Date: / / Page No.: 1

The OSI Security Architecture

> Security Attack
> Security Mechanism.
> Security Serivce.

Security Services

The processing or communication services that is provided by a system to give a specific kind of protection to system resources; security services implement security policies and are implemented by security mechanism.

SS
↳ Authentication.
- Peer entity auth". (PEA)
- Data origin auth". (DOA).
↳ Access Control.
↳ Data Confidentiality.
↳ Data Integrity.
↳ Non-repudiation.
↳ Availability.

(1) Authentication :- • Ensures that the entity (user, device or server) is who they claim it to be.
• Example :- Using username/passwords, biometrics, or digital certificates.

a) PEA :-
- what it is: It verifies the identity of other party in a communication session.
- when used: During live interactions or real-time communication (in a network communication).

- Purpose: To make sure you are actually talking to the correct person/device and not an imposter.
- Example: When 2 comp. set up a secure connection (like in SSL/TLS), each one verifies the other's identity.

(b) DOA:-
- It confirms who created or sent a piece of data.
- Used mainly for messages or files - not ongoing communication.
- Purpose is to ensure that the data received actually comes from the claimed source.
- Ex:- When you rx. an email, system checks if it ~~try~~ truly came from the person it says it came from (using digital signatures).

(2) Data Confidentiality :-
- Ensures that data is only accessible to those authorized to see it.
- Example: Encrypting emails so that only the intended recipient can read them.

(3) Integrity :- Ensures that the data has not been altered during transmission.
Example:- Using checksum, hash functions (like SHA-256) to detect changes.

(4) Non-repudiation :- Prevents a sender from denying that they sent a message.
Ex:- Digital sign. ensures proof of the origin & integrity of data.

(5) Access Control :-
- Restricts access to resources to authorized users only.
- Ex :- Firewalls, role-based access control (RBAC) systems.

(6) Availability :-
- Ensures that the services & data are available when needed, even under attack
- Ex :- Protection against Denial of Service (DoS) attacks.

Security Service Mechanisms :-

These are the tools and techniques used to implement the security services (like authentication, confidentiality, etc.). They are the actual methods that make security happen in a network.

Major types of SMs :-

(1) Specific security Mechanism

a) Encipherment (Encryption) :-
- Converts data into a secret code to protect confidentially.
- Ex :- AES, RSA encryption.

b) Digital Signatures :- - Proves the authenticity and integrity of a message
Ex :- Signing an email to verify the sender.

c) Access Control Mechanism :- Controls who can access what resources.
Ex :- Setting file permissions (read/write/execute).

4 Data Integrity Mechanisms :- Ensures that data is not tempered during tx.
Ex :- Hash func. like SHA-256, Message Authentication Codes (MAC).

5 Authentication Exchange :- Methods for verifying identity during communication.
Ex :- Password authentication, challenge-response protocols.

6 Traffic Padding :- Sends extra (fake) traffic to hide real data patterns & prevent attackers from gaining info. by observing traffic.
Ex :- Padding comm. with dummy messages.

7 Routing Control :- Ensures data follows secure routes & avoids compromised paths.
Ex :- Choosing secure networks routes dynamically.

8 Notarization :- Involves a trusted 3rd party to verify the exchange of info
Ex :- Using a notary services to witness a digital transactions.

Pervasive Security Mechanisms :-
1. Trusted functionality :- The system provides func. that are guaranteed to work securely.
Ex :- Secure boot processes, trusted operating system kernels.
2. Security Labels :- Labels are attached to data or resources indicating their security level (like "Confidential", "Top Secret").

3. Event Detection :- Detects security-related events such as intrusion attempts or abnormal behaviour.
Ex:- Intrusion Detection System (IDS) monitoring network traffic.

4. Security Audits :- Recording & examining security-relevant activities for accountability & forensic analysis.
Ex:- Log files that track login attempts, file access, & system changes.

5. Resource Protection :- Protects resources like memory, CPU, or storage from being misused or overloaded.
Ex:- Preventing one process from accessing another process's memory.

Security Attack (S.A) :-
Action that compromises the security of an individual or an organization.
If attacks are successfully launched then affect of the attack would be loss of data, corruption of data, ransom attacks, injection of viruses, etc.

(1) Passive Attacks :-
- Attempts to learn or make use of information from the system
- Does not affect system resources
- Eavesdropping or monitoring of tx.
- Goal: Obtain information that is being transmitted.
  a) Release of message content. (RMC).
  b) Traffic Analysis. (TA).

a) RMC :- Attacker reads the contents of messages.
Ex:- Reading your private emails or listening to a phone call.

b) TA :- •) Even if the message is encrypted, the attacker studies patterns:
- Who is communicating with whom?
- How often are the message sent?
- How large are the messages?
•) From these clues, sensitive info. can still be infered.

(2) Active Security Attacks :- When attacker tries to modify, disrupt, or forge communication or system operations.
• Attacker interferes with the system.
• More dangerous bcz it changes data, services or both.

a) Masquerade :- •The attacker pretends to be someone else.
Ex:- A hacker logs into a bank account pretending to be a real user.

b) Replay :- Attacker captures a message & then re-sends it later to trick the system.
Ex:- Replaying a valid money transfer message to transfer money again.

c) Modification of Messages :- Attacker changes the contents of a message in transit.
Ex:- Changing the receiver's bank account number in an online payment.

d) Denial of Service (DoS) Attack :- Attacker disrupts normal services so that legitimate users can't access them.
Ex:- flooding a server with to much traffic to make a website crash.

Date: / / Page No.: 7

Passive Attack	Active Attack
• Hard to Detect	• Hard to Prevent
• Neither sender or rx. is aware of the attack	• Physical, software & network Vulnerabilities.
• Encryption prevents the success of the passive attack	• Detect & recover from any disruption or delays
• More emphasis on prevention than	• If the detection has a deterrent effect, it may also contribute to prevention.

Access Control Matrix.

It is a simple way to represent who (which users) can access what (which resources) & how (what type of access like read, write, execute).

It's basically a table where

Rows: Subjects (Users, Processes)

Columns: Objects (files, Database, Printers)

Entries = What actions are allowed (like read, write, execute).

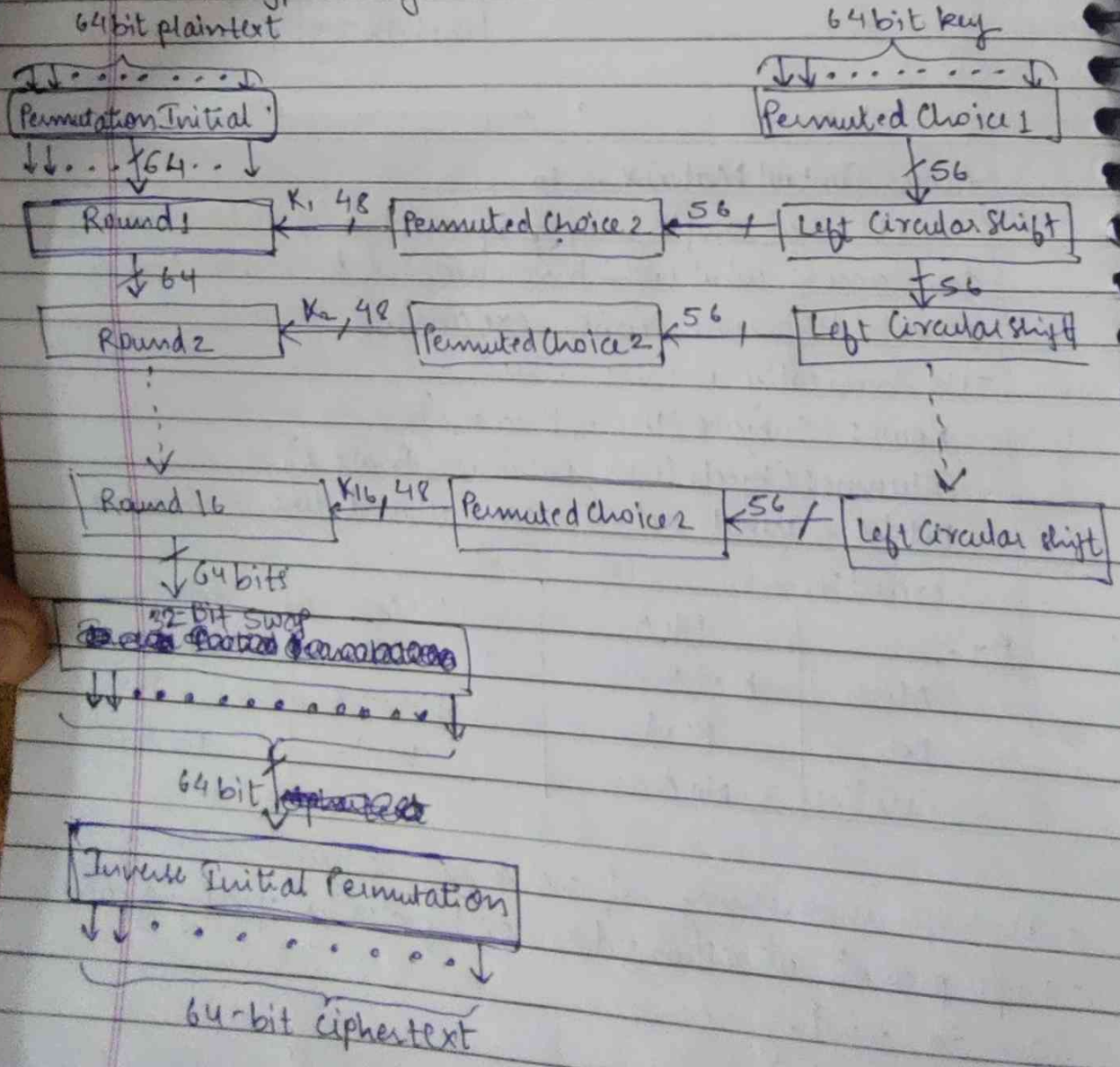
Ex:-	File A	File B	Printer
Alice	Read, Write	Read	No Access
Bob	Read	No Access	Print
Carol	No Access	Write	No Access.

- It helps in designing & managing permissions easily.
- Helps prevent unauthorized access by clearly specifying who can do what.

Data Encryption Standard (DES)

* Symmetric block cipher (same key used for both encryption & decryption). → (Take group of bits as input).
* Adopted by NIST in 1977
* Input size : 64 bits, Output : 64 bits, Main key : 64 bits
  Subkey : 56 bits, Round key : 48 bits, No. of rounds : 16

DES Encryption Algorithm.



Date: / / Page No.: 9

Initial Permutation

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

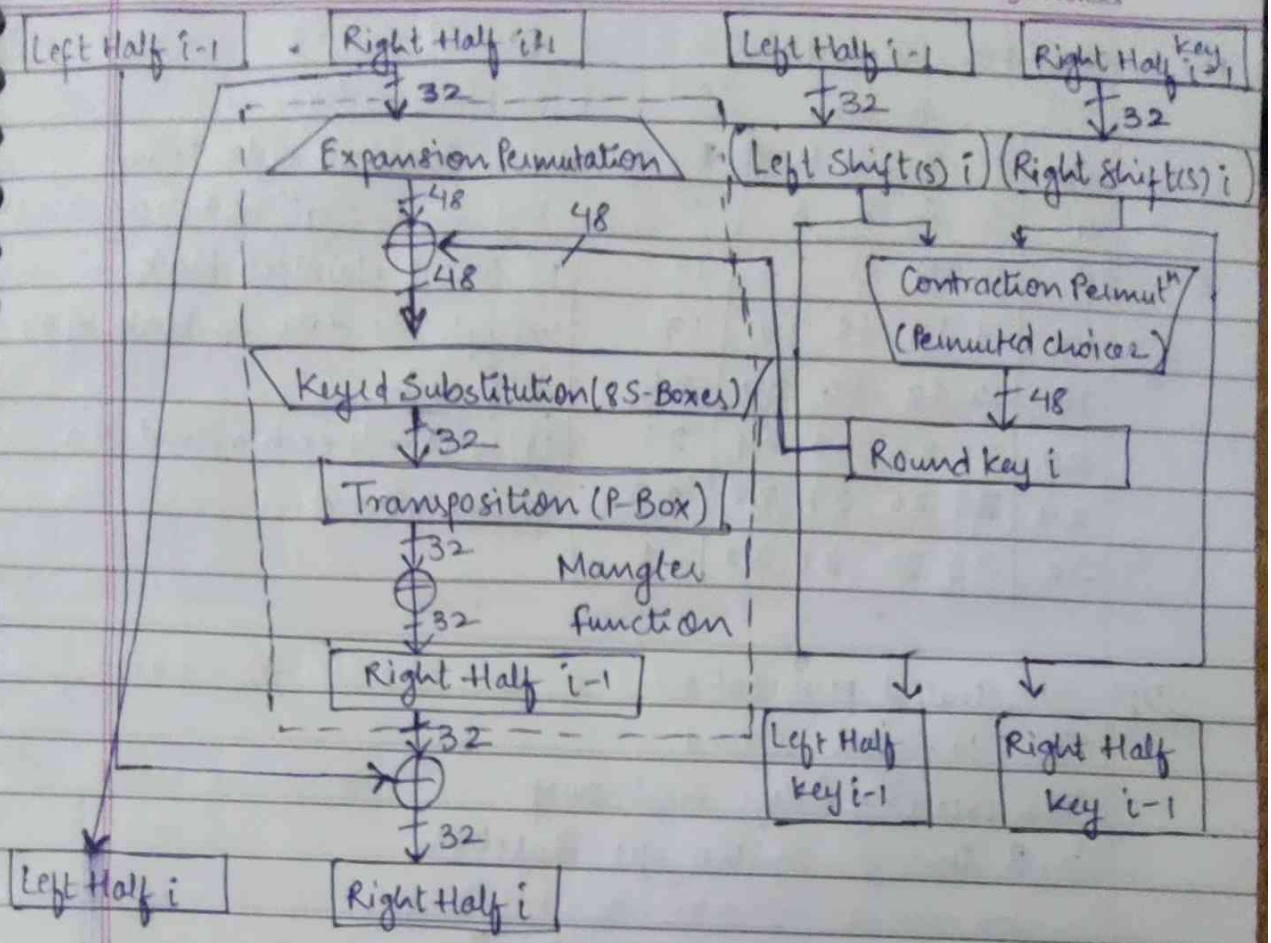
⇒

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Inverse Initial Permutation

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

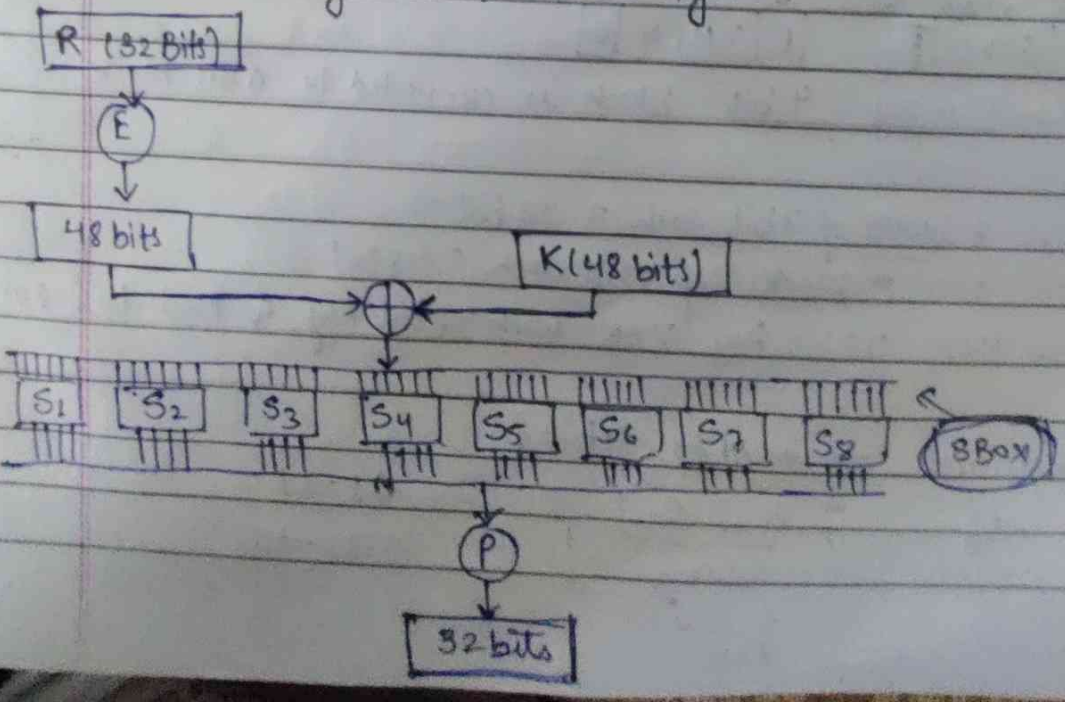
Date: / / Page No.: 11



$L_i = R_{i-1}$

$R_i = L_i \oplus F(R_{i-1}, K_i)$

The f function (Mangler fun) in DES Algo.



The Expansion Permutation

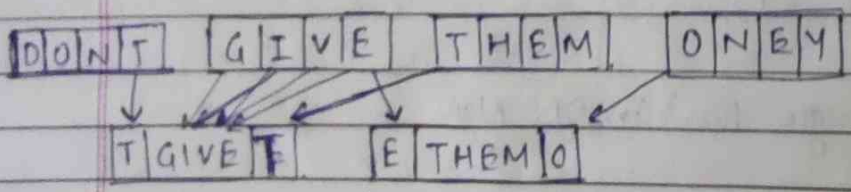
8bit	32 bit				8bit
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

DES

(i) Block cipher,
(ii) Symmetric cipher (same key for encryption + decryption)
(iii) 64 bit plaintext block - encrypts the data in blocks of 64 bits each.
(iv) 16 rounds each round is a fiestal round.

Steps (i) Initial permutation
(ii) 16 fiestal rounds
(iii) swapping / left right swap.
(iv) Inverse initial permutation.

Q - What happens in expansion box?
32 bit data will be → 1's & 0's form

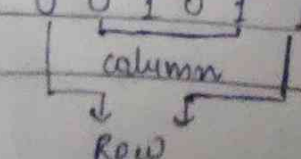


So here every 4 bit block is converted to 6 bit block

There were 8 blocks of 4 bit each = 32 bit
Now, there are 8 blocks of 6 bit each = 48 bit
Now these 48 bit key XOR with 48 bit key & sent to S-Box

Que How 6 bits converted to 4 bit?

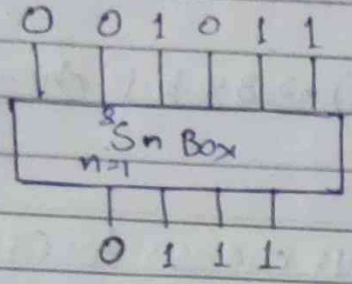
eg 0 0 1 0 1 1



01 → 1
0101 → 5

Date: / / Page No.: 13

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



Que How 16 subkeys are generated?

Actually we have 64 bit key which go as a i/p to PC-1 (permuted choice-1) & we get o/p as 56 bit key.

Inside PC-1 -

64 bit key divided into → 8 parts each of 8 bit $8*8 = 64$ bit

for each part last bit → discarded

i.e bit → 8, 16, 24, 32 - - - 64 discarded

Hence, 8 parts of 7 bits = $8 \times 7 = 56$ bits each

O/p of PC-1 is 56 bits which is then divided into 2 parts of 28 bits each → C_0, D_0

Now, these bits are shifted with left shift in each round.

in Rounds, i = 1, 2, 9, 16 → 1 shift i.e rotated left by 1 bit.

two halves rotated by 2 bits

Inside PC-2 56 bit → 48 bit using a predefined table

Then we get our 1st key for Round 1

DES Decryption :- reverse of DES Encryption.
64 bit plain text → 64 bit cipher text
64 bit Ciphertext → 64 bit plaintext
↙ K1 → K16 , K2 → K15 . . . . . K16 → K1 .
(keys)

Avalanche Effect in DES.

* Strong.
* 1 bit change in plaintext (PT) → 34 bits change in (CT) Cipher text on average.

* 1 bit change in key - 35 bits changes in CT on avg.

Strength of DES.

(1) • The use of 56 bit keys
  * Subkey size : 56 bit keys
  * 2^{56} possible keys
  * 7.2×10^{16} keys
  * A brute force attack appears impractical.
  * key space has to be secured.
  * One DES encryption per microsecond.
  * More than a thousand year to break the cipher.
• Insecure DES
  • DES cracker - $250,000 , own cracker
  • The attack took less than 3 days
  • Alternatives - AES & triple DES

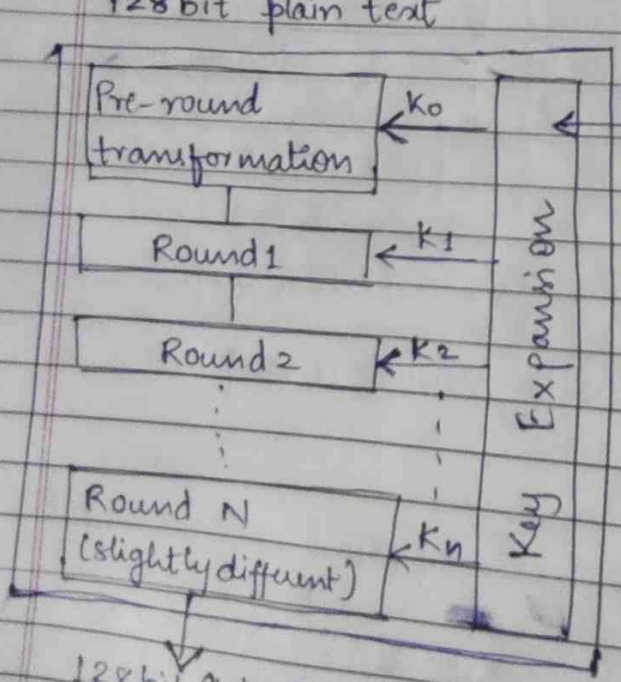
Date: / / Page No. 16

Advanced Encryption Standard (AES).

* Symmetric key block chain cipher (i.e same key used for encryption & decryption).
* Estab. in 2001 by the US NSIT.
* Fixed block size = 128 bits i.e 16 bytes = 4 words (∵ 1 word = 32 bits).

Rounds	No. of bits in key	
10	128	AES-128 version
12	192	-11- - 192 -11-
14	256	-11- - 256 -11-

* no. of keys generated by key expansion algo. = no. of rounds + 1

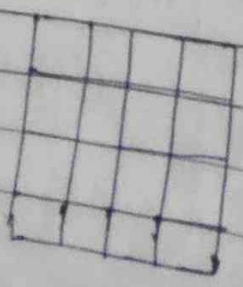


Cipher Key (128 or 192 or 256 bit).

General Design of AES Encryption.

State
16 byte (4x4) matrix
Store intermediate result

Input array



(4x4 i.e 16 bytes i.e 128 bits or 4 words).

State Array

	$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
1st byte of 0th word	$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
2nd byte of 0th word	$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
	$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

→ $[W_0, W_1, W_2, W_3]$

↳ 3rd byte of 1st word.

Key → 128 bits or 4 words.

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_{15}

expand Key Algo →

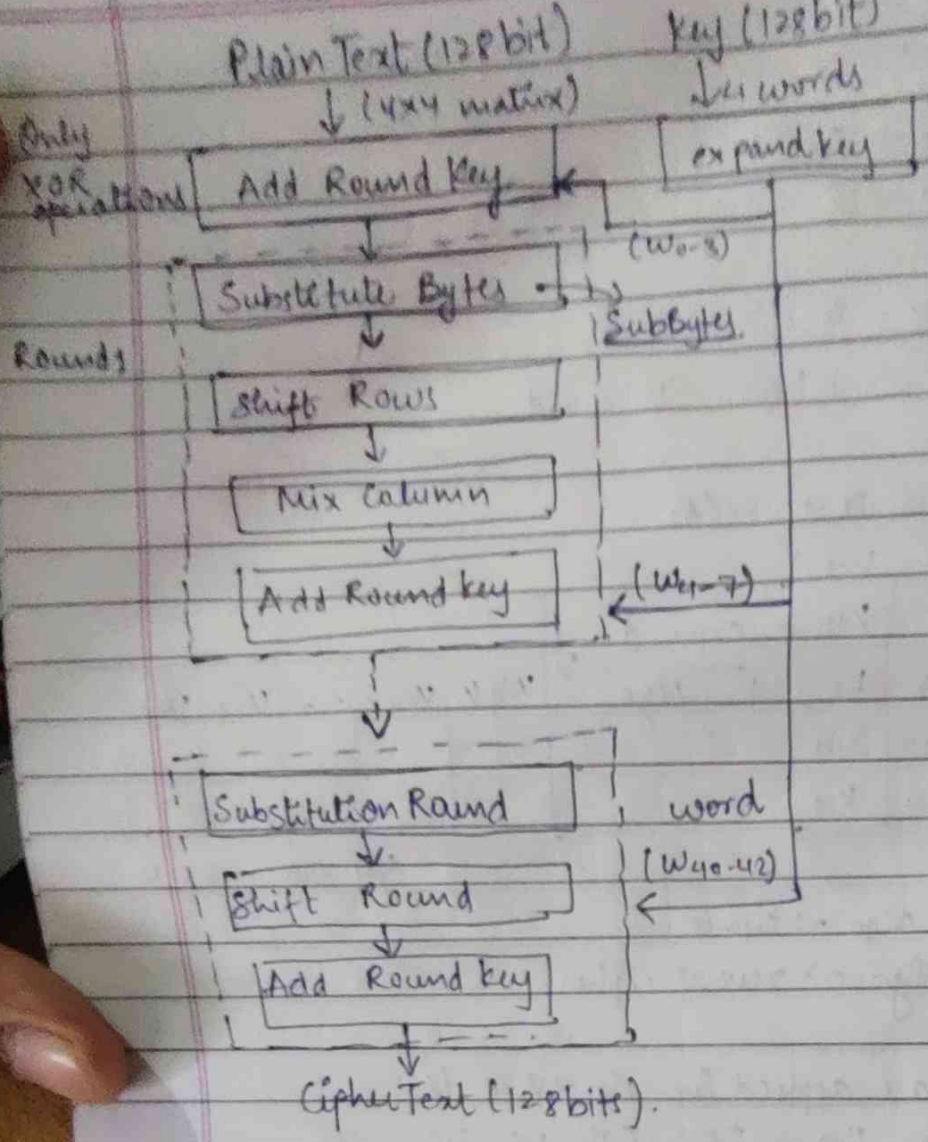
W_0	W_1	W_2	. . .	W_{42}	W_{43}

44 words

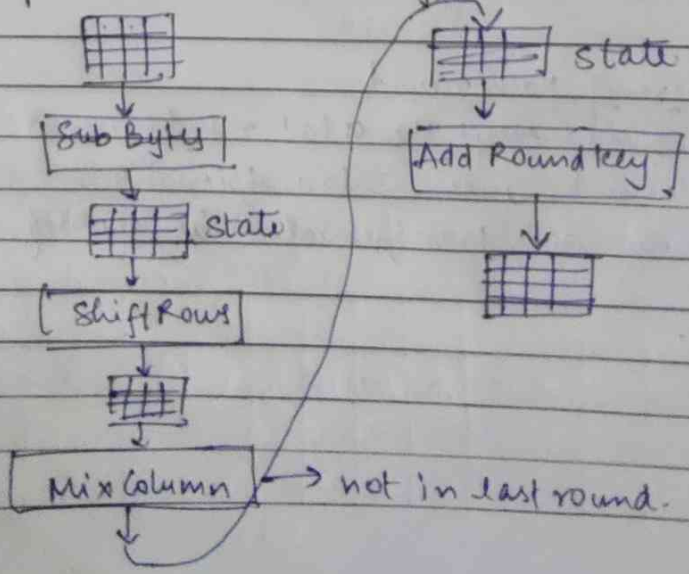
* Encryption Algo → cipher
* Decryption Algo → reverse cipher.
  ↓
  Round keys are applied in reverse order
* Much stronger than DES (all i.e 2DES, 3DES).

Structure of Each round

* Each round except the last round uses 4 transformations
* last round uses 3 transformations
  i.e mix col^n transformation is missing.



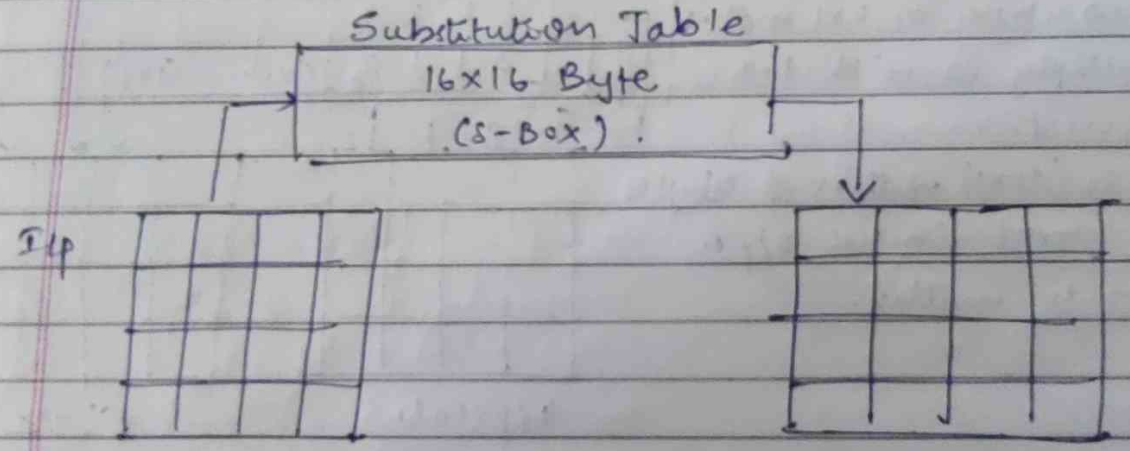
Structure of each Round at encryption side



Transformations :- 4 types substitution, permutation, mixing & key-adding

(1) Substitution :- AES, like DES uses substitution. But mechanism is diff. Substitution is done for each byte. Only 1 table is used for transformation of bytes.
which means that if 2 bytes are same, the transformation is also same.

Sub-Bytes :- at enc. side we interpret the byte as 2 hexadecimal digits
1st hexadecimal dig → row } of substitution
2nd —11— —11— → col } table.



(2) Permutation → In this we permute/shift the bytes. In DES, permutation was done at bit level. AES :-
—11— is —11— byte level

Shift Rows
* shifting is done to left
* no. of shifts depends on the row of the state matrix.

Row0	63	C9	FE	30		63	C9	FE	30	No shift
Row1	F2	F2	63	26	Shift Rows →	F2	63	26	F2	1 byte shift
Row2	C9	C3	7B	D4	(shift left)	7D	D4	C9	C3	2 byte shift
Row3	BA	63	82	D4		D4	BA	63	82	3 byte shift.

In decrypt^n we use Inv shift Row (shifting right).
no. of shift same

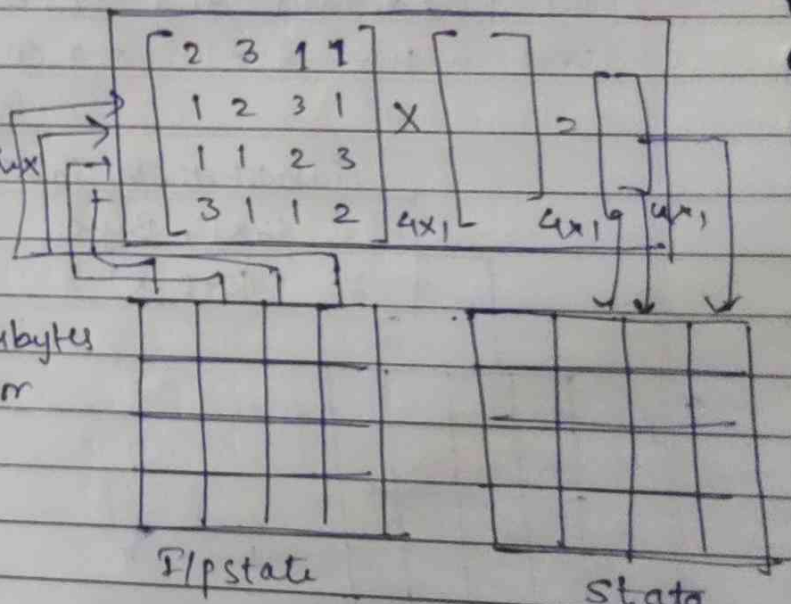
Note shift[Row] transformations are
& Inv —"— inverse of each other.

③ Mixing

Mix Col's for enc

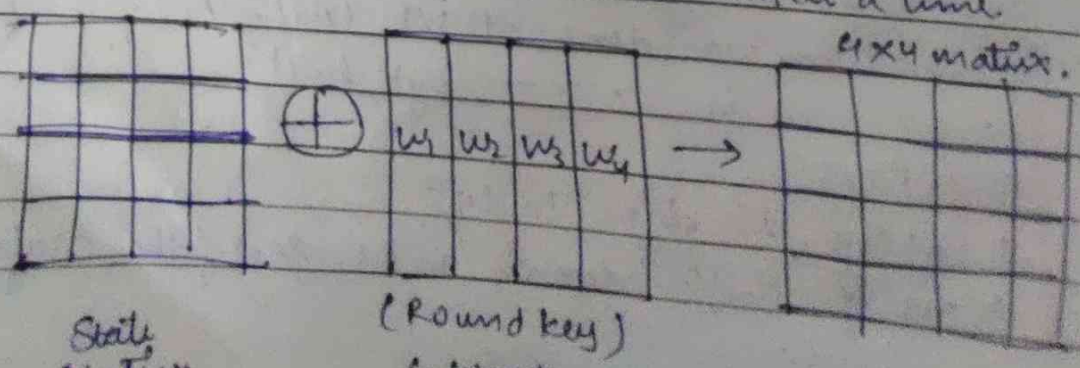
Take each word/column
ie 4 bytes or 4×1 matrix
& multiply it with the
const. matrix

The o/p is (4×1) matrix of 4 bytes
& is stored in the o/p or
state matrix.



④ Key Adding

Add Round key → also proceeds 1 column at a Time



AUTHENTICATION

* It is one of the 5 principles of security.
* Verifying the authenticity of the msg is v. imp.
* An authenticator must be there to authenticate the message.

Methods to produce authentication.

(1) Message Encryption. → Cipher text
(2) Message Authentication Code (MAC)
(3) Hash Functions → hash value

MAC (Message Authentication Code)

* We will use a secret key to generate a small fixed size block of data called MAC or cryptographic checksum.
* It is then appended with the message.
* The communication parties will share a secret common key which will be used to create a MAC.

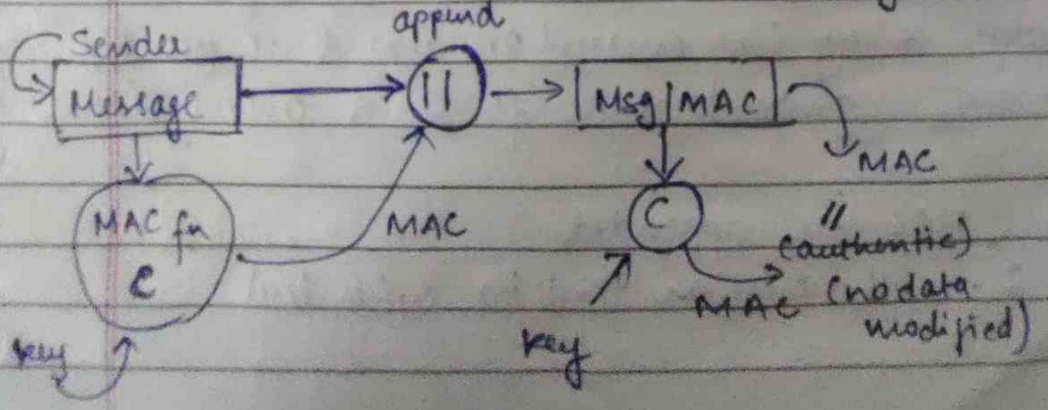
Let A → sender
B → receiver.

when A sends a msg to B, it calculates the MAC as a function of the message & the key

MAC = C(K, M).

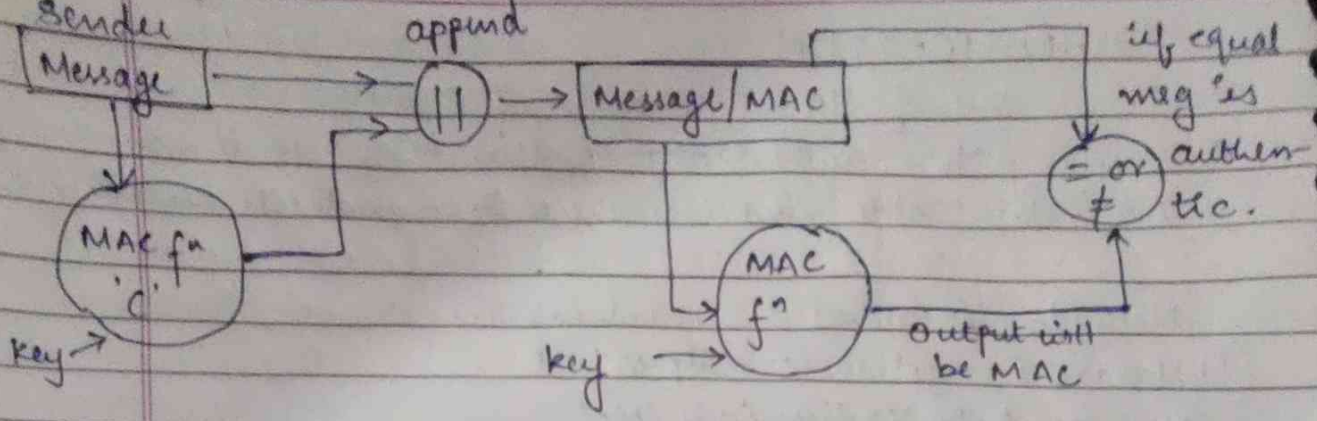
where,

M = input message
C = MAC function
K = shared secret key.



(1) MAC for Authentication

Date: / / Page No. 22



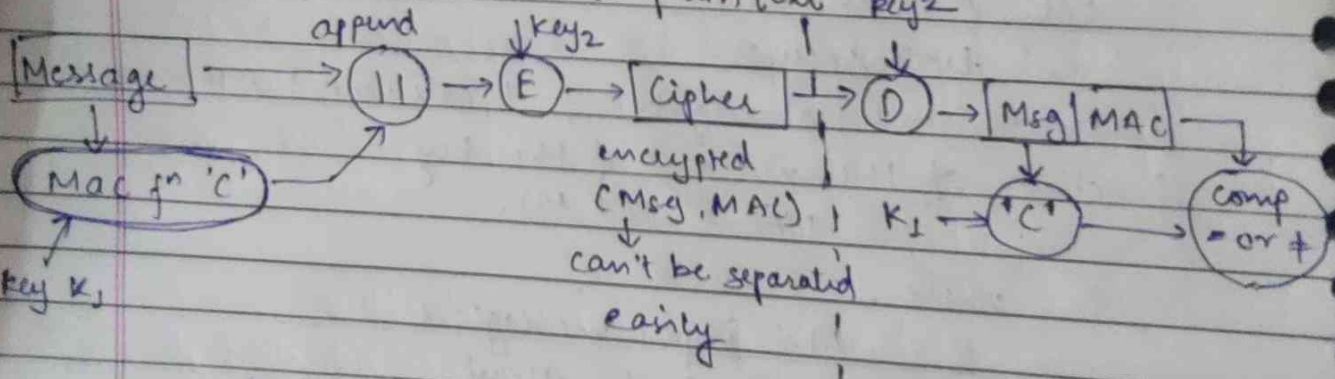
We separate the msg & the MAC.

Only authentication is achieved.

Drawbacks (i) No confidentiality bcz if 3rd party come in b/w then he can get the msg :. no security.

(2) MAC for Authentication & Confidentiality

(a) authentication tied to plain text



Sender's side

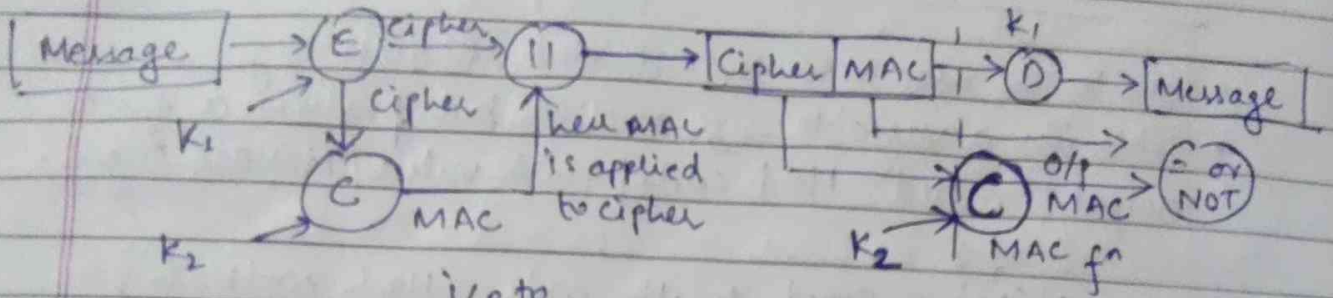
* Apply message + key, in encryption algo & we get O/P
* This O/P + key2 given to MAC fn C & output will be MAC
* Now, cipher & MAC are appended

:. authentication tied to cipher text

Receiver's side

Date: / / Page No.: 23

(b) authentication tied to ciphertext



i/p to

* we will give it as decryption Algo using K_1 & o/p → message.
* Take cipher apply it to MAC func (C) & o/p → MAC
* Now compare this with separated MAC

if = Msg authentic else NOT.

significance of MAC

(i) Ensures that rx. knows whether the msg is altered or not.

(ii) Authentication Ensured i.e receiver is ensured that the msg came from the correct sender.

HASH FUNCTIONS

- Similar to ~~was~~ MAC but it doesn't uses key
- Takes in a variable size message but produces a fixed output. called Hash code / Hash value / Message Digest.

$H(\overset{msg}{M})$ = fixed length code (Hash code 'h')
↳ act as an authenticator

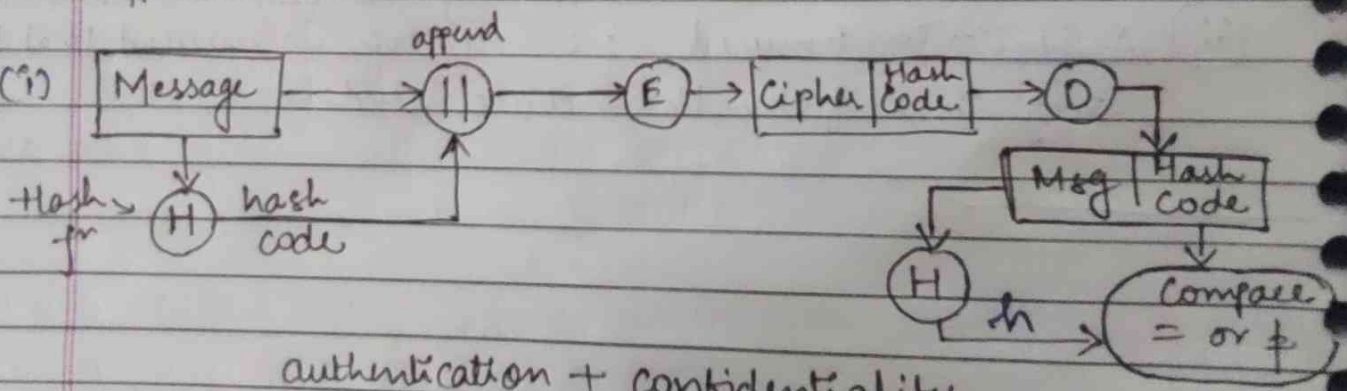
- The only input is message.
- A hash value 'h' is generated by a fn H.

$H(M)$ = fixed length code 'h'
↓ variable length ↓ hash code.

They are also called compression function

There are diff methods to provide authentication in diff situations.

(i)

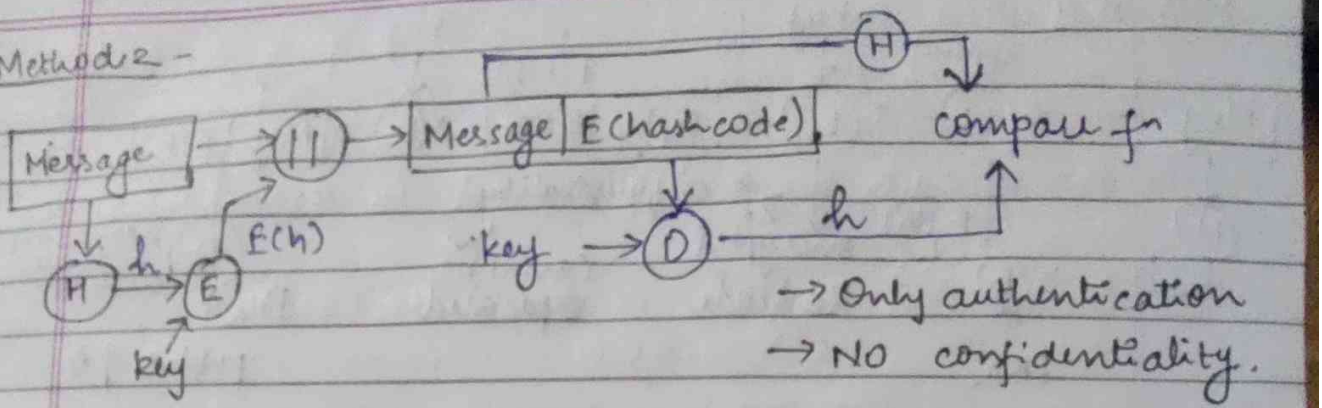


authentication + confidentiality maintained b/c msg was encrypted before sending.

if both hashcodes equal in the end b/c = bcz only A & B shares the secret key, the msg must have come from A & has not been altered.

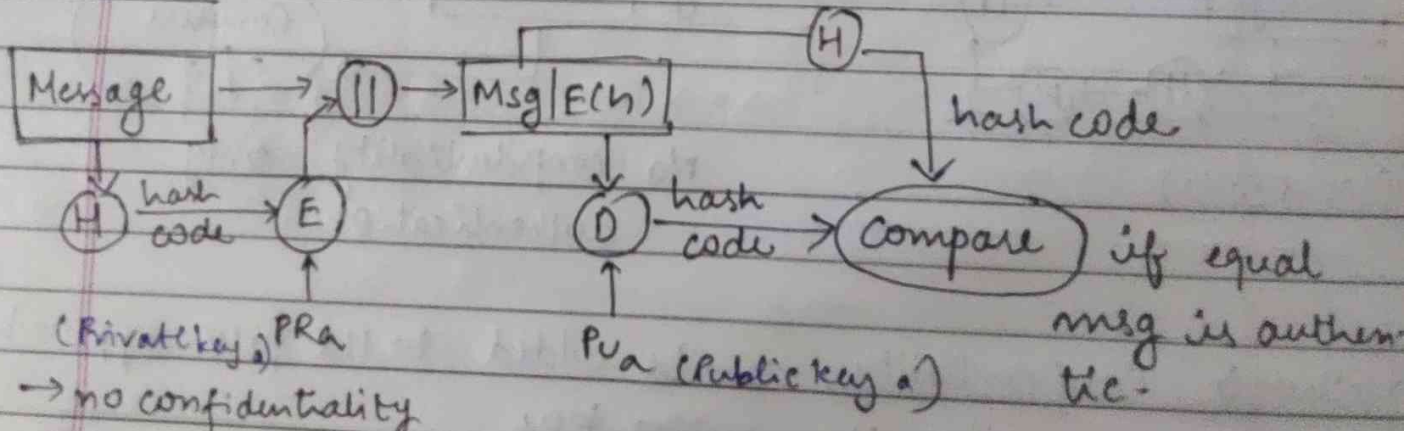
Date: / / Page No.: 25

Method 2 -



Only hash code is encrypted, using symmetric encryption If you need only authentication & no confidentiality & your msg is not private msg so we can use it bcz the processing time will be less bcz we are not encrypting the msg. We are encrypting the hash code

Method 3 :-

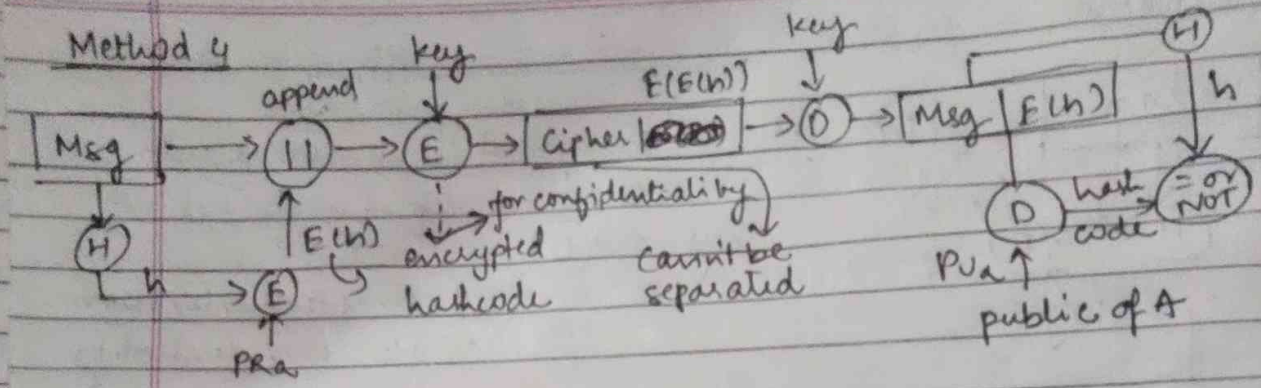


Processing time will be less & as the msg is not encrypted.

(same concept as above but using Assymetric key crypto).

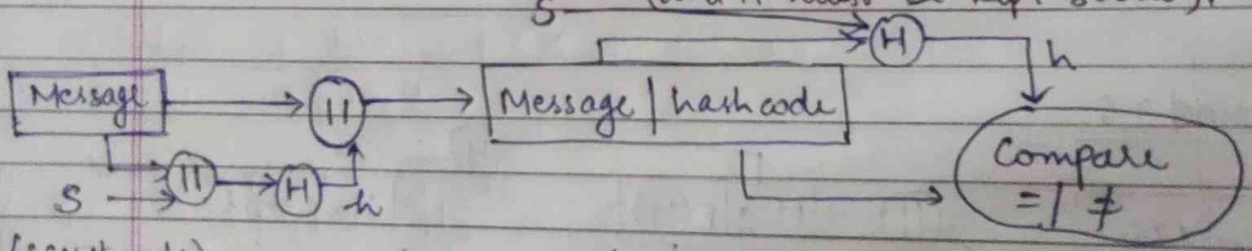
Date: / / Page No.: 26

Method 4



when we need both confidⁿ + authⁿ.
using symmetric key.

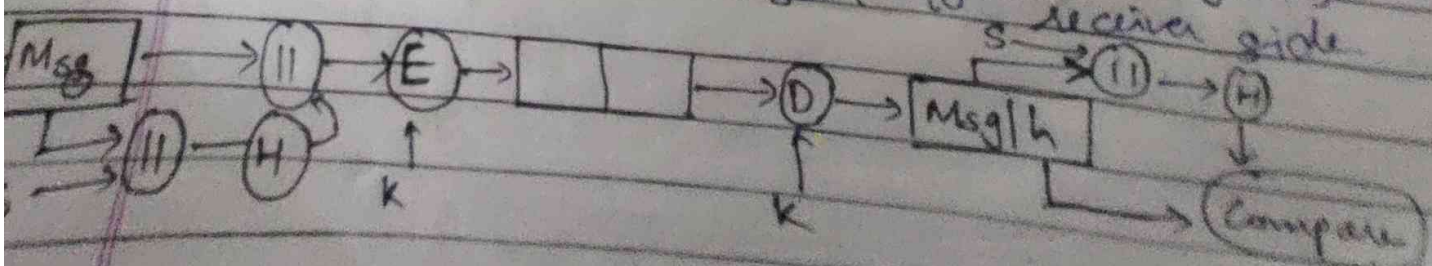
Method 5 Sender and Receiver will have a secret code 's' (and it must be kept secret).



No confidentiality only authentication.

Method 6 Confidentiality can be added to the prev. approach by encrypting the entire key

Take the msg & append with the ~~secret~~ secret code 's'. Then apply Hash fn. It gives 'h'. Now append 'h' & msg. Now encrypt using key 'k', We get encrypted (msg + h). Now it will be sent to receiver side



PKI:- Public Key Infrastructure

Date: / / Page No. 32

PKI is a framework of technologies & policies that enables secure communication using public key crypto....

It helps people, websites and systems prove their identities & exchange data securely over the internet (online banking, secure emails, etc.).

Key Concepts of PKI

Component	Description.
Public key	Shared openly; used to encrypt data or verify signature. ↓ ↓ ↓ ↓
Private Key	Kept secret; decrypt create digital sign.
Certificate Authority (CA)	Trusted third party that issues & verifies digital certificates. (DC)
DC	A file that binds a public key to a person, web., or entity - issued by a CA.
Registration Authority (RA)	Verifies user identities on behalf of CA before certificates are issued.
Certificate Revocation List (CRL)	A list of revoked & or invalid certif'—. (ex expired, compro.).

How PKI Works?

1. A user or website generates a public-private key pair.
2. They request a certificate from a CA (like DigitalCert, etc).
3. CA verifies their identity (through RA if needed).
4. CA issues digi. cert. signing it with its own private key.
5. Other users can verify the certificate using the CA's public key.
6. Data can now be securely exchanged.

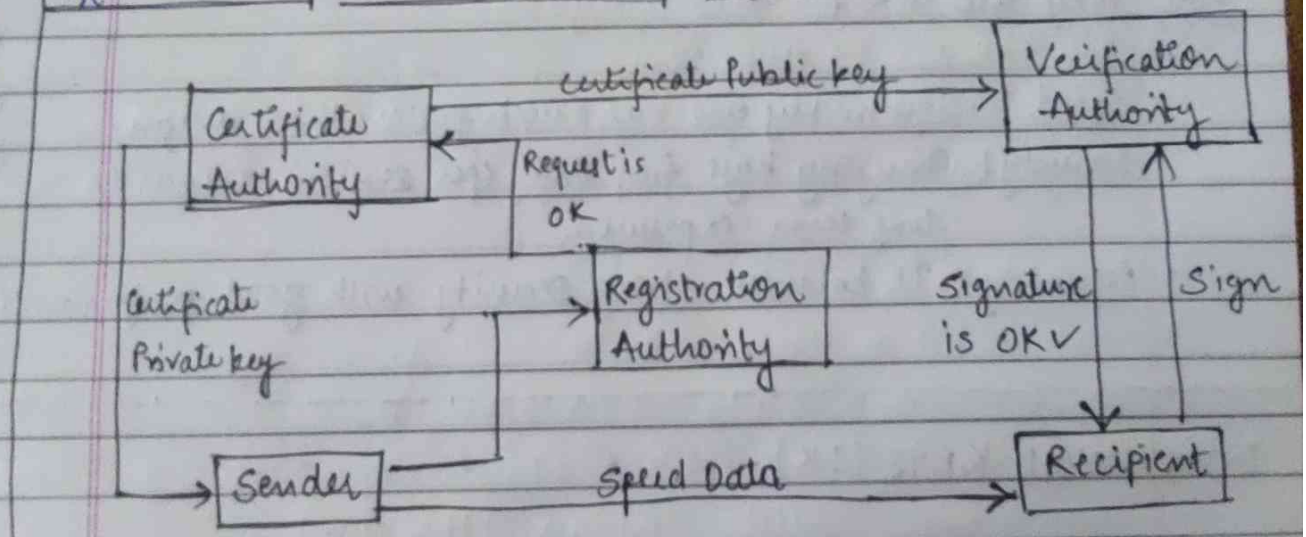
Date: / / Page No.:

Where PKI is used:
- HTTPS (SSL/TLS) → Secure browsing
- Email encryption (eg S/MIME).
- Digital Signatures → Document integrity & authenticity.
- VPN authentication.
- IOT device security.

SessionKey

A temp. enc. key used to secure a single comm. session between two parties.

- Ensures that the data exchanged during that session stays private & secur



Public Key Infrastructure

Session key cont...

Key Points:

Features	Details
Type	Symm. key (same for enc. & dec.).
Lifespan	short Term (only for duration of 1 session).
Purpose	To encrypt & decrypt message during a session.
Generation	Created at the start of a session & discarded after it en
Advantage.	Even if SK is compromised, only that session is affected - not all communication.

Date: / / Page No. 34

Que How session keys work?

1. 2 parties (client & server) agree to establish a secure session.
3. A session key is used for enc. & dec. messages during that session.
2. A SK is generated.
   - Sometimes it's gen. by one party & securely shared.
   - —"— both parties contribute random numbers to create it.
4. After session ends, the SK is discarded & a new one is gen. for next session.

Que Why use SK?

- Speed: Symm. key enc. is much faster than assym.
- Security: Changing keys frequently (per session) reduces the long term exposure.
- Efficiency: It balances strong security with good performance.

INTERCHANGE KEYS (IK)

IK are enc. keys specially used to exchange other keys specially SK.

Most often used in assym. crypto. (public private key systems) where:

- Public keys are used to enc. the session key (or data).
- Private keys are used to dec. it.

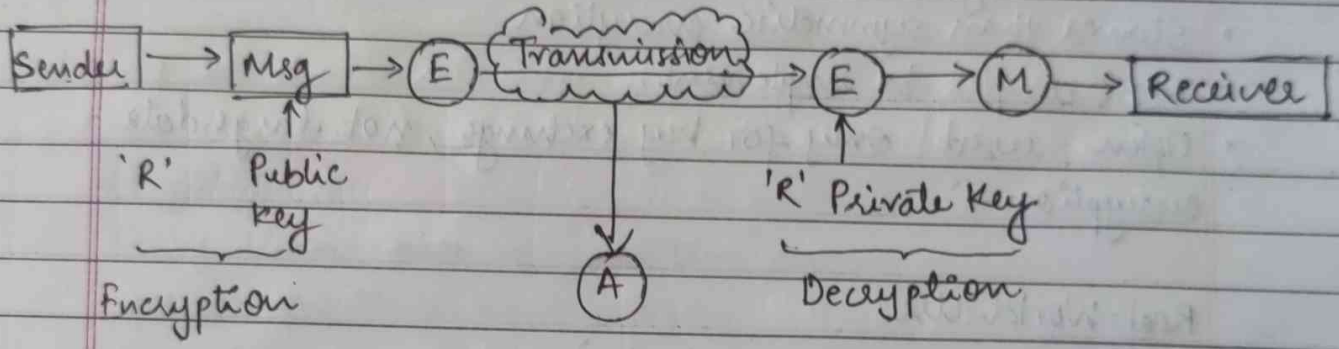
Purpose:

To securely deliver SK from one party to another, especially when using a public network.

How IK works?

1. Alice wants to send a SK to Bob.
2. She uses Bob's public IK to enc. the session key.
3. Bob receives the enc. SK & uses his private IK to decrypt it.
4. Now both parties have the same SK & can use it for fast symm. enc. during the session.

Assymmetric / Public Key Cryptography.



It is a type of encryption that uses 2 different keys:
- A public key (shared openly).
- A private key (kept secret).

These 2 keys are mathematically related, but you can not derive one from the other easily.

Public Key — Used to encrypt data or verify a digital sign.
Private Key — 〃 — decrypt 〃 — or sign data digitally.
Data encrypted with one key can only decrypted with the other.

Qu — How it works (Example):

Secure Message Sending:

1. Alice wants to send a secret message to Bob.
2. She encrypts it using Bob's public key.
3. Only Bob can decrypt it - using his private key.

Digital Signature:

1. Bob wants to prove he sent a message.
2. He creates a digital signature using his private key.
3. Anyone can verify that signature using Bob's public key.

Advantages :-

- No need to share private keys.
- Supports authentication (via digital signatures).
- Ensures confidentiality & integrity.

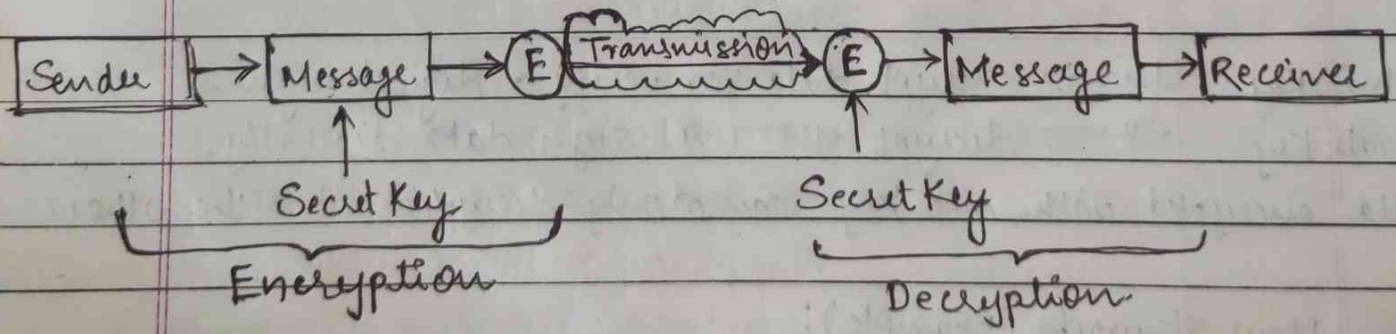
Disadvantages :-

- Slower than symmetric encryption.
- More complex to implement.
- Often used only for key exchange, not large data encryption.

Real-World Use

- SSL/TLS (for HTTPS websites).
- Digital signature (PDFs, code signing).
- Email encryption (PGP).
- Cryptocurrency wallets (Bitcoin, Ethereum).

Symmetric Key Cryptography



An encryption method where same key is used for both encrypti & decryption.

- Simple, fast, & widely used for securing data during communication or storage.

One key to rule them all - both sender & receiver must have the same secret key & keep it private.

How it Works ?

1. Alice wants to send a message to BOB.
2. She encrypts the message using a shared secret key
3. Bob decrypts the message using the same key.

* If someone else gets the key, they can read or alter the message - so key secrecy is very important.

Advantages :-

- faster than ~~RSA~~ asymmetric enc.
- Less computational power req.
- Efficient for real-time communication

Disadvantages :-

- Key distribution is risky
- No built in authentication
- Scalability Problem.

Common Algo :- AES, DES, 3DES, RC4 (~~RC~~ stream cipher) *.

Real World use :-

- Wi-fi Encryption
- file Enc.
- Database Enc.
- Secure Messaging apps.

R.SA Algorithm

- Rivest-Shamir-Adleman developed in 1978.
- Asym. Cryptographic Algorithm. (2 keys) i.e public & private key concept is used here.
- → Public key : known to all users in N/w.
- → Private key : kept secret, not sharable to all.

If public key of user A is used for encryption we have to use the private key of same user for decryption.

The RSA scheme is a block cipher in which the plain text & cipher text are integers b/w 0 & n-1 for some value n.

1. Key Generation

(i) Select 2 large prime nos. p and q.

(ii) Calculate $n = p * q$.

(iii) Calculate $\phi(n) = (p-1) * (q-1)$ // eulers taitent 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 $ed \equiv \text{Mod } \phi(n)$.

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

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

private key = $\{d, n\}$

Let $p = 3, q = 11$

$n = p * q = 3 \times 11 = 33$

$\phi(n) = 2 \times 10 = 20$

Given $e = 7$ as $1 < 7 < 20$ & $\gcd(7, 20) = 1$

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

$ed = \bmod \phi(n)$

$7 \times d = \bmod \phi(n)$

$7 * d \bmod \phi(n) = 1$

$(7 * d) \bmod 20 = 1 \quad (\because d = 3)$

↳ multiplicative inverse of 7

Since $e = 7, d = 3$

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

private key = $\{d, n\} = \{3, 33\}$

Encryption Plaintext → $M < n$ (always)

$C = M^e \bmod n$

Let $M = 31$

- $C = 31^7 \bmod 33 = 4 \Rightarrow C = 4$

Decryption

$M = C^d \bmod n$

$M = 4^3 \bmod 33 = 31$

$\boxed{M = 31}$

Buffer Overflows :-

It is a type of software vulnerability that occurs when a program writes more data to a buffer (temporary storage area in memory) than it can hold.

This extra data can overwrite adjacent memory, potentially leading to :

- program crashes.
- Data corruption
- Or even unauthorized code execution (security threat).

How it happens :

1. A program allocates a buffer (say 8 bytes) for user input
2. The user input more than 8 bytes.
3. The extra overflow into other parts of memory.
4. This can overwrite :
    - Variables
    - Return addresses.
    - Or inject malicious code.

Example :-

```
char buffer[8];
gets(buffer); // If user inputs more than 8 characters
                → overflow.
```

Dangers :-

- Crashing applications.
- Gaining root / system access.
- Spreading Malware.
- Taking control of servers or systems.

Date: / / Page No.: 41.

Prevention Techniques.	Description.
(1) Input Validation	Always check input length before storing
(2) Bounds Checking	Use safe func like fgets() instead of gets().
(3) Stack Canaries	Special Values used to detect overflows before return
(4) ASLR (Address Space Layout Randonization).	Randomizes memory locations to prevent exploits.
(5) Use of High Level Languages.	Languages like Python & Java manage memory automatically.

Date: / / Page No. 42

CROSS-SITE SCRIPTING (XSS)

XSS is a security flaw in web applications where attackers inject malicious scripts into web pages that other users view and interact with.

Example:- <script> alert 'XSS Detected!'); </script>

- Steal cookies, session tokens, or login credentials.
- Trick used into performing unwanted actions.
- Deface websites
- Redirect users to malicious websites.
- Install malware.

Q How it works?

1. A website displays user input (like comments or search terms) without properly sanitizing it.
2. An attacker injects a malicious Javascript snippet.
3. The site unknowingly sends that script to other users.
4. The script runs in users browser with the same permission as the trusted ~~browser~~ site.

Types of XSS

(1) Stored XSS :- Malicious script is stored on the server (e.g in a comment or post), & served to other users.

(2) Reflected XSS :- Script is a part of a URL or form submission, & reflected in the response (eg. search results).

(3) DOM based XSS :- Vulnerability in the client-side Javascript code, not the server.

Prevention :-

- Escape/sanitize user input - convert <, > etc. to HTML-safe eq.
- Use content Security Policy (CSP) to restrict script sources.
- Use HTTPOnly flags on cookies to prevent theft.
- Use trusted library (e.g DOMPurify) to clean HTML.
- Never directly insert untrusted data into HTML, JS or CSS.

Command Injection

It is a critical web security vulnerability that allows an attacker to execute arbitrary system commands on a server - directly through a vulnerable application.

It's often caused by improper input validation when user input is passed to a system shell (like bash, cmd or sh).

Example

Imagine a website lets users "ping" an IP address by enting:

```
http://example.com/ping?ip=8.8.8.8
```

Behind the scenes, it runs:

```
ping 8.8.8.8
```

Now an attacker could try

```
http://example.com/ping?ip=8.8.8.8;ls
```

This executes

```
ping 8.8.8.8; ls
```

This cause the system to list files on the server - not intended behaviour.

Date: / / Page No.: 44.

Commonly Targeted Commands.

Unix/Linux: ls, cat/etc/passwd, rm, whoami
Windows: dir, type, net user, del.

Attack Payload Examples

Payload	Action
8.8.8.8; whoami	Shows current user
127.0.0.1 && rm -rf /	Deletes all files (dangerous).
www.google.com & net user	Lists user accounts (on Windows).

Detection

(1) Manual Testing with payloads like:
- ;, &&, ||, |,
- inject into any user input fields or URL parameters.

(2) Tools
- Burp Suite (Intruder module).
- OWASP ZAP
- Nikto
- Commix (specifically for command injection).

(3) Logs:
- Look for unusual shell command errors in server logs.

Prevention Practice	Description
-> Avoid exec calls	Use safe, high-level lang functions instead.
-> Input Validation	Strictly validate & sanitize user input.
-> Whitelisting	Only allow pre-approved input values (e.g. IP addresses).
-> Escape Shell Characters	Properly escape special characters.
-> Use parameterized APIs	Avoid string concatenation in system calls.

ACCESS CONTROL MECHANISMS.

Access control refers to rules & techniques that determine who can access or modify what resources in a system. It protects data from unauthorized access & misuse.

Components	Description
Subject	The user or system requesting access.
Object	The resource (file, database, applicn, etc).
Access Rights.	The allowed actions (read, write, execute, delete).

Types of ACM :-

(1) Discretionary Access Control (DAC).
- Access is based on owner's direction
- The owner/user can grant or revoke permission.
- Ex: Windows file permissions.

✓ Flexible

✗ Less secure (users can grant access to others)

2. Mandatory Access Control (MAC).
- Access is based on security labels (e.g Top Secret, confidential)
- System enforces rules, not users.
- Common in military & government systems.

✓ High Security.
✗ Rigid & hard to manage.

3 Role-Based Access Control (RBAC).
- Access is granted based on a user's role in an organization.
- Example:- Admins have full access; Employees have limited access.

✓ Scalable & manageable
✗ May be too generalized in complex systems

4. Attribute-Based AC (ABAC).
- Access is based on attributes (user, rsrc, environment).
- Policies are enforced based on dynamic conditions (e.g time, location).

✓ Very flexible & fine-grained.
✗ More complex to configure.

5. Rule-Based Access Control.
- Uses predefined rules set by the system administrator.
- Often used in firewalls & routers (e.g allow traffic from IP x to port y).

Enforcement Methods	Example
Access Control Lists (ACLs)	Per-object lists defining which users/roles can perform which actions.
Capability Lists	Per-subjects lists defining what resources that users can access.
Access Control Matrix.	A table showing which subjects can access which objects & how.

CONFINEMENT PROBLEM.

It refers to the challenge of ensuring that a program or process cannot leak sensitive data to unauthorized users - even if it processes confidential info.

Goal :- To confine a program in such a way that:

- It can operate on sensitive data.
- But cannot leak that data outside its allowed boundaries.

Que Why its Hard?

A program can leak data through:

- Direct Channels: Writing to a file, sending over a network.
- Covert Channels: Indirectly communicating Data using system properties (e.g. CPU usage, file locks, response timing).

Example

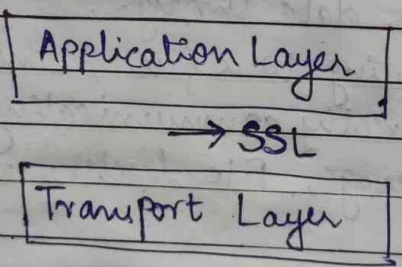
Scenario	Risk
A browser plugin reads password data	It might send the data to an ext. server
A cloud function processes private info.	It may log sensitive o/p somewhere insecure.

Security Measures (Partial Solutions):

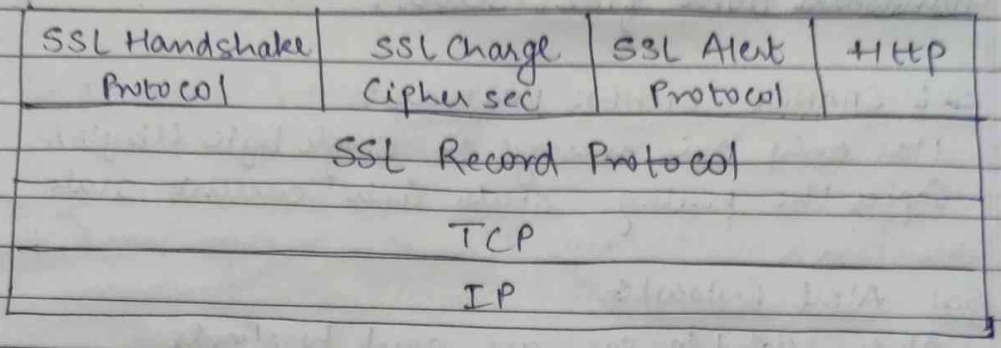
Approach	Description.
Sandboxes	Limit what the program can access (e.g. file system, network).
Mandatory Access Control	Enforce strict policies on data access.
Info. Flow Control	Track and control data moves between processes.
Audit Logging.	Monitor access and data movements.
Data Loss Prevention (DLP).	Detect & block data exfiltration attempts.

SECURE SOCKET LAYER (SSL).

- To provide security for (communication b/w 2 users).
- Ensures integrity, authentication & confidentiality.
- Lies between application layer & transport layer of TCP/IP.

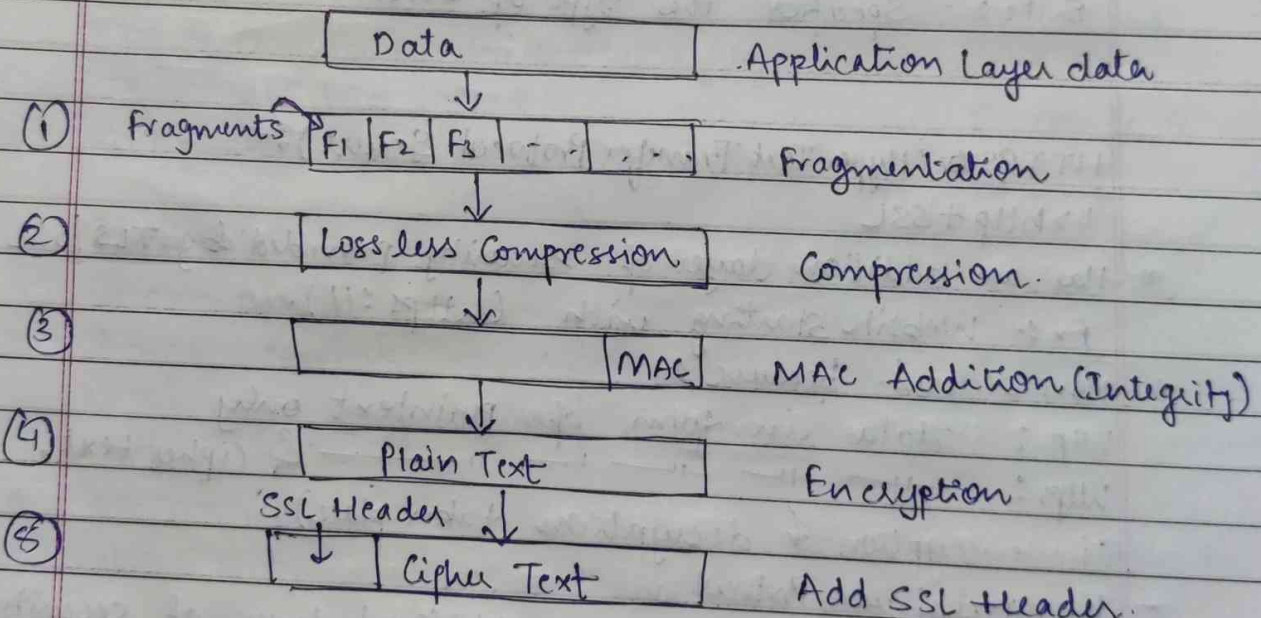


Protocol Stack of SSL :-



SSL Record Protocol :- It has 2 services

(1) Confidentiality → by encryption
(2) Message Integrity → by MAC.



SSL Handshake Protocol :

→ Ensures Authentication.
→ Most complicated part in SSL.
→ Key exchange b/w client & server.

Working :-

(1) connection establishment with server.
(2) Key exchange from server to client. } authentication
(3) —"— —"— —"— client to server }

4 Handshake done from server.

SSL CHANGE CIPHER PROTOCOL:-

- Has only one message :- single byte (1 byte).
- Copies the pending state into current state

SSL Alert Protocol:-

Alerts related to SSL are sent to clients
- has 2 bytes.

Byte 1 - Can have values as (1) or (2)
1 - warning 2 - fatal Error. (terminate)

Byte 2 - Specifies the type of error

HTTPS (Hyper Text Transfer Protocol Secure):-

↳ http + SSL

* Has an addition layer of security provided by TLS/SSL
Ex:- Website starting with (https://).
* More secured. because:
http: data in form of Plaintext only
https: -"- -"- -"- -"- -"- & cipher text.
(i.e encryption & decryption takes place).
- Transport Layer Protocol
- heavier than http (has an additional layer of security).
- Runs on port no. 443 of server.
- uses a certificate authority. (CA).
- slower than http

Main Usage:- Banking websites
Login Credentials

Intrusion Detection System (IDS).

Intruder :- A person trying to get unauthorized access to a system or a ~~person~~ network

* **Outside Intruder (masquerader)** :- not having any authorized access to the system
* **Inside Intruder (misfeasor)** :- having some authorized access to the system but with some restriction & tries to misuses the info.

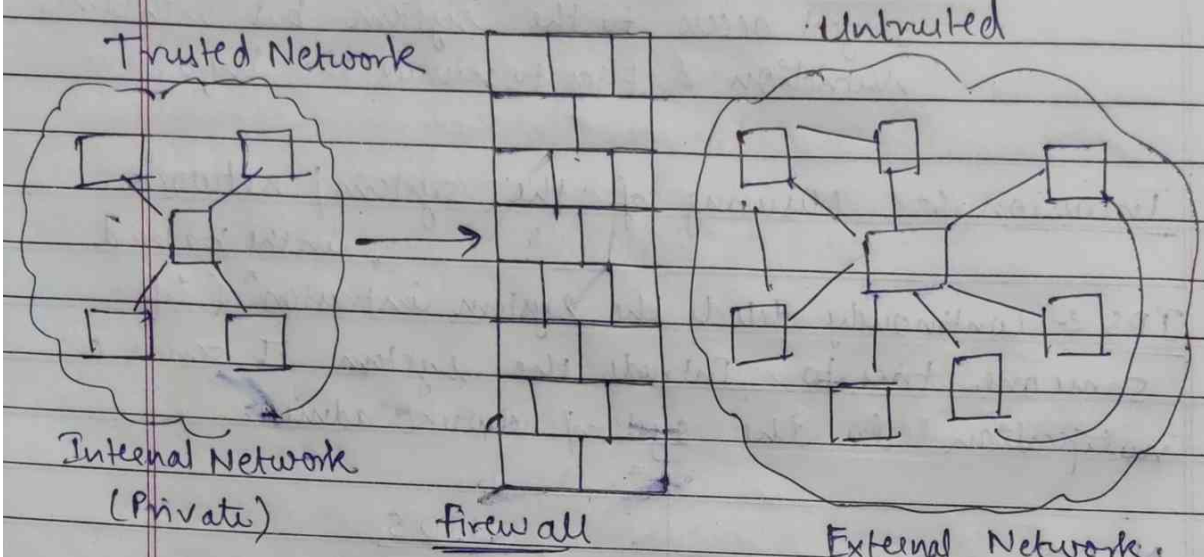
Intrusion :- Misuing of the system/network. in the backend

IDS :- Continously detects the system intrusion & if someone tries to Intrude the system it sends a alert notification to the system/network admin.

	NIDS	HIDS
*	Network Based	* Host Based.
*	Analysis & Matches traffic to the library of known attack.	* Installed on individual host or device on nw.
*	Monitors, capture & analyse network traffic.	* It monitor data packets from the device only & will alert the admin if suspicious activity is detected.
*	Detect malicious data present into packets	* Snapshot Existing System → Previous System.
→	Analysis is very difficult in busy network.	* Files detected or modified.

Firewalls

* Monitors & control Incoming & outgoing traffic based on predefined rule
* Acts like a barrier * Host based & network based firewall.
  Software ✓ → Hardware



Packet filtering firewall (layer-4).

* Check IP header, TCP header.
* Works on Network and Transport layer.
* Can block IP address, full Network
* Can block a service (http, ftp etc.).

Don not allow:

Rule No.	Source IP	Source Port	Destination IP	Dest Port
1	179.2.4.80	Any	Any	Any
2	152.32.0.0	Any	Any	Any
3	Any	Any	172.9.0.3	Any
4	Any	80	Any	Any
5	Any	Any	Any	21

Date: / / Page No.: 53

Application (Proxy) Firewall (Layer-5)