Network Security

Page No.:
Date : / /

Unit - 3

Message Authentication Requirements

1) Revelation - Releasing content of message to someone who does not have an appropriate cryptographic key.

2) Analysis of Traffic - Determination of pattern of traffic through the duration of connection & frequency

3) Deception - Adding out of content messages from a fraudulent source into a communication network.
→ leads to mistrust

4) Modification in the content - Inserting, deleting/modifying the contents of message

5) Modification in the sequence - Changing order of messages between parties

6) Modification in the timings - Replay & delay of messages sent between parties

7) Source Refusal - When the source denies being the originator of a message

8) Destination Refusal - When the receiver denies the reception.

Page No.:
Date : / /

Message Authentication Functions

Lower level
↓
need for a function producing authenticator i.e., value that helps in authentication of message

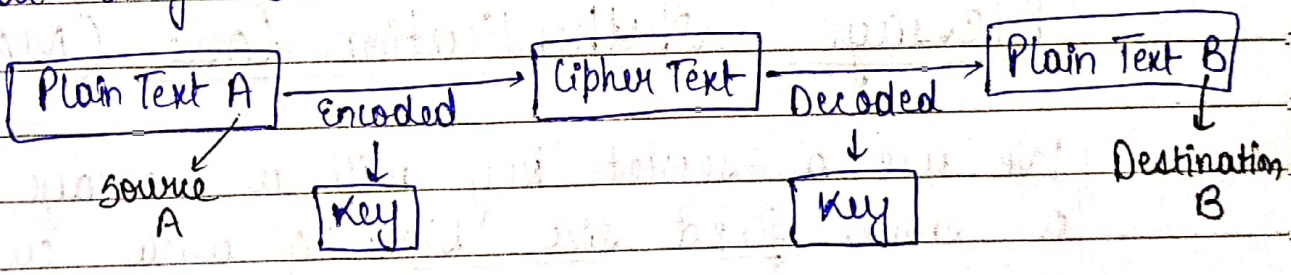
Higher level
↓
lower level function is used to help receivers verify the authenticity of messages

1) Message Encryption

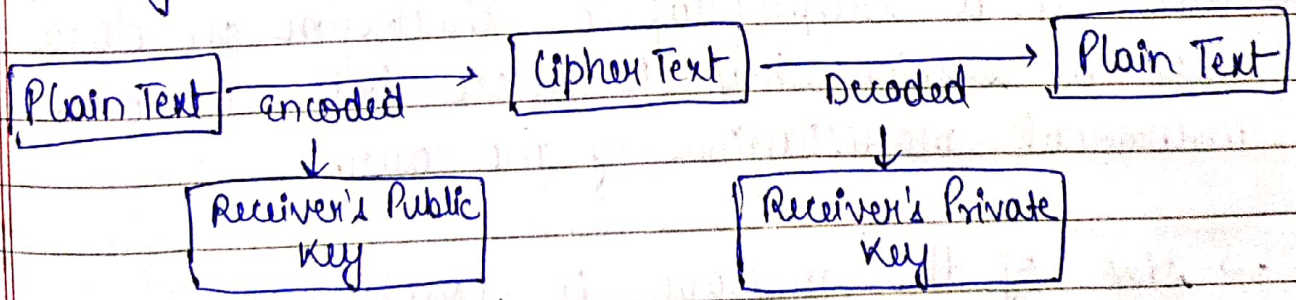
↳ data converted into some ~~unreadable~~ non-understandable form (cipher text) & then sent further

i) Symmetric Key Encryption

→ message is encoded & decoded using a secret key that only sender and receiver share



ii) Asymmetric or Public-Key Encryption



2) Message Authentication Code (MAC)
→ It is a security code that the user of a computer has to type in order to access any account or portal.
→ maintain information integrity
→ confirms authenticity of message

3) Hash function
→ mathematical function that can convert a numerical value into another numerical value that is compressed
→ input can be of any length but output is of fixed length
→ produces a hash value. or message digest

Message Authentication Code (MAC)

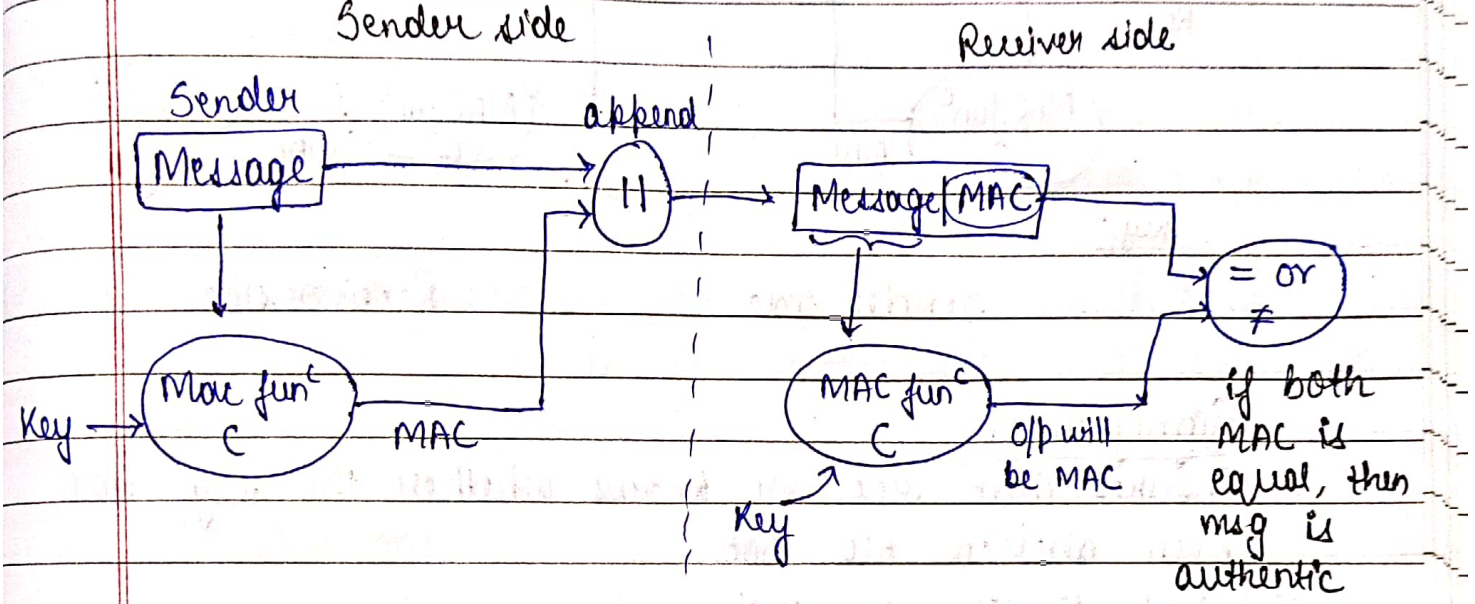
We use a secret key with a message to generate a small fixed size block of data called MAC or cryptographic checksum.

→ MAC is a cryptographic checksum on data that uses a session key to detect both accidental and intentional modifications of the data.

→ size of the message is compressed
→ message is appended with MAC & then sent
→ communicating parties will share a secret common key

$MAC = C(K, M)$, where C → MAC function
K → key
M → message input

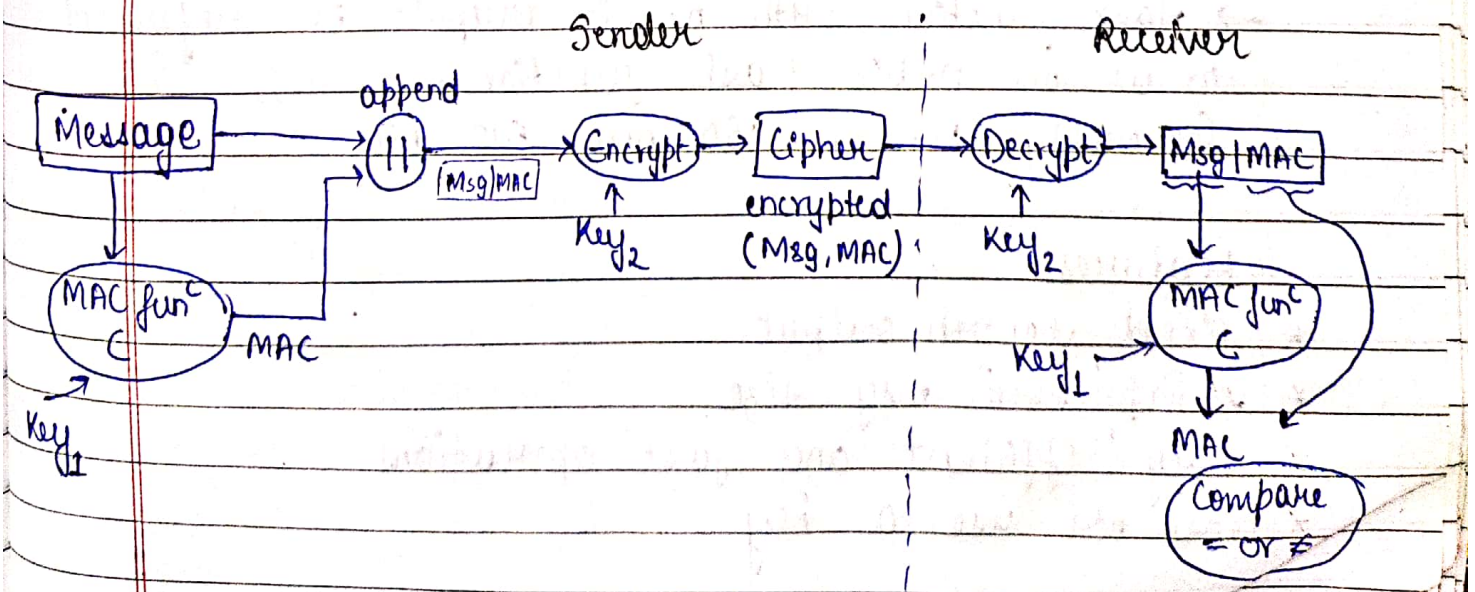
1) MAC - for authentication



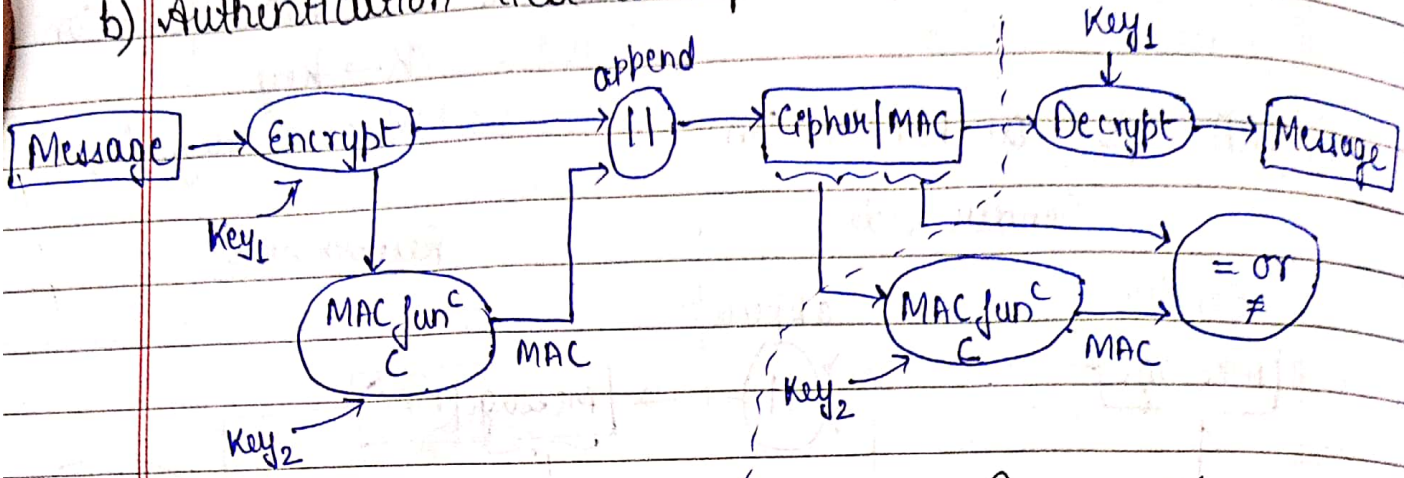
→ only authentication is achieved
→ no confidentiality

2) MAC - for authentication & confidentiality

a) Authentication tied to plain text



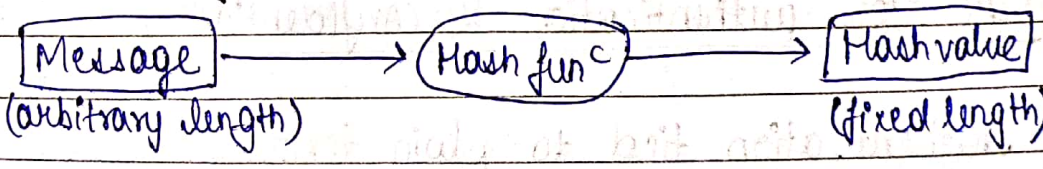
b) Authentication tied to cipher text



Significance

1) Ensures that receiver knows whether the msg has been altered or not
2) Authentication ensured

Hash functions (Compression function)



→ Hash function with n-bit output is referred to as an n-bit hash function.
Generally between 160 and 512 bits

Features

* fixed length output
* compressed msg size
* very efficient and fast operations
* does not use a key

Page No.:
Date : / /

Properties

* Pre-image Resistance - It should be computationally hard to reverse a hash function

* Second Pre-Image Resistance - It should be hard to find a different input with the same hash

* Collision Resistance - It should be hard to find two different inputs of any length that result in the same hash

Applications

1) Password Storage
2) Data Integrity Check
   ↳ used to generate checksums on data files.

Security of Hash functions and MACs

Brute force attacks Cryptanalysis

Brute force attacks

i) Hash functions
→ The strength of a hash fun^{c} against brute force depends solely on the length of the hash code

ii) MACs

→ The level of effort for a brute-force attack on a MAC algorithm can be expressed as $\min(2^k, 2^n)$ where $k \rightarrow$ key length & $n \rightarrow$ no. of message bits.

Cryptanalysis

Cryptanalysis attacks on hash functions & MAC algorithms seek to exploit some property of the algorithm to perform some attack other than an exhaustive search.

Birthday Attacks

→ Attack on Hashing algorithms
→ Collision Resistance of hash functions is the main concern
→ based on the Birthday Paradox
→ brute force attack.

Birthday paradox problem:

In a room full of n students, if asked how many students have birthday on a particular date (say October 10), then its probability

$$P = 1 - (364/365)^{30} \approx 7.9\%$$

However, the probability that at least one student has the same birthday as any other student

is around 70% using the formula:

$$P = 1 - \frac{365!}{(365-n!)*(365)^n}$$ (if $n=30$, then $P=70\%$)

Based on this, finding a specific hash # collision is more difficult than finding a matching hash collision with the same values.

Common use of the birthday paradox attack is digital signature susceptibility.

→ We should use a very strong combination and a long sequence of bit length to prevent birthday or brute force attacks.

MD5 (Message Digest Algorithm)

↳ developed by Ron Rivest
↳ fast and produces 128-bit message digests

Working

Step1: Add padding with the original message
↳ Padding is done such that the total length is 64 bit less than the exact multiple of 512.

eg→ 1000 bits + 472 bits (padding)
512 x 2 = 1024
512 x 3 = 1536 ✓
→ 64 bits less than 1536

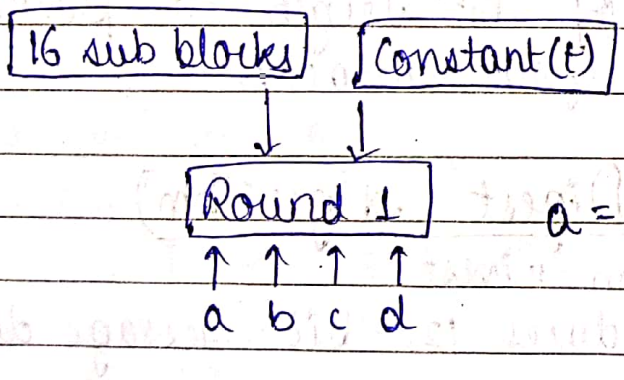
Page No.:
Date: / /

Step 2: Add length to it & that length is equal to (length mod 2^{64}) $\times 64$
↳ then it becomes exact multiple of 512

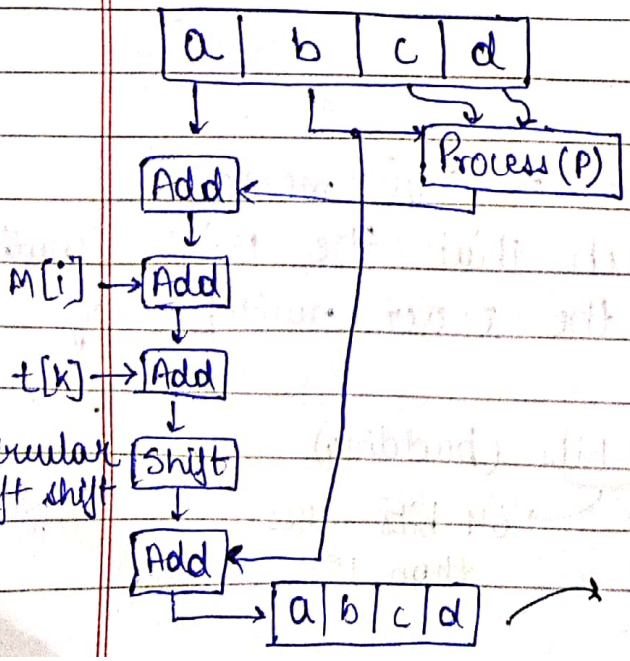
Step 3: Divide that full thing into blocks of 512-bit

Step 4: Initialize 4- chaining variables of 32 bit each (A, B, C & D)

Step 5: Process blocks
↳ Copy four chaining variables into corresponding variables (A=a, B=b, C=c, D=d)
↳ Divide 512- bits block into 16 blocks (32 bit each)
↳ Four rounds



$a = b + ((a + \text{Process } P(b,c,d) + M[i] + T[k])) << \text{shift}$



Page No.:
Date : / /

Secure Hash Algorithm (SHA)

↳ modified version of MD5
↳ output is a message digest of 160 bits in length

SHA properties:

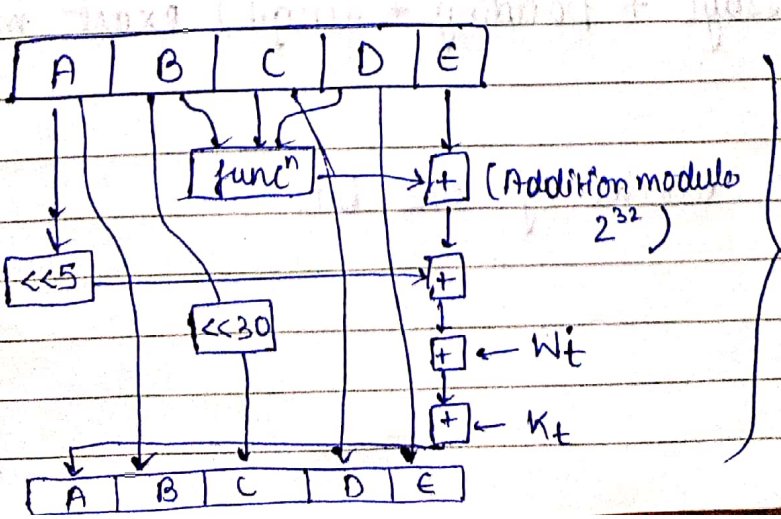
i) Generating original message from digest
ii) finding two messages generating same digest

Working:

i) Message + Padding (64 bit less than exact multiple of 512)
ii) Append length (where length = length mod 2^{64})
iii) Divide the i/p into 512-bit blocks
iv) Five chaining variables A, B, C, D, E of 32 bit each
v) Process blocks
   - ↳ Copy of chaining variables
   - ↳ Divide 512 bits into 16 sub blocks (32 bit each)
   - ↳ Four rounds (20 steps) = 80 steps

$$abcde = (e + Process P + s^5(a) + W[t] + K[t]), a, s^{30}(b), c, d$$

Each round,



In first round: f =
$K_1 = B.C \wedge \bar{B}.D$
Second round:
$K_2 = B \oplus C \oplus D$
$K_3 = B.C \wedge B.D \wedge C.D$
$K_4 = B \oplus C \oplus D$

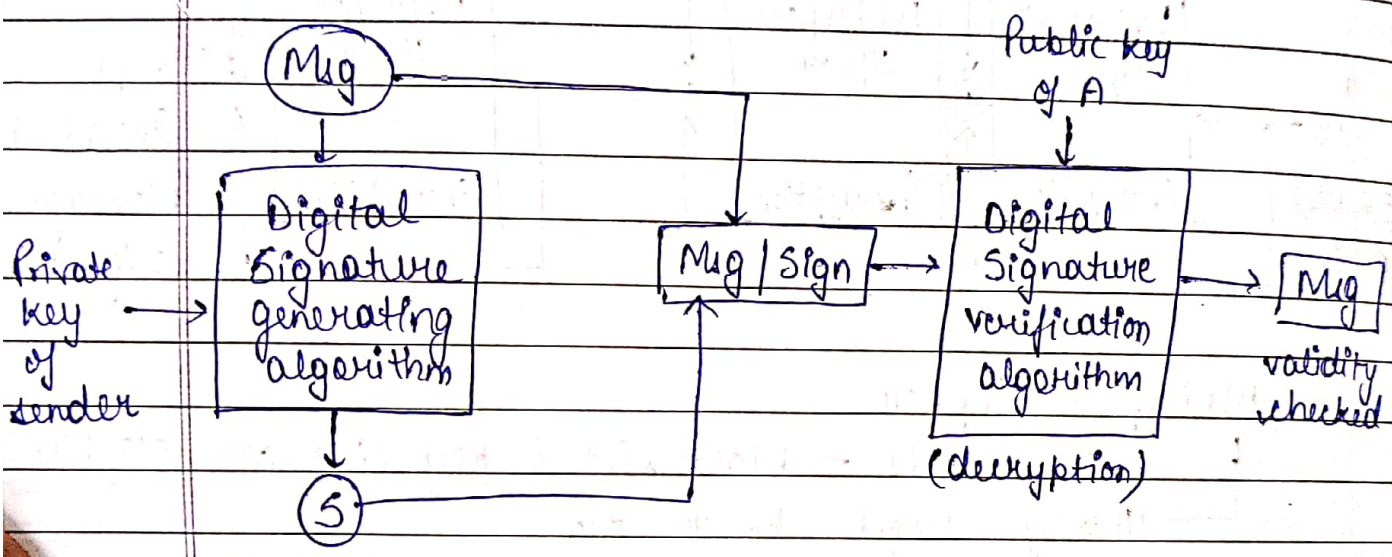
Page No.:
Date : / /

Comparison between MD5 & SHA

	MD5 (Faster)	SHA (Secure)
O/p length:	128 bits	160 bits
Attack to find original message:	2^{128} operations	2^{160} operations
Two msg with same MD:	2^{64} operations	2^{80} operations
Successful Attacks:	Some reported incidents of MD5 break	No such chain
Speed:	Faster	Slower

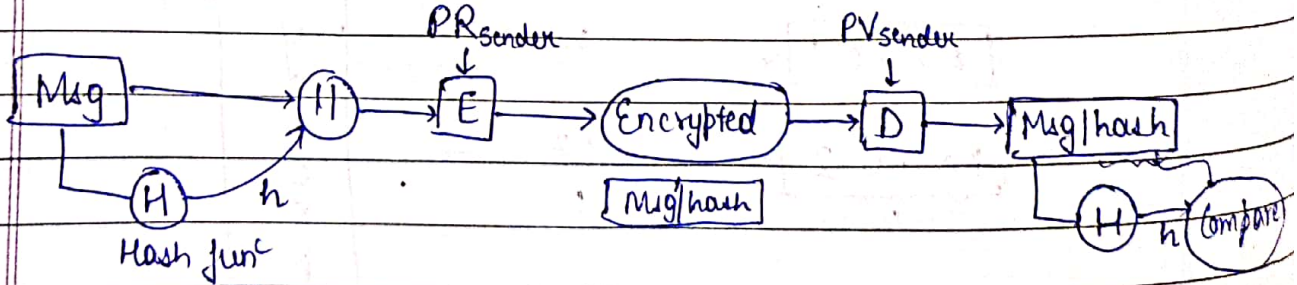
Digital Signature

→ imp role in e-commerce, online transaction etc
→ based on asymmetric key cryptography
→ used for authentication, non-repudiation & msg integrity
→ not used for confidentiality



Digital Signature

Actually, (Msg integrity)



Note→ The signature must use some info unique to the sender to prevent both forgery and denial

→ When we sign a document digitally, we send the signature on a separate document

Date
Page

Properties:

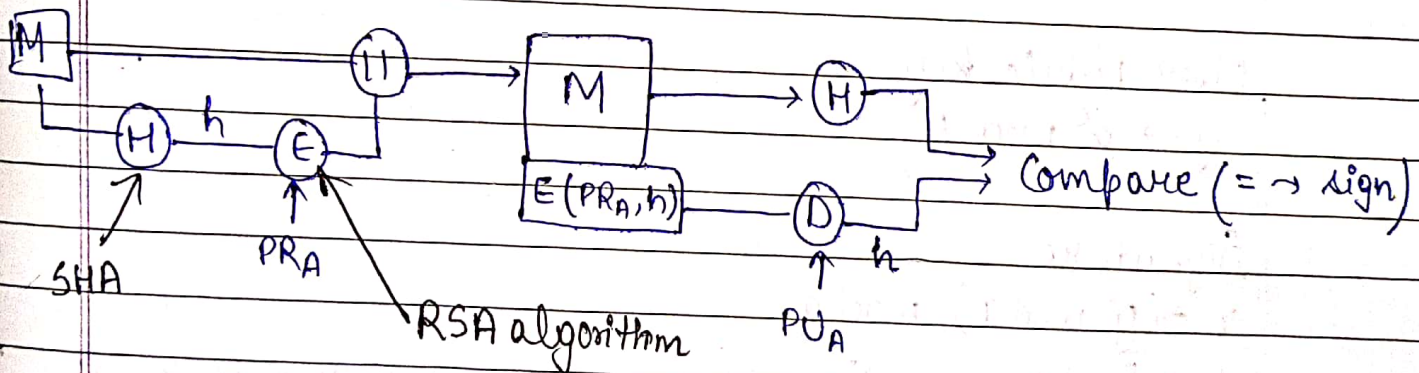
1) must verify the author, date & time of signature
2) must authenticate the contents at the time of signature
3) must be verifiable by third parties, to resolve disputes

Approaches

- RSA
- DSS / DSA

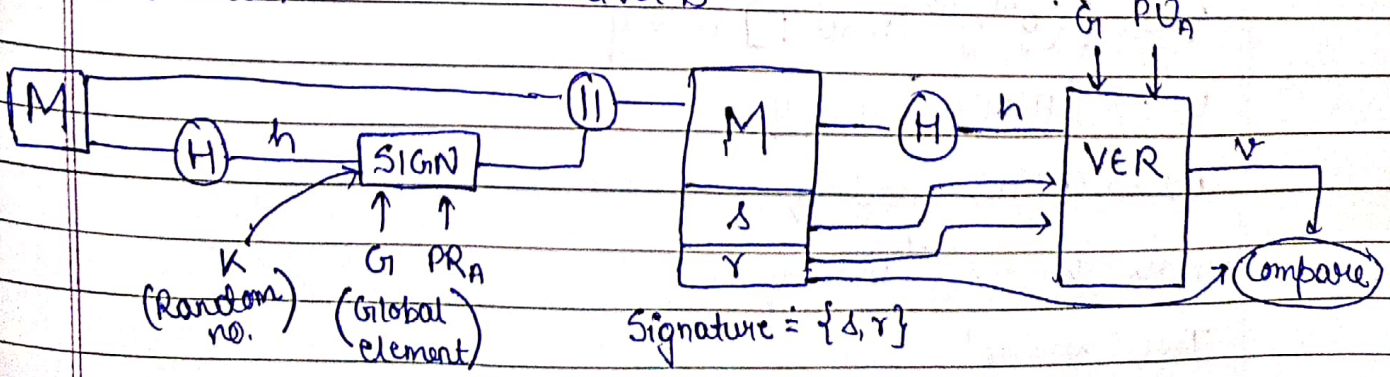
RSA Approach

Sender A ⟶ Receiver B



DSS Approach

Sender A ⟶ Receiver B



Algorithm

Global Public Key Component

* $P \rightarrow$ Prime number $\quad 2^{l-1} < P < 2^{l}$
* $q \rightarrow$ Prime divisor of $(P-1)$
* $g \rightarrow h^{(P-1)/q} \bmod P$

  h any integer

  $1 < h < p-1$

User Private key

$x \rightarrow$ Random number $\quad 0 < x < q$

User Public key

$y \rightarrow g^{x} \bmod P$

Signature

$r \rightarrow (g^{k} \bmod P) \bmod q$

$s \rightarrow [k^{-1}\{H(M)+xr\}] \bmod q \rightarrow$ k any integer $0 < k < q$

Signature $= \{r, s\}$

Verifying

$V \rightarrow [(g^{u1} y^{u2}) \bmod P] \bmod q$

$u1 \rightarrow [H(M')w] \bmod q$

$u2 \rightarrow (r')w \bmod q$

where $w \rightarrow (s')^{-1} \bmod q$

Test: $v = r'$