HASH FUNCTIONS

- A **hash function** is a function that takes an input (or 'message') and returns a fixed-size string of bytes.
- The output, typically a number, is called the **hash code** or **hash value**.
- The main purpose of a hash function is to efficiently map data of arbitrary size to fixed-size values
- A hash function transforms one numerical input value into another compressed numerical value.
- It is also a process that turns plaintext data of any size into a unique ciphertext of a predetermined length.
- A **cryptographic hash function (CHF)** is an equation that is widely used to verify the validity of data.
- It has many applications, particularly in information security (e.g. user authentication).
- A CHF translates data of various lengths of the message into a fixed-size numerical string the hash.
- A cryptographic **hash function is a single-directional work**, making it extremely difficult to reverse to recreate the information used to make it.

How Does a Cryptography Hash Function Work?

- The hash function accepts data of a fixed length. The data block size varies between algorithms.
- If the blocks are too small, padding may be used to fill the space. However, regardless of the kind of hashing used, the output, or hash value, always has the same set length.
- The hash function is then applied as many times as the number of data blocks.

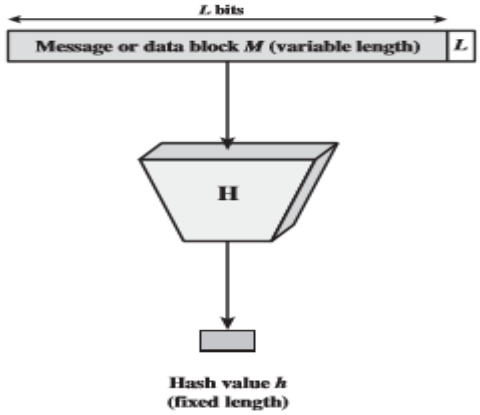


Figure 11.1 Black Diagram of Cryptographic Hash Function; $h = H(M)$

KEY PROPERTIES OF HASH FUNCTIONS

- **Deterministic**: A hash function must consistently produce the same output for the same input.
- **Fixed Output Size**: The output of a hash function should have a fixed size, regardless of the size of the input.
- **Efficiency**: The hash function should be able to process input quickly.
- **Uniformity**: The hash function should distribute the hash values uniformly across the output space to avoid clustering.
- **Pre-image Resistance**: It should be computationally infeasible to reverse the hash function, i.e., to find the original input given a hash value.
- **Collision Resistance**: It should be difficult to find two different inputs that produce the same hash value.
- **Avalanche Effect**: A small change in the input should produce a significantly different hash value.

APPLICATIONS OF CRYPTOGRAPHIC HASH FUNCTIONS

1. MESSAGE AUTHENTICATION

- Message authentication is a system or service that verifies the integrity of a communication.
- It ensures data is received precisely as transmitted, with no modifications, insertions, or deletions, a hash function is used for message authentication, and the value is sometimes referred to as a **message digest**.
- Message authentication often involves employing a message authentication code (MAC).
- MACs are widely used between two parties that share a secret key for authentication purposes.
- A MAC function uses a secret key and data block to generate a hash value, that identifies the protected communication
- A variety of ways in which a hash code can be used to provide message authentication are:

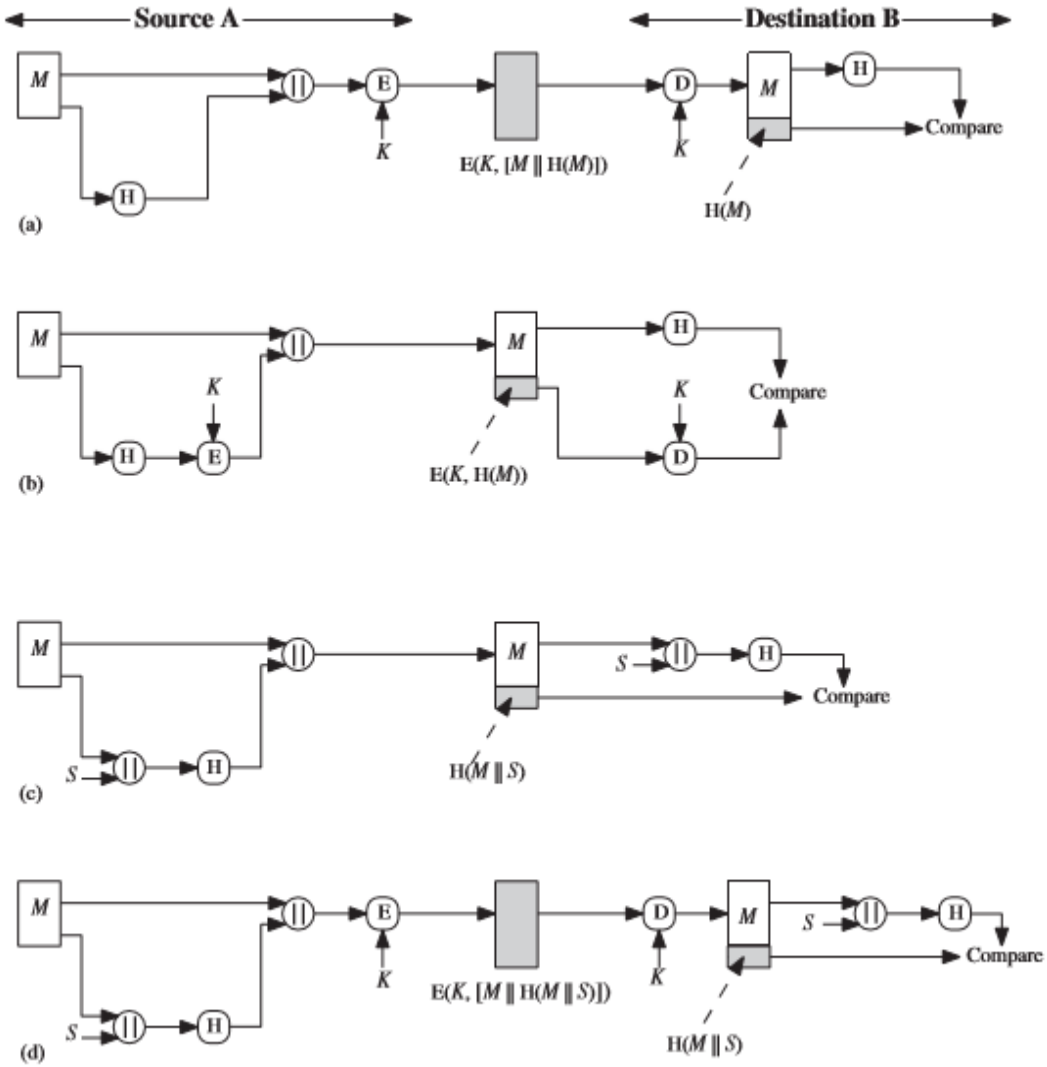


Figure 11.2 Simplified Examples of the Use of a Hash Function for Message Authentication

a) The message plus concatenated hash code is encrypted using symmetric encryption. Because only A and B share the secret key, the message must have come from A and has not been altered. The hash code provides the structure or redundancy required to achieve authentication. Because encryption is applied to the entire message plus hash code, confidentiality is also provided.

b) Only the hash code is encrypted, using symmetric encryption. This reduces the processing burden for those applications that do not require confidentiality.

c) It is possible to use a hash function but no encryption for message authentication. The technique assumes that the two communicating parties share a common secret value .A computes the hash value over the concatenation of and and appends the resulting hash value to .Because B possesses ,it can recompute the hash value to verify.Because the secret value itself is not sent,an opponent cannot modify an intercepted message and cannot generate a false message.

d) Confidentiality can be added to the approach of method (c) by encrypting the entire message plus the hash code

2. DIGITAL SIGNATURES

- The digital signature application is comparable to message authentication.
- Digital signatures operate similarly to MACs.
- Digital signatures encrypt message hash values using a user's private key.
- The digital signature may be verified by anybody who knows the user's public key.

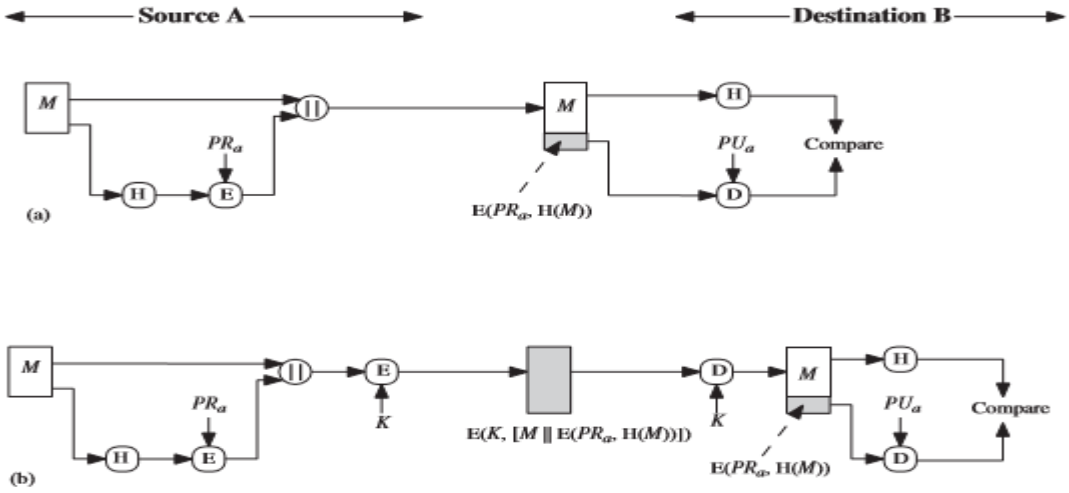


Figure 11.3 Simplified Examples of Digital Signatures

3. DATA INTEGRITY CHECK

- Hash functions are most commonly used to create checksums for data files.
- This program offers the user with assurance that the data is correct.
- The integrity check allows the user to detect any modifications to the original file.
- It does not assure uniqueness. Instead of altering file data, the attacker can update the entire file, compute a new hash, and deliver it to the recipient.

4. ONE-WAY PASSWORD FILE

- Hash functions are commonly used to create a one-way password file
- A scheme in which a hash of a password is stored by an operating system rather than the password itself. Thus, the actual password is not retrievable by a hacker who gains access to the password file.
- In simple terms, when a user enters a password, the hash of that password is compared to the stored hash value for verification. This approach to password protection is used by most operating systems.
- Hash functions can be used for intrusion detection and virus detection

SECURITY REQUIREMENTS FOR CRYPTOGRAPHIC HASH FUNCTIONS / PROPERTIES

Requirement	Description
Variable input size	H can be applied to a block of data of any size.
Fixed output size	H produces a fixed-length output.
Efficiency	$H(x)$ is relatively easy to compute for any given x, making both hardware and software implementations practical.
Preimage resistant (one-way property)	For any given hash value h, it is computationally infeasible to find y such that $H(y) = h$.
Second preimage resistant (weak collision resistant)	For any given block x, it is computationally infeasible to find $y \neq x$ with $H(y) = H(x)$.
Collision resistant (strong collision resistant)	It is computationally infeasible to find any pair (x, y) such that $H(x) = H(y)$.
Pseudorandomness	Output of H meets standard tests for pseudorandomness.

- The **first three properties** are requirements for the practical application of a hash function.
- The **fourth property, preimage resistant, is the one-way property**: it is easy to generate a code given a message, but virtually impossible to generate a message given a code. This property is important if the authentication technique involves the use of a secret value .
- However, if the hash function is not one way,an attacker can easily discover the secret value :If the attacker can observe or intercept a transmission, the attacker obtains the message ,and the hash code .The attacker then inverts the hash function to obtain .Because the attacker now has both and ,it is a trivial matter to recover .
- The **fifth property, second preimage resistant,** guarantees that it is impossible to find an alternative message with the same hash value as a given message. 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 message plus its encrypted hash code;
    - second, generate an unencrypted hash code from the message;
    - third, generate an alternate message with the same hash code.

A hash function that satisfies the first five properties is referred to as a weak hash function.

- The **sixth property, collision resistant**,is also satisfied, then **it is referred to as a strong hash function**.
- A strong hash function protects against an attack in which one party generates a message for another party to sign.
- The **final requirement, pseudorandomness**, has not traditionally been listed as a requirement of cryptographic hash functions but is more or less implied.

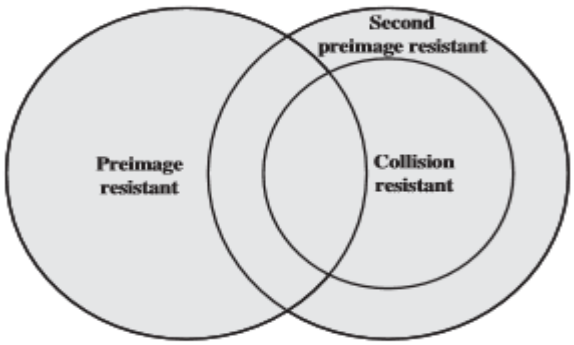


Figure 11.5 Relationship Among Hash Function Properties

- A function that is collision resistant is also second preimage resistant, but the reverse is not necessarily true.
- A function can be collision resistant but not preimage resistant and vice versa.
- A function can be collision resistant but not second preimage resistant and vice versa.

ATTACKS ON HASH FUNCTIONS

As with encryption algorithms, there are two categories of attacks on hash functions: **brute-force attacks and cryptanalysis.**

A **brute-force attack** does not depend on the specific algorithm but depends only on bit length. In the case of a hash function, a brute-force attack depends only on the bit length of the hash value.

A **cryptanalysis**, in contrast,is an attack based on weaknesses in a particular cryptographic algorithm.

Here are some types of brute force attacks on hash functions:

One-way function inversion

An attacker evaluates the hash function with every possible input until they find the desired output hash value

Preimage And Second Preimage Attacks

- For a preimage or second preimage attack,an adversary wishes to find a value such that H(y) is equal to a given hash value ,h .
- The brute-force method is to pick values of y at random and try each value until a collision occurs.
- For an m-bit hash value,the level of effort is proportional to 2^m. Specifically,the adversary would have to try,on average,2^{m-1} values of y to find one that generates a given hash value h .
- Ie, An attacker evaluates the hash function with every possible input until they find a second input that maps to the same output as a given input.

Collision Resistant Attacks

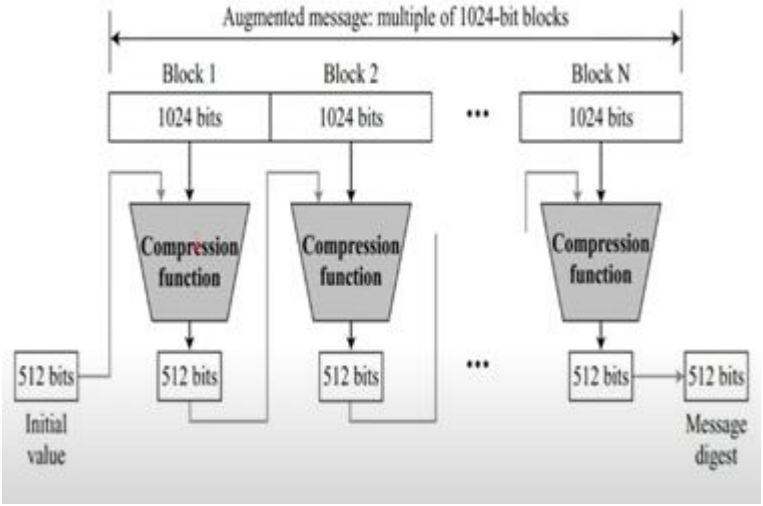
- For a collision resistant attack, an adversary wishes to find two messages or data blocks, x and y , that yield the same hash function: H(x) =H(y). This turns out to require considerably less effort than a preimage or second preimage attack
- Ie, An attacker tries to find two inputs that map to the same output. A variation of this attack, called a Birthday attack, is based on the Birthday Paradox.

SHA ALGORITHM

- The most widely used hash function has been the Secure Hash Algorithm (SHA).
- SHA was developed by the National Institute of Standards and Technology (NIST)
- The actual standards document is entitled "**Secure Hash Standard**."
- SHA is based on the hash function MD4, and its design closely models MD4

SHA 512

- SHA 512 is the version of SHA with 512 bit message digest.
- SHA 512 creates a digest of 512 bits from a multiple block message.
- Each block is 1024 bits in length.
- The algorithm takes as input a message with a maximum length of less than bits and produces as output a 512-bit message digest.
- The input is processed in 1024-bit blocks



- Message digest is initialized to a predetermined value of 512 bits
- Algorithm mixes the initial value with the first block of message to create the first intermediate message digests of 512 bits.
- Digest is then mixed with the second block to create the second intermediate digest.
- Finally , the (N-1) th digest is mixed with the Nth block to create the Nth digest.
- When the last block is processed, the resulting digest is the message digest for the entire message.

STEPS IN SHA 512

1. **Message Preparation**

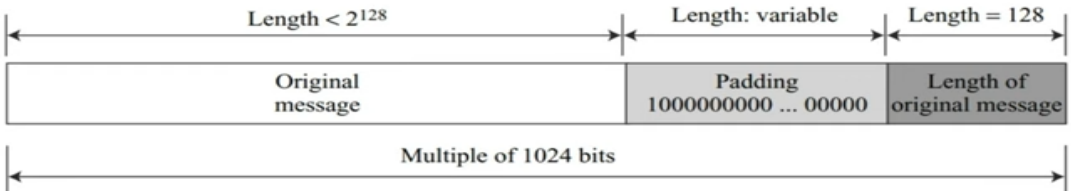
   - SHA 512 insists that the length of the original message be less than 2^{128} bits

2. **Append padding bits**.

   - The message is padded so that its length is congruent to 896 modulo 1024 .
   - Padding is always added, even if the message is already of the desired length.
   - Thus, the number of padding bits is in the range of 1 to 1024.
   - The padding consists of a single 1 bit followed by the necessary number of 0 bits.

3. **Append length.**

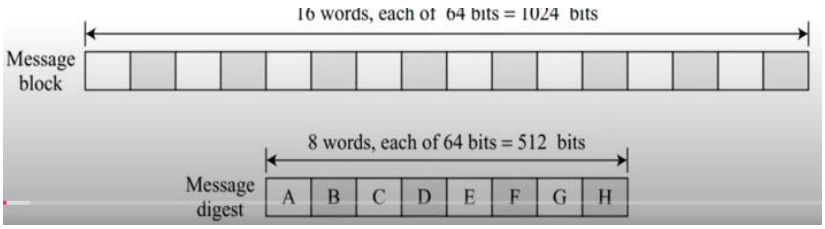
- A block of 128 bits is appended to the message.
- This block is treated as an unsigned 128-bit integer (most significant byte first) and contains the length of the original message (before the padding).
- The outcome of the first two steps yields a message that is an integer multiple of 1024 bits in length.
- So that the total length of the expanded message is N x 1024 bits



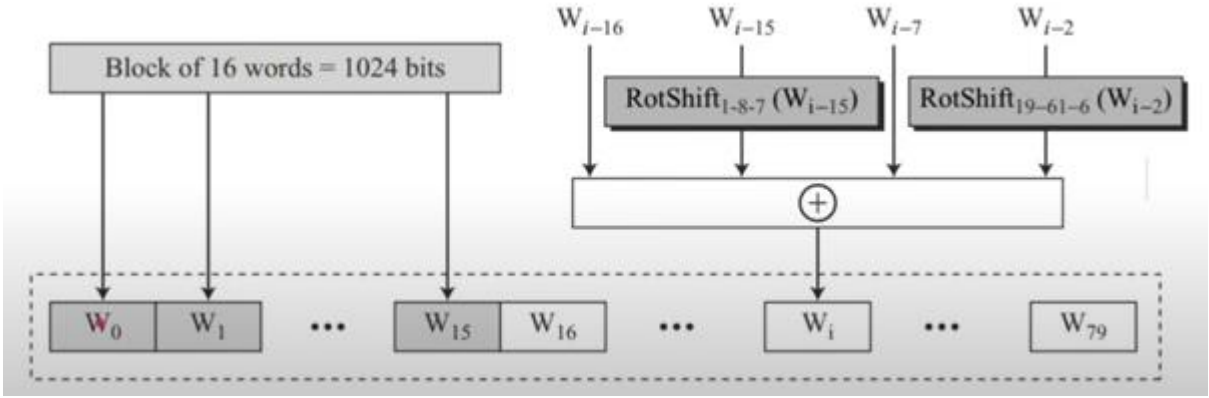
- **SHA 512 operates on words . ie,** it is word oriented
- **A** word is defined as 64 bits
- After the padding and length field are added to the message, each block of the message consists of **sixteen – 64 bit words.**

4. **Initialize hash buffer.**

- A 512-bit buffer is used to hold intermediate and final results of the hash function.
- The buffer can be represented as eight 64-bit registers (A,B,C,D,E,F,G,H).
- These registers are initialized to 64-bit integers (hexadecimal values).
- These words were obtained by taking the first sixty-four bits of the fractional parts of the square roots of the first eight prime numbers.

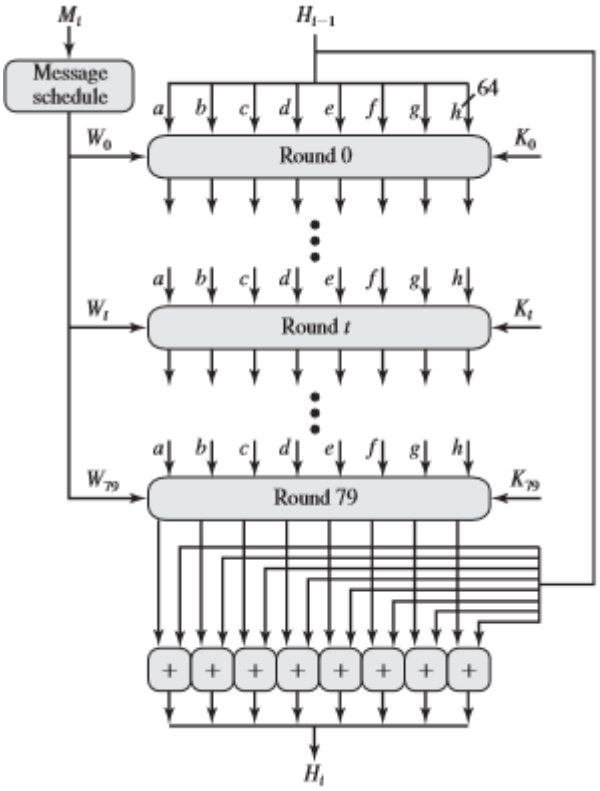


- 16 word block needs to be expanded to 80 words , from W0 to W79
- 1024 bit block be the first 16 words, rest of the words come from already made words according to the operation.



5. **Compression Function**

- The heart of the algorithm is a module that consists of 80 rounds
- Each round takes as input the 512-bit buffer value, A B C D E F G H ,and updates the contents of the buffer.
- At input to the first round,the buffer has the value of the intermediate hash value, H_{i-1} .
- Each round t makes use of a 64-bit value W_i, derived from the current 1024-bit block being processed .
- These values are derived using a message schedule described subsequently.
- Each round also makes use of an additive constant K_i , indicates one of the 80 rounds.
- The output of the eightieth round is added to the input to the first round to produce H_i.
- The addition is done independently for each of the eight words in the buffer



6. **Output.**

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

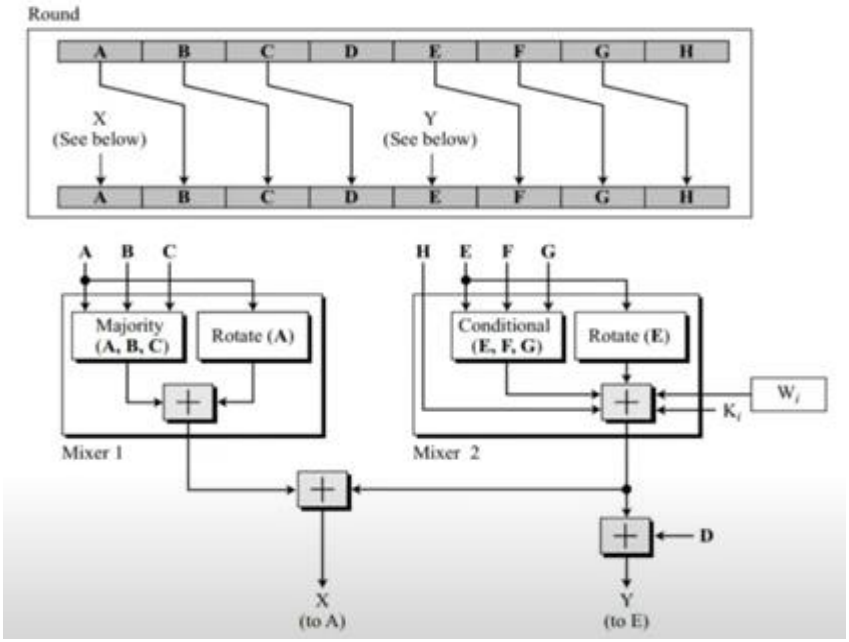
SHA-512 Round Function

- In each round, eight new values for the 64 bit buffers are created from the values of the buffers in the previous round.
- Six buffers are the exact copies of one of the buffers in the previous round.

  A→B , B→C , C→D , E→F , F→G , G→H

- Two of the new buffers, A and E, receive their inputs from some complex functions.
- And that involves some of the previous buffers , the corresponding word for this round W_i, and the corresponding constant for this round K_i
- There are two mixers , three functions ,and several operators.
- Each mixer combines two functions:-
  - Majority function is a bitwise function, and it takes corresponding bits of A,B,C
  - Conditional function is also a bitwise function, and it takes corresponding bits of E,F,G

- Rotate function, right rotate the 3 instances of same buffer (A or E) and applies the XOR operation on the results.



SHA 512

Message Digest Size	512
Message Size	$< 2^{128}$
Block Size	1024
Word Size	64
Number of Steps	80

MESSAGE AUTHENTICATION CODE

- Message authentication is a mechanism or service used to verify the integrity of a message.
- Message authentication assures that data received are exactly as sent by (i.e.,contain no modification, insertion, deletion, or replay) and that the purported identity of the sender is valid.
- Symmetric encryption provides authentication among those who share the secret key.
- A **message authentication code (MAC)** is an algorithm that requires the use of a secret key.
- A **MAC takes a variable-length message and a secret key as input and produces an authentication code**.
- A recipient in possession of the secret key can generate an authentication code to verify the integrity of the message.

It involves the use of a secret key to generate a small fixed size block of data, known as **cryptographic checksum or MAC**, that is appended to the message.

Assumes two communicating parties , A and B, share a common secret key. When A has a message to send to B, it calculates the MAC as a function of the message and the key:

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

Where, M = input message

C = MAC function

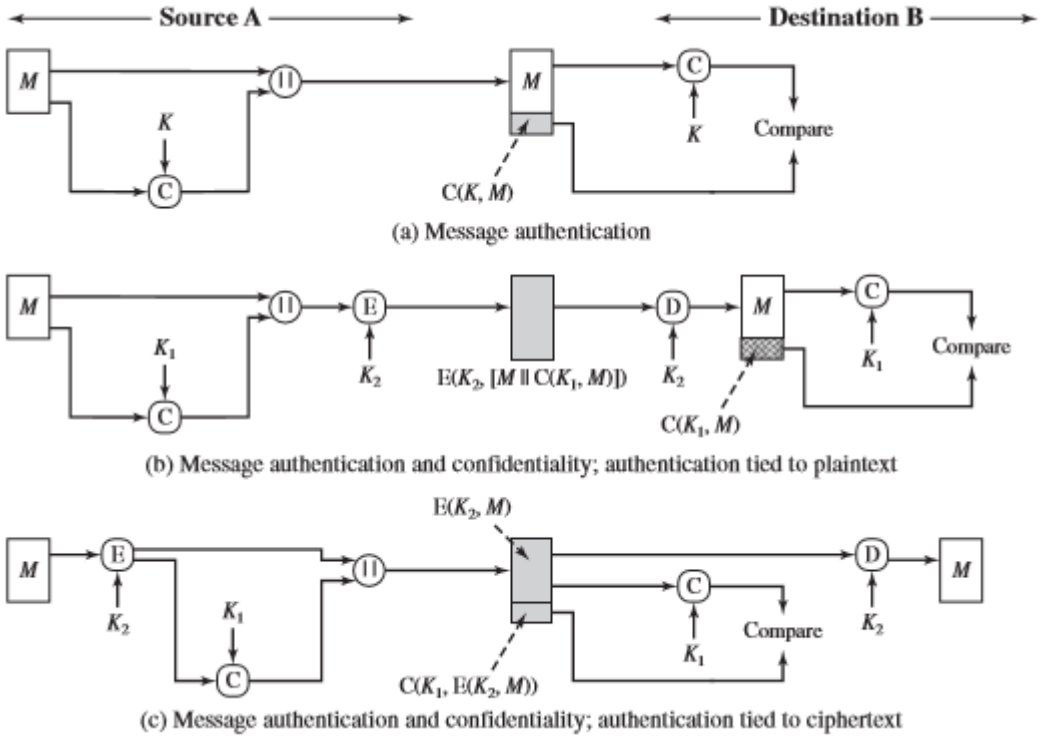
K = shared secret key

MAC = Message Authentication Code

Message plus MAC are transmitted to the intended recipient. Recipient performs the same calculations on the received message, using the same secret key, to generate a new MAC. Then received MAC is compared to the calculated MAC.

If both the MAC values match:

- The receiver is assured that the message has not been altered
- The receiver is assured that the message is from the alleged sender.
- If the message includes a sequence number then the receiver can be assured of the proper sequence.



MAC – REQUIREMENTS

- If an opponent observes M and MAC (K,M), it should be computationally infeasible for the opponent to construct a message M' such that
  MAC(K, M') = MAC (K,M)

First Requirement **:**

- An opponent must not able to reconstruct a new message to match a given tag, even though the opponent does not know and does not learn the key.

- MAC (K,M) should be uniformly distributed in the sense that for randomly chosen message, M and M', the probability that MAC (K,M) = MAC (K,M') is 2^{-n} , where n is the number of bits in the tag

Second requirement :

- Deals with the need to thwart a brute-force attack based on chosen plaintext

- If we assume that the opponent does not know K but does have access to the MAC function and can present messages for MAC generation, then the opponent could try various messages until finding one that matches a given tag
- If the MAC function exhibits uniform distribution, then a brute-force method would require, on average 2^{n-1} attempts before finding a message that fits a given tag

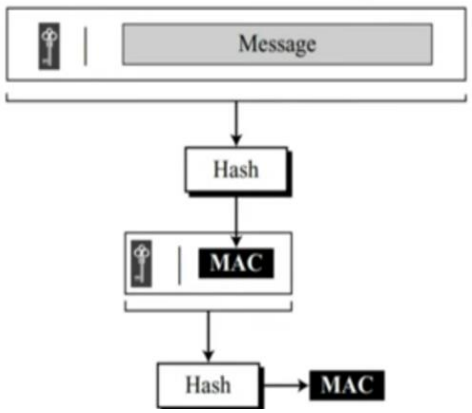
    - Let M' be equal to some known transformation on M.
      Ie, M' = f (M)
      Pr [MAC (K,M) = MAC (K ,M')] = 2^{-n}

MAC PROTOCOLS

HMAC (Keyed-Hash Message Authentication Code) and CMAC (Cipher-based Message Authentication Code) are both types of message authentication codes that **provide message integrity and authenticity**. They are commonly used in cryptography and computer security applications to verify that a message has not been tampered with and that it came from a trusted sender

Nested MAC

To improve the security of a MAC, nested MACs were designed in which hashing is done in two steps. In first step, the key is concatenated with the message and is hashed to create an intermediate digest. In second step, the key is concatenated with the intermediate digest to create the final digest



HMAC

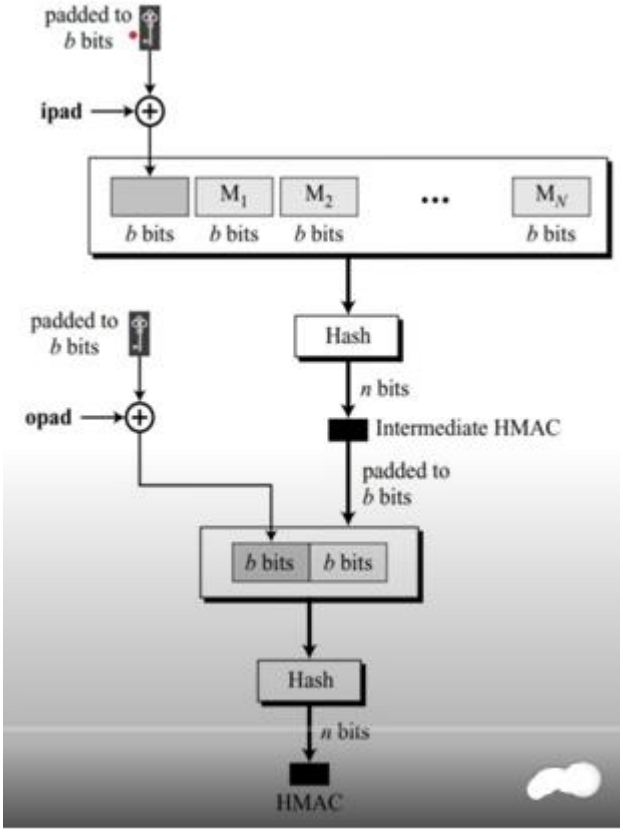
- **HMAC (Hash-Based Message Authentication Code)** is a cryptographic technique that ensures data integrity and authenticity using a hash function and a secret key. Unlike approaches based on signatures and asymmetric cryptography. Checking data integrity is necessary for the parties involved in communication.
- HTTPS, SFTP, FTPS, and other transfer protocols use HMAC.
- Digital signatures are nearly similar to HMACs i.e. they both employ a hash function and a shared key. The difference lies in the keys i.e. HMAC uses a symmetric key(same copy) while Signatures uses an asymmetric (two different keys).

Working of Hash-based Message Authentication Code

- HMACs provides client and server with a shared private key that is known only to them. The client makes a unique hash (HMAC) for every request. When the client requests the server, it hashes the requested data with a private key and sends it as a part of the request. Both the message and key are hashed in separate steps making it secure. When the server receives the request, it makes its own HMAC. Both the HMACS are compared and if both are equal, the client is considered legitimate.

- The formula for HMAC**:**

HMAC = hashFunc (secret key + message)



Algorithm

1. The message is divided into N blocks, each of b bits
2. The secret key is left padded with 0's to create a b bit key
   Note that it is recommended that is the secret key(before padding) be longer than n bits, where n is the size of the HMAC.
3. The result of step 2 is X-ORed with a constant called ipad (input pad) to create a b bit block. The value of ipad is b/8 repetition 36 in hexadecimal
   The b/8 repetition refers that repeating this sequence "b" times where b represents the block size divided by 8.
4. The resulting block is prepended to the N block message. The result is N+1 blocks
5. The result of step 4 is hashed to create an n bit digest, called as intermediate HMAC
6. The intermediate n bit HMAC is left padded with 0's to make a b bit block
7. Steps 2 and 3 are repeated by a different constant opad (output pad)
8. The result of step 7 is prepended to the block of step 6
9. The result of step 8 is hashed with the same hashing algorithm to create the final n bit HMAC.

Security of HMAC

- The security of any MAC function based on an embedded hash function depends in some way on the cryptographic strength of the underlying hash function. The appeal of HMAC is that its designers have been able to prove an exact relationship between the strength of the embedded hash function and the strength of HMAC.
- The probability of successful attack on HMAC is equivalent to one of the following attacks on the embedded hash function.

1. The attacker is able to compute an output of the compression function even with an that is random, secret ,and unknown to the attacker.

2. The attacker finds collisions in the hash function even when the is random and secret.

- HMAC provides higher security than traditional MACs due to its two-step hashing process, making it resistant to certain types of attacks.
- Despite challenges like key management and potential hash collisions, HMAC remains a robust and efficient method for securing data in various applications, including email verification, IoT, and password reset mechanisms.

Advantages of HMAC

- HMACs are ideal for high-performance systems like routers due to the use of hash functions which are calculated and verified quickly unlike the public key systems.
- Digital signatures are larger than HMACs, yet the HMACs provide comparably higher security.
- HMACs are used in administrations where public key systems are prohibited.

Disadvantages of HMAC

- HMACs uses shared key which may lead to non-repudiation. If either sender or receiver's key is compromised then it will be easy for attackers to create unauthorized messages.
- Securely managing and distributing secret keys can be challenging.
- Although unlikely, hash collisions (where two different messages produce the same hash) can occur.
- The security of HMAC depends on the length of the secret key. Short keys are more vulnerable to brute-force attacks.
- The security of HMAC relies on the strength of the chosen hash function (e.g., SHA-256). If the hash function is compromised, HMAC is also affected.

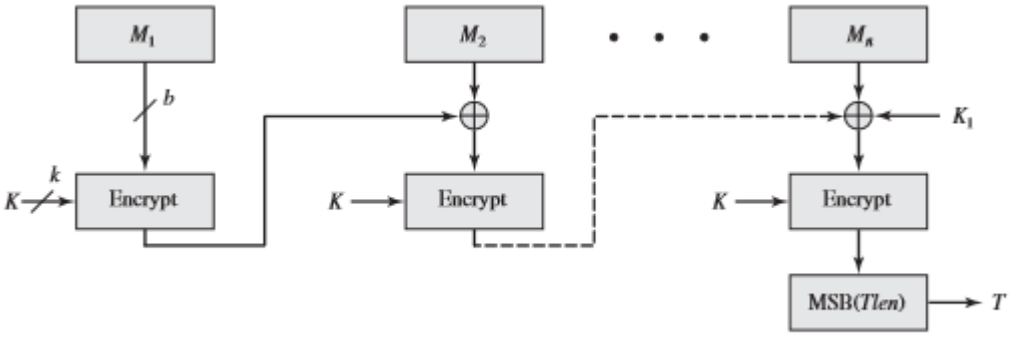
Applications of HMAC

- Verification of e-mail address during activation or creation of an account.
- Authentication of form data that is sent to the client browser and then submitted back.
- HMACs can be used for Internet of things (IoT) due to less cost.
- Whenever there is a need to reset the password, a link that can be used once is sent without adding a server state.
- It can take a message of any length and convert it into a fixed-length message digest. That is even if you got a long message, the message digest will be small and thus permits maximizing bandwidth.

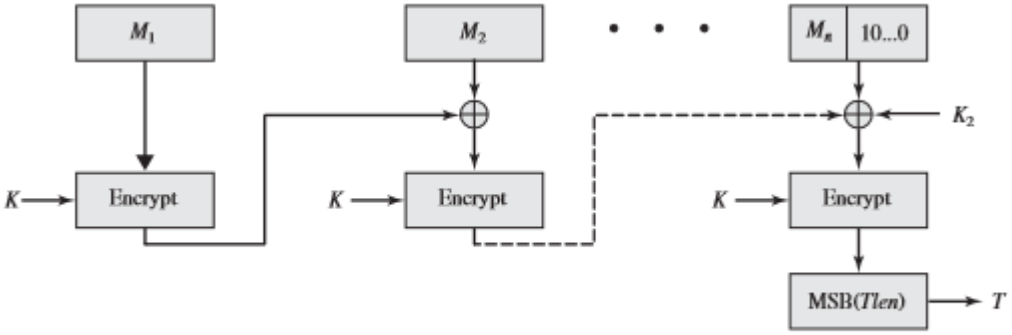
CMAC

- **Cipher-based message authentication codes** (or CMACs) are a tool for calculating message authentication codes using a block cipher coupled with a secret key.
- We use a CMAC to verify both the integrity and authenticity of a message.
- Mainly it can be use with AES and 3DES.
- It is similar to the cipher block chaining (CBC) mode
- The idea of CMAC is to create one block of MAC from N blocks of plaintext using a symmetric key cipher N times.
- First ,we define the operation of CMAC when the message is an integer multiple n of the cipher block length b.

- The message is divided into blocks (M1, M2, M3…. Mn)
- The algorithm makes use of a k bit encryption key K and an n-bit constant, K1.



(a) Message length is integer multiple of block size



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

Figure 12.8 Cipher-Based Message Authentication Code (CMAC)

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

where

T = message authetication code, also referred to as the tag
$Tlen$ = bit length of T
$\mathrm{MSB}_s(X)$ = the s leftmost bits of the bit string X

- If the message is not an integer multiple of the cipher block length, then the final block is padded to the right (least significant bits) with a 1 and as many 0s as necessary so that the final block is also of length b .
- The CMAC operation then proceeds as before, except that a different n -bit key K2 is used instead of K1

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

$$L = \mathrm{E}(K, 0^n)$$
$$K_1 = L \cdot x$$
$$K_2 = L \cdot x^2 = (L \cdot x) \cdot x$$

- where multiplication (.) is done in the finite field GF(2^n) and x and x^2 are first- and second-order polynomials that are elements of GF(2^n).
- Thus, the binary representation of x consists of n-2 zeros followed by 10; the binary representation of x^2consists of zeros followed by 100.
- To generate K1 and K2 , the block cipher is applied to the block that consists entirely of 0 bits.
- The first subkey is derived from the resulting ciphertext by a left shift of one bit and, conditionally, by XOR ing a constant that depends on the block size.
- The second subkey is derived in the same manner from the first subkey.

DIGITAL SIGNATURES

- A digital signature is an authentication mechanism that enables the creator of a message to attach a code that acts as a signature.
- Typically the signature is formed by taking the hash of the message and encrypting the message with the creator's private key.
- The signature guarantees the source and integrity of the message.
- The digital signature standard (DSS) is an NIST standard that uses the secure hash algorithm (SHA).
- The most important development from the work on public-key cryptography is the digital signature.
- The digital signature provides a set of security capabilities that would be difficult to implement in any other way.

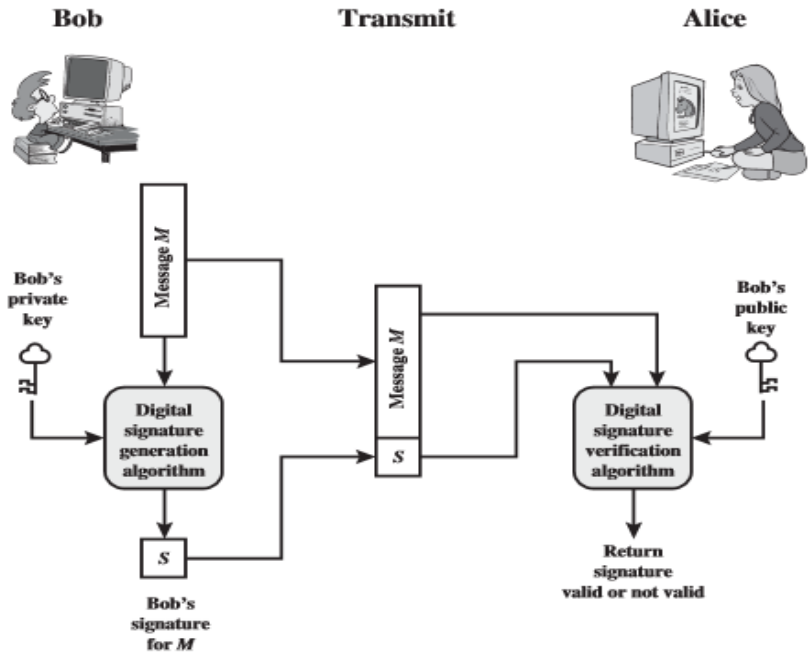


Figure 13.1 Generic Model of Digital Signature Process

Why Digital Signatures ?

- Suppose that John sends an authenticated message to Mary. Following disputes arise:

1. Mary may forge a different message and claim that it came from John. Mary would simply have to create a message and append an authentication code using the key that John and Mary share.

2. John can deny sending the message. Because it is possible for Mary to forge a message, there is no way to prove that John did in fact send the message.

- In situations where there is not complete trust between sender and receiver, something more than authentication is needed.
- The most attractive solution to this problem is the digital signature.

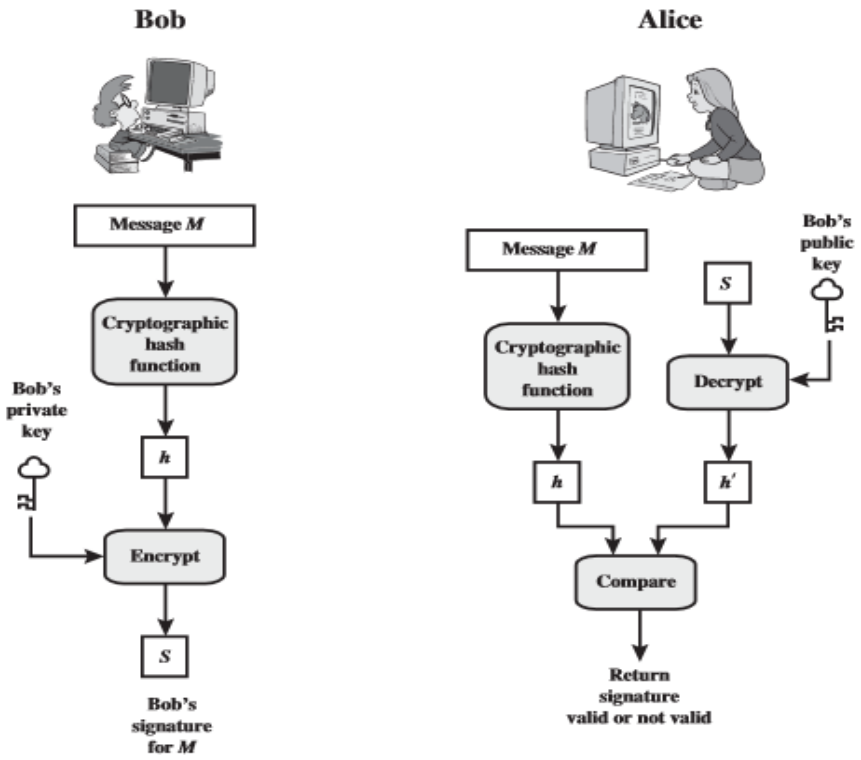


Figure 13.2 Simplified Depiction of Essential Elements of Digital Signature Process

- **The digital signature must have the following properties:**
    - It must verify the author and the date and time of the signature.
    - It must authenticate the contents at the time of the signature.
    - It must be verifiable by third parties, to resolve disputes.

ATTACKS AND FORGERIES IN DIGITAL SIGNATURES

Attacks :-

• **Key-only attack** : C only knows A's public key.

• **Known message attack :** C is given access to a set of messages and their signatures.

• **Generic chosen message attack** : C chooses a list of messages before attempting to breaks A's signature scheme, independent of A's public key C then obtains from A valid signatures for the chosen messages. The attack is generic, because it does not depend on A's public key; the same attack is used against everyone.

• **Directed chosen message attack** : Similar to the generic attack, except that the list of messages to be signed is chosen after C knows A's public key but before any signatures are seen.

• **Adaptive chosen message attack** : C is allowed to use A as an "oracle."This means the A may request signatures of messages that depend on previously obtained message–signature pairs.

Forgeries :-

• **Total break** : C determines A's private key.

• **Universal forgery** : C finds an efficient signing algorithm that provides an equivalent way of constructing signatures on arbitrary messages.

• **Selective forgery** : C forges a signature for a particular message chosen by C.

• **Existential forgery** : C forges a signature for at least one message. C has no control over the message. Consequently, this forgery may only be a minor nuisance to A.

DIGITAL SIGNATURE REQUIREMENTS

• The signature must be a bit pattern that depends on the message being signed.

• The signature must use some information unique to the sender to prevent both forgery and denial.

• It must be relatively easy to produce the digital signature.

• It must be relatively easy to recognize and verify the digital signature.

• It must be computationally infeasible to forge a digital signature, either by constructing a new message for an existing digital signature or by constructing a fraudulent digital signature for a given message.

• It must be practical to retain a copy of the digital signature in storage.

DIRECT DIGITAL SIGNATURE

- Direct digital signature refers to a digital signature scheme that involves only the communicating parties (source ,destination).
- It is assumed that the destination knows the public key of the source.
- Confidentiality can be provided by encrypting the entire message plus signature with a shared secret key (symmetric encryption).
- The validity of the scheme just described depends on the security of the sender's private key.

Disadvantages of Direct Digital Signature :-

- If a sender later wishes to deny sending a particular message, the sender can claim that the private key was lost or stolen and that someone else forged his or her signature.
- Some private key might actually be stolen from X at time T. The opponent can then send a message signed with X's signature and stamped with a time before or equal to T.

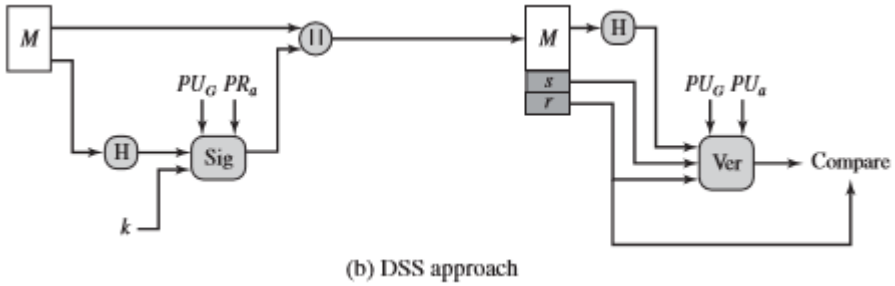
Solution :-

- Require every signed message to include a timestamp (date and time)
- Require prompt reporting of compromised keys to a central authority.
- Use of a digital certificate and certificate authorities.

CLASSIFICATION OF SIGNATURE SCHEMES

1. DIGITAL SIGNATURE STANDARD (DSS)

- The DSS uses an algorithm that is designed to provide only the digital signature function.
- It cannot be used for encryption or key exchange



(b) DSS approach

- The DSS approach also makes use of a hash function.

At sender's side,

- 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.
- Consider this set to constitute a global public key (PU_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 (PU_a) which is paired with the sender's private key.
- The output of the 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.

The Digital Signature Algorithm

- The DSA is based on the difficulty of computing discrete logarithms and is based on schemes originally presented by ElGamal
- There are three parameters that are public and can be common to a group of users.
- A 160-bit prime number ***q*** is chosen
- Next, a prime number ***p*** is selected with a length between 512 and 1024 bits such that ***q*** divides ***(p-1)*** .
- Finally, ***g*** is chosen

Global Public-Key Components

p prime number where $2^{L-1} < p < 2^L$ for $512 \le L \le 1024$ and L a multiple of 64; i.e., bit length of between 512 and 1024 bits in increments of 64 bits

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

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

User's Private Key

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

User's Public Key

$y = g^x \bmod p$

User's Per-Message Secret Number

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

Signing

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

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

Signature = (r, s)

Verifying

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

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

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

$v = [(g^{u1} y^{u2}) \bmod p] \bmod q$

TEST: $v = r'$

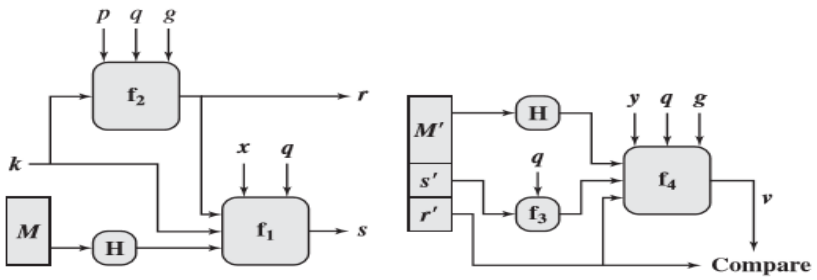
M = message to be signed

$\mathrm{H}(M)$ = hash of M using SHA-1

M', r', s' = received versions of M, r, s

Figure 13.4 The Digital Signature Algorithm (DSA)

- To create a signature, a user calculates two quantities, ***r*** and ***s*** , that are functions of the public key components ***(p,q,g)*** , the user's private key (***x***) , the hash code of the message H(M) , and an additional integer ***k*** that should be generated randomly or pseudorandomly and be unique for each signing.
- At the receiving end, verification is performed. .The receiver generates a quantity ***v*** that is a function of the public key components, the sender's public key, and the hash code of the incoming message. If this quantity matches the ***r*** component of the signature, then the signature is validated.
- Given the difficulty of taking discrete logarithms, it is infeasible for an opponent to recover ***k*** from ***r*** or to recover ***x*** from ***s*** .
- Another point worth noting is that the only computationally demanding task in signature generation is the exponential calculation $g^k \bmod p$.
- Because this value does not depend on the message to be signed, it can be computed ahead of time.



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

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

(a) Signing

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

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

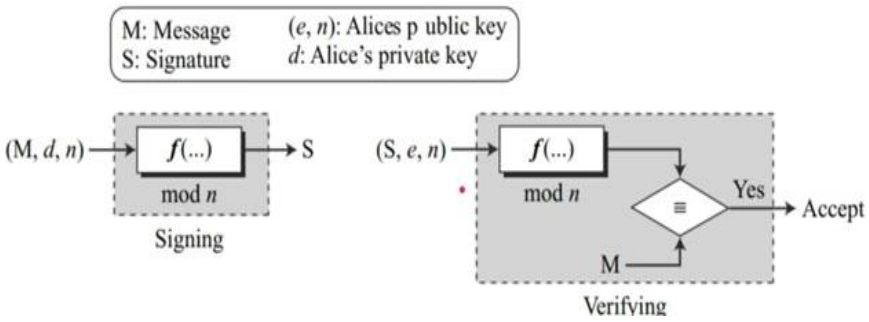
$= ((g^{(H(M')w) \bmod q} y^{r'w \bmod q}) \bmod p) \bmod q$

(b) Verifying

Figure 13.5 DSS Signing and Verifying

2. RSA DIGITAL SIGNATURE

- RSA idea can also be used for signing and verifying a message. In this case it is also called as the RSA Digital Signature Scheme
- Digital signature schemes changes the roles of the private and public keys.
- First, the private and public keys of the sender, not the receiver, are used.
- Second, the sender uses own private key to sign the document, the receiver uses the sender's public key to verify it
- If we compare the scheme with the conventional way of signing, the private key plays the role of sender's own signature and the sender's public key plays the role of the copy of the signature that is available to the public



- Signing and verifying sites use the same function, but with different parameters.
- Verifier compares the message and the output of the function for congruence.
- If the result is true, the message is accepted.

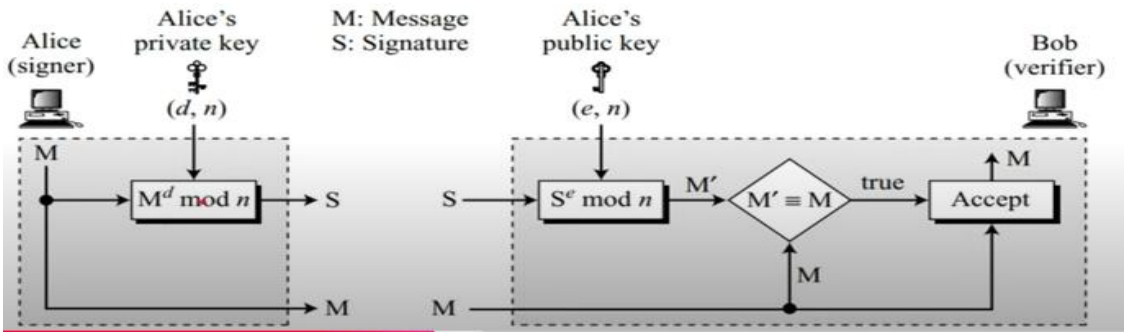
ALGORITHM – RSA DIGITAL SIGNATURE

1. Key Generation

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

1. Sender chooses two primes p and q , calculates n= p*q and φ (n) = (p-1) (q-1)
2. Choose e , the public exponent and calculates d, the private exponent such that e * d = 1 mod φ (n)
3. Sender keeps d and publically announces n and e

Ie, in RSA digital signature scheme, d is private and e , n are public



2. **Signing**

   Sender creates a signature out of the message using their private exponent, S = M^d mod n and send the message and the signature to receiver.

3. **Verifying**

   1. At the receiver end, receives M and S
   2. Receiver applies Sender's public exponent to the signature to create a copy of the message

      M' = S^e mod n
   3. Receiver compares the value of M' with the value of M
   4. If the two values are congruent, Receiver accepts the message.

M' = M mod n → S^e = M mod n → M^{d*e} = M mod n

3. ELGAMAL DIGITAL SIGNATURE SCHEME

- Elgamal digital signature scheme uses the same keys as in Elgamal cryptosystems, but the algorithm is different.
- In the signing process, two functions create two signatures ; in the verifying process the outputs of two functions are compared for verification.
- One function is used both for signing and verifying but the function uses different inputs.
- Message is part of the input to function2 when signing; it is part of the input to function1 when verifying.
- Calculations in function1 and function3 are done modulo p and modulo (p-1) in function2

Algorithm :

- As with ElGamal encryption,the global elements of ElGamal digital signature are a prime number q and α, which is a primitive root of q.
- User A generates a private/public key pair as follows.

  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 pubic key is $\{q, \alpha, Y_A\}$.

- **To sign a message** M , user A first computes the hash m = H (M) ,such that m is an integer in the range 0<= m<= (q-1) . A then forms a digital signature as follows.

  1. Choose a random integer K such that $1 \leq K \leq q - 1$ and $\gcd(K, q - 1) = 1$. That is, K is relatively prime to $q - 1$.
  2. Compute $S_1 = \alpha^K \bmod q$. Note that this is the same as the computation of C_1 for ElGamal encryption.
  3. Compute $K^{-1} \bmod (q - 1)$. That is, compute the inverse of K modulo $q - 1$.
  4. Compute $S_2 = K^{-1}(m - X_A S_1) \bmod (q - 1)$.
  5. The signature consists of the pair (S_1, S_2).

- Any user B can **verify the signature** as follows.

  1. Compute $V_1 = \alpha^m \bmod q$.
  2. Compute $V_2 = (Y_A)^{S_1}(S_1)^{S_2} \bmod q$.

- The signature is valid if V1 = V2.

$\alpha^m \bmod q = (Y_A)^{S_1}(S_1)^{S_2} \bmod q$	assume $V_1 = V_2$
$\alpha^m \bmod q = \alpha^{X_A S_1}\alpha^{KS_2} \bmod q$	substituting for Y_A and S_1
$\alpha^{m-X_A S_1} \bmod q = \alpha^{KS_2} \bmod q$	rearranging terms
$m - X_A S_1 \equiv KS_2 \bmod (q-1)$	property of primitive roots
$m - X_A S_1 \equiv KK^{-1}(m - X_A S_1) \bmod (q-1)$	substituting for S_2