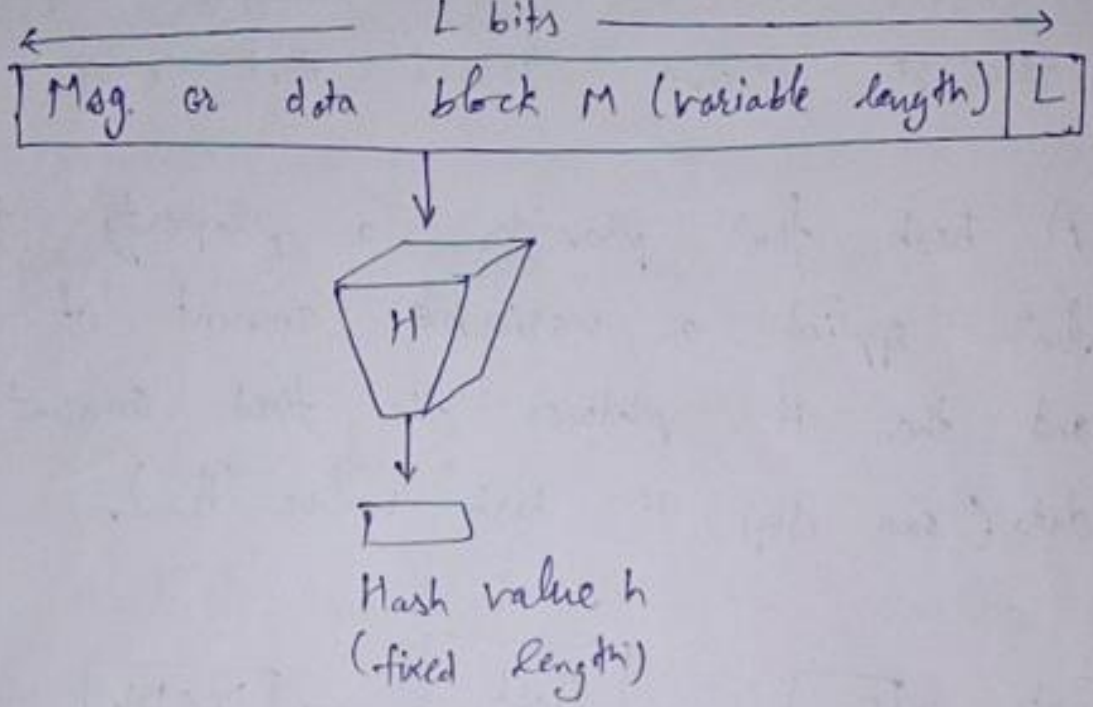
## * Introduction of Hash Function :

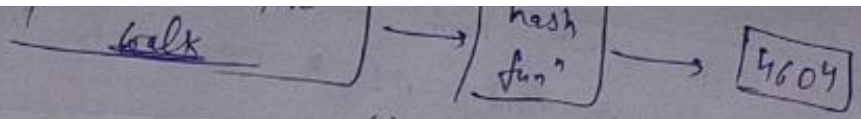
- In hash fun$^n$ H accepts a variable length block of input data called as 'M' and produces the fixed size hash value can be represented as

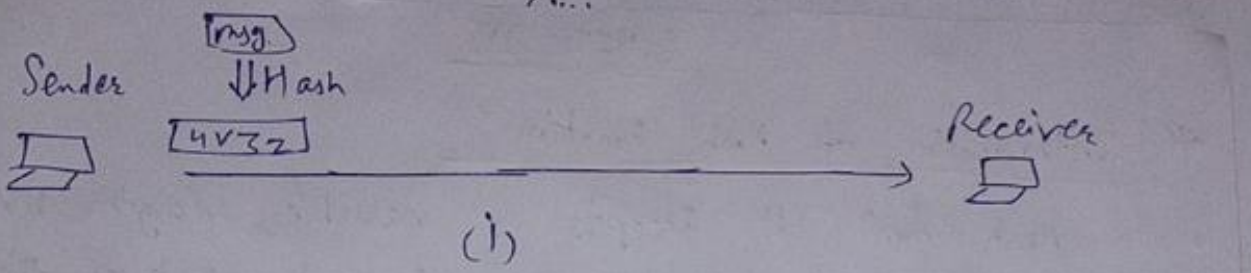
$$h = H(M)$$



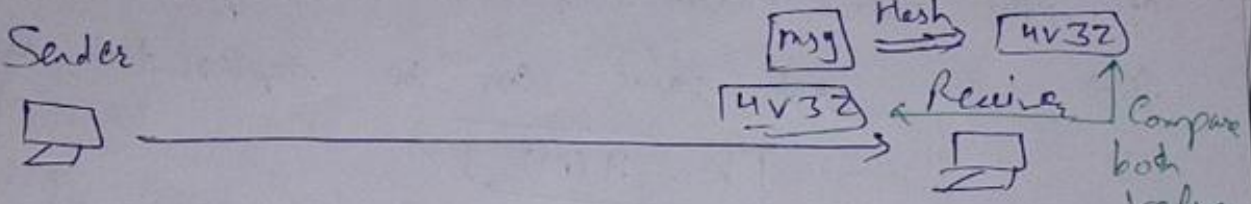

[ Block diagram of hash fun$^n$ ]

- When hash fun$^n$ provides security then is called Cryptographic hash fun$^n$.
- Hash fun$^n$ protects the Integrity of the msg. If Encryption process is apply on msg. with hash fun$^n$, it is also provide authentication and Confidentiality.



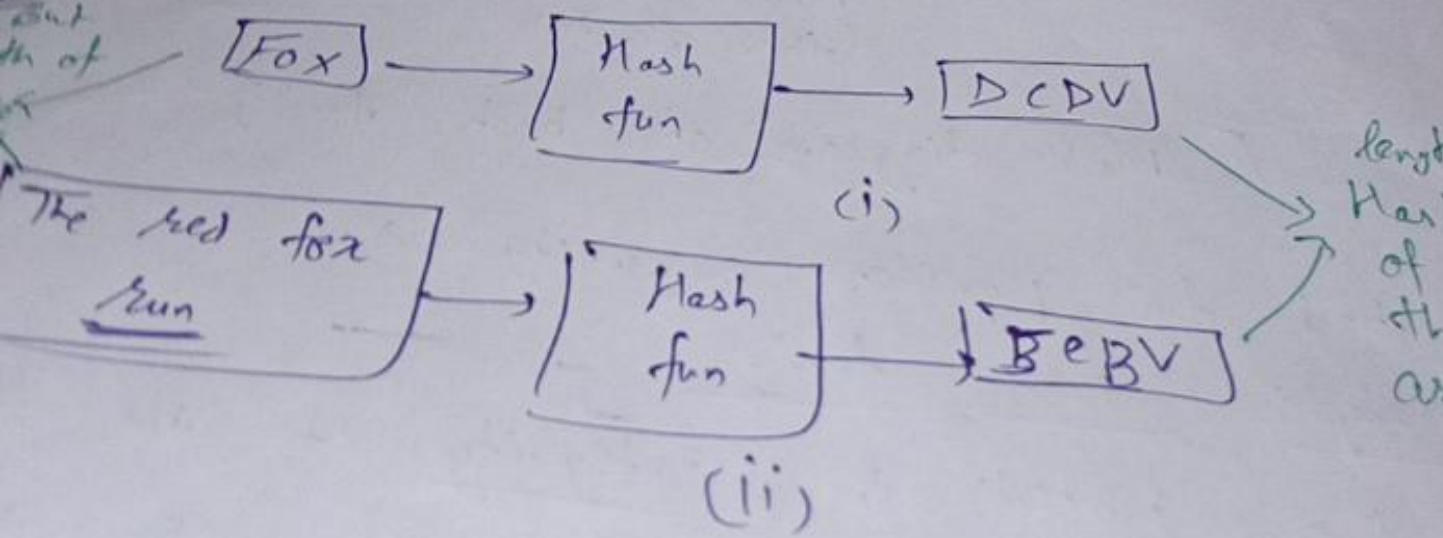




(i)

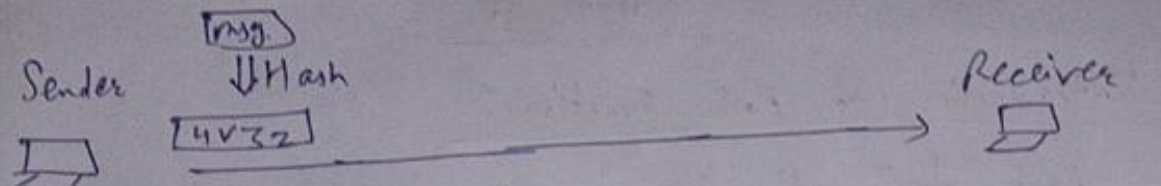



if both value same then Receiver accept the values otherwise Receiver discard that msg.

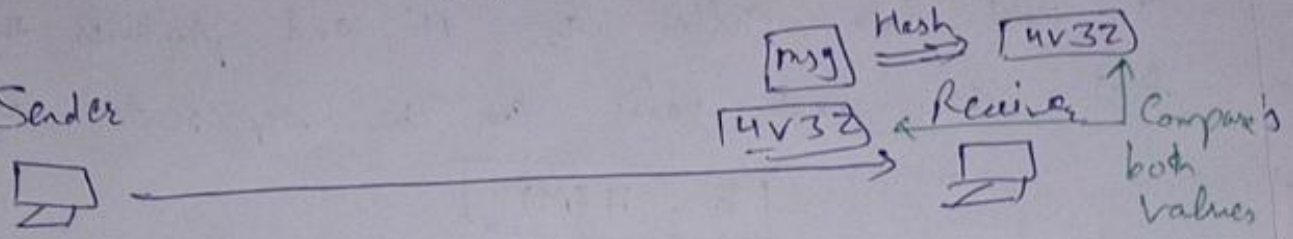
- A hash fun$^n$ provides a property that has fun$^n$ applied on variable amount of data (m) and then it produces the fixed amount of output data ( can say it hash value 'h').




bit or bits changes in the data, the fun$^n$ output data will also change.

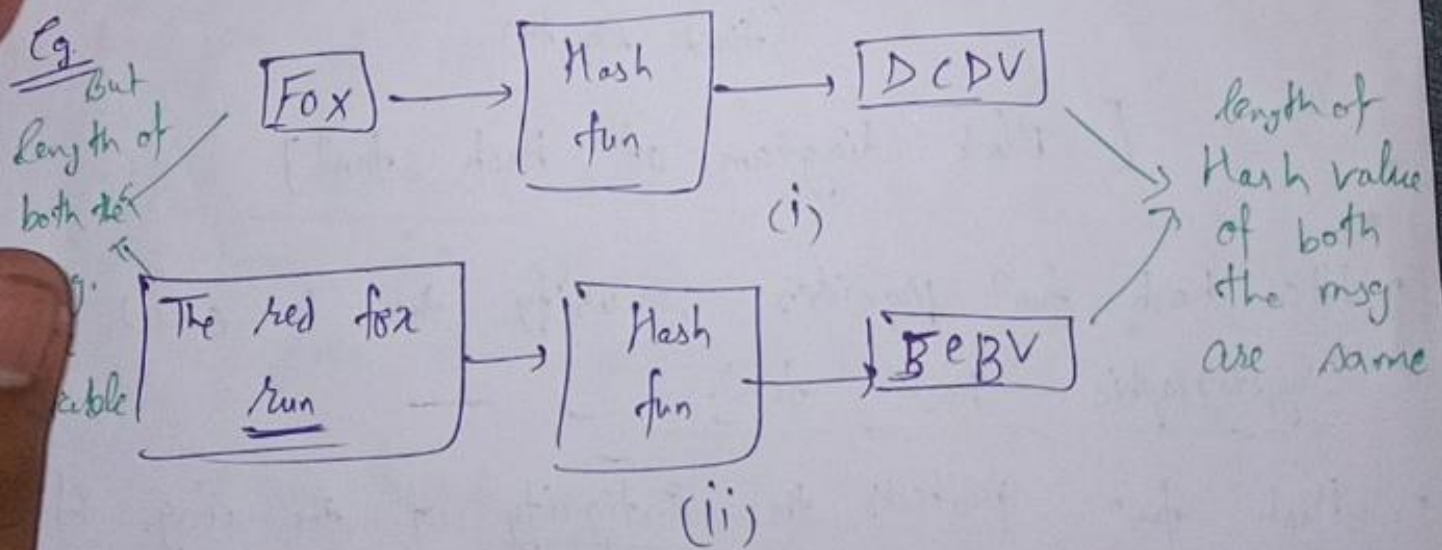



(i)

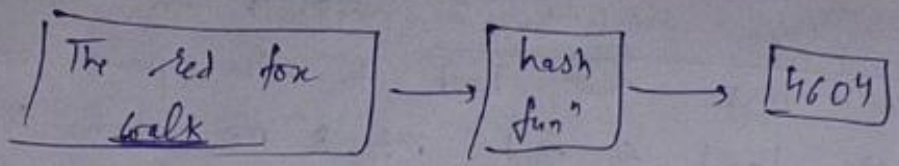



if both value same then Receiver accept the values otherwise Receiver discard that msg.

- A hash fun$^n$ provides a property that has fun$^n$ applied on variable amount of data (m) and then it produces the fixed amount of output data (can say it hash value 'h').




If any bit or bits changes in the data, then whole hash fun$^n$ output data will also change.




(iii)

- Cryptographic has fun^n is one-way fun^n, which is practically infeasible to invert. (it means we can get the original msg. from hash value.) (like in fig (i) from DCDV we can't get original msg. Fox.)
- The most popular hashing algo. is MD5 and SHA. (Msg Digest 5) & (~~Simple~~ Secure Hash Algo.).

## #Properties of hash fun^n:

a) Compression: o/p of hash fun^n is mush smaller than the size of i/p.

b) Pre-Image Resistance: difficult to find the input from given hash fun^n o/p i.e. $X = H(m)$. So if X is given then difficult to find msg. m.

- Weak Collision Resistance: given msg. m1, it is difficult to produce another msg m2 such that H(m1)=H(m2) i.e. it is infeasible to find two diff. msgs. with the same hash value.

↳ Process the entire n-block of the input data.

- Strong collision Resistance: difficult to find any two diff. msg. that hash to the same value. i.e. it is hard to find m1 & m2 such that same hash value H(m1) = H(m2).

- # Strong and Weak Collision Resistance are not the same: Weak collision resistance is bound to a particular input, where as strong collision resistance applies to any two arbitrary inputs.

2) # Characteristics of Hash Function:

3) • it is quick to calculate hash value (h) for any given msg. i.e. $z = H(m)$ ( m = msg., H = Hash fun^n, z = generated value from Hash ).

- Hash fun^n (H) can be applied to variable length of data block.

4) • A small change in a msg. should change the hash value.

- Hash fun^n has one-way property, it is impossible to generate msg. from given hash value. (can not decrypt & can not get original msg. from Hash fun^n.)

- Hash fun$^n$ uses all the input data.
- Hash fun$^n$ 'uniformly' distributes the data across the entire set of possible hash values
- the hash fun$^n$ generates very diff. Hash values for the similar msg.

## * Simple Hash fun$^n$ :

- Here, there are two simple hash fun$^n$, all hash fun$^n$ are operating using same principle.

1) The msg. file is like a simple input it open a sequence on n. bit blocks. (if msg. file is too long then it is converted into n. bit blocks.)

2) When input is processed only one block at the given time in iterative fashion to generate an n. bit hash fun$^n$.

- The simple hash fun$^n$ is bit-by-bit XORing done of every block.
- This can be shows the following ways:

$$C_i = B_{i_1} \oplus B_{i_2} \oplus \ldots \oplus B_{i_m}$$

where $C_i = C_i$ is $i^{th}$ bit if hash code, $1 <= i <= n$

$m$ = $n$ is the no. of block in the input

$b_{ij}$ = $i^{th}$ bit in $j^{th}$ block

$\oplus$ = XORing operation

| | bit 1 | bit 2 | .... | bit n |
|---|---|---|---|---|
| block 1 | $b_{11}$ | $b_{21}$ | .... | $b_{n1}$ |
| block 2 | $b_{12}$ | $b_{22}$ | .... | $b_{n2}$ |
| | ⋮ | ⋮ | ⋮ | ⋮ |
| block m | $b_{1m}$ | $b_{2m}$ | | $b_{nm}$ |
| hash code | $C_1$ | $C_2$ | | $C_n$ |

- When this operation performs it produces a simple parity for each bit location this process is known as a redundancy check.

- To increase the complexity and improve performance use simple ways i.e. one-bit circular shift, and also rotation on hash value after the every block is processed. (permutation is performed inside the hash fun$^n$ to increase the complexity).

- The steps are summarized by as follows to increase complexity:

↳ Set n-bit hash value initially zero.

↳ process the each and every successive n-block of data is as follows:

⇒ the current hash value is rotate to the left by one bit.

⇒ perform the XOR operation in b/w block and hash value.

## * Its Requirements & Security (Hash fun$^n$):

| Requirement | Description |
|---|---|
| 1) variable input size | H can be applied to a block of data of any size. |
| 2) Fixed o/p size | H produces a fixed length o/p. |
| 3) Efficiency | H(x) is relatively easy to compute for any given, making both HW and SW implementations practicals. |

The first 3 properties are requirements for the practical app of hash fun$^n$.

| | |
|---|---|
| 4) preimage resistant (one-way property) | for any given hash value, it is computationally infeasible to find y such that H(y) = h. |
| The 4 property mysg. but virtually | easy to generate a code given a impossible to generate a msg. given a code. |

If the hash fun$^n$ is not one way, then an attacker can easily discover the secret value.

| | |
|---|---|
| 5) Second preimage resistant (Weak Colligion resistant) | For any given block, it is Computationally infeasible to find $y \neq x$ with $H(y) = H(x)$. |

From the 5 property it is impossible to find an alternative msg. with the same hash value as a given msg.

This prevents forgery when an encrypted hash Code is used. If this property were not true an attacker would be capable of the following sequence: First, observe or intercept a msg plus its encrypted hash code; Second, generate an unencrypted hash Code from the msg; third, generate an alternate msg. with the same hash Code.

| | |
|---|---|
| 6) Collision resistant (Strong collision resistant) | It is Computationally infeasible to find any pair $(x, y)$ such that $H(x) = H(y)$. |

If 6 property is satisfied then it is referred to as a strong hash fun$^n$.

also written as $C = E_K[\cdot]$.

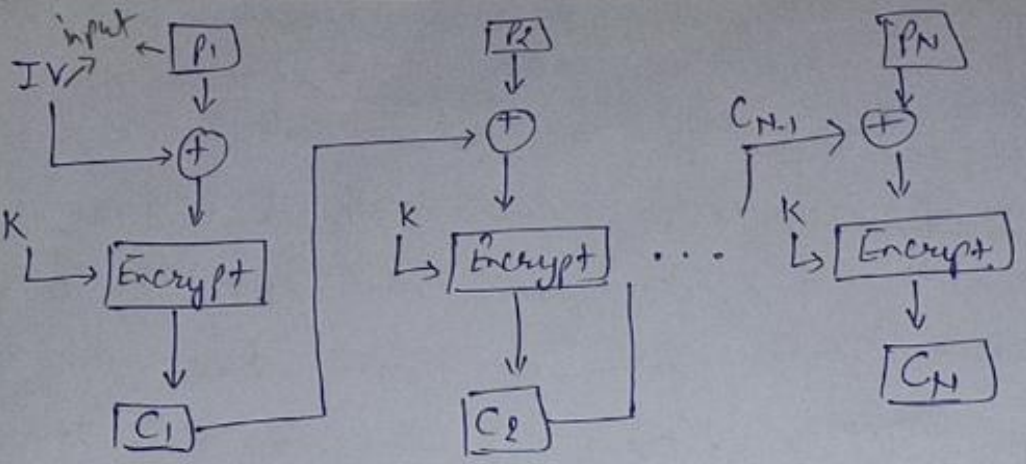
* MAC based on Hash Function (HMAC):

A strong hash fun" protects against an attack in which one party generates a msg. for another party to sign.

for eg: Suppose Bob writes an IOU msg., sends it to Alice, and she signs it. Bob find two msgs. with the same hash, one of which requires Alice to pay a small amount and one that require a large payment. Alice signs the first msg. and Bob is then able to claim that the second msg. is authentic.

* Hash Function based on Cipher Block Chaining:

- Two major categories of hash fun" are: dedicated hash fun" and block cipher based hash fun".
- Block cipher is a popular encryption - decryption primitives. To encrypt, the block cipher accepts a key K and a plain text block P as input and produces a cipher text block $C = E(K, P)$, also written as $C = E_K(P)$.




82
75

- By the definition of CBC (Cipher Block Chaining):

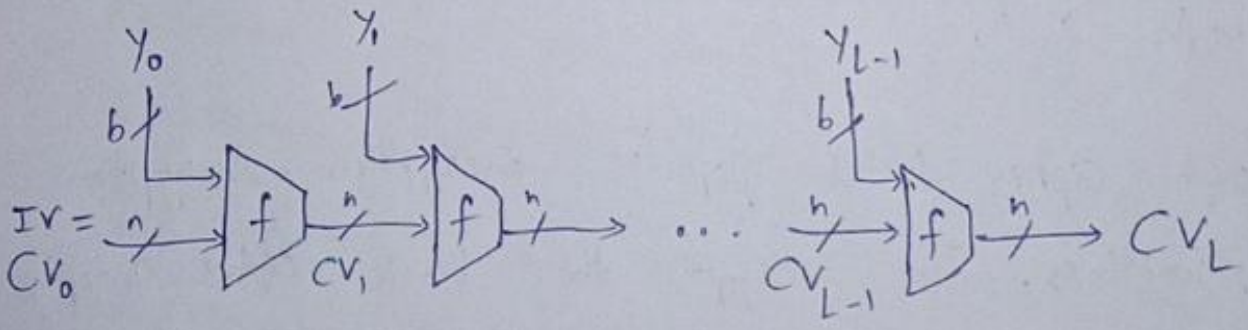
$$\boxed{CBC: C_j = E(K, [C_{j-1} \oplus P_j])}$$

for $C_1$ if $j=1$ then

$$C_1 = E(K, [C_0 \oplus P_1])$$

$$C_1 = E(K, [IV \oplus P_1]) \quad [\because C_0 \equiv IV]$$

## How Hash fun$^n$ work based on CBC:




IV = initial value
$CV_i$ = Chaining variable
$Y_i$ = ith input block
f = Compression algo. /fun$^n$.
has an initial value of the algo. [$CV_0$].

L = no. of input blocks
n = length of hash code
b = length of input block

... chaining variable that is specified as part

- The hash fun$^n$ takes an input msg. and partitions it into L fixed-sized blocks of b bits each.

$Y_0 \ldots Y_{L-1}$ = L fixed sized blocks

- if necessary, final block is padded to bits.
- The final block also includes the value of the total length of the input to the hash fun$^n$.
- Because the inclusion of length makes the job of the opponent more difficult.
- Either the opponent must find two msg. of equal length that hash to the same value or two msg. of differing lengths that, together with their length values, hash to the same value.
- the hash algo. alt involves repeated use of a compression fun$^n$, f, that takes two inputs (an n-bit input from the previous step, called the chaining variable, and a b-bit block) and produces an n-bit output.
- At the start of hashing, the chaining variable has an initial value that is specified as part of the algo. [CV$_0$].

- The final value of the chaining variable is the hash value. Often, hence the term Compression. [$CV_L$].

The hash fun$^n$ can be summarized as:

$$CV_0 = IV = \text{initial } n\text{-bit value}$$
$$CV_i = f(CV_{i-1}, Y_{i-1}) \quad 1 \leq i \leq L$$
$$H(M) = CV_L$$

## ✳ Secure Hash Algorithm (SHA-1):

### # Introduction:

- The SHA was developed by National Institute of Standards and Technology (NIST). It is based on MD4 algo. Based on diff. digest lengths, SHA includes algo. such as SHA-1, SHA-256, SHA-384 and SHA-512. (versions of the SHA or output of the Hash algo.).

- Unlike encryption, given a variable length msg x, a secure hash algo. computes a fun$^n$ H(x) which has a fixed bits. When a msg of any length is less than $2^{64}$ bits is input, the SHA-1 produces a 160-bit output called msg. digest. (Input msg. is the length of variable fun$^n$ or final ... mean. output of hash

= and output of that msg or we can say hash value or hash code is the fixed size).

- SHA-1 called secure because it is computionally infeasible to find a msg which corresponds to a given msg digest or to find two diff. msg. which produces the same msg. digest. (Strong Collision resistant property is true in SHA-1).

- The most commonly used hash fun$^n$ from the SHA family is SHA-1. SHA-1 is used in SSL/TLS, PGP, SSH, MIME and IPSec for security and authentication purpose.

S.S.L - Secure Socket Layer
TLS - Transport Layer Security
PGP - Pretty Good Privacy in mail communicat$^n$
SSH - Secure Shell
MIME - Mail Communicat$^n$
IPSec. - IP Security

# Features of SHA-1:

- Msg or data file used as input in SHA-1 to compute a msg digest mean. output of hash fun$^n$ or final hash value. (size of msg. is variable or output is fixed size).

- the msg. or data file should be considered to be a bit string.
- the length of the msg. is the no. of bits in the msg. (the empty msg. has length 0.)
- the purpose of msg. padding is to make the total length of a padded msg. a multiple of 512.
- Eg: if any msg. length is 1000 bits, we padded 24 bits to make msg. into multiple of 512 bits. So the msg. is 1000 + 24 = 1024 bits. length now it is divided into 512 bits. that mean multiple of 512 bits.
- the SHA-1 sequentially processes blocks of 512 bits when computing the msg. digest.

## # Working of SHA-1:

- SHA-1 works with any input msg. that is less than $2^{64}$ bits in length.
- the output of SHA is a msg. digest, which is 160 bits in length.

(Input msg. length can be variable & output msg. length should be fixed.)

## ✳ MAC based on Hash Function (HMAC):



### Step 1: Padding

- the first step of SHA-1 is added padding to the end of original msg. to prepare msg. in multiple of 512 bits.

### Step 2: Append Length

- the length of msg. excluding the length of the padding is now calculated and appended to the end of the padding as 64-bit block. (msg. length is 64 bits short of multiple of 512.)

(append length means find the length of whole msg. and appended to the last block of the input msg. purpose of appending the length is to increase the complexity for this algo.)

### Step 3: Divide the Input into 512 bit blocks:

- the input msg. is now divided into blocks, each of length 512 bits.

for eg. if msg. is 1024 bits. then it is divided into 2 blocks and length of the both block is 512 bits.

## Step 4: Initialize Chaining Variables:

- Now, five chaining variables A to E are initialized.

| A | B | C | D | E |
|---|---|---|---|---|

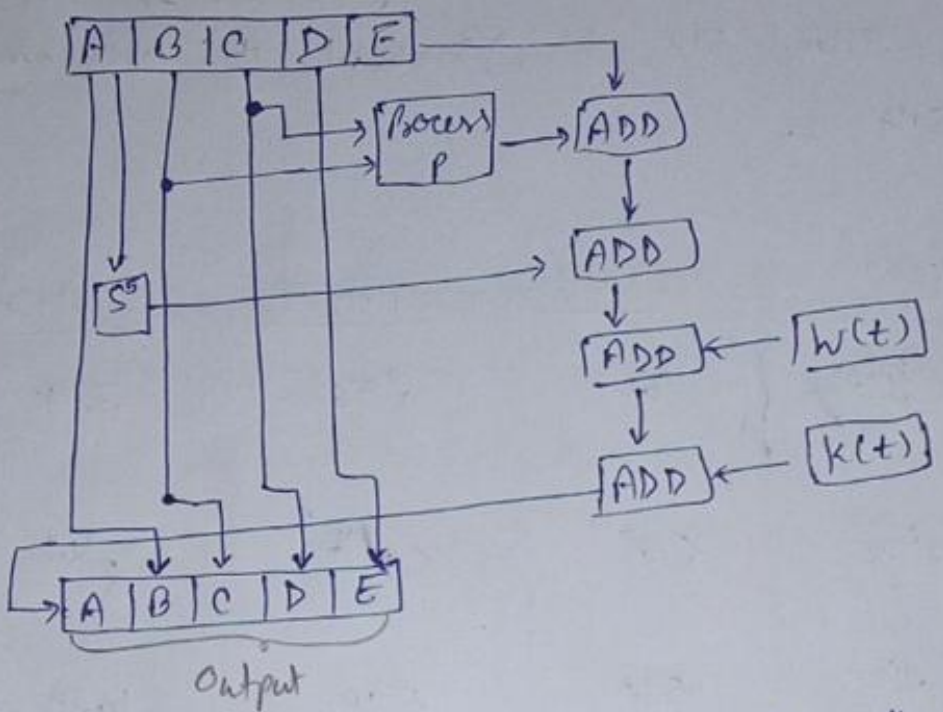
- Each of 32 bits variable produces 160 bits length of msg. digest. (means variable A is 32 bits, B is 32 bits and so on ... so total length of msg. digest is 160 bits.)

## Step 5: Process Block & output:

- Combination of A-E chaining variable is called ABCDE, will be considered as a single register. (160 bits, 1 single register is used)
- Now divided the current 512 bit block into 16 sub blocks, each consisting of 32 bits (32 X 16 = 512).
- SHA-1 perform fours rounds.
- Each round takes the current 512 bit block, the register ABCDE and constant K(t) (where t=0 to 79) as input.
- SHA-1 consists of fours rounds, each round containing 20 iteration. So total iteration is 80.

# HMAC Concept:

- the logical operation of a single SHA-1 (78) iteration looks as shown in fig.




- Mathematically, an iteration consists of the following operation:

$$ABCDE = E + \text{Process } P + S^5(a) + W(t) + K(t)$$

Here, $W(t)$ = expanded msg. word of round $t$;
$K(t)$ = Round of $k$. constant of round $t$;

## * Comparison Based on parameters of diff. SHA version :

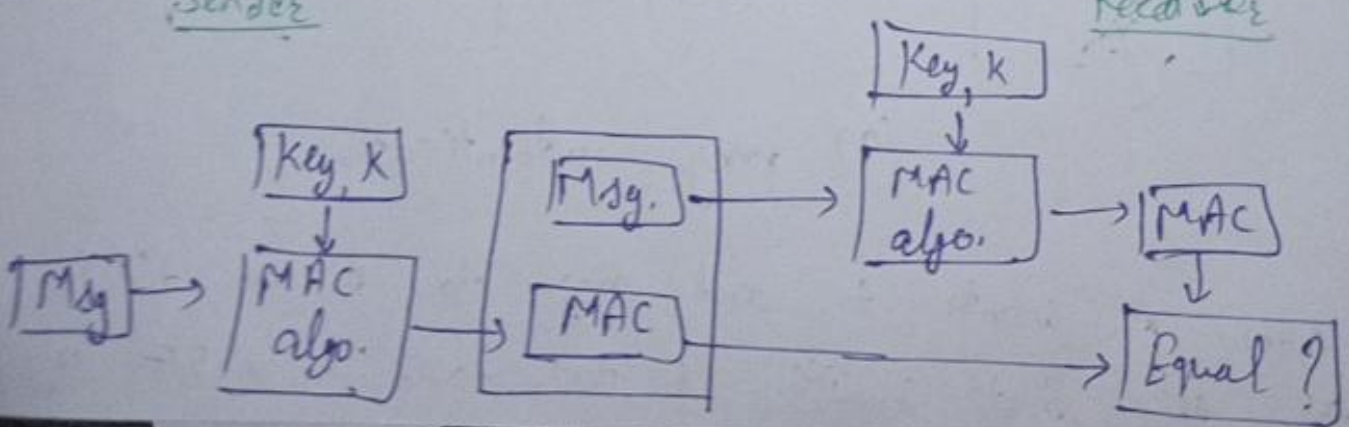
- SHA-1 was cracked in the year 2005. New hash fun" SHA-512 is introduced. to overcome problem of SHA-1.

| Parameters | SHA-1 | SHA-256 | SHA-384 | SHA-512 |
|---|---|---|---|---|
| 1) Msg digest size. | 160 | 256 | 384 | 512 |
| 2) Msg size | $< 2^{64}$ | $< 2^{64}$ | $< 2^{128}$ | $< 2^{128}$ |
| 3) Block size | 512 | 512 | 1024 | 1024 |
| 4) word size | 32 | 32 | 64 | 64 |
| 5) No. of steps | 80 | 64 | 80 | 80 |
| 6) Security level | 80 | 128 | 192 | 256 |

SHA-512 is more secure in compare of others.

## * Msg. Authentication Code (MAC) :

- Msg. authentication is a mechanism or service used to verify the integrity of a msg. Msg. authentication guarantees that the sender of the msg. is authentic.
- A MAC algo, sometimes called a keyed hash fun$^n$ accepts as input a secret key an arbitrary-length msg. to be authenticated, and outputs a MAC. (with the help of key we achieve the authentication)
- MAC value protects both a msg's data integrity as well as its authenticity, by allowing verifies to detect any changes.
- the MAC is a small size fixed-size block of data that is generated based on a msg M of variable length using secret key K as follows.



Before sending a msg., Sender generate MAC with the help of MAC algo. and key K; and MAC is appended with the msg. and send to the receiver.

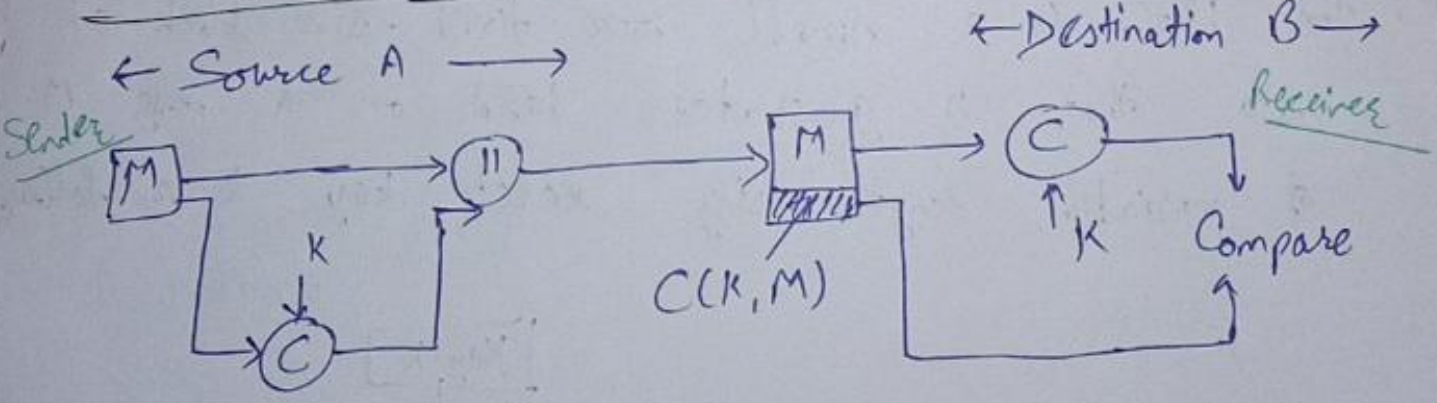
At receiver side both MAC and Msg are seperate and generate MAC from the original msg. with the help of MAC algo & key, K.

If appended MAC and generated MAC both are same it means there is no modification in msg., and the msg. is authenticate.

- MAC is also called cryptographic checksum.

$$MAC = C(K, M)$$

## # Applications of MAC:




(a) msg. authentication

- Sender A wants to send msg. to B via MAC, then the first condition is to share a secret key K.

Sender A sends a msg M to receiver B. Here for the cryptography checksum 'C' generation at that time key K is used for the source and the destination, that mean same key is shared by source and destination.

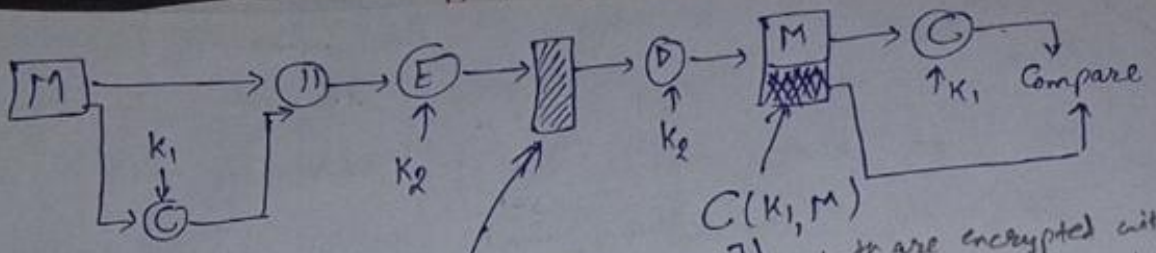
1) Sender A calculates the MAC from msg. M by applying K. Append MAC to the msg. M.

2) A Sends the original msg. M and the MAC to B.

3) When B receives msg, B also uses K to calculate own MAC over msg M.

4) B now compares MAC1 and MAC2. If both are same, B assures that msg is not altered during transmission. If it is not matched, it means B can reject the msg.

(Here integrity is acheived & with the help of K authentication is also acheived).




(b) Msg. authentication & confidentiality ; authentication tied to plaintext

- Sender A want to send msg to B via MAC, then the first condition is to share a secret key $K_1$.

1) Sender A calculates the MAC from msg. M by applying $K_1$. Append MAC to the msg M.

2) Encrypt msg. and appended MAC using key $K_2$. A sends the generate cipher text (Msg + MAC) to B.

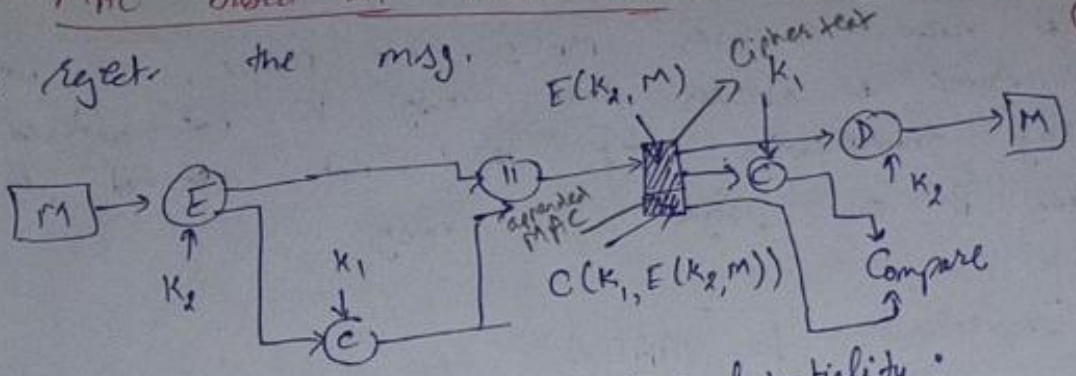
3) When B receives cipher text, decrypt using key $K_2$. B also uses $K_1$ to calculate own MAC over msg. M.

4) B now compares MAC1 and MAC2. If both are same, B assures that msg. is not altered during transmission. And if it is not matched then

4) When ... own MAC over msg. ... get original msg.

## MAC based on Hash Function

B reject the msg.




(c) Msg. authentication and confidentiality ; authentication tied to Cipher text

- Sender A wants to send msg. to B via MAC, then the first condition is to share a secret key $K_1$.

1) Sender A encrypt the msg and generate Cipher text using key $K_2$. (MAC is generated from Cipher text not from original text. So we say authentication tied with the Cipher text)

2) Sender A calculates the MAC from Cipher text by applying $K_1$. then append MAC to cipher text.
(Receiver will remove this Cipher text and MAC. Receiver will generate a MAC from the cipher text and compare with the append MAC and decrypt the msg. M.

3) A sends appended Cipher text plus MAC to B.

4) When B receives msg, B also uses $K_1$ to calculate own MAC over msg. M. Decrypt cipher text using $K_2$ and get original msg.

(M MAC).

5) B now compares MAC1 and MAC2. If both are same, B assures that msg. is not altered during transmission. If it is not matched, it means B can reject the msg.

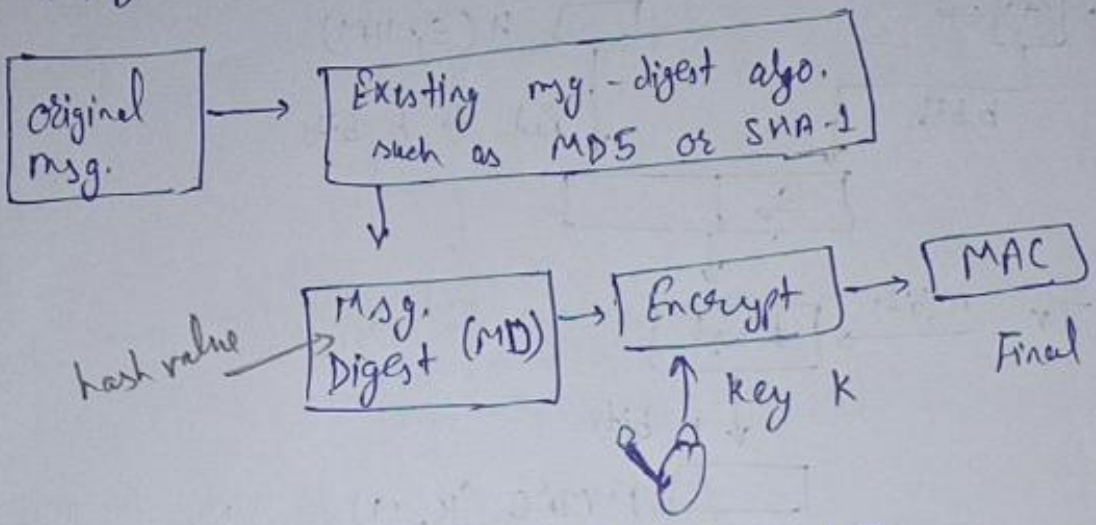
## # Importance of MAC:

- MAC ensures that only receiver can identify the original msg. Even if attacker modifies the msg M, but cannot modify MAC. In case of MAC is modified by attacker, receiver's calculations of MAC will differ from it. (It means receiver can not accept that msg.)
- MAC cannot modify easily because MAC is encrypted by secret key K, but key is only known by sender and receiver only.
- MAC is one-way fun". (It means if msg. is given we generate MAC but if MAC is given we cannot generate a msg.)
- MAC provides data integrity and authentication both. Also apply encryption and achieve Confidentiality.

## ✳ MAC based on Hash Function (HMAC):

### # HMAC Concept:

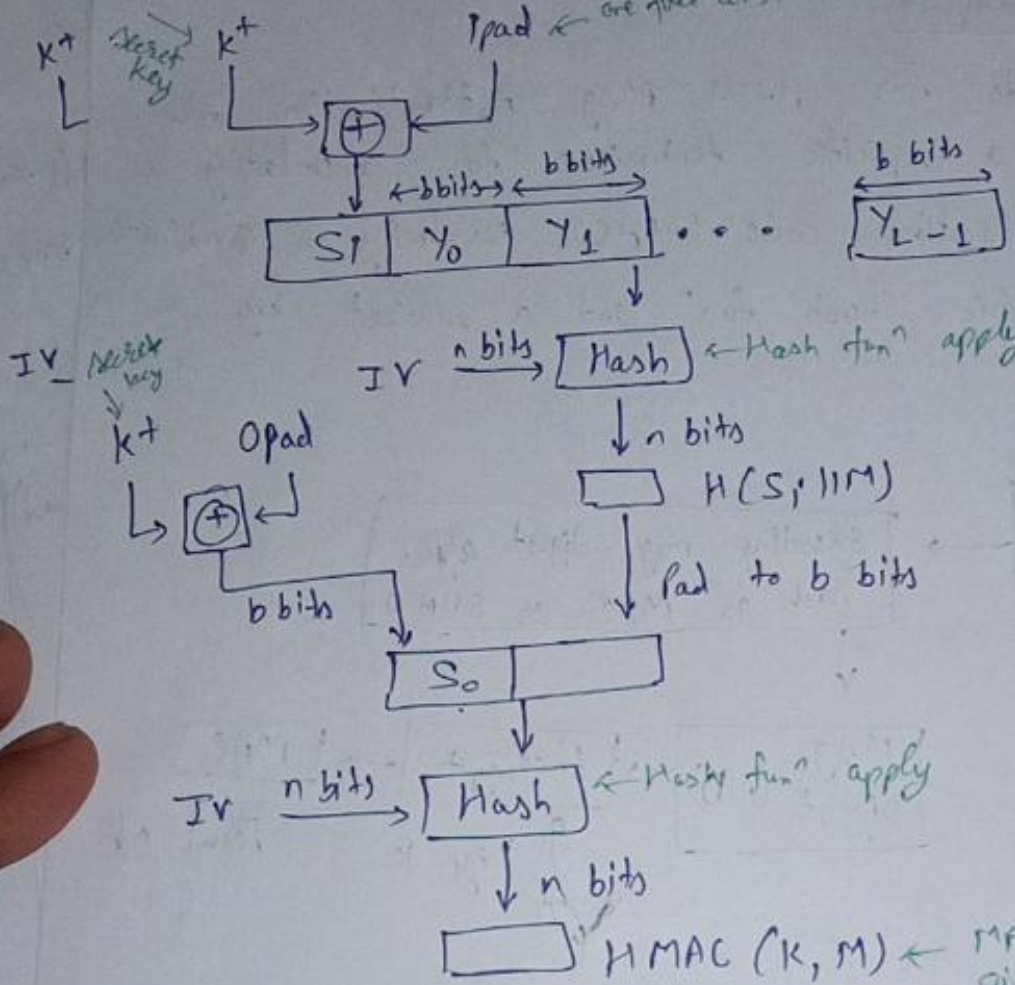
- HMAC stands for HASH Msg. Authentication Code (HMAC) is a specific technique for calculating a msg. authentication code (MAC) involving a combination of cryptographic hash fun$^n$ and a secret key Cryptography.




[ Block diagram of HMAC Concept ]

- In HMAC Concept we acheive integrity as well as authentication.

# HMAC Structure & Implementation:




[ HMAC Structure ]

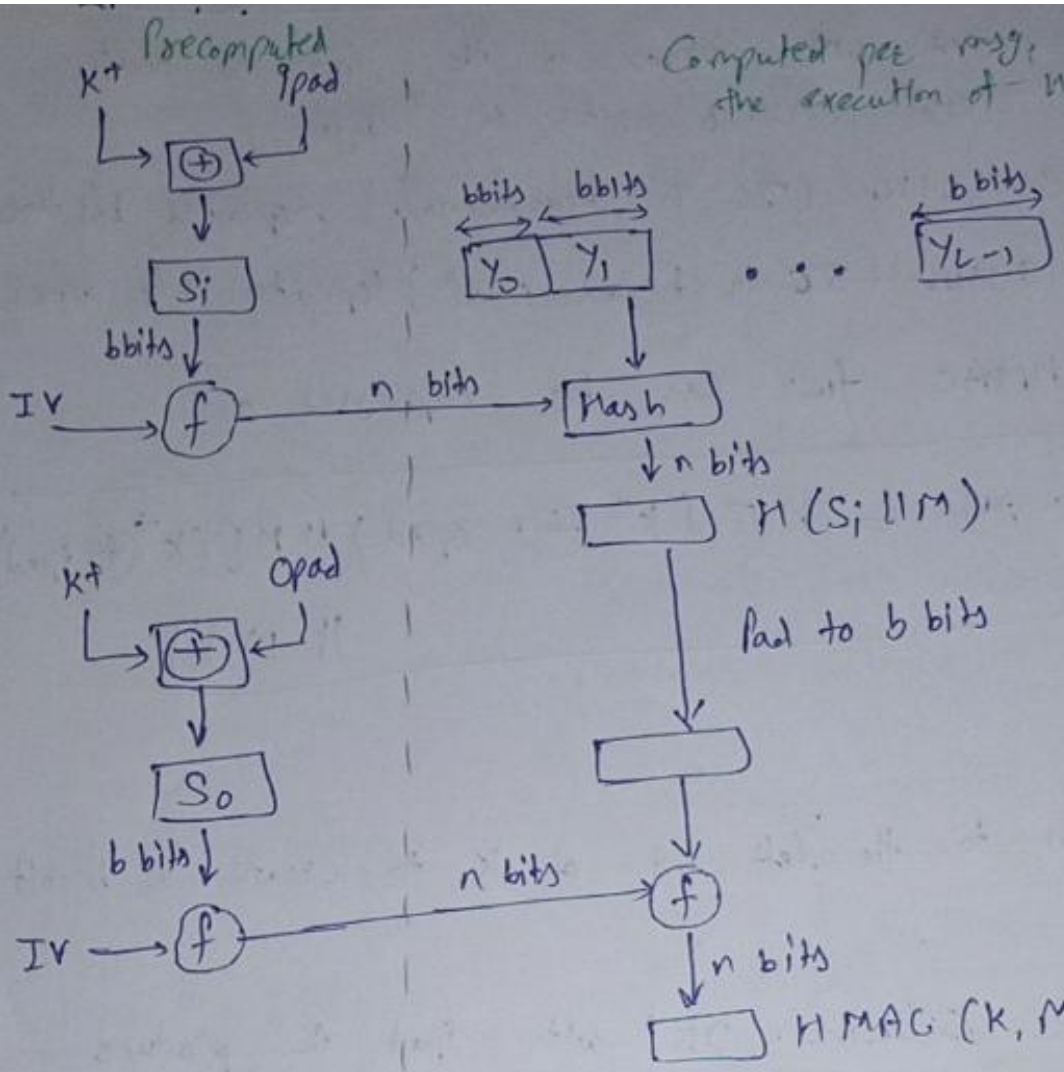
## Notations for both the diagram:

H = Embedded hash fun^ (eg. MD5, SHA-1, RIPEMD-160)

IV = Initial value input to hash fun^

M = msg. input to HMAC (including the padding specified in the embedded hash fun^)

$Y_i$ = $i^{th}$ block of M, $0 \leq i \leq (L-1)$




[ HMAC Implementation ]

L = no. of blocks in M

b = no. of bits in a block

n = length of hash code produced by embedded hash fun^n.

K = secret key; recommended length is $\geq n$; if key length is greater than b, the key is input to the hash fun^n to produce an n-bit key

$K^+$ = K padded with zeros on the left so that the result is b bits in length

ipad = 00110110 (36 in hexadecimal) repeated b/8 times

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

The HMAC form can be expressed as:

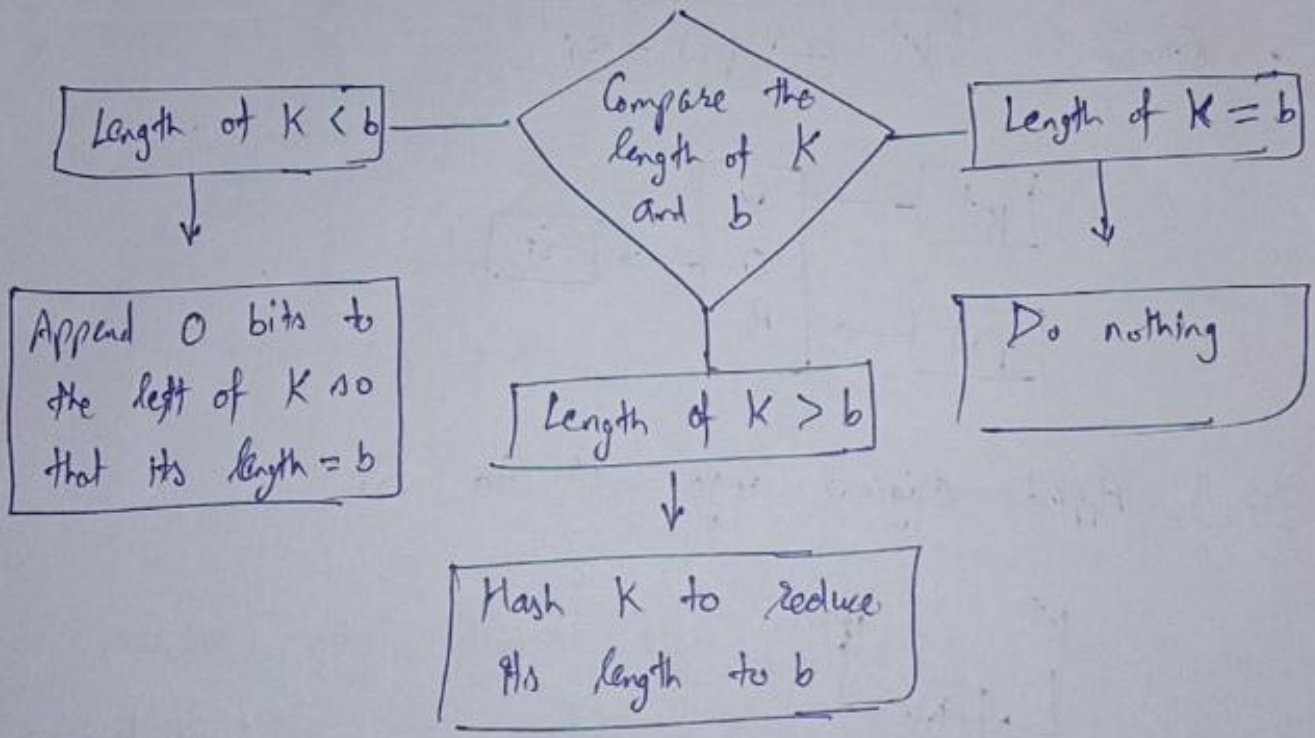
$$HMAC(K, M) = H[(K^+ \oplus opad) \,||\, H[(K^+ \oplus ipad) \,||\, M]].$$

Algorithm:

1) Append zeros to the left end of K to create a b bit string $K^+$.
2) XOR (bitwise exclusive-OR) with ipad to produce the b-bit block Si.
3) Append M to Si.
4) Apply H to the stream generated in step 3.
5) XOR $K^+$ with opad to produce the b-bit block So.
6) Append the hash result from step 4 to So.
7) Apply H to the stream generated in step 6 and output the result.

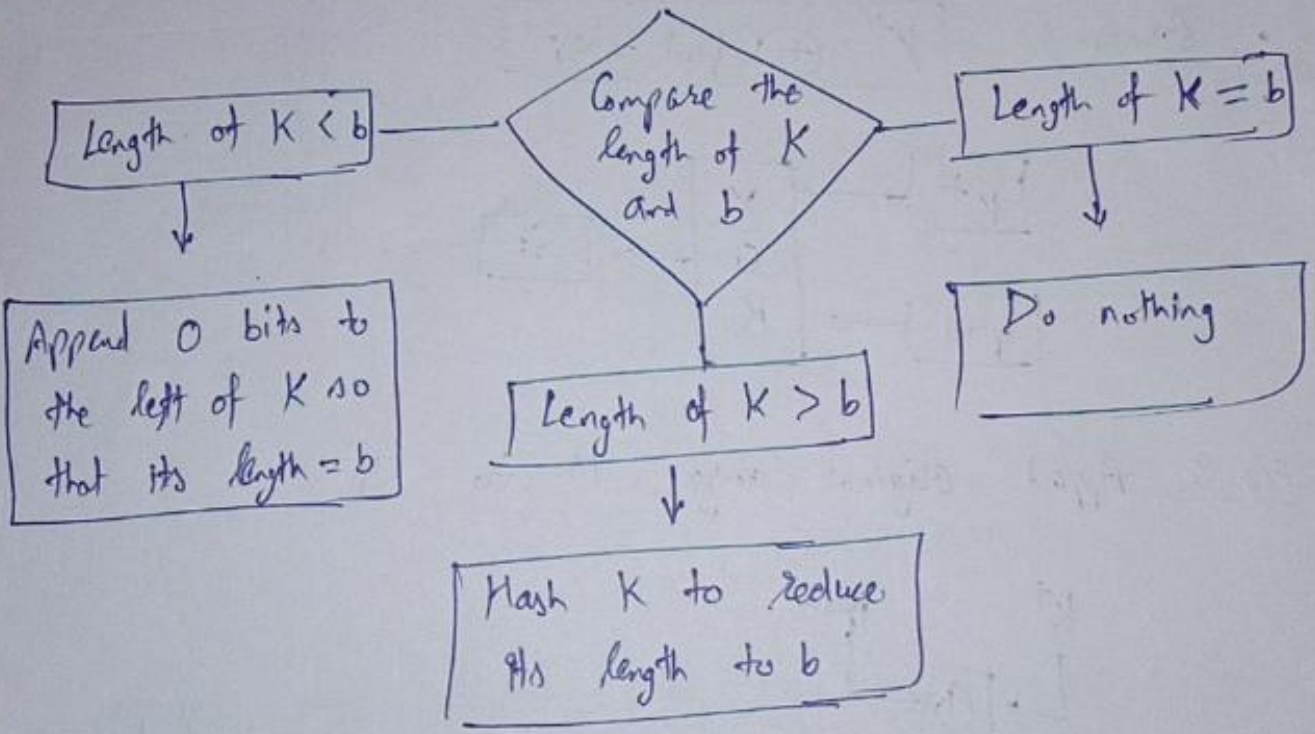
Step 1: Make the length of $K^+$ equal to b.

- If length of $K^+ < b$: add 0 bit as required to the left of K.
- If length of $K^+ = b$: we do not take any action, & proceed to step 2.
- If length of $K^+ > b$: we need to trim K, for this, we pass K through the msg.-digest algo. (H) selected for this particular instance of HMAC.

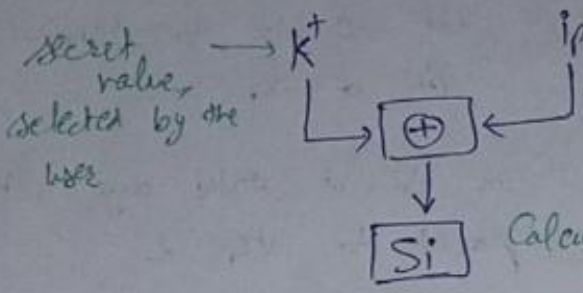


**Step 1**: Make the length of $K^+$ equal to b.

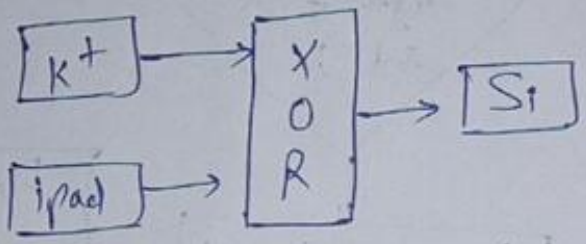
- if length of $K^+ < b$ : add 0 bit as required to the left of K.
- if length of $K^+ = b$ : we do not take any action, & proceed to step 2.
- if length of $K^+ > b$ : we need to trim K, for this, we pass K through the msg. digest algo. (H) selected for this particular instance of HMAC.



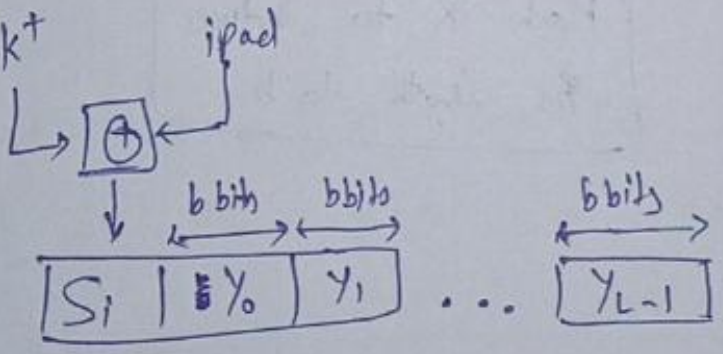
Step 2: XOR $K^+$ with ipad to produce Si.




- XOR $K^+$ (the o/p of step 1) and ipad to produce a variable called Si.
- Here ipad = 00110110 (36 in hexadecimal) repeated b/8 times.
- Equation: $K^+ \oplus ipad = Si$




Step 3: Append original msg. M to Si

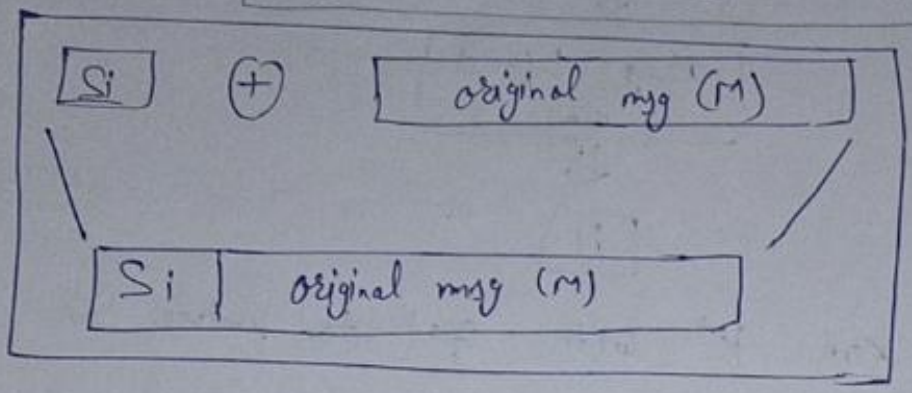



- Take the original msg (M) and simply append it to the end of Si.

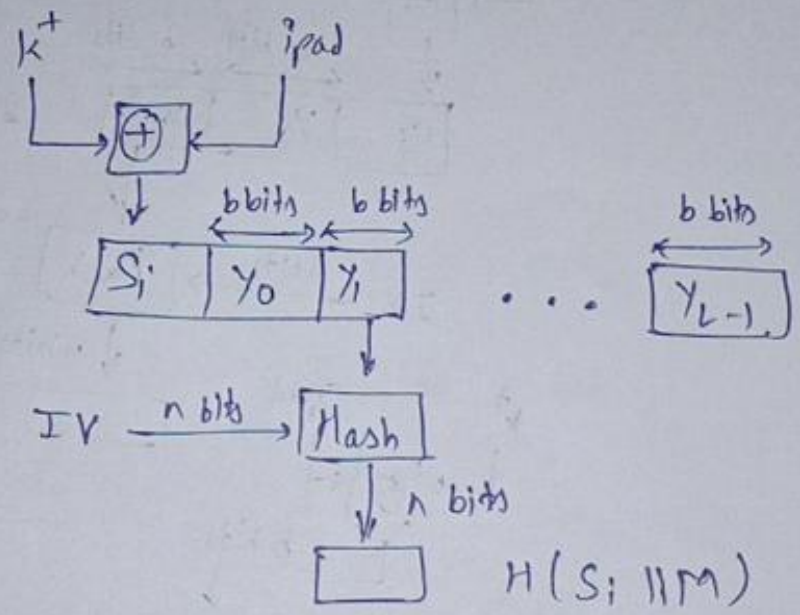
Original msg. is divided into fixed size of block & these block append with secret value Si.

- Equation:

$$[(K^+ \oplus ipad) \,||\, M] = S_i \,||\, M$$




Step 4: Apply msg - digest algo.




- the selected msg. digest algo (eg. MD5, SHA-1, etc.) is applied to the output of step 3.

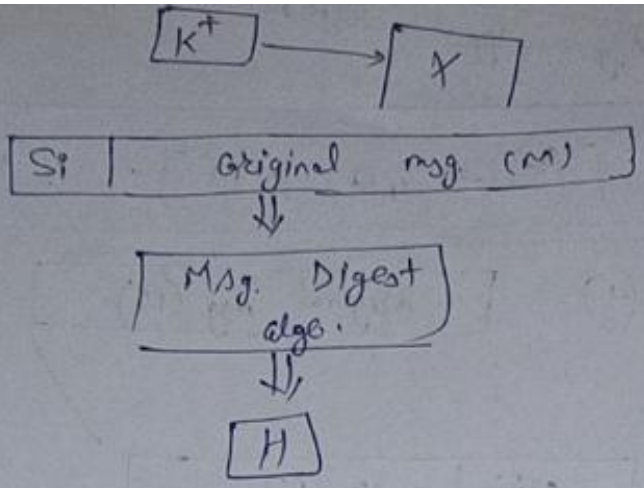
- Equation:

$$H[(K^+ \oplus ipad) \,||\, M] = H(S_i \,||\, M)$$

Hash value of this appended with msg

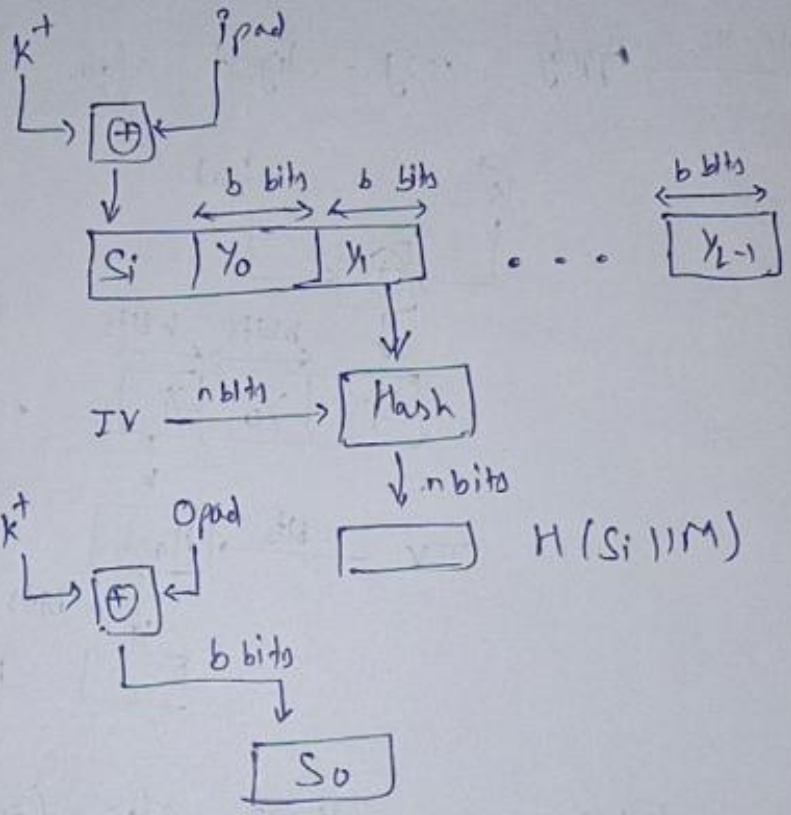
also we can write it

• The



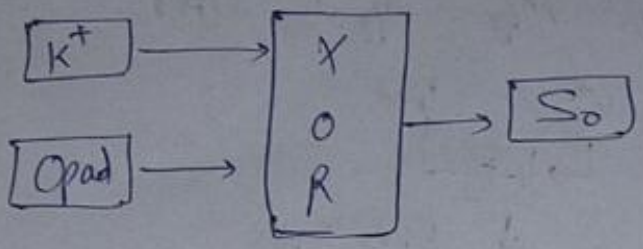



Step 5: XOR K+ with opad to produce So

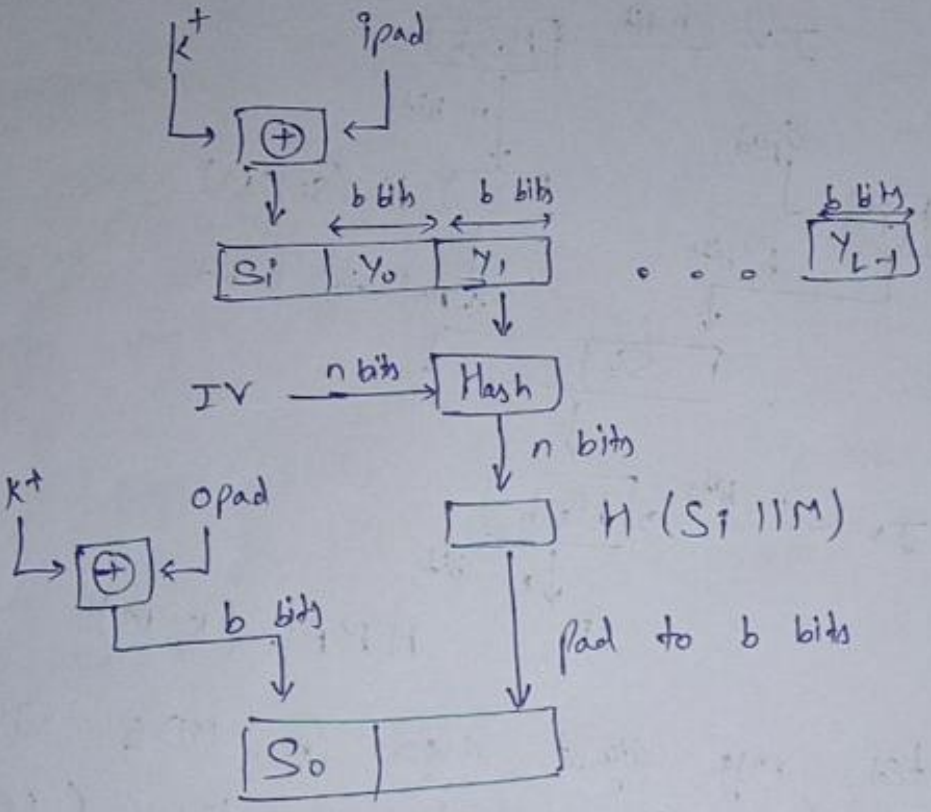



- XOR K+ (the o/p of step 1) with opad to produce a variable called as So.
- Here opad = {00111100 (5C in Hexadeimal) repeated b/8 times. (given in algo.
- Equation: $K^+ \oplus opad = So$

Advantage :

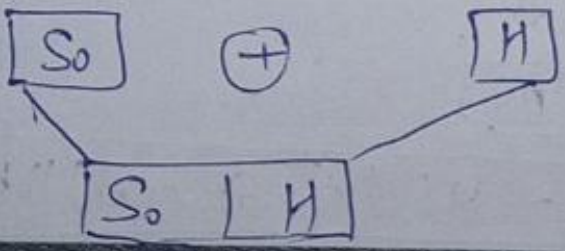



Step 6: Append H to So

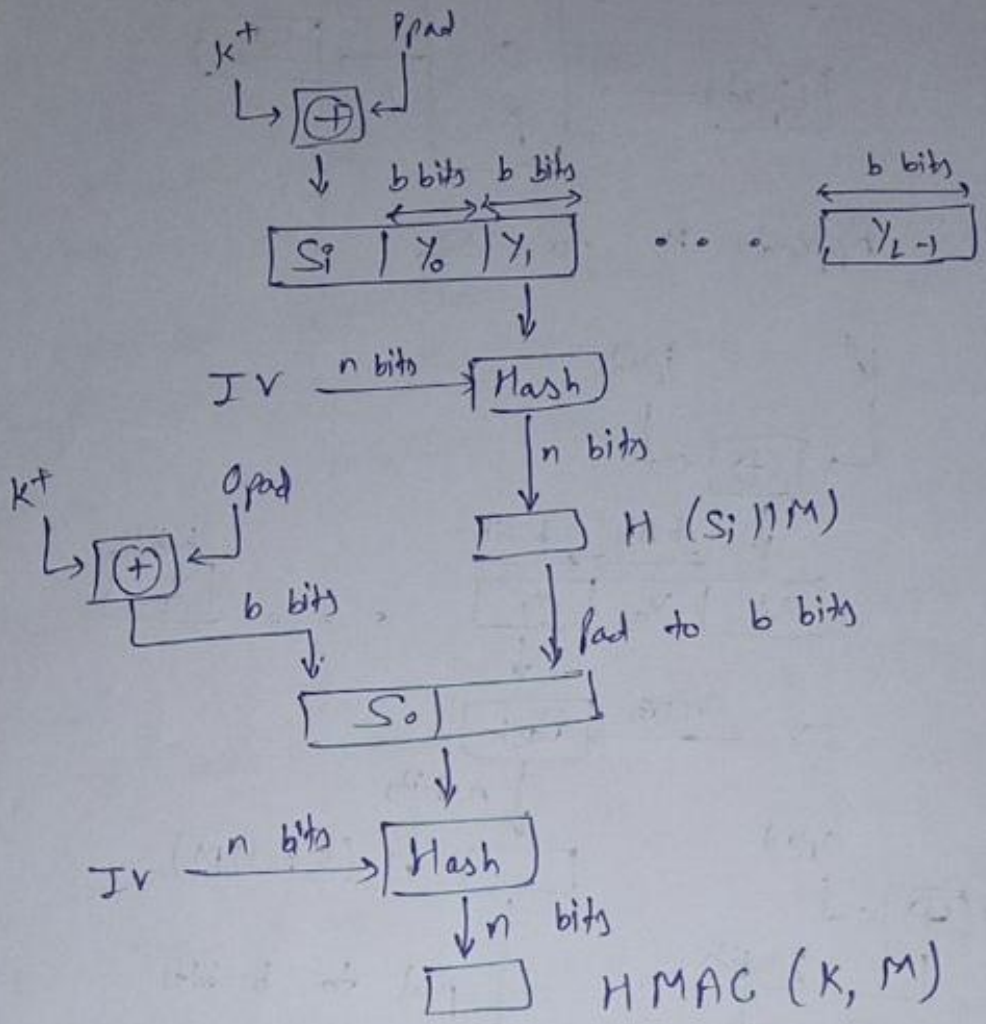



- Append the msg digest calculated in step 4 to the end of So.
- Equation:

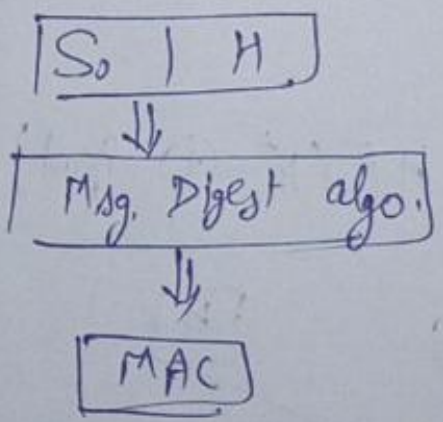
$$(K^+ \oplus opad) \,||\, H[(K^+ \oplus ipad) \,||\, M] = So \,||\, H(Si \,||\, M)$$



Step 7: Apply Msg. - digest - algo.




The selected msg. digest algo. (eg. MD5, SHA-I, etc.) is applied to output of step 6 (i.e. to the Concatenation of So and H). Finally we got MAC.




- Equation:

$$HMAC(K, M) = H[(K^{+} \oplus Opad) \,||\, H[K^{+} \oplus Ipad) \,||\, M]$$

## Advantage :

- HMAC is faster to Compute and verify digital signatures because they use hash fun" rather than public key.
- HMACs can be some cases where the use of public key cryptography is prohibited.
- HMACs are much smaller than digital Signatures.

## Disadvantage :

- Key exchange is main issue, so can't prevent against replay of msg. attack. Or man-in-middle attack.
- HMAC can not be used if the no. of receivers is greater than one.
- If multiple parties share the same symmetric key. How does a receiver know that the msg. was prepared and sent by the sender.

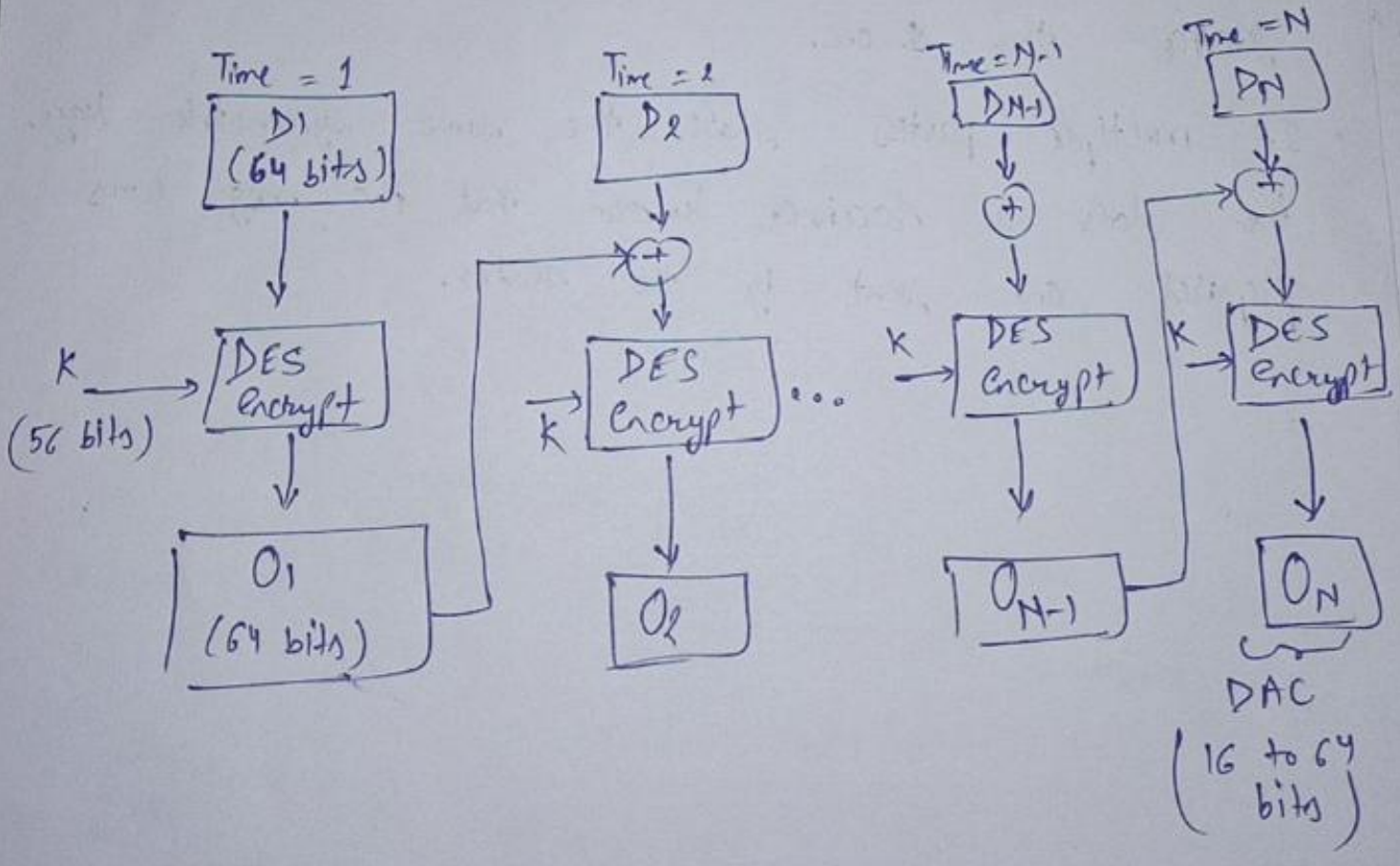
## * MAC based on Block Cipher:

two methods are used in MAC based on Block Cipher.

- DAA (Data Authentication ~~Algo~~ Algo.)
- CMAC (Cipher Based Msg Authentication Code)

## # Data Authentication Algorithm:

- One of the most widely used MACs which is based on block cipher referred to as the Data Authentication Algo.
- the algo. is designed using the Cipher Block Chaining mode of operation of DES.



... AES or triple (89)

- The data (eg. msg, record, file or prog.) to be authenticated are grouped into fixed size 64-bit blocks: $D_1, D_2, \ldots D_N$ (all blocks are 64 bits and we can say in the last block are can padded some extra bits to make it 64 bits fixed size block).

(88) (90)

hanged to

- If necessary, the final block is padded on the right with zeros to make a 64 bit blocks. Using the DES encryption algo. and a secret key, a data authentication code is calculated as follow:

$$O_1 = E(K, D_1)$$
$$O_2 = E(K, [D_2 \oplus O_1])$$
$$O_3 = E(K, [D_3 \oplus O_2])$$
$$\vdots$$
$$O_N = E(K, [D_N \oplus O_{N-1}])$$

- The DAC consists of either the entire block $O_N$ or the leftmost M bits of the block, with $16 \leq M \leq 64$.

For overcome last method's ~~pro~~ Drawback of DAA, we will use CMAC.

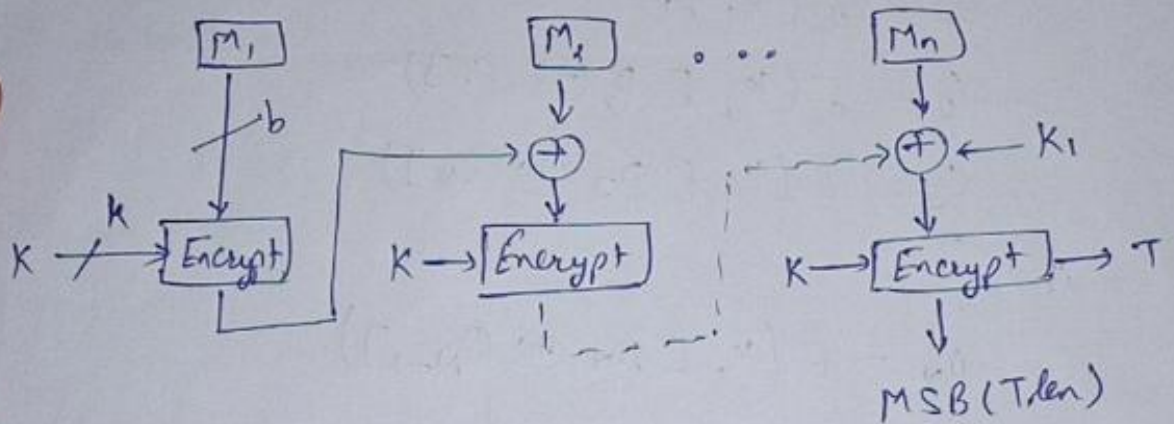
pro...

# Cipher Based Msg Authentication Code (CMAC):

→ • DAA, which is now obsolete (which is not used now a days.). Then CMAC, which is designed to overcome the deficiencies of DAA.

→ • CMAC mode of operation for use with AES and triple DES. (it gives more security in comparison of DAA)

• First, let us define the operation of CMAC when the msg. is an integer multiple n of the cipher block length b.




in this encryption process we can apply AES or triple DES it means it gives more security.

→ If AES is applied then 128 bit, 192 or 256 bit block is divided.

→ If triple DES applied then 64 bit block is used.

→ msg block is divided into fixed size block as per the given algo. like AES or triple DES.

variable b is used, because of AES or triple DES.

→ Encryption key is applied, k is variable because it may be AES or triple DES.

→ if AES is used the key size is 128 bits

→ if triple DES is used the 64 bit 3 times key is applied.

- For AES, b = 128 and for triple DES, b = 64.
- The msg. is divided into n blocks $(M_1, M_2, \ldots M_n)$.
- For AES, the key size is 128, 192 or 256 bits; for triple DES, the key size is 112 or 168 bits.
- CMAC is calculated as follow:

$$C_1 = E(K, M_1)$$
$$C_2 = E(K, [M_2 \oplus C_1])$$
$$C_3 = E(K, [M_3 \oplus C_2])$$
$$\vdots$$
$$C_n = E(K, [M_n \oplus C_{n-1} \oplus K_1])$$
$$T = MSB_{Tlen}(C_n)$$

## * Digital Signature:

# Definition: A digital signature is a mathematical technique used to validate the authenticity and integrity of a msg. or digital document.

- A digital signature is defined the signature generated electronically from the digital computer to ensure the identity of the sender and content of the msg. cannot be modified during transmission process.

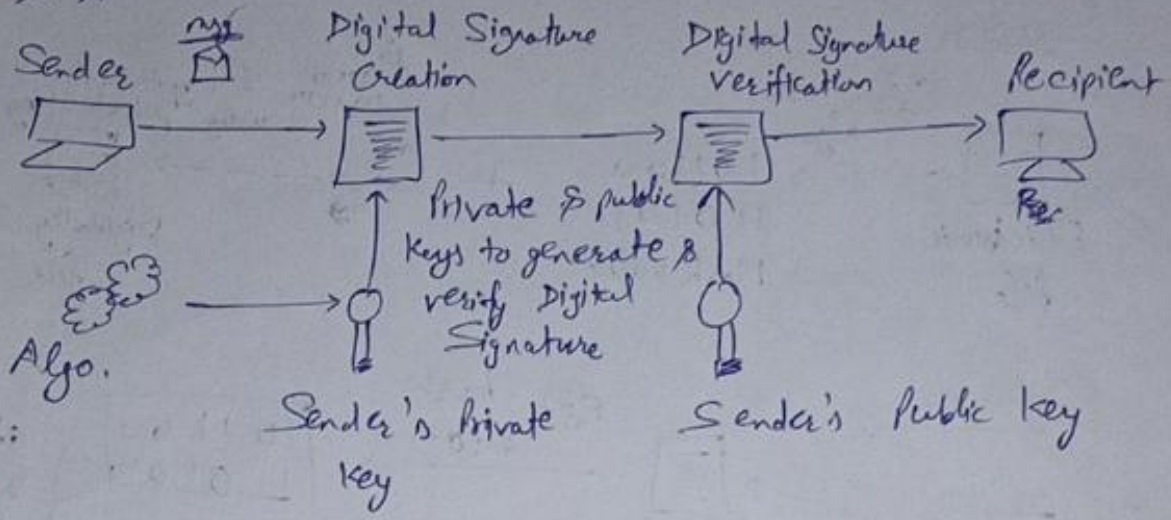
## # Purpose of Digital Signature:

- Concept of digital signature is that sender of a msg uses a signing key (private key) to sign the msg. and send that msg. and its digital signature.
- The receiver uses a verification key (Public key) of the sender only to verify the origin of the msg. and make sure it has not been tempered with while in transmission.
- Digital signature techniques acheive the "authenticity" and "Integrity" of the data over internet.

produce a signature.

6. Last apply H to the stream generated in step 5 and output the result. (90)

Therefore,




# Process of Digital Signature:

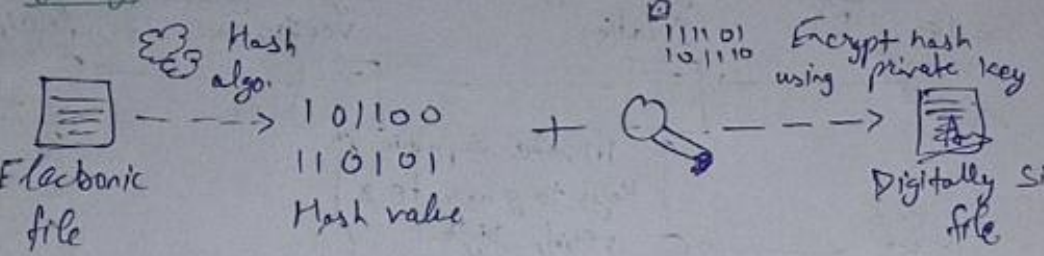
- Hash value of a msg. when encrypted with the private key of a user is, his digital signature on that e-document. Digital signature is an example of asymmetric key cryptography which uses three diff. algo. to complete the process.

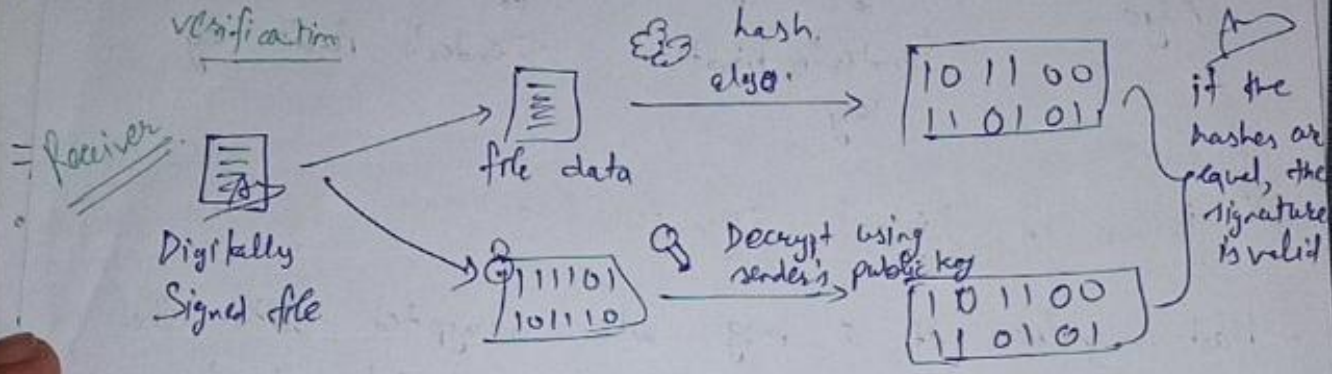
1) First step is key generation algo. which generates private key and a corresponding public key.

2) Next step is signing algo. which selects sending msg. and a private key generated in step 1, to produce a signature.

3) third step is signature verifying algo. which verify the authenticity of sending msg. and public key.

2) Signing




# Properties of Digital Signature:

- In situations where, there is no complete trust b/w sender and receiver, sth more than authentication is needed. The most attractive solution to this prob. is the digital signature. The digital signature must have the following properties;

1) It must verify the author and the date and time of signature.

6. And apply H to the stream generated in step 6 and output the result. (93) (91)

2) It must authenticate the contents at the time of signature.

3) It must be verifiable by third parties, to resolve disputes.

- Thus, the digital signature fun" includes the authentication fun".

# Advantages:

- Authentication: identification of person that signs
- Integrity of data: Every change will be detected.
- Non repudiation: Author can not be denied of his work.
- Imposter Prevention: Elimination of possibility of committing fraud by an imposter.

# Disadvantages:

- Expiry: In fast technology senerio, many of these tech products have a short life.
- Certificates: Both sender & receiver may have to buy digital certificates.
- Software: Seider & receiver have to ~~by~~ buy verificat" software or pay to third party for verification.

# Digital Signature Requirements:

1) The digital signature must be a bit pattern that depends on the msg. being signed.
2) The signature must use some unique info. of the sender to prevent both forgery and denial.
3) It must be relatively easy to produce the digital signature.
4) It must be relatively easy to recognize and verify the digital signature.
5) It must be computationally infeasible to forge a digital signature, either by constructing a new msg. for an existing digital signature or by constructing a fraud digital signature for a given msg.
6) It must be practical to retain a copy of the digital signature in storage.

# Digital Signature Security:

- Msg. Authentication: A digital signature technique can provide msg authentication. Digital signature is used to establish proof of identities and ensure that the origin of an electronic msg. is correctly identified.

- Msg Integrity: Digital signature are used to detect unauthorized modification to data which assures that the contents of msg. are not changed after sender sends but before it reaches to intended receiver.

- Non-Repudiation: There are situation where a user sends a msg. and after on refuses that he had sent that msg. That is known as non-repudiation because the person who signed the document cannot repudiate the signature at a later time.

- We can prevent man in the middle attack, Replay attack, Masquerade, Impersonation attack.

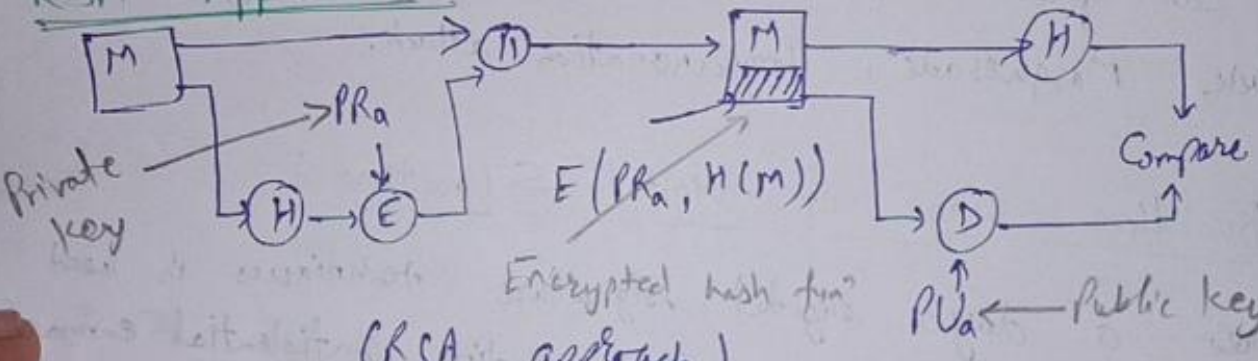
# Realtime usage of Digital Signature:

- Now a day's digital signature techniques is used in many app areas like sending confidential e-mail, during secure payment transfer and possibly all Sw companies, universities, educational institutions those want to achieve authentication and integrity of their confidential info.

- The national Institute of standards and Technol (NIST) has published Federal Information Processing Standard FIPS 186, known as DSS.

- The DSS makes use of SHA and presents a new DS technique, the DSA.
- Latest version also incorporates DSA based on RSA and on elliptic curve cryptography.

RSA Approach:




(RSA approach)

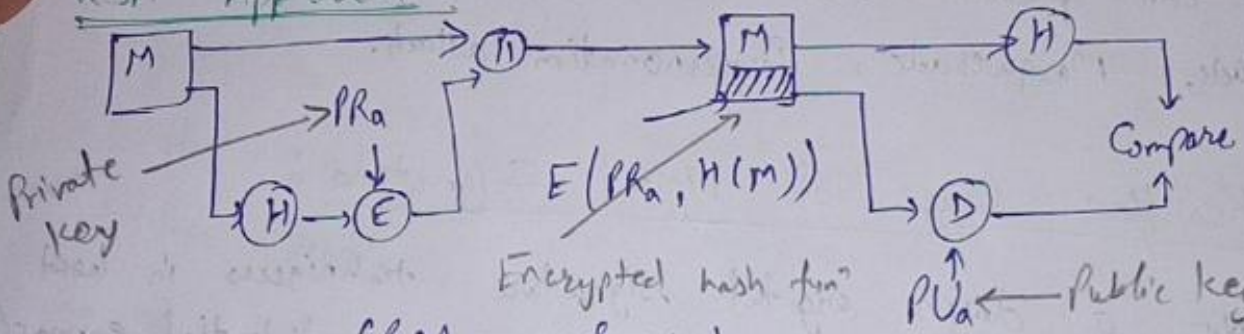
- the msg to be signed is input to a hash fun that produces a secure hash code to fixed length.
- this hash code is then encrypted using the sender's private key to form the signature.
- Both the msg and signature are then transmitted.
- Receiver takes the msg and produces a hash code.
- Receiver also decrypts the signature using the sender's public key.
- signature is input to a

## * Digital Signature Algo.:

### # Digital Signature Standard (DSS):

- The national Institute of standards and Technology (NIST) has published Federal Information Processing Standard FIPS 186, known as DSS.
- The DSS makes use of SHA and presents a new DS technique, the DSA.
- Latest version also incorporates DSA based on RSA and on elliptic curve cryptography.

RSA Approach:



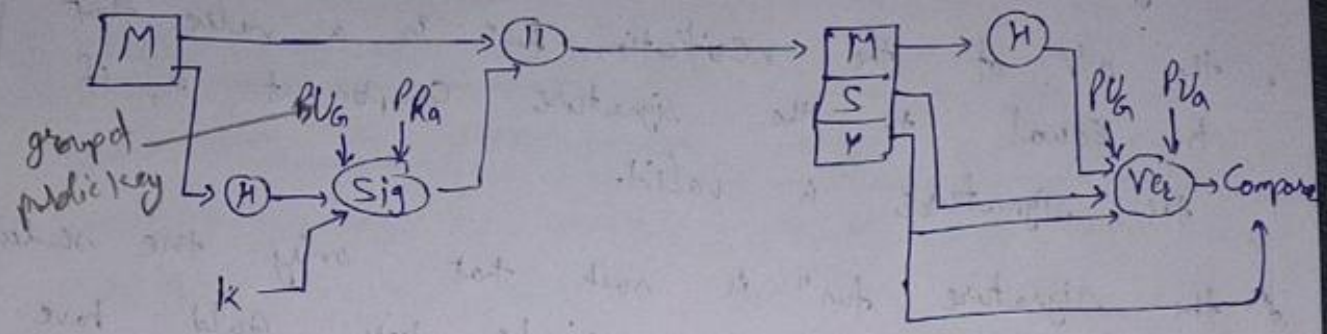

(RSA approach)

- the msg to be signed is input to a hash fun that produces a secure hash code to fixed length.
- this hash code is then encrypted using the sender's private key to form the signature.
- Both the msg. and signature are then transmitted.
- Receiver takes the msg. and produces a hash code.
- Receiver also decrypts the signature using the sender's public key.

signature is input to a verifi... incoming

- If the calculated hash code matches the decrypted signature, the signature is accepts as valid.

## DSS Approach :




- It is also makes use of a hash fun$^n$.
- The hash code is provided as input to a signature fun$^n$ along with a random no. k, generated for this particular signature.
- The signature fun$^n$ also depends on the sender's private key (PRa), and also a set of parameters known to a group of communicating principle.
- Consider this set to constitute a global public key (PUG).
- The result is a signature, consisting of two components, labeled s and r.
- At the receiver end, the hash code of the incoming msg is generated.
- Signature is input to a verification fun$^n$.

- the verification fun" also depends on the global public key as well as the sender's public key (PUa), which is paired with the sender's private key.
- the o/p of the verification fun" is a value that is equal to the signature component r, if the signature is valid.
- the signature fun" is such that only the sender, with knowledge of the private key, could have produced the valid signature.

## # Digital Signature Algo. (DSA):

### Key Generation Process:

Global Public key Components:

p: prime no. where $2^{L-1} < p < 2^L$ for $512 \leq L \leq 1024$ and L a multiple of 64; i.e. bit length of b/w 512 and 1024 bits in increments of 64 bits

q: prime divisor of (p-1), where $2^{159} < q < 2^{160}$; ie. bit length of 160 bits

g: $h^{(p-1)/q} \bmod p$; where h is any integer with $1 < h < (p-1)$ such that $h^{(p-1)/q} \bmod p > 1$

(Meaning of global key component is in the nw all the users are connected with each other and the value of p, q, g are the same for all the users.)



User's Private Key:

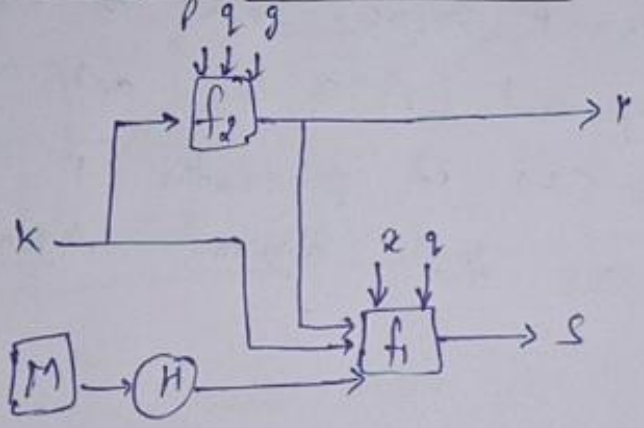
x random or pseudo random integer with $0 < x < q$

(x is called as sender's private key which is selected by a particular user.)

User's Public Key:

$y = g^x \bmod p$

Create Digital Signature:




$S = f_1(H(M), k, x, r, q) = (k^{-1}(H(M) + xr)) \bmod q$

$r = f_2(k, p, q, g) = (g^k \bmod p) \bmod q$

(a) Signing

## User's Per - Msg Secret No:

k = random or pseudo random integer with $0 < k < q$

(• the value of k always diff. whenever your msg. is different or new msg. is arrived.

• from the msg. hash value is generated and output of the hash value is applied to the $f_1$ fun$^n$.

• two diff. component r and s generated. the combination of r and s is the digital signature which is generated from the msg.

• Input of the $f_1$ is hash value (H), per msg. secret no. (k), output of the $f_2$, user's private key (x), global component of the users ($g_1$).

• $f_2$ fun$^n$ generates r value. Input of the $f_2$ fun$^n$ are global component (p, q, g), and user's per msg. secret no. (k), it generates r and S. r and s is our final digital signature.)

## Signing:

$$r = (g^k \bmod p) \bmod q$$

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

$$\text{Signature} = (r, s)$$

## Signature Verification :

### Verifying

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

$u_1 = [H(M')\, w] \bmod q$

$u_2 = (r')\, w \bmod q$

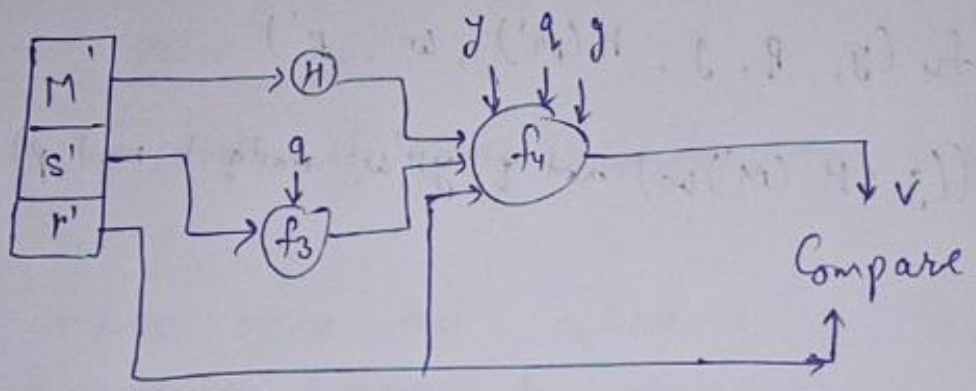
$v = [(g^{u_1}\, y^{u_2}) \bmod p] \bmod q$

TEST : $v = r'$

$M$ = msg. to be signed

$H(M)$ = hash of $M$ using SHA-1

$M'$, $r'$, $s'$ = received version of $M$, $r$, $s$




[• from the $M'$, hash value ($H$) is generated.

• $s'$ is the input of the $f_3$ fun? and $q$ is a global public key component and output of the $f_3$ fun? is $w$. So output of the $f_3$ is the input of the $f_4$.

- Output of the $f_4$ func$^n$. is $V$. $V$ is a value which is compare with the $r'$. If
- If $r'$ and $V$ both are same it means our digital signature is OK. there is no forgery in the msg. signature.
- from the original msg. Hash value ($H$) is generated and input to the $f_4$. now $y$ is the public key of the sender, $q$ and $g$ both are the Global public key component. and another input is $r'$ given to the $f_4$ and generate $V$.

$$w = f_3(s', q) = (s')^{-1} \bmod q$$

$$V = f_4(y, q, g, H(M'), w, r')$$

$$= ((g^{(H(M')w) \bmod q} \; y^{r'w \bmod q}) \bmod p) \bmod q$$

## * Digital Signature Scheme (Elgamal & Schnorr) (16)

- following are widely used digital signature schemes to generate the digital signature:
  1) Elgamal Scheme
  2) Schnorr Scheme
  3) Digital Signature Algorithm (DSA algo). based on Digital Signature Standard (DSS)

## # Elgamal Scheme:

- this scheme is variant of digital signature algo.
- this scheme is based on computing assumption of large prime no.
- It is computationally very complex to compute $S_1$ and $S_2$.
- this scheme assure that authenticity of msg. in sent by sender / signer to verifier.
- As with Elgamal encryption, the global elements of Elgamal digital signature is based on prime no. q and $\alpha$, which is a primitive root of q.

(global elements means all the users which is connected in n/w with other users So here q and $\alpha$ are the same no for all the users).

## Generating private key and public key pair:

1) Generate a random integer $X_A$, such that $1 < X_A < q-1$.
2) Compute $Y_A = \alpha^{X_A} \bmod q$
3) A's private key is $X_A$; A's public key is $\{q, \alpha, Y_A\}$.

for eg: $q = 19$, $\alpha = 10$

1) sender choose $X_A = 16$.
2) Compute $Y_A = \alpha^{X_A} \bmod q = 10^{16} \bmod 19 = 4$. So $Y_A = 4$.
3) Sender's private key is 16; Sender's public key is $\{q, \alpha, Y_A\} = \{19, 10, 4\}$.

3 1)(3)

Sender wants to sign a msg. with hash value $m = 14$.

## Create Digital Signature:

1) Choose a random integer $K$ such that $1 \leq K \leq q-1$ and $\gcd(K, q-1) = 1$. $K$ is relatively prime to $q-1$.
2) Compute $S_1 = \alpha^K \bmod q$.
3) Compute $S_2 = K^{-1}(m - X_A S_1) \bmod (q-1)$.
4) The signature consists of the pair $(S_1, S_2)$.

for eg:

1) Sender chooses $K = 5$, which is relatively prime to $q-1 = 18$.

... the msg. for signature

msg. is

2) $S_1 = \alpha^K \bmod q = 10^5 \bmod 9 = 3.$ $S_1 = 3$

3) $S_2 = K^{-1}(m - X_A S_1) \bmod (q-1) = 11\,(14 - (16)(3)) \bmod 18 = -374 \bmod 18 = 4.$ $S_2 = 4$

## Signature verification:

1) Calculate $V_1 = \alpha^m \bmod q$

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

for eg:

1) $V_1 = \alpha^m \bmod q = 10^{14} \bmod 19 = 16$ $V_1 = 16$

2) $V_2 = (Y_A)^{S_1} (S_1)^{S_2} \bmod q = (4)^3 (3)^4 \bmod 19$

$= 5184 \bmod 19 = 16$ $V_2 = 16$

# Schnorr Scheme:

- the schnorr signature scheme is also based on discrete ~~algo~~. logarithms.
- the schnorr scheme minimizes the msg. dependent amount of computation required to generate a signature. (very less bits or bytes of the msg. is used.)
- the main work for signature generation does not depend on the msg. (Schnorr is faster as compared to Elgamal)

- The scheme is based on using a prime modulus $p$, with having a $(p-1)$ prime factor of $q$ appropriate size - that is, $p = 1 \pmod q$. Typically, we use $p = 2^{1024}$ and $q = 2^{160}$.
  Input Output
- Thus, $p$ is a 1024-bit num, and $q$ is a 160 bit no., which is also the length of the SHA-1 hash value.

## Generating private key and public key pair:

1) Choose primes $p$ and $q$, such that $q$ is a prime factor of $p-1$.
2) Choose an integer $\alpha$, such that $\alpha^q = 1 \bmod p$. The values $\alpha$, $p$ and $q$ comprise a global public key that can be common to a group of users.
3) Choose a random integer $s$ with $0 < s < q$. This is the user's private key.
4) Calculate $v = \alpha^s \bmod p$. This is user's public key.

## Create Digital Signature:

Choose a random integer $r$ with $0 < r < q$ and compute $x = \alpha^r \bmod p$. This computation is a pre-processing stage which independent of the msg.

M to be signed.

2) Concatenate the msg with the hash the result to compute the value:

hash value $\rightarrow e = H(M \| x)$

4) Compute $y = (r + se) \bmod q$. The signature consist of the pair $(e, y)$.

## Signature verification:

1) Compute $x' = \alpha^y v^e \bmod p$

$= \alpha^y \alpha^{-se} \bmod p \quad (\because v = \alpha^{-s} \bmod p)$

$= \alpha^{(y - se)} \bmod p$

$= \alpha^r \bmod p \quad (\because y = r + se)$

$= x$

So Here $x' = x$.

2) Verify that $e = H(M \| x)$

Hence, $H(M \| x') = H(M \| x)$.

M to be signed.

2) Concatenate the msg with the hash the result to compute the value:

hash value $\rightarrow e = H(M \| x)$

4) Compute $y = (r + se) \bmod q$. The signature consist of the pair $(e, y)$.

## Signature verification:

1) Compute $x' = \alpha^y v^e \bmod p$

$= \alpha^y \alpha^{-se} \bmod p \quad (\because v = \alpha^{-s} \bmod p)$

$= \alpha^{(y - se)} \bmod p$

$= \alpha^r \bmod p \quad (\because y = r + se)$

$= x$

So Here $x' = x$.

2) verify that $e = H(M \| x)$

Hence, $H(M \| x') = H(M \| x)$.

## * Comparison of Elgamal and Schnorr

- Elgamal signature scheme is more time consuming in compare to Schnorr scheme.
- Schnorr scheme is 6 times faster than Elgamal and produce signature which is 6 times smaller.