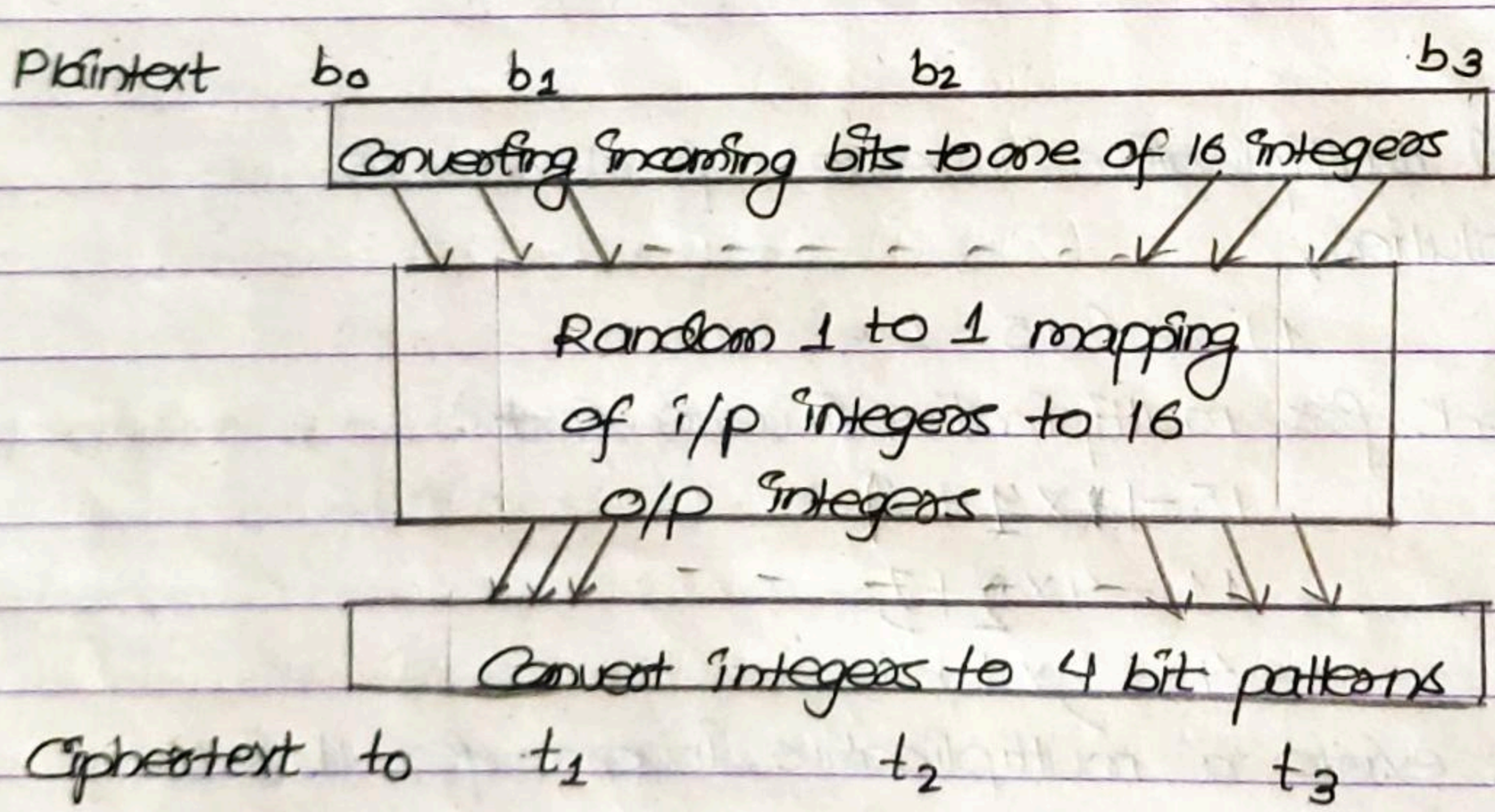
4 MODERN SYMMETIC CIPHERS

1) Binary Block Substitution

In a modern block cipher, we replace a block of N-bits from plaintext with a block of N-bits from a ciphertext. This general idea is illustrated in the figure below: [for n=4 (although N is set to 64 or its multiples)]



⇒ Firstly, the 4 bit patterns are converted into 2^4 (=16) different possible patterns we can represent these pattern by an integer between 0 & 15. So, bit pattern will be,

0000 → 0

0001 → 1

⋮

1111 → 15

ideal
⇒ In block cipher, relationship between i/p & o/p block is completely random. But it must be convertible for decryption work. Therefore,
⇒ It has to be 1 to 1 mapping (meaning that at each i/p block it is mapped to a unique o/p block).

⇒ The mapping between i/p & o/p block can also be constructed as a mapping from integers corresponding to i/p block to the integers corresponding to the o/p blocks.

⇒ The encryption key for the ideal block cipher is the code-block itself meaning the table shows relationship between i/p & o/p blocks.

⇒ In the above figure, an ideal block cipher is shown that uses block of size 4. Each block of 4-bits in the plaintext is converted into 4 ciphertext bits.

Size of encryption key for binary block substitution

Key

Consider a 64 bit block encryption with a 64 bit ~~block~~, we can consider each i/p block as 2^{64} integers and for each such integer we can specify o/p block of 64 bits. We can construct the code book by displaying just the o/p blocks in the order of corresponding i/p blocks such a code ~~block~~ block will be of size 64×2^{64} i.e. nearly equal to 10^{21}. This implies that the encryption key for an ideal block cipher using 64 bit blocks will be of size 10^{21}. This size of the encryption key makes the ideal block cipher and in practical idea thinking of logical issues related to transmission, storage and processing such large keys.

5th June, 2019
Wednesday

Shannon's Theory of Confusion & Diffusion

⇒ Block cipher looks extremely large substitution as we need table of 2^{64} entries for 64-bit blocks.

⇒ Using the idea of product cipher in 1949, Claude Shannon introduced idea of substitution permutation (S-P) network called modern substitution transpose product cipher. These form basics for modern block cipher.

⇒ S-P network based on two primitive cryptographic operation we've seen → substitution (S-box)
→ permutation (P-box)

⇒ provides confusion and diffusion in messages.

Diffusion

⇒ dissipates the statistical structure of plaintext over block of ciphertext.

The mechanism of diffusion seeks to make the statistical relationship between the plaintext & ciphertext as complex as possible in order to deduce the key.

Confusion

⇒ makes the relationship between ciphertext & key as complex as possible to make the relationship between the statistics of ciphertext and the value of encryption key as complex as possible, against thwart attempts to discover the keys.

Feistel Cipher Structure

⇒ It implements Shannon's diffusion and confusion theory (S-P network concept) based on invertible product cipher.

⇒ In early 1970 IBM developer Horst Feistel introduced Feistel cipher and it requires first implemented by Feistel and Don Coppersmith in Lucifer cipher.

⇒ The Feistel cipher is a system based on same basic algorithm for encryption and decryption. The Feistel cipher structure consist of multiple rounds of processing of the plaintext with each round consisting of substituting round followed by a permutation round as shown in the figure below:-

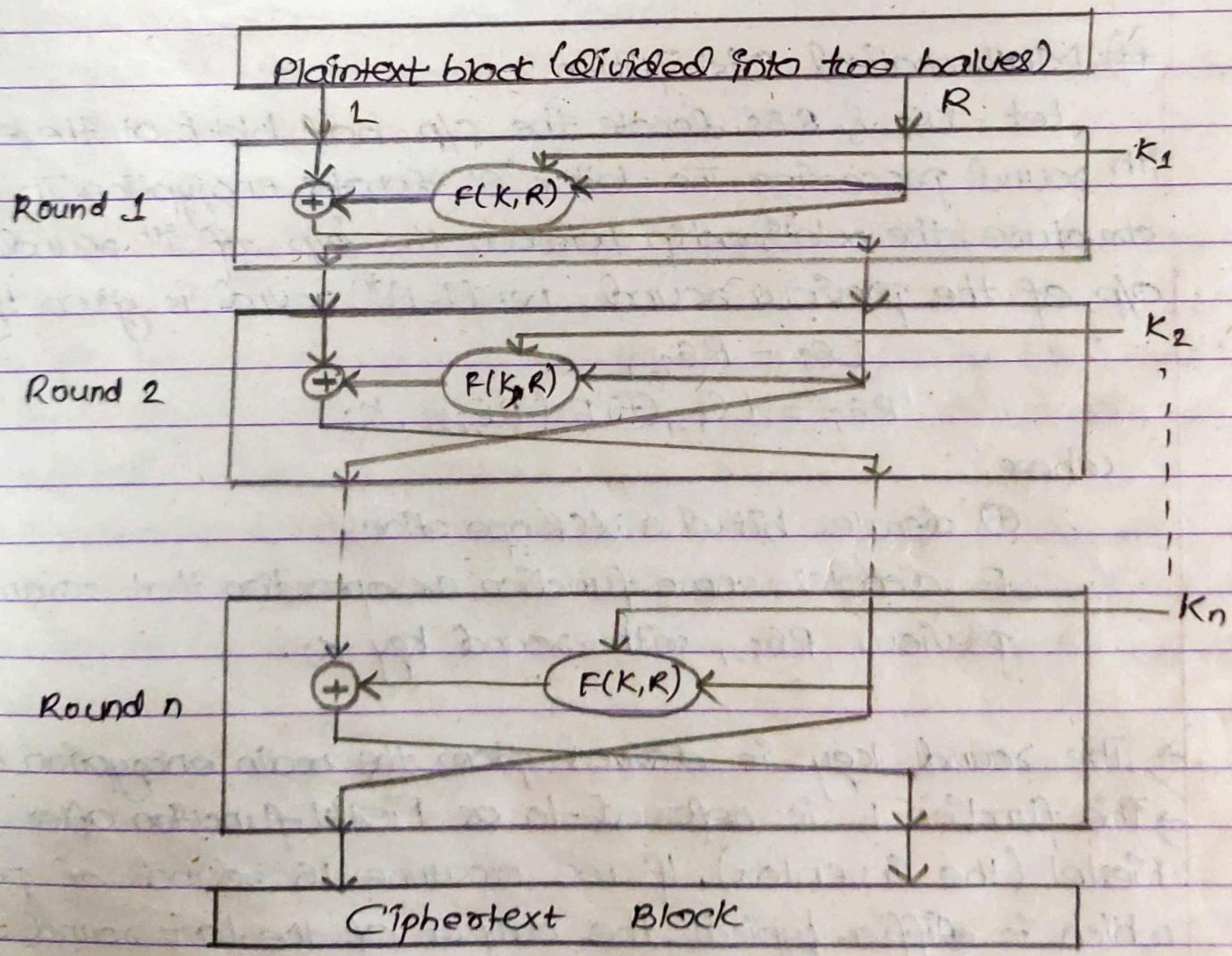


Fig: Feistel structure of Symmetric key Cryptography

6th June, 2019
Thursday

⇒ The plaintext input block is divided into two halves at each round and we denote these halves with L and R for left block and right block respectively. In each round, the right half of the block i.e. 'R' goes through an operation that depends upon R and the encryption key K.

⇒ The permutation step at the end of each round consists of swapping the modified L & R. Hence the current L for the next round would be R and the current R for the next round would be L. The process continues for the next 'N' rounds.

Mathematical Description

Let LE_i & RE_i denote the o/p half block at the end of i^{th} round processing. The letter 'E' denotes encryption. In Feistel structure, the relationship between the o/p of i^{th} round and o/p of the previous round i.e. $(i-1)^{th}$ round is given by.

$$LE_i = RE_{i-1}$$

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

where,

$\oplus$ denotes bit wise XOR operation

F denotes some function or operation that scrambles previous RE_{i-1} with round key K_i.

⇒ The round key is derived from the main encryption key.

⇒ The function F is referred to as Feistel function after Horst Feistel (the inventor). If we assume 16 rounds of processing which is typical the output of the last round of the processing is given by,

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

$LD_1 = LE_{16}$

Decryption of ciphertext based on Feistel structure

The decryption algorithm is exactly the same as encryption algorithm with the only difference that the round key are used in reverse order. The output of each round during decryption is the input to the corresponding round during encryption except for left right switch between the two halves. This property holds true regardless of the choice of Feistel function.

Data Encryption Standard (DES)

⇒ IBM in 1974 created DES which was based on their Lucifer algorithm.

⇒ Symmetric key block cipher

⇒ Remarkably well engineered algorithm which has a powerful influence in cryptography.

⇒ DES has a key length of 56 bits (+8 parity bits) and a block length of n=64 bits. It consists of 16 rounds of what is called "Feistel Network".

⇒ Electronic Frontier Foundation broke DES in 22hrs & 15 mins.

⇒ NIST then issued a new directive which required the organization to be triple DES (3-DES) i.e. three consecutive application of DES.

⇒ The round keys are generated from the main key by permutation.

⇒ The key length is of 48 bits.

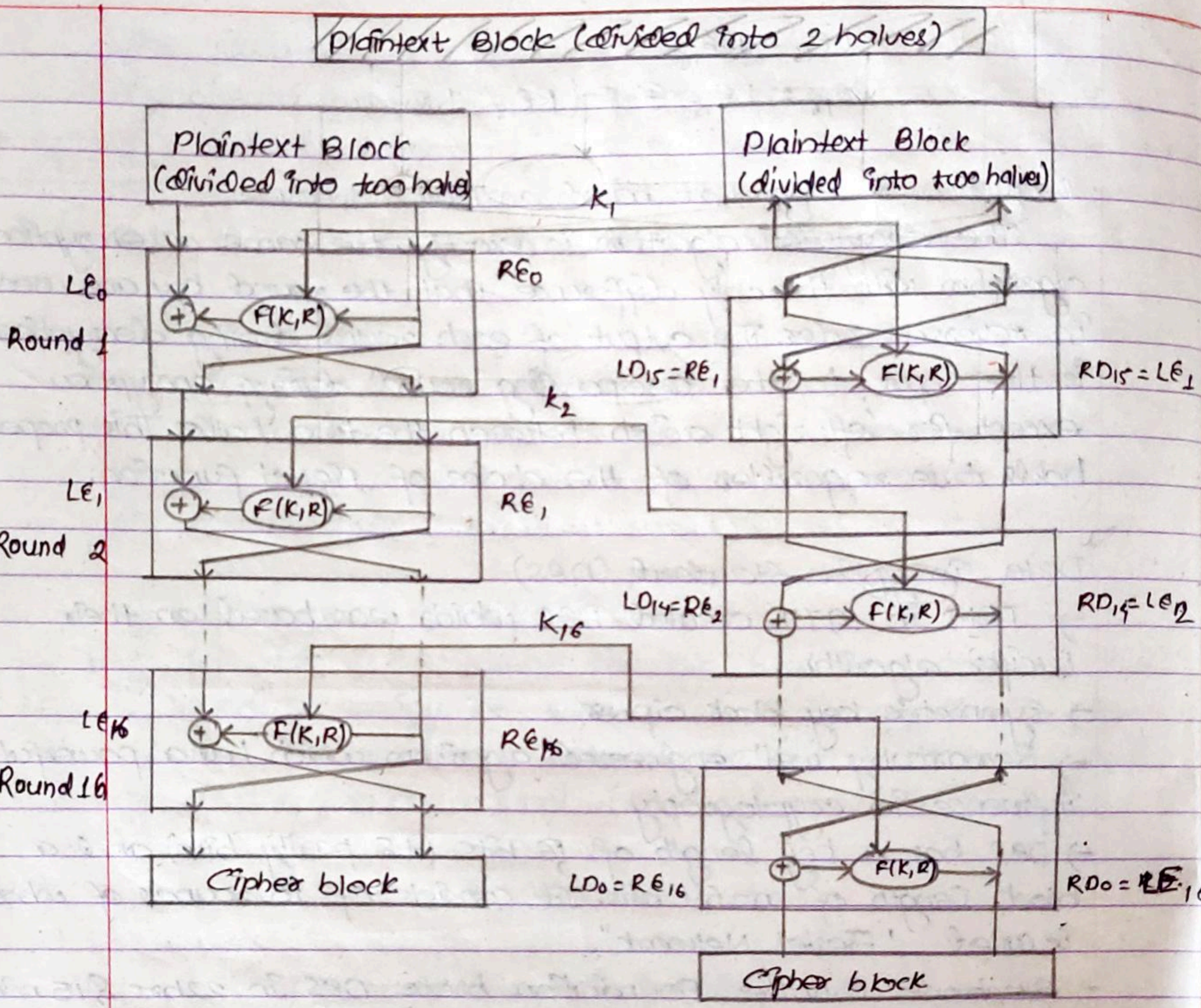


Fig :- Encryption & Decryption in Fiestel structure with round of 16

To prove the claim of encryption and decryption, let RD_i & LD_i denote the left half and the right half i^{th} round. Now, we know that

LD_1 & RD_1 occured in the first decryption round. The relationship between two halves that are input to the first decryption and the output of the decryption algorithm is given by

$$LD_0 = RE_{16} \quad \& \quad RD_0 = LE_{16}$$

Now, we can write the following equation for the output of the first decryption round

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

Again,

$$RD_1 = LD_0 \oplus F(RD_0, K_{16})$$
$$= RE_{16} \oplus F(RE_{15}, K_{16})$$
$$= \left(LE_{15} \oplus F(RE_{15}, K_{16}) \oplus F(RE_{15}, K_{16})\right)$$
$$= LE_{15}$$

This means that except for the left-right switch, the output of the first round of decryption is same as input of the last stage of encryption round.

$LD_0 = RE_{16}$	$RD_1 = LE_{15}$
$RD_0 = LE_{16}$	$LD_1 = RE_{15}$

The following identities have been used to derive the above expression.

12th June, 2019
Wednesday

$A \oplus (B \oplus C) = (A \oplus B) \oplus C$

$A \oplus 0 = A$

$A \oplus A = 0$

Algorithm Implementation of DES

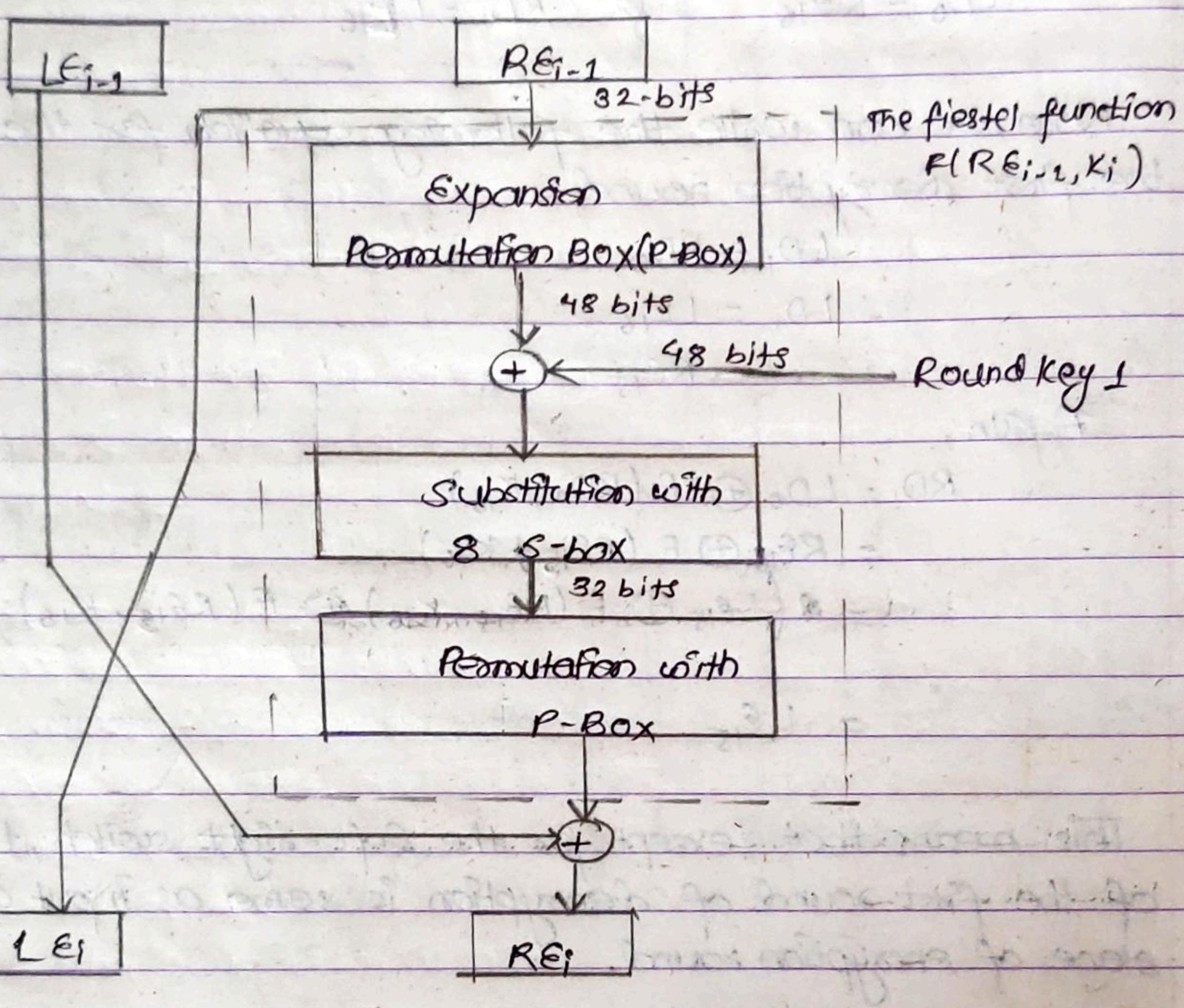
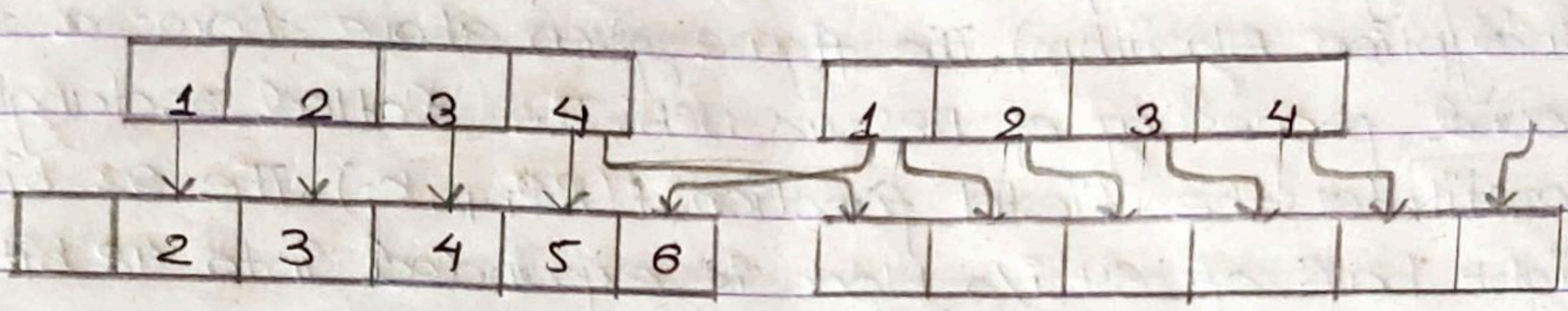


Fig: Single Round Processing of DES

① Expansion P-Box (32 bits → 48 bits)



1 → 2 , 2 → 3 , 3 → 4 , 4 → 5

6 → 1st bit of next block

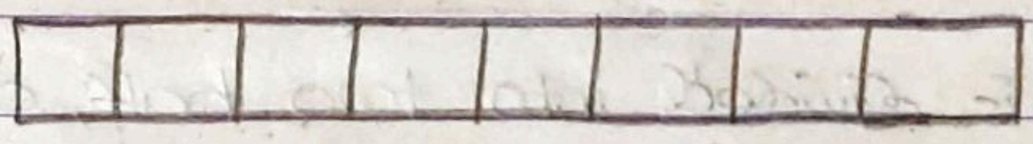
② Whitener (XOR-operation)

XOR between 48 bits, key & 48 bit o/p from expansion box.

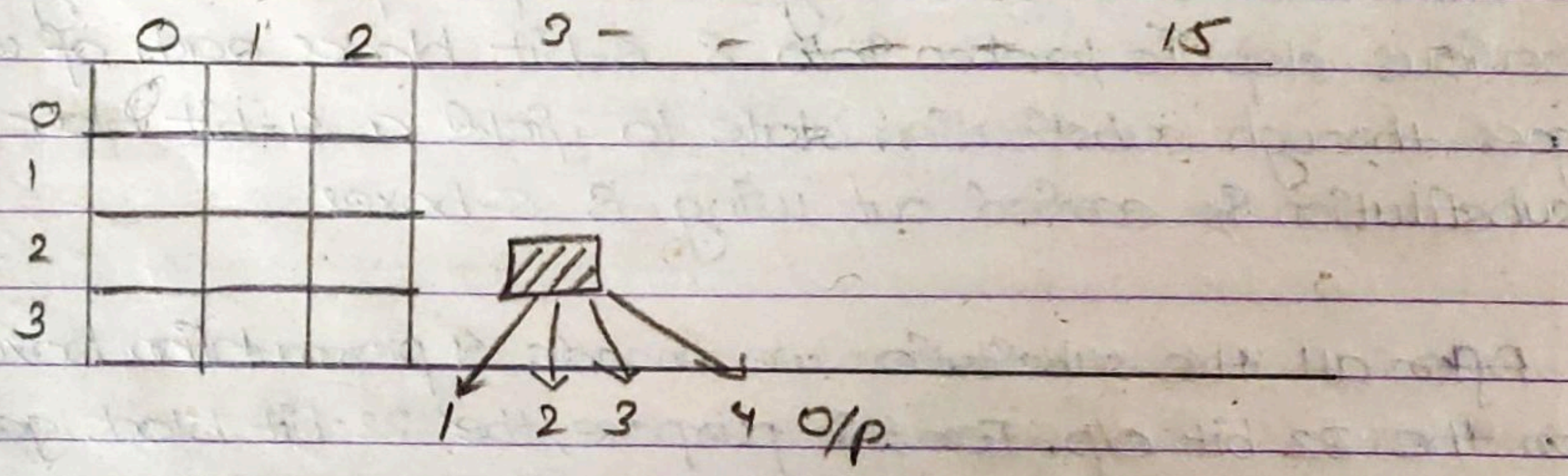
③ Substitution (S-Box)

⇒ 6 bit i/p —transform→ 4 bit o/p.

⇒ 8 bit are used



Each S-box has a different table.



The algorithm implementation of DES is called DEA (Data Encryption Algorithm). The figure given above shows a single round processing of DES or DEA. The dotted rectangle constitutes the Feistel function $f(R_{i-1}, K_i)$. The 32 bit right half of 64 i/p block is expanded into 48 bit block using expansion P-boxes. This is refered to as expansion permutation step or E-step. The E-step consists of the following :-

1) First, the 32 bit block is divided into 8-4 bit nibbles then an additional bit on the left of each bit is added which is the last bit of the previous nibble.

2) An additional bit to the right of the each nibble is attached at the end which is the beginning of next 4 bit ~~level~~ nibble.

The 56 bit key is divided into two half, each half is separately shifted and combined to yield a 48 bit key. The 48-bit of expanding output produced by that E-step is XOR with the round key. The output produced by previous step is broken into 8 6-bit blocks each of which goes through substitution state to yield a 4-bit block. The substitution is carried out using 8 S-boxes.

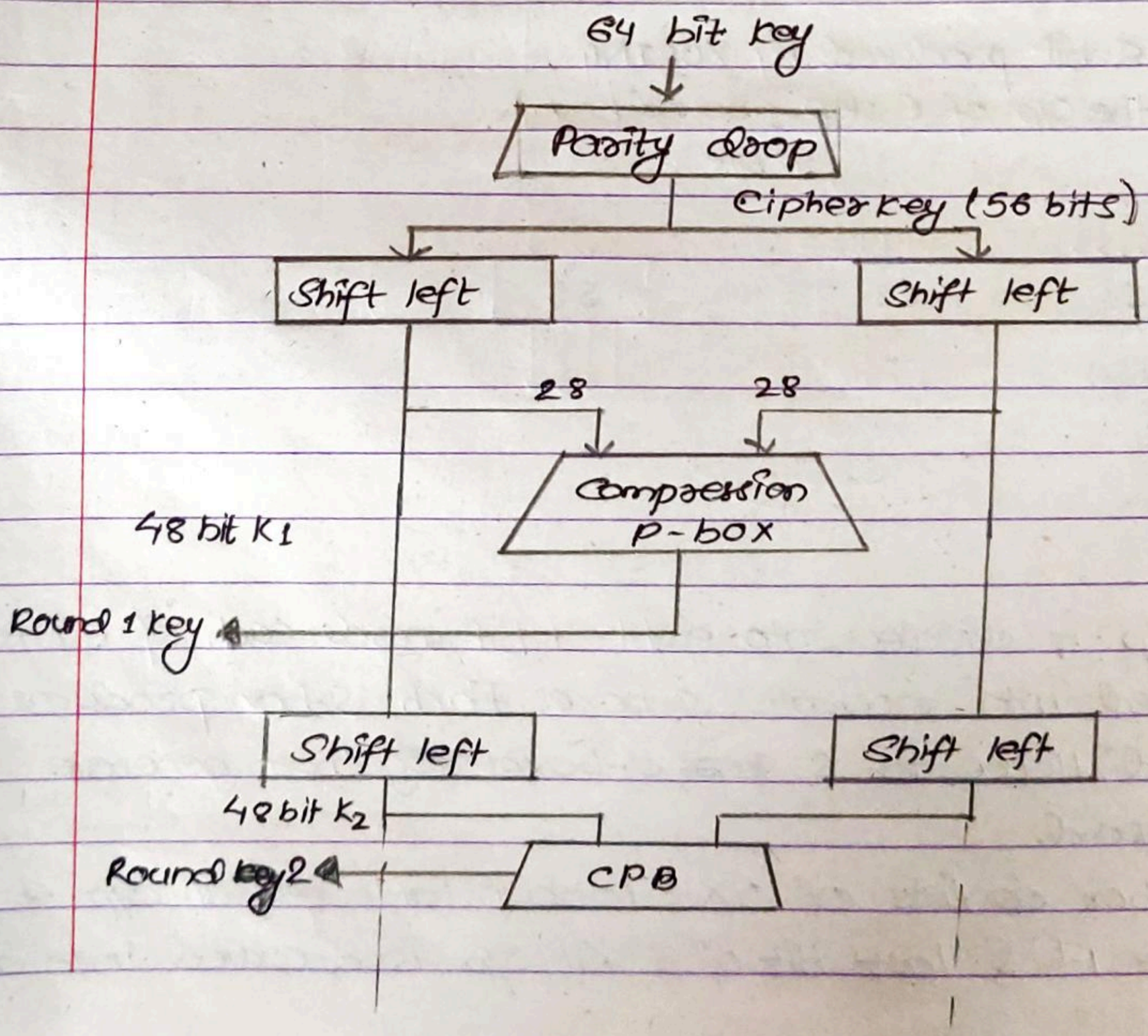
After all the substitution we encode 4 permutation boxes on the 32 bit o/p. For this purpose, the 32-bit block goes through P-box permutation. The o/p of P-box is then XOR with the left half of 64-bit block that we started with and this o/p will act as the right half of next round.

The main goal of substitution step implemented by S-block is to introduce diffusion i.e. each plaintext bit must be acting to affect as many ciphertext bits as possible.

The strategy of creating many rounds key from the main key is meant to introduce confusion into the encryption process i.e. the key must affect as many bit as possible of the o/p of the ciphertext.

Key Generation

The round key generator generates 16 48-bits key from the 56-bit cipher key. Actually the key length is 64 bit out of which 8 parity bit has been dropped.

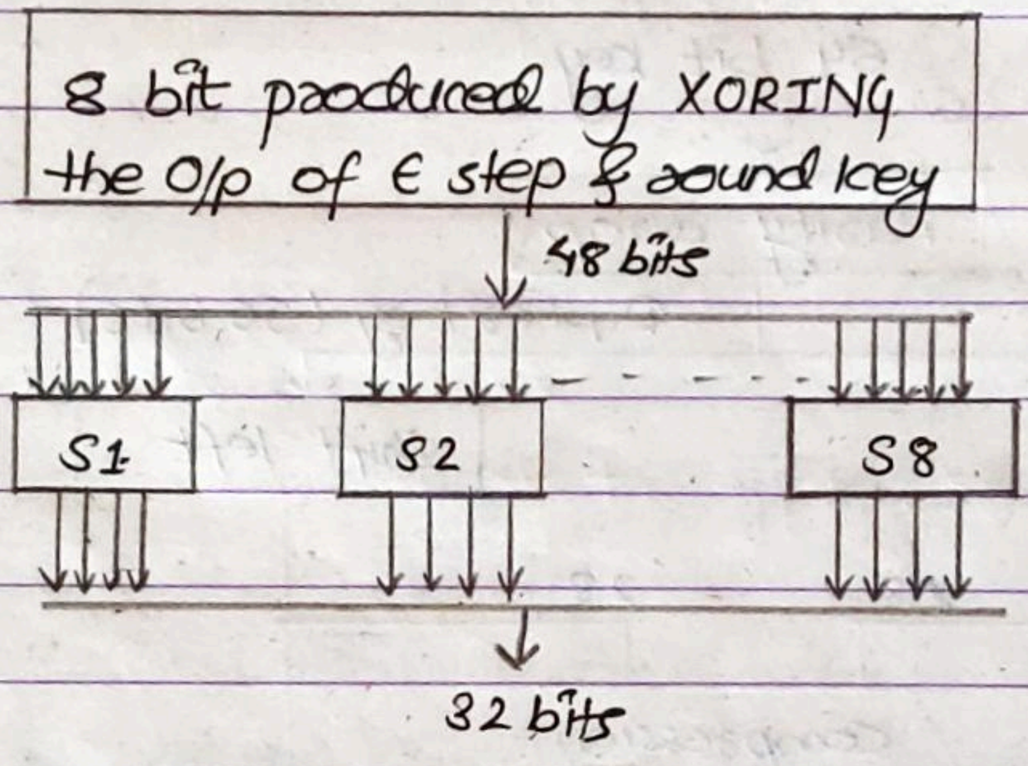


The left shift operation occurs according to the following scheme.

Rounds	Bit shift
1, 2, 9, 16	1 bit
others	2 bit

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
key shift	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1
Total shift	1	2	4	6	8	10	12	14	15	17	19	21	23	25	27	28

S-Box Substitution Step



48 bit i/p is divided into eight 4 bit words each of 6 bits which are fed into separate S-boxes. Each S-box produces a 4 bit word. Hence, the 8 S-boxes together generate 32 bit o/p word.

Each S-box consists of 4X16 lookup table for an o/p 4 bit word. The 1st & last bit of 6 bit o/p is decoded into one

of 4 rows & middle 4 bit decoded into 16 columns of the lookup table.

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

Fig: Table for S-box S1

Permutation in the Feistel Function

The last step of Feistel function is permutation with P-box. The permutation is shown below:

P-box Permutation							
15	6	19	20	28	11	27	16
0	14	22	25	4	17	30	9
1	7	23	13	31	26	2	8
18	12	29	5	21	10	3	24

The permutation table says, 0th o/p bit will be the 15th i/p bit & 1st bit be the 6th i/p bit & so on, for all 32 bits o/p that is obtained from the i/p.

The basic process of enciphering a 64 bit data block using DES consists of
- an initial permutation

-16 rounds of complex independent calculation 'f'.
- A final permutation being inverse of IP.

This can be described as

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

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

and forms one round in the S-P network.

- An initial permutation of the key P(i) which selects 56 bits into 2 halves of 28 bits.
- The 16 round consist of:
  #) selecting 24 bit of each halves & permutating them by P(2) for use in 'f'.
  #) rotating each half either 1 or 2 places depending upon the key rotation schedule (KS).
  #) This can be described functionally as:

$$K(i) = P2(KS(PC1(K, i)))$$

48 bit i/p is divided into eight 4 bit words each of 6 bits which are fed into separate S-boxes. Each S-box produces a 4 bit word. Hence, the 8 S-boxes together generate 32 bit o/p word.

3rd July, 2019
Wednesday

(4)

Modes of block Cipher / stream Cipher

The way we use a block cipher is called mode of use & four have been defined for DES by ANSI standard.

#) Block modes
splits messages in blocks (ECB, CBC)

① Electronic Code Book
⇒ where the message (broken into independent 64 bits) blocks are encrypted.

$$C(i) = DES_{-}(K_i)\ (P_{-}(i))$$

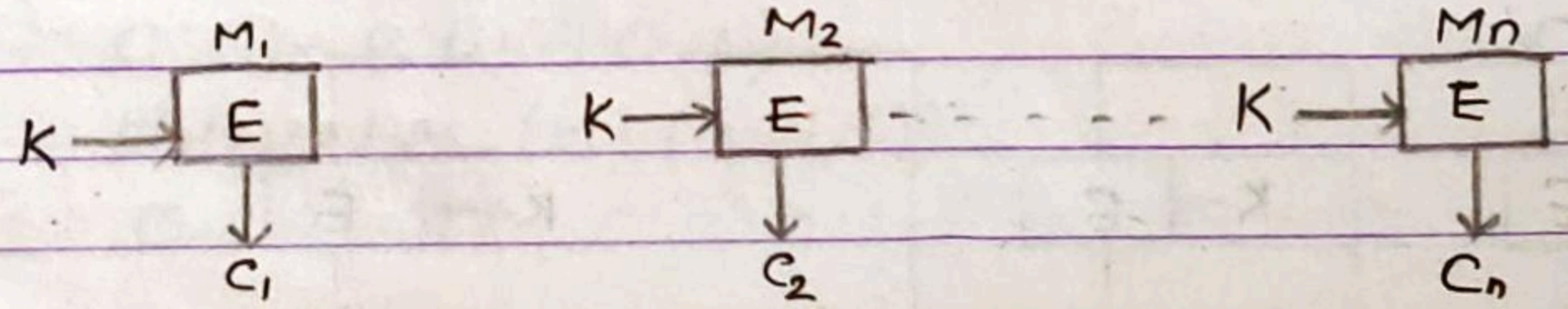


Fig: ECB

② Cipher Block Chaining
⇒ Again the message is broken into 64 bit blocks, but they are linked together in the encryption operation with an initial vector. (IV)

$$C(i) = DES_{-}(K_i)(P(i) \oplus C(i-1))$$

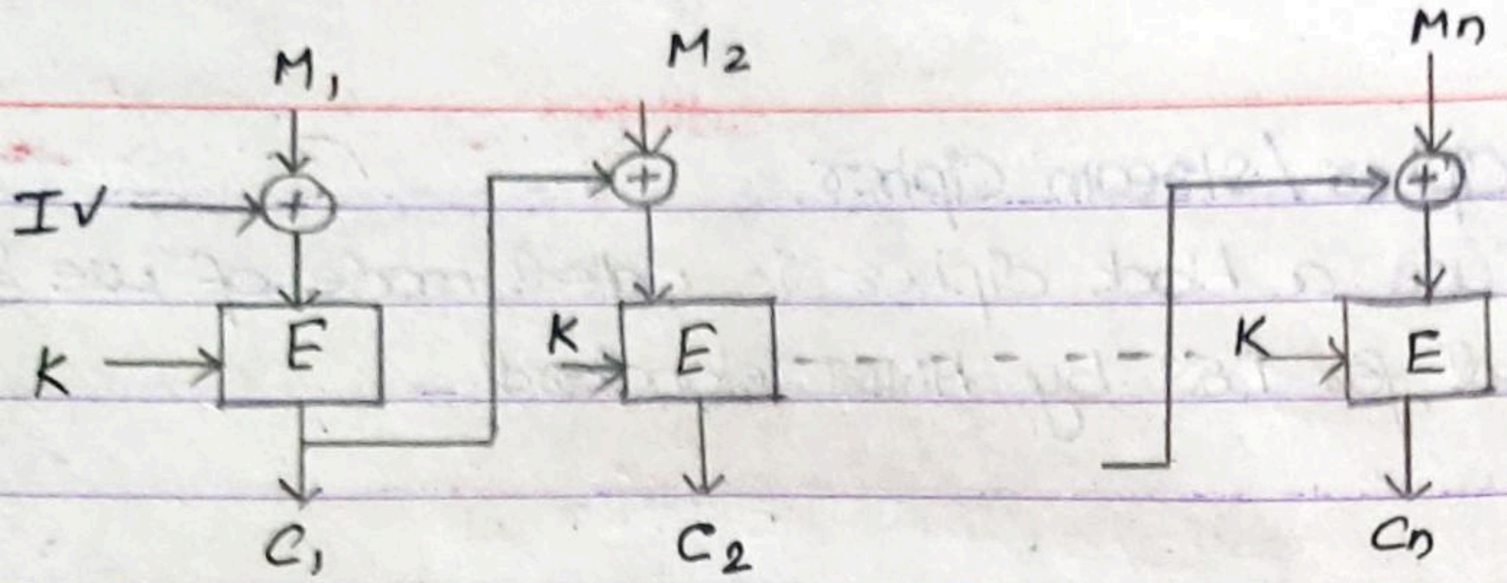


Fig: CBC

#) Stream Modes

This is one bit stream modes (CFB, OFB)

① Cipher Feedback Mode

The message is treated as a stream of bits, added to the o/p of DES, with the result being feedback for next stage.

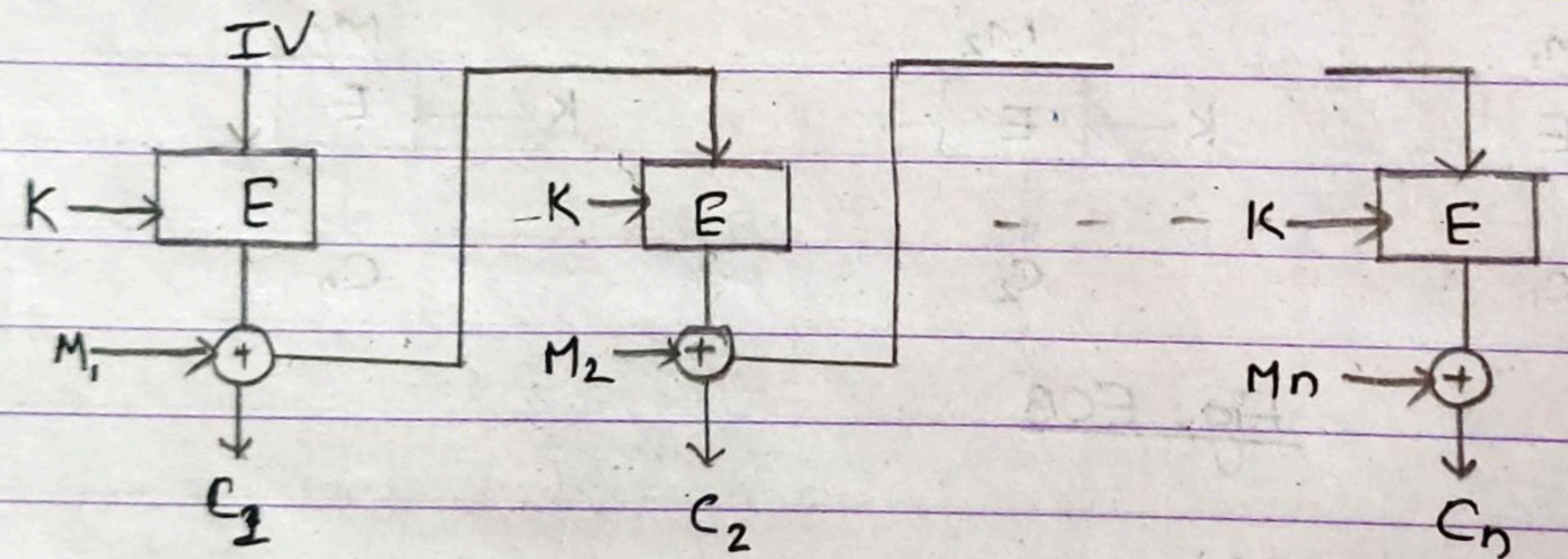


Fig: Cipher feedback mode

$$C_{(i)} = P_{(i)} \oplus DES_{(ki)}(C_{(i-1)})$$

$$C_{(-1)} = IV$$

(ii) Output Feedback Mode

The message is treated as a stream of bits, added to the message but with the feedback being independent of the message.

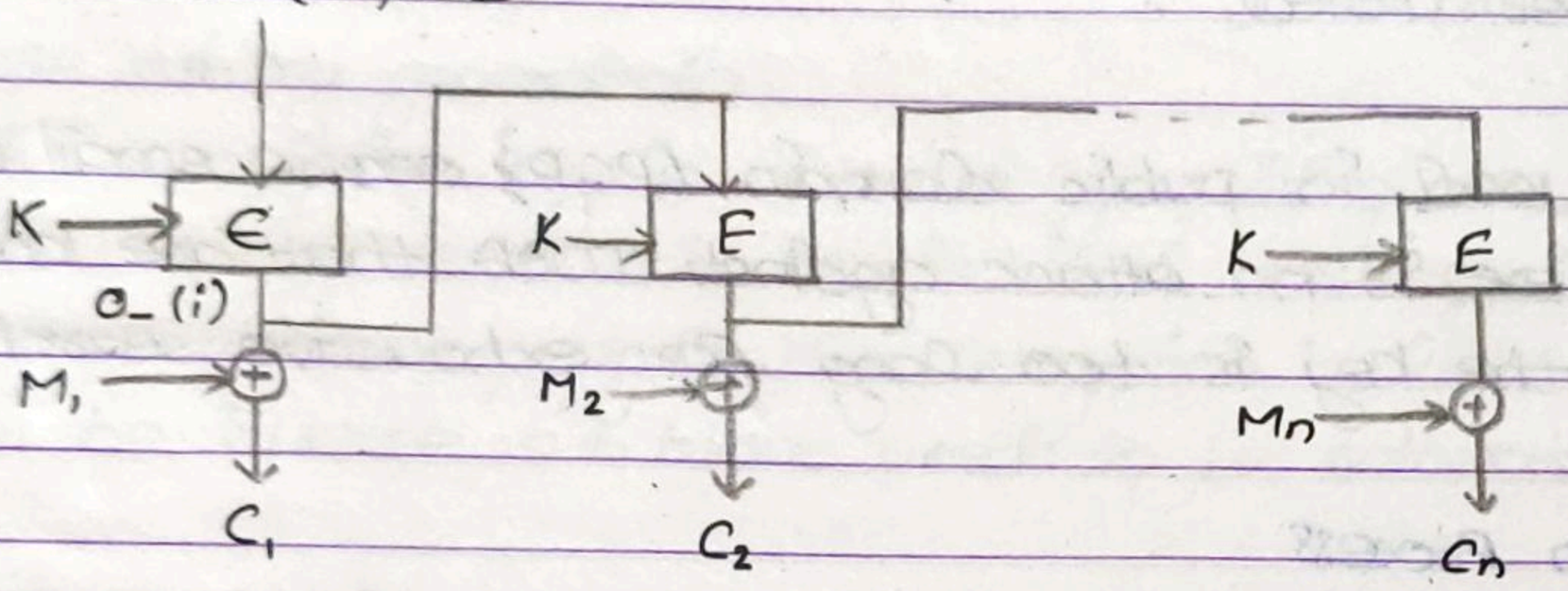


Fig: Output Feedback Mode

$C_{-}(i) = P_{-}(i) \oplus O_{-}(i)$

$O_{-}(i) = DES_{-}(K_i)\ (O_{-}(i-1))$

$O_{-}(-1) = IV$

Triple DES or 3DES can be used to make the DES algorithm more robust and secure against unauthorized and unauthenticated approaches.

#) International Data Encryption Algorithm (IDEA)

IDEA was developed by James Massey & Xuejia Lai in 1990 at ETA Zurich. It was originally called IPESS. In 1992, the name was changed to IDEA. IDEA encrypts 64 bit message box using 128 bits key, to obtain 64 bit cipher blocks. It is based on mixing operation from different algebraic groups (XOR, addition mode 2^{16}, multiplication mod $2^{16}+1$).

An operation is performed on 16 bit block with no permutation. Hence, it is very efficient in software implementation. IDEA is patented in Europe and US, however non-commercial use is freely permutated.

It is used in public domain (PGP) secure email system. Currently there is no attack against IDEA that are known to us as the key is too long for exhaustive search.

- Encryption Process

⇒ It encrypts 64 bit plain text into 64 bit cipher text block using a 128 bit key (i/p) say K. It is based on novel generalization of Feistel structure. It consists of 8 computational identical rounds followed by an o/p transformation. A round 'r' consist of 6 16 bit sub keys $K_i^{(r)}$ $(1 \le i \le 6)$. To transform the 64 bit input 'x' into an output of 4 16 bit blocks which are the input to the next round.

⇒ Round 8 output enters to the output transformation employing for additional sub keys $K_i^{(9)}$ for $1 \le i \le 4$ to produce the final ciphertext $y = \{y_1, y_2, y_3, y_4\}$. The dominant design concept of IDEA is mixing operation from three different algebraic groups of 2^n elements.

⇒ The corresponding group operations of sub-blocks a & b of bit length n=16 are

① bitwise XOR : $a \oplus b$

② addition mod $2^n = (a+b)$ AND 0XFFFF (16 bit binary operation)

denoted by ⊞ &

③ modified. multiplication: mod 2^n+1 with $0 \in Z_{2^n}$ associated with $Z_n \in Z_{2^n+1}$; denoted by $a \odot b$.

#) IDEA subkey generalization

The encryption key is obtained by splitting 128 key bit into 8 16 bit subkeys once this key is used the key is rotated 25 bits and broken up again. The decryption is little more complex since the inverse sub block needs to be calculated.

#) IDEA Algorithm

The input is 64 bit plain text $m = m_1, m_2, m_3, m_4$. The output is 64 bit ciphertext $y = y_1, y_2, y_3, y_4$. The key length is 128 bits.

~~The~~

① key schedule

compute 16 bit sub keys for each of the 8 rounds. $K_1^{(1)}$ $K_2^{(2)}$ ... $K_{16}^{(8}$

② Divide the 64 bit message block into 4 16 bit blocks

$m_1, m_2, \dots m_{64} \Rightarrow \underbrace{x_1}_{16\text{ bit}}\ \underbrace{x_2}_{16\text{ bit}}\ \underbrace{x_3}_{16\text{ bit}}\ \underbrace{x_4}_{16\text{ bit}}$

③ For round r from 1 to 8 do

ⓐ $x_1 \leftarrow x_1 \odot K_1^{(r)}$, $x_4 \leftarrow x_4 \odot K_4^{(r)}$

$x_2 \leftarrow x_2 \boxplus K_2^{(r)}$, $x_3 \leftarrow x_3 \boxplus K_3^{(r)}$

ⓑ $t_0 \leftarrow K_5^{(r)} . (x_1 \oplus x_3)$, $t_1 \leftarrow K_6^{(r)} (t_0 \boxplus (x_2 \odot x_4))$, $t_2 \leftarrow t_0 \boxplus t_1$

③ $x_1 \leftarrow x_1 \oplus t_1$, $x_4 \leftarrow x_4 \oplus t_2$, $a \leftarrow x_2 \oplus t_2$, $x_2 \leftarrow x_3 \oplus x$,

$x_3 \leftarrow a$

④ o/p transformation

$y_1 \leftarrow x_1 \odot k_1^{(9)}$, $y_4 \leftarrow x_1 \odot k_4^{(9)}$

$y_2 \leftarrow x_3 \boxplus k_2^{(9)}$, $y_3 \leftarrow x_2 \boxplus k_3^{(9)}$

⇒ The input to the output transformation is 128 bit key and the output is 52 bit

$128 - K_1, K_2 - - - - K_{128}$

52-bit subkeys

$K_1', K_2' - - - K_6'$

$K_1^2, K_2^2 - - - K_6^2$

⋮

$K_1^8 - - - - K_6^8$

Single round of IDEA

$\odot$ ⇒ Multiplicative mod 2^n+1

$\oplus$ ⇒ bitwise XOR

$\boxplus$ ⇒ Addition mod 2^n.

step ③ of above algorithm can also be represented as:

① $X_1 * K_1$

② $X_2 + K_2$

③ $X_3 + K_3$

④ $X_4 * K_4$

⑤ Step ① ⊕ Step 3

⑥ Step 2 ⊕ Step 4

⑦ Step 5 * K5

⑧ Step 6 + Step 7

⑨ Step 8 * K_6

⑩ Step 7 + Step 9

⑪ Step 1 ⊕ Step 9 → R_1

⑫ Step 3 ⊕ Step 9 → R_2

⑬ Step 2 ⊕ Step 10 → R_3

⑭ Step 4 ⊕ Step 10 → R_4

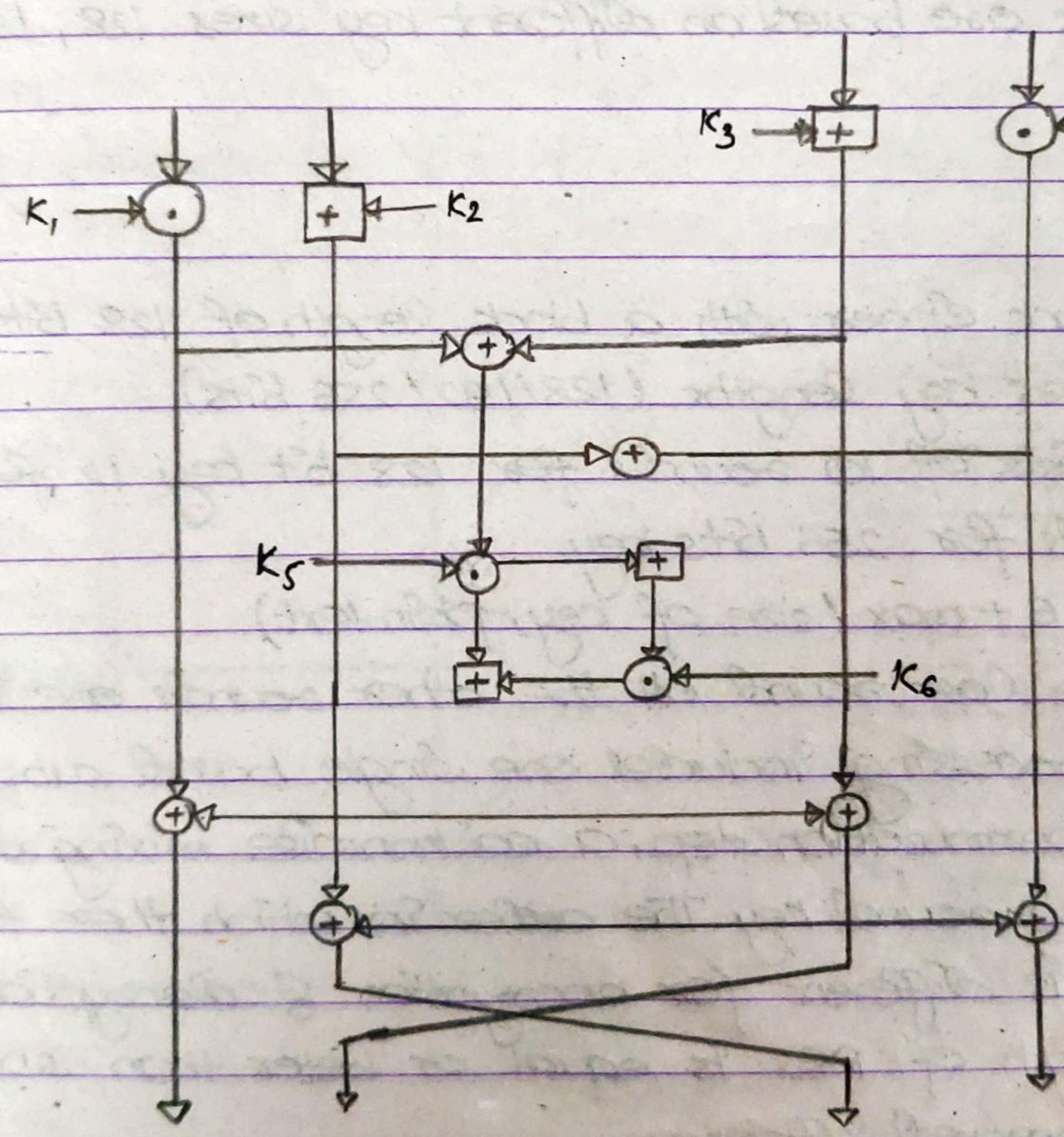


Fig: Single round of IDEA

#) Advanced Encryption Standard (AES)

⇒ Rijndael is a family of block ciphers developed by Belgian cryptographers Vincent Rijmen and Joen Daemen.

⇒ It was submitted as an entry to NIST competition to select as an Advanced Encryption Standard (AES) to replace DES.

⇒ In 2001, Rijndael won the competition and the 128, 192 & 256 bit versions of Rijndael were officially selected as Advanced Encryption Standard.

⇒ The 3 variants are based on different key sizes 128, 192 & 256 bits.

Features:

⇒ AES is a block cipher with a block length of 128 bits.

⇒ has 3 flavours key lengths (128/192/256 bits)

⇒ Encryption consists of 10 rounds for 128 bit key 12 for 192 bits key & 14 rounds for 256 bits key.

*No. of round = 6 + max (size of key, plain text)

⇒ Except for the last round all the other rounds are identical.

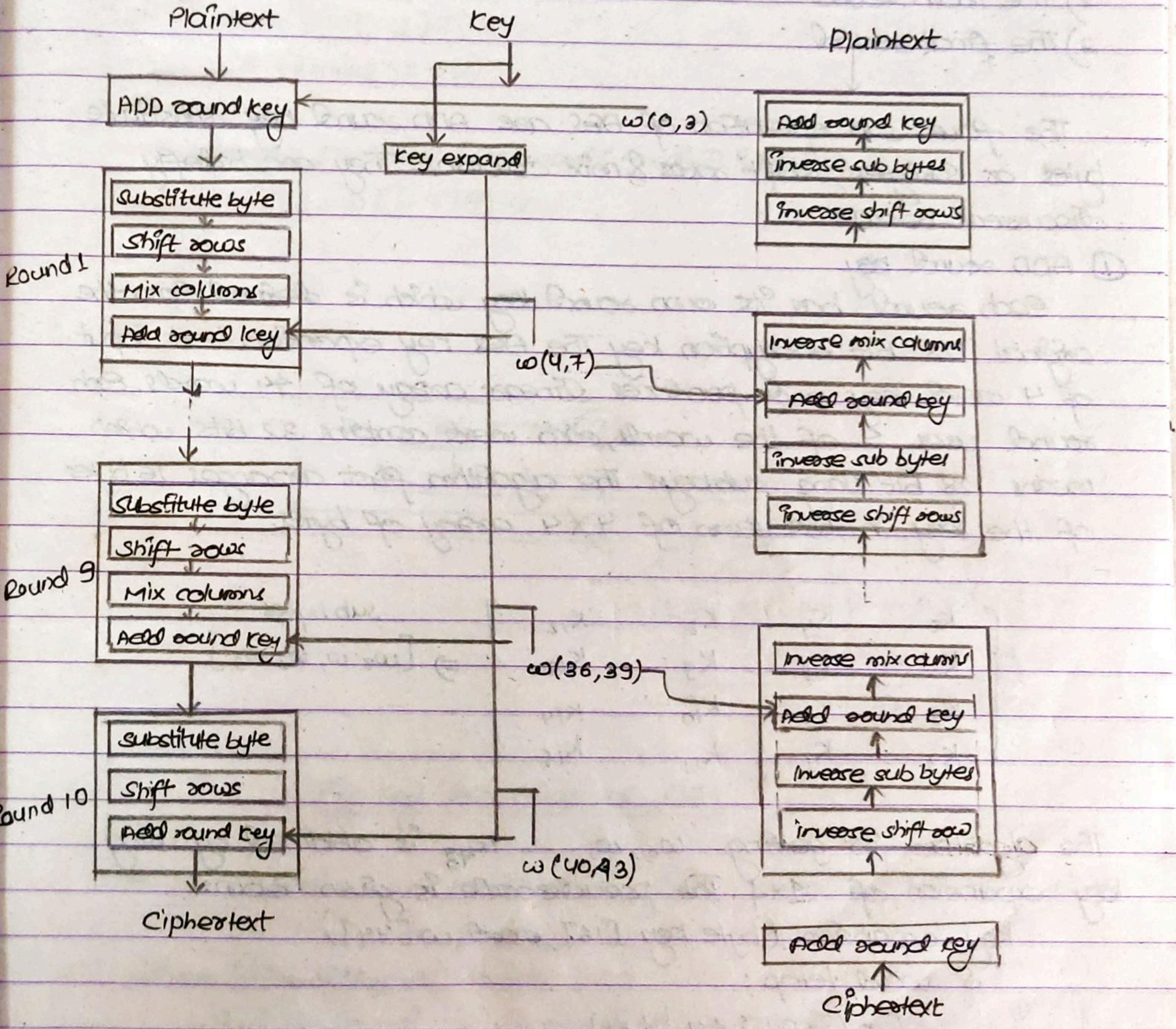
⇒ Each round processing includes one single based substitution step, a row-wise permutation step, a column wise mixing step and addition of the round key. The order in which these four steps are executed is different for encryption & decryption.

⇒ Security strength of AES is equal or better than 3DES and significantly improved efficiency

⇒ The 128 bit block is represented as a 4×4 matrix of bytes as shown below:

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

⇒ This 4×4 matrix of byte is referred to as state array.



The encryption phase of AES can be divided into 3 phases:

1) Initial round
2) The main round
3) The final round

The four sub operation of AES are ADD round key, substitute bytes or SubBytes, shift rows & mix columns. They are briefly discussed below:

① ADD round key

Each round has its own round key which is derived from the original 128 bit encryption key. The AES key expansion text input of 4 word key and produces linear array of 44 words. Each round uses 4 of the words, each word contains 32 bits which means 128 bit long subkeys. The algorithm first arranges 16 bytes of the key in the form of 4×4 array of bytes.

$$\begin{bmatrix} K_0 & K_4 & K_8 & K_{12} \\ K_1 & K_5 & K_9 & K_{13} \\ K_2 & K_6 & K_{10} & K_{14} \\ K_3 & K_7 & K_{11} & K_{15} \end{bmatrix} \xrightarrow{\text{sub bytes}} \Rightarrow [w_0\ w_1\ w_2\ w_3]$$

The algorithm for getting $w_0, w_1, \ldots w_{43}$ is obtained by using key expansion of 4×4. The pseudocode is given below:

```
key-expansion (byte key[16], word w[44])
{ word temp;
    for (i=0; i<4; i++)
    {
        w[i] = (k[4*i], k[(4*i)+1], k[(4*i)+2], k[(4*i),3]);
```

```
    }
for (i=4; i<=43; i++)
{
  temp = w[i-1];
  if (i mod 4 == 0)
{
  temp = subword (Rotword (temp)) ⊕ Rcon [i/4];
  w[i] = w [i-4] ⊕ temp;
  }
}
```

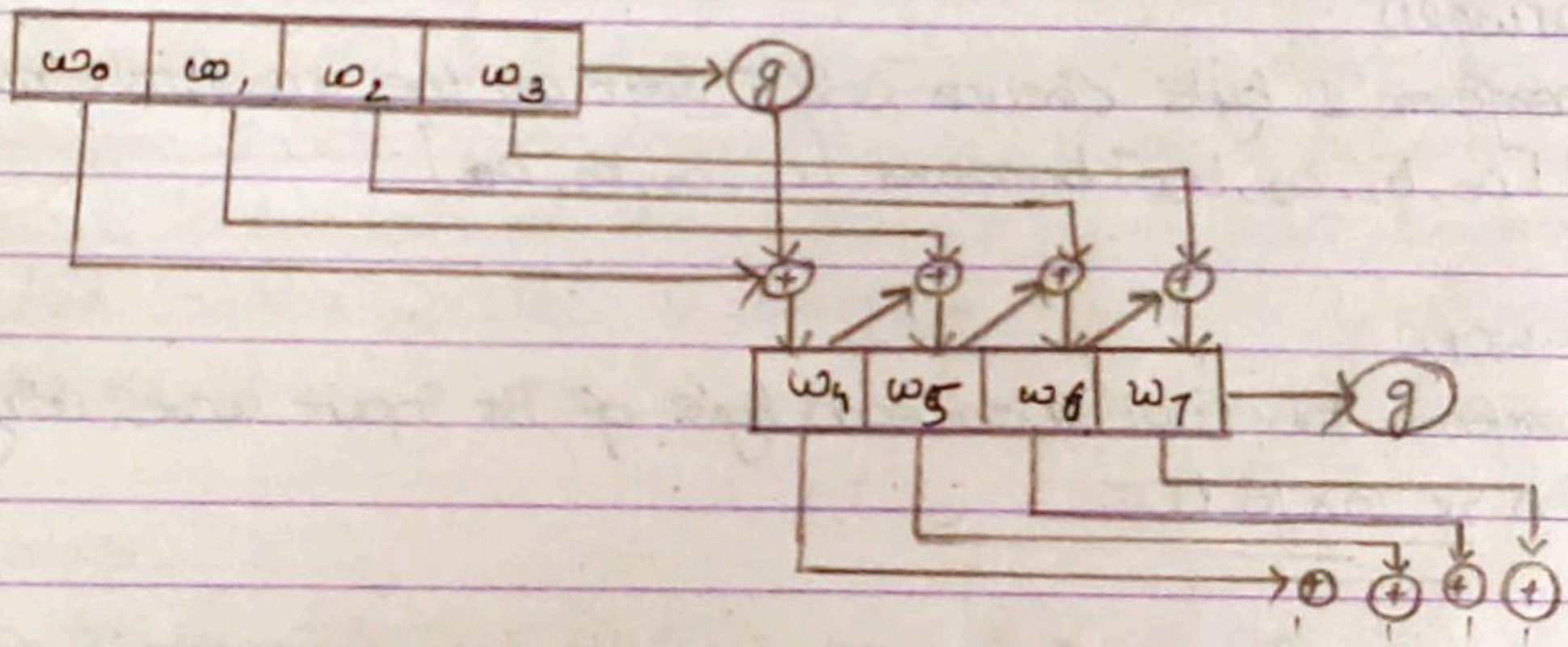


fig: key expansion of AES

Let us consider we have four words of round key of i^{th} round.

$w_i, w_{i+1}, w_{i+2}, w_{i+3}$

when ($i \bmod 4 \neq 0$) then

w[i] is found by immediately preceding word w[i-1] XOR-ed with four positive back word [i-4] i.e.

$w[i] = w[i-4] \oplus w[i-1]$

Gf = Galois Field.

$$w[i+5] = w[i+1] \oplus w[i+4] \quad -①$$
$$w[i+6] = w[i+2] \oplus w[i+5] \quad -②$$
$$w[i+7] = w[i+3] \oplus w[i+6] \quad -③$$

for a word those position in w array is multiple of 4, a more complex function is used.

⇒ The beginning of each round key is obtained by,

$$w_{i+4} = w_i \oplus g(w_{i+3}) \quad -④$$

where g consists of following substitution:

① ROTWORD

⇒ perform 1 byte circular shift left on the word. This means word $[b_0, b_1, b_2, b_3]$ becomes $[b_1, b_2, b_3, b_0]$

② SUBWORD

⇒ perform substitution on each byte of its input word using s-box which is 16×16 LUT.

③ The result of step ② is XOR-ed with round constant Rcon[j].

The round constant is a word in which the rightmost 3 bytes is always zero. Thus, it has only effect on the left byte of the word.

$$Rcon(j) = [RC[j], \overrightarrow{0x00^0}, \overrightarrow{0x00^0}, \overrightarrow{0x00^0}]$$

with $RC[j] = 1$; $RC[j] = 2 \cdot RC[j-1]$ and the multiplication is defined over the field of GF(2^8).

⇒ The addition of round constant is for the purpose of destroying any symmetry that may have been introduced in other steps in key expansion algorithm

5th July, 2019
Friday

(5)

Sub-Bytes

⇒ This round uses byte by byte substitution during the encryption forward process. The corresponding substitution for the decryption process is called Inverse sub bytes.

⇒ This step consists of a 16×16 lookup table (LUT) to find a replacement for a byte in the input state array. The entries in the LUT are created using the notion of multiplicative inverse of $GF(2^8)$ (upto 256 bits and represent in hexadecimal as 0-0f).

⇒ For a particular round, each byte is mapped to a new byte in the following way. The leftmost nibble from the left half determines the row and the rightmost nibble determines the column. The S-box transformation of 95 or 0X23 (hex) can be found in the cell at the intersection of row labeled 20 and the column labeled 03. Therefore, decimal 35 becomes 0X26 or decimal 38.

Shift rows

This step is called shift rows for shifting the rows of the state array during the forward process, the operation is called inverse shift rows during the transformation in decryption state. In this operation, the first row is unaltered, second and third row (remaining rows i.e. 2nd to fourth) are shifted to the ~~right~~ left in increasing order.

2nd ⇒ 1 shift ~~right~~ left

3rd ⇒ 2 shift left

4th ⇒ 3 shift left

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,0}$
$a_{2,2}$	$a_{2,3}$	$a_{2,0}$	$a_{2,1}$
$a_{3,3}$	$a_{3,0}$	$a_{3,1}$	$a_{3,2}$

Mix columns

The mix column operation is basically a substitution that uses GF(28) in which each column is operated individually. Each byte of the column is mapped into a new column by using a function of all four bytes in the column.

The transformation is visually shown in the figure below:

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

X

2	3	1	1
1	2	3	1
1	1	2	3
3	1	1	2

=

$b_{0,0}$	$b_{0,1}$	$b_{0,2}$	b_{0}
$b_{1,0}$	$b_{1,1}$	$b_{1,2}$	b_{1}
$b_{2,0}$	$b_{2,1}$	$b_{2,2}$	b_{2}
$b_{3,0}$	$b_{3,1}$	$b_{3,2}$	b_{3}

Fig: Inverse MIXcolumns

Advantages of AES

⇒ One of the primary advantage of AES is its ubiquity
⇒ standard used by the US government.
⇒ is relatively fast in both hardware and software.
⇒ allows users to pick a tradeoff between speed and security.
⇒ Increased key length increases the execution time of both encryption and decryption.
⇒ uses a single S-box for all bytes in all rounds.

Disadvantage

⇒ AES has very simple key schedule & simple encryption operations. Many AES attacks are based upon the simplicity of this key schedule & it is possible that one day an attack will be created to break AES encryption.

Characteristics	DES	IDEA	AES
Basic	Data block divided into 2 halves.	Same as AES	Entire data block is processed as a single matrix.
Principle	Feistel cipher	works on substitution & permutation.	works on substitution & permutation.
Plaintext	64 bits	64 bits	128, ~~192, 256~~ bits
Key size	56 bits	128 bits	128 bits 192, 256 bits
Round	16 rounds	8 rounds	10-128 bits 12-192 bits 14-256 bits
Speed	slower	faster than DES / slower than AES	faster
Security	less secure	more secure	more secure than IDEA

Characteristics	DES	IDEA	AES
Round Name	Expansion, permutation, X-OR, P-Box, S-box, Swap	GF(2^{16}) GF($2^{16}+1$) X-OR	sub Bytes Shift Rows Mix Columns Add Round keys