19/08/22

Symmetric encryption → for large blocks of data

Asymmetric → for personalized data (Eg): Email

Security and threat & security attack

Threat - possible violation of any of the rights in CIA

Attack - implementing a threat
↳ not a possibility anymore but a reality.

CIA Security Triad:

✓ ~~Authentication~~ < ~~Only authorized person~~

✓ Integrity - Change in data can be done only by ~~authenticated people~~ authorized

✓ Confidentiality - ~~message can be accessed only~~

✓ Availability - No DoS, one can always access the services.

3 levels of attack:

✓ Low → not a total disruption (minor data loss, financial loss)
✓ Medium → some disruption / damage
✓ High → catastrophic damage

Low confidentiality - Faculty info
Med. confidentiality - Student roll no.
High confidentiality - Endsem marks (cannot be revealed to public)

Low integrity - online polls
Med. integrity -
High integrity - patient details (of allergies, etc..) in a hospital DB

High security - Online exams.

RFC - Request for Comment

A document prepared by any of the members of a committee / internet society submitted to the committee (for discussion)

The committee reviews the protocols / rules & places the proposal in one of the 5 categories:

Recommended
Required
Elective
Useful
Not Recommended

The proposal goes through 3 stages

1. Draft

2. Proposed - implementation

3. Available / Internet Standard - Accepted for use

IP, ... are all RFCs.

IETF → Internet society

OSI - defines the functions of layer by layer and implements security at each layer

NIST maintains security standard for the internet

National Institute of Stds & Tech

Book :

Network security

- Stallings

~~Computer Security~~

Accountability :

100% security cannot be guaranteed. So a track record containing all activities of a user and he is held accountable if any threat/violation is found

security breach

security mechanism :

identify threats, try to correct it or prevent it

security service :

Attack

Passive

Releasing of sensitive information but no attack on the system itself or data

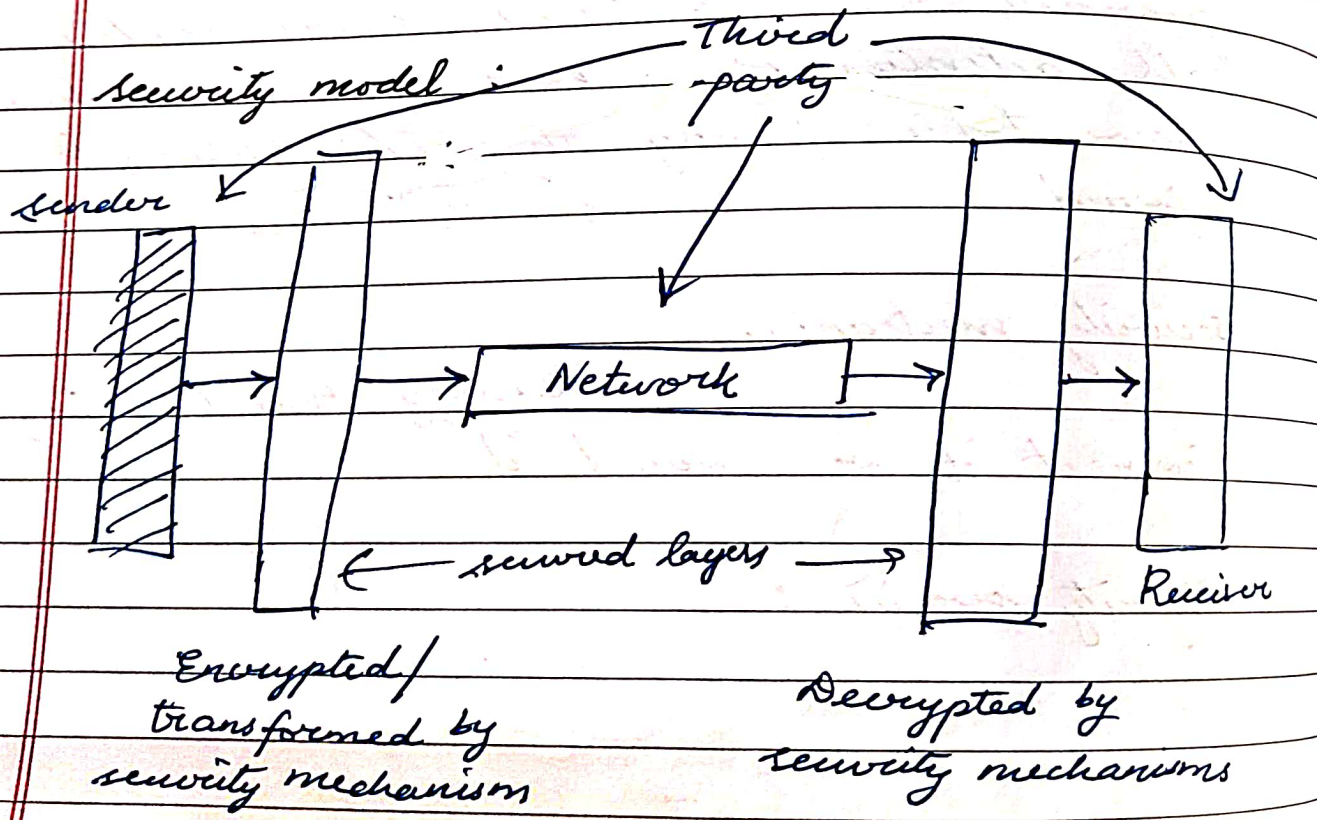
(Eg): Traffic analysis

Active

(Eg): DoS, Message masquerading (hacker changes a user's info → also Replay, Modification of Message.

Masquarade – hacking someone's Fb account and posting sth (not done by the account owner) Ex

Replay – duplicate a site (fb) to get the bank account details (say) of a user.



Attack - implementation of threat

Prevention of passive attacks - through encryption

Replay - Breach of Trust
cannot be detected

Access control - Concerned with the channel

Connectionless confidentiality
↓
no secure connection btw the nodes
(Eg): UDP

Without Recovery - just detect & send a msg to the user
(so he can resend the data)

25/08/22

Security mechanisms :

They are used to ensure & provide security service

Complete mediation → difficult to achieve
(time & resource consuming)
↓
no way to escape from verifying your authenticity

Open Source → people can find the vulnerabilities and contribute

UT - User Terminal

Gatekeeper → (Eg): firewall

Worm - just blocks your system from responding
(does not affect the system itself).
replicates repeatedly

Based on research paper

Security Issues in TCP/IP suite

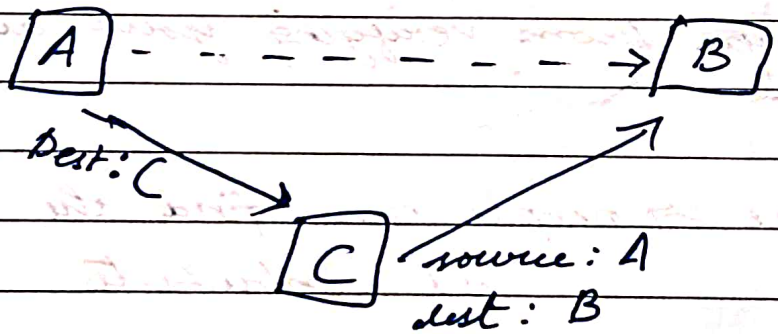
1. Sniffing / Eavesdropping:

✓ Can happen in both TCP & IP layer
✓ An attacker installs itself to src or dest, listens to the traffic and modifies it

2. Spoofing:

✓ Attacker ~~sends~~ gets the packet from src, modifies it & sends to dest.
(Attacker copies the src address of src and sends msgs to dest using that address and pretending to be ~~the~~ src).
⇒ Virus attacks src computer and sends the data from src to attacker instead of actual destination.

example?



3. Buffer overflow → inherent to the prog. lang

Limit of buffer reached (extra bits flow to adjacent memory locations → excess bits overwrite existing memory locations)

This vulnerability is exploited in this attack by using a malicious code to modify those extra bits

↳ Uses stack or heap.

(Eg): C, (Nowadays we use modern lang. that handle such buffer overflows)

4.

4. Poisoning ~~of ARP poisoning~~

Corrupting data packets to make new illegal packets.

Attack router so that it repeatedly sends packets to a node eventually leading to a DOS for the dest or blocking the router itself

5. Illegal packets → Ping of Death
combination of actions

Illegal ~~data bit patterns~~ → assumed to never occur.

TCP layer fns → ACK, RST, FIN, SYN.

(Eg): FIN + RST ⇒ cannot reset after finishing?

Other eg — SYN + FIN
SYN + RST

6. Storms :

There is a threshold for # msgs generated & considered to be normal

If a worm is installed that keeps replicating and generating a msg ⇒ Floods the network

May crash the communication network itself and lead to DoS

eventually threshold is crossed

7. Denial of Service & distributed DoS

Involves making a system as zombie

↓

system that keeps generating unnecessary msgs leading to blocking of a system

distributed → many zombies scattered over the network

↓

all of them have to be identified so it is difficult.

8. ARP Poisoning (Man-in-the-middle) :

↓

Address Resolution Protocol ⇒ IP → MAC

(to find IP or MAC address)

Reverse ARP → RARP

⇒ MAC → IP

Table is maintained ⇒ ARP cache !

(IP → MAC !)

Malicious program attacks the table to modify the IP address of dest. to that of the attacker (in the ARP cache)

~~Assume~~ we are assuming the ~~RARP~~ response ~~is~~ corresponds to the most recent ARP request

9. ICMP exploits :

↓

works in the network ~~(IP layer)~~. (uses datagrams of IP).

~~transport~~ layer

(can be used with both TCP, UDP).

Usually ICMP packet is encapsulated by IP packet

Attacker can use the decoding of ICMP echo packet ~~to broadcast~~ the address as the ~~victim addr,~~ set the ~~src~~ dest addr to victim addr in IP packet and broadcast the packet across the network

⇒ now all nodes send ICMP echo reply to the victim leading to it getting blocked

ICMP packets are encapsulated as IP datagrams.

✓ DoS is easy to achieve using ~~ICMP~~ ICMP exploits

✓ Can lead to Flooding / congestion in the network

✓ Ping of death also possible

29/08/22

IPv4 spoofing:

IPv4 does not check if src address is correct or not
⇒ assumes it is always correct

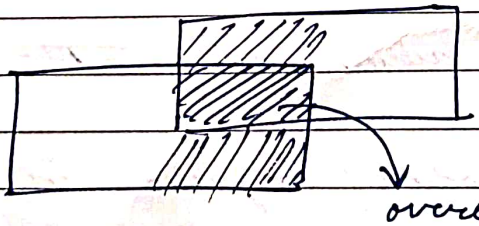
So an attacker can spoof the addr with his own

Detection is easy if src, dest are on the same network (router can perform this check) ↓
more chance of spoof attacks then router checks for remote access

Prevention → check if the path of the packet to get back to src is valid or not (has to pass through another router, it...

IP fragment attack:

No check if fragments (while defragmenting or ~~reassembling~~ reassembling) are in order or have overlaps
⇒ Attacker can create overlaps in IP packets



overlapping part of data is overwritten

(if port addr is overwritten, leads to unattended data
⇒ can result in DOS

generally used to bypass firewall

Routing exploits :

Bellman Ford → distributed algo (so time-to-time updation is needed).

5 attacks :

✓ DoS :

change router's addr or steal dest addr in the routing table

✓ Packet Misreading Attack

✓ Routing ~~Misleading~~ table exploits

✓ Hit and run

Attacker just tries to gain access to router and continues other attacks if granted (if not it tries with next router)

✓ Persistent attack

Persistent Hit & Run

Keeps trying until it gains access to router. (tries all possible attacks).

⇒ difficult to detect, more dangerous.

* UDP exploits :

UDP - connectionless

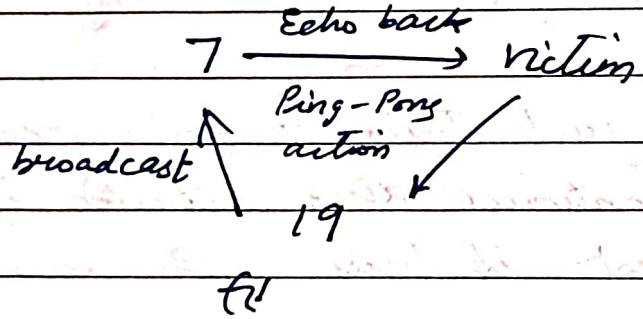
used bcos of prompt response

1. Flooding

Chargen
→ listens on port 19
provides stream of data to test a terminal

2. Echo
→ port 7.
simply resends the data

Hacker for Broadcast Src addr. at port 7 and send it to some intermediate router (modify dest. port to 19).



⇒ leads to congestion (unnecessary test data)

3. Fraggle

switch off chargen, echo to avoid flooding attacks.

TCP Exploits

1. Sequence number prediction:

random - only pseudo random
once we find the seed, we can find ISN.
can ~~it~~ then sniff the packets to get the sequence (initial ISN that starts ;).)

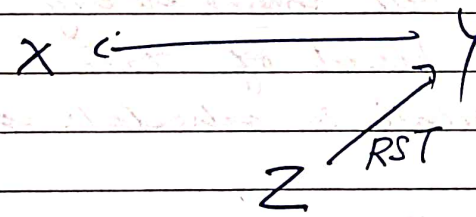
2. Closing connection:

FIN set $\Rightarrow$ TCP closes the connection (involves ACK).

Attacker sniffs IP of src and pretext? the src itself to be the dest.
~~(or)~~ attacker can simply set the FIN flag ~~in IP and in~~

3. TCP RESET Attack

✓ Closing connection like attack using RST flag to form a new ~~or~~ connection (tries to get ACK from dest.).



Y waits for ACK from X
but X is not aware of RST request at all (bcos Z made it)
$\Rightarrow$ Leads to ~~flattings~~ & DOS
Stall,

Reason for these attacks
TCP simply accepts datagrams as long as they are in the valid range irrespective of the order

4. Shrew attack:

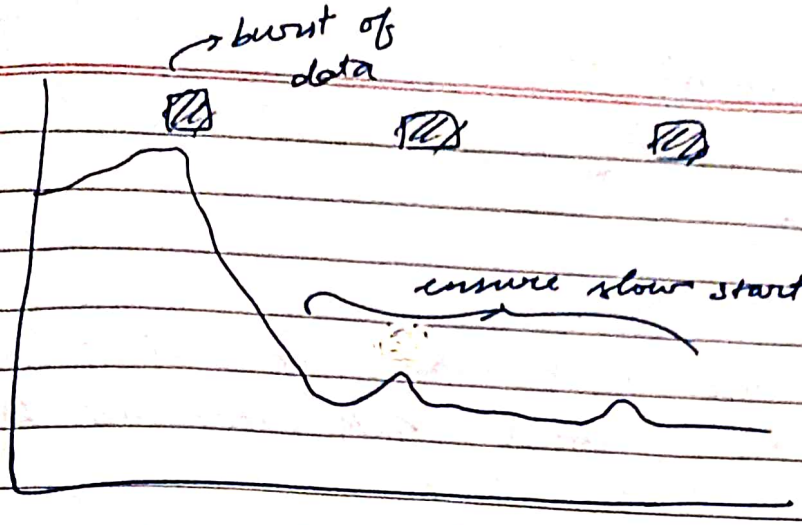
Exploits congestion control protocol in TCP

Congestion control mechanism in TCP cwnd

TCP throughput ↑ by dynamic congestion control protocol
Congestion window timer is increased exponentially whenever there is a chance for congestion until it reaches slow start phase tcd after which window increases only by 1
↓
limited transmission

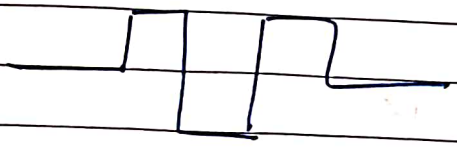
Attacker sends series / bursts of data at fixed frequency / interval
(to increase congestion window)
Eventually it reaches slow start phase and stays there till retransmission timer goes off out immediately after which another burst of data is sent
(easy to calculate & schedule such transmissions)
This way, attacker ensures it is always in slow start phase

Look at: State transition diag TCP (RFC 793).



21/9/22

For shrew attack, it is enough to send a square wave to & cause the attack



ACK Tricks :

Attacker sends duplicate (multiple) copies of ACK whenever a packet is received
↳ ACK division

Duplicate ACK Spoofing :
For certain amount of packets received, we send duplicate ACK (some packets may actually be lost while others are simply not acknowledged)
⇒ increases congestion window and forces slow start phase

Optimistic ACKing :
ACK not sent for current packet's sequence number but for some other sequence number (of packets yet to be received). The ~~se~~

current seq no

X ——————→

ACK
SYN-ACK
sent for X+K

Sender assumes X was received (even if it was lost) because ACK received for sequence no. X+K itself.
⇒ leads to modification of

Illegal segments :
Cannot send SYN + FIN (together).

S ——→ A ——————→ V
SYN-FIN

Once it receives ~~FIN~~ SYN
status becomes SYN-RCVD

timer initiated & when timer goes off connection is closed automatically.

receives FIN ↓

CLOSE-WAIT (closing initiated)

hereafter it is stalled & does not receive any requests from A or any other ~~node~~ sender node (S).

So V is in half-closed state (keeps waiting for CLOSE ACK from A).
⇒ denial of service attack at V.

Simultaneous Connections.

X Y

X ——SYN——→
←——SYN—— Y

both initiate connections together

TCP protocol :

Y & X waits for SYN-ACK but both receive SYN from each other

So SYN is treated as SYN-ACK itself
↳ from dest.

After SYN-ACK is received connection establishment timer goes off & status becomes SYN-RCVD.

If X is malicious, instead of sending SYN-ACK to Y, it sends back SYN itself

Now Y goes to SYN-RCVD state and ~~X~~ waits for ACK but there is no timer here.

SYN flooding | connection flooding (many instead of one port)

If X sends SYN back to all cores, then all cores of Y ~~are~~ ~~st~~ go to SYN-RCVD state and are stalled (X does not send ACK).

If Y is an FTP server, then it will be down for all ⇒ DoS.

~~Conn~~ Only way to escape is blocking the IP address of X

⇒ ~~dif~~ attack is strengthened by ~~using~~ IP spoofing and multiple attackers (all requests are now not from 1 src/ address).

⇓

difficult to identify all of the attackers

When blocking the addresses of attackers, it is possible to block some addresses from genuine people also if attacker spoofs and uses this address

src gets flooded with ACK
lowers bandwidth
src also blocked from accessing dest
(X) (S) (Y)

Connection Hijacking :

existing connection

A causes DoS at S

A tries to sniff the data transport btw S, D and can find / sequence no. (valid)
Now A sends ACK for that seq. no. to D.
⇒ D can assume all other packets are lost
(No ACK from A for other packets).
Now D resends the other packets to A ~~will sniff~~ again & A will send ACK with its own data for other seq. nos
⇒ ~~indirectly~~ A establishes a connection with D. (D is unaware that connection has been hijacked)

D assumes A is the legitimate src !
(If S sends data, since D already received invalid msg with the same seq. no from A, it simply discards the data from S
A can get any task done from D.)

After A gets the task done, it frees S from DoS and relieves D as well.

can involve confidential info as well getting from D.

as they are considered duplicate

Message Authentication Requirements

1. ~~No~~ Disclosure } Confidentiality
2. Traffic Analysis }
3. Masquerade } Authentication
4. Content Modification }
5. Sequence modification }
6. Timing modification }

↓

changing the time at which msg A is to be sent (store & send later).

⇓

old msgs (not current msgs)

change order of msg before giving a sequence no.

7. Source repudiation } Digital Signature
8. Destination repudiation → all services clubbed together can ensure this (does not fall under 1 service like confidentiality).

under 1 receiver cannot deny sending/receiving a message

02/09/22 Social Engineering :

Using sensitive ~~of~~ info for malicious purposes

(Eg):

- ✓ Baitware
- ✓ Scareware
- ✓ Phishing
- ✓ Spear Phishing
- ✓ Pretexting

Baitware - Tricks people (by giving ~~false~~ fake offers of cash prize, etc.) into giving sensitive info

Scareware - phone/laptop ~~get~~ getting infected with virus

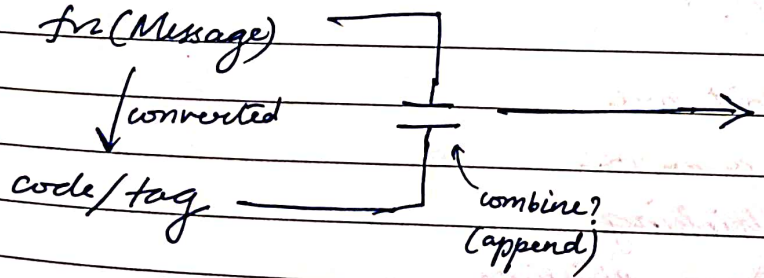
Phishing - Using fake websites to get sensitive info (generic and not customized)
↳ template is broadcasted to all users

Pretexting - Attacker pretends to be some genuine person you know to get sensitive info
(Attacker researches about the victim).

spear phishing - Identify potential victims
Carry out research about them and use it to create customized phishing attacks.
Can involve pretexting (to research is necessary to answer victims' qns).
Difficult to avoid.

social Engineering - Engineering on the psychology of victims to gather sensitive info.

Message Authentication

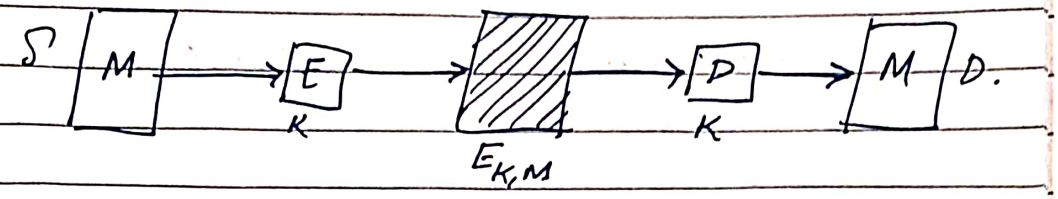


1. Hash fns
2. Encryption
3. Message Authentication Codes (MAC).

Hash fn: Maps any msg of arbitrary length to a fixed length number

2) Encryption:

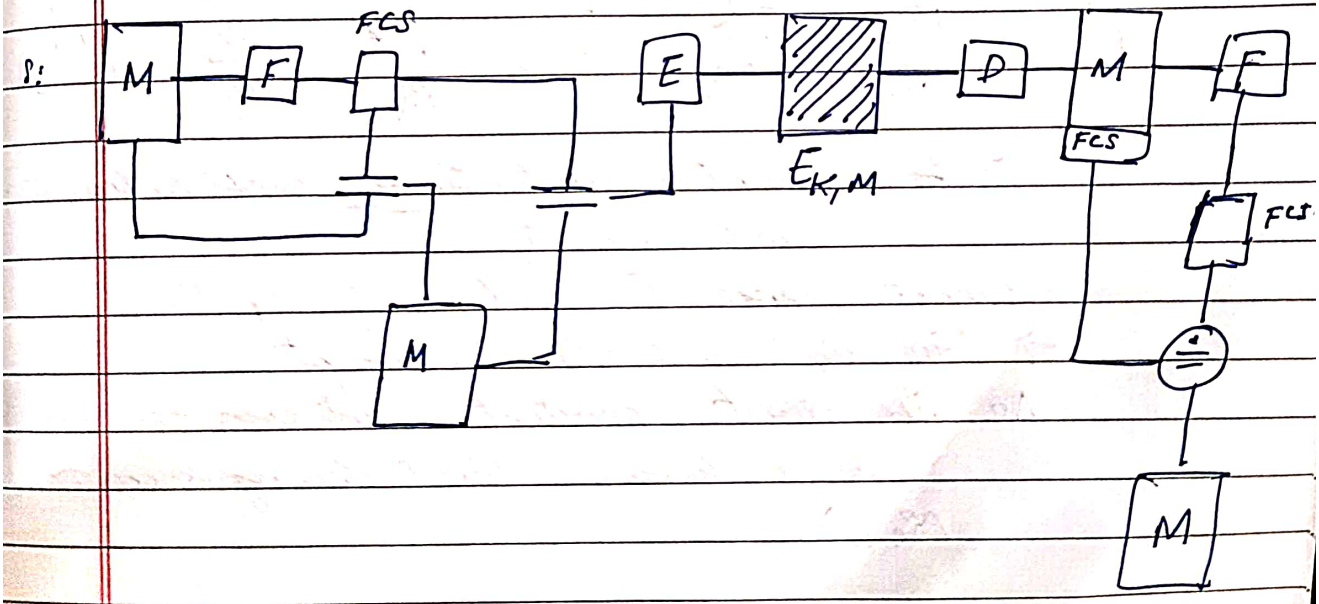
$S \xleftarrow[\text{(shared key)}]{K} \rightarrow D$ (symmetric)



⇒ offers both confidentiality & authentication (bcos key K is known only to S and D).

Frame Check Sequence (FCS):

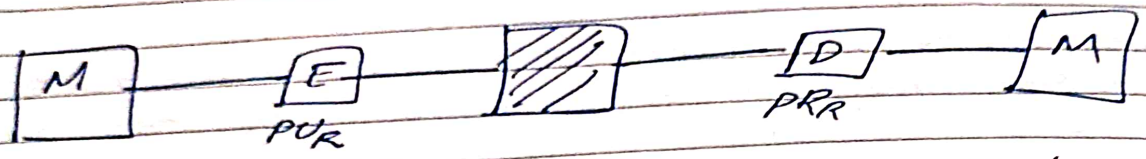
- ✓ similar to checksum
- ✓ Used to ensure that Message received at the destination is the correct/original message sent by S.
- ✓ We use another function (apart from encryption fn)
- ✓ This way we avoid revealing info about the msg and at the same time ensure correctness



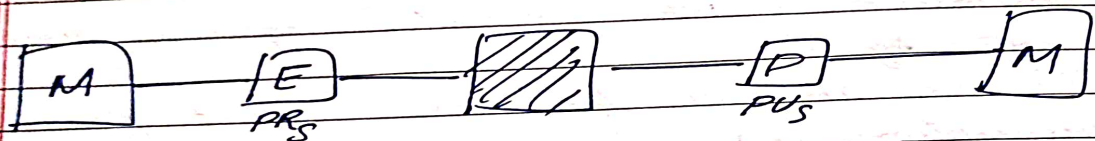
⇒ Internal error control mechanism (FCS before E ⇒ otherwise it is easier to

break the scheme because we can change the letters of the encryption

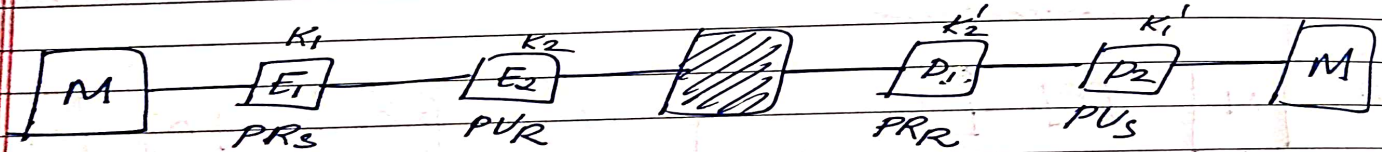
Using Asymmetric Cryptosystem



⇒ Only confidentiality is guaranteed
(because everyone has access to PU_R = Public key of receiver).



⇒ only authentication. (everyone can view the msg using PU_S)



⇒ to guarantees authenticity, confidentiality and security
(cannot change the order of E_1, E_2
⇒ first sign and then encrypt
No info about ciphertext – not even the sender can be known without having access to PR_R ??).

→ Many-to-one for (less vulnerable to crypto attacks ⇒ brute force becomes infeasible).

3) MAC = F(K, M).

↓

also called tag

(calculated & added to the msg - like checksum).

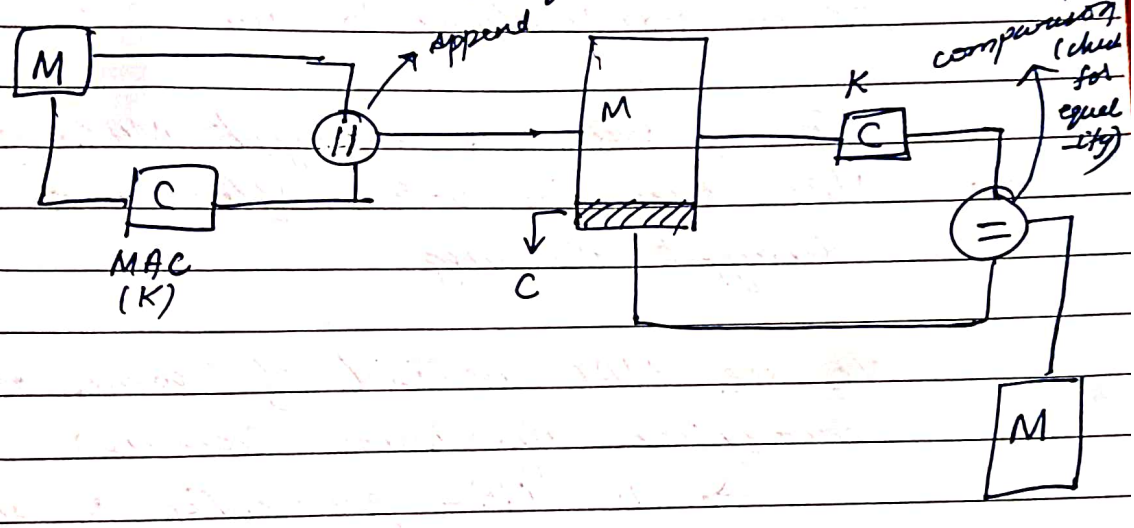
~~Effective for~~ only a small code appended to the msg so it does not increase the size of the msg much

04/09/22

C(K, M) = MAC

✓ a many-to-one mapping

✓ fixed length code cryptographic check sum / tag

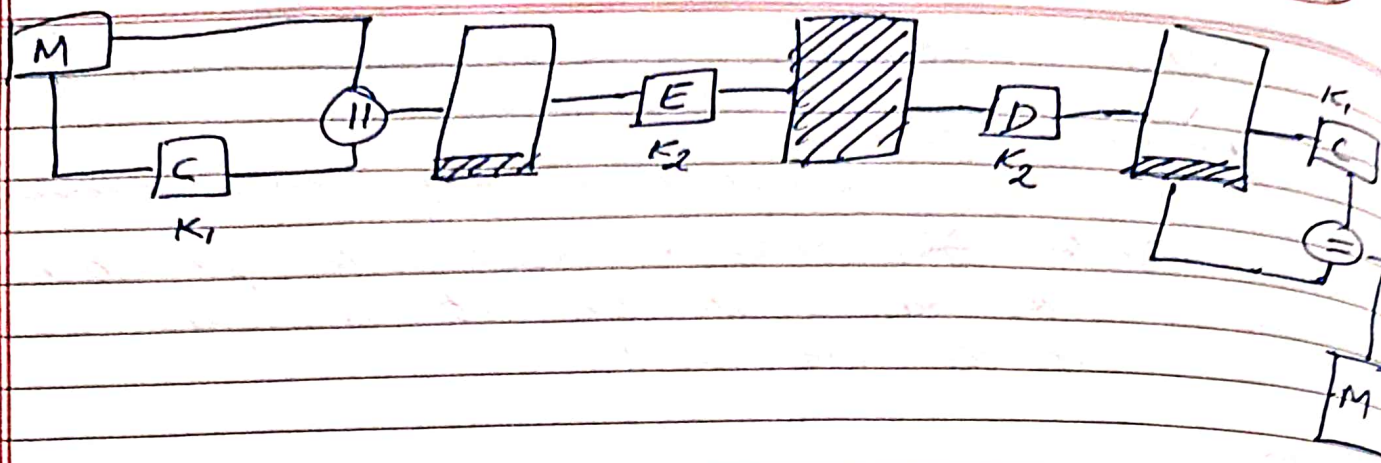


Sender & receiver knows key K.

→ Above scheme provides only authentication. For confidentiality, we should encrypt the message instead of sending it as such

⇓

uses another key

① 

② Encrypt first & then ~~find~~ do ~~the~~ MAC
⇒ but 1 is more popular
In 1, MAC is generated from orig msg
⇒ so modifications to msg ...?

?? In 2, once we decrypt MAC first, the MAC loses its value after ~~we~~ which we decrypt the msg

We use MAC instead of some other encryption algorithm because
⇒ 1. MAC generates a smaller code (~~cost~~ - cost-effective).

2. It is a many-to-one mapping

3. Most nodes - overloaded already
⇒ can't decrypt the MAC for every msg
so they perform a random sampling & ~~check~~ ~~the~~ decrypt the MAC only for a the sample and check the validity of the msg (they cannot do costly decryption operations either)
So MAC is preferred

Cryptanalysis of MAC

Importance of many-to-one mapping

Ciphertext C - is known (say).
but key K is not known.

Attack-1
Brute force attack

Let $|K| = k$, n = MAC length and $k > n$.

If $k = 100$ bit, $n = 10$ bit
we have 2^{100} keys and only 2^{10} msgs

On average $\frac{2^{100}}{2^{10}}$ msgs & ciphertexts are mapped to every possible message.

If attacker knows one pair of msg & MAC (M_1, MAC_1), he tries all possible ki to find the key used by MAC
↳ But he will still not be able to find the key because
$\frac{2^{100}}{2^{10}} = 2^{90}$ keys give the same MAC?

Now he uses knowledge of another pair (M_2, MAC_2) to find the key from the 2^{90} keys
$\Rightarrow \frac{2^{90}}{2^{10}} = 2^{80}$ keys still give the same MAC for both M_1 & M_2

$\Rightarrow$ He will have to try $2^{100} + 2^{90} + 2^{80} + \ldots + 2^{10}$ total trials
↳ we can stop when # keys = no. of bits of MAC bcos on avg only 1 key gives the MAC for each msg

Once we find a key, we still have to test the key on many more msgs (because we have only been considering the average case) to be sure that the key is correct.
⇒ This is true even if $k \leq n$

However, MAC is also prone to some attacks

Attacks based on structure of MAC

Consider $E(K, \Delta M)$ where E is DES scheme
where $\Delta M = M_1 \| M_2 \| \ldots \| M_e$
$\Delta M = M_1 \oplus M_2 \oplus \ldots \oplus M_e$
(XOR).

The attacker can come up with
$\Delta Y = Y_1 \| Y_2 \| \ldots \| Y_e \| \Delta M.$

(He only changes the message but not the MAC and sends ΔY, MAC to the dest receiver)

By properties of XOR
$MAC_{\Delta Y} = MAC_{\Delta M}$

So receiver will accept ΔY as the correct message because the MAC is ~~the~~ ~~as~~ the same for ΔY and ΔM.
⇒ Inherent flaw in MAC

Ensuring security of MAC

So we overcome this attack by ~~it~~ taking the following steps
1) uniformly distributing the MAC functions for n bit keys.

$Pr[MAC(M_i, K_i) = MAC(M_i, K_j)] = \frac{1}{2^n}$

⇒ key can be only one

2) It should be computationally difficult to ~~check~~ find K such that

$MAC(M_1, K) = MAC(M_2, K)$.

3) If M' is obtained from M by applying some fn f

i.e $f(M) = M'$

then $Pr(MAC(M, K) = MAC(M', K)) = \frac{1}{2^n}$

(It should be difficult to find key & and msg).

$min(K, n) = 128$

⇒ computational overhead for brute force attack is 2^{128} (difficult enough

Computation Resistance :

Finding if 2 messages were encrypted using the same key is as difficult as doing the brute force attack

If $min(K, n) = 128$

, complexity is $2^{128} \cong 10^{42}$

10^{43}s is the age of the universe

(so practically impossible to break it)

22/08/22

Cryptographic Hash function

Collision resistance
→ achieved by uniform distribution of key values
$P_r(k) = \frac{1}{n}$ (where n = total # keys)

MD5
produces 128-bit digest
→ Easier to break

Padding:

for MD5.

(message + 1 + as many 0s as reqd + (last 64 bits for storing length of message modulo 2^{64}) = such that entire size is ÷ble by 512)

Padding

128 for SHA-512 (everything else is the same)

Linear fns:

1. Additive property
$f(x+y) = f(x) + f(y)$

2. Homogeneity
$f(kx) = kf(x)$.

→ Easier to break
so we use non-linear fns for cryptography

(x) MUL, NOT, OR, AND → non-linear operations
XOR → linear operation?

MD5 :

512 bits

one ~~bit~~ block = 32 bits

⇒ # blocks = 16.

For each rounds, we do operations on 16 blocks

Do this for 4 rounds.

(using 4 different fns).

modulo ⇒ just left shift and discard the bits outside the block

A, B, C, D → states?

↳ fixed ~~initially~~ for an algo (like the key K).

circular shift ⇒ bring left most bits to right ex

implements ——?

1. Msg size is 2590 bits

Padding in SHA-512?

for 1 of padding bit

$2590 + 128 + 1 + x = 1024k$.

$x = 3072 - 2719$

$x = 353.$

10^{353} ⇒ 354 bits of padding

⊕ ⇒ modulo operation

128
11
117

25
1
27
30
27
3