Info Security

Authentication:

↳ It is one of 5 principle of security
↳ verifying the authenticity is ma
↳ authenticator must be there to authenticate the message

Methods:

(i) Message Encryption → cipher text
(ii) Message Authentication Codes → (MAC)
(iii) Hash-function → hash value

(i) MAC

Message Authentication Code
↳ also called cryptographic checksum

↳ fixed block size data
2 MB → 2 KB

↳ It is appended with the message
↳ Communication Parties will share private key

=> We will use a secret key to generate a small fixed size/block of data called MAC or Cryptographic Checksum

Explanation of Mechanism

Let A -> Sender
B -> Reciever

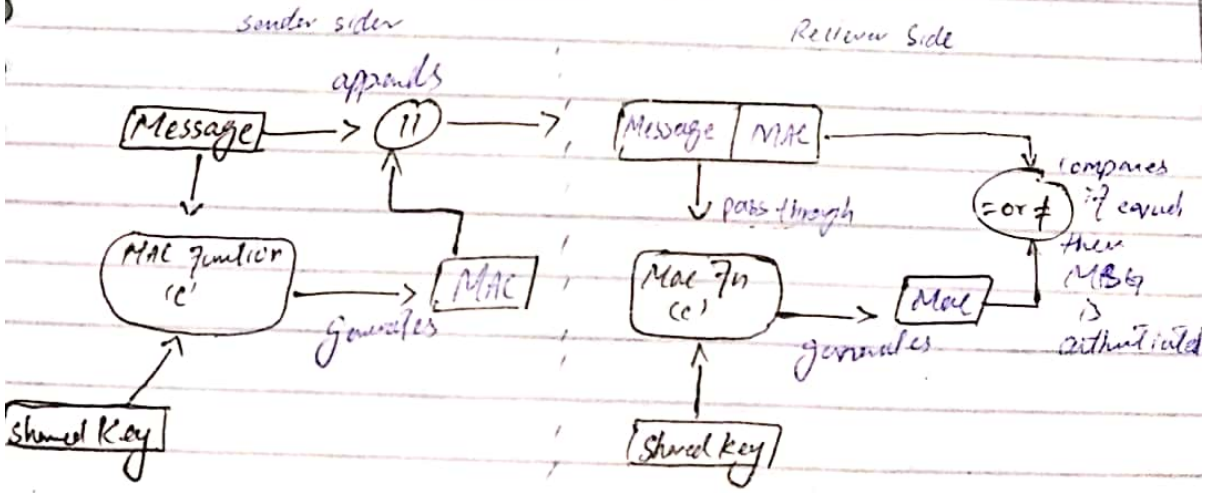
when A sends message to B, it calculate its MAC by passing it with some MAC function. MAC function takes Message & shared secret key as parameters and generates MAC as

$$MAC = C(K, M)$$

where
M = Message from A
K = Shared Key
C = MAC function

Diagrams

(i) MAC - for authentication



=> Only authentication can be achieved using this technique
=> Integrity and Confidentiality can't be achieved

(2) MAC - for authentication & confidentiality

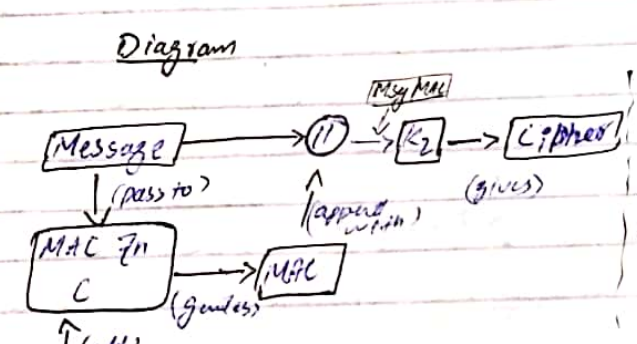
This mechanism has 2 scenarios

(a) authentication tied to plain text

we Encrypt the [Message | MAC] block by using an extra key K_2 and decrypt on reciever's side

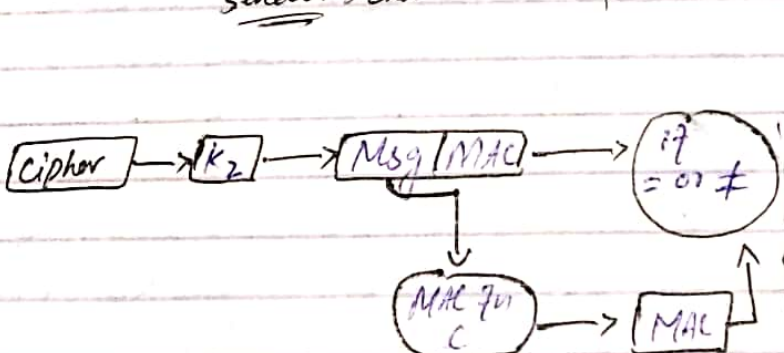
After decryption it is compared.

Diagram



Msg pass to MAC fn and genrates MAC MAC appends with K_2 and genrates cipher text

sender's end



Msg recieved and decrypted using K_2. Then MSG is passed to MAC fn. The resultant MAC is compared

reciever's end

=> this way authentication & confidentiality is achieved.

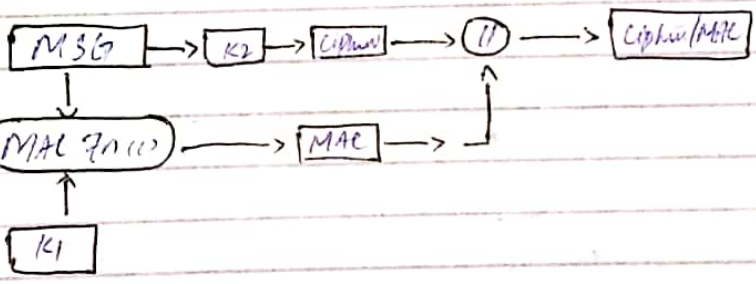
(b) Authentication tied to cipher text

=> Msg is first encrypted by K_2. Then MAC generated from original MSG using MAC function is appended and send to reciever

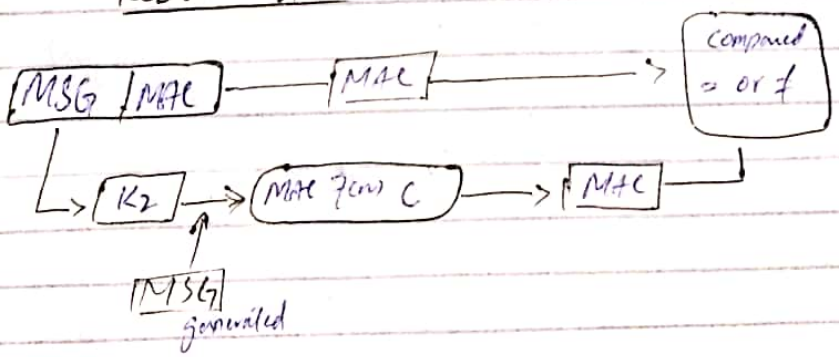
=> At reciever's end, the ciphere text is decrypted with K_2 and passed to MAC function. Then resultant MAC is compared with recieved MAC

Diagrams

Senders' END



Reciever's END



=> Both authentication & confidentiality

(Same keys) (= MAC means no changing)

Digital Signature

Need for
no proof for that
A send the msg to B
because actual 'A' can be
hacked or changed his mind
and B can't do anything

=> Msg/data comming from authenticated person.

=> V. imp in ecommerce, online transaction etc.
=> based on A-Symmetric cryptography
encryption -> private key
decryption -> public key

=> Used for
↳ msg authentication
↳ non repudiation / denying
↳ msg integrity

=> Don't used for confidentiality

Diagram Explanation

=> At the send's side

msg is passed to some hash function 'H' which is a digital signature generating algo and accepting private key of sender

This function will generate hash code 'h' which is append with original msg and then encrypted using private key of Sender

=> At the reciever's end

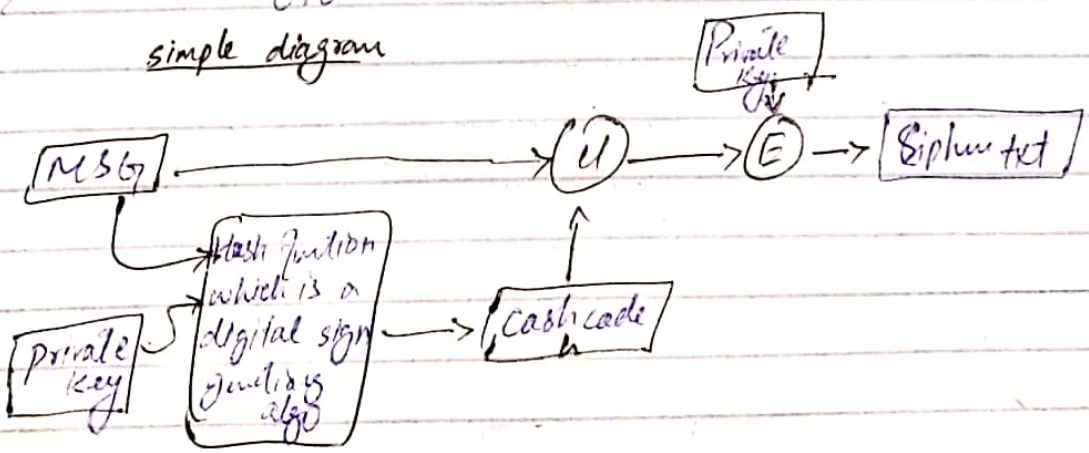
both encrypted msg and hash code is recieved at reciever's end.

the encrypted msg is decrypted using public key of sender

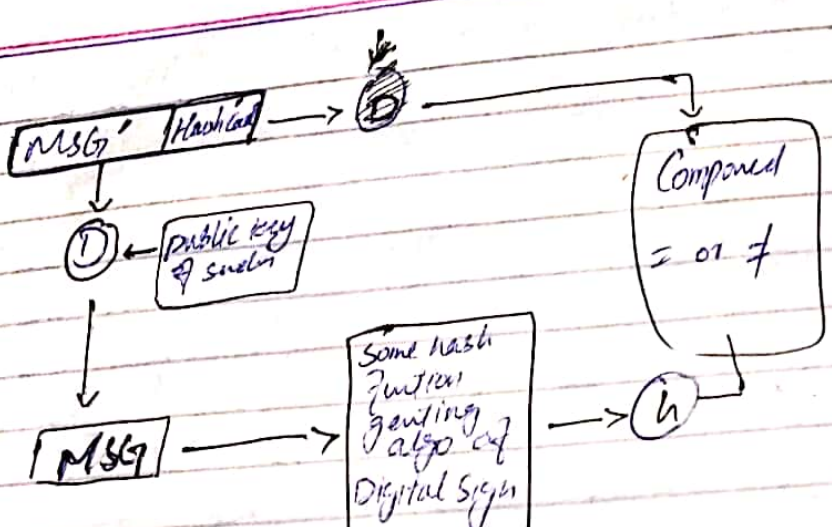
Then this msg is passed to hash function 'H' which generates hash code 'h'

Hash code 'h' is compared with recieved one

simple diagram



Send's END



Reciever's END

=> Achieving Confidentiality using (E) and (D)

=> Integrity

if changed then hash values don't match

=> Sender sends 2 doc MSG and Sign

=> Sign must be some unique info to sender

-> to prevent forgery and denial

=> It must be easy to produce digital sign

-> " " " " " recognize & verify " "

(i) Key gen algo - to gen private key

I/p : M + Private key

(ii) Signing Algo : o/p : Digital sign

(iii) Verifying Algo : using Public key & sign

Hash Function

↳ Technique to achieve authenticity
↳ similar to MAC
but it doesn't use a key

↳ Hash function takes variable size, message and produce a fixed length output called Hash code / Hash value OR Message Digest

↳ Only take message as I/P

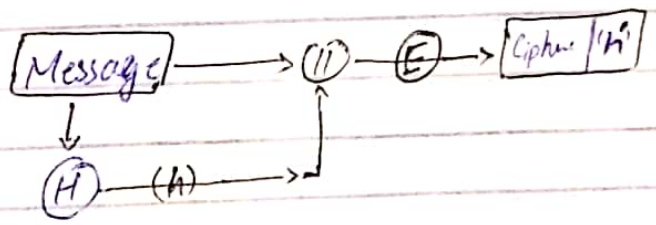
↳ A hash value 'h' is generated by a Hash function

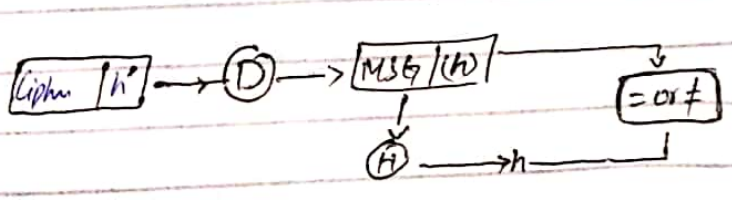
$H(M)$ = fixed length code 'h'

↳ Hash function is also called Compression Function

=> Diff methods to provide authentication

(i) Same as Digital Signature

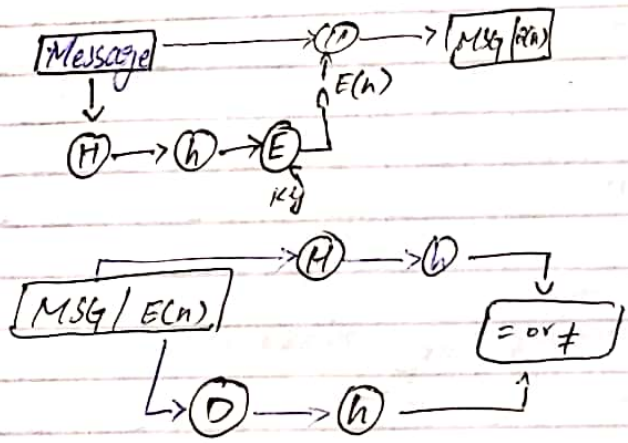




=> This gives authentication + confidentiality
↳ msg is encrypted

- b/c only A & B shares the secrete keys, the msg has come from A and has not been altered

(2)



=> Message is passed to Hash function which yield hash code 'h'. Then 'h' is encrypted and appended with Message

=> Appended message is passed to Hash function which yields 'h'. The encrypted h is decrypted and compared with hash code h

=> Only authentication

=> No confidentiality -> msg is not encrypted

=> We can use this scenario for no confidentiality, only authentication is needed, msg is not private, this reduces processing time.

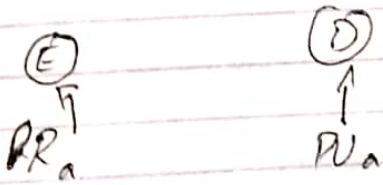
Method - 3

This method is same as the previous method
The only difference is that we use Asymmetric key for encryption/decryption process

For encryption
We use private key of Sender

For decryption
We use public key of Sender

* Same Diagram as Method - 2 just added



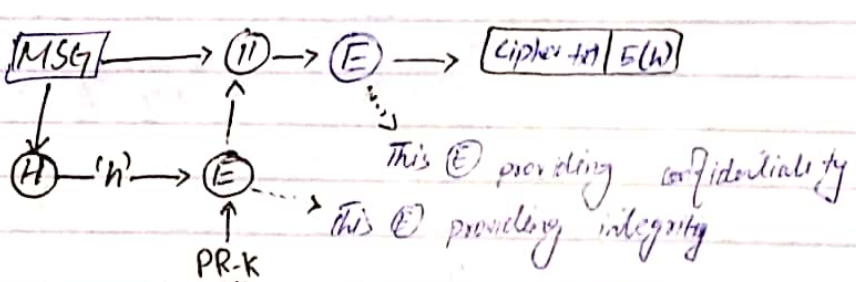
=> Only authentication
=> No confidentiality
=> Fast processing time

Method-4

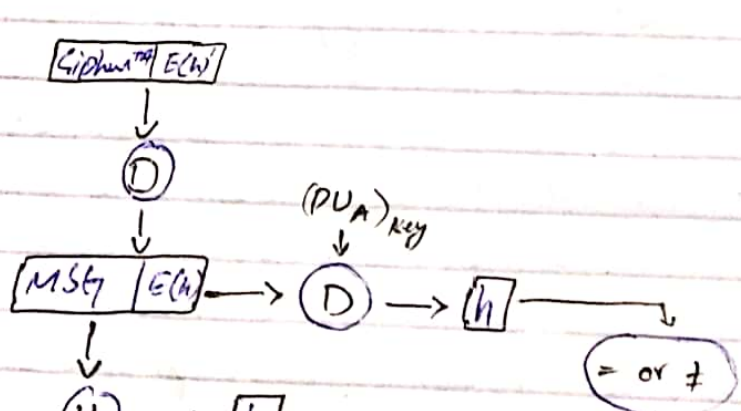
In this technique, we add an extra key after appending MSG and E(h) for encryption of MSG and E(h) This helps us achieving the authentication and confidentiality at the same time

Diagram

Senders End



Recievers End



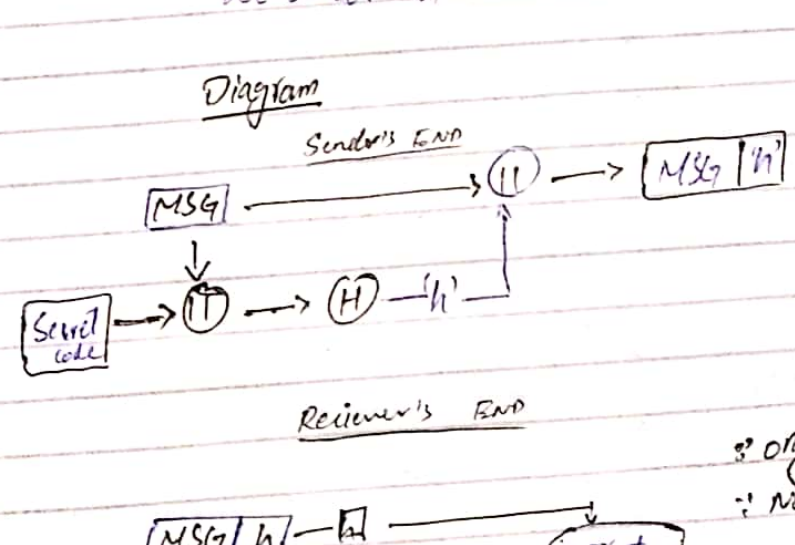
Method-5

In this message we use a secret code.
This **secret code** is known by both Sender and reciever.

The sender appends the message with this **secret code** and then pass it to Hash function H
The resultant hash cod 'h' is appended with orignal msg and sent forward.

The reciever recieves it and appends msg with **secret code**, pass to Hash function H and compare resultant hash code with recieved 'h'

Diagram

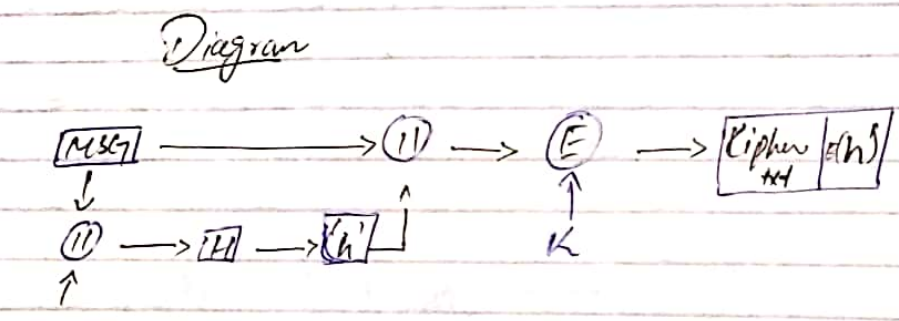


∴ only authentication
∴ No confidentility

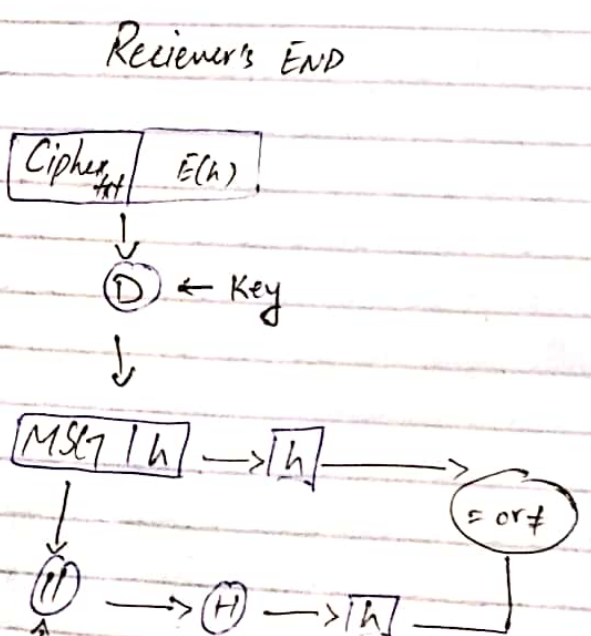
Method-6

This method is similar to previous method but in this method we also achieve confidentiality by inducing an extra Encryption method after appending MSG and hash code.

Diagram



Reciever's END



* Both integrity and confidentiality is achieved

Fire Walls

Set of rules and policies that are required for accessing data after passing through Firewall.

=> A network device / software / program / rules

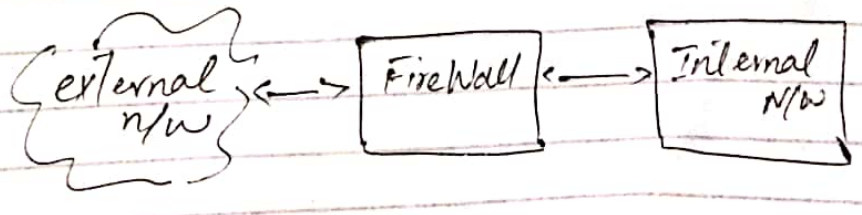
=> Can be a hardware or software device

=> All data passes through Firewall to prevent unauthorized access

=> After examining, the Firewall either blocks or allows the data to pass through it

i.e only authorized traffic (defined by the local security policy) will be allowed to pass

=> Firewall establishes a barrier b/w secure internal n/w and Outside untrusted n/w (like internet)



Definition:

A firewall is a n/w security system that monitors and controls incoming and outgoing n/w traffic based on predefined security rules.

Types

(i) Packet Filtering Firewalls / Routers
(ii) Application level gateway
(iii) Circuit level gateways

Packet Filtering Firewalls:

=> Applies a set of rules to each incoming IP-packet and then forwards or discards the packet

=> Rules are based on

source IP, destination IP address protocols and ports etc

=> If the rules matches then the corresponding action will be taken against the request

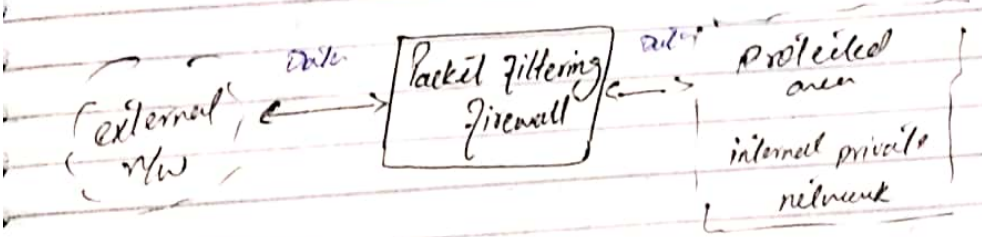
=> Otherwise, default action will be carried out (ie discard or pre-defined)

=> It analysis the traffic at transport layer & network

=> The firewall maints a filtering table which decides the packet will be discarded or forwarded

=> Filtering table filters the incoming request acc to the table and provides access accordingly

e.g. if source port = 4070 then accept if else deny request



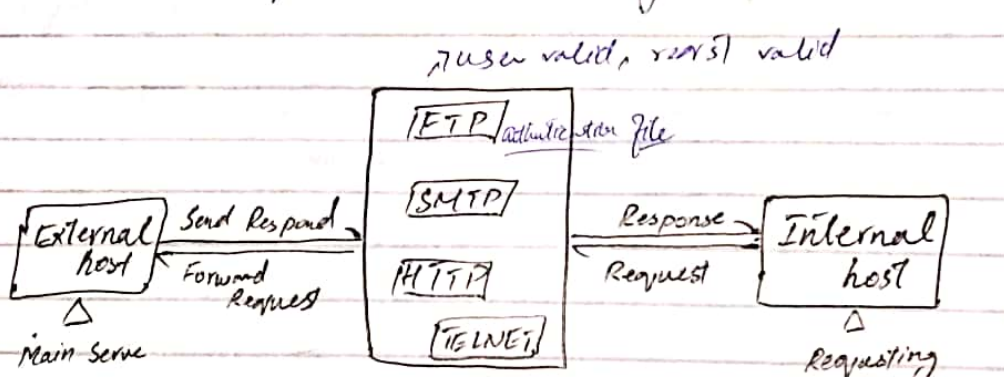
=> Simple but less Secure.

2. Application lvl gateways

=> Also called proxy servers
=> More secure than packet filtering layer
=> connects users using TCP/IP applications like (FTP, SMTP, HTTP etc) & authentication credentials
=> Our system sends request to main servers, the request is passed to proxy server first which lies blw our system and main system. The proxy server checks the request if it matches protocols and rules.

=> after confirming authenticated request, the request is passed to main server for request response.

=> The main server send respond to proxy server according to the request.
Proxy server checks the respond from main server either it follows protocols or not and transfer/forward response to the system



Proxy Server

∵ checks requests and respond from both sides and forward if follows protocols else denies request

∵ Processing overhead bcz of continues checking

=> prevents DOS attack
=> Only allowing by valid authenticated users

Circuit-lvl-Gateway

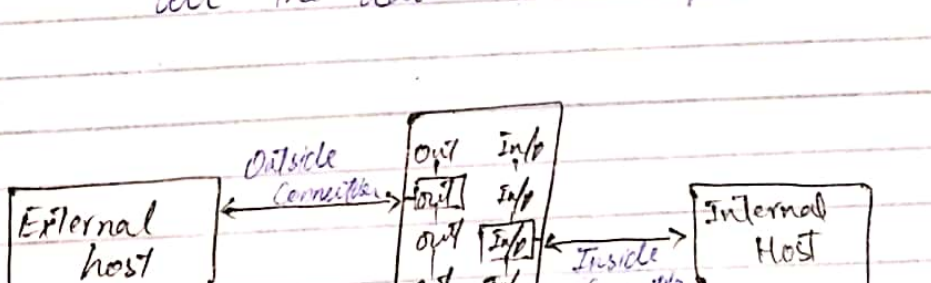
=> used two TCP connections

(i) b/w internal host and Gateway
(ii) b/w external host and Gateway

=> establishes a sure connection b/w internal host, external host with gateway before processing requests and data

=> In this way, our overhead processing cost is reduced.

=> Security checks are done before setting up a connection. Once the connection is established, all the data will be passed.



Circuit-lvl-Gateway

=> If connection requests follows protocols then connection is established

=> Faster than application lvl gateways / firewalls bcz there are less evaluations.

Intrusion Detection System (IDS)

Intruder:

An intruder is an unauthorized person who is trying to get access to the system

⇒ with criminal intentions
steal data
imbalance environment

Outside Intruder:

also called Masquerade

↳ has not authorized access to system
↳ steal credentials to access system

Inside Intruder:

also called Misfeasor

↳ already have some authorized access with some restrictions
↳ misuse the resources of company
↳ leaks info
↳ more harmful
↳ unable to identify

Intrusion:

Unauthorized access to the system and use of resources
↳ it is the task carried out by Intruder

Intrusion Detection System:

↳ Continuous monitors system
↳ " " data traffic
↳ checks if any unauthorized acces to system or suspicious activity
↳ if detects informs the system admin
↳ continously runs in the system backgr

↳ detects by checking if the activity is following the policies, methods that are pre defined by administrator

Intrusion Detection System Methods

Two types of detection system methods

(i) Signature Based IDS

(ii) Anomaly Based IDS

(i) Signature Based IDS:

↳ Pattern
search data patterns in data packets

↳ DB of attack patterns
it creates data base of already know attack patterns that have been carried out checks every data packet

↳ Detect know Packet
signature based IDS checks each data packet and compares it with data packets in data base
It detects attack if data pattern matches with some patter in already known data base

↳ Cannot identify new attacks
A new attack pattern can't be detected because it don't match any predefined pattern in Data Base

(ii) Anomaly Based IDS

Continuous detect system and give alert on any deviation

↳ Deviation

Change/deviation in the normal behaviour of some personal or system accessing resources

IDS Types

NIDS	HIDS
Network Based IDS	Host Based IDS
↳ Network Based	↳ Host Based
↳ ~~Analysis~~: Monitors, Captures and analyze network traffic	↳ HIDS are installed on host or end devices on network
↳ Network traffic = data packet	↳ HIDS continously monitor data packets from the devices only
↳ continously monitoring	
↳ Detect malicious data present into packets	
↳ checks their patterns, rules, behaviours	↳ In case, it detects some suspicious activity, it alerts the admin
↳ If finds some malicious data, NIDS captures it and analyze data packet	↳ Method to detect uses snapshot
↳ Analysis: includes matching data packet with library of known attacks	Existing system ⇌ Previous system

↳ attack is detected before it intrudes and damages the system

↳ Analysis is very difficult in busy networks

↳ Slow detection

↳ HIDS takes snapshots of existing and previous system.

=> If both system matches then there is no intrusion.

=> If both system's snapshot doesn't matches then there is some intrusion

=> Some fiels are modified and some suspicious activity is detected so HIDS alerts the system.

Malwares

Advanced Presistent Threats (APTs)

Cyber crime, directed to some business organization and to some political targets

uses vide variety of intrusion technologies and malwares,

applied prisistently to target over an extended time

combination of 3 words

Advanced: wide variety of technology, malwares and custom build malwares and attack

Presistent: Target system is attacked continously until the goal is achieved

Threat: give threats to selected targets as a result of the organized and well-funded attacks carried out to specifically chosen target.

Adware:

Includes advertising that is integrated into software. It can result in pop-up ads or redirection of a browser to a commercial site.

Attack kits:

Set of tools for generating new malware automatically using a variety of supplied propagation and payload mechanism.

Autorooter:
tools to break into new machines

Backdoor:
→ attack that bypass a normal security check, and allows unauthorized access

Downloaders:
code that installs other malwares on the target system
⇒ First insert into system then this code imports large malware packages

Drive by Download:
attack on a website that through exploits vulnerability to attack a client system when the site is viewed.

Exploits:
Code specific to a single vulnerability or set of vulnerabilities.

Floddors:
(DOS) large amount of data is requested to attack on networked computer system by carrying out some form of DOS attack

Keyloggers:
Captures keys stored on system

Logic bomb:
code inserted in malware by intruder

Symmetric Cryptography

Symmetric key encryption / cryptography is also called public-key cryptography.

=> It uses only one key for encryption and decryption process

=> In asymmetric key cryptography two keys (public key and private key) are used for both encryption and decryption process

= The most popular symmetric-key system is Data Encryption Standard (DES)

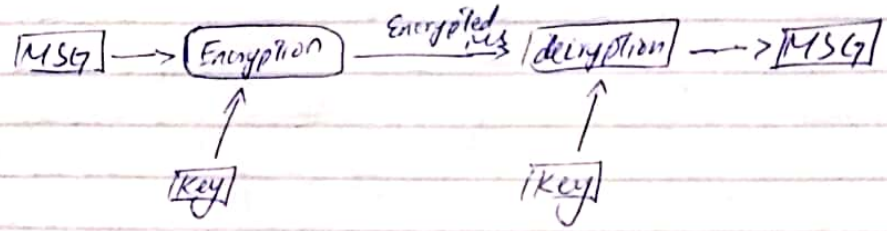
↳ Include 5 ingredients
(i) Plain txt: refers to msg or data
(ii) Encryption Algo: procedures and steps applied on plain txt
(iii) Secret key: input to algo, substitution & transformation on txt based on this
(iv) Cipher txt: encrypted, scrambled txt, output of algo
(v) Decryption Algo: cipher txt $\xrightarrow{\text{if(key)}}$ MSG

Cryptography

"Method of protecting information and communication through the use of codes so that only those for whom the info is intended can read and process it"

or

"The process of protecting info by transforming it into an unreadable form"



Encryption: process of transforming info into unreadable form

Decryption: " " " " readable form

Key: String of bits used by algo for encryption, decryption process

Types

(i) Symmetric Key Cryptography: => use only 1 key for encryption & decryption, eg DES
Also called secret key or private key cryptography

(ii) Asymmetric Key Cryptography
=> uses 2 keys for Encryption and decryption. Also called public key cryptography

2 keys -> Public key: known to everyone
-> Private key: know to that particular person.

=> A message that is encrypted using a private key can only be decrypted by using a public key and vice versa

popular asymmetric key algo

RSA, DSA, ---

(iii) Hash Function

Msg of any length is encrypted or converted into hash code of some fixed length by using some Hash Function.

Also called msg digest

Symmetric key cryptography	Asymmetric cryptography
(i) public key cryptography	(i) private key cryptography
(ii) only used 1 key	(ii) used 2 key
(iii) used public key	(iii) uses public and private key
(iv) faster in execution	(iv) slower than symmetric
(v) less complex, less computational power	(v) more complex, more computational power
(vi) used for bulk data (ie faster)	(vi) used for securely exchanging secret key.

Disadvantages:
sharing same key b/w the sender and reciever is not safe

No problem of key shring b/w sender and reciever b/c private key concept

Common Algos:
DES, AES, 2DES, 3DES

Common Algo
RSA, DSA etc

Symmetric Encryption Principles

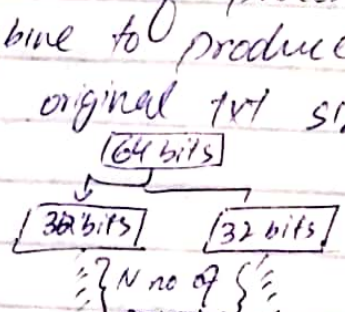
Feistel Cipher Structure

=> Most of the block cipher technique follows this structure

=> plain txt is divided into 2 equal parts ie Lo and Ro e.g 64 bit → 32 bit + 32 bits

=> the both 2 halves of data passes through N number of processing rounds and then combine to produce the cipher txt block of original txt size

i.e

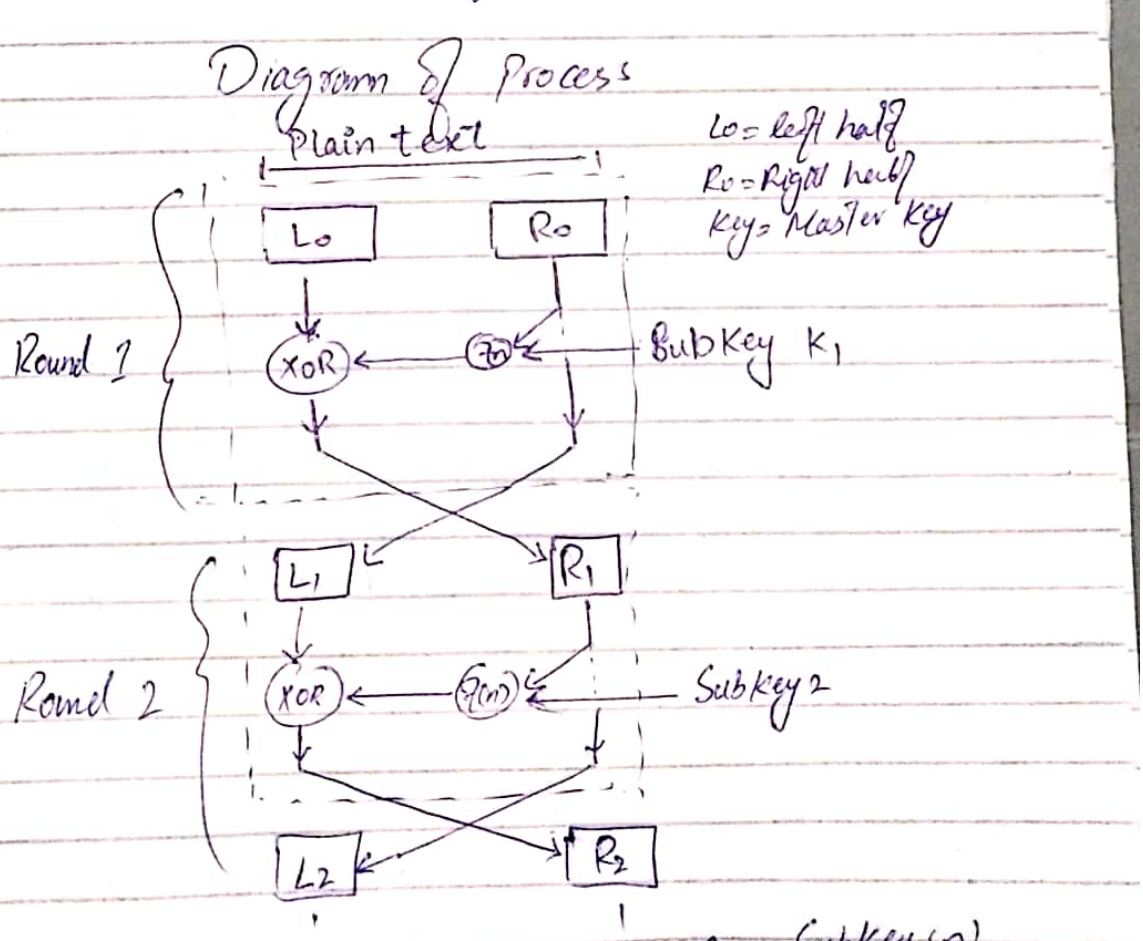


=> On the right half R_0 we apply a function and in the function we will use a subkey generated from the master key.
The no of subkeys generated from the master key will be equal to the number of processing rounds

=> The o/p of this function is XORed with the left half L_0 and then their o/p will be swapped

This is single round.

Diagram of Process



=> We will have n-rounds → depending upon the algorithm carried out on plain txt

=> All rounds will have same str.

=> Terms

Block size → Larger block size, more secure
size of txt

Key size → Larger key means more secure encryption but decreases processing speed

no of rounds → times process carried out
more round means → more secure

Subkey generation → more complex subkeys algo make difficult for attackes to steal data

Functions → more complex f(n), more secure data and difficult to attack

DES

⇒ Data Encryption Standard

=> DES is an a symmetric cryptography

(i) DES is a block cipher

(ii) Symmetric cipher
(use same key for encryption + decryption)

(iii) Process a 64 bit plain txt block
It encrypts the data in blocks of size 64 bits each

(iv) It processe 16 rounds
and each round is a Feistel cycle round.

Steps

(i) Initial Permutation e.g. ABC ← ABC, BAC, ACB, ...

(ii) 16 Feistel rounds

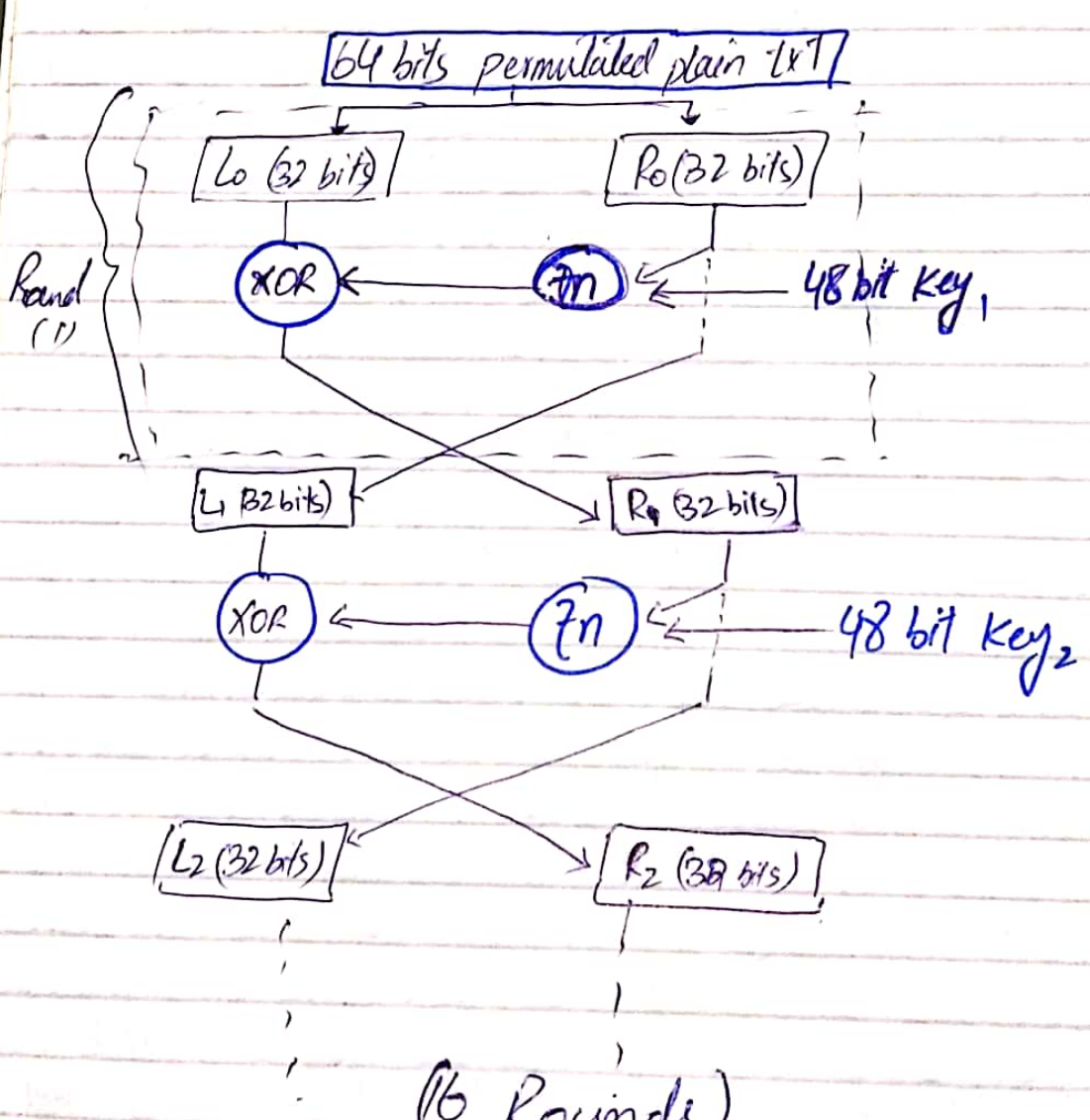
(iii) Swapping / left right swap

(iv) Final permutation / inverse initial permutation

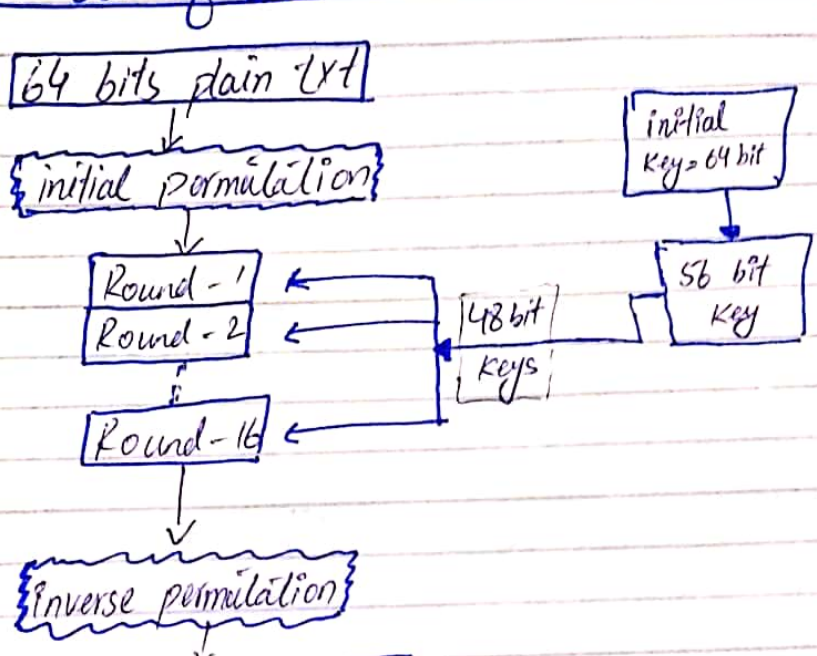
Key

64 bit key is used as master key
It is converted in 56 bit key and
then further in 48 bits subkey

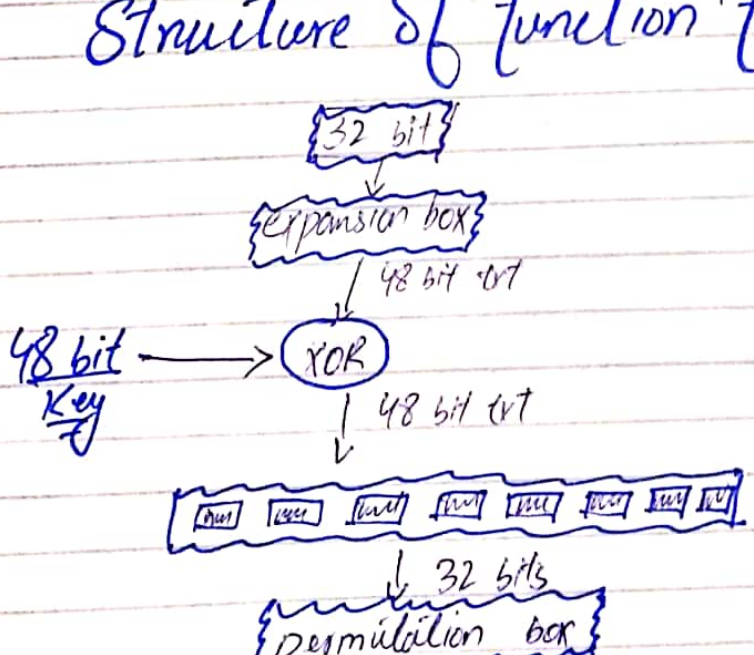
Feistel rondss



Basic Str of DES



Structure of function 'fn'



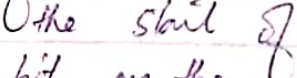
S boxes
↳ substitution boxes
↳ 4 bits each
↳ each box contains 6 bits data

=> 32 bits of data plain txt is passed through expansion box which converts it into 48 bits.
Then this 48 bits is XORed with 48 bit key.
Then this 48 bits are passed to substitution boxes that converts it into 32 bits

What happens in expansion box?

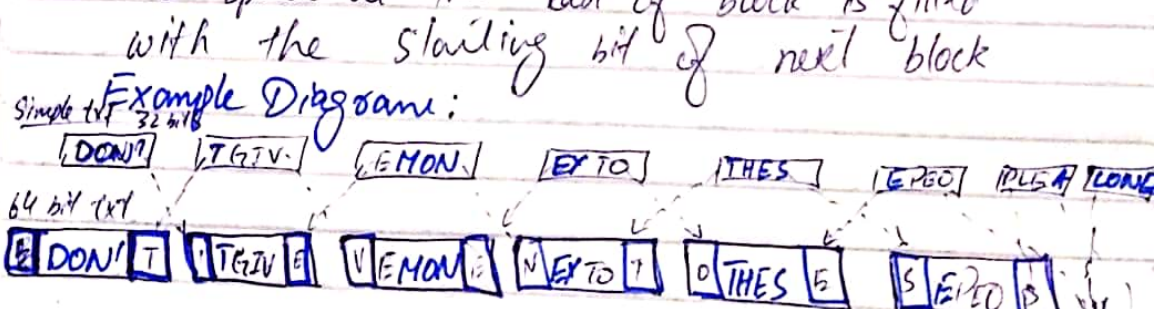
Expansion box converts 32 bits txt into 48 bits so that it can be XORed with 48 bits key.

The permuted txt is in the form of 32 bit i.e
8 blocks containing 4 bits each.
So to convert it into 48 bit we induce extra 2 bits space in the start and end of each blocks.
The space at the start of block is filled by the the bit on the last of previous block e.g



The space at the last of block is filled with the starting bit of next block

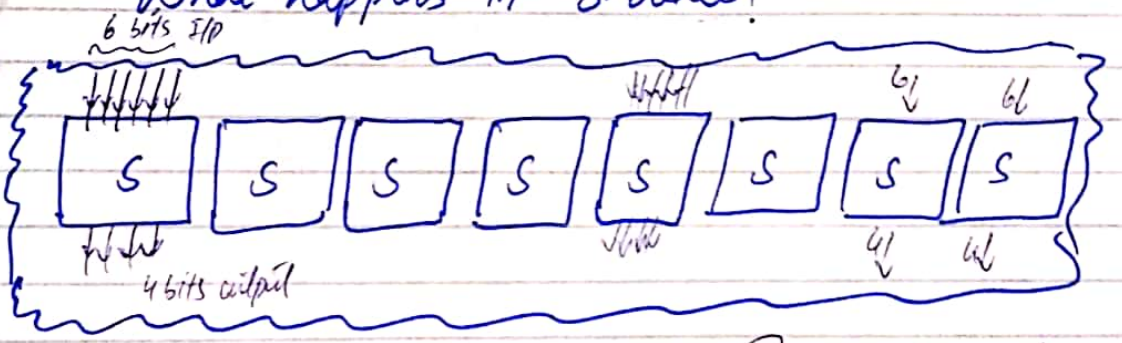
Simple txt Example Diagram:
32 bit



Now this 48 bit txt can be XORed with 48 bit Key

After this process the 48 bits txt is processed through S-boxes that converts 48 bits txt into 32 bit

What happens in S-boxes?



S-boxes takes 8 inputs of 6 bits each and generates output of 8 blocks combining 4 bits each

Input = 8 x 6 bits = 48 bits

Output = 8 x 4 bits = 32 bits

How 6-bits are converted into 4 bits

6 bits are converted into 4 bits in S-boxes
The first bit and last bit in 6 bit block represents the row number in combined form

The 4 bits in the centre represents the column number of table

The resultant element in table searched by row and colum is then converted into 4 bit binary number which is the desired output.

eg

6 bit binary number as input of S-block

0 0 1 0 1 0

01 represents row number i.e 00->0, 01->1, 10->2, 11->3

So row is 1

0101 represents colum number 5 i.e 0000->0, 0001->1, 0010->2, 0011->3, 0100->4, 0101->5

So the desired output will be the 4 digit binary number representing the element in row #1 and column #5

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

So the resultant 4 bit binary output will be 4 bit binary number of 7
i.e

0111

I/P 6 bits = 001011

O/P 4 bits = 0111

Each S-box has a different table

Key Generation in DES

The original master key is of 64 bits. It is first converted into a 56 bits key
And then it is further converted into 48 bits keys for each round
ie
16 rounds means 16 diff 48 bits keys

How 64 bits keys is converted in 56 bits and then into 48 bits keys

⇒ 64 bit key is passed through PC-1 (permuted-choice 1)

inside PC-1 64 bits key is divided into 8 blocks each containing 8 bits

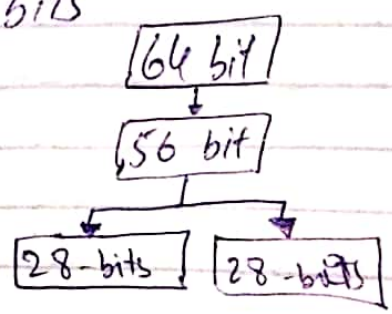
[12345678] [9 10 11 12 13 14 15] [16 17 18 19 20 21 22] . . . .

For each pair of PC-1, the last bit is discarded

ie [1 2 3 4 5 6 7 (8)] ↳ discarded

So after discarding last bit from every block, we have remaing 8 blocks each containing 7 bits so 64 bit key is converted into 56 bit key.

So now this 56 bit key is further divided into two halves C_0 and D_0 each containing 28 bits



=> Now at each round the C_0 and D_0 keys are passed through left shift process

↳ At some rounds, only 1 bit is shifted

i.e

at round i = 1, 2, 9, 16 only 1 bit is rotated left by 1 bit shifted

↳ At some rounds, 2 bits are shifted

i.e

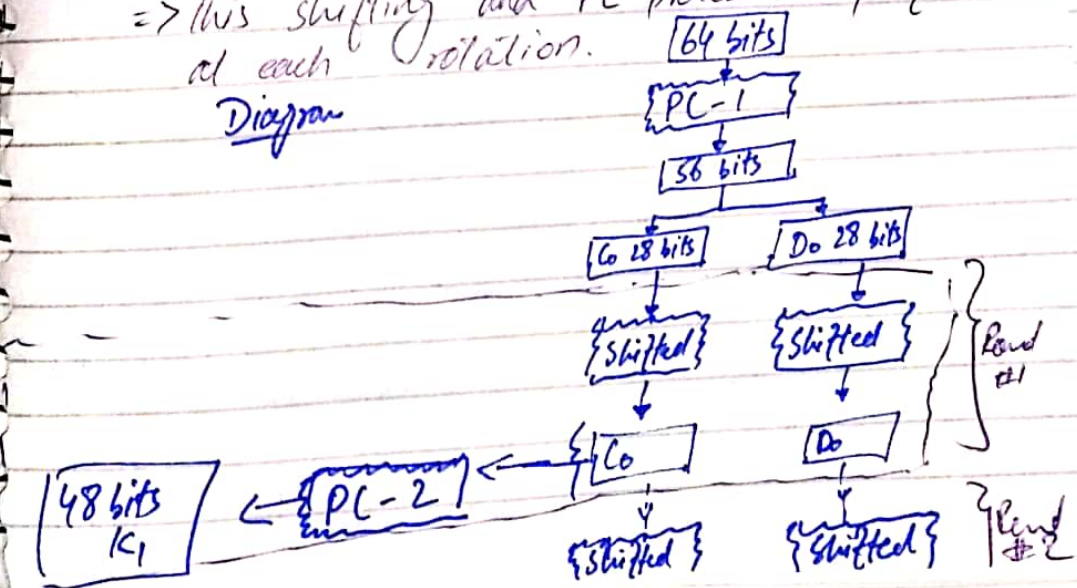
at round i = 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15

2 bits are shifted or roted left by 2 bits

=> After shifting the C_0 and D_0 are again passed through some PC-2 function and convert this 56 bits keys into 48 bits key by apply some permuted-choice 2 procedure.

=> This shifting and PC process are performed at each rotation.

Diagram



DES-Analysis

Properties

Avalanche effect:

It means a small change in plain text (or key) should create a significant change in the cipher text

=> DES proved to be strong with regard to this property

e.g

Plaintext : 0000000000000000

Ciphertext : 4798FA6216E3...

Plaintext : 0000000000000001

Ciphertext : 5869DBC1017AG

Completeness effect:

It means, that each bit of the cipher text needs to be depend on many bits on the plain txt.

=> The confusion and defusion produced by D-boxes and S-boxes in DES, show a very strong completeness effect

AES

(Advanced Encryption Standard)

=> AES is a symmetric key cryptography

=> 2 blocks cipher
processes are carried out on the blocks of text

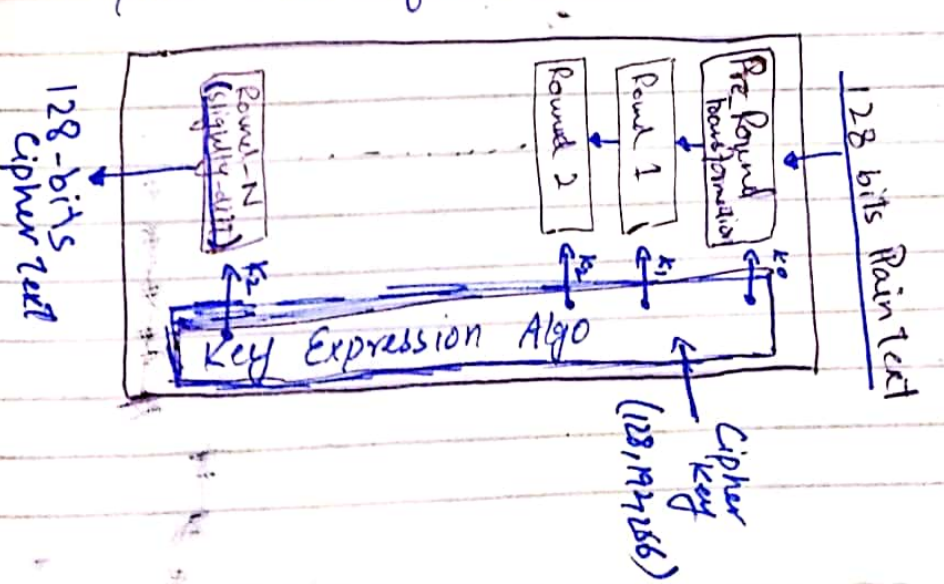
=> Formed in 2001 by the US NSIT
(National Institute of Standard and Technology)

=> Fixed block size
e.g. 128 bits = 16 byte = 4 words (* 1 word = 32 bits = 4 bytes)

=> The no of bits in key depends upon the no of rounds
i.e

Rounds	No of bits in key	AES-Version
10	128	AES-128 Version
12	192	AES -192 Version
14	256	AES – 256 Version

=> General Str. of AES

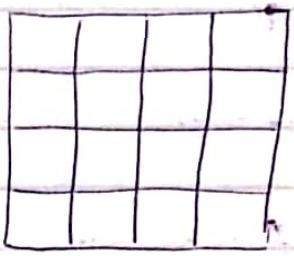


=> Cipher key of (128, 192, 256) bits is passed through Key expression algorithm

=> Key Expression Algorithm generates 'n+1' no of keys
∴ n = no of round
1 extra key for Pre-Round Transformation

=> After Nth Round the txt is converted into 128 bits of cipher text

1 bit = 0 or 1
1 byte = 8 bits
1 word = 4 bytes
Block size =
State
↳ matrix of 4x4
↳ 16 bytes = 4x4
↳ stores intermediate result of input array
↳ Input array of 4x4



i.e
4x4 = 16 byte = 128 bits
= 4 words

=> State Array

S(0,0)	S(0,1)	S(0,2)	S(0,3)
S(1,0)	S(1,1)	S(1,2)	S(1,3)
S(2,0)	S(2,1)	S(2,2)	S(2,3)
S(3,0)	S(3,1)	S(3,2)	S(3,3)

Eg
∴ S(0,0)
↳ Representing word #
↳ Representing Byte #

e.g
S(1,3)
-this means that
byte # 1 or word #3
or
1st byte of 3rd word

=> It stores intermediate result of steps.

Key

No of bits in key depends upon no of rounds

e.g 12 rounds

so no of bits in key = 128 bits

which is = 16 byte = 4 words

In matrix form

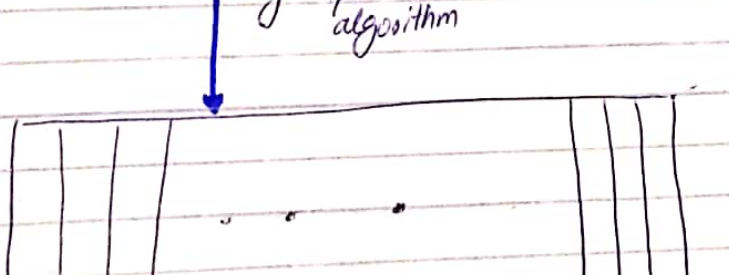
Key

W_0	W_1	W_2	W_3
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_{16}

$\therefore W_0 .. W_3$ are words

$\therefore K_0 = 1$ byte

Key expension algorithm



4 words key generates 44 words after passing through Key expension algorithm.

=> Encryption: plain txt into cipher text

=> Decryption: cipher txt into plain text OR reverse cipher

=> Round keys are applied in reverse order

=> Much stronger than 2DES and 3DES

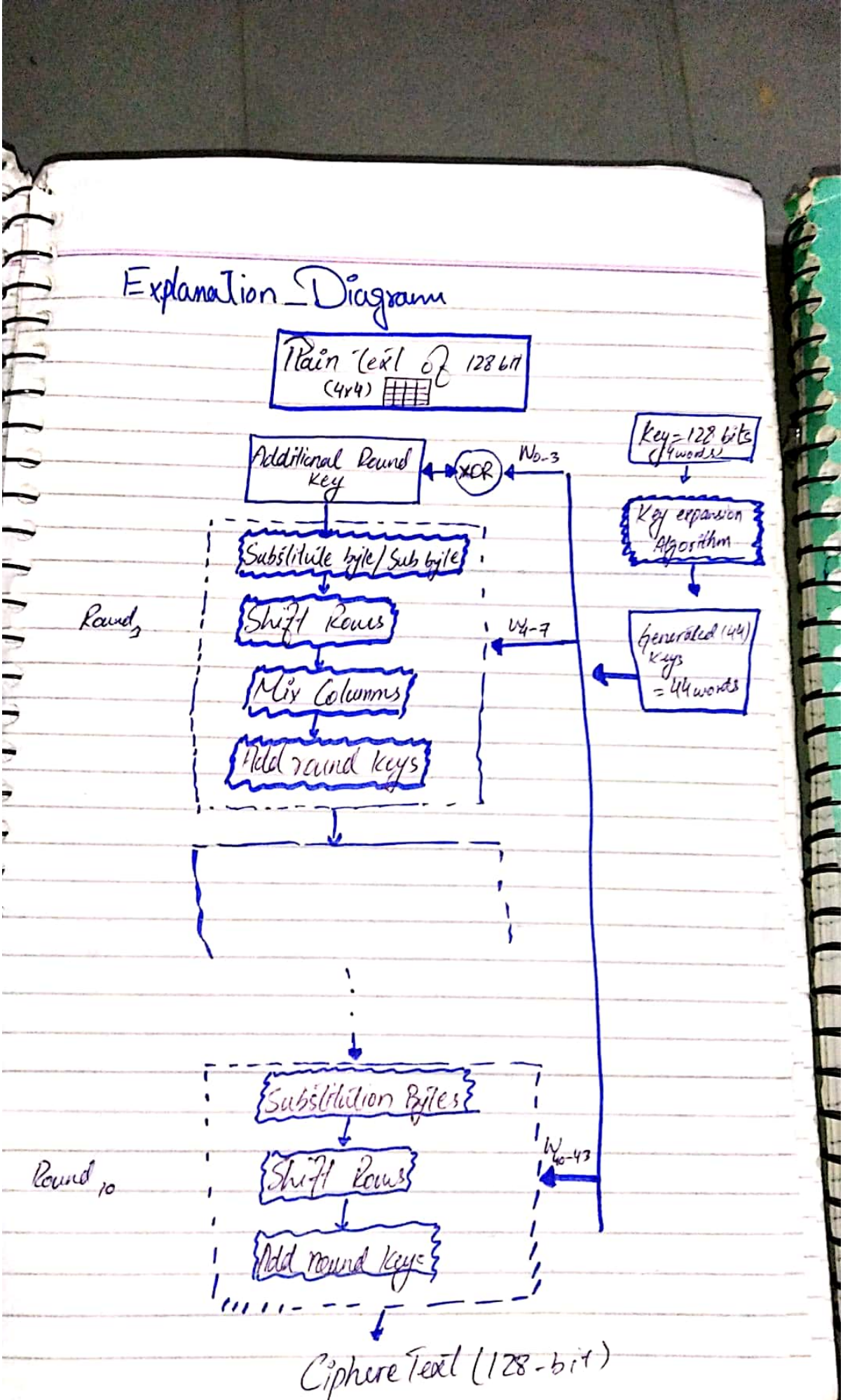
=> Structure of Each Round

Eeach round except the last round, contains 4 transformation

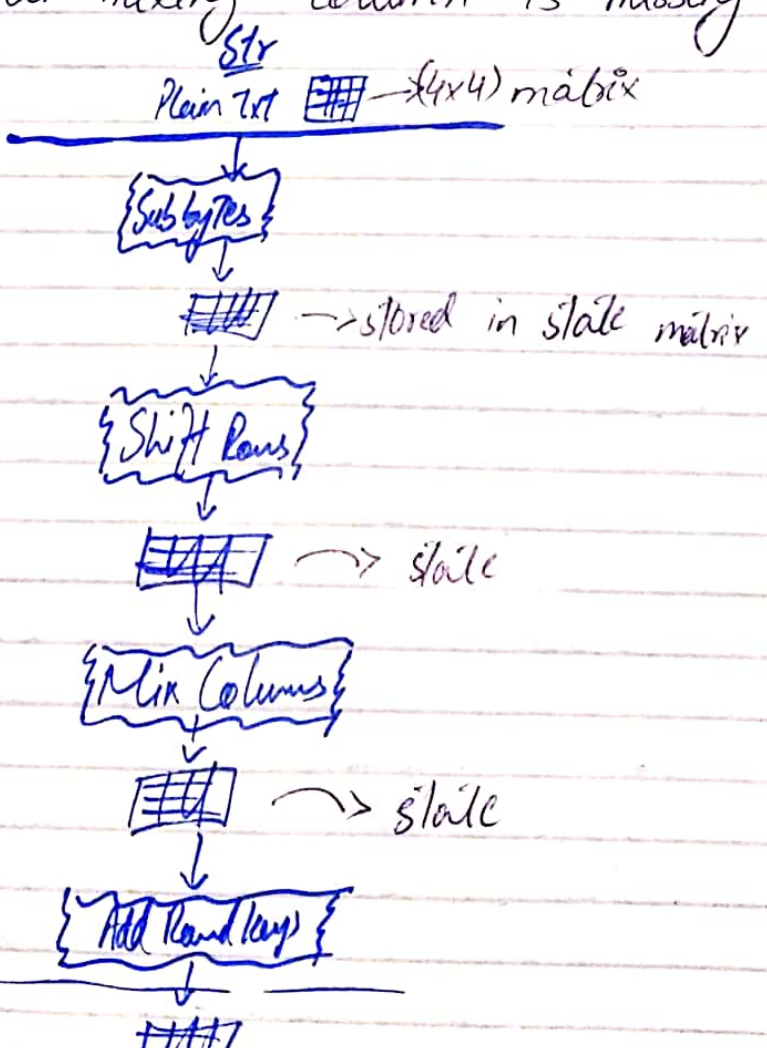
i.e

- Substitute Bytes
- Shift Rows
- Mix Columns
- Add Round keys

The last round don't contain Mix Columns transformation



=> Plain text is passed through Additional round which simply xored 4 words key with 4 words plain text
=> Total 44 words keys are generated by key expansion algorithm.
=> First 4 words are used in First Additional round
=> Remaining 40 words keys are used in 10 rounds, each round utilizes 4 word key.
=> Each round contains 4 steps
=> Only the last round contain 3 steps in which mixing column is missing



TRANSFORMATIONS

→ 4 types

Substitution → Sub bytes
permutation → shifting rows
mixing → Mixing columns
key-adding → Round key adding

(i) Substitution:

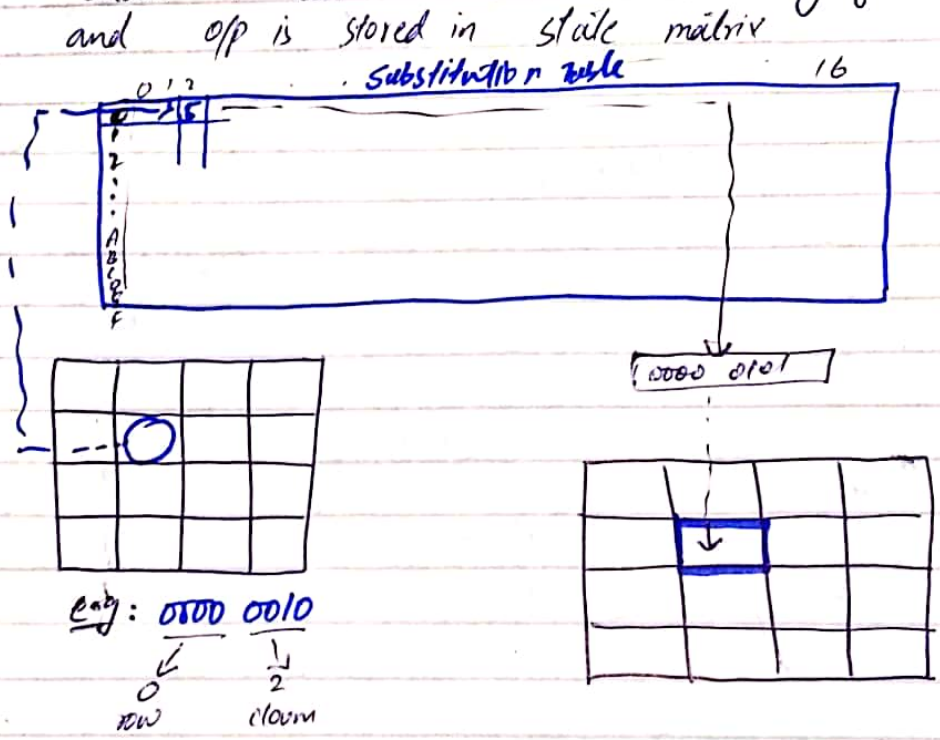
AES like DES uses substitution.
The mechanism is different.
Substitution is done for each byte in 4×4 matrix
Only one table is used for transformation of bytes.
which means if 2 bytes in 4×4 is same, the transformation is also same.

Sub-byte

At encryption side,
the hexadecimal byte is divided into two halves
The first half refers rows
The 2nd half refers columns

eg 0000 0010
↓ rows ↓ columns

The rows and columns generated from I/p 4x4 is searched in substitution table and the value corresponding to that rows and columns is converted back into binary form and o/p is stored in state matrix



So the element on 0th row and 2nd column is 5.

We convert this 5 in binary number of 8 bits and write the 1 byte in state matrix in same corresponding row and column

The element is 0
Its 8 bit binary number would be
(0000 0101)

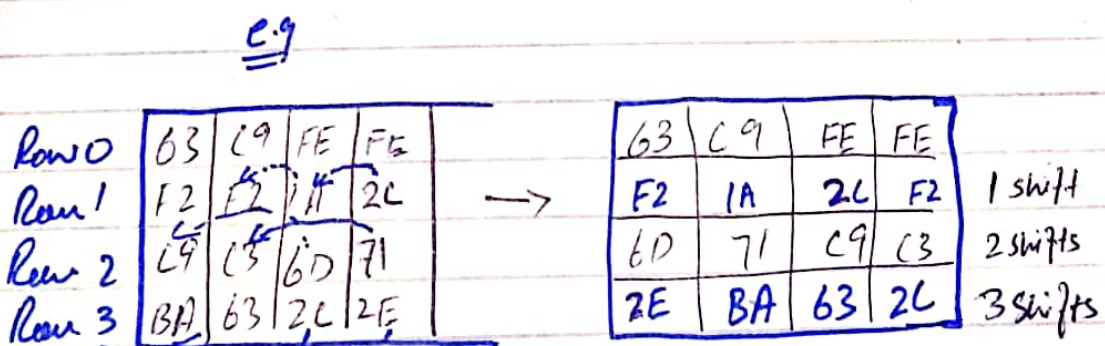
2- Permutation

Permutation is left shifting of rows
In DES, we do shifting on bits level.
In AES, we do shifting on bytes level.

Shifting Rows

Shifting is done to the left
No of shifts depends upon no of row
i.e Row 0 → No shifts
Row 1 → 1 time shift to left
Row 2 → 2 " " "
Row 3 → 3 " " "

e.g



3- Mixing

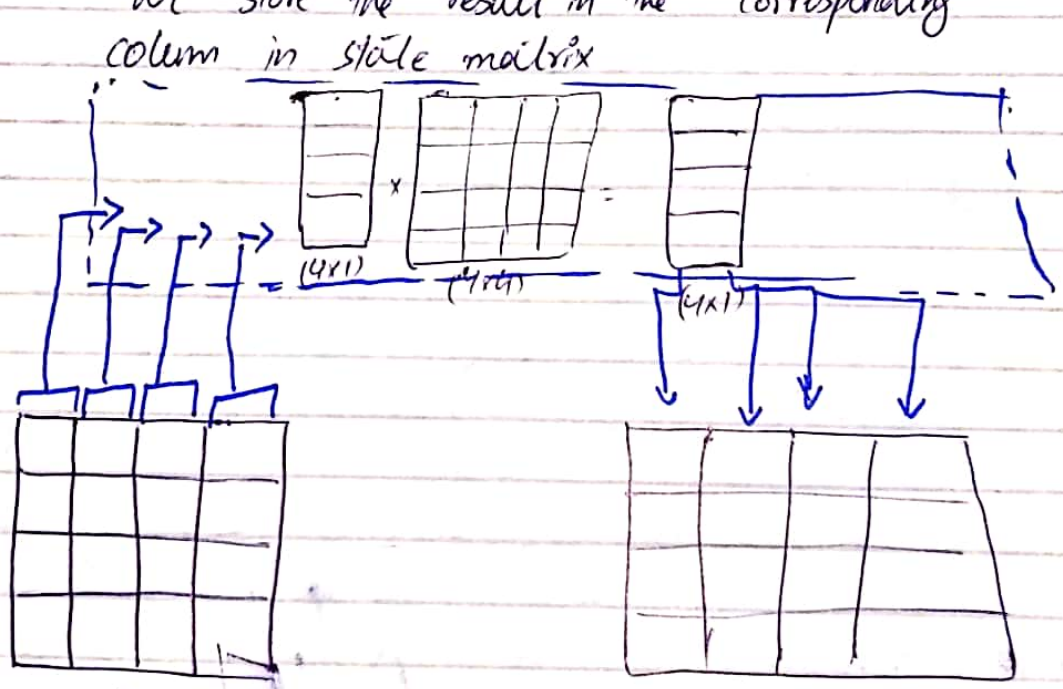
MixColumns

for encryption process

We take each column of state matrix and multiply it with a constant matrix

(4x4) x (4x1) so our result will be in (4x1) form

We store the result in the corresponding colum in state matrix



I/P from state matrix

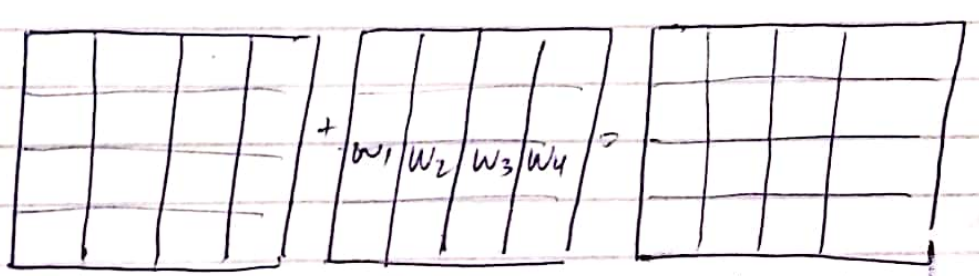
O/P from State Matrix

4. Key Adding

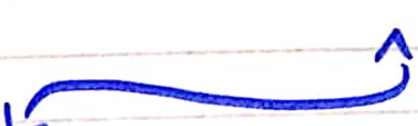
Add Round Key

XOR

Now we added the o/p from state matrix and 4 word key each colum is XORed with 1 word



=> The result output is considered as cipher text if it is the last rounds



Block Cipher Modes of operations.

For different types of messages we uses different modes of operations.

=> 5 diff modes of operations are

(i) ECB -> Electronic CodeBook Mode
(ii) CBC -> Ciphere Block Chain Mode
(iii) CFB -> Ciphere FeedBack Mode
(iv) OFB -> Output FeedBack Mode
(v) CTR -> Counter Mode

ECB
(Electric CodeBook Mode)

=> Simplest mode of operation
=> plain text is divided into blocks of **fixed sizes**
=> If message is not a multiple of block size, then padding is done.
=> Takes one block at a time and encrypts it
=> **Same Key** is used for both **encryption & decryption**

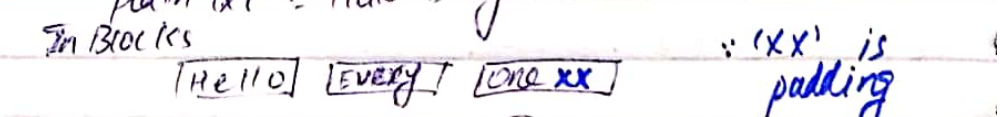
e.g
fixed block size = 5
plain txt = Hello Everyone

In Blocks

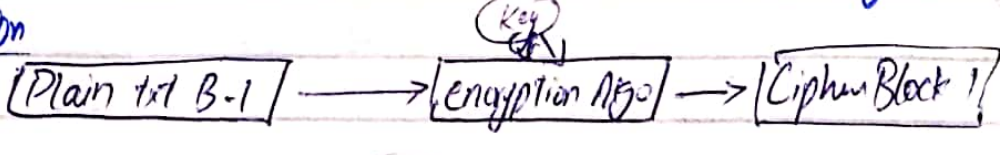
HELLO	EVERY	ONE XX

∵ 'XX' is padding

Encryption



Decryption



=> ECB

is best for short amount of data e.g key

it is not secure for lengthy data

if identical blocks appears, it generates identical ciphere txt for both

CBC ⟷

Cipher Block Chain Mode

CBC overcomes the limitations of ECB i.e

if both blocks are identical their ciphere txt will be identical

=> The first block of plain txt is XORed with **initialization vector**.

∵ **Initialization Vector:**

IV is the block of same size, containing some data.

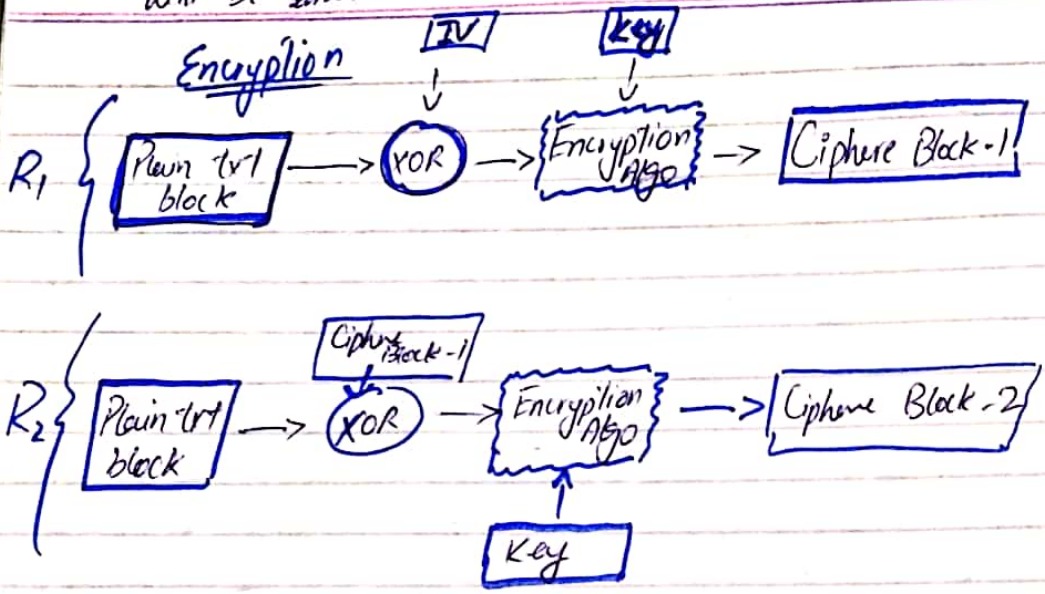
=> The I/P of next 2nd block is XORed with the output ciphere block of previous txt block.

=> Same key are used for encryption process in each round.

⇒ If we have 2 diff block, it will produce diff ciphers

Lim: If 2 identical msg, & we use same IV, cipher will be same.

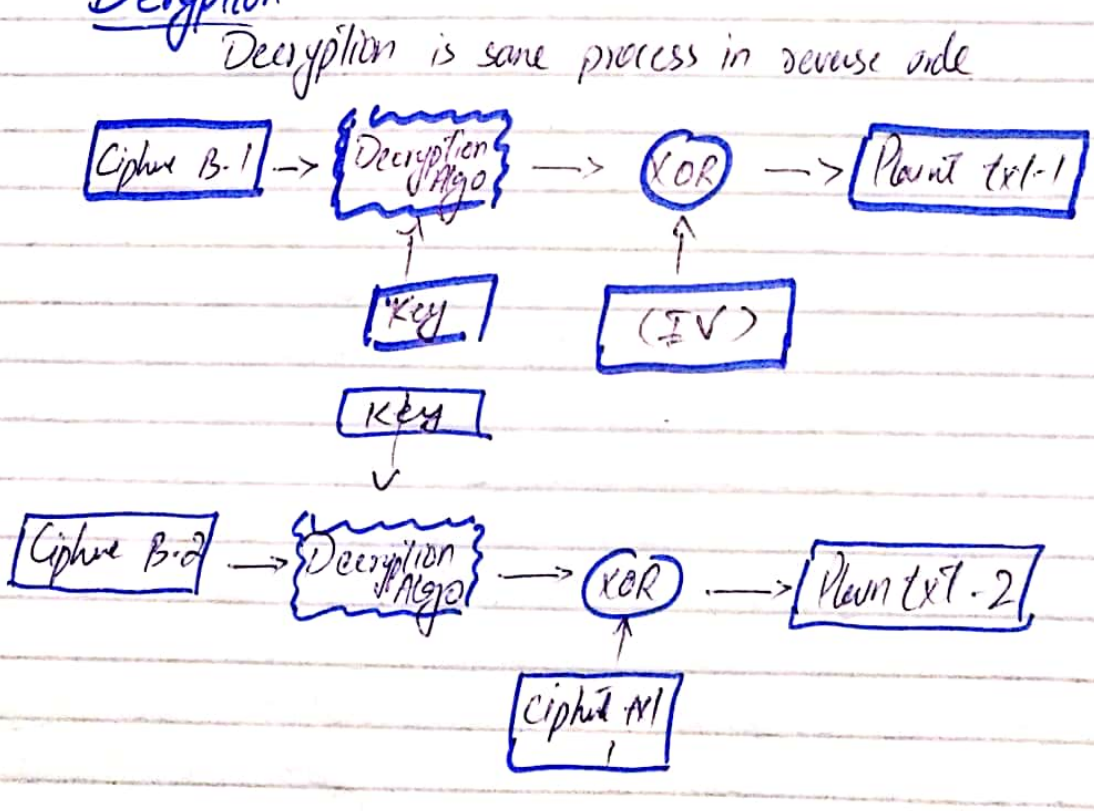
Encryption



This chain of using Cipher Blocks of previous txt block continues, That's why it is called Cipher Block Chain Mode.

Decryption

Decryption is same process in reverse side



Info Sec

Cipher Block Operation

(3) Cipher feedback mode (CFB)

In cipher feedback mode the plain txt is converted into s-bits.

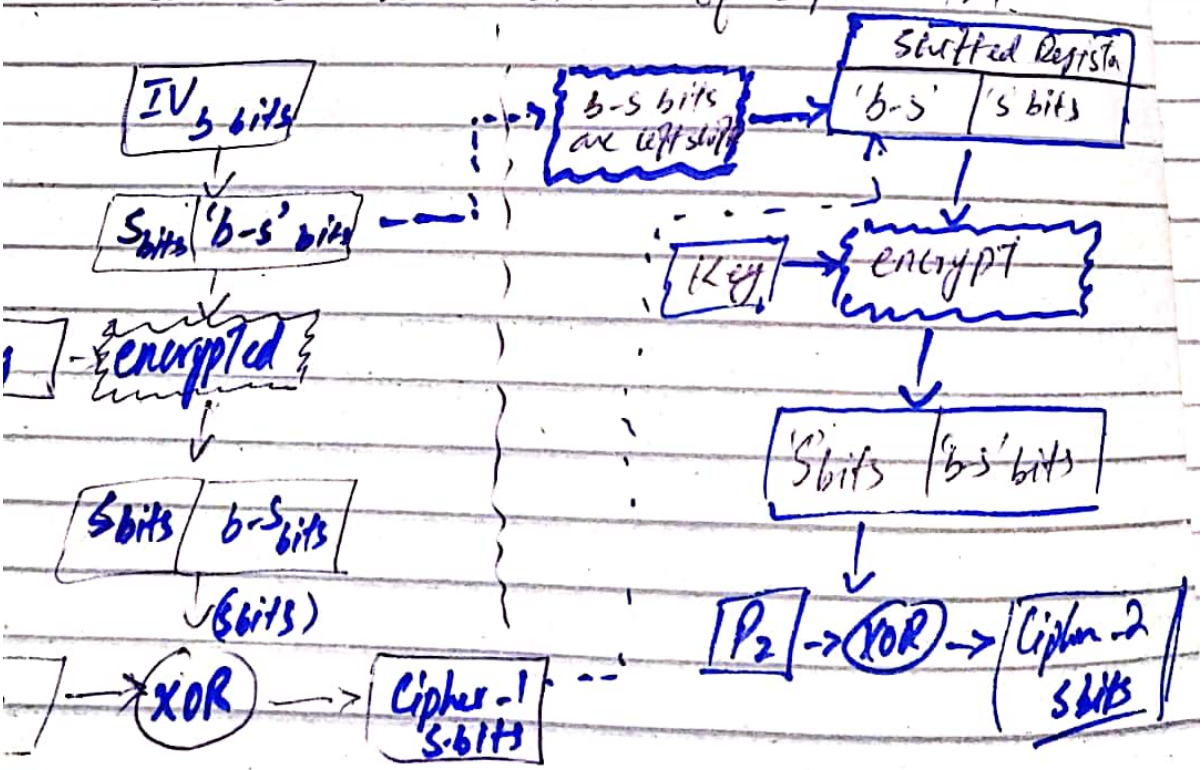
s-bits can have any numbers

The initial vector IV contain s-bits init

IV encrypted using key, than it is divided into 's' bits and 'b-s' bits.

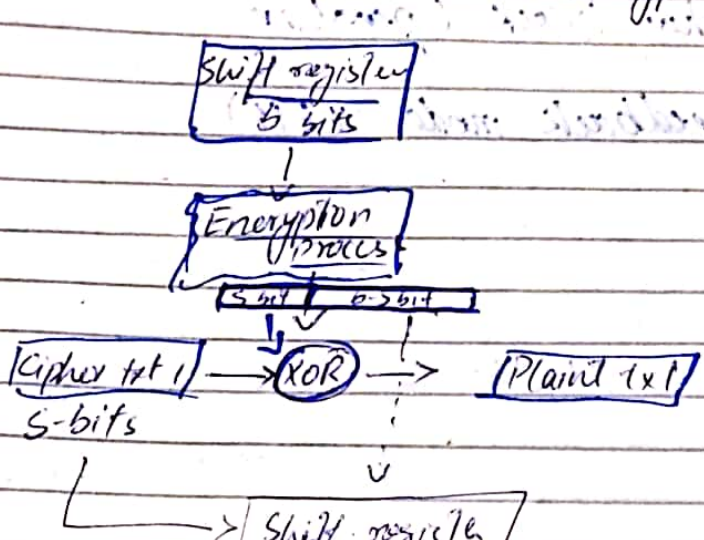
The 's' bits are XORed with plain txt

The 'b-s' bits are left shifted and combined with 's' bits of cipher txt



For decryption

same as encryption process

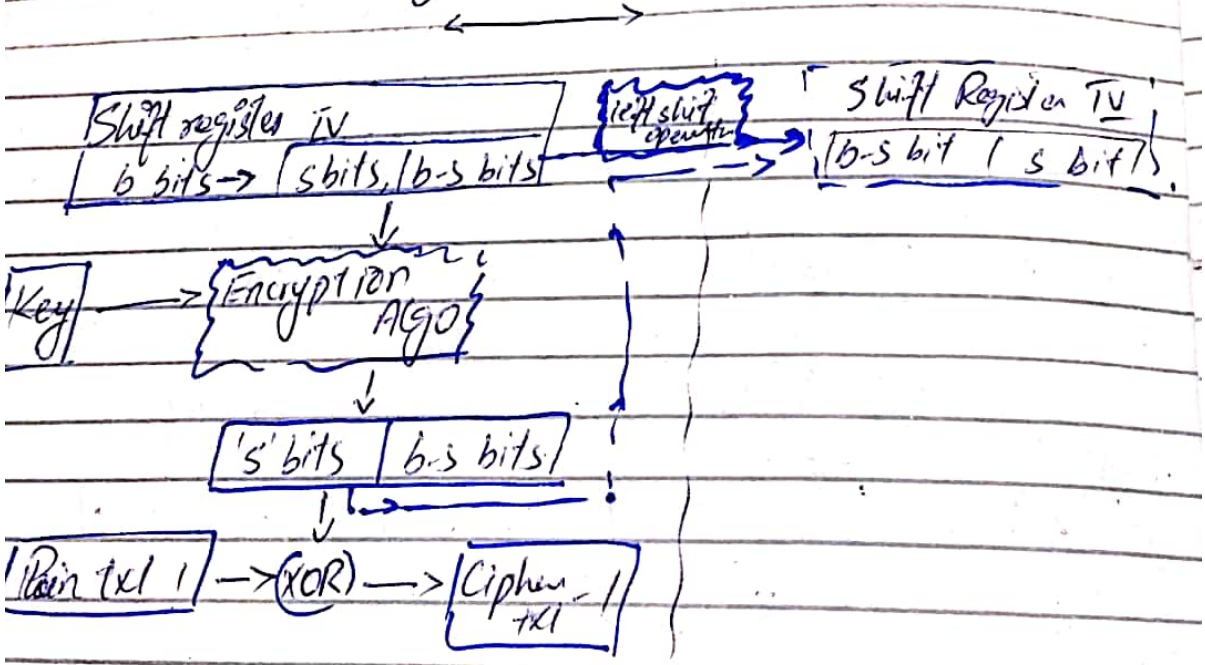


⇒ Encryption process is used for both encryption and decryption.

(4) Output FeedBack Mode

=> Similar to the structure of CFB

=> The s-bits that are used for the encryption of of next plain txt are the s-bits of ~~input vector~~ initial vector after encryption through key.



Decryption is same as encryption process

We just XOR 's' bit with cipher txt in decryption.

We use same Encryption Algorithm in decryption process.

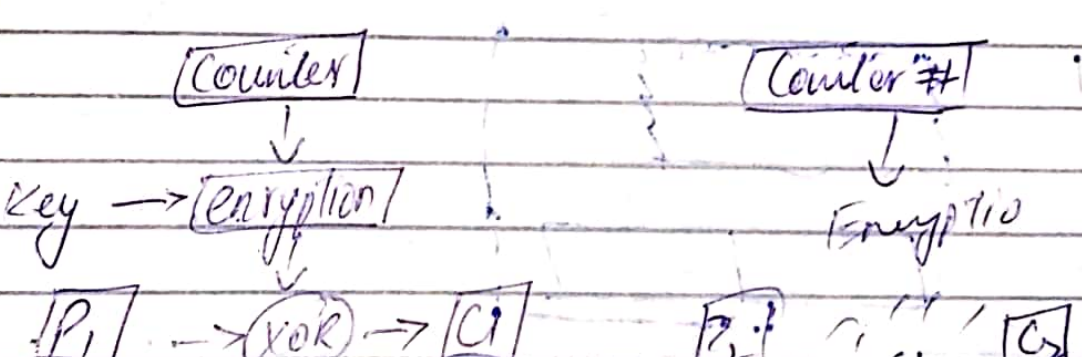
(5) Counter Mode (CTR)

=> Simple and fast

=> a counter, equal to the plain text block size is used

=> Counter is initialized to the same value and then incremented by 1 for each subsequent block

Diagram



Decryption

Same as encryption process
Uses same encryption algo
Chang P1 with C1

No chaining

RSA Algorithm

=> The acronym RSA is made from the initial letters of the surnames of:
Ron Rivest, Adi Shamir and Leonard Adleman

=> RSA algo was developed in 1978

=> It is an asymmetric algorithm using 2 keys
i.e public key and private key

=> Public Key:
Key know to all users in network

=> Private key:
Kept secret, not shared to all

=> If public key of user A is used for encryption, then we have to use the private key of same user of encryption

=> The RSA algo is a Block cipher in which the plain txt and cipher txt are integers blw 0 and n-1 for some value of 'n'

Algorithm

1- Key Generation

(i) Select 2 large prime number as 'p' and 'q'

(ii) Calculate $n = p * q$

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

$\because \phi$ = euler's totient function

(iv) Choose value of 'e'

$1 < e < \phi(n)$ and $\gcd(\phi(n), e) = 1$

(v) Calculate

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

i.e $ed \equiv \bmod \phi(n)$

$ed \bmod \phi(n) \equiv 1$

(vi) Public key = $\{e, n\}$

(vii) Private key = $\{d, n\}$

2- Encryption

$$C = M^e \bmod n$$

∵ e = value calculated in key gen method

∵ C = cipher txt

∵ M = plain txt

∵ M = no of integers/digits in plain text and it should be $M < n$

e.g Plain txt = Abhi

so $M = 4$

3- Decryption

$$M = C^d \bmod n$$

∵ M = plain txt

∵ d = calculated in key Gen Method

∵ Cipher txt

Note:

Public key $\{e, n\}$ is used in encryption process

so

Private key $\{d, n\}$ is used in decryption process.

Example

We are supposed to take some large prime numbers but for the sake of example we'll take small number

$p = 3, q = 11$

(ii) $n = p * q = 3 \times 11$

$\boxed{n = 33}$

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

$= (2) * (10)$

$\boxed{\phi(n) = 20}$

(iv) chose value of e

let $e = 7$

so $1 < 7 < 20 \Rightarrow (T)$ and $\gcd(7, 20) = 1$

(v) now

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

$de \bmod \phi = 1$

$(7 * d) \% 20 = 1$

20 → 21 / 21%20=1
40 → 41
60 → 61
80 → 81

$\therefore 21 \% 20 = 1$

so $\boxed{d = 3}$

$(7 * 3) \% 20 = 1 \Rightarrow (T)$

(vi) Private key = (d, n)

(vii) Public key = (e, n)

Encryption

$C = M^e \bmod n$

Let $M = 31$; so

$M < n$, $31 < 33 = \phi(T)$

$C = (31)^7 \bmod 33$

$\boxed{C = 4}$

Decryption

$M = C^d \bmod n$

$= 4^3 \bmod 33$

$\boxed{M = 31}$

Diffie-Hellman Key Exchange

=> It is not an encryption algorithm.
=> It is used to exchange secret keys between ...2 users.

➔ We will use a symmetric encryption to exchange the secret keys b/w users.

➔ We use this algorithm bcz when ↓ we are sending a key to reciever, transition it can be attacked in b/w. ~~algorithm~~.

Algorithm

(i) consider a prime number 'q'

(ii) select α such that it must be the primitive root of q and $\alpha < q$

α is a primitive root if

$\alpha^1 \bmod q$
$\alpha^2 \bmod q$
$\alpha^3 \bmod q$
⋮ ⋮
$\alpha^{q-1} \bmod q$

gives result within = $\{1, 2, 3, \ldots q-1\}$

Example let $q = 7$ = 5

$5^1 \bmod 7 = 4$

$5^2 \bmod 7 = 6$ $= \{1, 2, \ldots 6\}$

$5^3 \bmod 7 = 2$ =

$5^4 \bmod 7 = 3$ $\neq \{1, 2, 3, \ldots q-1\}$

$5^5 \bmod 7 = 1$

So we

can take 5 as α

and it

satisfies both condition

(i) it is primitive route of 7

(ii) $5 < 7$ ⇒ (T)

(iii)

$\therefore X \rightarrow$ private key of user

$\therefore Y \rightarrow$ public key of user

assume X_A (private key of A) and $X_A < q$.

calculate

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

(iv)

assume X_B (private key of B) & $X_B < q$

calculate

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

Now we will calculate Secretes keys

=> To calculate the secret keys both sender and receiver will use public keys

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

- K_1 = Key of user A
- X_A = Private Key of user A
- Y_B = Public " " B

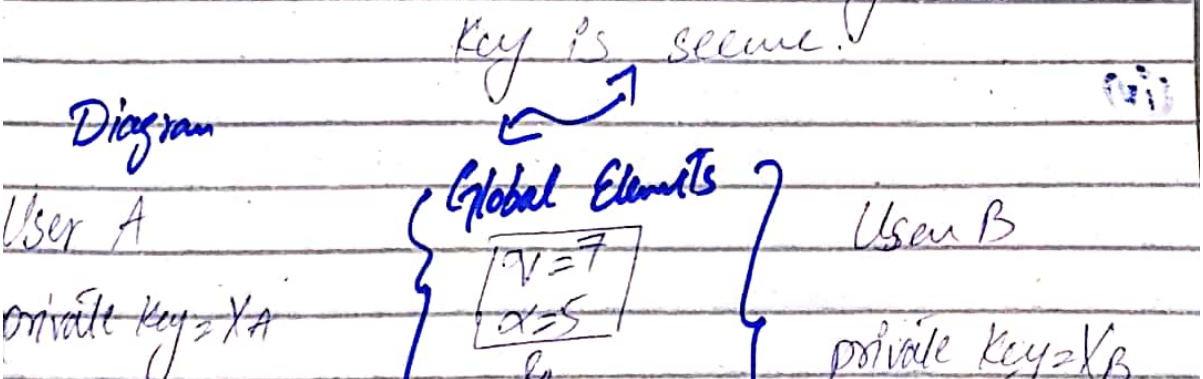
$$K_2 = (Y_A)^{Y_B} \bmod q$$

- K_2 = Key of user B
- X_B = Private Key of " "
- Y_A = Public " " " "

if $K_1 = K_2$

then we can say that Key is secure.

Diagram



Example

(i) $q = 7$

(ii) $\alpha = 5$

$\Rightarrow \alpha$ is primitive root — (1)

$\Rightarrow \alpha < q$ — (1)

$\Rightarrow$ we can also take 3 as 'α'

(iii)

Assume $X_A = 3$

so

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

$= 5^3 \bmod 7$

$Y_A = 6$

(iv)

Assume $X_B = 4$

so

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

$= 5^4 \bmod 7$

$Y_B = 2$

Secret key calculation

(P1)

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

$= 2^3 \bmod 7$

$K_1 = 1$

(P2)

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

$= 6^4 \bmod 7$

$K_2 = 1$

$\Rightarrow$ Keys K_1 & K_2 are succesfully exchanged