UNIT-4

Cryptographic Hash Functions and Digital Signatures

Message Authentication: Message authentication is a mechanism used to verify the integrity of a message. It ensures that data received are exactly as sent by and the identity of sender is valid. A message, file, document etc. is said to be authentic when it is genuine and comes from it's concerned source. The service used to provide message aunthentication is a Message Authentication Code (MAC). Message authentication is concerned with:

→ Protecting the integrity of message.
→ Validating identity of originator.
→ Non-repudiation of origin.

Message Authentication Functions:

Any message authentication or digital signature mechanism has two level of functionality. At the lower level, there must be some sort of function that produces an authenticator: a value to be used to authenticate a message. This lower-level function is then used as a primitive in a higher-level authentication protocol that enables a receiver to verify the authenticity of a message.

The types of functions that may be used to produce an authenticator may be grouped into three classes:

i) Hash functions: A function that maps a message of any length into a fixed-length hash value, which serves as the authenticator.

ii) Message encryption: The ciphertext of the entire message serves as it's authenticator.

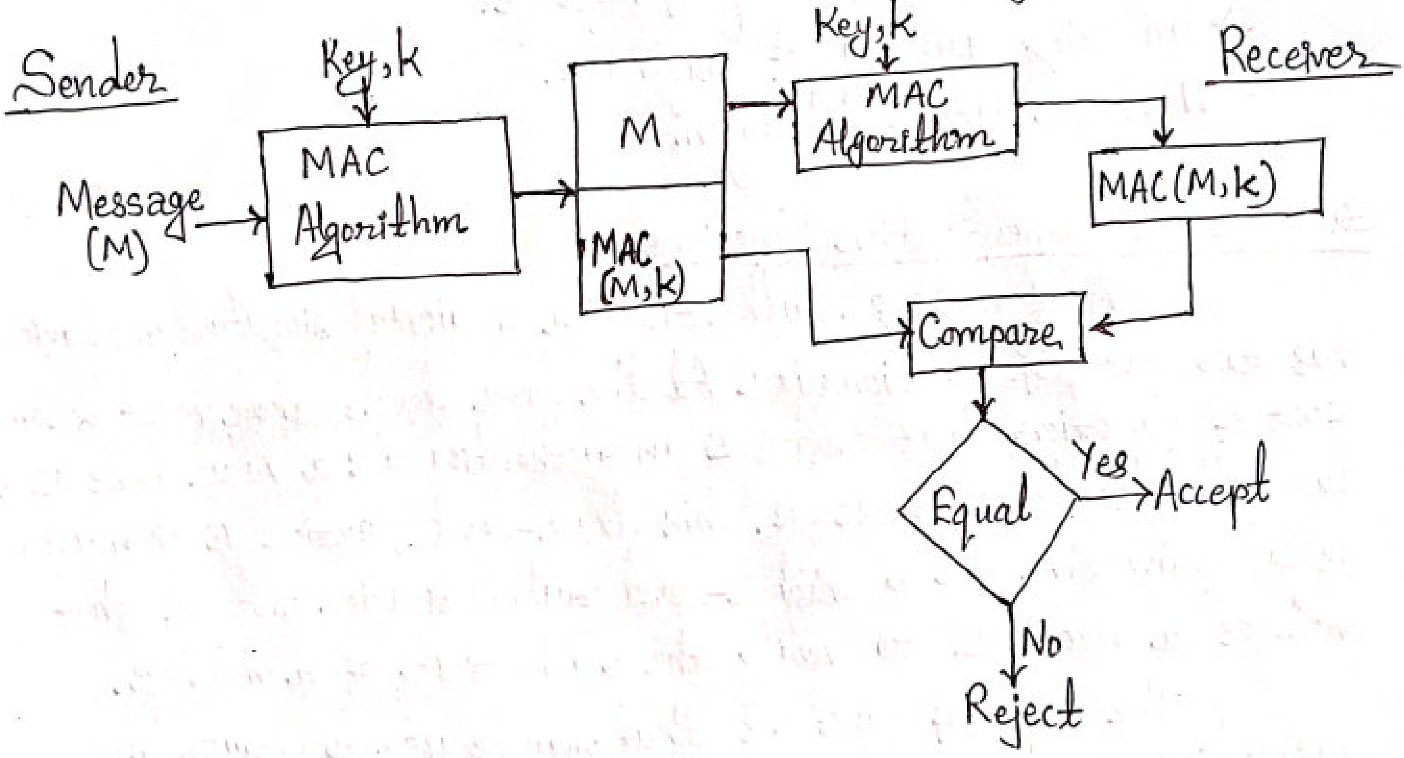
iii) Message Authentication Code (MAC): A function of the message and a secret key that produces a fixed-length value that serves as the authenticator.

⊛. Message Authentication Codes (MAC):

Message authentication code is a function of the message and a secret key that produces a fixed-length value that serves as the authenticator. This technique assumes that the sender and a receiver share a common secret key k. When a sender has a message to send receiver it calculates the MAC as a function of message and key.

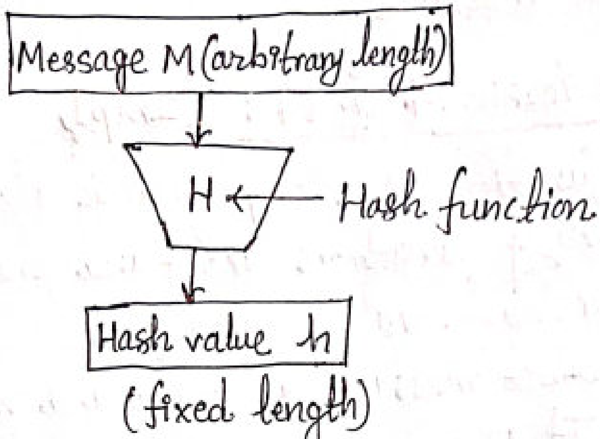
$$MAC = C(M, k)$$

where, C → MAC function.
M → Input Message
k → Shared key.



The message plus MAC are transmitted to the intended receiver. The receiver performs the some calculation on received message using the same secret key to generate a new MAC. Then the received MAC is compared to the calculated. If the received MAC matches the calculated MAC then the receiver is assumed that the message has not been altered.

⊛. Hash Function: It is a function that maps a message of any length into a fixed length hash value, which serves as the authenticator. Cryptographic hash functions are an important tool of cryptography and play a fundamental role in efficient and secure information processing.



⊛. Properties of Hash Function:

1. **One-way property:** It is computationally hard to find the input (message) from the output (hash value) i.e, for any given hash value h, it should be difficult to find any message m such that $H(m)=h$. [Imp]

2. **Weak collision resistance:** For any given input m_1, it is computationally hard to find a different input m_2 such that $H(m_1)=H(m_2)$.

3. **Strong collision resistance:** It is computationally hard to find two different message m_1 and m_2 such that $H(m_1)=H(m_2)$.

4. Produces a fixed-length output.

5. Hash function can be applied to a block of data of any size.

⊛. Applications of Hash Functions:

→ Message authentication
→ Digital signature
→ Password and security
→ Encryption
→ Message digest.

we did not describe these here because these topics will come before or after in this unit like we already discussed message authentication in beginning. If asked in exam, should be described.

⊗. Message Digests:

A message digest is a cryptographic hash function containing a string of digits created by a one-way hashing formula. Message digests are designed to protect the integrity of a piece of data or media to detect changes and alternations to any part of a message.

1) Message Digests Version 4 (MD4): [Imp]

The MD4 function is a cryptographic algorithm that takes a message of arbitrary length as input and produces a 128-bit message digest or hash value as output.

Suppose a b-bit message as input and we need to find its message digest. It is assumed that the bits of the message are $m_0, m_1, \ldots\ldots, m_{b-1}$.

Step1: Append padded bits:

The message is padded so that it's length is congruent to 448, modulo 512. A single '1' bit is append to the message and then '0' bits are appended so that the length in bits equals 448 modulo 512.

Message	100000	

(Message length + padded bits) % 512 = 448.

Step2: Append length:

A 64-bit representation of b is appended to the result of the previous step. The resulting message has a length that is an exact multiple of 512 bits.

Message	100000	64 bits

(Message length + padded bits + 64 bits) % = 0.

Step3: Initialize MD buffer:

A 4-word buffer (A, B, C, D) is used to compute the message digest. Here each of A, B, C, D is a 32-bit register initialized to the following values in hexadecimal, low-order byte first:

A: 01 23 45 67

B: 89 ab cd ef

C: fe dc ba 98

D: 76 54 32 10

Step4: Process message in 16-word blocks:

It contains three passes (rounds) with 16 steps or operation each. We first define three auxiliary functions that each take as input three 32-bit words and produce as output one 32-bit word.

Pass 1: $F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$ [Step 0 to 15]

Pass 2: $G(X,Y,Z) = (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$ [Step 16 to 31]

Pass 3: $H(X,Y,Z) = X \oplus Y \oplus Z$ [Step 32 to 47]

See the operation of passes in book for detail

Step5: Output:

After all, rounds have performed the buffer A, B, C, D contains the MD4 output starting with lower bit A and ending with higher bit D.

2) Message Digest Version 5 (MD5):

The MD5-message digest algorithm is widely used cryptographic hash function producing a 128-bit (16-byte) hash value, typically expressed in text format as a 32 digit hexadecimal number. MD5 were invented by Ron Rivest as an improvement version of MD4.

Operations:

Step1 to Step3 : Same as of MD4

Step4: It contains 4 passes, with 16 steps or operation each. We first define four auxiliary functions that each take as input three 32-bit words and produce as output one 32-bit word.

Pass1: $F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$ [Step 0 to 15]

Pass2: $G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$ [Step 16 to 31]

Steps 1 & 3 same as MD4

Pass3: $H(X,Y,Z) = X \oplus Y \oplus Z$ [Step 32 to 47]

Pass4: $H(X,Y,Z) = Y \oplus (X \wedge \neg Z)$ [Step 48 to 64]

only pass 3 and 4 different here all other are same as MD4

same as MD4

Step5: Output:

After all rounds have performed the buffer A, B, C, D contains the MD5 output starting with lower bit A and ending with higher bit D.

Secure Hash Algorithms (SHA):

The secure hash algorithm is a family of cryptographic hash functions published by the National Institute standards and Technology (NIST) as a U.S. Federal Information Processing Standard (FIPS). It includes following variations:

1) Secure Hash Algorithm-1 (SHA-1): [Imp]

SHA-1 is a cryptographic hash function which takes an input and produces the output of 160 bits hash value known as a message digest. It works for any input message that is less than 2^{16} bits.

Operations:

1. Message Padding: Padding is performed by appending a single "1" bit to the message, and then enough zero bits are append so that the length in bits of the padded message becomes congruent to 448, modulo 512.

2. SHA-1 Message digest computation: SHA-1 consists of 80 iterations. Each time it process the 512 bits message and at last it produces the output of 160 bits hash value.

Let the 160 bits hash value be ABCDE.

Initially,

A = 67 45 23 01

B = ef cd ab 89

C = 98 ba dc fe

D = 10 32 54 76

E = c3 d2 e1 f0.

In each iteration, these values are updated as;

$B = \text{old } A$

$C = \text{old } B \lll 30$

$D = \text{old } C$

$E = \text{old } D$

$A = E + (A \lll 5) + W_t + K_t + f(t, B, C, D)$

where,

$W_t = W_{n-3} \oplus W_{n-8} \oplus W_{n-14} \oplus W_{n-16}$ is the t^{th} 32 bits word block and K_t is the constant depending upon the value of t given by following relations:

$K_t = 5a827999$ if $0 \le t \le 19$

$K_t = 6ed9eba1$ if $20 \le t \le 39$

$K_t = 8f1bbcdc$ if $40 \le t \le 59$

$K_t = ca62c1d6$ if $60 \le t \le 79$

Again, $f(t, B, C, D)$ is a function that varies according to the following relations:

$f(t,B,C,D) = (B \wedge C) \vee (\neg B \wedge D)$ if $0 \le t \le 19$

$f(t,B,C,D) = B \oplus C \oplus D$ if $20 \le t \le 39$

$f(t,B,C,D) = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$ if $40 \le t \le 59$

$f(t,B,C,D) = B \oplus C \oplus D$ if $60 \le t \le 79$

2) Secure Hash Algorithm-2 (SHA-2): [Imp]

SHA-2 is a family of two similar hash functions known as SHA-256 and SHA-512, with different block sizes. Both algorithm belongs to SHA-2. They differ in the word size. SHA-256 uses 32-bit words where SHA-512 uses 64-bit words. There are also truncated versions of each standard, known as SHA-224, SHA-384, SHA-512, SHA-512/224, SHA-512/256.

⊛. Parameters for various version of SHA:

Algorithm	Message Digest Size	Message Size	Block Size	Word Size	No. of Step
SHA-1	160	$< 2^{64}$	512	32	80
SHA-224	224	$< 2^{64}$	512	32	64
SHA-256	256	$< 2^{64}$	512	32	64
SHA-384	384	$< 2^{128}$	1024	64	80
SHA-512	512	$< 2^{128}$	1024	64	80

<u>Note:</u> All sizes are measured in bits.

⊛. SHA-512:

SHA-512 is a hashing algorithm used to convert text of any length into a fixed-size string. Each output produces a SHA-512 length of 512 bits (64 bytes).

The algorithm is commonly used for email addresses hashing, password hashing, and digital record verification. SHA-512 is also used in blockchain technology, with the most notable example being the BitShares network. SHA-512 undergoes following stages:

i) Input formatting
ii) Hash buffer initialization
iii) Message Processing
iv) Output

⊗. Digital Signature: [Imp]

Digital signature is an electronic signature that can be used to authenticate the identity of the sender of a message and to ensure that the original content of the message or document that has been sent is unchanged.

Digital Signature schemes normally gives two algorithms, one for signing which involves the user's secret or private key and one for verifying signatures which involves the user's public key.

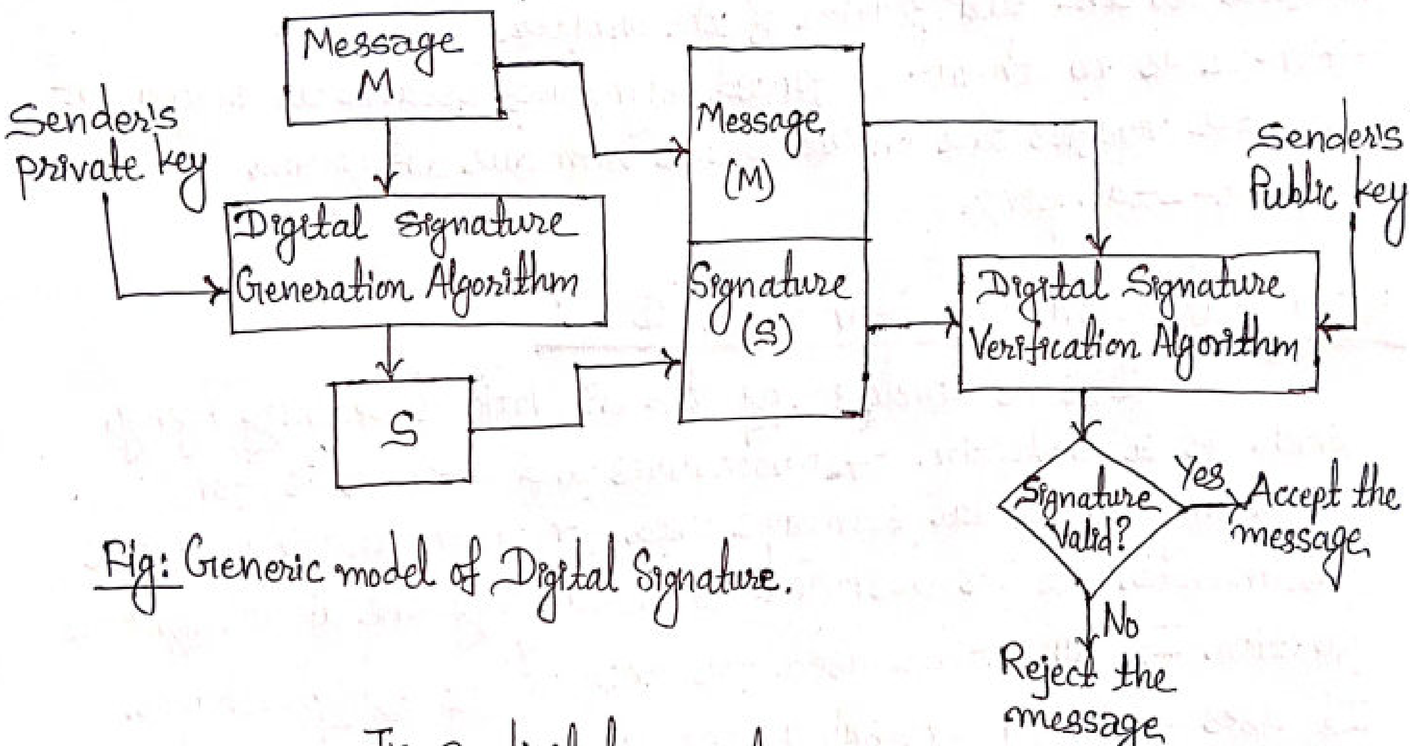


Fig: Generic model of Digital Signature.

In a digital signature process, the sender uses a signing algorithm to sign the message. The message and the signature are sent to receiver. The receiver receives the message and the signature applies verifying algorithm to the combination. If the result is true, the message is accepted otherwise it is rejected.

Direct Digital Signature: A direct digital signature involves only two parties, a sender and a receiver. It is assumed that the receiver knows the public key of the sender. A digital signature may be formed by encrypting the entire message with the sender's private key or by encrypting a hash code of the message with the sender's private key.

→ There is a threat of forged digital signatures in direct digital signature.

Arbitrated digital signature: It involves three parties, a sender, a receiver and a trusted arbiter. Every signed message from a sender X to a receiver Y goes first to an arbiter A, who subjects the message and its signature to a number of tests to check its origin and content. The message is then dated and sent to Y with an indication that it has been verified to the satisfication of the arbiter.

→ There is no threat of forged signatures because an independent arbiter verifies the entire process with due dating and time-stamping.

⊛. Digital Signature Standard (DSS):

DSS is developed by the U.S National Security Agency, which is a collection of procedures and standards for generating a digital signature used for authenticating electronic documents. It is designed to provide only the digital signature function. It cannot be used for encryption or key exchange. It uses SHA for hashing the message.

⊛. The DSS Approach for Digital Signature: [Imp]

DSS approach makes use of hash function. The hash code is provided as input to a signature function along with a random number k generated for this particular signature. The signature function also depends on the sender's private key (PR_a) and a set of parameters known to a group of communicating principals. We can consider this set to constitute a global public key (PR_G). The result is a signature consisting of two components, labeled s and r.

At the receiving end, the hash code of the incoming message is generated. This plus the signature is input to a verification function. The verification function 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 output of verification function is a value that is equal to the signature component r if the signature is valid. The signature function is such that only the sender, with knowledge of the private key, could have produced the valid signature.

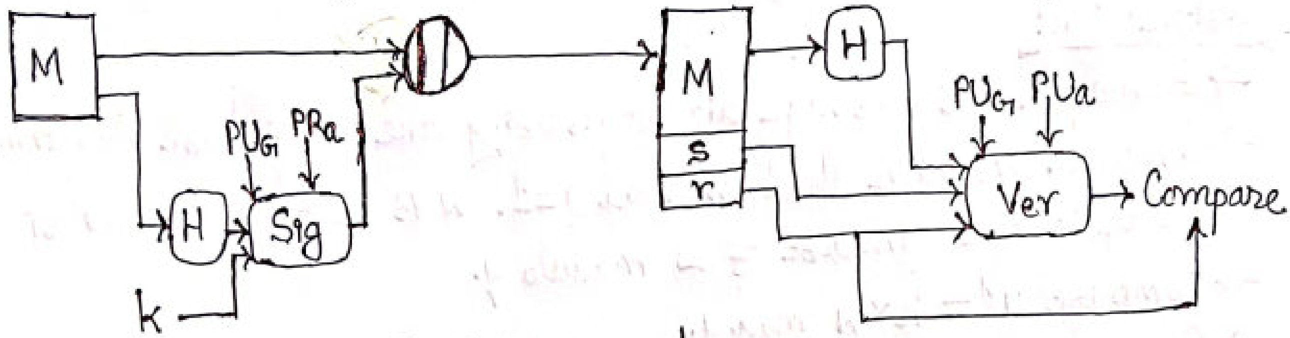


Fig: DSS Approach

⊛. Digital Signature Algorithm (DSA):

→ It creates a 320 bit signature with 512–1024 bit security.

→ It consists of 2 parts: generation of a pair of public key and private key & generation and verification of digital signature, which can be described as;

1) Generation of a pair of public key and private key:

→ Choose a prime number q, which is called the prime divisor.

→ Choose another prime number p, such that $p-1 \bmod q = 0$. p is the prime modulus.

→ Choose an integer g, such that $1 < g < p$, $g^q \bmod p = 1$ and $g = h^{(p-1)/q} \bmod p$.

→ Choose an integer, such that $0 < x < q$.

→ Compute $y = g^x \bmod p$.

→ Package the public keys as $\{p, q, g, y\}$.

→ Package the private keys as $\{p, q, g, x\}$.

2) Signature generation and signature verification:

Generation:

→ Generate the message digest h, using a hash algorithms like SHA-1.
→ Generate a random number k, such that $0 < k < q$.
→ Compute $r = (g^k \bmod p) \bmod q$. If $r = 0$, select a different k.
→ Compute i, such that $k^i \bmod q = 1$. i is called the modular multiplicative inverse of k modulo q.
→ Compute $s = i(h + rx) \bmod q$. If $s = 0$, select a different k.
→ Package the digital signature as $\{r, s\}$.

Verification:

→ Generate the message digest h, using the same hash algorithm.
→ Compute w, such that $sw \bmod q = 1$. w is called the modular multiplicative inverse of s modulo q.
→ Compute $u1 = h \times w \bmod q$.
→ Compute $u2 = r \times w \bmod q$.
→ Compute $v = ((g^{u1} \times y^{u2}) \bmod p) \bmod q$.
→ If $v == r$, the digital signature is valid.

⊛. RSA Approach for Digital Signature: [Imp]

The message to be signed is input to a hash function that produces a secure hash code of fixed length. This hash code is then encrypted using the sender's private key (PRa) to form the signature. Both the message and the signature are then transmitted.

The recipient takes the message and produces a hash code. The recipient also decrypts the signature using the sender's public key (PUa).

If the calculated hash code matches the decrypted signature, the signature is accepted as valid. Because only the sender knows the private key, only the sender could have produced a valid signature.

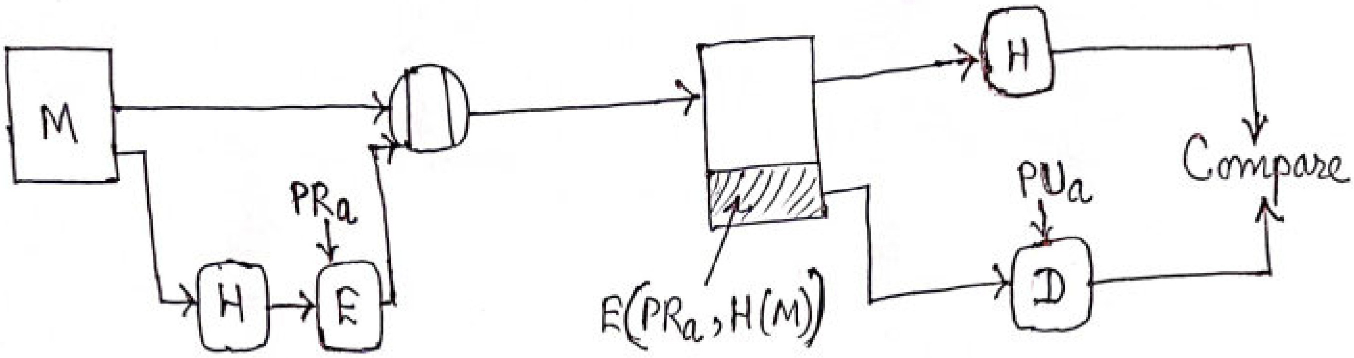


Fig: RSA Approach