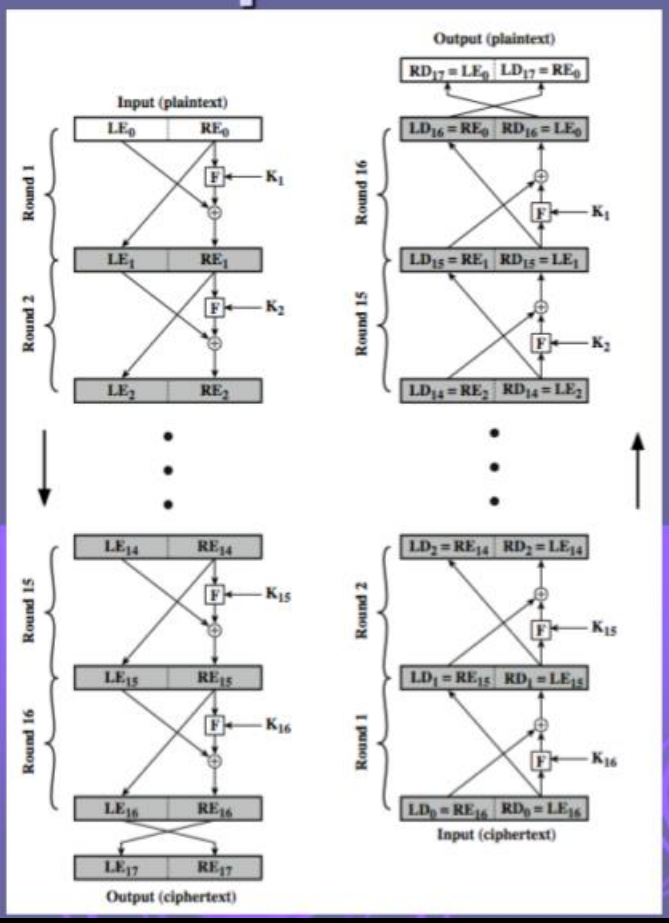
CNS

Feistel Cipher Structure



Data Encryption Standard - DES

64-bit - Plaint text

64 bit - key → (cast (parity) 8 bit drop)

↓

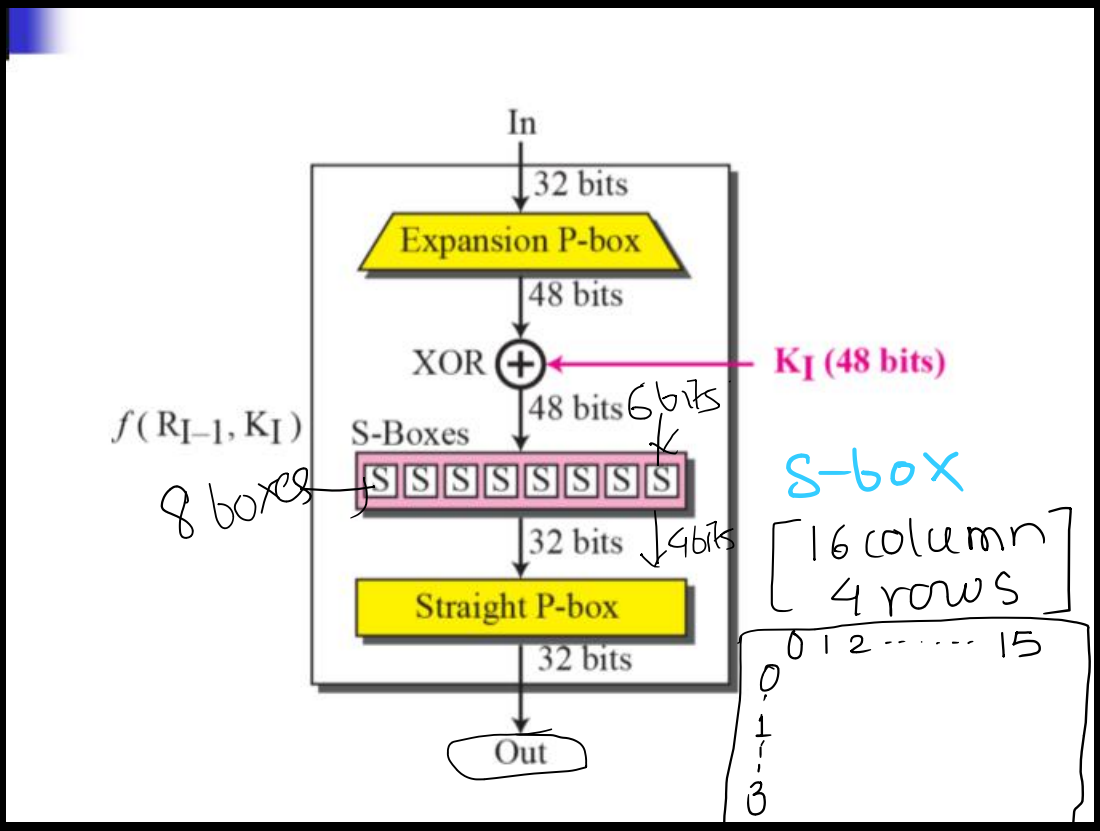
56 bit - Cipher key

↓

48 bit - 16 Round sub key

(Block cipher, Product Cipher

DES function



1) Ceaser Cipher

→ Substitution ciphers
→ Monoalphatic Ciphers (one to one char mapping)
→ $C = (P+k) \% 26$ } General Ceaser Cipher
→ $P = (C-k) \% 26$
→ In actual ceaser cipher, key=3
→ key range = {0, 1, ..., 25}

2) Playfair Ciphers

→ Substitution cipher
→ polyalphatic cipher
→ Playfair key matrix
5x5 matrix
→ fill letter of key (w/o duplicate)
→ then fill remaining letter in alphabetic order
→ keep i & J in same field

→ Make "diagram", i.e. group of two letters

→ avoid repeating some letter instead keep bogus char 'X'

→ At end of odd length string keep "z" as bogus char.

→ Now take each diagrams and see in key matrix to substitute the word accordingly

→ same row / or column then take next element in cyclic order

→ If diff^n then then take letter of same row but in column of other letter of pair/diagram

3) Vigenere Cipher

1) Polyalphabetic cipher
2) Substitution cipher.
3) key is repeated till length of plain text then use formula

$$C = (P_i + K_i) \% 26$$

* Transposition Cipher (keyless cipher)

→ only key used = length

1) Railfence cipher

→ Transposition cipher
→ numeric key = depth (is given)
→ write plain text in zig-zag order
→ Cipher: Read the text row wise

2) Columnar transposition Cipher / row transposition Cipher

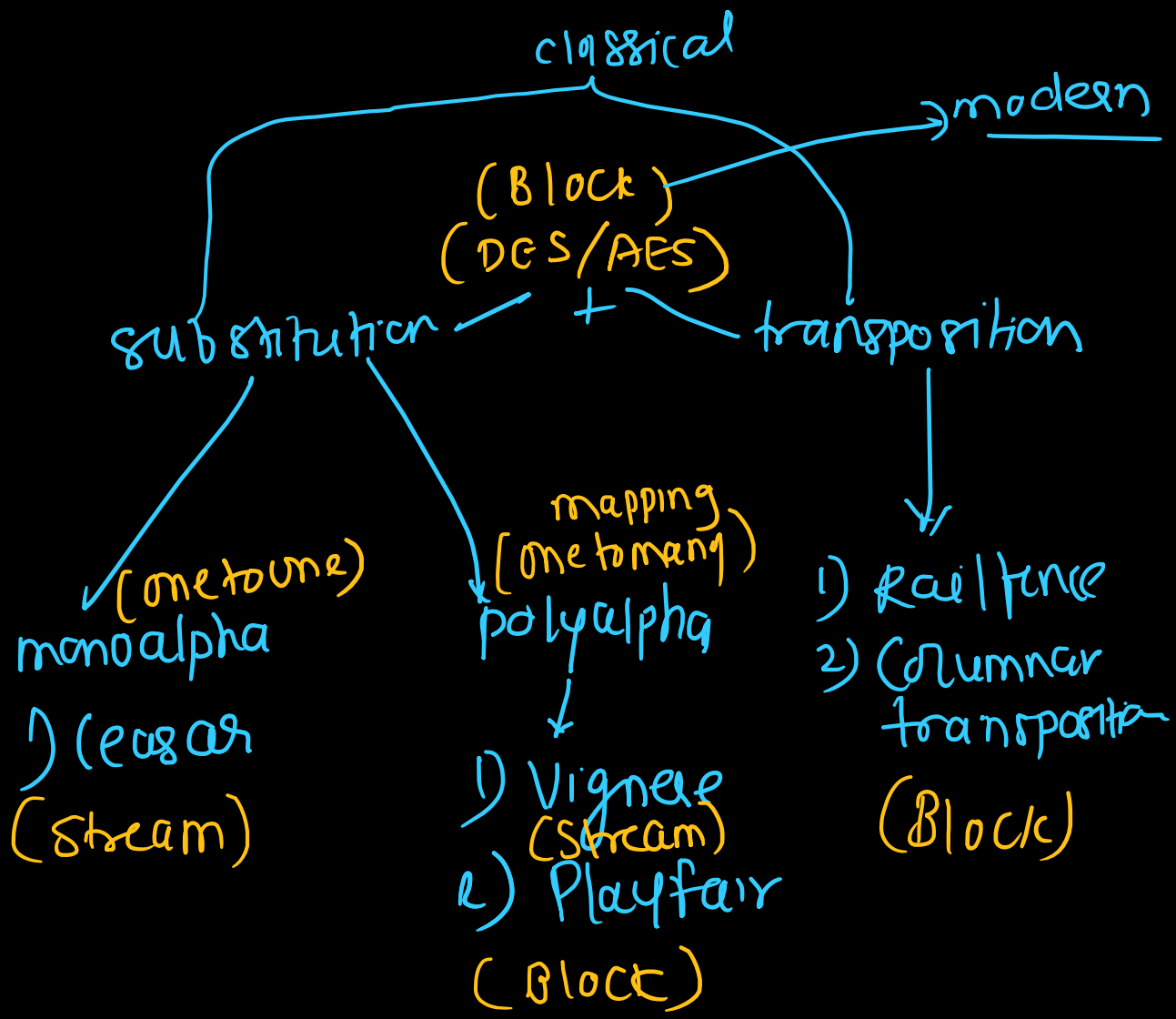
→ Transposition cipher
→ write plain text row wise
→ read cipher text column wise in an order given by key

→ numeric key giving (4 3 2 1 5) direct order. OR (src-ppt)

→ char key by which u make order (in ascending order of letter) (src-wikipedia)

Product Cipher - used in modern cipher

⇒ Combination of substitution & transposition
→ eg. DES, AES



AES

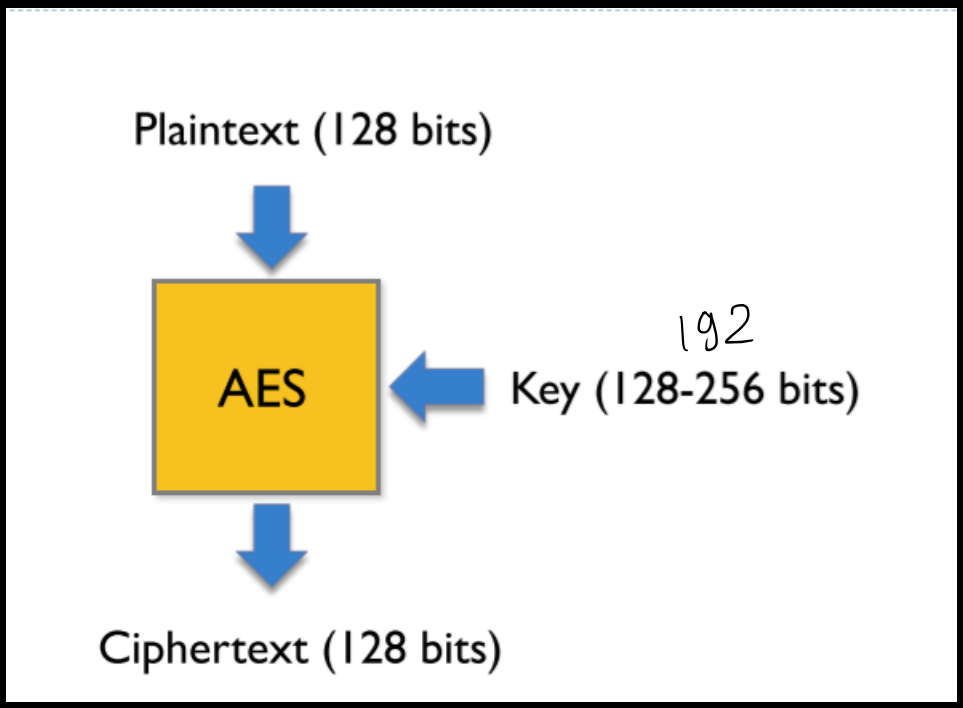
(Advance Encryption Standard)

1) for replacing DES (key size small)

2) Triple DES (slow, small block) 64 bit

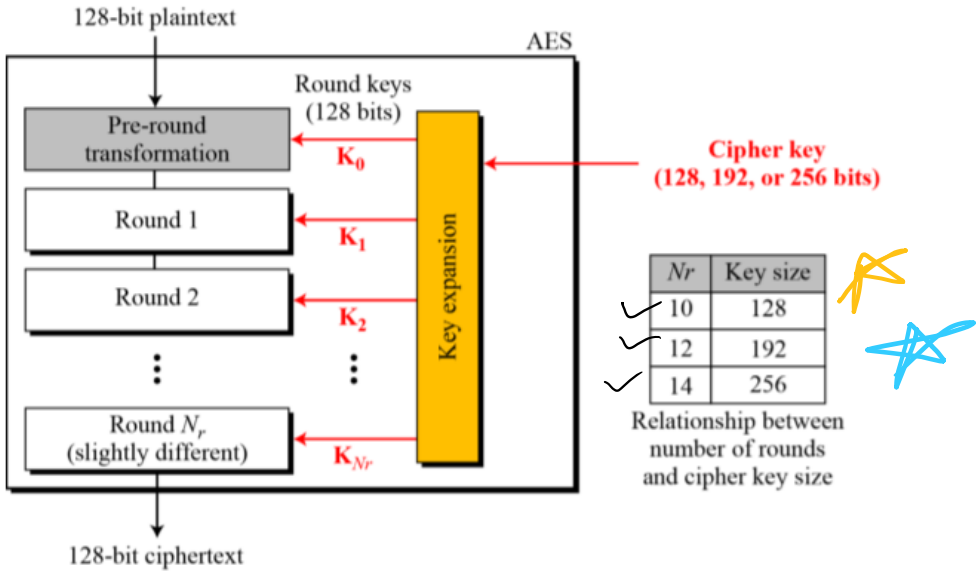
3) AES Cipher - (Rijndael cipher)

- Block size - 128 bit
- key (128 / 192 / 256 bit)
  - Round key - 128 bit (always)
- follows iterative approach rather than feistal as in DES
- process data as block of 4×4 matrix i.e 4 columns of 4 bytes (128 bit)
- operates on entire data block in every round



Multiple rounds

- Rounds are (almost) identical
  - First and last round are a little different



Nr	Key size
10	128
12	192
14	256

Relationship between number of rounds and cipher key size

High Level Description

Key Expansion

- Round keys are derived from the cipher key using Rijndael's key schedule

Initial Round

- AddRoundKey : Each byte of the state is combined with the round key using bitwise xor

Rounds

- SubBytes : non-linear substitution step
- ShiftRows : transposition step
- MixColumns : mixing operation of each column.
- AddRoundKey

Final Round

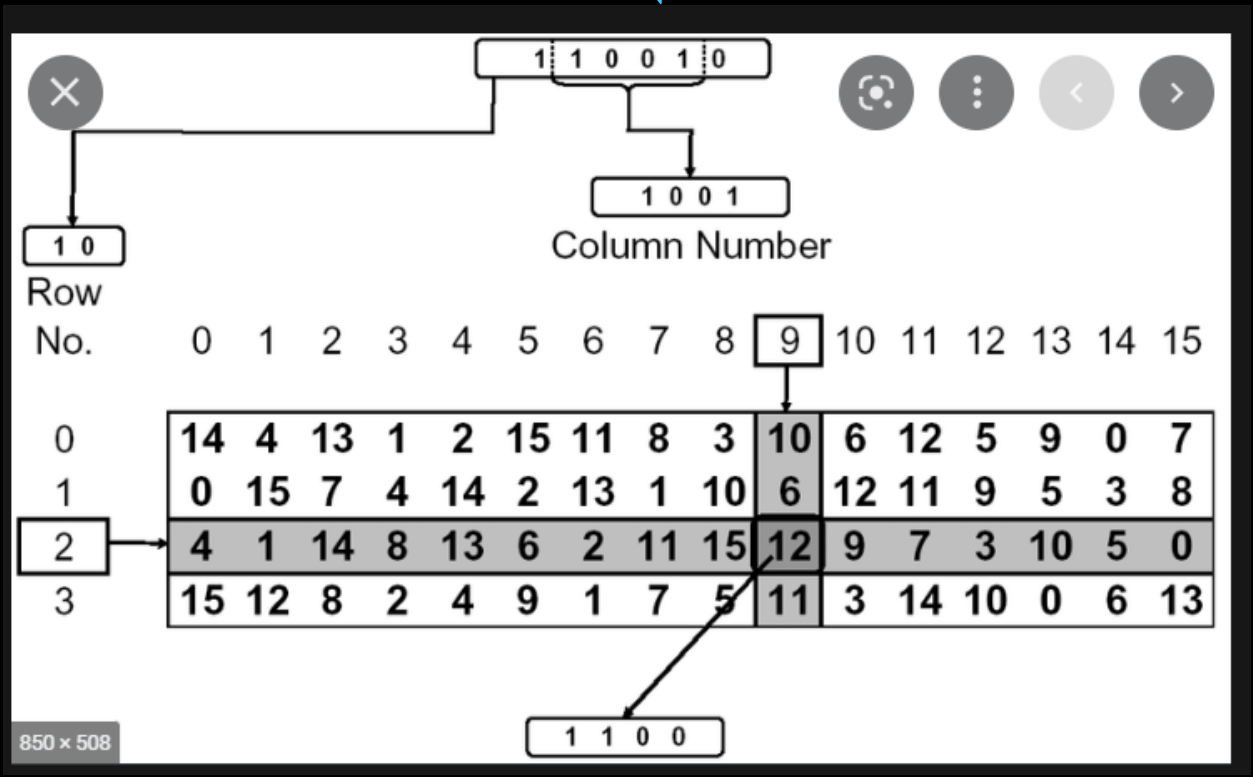
- SubBytes
- ShiftRows
- AddRoundKey

No MixColumns

DES


Round key 01
48 bits
Round key 02
48 bits
Round key 16
48 bits
Round-Key Generator
A complex combination of shifting, spliting, and combining units
Cipher key
56 bits
Parity drop
Key with parity bits
(64 bits)

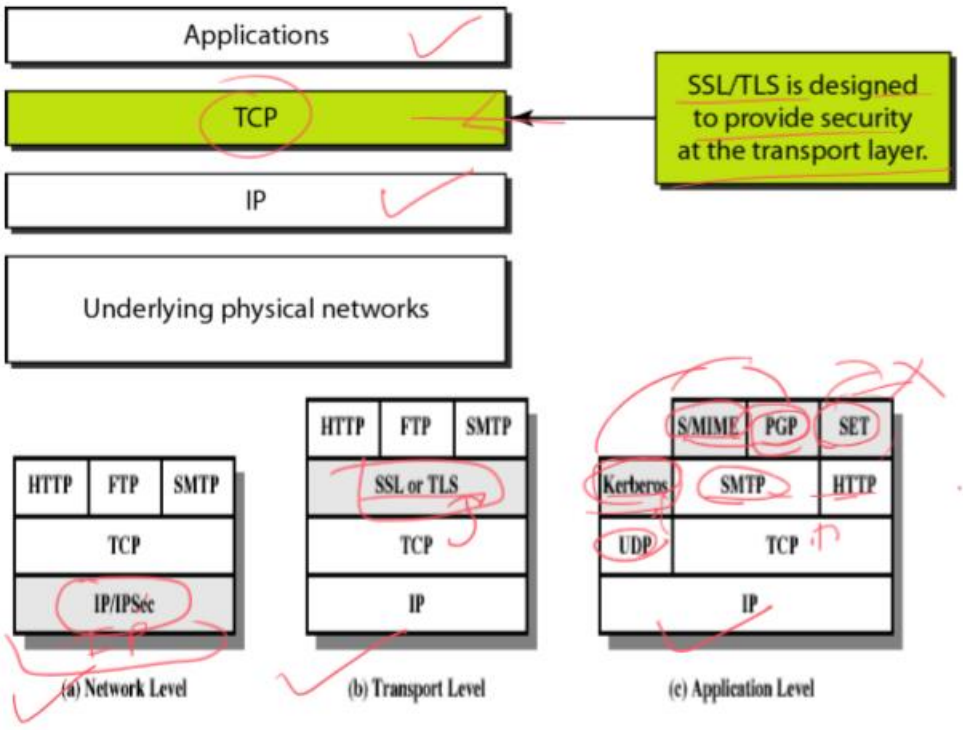
S-box


1 1 0 0 1 0
1 0 0 1
Column Number
1 0
Row
No. 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 14 4 13 1 2 15 11 8 3 10 6 12 5 9 0 7
1 0 15 7 4 14 2 13 1 10 6 12 11 9 5 3 8
2 4 1 14 8 13 6 2 11 15 12 9 7 3 10 5 0
3 15 12 8 2 4 9 1 7 5 11 3 14 10 0 6 13
1 1 0 0
850 × 508

- SSL - Secure Socket layer
- TLS - IETF version
  - ↳ Transport layer security
- Transport layer protocol

* SSL/TLS - both uses TCP protocol for reliability

Web Traffic Security Approaches



→ SSL - two layer protocol

4 protocol

1) Handshake protocol
2) Change Cipher Spec protocol
3) Alert protocol
4) Record protocol

Record protocol → confidentiality
↳ integrity

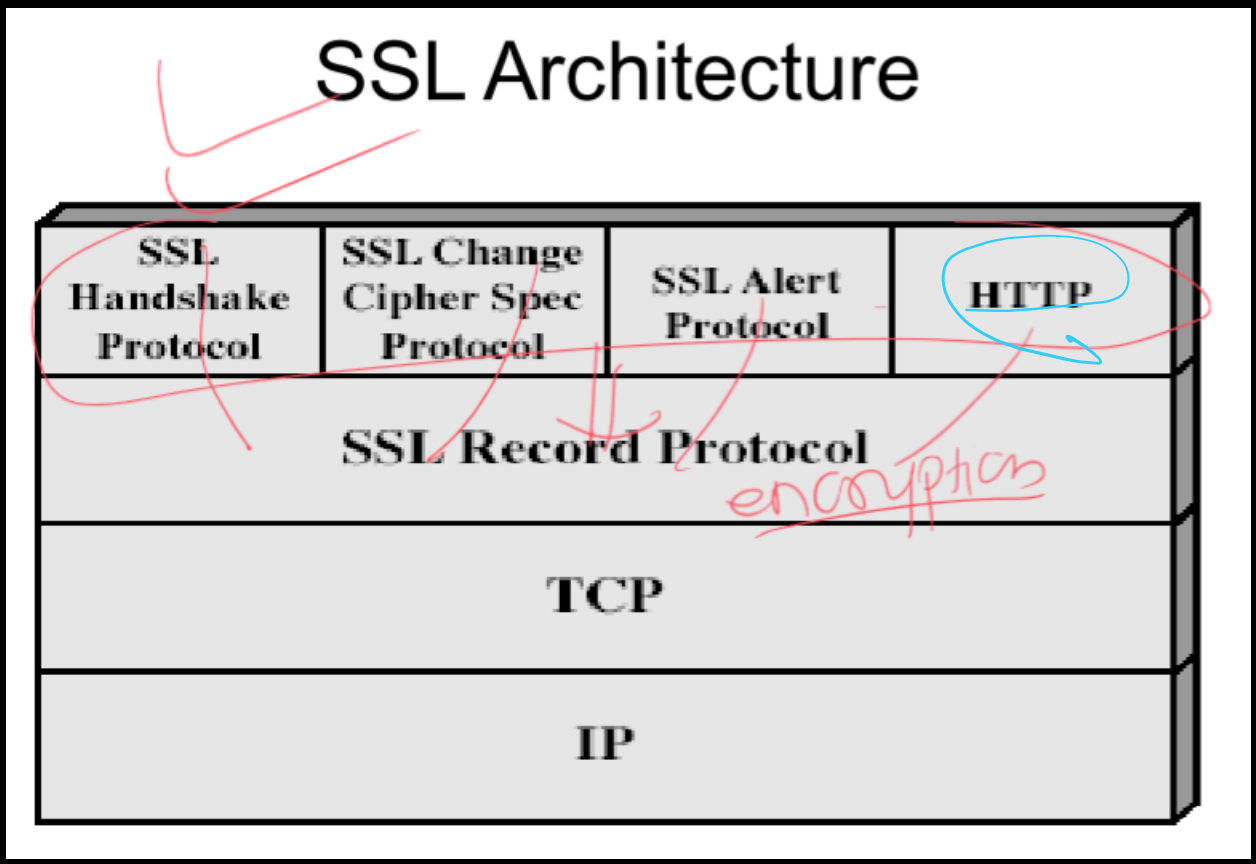
Msg → fragment → Compress (Lossless)
16384 bytes

Add MAC (Hash value)

encrypt (padding - optional)

SSL Record Header

↳ basic security service to higher layer protocol HTTP



management of SSL exchanges
↳ Handshake, ChangeCipherSpec, Alert

SSL Session state

- Session Identifier: byte sequence chosen by server
- Peer certificate X509.v3 certificate
- Compression method: compress prior to encryption
- Cipher Spec: data encryption algorithm and hash algorithm
- Master Secret: 48 byte secret
- Is Resumable: can session be used to initiate new connections

SSL session state

Change Cipher Spec Protocol

Cipher Spec

↳ current by updating algorithm (encryption and hash) to be used

---

SSL Alert protocol

↳ SSL related alert convey to peer node

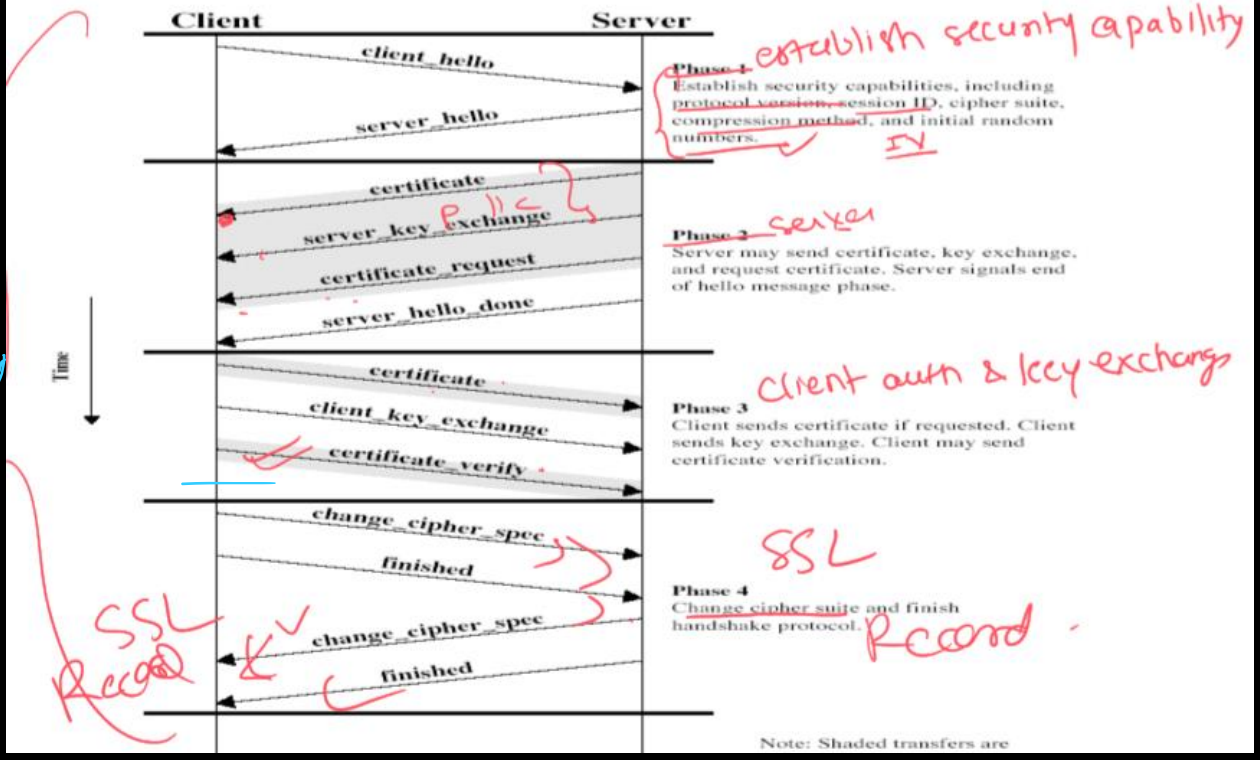
↳ uses Record protocol

→ Types of alert

→ unexpected msg, compression failure, bad record mac, handshake failure

→ no certificate, bad certificate unsupported certificate, certificate revoked, certificate expired, certificate unknown, close notify

SSL Handshake Protocol



Key exchange Methods

-)

IPSec - Network layer security

↳ not just a protocol

↳ it is a set of services and functions, and protocol

↳ services

1) confidentiality (encryp
2) Authentication & integrity
   ↳ Hashing + public key
   digital signature

3) Replay attack

4) decide of security algo & key

Transport mode

↳ end-host

Tunnel mode

1) end- Router/Gateway

SHA 512

1) ↳ SHA 2 group

SHA 256

used in bitcoin blockchain for hashing

Block size = 512 bits
Digest size = 256
Msg size = $2^{64}-1$

(Bandwidth save as value small)
word size = 32

SHA 512

↳ email address hashing, password hashing, digital record verification

Block size = 1024 bits
Length = $2^{128}-1$
Digest size = 512
word size = 64

TLS / SSL / PGP - SHA 256 algorithms

SHA 512

→ Initial hash value - 512 bits (8 words (64 bits each)) (Initial buffer)
→ Round constants - 80 word (each 64 bits)

→ Round - 80

1) Initial vector
(buffer value)

→ first 8 prime numbers
(2, 3, 5, 7, 11, 13, 17, 19)
ka square root
ka floating ka
64 bits ka their
fraction part

→ for storing the chaining process

2) 80 Round constant

80 prime ka cuberoot
ka first 64 bits ka
fractional part

3) 80 words from 1024 bit block i.e. from 16 words

→ ① starting 16 words
i.e. W[0] - W[15] ← 16 words
→ dividing 1024 bits block into 16 words of 64 bit each.

② W[16] - W[79]

↳ sigma1 | sigma2

↳ loop from $i = 16$ to $i <= 79$

sigma1 = (w[i-15] right rotate 1) XOR (w[i-15] RR 8) XOR (w[i-15] RR 7)

sigma2 = (w[i-2] RR 19) XOR
(w[i-2] RR 61) XOR
(w[i-2] RRG)

→ **w[i] = w[i-16] + sigma1 + w[i-7] + sigma2**

1. Input formatting
2. Hash buffer initialization
3. Message Processing ⟹ round const + schwards
4. Output

① 3 part of formatted msg

→ original msg. ($2^{128}-1$) (size limit)
→ padding (start from 1 then 0's)
→ length (128 bits) of original msg

3) Message Processing

- Round constant
- Working

① formatted msg divide into n blocks each of size 1024 bits.

② Then first block along with initial hash value given to compression func having round func and addition func in H

③ In msg processing before going round func we

derive 80 words from 1024 bit block,

④ There is total 80 rounds in round func. each round take one word, one round constant and previous block hash value or IV.

⑤ each round has majority func, conditional func and Rotation func.

⑥ output from round func is added to previous

hash value.

7. And the final output is stored hash buffer register and give to next block

8. It goes on till last block, whose output is the final hash value for the msg.

Kerberos

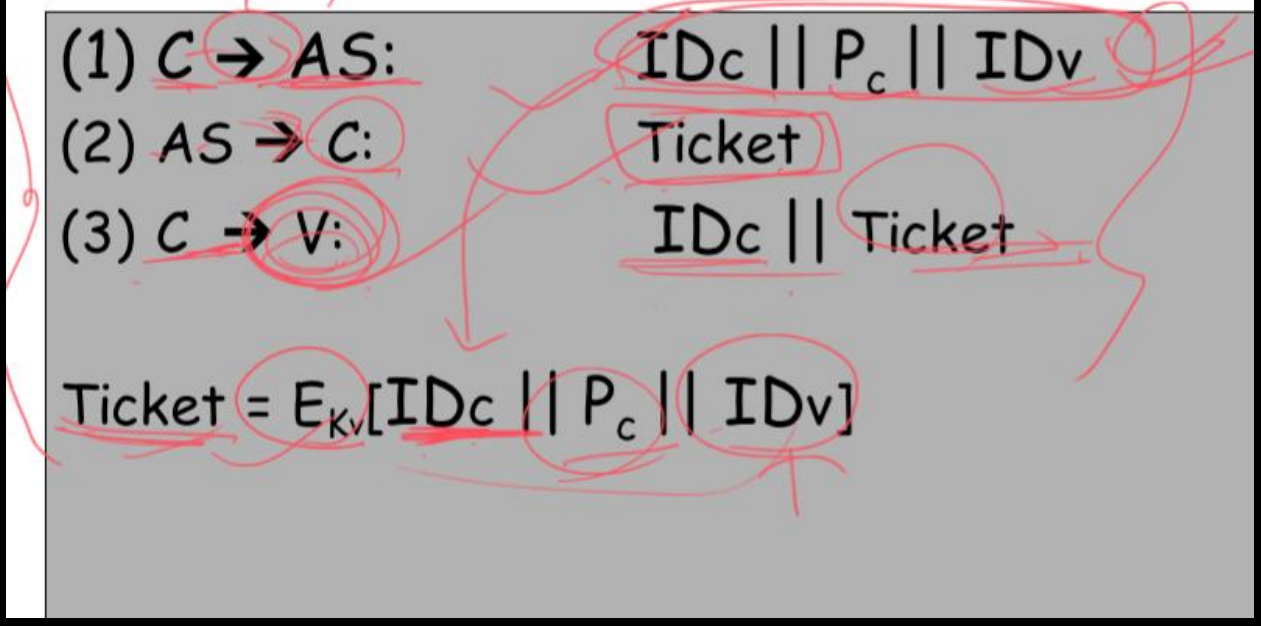
→ Authentication application

→ centralized authentication server.

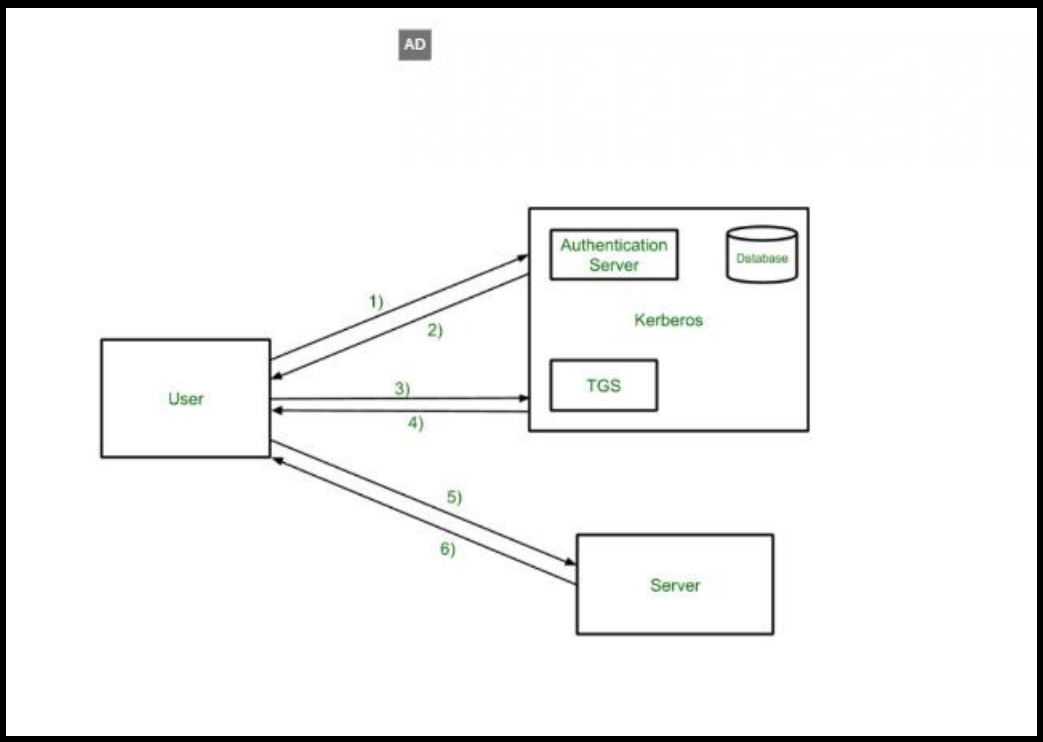
→ use symmetric key encryption (Two version 4 & 5)

→ version - 4 - DES

A Simple Authentication Dialogue



- **Step-1:**
  User login and request services on the host. Thus user requests for ticket-granting service.
- **Step-2:**
  Authentication Server verifies user's access right using database and then gives ticket-granting-ticket and session key. Results are encrypted using the Password of the user.
- **Step-3:**
  The decryption of the message is done using the password then send the ticket to Ticket Granting Server. The Ticket contains authenticators like user names and network addresses.
- **Step-4:**
  Ticket Granting Server decrypts the ticket sent by User and authenticator verifies the request then creates the ticket for requesting services from the Server.
- **Step-5:**
  The user sends the Ticket and Authenticator to the Server.
- **Step-6:**
  The server verifies the Ticket and authenticators then generate access to the service. After this User can access the services.


AD
Authentication Server
Database
Kerberos
TGS
User
1)
2)
3)
4)
5)
6)
Server

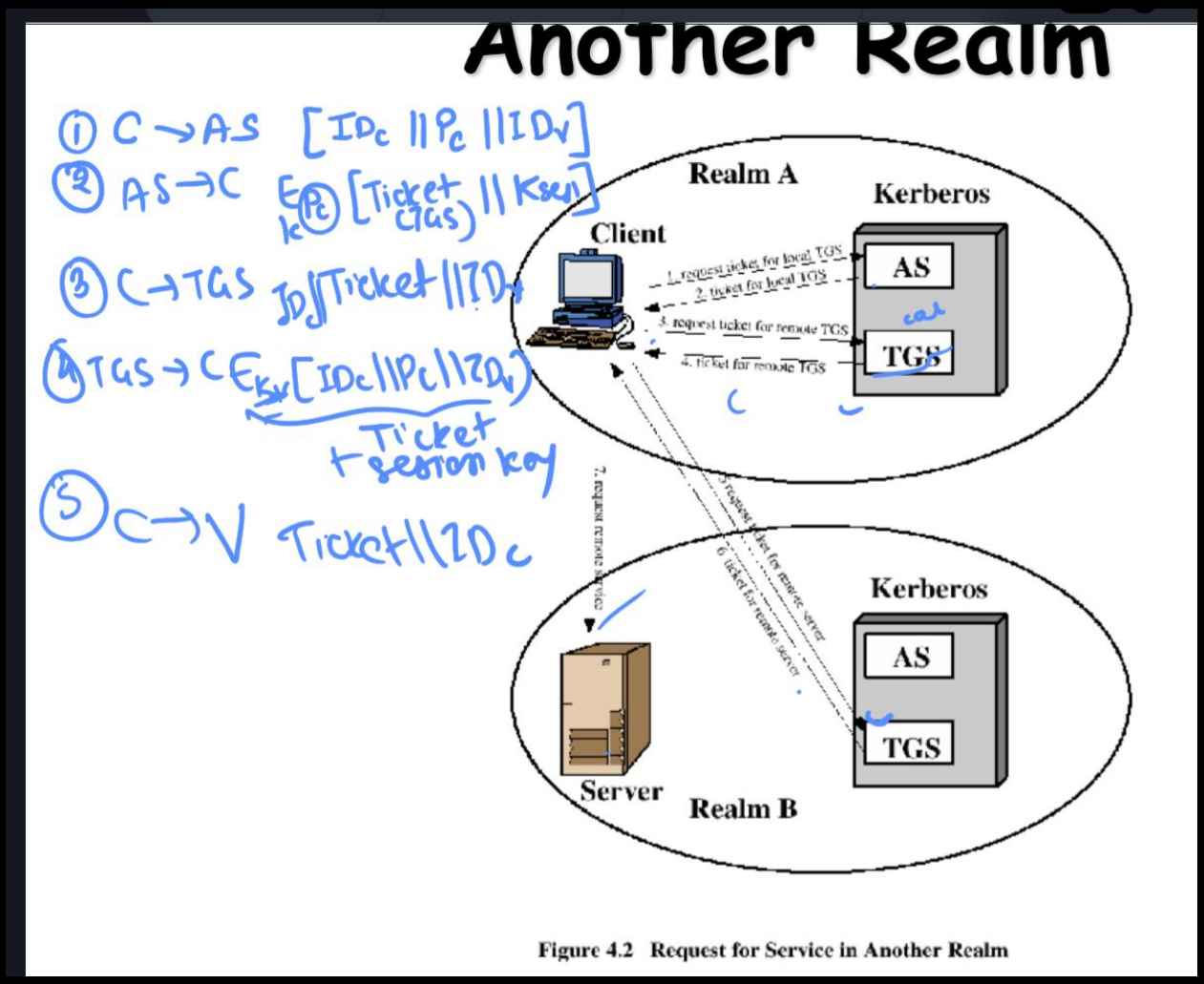


Figure 4.2 Request for Service in Another Realm

S.No.	Kerberos Version 4	Kerberos Version 5
1.	Kerberos version 4 was launched in late 1980s.	Kerberos version 5 was launched in 1993.
2.	It provides ticket support.	It provides ticket support with extra facilities for forwarding, renewing and postdating tickets.
3.	Kerberos version 4 works on the Receiver-makes-Right encoding system.	Kerberos version 5 works on the ASN.1 encoding system.
4.	It does not support transitive cross-realm authentication.	It supports transitive cross-realm authentication.
5.	It uses Data Encryption Standard technique for encryption.	It uses any encryption techniques as the cipher text is tagged with an encryption identifier.
6.	In Kerberos version 4, the ticket lifetime has to be specified in units for a lifetime of 5 minutes.	In Kerberos version 5, the ticket lifetime is specified with the freedom of arbitrary time.

X.509 Auth service

1) version
2) certificat

Mod-5 electronic mail security

PGP → S/MIME

↳ Pretty good privacy

S/MIME ↑ IETF standard version of PGP

(open source)

give confidentiality + authentication

for email + file storage applications

① Authentication only

→ msg → hash

↓

e_k [hash]
private

↓

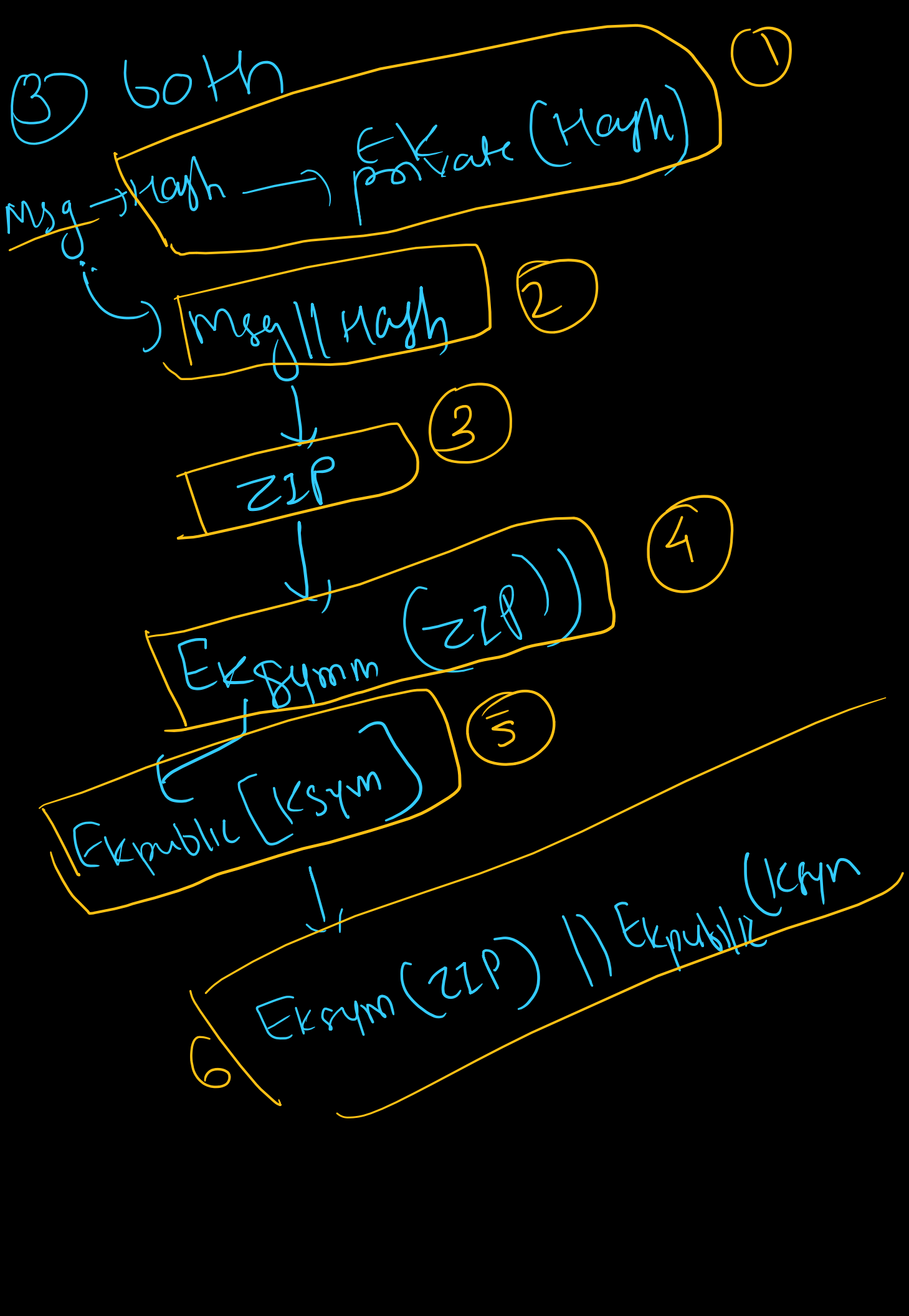
msg || (Hash)

② Confidentiality

msg → compress (zip)

→ E_{Ksym}(msg)

$E_{public}(K_{sym})$


(3) both
msg
Hash
$E_{Kprivate}$(Hash)
①
msg || Hash
②
ZIP
③
E_{Ksym}(ZIP)
④
$E_{Kpublic}$[Ksym]
⑤
⑥
E_{Ksym}(ZIP) || $E_{Kpublic}$(Ksym

① Compression

→ ZIP

② → email compatibility

→ use radix 64

③ segmentation & reassembly

S/MIME

→ Secure / Multipurpose Internet Mail Extensions
↳ industry standard

→ PGP — for personal use

→ SMTP

:exe /img/video

MIME

↳ support other binary also

IPSec

↳ For network layer security

→ Two modes

↙ tunnel mode

IP header also encrypted

↘ transport mode

only datagram from transport layer encrypted

→ It is a set of protocol & services

→ Two protocol

① AH — Authentication Header

② ESP — encapsulating security payload

→ IPsec Provide

① Authentication
② confidentiality
③ Integrity
④ network security
Rom attack like
replay attack

→ IPsec operation

① setting up secure path between two devices that want secure communication among many insecure intermediate systems.

② Device agree of set of security protocol so it can understand format of data received and send. (AH or ESP)

③ Device decide encryption algorithm
ESP → DES / AES / TDES

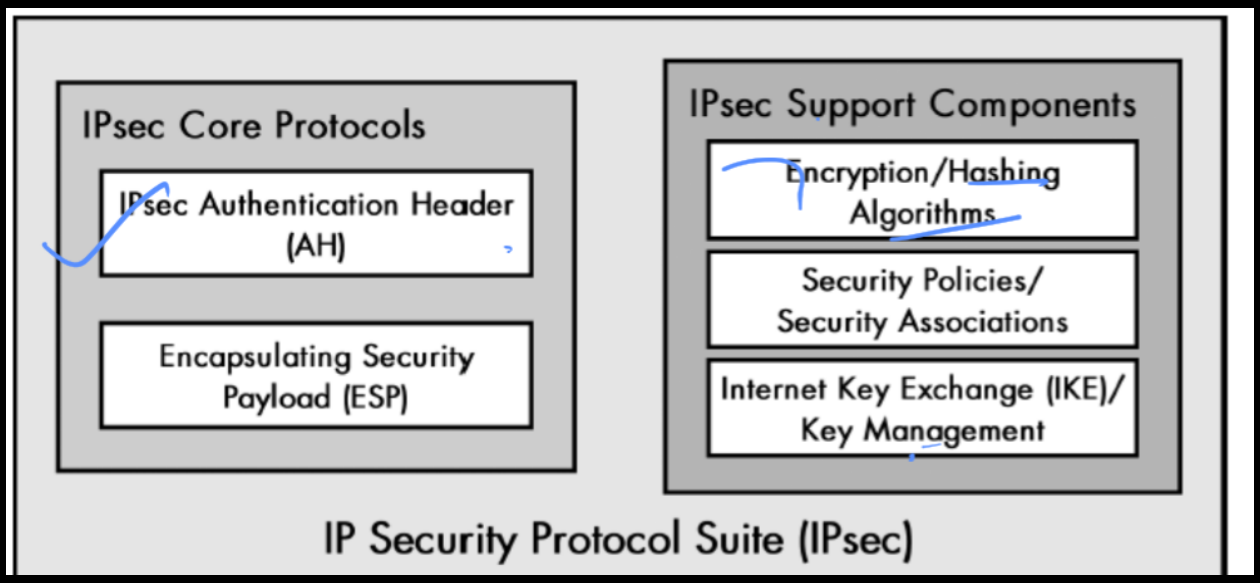
④ Device exchang key

⑤ provide confidentiality & authentication to IP layer.

IPsec Core protocols

2 protocol

1) AH — integrity
2) ESP — integrity + confidentiality (privacy)



Ipsec Architecture

Host — ipsec on host/client

Router

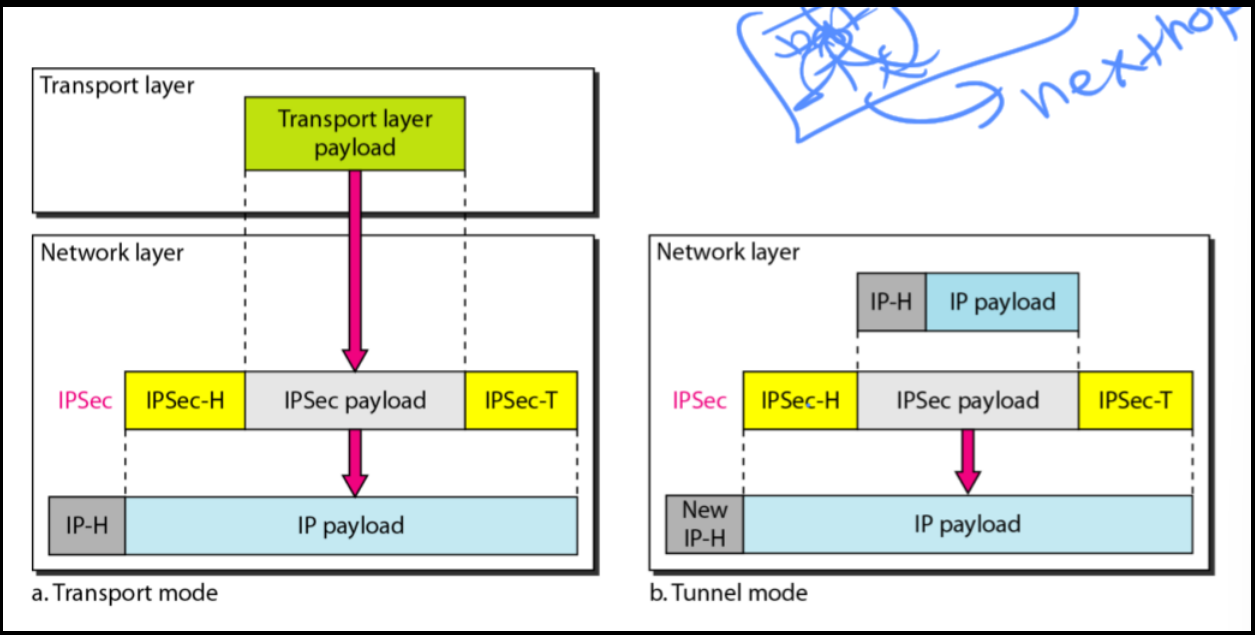
Ipsec Mode

1) Transport-mode

↳ only transport layer payload is protected by AH/OR ESP and its header are added

↳ It does not protect IP header

↳ It is used in host-to-host data protection.



33333333333333333

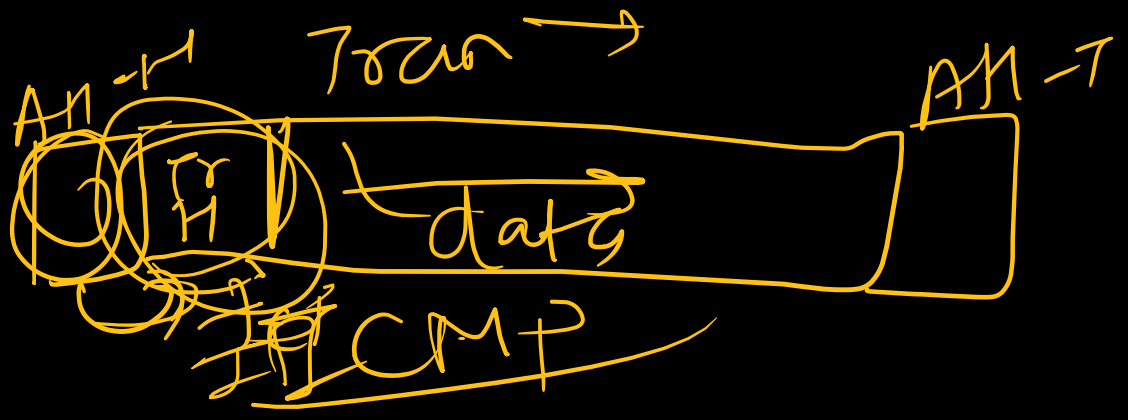
2) Tunnel mode

→ It protect IP payload along with original IP header.

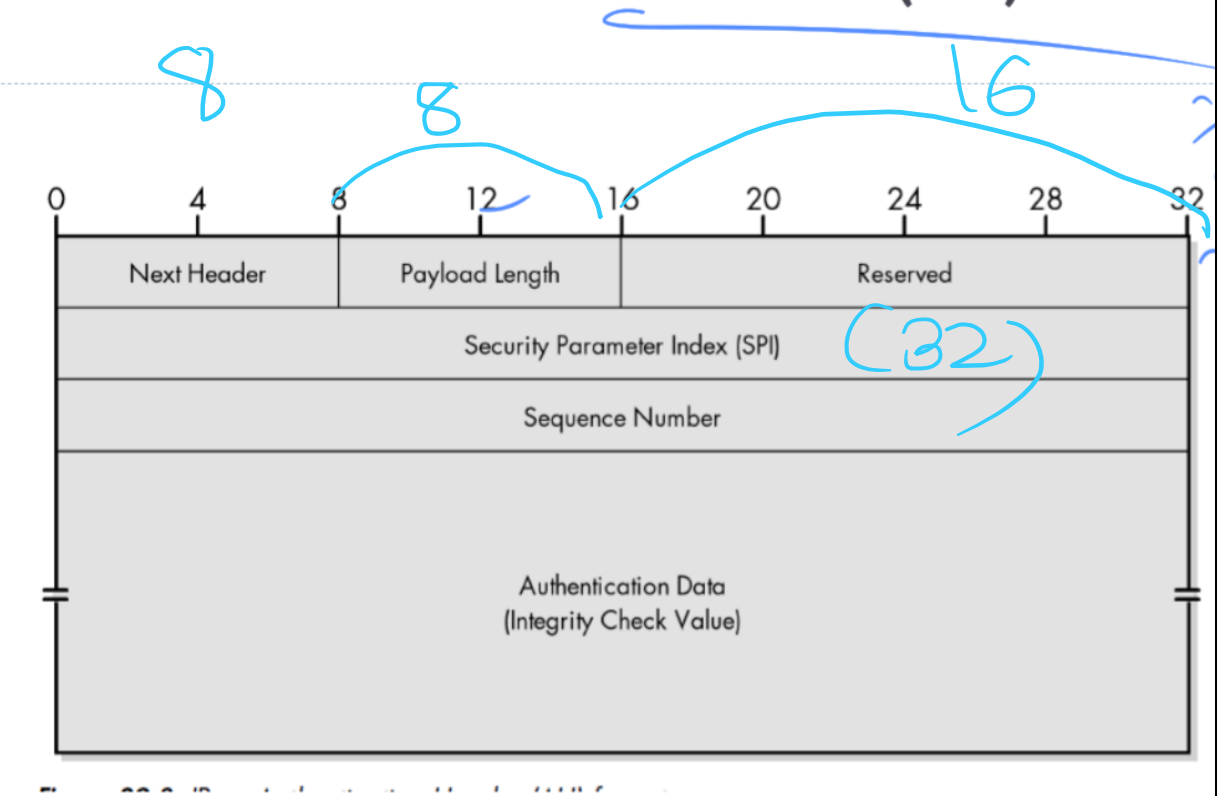
IPsec AH

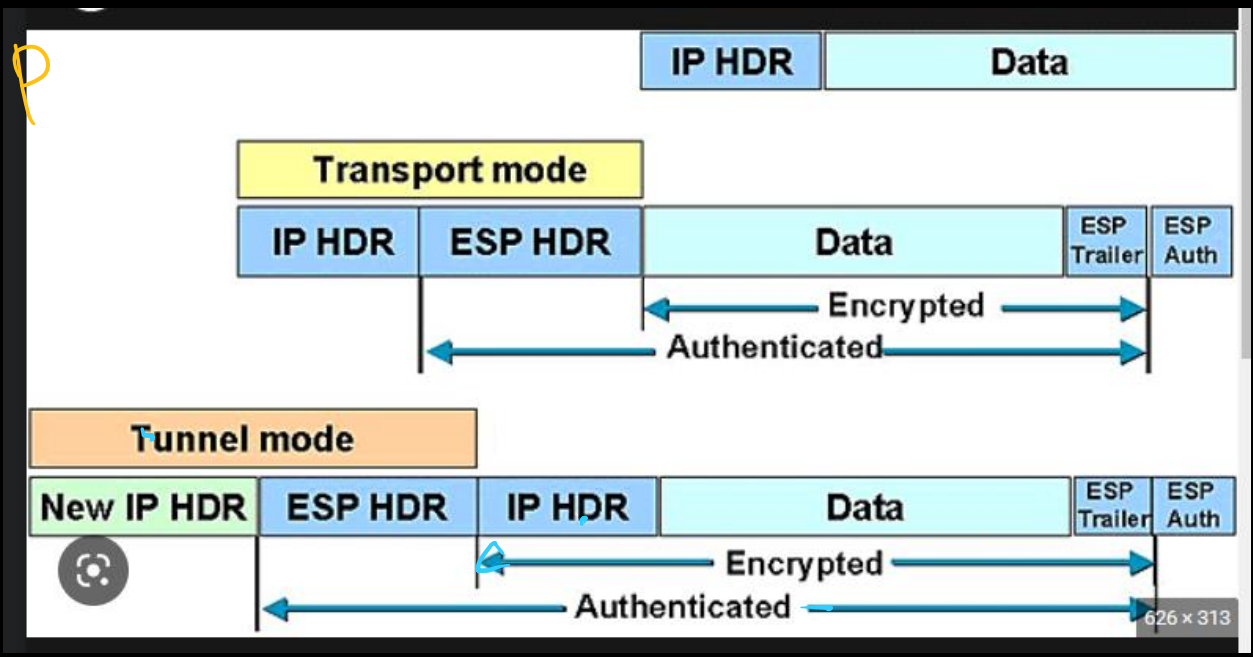
1. Header

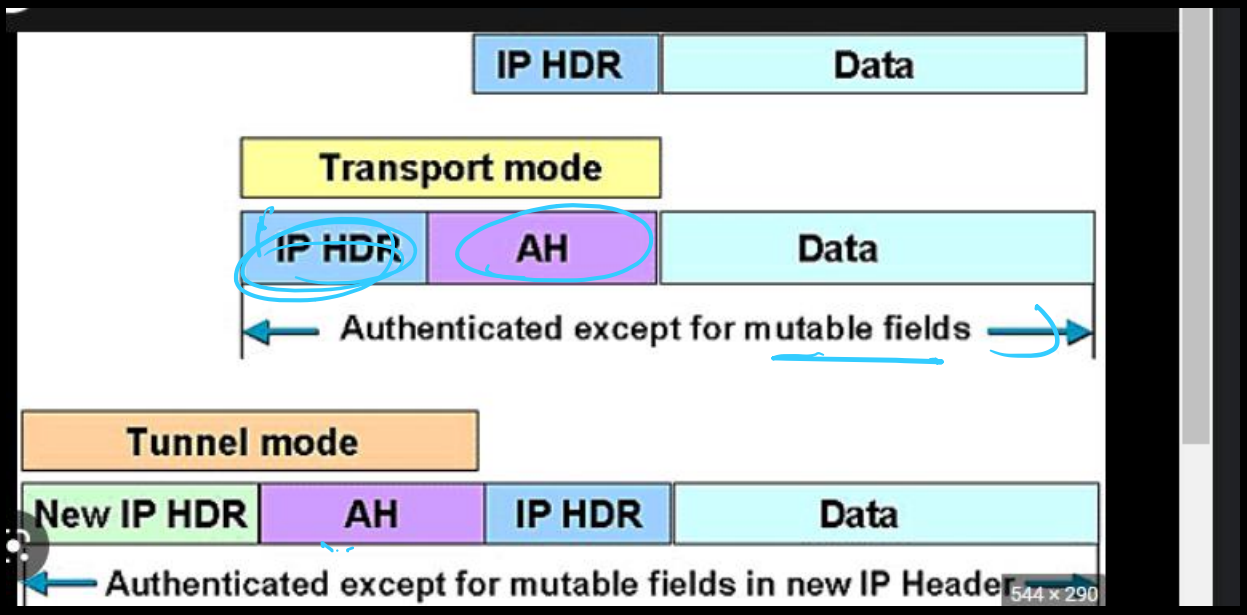
2. Next header



IPSec Authentication Header (AH)







IP datagram

↳

IP header data

ESP

authentication, integrity | privacy

- add header & trailer.

Transport Mode vs. Tunnel Mode

Transport Mode	Tunnel Mode
IP payload is encrypted	IP payload is encrypted
IP header is not encrypted	IP header is encrypted
Original IP header is used for routing decisions	New IP packet encapsulates the encrypted one with a new header that is used for routing decisions
Provides protection for the payload from end to end	

Encapsulating Security Payload (ESP) Protocol in transport mode

- ***Security parameter index:*** the 32-bit security parameter index field is similar to that defined for the AH protocol.
- ***Sequence number:*** the 32-bit sequence number field is similar to that defined for the AH protocol.
- ***Padding:*** this variable-length field (0 to 255 bytes) of 0s serves as padding.
- ***Pad length:*** the 8-bit pad length field defines the number of padding bytes.

Services	AH	ESP
Access control	Yes	Yes
Message authentication (message integrity)	Yes	Yes
Entity authentication (data source authentication)	Yes	Yes
Confidentiality	No	Yes
Replay attack protection	Yes	Yes

Security Association

→ It is a mechanism used by IPsec to decide security parameter between nodes during the connection.

→ This mechanism used by IPsec make IP connection oriented protocol

Security Association

→ A set of security parameter established between sender & receiver during first time of their communication.

→ Two type

① Inbound SA

From	Protocol	Authentication	Encry

② outbound SA

To	Protocol	Authentication	Encryption.

Security Association Database (SADB)

→ SADB is a collection of SAs.

→ It is 2D table having SA define in single row.

→ two SADBs - inbound and outbound.

SPI

↳ It is parameter to differentiate among SA in SADBs.

→ SPI - (destination/source add) and protocol (AH or ESP)

Services provided by IPsec

① Access Control

→ Provide access control indirectly by using SADB.

② Message Authentication

→ provide message authentication using authentication data for preserving integrity of message.

③ Entity Authentication

Authentication of sender is done by using SA and keyed-

hash value of data sent by sender is both AH and ESP

④ Confidentiality

ESP provide confidentiality by encrypting message.

AH doesn't provide Confidentiality

⑤ Replay attack protection.

→ using sequence number replay attack is prevented by both ESP and AH.

How to prevent IP spoofing

IP spoofing easily done when there trust relationship exist between machine.
(i.e. mostly in local network)

→ (1) So to prevent IP spoofing, we can use access control list to deny private IP address

(2) Filtering both inbound & outbound traffic.

(3) Reject the packet coming from outside of local network that have source IP address of inside by configuring router and switches

(4) Use protocol like IPsec to reduce the risk of spoofing.

(5) Enable encryption session on router so that trusted hosts outside network can communicate securely with local hosts

Routing (RIP) Attack

RIP = Routing Information Protocol

① ↳ It is used to share routing info such as shortest-paths, and advertising routes/path out from local network.

② ↳ In original version of RIP, there was no built in authentication for verifying the information provided in RIP packet.

↳ Using this bottleneck, attacker forge RIP pk by claiming his host "X" has fastest path out of network. Due to which All packets going out from local network are routed through X, where they could be modified or examined.

How to prevent / mitigate RIP attack.

→ New version of RIP has simple password authentication algorithm, making RIP attack harder to happen.

② IPsec can be used for authentication and encryption of each IP packet transfered during communication session.

ICMP Attack

Internet control Message protocol.

→ It is used by network devices like routers to send error messages

→ There is no authentication is ICMP, which give rise to ICMP attacks resulting as denial of services or packet interception is done by attackers

In ICMP attack,

① Attacker send forged ICMP-echo packet to vulnerable's network broadcast address.

② All system present in vulnerable network send ICMP echo replies to victim (system), consuming the resources of target/victim system resulting in denial of servic (DOS) to legitimate traffic.

Mitigation / P

Intrusion Detection

① Signature based