we get CT.

Encryption happens in each round :-

We divide the input plain text into two registers A and B each of size w bits. After undergoing the Encryption process the result of A and B together forms the Cipher Text block.

1. One time initialization of PT blocks A & B by adding S[0] & S[1] to A & B respectively. These operations are mod 2^w.
2. XOR A and B. $A = A \wedge B$.
3. Left shift new value of A & B.
4. Add S[2*i] to the o/p of previous step. this is the new value of A.
5. XOR B with new value of A and store in B.
6. Left shift new value of B by A bits.
7. Add S[2*i+1] to the o/p of previous step. This is the new value of B.
8. Repeat entire procedure (Except One time Initialization) r times.

$A = A + S[0]$

$B = B + S[1]$

for $i = 1$ to r do:

$A = ((A \wedge B) \lll B) + S[2*i]$

$B = ((B \wedge A) \lll A) + S[2*i+1]$

return A, B

RC5, decryption can be defined as:

for i=r down to 1 do:

$B = ((B - S[2*i+1]) >>> A) \wedge A$

$A = ((A - S[2*i]) >>> B) \wedge B$

$B = B - S[1]$

$A = A - S[0]$

return A, B

IDEA (International Data Encryption Standard) (IDEA)

→ PlainText → 64 bits is divided into 4 blocks

That is size of each block is 16 bytes

64 bits → 16 bytes, 16, 16, 16

→ Key size → 128 bit

→ No. of rounds are 17. IDEA algorithm perform 17 rounds.

Based on no. of round the key size, 128 bit is divided into subkeys.

→ Figure Explanation.

→ First the PT is divided into 4 blocks that is 64 bit as i/p

→ ~~Once~~ This 4 blocks are applied to round 1 Operation. For this round 1 we should take i/p & produce key.

Key Size 128 bit Key, from 128 bit Key we have to perform Key expansion. From key Expansion we have to generate 4 Keys.

Round 1:-

→ We are giving for round 1 input as 16 bit and 4 Keys $(\underbrace{K1, K2, K3, K4}_{16\,bit})$

→ Round 2:-

For round 2 we give 1st round output and we generate Sub Keys Only 2 Keys (K5 & K6)

→ Round 17:-

For Round 17 the Sub Keys are 4 they are (K49, K50, K51, K52)

Then last we get Cipher Text of 16 bytes that is 64 bit.

→ Key Expansion: 128 bit is Expanded into 52 Sub Keys.

→ Odd round ~~take~~ generates 4 keys.

→ Even round generates 2 Keys.

Key Expansion:-

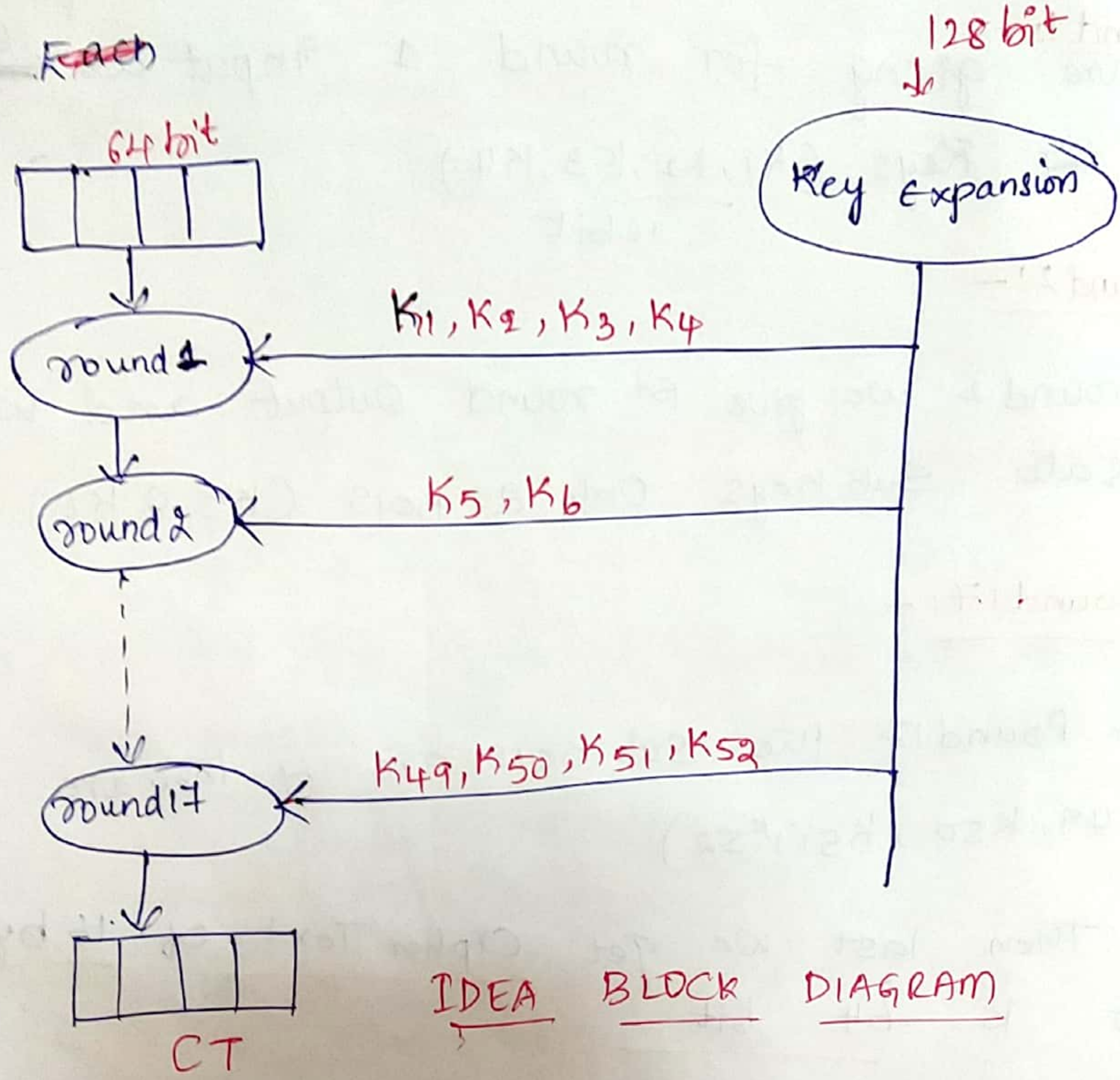
128-bit is divided into K1 to K8.

Each Key (K1) has 16 bytes.

$\boxed{16 \times 8 = 128 \text{ bits}}$

→ According to our algorithm, Our Key is Converted into 52 Sub Keys.

→ ~~Each~~

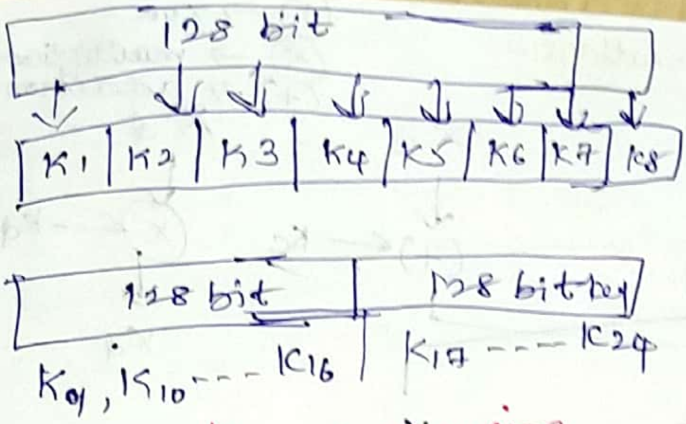


IDEA BLOCK DIAGRAM

→ Each Sub Key has 16 bytes.

The procedure for generating next 8 Keys is, 128 bit append again 128 bit, from 1st bit Generate Key from 25th bit $K_9, K_{10} --- K_{16}$.

After 128-bit Key then $K_{17}, K_{18} ---- K_{24}$.



Key expansion

Round Operations:-

Rounds are of two types:

1. Odd Round
2. Even Round.

→ For each round the PT size is 64 bits. Assume that that is given to 4 blocks. For odd round. For, they are x_a, x_b, x_c & x_d. for odd round. This is the Even round x_a, x_b, x_c & x_d. intially the blocks are divided into block size 4 blocks.

Odd round x_a, x_b, x_c, x_d

Keys are k_a, k_b, k_c, k_d

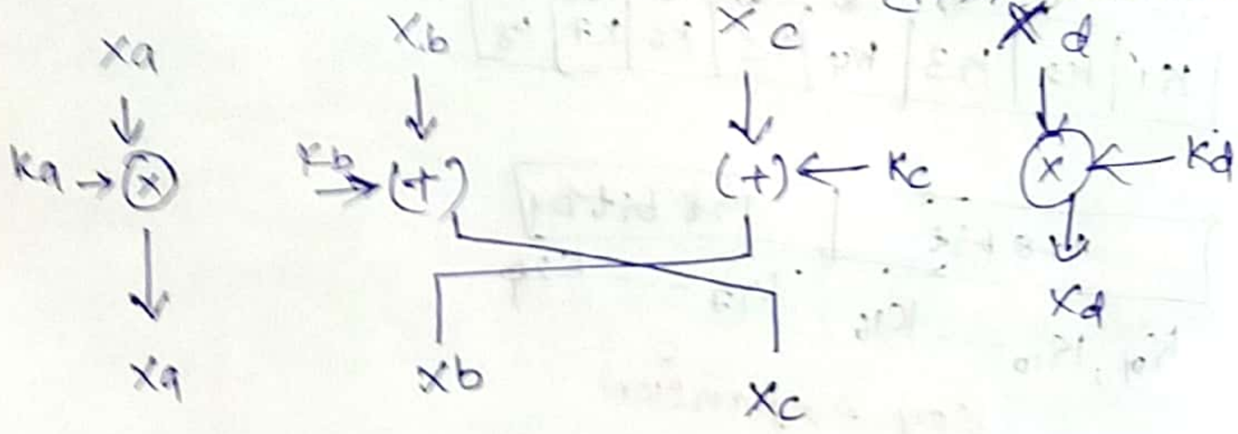
Even round x_a, x_b, x_c, x_d

Keys are k_e, k_f.

By using this keys how we perform operation on each round.

odd round Operations:-

(+) → XOR
(×) → multiplication
(+) → addition



On first block Xa, we apply this Xa, Ka

Xb, Xc values are interchanged.

odd round:

$X_a = X_a \otimes K_a$

$X_b \Rightarrow X_c (+) K_c$

$X_c = X_b (+) K_b$

$X_d = X_d \otimes K_d$

Even Round :-

$X_a \quad X_b \quad X_c \quad X_d$

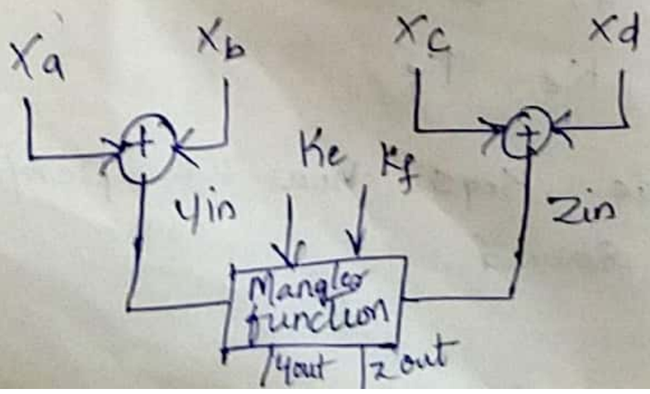
Keys ⇒ $K_e \quad K_f$

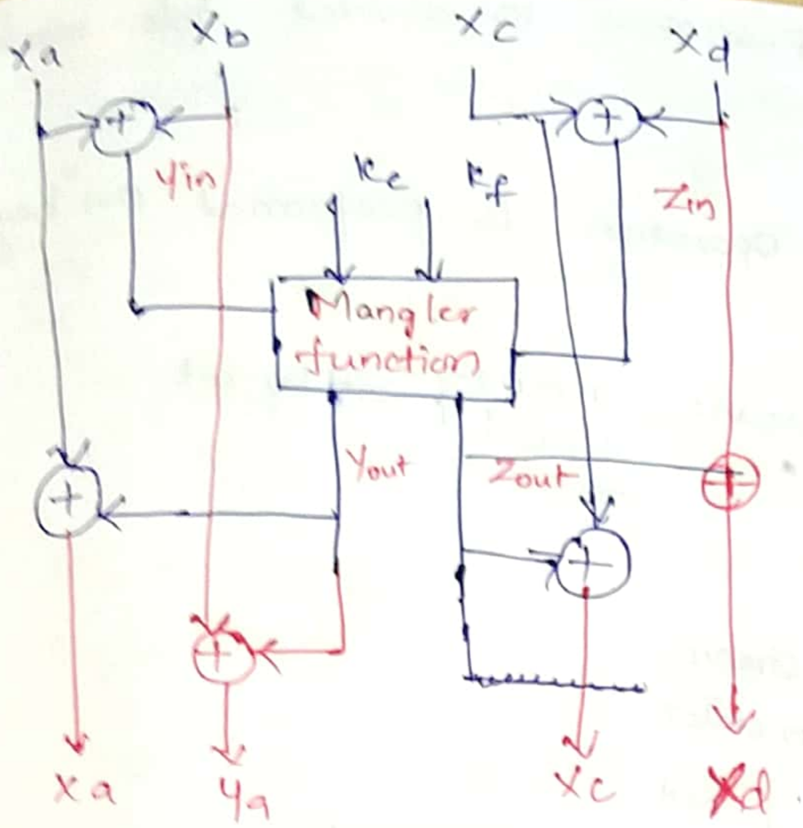
Consider 2 parameters : $X_a \oplus X_b = Y_{in}$

$X_c \oplus X_d = Z_{in}$

To get result we apply Mangler function

Figure:-





$X_a = X_a \oplus y_{out}$

$X_b = X_b \oplus Y_{out}$

$X_c = X_c \oplus Z_{out}$

$X_d = X_d \oplus Z_{out}$.

In this the 4 block (X_a, X_b, X_c, X_d) are converted into Yin & Zin. This 2 blocks and 2 keys are applied to function. After applying keys to function that produces the result as Yout & Zout.

→ The values of Yout & Zout is :-

$Y_{out} = ((K_e \otimes Y_{in}) + Z_{in}) + K_f$

$Z_{out} = ((K_e \otimes Y_{in}) + Y_{out})$.

Stream Cipher:- Plain Text is divided into number of Streams.

→ Bitwise XOR Operation is performed on "Key and "plainText".

Ex: Bit means
1 0 0 1
1 1 0 1 — bit by bit.

Encryption:-

Key stream Generator

↓ Key

Plain Text → ⊕ → C (Cipher Text)

Decryption:-

Keystream Generator

↓ Key

Cipher Text → ⊕ → plainText

→ In this "Key Stream Generator" will generate the "Key"

Encrypt

$m_1 \quad m_2 \quad m_3 \; ---- \; m_i \rightarrow$ Plain Text

$\oplus \; k_1 \quad k_2 \quad k_3 \; ---- \; k_i \rightarrow$ Keys

$c_1 \quad c_2 \quad c_3 \; ---- \; c_i \rightarrow$ CT

Decryption:-

(CipherText ⊕ Key ⇒ PlainText) decryption.

$$\begin{array}{lllllll} & C_1 & C_2 & C_3 & C_4 & --- & C_i \\ \oplus & K_1 & K_2 & K_3 & K_4 & ---- & K_i \\ \hline & P_1 & P_2 & P_3 & P_4 & ----- & P_i \end{array}$$ PlainText

Ex:-

$$\begin{array}{ll} PT \rightarrow & 1100 \\ Key \rightarrow & \oplus 1011 \\ \hline CT \rightarrow & 0111 \end{array} \qquad \begin{array}{ll} & 0111 \rightarrow CT \\ \oplus & 1011 \rightarrow Key \\ \hline & 1100 \rightarrow PT. \end{array}$$

RC4 Algorithm:-

→ RC4 Algorithm is a stream Cipher Algorithm.

→ RC4 is designed in 1987 by Ron Rivest.

→ It is a variable key size stream cipher with byte-oriented operations.

→ The Algorithm is based on the use of a random permutation.

RC4 Algorithm has 3 steps. they are:

1. Key scheduling.
2. Key stream Generation.
3. Encryption & decryption.

1 Key Scheduling:

→ We have several itterations.

→ The itterations will be depend on the size of S-array.

For Example:- the size of S-array is having "8". We have to do "0 to 7" iterations.

In, Key scheduling we have Algorithm.

$j = 0$

for $i = 0$ to 255 do

$j = [j + S(i) + T(i)] \bmod 256$

Swap (S[i], S[j]);

S(i) means → State vector

T(i) means → it is key array. It is Temporary vector

256 means ⇒ Depends on the size.

If the size is 8.

then i = 0 to 7 do

Ex:- then $j = [j + S(i) + T(i)] \bmod 8$

Swap (S[i], S[j]);

Ex:- S-array = $\begin{array}{cccccccc} S[0] & S[1] & S[2] & (3) & (4) & (5) & (6) & (7) \end{array}$ $\begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{bmatrix}$

Key array = [1 2 3 6] already given the key array.

plain Text = [1 2 2 2] This is also given

→ Initialise T-array with key

→ Size of 'S'-array & T-array Should be Equal. We have to repeat the key.

T = $\begin{array}{cccccccc} T(0) & T(1) & & & & & & \end{array}$ $\begin{bmatrix} 1 & 2 & 3 & 6 & 1 & 2 & 3 & 6 \end{bmatrix}$

why we are repeating key in T-array, means the size of T-array should be equal to S-array.

Ex:- 1 iteration:

1. $j = 0$

for $i = 0$ to 7

$j \Rightarrow [0+0+1] \bmod 8$ $\quad [j + S[i] + T(i)] \bmod 8$

$j \Rightarrow 1 \bmod 8 \Rightarrow 1$

$j \Rightarrow 1$

swap $S(0)$ and $S(1)$

$$S \Rightarrow \begin{array}{cccccccc} S(0) & S(1) & S(2) & S(3) & & & & \\ \end{array}$$
$$S \Rightarrow [1 \quad 0 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7]$$

2. For $i = 1$, $j = 1$

$j \Rightarrow [1+0+2] \bmod 8$

$\Rightarrow 3 \bmod 8$

$j \Rightarrow 3$

swap $S(1)$ & $S(3)$ $\rightarrow$ Swap $(S[i], S[j])$;

$S \Rightarrow [1 \quad 3 \quad 2 \quad 0 \quad 4 \quad 5 \quad 6 \quad 7]$

3. For $i = 2$, $j = 3$

$j \Rightarrow [3+2+3] \bmod 8$

$\Rightarrow 8 \bmod 8$

$j \Rightarrow 0$

swap $S[2]$ & $S(0)$ $\Rightarrow S \Rightarrow [2 \quad 3 \quad 1 \quad 0 \quad 4 \quad 5 \quad 6 \quad 7]$

We should do upto $i = 7$.
→ 8 itteration. After 8^{th} itteration we got

$\Rightarrow [\overset{S(0)}{0} \;\; \overset{S(1)}{7} \;\; \overset{S(2)}{1} \;\; 4 \;\; 6 \;\; 2 \;\; 5 \;\; 3]$

Step 2:- Stream Generation:

No. of Iterations = Size of Key.

Ex: Size of Key = 4

0 to 3,

1. i, j = 0;

```
while (true)
i = (i+1) mod 256;
j = (j + S[i]) mod 256;
Swap (S(i), s(j)); → we should not use state array
t = (S[i] + s[j]) mod 256
K = S[t];
```

Last 8^{th} iteration we get state array that should be taken in i, j ...

$4)7(1$
$\underline{4}$
(3)

$i = (0+1) \bmod 4$

$i = 1 \bmod 4 \Rightarrow 1$.

$4)11(2$
$\underline{8}$
(3)

$j \Rightarrow (0 + 7) \bmod 4 \Rightarrow 7 \bmod 4$

$j \Rightarrow 3$.

Swap (S(1) & S(3));

$\Rightarrow [0 \;\; 4 \;\; 1 \;\; 7 \;\; 6 \;\; 2 \;\; 5 \;\; 3]$

$t = (4 + 7) \bmod 4$

$\Rightarrow 11 \bmod 4 \Rightarrow 3 \Rightarrow t = 3$

$K = S[t] \Rightarrow K = S[3]$

$\boxed{K(0) = 7}$

2. i,

i=

j

3.

4.

2. $i, j = 1;$

$i = (1+1) \bmod 4 \Rightarrow i \Rightarrow 2 \bmod 4 \Rightarrow 2 \Rightarrow i.$

$j = (1+4) \bmod 4 \Rightarrow 5 \bmod 4 \Rightarrow j \Rightarrow 1.$

swap $S(i), S(j)$

swap $\Rightarrow S(2), S(1)$

$\Rightarrow [0 \; 1 \; 4 \; 7 \; 6 \; 2 \; 5 \; 3]$

$t = (S(2) + S(1)) \bmod 4$

$\Rightarrow (4+1) \bmod 4 \Rightarrow 5 \bmod 4 \Rightarrow t = 1.$

$K = S(1);$ $\boxed{\Rightarrow K1 \Rightarrow 1}$

3. $i, j = 2;$

$i = (2+1) \bmod 4 \Rightarrow 3 \bmod 4 \Rightarrow i \Rightarrow 3$

$j = (2+4) \bmod 4 \Rightarrow 6 \bmod 4 \Rightarrow j \Rightarrow 2$

Swap $S(3), S(2)$

$\Rightarrow [0 \; 1 \; 7 \; 4 \; 6 \; 2 \; 5 \; 3]$

$t \Rightarrow (S(3) + S(2)) \bmod 4 \Rightarrow (4+6) \bmod 4 \Rightarrow 10 \bmod 4$

$\boxed{K(2) \Rightarrow 7}.$ $t \Rightarrow 2.$

4. $i, j = 3;$

$i = (3+1) \bmod 4 \Rightarrow 4 \bmod 4 \Rightarrow i = 0$

$j = (3+0) \bmod 4 \Rightarrow 3 \bmod 4 \Rightarrow j = 3$

swap $S(0), S(3)$

$\Rightarrow [7 \; 1 \; 7 \; 0 \; 6 \; 2 \; 5 \; 3]$

$t \Rightarrow (S(0) + S(3)) \bmod 4 \Rightarrow 4 + 0 \bmod 4 = 4 \bmod 4$

$t = 0$

$\boxed{K(3) \Rightarrow 4}$

te array.
so we get
that should
in i, j values

2)4(2
4
0

4)10(2
8
2

K= [7 1 7 4]

∵ New key array is obtained.

For Encryption and decryption we will use new key

→ **Encryption and Decryption:-**

Encryption:- PT XOR New key.

(First convert into binary)

Ex: PT- 1 2 2 2

PT →	0001	0010	0010	0010
New key →	7	1	7	4
	0111	0001	0111	0100

↳ After converting into binary we should perform "XOR ⊕" operation then we will get CT (Cipher Text)

Decryption:- CT XOR New key.

CT
⊕ New key then we get PT (Plain Text)

~~Distribution~~:- **Assymetric Key Ciphers:-**

Symmetric key means only one key.

Sender $\xrightarrow[\text{Key}]{}$ Receiver

Sender encrypts msg by using a key Receiver decrypts the msg with the same key.

~~Assymetric~~

The attacker can attack while sending the msg. Then receiver decrypts the msg and he will read that msg. To overcome this problem we are moving to another technique that is Assymmetric key cryptography. ~~and~~ and public key

Here, in Assymmetric key there are two keys.
→ 1 is for Encryption & other key is for decryption.

The Elements we used in Public key Cryptography are:-

1. plain Text → This is i/p to the Algorithm
2. Encryption Algorithm
3. Keys
4. Decryption Algorithm
5. Cipher Text.

→ Encryption Algorithm with ~~some~~ ~~we~~ we use converts the PT into some Unreadable format.

Keys: To perform ~~Keys~~ ~~we~~ ~~use~~ Encryption Algorithm use some keys.

Keys are divided into 2 types:

1. Public Key
2. Private Key

} for Encrypt^n & Decrypt^n we use this keys.

4. Decryption Algorithm: When sender sends

msg to Receives the Receiver will decrypt msg into Readable format

5. **Cipher Text:** Unreadable format.

→ To convert PT to CT we use Encryptn Algorithm.

→ To convert CT to PT we use decryption Algorithm.

→ To perform Encryptn algorithm (or) decryption Algorithm we are using Keys.

Public Key Cryptography:-

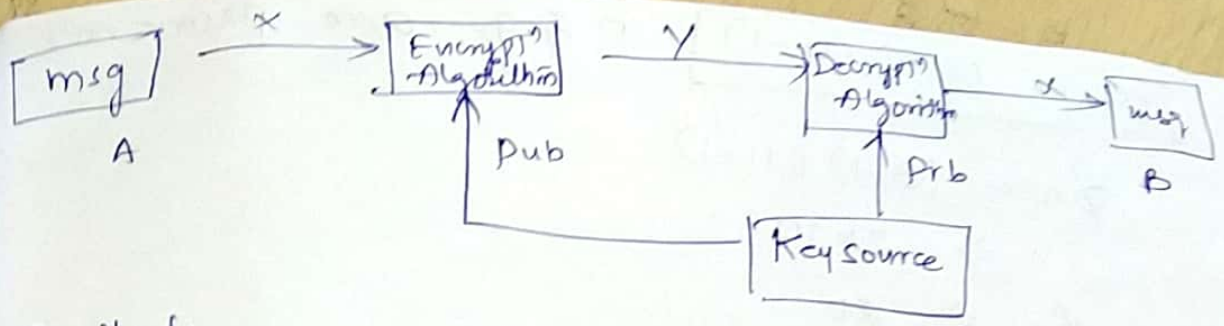
→ The procedure for sending msg to Receiver are Both Sender & Receiver both generates a pair of Keys (2 Keys)

→ Among 2 Keys 1 Key is placed in Public register that means the key access as Public Key. and 1 key is kept as secret that is private that is private Key.

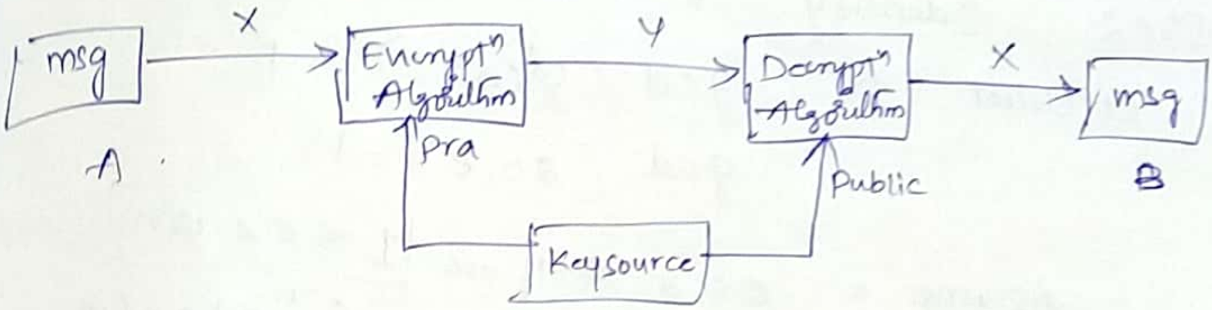
→ Once these 2 Keys are Generated we are performing Encryption and Decryption Algorithm

For Ex:

→ ~~Sender~~ ~~Sends~~ User A wants to send msg ~~to~~ first, it performs Encryptn Algorithm. To perform Encryptn Algorithm we use Public Key of B.



Similarly,



This process is called Authentication.

RSA algorithm :-

(Rivest - Shamir - Adleman) - Author.

→ It is Assymetric Key Algorithm and it is Block Cipher Algorithm.

There are 3 Steps:

1. Key generation
2. Encryption
3. Decryption.

1. Key Generation:

Step 1: Select 2 large ~~numbers~~ prime numbers.
(For Security we select large numbers)

$p = 3 \qquad q = 11$

$n = p \times q$

$n = 3 \times 11 = 33$

$\boxed{\phi(n) = (p-1)(q-1)}$ p & q are prime no's

$\phi n = (3-1)(11-1)$

$\Rightarrow 2 \times 10$

$\phi n \Rightarrow 20$

Step 2: Identify 'e' value.

Condition is $\gcd(\phi(n), e) = 1$

$\gcd(20, e) = 1$.

Assume = $e \Rightarrow 2$ $\Rightarrow$ If we take 2 then '0' $\boxed{1 < e < \phi(n)}$

$2)\,20\,(10$
20
(0)

If we assume = $\boxed{e \Rightarrow 3}$

the GCD of 3 & 20 is ~~Should be~~ "1"

$3)\,20\,(6$
18
$2)\,3\,(1$
2
(1)

Step 3: calculate 'd' value.

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

$d * 3 \bmod 20 = 1$

$\because 7*3 \bmod 20 \Rightarrow$ ~~d~~
$21 \bmod 20 \Rightarrow 1$.

$d = 7$.

To perform Encrypt^n & Decrypt^n.

$m = 5$

$\boxed{C = M^e \bmod n}$

$= 5^3 \bmod 33$

$= 125 \bmod 33$

$33)\,125\,(3$
99
(26)

CT

DEC

PT

ELGA

A

we

3 S

1.

1. S

2. Se

as

3.

4.

CT ⇒ 26

public key = {e, n} = {7, 33}

Private key = {d, n} = {3, 33}

Decryption:

$$\boxed{M = c^d \bmod n}$$

$\Rightarrow 26^7 \bmod 33$

$\Rightarrow 26^5 \times 26^2 \bmod 33$

PT. $\Rightarrow \cancel{26^1 \times 26^2 \times 26^2}$ 5.

ELGAMAL CRYPTOGRAPHY:-

Assymmetric key Cryptography

we use 2 different keys in Assymmetric key.

3 steps: 1. Key Generation

2. Encryption

3. Decryption.

1. Key Generation:

1. Select large prime number (P) = $\boxed{P = 11}$

2. Select a decryption key and it is also called as private key.

$$\boxed{d = 3}$$

3. Select second part of encryption key 'e1'

$e1 = 2$

4. Calculate third part of encrypt^n key 'e2'

$$\boxed{E_2 = e_1^d \bmod P}$$

$\Rightarrow (2)^3 \bmod 11$

$\Rightarrow 8 \bmod 11$

$\Rightarrow 8.$

$\boxed{e2 = 8}$

5. Public calculating publickey = $(E1, E2, P)$

Private Key = d.

$\boxed{\text{pub key} = (2, 8, 11)}$

Keys are Generated.

2. Encryption:-

1. Select random Integer (R)

$R = 4$

C1 & C2 are Cipher Text

2. Calculate $\boxed{C_1 = E_1^r \bmod P}$

$\Rightarrow 2^4 \bmod 11$

$\Rightarrow 16 \bmod 11$

$\boxed{C1 \Rightarrow 5}$.

3. Calculate $C2 \Rightarrow \boxed{(PT \times e2^R) \bmod P}$

PT assume as "7".

$\Rightarrow (7 \times 8^4) \bmod 11$

$\Rightarrow 28672 \bmod 11$

$\Rightarrow 6$

$\boxed{C2 = 6}$

$\boxed{C_1, C_2 = 5, 6}$

3. Decryption:

1. $PT = \left[C_2 \times \left((C_1)^{D} \right)^{-1} \right] \bmod P$

$\Rightarrow \left(6 \times \left((5)^{3} \right)^{-1} \right) \bmod 11$

$\Rightarrow 6 \, (5^3)^{-1} \bmod 11$

$\Rightarrow 6 \, (125)^{-1} \bmod 11$

$\Rightarrow 125 \times x \bmod 11 = 1$

If $x = 3 \Rightarrow 125 \times 3 \bmod 11 = 1$

$\Rightarrow 375 \bmod 11$

$\Rightarrow 1$

$\boxed{\therefore x = 3}$

$\Rightarrow 6 \times x \bmod 11$.

$\Rightarrow 6 \times 3 \bmod 11$

$\Rightarrow 18 \bmod 11$

$\Rightarrow 7$.

$\boxed{PT = 7}$.

DIFFIE-HELLMAN KEY EXCHANGE ALGORITHM:

→ It is not an encryption/ Decryption algorithm.

→ It is Used to exchange Keys between Sender & Receiver.

→ Assymetric Cryptography.

Procedure:-

1) Consider a prime number q

Let $q = 7$.

2) Select α such that $\boxed{\alpha < q}$ and α is primitive root of q.

Such that "α" Should be Less than "q".

Primitive root:-

$\alpha^1 \bmod q$

$\alpha^2 \bmod q$

$\alpha^3 \bmod q$

⋮

$\alpha^{q-1} \bmod q$ Should have values $\{1, 2, 3 \cdots q-1\}$

Ex:- $\alpha = 3$ and $q = 7$ (1, 2, 3, 4, 5, 6)

$3^1 \bmod 7 = 3$

$3^2 \bmod 7 = 2$

$3^3 \bmod 7 = 6$

$3^4 \bmod 7 = 4$

$3^5 \bmod 7 = 5$

$3^6 \bmod 7 = 1$

You α,

3)

Ex:-

4)

5)

Scanned with CamScanner

You should get 1 to 6 less than 7. Because α is primitive root of q

$\therefore$ 3 is primitive root of 7.

3) Assume X_A (Private key of A) and $\boxed{X_A < q}$

Calculate $\boxed{Y_A = \alpha^{X_A} \bmod q}$.

X – Private key

Y – Public key

Ex:- $q = 7$ and $\alpha = 5$.

and let $X_A = 3$

$Y_A = (5)^3 \bmod 7$

$\Rightarrow 125 \bmod 7$

$\Rightarrow 6$

$\boxed{Y_A = 6}$

X_B – Pvt key of B

Y_B = Public key of B.

4) Assume X_B and $\boxed{X_B < q}$

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

$X_A, X_B = (3,4)$

$Y_A, Y_B = (6,2)$

Let $X_B = 4$, $\alpha = 5$

$Y_B = (5)^4 \bmod 7 \Rightarrow 625 \bmod 7$

$\boxed{Y_B = 2}$

5) Calculate secret keys K_1 and K_2.

→ In Diffie Hellman Exchange we exchange the keys. Key exchange is done successfully or not we should check.

K_1 = Person A and K_2 = Person B.

$\boxed{K_1 = (Y_B)^{XA} \bmod q}$ $\boxed{K_2 = (Y_A)^{XB} \bmod q}$

After calculating, if $K_1 = K_2$ then Success

$K_1 = (2)^3 \bmod 7 = 8 \bmod 7 = 1 \Rightarrow \boxed{K_1 = 1}$

$K_2 = (6)^4 \bmod 7 \Rightarrow 1296 \bmod 7 = 1 \Rightarrow \boxed{K_2 = 1}$

$\boxed{K_1 = K_2}$ ∵ Sucess

Key exchanged Successfully.

→ Both of them (K_1 & K_2) Should be Equal. If, it is Equal then Key Exchange done Successfully.

→ In this, Sender and Receiver the Key exchange is done.

→ By Calculating the values of <u>K1 & K2</u> we are checking that Key exchange is done Successfully or not.

→ How, we find the values K_1 & K_2. By Using X_A, X_B, Y_A & Y_B. By using Both Private & Public Key.