

- ☒ Message authentication is concerned with:
 - ☒ protecting the integrity of a message
 - ☒ validating identity of originator
 - ☒ non-repudiation of origin (dispute resolution)
- ☒ Will consider the security requirements
- ☒ Three alternative functions used:
 - ☒ message encryption
 - ☒ message authentication code (MAC)
 - ☒ hash function

Security Requirements

- ☒ Disclosure
- ☒ Traffic analysis
- ☒ Masquerade
- ☒ Content modification
- ☒ Sequence modification
- ☒ Timing modification
- ☒ Source repudiation
- ☒ Destination repudiation

Message Encryption

- ☒ Message encryption by itself also provides a measure of authentication
- ☒ If symmetric encryption is used then:
 - ☒ receiver know sender must have created it
 - ☒ since only sender and receiver now key used
 - ☒ know content cannot of been altered
 - ☒ if message has suitable structure, redundancy or a checksum to detect any changes

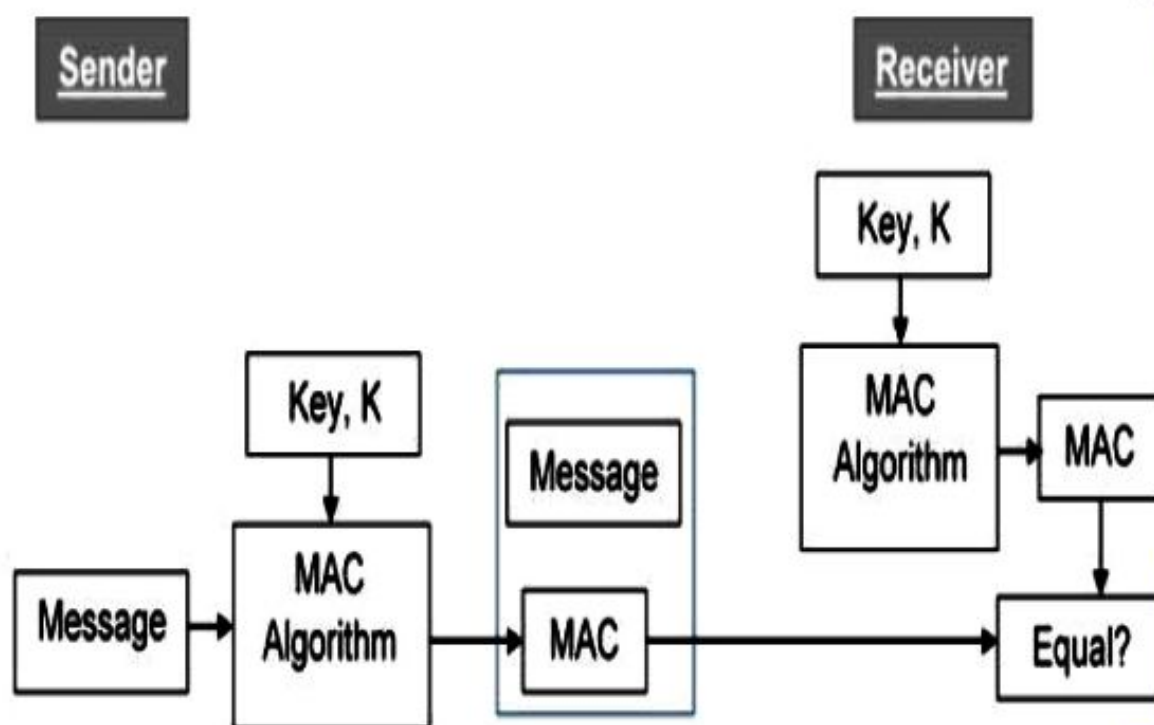
Message Encryption

- ☒ If public-key encryption is used:
 - ☒ encryption provides no confidence of sender
 - ☒ since anyone potentially knows public-key
 - ☒ however if
 - ☒ sender **signs** message using their private-key
 - ☒ then encrypts with recipients public key
 - ☒ have both secrecy and authentication
 - ☒ again need to recognize corrupted messages
 - ☒ but at cost of two public-key uses on message

Message Authentication Code (MAC)

- Generated by an algorithm that creates a small fixed-sized block
 - depending on both message and some key
 - like encryption though need not be reversible
- Appended to message as a **signature**
- Receiver performs same computation on message and checks it matches the MAC
- Provides assurance that message is unaltered and comes from sender

MAC Process



- ✘ The sender uses some publicly known MAC algorithm, inputs the message and the secret key K and produces a MAC value.
- ✘ Similar to hash, MAC function also compresses an arbitrary long input into a fixed length output. The major difference between hash and MAC is that MAC uses secret key during the compression.
- ✘ The sender forwards the message along with the MAC. Here, we assume that the message is sent in the clear, as we are concerned of providing message origin authentication, not confidentiality. If confidentiality is required then the message needs encryption.

- ✘ On receipt of the message and the MAC, the receiver feeds the received message and the shared secret key K into the MAC algorithm and re-computes the MAC value.
- ✘ The receiver now checks equality of freshly computed MAC with the MAC received from the sender. If they match, then the receiver accepts the message and assures himself that the message has been sent by the intended sender.
- ✘ If the computed MAC does not match the MAC sent by the sender, the receiver cannot determine whether it is the message that has been altered or it is the origin that has been falsified. As a bottom-line, a receiver safely assumes that the message is not the genuine.

Message Authentication Codes

- As shown the MAC provides authentication
- Can also use encryption for secrecy
 - generally use separate keys for each
 - can compute MAC either before or after encryption
 - is generally regarded as better done before
- Why use a MAC?
 - sometimes only authentication is needed
 - sometimes need authentication to persist longer than the encryption (eg. archival use)

Note that a MAC is not a digital signature

MAC Properties

- A MAC is a cryptographic checksum
$$\text{MAC} = C_K(M)$$
 - condenses a variable-length message M
 - using a secret key K
 - to a fixed-sized authenticator
- Is a many-to-one function
 - potentially many messages have same MAC
 - but finding these needs to be very difficult



Requirements for MACs

- ⊠ Taking into account the types of attacks
- ⊠ Need the MAC to satisfy the following:
 1. knowing a message and MAC, 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

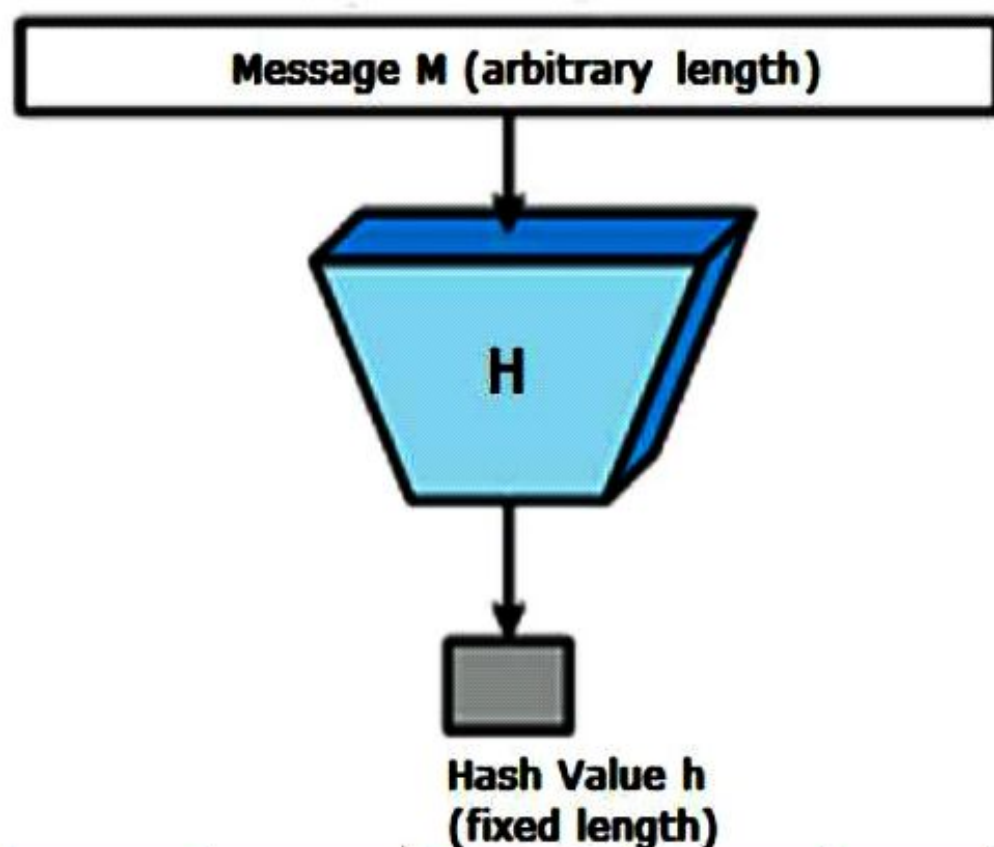
Limitations of MAC

- ⊠ **Establishment of Shared Secret.**
 - ⊠ It can provide message authentication among pre-decided legitimate users who have shared key.
 - ⊠ This requires establishment of shared secret prior to use of MAC.
- ⊠ **Inability to Provide Non-Repudiation**
 - ⊠ Non-repudiation is the assurance that a message originator cannot deny any previously sent messages and commitments or actions.
 - ⊠ MAC technique does not provide a non-repudiation service. If the sender and receiver get involved in a dispute over message origination, MACs cannot provide a proof that a message was indeed sent by the sender.
 - ⊠ Though no third party can compute the MAC, still sender could deny having sent the message and claim that the receiver forged it, as it is impossible to determine which of the two parties computed the MAC.

Hash Functions

- ☒ Condenses arbitrary message to fixed size
 $h = H(M)$
- ☒ Usually assume that the hash function is public and not keyed
 - ☒ cf. MAC which is keyed
- ☒ Hash used to detect changes to message
- ☒ Values returned by a hash function are called **message digest** or simply **hash values**.
- ☒ Can use in various ways with message
- ☒ Most often to create a digital signature

14



Requirements for Hash Functions

1. Can be applied to any sized message M
2. Produces fixed-length output h
3. Easy to compute $h=H(M)$ for any message M
4. Given h is infeasible to find x s.t. $H(x)=h$
 - one-way property
5. Given x is infeasible to find y s.t. $H(y)=H(x)$
 - weak collision resistance
6. Infeasible to find any x,y s.t. $H(y)=H(x)$
 - strong collision resistance

Simple Hash Functions

- ⊠ Are several proposals for simple functions
- ⊠ Based on XOR of message blocks
- ⊠ Not secure since can manipulate any message and either not change hash or change hash also
- ⊠ Need a stronger cryptographic function

Block Ciphers as Hash Functions

- ❑ Can use block ciphers as hash functions
 - ❑ using $H_0=0$ and zero-pad of final block
 - ❑ compute: $H_i = E_{M_i} [H_{i-1}]$
 - ❑ and use final block as the hash value
 - ❑ similar to CBC but without a key
- ❑ Resulting hash is too small (64-bit)
 - ❑ both due to direct birthday attack
 - ❑ and to “meet-in-the-middle” attack
- ❑ Other variants also susceptible to attack

Hash Functions & MAC Security

Like block ciphers have:

- ❑ **brute-force** attacks exploiting
 - ❑ strong collision resistance hash have cost $2^{m/2}$
 - ❑ have proposal for h/w MD5 cracker
 - ❑ 128-bit hash looks vulnerable, 160-bits better
- ❑ MACs with known message-MAC pairs
 - ❑ can either attack key space (cf key search) or MAC
 - ❑ at least 128-bit MAC is needed for security

Like block ciphers have:

☒ **brute-force** attacks exploiting

- ☒ strong collision resistance hash have cost $2^{m/2}$
 - ☒ have proposal for h/w MD5 cracker
 - ☒ 128-bit hash looks vulnerable, 160-bits better
- ☒ MACs with known message-MAC pairs
 - ☒ can either attack keyspace (cf key search) or MAC
 - ☒ at least 128-bit MAC is needed for security

Hash Functions & MAC Security

- ☒ **cryptanalytic attacks** exploit structure
 - ☒ like block ciphers want brute-force attacks to be the best alternative
- ☒ have a number of analytic attacks on iterated hash functions
 - ☒ $CV_i = f[CV_{i-1}, M_i]; H(M) = CV_N$
 - ☒ typically focus on collisions in function f
 - ☒ like block ciphers is often composed of rounds
 - ☒ attacks exploit properties of round functions

Hash and MAC Algorithms

❏ Hash Functions

- ❏ condense arbitrary size message to fixed size
- ❏ by processing message in blocks
- ❏ through some compression function
- ❏ either custom or block cipher based

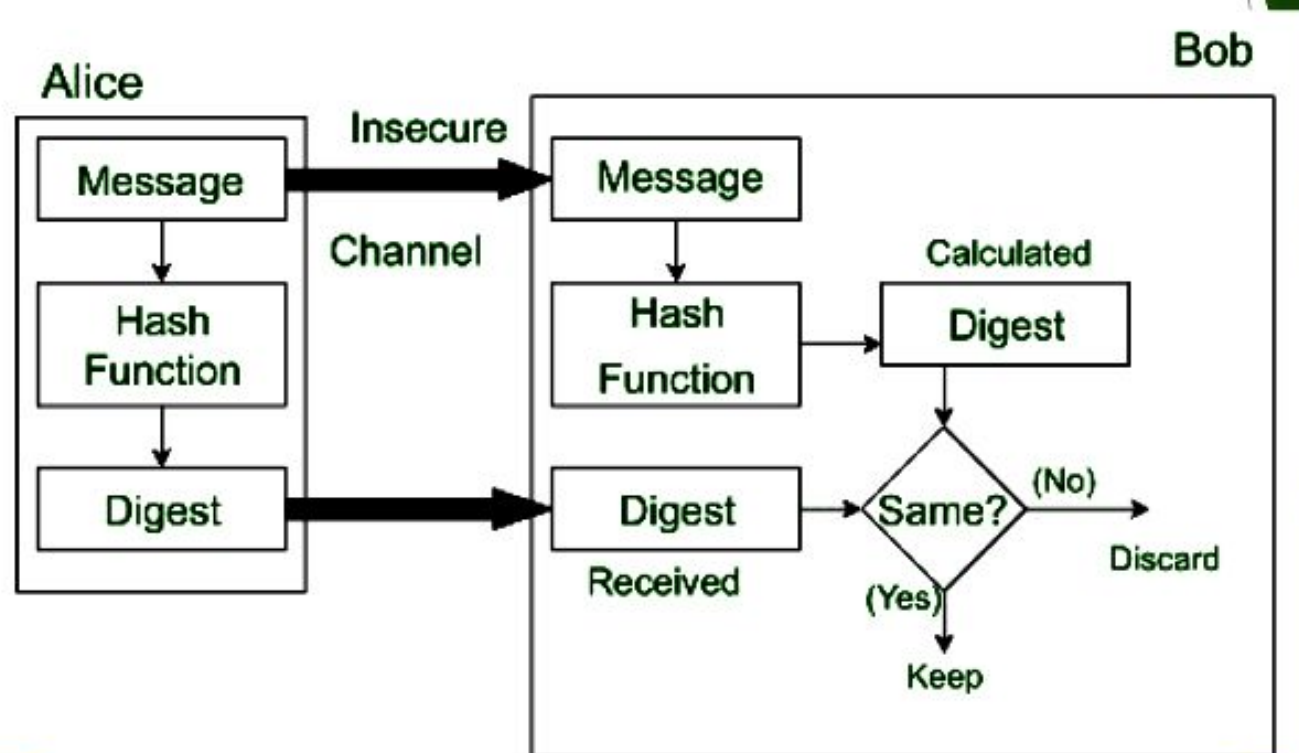
❏ Message Authentication Code (MAC)

- ❏ fixed sized authenticator for some message
- ❏ to provide authentication for message
- ❏ by using block cipher mode or hash function

Message Digest (MD) Algorithm

- ❏ **Message Digest** is used to ensure the integrity of a message transmitted over an insecure channel (where the content of the message can be changed).
- ❏ The message is passed through a Cryptographic hash function. This function creates a compressed image of the message called **Digest**.
- ❏ Message digests are also called one-way *hash functions* because they produce values that are difficult to invert, resistant to attack, effectively unique, and widely distributed.
- ❏ MD Algorithm Developed by the Ron Rivest.
- ❏ Total Six Version Available of MD Algorithm: MD1, MD2, MD3, MD4, MD5 & MD6.
- ❏ Currently Used MD5 Algorithm.

Message Digest Working Diagram



MD5 Algorithm

- ❑ MD5 was most popular and widely used hash function for quite some years.
- ❑ The MD family comprises of hash functions MD2, MD4, MD5 and MD6. It was adopted as Internet Standard RFC 1321. It is a 128-bit hash function.
- ❑ MD5 digests have been widely used in the software world to provide assurance about integrity of transferred file. For example, file servers often provide a pre-computed MD5 checksum for the files, so that a user can compare the checksum of the downloaded file to it.

Algorithm Steps

❏ Padding

The padding consists of a single 1 bit in the first column, followed by enough zeros to pad the message to the required length till the 512 bit. Padding is always used, even if the original length of M happens to equal $448 \bmod 512$. As a result, there is at least one bit of padding, and at most 512 bits of padding. Then the length in bits of the message uses before padding is appended as a 64-bit block.

The padded message is a multiple of 512 bits and, it is also a multiple of 32 bits.

❏ Append the Length Field

In this we calculate the length of the original message. This length is added at the last after the padding bits in the original message.

The final message will be used for hashing to generate the message digest.

❏ Divide the Input Final Message into 512 bits block

❏ Initialize Chaining Variable

❏ MD5 uses the four changing variable. They are identified as A, B, C & D. Each of these 32 bit number.

A	01	23	45	67
B	89	AB	CD	EF
C	FE	DC	BA	98
D	76	54	32	10

❏ Process Block:

Divide Current 512 Bits Block into 16 Sub-Blocks of 32 Bits.

- MD5 uses four auxiliary functions that each take as input three 32-bit words and produce as output one 32-bit word.

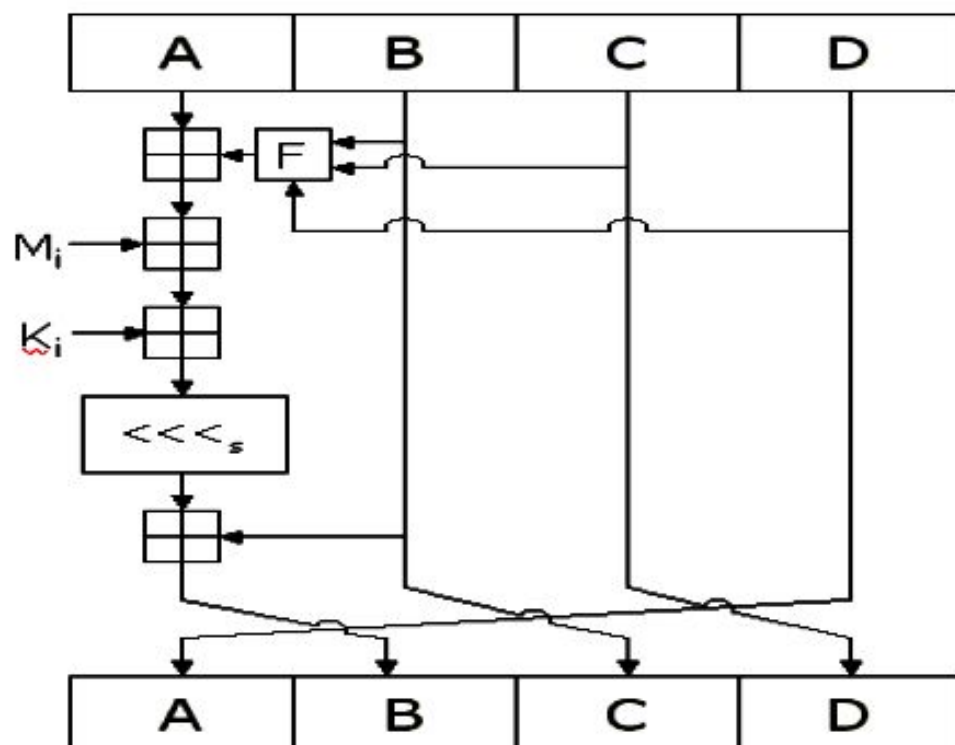
$$F(X, Y, Z) = (X \wedge Y) \vee (\neg X \wedge Z)$$

$$G(X, Y, Z) = (X \wedge Y) \vee (Y \wedge \neg Z)$$

$$H(X, Y, Z) = X \oplus Y \oplus Z$$

$$I(X, Y, Z) = Y \oplus (X \vee \neg Z)$$

- The uses of the four buffers (A, B, C, and D) are now combined with the words of the input using the four auxiliary functions (F, G, H and I). Here, there are four rounds, each involves 16 basic operations to perform



One operation of MD5

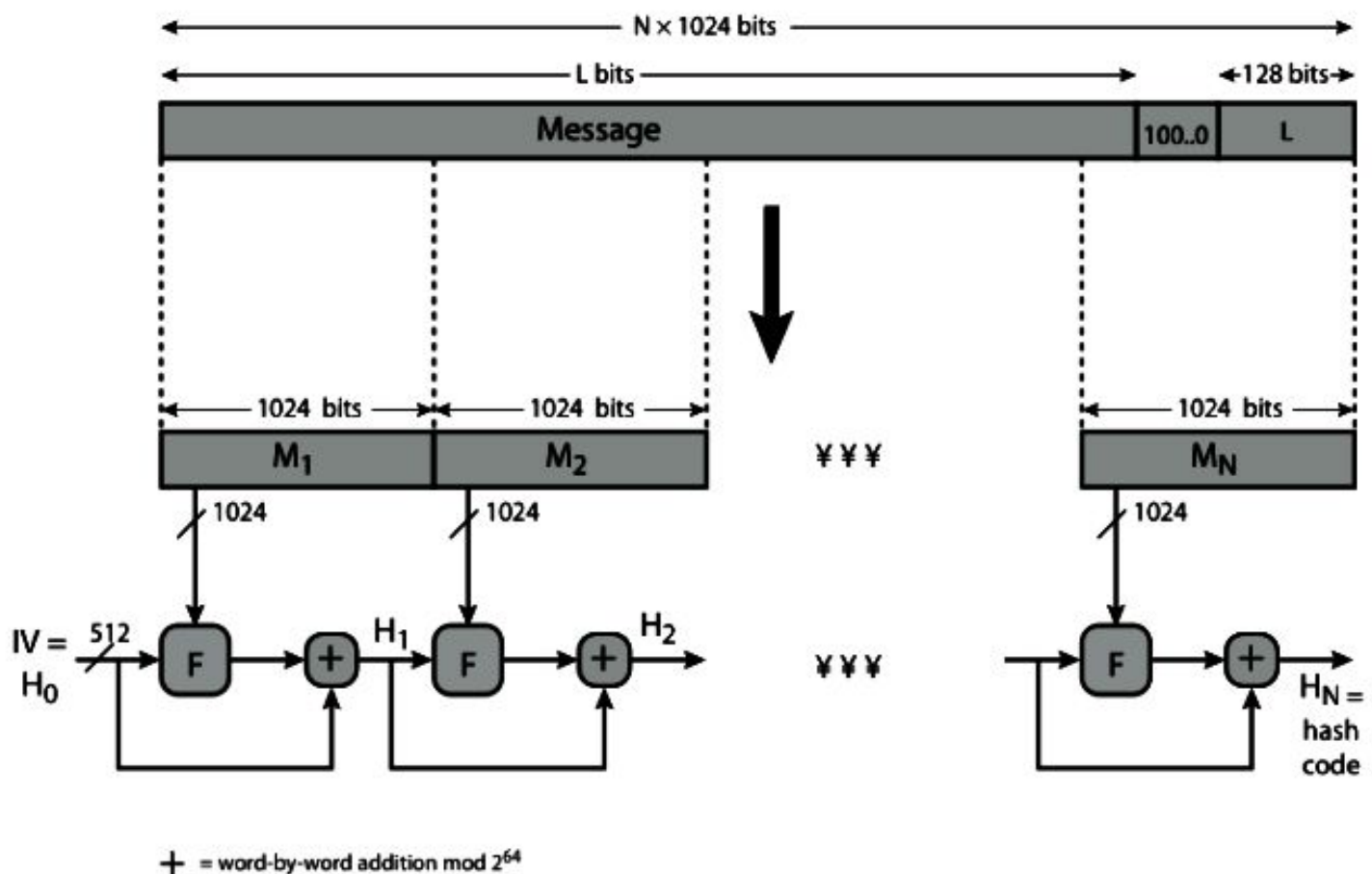
Secure Hash Algorithm

- ❑ SHA originally designed by NIST & NSA in 1993
- ❑ Was revised in 1995 as SHA-1
- ❑ US standard for use with DSA signature scheme
 - ❑ standard is FIPS 180-1 1995, also Internet RFC3174
 - ❑ nb. the algorithm is SHA, the standard is SHS
- ❑ Based on design of MD4 with key differences
- ❑ Produces 160-bit hash values
- ❑ Recent 2005 results on security of SHA-1 have raised concerns on its use in future applications

Revised Secure Hash Standard

- ❑ NIST issued revision FIPS 180-2 in 2002
- ❑ Adds 3 additional versions of SHA
 - ❑ SHA-256, SHA-384, SHA-512
- ❑ Designed for compatibility with increased security provided by the AES cipher
- ❑ Structure & detail is similar to SHA-1
- ❑ Hence analysis should be similar
- ❑ But security levels are rather higher

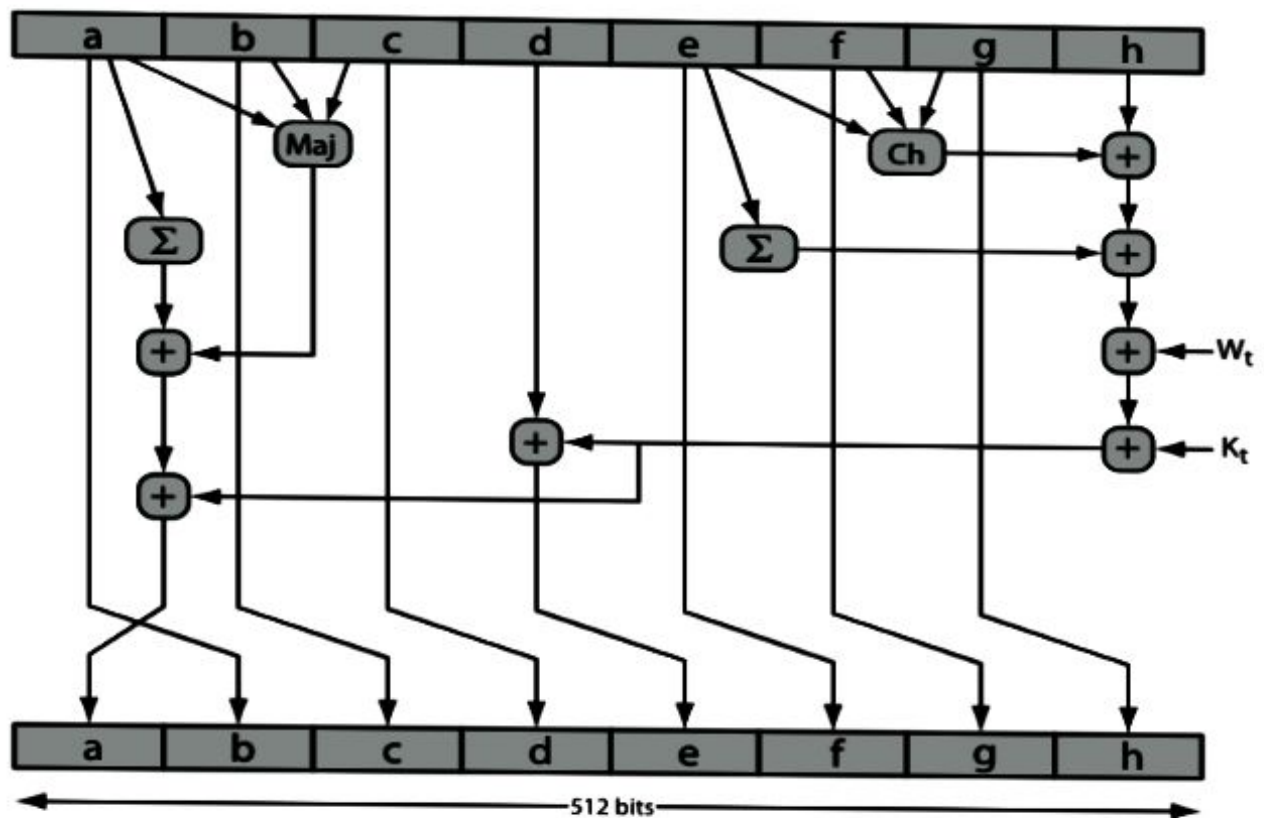
SHA-512 Overview



SHA-512 Compression Function

- ⊠ Heart of the algorithm
- ⊠ Processing message in 1024-bit blocks
- ⊠ Consists of 80 rounds
 - ⊠ updating a 512-bit buffer
 - ⊠ using a 64-bit value W_t derived from the current message block
 - ⊠ and a round constant based on cube root of first 80 prime numbers

SHA-512 Round Function



Difference Between MD5 & SHA-1

S.NO	MD5	SHA1
1.	MD5 stands for Message Digest.	While SHA1 stands for Secure Hash Algorithm.
2.	MD5 can have 128 bits length of message digest.	Whereas SHA1 can have 160 bits length of message digest.
3.	The speed of MD5 is fast in comparison of SHA1's speed.	While the speed of SHA1 is slow in comparison of MD5's speed.
4.	To make out the initial message the aggressor would want 2^{128} operations whereas exploitation the MD5 algorithmic program.	On the opposite hand, in SHA1 it'll be 2^{160} that makes it quite troublesome to seek out.
5.	MD5 is simple than SHA1.	While SHA1 is more complex than MD5.
6.	MD5 provides indigent or poor security.	While it provides balanced or tolerable security.
7.	In MD5, if the assailant needs to seek out the 2 messages having identical message digest then assailant would need to perform 2^{64} operations.	Whereas in SHA1, assailant would need to perform 2^{80} operations which is greater than MD5.

MACS Based on Hash Functions:

HMAC

- ❑ **HMAC algorithm** stands for Hashed or Hash based Message Authentication Code.
- ❑ It is a result of work done on developing a MAC derived from cryptographic hash functions.
- ❑ HMAC is a great resistant towards cryptanalysis attacks as it uses the Hashing concept twice.
- ❑ HMAC consists of twin benefits of Hashing and MAC, and thus is more secure than any other authentication codes.
- ❑ RFC 2104 has issued HMAC, and HMAC has been made compulsory to implement in IP security.
- ❑ The FIPS 198 NIST standard has also issued HMAC.

- ❑ Hash-based message authentication code (HMAC) provides the server and the client each with a public and private key.
- ❑ The public key is known, but the private key is known only to that specific server and that specific client.
- ❑ The client creates a unique HMAC, or hash, per request to the server by combining the request data and hashing that data, along with a private key and sending it as part of a request.
- ❑ The server receives the request and regenerates its own unique HMAC. The server compares the two HMACs, and, if they're equal, the client is trusted and the request is executed. This process is often called a secret handshake.

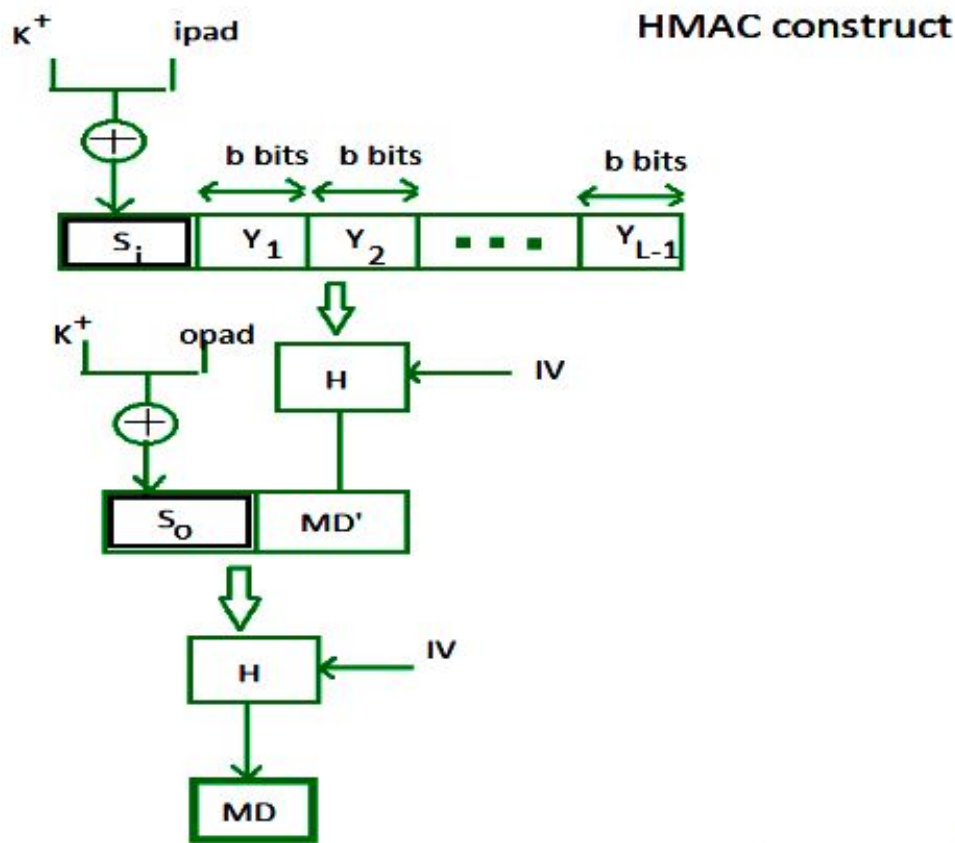
Objectives of HMAC

- ❏ As the Hash Function, HMAC is also aimed to be one way, i.e, easy to generate output from input but complex the other way round.
- ❏ It aims at being less effected by collisions than the hash functions.
- ❏ HMAC reuses the algorithms like MD5 and SHA-1 and checks to replace the embedded hash functions with more secure hash functions, in case found.
- ❏ HMAC tries to handle the Keys in more simple manner.

HMAC Algorithm

- ❏ The working of HMAC starts with taking a message M containing blocks of length b bits.
- ❏ An input signature is padded to the left of the message and the whole is given as input to a hash function which gives us a temporary message digest MD' .
- ❏ MD' again is appended to an output signature and the whole is applied a hash function again, the result is our final message digest MD .

Simple Structure of HMAC



$$S_i = K^+ \oplus \text{ipad}$$

where K^+ is nothing but K padded with zeros on the left so that the result is b bits in length

$$S_0 = K^+ \oplus \text{opad}$$

where ipad and opad are 00110110 and 01011100 respectively taken $b/8$ times repeatedly.

$$\text{MD}' = H(S_i || M)$$

$$\text{MD} = H(S_0 || \text{MD}') \quad \text{or} \quad \text{MD} = H(S_0 || H(S_i || M))$$

Description of HMAC Algorithm

- Append zeros to the left end of K to create a b -bit string K^+ (e.g., if K is of length 160 bits and $b = 512$, then K will be appended with 44 zeroes).
- XOR (bitwise exclusive-OR) K^+ with ipad to produce the b -bit block S_i .
- Append M to S_i .
- Apply H to the stream generated in step 3.
- XOR K^+ with opad to produce the b -bit block S_o .
- Append the hash result from step 4 to S_o .
- Apply H to the stream generated in step 6 and output the result.

HMAC Security

- Security of HMAC relates to that of the underlying hash algorithm •
- If used with a secure hash functions (s.t. SHA1) and according to the specification (key size, and use correct output), no known practical attacks against HMAC •
- In general, HMAC can be attacked as follows:
 - brute force on the key space
 - attacks on the hash function itself
 - birthday attack, although the use of key makes this attack more difficult
 - attacks against the compression function

MACs based on Block Ciphers

We look at two MACs that are based on the use of a block cipher mode of operation.

1. Data Authentication Algorithm (DAA)
2. Cipher Based Message Authentication Codes (CMAC)

Data Authentication Algorithm

- ⌘ Data Authentication Algorithm(DAA) is a widely used MAC based on DESCBC.
- ⌘ Send final block as the MAC or the leftmost M bits ($16 \leq M \leq 64$) of final block.
- ⌘ If necessary, the final block is padded on the right with zeroes to form a full 64-bit block.

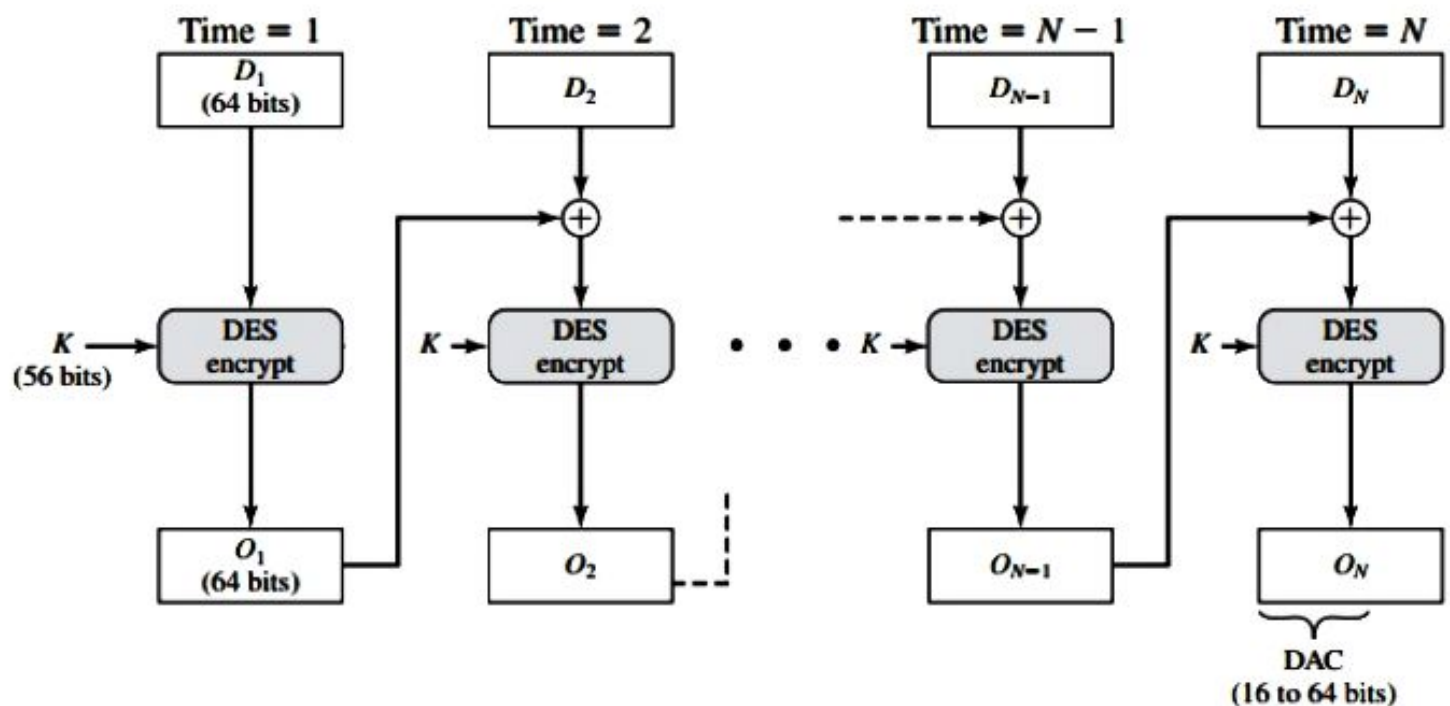
$$O1 = E(K, D)$$

$$O2 = E(K, [D2 \oplus O1])$$

$$O3 = E(K, [D3 \oplus O2])$$

$$ON = E(K, [DN \oplus ON-1])$$

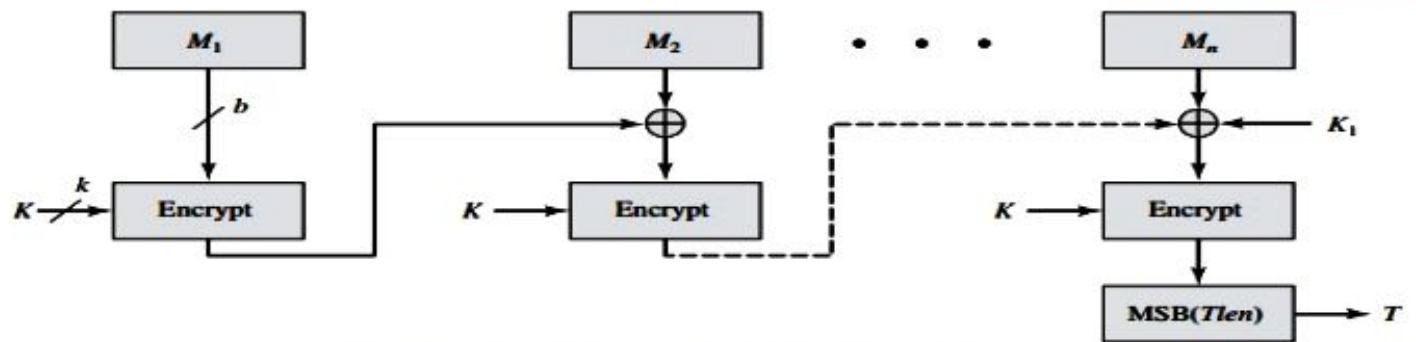
DAA Structure



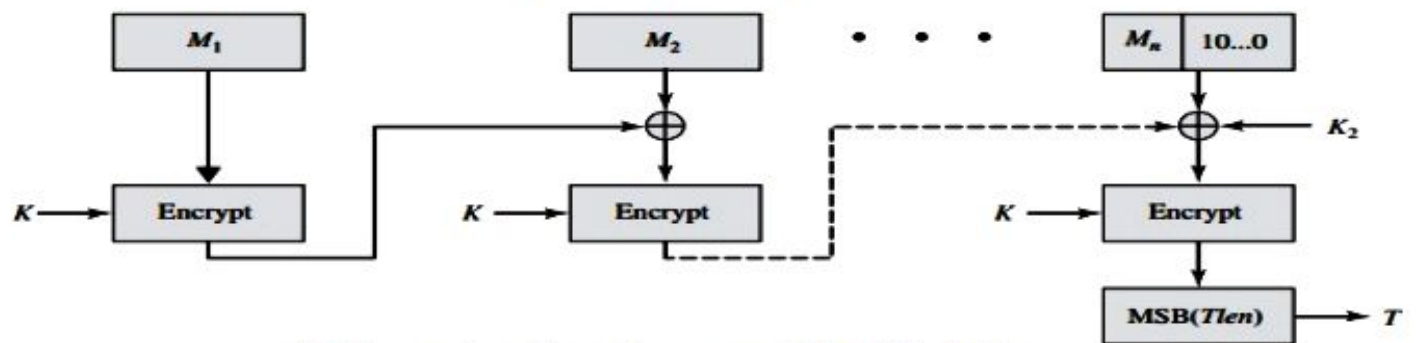
Cipher-Based Message Authentication Code (CMAC)

- ❑ CMAC (Cipher-based Message Authentication Code) is a block cipher based MAC algorithm.
- ❑ It may be used to provide assurance of the authenticity and, hence, the integrity of binary data.
- ❑ This mode of operation fixes security deficiencies of CBC-MAC.

CMAC Structure



(a) Message length is integer multiple of block size



(b) Message length is not integer multiple of block size

CMAC is calculated as follows

$$\begin{aligned}
 C_1 &= E(K, M_1) \\
 C_2 &= E(K, [M_2 \oplus C_1]) \\
 C_3 &= E(K, [M_3 \oplus C_2]) \\
 &\vdots \\
 &\vdots \\
 &\vdots \\
 C_n &= E(K, [M_n \oplus C_{n-1} \oplus K_1]) \\
 T &= \text{MSB}_{Tlen}(C_n)
 \end{aligned}$$

where

T = message authentication code, also referred to as the tag

$Tlen$ = bit length of T

$\text{MSB}_s(X)$ = the s leftmost bits of the bit string X

⌘ Advantages

1. Can use existing encryption functions.
2. Encryption functions have properties that resist pre image and collision attacks.

⌘ Disadvantage

1. Encryption algorithms (particularly when chained) can be much slower than hash algorithms

Digital Signatures

- ❑ Digital signatures are the public-key primitives of message authentication.
- ❑ In the physical world, it is common to use handwritten signatures on handwritten or typed messages.
- ❑ They are used to bind signatory to the message.
- ❑ Similarly, a digital signature is a technique that binds a person/entity to the digital data.
- ❑ This binding can be independently verified by receiver as well as any third party.
- ❑ Digital signature is a cryptographic value that is calculated from the data and a secret key known only by the signer.

Digital Signature Properties

- ❑ 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 & verify
- ❑ Be computationally infeasible to forge
 - ❑ with new message for existing digital signature
 - ❑ with fraudulent digital signature for given message
- ❑ Be practical save digital signature in storage

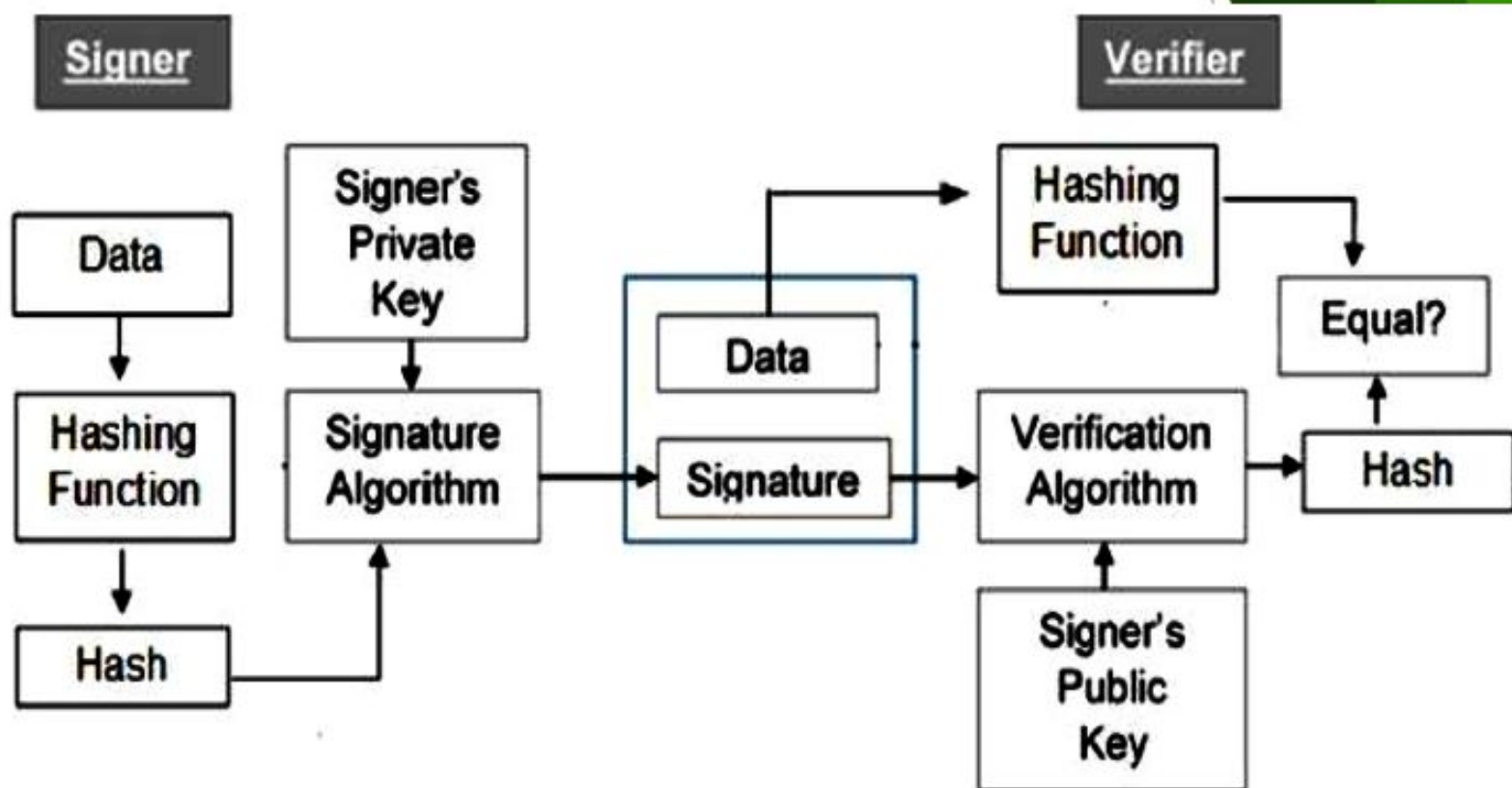
Direct Digital Signatures

- ⊠ Involve only sender & receiver
- ⊠ Assumed receiver has sender's public-key
- ⊠ Digital signature made by sender signing entire message or hash with private-key
- ⊠ Can encrypt using receivers public-key
- ⊠ Important that sign first then encrypt message & signature
- ⊠ Security depends on sender's private-key

Arbitrated Digital Signatures

- ⊠ Involves use of arbiter A
 - ⊠ validates any signed message
 - ⊠ then dated and sent to recipient
- ⊠ Requires suitable level of trust in arbiter
- ⊠ Can be implemented with either private or public-key algorithms
- ⊠ Arbiter may or may not see message

Model of Digital Signature



Process

- Each person adopting this scheme has a public-private key pair.
- Generally, the key pairs used for encryption/decryption and signing/verifying are different. The private key used for signing is referred to as the signature key and the public key as the verification key.
- Signer feeds data to the hash function and generates hash of data.
- Hash value and signature key are then fed to the signature algorithm which produces the digital signature on given hash. Signature is appended to the data and then both are sent to the verifier.

Process

- ❑ Each person adopting this scheme has a public-private key pair.
- ❑ Generally, the key pairs used for encryption/decryption and signing/verifying are different. The private key used for signing is referred to as the signature key and the public key as the verification key.
- ❑ Signer feeds data to the hash function and generates hash of data.
- ❑ Hash value and signature key are then fed to the signature algorithm which produces the digital signature on given hash. Signature is appended to the data and then both are sent to the verifier.

7

- ❑ Verifier feeds the digital signature and the verification key into the verification algorithm. The verification algorithm gives some value as output.
- ❑ Verifier also runs same hash function on received data to generate hash value.
- ❑ For verification, this hash value and output of verification algorithm are compared. Based on the comparison result, verifier decides whether the digital signature is valid.
- ❑ Since digital signature is created by 'private' key of signer and no one else can have this key; the signer cannot repudiate signing the data in future.

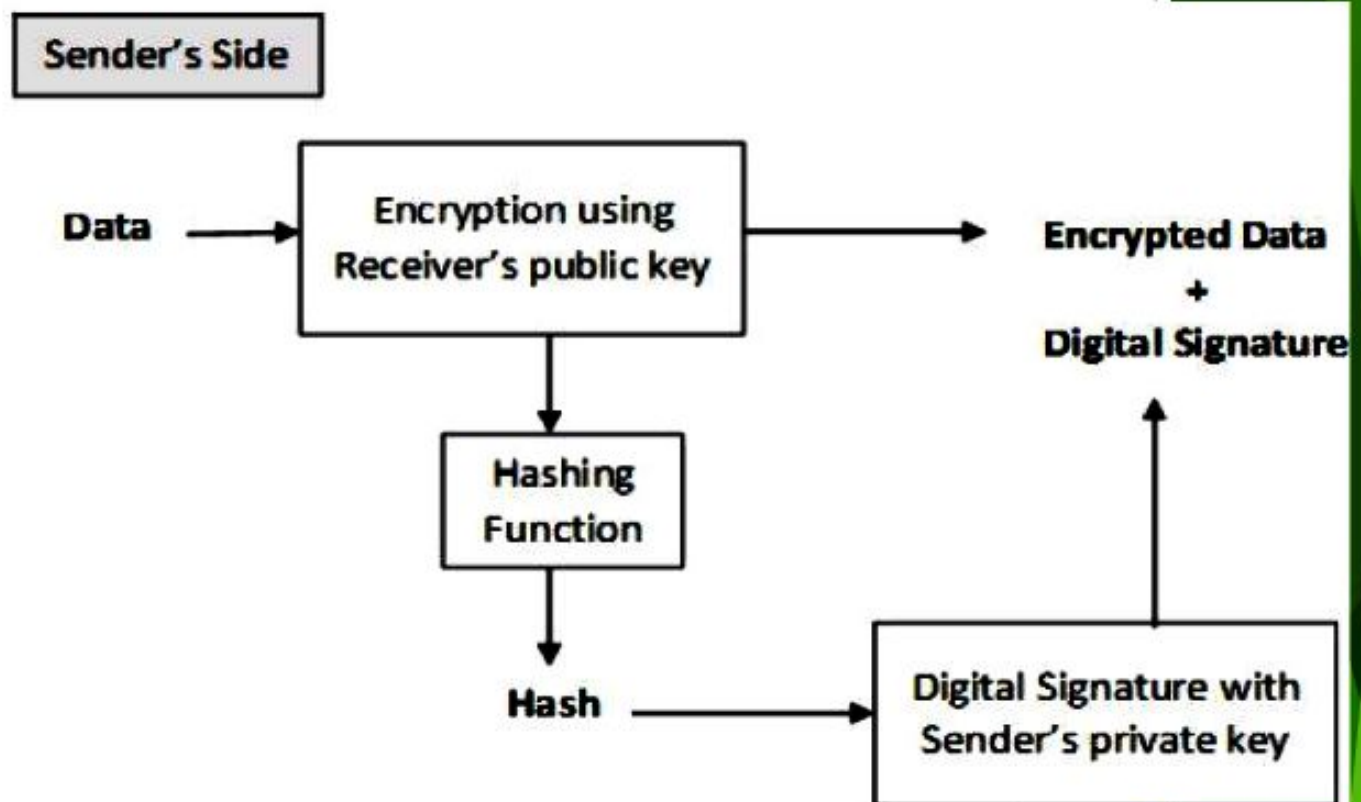
Importance of Digital Signature

- ❏ **Message authentication** – When the verifier validates the digital signature using public key of a sender, he is assured that signature has been created only by sender who possess the corresponding secret private key and no one else.
- ❏ **Data Integrity** – In case an attacker has access to the data and modifies it, the digital signature verification at receiver end fails. The hash of modified data and the output provided by the verification algorithm will not match. Hence, receiver can safely deny the message assuming that data integrity has been breached.
- ❏ **Non-repudiation** – Since it is assumed that only the signer has the knowledge of the signature key, he can only create unique signature on a given data. Thus the receiver can present data and the digital signature to a third party as evidence if any dispute arises in the future.

Encryption with Digital Signature

- ❏ In many digital communications, it is desirable to exchange an encrypted messages than plaintext to achieve confidentiality.
- ❏ In public key encryption scheme, a public (encryption) key of sender is available in open domain, and hence anyone can spoof his identity and send any encrypted message to the receiver.
- ❏ This makes it essential for users employing PKC for encryption to seek digital signatures along with encrypted data to be assured of message authentication and non-repudiation.
- ❏ This can archived by combining digital signatures with encryption scheme.
- ❏ There are **two possibilities**, **sign-then-encrypt** and **encrypt-then-sign**.

Encrypt-Then-Sign



Digital Certificate

- ❏ Digital certificate is issued by a trusted third party which proves sender's identity to the receiver and receiver's identity to the sender.
- ❏ A digital certificate is a certificate issued by a Certificate Authority (CA) to verify the identity of the certificate holder.
- ❏ The CA issues an encrypted digital certificate containing the applicant's public key and a variety of other identification information.
- ❏ Digital certificate is used to attach public key with a particular individual or an entity.

Digital Certificate Contains

- ✘ Name of certificate holder.
- ✘ Serial number which is used to uniquely identify a certificate, the individual or the entity identified by the certificate
- ✘ Expiration dates.
- ✘ Copy of certificate holder's public key.(used for decrypting messages and digital signatures)
- ✘ Digital Signature of the certificate issuing authority.

Digital certificate is also sent with the digital signature and the message.

Digital Certificate v/s Digital Signature

FEATURE	DIGITAL SIGNATURE	DIGITAL CERTIFICATE
1. Basics / Definition	Digital signature is like a fingerprint or an attachment to a digital document that ensures its authenticity and integrity.	Digital certificate is a file that ensures holder's identity and provides security.
2. Process / Steps	Hashed value of original message is encrypted with sender's secret key to generate the digital signature.	It is generated by CA (Certifying Authority) that involves four steps: Key Generation, Registration, Verification, Creation.
3.Security Services	Authenticity of Sender, integrity of the document and non-repudiation .	It provides security and authenticity of certificate holder.
4.Standard	It follows Digital Signature Standard (DSS).	It follows X.509 Standard Format

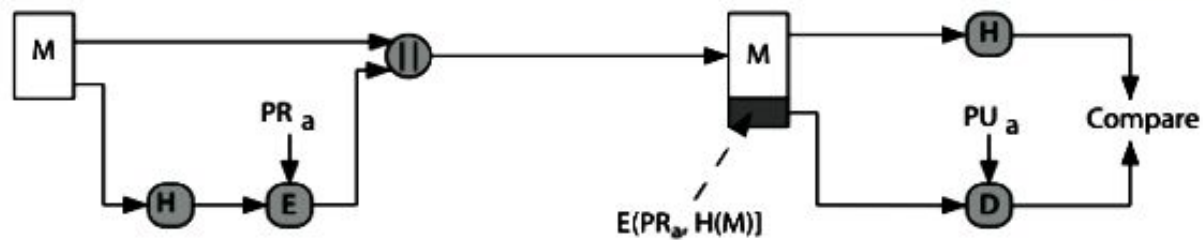
Digital Signature Standard (DSS)

- ❑ US Govt approved signature scheme
- ❑ Designed by NIST & NSA in early 90's
- ❑ Published as FIPS-186 in 1991
- ❑ Revised in 1993, 1996 & then 2000
- ❑ Uses the SHA hash algorithm
- ❑ DSS is the standard, DSA is the algorithm
- ❑ FIPS 186-2 (2000) includes alternative RSA & elliptic curve signature variants

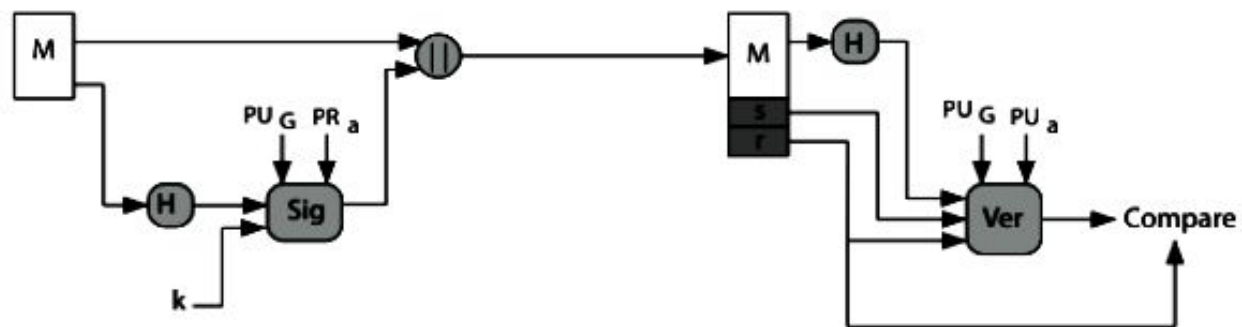
Digital Signature Algorithm (DSA)

- ❑ Creates a 320 bit signature
- ❑ With 512-1024 bit security
- ❑ Smaller and faster than RSA
- ❑ A digital signature scheme only
- ❑ Security depends on difficulty of computing discrete logarithms
- ❑ Variant of ElGamal & Schnorr schemes

Digital Signature Algorithm (DSA)



(a) RSA Approach



(b) DSS Approach

17

DSA Key Generation

- ✧ Have shared global public key values (p, q, g) :
 - ✧ choose q , a 160 bit
 - ✧ choose a large prime $p = 2^L$
 - ✧ where $L = 512$ to 1024 bits and is a multiple of 64
 - ✧ and q is a prime factor of $(p-1)$
 - ✧ choose $g = h^{(p-1)/q}$
 - ✧ where $h < p-1$, $h^{(p-1)/q} \pmod{p} > 1$
- ✧ Users choose private & compute public key:
 - ✧ choose $x < q$
 - ✧ compute $y = g^x \pmod{p}$

DSA Signature Creation

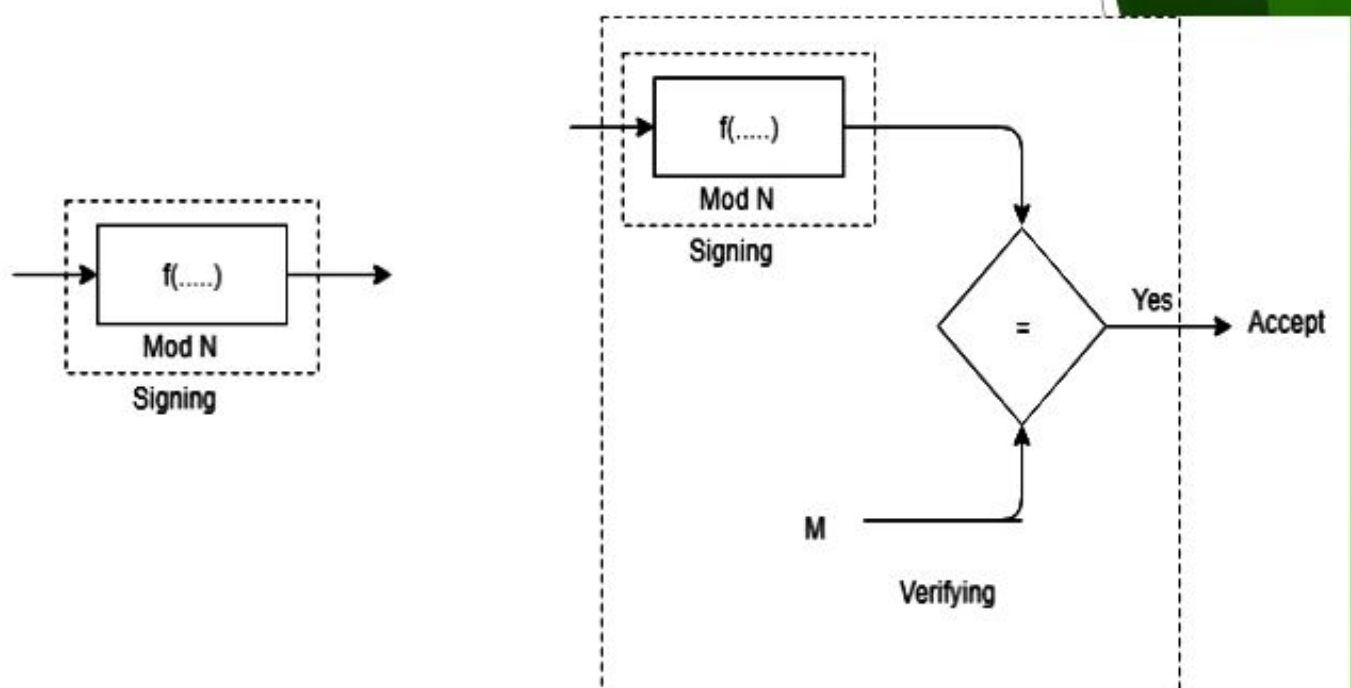
- ✘ To **sign** a message M the sender:
 - ✘ generates a random signature key k, $k < q$
 - ✘ k must be random, be destroyed after use, and never be reused
- ✘ Then computes signature pair:
$$r = (g^k \pmod p) \pmod q$$
$$s = (k^{-1} \cdot H(M) + x \cdot r) \pmod q$$
- ✘ Sends signature (r,s) with message M

DSA Signature Verification

- ✘ Having received M & signature (r,s)
- ✘ To **verify** a signature, recipient computes:
$$w = s^{-1} \pmod q$$
$$u_1 = (H(M) \cdot w) \pmod q$$
$$u_2 = (r \cdot w) \pmod q$$
$$v = (g^{u_1} \cdot y^{u_2} \pmod p) \pmod q$$
- ✘ If $v=r$ then signature is verified

RSA Digital Signature Scheme

- ❑ RSA idea is also used for signing and verifying a message it is called RSA digital signature scheme.
- ❑ Digital signature scheme changes the role of the private and public keys
- ❑ Private and public keys of only the sender are used not the receiver
- ❑ Sender uses her own private key to sign the document and the receiver uses the sender's public key to verify it.
- ❑ The signing and verifying sets use the same function, but with different parameters. The verifier compares the message and the output of the function for congruence. If the result is true the message is accepted.



Key Generation in RSA

Key generation in RSA digital signature scheme is exactly the same as key generation in RSA cryptosystem.

- ☒ Chooses two primes p, q and compute $n = p \cdot q$
- ☒ Compute $\phi(n) = (p-1)(q-1)$
- ☒ Chooses e with $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$, (e is prime)
- ☒ Solves $d \times e \bmod \phi(n) = 1$
- ☒ Public Key (e, n) , Private Key (d, n)

4

Working of RSA digital signature scheme

Sender A wants to send a message M to the receiver B along with the digital signature S calculated over the message M

- ☒ Step 1: The sender A uses the message digest algorithm to calculate the message digest $MD1$ over the original message M
- ☒ Step 2: The sender A now encrypts the message digest with her private key. The output of this process is called the digital signature.
- ☒ Step 4: After the receiver B receives the original message M and the sender A's digital signature, B uses the same message digest algorithm which was used by A and calculate its own message digest $MD2$.
- ☒ Step 5: The receiver B now uses the sender's A's public key to decrypt the digital signature. Note that A had used his private key to decrypt the message digest $MD1$ to form the digital signature. Therefore only A's public key can be used to decrypt it. The output of this process is the original message digest which was calculated by A ($MD1$) in step 1.

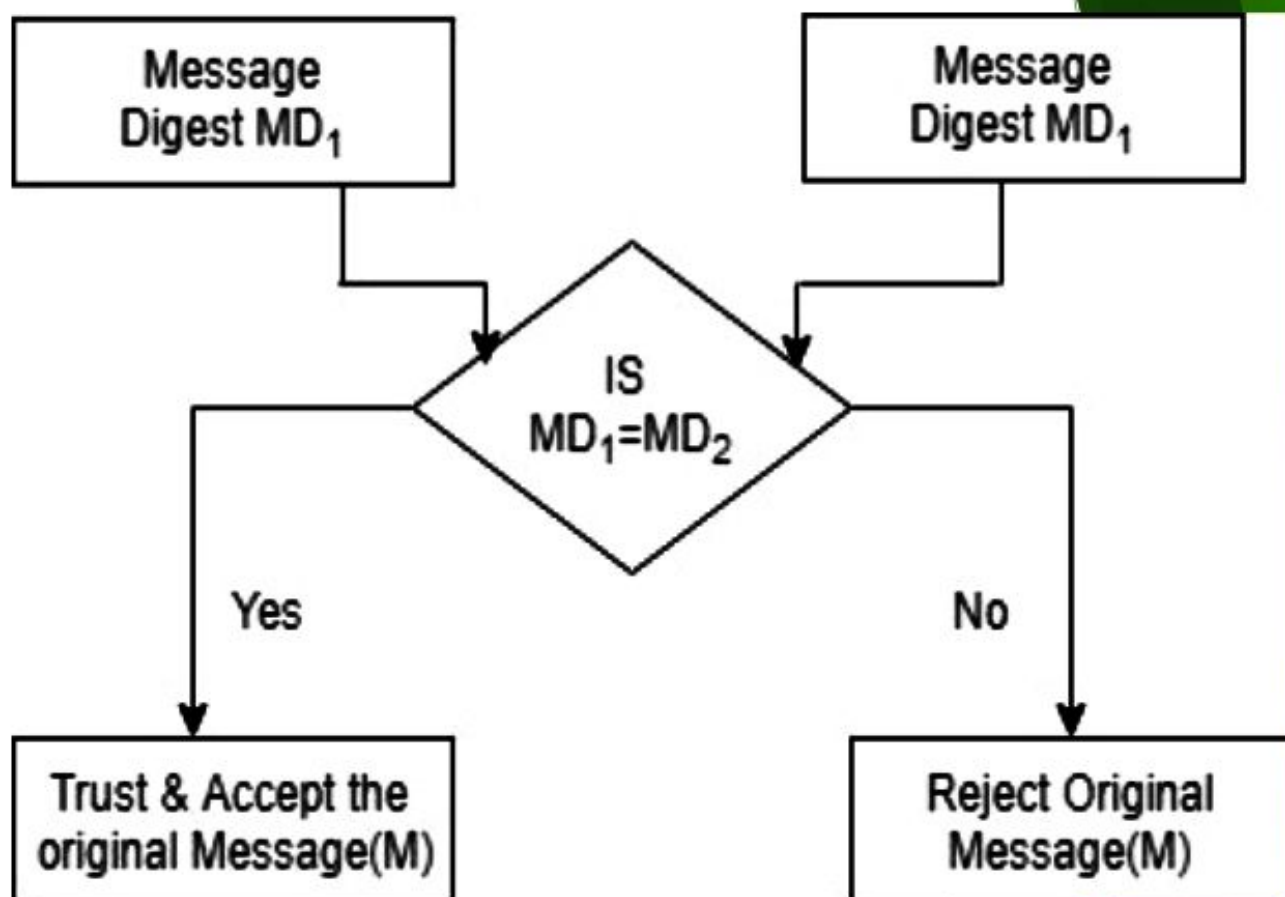
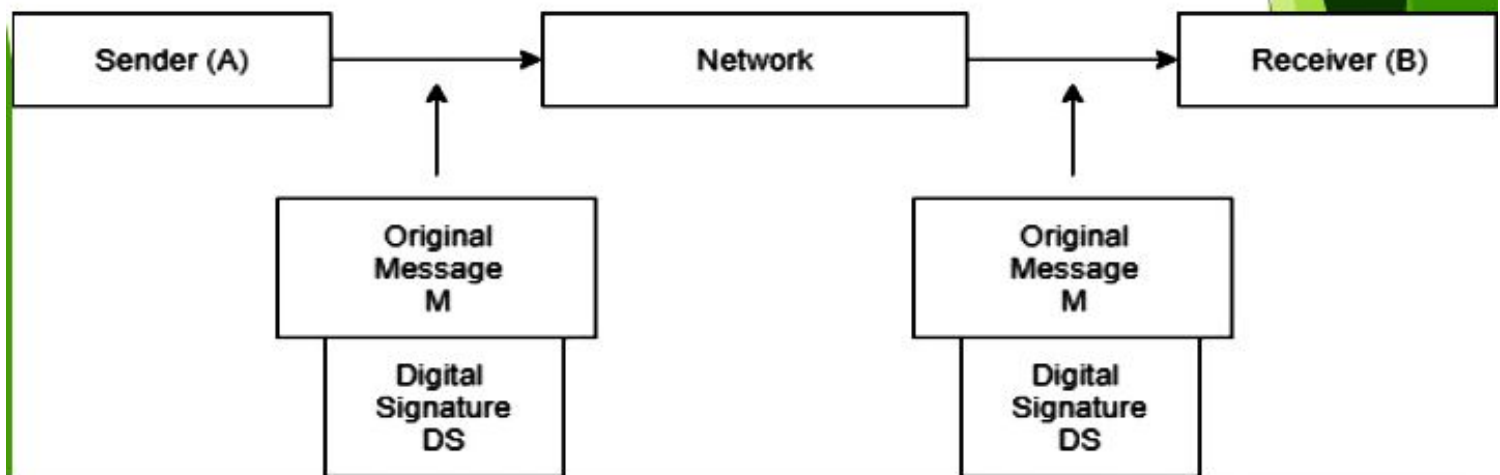
Step 6: B now compare the following two message digests.

MD2, which it had calculated in step 4

MD1, which is retrieved from A's digital signature in step 5

If $MD1 = MD2$ the following facts are established:

- B accepts the original message (M) as the correct, unaltered message from A
- B is also assured that the message came from A and not from someone else attached, posing as A



ElGamal Digital Signature Scheme

- ✘ This scheme used the same keys but a different algorithm
- ✘ The algorithm creates two digital signatures, these two signatures, a
- ✘ The key generation process is the same as that of El-gamal algorithms are used in the verification phase.
- ✘ The Public Key (e_1, e_2, p) and private key d .

Signature

This process works as follows

- ✘ The sender selects a random number r
- ✘ The sender computes the first signature s_1 using
$$s_1 = e_1^r \bmod p$$
- ✘ The sender computes the second signature s_2 using the equation
$$s_2 = (M - d \times s_1) \times r^{-1} \bmod (p-1)$$

Where P = large prime number

M = original message that needs to be signed

- ✘ The sender sends M, s_1, s_2 to the receiver.

For eg. Let $e_1 = 10, e_2 = 4, p = 19, M = 14, d = 16$ & $r = 5$

Then, $s_1 = e_1^r \bmod p = 10^5 \bmod 19 = 3$

$s_2 = (M - d \times s_1) \times r^{-1} \bmod (p-1)$

$= (14 - 16 \times 3) \times 5^{-1} \bmod (18) = 4$

- ✘ Then these signatures s_1 and s_2 are sent to the receiver.

$$s_2 = (M - d \times s_1) \times r^{p-2} \bmod (p-1)$$

Where P = large prime number

M = original message that needs to be signed

The sender sends M, s_1, s_2 to the receiver.

For eg. Let $e_1 = 10$, $e_2 = 4$, $p=19$, $M=14$, $d=16$ & $r=5$

Then, $s_1 = e_1^r \bmod p = 10^5 \bmod 19 = 3$

$$s_2 = (M - d \times s_1) \times r^{p-2} \bmod (p-1)$$

$$= (14 - 16 \times 3) \times 5^{18} \bmod (18) = 4$$

Then these signatures s_1 and s_2 are sent to the receiver.

Verification

This process works as follows.

The receiver performs the 1st part of verification called v_1 using the equation

$$v_1 = e_1^M \bmod p$$

The receiver performs the 2nd part of verification called as v_2 using the equation

$$v_2 = e_2^{s_1} s_1^{(s_2)} \bmod p$$

Eg.

$$v_1 = e_1^M \bmod p = 10^{14} \bmod 19 = 16$$

and

$$v_2 = e_2^{s_1} s_1^{(s_2)} \bmod p = 4^3 3^4 \bmod p = 5184 \bmod 19 = 16$$

Since $v_1 = v_2$, the signature is valid.