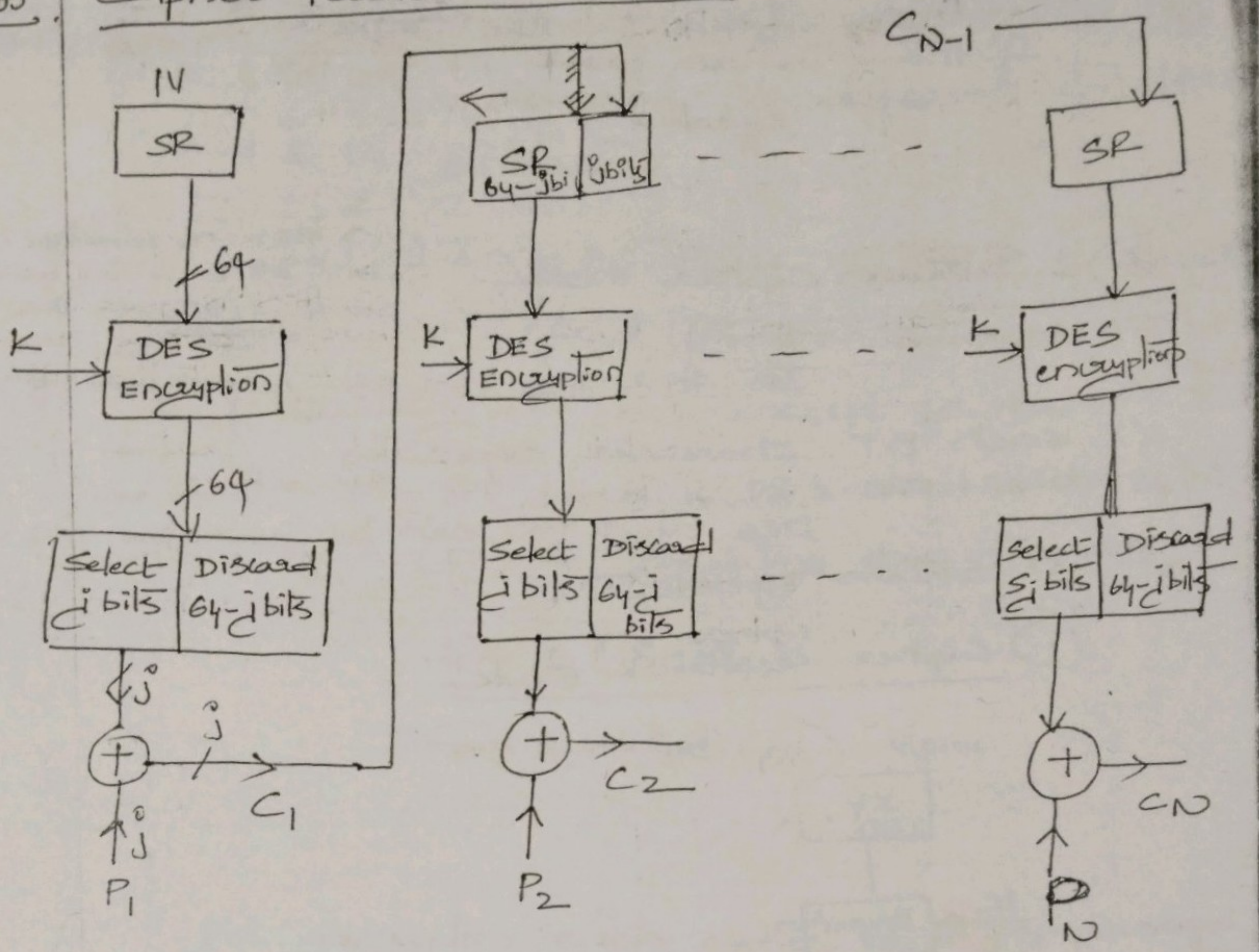
18/8/05

Cipher feedback mode



The stream cipher eliminates the need to pad a message to be an integral no. of blocks.

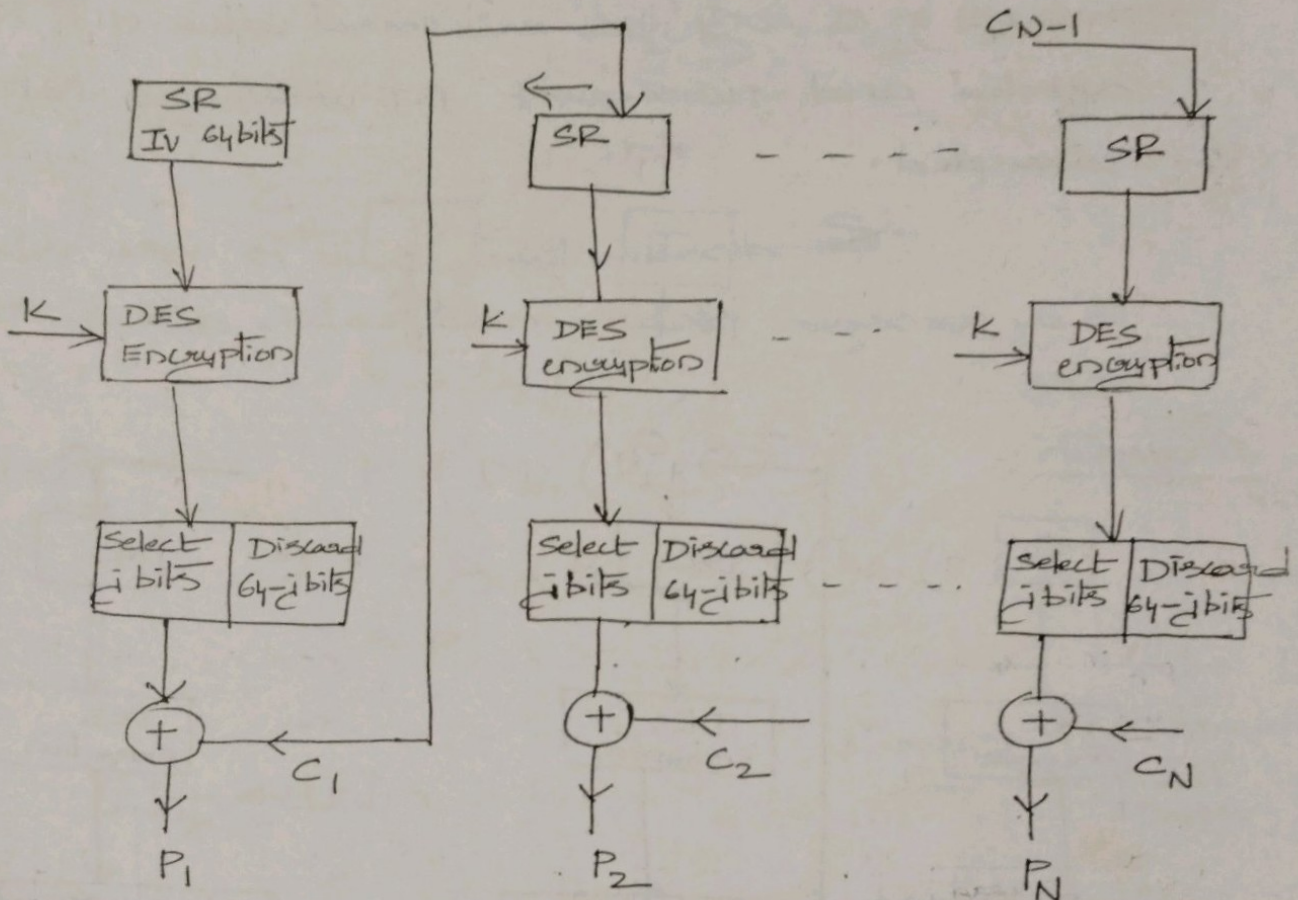
In CFB method it is assumed that the unit of transmission is 'j' bits where j is 8 bit value. The i/p to the encryption function is 64 bit shift Register (SR) that is initially set to some Initialization vector (IV). The left most (MSB) j bits of encryption function are X-ORed with 1st unit of P.T. P_1 to produce the 1st unit of C.T. C_1.

i.e. $C_1 = P_1 \oplus S_j(E_K(IV))$

The content of SR are shifted left by j bits and C_1 is placed in the right most j bits of SR. This process continues all the P.T. units have been encrypted.

For decryption the same scheme is used except that the received C.T. unit is X-OR ed with o/p of encryption function to produce the P.T.

In decryption, the encryption function is only used, only the difference is for X-OR operation & C.T. is given as i/P & obtaining the P.T.



Decryption

The drawback of this method is if any one C.T. block is currupted during transmission then all the successive P.T. blocks also be currupted. because the C.T. is also taken as feed back in the SR.

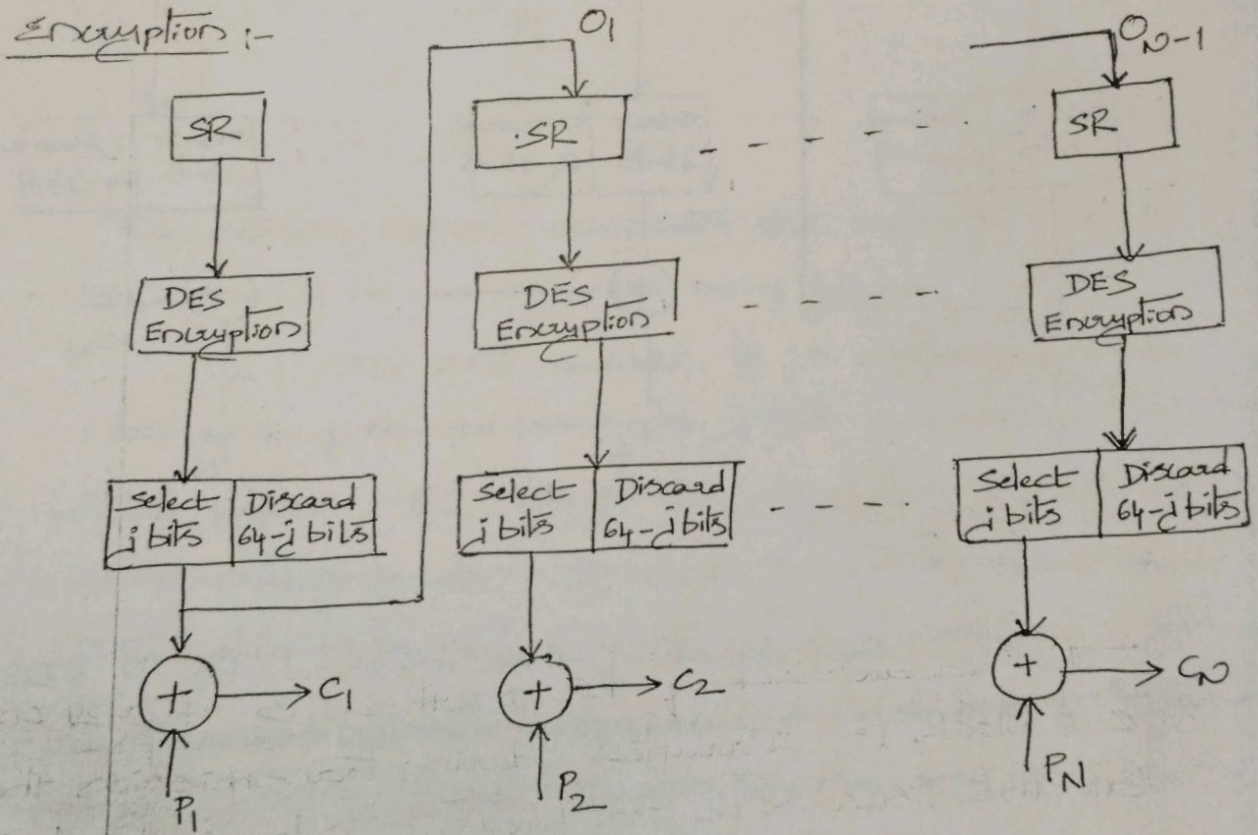
O/P feedback mode :- (OFB).

It is similar to the structure of CFB but the o/p of encryption function that is feedback to the SR in OFB.

The advantage is bit error in transmission do not propagate. For example if a bit error occurs in C_1, only the recovered value of P_1 is affected and subsequent P.T. units are not corrupted.

The disadvantage is it is more vulnerable to a message stream modification attack than CFB.

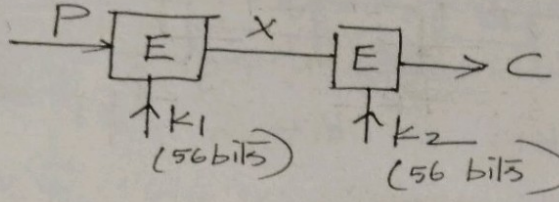
Encryption :-



V.V.

Q:- Explain double DES algorithm with attacks.

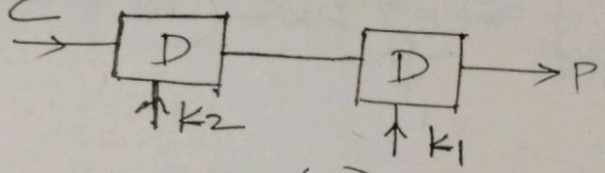
Encryption



$$X = E_{K_1}(P)$$

$$C = E_{K_2}(X) = E_{K_2}(E_{K_1}(P))$$

Decryption

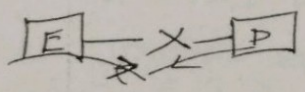


$$X = D_{K_2}(C);$$

$$P = D_{K_1}(D_{K_2}(C))$$

$$\Rightarrow P = D_{K_1}(D_{K_2}(E_{K_2}(E_{K_1}(P))))$$

$$= P$$



Known P.T	Corresponding C.T
1111---1	101010---

$K_{10} \rightarrow X_{10}$

$K_{11} \rightarrow X_{11}$

:

$K_{12^{56}} \rightarrow X_{12^{56}}$

$Y_{10} \leftarrow K_{20} \xleftarrow{D} C$

$C_{12}^{y} - K_{21} \xleftarrow{D} C$

(K_{10} should match with any one of X..)

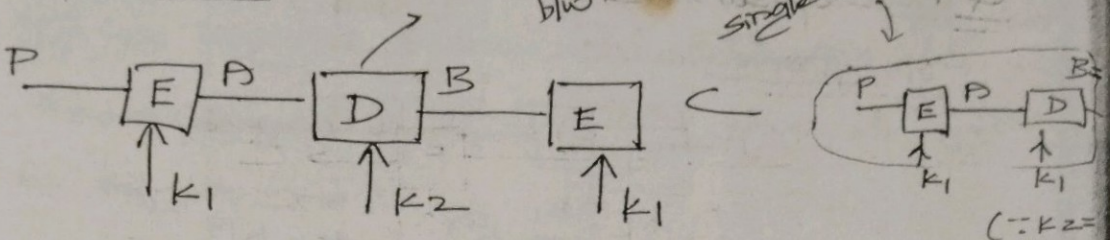
K_{10} → key of all zeros

K_{11} → key of all 1's.

K_{1i}, K_{2j} → master keys.

Triple DES

To have compatability b/w Triple DES & single DES

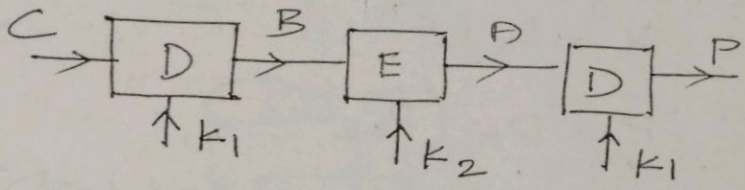


$$A = E_{K_1}(P)$$

$$B = D_{K_2}(A)$$

$$C = E_{K_1}(B)$$

$$= E_{K_1}\left(D_{K_2}\left(E_{K_1}(P)\right)\right)$$



$$B = D_{K_1}(C)$$

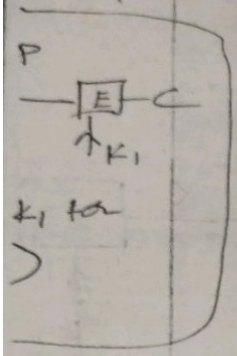
$$A = E_{K_2}(B)$$

$$P = D_{K_1}(A)$$

$$= D_{K_1}\left(E_{K_2}\left(D_{K_1}(C)\right)\right)$$

$$= D_{K_1}\left(E_{K_2}\left(D_{K_1}\left(E_{K_1}\left(D_{K_2}\left(E_{K_1}(P)\right)\right)\right)\right)\right)$$

20/8/05

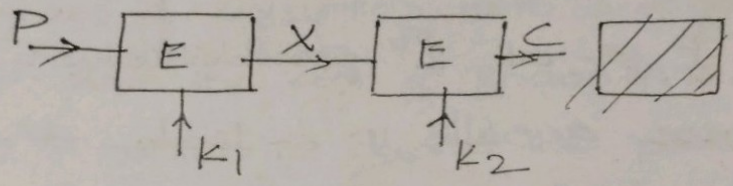


Double DES algorithm :-

The simplest form of multiple encryption has 2 encryption stages and 2 keys.

For a given P.T. 'P' and 2 encryption keys K_1 & K_2, C.T. 'C' is generated as,

$$C = E_{K_2}(E_{K_1}(P))$$



Decryption requires that the keys should be applied in the reverse order.

$$P = D_{K_1}(D_{K_2}(C))$$

Thus in double DES performing encryption 2 times with 2 different keys K_1 and K_2, which involves a key length of 56×2 (= 112 bits), resulting in increasing cryptographic strength. But there are certain weaknesses of double DES.

1) Reduction to a single stage :-

It is possible to find a key K_3 such that

$$E_{K_2}(E_{K_1}(P)) = E_{K_3}(P).$$

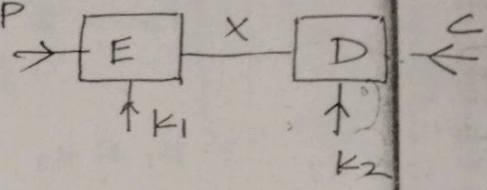
Thus the double encryption will be useless because the result would be equivalent to single encryption with a single 56 bit key.

2) Meet in the middle attack :-

It is based on the observation that,

$$C = E_{K_2}(E_{K_1}(P))$$

then, $\boxed{X = E_{K_1}(P) = D_{K_2}(C)}$



Given a known pair (P, C), the attack proceeds as,

(i) encrypt P for all 2^{56} possible values of K_1. Store these results in a table and then sort the table by the values of X.

(ii) Decrypt C by using all 2^{56} possible values of K_2. As each decryption is produced, check the result against the table for a match.

If a match occurs then test the 2 resulting keys against another P.T - C.T. pair. If the 2 keys produce the correct C.T., accept them as the correct keys.

Thus because of X we are able to calculate the keys K_1 and K_2 known as, meet in the middle attack.

combinations of P.T

combinations of 2 keys combinedly

$2^{64} - 2$

$1 - ?$

For double DES, the average produces an false alarms on the first (P,C) pair

$$2^{48}\left(2^{112}/2^{64}\right)$$

(2 keys; P.T.; wrong keys)

In meet in the middle attack the 2 blocks of P.T, C.T. is considered, therefore the generation of false alarm for 2nd pair is reduced to $2^{48}/2^{64} = 2^{-16}$. Therefore the

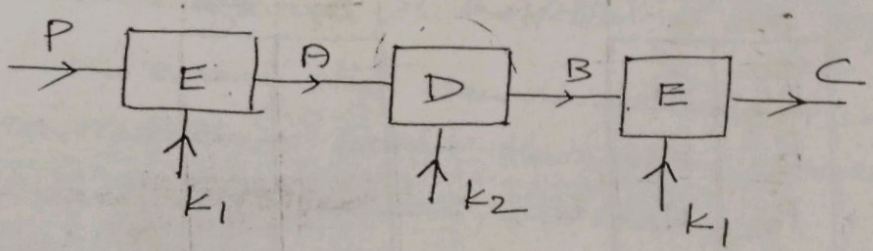
probability of calculating correct keys is, $1-2^{-16}$.
Thus meet in the middle attack has more probability that we can determine the keys K_1 and K_2.

⇒ Tripple DES algorithm with 2 keys

To overcome the meet in the middle attack 3 stages of encryption with 3 different keys can be used. This raises the cost of known P.T. attack increases to 2^{112} which is beyond the practicle way of attacking. But the drawback is requiring a key length of $56 \times 3 = 168$ bits.

Therefore as an alternative Tripple DES proposed by Tuchman is performing tripple-encryption with only 2 keys as encrypt-decrypt-encrypt (E-D-E sequence) as,

$$C = E_{K_1}(D_{K_2}(E_{K_1}(P)))$$

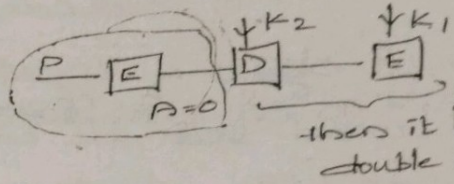


There is no cryptographic significance to the pair use of decryption in the middle stage. It's only advantage is that it allows the users of tripple DES to decrypt data encrypted by users of single DES.

The decryption of Tripple DES is performed as

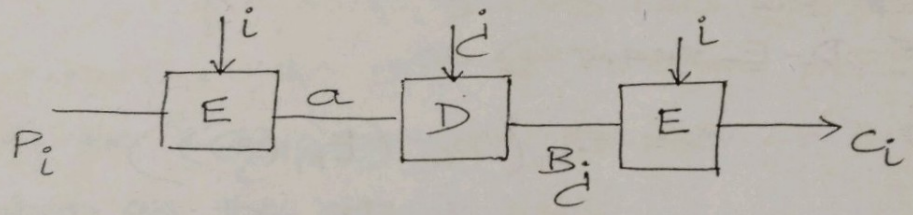
$$C = D_{K_1}\left(E_{K_2}\left(D_{K_1}(P)\right)\right)$$

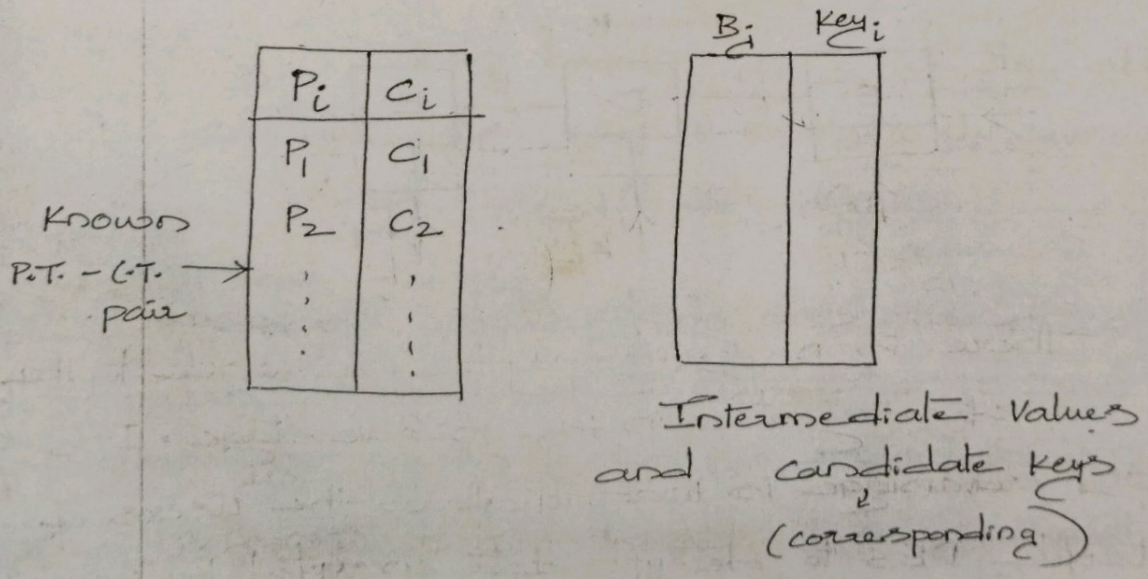
Attacks: (i) There is no practicle crypt analytic attack on Tripple DES. The proposal came from Hellman as finding the P.T. values that produce a 1st intermediate value A=0 and then using meet in the middle attack to determine K_1 & K_2.



(ii) A known P.T. attack is also possible by observing A & C values then the problem reduces to that of an attack on double DES.

The attack proceeds as follows.





The following steps are used to attack on Tripple DES.

(i) Obtain 'n' ((P,C) pairs). This is the known P.T., as shown in the table the P.T. & C.T. are stored.

(ii) Select an arbitrary value 'a' as 'A' in Tripple DES Create a second table for each of 2^{56} possible keys calculate the P.T. value P_i that produces 'a' as:

$$P_i = D_i(a). \quad (\text{or } a = E_i(P))$$

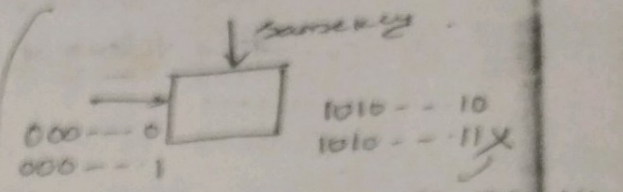
For each P_i that matches an entry in table-1, create an entry in table-2 consisting of K_1 value and the value of B that is produced for the (P,C) pair from table-1 and assuming that value of K_1.

Thus some entries are made in table-2 based on the values of B.

(iii) There are set of values of K_1 in table-2 and it is possible to search for a value of K_2. For each of 2^{56} possible keys, $K_2 = i$ calculate the second intermediate value for selected 'a' as, $B_i = D_i(a)$

At each step search in table-2 and if any match is found then the corresponding keys are identified as K_1 and K_2.

⇒ Avalanche effect :-



∴ even though there is one bit of change in P.T. there should be large change in C.T. (for same keys)

- The desirable property of encryption algorithm is that even for small changes in either P.T. or key it should produce significant (large) changes in C.T.

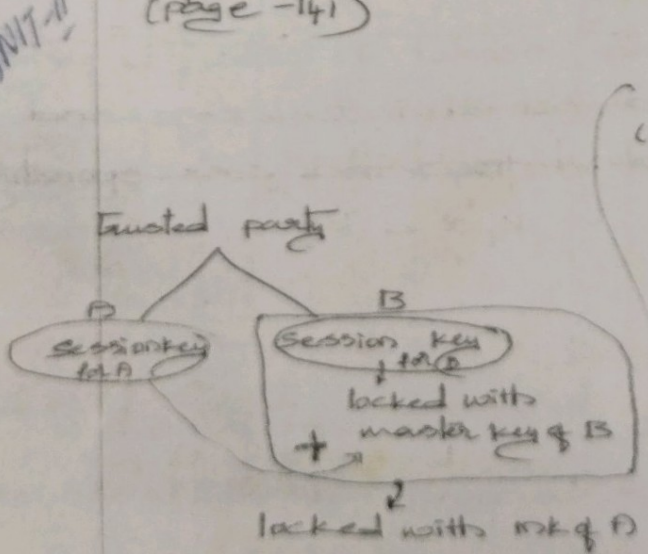
It means change in one bit of P.T. or one bit of key should produce ~~many~~ a change in many bits of the C.T.

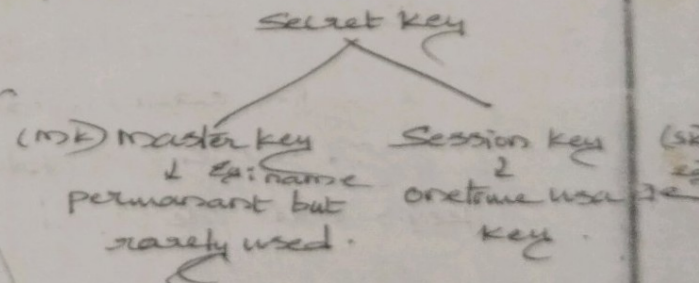
DES algorithm follows more Avalanche effect, for one bit change in P.T. causes 34 bits change in C.T. at the end of 16 rounds, for same key. It also follows more Avalanche effect for changes in key. There are 35 bits change in C.T. even for one bit change in the key after 16 rounds.

(short) ⇒ **Key distribution :-** → is needed for decryption (by the receiver)

(page -141)

UNIT-II





perform encryption with session key & lock it with master key

ie. mk are useful to distribute SKs

→ we are going to select a trusted party which distributes the session key

(key authority)

For conventional encryptions the 2 parties to an exchange must possess the same key and that key must be protected from access by others. The strength of cryptographic systems depends on key distribution techniques i.e. means of delivering a key to 2 parties who wants to exchange the data without allowing others to see the key.

For 2 parties A & B key distribution can be achieved in no. of ways.

1) A key can be selected by user 'A' & physically delivered to B.

2) A third party can select the key & physically delivered it to users A & B.

3) If A and B are having old key, then the new key can be transmitted by encrypting with old key.

4) If A and B has an encrypted connection to a 3rd party C, C can deliver a key on the encrypted links to A & B.

25/8/05

1) Program for DES key generation

```
class DESSubkeyTest
{
  public static void main(String args[])
  {
    try {
        byte[]   thekey = getTestkey();
      byte[][]   subkeys = getSubkeys(thekey);
      boolean  ok = validateSubkeys(subkeys);
      System.out.println("DES result:"+ok);
    }catch(Exception e)
     {
       e.printStackTrace();
     }
  }
```

```
static final int[]  PC1 = {57,49,41,33,25,17,9
static final int[] PC2 = {14,17,11,--};
static final int[] SHIFTS = {1,1,2,2,2,2,2,2,1,2,2,2,2,2,2};

private static byte[][] getSubkeys(byte[] thekey)
  throws Exception
  {
    printBytes(thekey, "Inputkey");
    int activekeysize = PC1.length;   (56)
    int numofsubkeys = SHIFTS.length;
    byte[] activekey = SelectBits(thekey, PC1);
    printBytes(activekey, "After
```

```
int halfkeysize = activekeysize/2;
byte[] c = selectBits(activekey, 0, halfkeysize);
byte[] d = selectBits(activekey, halfkeysize, halfkeysize);
byte[][] subkeys = new byte[numofsubkeys][];
for(int k=0; k<numofsubkeys; k++)
{
    c = rotateleft(c, halfkeysize, SHIFTS[k]);
    d = rotateleft(d, halfkeysize, SHIFTS[k]);
    byte[] cd = concatenateBits(c, halfkeysize
                                d, halfkeysize);
    printbytes(cd, "subkey #"+(k+1)+"after
                    shifting");
    subkeys[k] = selectBits(cd, PC2);
    PrintBytes(subkeys[k], "subkey #"+(k+1)+
                "after PC2");
}
return subkeys;
}
private static byte[] rotateleft(byte[] in, int
                                 len, int step)
{
    int numofBytes = (len-1)/8 + 1;
    byte[] out = new byte[numofBytes];
    for(int i=0; i<len; i++)
    {
        int val = getBit(in, (i+step)%len);
        setBit(out, i, val);
```

c = 1st 28 bits of key
d = next 28 bits of key
SHIFTS[k] -> indicates step 1 or 2 (1 bit rotation / 2 bit rotation)
step = 1 or 2
halfkeysize -> for length
halfkeysize -> for position
in -> i/p
len, int step -> iteration number
numofBytes = (len-1)/8 + 1 -> allocating memory for numofBytes number
(i+step)%len -> from here we get pos
out -> to handle the data into out
len -> length of the array (key word)

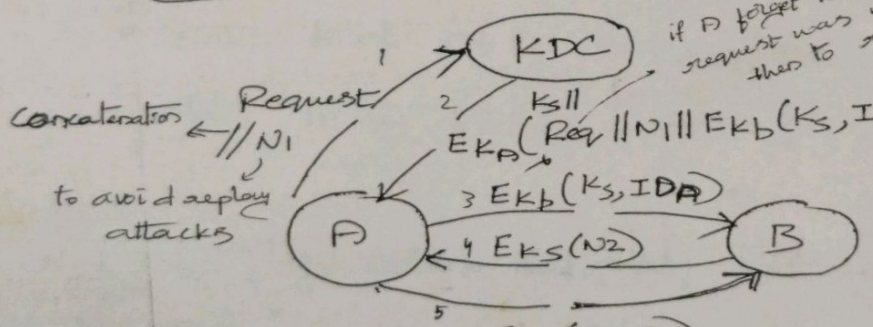
⇒ Key distribution :-

The options (1.) & (2.) uses manual delivery of key.

The option (3.) is the users will be having old key and whenever they want to exchange the key they will encrypt the new key with old key.

The 4th technique is key distribution centre based on use of hierarchy of keys. There are 2 levels of keys are used. The communication b/w end systems is encrypted by using a temporary key known as session key. It is used for the duration of logical connection & discarded after the end of session. The session keys are transmitted in encrypted form by using master key that is shared by key distribution centre (KDC) & end systems.

Key Distribution Centre scenario :-



(identity of A + message + SK) → locked with mk of B i.e. $E_{K_b}(K_s, ID_A)$.

→ A requests KDC for SK that is to be used b/w A & B. (2 SK's (same) are required i.e. one for A & one for B

1, 3, 3 for key distribution
4, 5 for authentication

2) Kerberos

Key distribution by using KDC is having the following steps.

Assume that user A wants to establish a logical connection with B and requires one time session key (SK) to protect the data Transmission. A has a secret key K_A, known as master key (of A) with KDC. Similarly B shares the master key K_B with KDC.

Step1 :- A issues a request to KDC for a session key the message includes identity of A & B (i.e. I am A & I want key for B) & a unique identifier No, called Nonce. (Time stamp, Counter & random number)

Step2 :- KDC response with a message encrypted by using K_A. Thus A receives the message which contains one time session key K_s to be used for the session. The original request message so that A can verify the reply belongs to which message. In addition the message also include 2 items for B, Encrypted by using K_B. It contains one time session key K_s & also identifier of A.

Step3 :- 'A' stores SK & sends the message intended for B received from KDC to user B. Now B decrypts & learns the SK & identifies the communicating party as A.

Thus the SK is delivered to A & B. But there are 2 additional steps for identifying

A&B each other.

Step4:- with SK B sends nonce N_2 to A.

Step5:- A response by sending $f(N_2)$, where f is a function that performs some transformations on N_2, by encrypting with K_s.

3/9/05

Random number generation :-

Random numbers plays an important role in the use of encryption for various n/w security applications.

A no. of n/w security algorithms based on cryptography makes use of random numbers.

For example,

(i) Reciprocal authentication schemes; to prevent replay attacks random numbers are used as nonce.

(ii) session key generation which is done by key-distribution centre or by one of the principles (see xtra)

(iii) generation of keys for public key encryption algorithms.

---

random numbers

Natural — pseudo

pseudo → man or computer make random numbers

→ random numbers
→ the relation b/w series of numbers is non deterministic

→ random numbers are used to generate session key because if intruder catches one key he can not expect next key.

⇒ Pseudo random number generation :-

- The technique generated by Lehmer which is also known as "Linear congruential method."
- The algorithm is parameterised by 4 numbers

(i) m : modulus $(m > 0)$

(ii) a : multiplier $(0 \leq a < m)$

(iii) c : increment $(0 \leq c < m)$

(iv) x_0 : the starting value (or) seed value.
$(0 \leq x_0 < m)$

- The sequence of random numbers (X_n) is obtained by using the following iterative equation

$$X_{n+1} = (ax_n + c) \bmod m.$$

Example :- Consider $a = 7$, $c = 0$, $m = 32$ and $x_0 = 1$. Then the random numbers are generated as,

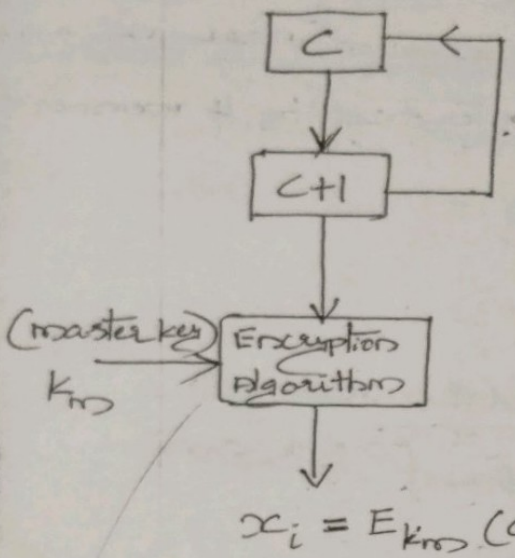
$$x_1 = (7 \times 1 + 0) \bmod 32$$
$$= 7$$

$$x_2 = (7 \times 7 + 0) \bmod 32 \quad (\because x_2 = (ax_1 + c) \bmod m)$$
$$= 17.$$

$$x_3 = (17 \times 7 + 0) \bmod 32$$
$$= 23.$$

$$x_4 = (23 \times 7 + 0) \bmod 32$$
$$= 1.$$

⇒ Cryptographically generating random numbers :-



counter	random no.
1	50
2	182
3	20
4	35
:	:

For cryptographic applications, by taking advantage of encryption logic to produce random numbers. A no. of means have been used.

(1) Cyclic encryption :-

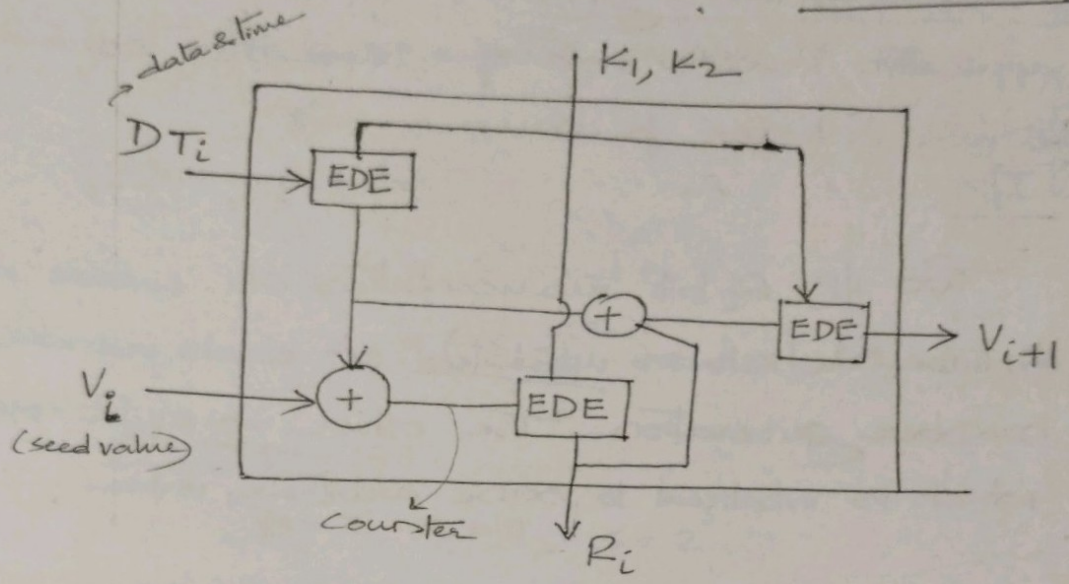
As shown in the diagram the procedure is used to generate session keys from master key. A counter (C) with N period provides i/p to the encryption logic.

The encryption logic generates the C.T. based on the counter value and the master key. After each key is produced the counter is incremented by one. Thus random numbers are produced by this method for entire period (no. of bits or length of counter)

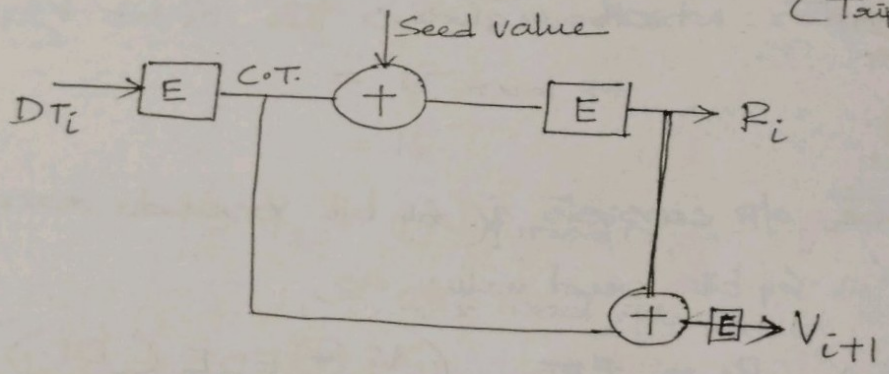
numbers :-

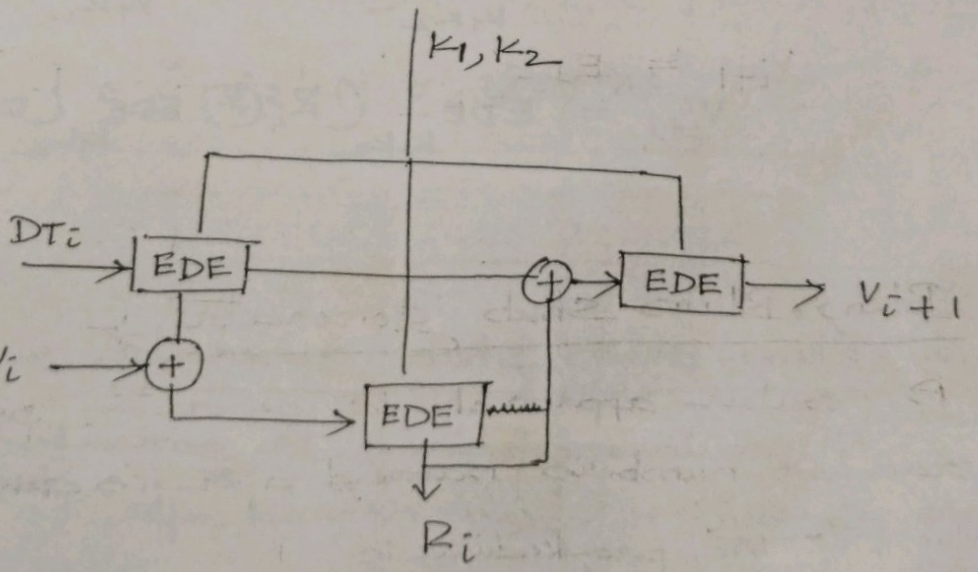
(2) ANSI X 9.17 pseudo random number generator :-

random no.
50
182
20
35



EDE → Encryption - Decryption - Encryption (Tripple DES).





One of the strongest pseudo random number generator commonly used in financial security application makes use of triple DES, contains,

1) I/P

Two i/P, 64 bit representation of current date & time, which is updated on each random number generation. The other is 64 bit seed value, which is initialized to some arbitrary value.

2) Keys

The generator makes use of Tripple DES encryption which requires two 56 bit keys K_1 & K_2

3) O/P

The o/p consists of 64 bit pseudo random number & a 64 bit seed value as,

$$R_i = EDE_{K_1K_2}\left(V_i \oplus EDE_{K_1K_2}(DT_i)\right)$$

$$V_{i+1} = EDE_{K_1K_2}\left(R_i \oplus EDE_{K_1K_2}(DT_i)\right)$$

3) Blum-Blum shub generator :-

A popular approach for generating pseudo random numbers named after it's developers.

The procedure is,

Step1:- Select 2 large prime numbers p and q with both having remainder 3 when they are divided with 4. i.e. $P \equiv q$

Step 2 :- Calculate product $n = p \times q$.

Step 3 : Select a random number 's' such that which is relative prime to n.

Then sequence of random numbers are generated as,

$$X_0 = s^2 \bmod n$$

$$X_i = (X_{i-1})^2 \bmod n \quad \text{for } i = 1 \text{ to } \infty$$

Example :-

$p = 7,\ q = 11,\ s = 2$

Then,

$X_0 = 4 \bmod 77$

$= 4$

$X_1 = (X_0)^2 \bmod n$

$= 4^2 \bmod 77$

$= 16.$

$X_2 = (X_1)^2 \bmod n$

$= 256 \bmod 77$

$= 25.$

$n = 192649 = 383 \times 503$