XOR :
0 1 → 1
0 0 → 0

* Stream Cipher :

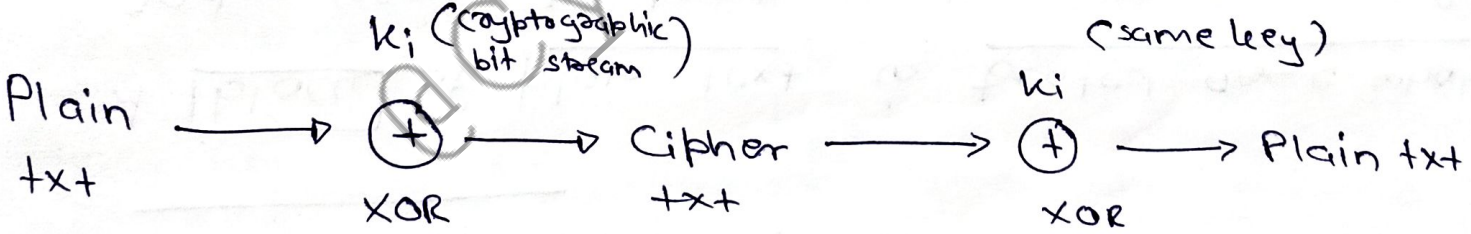
Used to convert Plain text → Cipher text

(A) Stream cipher encrypts a digital data stream

↓

One bit or one byte } at a time

→ A symmetric key cipher
i.e = 1 key for encryption & decryption both.



eg. Message : 1 0 1 1 0 1 1 1
At sender side

key : 0 1 0 0 1 0 1 1 (+)

Cipher txt : 1 1 1 1 1 0 0
(recieved end)

To decrypt:

```
  11 11 11 00   <-- cipher text
⊕ 01 00 10 11   <-- key
  ---------
  10 11 01 11   <-- decrypted msg (plain text)
```

* Block cipher:

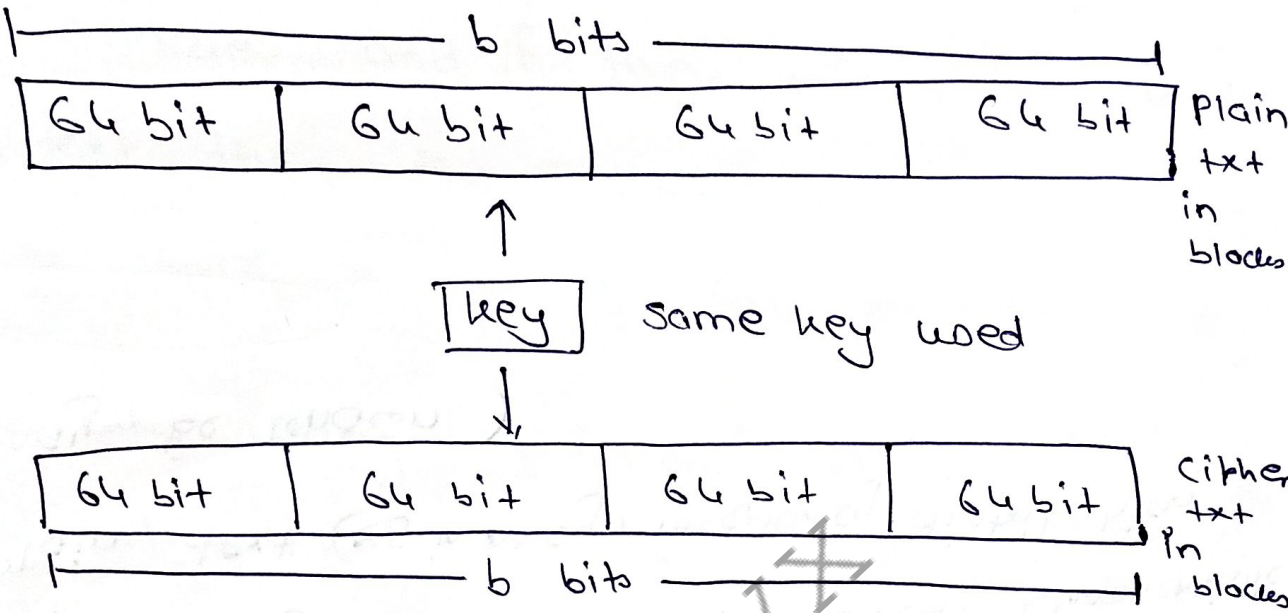
A **block** of plain text is treated as a whole and used to produce the cipher text of equal length.

Typically a block of 64 and 128 bits is used.

→ Symmetric key cipher
(1 key only)

key will be applied on each block.

eg.



BLOCK CIPHER	STREAM CIPHER
(1) Plain → Cipher via block method.	(1) 1 bit or byte of plain txt → Cipher txt
(2) Uses 64 bits or more.	(2) Uses 8 bits.
(3) Complexity = simple	(3) Complexity = difficult
(4) Uses confusion & defusion	(4) Uses only confusion
(5) Reverse encryption = hard	(5) Reverse encryption = easy
(6) Algorithmic modes :	(6) Algorithmic modes :
→ ECB (Electronic Code Block)	→ CFB (Cipher feedback)
→ CBC (Cipher Block chaining)	→ OFB (Output feedback)

✡ Confusion & Diffusion:

introduced by Claud Shannon

↳ AkA "Shannon Theory."

Confusion: hiding the relⁿ betn

↳ cipher txt & plain txt

Diffusion: hiding the relⁿ betn

↳ plain txt & cipher txt

Shannon Theory says that, if a attacker has some knowledge of the statistical characteristi of Plain text (eg. in msg, frequency distribution may be known)

so if these statistics are known in any way to an attacker and if they reflect back in the cipher text then the cryptoanalyst (i.e attacker) may be able to deduce the encryption key.

That is why using confusion & diffusion we hide the relationship.

In diffusion: change in 1 bit of plain text
↓
change in half or more bit of cipher.

In confusion: each bit of cipher txt should depend on key.

Fiestel Cipher Structure:

S-1 Plain text is divided into (2 parts) i.e L_0 & R_0

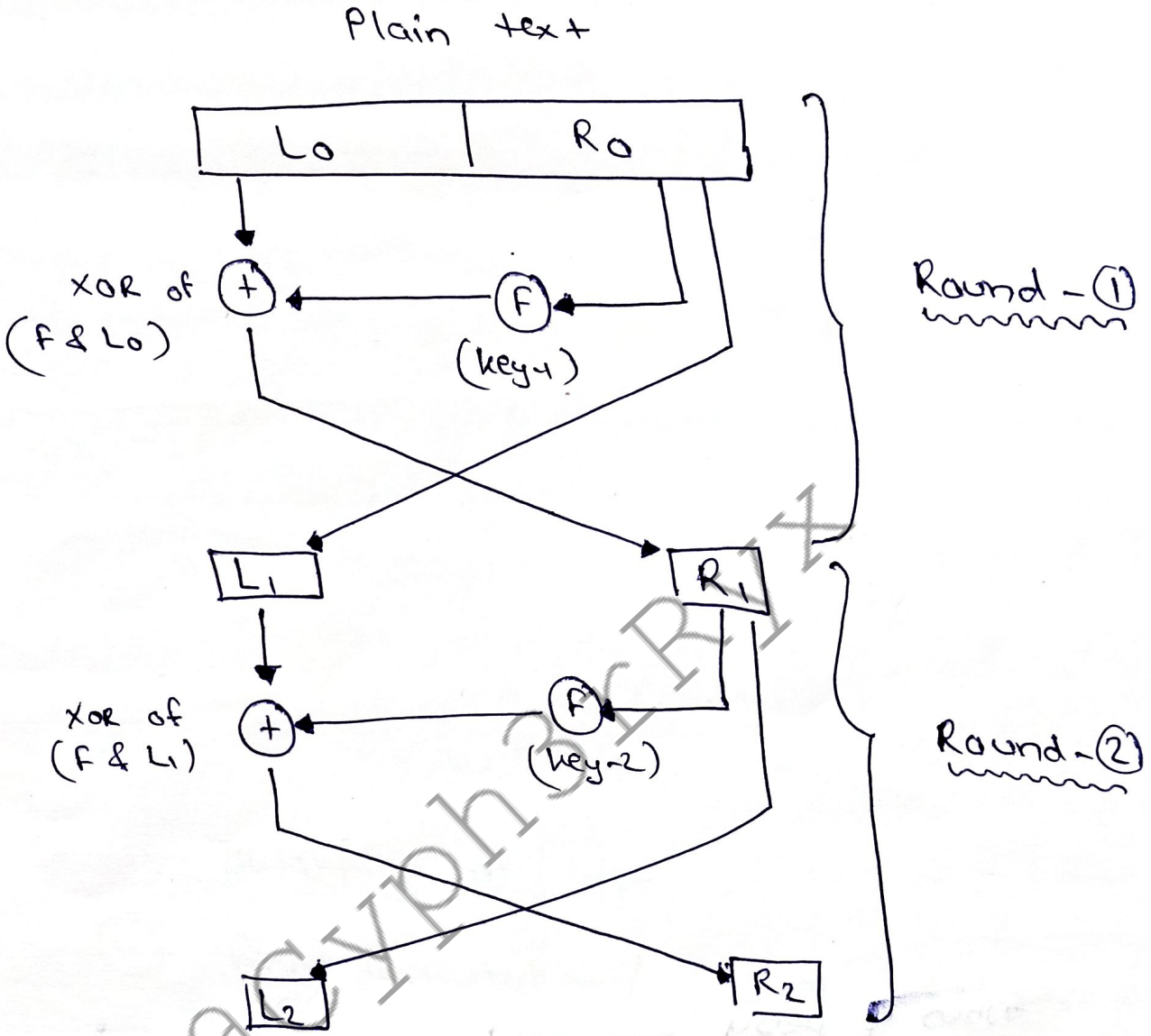
S-2 Both halves of the data passes through 'N' rounds of (processing) and then (combine) to produce the cipher text block.

S-3 On the right half, we apply a fxn & in that fxn we will use a (subkey) generated from master key.

S-4 The o/p of (S-3) is XORed (with) the left half of the left side and then their o/p will be swapped.

This S-1 to S-4 is for one single round.

Diagram:



Explanation:

(1) Plain txt in Lo & Ro

(2) Ro is processed via applying fxn with a key-1

(3) XOR opn takes place betn fxn o/p and Lo.

(4) XOR o/p is taken as R_1 for the next round and the original Ro is taken as L_1 for next round.

(1) Block size : larger size → more security

(2) key size : larger size → more security & less speed of encryption & decryption.

(3) No. of rounds : more rounds → more security

(4) subkey gen. algo : more complex → hard to crack

(5) fxn f : more complex → hard to crack.

Annexure No :

DES (Data Encryption Standards)

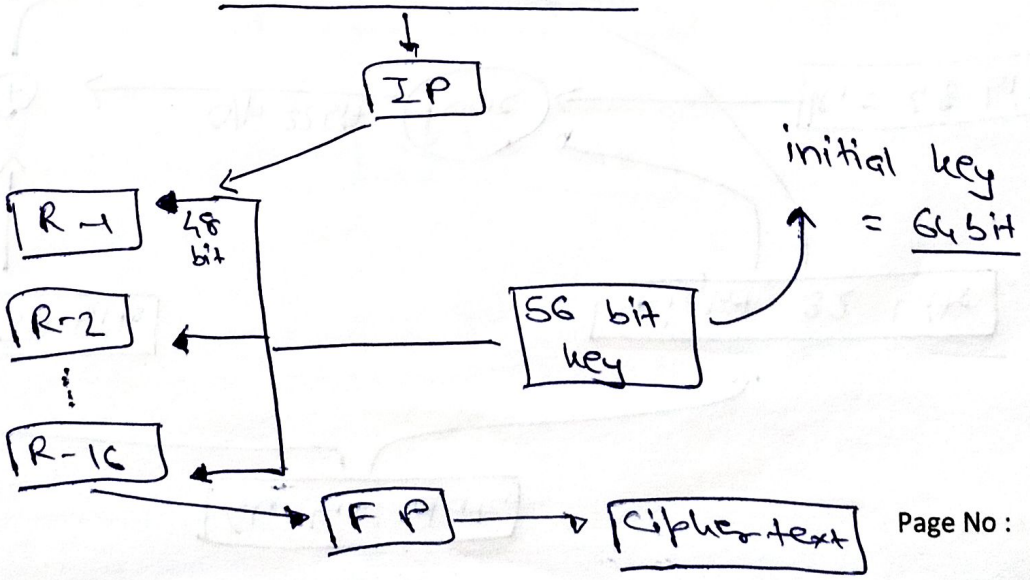
Notes:

1. Block cipher
2. Symmetric cipher
3. 64 bit plain text block
4. 16 rounds of processing & each round is a fiestel round.

Steps:

1. Initial Permutation
2. 16 fiestel rounds
3. Swapping of o/ps
4. final Permutation / Inverse initial permutation.

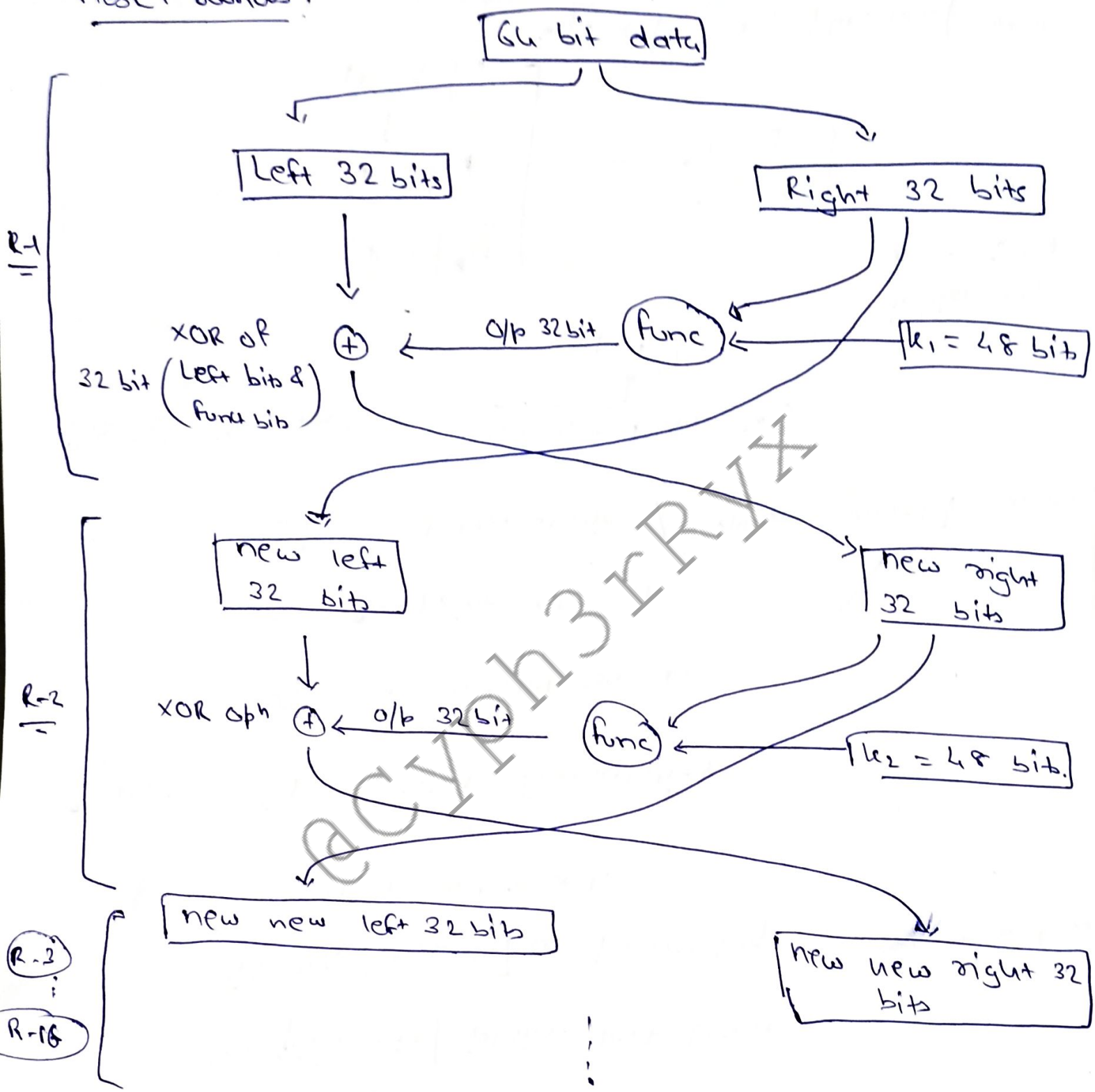
Basic Structure:



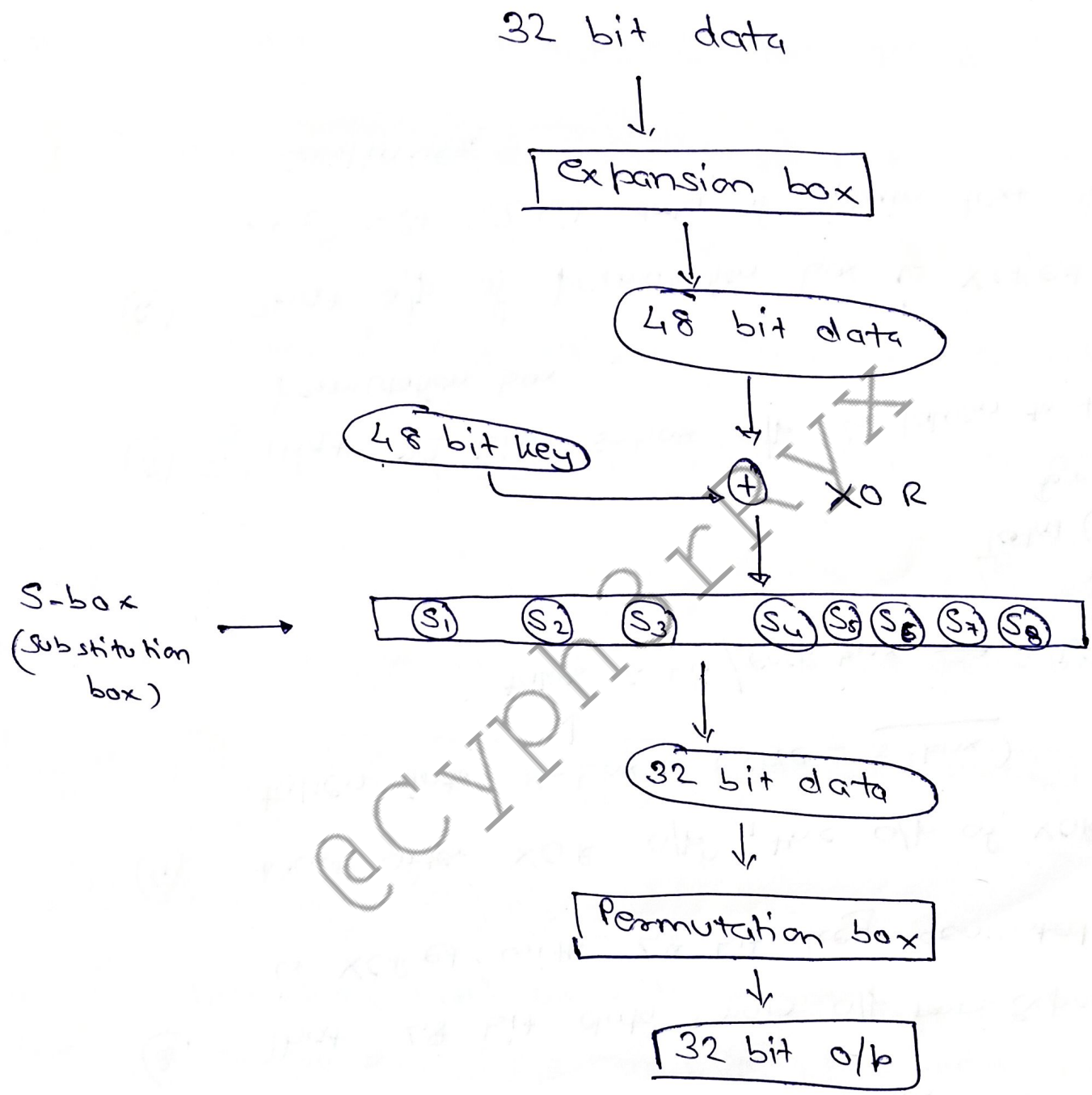
Enrollment No :

Page No :

Fiestel rounds.



* What happens in "function" ? OR function definition :



Explanation:

(1) Takes a 32 bit of data as input.

(2) Input is processed in Expansion box and gives a 48 bit data stream

(3) That 48 bit data stream o/p from Expansion box is XORed with 48 bit key generated.

(4) Now after XOR o/p, the o/p of XOR is taken into S-boxes (size = 8 box)

↓

takes 6 bit / each box → 4 bit / each box

↓

Total 32 bit for 8 box

(5) That 32 bit S-box o/p is taken to the permutation box

(6) That o/p of permutation box is XORed with the left 32 bit data of plain text & round continues.

But the question remains, what happens in expansion box though?

⇒ 32 bit of data is in 1's & 0's form

for clear explanation let's take a text example

String: DONT GIVE THE MONEY TO THAT PERSON

Now each letter of the string is divided into 4 bit block

D	O	N	T		G	I	V	E		T	H	E	m		O	N	E	Y		T	O	T	H		A	T	P	E

R	S	O	N

Now each block is converted from 4 bit to 6 bit

	DONT			GIVE			THEm			ONEY			TOTH	

	ATPE			RSON	

Now for left bit (first) we will consider the last letter of the previous block i.e from | D | O | N | T | we'll take 'T' as the first bit for | G | I | V | E | & same goes for Right bit (last) we will consider the first letter of the next bit. | G | I | V | E | I | from | T | H | E | m |

↳ final o/p | T | G | I | V | E | T |

N	DONT	G		T	GIVE	T		E	THEM	O		M	ONEY	T
Y	TOTH	A		H	ATPE	R		E	RSON	D				

for DONT & RSON we will consider it as globe structure so for 1st bit of DONT we'll consider last letter of RSON and for last bit of RSON well consider first letter of DONT ✓ easy

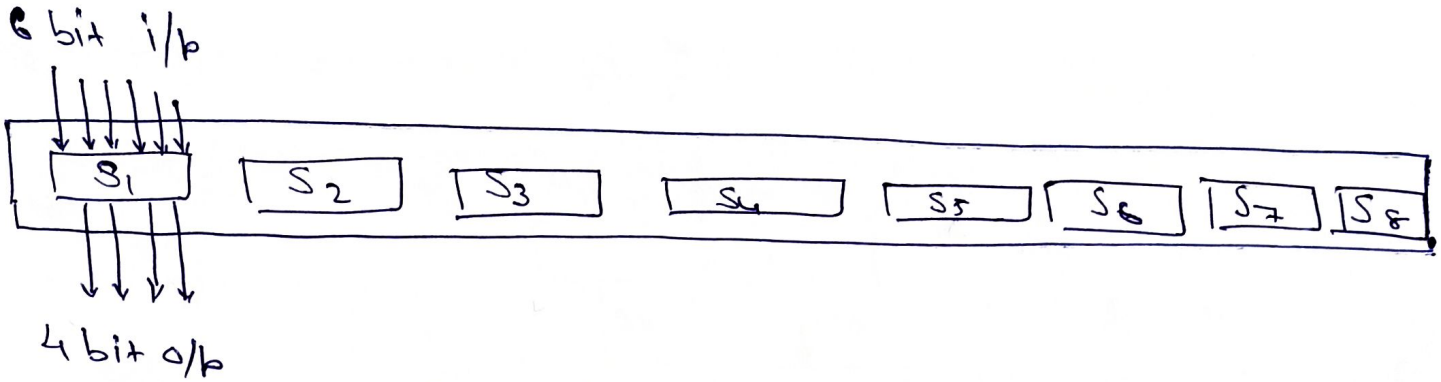
So here every 4 bit block is converted into 6 bit block

There were 8 block of 4 bits = 32 bit

Now there are 8 block of 6 bit = 48 bit

Now these 48 bit XOR with 48 bit key and then sent to S-box.

What happens in S-boxes ?



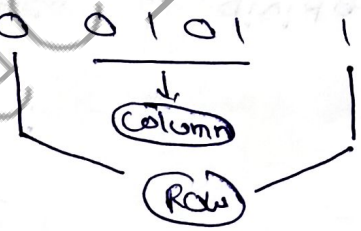
So $4 \times 8 = 32$ bit ... this will go in (permutation box)

How are 6 bit converted into 4 bit?

Eg. "0 0 1 0 1 1" is the 6 bit data

So first and last number are taken i.e: 0 1

Here



01 in decimal is (1)

0101 in decimal is (5)

Now we have to see S-box table in which we need to check Row - (1) & column - (5) and we get the number in decimal format (i.e: 7); we convert it into binary format [0 1 1 1] ⟶ this is the 4 bit o/p of 001011 input.

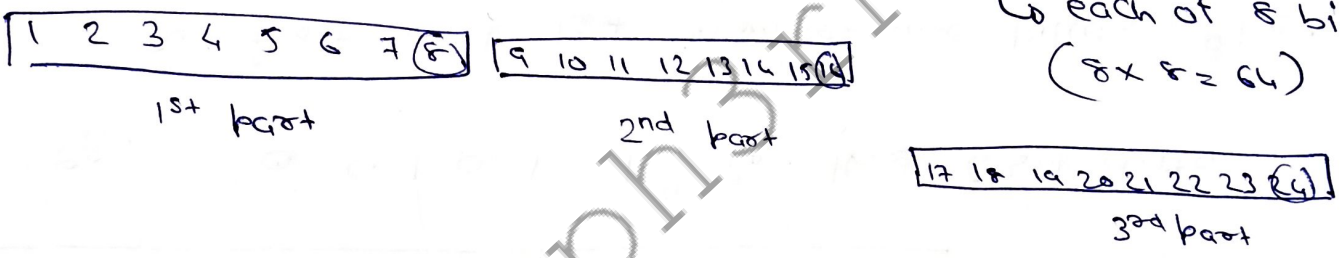
Now we have another question..,

How are 16 subkeys are generated in each round?

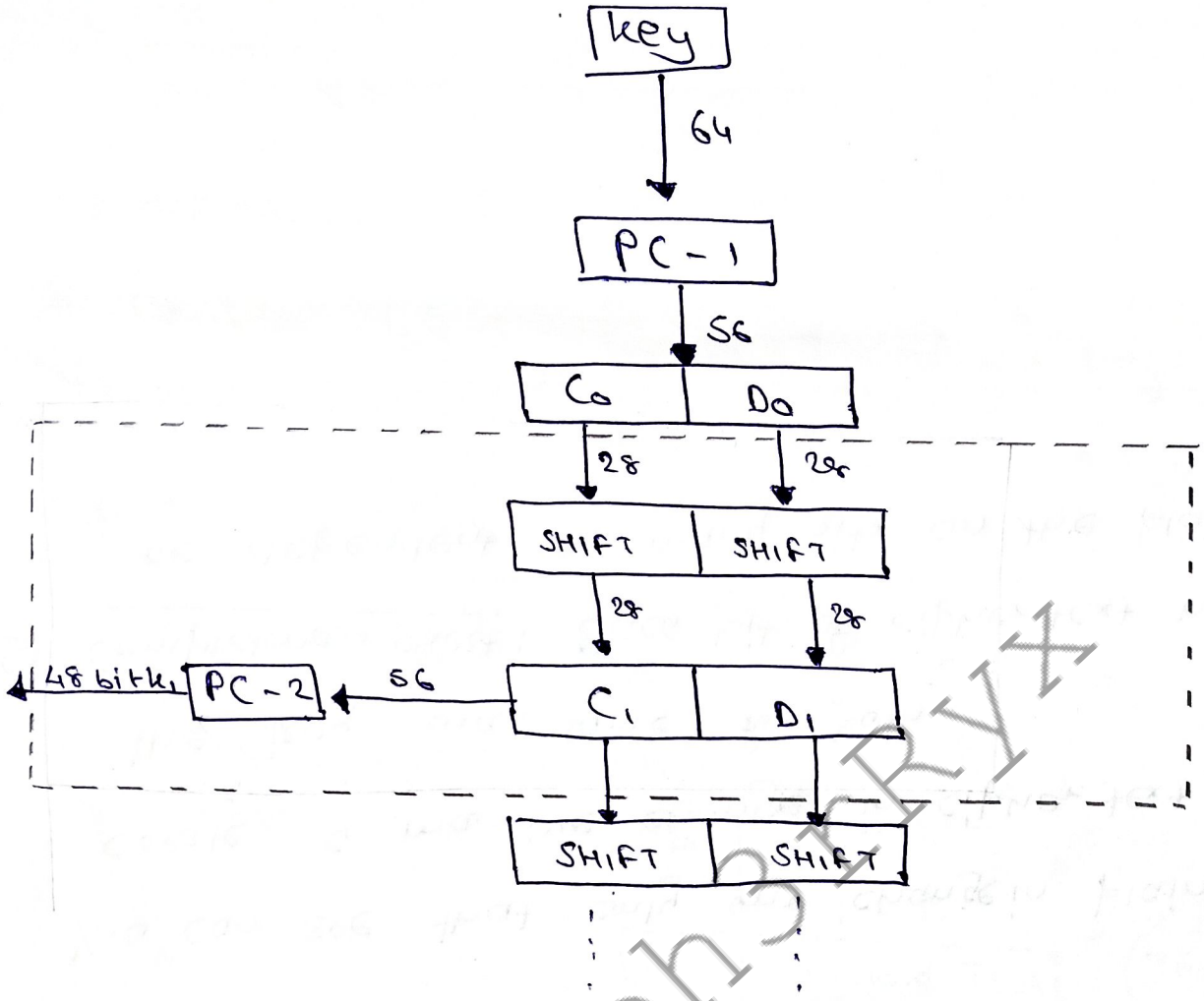
So the main key is actually of 64 bits which goes as input to PC-1 (Permuted Choice-1) and we get output as 56 bit key.

Inside PC-1 :

Sol 64 bit key is divided into 8 parts
↳ each of 8 bits
($8 \times 8 = 64$)



Discard last bit from every part so that we have 8 bit discarded in general [∴ $64 - 8 = 56$ bits]



PC-1 is a 56 bits which is then divided into C_0 & D_0 of 28 bit each then,

these bits are shifted with left shift in each round

→ for rounds 1, 2, 9, 16 we shift only (1) bits

→ for rounds remaining than 1,2,9,16 we shift (2) bits

After shifting we get (C_1, D_1) which goes in PC-2 as 56 bits.
28, 28

and in PC-2 last bits will be discarded and we get 48 bits key.

DES analysis

(1) Avalanche effect:

It says that a small effect / change in plain text or key should create a significant change in the cipher text.

eg. k = ABC1234

Plain - 0000000
Cipher - 1AD74C5
} 1st set

Plain - 0000001
Cipher - 0497C50
} 2nd set

You can see that only one change in plain text can create a massive change in cipher text even though the key was same for both.

(2) Completeness effect: Each bit of cipher text needs to be dependent on many bits on the plain text.

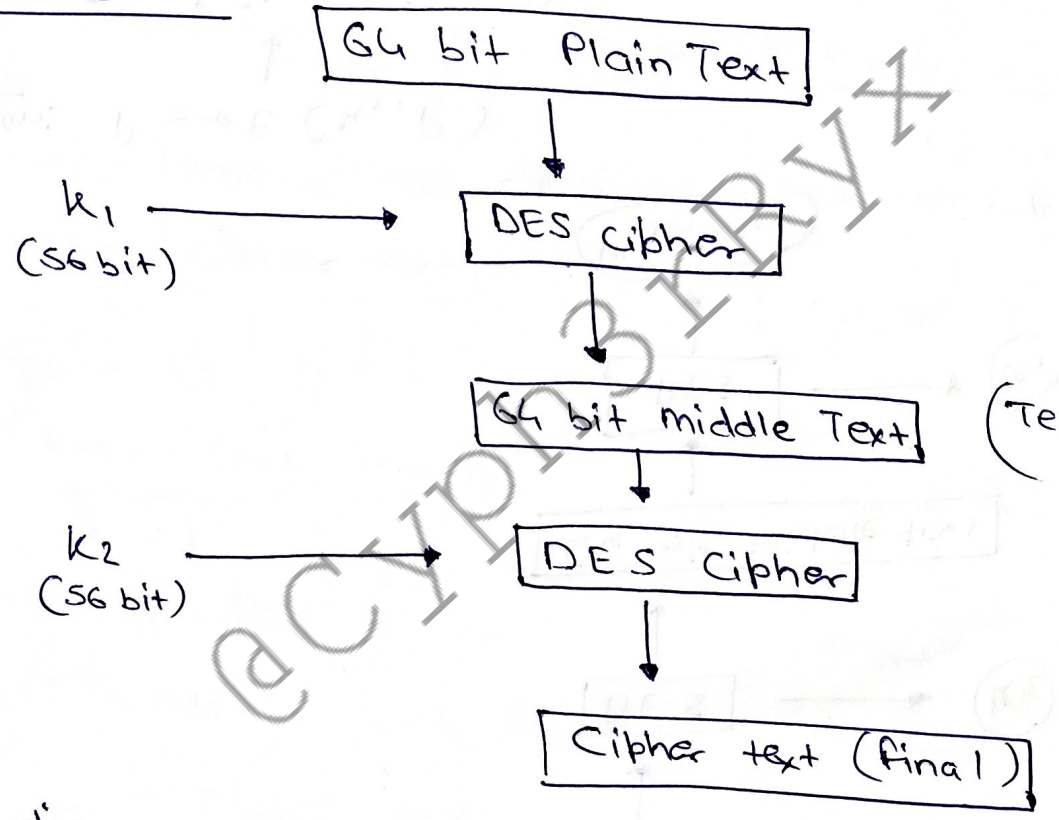
* Double DES :

DES alone is vulnerable so we use "multiple DES

→ Double DES uses 2 different keys

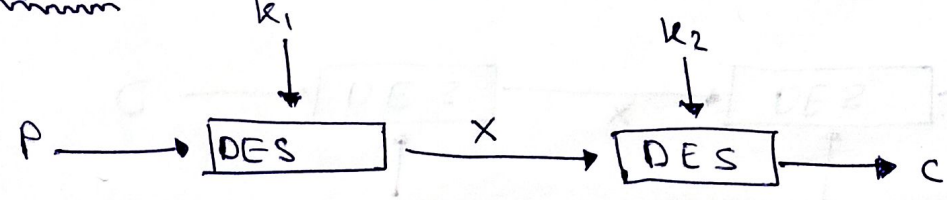
(56 + 56) = 112 bit keys

Block Diagram:

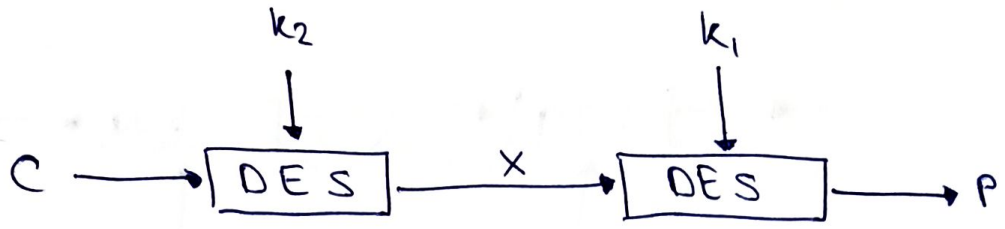


Simple diagram:

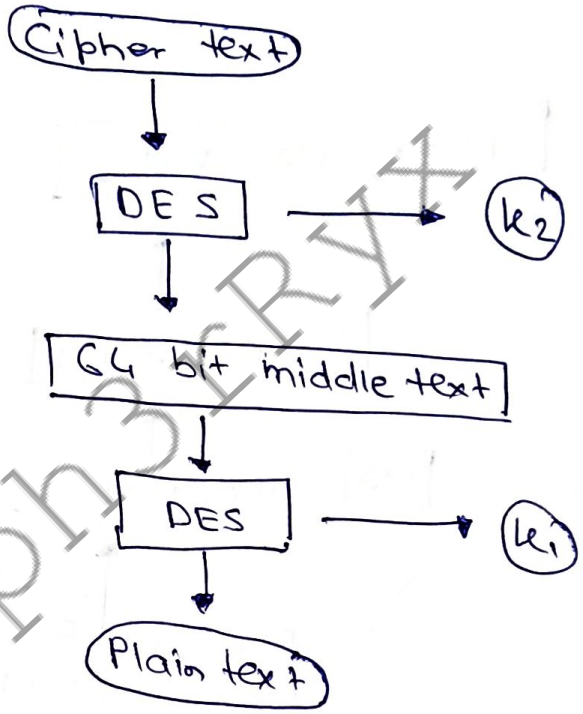
Encryption :



In decryption of 2DES: keys are swapped.



Block diagram:



Encryption: $P \rightarrow E(k_1, P)$

$C \rightarrow E(k_2, E(k_1, P))$ = Cipher

Decryption: $P \rightarrow D(k_1, D(k_2, C))$ = Plain

1st this will happen

Meet in the middle attack:

→ We have a problem with a temporary middle text (64 bit)

→ This attack is a drawback of 2DES.

Definition: This attack involves encryption from 1 end and decryption from the other end and then "matching the results in the middle".

Note: This attack does requires some knowing of Plain text / Cipher text.

Some pair of

Plain Text known

↓

encrypts pair for all 2^{56} possible values of k_1

Plain	Cipher	middle.
---	----	
---	----	

TABLE - (1)

sort the results in table (1)

Cipher text known

↓

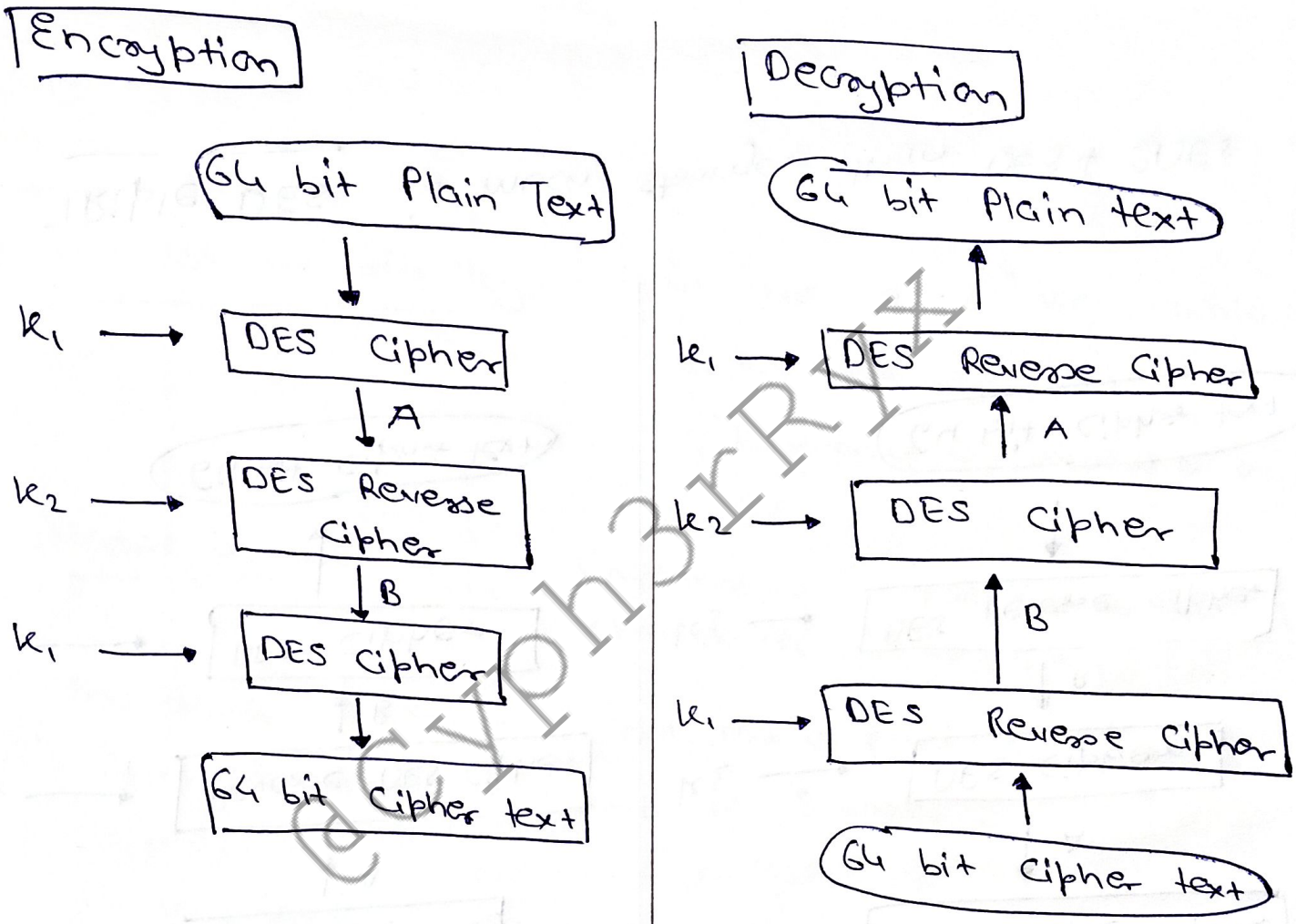
decrypts pair of all 2^{56} possible values of k_2

Cipher	middle	Cipher
----	----	
----	----	

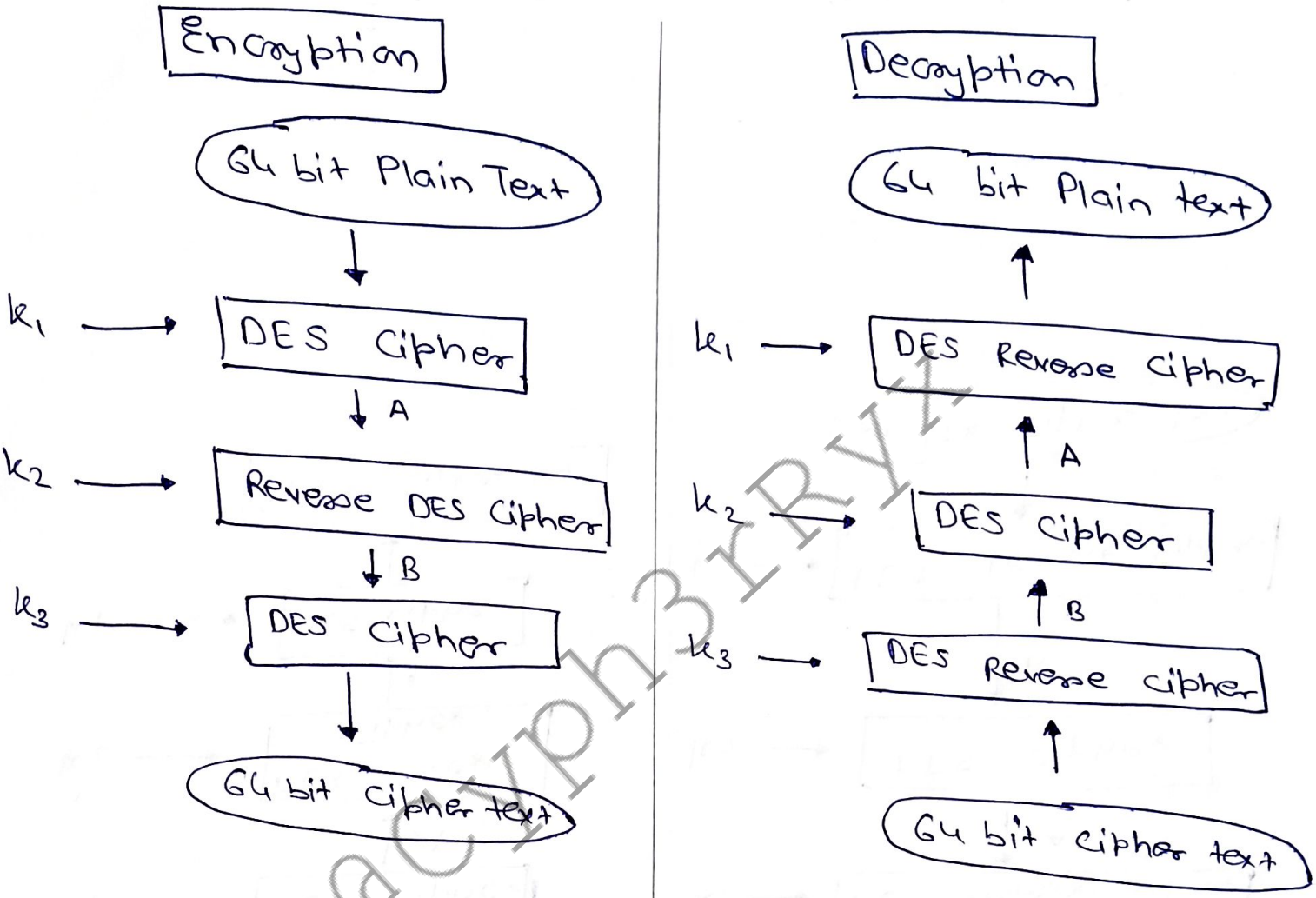
TABLE (2)

* Triple DES (3DES):

(1) Using 2 keys:



② Using 3 keys:



→ Triple DES is much stronger than DES & 2DES

Weakness of DES:

1. Weakness in S-boxes
2. Weakness in P-boxes
3. Weakness in key

* Strength of DES:

1. **Use of 56 bit keys:** Possible 2^{56} keys which is 7.2×10^{16} keys
   ∴ Bruteforce is impractical on it.

2. **Nature of DES algo:** Cryptanalysis is possible by exploiting characteristic of DES algo.
   The design of S-box is not made public & so far no fatal weakness is found in S-boxes.

3. **Timing Attacks:** Timing attacks exploits the fact that an encryption / decryption algo often takes slightly different amount of time on different inputs.
   DES is fairly resistant to a successful timing attacks.

* Block cipher design principles:

① **DES design criteria:** focus on design of S-box & P-box function.

② **No. of rounds:** ↑ no. of rounds ↑ is to perform cryptanalysis

③ **Design of function F:** Algorithm should have a great Avalanche effect.

④ **Key Schedule Algo:** select the subkeys to maximize the difficulty of deducing individual subkeys

Caeser cipher: let say $k = 3$ shift = (3)

A	B	C	D	E	F
(x) (z)	(y)	(z)	(a)	(b)	(c)
G	H	I	J	K	L
(d)	(e)	(f)	(g)	(h)	(i)
M	N	O	P	Q	R
(j)	(k)	(l)	(m)	(n)	(o)
S	T	U	V	W	X
(p)	(q)	(r)	(s)	(t)	(u)
Y	Z				
(v)	(w)				

x, y, z are forwarded in A, B, C

Plain: I LOVE RYX

Cipher: f ilsb ovu

Playfair:

① key: MONARCHY

PT: BALLOON

M	O	N	A	R
C	H	Y	B	D
E	F	G	I/J	K
L	P	Q	S	T
U	V	W	X	Z

column = ↓

row = →

rectangle = ← →

BA LL OO N

↓ ↓ ↓ ↓

BA LX LO ON

BA LX LO ON

↓ ↓ ↓ ↓

IB SU PM NA

Cipher txt = IBSUPMNA

② key: keyword

PT: COME TO THE WINDOW

K	E	Y	W	O
R	D	A	B	C
F	G	H	I/J	L
M	N	P	Q	S
T	U	V	X	Z

COME TO THE W

LC NK ZK VF ↓ YO

IN DO WX

HP CE BW

CT: LC NK ZK VF YO HP CE BW

Hill cipher:

A → 0	F → 5	K → 10	P → 15	U → 20
B → 1	G → 6	L → 11	Q → 16	Y → 21
C → 2	H → 7	M → 12	R → 17	W → 22
D → 3	I → 8	N → 13	S → 18	X → 23
E → 4	J → 9	O → 14	T → 19	Y → 24
				Z → 25

Encryption:

$$K = \begin{bmatrix} H & I \\ L & L \end{bmatrix} = \begin{bmatrix} 7 & 8 \\ 11 & 11 \end{bmatrix}$$

PT: EXAm $\Rightarrow$ $P_1 = \begin{bmatrix} E \\ X \end{bmatrix} = \begin{bmatrix} 4 \\ 23 \end{bmatrix}$

$C_1 = K \times P_1 \bmod 26$

$C_2 = K \times P_2 \bmod 26$

$P_2 = \begin{bmatrix} A \\ m \end{bmatrix} = \begin{bmatrix} 0 \\ 12 \end{bmatrix}$

$$C_1 = \begin{bmatrix} 7 & 8 \\ 11 & 11 \end{bmatrix} \times \begin{bmatrix} 4 \\ 23 \end{bmatrix} \bmod 26$$

$$= \begin{bmatrix} (7\times4) + (8\times23) \\ (11\times4) + (11\times23) \end{bmatrix} \bmod 26$$

$$= \begin{bmatrix} 212 \\ 297 \end{bmatrix} \bmod 26$$

$$= \begin{bmatrix} 4 \\ 11 \end{bmatrix} = \begin{bmatrix} E \\ L \end{bmatrix}$$

How to find mod:

$297 / 26 = 11.42$

≈ 11

$11 \times 26 = 286$

$297 - 286 = (11)$

$$C_2 = \begin{bmatrix} 7 & 8 \\ 11 & 11 \end{bmatrix} \times \begin{bmatrix} 0 \\ 12 \end{bmatrix} \bmod 26$$

$$= \begin{bmatrix} (7\times0) + (8\times12) \\ (11\times0) + (11\times12) \end{bmatrix} \bmod 26 = \begin{bmatrix} 96 \\ 132 \end{bmatrix} \bmod 26$$

$$= \begin{bmatrix} 18 \\ 2 \end{bmatrix} = \begin{bmatrix} S \\ C \end{bmatrix}$$

Cipher text = ELSC

Decryption:

C = ELSC

k = HILL $= \begin{bmatrix} H & I \\ L & L \end{bmatrix}$

$$k^{-1} = \frac{1}{(HL) - (IL)} \begin{bmatrix} L & -I \\ -L & H \end{bmatrix}$$

$$= \frac{1}{(7\times11) - (8\times11)} \begin{bmatrix} 11 & -8 \\ -11 & 7 \end{bmatrix}$$

$$= \frac{1}{(77 - 88)} \begin{bmatrix} 11 & -8 \\ -11 & 7 \end{bmatrix}$$

$$= \frac{1}{-11} \begin{bmatrix} 11 & -8 \\ -11 & 7 \end{bmatrix}$$

$k^{-1} = ABCD$

$= \begin{bmatrix} A & B \\ C & D \end{bmatrix}$

$$k^{-1} = \frac{1}{(AD) - (BC)} \begin{bmatrix} D & -B \\ -C & A \end{bmatrix}$$

Now to remove minus we do this

for -11 = $N - (11 \bmod N)$
= $26 - (11 \bmod 26)$
= $26 - 11$
= 15

for -8 = $N - (8 \bmod N)$
= $26 - 8 \bmod 26$
= $26 - 8$
= 18

$$K^{-1} = \frac{1}{15}\begin{bmatrix} 4 & 18 \\ 15 & 7 \end{bmatrix}$$

$A \times A^* \bmod N = 1$
$\therefore 11 \times A^* \bmod N = 1$
$\therefore \boxed{A^* = 7}$

$$K^{-1} = \frac{1}{15}\begin{bmatrix} 11 & 18 \\ 15 & 7 \end{bmatrix} = 7\begin{bmatrix} 11 & 18 \\ 15 & 7 \end{bmatrix}$$

$$= \begin{bmatrix} 77 & 126 \\ 105 & 49 \end{bmatrix}$$

$$K^{-1} = \begin{bmatrix} 77 & 126 \\ 105 & 49 \end{bmatrix} \bmod 26$$

$$K^{-1} = \begin{bmatrix} 77 \bmod 26 & 126 \bmod 26 \\ 105 \bmod 26 & 49 \bmod 26 \end{bmatrix} = \begin{bmatrix} 25 & 22 \\ 1 & 23 \end{bmatrix}$$

$$P_1 = \begin{bmatrix} 25 & 22 \\ 1 & 23 \end{bmatrix} \begin{bmatrix} 4 \\ 4 \end{bmatrix} = \begin{bmatrix} 342 \\ 257 \end{bmatrix}$$

$$P_2 = \begin{bmatrix} 25 & 22 \\ 1 & 23 \end{bmatrix} \begin{bmatrix} 18 \\ 2 \end{bmatrix} = \begin{bmatrix} 494 \\ 64 \end{bmatrix}$$

$P_1 \bmod 26 = PT_1$

$\therefore \begin{bmatrix} 342 \\ 257 \end{bmatrix} \bmod 26 = PT_1$

$\therefore PT_1 = \begin{bmatrix} 4 \\ 23 \end{bmatrix} = \begin{bmatrix} E \\ X \end{bmatrix}$

$P_2 \bmod 26 = PT_2$

$\therefore \begin{bmatrix} 494 \\ 64 \end{bmatrix} \bmod 26 = PT_2$

$PT_2 = \begin{bmatrix} 0 \\ 12 \end{bmatrix} = \begin{bmatrix} A \\ M \end{bmatrix}$

$PT_1 \ddagger PT_2 =$ **EXAM**

Vernam cipher / One-Time-Pad cipher:

XOR

0 1 → 1

1 0 → 1

1 1 → 0

0 0 → 0

We need to make an XOR opn b/w pt & kt

Encryption:

P = 101011010

k = 101

```
P  1 0 1 0 1 1 0 1 0
k  1 0 1 1 0 1 0 1 0
⊕  0 0 0 1 1 0 1 1 1
```

CT = 000110111

Decryption:

C = 000110111

k = 101

```
C  0 0 0 1 1 0 1 1 1
k  1 0 1 1 0 1 0 1 0
⊕  1 0 1 0 1 1 0 1 0
```

PT = 101011010

* Block cipher modes of operations :

For different types of messages, we need different mode of operation.

(1) ECB : (Electronic Code Book)

→ Plain text is divided into a no. of fixed size blocks.

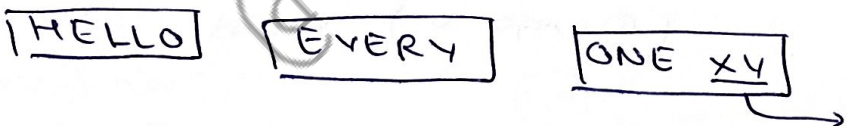
→ If message doesn't fit the size of block then do add additional character to fill that block size.

→ Take one block at a time and encrypt it.

→ same key is used for encryption and decryption

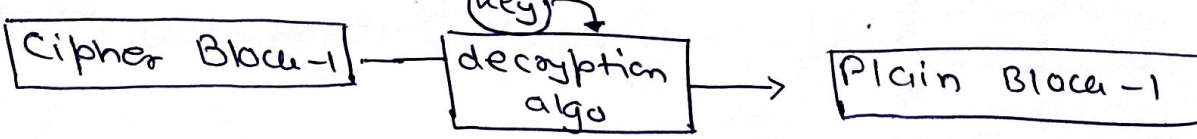
eg. PT: Hello Everyone

Block size : 5



Encryption:

Decryption :



Note:

↳ Best for short amount of data, such as key.

↳ Not secure for lengthy data

If identical block appears then this mode will produce same cipher

for eg. Hello (in block ①) Hello (in block 10)

↳ ABSFG ↳ *#$49

different o/p in CBC but not in ECB

To overcome this we have CBC,

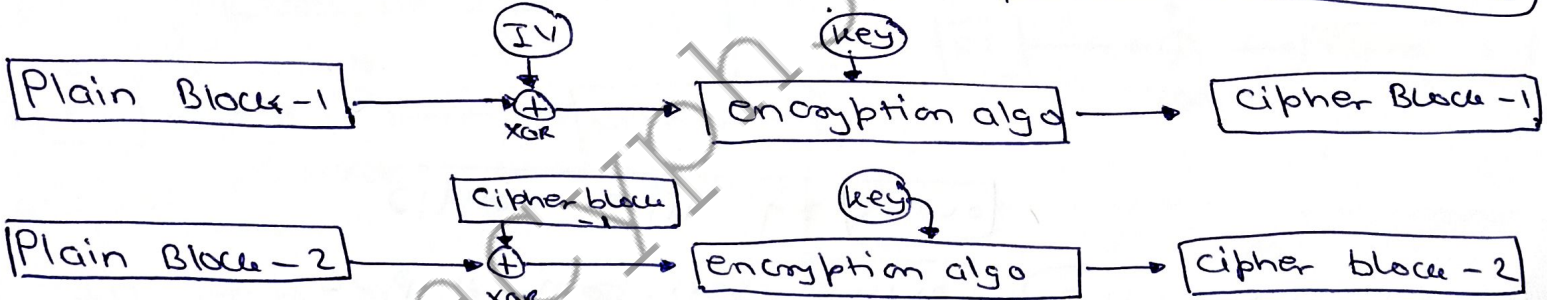
CBC — Cipher Block chaining mode

To overcome the limitation of ECB, in CBC 2 same block of text will give different cipher text as in above eg. of "Hello".

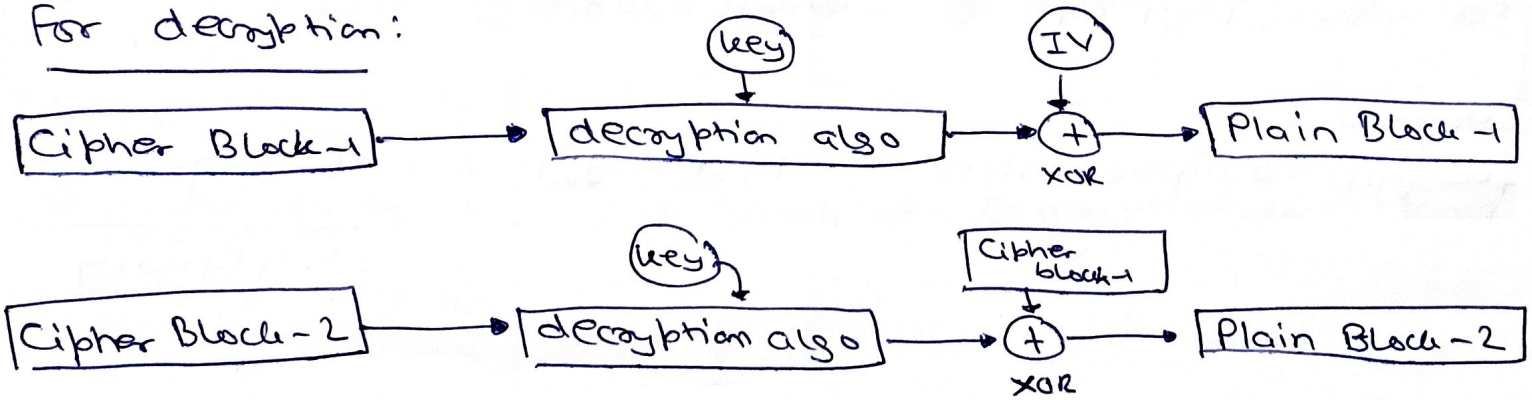
I/p to the encryption algo is XOR of the current plain text block and the previous cipher text block.

Same key will be used for encryption & decryption.

For first round we have IV → Initialization Vector



For decryption:



Note:

↳ IV must be known to both parties,
but not known to 3rd party.

Limitation:

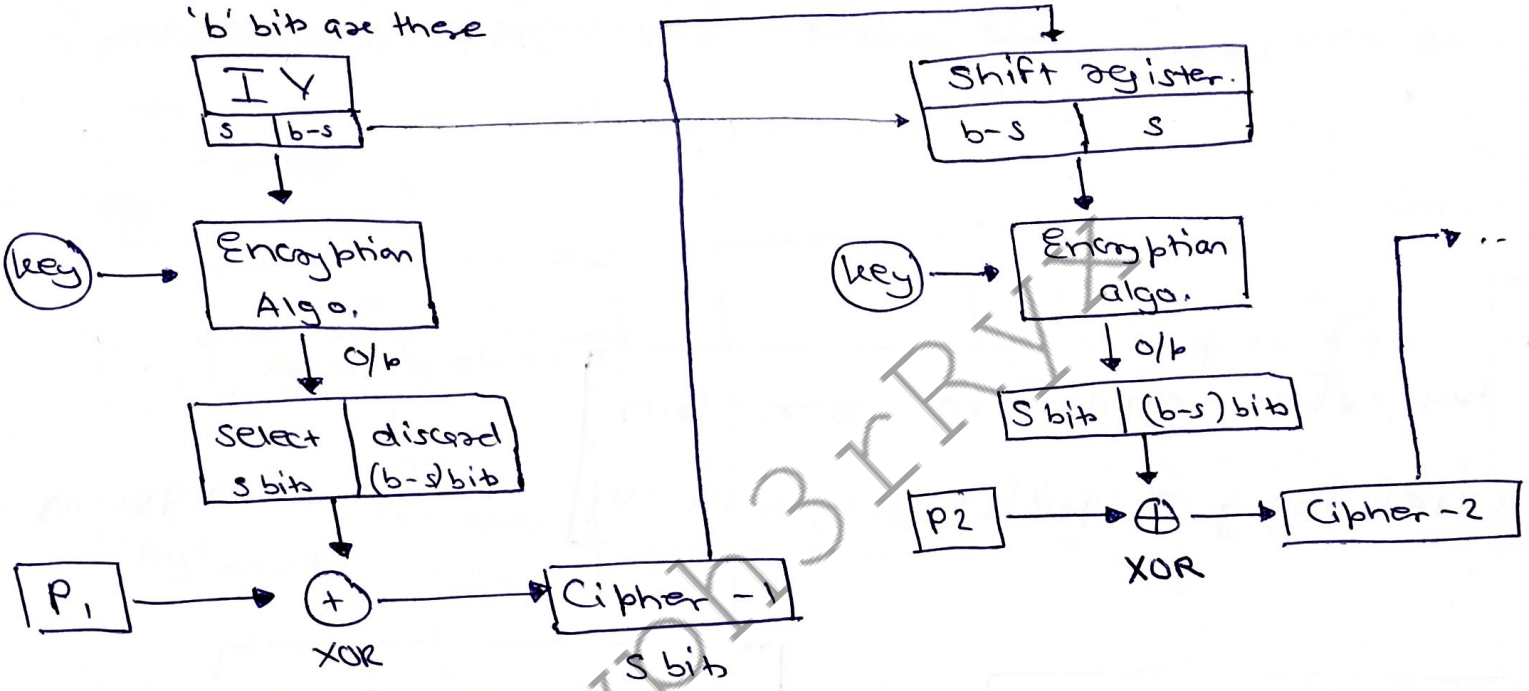
↳ As 2 different blocks will produce different ciphers

thus if we have 2 identical messages and if we use same (IV) then the cipher will be same.

CFB : Cipher feedback mode

The Plain text is divided into segment of 's' bits.
"s" can have any value.

Encryption:

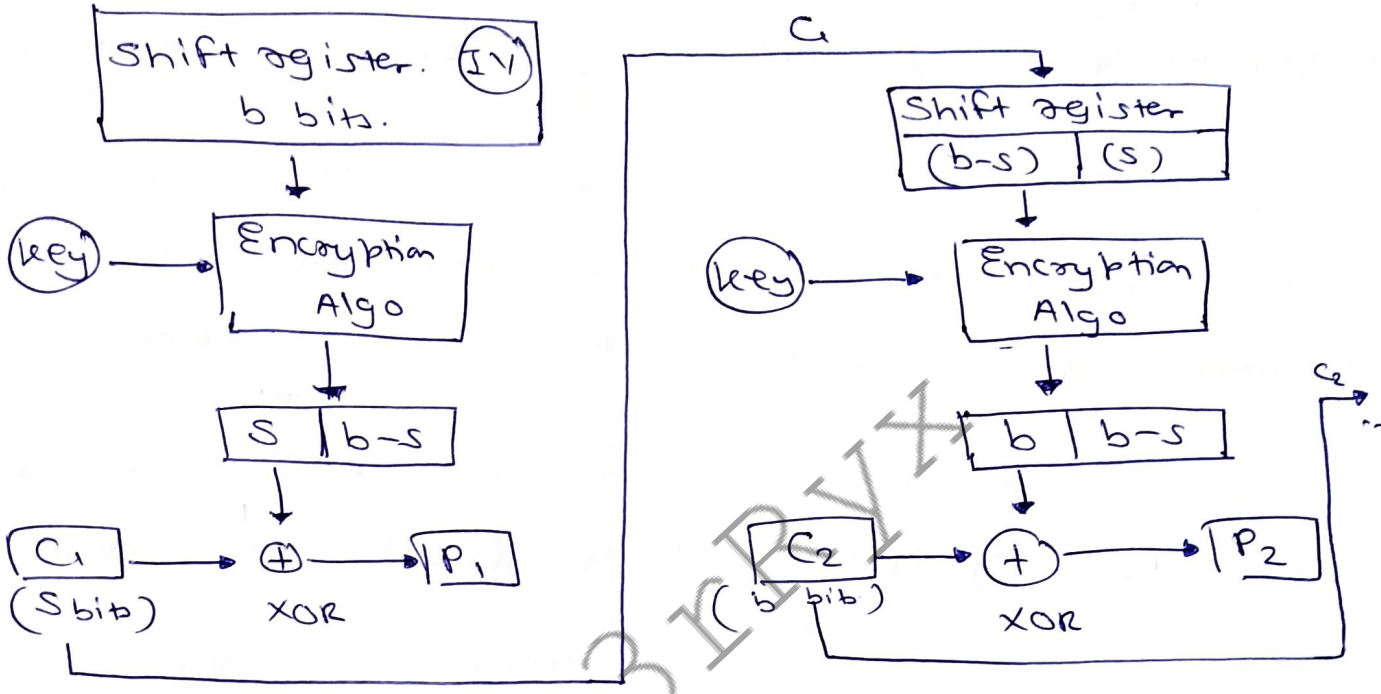


In shift register (b-s) bit is loaded from a left shift done on IV and (s) bits is loaded from cipher 1

As the name & figure suggest, we give cipher block as a feedback to the next block of encryption with same new specification.

IV is used for initial (1st) encryption & o/p bits are divided into 's' bits
'b-s' bits.

Decryption:



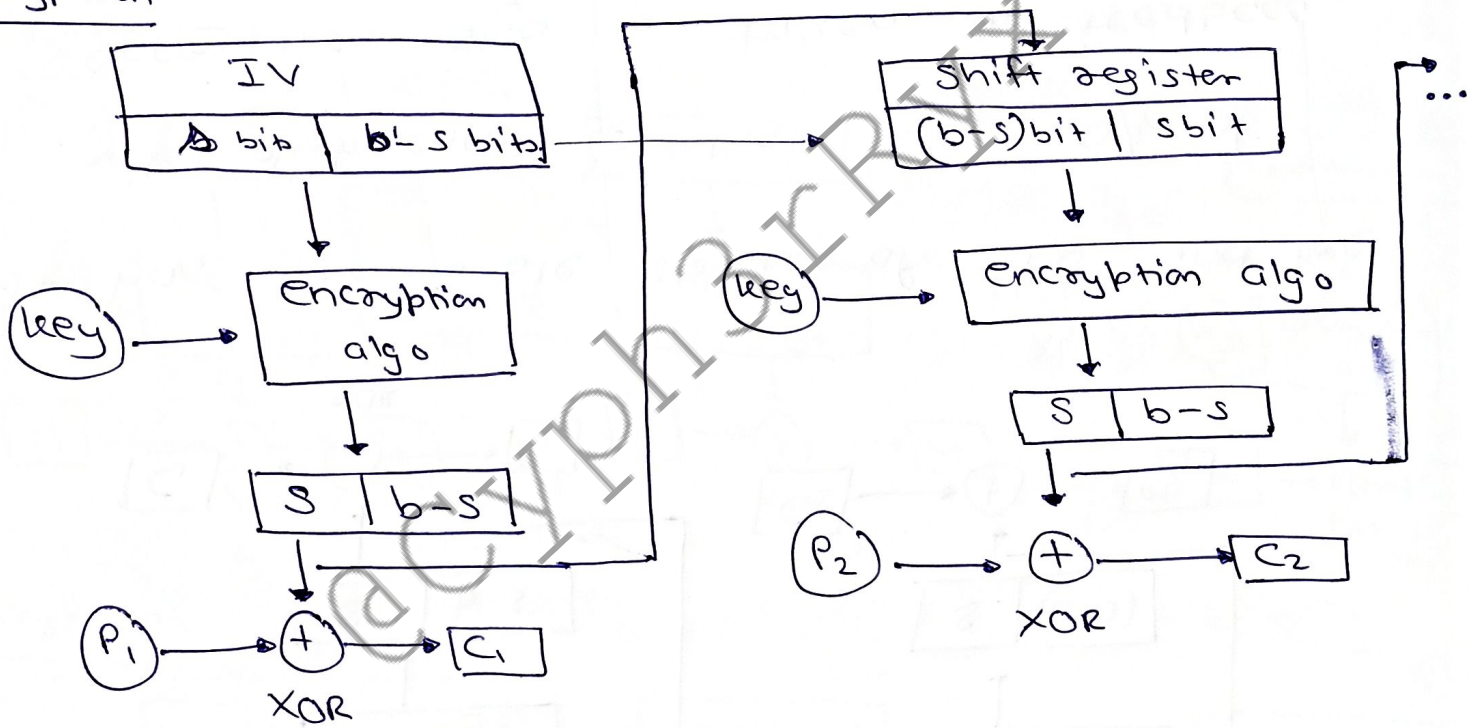
Notable thing is **in both Encryption & Decryption we use Encryption Algorithm.**

OFB (Output feedback mode) :

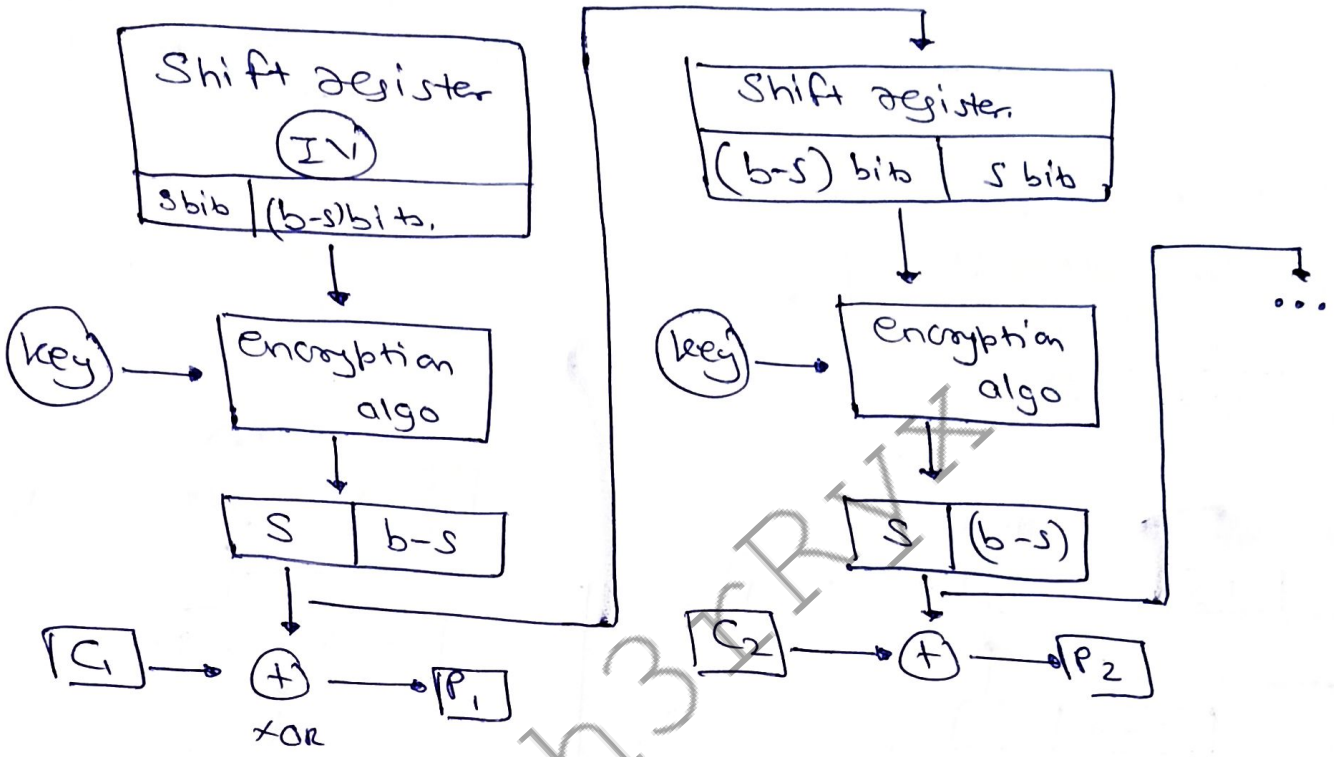
The structure is same to CFB

- The o/p of encryption fxn is given as feedback to the shift register in OFB, whereas in CFB the cipher text is given as feedback to shift register.

Encryption :



Decryption:

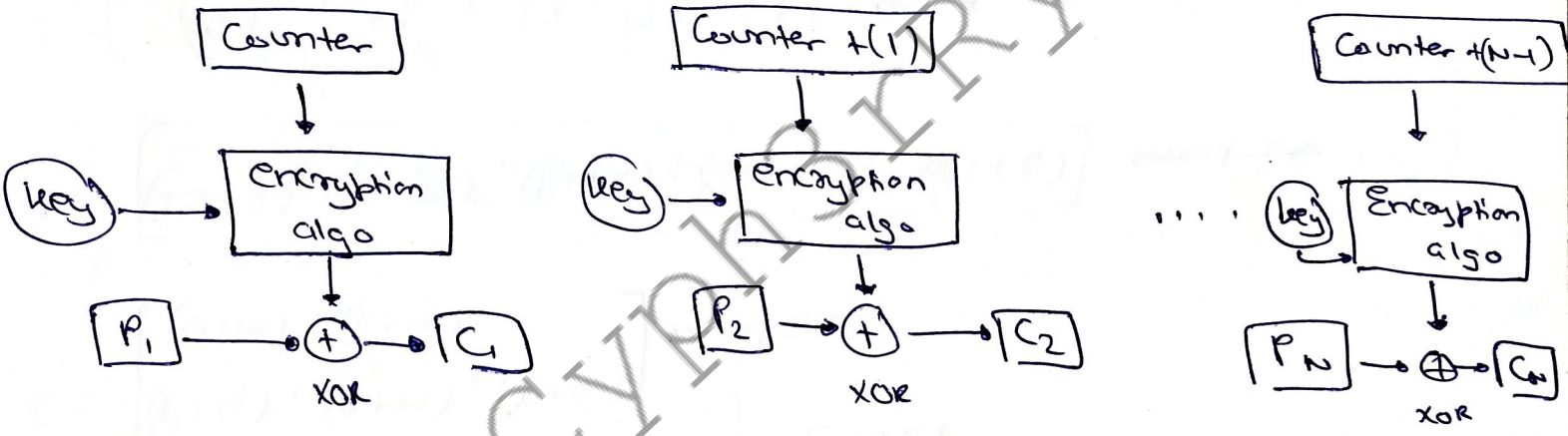


It has the whole structure of CFB and the vital difference is that here o/p of encryption algo is taken as feedback.

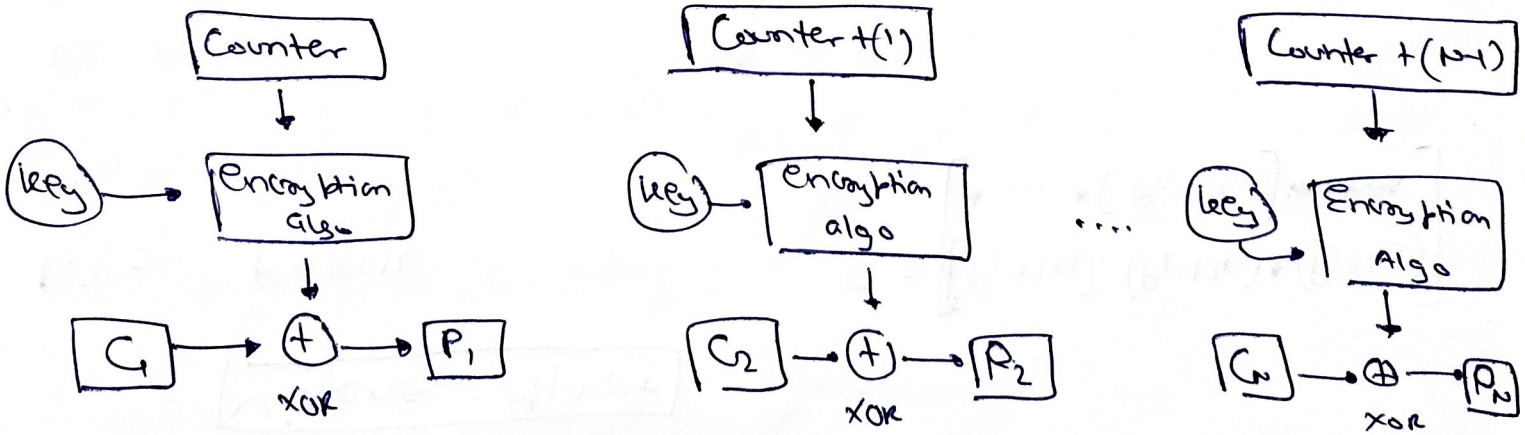
Counter mode :

- Simple & fast
- A counter, equal to plain text block size is used.
- A counter is initialized with some value and then incremented by (1) for each subsequent block.

Encryption :



Decryption :



Polyalphabetic cipher:

Vigener cipher

P.T = PARUL

key = cem

$$C = \left[(P_1 + k_1), (P_2 + k_2), (P_3 + k_3), \ldots, (P_n + k_n) \right] \bmod 26$$

P = PARUL

k = cemcmc

A → 0

B → 1

:

Z → 25

$$C = \left[(P + c), (A + m), (R + c), (U + m), (L + c) \right] \bmod 26$$

$$C = \left[(15 + c), (0 + 12), (17 + 6), (20 + 12), (11 + 6) \right] \bmod 26$$

$$C = (21, 12, 23, 32, 17) \bmod 26$$

Mod 26 is only used whenever the addition exceeds 25

$32 \bmod 26 = 6$

C = (21, 12, 23, 6, 17)

P = C = VMXGR

Decryption : $P = [(C_1 - k_1), (C_2 - k_2), (C_3 - k_3), \ldots (C_n - k_n)]$

Eg. 2 — P = PARUL UNIVERSITY

k = PIET

P = PARUL UNIVERSITY

k = PIETPIETPIETPIE

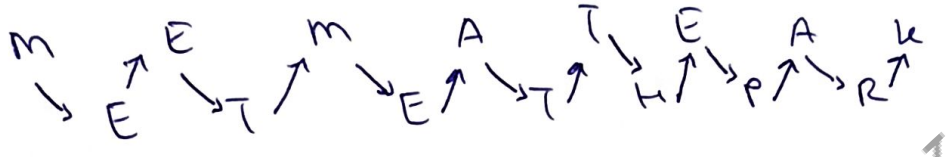
C = [(15+15), (0+8), (17+4), (20+19), (11+15), (20+8), (13+4), (8+19), (21+15), (4+8), (17+4), (18+19), (8+15), (19+8), (24+4)]

C = EIVNA CRBkmvLXBC

Railfence technique: (Transposition Technique)

MEET ME AT THE PARK

key Shift : (2)



CT: MEMATEAKETETHPR

key shift : (3)

CT: M M T A E T E T H P R E A E K

We can make it more difficult by rows and column.

Eg. KILL THE CORONA

k = 351624

	1	2	3	4	5	6
	K	I	L	L	T	H
	E	C	O	R	O	N
	A	X	Y	Z	W	B

↳ Padding.

CT will be according to key i.e 351624

LOY TOW KEA HNB ICX LRZ

RSA Algorithm:

If public key of [User A] is used for (Encryption) then we have to use private key of [A] for (Decryption)

key generations:

(1) Select 2 large prime nos. "p" & "q"

(2) Calculate $n = p \times q$

(3) $\phi(n) = (p-1) * (q-1)$

(4) Choose value of e

$1 < e < \phi(n)$ & $\gcd(\phi(n), e)$

↓ Private key ↓ Public key

(5) $d = e^{-1} \bmod \phi(n)$

$\Rightarrow ed = 1 \bmod \phi(n)$

$\therefore ed \bmod \phi(n) = 1$

(6) Public = $\{e, n\}$

Private = $\{d, n\}$

(7) Encryption => $PT^e \bmod n$

Decryption => $CT^d \bmod n$

Example :

let $p = 3$ $q = 11$

$n = p \times q = 33$

$\phi(n) = (p-1) \times (q-1) = (3-1)(11-1) = (2)(10) = \boxed{20}$

let $e = 3 \Rightarrow 1 < 3 < 20$

$\gcd(20, 3) = \boxed{1}$

$d = e^{-1} \bmod \phi(n)$

$de \bmod \phi(n) = 1$

$d(3) \bmod 20 = 1$

$d \cdot 3 \bmod 20 = 1$

$\therefore \boxed{d = 7}$

$e = 3$ encryption key

$d = 7$ decryption key

Public key = $\{e, n\}$

$= \{3, 33\}$

Private key = $\{d, n\} = \{7, 33\}$

Encryption $\Rightarrow C = 31^3 \bmod 33$

$= \boxed{25}$

Decryption $\Rightarrow P \Rightarrow 25^7 \bmod 33$

$= \boxed{31}$

Deffie Hellmann:

- Not an encryption algorithm
- Used to exchange secret keys betⁿ 2 users.

Algo:

(1) q is a prime no.

(2) α is primitive root of q

$$\Downarrow$$

$\alpha^1 \bmod q =$

$\alpha^2 \bmod q =$

$\vdots$

$\alpha^{q-1} \bmod q =$

Result = $\{1, 2, 3, \ldots, q-1\}$

order don't matter.

Check all values for all $\{1, 2, 3, \ldots, q-1\}$ by this

(3) $X \rightarrow$ Private key

$Y \rightarrow$ Public key

Person A:

- X_A (Private key)
- $Y_A = \alpha^{X_A} \bmod q$

Person B:

- X_B (Private key of B)
- $Y_B = \alpha^{X_B} \bmod q$

Now we will exchange the secret keys.

To exchange both,

sender & reciever

will use publickey

$K_A = (Y_B)^{X_A} \bmod q$; $K_B = (Y_A)^{X_B} \bmod q$

$K_A = K_B$

→ key is exchanged

Example:

Let $q = 7$

$\therefore \alpha < q$ i.e $= 5$ $\boxed{\alpha = 5}$

for 7 we have 3 & 5 both as a primitive root so we can choose any one of them.

key for P-1

let, $X_A = 3$ $(\because X_A < q)$

$Y_A = \alpha^{X_A} \bmod q$

$= (5)^3 \bmod 7$

$= 125 \bmod 7$

$\boxed{Y_A = 6}$

key for P-2

let, $X_B = 4$ $(\because X_B < q)$

$Y_B = \alpha^{X_B} \bmod q$

$= (5)^4 \bmod 7$

$\boxed{Y_B = 2}$

Secret key P-1

$K_1 = (Y_B)^{X_A} \bmod q$

$= (2)^3 \bmod 7$

$= 8 \bmod 7$

$\boxed{K_1 = 1}$

Secret key P-2

$K_2 = (Y_A)^{X_B} \bmod q$

$= (6)^4 \bmod 7$

$\boxed{K_2 = 1}$

$K_1 = K_2$

key exchanged

☺ Woohoo...

Annexure No :

Extended Euclidean Algo:

2 integer a & b we often need to find 2 integer s & t such that

$$(s * a) + (t * b) = \gcd(a, b)$$

Find gcd of (161, 28) & value of s & t
a, b

$s = s_1 - q s_2$ → $t = t_1 - q t_2$

q	r_1	r_2	r	s_1	s_2	s	t_1	t_2	t
5	161	28	21	1	0	1	0	1	-5
1	28	21	7	0	1	-4	1	-5	6
3	21	7	0	1	-4	4	-5	6	-23
	(7)	0		(-4)	4		(6)	-23	
	↓ GCD			↓ s			↓ t		

GCD = 7

s = -4

t = 6

verify it via formula

$$(s \times a) + (t \times b) = \gcd(a, b)$$

$$\therefore (-4 \times 161) + (6 \times 28) = \gcd(161, 28)$$

$$\therefore -161 + 168 = \gcd(161, 28)$$

$$\therefore \boxed{7 = \gcd(161, 28)}$$

Verified.

Enrollment No :

Page No :

* Euclidean Algorithm:

↳ used for finding gcd.

$$\gcd(a,b) = \gcd(b, a \bmod b)$$

$$\gcd(a,0) = a$$

Eg. $\gcd(11,7)$

$\Rightarrow \gcd(7, 11 \bmod 7)$

$= \gcd(7,4)$

$\Rightarrow \gcd(4, 7 \bmod 4)$

$= \gcd(4,3)$

$\Rightarrow \gcd(3, 4 \bmod 3)$

$= \gcd(3,1)$

$\Rightarrow \gcd(1, 3 \bmod 1)$

$= \gcd(1,0)$

$\Downarrow$

1

gcd (2740, 1760)

q	r_1	r_2	r ($r_1 \% r_2$)
1	2740	1760	980
1	1760	980	780
1	980	780	200
3	780	200	180
1	200	180	20
9	180	20	0
	20	0	

↑ gcd

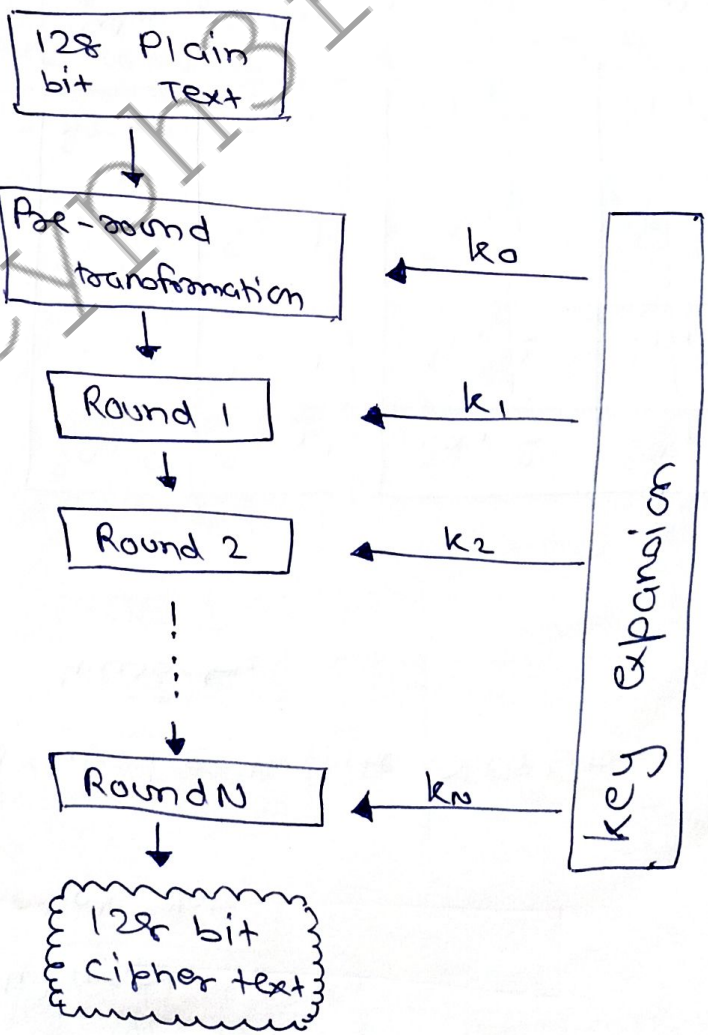
* A.E.S Algorithm:

Advance Encryption Standards

Fixed size = 128 bits for blocks

Rounds	No. of bits in key
10	128
12	192
14	256

- bit ⇒ 0 or 1
- 1 byte ⇒ 8 bit
- 1 word ⇒ 32 bits/4 byt
- Block size ⇒ 128 bits
  16 byte



State : 16 byte (4×4) matrix
Stores immediate data

INPUT ARRAY : 4×4 i.e 16 byte => 128 bits.
4 words.

- State array:

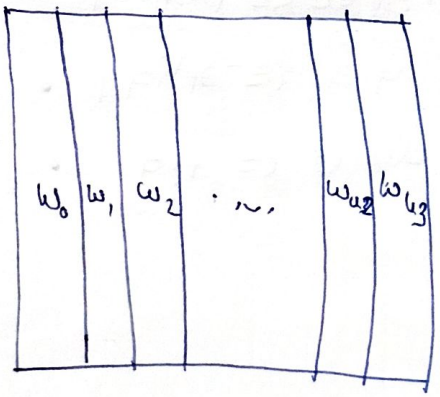
$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$	w_0
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$	w_1
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$	w_2
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$	w_3

$[w_0, w_1, w_2, w_3]$

- Key expansion: key = 128 bit => 4 words.

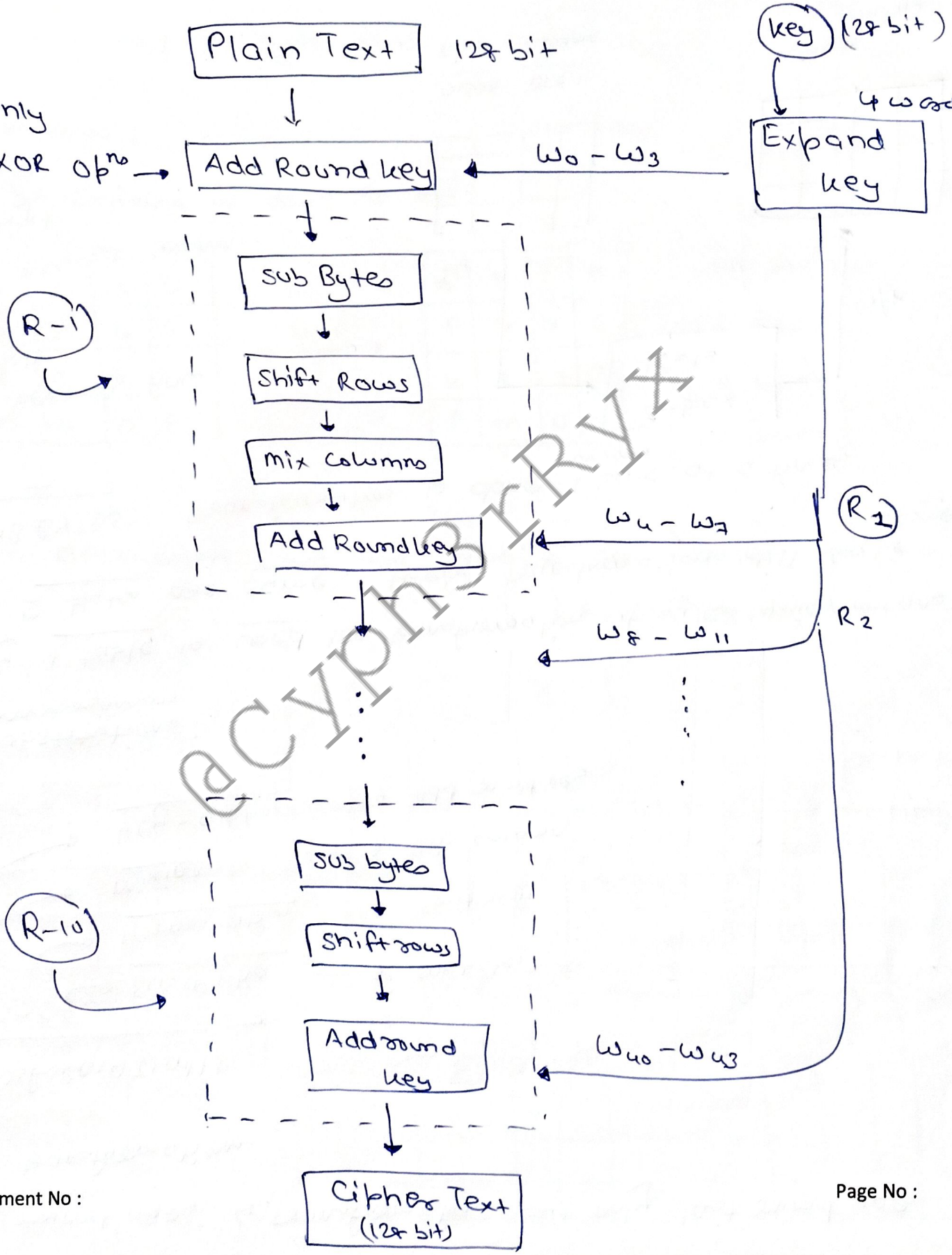
k_0	k_4	k_8	k_{12}
k_1	k_5	k_9	k_{13}
k_2	k_6	k_{10}	k_{14}
k_3	k_7	k_{11}	k_{15}

Expand key Algorithm →



44 words.

Structure of each round:



Each round uses 4 transformation but only last round uses 3 transformations.

TRANSFORMATIONS:

- Substitution → sub bytes
- Permutation → Shift rows
- Mixing → mix column
- key-adding → Add round key

(1) Substitutions:

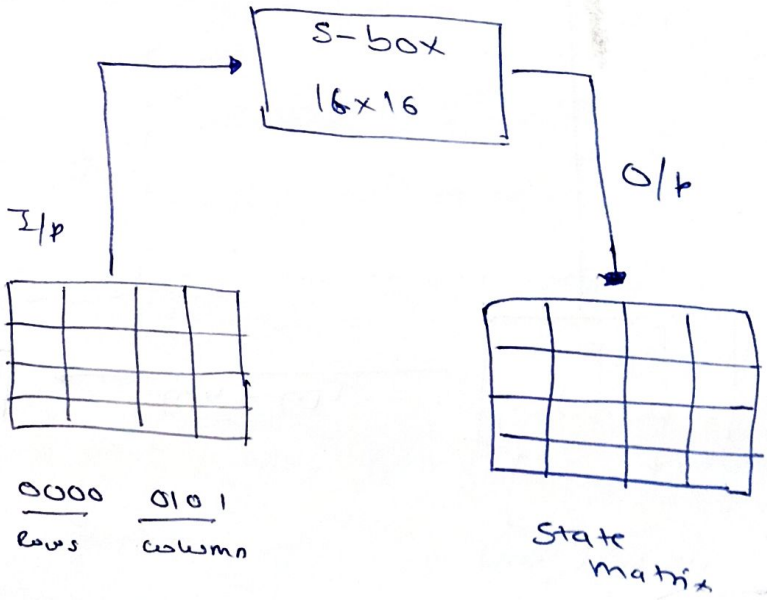
Only 1 table is used for transformation of bytes which means if 2 bytes are same, their transformation will also be same.

SUB BYTES:

Transformation is done 1 byte at a time.

1st hex digit ⇒ Row

2nd hex digit ⇒ Column



Permutation :

↳ done on byte level

Shift rows :

↳ Shifting is done to left.

A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P

Shift →

A	B	C	D
F	G	H	E
K	L	I	J
P	M	N	O

NO SHIFT
1 SHIFT
2 SHIFT
3 SHIFT

In decryption we use inverse shift rows
i.e Shifting to right

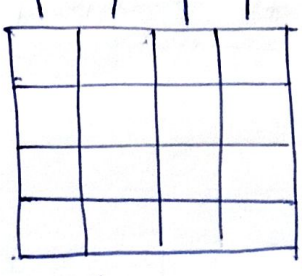
Mixing :

(S-1) Take each word/column i.e 4 byte / 4x1 matrix & multiply it with a constant matrix.

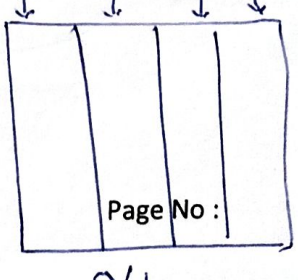
(S-2) The o/p (4x1) matrix is stored in state matrix

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \times \begin{bmatrix} \quad \\ \quad \\ \quad \\ \quad \end{bmatrix} = \begin{bmatrix} \quad \\ \quad \\ \quad \\ \quad \end{bmatrix}$$

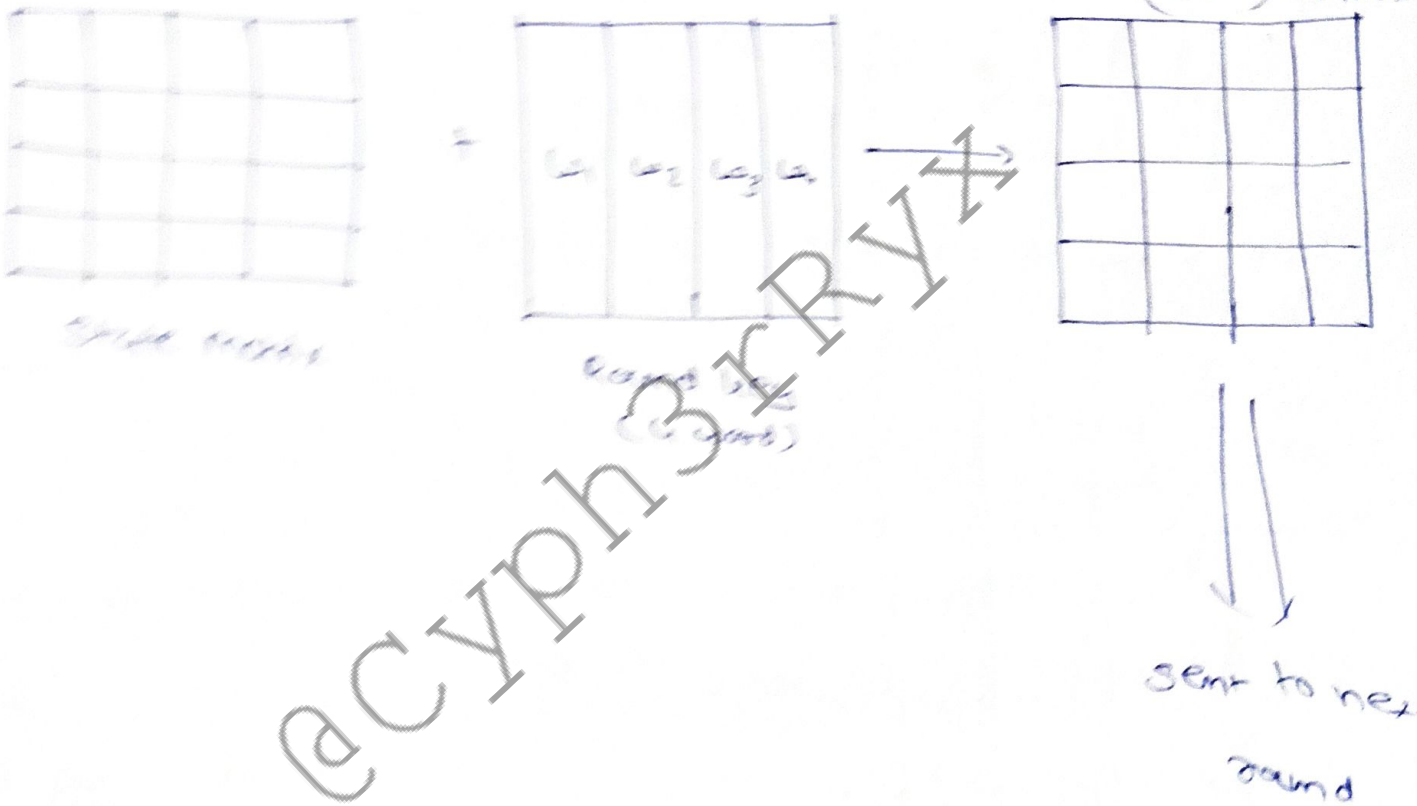
(constant) (4x1)



I/P



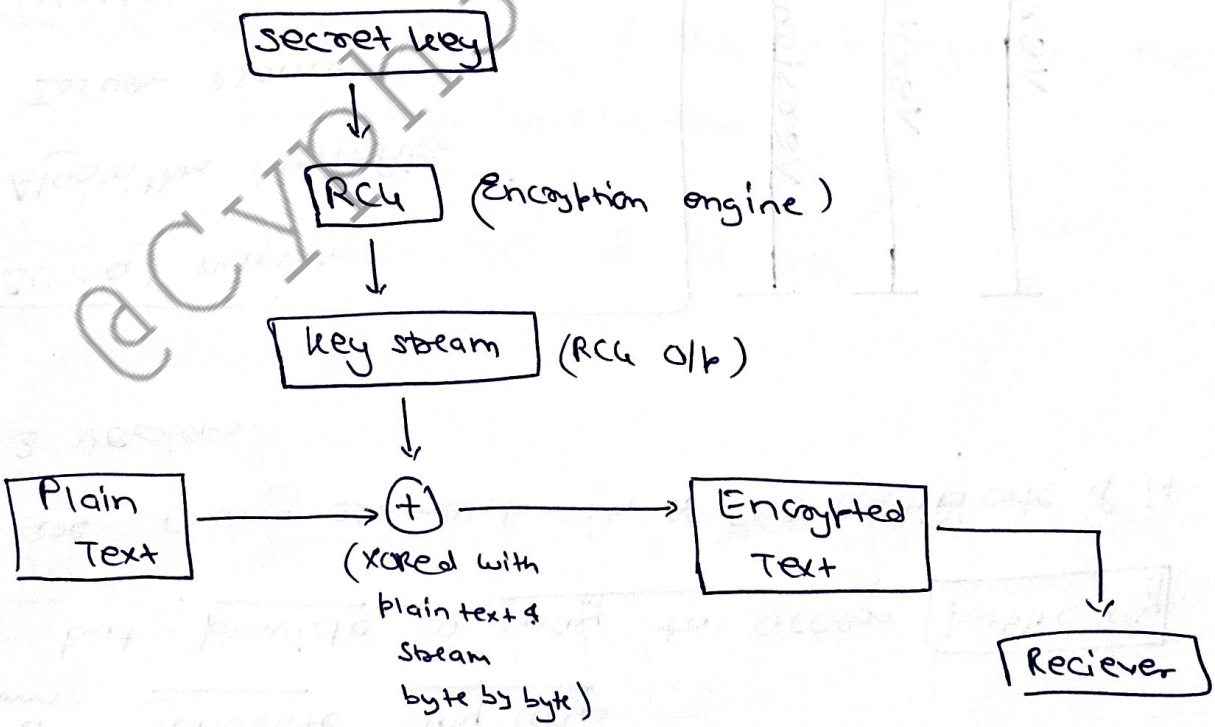
O/P



RC4:

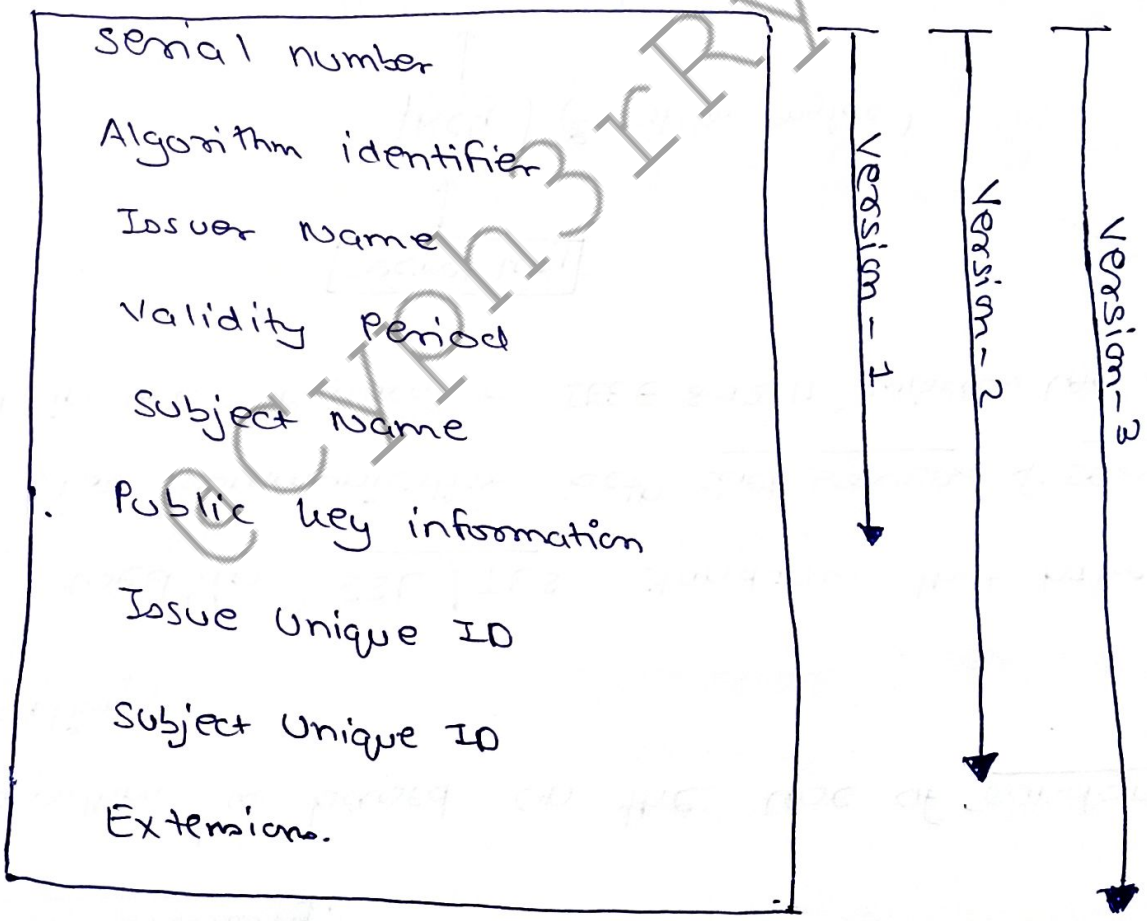
- Created by **Ron Rivest** in **1987** for **RSA security**.
- The algorithm is based on the use of **random permutation**.
- RC4 is used in **SSL / TLS** standards that have been defined for communication bet^n **web browsers** & **servers**.

Also used in WEP Protocol – IEEE 802.11 wireless LAN std.



X.509 :

- digital certificate accepted internationally
- does not generate any keys
  but provide a way to access public key.
- There are many elements in X.509 certificate & it has 3 versions.



(1) Serial no. : Certificate's serial no.

(2) Algorithm identifier : Algo used by the issuer for encryption

(3) Issuer Name : Name of the issuer / organization

(4) Validity Period : from - to (date & time)

(5) Subject Name : To whom we are providing the certificate.

(6) Public key Info. : To encrypt/decrypt the message, user will use public key & the info regarding that public key will be here.

(7) Issue Unique ID : Unique ID of Issuer

(8) Subject Unique ID : Unique ID of subject

(9) Extensions : Optional but add custom functionalities

Differential	Linear
(1) Block of bit	(1) Single bit
(2) Chosen plain text attack.	(2) known plain text attack
(3) known PT is 2^{55}	(3) known PT is 2^{43}
(4) Chosen PT is 2^{47}	(4) Chosen PT is 2^{43}