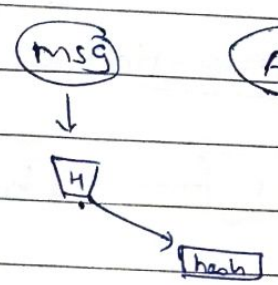
Network Security

PAGE No
DATE / /

* Hash fxn.



H ⇒ Hash fxn
h ⇒ Hash value

Applications:
1. Message authentication
2. Digital signature

Hash fxn is used to generate hash value of message

$$h = H(m)$$

if the hash value is assigned to a single message.

[No two hash have same message and no two message have same hash value]

→ Collison free property

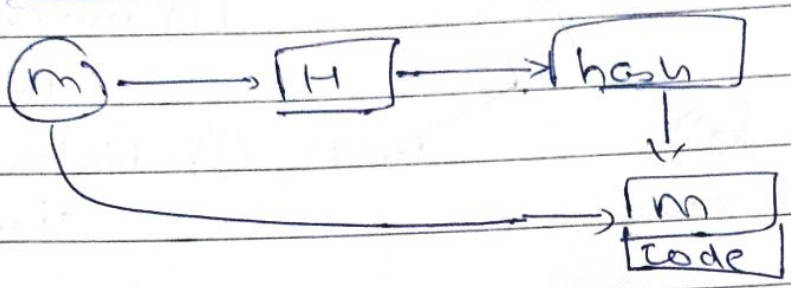
One-way property:

We can decode a message's hash value from hash fxn but,
we can't decode a original message through the hash value.

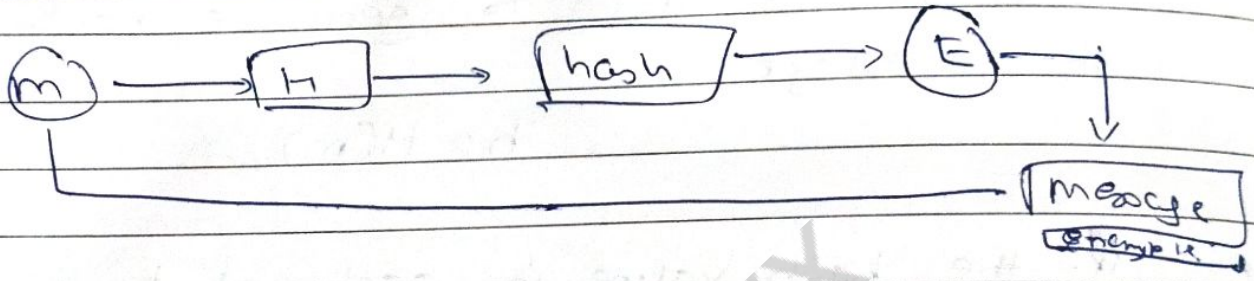
Confidentiality is achieved by encrypting message (E)

Authentication is achieved by hashing the message with a code
i.e concatinating original msg with hash value.

* Message authentication : (only authentication)



* Digital signature : (both authentication & confidentiality)



Note : Even value of 1 in XOR => result => 0
Odd " " " " " => result => 1

* Characteristic of hash fxn => important topic

H must have :

(1) variable i/p size
↳ H can be applied to any block of data

(2) fixed o/p size :
↳ produce fixed o/p.

(3) Efficiency :
↳ H(x) is easy to compute for any(x)

PAGE No.
DATE / /

(4) One way property:

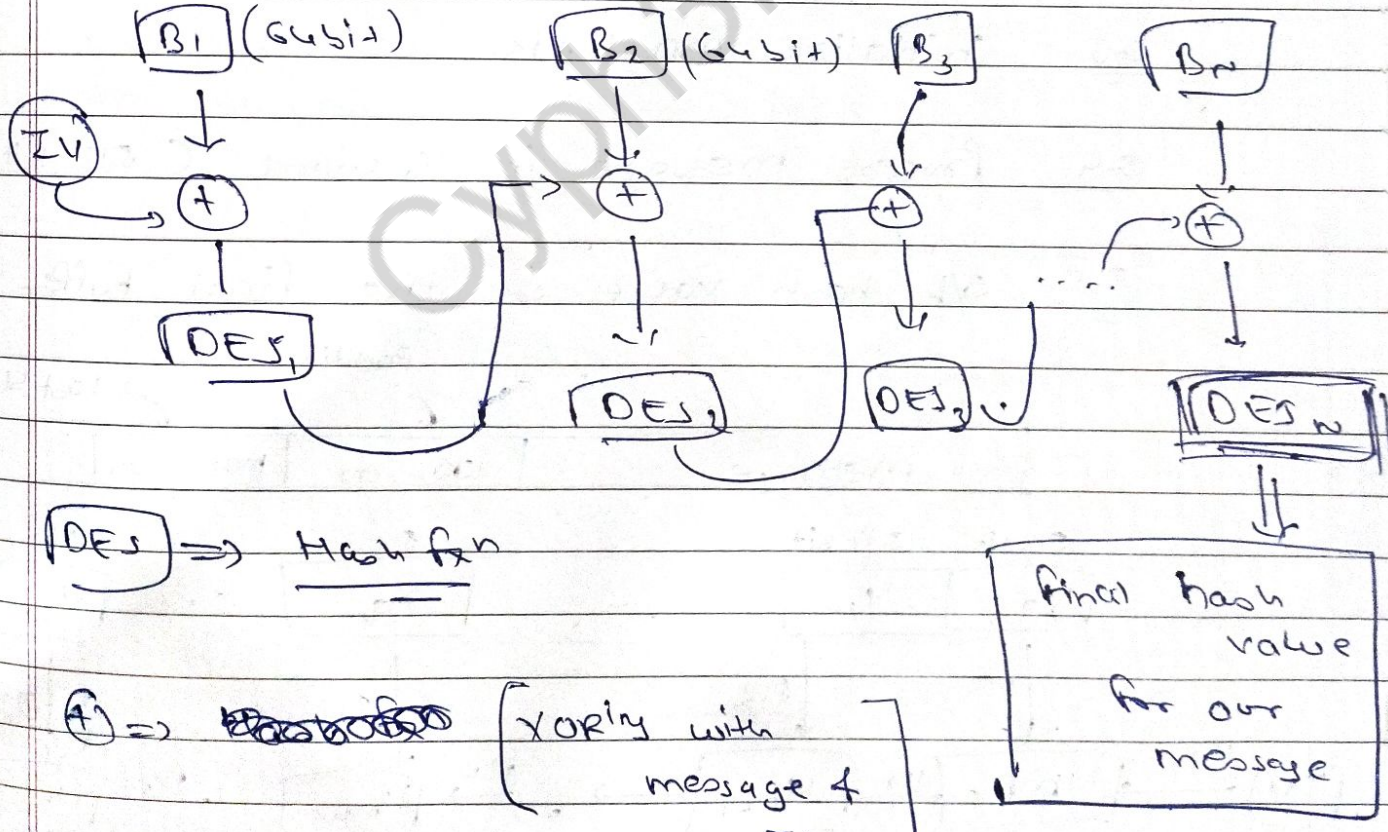
↳ just as before discussed

(5) Weak collision resistant

(6) Strong collision resistant : $H(x) = H(y)$

↳ collision free property.

✱ CBC in hash :



DES ⇒ Hash fxn

(+) ⇒ [redacted] [XORing with message & IV]

Note:

if we give 64 bit input ⇒ output will be 64 bit only.

↓ variable size

↓ fixed size

PAGE No.
DATE / /

✱ MD5 (Message Digest version 5)

⇓

generate only (128 bit o/p.)

↳ in the last block we have 448 bit block

~~128~~ bit block size of MD5 ⇒ 512 bit

↳ padding can be done in manner

100000....00

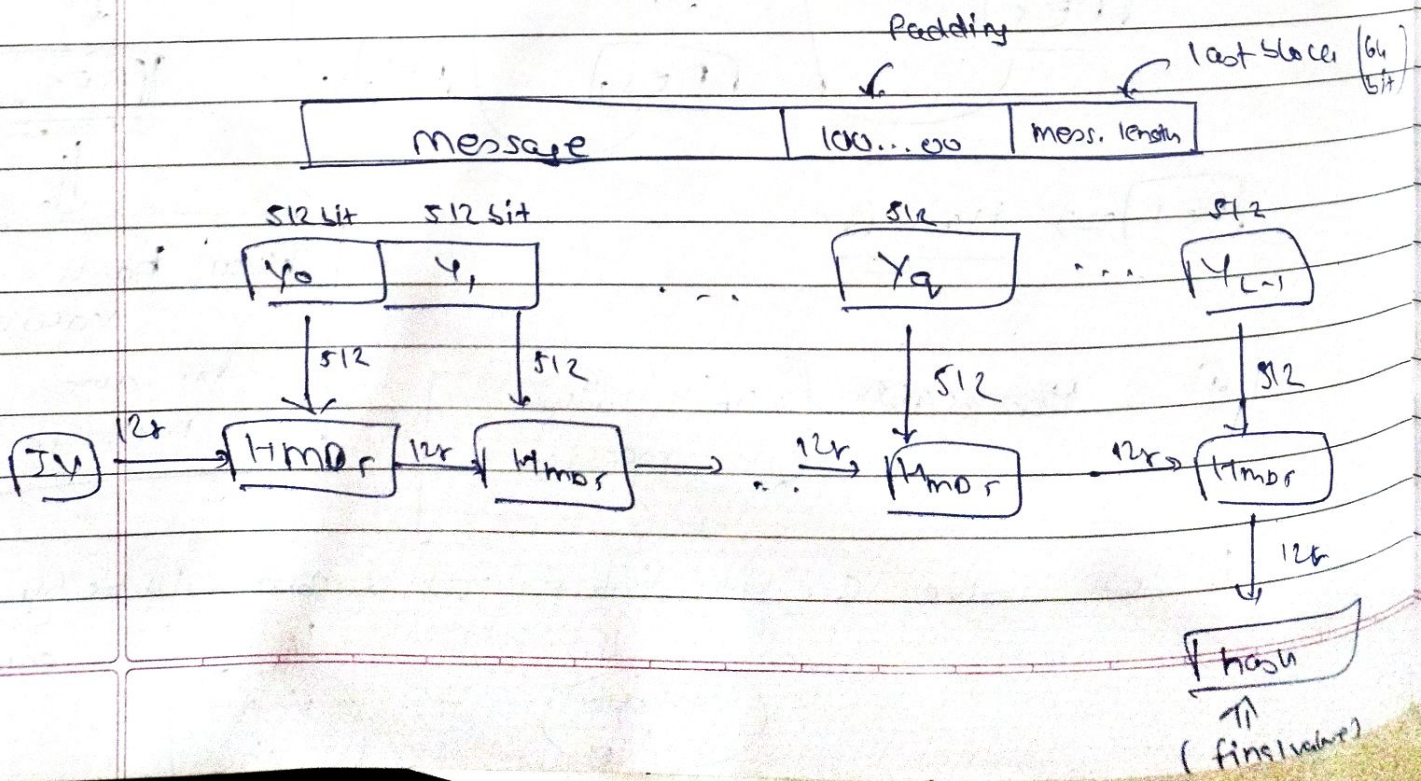
S-1 Append Padding bit

S-2 Append length

S-3 initialize MD buffer.

S-4 Process message in 16 word (512 bit) block

S-5 o/p hash value is the final buffer value.



each hexadecimal => 4 bit
(0...9, a...f)

PAGE No.
DATE / /

effective bit => 448 bit.
IV => (128 bit)
message length => l mod 2^{64} => (64 bit)

in each MD5 we have to perform 64 step.
4 rounds => 16 step each (total)

IV => 128 bit
4 word
1 byte = 8 bit
1 word => 32 bit

P: 01234567
Q: 89abcdef
R: fedcba98
S: 76543210

What happens in MD5 hash fxn?

m_i => 512 bit

512 bit → PQRS1 = f_A(PQRS0, m_i, T[1...16])
P, Q, R, S
128 bit → PQRS2 = f_B(PQRS1, m_i, T[17...32])
128 bit → PQRS3 = f_C(PQRS2, m_i, T[33...48])
128 bit → PQRS4 = f_D(PQRS3, m_i, T[49...64])
+ + + +
128 bit hash value
IV = 128 bit

PAGE No.
DATE / /

SHA1 : (Secure Hash Algorithm).

it is all the same as MD-5 but the difference here is:

→ MD5 has i/p value => 512 bit
SHA1 " " " => 512 bit

→ MD5 rounds => 4 round => 16 step each & 64 step total
SHA1 rounds => 4 round => 20 step each & 80 step total

→ MD5 word => A B C D
32bit 32bit 32bit 32bit => (128 bit)

SHA1 words => P Q R S T
32 32 32 32 32 => (160 bit)

∴ SHA1 IV => 160 bit
SHA1 O/P => 160 bit

↓

SHA1 has key with which each round is encrypted

↓

more secure

↓

slower than MD5

MD5 IV => 128 bit
MD5 O/P => 128 bit

↓

MD5 has no key

↓

less secure than SHA1

↓

faster than SHA1

PAGE No.
DATE / /

SHA versions:

5 types:

	SHA1	SHA-224	SHA-256	SHA 384	SHA 512
Message digest size.	160	224	256	384	512
Message size →	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block size →	512	512	512	1024	1024
Word size →	32	32	32	64	64
No. of steps. →	80	64	64	80	80

Words for SHA1:

P => 67 45 23 01

Q => ef cd ab 89

R => 98 ba dc fe

S => 10 32 54 76

T => c3 d2 e1 f0

for MD5

P => 01 23 45 67

reverse P => 10 32 54 76

reverse (P) for (P) of SHA1.

Q = 89 ab cd ef

reverse Q = 98 ba dc fe

reverse => ef cd ab 89 for SHA1

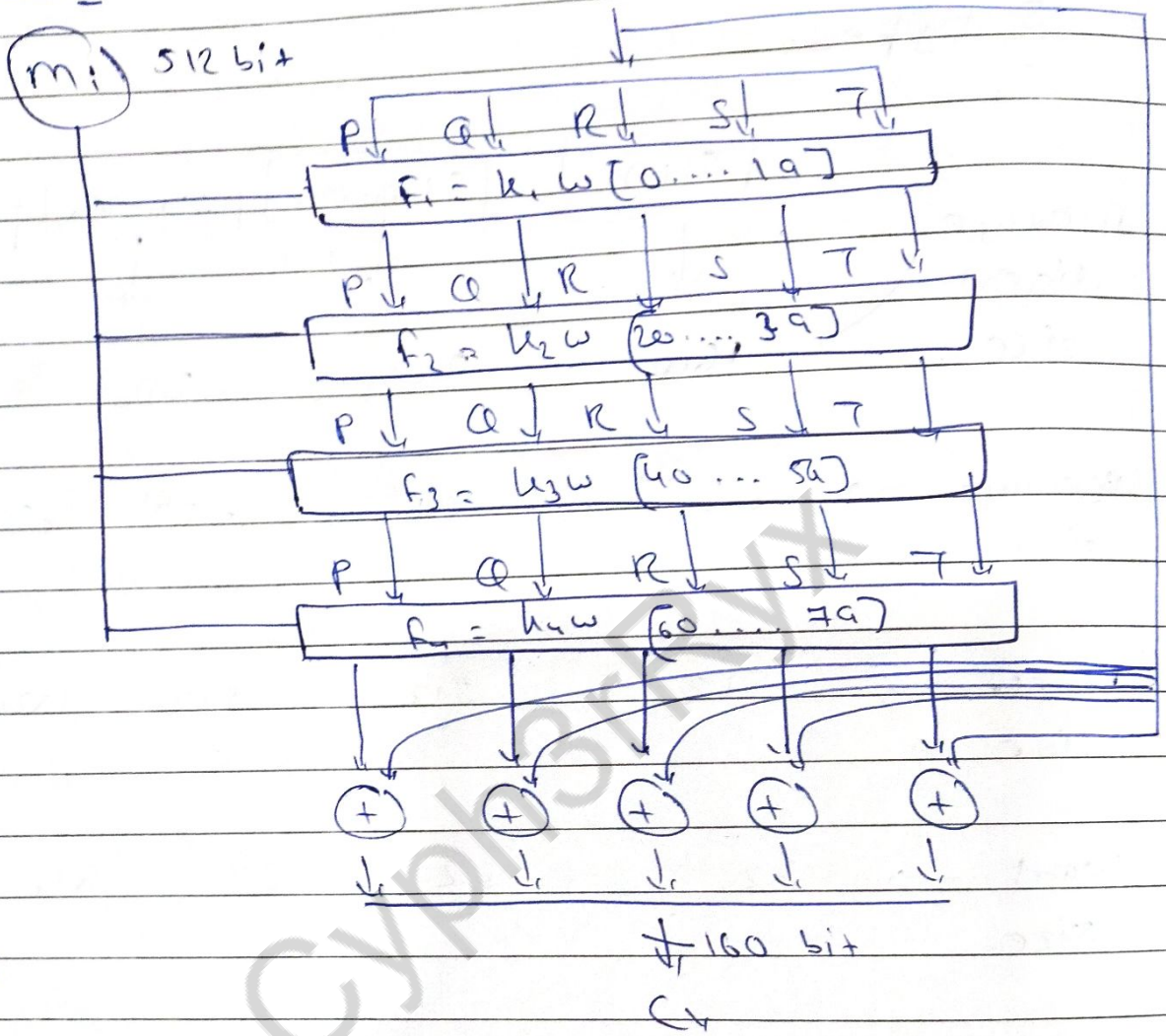
R => reverse of Q

S => reverse of P

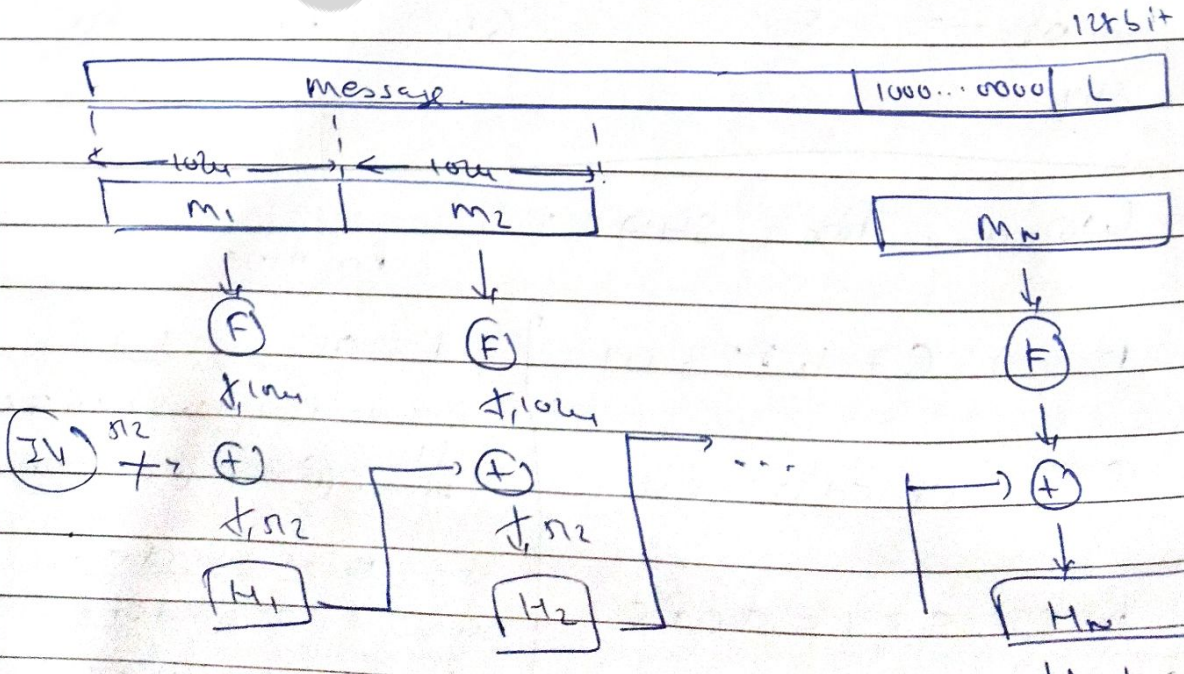
T => c d ef 4 3 2 1 0

complete item

Structure:



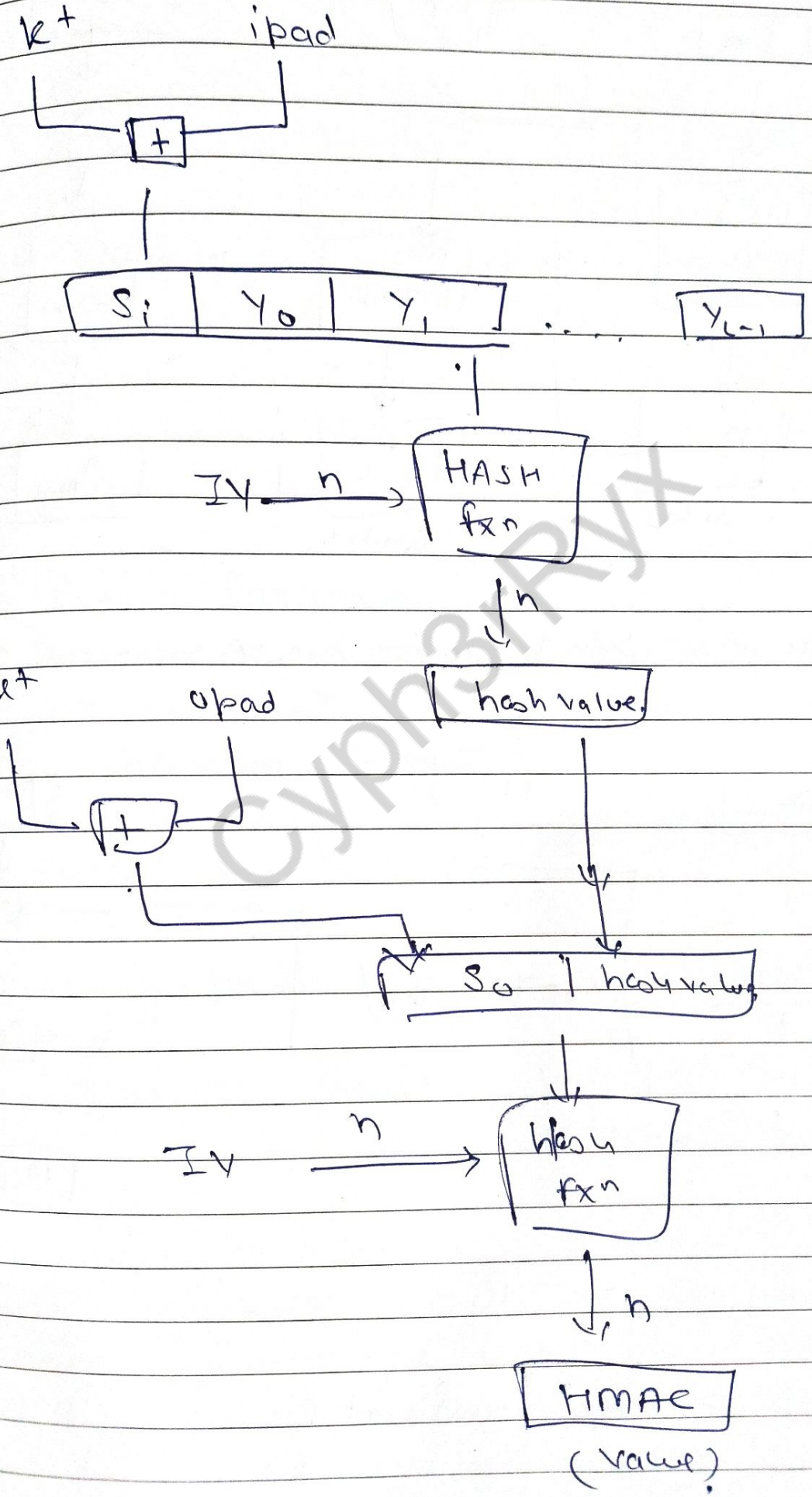
SHA-512



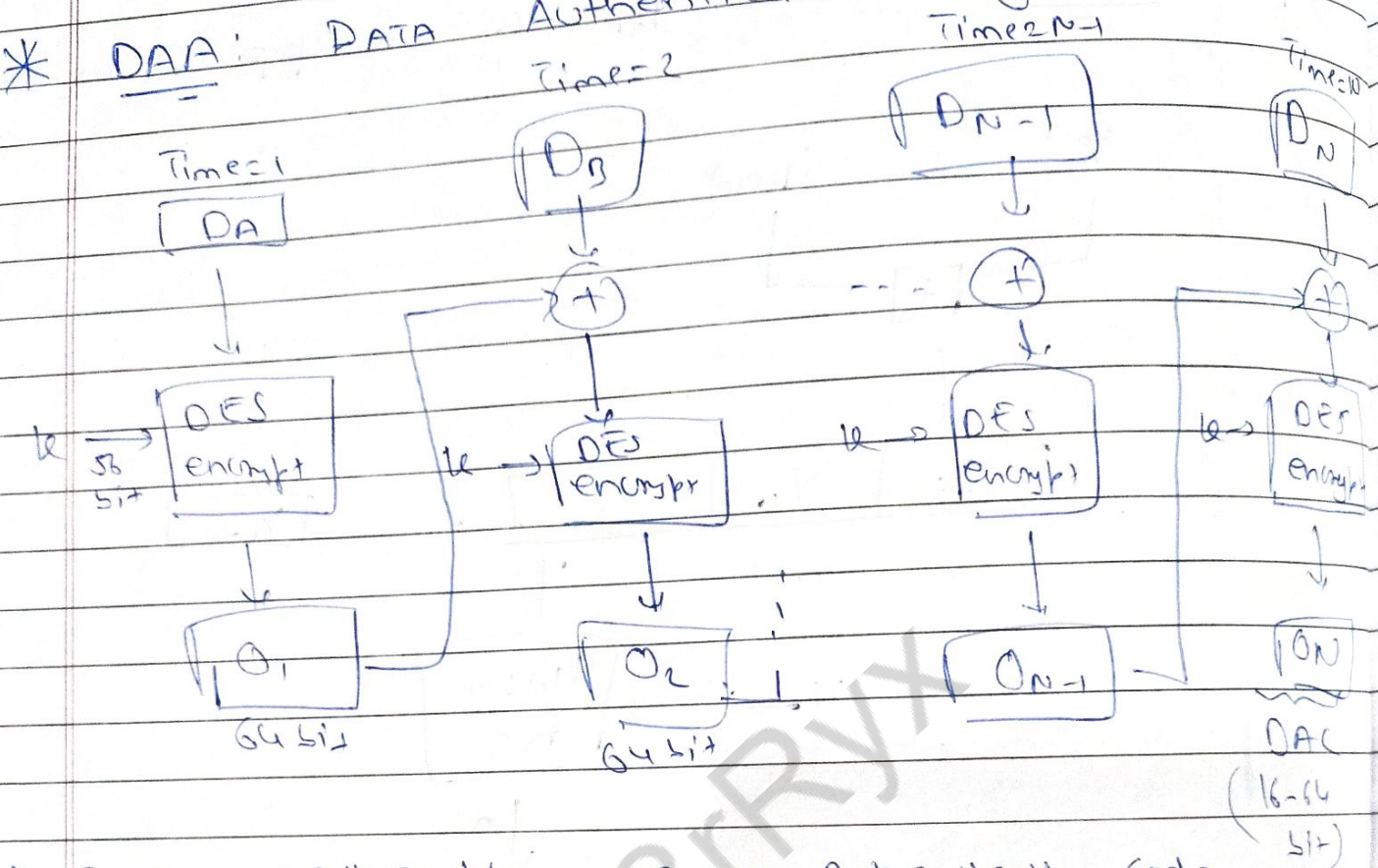
- Words: a, b, c, d, e, f, g, h.
- Rounds: 80

HMAC:

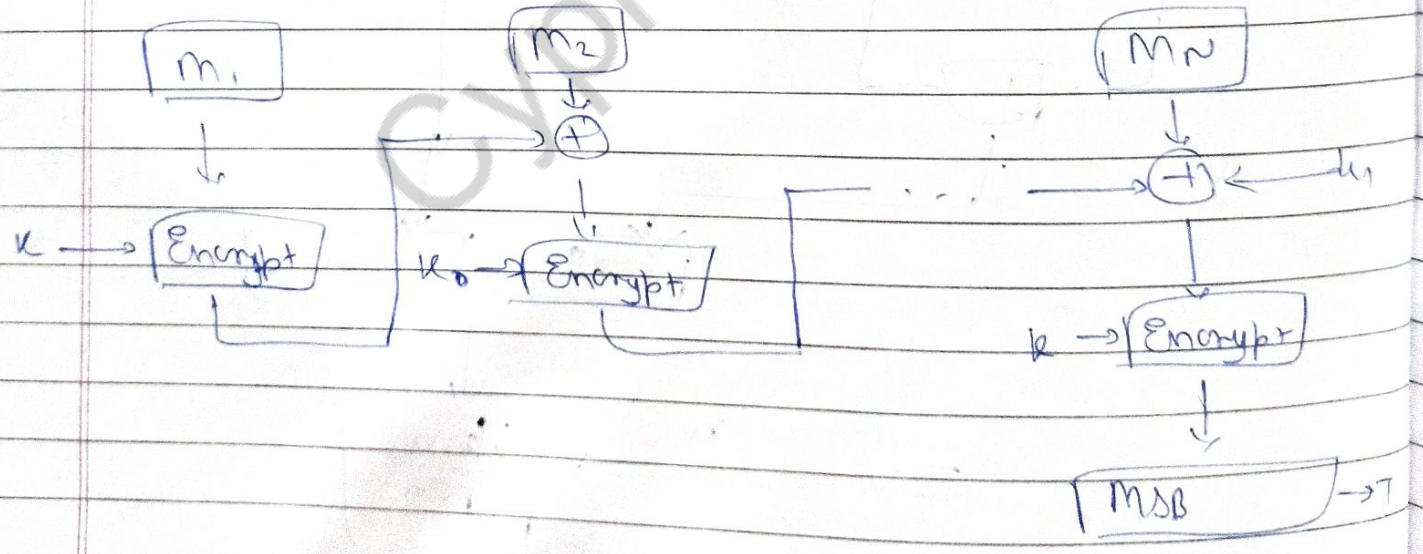
$ipad = 36$ & $opad = 56$ constant.



* DAA : Data Authentication Algorithm.



* CMAC : Cipher block message Authentication Code



Padding = 1000...000

LSB = Least Significant Bit

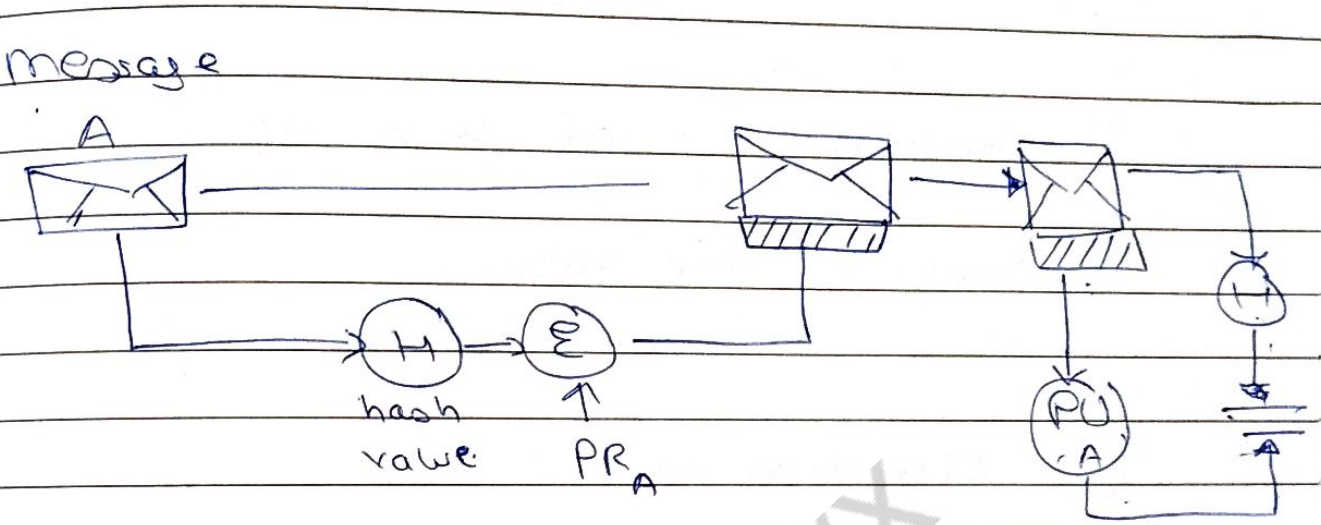
MSB = Most Significant Bit

1 0 1 1 0 0

MSB LSB

PAGE No.
DATE / /

* Digital Signature:



* Elgamal:

1. Public Parameter
2. Private key of sender (PR_A) ~~& Public key~~
3. Hash fxn
4. Random integer.

Algorithm steps:

① Using public parameter (q & a)

② Generate sender private key (X_A)
public key (Y_A)

③ The hash $m = H(m)$, $0 \leq m \leq q-1$

④ Random integer k with
$1 \leq k \leq (q-1)$
$\gcd(k, q-1) = 1$

(5) Compute temporary key:

$$S_1 = a^k \bmod q$$

(6) Compute k^{-1} of $k \bmod (q-1)$

(7) Compute the value:

$$S_2 = k^{-1}(m - X_A S_1) \bmod (q-1)$$

(8) Signature is (S_1, S_2)

Verification:

(1) $V_1 = a^m \bmod q$

(2) $V_2 = Y_A^{S_1} S_1^{S_2} \bmod q$

(3) Signature is valid if $V_1 = V_2$

Note: if they ask/give you a numerical then q, a, X_A, m & k will be provided by them.

PAGE No.
DATE / /

* Schnorr

① Choose suitable primes p, q

② Choose 'a' such that $a^q = 1 \bmod p$

③ Here (a, p, q) are public parameters.

④ Each user (A) generates a key:

(i) Choose secret key : $0 < s_A < q$

(ii) Compute the public key :
$v_A = a^{-s_A} \bmod q$

⑤ Random 'r' with $0 < r < q$ and
compute $x = a^r \bmod p$

⑥ Concatenate M with x & hash result to
$e = H(m \| x)$

⑦ Compute : $y = (r + se) \bmod q$

⑧ Signature is (e, y)

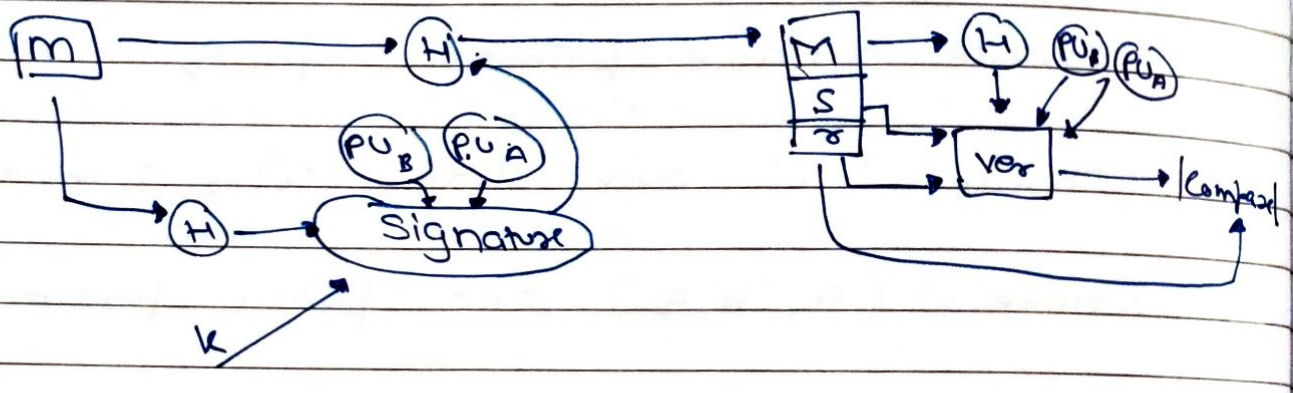
Verification:

① Compute : $x' = a^y v^e \bmod p$

② Verify : $e = H(m \| x')$

③ $x' = e$
$\therefore$ valid

DSS :



↳ $r = (g^k \bmod p) \bmod q$

↳ $s = k^{-1}(H(m) + xr) \bmod q$

STEPS:

1. have shared global public value (p, q, g)
   - choose 160 bit prime no. q
   - choose a large prime p with $2^{L-1} < p < 2^L$
     ↳ $L = 512$ to 1024 bits & is a multiple of 64
     ↳ such that q is a 160 bit prime divisor of $(p-1)$.
   - choose $g = h^{(p-1)/q}$
     ↳ $1 < h < p-1$ & $h^{(p-1)/q} \bmod p > 1$

User choose private & compute public key:

↳ choose random private key: $x < q$

↳ compute public key: $y = g^x \bmod p$

PAGE No.
DATE / /

Verification:

→ having recieved m & signature (r, s)

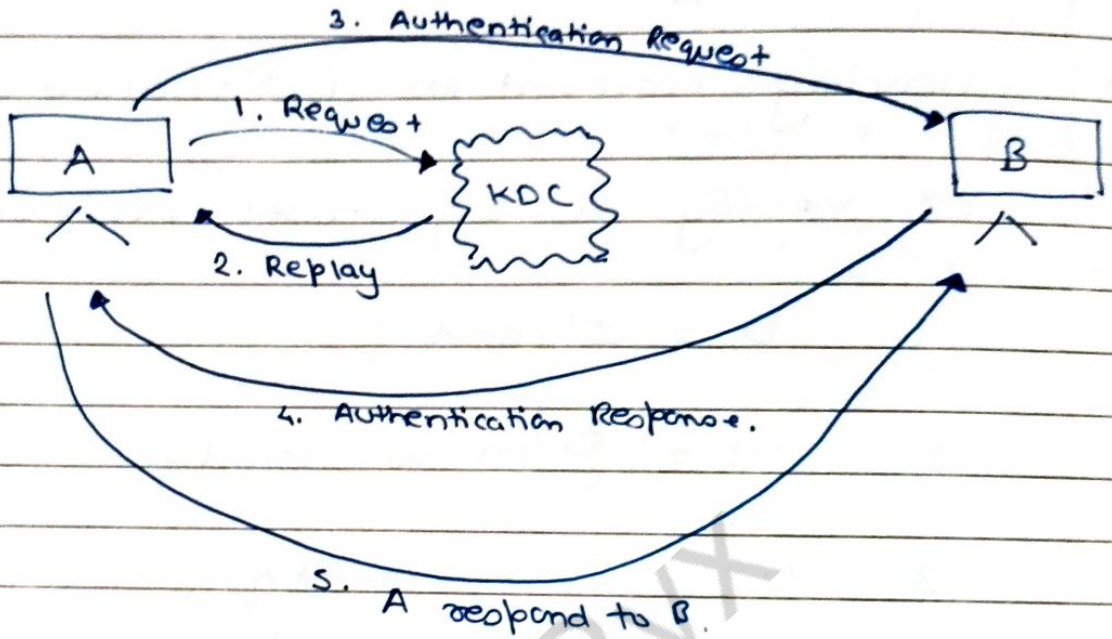
→ to verify a signature, reciever computes

1. $w = s^{-1} \bmod q$
2. $u1 = [H(m)\, w] \bmod q$
3. $u2 = (r w) \bmod q$
4. $v = [(g^{u1} y^{u2}) \bmod p] \bmod q$

→ if $v = r$ then signature is verified.

PAGE No.
DATE / /

* Needham - Schroeder Protocol :



Protocol overview:

1. $A \longrightarrow KDC : ID_A \,||\, ID_B \,||\, N_1$

Here ID_A = Sender's public key
ID_B = Reciever's public key
N = Nonse value

2. $KDC \longrightarrow A :$

$$E(k_A, [k_s \,||\, ID_B \,||\, N_1 \,||\, E(k_b, [k_s \,||\, ID_A])])$$

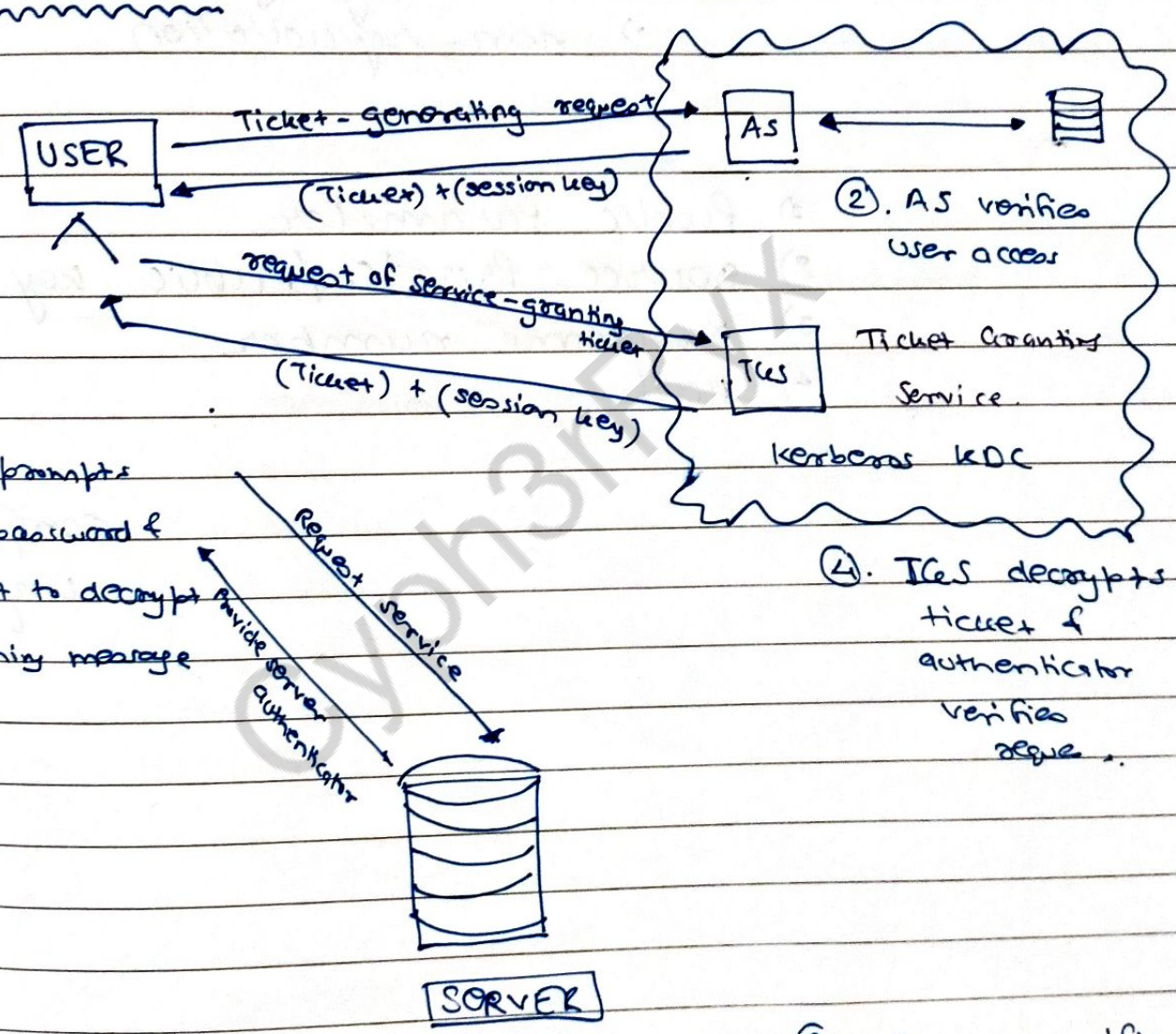
Here k_A = master key of sender
k_s = session key
k_b = master key of reciever
E = Encrypt

3. $A \rightarrow B : E(K_b, [K_s \| ID_A])$

4. $B \rightarrow A : E(K_s, [N_2])$

5. $A \rightarrow B : E(K_s, [f(N_2)])$

* KERBEROS:



⑤ User workstation sends ticket & authenticator to server

⑥ Server verifies & grant the access.

* OSI model:

Application layer
Presentation layer
Session layer
} HTTP, SMTP, FTP, DNS, SNMP, NFS

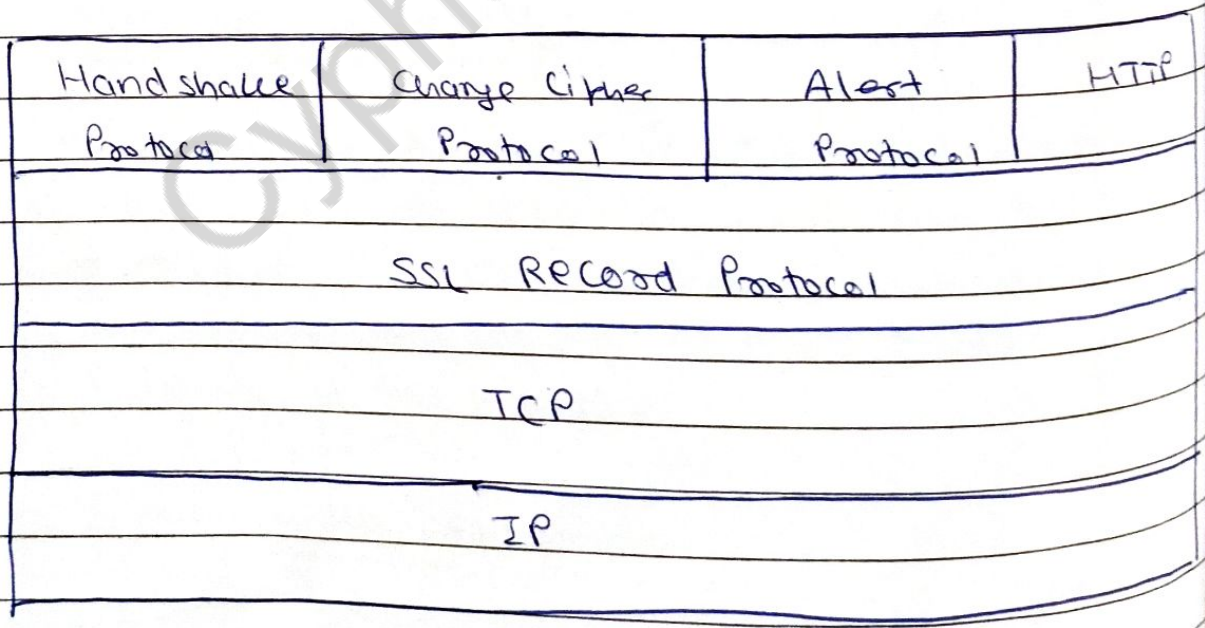
Transport layer ⟶ TCP, UDP

Network layer ⟶ IP

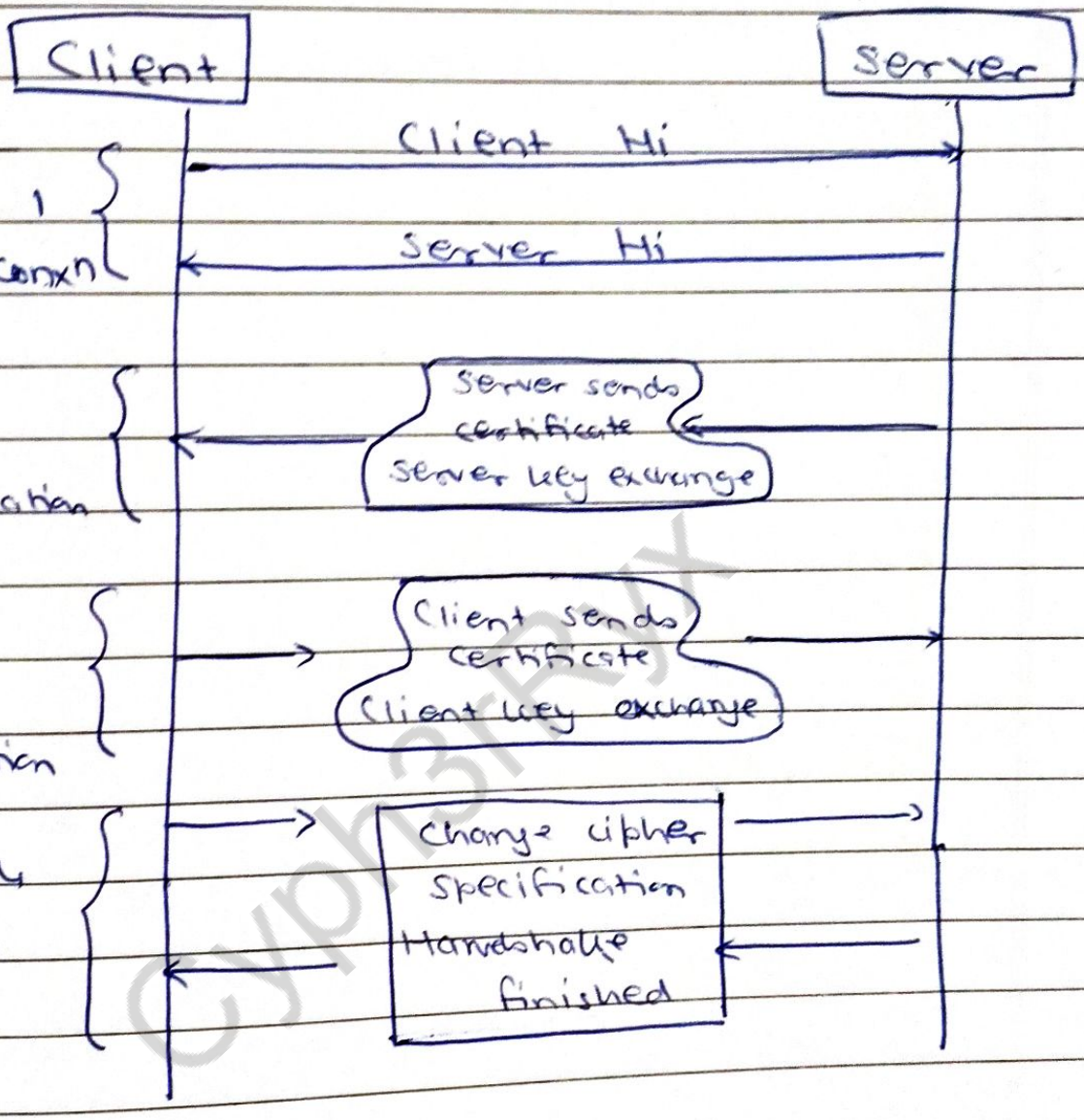
Data link layer ⟶ Device Driver

Physical layer ⟶ Network Adapter

* SSL: Secure Socket Layer.



Handshake Protocol : 4 Phase.



Annexure No :

Network Security

Most Hard Part

- ① Schnorr
- ② Elgamal
- ③ DSS vs RSA & DSA Approach
- ④ TLS
- ⑤ SSL
- ⑥ IPsec
- ⑦ Kerberos
- ⑧ Needham - Schröder.
  - (i) Symmetric
  - (ii) Asymmetric

So let start.

Elgamal :

→ DS → validate

→ Safe & secure

→ Assymetric key exchange.

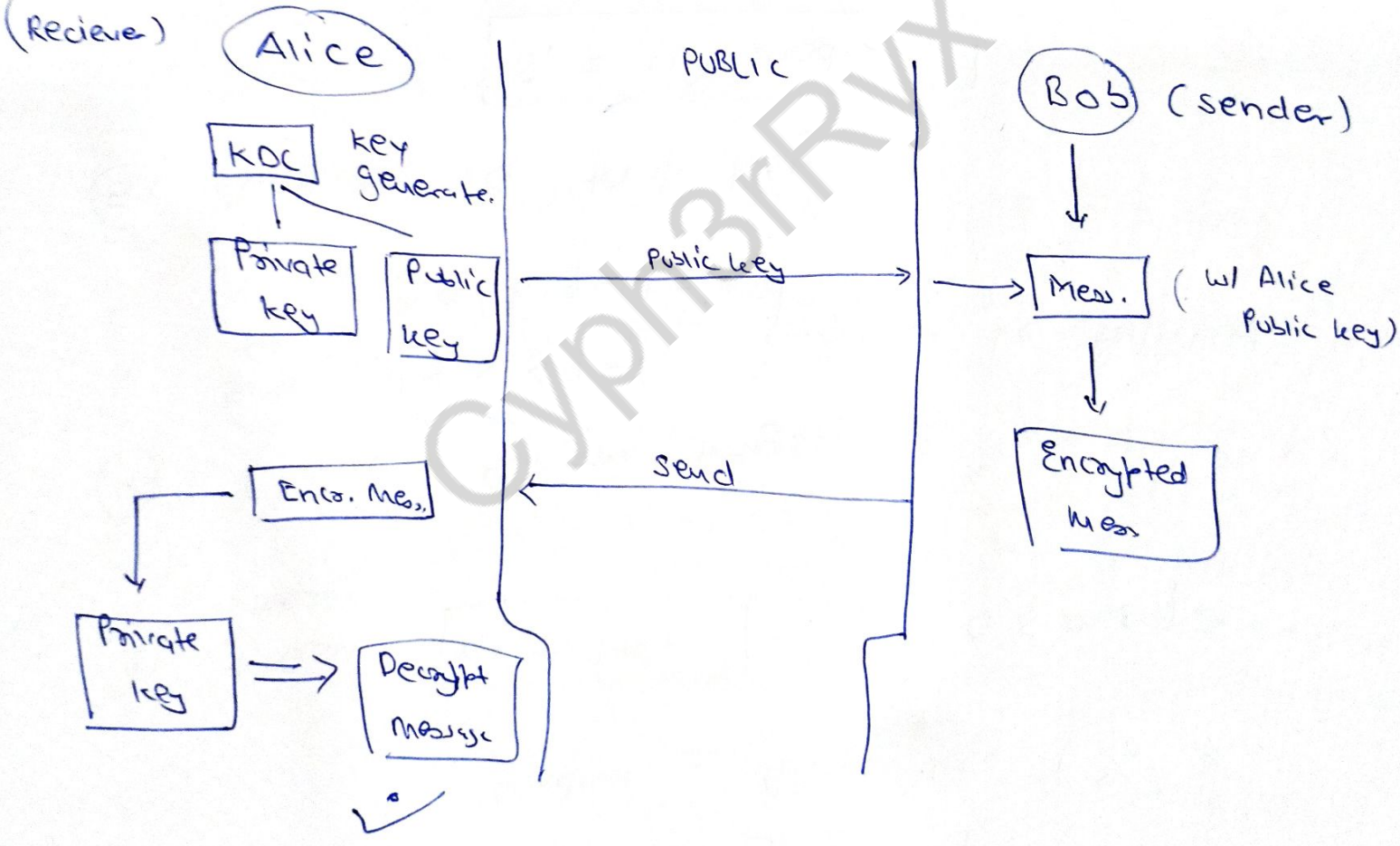
Developed by

Taher Elgamal → father of SSL

↓

DSS → major contrib.

Elgamal Cryptosystem



Signature sent by Bob is same as Alice

Algorithm :

(1) Select q & a

(2) Generate senders Private key (X_A)

Senders Public key (Y_A)

(3) Hash message $m = H(m)$; $0 \leq m \leq q-1$

(4) Choose Random integer "k"

such as, $1 \leq k \leq q-1$

$\gcd(k, q-1) = 1$ } $\Rightarrow$ must be fulfilled

(5) Compute a temp. key

$$S_1 = a^k \bmod q$$

(6) Compute k^{-1} of $k \bmod (q-1)$

(7) Compute the value :

$$S_2 = k^{-1}(m - X_A S_1) \bmod (q-1)$$

maxe one :

Signature $= (S_1 \& S_2)$

Verification :

(1) $V_1 = a^m \bmod q$

(2) $V_2 = Y_A^{S_1} S_1^{S_2} \bmod q$

if $V_1 = V_2 \implies$ Signature validate.

How to Remember this ?

(1) q & a => Q & A (Rythm)

(2) Private & Public => KDC ster. & $m = H(m)$

(3) k $0 \le k \le q-1$ => Random value
$\gcd(k, q) = 1$

(4) $S_1 = a^k \bmod q$ => (k) aa gaya toh S_1 find kara
usme a (power) mod (q)

(5) $k^{-1} = k \bmod (q-1)$ => (k) aa gaya toh k^{-1} find kara
usme k ka mod (q-1)

(6) $S_2 = k^{-1}(m - X_A S_1) \bmod (q-1)$
find karo

=> (k^{-1}) aa gaya toh S_2 find karo

(S_1 & S_2)

k^{-1} (maximum) mod (q-1)

(7) $V_1 = a^m \bmod q$ (am mod que)

(8) $V_2 = Y_A^{S_1} S_1^{S_2} \bmod q$ (Yasavan Savan Satu mod que)

Rewriting Algo:

(1) q & α

(2) X_A = Priv.

Y_A = Pub.

(3) $m = H(m)$

$0 \leq H(m) \leq (q-1)$

(4) $k \Rightarrow 0 \leq k \leq (q-1)$, $\gcd(k, q-1) = 1$ $\Big\}$ $S_1 = \alpha^k \bmod q$

(5) $k^{-1} = k \bmod (q-1)$

(6) $S_2 = k^{-1}(m - X_A S_1) \bmod q-1$

(7) S_1 & S_2

(8) $V_1 = \alpha^m \bmod q$

(9) $V_2 = Y_A^{S_1} S_1^{S_2} \bmod q$

(10) $V_1 = V_2 \Longrightarrow$ signature verified

Schnorr

Algorithm:

(1) p, q => prime no.

(2) Choose 'a' => $a^q = 1 \bmod p$

(3) Public Parameter (a, p, q)

(4) User - A generates a key

(i) secret key => $0 < S_A < q$

(ii) Public key => $y_A = a^{-S_A} \bmod q$

(5) Random 'r' => $0 < r < q$

(6) $x = a^r \bmod p$

(7) Concatenate "m" w/ x & hash => $e = H(m \| x)$

(8) $S_1 = (r + Se) \bmod q$

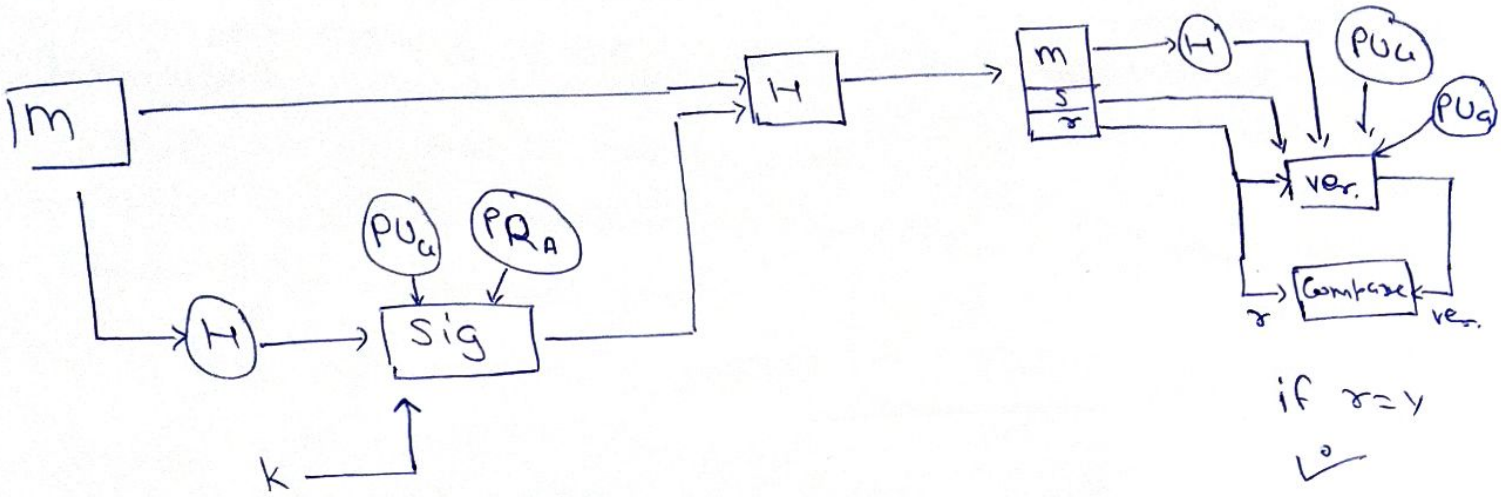
(9) Signature = (e, S_1)

Verification

(1) $x' = a^{S_1} y_A^e \bmod p$

(2) $e = H(m \| x')$

DSS :



Hash code provided as i/p with random "k"

Signature depend on Private key of A, Public key (global) } ⇒ Hashed Result (S & r) w/ message

message w/ s & r

↓ hashed

w/ PU_G & PU_A also s & r } ⇒ Version

↓ in-take

compared w/ (r) & (version)

if r = version

⇒ sig. verified.

Algorithm : (I) key Generation

Shared values => p, q, g

- For q => choose 160 bit prime no.
- for p => choose large prime no.

  ; $2^{L-1} < p < 2^L$ [L = 512 - 1024 bits]

- for g => $g = h^{(p-1)/q}$; $1 < h < (p-1)$

  & $h^{(p-1)/q} \bmod p > 1$

↳ Choose Private key & Compute public key.

Random private : $x < q$

Compute public : $y = g^x \bmod p$

(II) Signature Creation :

(i) Generate (k) ; $k < q$

↳ random & one time use

(ii) Compute [signature] : $r = (g^k \bmod p) \bmod q$

$s = [k^{-1}(H(m) + xr)] \bmod q$

Send signature (s, r) along w/ message (m)

(III) Signature Verification :

recieve s, r w/ (m) Verification is done via.

(i) $w = s^{-1} \bmod q$

(ii) $u_1 = [H(m)\, w] \bmod q$

(iii) $u_2 = (r \cdot w) \bmod q$

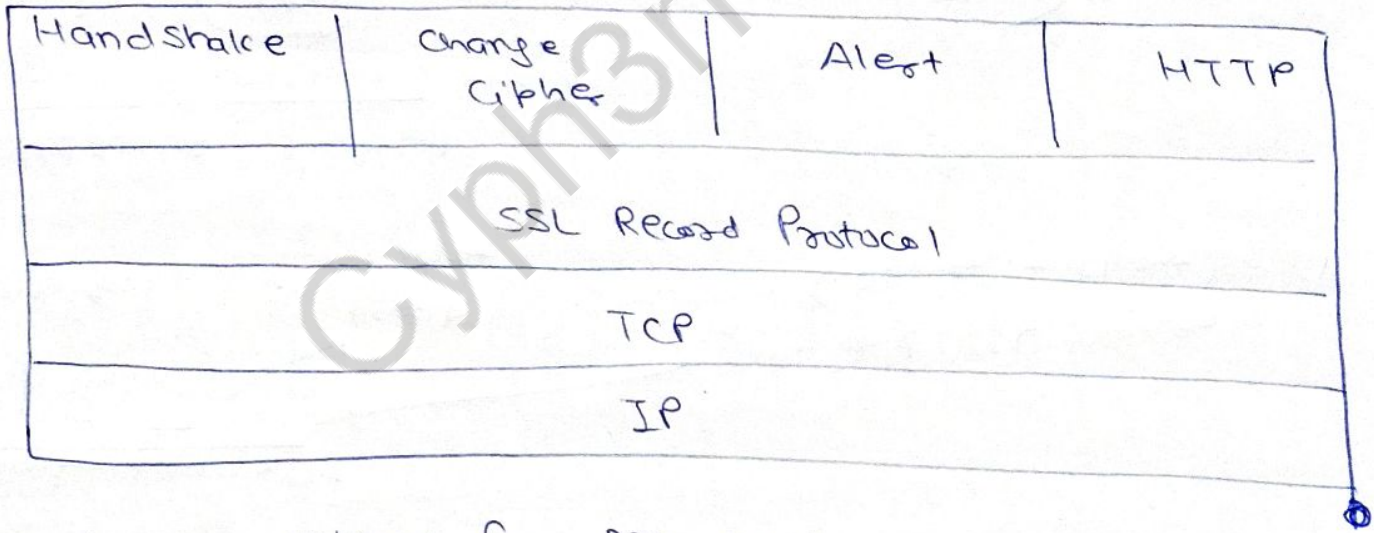
(iv) $v = [(g^{u_1} y^{u_2}) \bmod p] \bmod q$.

if $v = r$ [Signature verified]

SSL = Secure Socket Layer.

- netscape
- Transport Layer Security
- secure data transfer b/w [Web server] & [Client]

Protocol Stack:



First layer: support for SSL session connxn management & connxn establishment

Second layer: [Message authetication], [Confidentiality] & [Integrity].

Third layer: Transport

& Layer

fourth layer: Internet.

SSL Architecture: (1) SSL session:

association b/w client & server.

created via Handshake Protocol.

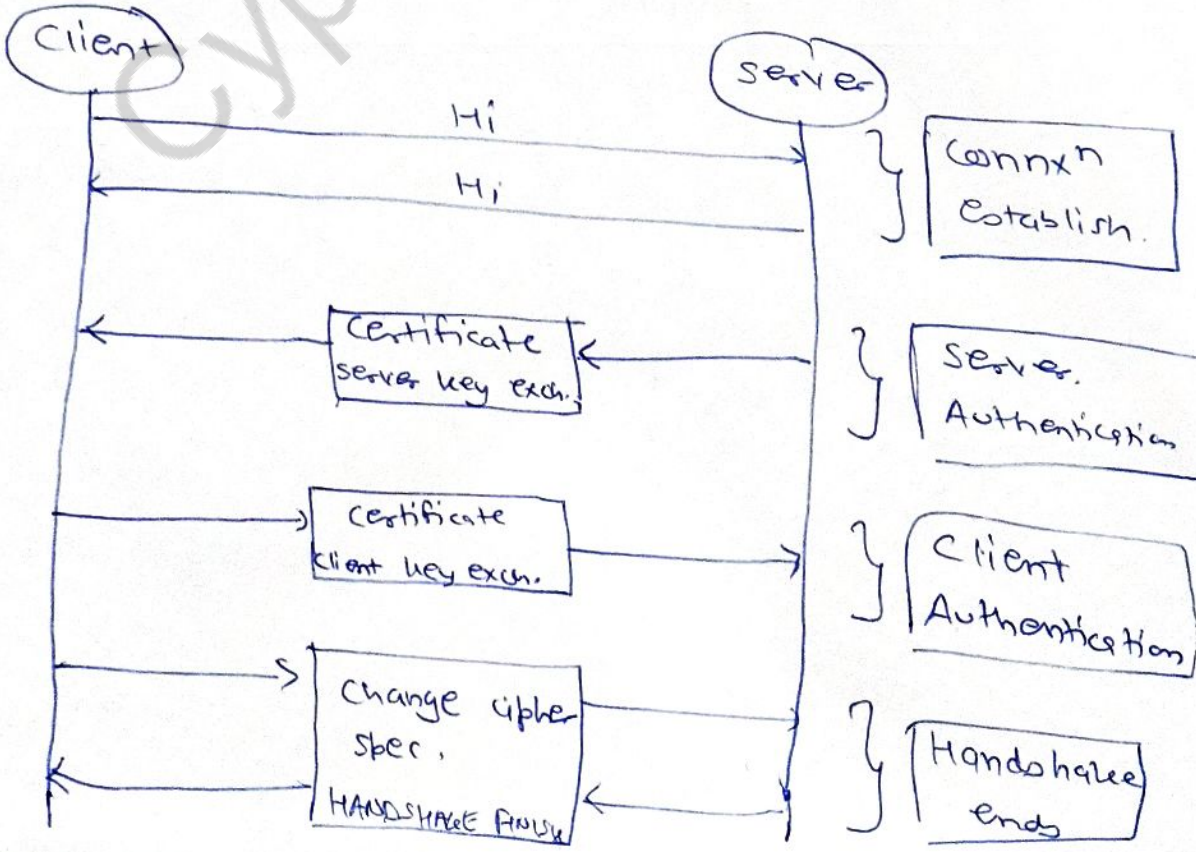
shared by multiple SSL connxn.

defined by cryptographic parameter

(2) SSL connxn:

- transient
- peer to peer
- only one SSL session (server - client)

(3) Handshake Protocol



(4) SSL change cipher Protocol :

- Uses Record Protocol of SSL
- a single message
- pending state → To → current state
- Updating cipher suite in use.

(5) SSL Alert Protocol :

- warning — alert — notify

about., unexpected message
error
handshake failure
bad record mac
illegal parameter.

via

Alert codes : eg. 0 → close_notify (no message from sender)
10 → unexpected_message.
40 → handshake_failure.
20 → bad_record_mac
42 → certificate_expired
no_certificate
bad_certificate.

⑥ HTTP :

↳ Port (80) | Complete the connxn b/w client & server

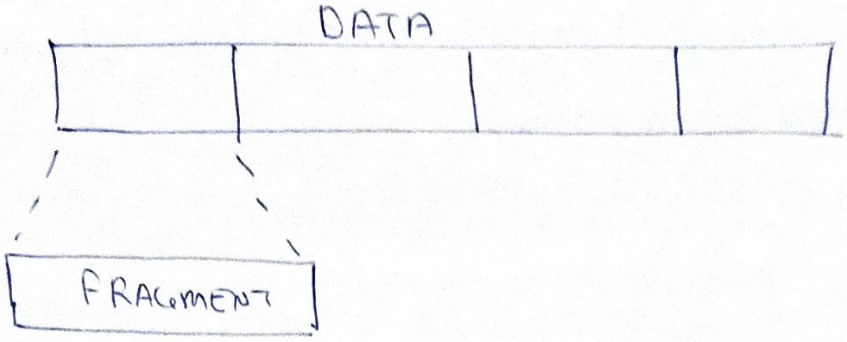
header.

⑦ SSL Record Protocol .

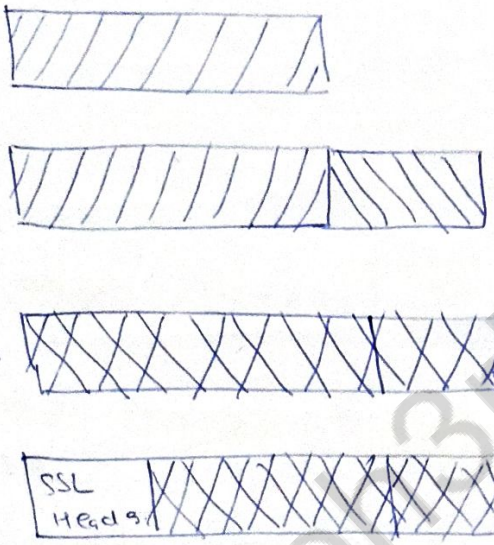
Confidentiality → message compressed before encryption
→ use symmetric encryption
→ message known to only certified vendors & recievers

Integrity → no one can change the data
→ use of MAC w/ shared secret key
→ similar to HMAC w/ diff. padding.

SSL Record Protocol diagram:



- Compress
- Compress + MAC
- Encryption
- SSL Header Appended



Payloads:

1. Change Cipher — 1 — 1 byte.
2. Alert — 1byte (Level) 1byte (Alert)
3. Handshake — 1 byte (TYPE) 3byte (length) ≥0byte (content)
4. HTTP : — ≥1 byte (Content)

Version:

SSL 1 → High insecurity
SSL 2 → 1995
SSL 3 → 1996
TLS 1.0 → 1999
TLS 1.1 → 2006
TLS 1.2 → 2008
TLS 1.3 → 2018

Enrollment No :

Page No :

Transport Layer Security (TLS):

- Derived from **SSL**
- ensures **no third party** tampers w/ message.

Benefits:

1. TLS / SSL Encryption
2. Work efficiently on most OS / web server.
3. Algorithm authentication
4. Easy deployment
5. Easy use.

Working of TLS:

(1) Client sends no. of specs.

(2) Server checks version supported by them both

(3) Server picks cipher suite & compression method.

(4) Server provides certificate

(5) Client verify certificate

(6) "key exchange occurs."

(7) Now server - client can communicate

(8) To close connxn : • Finish the process
• Termination known at both sides.

NOTE: Connxn can't be compromised in betn.

SSL	TLS
(1) Secure Socket Layer	(1) Transport Layer security
(2) Complex	(2) simple
(3) MAC is used	(3) Hashed MAC is used
(4) Less secure than TLS	(4) High secure
(5) SSL → Version 3.0	(5) TLS → version 1.0
(6) Slower	(6) Faster
(7) ress reliable	(7) more reliable
(8) Depreciated	(8) widely used
(9) uses Port to setup Explicit connxn	(9) Protocol to setup implicit connxn

IP sec Security Protocol

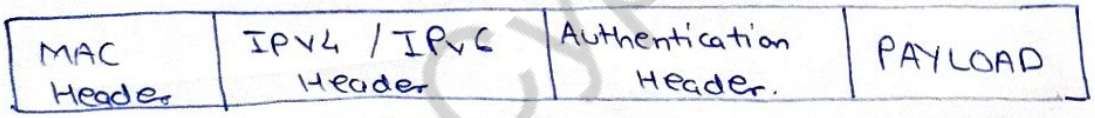
IETF (Internet Engineering Task force)

[Provide suite of Protocols] → for → [communication betn 2 points across IP network]

⇓

w/ confidentiality
integrity
authentication

Components :



1. Encapsulating Security Payload (ESP)
2. Authentication Header (AH)
3. Internet key Exchange (IKE)

(i) ESP :

- Data integrity
- Data encryption
- Data authentication
- Anti replay

→ Protocols provided for Authentication for payload.

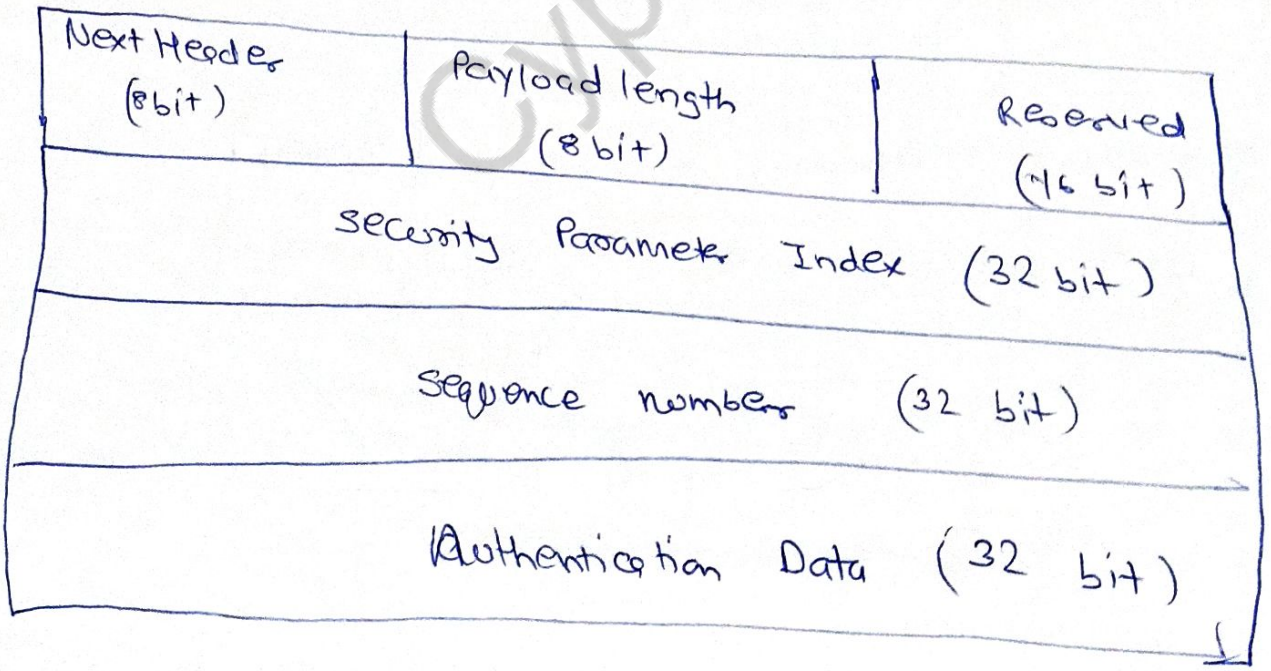
(ii) Authentication Header (AH) :

- Connectionless integrity
- Data origin authentication

2 main advantage :-

(i) message integrity : message not modified

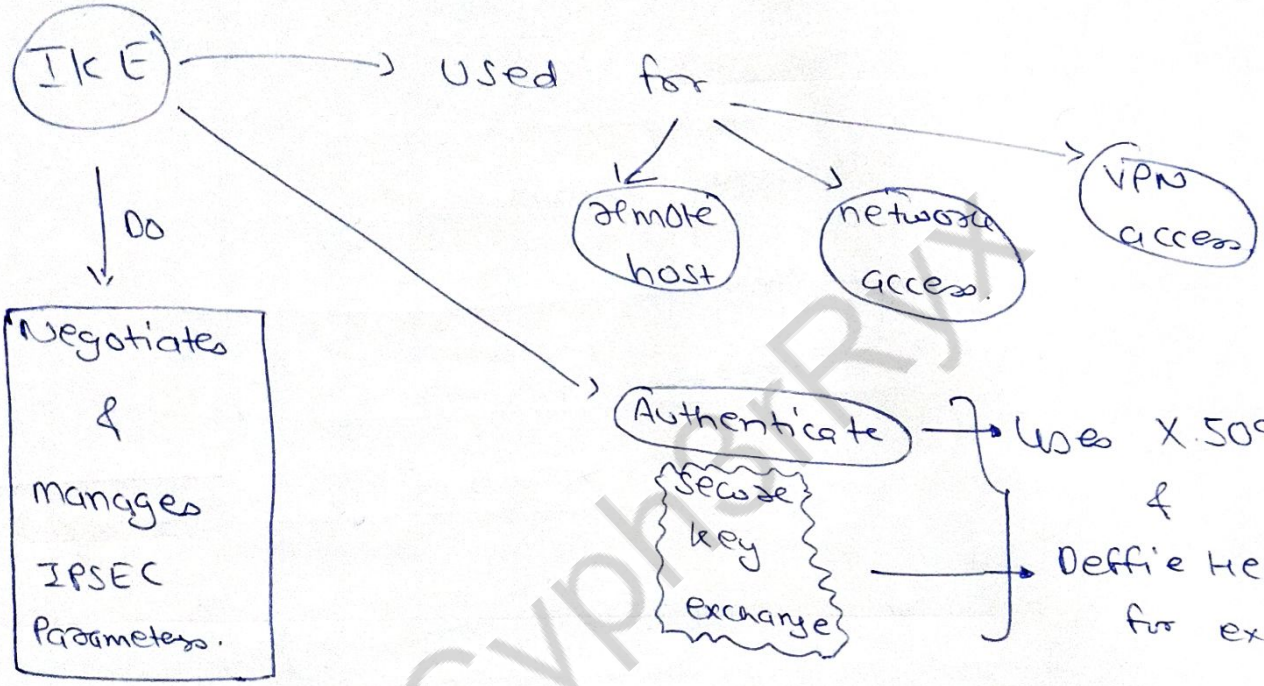
(ii) source authentication : source exactly from who you expect.



(iii) IKE → (Internet Key Exchange)

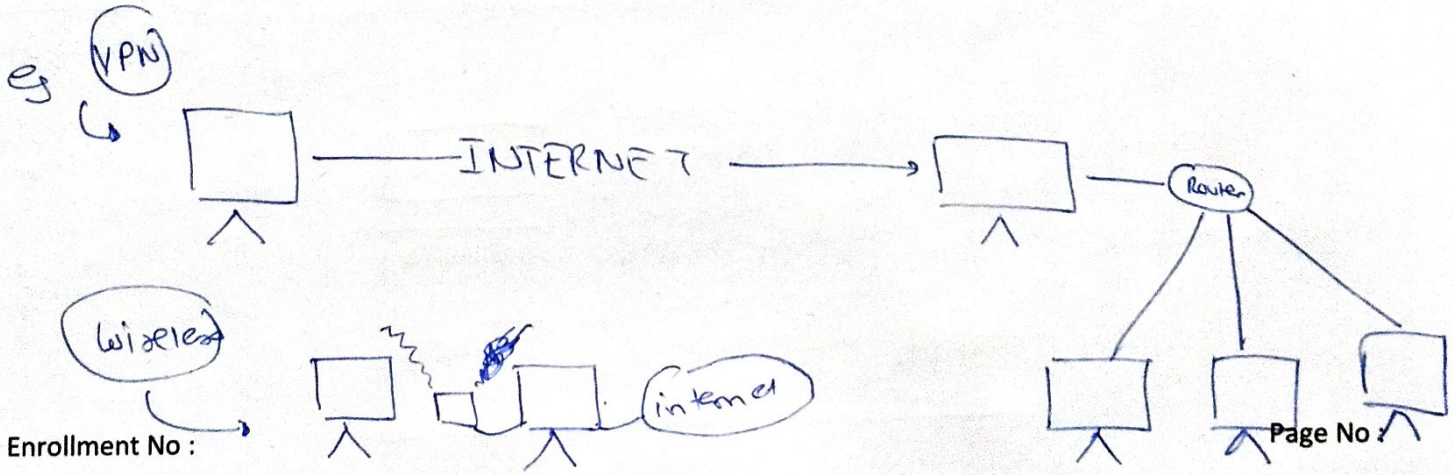
key management Protocol

→ secure, authenticated communication.



↳ Also used for setting up a security association (SA) using IPsec.

eg VPN



Use:

1. Encrypt data.
2. Provide security for routers.
3. Provide authentication without encryption.
4. Protect network data w/ VPN

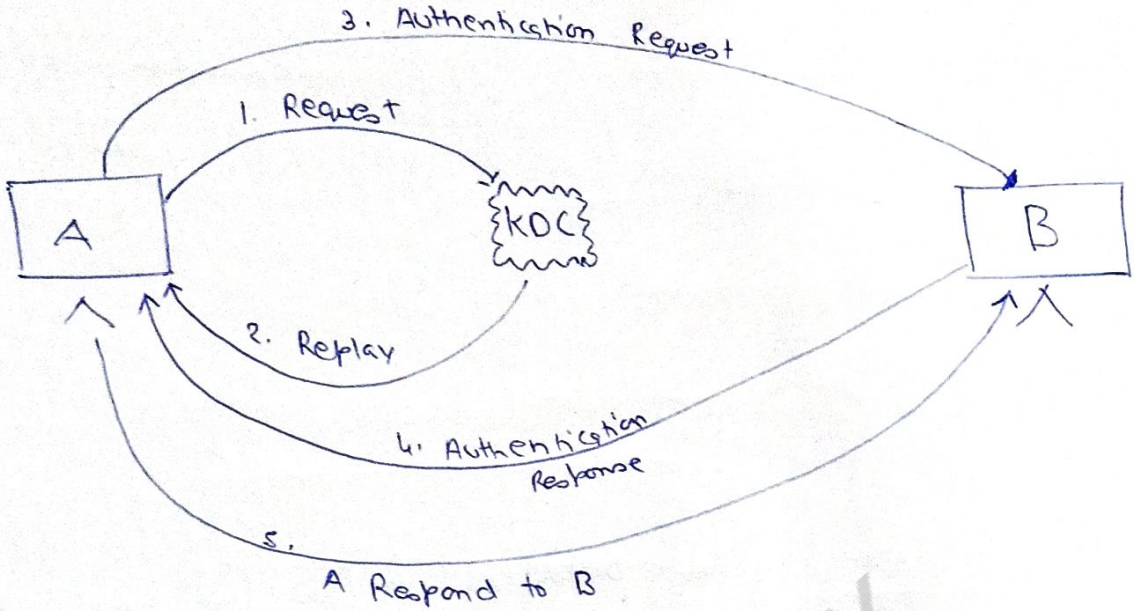
Advantage:

1. flexible
2. scalable
3. wide compatability
4. Strong security

Disadvantage:

1. Complex
2. Limited protection
3. key manggement
4. Performance impact.

Needham Schroeder



Also called " Remote User Authentication "

w/ symmetric encryption

Request (1). $A \longrightarrow KDC : ID_A \,||\, ID_B \,||\, N_1$

Here ID_A is senders Public key [Identity of A]

ID_B is Recievers ~~Private~~ Public key [Identity of B]

N_1 is Nonce value

Response (2) $KDC \longrightarrow A : E(K_A, [K_S \,||\, ID_B \,||\, N_1 \,||\, E(K_b, [K_S \,||\, ID_A])])$

↳ Here, (KDC) will generate one session key for comm betn A & B & send it to A

↳ The encryption is done via master key (K_A) of A.

Auth Request (3) $A \longrightarrow B : E(k_b, [k_b \| ID_A])$

Now (A) will send the 2nd portion to (B).

Encrypted via $[k_b]$.

↓

[master key of B]

k_b = session key

k_A & k_B = master key of A & B

E = Encrypt

Auth response (4) $B \longrightarrow A : E[k_b, [N_2]]$

- Once Authentication is achieved
  - (B) will send one message to (A) indicating session key & Nonce value to validate the request.

A responds to B (5) $A \longrightarrow B : E[k_b, f(N_2)]$

- User (A) sends response to (B) using secret key / session key.

One way authentication in Needham Schroeder :

We can remove last 2 step from the protocol.
So only 3 step will remain

(Request) S-1 $A \longrightarrow KDC : ID_A \| ID_B \| N_1$

- User (A) communicate w/ (KDC) to get session key

(Response) S-2 $KDC \longrightarrow A : E(K_A, [K_s \| ID_B \| N_1]) \| E(K_B, [K_s \| ID_A])$

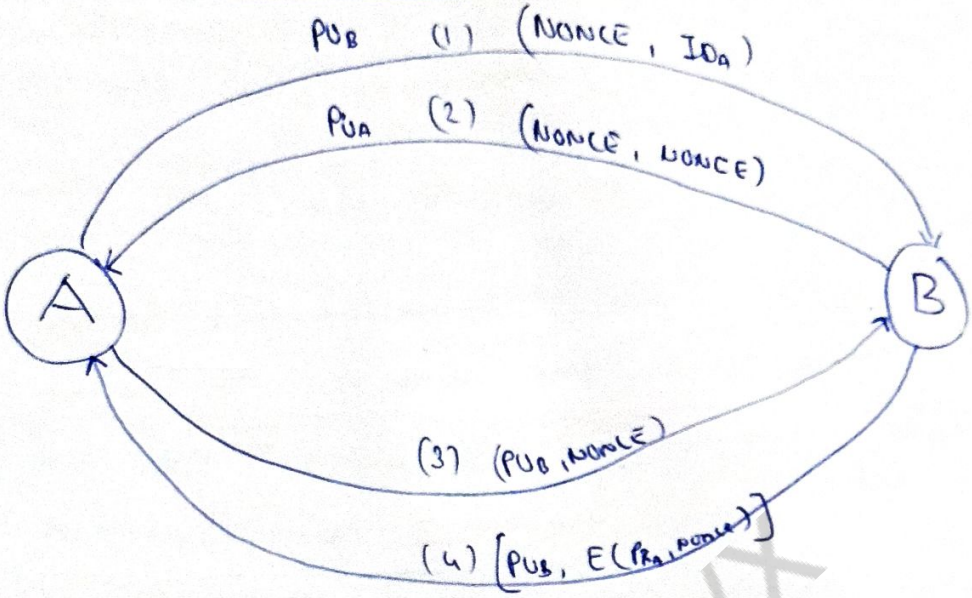
- (KDC) reply with session key

  2 message → first encrypted w/ (A)
  → second encrypted w/ (B)

(A to B) S-3 $A \longrightarrow B : E(K_B, [K_s \| ID_A]) \| E(K_s, m)$

- (Send) second portion of the above recieved message from KDC to B via A
- Also attach a message which is encrypted via session key so that the verification part can be removed.

Remote User Authentication w/ Assymetric Encryption.



if A→B
then PUB
if B→A
then PUA
except (4)

Mutual authentication:-

Request ① $E[PU_b, [N_1 || ID_A]]$

↳ A ⟶ B

↳ Encrypted via Public key - (B) (PU_B)

↳ on Decryption it get identity of A & nonce value

Response ② $E[PU_A, [N_1 || N_2]]$

↳ B → A

↳ Encrypted via Public key - (A)

↳ Decryption via Private key - (A)

Annexure No :

send (3) $E[PU_b, N_2]$

↳ A → B

↳ send message from A to B which is encrypted via public key of B (PU_B)

recieve (4) $E[PU_b, E(PR_A, K_s)]$

↳ B → A

↳ recieve message from B to A which is encrypted via public key of B (PU_B)

decryption via private key of B

↳ on decryption also the content is (encrypted) via private key of A.

ONE WAY:

for (confidentiality):

$A \rightarrow B : E(PU_b, K_s) \,||\, E(K_s, M)$

for comm.,

(i) Encrypt the message using session key &

(ii) Encrypt the session key using public key of B

(Double encryption)

for Authentication:,

$A \rightarrow B : M \,||\, E[PR_a, H(M)]$

User A → B,

(Append) message w/ DS

$H(M)$

& DS is encrypted via PR_a

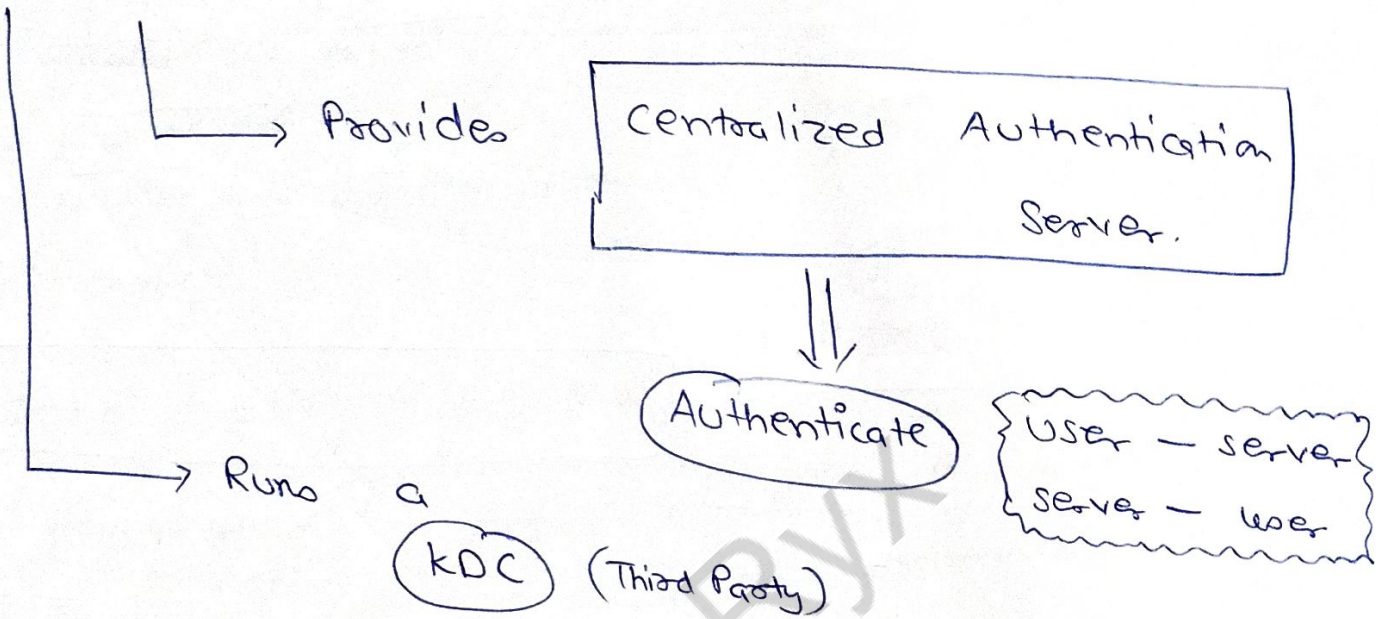
and thus (DS) will be decrypted via PU_a

∴ Authentication (done)

Enrollment No :

Page No :

Kerberos



Components:

1. Authentication server (AS):
   - initial authentication & ticket for TGS
2. Database:
   - verify access rights of users.
3. Ticket Granting server (TGS):
   - Issues the ticket for the servers.

Ticket :

↳ Each [request] requires a [ticket].

↓

Provides [client] access to [server]

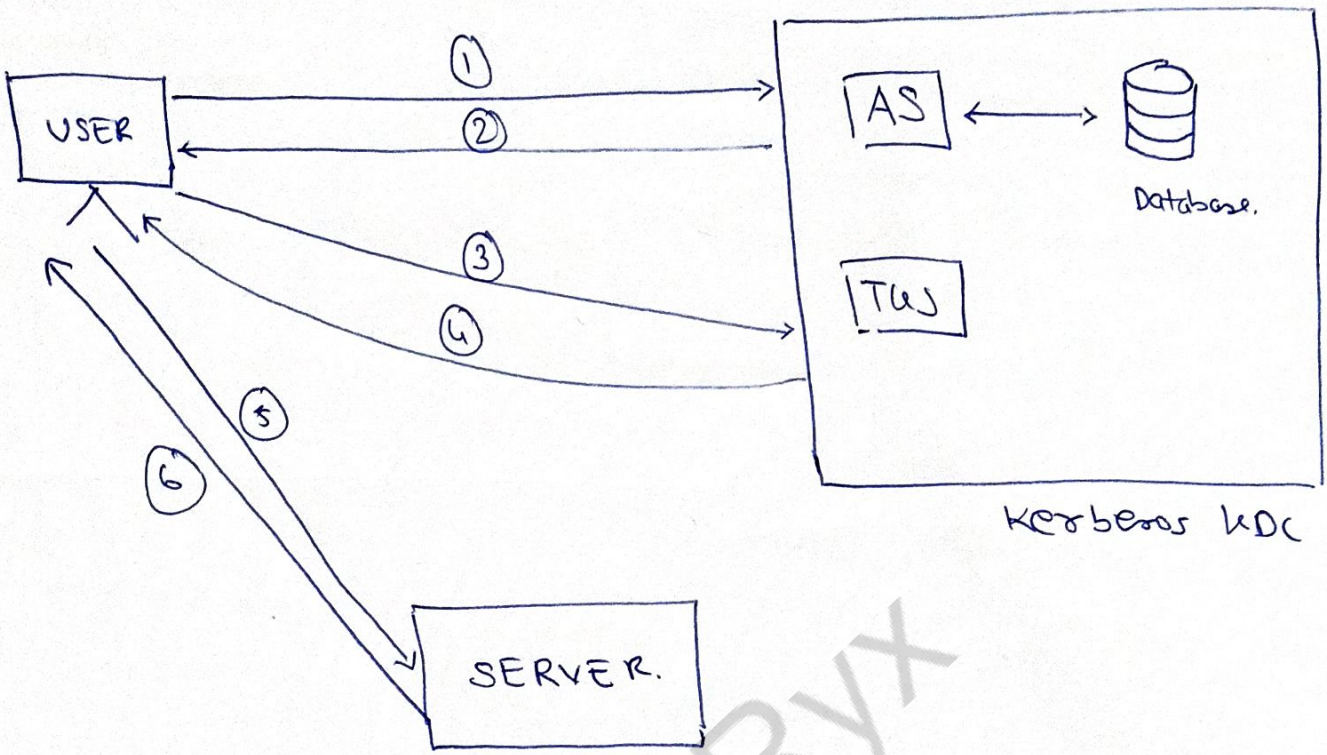
[Tickets] → given by → [TGS]

Ticket Granting Service

TGS → Seals each ticket w/ secret key of server

Sealed Tickets → [send to server] → Get → [Access to the server]

[Access to the server] → [once you're in the server] ⟹ Tickets are useless & expires after use.



S-1 : User login & request services on host

: (1) Request for Ticket-Granting service

S-2 : Authentication server verifies user access

: (2) Gives (Ticket Granting Ticket) + (session key)

S-3 Decrypts the result message using password

∴ (3) Sends the request to TGS

(Ticket has username & user addresses)

S-4 Ticket Granting service decrypt the Ticket

∴ (4) TGS sends (Ticket) + (session key)

S-5 User sends Ticket & Authenticator to server.

∴ (5) User sends request to server

S-6 Server verifie the request & give user access

∴ (6) Provide server authentication & services.

Limitation :

(1) Each network service is modified individually

(2) Requires always on

(3) Less scalable

(4) Result in loss of trust

(5) Stores all passwords with encryption via Single key