Chapter 8

Encipherment Using Modern Symmetric-key Ciphers

Use of Modern Block Ciphers :-

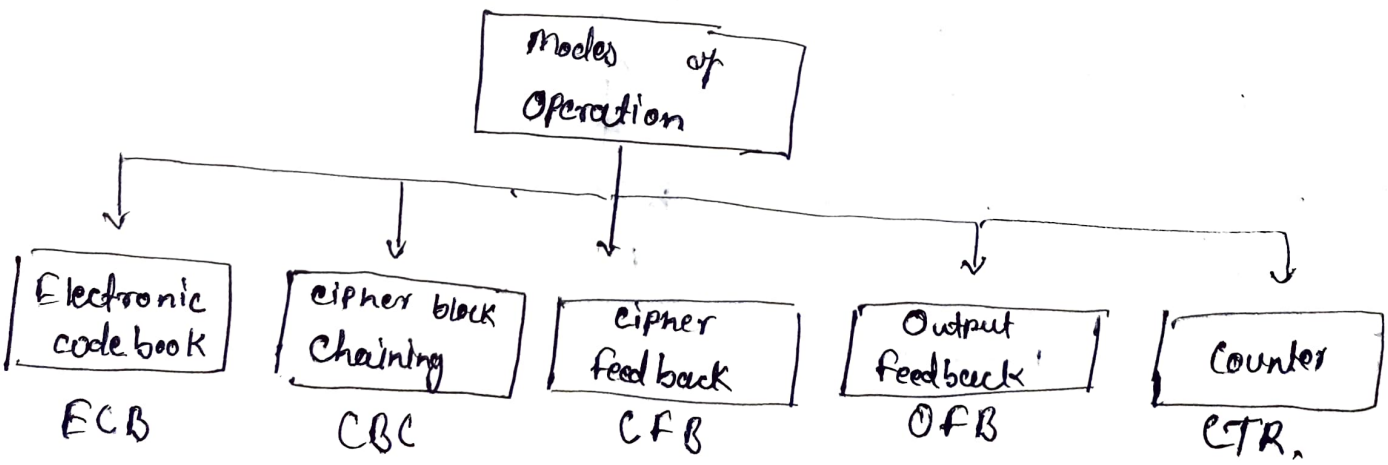
Symmetric-key encipherment can be done using modern block ciphers. The two modern block ciphers DES & AES are designed to encipher and decipher a block of text of fixed size.

DES encrypts ~~a block~~ and decrypts a block of 64 bits, AES encrypts and decrypts a block of 128 bits.

In real life applications, the text to be enciphered is of variable size and normally much larger than 64 or 128 bits.

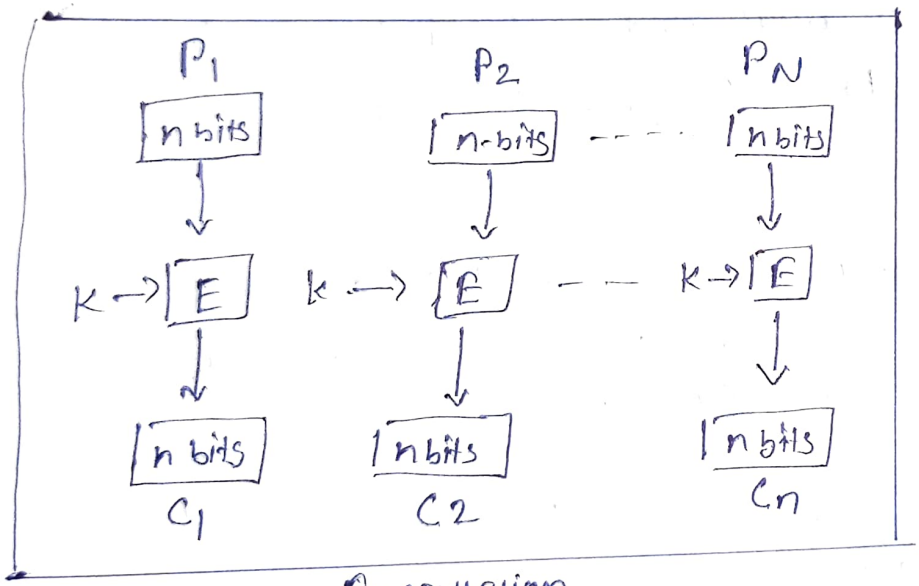
"Modes of Operations" have been devised to encipher text of any size employing either DES or AES.

We have 5 modes of operation :-



1. Electronic Codebook (ECB) Mode :-

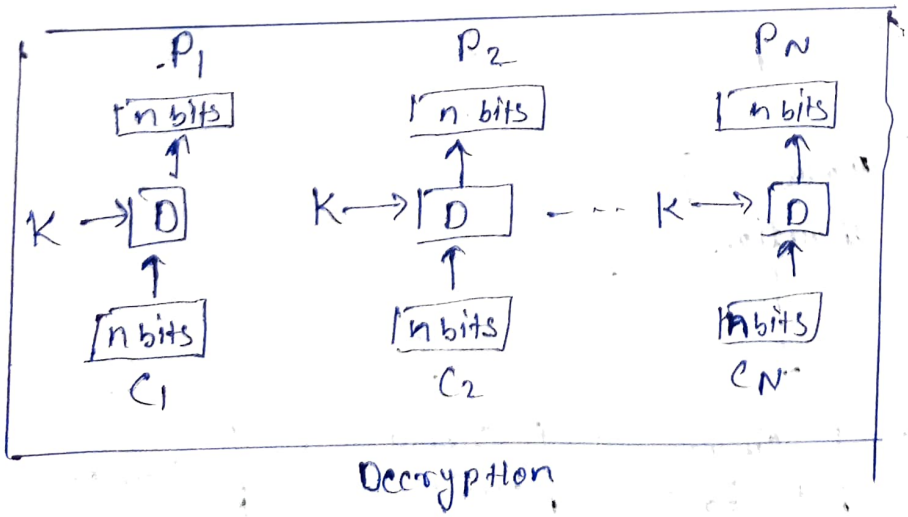
The simplest mode of operation is called the Electronic Codebook mode. The plaintext is divided into N blocks. The block size is n bits. If the plaintext size is not multiple of the block size, the text is padded to make the last block the same size as the other blocks. The same key is used to encrypt & decrypt each block.



Encryption

$C_i = E_K(P_i)$

$P_i = D_K(C_i)$



Decryption

Security Issue :-

Following are Security issue in CBC mode:-

① Pattern at block level are preserved.

Example :- ~~equal~~ same blocks in the plaintext become same block in the ciphertext.

If the Attacker finds out that ciphertext blocks 1, 5, 10 are the same., He knows that Plaintext blocks 1, 5, 10 are the same.

This is a leak in Security.

Because if we found the plaintext of one block then we are able to find other blocks also using exhaustive search attack.

② Because Each block is independent of each other, so it creates a opportunities for attacker to exchange some ciphertext block without knowing the key.

Example :- If Attacker know that a block 8 always conveys some specific information. ~~She~~ He can replace this block with other data. (Salary Block replaced By Employee)

Error Propagation :- A single Bit error in transmission can create error in several in the corresponding block. However the error does not have any effect on the other blocks.

Ciphertext Stealing :- we know that in ECB mode the padding must be added when we have less the n bits (block size). But in some real world Scenario when we want to encrypt the Data (or file) for security we don't want to store the data more then the original data Because of Memory sortage.

The Ciphertext Stealing (CTS) is a technique used for dealing with this issue in ECB.

In this technique the last two plaintext blocks P_{N-1} and P_N are encrypted differently

Applications of ECB :- (1) The ECB is Not secure for message of more than one Block.

(2) Another Advantage of this mode is that we can use parallel processing.

2. Cipher Block Chaining (CBC) Mode:

In this mode, each plaintext block is exclusive-ored with the previous ciphertext block before being encrypted.

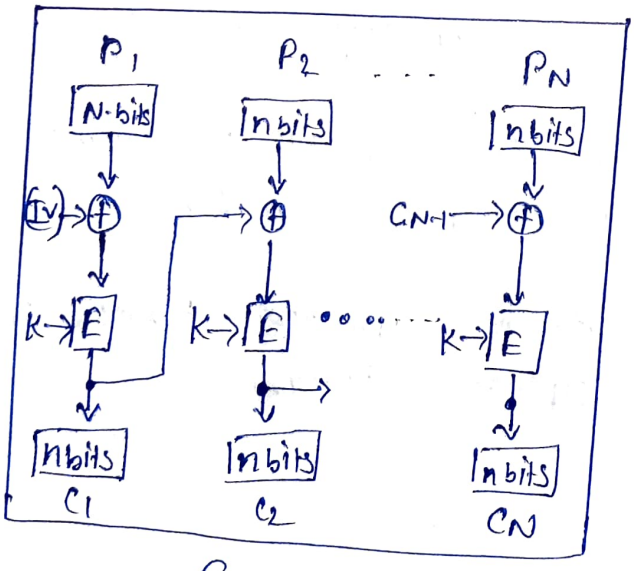
When a block is enciphered, the block is sent but a copy of it is kept in memory to be used in the encryption of the next block.

We have to think about that there is no ciphertext block before the first block. In this case the initialization vector (IV) is used.

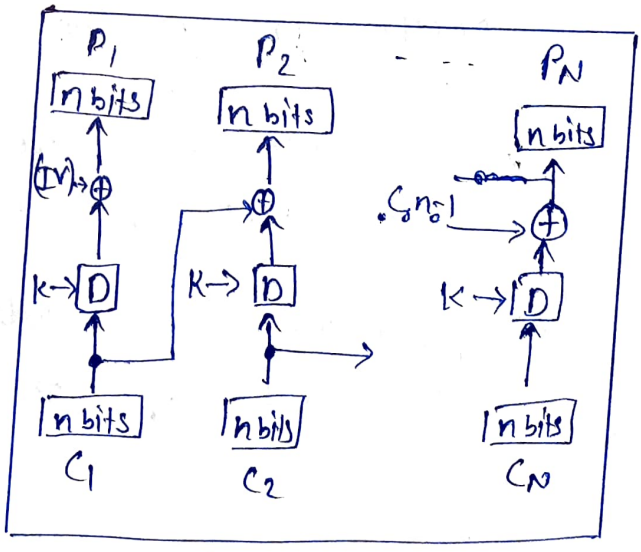
The Sender and receiver agree upon a specific Predetermined IV.

At the Sender side exclusive-oring is done before encryption.

At the receiver site exclusive-oring is done after decryption



Encryption



Decryption.

Initialization vector (IV) :- The Initialization vector (IV) Should be known by the Sender and the receiver.

Although keeping the vector secret is not necessary, the integrity of vector plays an important Role in the Security of CBC mode.

(IV) Should be kept from change.

If the attacker change the bit values of the (IV), It can change the bit value of the first block.

Security issue in CBC mode:-

(1) As we know that same plaintext Blocks Belongs to that message are encrypted into different ciphertext blocks. So that pattern at Block level are Not Preserved.

But If two message are equal, & their encipherment is the same if they use the same (IV).

As for this ~~reson~~, reason, some people recommend the use of a timestamp as an IV.

(2) Attacker can add some ciphertext blocks to the end of the ciphertext Stream.

Error Propagation :-

In the CBC mode single Bit error is self-recovered.
means. the single bit error affects only one or two blocks. then in the later block. it is recovered.

Ciphertext Stealing can also be applied to CBC.

---

3. Cipher feedback mode (CFB) :-

ECB and CBC encrypt and decrypt "block" of the message. The Block size is Predetermined according to the Encryption Algorithm (AES and DES)
64 bits 128 bits

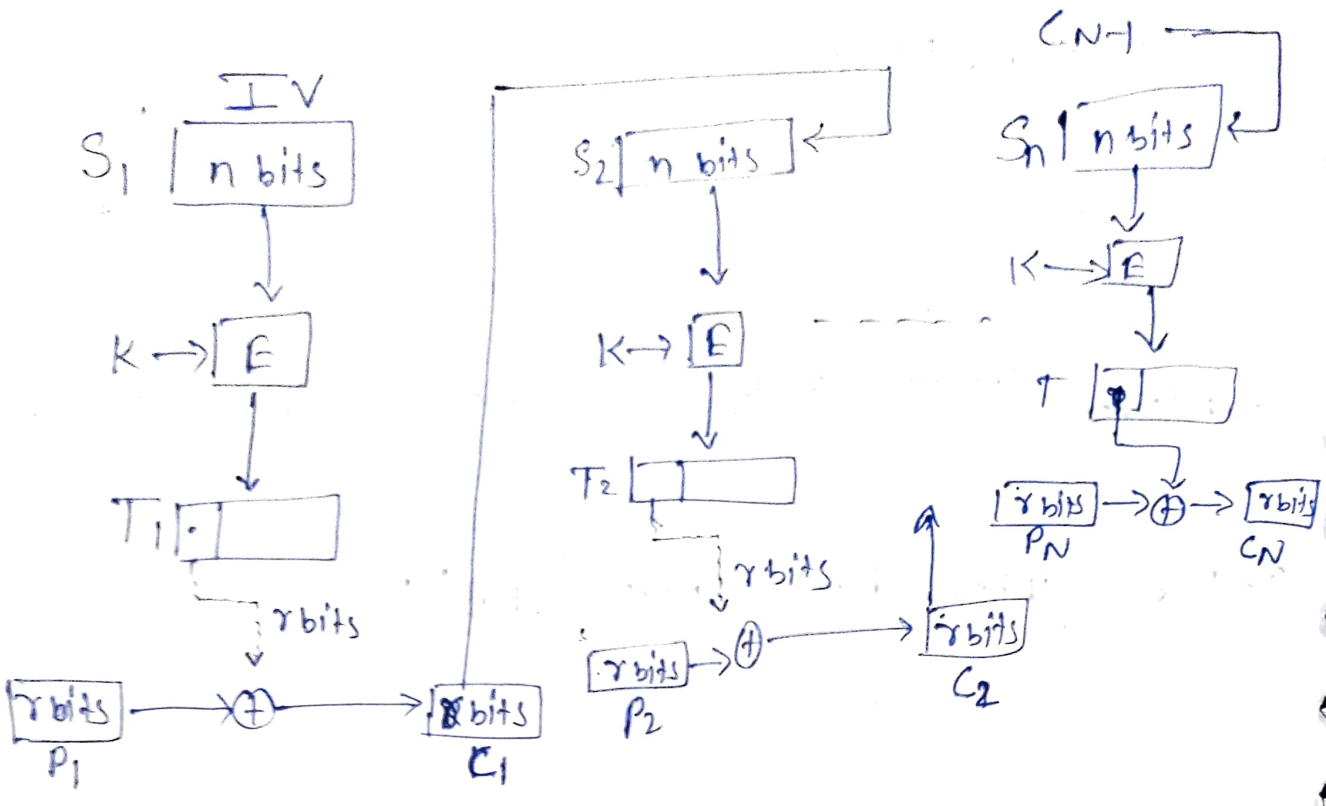
for this reason we have to add padding if the plaintext is smaller then the blocksize.

But in CFB mode the size of ~~Plaintext~~ ciphertext is same as Plaintext.

In this mode the size of the plaintext & the ciphertex is r Not n as AES & DES.

The AES or DES is Not used for Encryption or Decryption of the plaintext in this mode.
It is used for ~~Encr~~ Encryption of data of Shift register.

Encryption is Done By exclusive-oring the r-bit Plaintext block with r bits of Shift register.
Similarly the Decryption is also done by XORing the r bit ciphertext with r bits of Shift register



First the IV has a ~~n~~ n bits that is given to shift register S_1, then ~~after~~ there is No shifting is performed, It is given to the Encryption algorithm. ~~At the~~ After this it is given to the Temprory register T_1 from these n bits in this we took leftmost r bits, which are XoRed with r bit plaintext P_1 & converted to r bit ciphertext C_1. ~~then these r bits of ciphertext~~

After this the bits of shift register S_2 is shifted left By r bits then we append r bit cipher from C_1 from back then this process is continue until all blocks are not ciphered.

CfB as Stream cipher :-

The CfB is Non-synchronous Stream Cipher in which the key Stream is dependent on the ciphertext.

Some interesting point about this:-

1. No padding is required Because of this It can be named as Stream cipher.
2. It is useful in Real time communication
3. But CfB is less Efficient then CBC or ECB Because Block size is small (r)

Security issue:-

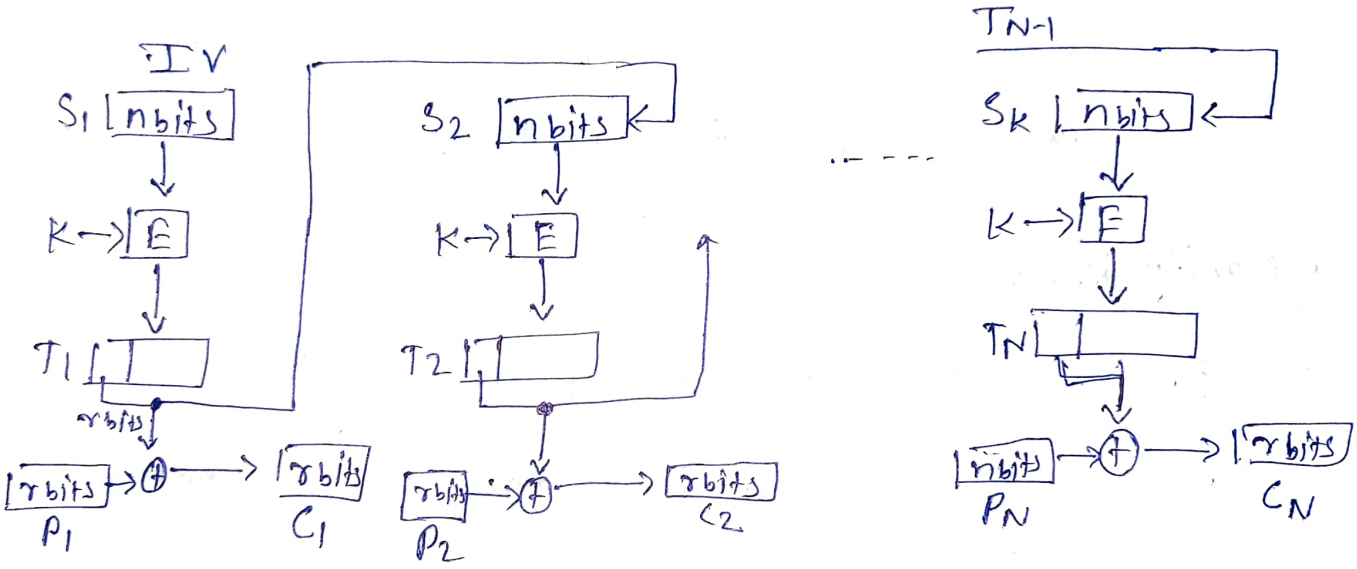
1. Just like CBC the patterns of the block level are not preserved.
2. At every time when we have to send message we have to select different IV.
3. Attacker can add some ciphertext Block to the end of the ciphertext Stream.

Error Propogation :- After the sigle Bit error in the ciphertext, the System recovers from the error.

Application :- The CfB mode is used for encipher blocks of small size such as one character or bit at a time.
There is No need for padding.

Output feedback (OFB) Mode :-

Output feedback mode (OFB) is very similar to CFB mode, with on difference : each bit in the ciphertext is independent of the previous bit or bits. This avoids the error propagation. If an error occurs in transmission, It does not affect the bits that follow



OFB as a Stream Cipher :- Like CFB, It is also Think as Stream cipher. The key stream, however is independent from the plaintext or ciphertext, which means that the Stream cipher is syncronous.

All the things are similar to CFB.

5. Counter (CTR) Mode :-

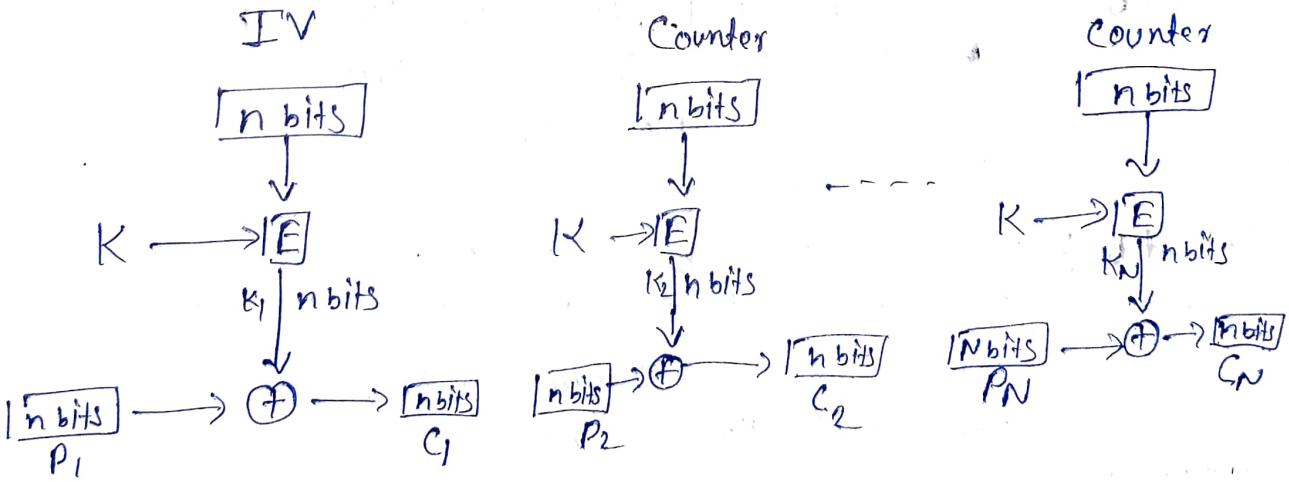
In the counter (CTR) mode, there is no feedback.

The pseudorandomness in the key stream is achieved using a counter.

The n-bit counter is initialized to a Pre-determined value (IV) and increamented Based on the Predetermined rule (mod 2^n).

To Provide a better randomness, the increment value can depend on block number to be Increamented.

The Plantext and ciphertext block have the same block size as a underlying cipher (AES & DES), Plaintext blocks of size n are encrypted to create ciphertext blocks of size n.



CTR uses the same Encryption function for Both Encipherment and decipherment.

CTR as a Stream Cipher :- Like CFB & OFB, CTR is actually a Stream Cipher.

Security issue :- Same as OFB mode.

Error Propagation :- Single Bit Error in the ciphertext effects only the corresponding bit in the Plaintext.

Summary about these modes :-

(1) ECB :- Each n-bit Encrypted independently with same key
It is a Block cipher.
Data unit size = n bits

(2) CBC :- Same as ECB., But each block first XORed with the previous ciphertext.
It is a Block cipher.
Data unit size n bits

(3) CFB :- Each r-bit block is XORed with an r-bit key. which is a part of previous cipher text
It is a Stream cipher.
Data unit size $r \leq n$ bits.

(4) OFB :- Same as CFB, but the shift register is updated by the previous r-bit key.
It is a Stream cipher.
Data unit size $r \leq n$ bits

5. CTR :- Same as OFB but counter is used instead of shift register.
It is a Stream cipher.
Data unit size is n bits.

USE of Stream ciphers :- The five modes of operation Enable the use of block cipher for encipherment of message or files in large units (ECB, CBC & CTR) & small units (CFB & OFB).

Stream cipher is more efficient for real-time Processing.

One of the Stream cipher is "RC4"

RC4 :- RC4 is a Stream cipher that was designed for RSA Data security.

RC4 is used in many data communication and networking protocols. including SSL/TLS & IEEE 802.11 wireless LAN.

It is a Byte oriented Stream cipher in which a byte of Plaintext is XORed with a byte of key to produce a Byte of ciphertext.

Data structures used :-

① State array S :- RC4 is based on the concept of state.

The size of state array is 256 where each indices contains an 8 bit Number from 0 to 255.

During the Encryption one block is selected & XORed with 8 bit plaintext to create a ciphertext.

The Block is selected randomly Based on Key. value.

② Key Array :- It is also similar to the State array. The size of this key array is also 256. ~~where~~ ~~each~~ each indices contains an 8 bit Number from (0-255)

It is used for selecting a random State for Encryption.

Steps in this RC4 :-

Step1 :- Key initialization :- Here we initialize a Key Array using a cipher key.

If the cipher key is less then 256 bytes then we repeate a cipher key untill it ~~Become~~ all the indices of key array is Not initialized.

Example :- Assum cipher key is of 4 bytes

(A B C D)

then the key Array is look like :-

K= | A | B | C | D | A | B | C | D | ..... | A | B | C | D |

0 1 2 3 4 5 6 7 252 253 254 255

Step2 :- Initialization of State Array :- first we initialize a state array By integer start from 0.

Ex :-

S= | 0 | 1 | 2 | 3 | 4 | 5 | - - - - - - | 254 | 255 |

0 1 2 3 4 5 254 255

then we randomly suffled the state array with the help of key array.

~~for i~~ ~~S[n] =~~

loop for 0 to 255 [n = (n + S[i] + K[i]) mod 256
Swap (S[i], S[n])]

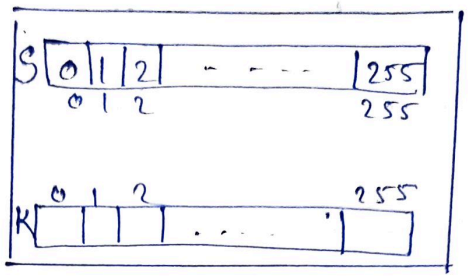
Step3 :- Key Stream Generation :- ~~The key Stream~~ is The keys in the key stream is genereted one by one.

i = (i+1) mod 256
n = (n + S[i]) mod 256
Swap(S[i], S[n])
K ← S[(S[i] + S[n]) mod 256]

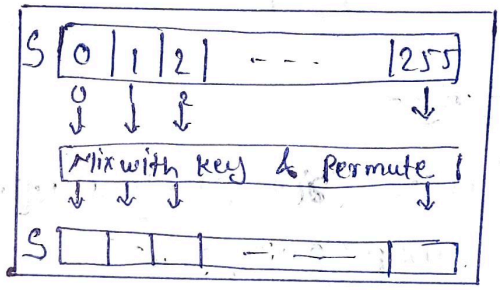
where i & n are indices from 0 – 255 which initialized with 0.

Step 4 :- Encryption/Decryption :- After k has been created, the plaintext byte is encrypted with k to create the ciphertext byte.

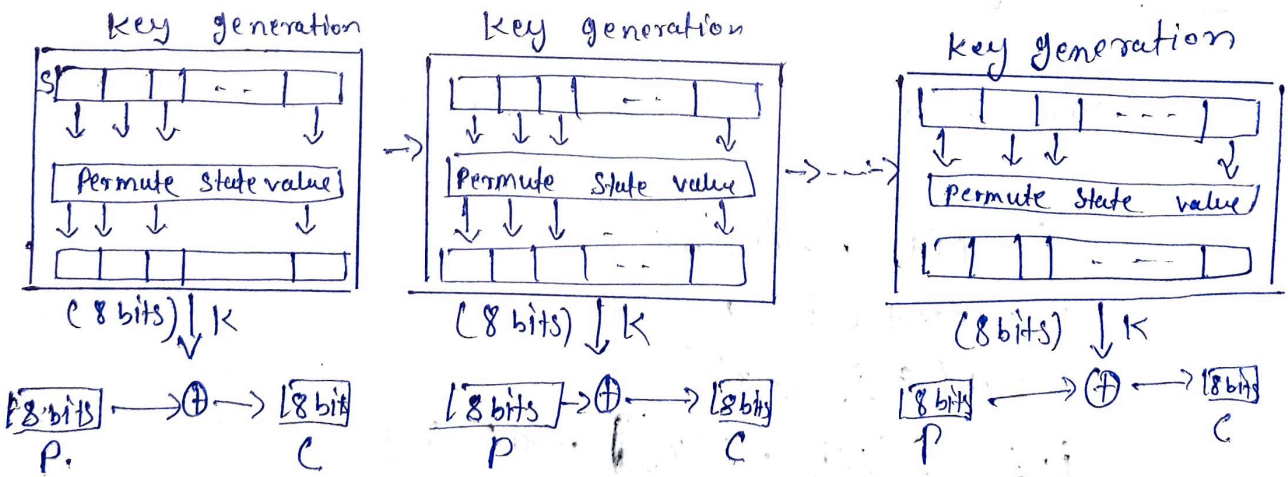
Step 1 :- State & key initialization



Step2 :- Initial State permutation



Step 3 & 4



Security issue :- It is believed that the cipher is secure if the size is at least 128 bits we can also use Different a cipher key for different session.

 # IDEA Algorithm :- (International data Encryption Algorithm)

It is a Block cipher, Replacement for DES

Key size 128 bits

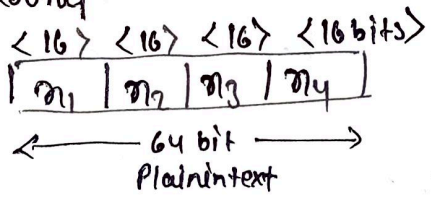
Block size 64 bits

Rounds = 8 Rounds.

we have 52 keys each round consumes 6 key & the remaining key used for final output from the 8th round. Each key is of 16 bits

Procedure :-

For one Round



(1) Multiply n_1 & first key ($S_1 = n_1 \times K_1$)

(2) Add n_2 & second key ($S_2 = n_2 + K_2$)

(3) Add n_3 & 3rd key ($S_3 = n_3 + K_3$)

(4) Multiply n_4 & 4th key ($S_4 = n_4 + K_4$)

(5) ~~$S_5 = S_1 + \ldots$~~

(5) $S_5 = S_1 \oplus S_3$

(6) $S_6 = S_2 \oplus S_4$

(7) ~~$S_7 = S_5 \oplus K_5$~~ $S_5 \times K_5$

(8) ~~$S_8 = S_6 \times K_6$~~ $S_6 \times S_7$

(9) $S_9 = S_8 \times K_6$

(10) $S_{10} = S_7 + S_9$

New $n_1' = S_1 \oplus S_9$

New $n_2' = S_3 \oplus S_9$

New $n_3' = S_2 \oplus S_{10}$

New $n_4' = S_4 \oplus S_{10}$

} Inputes for the Next Round

After the 8 Rounds.

(1) Multiply $n_1' \& K_1 \longrightarrow y_1$

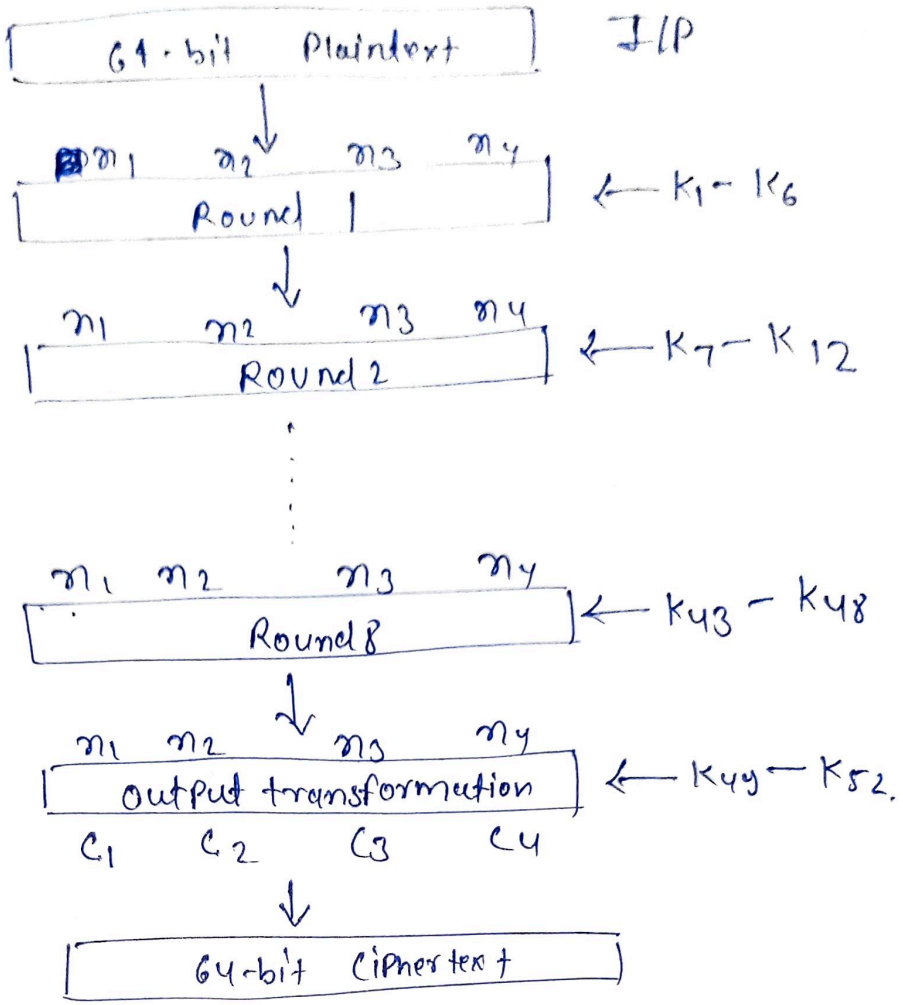
(2) ~~Multiply~~ Add $n_2' \& K_2 \longrightarrow y_2$

(3) Add $n_3' \& K_3 \longrightarrow y_3$

(4) Multiply $n_4' \& K_4 \longrightarrow y_4$

then Concatinate

$y_1 \; y_2 \; y_3 \; y_4$


64-bit Plaintext
I/P
x_1 x_2 x_3 x_4
Round 1
$K_1 - K_6$
x_1 x_2 x_3 x_4
Round 2
$K_7 - K_{12}$
x_1 x_2 x_3 x_4
Round 8
$K_{43} - K_{48}$
x_1 x_2 x_3 x_4
Output transformation
$K_{49} - K_{52}$
C_1 C_2 C_3 C_4
64-bit Ciphertext