SHA-1

- input message less than 2^{64} bits length
- output message is 160 bits in length.

Step 1 :- Padding :- Add padding to end of original message in such a way that the length of message is 64 bit ~~shorter~~ of multiple of 512.

Step 2 :- Append Length :- Append 64-bit block to the message. Now message becomes multiple of 512.

Step 3 :- Divide the input into 512-bit Blocks :-

Step 4 :- Initialize Chaining Variables :-

- A to E variables are Initialized (A = 32 bits
  32 × 5 = 160 bits).

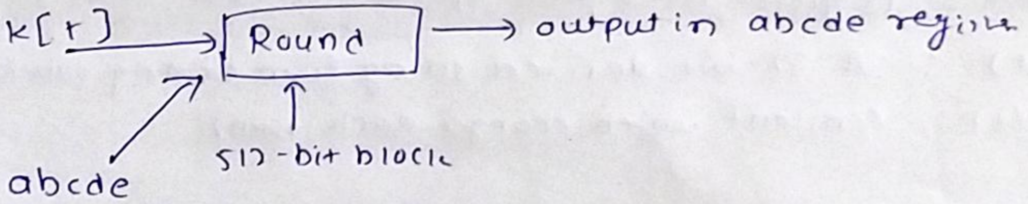
Step 5 :- Process Blocks.

Step 5.1 :- Copy A-E into variable a-e. The combination called abcde as a single register for storing temporary results. and final results.

Step 5.2 :- Divide 512-bit block into 16-sub-block, each of 32 bits. (32 × 16 = 512).

Step 5.3 :- - SHA has four rounds, each round of 20 step.

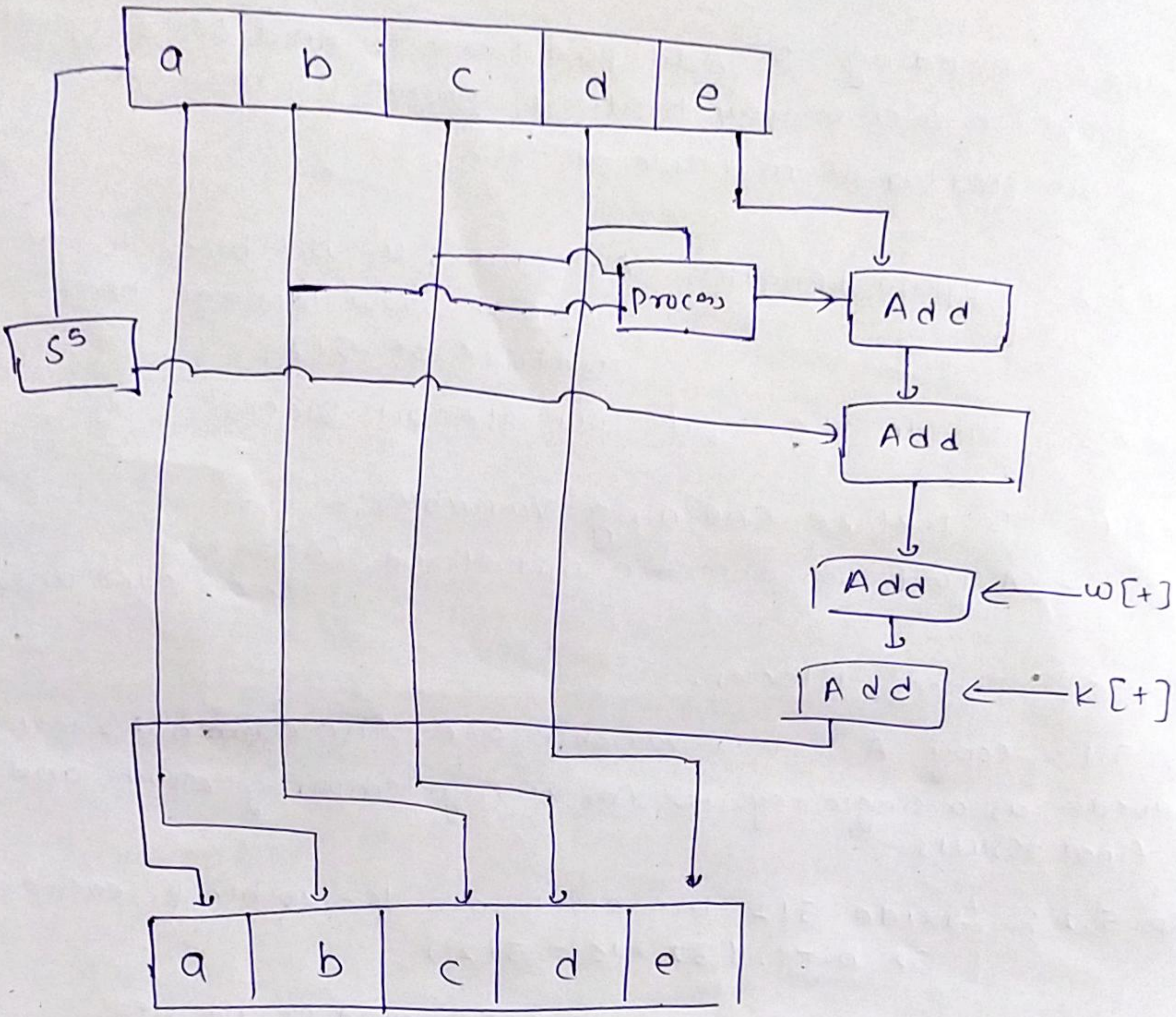
- Each round take input 512-bit block, registers abcde, constant K[t] as three inputs



- Update abcde register after each step.
- K[t] constant value, SHA use four constant values of K[t]

Round		Round	
1	10 to 19	3	40 to 59
2	20 to 39	4.	60 to 79

Operation in Each Round of SHA :-



S^5 :- Circular shift by 5 bit position.

Process :- logical AND, ~~OR~~ OR operations that changes each roun,

w(t) :- A 32-bit derived string from existing sub-block.

K(t) :- constant which changs each roun)

- input msg less than 2^{128} bits length

SHA-512

- output msg 512 bits in length

Step 1 :- Padding :- Add padding to end of original message in such a way that length of message is 128 bit short that multiple of 1024 bit.

Step 2 :- Append Length :- Append 128 - bit block to the message. Now message becomes multiple of 1024 bit.

Step 3 :- Divide Input into 1024 bit blocks.

Step 4 :- Initilize chaining variables.

- A to H variables are Initialized (8 x 32 = 128 bits).

Step 5 :- Process Block

Step 5.1 :- Copy A-H into variable ~~A to~~ a-h . The comb^n abcdefgh will be single register for storing results.

Step 5.2 :- Divide the 1024 - bit block into 16 - sub blocks each of 64 bits.

Step 5.3 :- - SHA has ~~four~~ 80 rounds ~~each round is step~~.

- Each round has 1024 - bit block, the register abcdefgh and k[t] [t = 0 to 9) has three inputs.
- ~~It~~ It then updates abcdefgh for each round.

Each round consist of following procedure :-

Temp 1 = h + ch(e, f, g) + sum e + W_t + K_t

Temp 2 = sum a + maj (a, b, c)

a = Temp 1 + Temp 2.

b = a

c = b

d = c

e = d + Temp 1

f = e , g = f , h = g.

maj, ch = Process of AND, OR operation

W_t = derived from 512-bit input block

K_t = constant.

* Compare MD5 and SHA-1.

	MD5	SHA.
message-digest length	128 bits.	160 bits
	2^{128} operations to break	2^{160} operation to break.
	Less secure than SHA	more secure than MD5.
	Faster than SHA	~~Less~~ Slower than MD5.
	Attack are reported	No attack are reported

* Compare All SHA's:-

	SHA 1	SHA-256	SHA-384	SHA-512
message digest	160	256	384	512
message size	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block size	512	512	1024	1024
Steps in Algo	80	80	80	80
word size (in bits)	32	32	64	64.

* HMAC:-

- Hash-bored message Authentication code.
- HMAC reuse the existing message digest algorithm (MD5, SHA 1) to produce the MAC.
- Used In IP security and SSL protocol on Internet

working:-

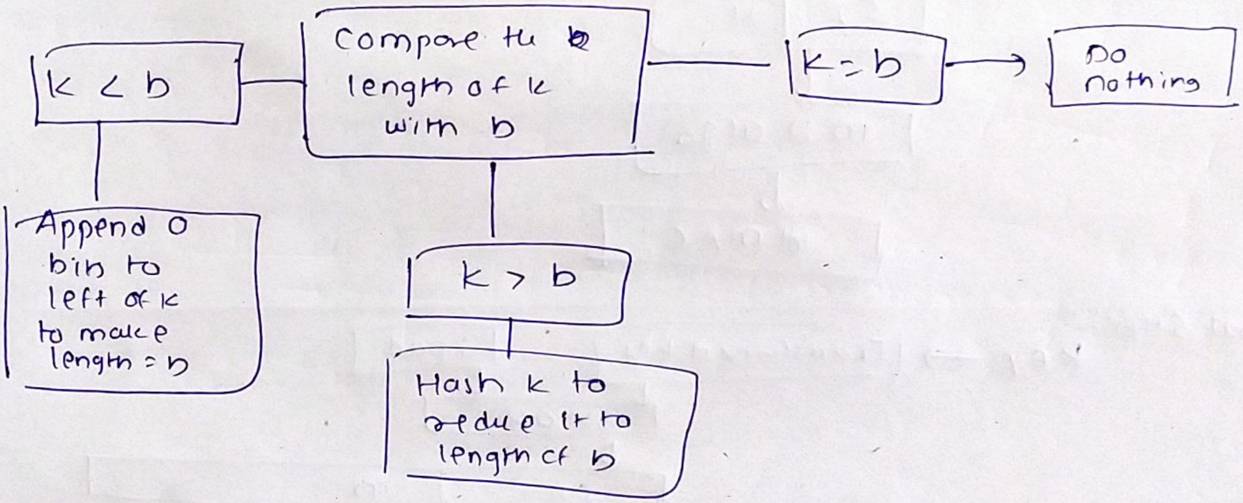
ipad = 00110110 string repeated b/8 times

opad = 01011010 repeated b/8 times.

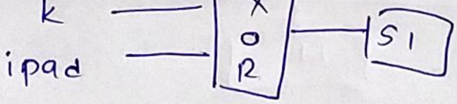
b = bits in ~~of~~ each block.

k = Symmetric key.

Step 1 :- make length of k and b equal.


Compare the length of k with b
k < b
Append 0 bits to left of k to make length = b
k = b
Do nothing
k > b
Hash k to reduce it to length of b

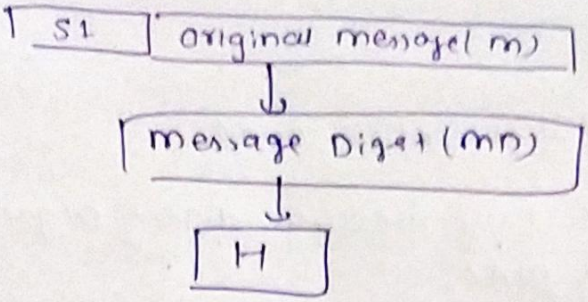
Step 2 :- XOR k with ipad to produce S1 :-



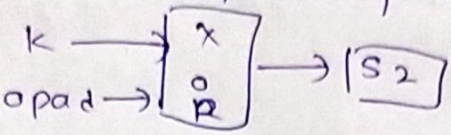
Step 3 :- Append message to S1.

| S1 | Original message |

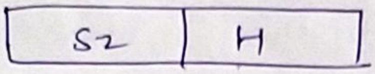
Step 4:- MD algo are applied to output of step 3.



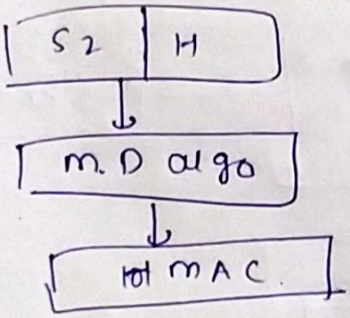
Step 5 :- XOR k with opad to produce S2



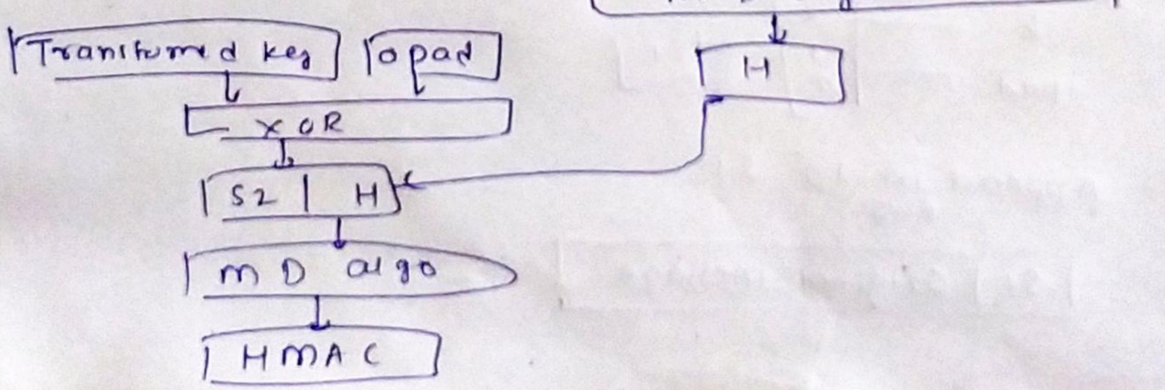
Step 6 :- Append H to S2



Step 7 :- MD algo applied to step 6 :-



Total Steps:-



* Digital Signature :-

- developed for performing digital signatures.

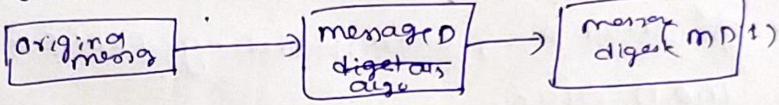
- DS make use of SHA-1 algorimm for calculating message digest and uses to perform digital signature.

- Like RSA, DSA is asymmetric-key cryptographic technique

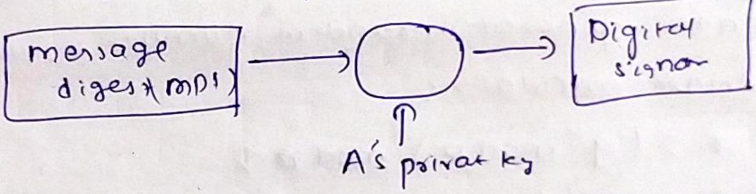
But RSA → encryption
DSA → digital signature.

working of Digital signatures:-

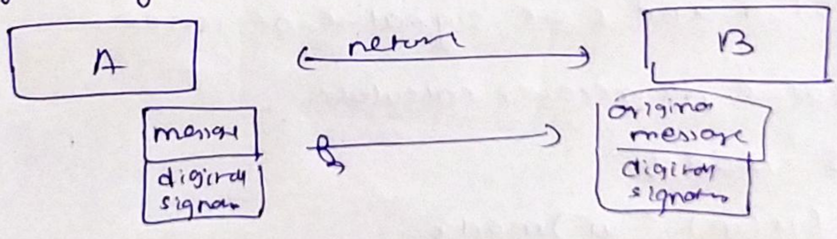
Step 1 :- A uses SHA-1 algo to generate the message digest of original message.



Step 2 :- Now ~~[illegible]~~. message digest is encrypted with ~~[illegible]~~ ('A') private key. The output is digital signature



Step 3 :- Now sender A sends original message (m) along with Digital signature (DS) to receiver B



Step 4 :- Receiver B receives the original message and digital signature. Now B uses same message and MD algo to generate it's own message digest (MD2)

Step 5 :- Receive B decrypt digital signature with the A's public key. and checks if (MD1 = MD2).

Step6 :- If $MD1 \neq MD2$, then it rejects the message.

*k - Principle of digital signature is strong, secure and reliable.

- with DSS Approach:-

Variables in DSA :-

p = prime number of length L bits.

L = no. multiple of 64 betw 512 and 1024.

q = A 160-bit prime facter of $(p-1)$.

$g = h^{(p-1)/q} \mod p$.

x = A number less than q., private key.

$y = g^{x} \mod p$. , corresponding public key.

H = MD algo.

For sending message:-

1. The sender generates random number k, which is less than q.
2. The sender calculates:

$$r = (g^{k} \mod p) \mod q.$$

$$s = (k^{-1}(H(m) + xr)) \mod q.$$

The values of r and s are signature of sender.

o verify signature, the receiver calculates

$$w = s^{-1} \mod q$$

$$u_1 = (H(m) * w) \mod q$$

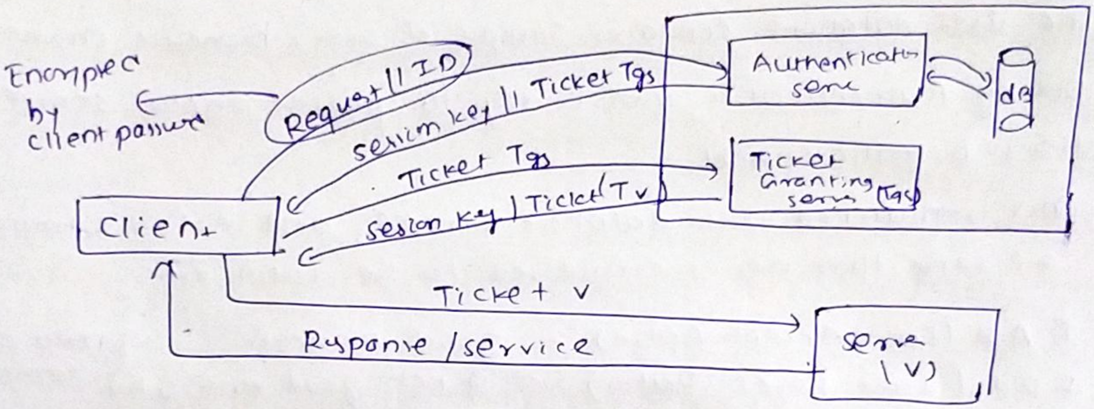
$$u_2 = (rw) \mod q.$$

$$v = ((g^{u_1} * y^{u_2}) \mod p) \mod q.$$

$v = r$, the signature is verified. Otherwise rejected.

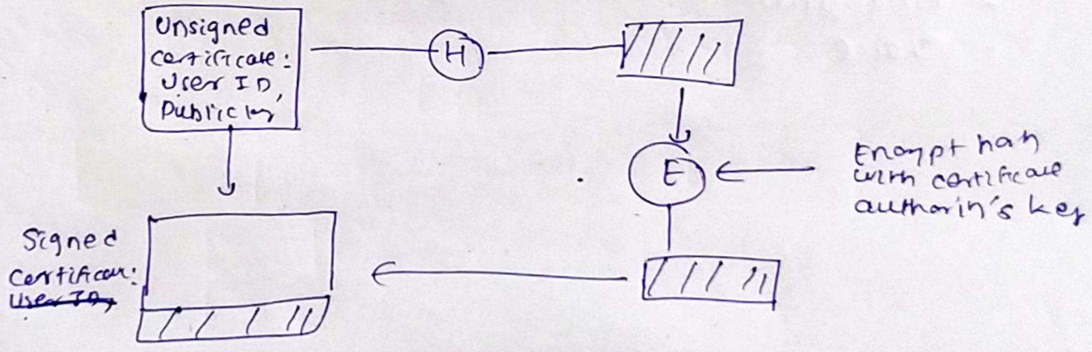
*KERBEROS.

- It is a authenticcation protocol.
- Designed at MIT.
- Authenticatin for client/server applications by using secret-key crptrography.



* X.509 Authentication:-

- X.509 is digital certificate built by ITU., ~~in which~~
- X.509 digital certificate is certificate-based-authentication security framework that is used for secure transaction processing.


Unsigned certificate: User ID, Public key
H
E
Encrypt hash with certificate authority's key
Signed certificate: ~~User ID~~

- format:-

Version number
Serial number
Algorithm
Issuer Name
validity period
subject name
subject public key
signature.

* Biometric Authentication:

- Biometric device ⇒ trying to prove who you are
- Biometric device work on human characteristics, such as fingerprint, voice or Iris of eye.
- The user database contains sample of user's biometric characteristics
- During Authentication, user is req. to provide another sample of user's biometric charact.
- This is matched with database sample, and if two sample are same then user is considered to be valid one
- FAR (False Accept Ratio) - good enough } map average sample
  FRR (False Reject Ratio) - not good enough }
- Biometric Techniques:

Physiological
- Face
- Iris
- Fingerprint
- Voice

Behavioural
- Keystroke
- Signature