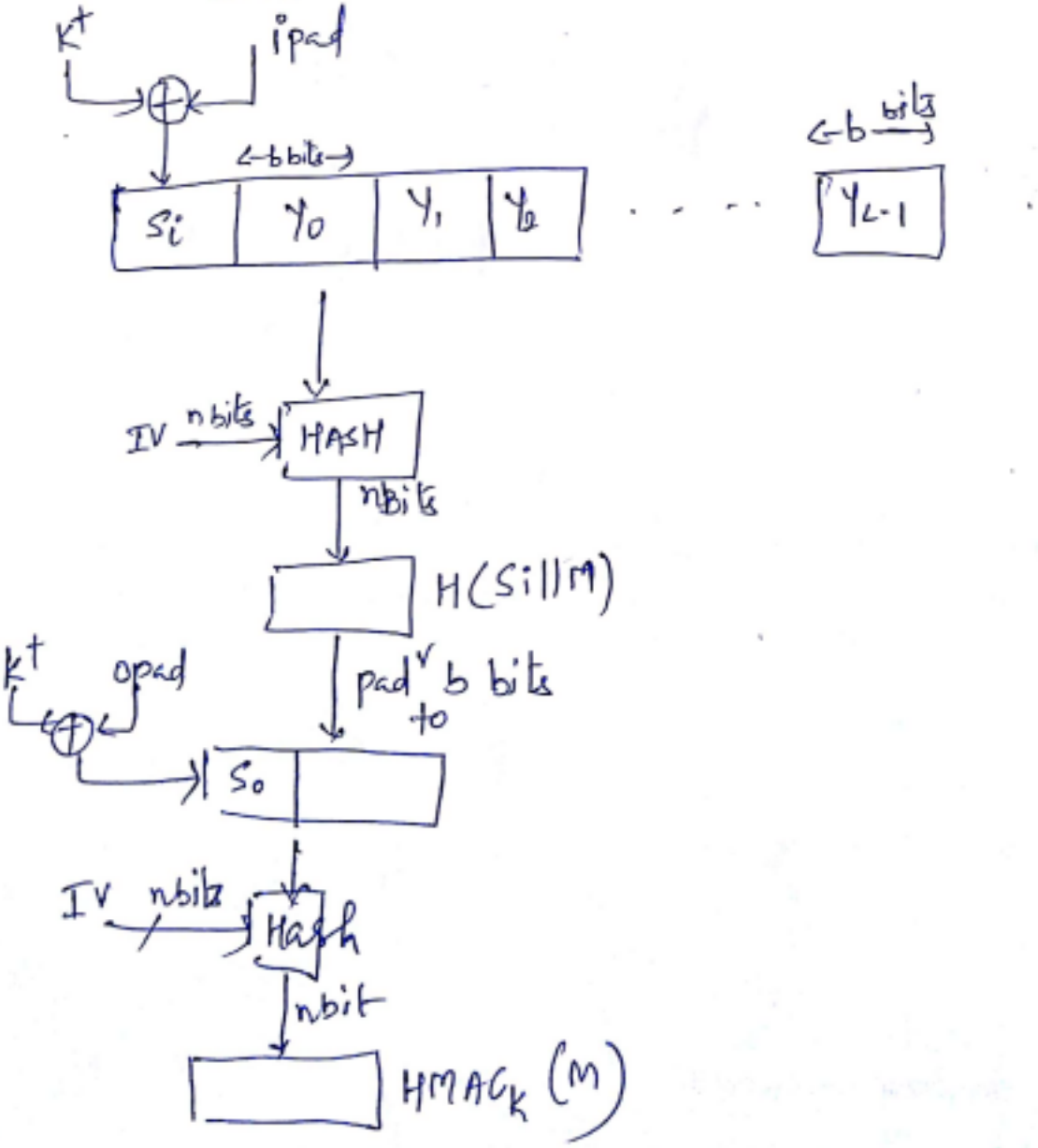
HMAC

- Hash based MAC

- now a days MAC is derived from cryptographic hash function.

because

1) cryptographic hash functions such as MD5 & SHA-1 generally executes faster in SW than symmetric block ciphers such as DES.



H = embedded hash function (MD5, SHA1...

IV = initial value i/p to hash fun.

M = msg i/p to HMAC (including specific

Y_i = i^{th} block of M.

L = no. of block in M.

b = no. of bits in a block

n = length of hash code

K = secret key ; if key length is great
b ; the key is i/p to the hash
to produce n bit key.
if key length is $\geq n$

K^+ = K padded with 0's on the left
So the result is b bits in length

i.e $K_{\text{length}} = b_{\text{length}}$.

ipad = 00110110 (36 in hex)
repeated b/8 times. (512/8 = 64 times)

opad = 01011100 (5C in hex)
repeated b/8 times.

- to get 2 different states out of 1 single master key.
- to minimize potential errors in implementation

HMAC is expressed as

$$HMAC_K(M) = H\left[(K^+ \oplus opad) \,||\, H[(K^+ \oplus ipad) \,||\, M]\right]$$

1. append 0's to the left end of K to create a b-bit string K^+.

   eg. if K is of length 160 bits &
   b = 512
   then K will be appended with
   44 0's

2. XOR K^+ with ipad to produce b-bit block S_i.

3. Append M to Si
4. Apply H to the stream
5. XOR K^+ with opad.
   to produce So of b-bi
6. Apped H result with So
7. Apply H to the stream.
   R & generate the O/P.

Digital Signatures

- In situation where there is not complete trust bet^n sender & receiver.
- The most attractive Sol^n to this problem is digi. sign.
- It is analogous to hand-written signature.
- It must have following property.
  1) it must verify the author & the date & time of signature.
  2) It must authenticate the contents at the time of signature
  3) It must be verifiable by 3rd parties, to resolve disputes.

⇒ Thus signature function includes, authentication function.

Requirements of digital signature

→ Sign. must be a bit pattern that depends on msg being signed.

→ The sign must use some unique info to prevent forgery & denial.

→ It must be easy to prodnee digi.

→ It must be easy to recognize & Ve
digi. sign.

→ It should be computationaly infeas
~~prevent~~ both forge digi. sign.

Scanned by CamScanner

Direct digital sign.

→ direct digi. sign. Involves only communicating parties. (source, Destination)

→ It is assumed destination knows public key of the source.

→ It is formed by encrypting msg or hash code by sender's private keys.

→ All direct schemes described has common weakness.

The validity of scheme depends on the security of sender's private key.

- bcz later sender can claim that the private key was stolen & some1 else forged his signature.
- one sol'n to require every signed msg to include a time stamp. (date & time)

Arbitrated digital Signature

- every signed msg from A to B ~~first~~ goes first to an arbiter (Someone who has power to settle matters) who checks its origin & content.

- all the parties must have great deal of trust on arbitration mechanism.

Arbitrated digi. sign. Techniques

(a) Conventional encryption, Arbiter sees the msg.

secret key K_{XA}

(1) $X \longrightarrow A : M \,||\, E_{Xa}[ID_X \,||\, H(M)]$

(2) $A \longrightarrow Y : E_{Kay}[ID_X \,||\, M \,||\, E_{Kxa}[ID_X \,||\, H(M) \,||\, T]]$

secretkey bet A & Y

→ It is assumed that Sender X & arbiter A share a secret key K_{xa} & that A & Y shares Kay.

- X constructs a msg M & computes is hash value H(M)
- Then X transmits Msg & signature to A.
- The signature contains identifier ID_X of X + hash value. all encrypted by K_{xa}.

→ A decrypts the sig. & checks the hash value to validate the msg.

→ Then A transmits a msg to Y. encrypted wit Kay.

→ msg includes
ID_x, original msg M, timestamp, &
& signature

→ Timestamp informs Y that this msg is timely & not a replay.

→ Y stores M & signature.

⇒ In case of disputes, Y, who claims to have received M from X, sends the following msg to A : signature

$E_{Kay}[ID_x \| M \| E_{Kax}[ID_x \| H(M)]]$

→ Arbiter recovers ID_x, M & the sign. then uses Kxq to decrypt the sign & varify hash code.

- Y cannot directly check X's signature
- The signature is there to settle disp
- both sides must have high degree trust in A.

both sides must trust A to resolve disputes fairly.

Scanned by CamScanner

(b) Conventional encryption,
Arbiter does not sees msg.

(1) $X \rightarrow A$:

$ID_X \| E_{Kxy}[M] \| E_{Kxa}[ID_X \| H(E_{Kxy}[M])]$

(2) $A \rightarrow Y$:

$E_{Kay}[ID_X \| E_{Kxy}[M] \| E_{Kxa}[ID_X \| H(E_{Kxy}[M]) \| I]]$

- provides arbitration but also assure confidentiality.
- In this case, both X, Y shares a secret key K_{xy}.
1) - X transmits its identifier, a copy of msg M encrypted by key K_{xy}. & sign. to A.

$ID_X \| H(E_{Kxy}[M])$

→ signature consists of identifier & the hash value of encrypted msg.

⇒ A decrypts the signature & checks the hash value to validate the msg
A is working with encrypted Msg M

→ A then transmits everything received from X, plus timestamp all encrypted with key to Y

A KR

(3) public-key encryption :→

Arbiter does not see Msg

(1) $X \rightarrow A$:

~~$ID_X \| E_{KR_X}[ID_X \| E_{KU_Y}(T)]$~~

$ID_X \| E_{KR_X}[ID_X \| E_{KU_Y}(E_{KR_X}[M]) \| H(M)]$

(2) $A \rightarrow Y$:

$E_{KR_a}[ID_X \| E_{KU_Y}[E_{KR_X}[M]] \| T]$

$\Rightarrow$ X double encrypts the msg M

1st with x's private key KR_x

& Then with Y's public key KU_y

- This is signed secret version of msg.

- This signed msg + identifier is again encrypted with KR_x & together with ID_x is sent to A

⇒ inner double encrypted msg is secure from Arbiter.

→ A checks to make sure that X's Private / Public key pair is still v
If so, then it verifies the msg.

→ A then transmits to Y, encrypte

KR_a →

→ msg includes ID_x, E (M), Time sta