16 January 2024 09:40

* Cryptology → Cyrptography + Cryptoanalysis
  - Cryptology → Overall umbrella
  - Cyrptography → Encrypt / Decrypt
  - Cryptoanalysis → Find weakness / vulnerabilities

* NIST → Standardizes cryptographic Algo.

* Example →

ATM1 → PIN1 + X = Y_1
- PIN1 → Plaintext
- X → Secret key
- Y_1 → write (Ciphertext)
- + → Encryption function

Similarly

ATM 2 → PIN2 + X = Y_2

[Encryption : $E(P, K) = C$
Decryption : $D(C, K) = P$]

*

Crypto
- Symmetric Key Crypto → One Secret for both encryption & decryption
- Public Key Crypto → Two Keys
  - Public (for encryption)
  - Secret (for decryption)

$E(P, e_k) = C$

$D(C, d_k) = P$

* Functionalities of Cryptography

1) Confidentiality (Secrecy)
2) Integrity (Anti-tampering)
3) Authentication (Verify that user & msg is authenticate "coming from you only")
4) Non repudiation (you cannot deny that this message is not coming from you)

Domain

Encryption: $P \times Enc \longrightarrow C$

Decryption: $C \times Dec \longrightarrow P$

* Caesar Cipher

Shift every alphabet by three characters.

A → 0
B → 1
⋮
Z → 25

Agreed no = 3 (fixed in caesar)

So,

$E(x, 3) = (x+3) \% 26$

$D(c, 3) = (c+26-3) \% 26$

(In modulus 26 system,

$Z \to 25$ ∫ in caesar

(In modulus 26 system, 23 is additive inverse of 3)

* Function –

$f: A \to B$ is a relation blw A & B if $a, b \in A$ and $a = b$ then $f(a) = f(b)$

↓

$[R \subseteq A \times B]$

- One-to-one → $f(a) = f(b) \Rightarrow a = b$
- Onto → $f: A \to B$ then $\forall\, b \in B \ \exists\, a \in A$ st $f(a) = b$
- Bijective → Both one-one & onto
- Permutation → Let π be a permutation on set S then $\pi : S \to S$ is a bijection from S to S.

$$\pi : \begin{pmatrix} 1 & 2 & 3 & 4 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ 2 & 3 & 1 & 4 \end{pmatrix}$$

→ One-way: You can find but limited due to computation.

Example

$N = p \times q$ → this computation is easy

↑ Find ↑ Given

Now, if asked given N → find p & q → hard computation

* Substitution Box / S-Box :–

$S: A \to B$ with $|B| \leq |A|$

$S: \{1,2,3,4\} \to \{1,2,3\}$ can't be one-one

* Transposition Cipher :–

$M = m_1 m_2 \ldots m_t$

c : permutation on t elements

Encryption] $C: m_{c(1)} m_{c(2)} \ldots m_{c(t)}$

Decryption] $M = C_{c^{-1}(1)} C_{c^{-1}(2)} \ldots C_{c^{-1}(t)}$

$m_1 m_2 \ldots m_6$

M : CAESAR

↓

C : RSCEAA

e : (Secret key)

	1	2	3	4	5	6
↓	↓	↓	↓	↓	↓	↓
	6	4	1	3	5	2

Example

M : CAESAR

$d = e^{-1}$:

	1	2	3	4	5	6
↓	↓	↓	↓	↓	↓	↓
	3	6	4	2	5	1

* Substitution Cipher–

$\mathcal{A} = \{A, B, C \ldots Z\}$

e : substitution from $\mathcal{A}$ to $\mathcal{A}$

$\mathcal{A}$ = {A, B, C ... Z}

e : substitution from $\mathcal{A}$ to $\mathcal{A}$

C = e(m1) e(m2) _ ... e(mt)

e(A) = Z, e(B) = D, e(C) = A } Example

So, ABC → ZDA

* Affine Cipher –

A	B	C	- - - -	Z
↓	↓	↓		↓
0	1	2	- - - -	25

$\mathcal{A}$ = set of alphabets

$\mathbb{Z}_{26}$ = {0, ... 25}

$\mathcal{A} \rightarrow \mathbb{Z}_{26}$

k = secret key = $(a, b) \in \mathbb{Z}_{26} \times \mathbb{Z}_{26}$

Encryption: $e(x, k) = (ax + b) \bmod 26 = C$ → GCD b/w a & 26 = 1

x : plaintext

$x \in \mathbb{Z}_{26}$

Decryption :

* Playfair Cipher –

Secret key = PLAYFAIREXAMPLE (I=J)

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

M : HIDE (even length)

HI DE

B M O D

Eg1

M : BALL (if repitition)

BA LX LX

Add extra X

Eg2

M : CSE (odd length)

CS EX

Eg3

Same process for decryption.

Problem : HJDE → BMOD → HIDE now

check HIDE tha ya HJDE

2 possible combinations.

Total No. of Bijective mapping = 26 !

And total no. of possible k (a, b) = 12 × 26

* Hill Cipher –

And total no. of possible $k(a,b) = 12 \times 26$

★ Hill Cipher –

Secret key → $A = (a_{ij})_{n \times n}$ → Invertible

$M = m_1 m_2 \dots m_n$

$C = A \cdot M$

$C_i = \sum_{j=1}^{n} a_{ij} m_j$

$M = A^{-1} C$ → Decryption

★ Symmetric key

→ Block Size

Block Cipher

Stream Cipher

$M = m_0 || m_1 || \dots || m_n$

Encryption : $C = Enc(m_0, k) || Enc(m_1, k) \dots Enc(m_n, k)$

$C = C_0 || C_1 || \dots || C_n$

Decryption : Same – Block by Block

$len(M) = n$, $len(m_i) = l$

⇒ For odd length messages –

$M = \underset{n}{m_0} || \underset{n}{m_1} || \underset{l}{m_2}$

length $= 2n + l$, $l < n$

add zeros to last so that equal distribution happens.

$l + r = n$ → added 0s.

$C_0 = Enc(m_0, k)$, $C_1 = Enc(m_1, k)$

$C_2 = Enc(m_2 || 0^{n-l}, k)$

Now, $C = C_0 || C_1 || C_2$

⇒ This was ECB mode of operation

Electronic Control Book

⇒ This mode is not secure bcs –

1) If two blocks are same, Encrypted blocks are same → revealing info.
2) We need to share how many extra 0s we added in odd length.

★ Product Cipher – (type of Block Cipher)

Combines 2 or more transformation in a manner

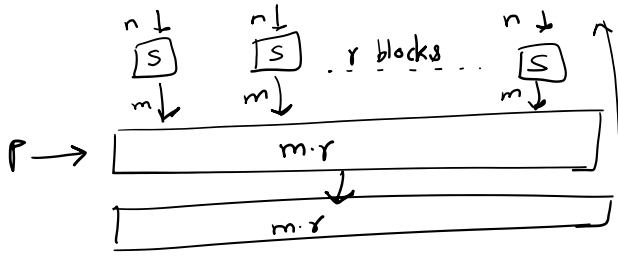
(Any block cipher in any one or faisth

Combines 2 or more transformation in a manner intending that the resulting cipher is more secure than the individual components.

{ Any block cipher lands in any one type → SPN or Feistel

Substitution - Permutation Network (SPN)

⇒ Encryption: $S: \{0,1\}^n \longrightarrow \{0,1\}^m$; $P: \{0,1\}^{mr} \longrightarrow \{0,1\}^{mr}$

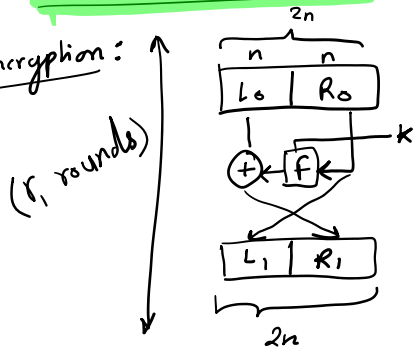


r, number of times (r, rounds)

⇒ For decryption: Simply back-track usually S-Box are invertible

Feistel Network -

⇒ Encryption:



$f: \{0,1\}^n \times \{0,1\}^k \longrightarrow \{0,1\}^n$

$L_1 = R_0$

$R_1 = L_0 \oplus f(R_0, k)$ ← XOR

⇒ Decryption: $L_0 = R_1 \oplus f(\underbrace{L_1}_{R_0}, k)$

$R_0 = L_1$

If odd length : simply pad & add zeros to make even.

★ Iterated Block Cipher -

An iterated block cipher is a block cipher involving sequential repitition of an internal fn called as round fn. The parameters include the no. of rounds r, the block size n, & the bit size k of the input key K from which r subkeys k_i (round keys) are derived.

$F(K_1, P) \longrightarrow C_1 \longrightarrow F(K_2, C_1) \longrightarrow C \rightarrow$ Ciphertext

2 round IBC → F: round fn; K_2: round key

$K \rightarrow G(K) \rightarrow K_1, K_2$

K: master key; G: Key scheduling algo

Key scheduling algo

$C \rightarrow F^{-1}(C, K_2) \rightarrow C_1 \rightarrow F^{-1}(C_1, K_1) \rightarrow P$

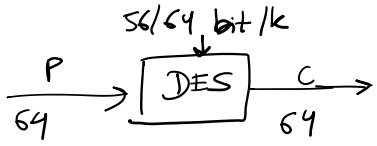
↳ F^{-1} (inverse) means effect reversal not neccessary f inverse.

* DES (Data Encryption Standard) :-

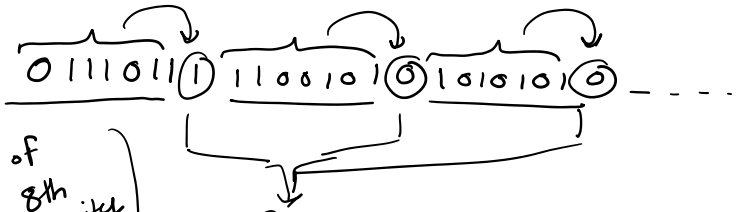
Designed by IBM.

Secret key = 64 bit

Plaintext Block size = 64 bit



64 bit key (8 for parity check)



If you know 56 bits then you can determine other 8 bits.

- DES is based on Feistel Network
- It is an IBC
- Number of rounds = 16

Key scheduling Algo (64 bit key)

$\rightarrow K_1, K_2, \ldots K_{16}$

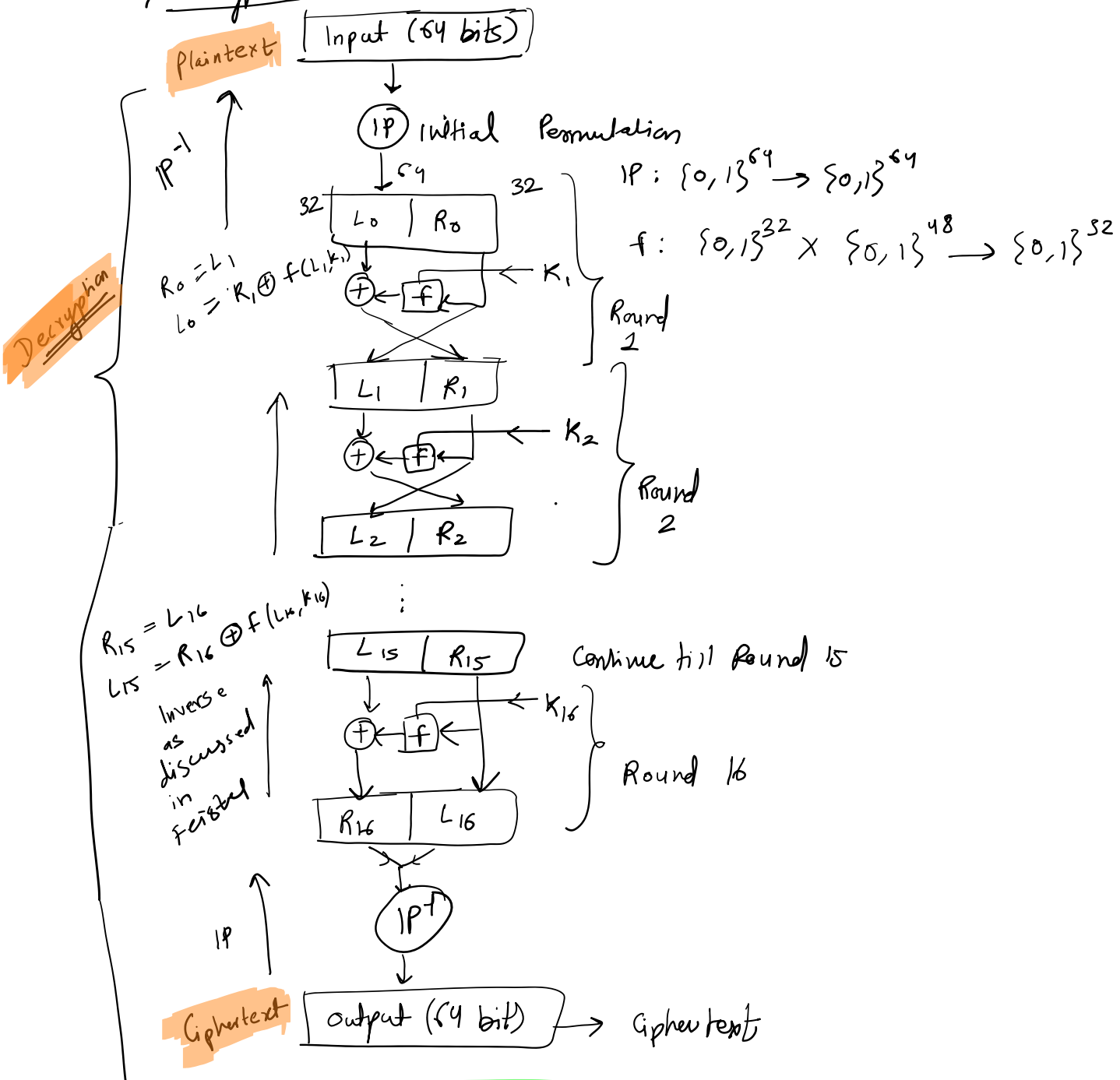
K_i s are of 48 bits.

$\underbrace{K}_{64} \rightarrow K_1, K_2, \ldots K_{16}$ (K_1, K_2: 48 each)

⇒ Encryption :

48 48

⇒ Encryption :



$IP : \{0,1\}^{64} \rightarrow \{0,1\}^{64}$

$f : \{0,1\}^{32} \times \{0,1\}^{48} \rightarrow \{0,1\}^{32}$

* **Internal Permutation (IP) —**

$IP \{0,1\}^{64} \rightarrow IP \{0,1\}^{64}$

$IP(m_1 m_2 \dots m_{64}) = m_{58}\, m_{50} m_{42} \dots m_7$

IP : $\begin{pmatrix} 1 & 2 & 3 & \dots & 64 \\ \downarrow\uparrow & \downarrow\uparrow & \downarrow\uparrow & & \downarrow\uparrow \\ 58 & 50 & 42 & & 7 \end{pmatrix}$

IP $\quad$ IP^{-1} (mapping changed in matrix)

* **f function algorithm —**

$f : \{0,1\}^{32} \times \{0,1\}^{48} \rightarrow \{0,1\}^{32}$

$f: \{0,1\}^{32} \times \{0,1\}^{48} \longrightarrow \{0,1\}$

$f(R_i, K_i) = X_{i+1}$

$f(R_i, K_i) = P\left[S\left(E(R_i) \oplus K_i\right)\right]$

P: just does permutation 32 to 32

S: S-box 48 to 32

E: expansion fn 32 to 48

* $E: \{0,1\}^{32} \longrightarrow \{0,1\}^{48}$

$E(x_1 x_2 \ldots x_{32}) = y_1 y_2 \ldots y_{48}$

E:

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

$E(x_1 x_2 \ldots x_{32}) =$

$= (x_{32}\, x_1\, x_2\, x_3\, x_4\, x_5$
$x_4\, x_5 \ldots$
$\ldots x_{32}\, x_1)$

* S-box of f :-

$S: \{0,1\}^{48} \longrightarrow \{0,1\}^{32}$

$S(x) = Y$

$x = B_1 B_2 B_3 B_4 B_5 B_6 B_7 B_8$

length of B_i is 6-bit

$S_1\ S_2\ S_3\ S_4\ S_5\ S_6\ S_7\ S_8$

$S_i: \{0,1\}^6 \longrightarrow \{0,1\}^4$ for all $i = 1,2 \ldots 8$

$S_i(B_i) = C_i$

(B_i: 6 bit, C_i: 4 bit)

$\therefore S(x) = \left[S_1(B_1)\ S_2(B_2) \ldots S_8(B_8)\right]$

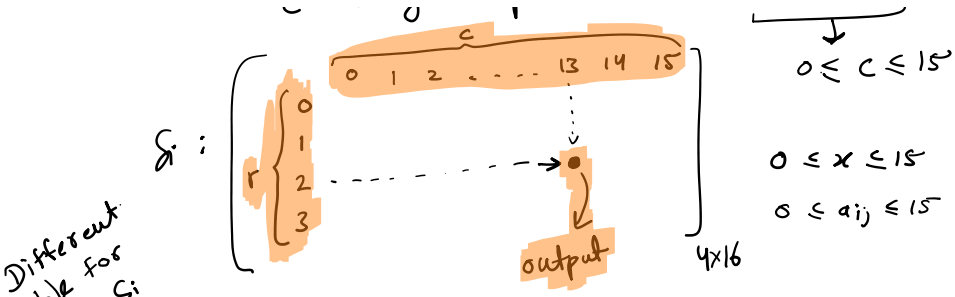
$B_i = b_1 b_2 b_3 b_4 b_5 b_6 \longrightarrow b_i \in \{0,1\}$

$r = (2b_1 + b_6) \quad 0 \le r \le 3$

c = integer representation $(b_2 b_3 b_4 b_5)$

$0 \le c \le 15$

	c															
	0	1	2	...	13	14	15



* **Permutation** —

$P : \{0,1\}^{32} \longrightarrow \{0,1\}^{32}$

P:

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11		4

$P(x_1 x_2 \ldots x_{32}) = (x_{16} x_7 \ldots x_4)$

* **Key Scheduling Algorithm**

$K \rightarrow$ key size of DES $(56+8)$ → parity bits.

$K_1 K_2 \ldots K_{16} \rightarrow$ each 48

→ Input : 64 bit Key $k = K_1 \ldots K_{64}$

→ output : 16 round keys K_i $1 \le i \le 16$, where length of K_i is 48

1) Define $1 \le i \le 16$, v_i where $v_i = 1$ $i \in \{1, 2, 9, 16\}$ else $v_i = 2$
2) Discard 8 parity check bits → $\tilde{K}$
3) $T = PC1(\tilde{K})$ $PC1 : \{0,1\}^{56} \longrightarrow \{0,1\}^{56}$
4) $(C_0, D_0) = T$ → length of C_0 & D_0 = 28
5) For $i = 1$ to 16 {

$C_i \leftarrow (C_{i-1} \circlearrowleft v_i)$ ↺ left circular shift

$D_i \leftarrow (D_{i-1} \circlearrowleft v_i)$

$K_i = PC2(C_i, D_i)$

}

round key

$PC2 : \{0,1\}^{56} \longrightarrow \{0,1\}^{48}$

PC1 :

C_0

57	49	41	.....	9
⋮				⋮
.....				36

PC1 is just doing some sort of permutation

D_0 [— — — — — ;

indexes are 8, 16, .. etc

L - - - - - - of R

D_0 [- - - - - - ; - - - - - -]

parity bits are missing 8, 16, -- etc

$PC1(K_1 K_2 \dots K_{64}) = PC1(K_1 K_2 K_3 \dots K_7 K_9 K_{10} \dots K_{15} K_{17} \dots K_{63})$

$= K_{57} K_{49} K_{41} \dots$

$PC1(\bar{K_1} \bar{K_2} \dots \bar{K_{64}}) = \bar{K_{57}} \bar{K_{49}} \dots \bar{K_9}$ } IMP

$= \overline{PC1(K)}$

same logic for PC1 and PC2 both.

14th bit will be 1st bit of PC2

PC2:

14	17	11	24	1	5
1					
.					
.					
.					
-	-	-	-	-	

these are input string's bit position

so 1st bit of output = 14th bit of input just like that

PC2 : Just deletes few bits as it is (refer textbook).

* COMPLEMENT PROPERTIES :—

$DES(m, k) = c$

$DES(\bar{m}, \bar{k}) = \bar{c}$ } VIMP

Key scheduling

$KS(k) = K_1 \dots K_{16}$

$KS(\bar{k}) = \bar{K_1} \dots \bar{K_{16}}$

since PC1, PC2 and leftshift (all are complement to complement mapping

In exam, explain why. all that is written below ↓

IP(m) ↓ IP($\bar{m}$)

complement of each other.

m		$\bar{m}$	
L_0	R_0	$\bar{L_0}$	$\bar{R_0}$
L_1	R_1	$\bar{L_1}$	$\bar{R_1}$

m: $R_1 = L_0 \oplus f(R_0, K_1)$

$\bar{m}$: $R_1 = \bar{L_0} \oplus f(\bar{R_0}, \bar{K_1}) = \bar{L_0} \oplus f(R_0, K_1)$

$\bar{R_1}$ actually

$E(\bar{R_0}) = \overline{E(R_0)} \oplus \bar{K_1}$

$= E(R_0) \oplus K_1$

complement cancel out

same for other

$m \rightarrow \bar{m}$ and $K \rightarrow \bar{K}$

$c \rightarrow \bar{c}$

* Brute Force attack complexity = 2^{56} [since size of key = 56 bits]

* **Brute Force attack complexity = 2^{56}** [since size of key = 56 bits]

$S_1 = \{K_1, K_2, \ldots K_{2^{56}}\}$ → Set of all possible keys.

* **Chosen plaintext attack** → like game. (2^{55})

m $\quad$ $\bar{m}$ → Given

all 0s $\quad$ all 1s

$DES(m, K) = C_1$

unknown → same $\quad$ not complement

$DES(\bar{m}, K) = C_2$

Attacker- → from the set S_1 above

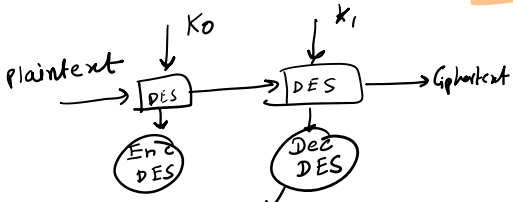
$DES(m, K_i) = C$

if $C \neq C_1$ discard K_i from S

[or $C \neq \bar{C}_2$ same thing] → if $\bar{C} \neq C_2$ discard $\bar{K}_i$ from S.

in every iteration I am deleting 2 keys checking 2 keys ∴ search reduced by half ∴ new complexity = 2^{55}

here attacker is only doing 1 decryption whereas what I said needs 2 decryption at once. I said that we calculate $DES(m, k_i)$ and $DES(m, \bar{k}_i)$ then discard k_i & $\bar{k}_i$ → Two calculations.



This will not decrypt bcs, it will decrypt if $K = K_0$ but here K_1 ∴ It will again do encryption only.

{ED, DD, EE, DE} → Anything can be used.

K → 128-bit = (concatenation) (K_0, K_1)

Among this 128, 16 are parity check bits.

∴ 112 bit security.

Exhaustive search → 2^{112}

* **Double DES:-**

Attacker is having one valid plaintext, ciphertext pair → Know plaintext attack.

P, C → Double DES

Select K_i

$Enc_{DES}(P, K_i) = X_i$

$Enc_{DES}(C, K_j) = Y_j$

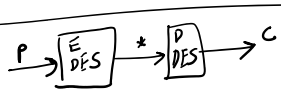
$[X_i, K_i]$ → Table 1 → store it in a table

$X_i = Y_j$ → Found correct!

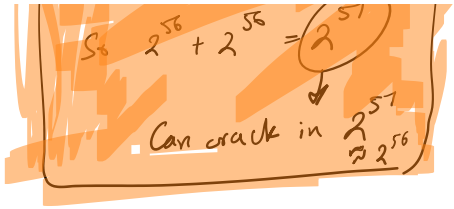
$[Y_j, K_j]$ → Table 2

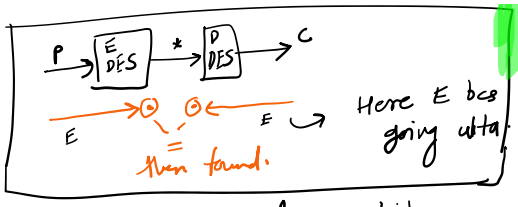
$D_0(X_i, K_i)$ in 2^{56} and $D_0(Y_j, K_j)$ in 2^{56}

So $2^{56} + 2^{56} = 2^{57}$



Meet in the middle attack



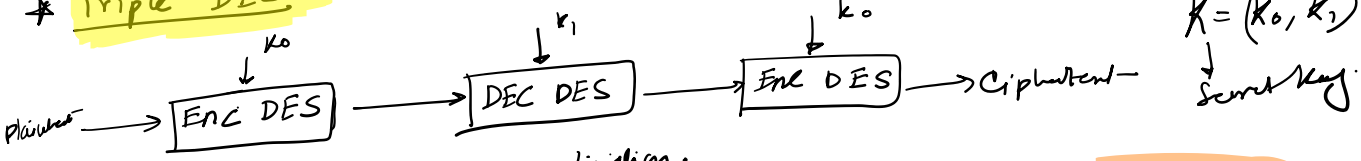


in the middle attack

Increasing m bit secret key which provides n bit security to 2m bit secret key then n bit security does not increase to 2m. as we saw above. This is not only for Double DES. It's for all. It will more or less provide n-bit only.

Vimp

* Triple DES :-



EDE, DEE, etc → any combination.

Here, we did twice length key and 2^{2n} bit security we got. (2^{112} bit security)

but in triple DES format

$K_0 - K_1 - K_0$ if EDE (thrice encryption)
if $K_0 - K_1 - K_1$ in EDE then this part no sense (only once encryption)

* Few Mathematical things :-

Search complexity $n \rightarrow n/2$ if quantum computers come.
else only method is exhaustive search (2^n).
To get n in quantum case, use 2n bit key in DES setup $\rightarrow 2^{2n/2} \rightarrow 2^n$ bit you will get.

$R \subseteq X \times Y$

A binary operation * on a set S is $S \times S \rightarrow S$
That is * is a rule which assigns to each ordered pair of elements from S to an element of S.

$* : S \times S \rightarrow S$

$*(a,b) = c$ where $a, b, c \in S$
$*(b,a) = d$ $d \in S$
it is not necessary that $d = c$

⇒ Group :-
A group $(G, *)$ consists of a set G with a binary operation * on G satisfying three axioms :-

1) The group operation is associative
$a * (b * c) = (a * b) * c \quad \forall\ a, b, c \in G$

2) There is an element $1 \in G$ called the identity element such that
$a * 1 = 1 * a = a \quad \forall\ a \in G$

3) For each $a \in G$, there exists an element $a^{-1} \in G$ called the inverse of 'a' such that :-

3) For each $a \in G$, there exists an element $a^{-1} \in G$ called the inverse of 'a' such that:—

$$a^* * a^{-1} = 1 = a^{-1} * a \quad \forall a \in G$$

→ If $a*b = b*a \ \forall a, b \in G$ then group G is called abelian (or commutative)

→ Lets consider :—

→ * : matrix multiplication over sq. matrix

↳ not commutative.

→ G = {set of all matrices which are invertible} but it will not be a commutative group.

(G, *)

→ Z : Set of integers. (commutative group)

(Z, +)

need some operation

→ (Z, *) : Not a group since $\left(\frac{1}{a}\right)$ (inverse) is not an integer $\notin Z$

→ (Z, −) : Not a group ∵ 1st and 2nd getting violated.

→ (Q, *) : No $\left(\frac{1}{0}\right)$ → not defined $\frac{1}{0} \notin Q$

→ $(Q - \{0\}, *)$: Yes

→ $(Z_n, +_n)$: Yes

→ $(Z_n, *_n)$: No → because for a^{-1} you need gcd = 1

→ $U_n = \{ x \in Z_n \setminus \{0\} \mid \gcd(x, n) = 1 \}$

→ set subtraction.

$(U_n, *_n)$: Yes

→ $\{Z_p \setminus \{0\}, *\}$: Yes

Prime number set

WRITTEN IN NOTEBOOK

* AES :—

...continued

→ All round func. need to be invertible

→ Round function :— (AES 128)

— Last round func. is diff from others.

— $f_1 = f_2 = \cdots = f_9$ (f_{10} is diff)

- Last round func. is diff
- $f_1 = f_2 = \cdots = f_9$ (f_{10} is diff)
- First 9 round functions consists of -

f_9 [f_{10} { • Subbytes • Shift rows • Mix columns }] ← True for all AES types.

↓ All

$\{0,1\}^{128} \rightarrow \{0,1\}^{128}$

f_1 to f_{n-1} f_n^{th}

for n rounded AES.

for 1st 1-9 rounds.

Subbytes-

$\{0,1\}^{128} \rightarrow \{0,1\}^{128}$

$x = x_0 x_1 \cdots x_{15}$

Size of x_i = 8 bit

$$\begin{bmatrix} x_0 & x_4 & x_8 & x_{12} \\ x_1 & x_5 & x_9 & x_{13} \\ x_2 & x_6 & x_{10} & x_{14} \\ x_3 & x_7 & x_{11} & x_{15} \end{bmatrix} \rightarrow \begin{bmatrix} S_{00} & S_{01} & S_{02} & S_{03} \\ S_{10} & & & \\ S_{20} & & & \\ S_{30} & & & \end{bmatrix}$$

S-BOX $S: \{0,1\}^8 \rightarrow \{0,1\}^8$; $S(0)=0$ → fixed constant

$(S_{ij})_{4\times4}$

1) $(C_7 C_6 C_5 C_4 C_3 C_2 C_1 C_0) \leftarrow (01100011)$

MSB ↑ ↓ LSB → in hexa : 63

2) $S(S_{ij}) = (a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0)$

3) For i=0 to 7

$$b_i = (a_i + a_{(i+4)\%8} + a_{(i+5)\%8} + a_{(i+6)\%8} + a_{(i+7)\%8} + c_i) \bmod 2$$

for making it circular

Acts as XOR inside. or +

4) $(b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0)$

output = S'_{ij} → one element of matrix

$$\begin{bmatrix} S_{00} & S_{01} & S_{02} & S_{03} \\ S_{10} & & & \\ S_{20} & & & \\ S_{30} & & & \end{bmatrix} \xrightarrow[\text{process}]{\text{Using above}} \begin{bmatrix} S'_{00} & S'_{01} & S'_{02} & S'_{03} \\ S'_{10} & & & \\ S'_{20} & & & \\ S'_{30} & & & \end{bmatrix}$$

If $S_{00}=0$ then $S'_{00} = 63_{hex}$

∵ we saw $S(0)=0$ in the S-box and hence c_i term remains in b_i

∴ we get 63 (hex).

$S: \{0,1\}^8 \rightarrow \{0,1\}^8$

$S: \{0,1\}^8 \longrightarrow \{0,1\}^8$

$S(x) = Y \qquad a_i \in \{0,1\}$

$x = (a_7\, a_6\, a_5\, a_4\, a_3\, a_2\, a_1\, a_0)$ (MSB ↑, LSB ↓)

$p(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5 + a_6 x^6 + a_7 x^7$

$\deg(p(x)) < 8$

$p(x) \in \left(F_2[x] / \langle x^8 + x^4 + x^3 + x + 1 \rangle, +, \times\right)$ ⇝ we say this

↓ field

∴ any non-zero poly will have an inverse ∴ field

find multiplicative inverse of $p(x)$ under modulo $(x^8 + x^4 + x^3 + x + 1)$

($p(x)$ would have an inverse ∴)

$p(x) \cdot q(x) \equiv 1 \pmod{(x^8 + x^4 + x^3 + x + 1)}$

$x^8 + x^4 + x + 1 \mid p(x)q(x) - 1$

$p(x)q(x) - 1 = h(x^8 + x^4 + x^3 + x + 1)$

$p(x) \cdot q(x) = 1 + h(x) \cdot (x^8 + x^4 + x^3 + x + 1)$

$1 = p(x) \cdot q(x) + h_1(x)(x^8 + x^4 + x^3 + x + 1)$ ← multiplicative inverse of $p(x)$

∴ By extended euclidean algo ↓ ↑ ⇝ we will use this (reverse)

$1 = \gcd(p(x), x^8 + x^4 + x^3 + x + 1)$ ↪ they are co-prime.

Example –

$S(01010011)$

$\{0,1\}^8 \to \{0,1\}^8$

$01010011 \longrightarrow x^6 + x^4 + x + 1$

$p(x) = x^6 + x^4 + x + 1$

$g(x) = x^8 + x^4 + x^3 + x + 1$

$p(x) \cdot q(x) \equiv 1 \pmod{\frac{x^8 + x^4 + x^3 + x + 1}{g(x)}}$

Perform :-

$x^6 + x^4 + x + 1 \;\big|\; x^8 + x^4 + x^3 + x + 1 \;\big|\; x^2 + 1$

$x^8 + x^6 + x^3 + x^2$

$x^6 + x^4 + x^2 + x + 1$

$x^6 + x^4 + x + 1 \;\big|\; x^6 + x^4 + x + 1 \;\big|\; x^4 + x^2$

x^6

$x^4 + x + 1$

$x^4 \;\big|\; x^2 \;\big|\; x + 1$

$x^2 + x$

x

$x + 1$

1 → gcd.

go from bottom to up

$1 = x^2 + (x+1)(x+1)$ ($1 \equiv 1 \pmod 2$)

$= x^2 + \left[x^6 + x^4 + x + 1 + (x^2)(x^4 + x^2)\right](x+1)$

take x^2 common ↗

$= (x+1)\left[x^6 + x^4 + x + 1\right] + \left[1 + (x^4 + x^2)(x+1)\right]x^2$

$1 = (x+1)(x^6 + x^4 + x + 1) + (1 + x^5 + x^4 + x^3 + x^2)x^2$

$1 = (x+1)(x^6 + x^4 + x + 1) + (1 + x^5 + x^4 + x^3 + x^2)[$

$(x^8 + x^4 + x^3 + x + 1)$

$+ (x^2 + 1)(x^6 + x^4 + x + 1)]$

↪ replaced x^2

$+ (x+1)(x^6+x^4+x+1)$ → replaced x^8

$1 = (1+x^5+x^4+x^3+x^2)(x^8+x^4+x^3+x+1)$

$+ [(x+1) + (x^2+1)(1+x^5+x^4+x^3+x^2)](x^6+x^4+x+1)$ → Match coeff. from above.

$1 = h(x)\cdot g(x) + [x+1+x^2+x^7+x^6+x^5 + x^4+1+x^5+x^4+x^3+x^2](x^6+x^4+x+1)$

$1 = h(x)\cdot g(x) + (x^7+x^6+x^3+x)(x^6+x^4+x+1)$

multiplicative inverse of x^6+x^4+x+1 under mod. $x^8+x^4+x^3+x+1$

$\therefore S(\overbrace{01010011}^{5}\ \overbrace{}^{3}) = (11001010)$ → inverse.

found sbox.

$= (q_7 q_6 \cdots q_0)$ → we saw formula above.

$b_i = (a_i + a_{(i+4)\%8} + \text{- - - - -}) \bmod 2$

$b_0 = (a_0 + a_4 + a_5 + a_6 + a_7 + c_0) \bmod 2$ → as taken from 11001010

$= (0+0+0+1+1+1) \bmod 2$

$= 1$

$b_1 = 0$

$\vdots$

$b_7 = 1$

$(b_7 b_6 b_5 \cdots b_0) = (\underbrace{1110}_{E}\ \underbrace{1101}_{D})$ → in hexa

Subbyte (53) = ED

initial input ↑ ↑ final output

⇒ Now, pehle se calculate karke table me store kar lenge.

Lookup table.

⇒ 16×16 table = 256 possibilities ($\because 2^8$)

⇒ first 4 MSB = rows, last 4 LSB = col., value in table = final output.

- Shift rows

$\{0,1\}^{128} \longrightarrow \{0,1\}^{128}$

$\begin{matrix} 0 \\ 1 \\ 2 \end{matrix}\begin{bmatrix} S_{00} & S_{01} & S_{02} & S_{03} \\ S_{10} & S_{11} & S_{12} & S_{13} \\ S_{20} & S_{21} & S_{22} & S_{23} \end{bmatrix} \rightarrow \begin{bmatrix} S_{00} & S_{01} & S_{02} & S_{03} \\ S_{11} & S_{12} & S_{13} & S_{10} \\ S_{22} & S_{23} & S_{20} & S_{21} \end{bmatrix}$ ← 1 left shift, ← 2 left shift

$$\begin{matrix}0\\1\\2\\3\end{matrix}\begin{bmatrix} & & & \\ S_{10} & S_{11} & S_{12} & S_{13} \\ S_{20} & S_{21} & S_{22} & S_{23} \\ S_{30} & S_{31} & S_{32} & S_{33}\end{bmatrix} \rightarrow \begin{bmatrix} S_{11} & S_{12} & S_{13} & S_{10} \\ S_{22} & S_{23} & S_{20} & S_{21} \\ S_{33} & S_{30} & S_{31} & S_{32}\end{bmatrix}$$

← 1 left shift

← 2 left shift

Total operations = 7 ∴ in const time.

— Mix Column —

Mixing of columns. $\{0,1\}^{128} \rightarrow \{0,1\}^{128}$

$$[\quad] \rightarrow [s'_{ij}]_{4\times4}$$

$$s'_{00} = \left((x)s_{00} + (x+1)s_{10} + (1)s_{20} + (1)s_{30}\right) \bmod (x^8+x^4+x^3+x+1)$$

$$\begin{bmatrix} S_0 & S_{01} & S_{02} & S_{03} \\ S_{10} & S_{11} & S_{12} & S_{13} \\ S_{20} & S_{21} & S_{22} & S_{23} \\ S_{30} & S_{31} & S_{32} & S_{33}\end{bmatrix}$$

$S_{32} = (x)S_{32} + (x+1)S_{02} + (1)S_{12} + (1)S_{22}$

$S_{13} = (x)S_{13} + (x+1)S_{23} + (1)S_{33} + (1)S_{03}$

work in circular.

Consider the column $c \in \{0,1,2,3\}$

for i=0 to 3

t_i = Binary to Poly (S_{ic})

$$u_0 = \left\{\left([x]*t_0\right) + \left([x+1]*t_1\right) + \left([1]*t_2\right) + \left([1]*t_3\right)\right\} \bmod (x^8+x^4+x^3+x+1)$$

u_1 = Similar Analogy.

s'_{ic} = Polynomial to Binary (u_i)

$$S' = \begin{bmatrix} x & x+1 & 1 & 1 \\ 1 & x & x+1 & 1 \\ 1 & 1 & x & x+1 \\ x+1 & 1 & 1 & x\end{bmatrix}\begin{bmatrix} S_{00} & S_{01} & S_{02} & S_{03} \\ S_{10} & S_{11} & S_{12} & S_{13} \\ S_{20} & S_{21} & S_{22} & S_{23} \\ S_{30} & S_{31} & S_{32} & S_{33}\end{bmatrix}$$

IMP

$\bmod (x^8+x^4+x^3+x+1)$

= Multiply this and the final matrix will store the correct multiplications. (S')

After getting polynomial from this, convert to binary by taking coeff.

Let's say → x^7+x^3+x = 10001010 → coeff = 1 jo jo terms hai

$$\begin{bmatrix} S'_{00} \\ S'_{10} \\ S'_{20} \\ S'_{30}\end{bmatrix} = \begin{bmatrix} x & x+1 & 1 & 1 \\ 1 & x & x+1 & 1 \\ 1 & 1 & x & x+1 \\ x+1 & 1 & 1 & x\end{bmatrix}\begin{bmatrix} S_{00} \\ S_{01} \\ S_{20} \\ S_{30}\end{bmatrix}$$

$S_{00} = 95$, $S_{10} = 65$, $S_{20} = FD$, $S_{30} = F3$ → In hexadecimal

$S_{00} = 1001\,0101 = x^7+x^4+x^2+1$

$\ldots^6 + x^5 + x^2 + 1$

$S_{00} = 10, S_{10} = \ldots$

$S_{00} = 10010010 = x^7 + x^4 + x^2 + 1$

$S_{10} = 01100101 = x^6 + x^5 + x^2 + 1$

$S_{20} = 11111101 = x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + 1$

$S_{30} = 11100011 = x^7 + x^6 + x^5 + x^4 + x + 1$

Now, $S'_{00} = (x)\left[x^7 + x^4 + x^2 + 1\right] + (x+1)\left[x^6 + x^5 + x^2 + 1\right] + (1)\left(x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + 1\right) + (1)\left[x^7 + x^6 + x^5 + x^4 + x + 1\right]$

$\mod (x^8 + x^4 + x^3 + x + 1)$

$= x^8 + x^5 + x^3 + x + x^7 + x^6 + x^3 + x + x^6 + x + x^2 + 1 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + 1 + x^7 + x^6 + x^5 + x^4 + x + 1$

$\mod (x^8 + x^4 + x^3 + x + 1)$

$= x^7 + x^4$

$S'_{00} = x^7 + x^4$

$= 10010000$

$= 90 \rightarrow$ In hexadecimal

and so on :—

— Key Scheduling Algo —

$K_1, K_2, \ldots\ldots K_{11}$

length of each K_i = 128 bit

$K = key[0] + key[1], \ldots\ldots key[15]$

length of $key[i]$ = 8 bit

i) ROTWORD $(B_0, B_1, B_2, B_3) = (B_1, B_2, B_3, B_0)$

length of B_i = 8 bit

ii) SUBWORD $(B_0, B_1, B_2, B_3) = (B_0', B_1', B_2', B_3')$

$B_j' = \text{subbyte}(B_i)$

RCon[1] = 01000000
⋮
RCon[10] = 36000000

} fixed constants.

for i = 0 to 3

$w[i] = (key[4i], key[4i+1], key[4i+2], key[4i+3])$

for i = 4 to 43

temp = w[i-1]

if $i \equiv 0 \pmod 4$ → if i divisible by 4

then temp = SUBWORD(ROTWORD(temp)) ⊕ RCon[i/4]

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

return $(w[0], w[1], \ldots\ldots w[43])$

return $(w[0], w[1], \ldots, w[43])$

length of $w[i] \rightarrow 32$ bits

$K_1 = w[0] \| w[1] \| w[2] \| w[3]$

$K_2 = w[4] \| w[5] \| w[6] \| w[7]$

$\vdots$

$K_{11} = w[40] \| w[41] \| w[42] \| w[43]$

} Round Keys we get (11 of them)

For decryption

Do ulta, { shift rows → invertible
sub-bytes → Invertible (∵ after doing all, we get look up table ∴ invertible)
mix column → Invertible (∵ the matrix jisse multiply karte is also invertible)

★ MODES of OPERATION

ECB, CBC, CFB, OFB, IG E

↓ v. simple ↓ most used

– ECB (Electronic Code Block) :–

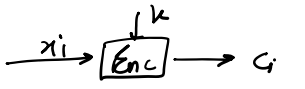
Input : Key K, n-bit plaintext $x_1, \ldots, x_t$

i) Encryption : $Enc(x_i, k) = C_i$

$1 \le i \le t \quad C = C_1 \ldots C_t$

ii) Decryption : $Dec(C_i, k) = x_i$

$i \le i \le t$



- Easy
- can do parallelly encryption
- Cons: if Block 1 = Block 2 then cipher 1 = cipher 2 } pata chal jayega ki 1=2.

CBC

Input : Key K, n-bit plaintext blocks $x_1, x_2 \ldots x_t$

i) Encryption : $C_0 = IV$ → Public → known to everyone, IV = Initial Vector

$C_j = Enc(C_{j-1} \oplus x_j, K)$

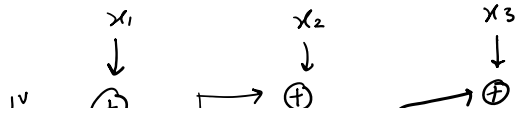
$i \le j \le t$

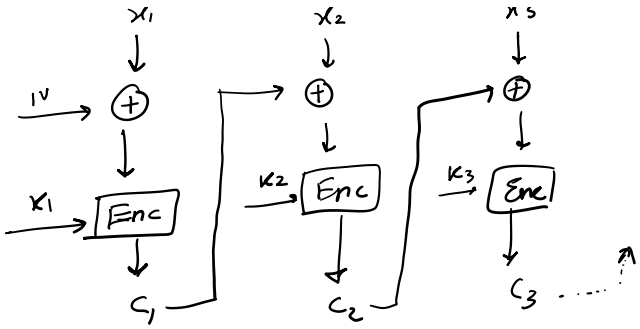
ii) Decryption : $C_0 = IV$

$x_j = Dec(C_j, k) \oplus C_{j-1}$

- If one block is not known, then other blocks also not known
- Block 1 (C_0) ke bina nahi hoga aage kaa ∴ C_0 must be known ∴ public
- Handles nicely if Block 1 = Block 2 Then CT 1 ≠ CT 2

Encryption Structure

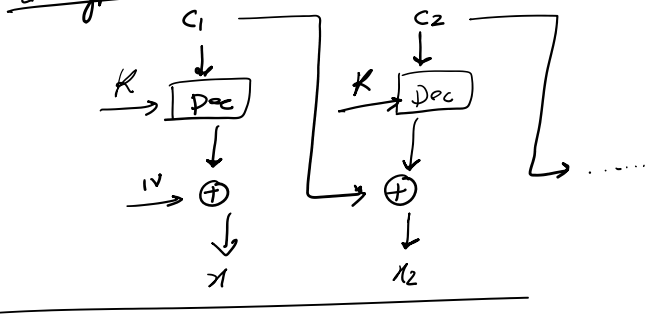




Block 1 = Block 2
Then CT 1 ≠ CT 2

- Key K same generally for all rounds ∵ heavy computation
- Whatsapp uses this.

Decryption



- Block Ciphers encrypt blockwise
- Stream Ciphers encrypt bitwise

Types
- Block — AES etc.
- Stream

$M = m_0\ m_1 \ldots\ldots m_L \qquad m_i \in \{0,1\}$
$K = k_0\ k_1 \ldots\ldots k_L$

$C = M \oplus K = (m_0 \oplus k_0)(m_1 \oplus k_1) \ldots\ldots (m_L \oplus k_L)$

$Enc: C_i = m_i \oplus k_i$

$Dec: m_i = C_i \oplus k_i$

Although C contains M in some form but you can't extract ∵ masking with K
∴ No other way left to crack
Can only guess.

* Most secure, Most efficient — if some conditions, among all types

* Shannon's Notion of Perfect Secrecy —
If there's a algo which does not reveal any info upon message than its perfect algo & perfectly secret

$P[M = m_i \mid C = C_{m_i}] = P[M = m_i]$ — in mathematical form

no extra advantage if C is known

before & after same probability

[Guess left ∴ Perfectly Secret]

If proven, then perfectly secure.

$m \in \{0,1\}$
$P(m=0) = p$
$P(m=1) = 1-p$

$K \in \{0,1\}$
$P[K=0] = 1/2$
$P[K=1] = 1/2$
} since randomly selected K.

$C = Enc(m, K) = m \oplus K$

$C \in \{0,1\}$

$c \in \{0,1\}$

$P(c=0) = P(m \oplus K = 0) = P(\{m=0, K=0\} \cup \{m=1, K=1\})$

$= P(m=0, K=0) + P(m=1, K=1)$ → ∵ m & K independent

$= P(m=0) \cdot P(K=0) + P(m=1)P(K=1)$

$= (p)(\frac{1}{2}) + (1-p)(\frac{1}{2})$

$P(c=0) = \frac{1}{2}$

VIMP

⊕ So even if message is bias, output (c) is getting randomized.

$\therefore P(c=0) = P(c=1) = 1/2$

$C = Ch_1 \mid m = m_1$ ✓ fixed if already occurred

So, $K = Ch_1 \oplus m_1$ ← then

$P(m=m_1 \mid C=Ch_1) = \frac{P(m=m_1, C=Ch_1)}{P(C=Ch_1)}$

Here it means we know m_1

Using Bayrian

Here we don't know m_1

$\therefore P(\cdots) \neq P(K=\cdots\oplus\cdots)$

$= \frac{P(C=Ch_1 \mid m=m_1) \times P(m=m_1)}{(1/2)}$

$C=Ch_1$ and $m=m_1$ only if $K=Ch_1 \oplus m_1$ ←

$= \frac{P(K=Ch_1 \oplus m_1) \times P(m=m_1)}{(1/2)}$

$= \frac{(1/2)\,P(m=m_1)}{(1/2)} = P(m=m_1)$

Hence proved ∴ Perfectly Secret.

1) $C = m \oplus K$

$C_1 = m_1 \oplus K$

$C_2 = m_2 \oplus K$

$C_1 \oplus C_2 = (m_1 \oplus K) \oplus (m_2 \oplus K) = m_1 \oplus m_2$

if $C_1 \oplus C_2 = 0$ then we get know $m_1 = m_2 = 0$ or $m_1 = m_2 = 1$

basically $m_1 = m_2$ → this is revealed.

$K = k_0 \ldots k_l$

$m = m_0 \ldots m_n$

$r = n - l$

$n > l$

$K' = K \,||\, k_0 \ldots k_{r-1}$ ← we know n & l bits

$C = m \oplus K'$

$C_{l+1} = m_{l+1} \oplus k_0$

$C_{l+1} = m_{l+1} \oplus k_0$

$C_0 = m_0 \oplus k_0$

$C_{l+1} \oplus C_0 = m_{l+1} \oplus m_0$ ← revealing info.

① ∴ length of Key ≥ length of message

② You cant use same key to encrypt different messages

} OTP — One Time Padding Method

— Not practical, as done → sender & receiver and pe how to make sure K generated randomly are same !!

M = 0101 1

K' = 01000

C = 0001 1

m = 5

k' = 4

0 ⊕ 1 = 0 ⊕ 1