UNIT - II

* Symmetric Key Ciphers :-

1. Block Cipher Principles
2. DES
3. AES
4. RC5 and Block Cipher Operation
5. Stream Ciphers
6. RC4

* ASymmetric Key Ciphers :-

1. Principles of Public key Cryptosystems
2. RSA Algorithm
3. Diffie-Hellman Key Exchange
4. Knapsack Algorithm

UNIT-II -

* Symmetric Key Ciphers :-

1. Block Cipher Principles
2. DES
3. AES
4. RC5 and Block Cipher Operation
5. Stream Ciphers
6. RC4

* ASymmetric Key Ciphers :-

1 Principles Of Public Key Cryptosystems
2. RSA Algorithm
3. Diffie-Hellman Key Exchange
4. Knapsack Algorithm

Unit-II

→ Encryption and Decryption :

- The process of encoding plain-text messages into cipher text messages is called encryption.
- The reverse process of transforming cipher-text message back to plain-text message is called decryption.
- Clear text or plain text signifies a message that can be understood by the sender, the receiver and also by anyone else who gets access to that message.
- When a plain text message is codified using any suitable scheme, the resulting message is called cipher text.
- Every encryption and decryption process has two aspects
  - * The algorithm and
  - * The key used for encryption and decryption.
- In general the algorithm used for encryption and decryption processes is usually known to everybody.
- However, it is the key used for encryption and decryption that makes the process of cryptography secure
- Broadly, there are two cryptographic mechanisms, depending on what keys are used.

Cryptography Techniques

Symmetric Key Cryptography

Asymmetric Key Cryptography

- Symmetric Key Cryptography involves the usage of the same key for encryption and decryption.
- Asymmetric Key cryptography involves the usage of one key for encryption and another different key for decryption.

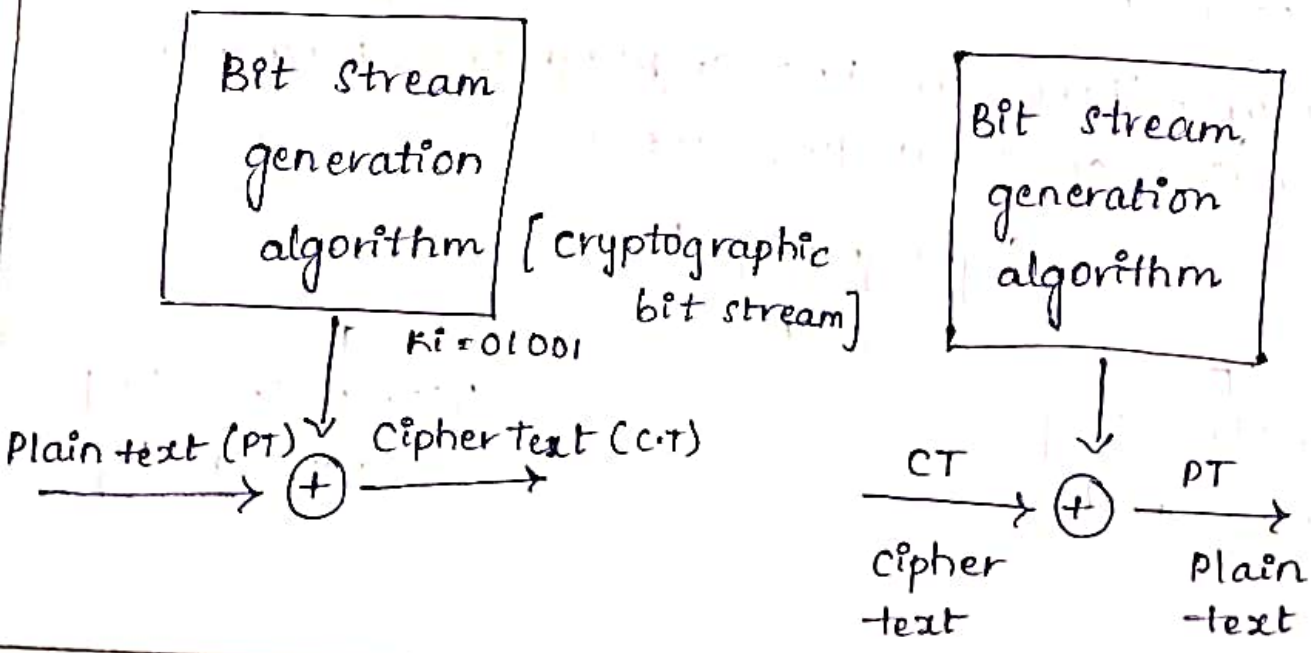
→ Stream Cipher and Block Cipher:

In cryptography these two are used for convert of plain text → cipher text.

(1) Stream Cipher: It is one that encrypts a digital data stream one bit or one byte at a time. The examples of stream ciphers are the autokeyed genere cipher and vernamcipher.

Here the bit stream generator is a key-enrolled algorithm and must produce a bit stream that is Cryptographically strong. It is a symmetric key cipher (i.e 1 key of encrypt + decrypt).

Encryption: Generates key in the form of bits.



Ex:

Let us message at sender sid is 10110110

To encrypt:- 1 0 1 1 0 1 1 0 ← message at sender side

0 1 0 1 0 1 0 1 ← Key

1 1 1 0 0 0 1 1 ← cipher

To decrypt :- 1 1 1 0 0 0 1 1 ← cipher Text

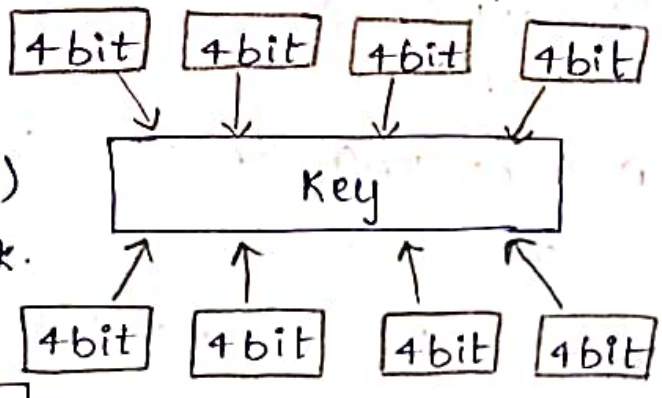
0 1 0 1 0 1 0 1 ← Key

1 0 1 1 0 1 1 0 ← Plain Text

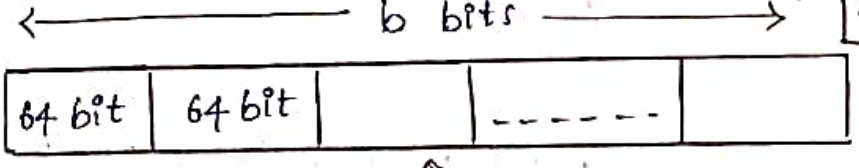
[Reverse side]

Draw Back: This can be introduce insurmountable logistical problems if the intended data traffic is very large.

(2) Block Cipher: It is one in which a block of Plain text is treated as a whole and used to produce a cipher text of equal length. Block bite of 64 or 128 bits is used.

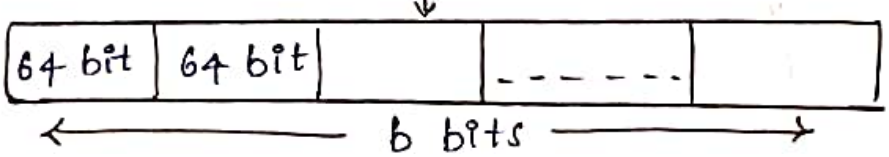


Symmetric Key Cipher (1 Key used)
Key will be applied on each block.



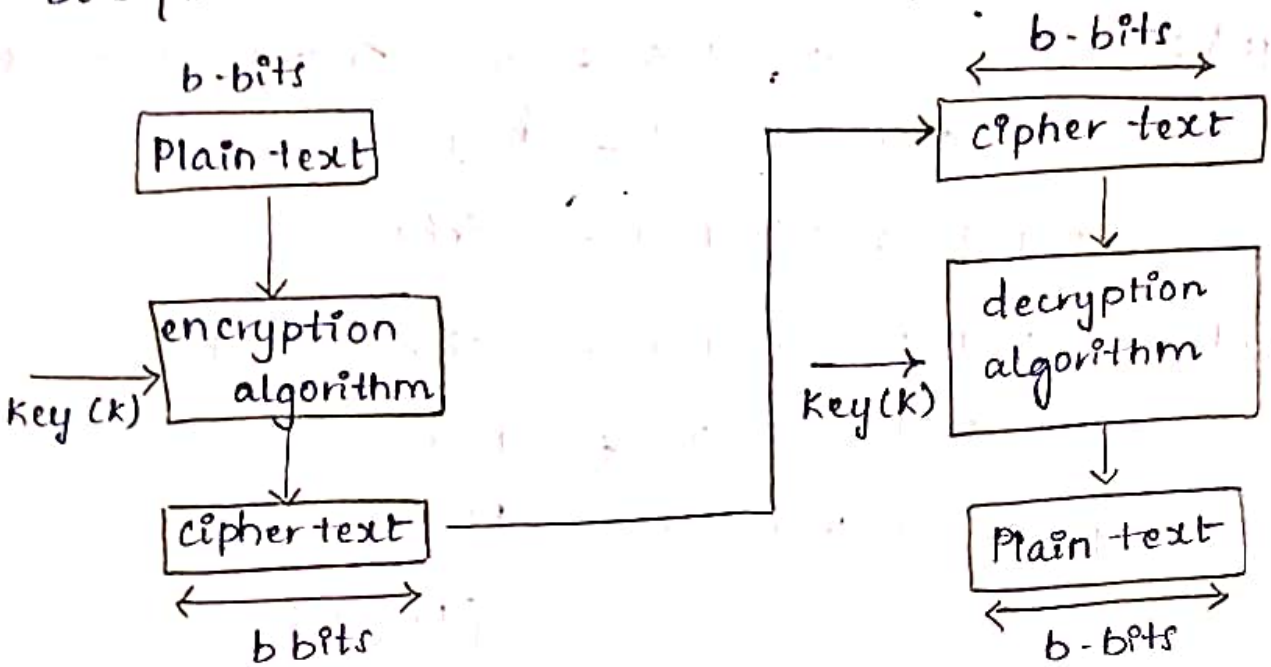
Plain Text in block

Cipher Text will also be in block



Ex: DES (64 bit block size) cipher & plain text

DES [DATA ENCRYPTION STANDARD]



Block Cipher

① Plain text → cipher text by taking PTS in blocks at a time.

② It uses 64 bits or more.

③ Complexity of block cipher is simple.

④ Uses confusion as well as diffussion concept.

⑤ In this reverse encryption text is hard.

→ modes :

⑥ ECB (Electronic Code Block)
CBC (Cipher Block Chaining)
algorithomic modes used.

Stream Cipher

① 1 bit or 1 byte of plain text → cipher text

② Stream cipher uses b bits

③ While stream cipher is more complex.

④ uses only confussion concept.

⑤ Reverse encrypted text is easy.

⑥ CFB (Cipher feed Back)
OFB (Output Feed Back)

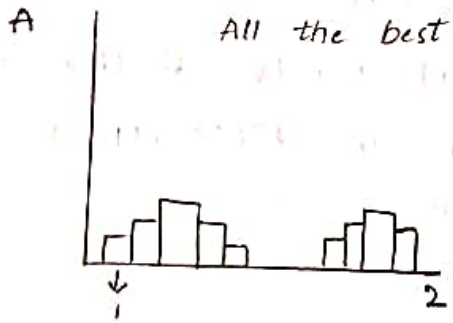
→ Confusion and Diffusion: The term confusion and diffusion were introduced by "Craude Shannon" to capture the two basic building blocks for any cryptographic system.

Let us assume the attacker has some knowledge of the statistical characteristics of the plain text.

[Example: Human readable message in some language.]

The frequency distribution of the various letters may be known. If there statistics are in any way reflected in the cipher text, the cryptanalyst.

i.e attacker maybe able to get the encryption key.



These shanon suggested & methods for to secure from the attackers.

(1) Confussion
(2) Diffusion
} Properties for creating a secure cipher.

→ Diffusion: In simple words is a symbol is in the PT - is changed several or all symbols in the CT will also change.

→ The idea of diffussion is to hide the relationship between cipher text and plain-text.

→ It means that if we change the single bit of the plain text, then (statistically) half of the bit in the cipher text should change and similarly.

→ If we change 1 bit of ciphertext then atleast one half of the plaintext bit should change.

→ Diffusion implies that each symbol in the ciphertext is dependent on some or all symbols in plaintext.

(2) Confussion : It hide the relationship (the plaintext and ciphertext as complex as possible again to attempt to discover the key). and the key.

→ If a single bit in the key is changed the mostly all bits of ciphertext will also be changed.

→ Confussion means that each bit of the ciphertext should depend on several parts of the key obscaring (make unclear or difficult to understand) the connection between the too.

→ Diffussion

(1) Make statistical relation between plaintext and ciphertext as complex as possible.

(2) If change in 1 bit of plaintext than half or more bits of ciphertext should change.

→ Confusion

(1) Makes relation blw key and cipher as complex as possible.

(2) Each bit of ciphertext should depend on the key.

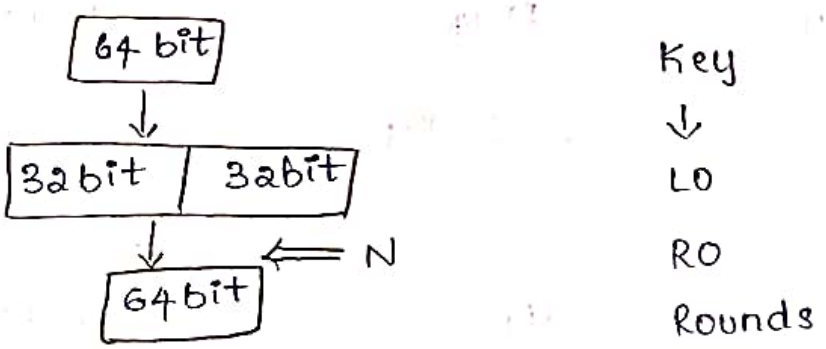
Fiestal Cipher Structure:

Most of the Block cipher technique follows the structure. Here the plain text is processed (divide) in to equal halves i.e LO, RO the I/P in the encryption algorithm are a plain-text block of length a bits and a key. The two halves of the data pass through n rounds of processing and then combine to produce the ciphertext block.

i has i/p (i-1, 2i-1) derived from the previous are round as well as a subkey Ki derived from the overall K.

On the right half we apply a function in the "f" we use a subkey generated from the masterkey.

The O/P of this is XORed with the left and then their O/P will be swapped.



(DES)

We will have nrounds depends on the algorithm. If any algorithm, we decide the plain-text in 2 halves and apply the function on RHS and XOR it with LHS in the O/P is swapped there that algorithm allows fiestal Structure.

Block size :

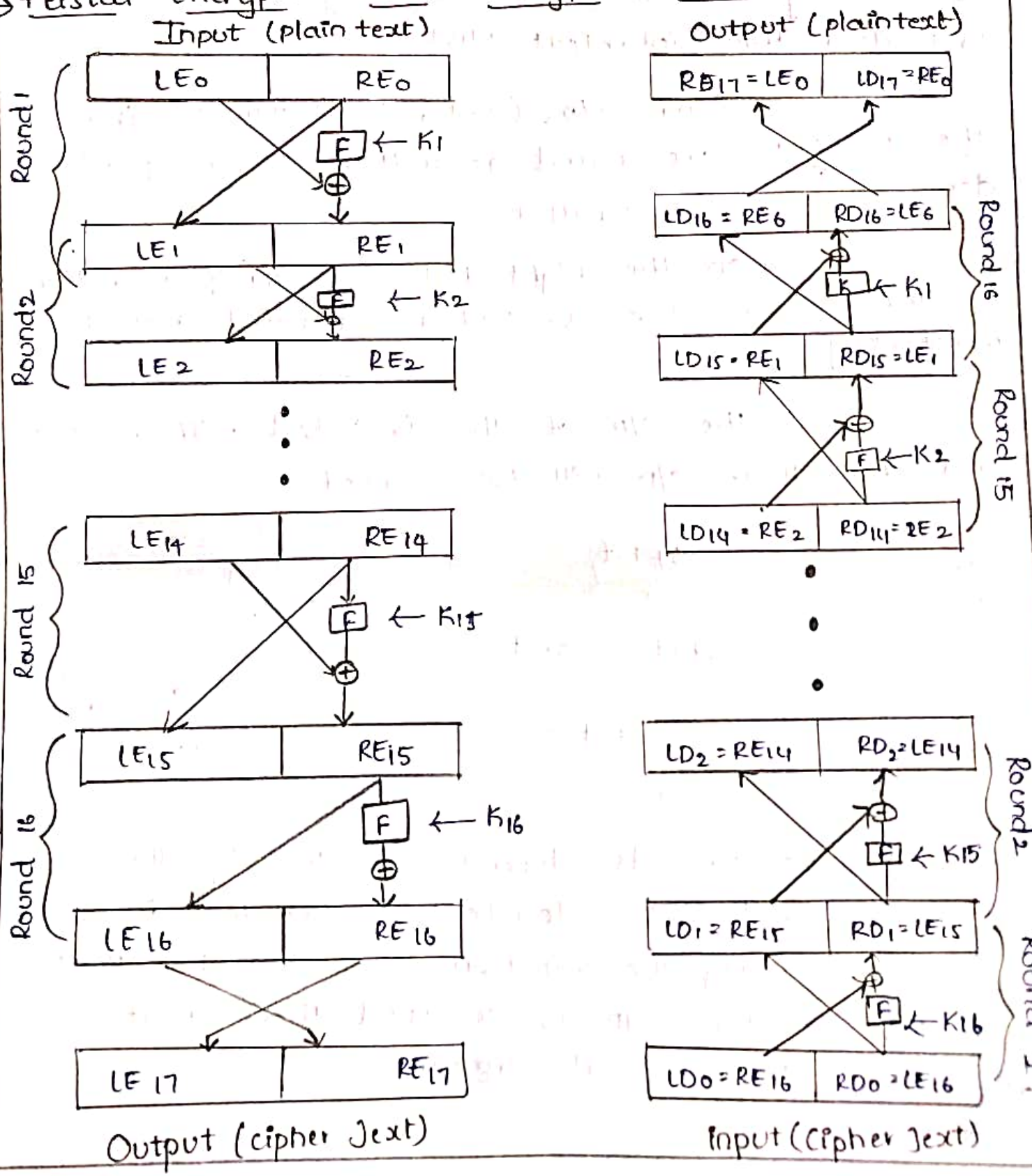
Key size

No. of rounds

Subkey generation algorithm

Round function [f]

→ feistal Encryption and Decryption (16 rounds)



Block Size: Larger Block sizes mean greater security (all other things being equal) but reduced encryption/ decryption speed for a given algorithm. The greater security is achieved by greater diffusion. Traditionally a block size of 64 bit has been considered a reasonable tradeoff and was nearly universal in block cipher design. However, the new AES uses a 128-bit block size.

Key size: Larger key size means greater security but may decrease encryption/ decryption speed. The greater security is actived by greater resistance to brute-force attacks and greater confusion. Key sizes of 64 bit or less are now widely considerd to be inadiquate and 128 bit has become a common size.

Number of Rounds: The essence of the fiestal cipher is that a single round offers inadequate security but that multiple rounds offer increasing security. A typical size is 16 rounds.

Subkey generation algorithm: Greater complexity in the algorithm should lead to greater difficulty of cryptanalysis.

Round function (F): Again, greater complexity generally means greater resistance to cryptanalysis.

Block Cipher:

Block cipher is an encryption method which divides the plain text into blocks of fixed size.

- Each block has an equal number of bits.
- At a time, block cipher operates only on one block of plain text and applies key on it to produce the corresponding block of cipher-text.
- While decryption also only one block of ciphertext is operated to produce its corresponding plain text.
- Data encryption Standard (DES) is the best example of it.

Block Cipher Principles:

- A block cipher is designed by considering its three critical aspects which are listed as below:
  (1) Number of Rounds.
  (2) Design of function F
  (3) Key schedule Algorithm.

(1) Number of Rounds:

- The number of rounds judges the strength of the block cipher algorithm.
- It is considered that more is the number of rounds difficult is for cryptanalysis to break the algorithm
- It is considered that even if the function F is relatively weak, the number of rounds would make the algorithm tough to break.

(2) Design of function:

- The function f of the block cipher must be designed such that it must be impossible for any cryptanalysis to unscramble the substitution.
- The criterion that strengthens the function f is its non-linearity.
- More the function f is nonlinear, more it would be difficult to crack it.
- While designing the function f it should be confirmed that it has a good avalanche property which states that a change in one-bit of input must reflect the change in many bits of output.
- The function f, should be designed such that it possesses a bit independence criterion which states that the output bits must change independently if there is any change in the input bit.

(3) Key Schedule Algorithm:

- It is suggested that the key schedule should confirm the strict avalanbe effect and bit Independence criterion.

⟹ Block Cipher Modes of Operation:

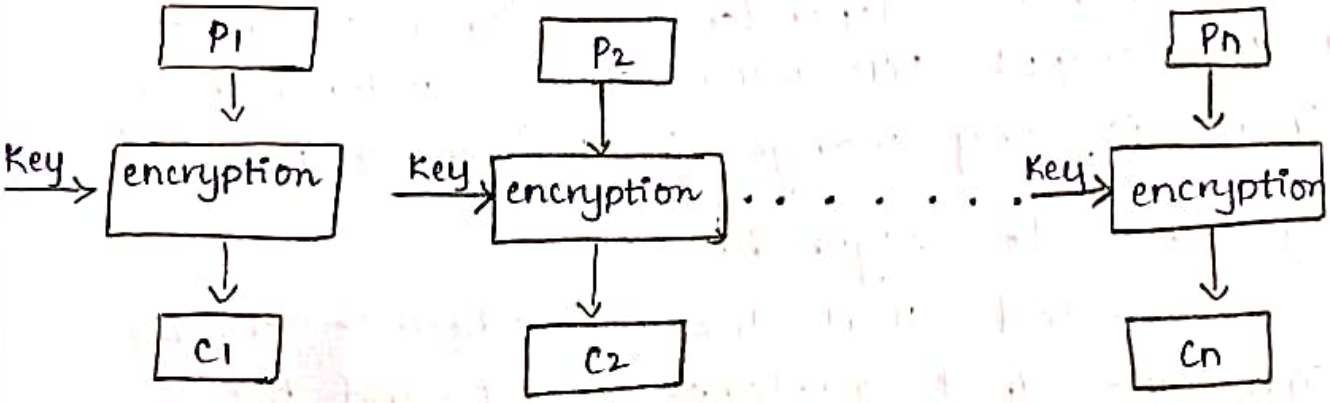
- There are five important block ciphers modes of operation.
- These five modes of operation enhance the algorithm so that it can be adapted by a wide range of applications which uses block cipher algorithm for

encryption.

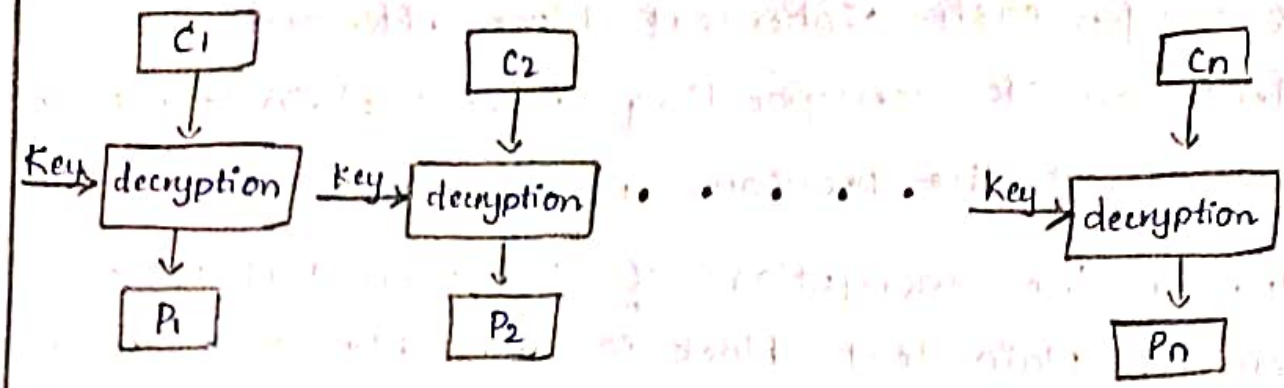
(1) Electronic Code Book Mode
(2) Cipher Block Chaining Mode
(3) Cipher Feedback Mode
(4) Output Feedback Mode
(5) Counter Mode.

(1) Electronic Code Book Mode: (ECB)

- In electronic code book mode the plain text is divided into the blocks, each of 64 bit.
- Each block is encrypted one at a time to produce the cipher block.
- The same key is used to encrypt each block.



- When the receiver receives the message i.e Cipher Text
- This ciphertext is again divided into blocks, each of 64-bit and each block is decrypted independently one at a time to obtain the corresponding plain text block.
- Here also the same key is used to decrypt each block which was used to encrypt each block.



- As the same key is used to encrypt each block of plain text there arises an issue that for a repeating plain text block it would generate the same cipher. and will ease the cryptanalysis to crack the algorithm
- Hence, ECB, is considered for encrypting the small messages which have a rare possibility of repeating text.

(2) Cipher Block Chaining Mode: (CBC)

- To overcome the limitation of ECB i.e the repeating block in plain text produces the same ciphertext, a new technique was required which is cipher block chaining Mode.
- CBC confirms that even if the plain text has repeating blocks its encryption won't produce same cipher block.
- To achieve totally different cipher blocks for two same plain text blocks chaining has been added to the block cipher.
- For this, the result obtained from encryption of the first plain text block is fed to the encryption of the next plain text box.

- In this way, each cipher-text block obtained is dependent on its corresponding current plain-text block input and all the previous plain-text blocks.
- But during the encryption of the plain text block, no previous plain text block is available so a random block of text is generated called Initialization Vector (IV)
- The encryption steps of CBC are :

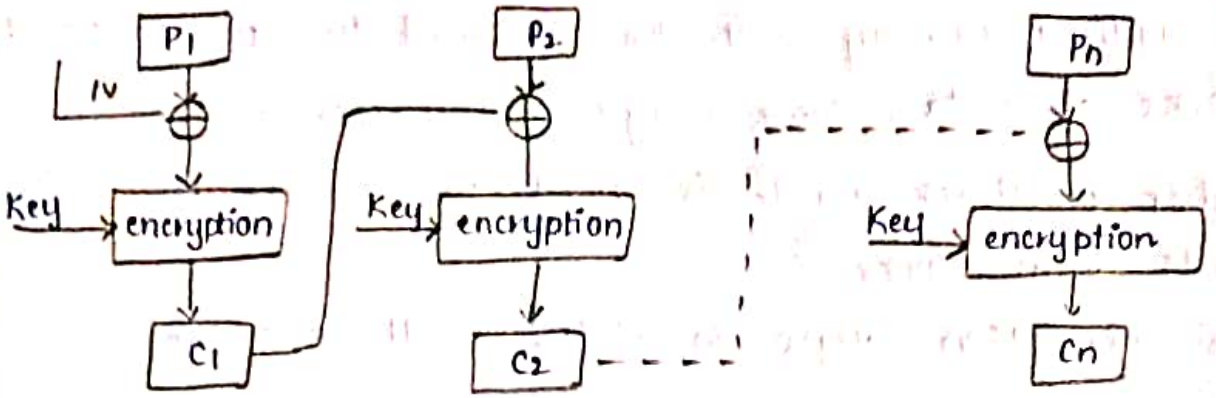
→ Step 1 :

The initialization vector and first plain text block are XORed and the result of XOR is then encrypted using the key to obtain the first ciphertext block.

→ Step 2 :

- The first ciphertext block is fed to the encryption of the second plain text block.
- for encryption of second plain text block, first cipher text block and second plain text is XORed and the result of XOR is encrypted using the same key in step 1 to obtain the second ciphertext block.
- Similarly, the result of encryption of second plain text block i.ue the second ciphertext block is fed to the encryption of third plain text block to obtain third cipher text block.
- And the process continues to obtain all the cipher text blocks.

- The steps of CBC are shown below:



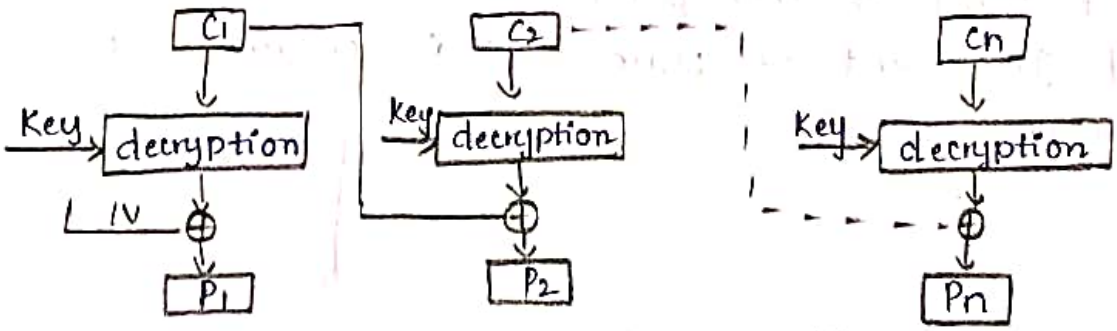
Decryption steps of CBC:

Step 1:

- The first ciphertext block is decrypted using the same key that was used for encrypting all plain text blocks.
- The result of decryption is then XORed with the initialization vector (IV) to obtain the first plain text book.

Step 2:

- The second ciphertext block is decrypted and the result of decryption is XORed with the first cipher text block to obtain the second plain text bbck.
- And the process continues till all plain text blocks are retrieved.



- There is a limitation in CBC that if we have two identical messages and if we use the same IV for both the identical messages it would generate the same cipher text block.

(3) Cipher feedback Mode : (CFB)

- All applications may not be designed to operate on the blocks of data, some may be character or bit-oriented
- Cipher feedback mode is used to operate on smaller units than blocks.
- The encryption steps in cipher feedback mode are:

Step 1 :

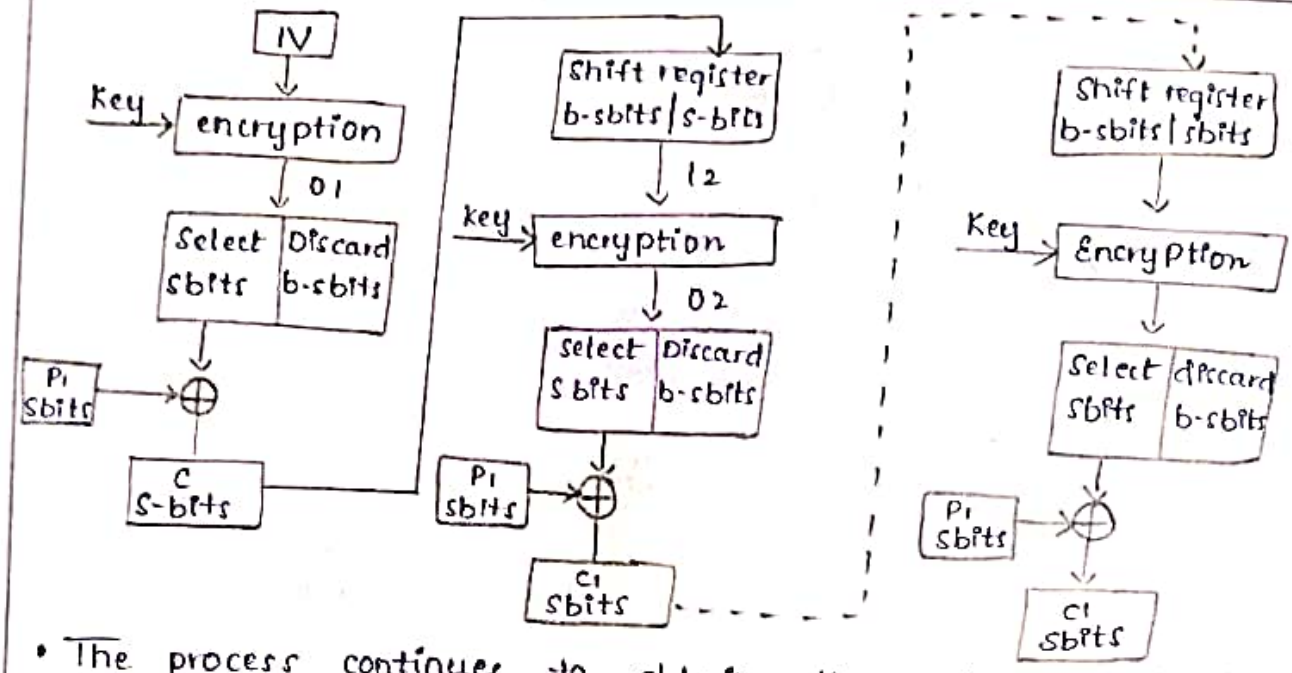
Here the initialization vector IV is kept in the shift register and it is encrypted using the key.

Step 2 :

- The left most s bits of the encrypted IV is then XORed with the first fragment of the plain text of s bits.
- It produces the first ciphertext c_1 of s bits.

Step 3 :

- Now the shift register containing initialization vector performs left shift by s bits and s bits c_1 replaces the right most s bits of the initialization vector.
- Then again, the encryption is performed on IV and the leftmost s bit of encrypted IV is XORed with the second fragment of plain text is obtain s bit ciphertext c_2.



- The process continues to obtain all crypto-cipher text fragments.

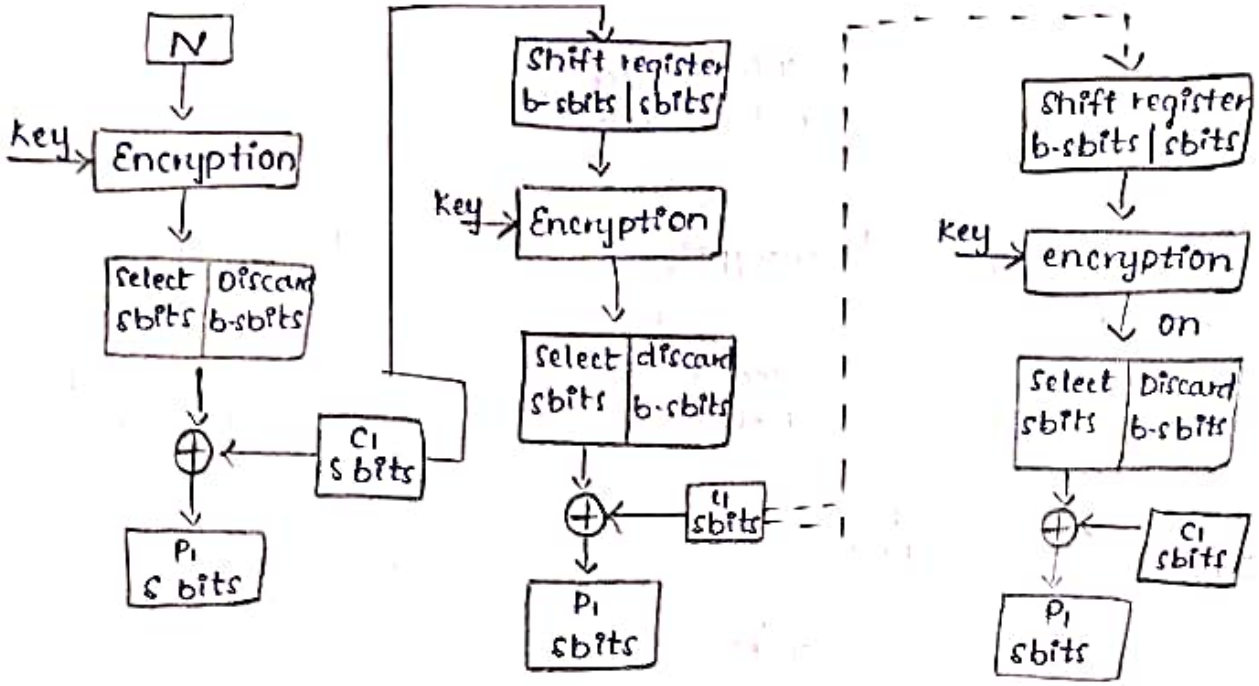
⇒ Decryption Steps:

Step 1:

- The initialization vector is placed in the shift register
- It is encrypted using the same key.
- Note that even in the decryption process the encryption algorithm is implemented instead of the decryption algorithm.
- Then from the encrypted IV sbits are XORed with the sbits ciphertext C_1 to retreive s bits of plain text P_1.

Step 2:

- The IV in the shift register is left-shifted by s bits and the sbits C_1 replaces the rightmost sbits of IV
- The process continues untill all plain text fragments and retrieved.



- It has a limitation that if there occur a bit error in any cipher text it would affect all the subsequent ciphertext units.

(4) Output feedback Mode: (OFB)

- The output feedback mode is almost similar to the CFB.
- The difference between CFB and OFB is that unlike CFB, in OFB the encrypted IV is fed to the encryption of next plain text block.
- The other difference is that CFB operates on a stream of bits whereas OFB operates on the block of bits

⇒ Steps for Encryption:

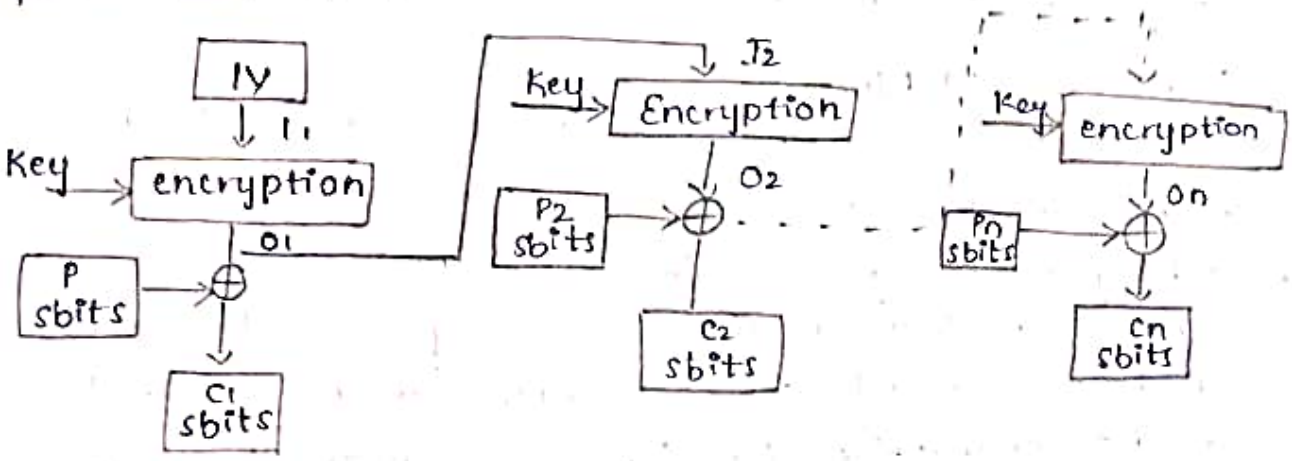
Step 1:

The initialization vector is encrypted using the key.

Step 2:

- The encrypted IV is then XORed with the plain-text block to obtain the ciphertext block.

- The encrypted IV is fed to the encryption of next plain text block.



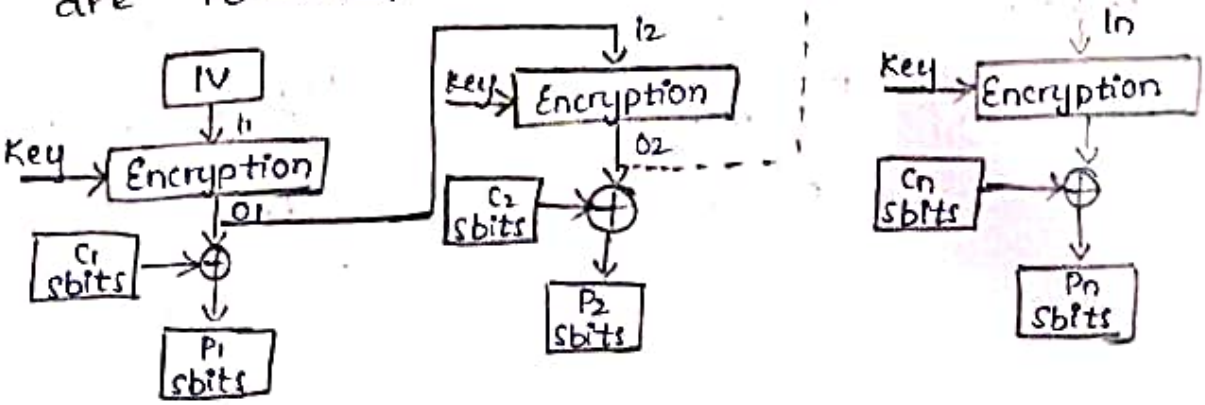
→ Steps for decryption:

Step 1:

- The initialization vector is encrypted using the same key used for encrypting all plain text blocks.
- In the decryption process also the encryption function is implemented.

Step 2:

- The encrypted IV is then XORed with the ciphertext block to retrieve the plain text block.
- The encrypted IV is also fed to the decryption process of the next ciphertext block.
- The process continues until all the plain text blocks are retreired.



- The advantage of OFB is that it protects the propogation of bit error.

(5) Counter (CTR) Mode:

- It is similar to OFB but there is no feedback mechanism in counter mode.
- Nothing is being fed from the previous step to the next step instead it uses a sequence of number which is termed as a counter which is input to the encryption function along with the key.
- After a plain text block is encrypted the counter value increments by 1.

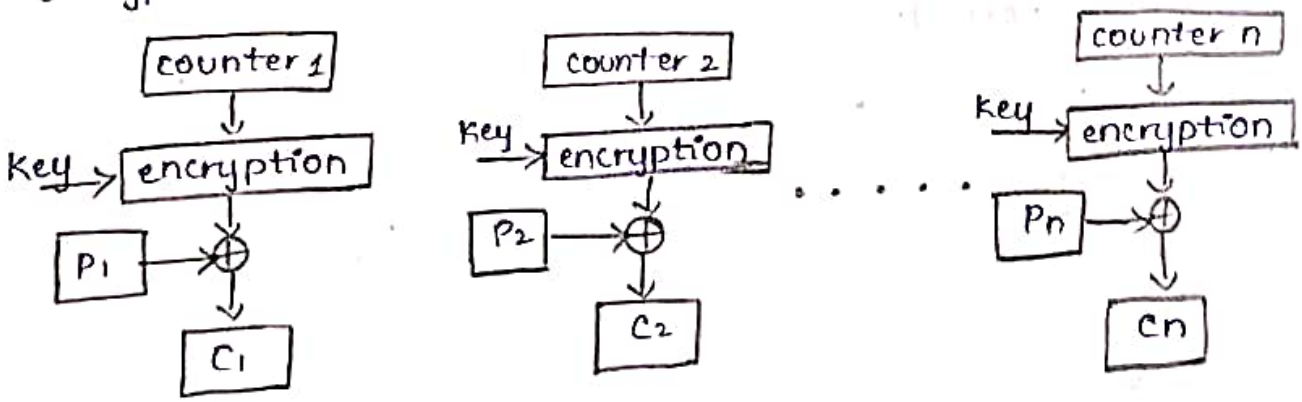
→ Steps of Encryption:

Step 1:

- The counter value is encrypted using a key.

Step 2:

- The encrypted counter value is XORed with the Plain text block to obtain a ciphertext block.
- To encrypt the next subsequent plain text block the counter value is incremental by 1 and step 1 and 2 are repeated to obtain the corresponding ciphertext.
- The process continues until all plain text block is encrypted.



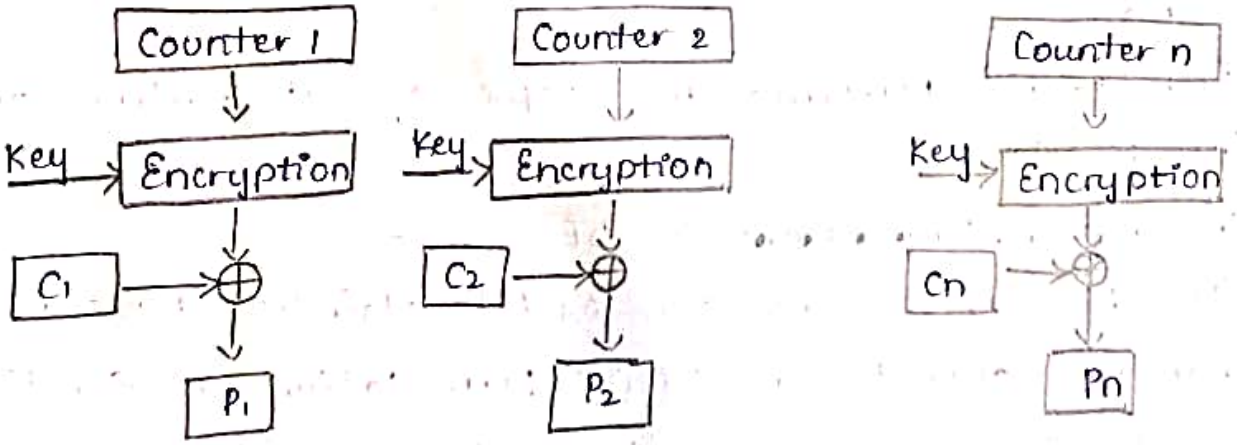
→ Steps for Decryption:

Step 1:

- The counter value is encrypted using a key.
- Here the encryption function is used in the decryption process.
- The same counter values are used for decryption as used while encryption.

Step 2:

- The encrypted counter value is XORed with the cipher text block to obtain a plain text block.
- To decrypt the next subsequent ciphertext block the counter value is incremented by 1 and step 1 and 2 are repeated to obtain corresponding plain text.
- The process continues until all cipher text block is decrypted.



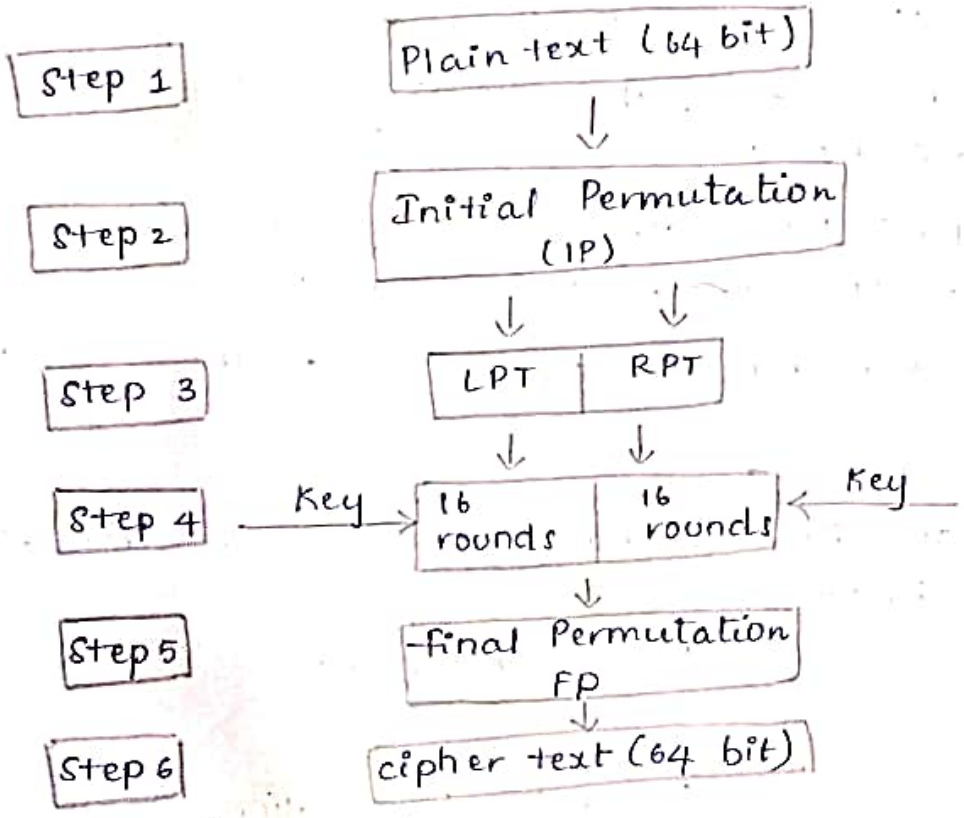
⟹ Data Encryption Standard : (DES).

* DES is a block cipher.
* It encrypts data in blocks of 64 bits each.
* That's is, 64 bits of plain text goes as the input to DES, which produces 64 bits of cipher text.
* The same algorithm and key are used for encryption and decryption.
* The key length is 56 bits.
* Actually, the initial key consists of 64 bits.
* However: every eighth bit of the key is discarded to produce a 56-bit key. (That is, bit positions 8, 16, 24, 32, 40, 48, 56 and 64 are discarded).
* DES is based on two fundamental attributes of cryptography namely substitution (also called confusion) and transposition (also called diffusion).
* DES consists of 16 steps, each of which is called a round.
* Each round performs the steps of substitution and transposition.
* The broad-level steps in DES are:

① In the first step, the 64 bit plaintext block is handled over to an initial permutation (IP) function.
② The initial permutation is performed on plain text
③ Next, the initial Permutation (IP) produces two halves of the permuted block, say left plain text (LPT) and right plain text (RPT).
④ Now, each of the LPT and RPT go through 16 rounds of encryption process, each with its own key.

⑤ In the end, LPT and RPT are rejoined, and a final permutation (FP) is performed on the combined block.

⑥ The result of this process produces 64-bit cipher.



⇒ Initial Permutation : (IP)

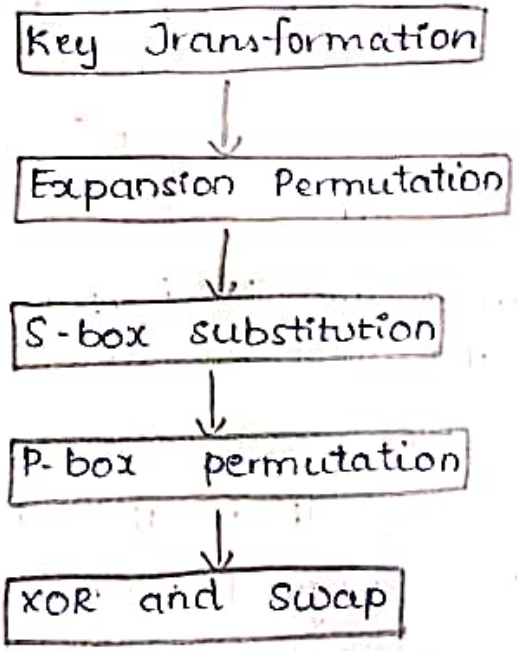
- The Initial Permutation (IP) happens only once and it happens before the first round.
- It suggests how the transposition should proceed.
- For example, it says that the IP replaces the first bit of the original plain-text block with the 58th bit of the original plain-text block, the second bit with the 50th bit of the original plain-text block and so on.
- The complete transposition table is shown below.

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	24	40	32	16	8
57	49	41	33	25	17	9	1	59	51	43	19	35	37	11	3
61	53	45	37	29	21	13	5	63	55	47	23	39	31	15	7

- After IP is done, the resulting 64-bit is divided into two half blocks.
- Each half block consists of 32-bits. We call the left block as LPT and right block as RPT.
- Now, 16 rounds are performed on these two blocks.

Rounds:

- Each of the 16rounds consists of the broad-level steps shown below.



Step 1: Key Jrans-formation

- In Key Jransformation from the 56-bit Key, a different 48-bit subkey is generated during each round.
- for this, the 56-bit key is divided into two halves each of 28 bits.

- These halves are circularly shifted left by one or two position depending on the round.
- for example for round number 1, 2, 9 or 16 the shift done by only one position, for other rounds, the circular shift is done by two positions.
- After an appropriate shift, 48 of the 56 bits are selected.
- Since the key-transformation process involves permutations as well as selection of a 48-bit subset of the original 56-bit key it is called compression permutation.

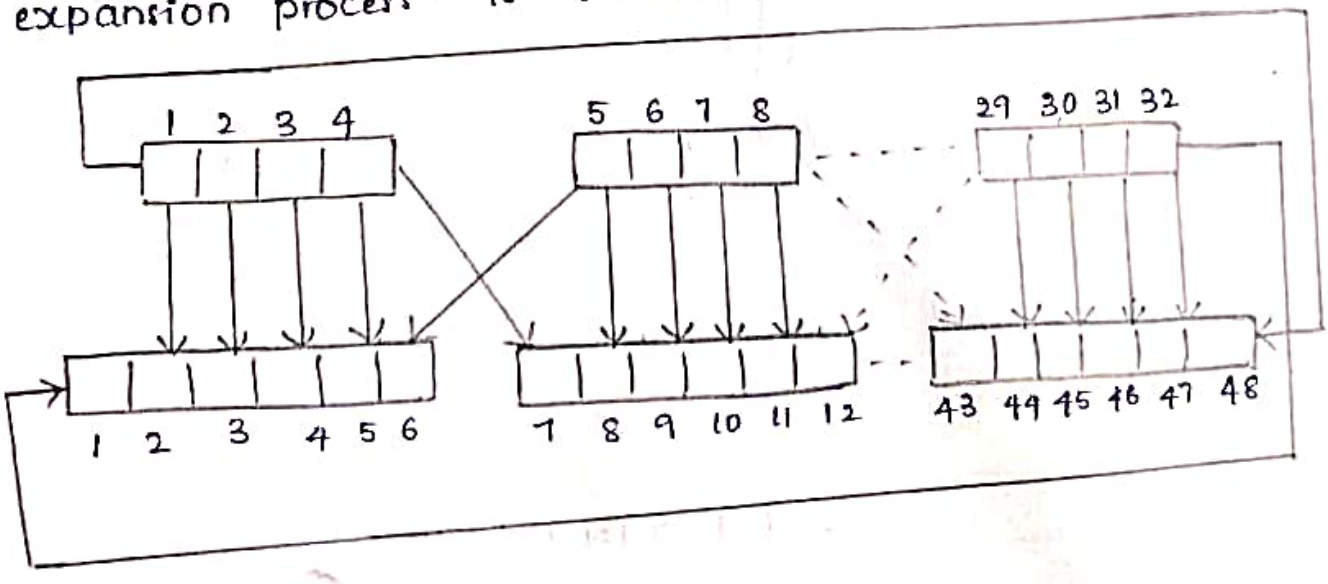
Step 2: Expansion Permutation

- During expansion permutation, the RPT is expanded from 32 bits to 48 bits. This happens as follows.

(1.) The 32-bit RPT is divided into 8 blocks, with each block consisting of 4 bits.

(2) Next each 4-bit block of the above step is then expanded to a corresponding 6-bit block.

-i.e per 4-bit block, 2 more bits are added. The expansion process is shown below.



- Now the 48-bit Key is XOR with 48 bit RPT and the resulting output is given to the next step, which is the S-box substitution.

Step 3: S box Substitution

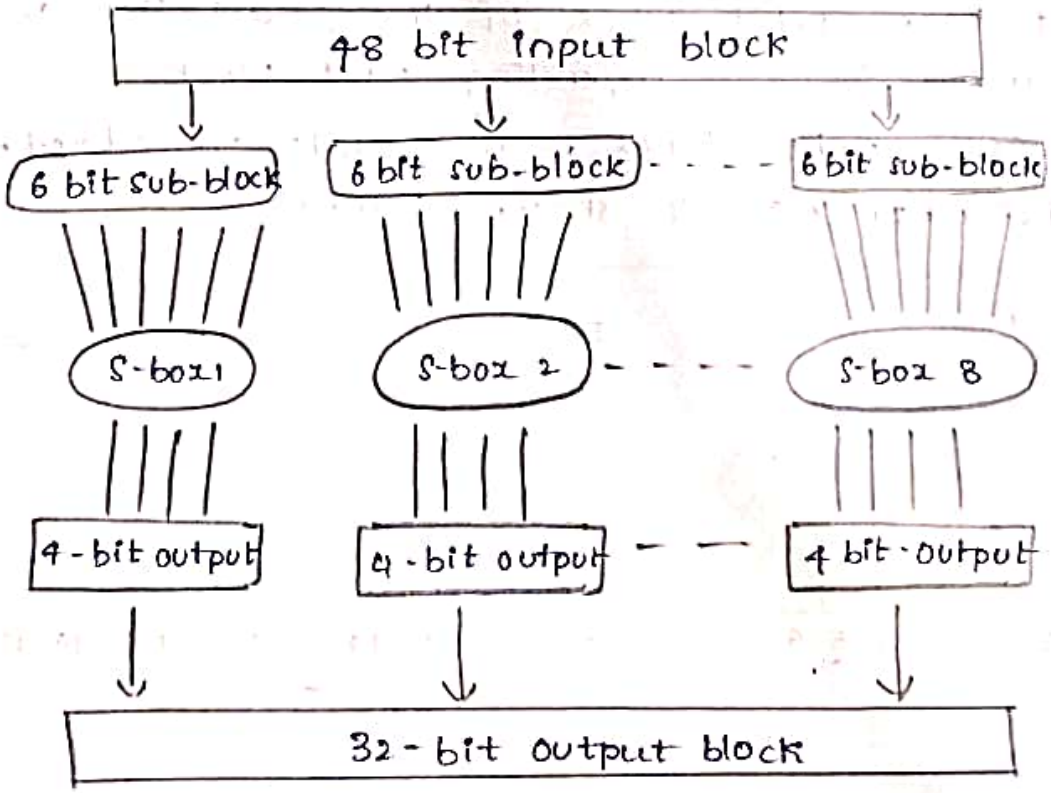
S box substitution is a process that accepts the 48 bit input from the XOR operation involving the compressed Key and expanded RPT, and produces a 32-bit output using the substitution technique

→ The substitution is performed by eight substitution boxes called as S-boxes.

→ Each of the eight S-boxes has 6-bit input and a 4-bit output

→ The 48 bit input block is divided into 8 sub blocks and each such sub-block is given to an S-box.

→ The S-box transforms the 6 bit input into 4 bit output as shown below.



The output of each S-box is then combined to form a 32-bit block, which is given to the last stage of a round, the P-box permutation.

Step-4 : Box Permutation :

→ The output of S-box consists of 32 bits. These 32 bits are permutated using a P-box.

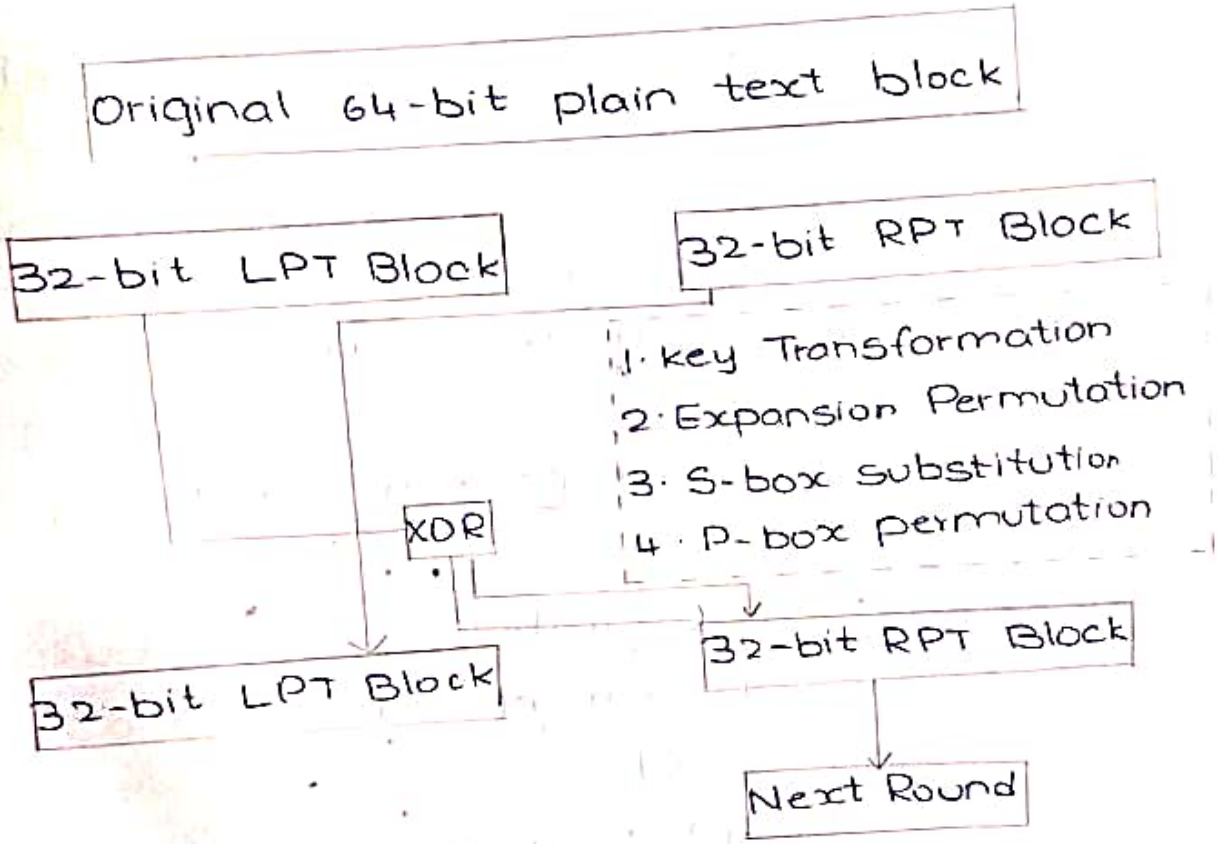
→ The straight-forward permutation mechanism involves single permutation i.e replacement of each bits with another bit without any expansion or compression.

Step 5 : XOR and Swap

→ At this juncture, the left half portion of the initial 64-bit plain text block, LPT is XORed with the output produced by P-box permutation.

→ The result of this XOR operation becomes the new right half i.e, RPT

→ The old right half i.e, RPT becomes the new left half, in a process of Swapping



→ Final Permutation

→ At the end of the 16 rounds, the final permutation is performed

→ For instance, the 40th input bit takes the position of the 1st output and so on.

→ The instance, the 40th input, output of the final permutation is the 64-bit encrypted block

→ RC4 (Rivest Cipher 4)

→ RC4 is a Stream Cipher

→ This means that the encryption happens byte-by-byte

→ However this can be changed to bit-by-bit encryption

→ RC4 generates a pseudorandom stream of bits called keystream

→ This is combined with the plain text using XOR for encryption

→ It uses a variable length key consisting of 1 to 256 bytes

→ This key is used to initialize a 256-bytes State Vector, with elements identified as S[0], S[1]..... S[255].

→ To perform encryption or decryption operation one of these 256 bytes of S is selected and proceed

→ we call the resulting output as k

→ After this the entries in S are permuted once again

→ Overall, there are two processes involved
(a) initialization of S (the State Vector) and
(b) Stream generation

1. Initialization of S

This process consists of the following steps

1. choose a key (k) of length between 1 and 256 bytes.
2. Set the values in the State vector S equal to the values from 0 to 255 in an ascending order
   i.e, Set S[0]=1, S[1]=1, ... S[255]=255
3. Create another temporary array T
   → If the length of the key K is 256 bytes, copy K into T as it is.
   → otherwise, after copying K to T, whatever are the remaining positions in T are filled with the values of K again.
   → At the end, T should be completely filled
   → The logic for implementing the above step is an follows

```
for i=0 to 255
    S[i]=i;
    T[i]=K [i mod keylen]
```

→ After this, the T array is used to produce initial permutation of S

→ For this the following logic is used

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

2. Stream Generation

→ Once the state vector S is initialized, the input key will be discarded.

→ In this step, we swap S[i] with another byte in S, as per the mechanism decided by the implementation of S to produce the key stream 'k'.

→ The logic is as follows

```
i = 0;
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]) mode 256;
    K = S[t];
```

→ After this, the plaintext is XORed with the generated keystream 'k' to produce the Corresponding cipher text

→ RC5 (Rivest Cipher 5)

→ RC5 is a symmetric-key block-encryption algorithm

→ The main features of RC5 are it is quite fast and requires less memory for execution, making it suitable not only for desktop computers, but also for Smart cards and other devices that have a small memory capacity

→ The basic parameters in RC5 are

(a) the word size (input plain-text block size)

(b) number of rounds and

(c) number of 8-bit bytes of the key

→ All the parameters are of variable length and consists of the sizes shown below

Parameter	Allowed values
word size in bits (RC5 encrypt 2-word blocks at a time)	16, 32, 64
Number of rounds	0-255
Number of 8-bit bytes in the key	0-255

→ A particular instance of the RC5 algorithm is denoted as RC5-w/r/b where

w = word size in bits.

b = number of 8 bit bytes in the key

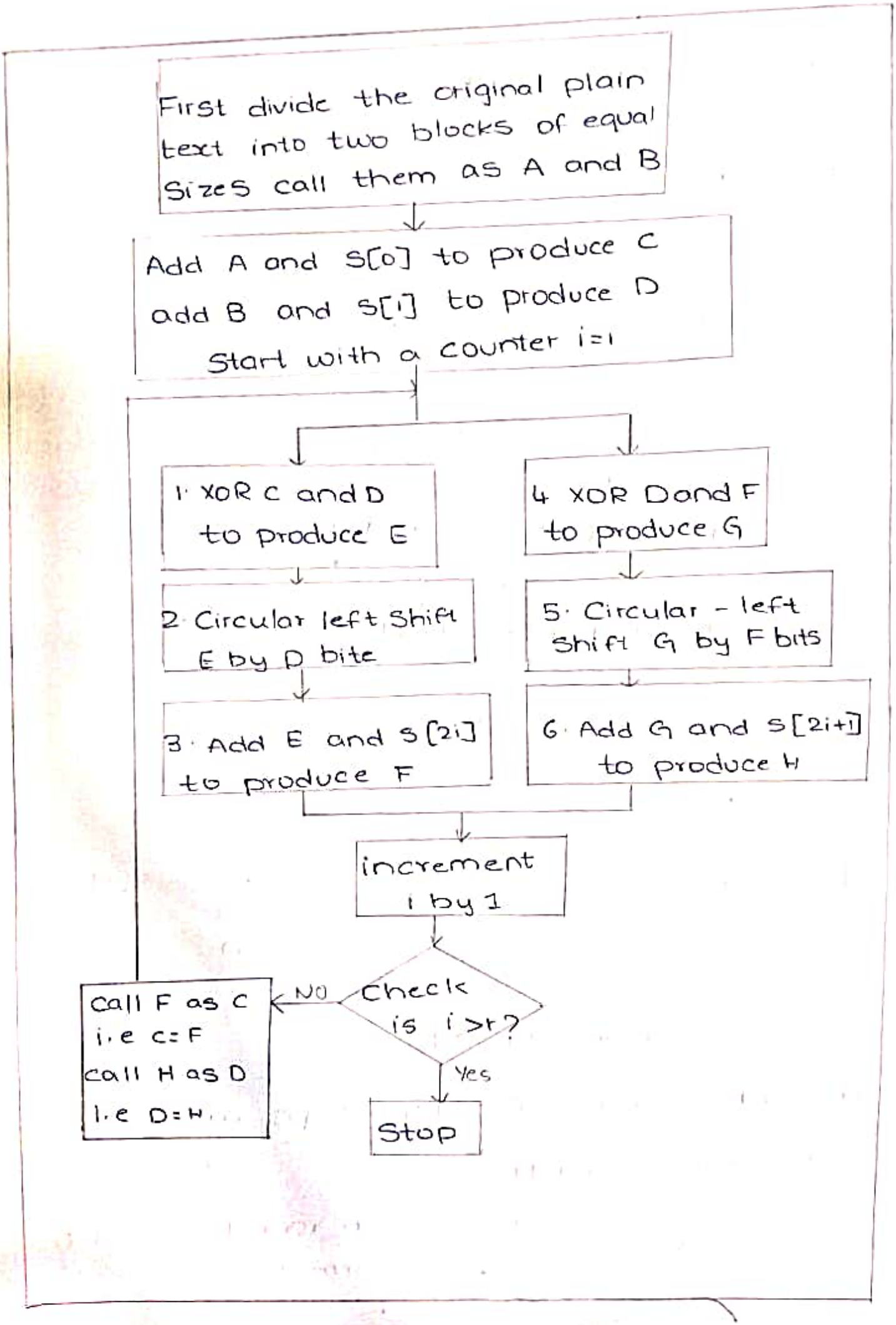
r = no. of rounds

→ Thus if we have RC5-32/16/16, it means that we are using RC5 with a block size of 64 bits (32x2 as RC5 uses 2 word blocks), 16 rounds for encryption and 16 bytes (i.e, 16x8 = 128 bits) in the key

→ RC5 principles of operation

The working of RC5 is shown below

(Next page)


First divide the original plain text into two blocks of equal sizes call them as A and B
Add A and S[0] to produce C
add B and S[1] to produce D
Start with a counter i=1
1. XOR C and D to produce E
2 Circular left Shift E by D bits
3. Add E and S[2i] to produce F
4 XOR D and F to produce G
5. Circular - left Shift G by F bits
6. Add G and S[2i+1] to produce H
increment i by 1
Check is i >r?
NO
Call F as C
i.e C=F
call H as D
i.e D=H
Yes
Stop

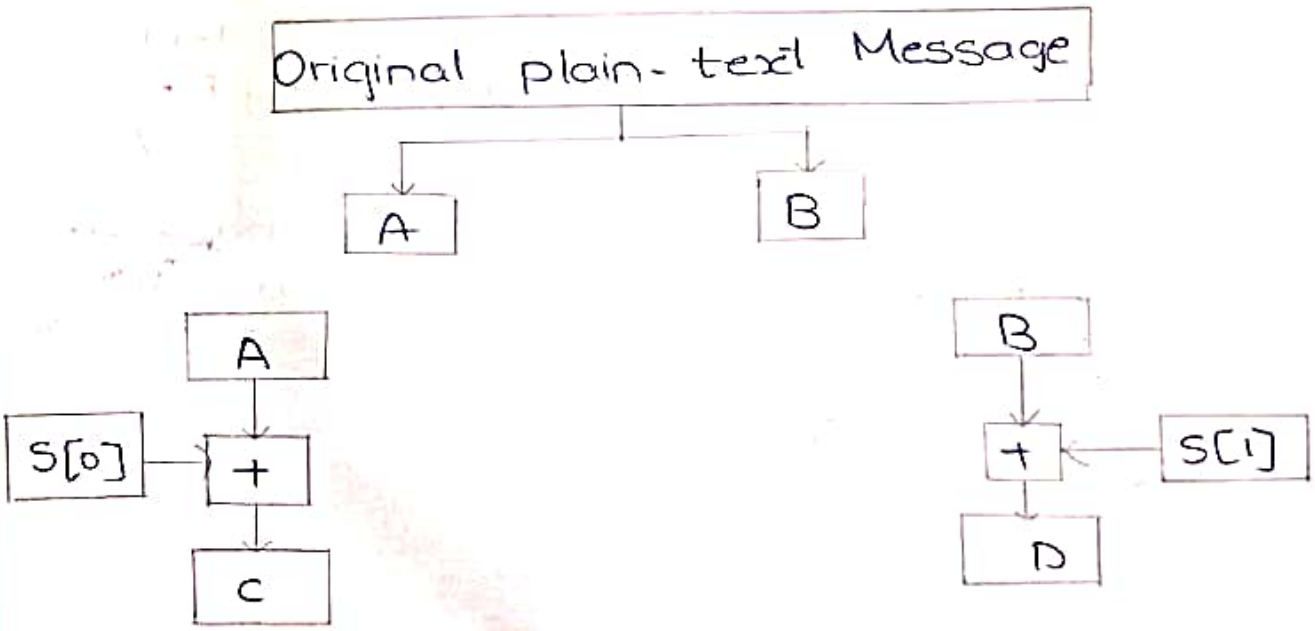
→ The step by step details of the algorithm is as below

1. One-time Initial Operation

→ This Consists of two simple steps

→ First the input plain text is divided into two equal sized blocks A and B

→ Then the first subkey i.e, S[0] is added to A and the second sub key i.e S[1] is added to B to produce C and D

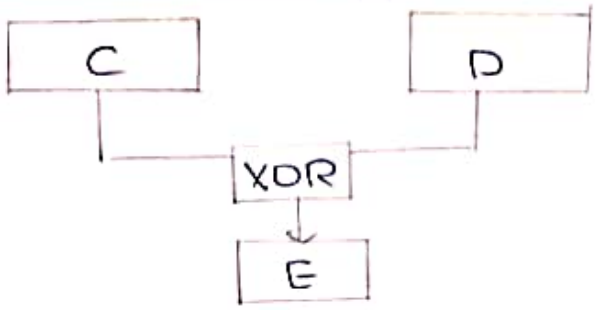


2. Details of one Round

In each round following steps are performed

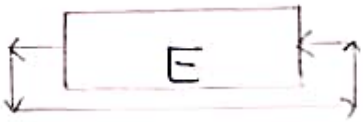
Step1 XOR C and D

→ In the first step of each round C and D are XORed together to form E



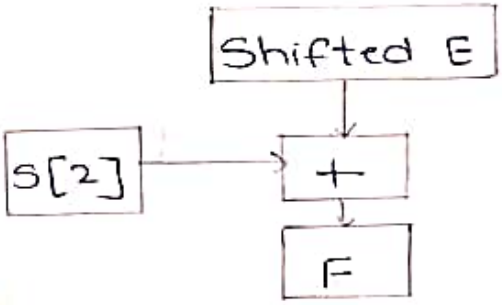
Step 2: Circular-left Shift E

→ Now, E is circular-left shifted by D position



Step 3: Add E and Next Subkey

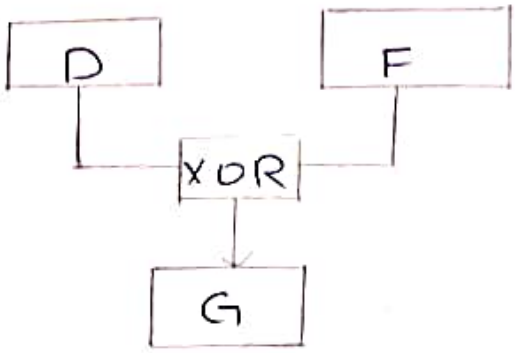
→ In this step, E is added to the next Subkey which is S[2] for the first round and S[2i] in general, for any round, where i starts with 1. The output of the process is F



Step 4: XOR D and F

→ This step is similiar to Step 1

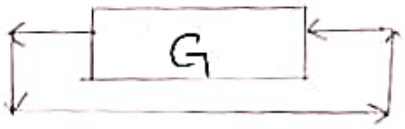
→ Here D and F are XORed to produce G



Step 5 : Circular - left shift G

→ This Step is similar to Step 2

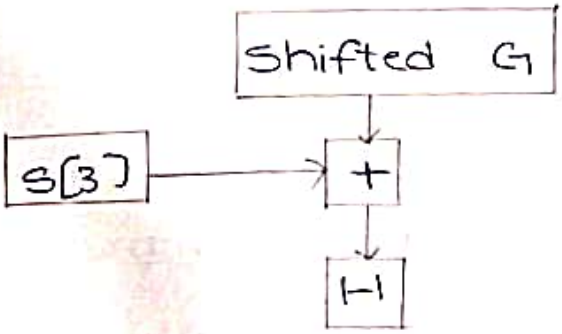
→ Here G is Circular - left Shifted by F positions



Step 6 : Add G and Next Subkey

→ In this Step G is added to next Subkey which is s[3] for the first Step and s[2i+1] in general

→ The output of this process is H



Step 7:

→ In this step, we check to see if all the rounds are over or not

→ For this, perform the following steps

- Increment i by 1
- check to see if i<r

→ Assuming that i in still less than r, we rename F as C and H as D and return back to step 1

3. Subkey Creation:

→ This is a two-step process

1. In the first step, the subkeys S[0], S[1].... .... are generated

2. The original key is called L. In the Second step the subkeys S[0], S[1]..... are mixed with the corresponding subportion of the original key L[0], L[1].....

Step 1: Subkey Generation

→ In this step two constants P and Q are used

→ The array of sub keys to be generated is called as S.

→ The mathematical form of the subkey

generation is shown below

$S[0] = P$

for $i = 1$ to $2(r+1)-1$

$S[i] = (S[i-1] + Q) \bmod 2^{32}$

Next i

Step 2: Subkey Mixing:

→ In this step, the subkeys $S[0], S[1]....$ are mixed with the subportions of the original key i.e, $L[0], L[1] .... L[c]$.

→ This process is shown mathematically as below

$i = j = 0;$

$A = B = 0;$

Do 3n times (where n is maximum of $2(r+1)$ and c)

$A = S[i] = (S[i] + A + B) <<< 3;$

$B = L[i] = (L[i] + A + B) <<< (A+B);$

$i = (i+1) \bmod 2(r+1);$

$j = (j+1) \bmod c$ (where c is last Subkey portion)

End

→ Advanced Encryption Standard (AES)

→ Also called Rinjindael Algorithm

→ AES is based on 128-bit blocks with 128-bit keys

→ The main feature of AES are

a) Symmetric and parallel structure:

→ This feature gives the implementers of the algorithm a lot of flexibility.

→ It also stands up well against Cryptanaly -sis attacks

b) Adapted to Modern Processors:

→ The algorithm works well with modern processors like pentinum, RISC and so on

c) Suited to Smart Cards:

→ The algorithm can work well with Smart Cards

→ AES is a block Cipher

→ The key size can be 128/192/256 bits

→ It 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.

→ 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 minimum number of rounds is 10 and the maximum number of rounds is 14

→ The steps described in the implementation of AES algorithm are

① Do the following one-time initialization process

(a) Expands the 16-bytes key to get the actual key block to be used

(b) Do one time initialization of the 16-byte plain-text block called State

(c) XOR the state with the key block

② For each round, do the following

(a) Apply S-box to each of the plain-text bytes

(b) Rotate row k of the plain-text, bytes block (i.e, state) k

(c) perform a mix columns operation.

(d) XOR the state with the key block

→ One time Initialization Process:

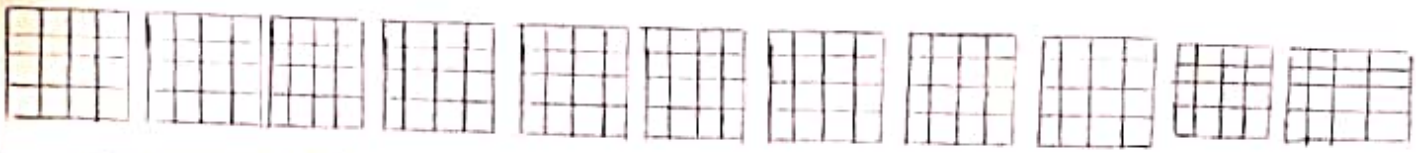
1. Expand the 16-byte key to get the actual key block to the used

→ The inputs to the algorithm are the key and plain text

→ The key size is 16 bytes (128 bits) in this case

→ This step expands this 16-byte key into 11 arrays each array contains 4 rows and 4 columns

16 byte key



expanded into 11 arrays each of Size 4x4

→ In other words, the original 16 byte key array is expanded into a key containing

$11 \times 4 \times 4 = 176$ bytes

→ One of these, 11 arrays are used in the initialization process and the other 10 arrays are used in the 10 rounds, one array per round

→

2. Do one time initialization of the 16-byte plain-text block called State

→ In this step, the 16-byte plain-text block is copied into the two-dimensional 4x4 array called State

→ The order of copying is in the column order

→ That is, the first four bytes of the plain-text block get copied into the first column of the state array, the next four bytes of the plain-text block get copied into the second column of the State array and so on.

→ This is shown in the figure below

16-byte plain-text block

B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15	B16

B1	B5	B9	B13
B2	B6	B10	B14
B3	B7	B11	B15
B4	B8	B12	B16

State array

3. XOR the State with the key block

→ Now, the first 16 bytes of the expanded key are XORed into the 16-byte state array

→ Thus, every byte in the State array is replaced by the XOR of itself and the corresponding byte in the expanded key

→ Processes in each round:

→ The following steps are executed 10 times, one per round

1) Apply S-box to each of the plain-text bytes

→ This step is very straight-forward

→ The contents of the state array are looked up into the S-box

→ Byte-by-Byte substitution is done to replace the contents of the state array with the repective entries in the S-box

2) Rotate row k of the plain-text block (i.e. state) by k bytes

→ Here, each of the four rows of the state array are rotated to the left

→ Row 0 is rotated 0 bytes, row 1 is rotated by 1 byte, row 2 is rotated by 2 bytes and row 3 is rotated by 3 bytes.

→ This helps in diffussion of data

Original array

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

Modified array

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

3 Perform a Mix-columns operation

→ In this step Matrix Multiplication is used

→ each column of the state array is multiplied with a constant matrix

4. XOR the state with the key block.

→ In this step, the resultant output of previous step is XOR-ed with the Correspon-ding key block

→ After all these rounds 128 bits of encrypted data is given back as output

→ RSA algorithm

→ The RSA algorithm is the most popular and proven assymmetric-key cryptographic algorithm

→ The RSA algorithm is based on the mathematical fact that is easy to find and multiply large prime numbers together, but it is extremely difficult to factor their product

→ The private and public keys in RSA are based on very large prime numbers

→ The algorithm itself is quite simple. However, the real challenge in the case of RSA is the selection and generation of the public and private keys

→ The whole process of how the public and private keys are generated and using them how we can perform encryption and decryption in RSA is shown below

1. Choose two large prime numbers P and Q
2. Calculate $N = P \times Q$
3. Select the public key E such that it is not a factor of $(P-1)$ and $(Q-1)$
4. Select the private key D such that the following equation is true

$$(D \times E) \bmod (P-1) \times (Q-1) = 1$$

5. For encryption, calculate the cipher text CT from the plain text as follows

$$CT = PT^E \bmod N$$

6. Send CT as the cipher text to the receiver
7. For decryption, calculate the plain text PT from the cipher text CT as follows

$$PT = CT^D \bmod N$$

→ Example of RSA:

1. Choose two large prime numbers P and Q
   let $P = 7$ and $Q = 17$
2. Calculate $N = P \times Q$
   We have $N = 7 \times 17 = 119$
3. Select the public key i.e, the encryption key E such that it is not a factor of $(P-1) \times (Q-1)$

→ Let us find $(7-1) \times (17-1) = 6 \times 16 = 96$

→ The factors of 96 are 2, 2, 2, 2, 2 and 3

→ we have to choose E such that none of the factors of E is 2 and 3

→ we cannot choose E as 4 because it has 2 as a factor

→ we cannot choose 15 because it has 3 as a factor

→ Let us choose E as 5 as it does not has factors as 2 and 3

4) Select the private key i.e, the decryption key D such that the following equation is true

$$(D \times E) \bmod (P-1) \times (Q-1) = 1$$

→ Let us substitute the values of E, P and Q in the equation

→ we have $(D \times 5) \bmod (7-1) \times (17-1) = 1$

→ That is, $(D \times 5) \bmod 96 = 1$

→ After some calculations, let us take $D = 77$. Then the following is true

$(77 \times 5) \bmod 96 = 385 \bmod 96 = 1$, which is what we wanted

5) For encryption, calculate the cipher text CT from the plain text PT as follows

$$CT = PT^E \bmod N$$

→ Let us assume that we wanted to encrypt plain text 10

Then we have

$$CT = 10^5 \bmod 119 = 100000 \bmod 119 = 40$$

6) Send CT as the Cipher text to the receiver

→ Send 40 as the Cipher text to the receiver

7) For decryption, calculate the plain text PT from the cipher text CT as follows:

$$PT = CT^D \bmod N$$

→ we perform the following

→ $PT = CT^D \bmod N$

→ i.e $PT = 40^{77} \bmod 119$

$= 10$

→ which is the Original plain text

→ Knapsack Algorithm:

→ knapsack Encryption algorithm is the first general public key cryptography algorithm

→ It is based on the knapsack problem

→ This is actually a simple problem

→ Given a pile of items, each with differ -ent weights, is it possible to put Some of them in a bag i.e, knapsack in such a way that the knapsack has a certain weight

→ That is if $M_1, M_2 \ldots M_n$ are the given values and S is the sum, find out bi so that

$$S = b_1 M_1 + b_2 M_2 + \ldots + b_n M_n$$

→ Each bi can be 0 or 1

→ A '1' indicates that the item is in the knapsack and a '0' indicates that it is not

→ A block of plain text equal in length to the number of items in the pile would select the items in the knapsack

→ The Cipher text is the resulting sum

→ For example: if the knapsack is 1,7,8,12,14, 20 then the plain text and the resulting Cipher text is as shown below:

plaintext	011011	111000	010110
knapsack	1 7 8 12 14 20	1 7 8 12 14 20	1 7 8 12 14 20
Cipher text	7+8+14+20 = 49	1+7+8 = 16	7+12+14 = 33

→ **Diffie - Hellman Key Exchange**

→ Diffie-Hellman key exchange is a method of digital encryption that securely exchanges cryptographic keys between two parties over a public channel

→ Let us assume that Alice and Bob want to agree upon a key to be used

for encryption/decryption messages

→ Then, the Diffie-Hellman key-exchange algorithm works as shown below

1. Firstly Alice and Bebo agree on two large prime numbers n and g.
   → These two integers need not to be kept Secret
2. Alice chooses another large random number 'x' and calculates A such that

$$A = g^x \bmod n$$

3. Alice sends the number A to Bob
4. Bob independently chooses another large random integer 'y' and calculates B such that

$$B = g^y \bmod n$$

5. Bob sends the number B to Alice
6. Alice now computes the Secret key K1 as follows

$$K1 = B^x \bmod n$$

7. Bob now computes the Secret key K2 as follows

$$K2 = A^y \bmod n$$

→ Example of Diffie-Hellman key exchange algorithm:

1. Firstly, Alice and Bob agree on two large prime numbers n and g
   → let $n=11$, $g=7$

2. Alice chooses another large random number x and calculates A such that

$A = g^x \bmod n$

→ let $x = 3$

Then we have $A = 7^3 \bmod 11 = 343 \bmod 11 = 2$

3. Alice sends the number A to Bob

→ Alice sends 2 to Bob

4. Bob independently chooses another large random integer 'y' and calculates B such that $B = g^y \bmod n$

→ let $y = 6$

Then $B = 7^6 \bmod 11 = 117649 \bmod 11 = 4$

5. Bob sends the number B to Alice

→ Bob sends 6 to Alice

6. Alice Computes the secret key K1 as follows

$K1 = B^x \bmod n$

we have $K1 = 4^3 \bmod 11 = 64 \bmod 11 = 9$

7. Bob now computes the Secret key K2 as follows $K2 = A^y \bmod n$

we have, $K2 = 2^6 \bmod 11 = 64 \bmod 11 = 9$

→ note that Diffie-Hellman key exchange algorithm can be used only for key agreement, but not for encryption (or) decryption of messages.

→ Principles of public key crypto systems:

→ Public key cryptography is a cryptographic technique that involves two distinct keys for encryption and decryption

→ That's why it is also known as asymmetric key cryptography

→ The two basic principles of public key cryptosystems are
- Confidentiality and
- authenticity

→ The public key crypto System is successful in achieving both these principles

→ To achive this, the message is encrypted using the senders private key

→ Now as the message is encrypted using the senders private key it is confirmed that the message has been prepared by the senders

→ No body is able to modify the message without having the senders private key.

→ So, public key crypto system has achieved authentication in both the terms data integrity and Source

→ Now, the message that was fixed first encrypted with the senders, private key encrypted using the intended receiver's public key.

→ The final cipher text can only be decrypted by the intended receivers private key which is only known to him.

→ In this way, the public key Cryptography achieves confidentiality.