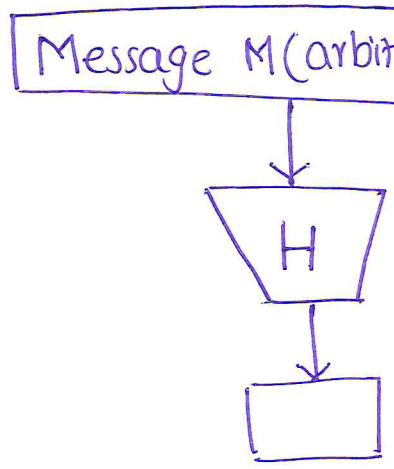
UNIT-4

Cryptographic Hash Functions & Digital Signatures

① Cryptographic Hash Function: It maps a bit string of arbitrary length to a bit string of short, fixed length.



H - Hash Function
h - Hash value

→ The values returned by a hash function are called "Message Digest" or "Hash Values".

→ Features of Hash Functions:

1. Fixed Length output
   * Hash Functions convert data of arbitrary length to a fixed length.
   * The hash values is much smaller than input data, hence hash function is also called "compression function".
   * Hash is smaller representation of larger data, so it is called "Digest".
   * Hash Function with n bit output is referred to as an n-bit hash Function.

2. Efficiency of Operation:

* The computation of $H(x)$ is a fast operation.
* Computationally, hash functions are much faster than a symmetric encryption.

(2) → Properties of Hash Functions:—

1. pre-Image Resistance: It is computationally hard to reverse a hash function.

In other words, if a hash function H produced a hash value 'z', then it should be difficult process to find any input value 'x' that hashes to 'z'.

2. Second pre-Image Resistance: It means that given an input and it's hash, it should be hard to find a different input with the same hash.

In other words, if a hash function H for an input 'x' produces hash value $H(x)$, then it should be difficult to find any other input value y such that $H(y) = H(x)$.

3. Collision Resistance: It means that it should be hard to find two different inputs of any length that result in the same hash.

In other words, For a hash function H, it is hard to find any two different inputs x and y such that $H(x) = H(y)$.

(3) Applications of Cryptographic Hash Functions:

1. Message Authentication:

→ Message authentication is a mechanism used to verify the integrity of a message. It assures that the data received are exactly as sent.

→ When a hash function is used to provide message authentication, then the hash value is referred as "message digest".

→ The Hash Function can be used for message authentication as follows:

1. The sender computes a hash value as a function of the bits in the message and transmits both hash value and the message.
2. The receiver performs the same hash calculation on the message bits and compares this value with incoming hash value.

   If there is a mismatch, the receiver knows that the message have been altered.

→ The Hash Function must be transmitted in a secure fashion.

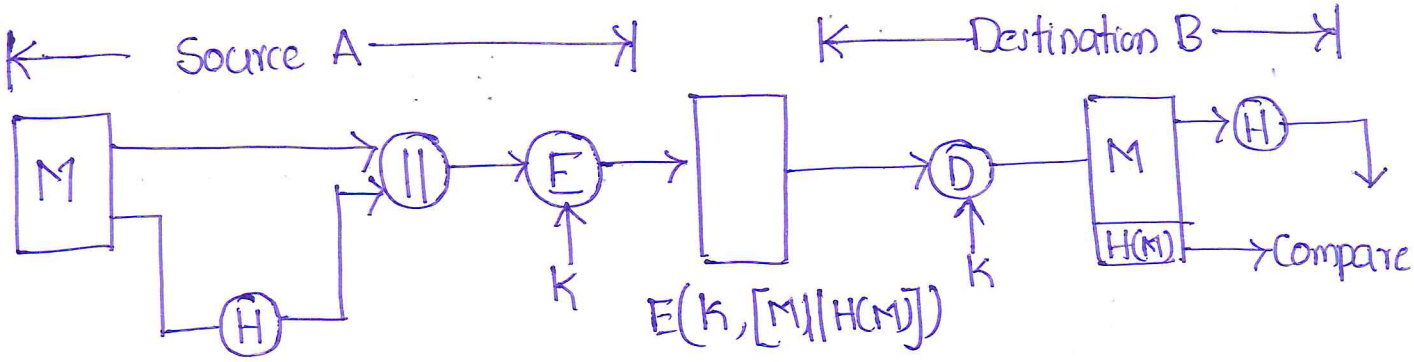


Figure: Hash Function used for Message Authentication

→ Message Authentication is achieved using a "Message Authentication Code (MAC)":

2. Digital Signatures:

→ The operation of the Digital Signature is similar to MAC.

→ In the case of Digital Signature,

The Hash value of the message is encrypted with user's privat key. Anay one who knows the user's public key can verify the integrity of the message associated with Digital Signature

→ The Hashcode is used to provide Digital Signature as follows:

a. The Hashcode is encrypted, using public key encryption with sender's private key. This provides authentication.

b. If confidentiality as well as digital signature is desired, then message plus private key encrypted hash code can be encrypted using symmetric key.

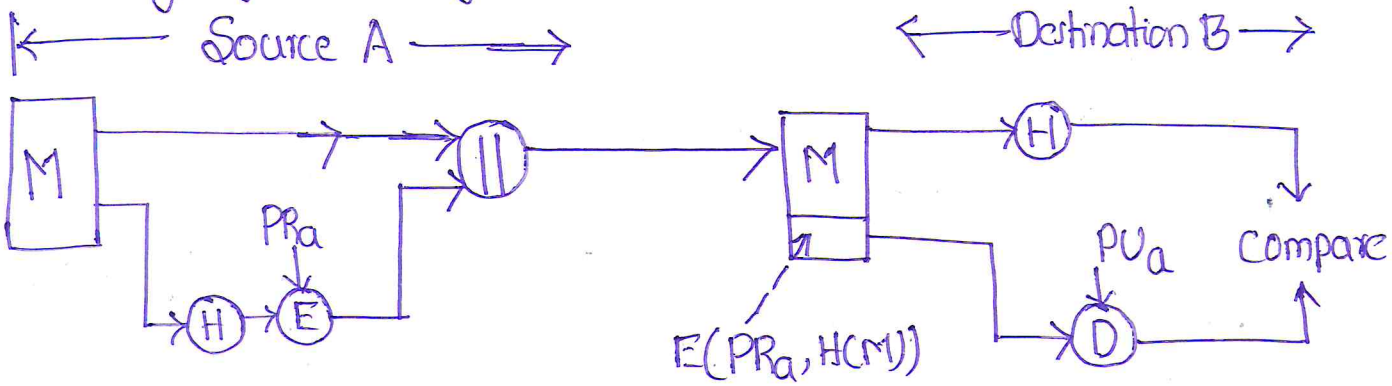


Fig: Hash Function used for Digital Signatures.

3. One-way Password File:

→ one-way password File is a scheme in which a hash of a password is stored by an operating system rather than password itself. Thus the actual password is not retrievable by hacker who gains access to password file.

→ When a user enters a password, the hash of that password is compared to the stored hash value for verification.

4. Intrusion Detection and Virus Detection:

→ Store H(F) for each file on a system and secure the hash values

→ An intruder need to change 'F' without changing H(F).

5. Pseudo Random Number Generator:

→ Hash Based PRNG is used for generation of Symmetric Keys.

④ Security of Hash Functions:-

There are two categories of attacks on hash functions:

a. Brute-Force attacks

b. Cryptanalysis

a. Brute Force Attacks:

→ Brute Force attack does not depend on the specific algorithm but depends only on bit length.

preimage and Second Preimage attacks

→ For this attack, an adversary wishes to find a value 'y' such that H(y) is equal to a given hash value.

→ The Brute force method is to pick values 'y' at random and try each value until a collision occurs.

→ For an m-bit hash value, the adversary would have to try 2^{m-1} valu

Collision Resistant Attacks:

→ For this attack, an adversary wishes to find two messages or data blocks, x and y that yeild same hash function H(x) = H(y).

→ This attack requires considerably less effort than preimage and second preimage attack.

BirthDay Paradox:-

(5) BirthDay Attack:-

→ The Birthday attack is a type of cryptographic attack. It is named because it exploits the "Birthday Paradox".

→ Birthday Paradox: If we chose random variables from a uniform distribution in the range '0' to 'n-1'. Then the probability that a repeated element is encountered exceeds ~~after~~ 0.5 after $\sqrt{n}$ choices.

→ According to Birthday Paradox, For an m-bit hash value, if we pick data blocks at random, we can expect to find two data blocks with same value within $\sqrt{2^m}$ attempts.

→ Strategy to exploit Birthday Paradox in Collision Resistant attack:

1. Source A is prepared to sign by appending the message with approximately m-bit hash code and encrypt with A's private key.
2. Opponent generates $\sqrt{2^m}$ variations of a valid message, all with essentially the same meaning. and also generates $\sqrt{2^m}$ variations of a desired fraudlent message
3. Two sets of messages are compared to find pair with same hash and opponent offers the valid variation for sign.
4. The signature is attached with the Fraudlent variation for transmission
5. Since both have same hash code, they will produce same signature.

(6) Secure Hash Algorithm:— (SHA)

→ SHA was developed by NIST and published as a Federal Information Processing standard (FIPS 180) in 1993.

SHA Versions:

1. SHA-0: SHA-0 has a weakness (Birthday Attack)
2. SHA-1: It produces 160-bit hash value.
3. SHA-2: It defined three versions of SHA with hash value lengths 256, 384, 512 bits, known as SHA-256, SHA-384, SHA-512 respectively.
4. SHA-3: It supports the same hash lengths as SHA-2 and its internal structure differ from rest of SHA family.

	SHA-1	SHA-256	SHA-384	SHA-512
Message Digest size	160	256	384	512
Message size	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block size	512	512	1024	1024
Word size	32	32	64	64
Number of steps	80	64	80	80

Table: Comparision of SHA parameters.

⑦ SHA-512 :-

Specifications:

Message Size : $< 2^{128}$ bits

Message Digest Size : 512 bits

Block size : 1024 bits

SHA-512 Logic:

Step ①: Append padding bits

* The message is padded so that the length is congruent to 896 mod 1024
* The no. of padding bits is in the range of 1 to 1024.
* The padding consists of a single 1 bit followed by necessary number of '0' bits.

Step ②: Append Length

* A block of 128 bits is appended to the message.
* The outcome of first two steps yields a message that is an integer multiple of 1024 bits.
* The expanded message is represented as sequence of 1024 bit blocks $M_1, M_2 \cdots M_n$ so that total length is $N \times 1024$ bits.

Step ③: Initialize Hash Buffer

* 512 buffer is used to hold intermediate and final results of the hash function.

* The bubber can be represented as eighth 64-bit registers (a, b, c, d, e, f, g, h).

a = 6A09E667F3BCC908
b = BB67AE8584CAA73B
c = 3C6EF372FE94F82B
d = A54FF53A5F1D36F1
e = 510E527FADE682D1
f = 9B05688C2B3E6C1F
g = 1F83D9ABFB41BD6B
h = 5BE0CD19137E2179

Step ④: Process message in 1024 blocks

* The heart of the algorithm is a module that consists of 80 rounds.
* Each round takes as input the 512 bit buffer value, abcdefgh and updates the contents of the buffer.
* At input to first round, the buffer has the value of intermedial hash value H_{i-1}
* Each round takes a 64 bit value W_t derived from 1024 bit block(M
* The output of eighteeth round is added to the input to the first round to produce H_i.

Step ⑤: Output

After all N 1024-bit blocks have been processed, the output from Nth stage is 512-bit message digest.

$$H_0 = IV$$

$$H_i = SUM_{64}(H_{i-1}, abcdefgh_i)$$

$$MD = H_N.$$

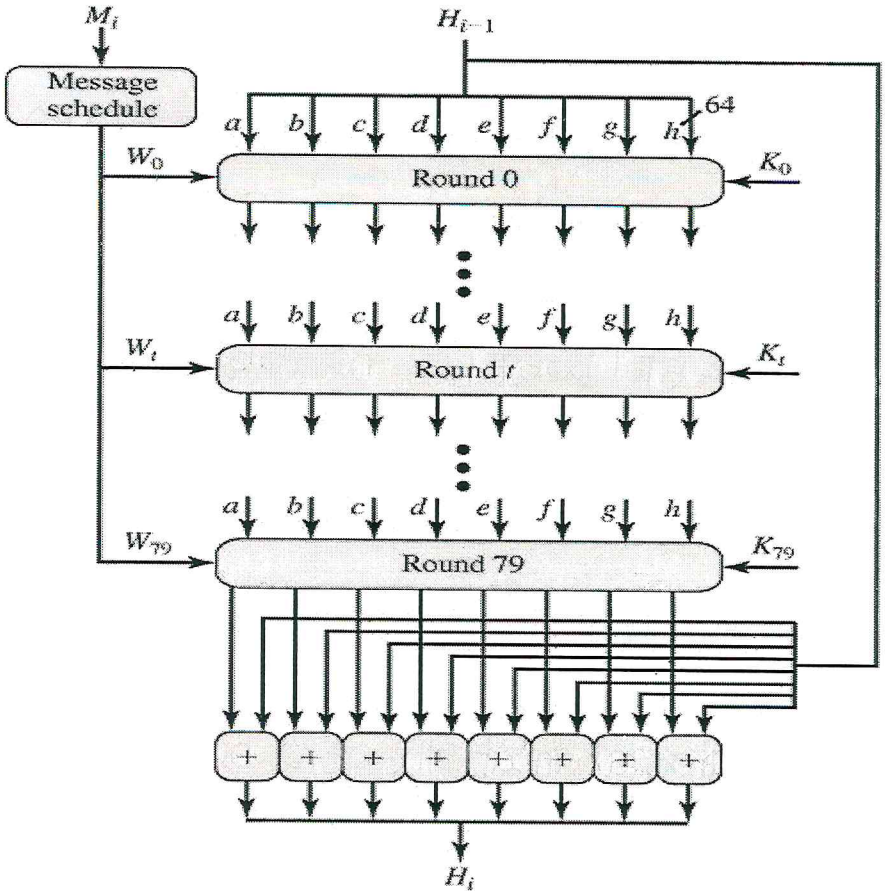


Figure: Processing a single 1024 Bit Block

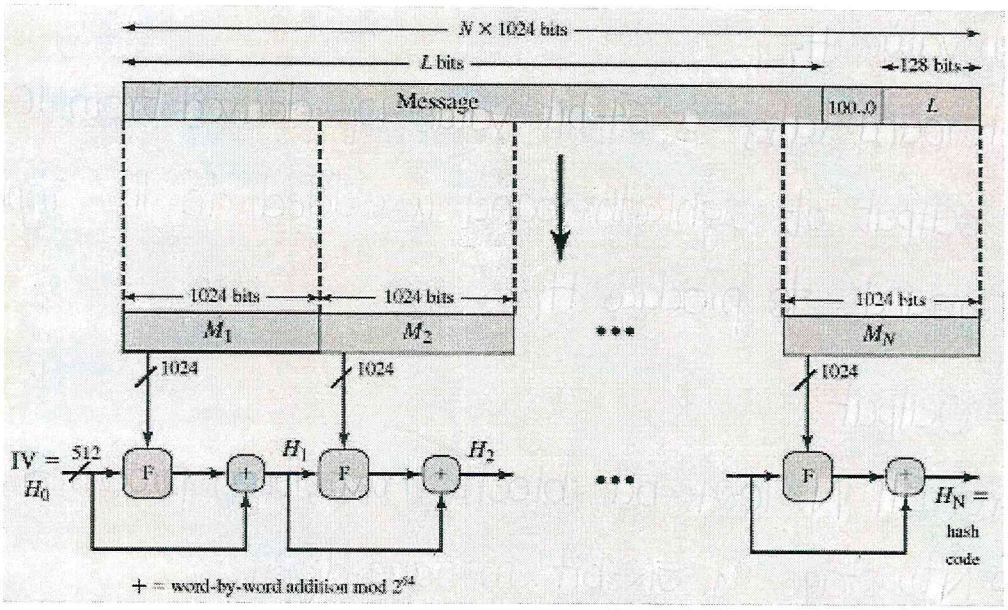


Figure: Message Digest Generation using SHA-512

(8) Message Authentication:-

→ Message authentication is a procedure to verify that

* Received message is from alleged source
* Message has not been altered.
* There is no change in message sequence.
* Message is not delayed or replay.

Message Authentication Functions:-

There are 3 Functions to provide message authentication:

① Hash Function

② Encryption

③ Message Authentication Codes (MAC)

① Hash Functions already discussed in previous sessions

② Encryption:-

→ Message encryption provides a measure of authentication.

→ Symmetric Encryption:

If symmetric encryption is used then,

* Receiver know sender must have created it, since only sender and receiver know the key to be used.
* Receiver know the content cannot be altered.

If message has suitable structure, redundancy or a checksum to detect any changes.

→ Assymetric Encryption:

The assymetric Encryption provides no confidence of sender, since anyone potentially knows public key.

* If the sender signs the message using private key then encrypt with receipients public key. Then it can provide secrecy and authentication.

(9) Message Authentication Code (MAC):

→ MAC also known as Cryptographic Checksum computed using message and some key.

$$MAC = C_K(M)$$

M : Variable length Message

K : Shared key between Sender and Receiver

$C_K(M)$: Fixed length Authenticator.

→ MAC is appended to the message at source, the receiver verifies the MAC by recomputing it.

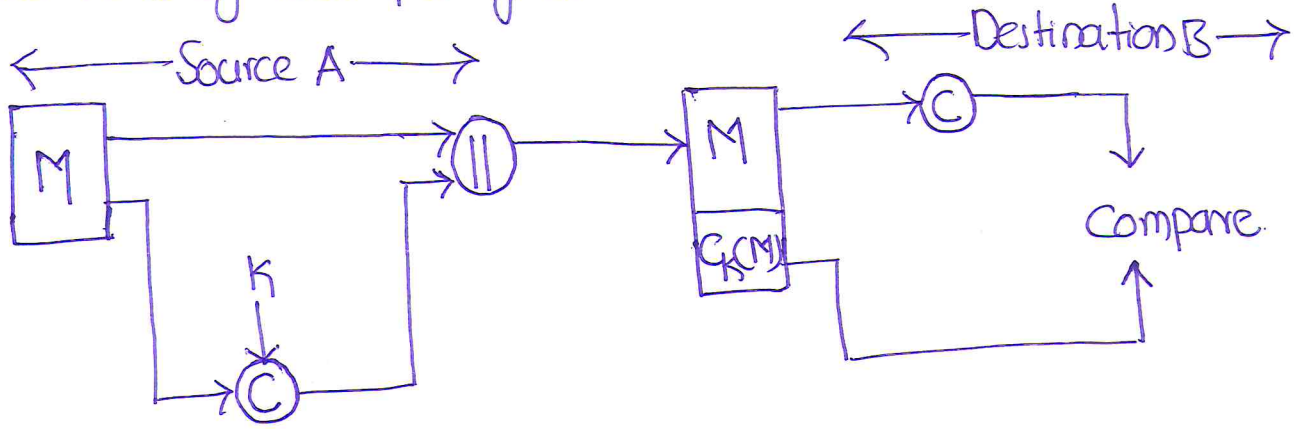


Fig: Message Authentication using MAC

→ Requirements for MACs:-

1. Knowing a message and MAC, it is infeasible to find another message with same MAC
2. MACs should be uniformly distributed
3. MAC should depend equally on all bits of the message

(10) HMAC (MACs Based on Hash Functions):

→ HMAC is specified as Internet Standard RFC2104.

HMAC Design objectives:

1. To use without modifications, available hash functions.
2. To allow for easy replacability of the embedded hash function
3. To preserve original performance of hash function without a significant degradation.
4. To use and handle keys in a simple way.
5. To have well understood cryptographic analysis of strength of authentication mechanism.

HMAC Algorithm:-

H = Embedded Hash Function (Ex: MD5, SHA-1, RIPEMD-160)

IV = Initial Value input to hash function.

M = Message

Y_i = i^{th} block of M

L = No. of blocks in M

b = no. of bits in a block

n = length of hashcode

K = secret key

K^+ = K padded with zeros

ipad = 00110110 repeated b/8 times

opad = 01011100 repeated b/8 times

HMAC can be expressed as,

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

The Algorithm can be described as

1. Append Zeros to the left end of K to create a b-bit string K^+
2. XOR, K^+ with ipad to produce the b-bit block S_i.
3. Append M to S_i.
4. Apply H to stream generated in step ③.
5. XOR, K^+ with opad to produce b-bit block S_o.
6. Append the hash result from step 4 to S_o.
7. Apply H to stream generated in step ⑥ and output the result.

Security of HMAC:-

→ The security of HMAC relates to the underlying Hash algorithm.

→ Attacking HMAC requires either

1. Brute Force attack on key
2. Birthday attack

→ choose a hash function based on speed and security constraints

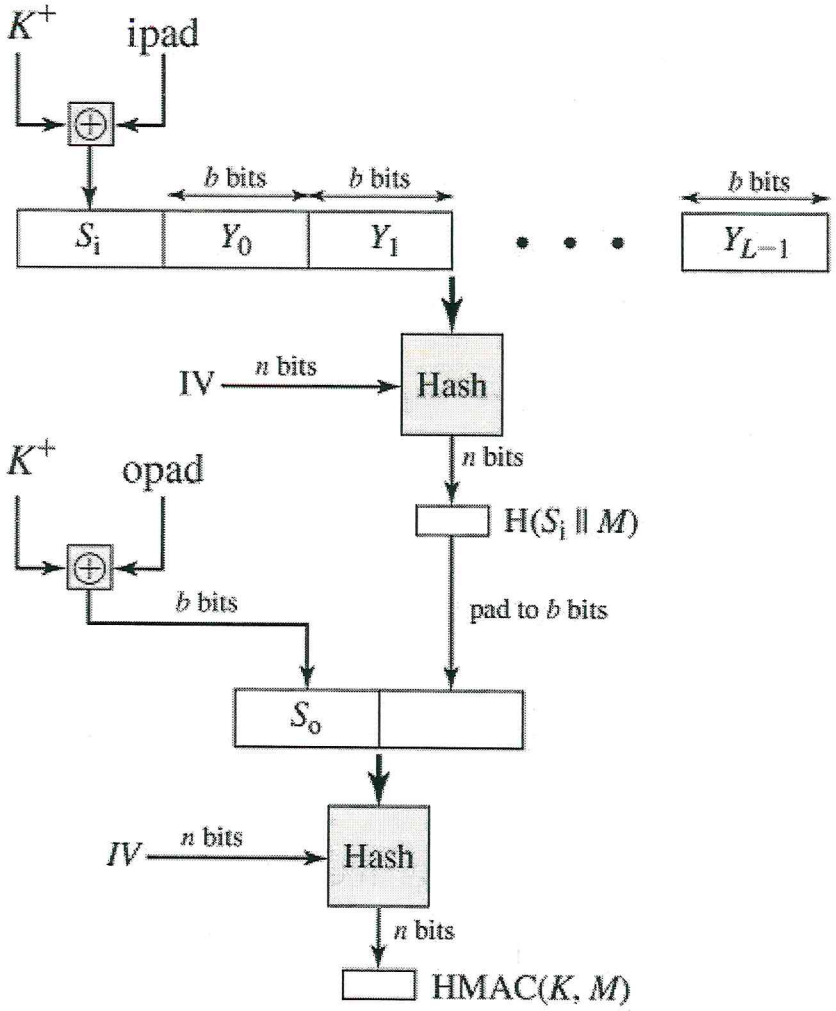


Figure: HMAC Algorithm

(11) CMAC (Cipher-Based Message Authentication Code):

When the message is an integer multiple 'n' of the cipher block length 'b'.

The algorithm makes use of

k-bit encryption key

b-bit constant

→ CMAC is calculated as

$$C_1 = E(K, M_1)$$

$$C_2 = E(K, [M_2 \oplus C_1])$$

$$C_3 = E(K, [M_3 \oplus C_2])$$

$$\vdots$$

$$C_n = E(K, [M_n \oplus C_{n-1} \oplus K_1])$$

$$T = MSB_{Tlen}(C_n)$$

where

T = MAC

$Tlen$ = bit length of T

$MSB_s(X)$ = The s leftmost bits of bit string X.

If the message is not an integer multiple of Cipher block length

Then, the final block is padded to the right with a '1' and as many 0's as necessary so that the final block is also of length b.

The CMAC operation is same as above.

→ The two n-bit keys are derived from k-bit encryption key as follows

$$L = E(K, 0^n)$$

$$K_1 = L \cdot x$$

$$K_2 = L \cdot x^2 = (L \cdot x) \cdot x$$

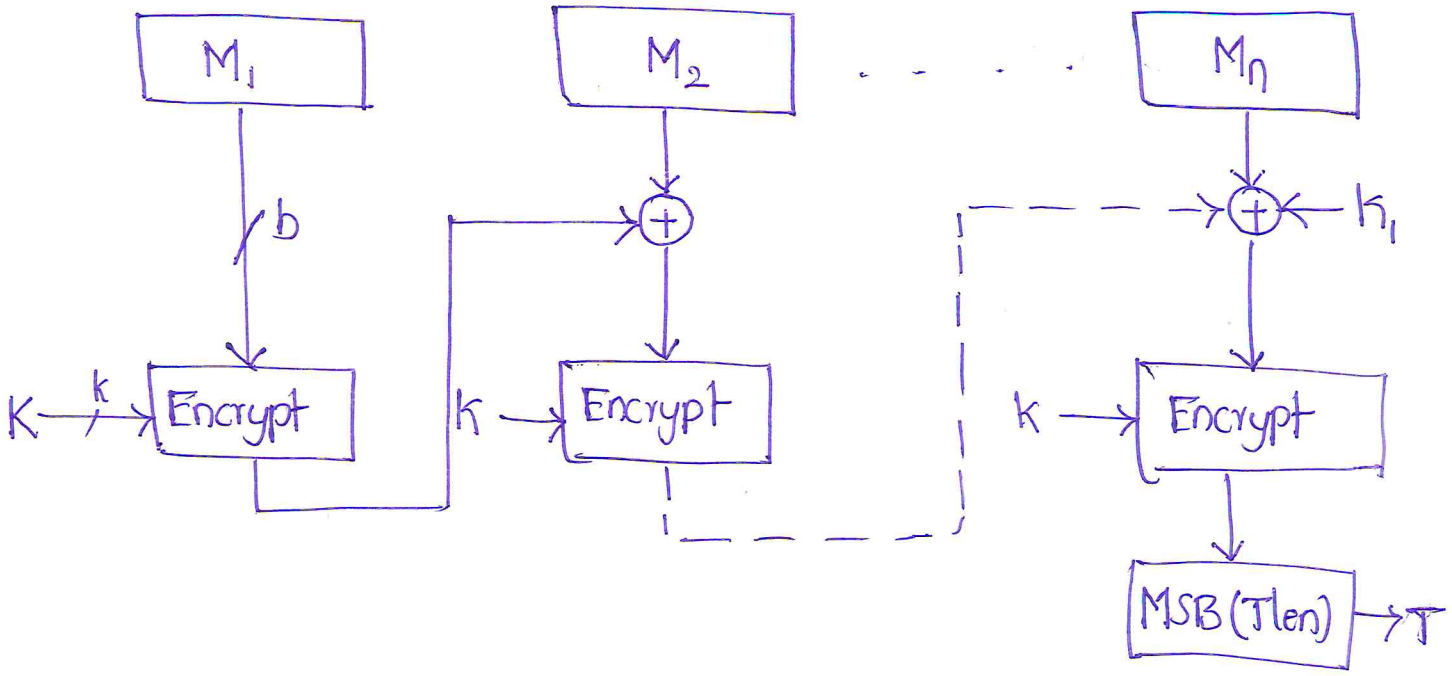


Figure: Message Length is integer multiple of Blocksize

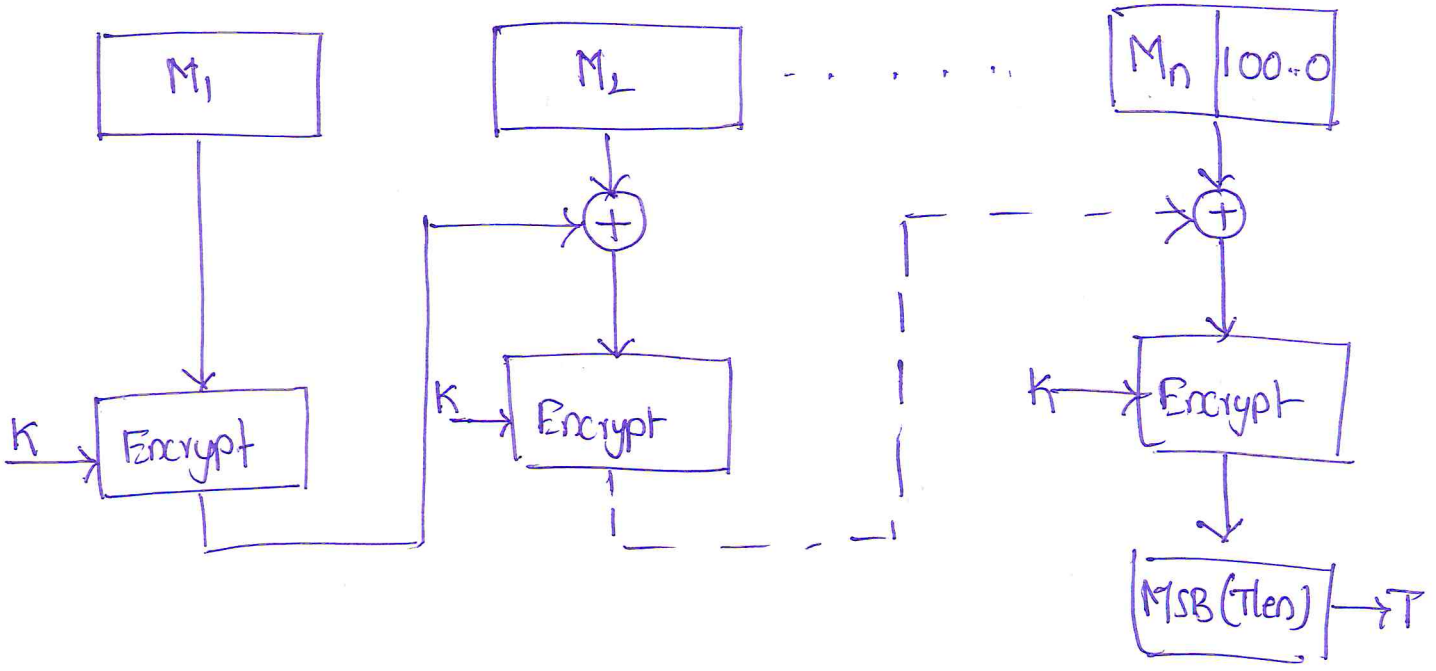


Fig: Message length is not integer multiple of blocksize

Fig: CMAC Operation

Digital Signatures:-

→ Digital Signature is a mathematical scheme for demonstrating the authenticity of digital messages or documents.

→ Digital Signatures provide the ability to

* Verify author, date & time of Signature
* Authenticate message contents
* Be verified by third parties to resolve disputes.

→ properties (or) Requirements:-

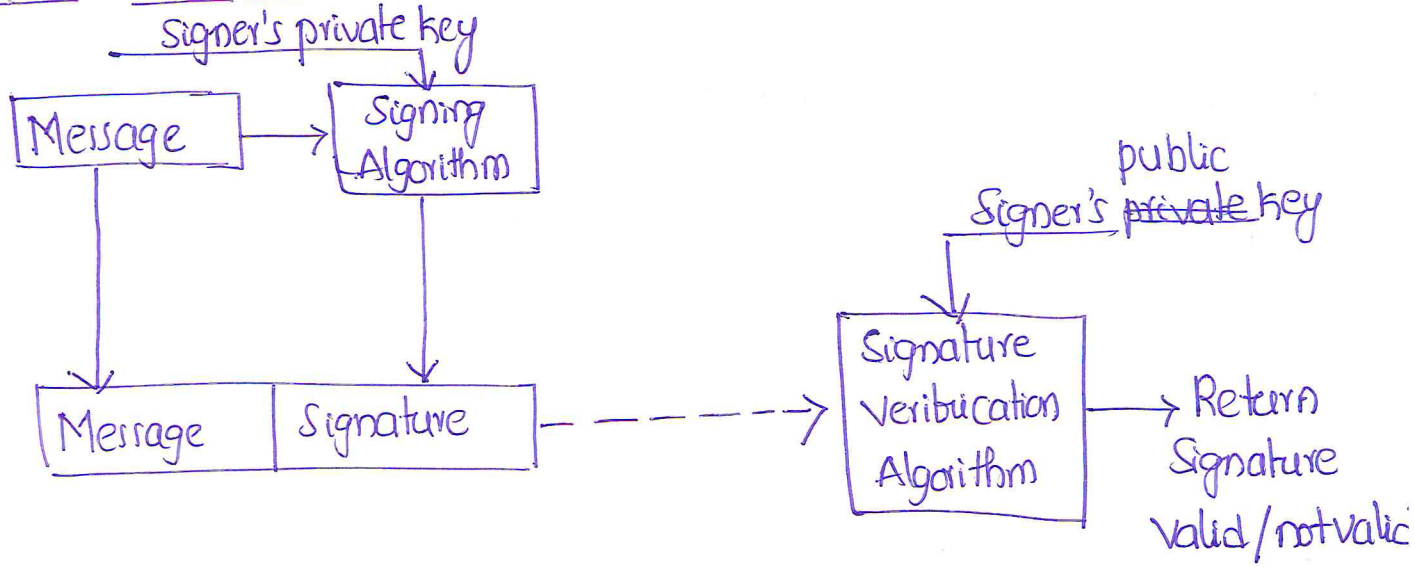
The digital Signature

* must depend on the message signed
* must use information unique to sender to prevent both forgery and denial.
* must be relatively easy to produce
* must be relatively easy to recognize and verify
* be computationally, infeasible to forge
* be practical to save digital signature.

→ There are two types of Digital Signature Schemes,

1. Direct Digital Signature: It involves only sender and receiver
2. Arbitrated Digital Signature: It involves use of arbiter A.
   The arbiter validates any signed message and then dated and sent to recipient.

Generic model of Digital Signature Process:



Digital Signature Requirements:

*

NIST Digital Signature Algorithms:-

→ The National Institute of Standards and Technology has published Federal Information Processing standard FIPS 186 known as "Digital Signature Algorithm" (DSA)

→ DSA makes used of SHA

DSA Approach:

→ DSA makes used of Hash Function

→ The hash code is provided as input to a signature function along with a random number 'k' generated for a particular signature.

→ The signature function also depends on sender's private key (PR_a) and a set of parameters

→ At the receiver end, the hash code of incoming message is generated and signature is input to a verification function. This verification function depends on global public key, and sender's public key.

Digital Signature Algorithm:

* Global Public key components:

p : prime number where $2^{L-1} < p < 2^L$

q : prime divisor of (p-1) where $2^{N-1} < q < 2^N$

$g := h^{(p-1)/q} \bmod p$

* User's private key

x: Random or pseudorandom integer $0 < x < q$

* User's public key

$y = g^x \bmod p$

* User's peer-message secret number

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

Signing Algorithm:

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

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

Signature = (r, s)

Verification Algorithm:

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

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

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

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

TEST : $v = r'$

M : message

H(M) : hash of M using SHA-1

M', r', s' = Received versions of M, r, s.

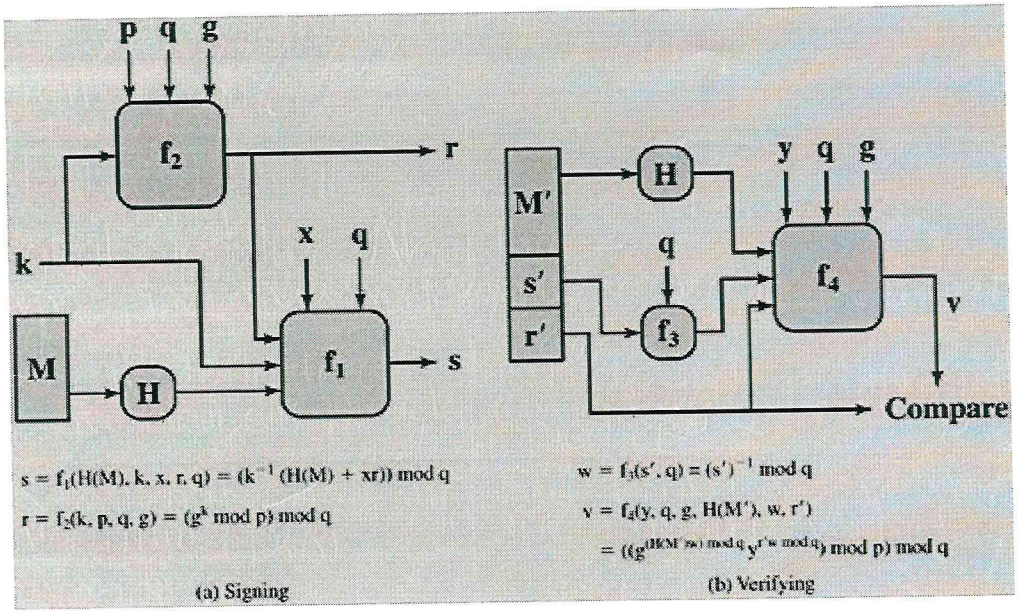


Figure: Signing and Verification in DSA

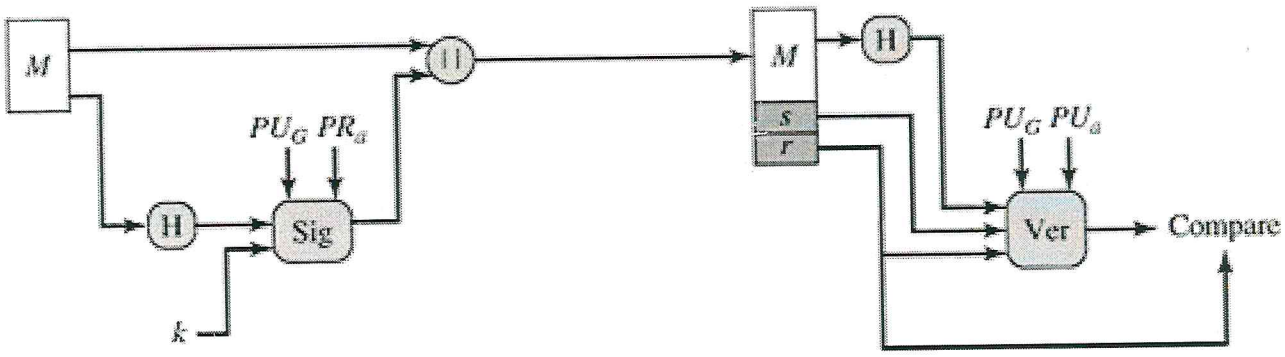


Figure: Overall DSA Approach