	55	47	39	31	23	15	D0
7	62	54	46	38	30	22	
14	6	61	53	45	37	29	
21	13	5	28	20	12	4	

28 bit

28 bit

PC-2 48 bit

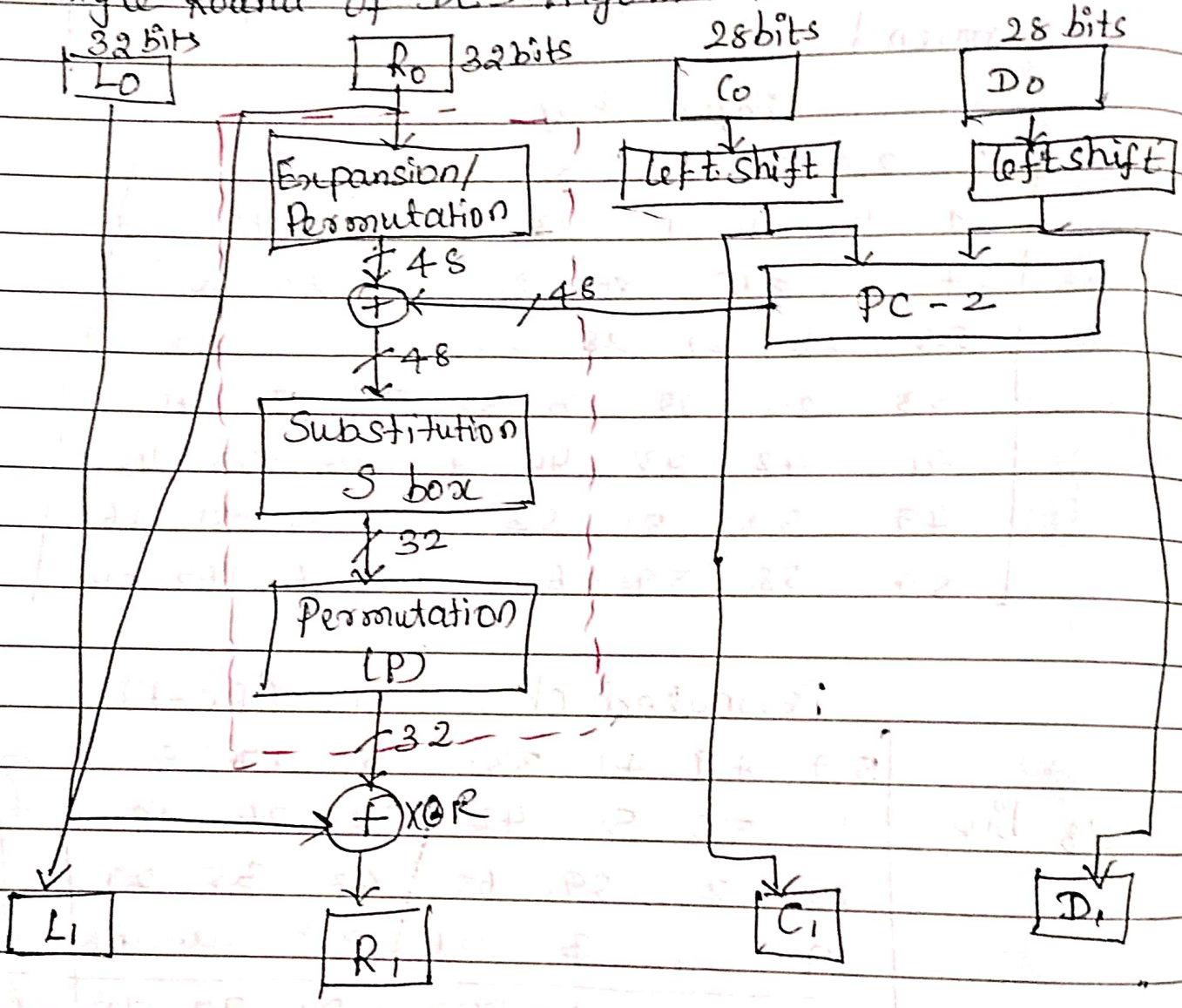
14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

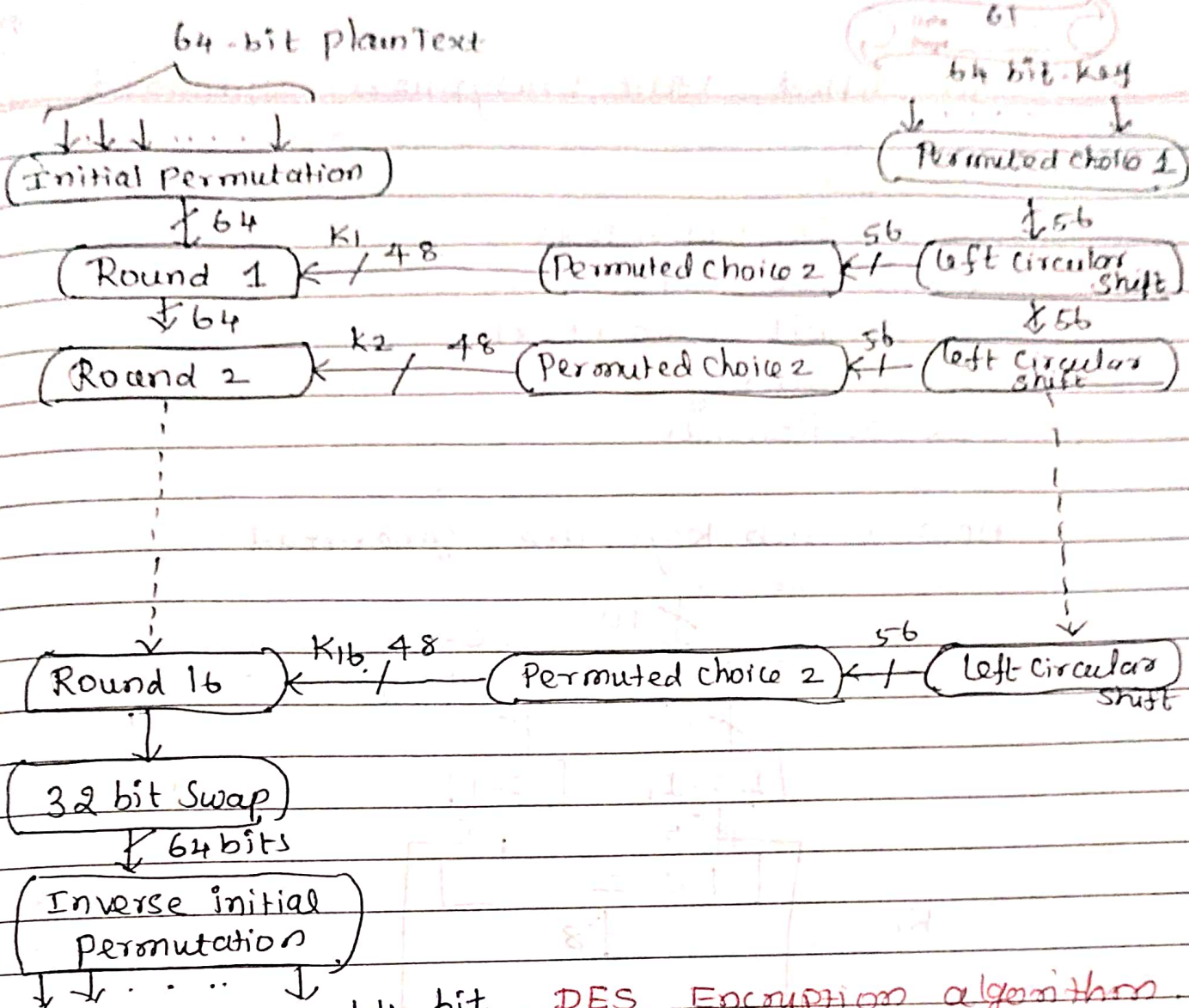
Schedule of left shifts

Round no :- 1, 2, 9, 16 → 1 Bits Rotated Rotated

Remaining Round → 2 " Rotated

Single Round of DES Algorithm.





DES Encryption algorithm.

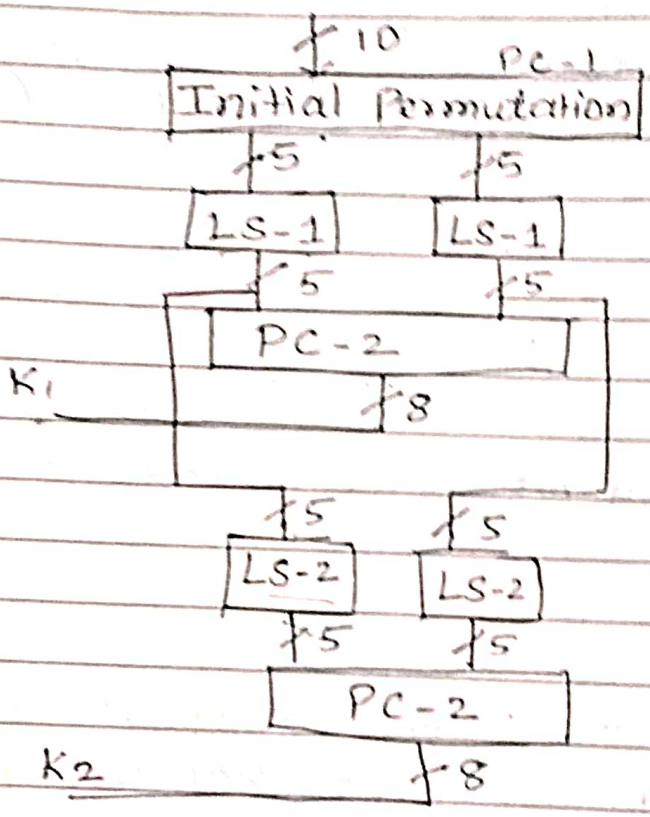
DES DEcryption

As with any Feistel cipher, decryption uses the same algorithm as Encryption, except that the application of the subkeys is reversed

Simplified Data Encryption Standard

S-DES
- 8 bit plain text
- 10 bit key
- 8 bit ciphertext
- 2 SubKeys
- 2 Rounds.

How 2 Sub Keys are generated



$R_1 = 1$
$R_2 = 2$

10
↓
8

Key = 1 0 1 0 0 0 0 0 1 0
1 2 3 4 5 6 7 8 9 10 → Position

plot $\frac{3}{2}$ K_1

Ip(Pc-1) = { 3 5 2 7 4 10 1 9 8 6 }

After = 1 0 0 0 0 0 1 1 0 0
P8-1

5 bit 5 bit

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

Round 1 left Circular shift by 1 bit

1 0 0 0 0 2 | 0 1 1 0 0 2

After left shift 1: 0 0 0 0 1 | 1 1 0 0 0

1 2 3 4 5 | 6 7 8 9 10 – Position

PC2 = 26 37 48 5 10 9

K1 = 1 0 1 0 0 1 0 0 → 8 bit K1

K2 After left shift 1

0 0 0 0 1 | 1 1 0 0 0

Round 2 left Circular shift by 2 bit

0 0 0 0 1 2 | 1 1 0 0 0 2

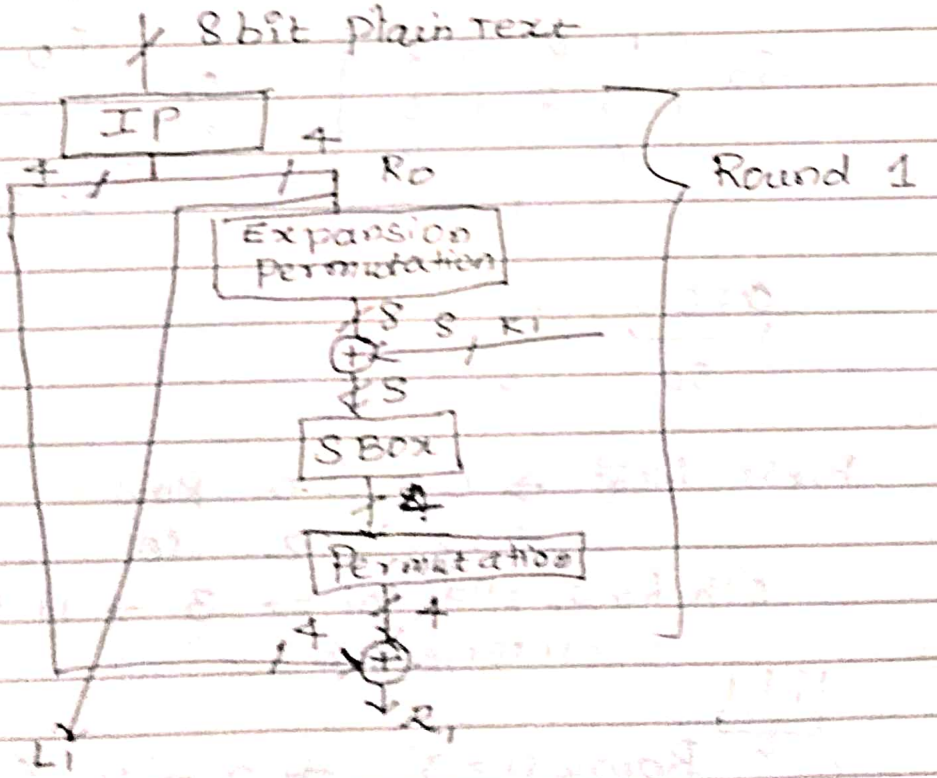
0 0 1 0 0 | 0 0 0 1 1

1 2 3 4 5 | 6 7 8 9 10

PC2 = 26 37 48 5 10 9

K2 = 0 1 0 0 0 0 1 1

Encryption:-



Encryption

8 bit plain text

P.T = 1 0 0 1 0 1 1 1
1 2 3 4 5 6 7 8

IP = {2, 6, 3, 1, 4, 8, 5, 7}

After IP = 0 1 0 1 1 1 0 1
L_0 R_0

EP = 4 bit to 8 bit

EP = {4 1 2 3 2 3 4 1}

Right (R_0) = 1 1 0 1
1 2 3 4

After EP = 1 1 1 0 1 0 1 1

K_1 = 1 0 1 0 0 1 0 0 ⊕

After XOR = 0 1 0 0 1 1 1 1

S0, S1 → 2 boxes

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

4 × 4

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

0 1 0 0 | 1 1 1 1
S0 S1

S0

First, Last → 00 ⇒ 0 Row
→ 10 = 2 Col

0th Row, 2nd Col ⇒ 3 → 11 (2 bit o/p)
intersection

S1

1 1 1 1

Row = 11 = 3 ⇒ 3 ⇒ 11 (2 bit o/p)
Col = 11 = 3

After S Box = 1111

Permutation = {2 2 4 3 1}

= 1111

XOR with 0101 (L0)

1010 (R1)

$L_1 \Rightarrow 1101$ $R_1 = 1010$

EP = {4 1 2 3 2 3 4 1}

R_1 = 1010

1 2 3 4

After EP = {0 1 0 1 0 1 0 1}

K_2 = 01000011 ⊕

After XOR = 00010110

$S_0, S_1 \rightarrow$ 2 boxes

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

0001 0110

S_0 S_1

S_0 0001

→ 01 = Row

00 = col

O/P 3 ⇒ 11

S_1 0110

→ 00 = Row = 0

11 = Col = 3

O/P 3 = 11

1111

Permutation = {2 4 3 1}

= 1111

XOR = 1101 (L0)

0010 (R1)

1010 0010

1 2 3 4 5 6 7 8

Swapping

~~C.T = 00101000~~

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

0010 1010

1 2 3 4 5 6 7 8

C.T 00111000

Decryption
Reverse process.

DES Analysis:-
properties
(i) Avalanche effect
(ii) Completeness effect

Avalanche effect

It means a small change in plain Text (or key) should create a significant change in cipher text.

Same key { plain Text 1 → Cipher Text 1 ; plain Text 2 → cipher Text 2] differ only by one bit around 29 bits are different

eg

plain Text 1 = 0000000000
Cipher Text 1 = 4789FD476E

Key = 2223451298

plain Text 2 = 0000000001
Cipher Text 2 = 0A4ED5C15A

Although the 2 plaintext blocks differ only in the right most bit, ~~the cipher text~~ there is a significant change in ciphertext blocks.

DES has been proved to be strong with regards to this property

completeness effect

It means that each bit of cipher text needs to depend on many bit on plain Text.

The confusion and Diffusion produced by P-Boxes and S-boxes in DES show a very strong completeness effect

Multiple DES

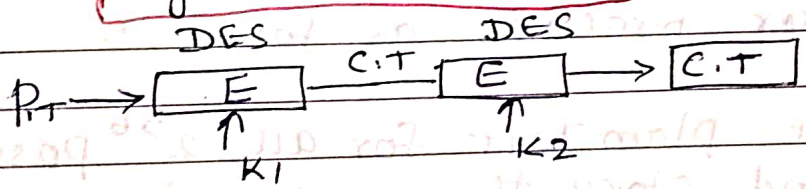
1. Double DES (2 DES) — Triple DES with 2 keys
2. Triple DES (3 DES) — " " " 3 "

Sine DES attack was vulnerable to brute force attack, variations of DES called Multiple DES were introduced.

1. Double DES (2 DES)

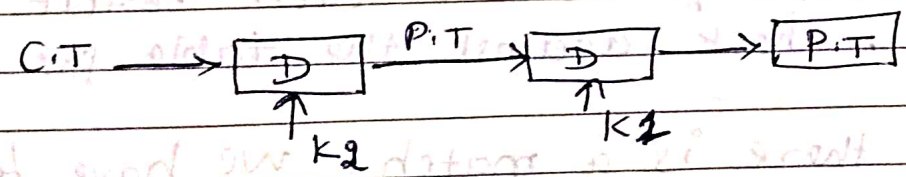
The simplest form of multiple encryption has 2 encryption stages and 2 keys.

Key = 56 x 2 = 112 bits.



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

Decryption



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

→ Double DES uses 2 keys k_1 and k_2
→ It performs DES on the original plain text using k_1 to get the encrypted text
→ It again performs DES on the encrypted text, but this time with other key k_2.
→ The original plain text encrypted twice with 2 different keys.

Draw back of ~~Dobad~~ Double DES.

Meet-in-the-middle attack. (MIM attack)

This attack involves encryption from one end and decryption from the other end and then "matching the results in the middle" and hence the name.

This attack requires knowing some plain text cipher text pairs.

let us assume plain text = P
cipher text = C.

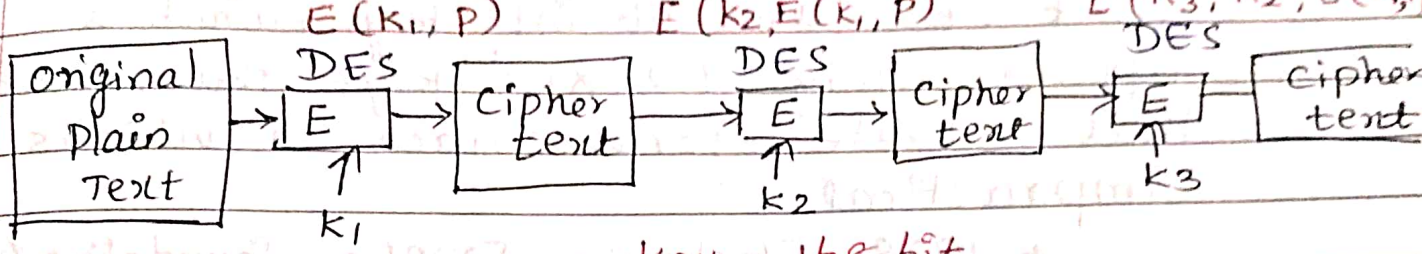
The attacker proceeds as follows :-

(i) encrypt plain text for all 2^{56} possible values of K_1 and store the results in a table and sort it.
(ii) Now decrypt cipher text using all 2^{56} possible value of k_2. As each result is produced, check against the table for a match.
iii) When there is a match, we have located a possibly correct pair of keys.

Triple DES with 3 keys.

→ The plain text is first encrypted with a key K_1, then encrypted with a second key K_2, and finally a third key, K_3.

→ Where K_1, K_2, K_3 are all different from each other.



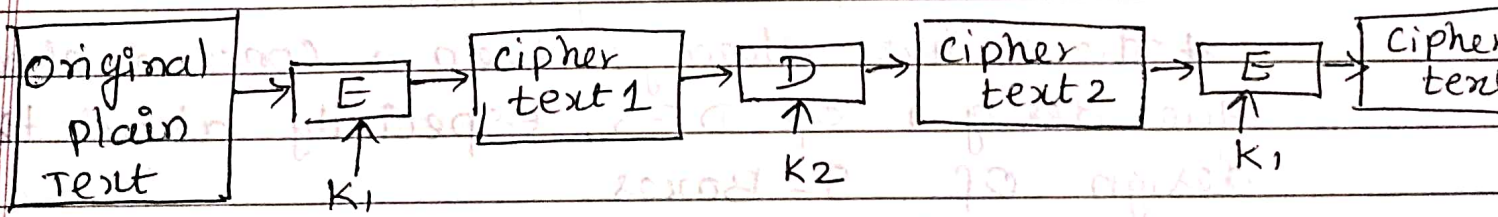
Key = 168 bit.

Triple DES with Two keys

1. Encrypt the plain text with K_1 $E(K_1, P)$
2. Decrypt the output of step 1 with K_2. $D(K_2, E(K_1, P))$
3. Finally encrypt the output of step 2 again with K_1.

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

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



The Strength of DES:-

Two main concerns with DES

① The length of the key

② The nature of the algorithm.

The length of the key

* DES uses 56-bit keys, which is approximately 7.2×10^{16} keys Thus, it seems that a brute-force attack on DES is impractical.

* 1998. Electronic Frontier foundation (EFF) announced that it had broken a new DES challenge using a special purpose "DES Cracker" machine that was built for less than $250,000.

* The attack took less than three days

* Hardware prices will continue to drop as speed increase, making DES worth less

* Fortunately, there are a no of alternative available in the market place.

The nature of the algorithm

* There has always been a concern about the design of DES, especially about the design of S-Boxes.

* Break it without having to search for the key

* The design criteria for the S-Boxes have been classified information

* Changing the S-Boxes slightly seems to weaken the algorithm.

Timing Attacks:-
* Attacks actual implementation of cipher
* use knowledge of consequences of implementation of derive knowledge of some / all subkey bits
* calculations can take varying times depending on the value of the inputs.

DES Weaknesses

① Weaknesses in S-Boxes.
Two specifically chosen inputs to an S-box can create same output
② Weaknesses in P-Boxes
Initial and Final permutations have no security benefits.
③ Weaknesses in key
Weak keys create same 16 round keys
key complement.

Differential and Linear cryptanalysis

Linear cryptanalysis.
⇒ Introduced by Matsui
⇒ The basic idea
⇒ TO find relations among certain bits of plaintext, ciphertext and key
⇒ In Linear cryptanalysis, the role of cryptanalyst is to identify the linear relation b/w some bits of plaintext

Some bits of the cipher text and some bits of the unknown key.

⇒ Linear cryptanalysis focus on statistical analysis against one round of decrypted cipher text

⇒ The attacker has a lot of plain text - cipher text pairs (Known plain text attack)

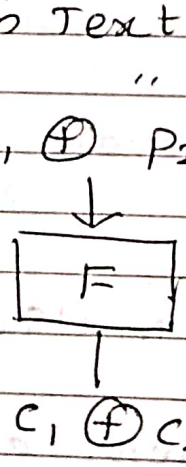
Differential cryptanalysis.

⇒ Differential attack is a chosen-plain Text attack

⇒ Differential analysis focuses on Statistical analysis of two inputs and two outputs of a cryptographic algorithm

⇒ input / output XOR difference.

⇒ differential attack involves comparing the XOR of two inputs to the XOR of the corresponding outputs.



$C_1 \oplus C_2 = P_1 \oplus P_2$

TO Find C_1 C_2

$P_1 \oplus K$ $P_2 \oplus K$

$C_1 \oplus C_2 = P_1 \oplus K \oplus P_2 \oplus K = P_1 \oplus P_2$

Block Cipher Modes of operation

Block Cipher Modes
- → Electronic Code Book (ECB)
- → Cipher Block chaining Mode (CBC)
- → Cipher Feedback mode (CFB)
- → OFB (output Feedback mode)
- → Counter Mode (CTR)

For different types of Messages, we need different Modes of operations.

Electronic Code Book (ECB)

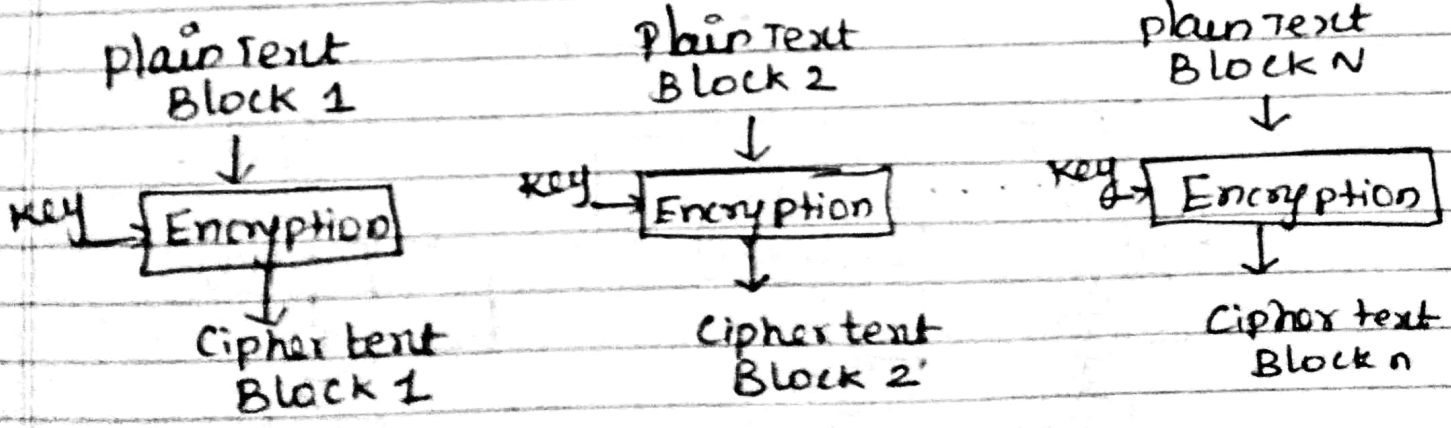
* The simplest mode of operation
* Plain Text Message is divided into Blocks of 64 bits each
* Block Size depends based on Algorithm
* Each such block is encrypted independently of the other blocks
* For all blocks same key is used for encryption
* Suitable only for small messages

eg

Block Size = 5

plain Text → Hello everyone

Hello | Every | oneXX → Padding



for decryption process

key
↓
Cipher Block 1 → Decryption algorithm → plaintext Block 1

this will happen for every Block.

disadv

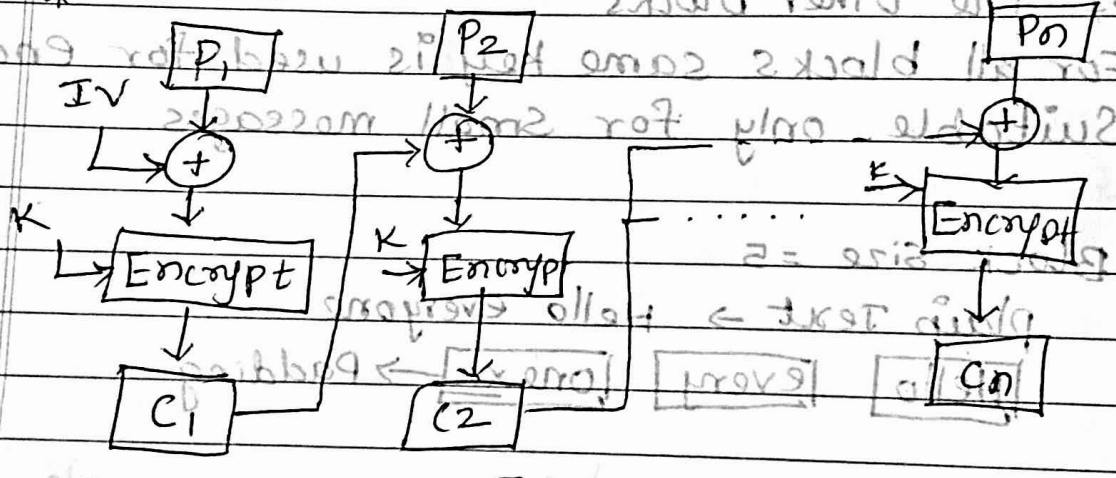
if identical blocks appear, then this mode produces same cipher.

Cipher Block Chaining Mode (CBC)

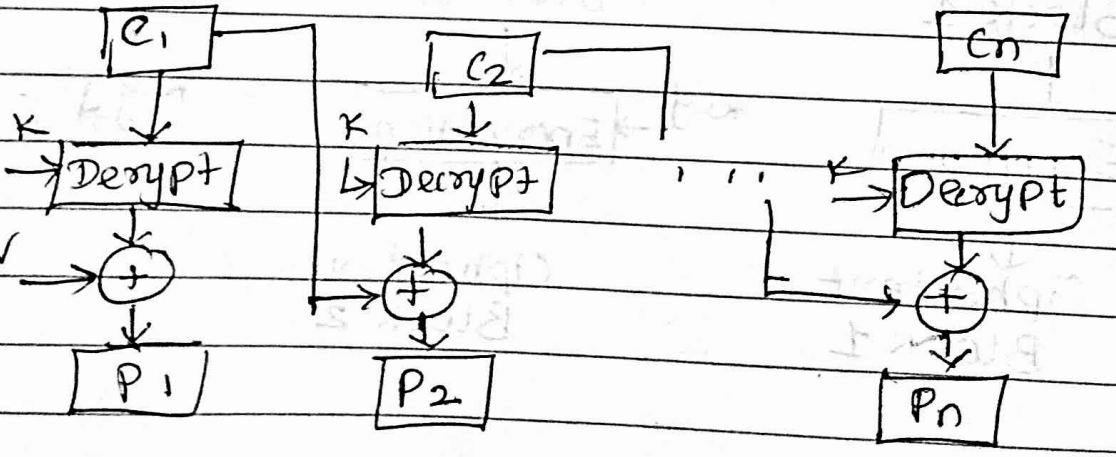
* TO overcome Security issues of ECB Mode (in ECB if same blocks appear then cipher text produced will be same)

Input to the encryption algorithm is XOR of the current plain text block and the preceding cipher text block; the same key is used for each block.

*



Encryption



Decryption

* In the first step; the first block of plain Text and a random block of text called Initialization Vector (IV) is used.
* The value of IV is generated randomly.
* IV must be known to both parties, but should be unpredictable by the 3rd parties

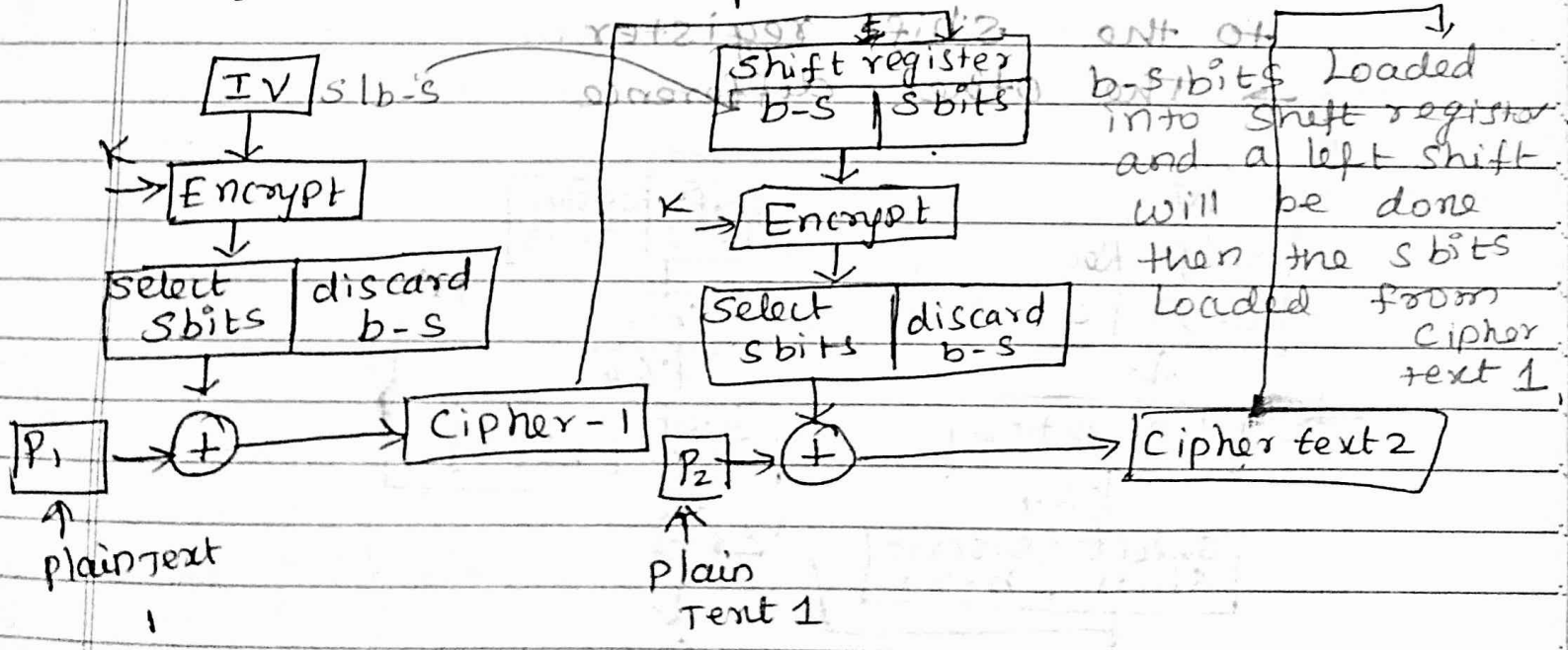
Limitation :- If we have 2 identical messages & if we use same IV, cipher will be same

Cipher feedback mode (CFB)

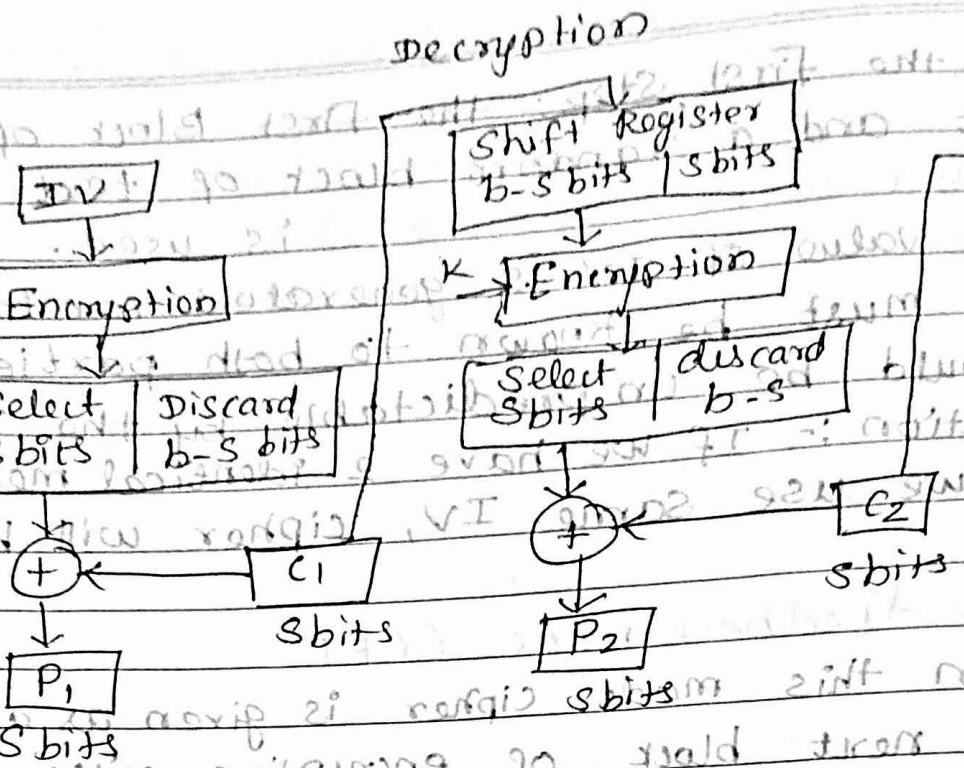
In this mode cipher is given as a feedback to the next block of encryption with some new specifications:-

→ 1st an IV (b bits) is used for 1st encryption and the o/p bits are divided as set of 's' and (b-s) bits

→ The left hand side 's' bits are selected and are applied an XOR operation with the plain text bits. The result is given to a shift register and the process continues.



Encryption

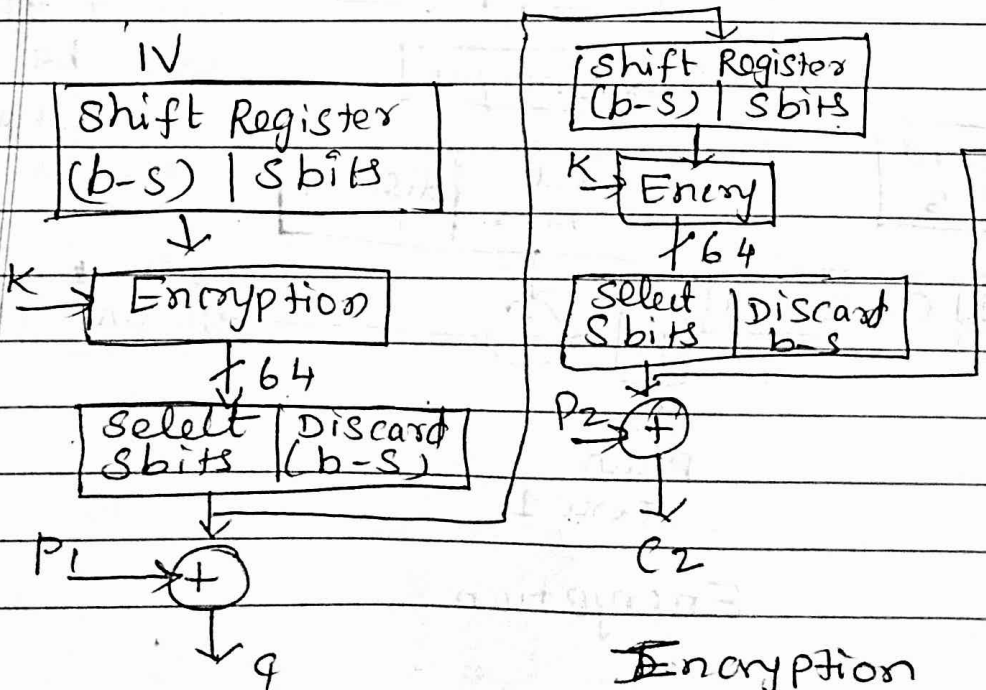


Output feedback (OFB) Mode

The OFB mode is similar in structure to that of CFB.

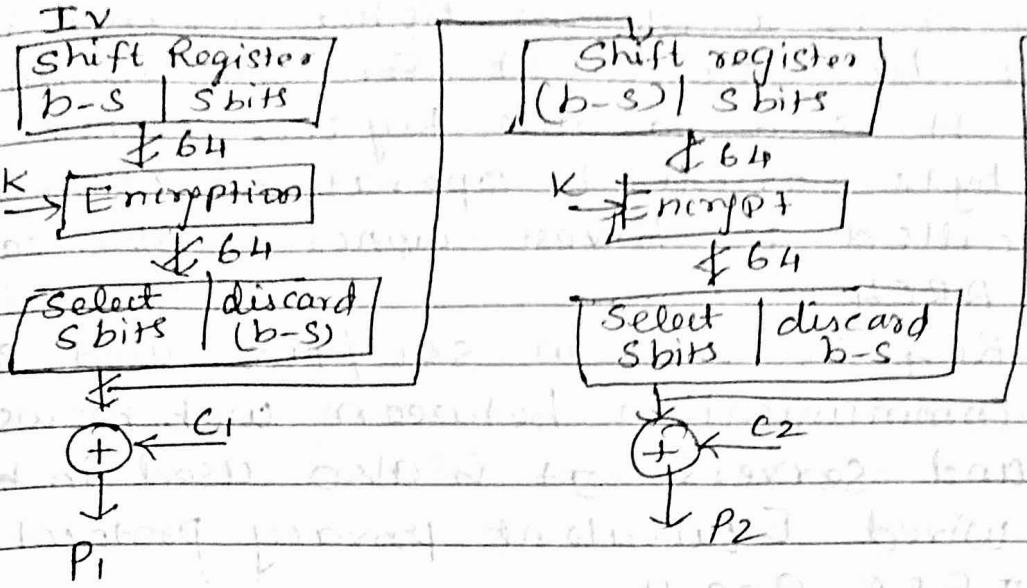
Its that output of the Encryption Function that is fed back to the Shift register in OFB, whereas in CFB, the ciphertext unit is feedback to the shift register.

→ The other difference



Encryption

Decryption



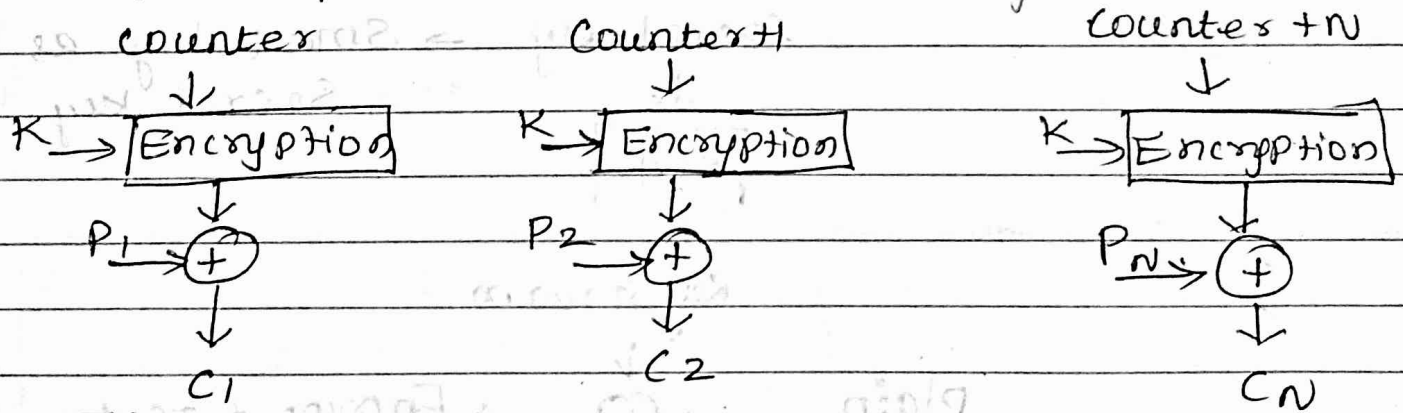
* Sender and receiver must remain in sync

Counter mode

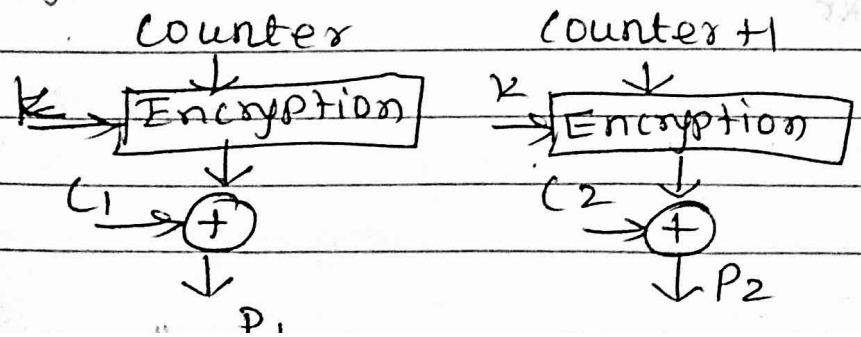
→ Uses sequence numbers called as counters as the inputs. usually a constant is used as the initial counter value

→ Incremented for every iteration.

→ The size of the counter block is same as the of the plain text block Encryption



Decryption



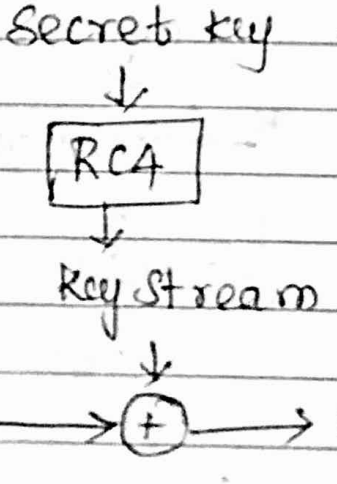
Stream cipher

RC4

* It is a Stream cipher designed in 1987 by Ron Rivest For RSA security.
* It is a variable keysize cipher with byte-oriented operations. It is also called as Rivest cipher 4. Also called ARC4
* RC4 is used in SSL/TLS, used for communication between web browsers and servers. It is also used in WEP Wired Equivalent privacy protocol in IEEE 802.11
* Symmetric Key Encryption
* 64 bit & 128 bit key sizes
* Strong and Easy.

It consists of 2 parts ① Key scheduling Algorithm (KSA) ② Pseudo-random generation Algorithm (PRGA)

RC4 Block diagram



RC4 inside
2 parts
① KSA
② PRGA

KSA
→ Generate state array

PRGA
→ Generate key stream
→ XOR Key stream with the data to generated encrypted stream

Key Scheduling algorithm (KSA)
To generate the key stream, the cipher makes use of a secret internal state which consists of 2 parts (S)
→ A permutation of all 256 possible bytes
→ Two 8 bit index pointers (denoted i & j)

It uses a state array S with 256 entries S[0], S[1], S[2] ... S[255]

S[0] = 0
S[1] = 1
⋮
S[255] = 255

Algorithm consist of 2 steps

Step 1

```
for i = 0 to 255 do
  S[i] = i
  T[i] = K[i mod (|K|)];
```

T → A Temporary Vector
K → Array of bytes of secret key
|K| → Key length

After initialising S, we need to Permute it and make it key dependent

Step 2 Permutation

```
j = 0;
for i = 0 to 255 do
    j = (j + S[i] + T[i]) mod 256
    Swap (S[i], S[j])
```

After Permuting the key dependent S is used for Stream Generation

PRGA Pseudo code

```
i = j = 0;
while (true)
    i = (i+1) mod 256;
    j = (j + S[i]) mod 256;
    Swap (S[i], S[j]);
    t = (S[i] + S[j]) mod 256
    K = S[t];
```

RC4 Encryption Example

lets Consider the Stream cipher RC4 but instead of the full 256 bytes, we will use 8 x 3 bits

That is the state Vector S is 4 x3 bit We will operate on 3 bits of plain text at a time Since S can take the value 0 to 7, which can be represented as 3 bit

P = {1, 2, 2, 2} K = {1 7 1 7}

k = {1, 7, 1, 7}

AES

AES is a block cipher intended to replace DES for commercial applications. It uses a 128-bit block size and a key size of 128, 192, or 256 bits.

AES does not use a Feistel structure. Instead, each full round consists of four separate functions: byte substitution, permutation, arithmetic operations over a finite field, and XOR with a key

Advanced Encryption Standard (AES) is a specification for the encryption of electronic data established by the U.S National Institute of Standards and Technology (NIST) in 2001. AES is widely used today as it is a much stronger than DES and triple DES despite being harder to implement.
Points to remember

- AES is a block cipher.
- The key size can be 128/192/256 bits.
- Encrypts data in blocks of 128 bits each.

That means it takes 128 bits as input and outputs 128 bits of encrypted cipher text as output. AES relies on the substitution-permutation network principle which means it is performed using a series of linked operations that involves replacing and shuffling the input data.

Working of the cipher :
AES performs operations on bytes of data rather than in bits. Since the block size is 128 bits, the cipher processes 128 bits (or 16 bytes) of the input data at a time.

The number of rounds depends on the key length as follows :

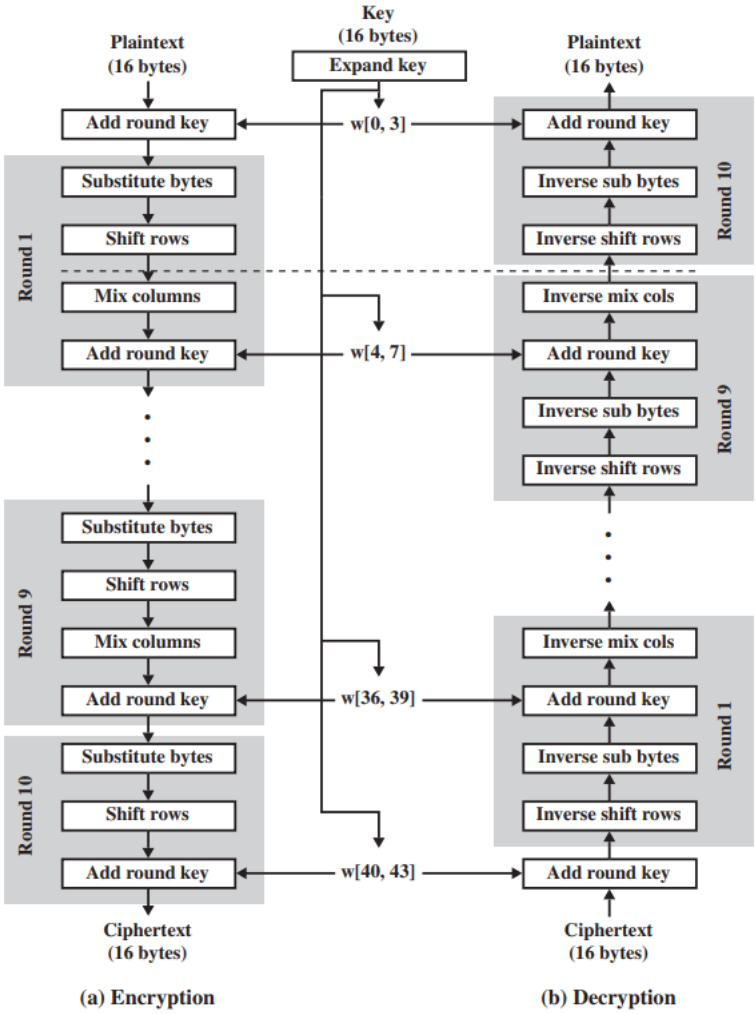
- 128 bit key – 10 rounds
- 192 bit key – 12 rounds
- 256 bit key – 14 rounds

Creation of Round keys :
A Key Schedule algorithm is used to calculate all the round keys from the key. So the initial key is used to create many different round keys which will be used in the corresponding round of the encryption.
Encryption :
AES considers each block as a 16 byte (4 byte x 4 byte = 128) grid in a column major arrangement.
[b0 | b4 | b8 | b12 |
| b1 | b5 | b9 | b13 |
| b2 | b6 | b10| b14 |
| b3 | b7 | b11| b15]
Each round comprises of 4 steps :

- SubBytes
- ShiftRows
- MixColumns
- Add Round Key

The last round doesn’t have the Mix Columns round.

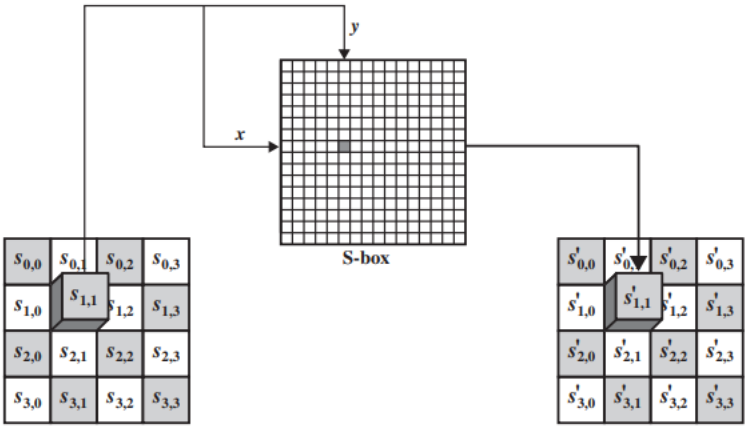
The Sub Bytes does the substitution and Shift Rows and Mix Columns performs the permutation in the algorithm.



SubBytes :

This step implements the substitution.

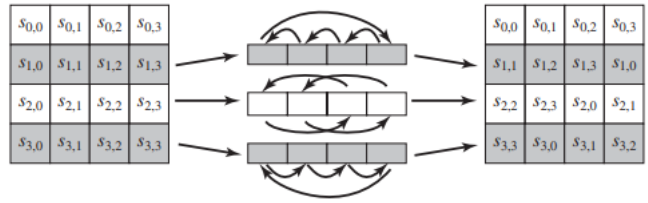
In this step, each byte is substituted by another byte. It is performed using a lookup table also called the S-box. This substitution is done in a way that a byte is never substituted by itself and not substituted by another byte which is a compliment to the current byte. The result of this step is a 16-byte (4 x 4) matrix like before.



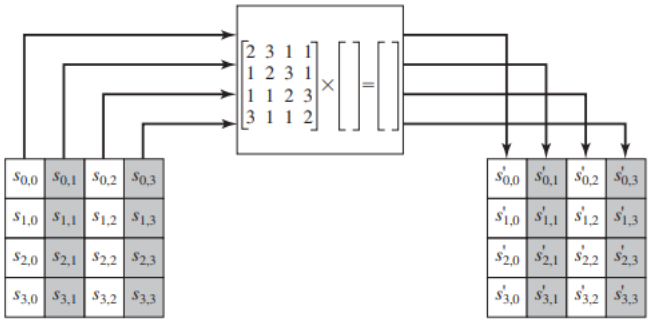
(a) Substitute byte transformation

ShiftRows :
This step is just as it sounds. Each row is shifted a particular number of times.

- The first row is not shifted
- The second row is shifted once to the left.
- The third row is shifted twice to the left.
- The fourth row is shifted thrice to the left.



(a) Shift row transformation



(b) Mix column transformation

Figure 5.7 AES Row and Column Operations

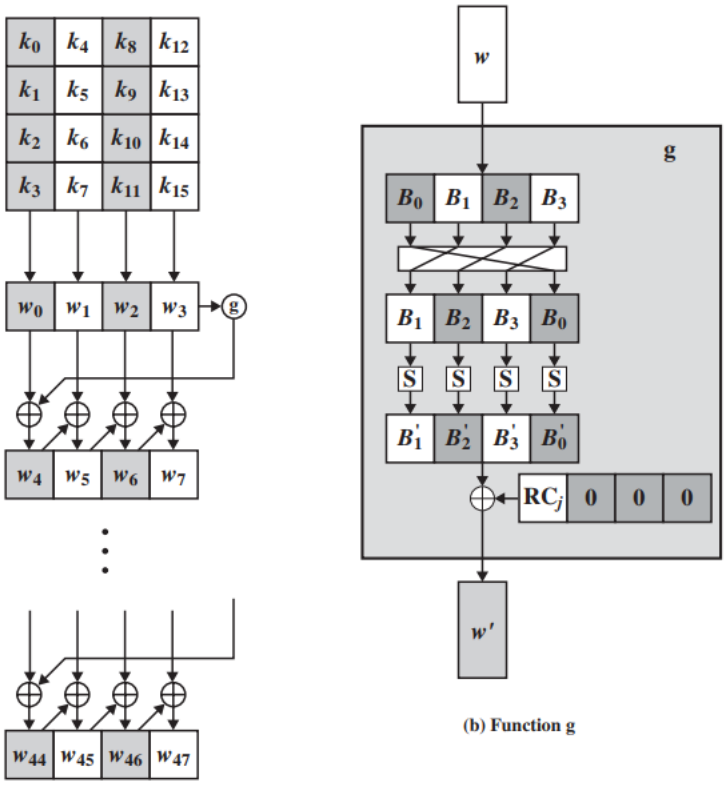
MixColumns :
This step is basically a matrix multiplication. Each column is multiplied with a specific matrix and thus the position of each byte in the column is changed as a result.

Add Round Keys :
Now the resultant output of the previous stage is XOR-ed with the corresponding round key. Here, the 16 bytes are not considered as a grid but just as 128 bits of data.

After all these rounds 128 bits of encrypted data is given back as output. This process is repeated until all the data to be encrypted undergoes this process.

47	40	A3	4C	AC	19	28	57	EB	59	8B	1B
37	D4	70	9F	77	FA	D1	5C	40	2E	A1	C3
94	E4	3A	42	66	DC	29	00	F2	38	13	42
ED	A5	A6	BC	F3	21	41	6A	1E	84	E7	D6

Key Expansion



(a) Overall algorithm

(b) Function g

Figure 5.9 AES Key Expansion

Principles of Public key Cryptosystem

Public key cryptography has become an essential means of providing confidentiality, especially through its need for key distribution, where users seeking private connection exchange encryption keys. It also features digital signatures which enable users to sign keys to check their identities.

The approach of public key cryptography derivative from an attempt to attack two of the most complex problems related to symmetric encryption. The first issue is that key distribution. Key distribution under symmetric encryption needed such as −

- that two communicants already shared a key, which somehow has been shared with them.
- the need for a key distribution center.

Public key Cryptosystem − Asymmetric algorithms depends on one key for encryption and a distinct but related key for decryption. These algorithms have the following characteristics which are as follows −

- It is computationally infeasible to decide on the decryption key given only information of the cryptographic algorithm and the encryption key.
- There are two related keys such as one can be used for encryption, with the other used for decryption.

A public key encryption scheme has the following ingredients which are as follows −

- **Plaintext** − This is the readable message or information that is informed into the algorithm as input.
- **Encryption algorithm** − The encryption algorithm performs several conversions on the plaintext.

- **Public and Private keys** − This is a set of keys that have been selected so that one can be used for encryption, and the other can be used for decryption.
- **Ciphertext** − This is a scrambled message generated as output. It is based on the plaintext and the key. For a given message, there are two specific keys that will create two different ciphertexts.
- **Decryption Algorithm** − This algorithm gets the ciphertext and the matching key and creates the original plaintext.
- The keys generated in public key cryptography are too large including 512, 1024, 2048 and so on bits. These keys are not simply to learn. Thus, they are maintained in the devices including USB tokens or hardware security modules.
- The major issue in public key cryptosystems is that an attacker can masquerade as a legal user. It can substitute the public key with a fake key in the public directory. Moreover, it can intercept the connection or alters those keys.
- Public key cryptography plays an essential role in online payment services and eCommerce etc. These online services are ensured only when the authenticity of the public key and signature of the user is ensured.
- The asymmetric cryptosystem should manage the security services including confidentiality, authentication, integrity, and non-repudiation. The public key should support the security services including non-repudiation and authentication. The security services of confidentiality and integrity are considered as an element of the encryption process completed by the private key of the user.

Simplified RC4 Example

Example

Steven Gordon

1 Simplified RC4 Example

Lets consider the stream cipher RC4, but instead of the full 256 bytes, we will use 8 x 3-bits. That is, the state vector **S** is 8 x 3-bits. We will operate on 3-bits of plaintext at a time since S can take the values 0 to 7, which can be represented as 3 bits.

Assume we use a 4 x 3-bit key of **K** = [1 2 3 6]. And a plaintext **P** = [1 2 2 2]

The first step is to generate the stream.

Initialise the state vector **S** and temporary vector **T**. **S** is initialised so the S[i] = i, and **T** is initialised so it is the key **K** (repeated as necessary).

S = [0 1 2 3 4 5 6 7]
T = [1 2 3 6 1 2 3 6]

Now perform the initial permutation on S.

```
j = 0;
for i = 0 to 7 do
            j = (j + S[i] + T[i]) mod 8
            Swap(S[i],S[j]);
end
```

For i = 0:
j = (0 + 0 + 1) mod 8
= 1
Swap(S[0],S[1]);

S = [1 0 2 3 4 5 6 7]

For i = 1:
j = 3
Swap(S[1],S[3])
S = [1 3 2 0 4 5 6 7];

For i = 2:
j = 0
Swap(S[2],S[0]);
S = [2 3 1 0 4 5 6 7];

For i = 3:
j = 6;
Swap(S[3],S[6])
S = [2 3 1 6 4 5 0 7];

For i = 4:
j = 3
Swap(S[4],S[3])
S = [2 3 1 4 6 5 0 7];

For i = 5:
j = 2
Swap(S[5],S[2]);
S = [2 3 5 4 6 1 0 7];

For i = 6:
j = 5;
Swap(S[6],S[4])
S = [2 3 5 4 0 1 6 7];

For i = 7:
j = 2;
Swap(S[7],S[2])
S = [2 3 7 4 0 1 6 5];

Hence, our initial permutation of **S** = [2 3 7 4 0 1 6 5];

Now we generate 3-bits at a time, k, that we XOR with each 3-bits of plaintext to produce the ciphertext. The 3-bits k is generated by:

```
i, j = 0;
while (true) {
      i = (i + 1) mod 8;
      j = (j + S[i]) mod 8;
      Swap (S[i], S[j]);
      t = (S[i] + S[j]) mod 8;
      k = S[t]; }
```

The first iteration:
S = [2 3 7 4 0 1 6 5]
i = (0 + 1) mod 8 = 1
j = (0 + S[1]) mod 8 = 3
Swap(S[1],S[3])
S = [2 4 7 3 0 1 6 5]
t = (S[1] + S[3]) mod 8 = 7
k = S[7] = 5

Remember, **P** = [1 2 2 2]

So our first 3-bits of ciphertext is obtained by: k XOR P
5 XOR 1 = 101 XOR 001 = 100 = 4

The second iteration:
S = [2 4 7 3 0 1 6 5]
i = (1 + 1) mod 8 = 2
j = (2 + S[2]) mod 8 = 1
Swap(S[2],S[1])
S = [2 7 4 3 0 1 6 5]

t = (S[2] + S[1]) mod 8 = 3
k = S[3] = 3

Second 3-bits of ciphertext are:
3 XOR 2 = 011 XOR 010 = 001 = 1

The third iteration:
S = [2 7 4 3 0 1 6 5]
i = (2 + 1) mod 8 = 3
j = (1 + S[3]) mod 8 = 4
Swap(S[3],S[4])
S = [2 7 4 0 3 1 6 5]
t = (S[3] + S[4]) mod 8 = 3
k = S[3] = 0

Third 3-bits of ciphertext are:
0 XOR 2 = 000 XOR 010 = 010 = 2

The final iteration:
S = [2 7 4 0 3 1 6 5]
i = (1 + 3) mod 8 = 4
j = (4 + S[4]) mod 8 = 7
Swap(S[4],S[7])
S = [2 7 4 0 5 1 6 3]
t = (S[4] + S[7]) mod 8 = 0
k = S[0] = 2

Last 3-bits of ciphertext are:
2 XOR 2 = 010 XOR 010 = 000 = 0

So to encrypt the plaintext stream **P** = [1 2 2 2] with key **K** = [1 2 3 6] using our simplified RC4 stream cipher we get **C** = [4 1 2 0].

(or in binary: **P** = 001010010010, **K** = 001010011110 and **C** = 100001010000)