UNIT IV INTEGRITY AND AUTHENTICATION ALGORITHMS

Authentication requirement – Authentication function – MAC – Hash function – Security of hash function: HMAC, CMAC – SHA – Digital signature and authentication protocols – DSS – Schnorr Digital Signature Scheme – ElGamal cryptosystem – Entity Authentication: Biometrics, Passwords, Challenge Response protocols – Authentication applications – Kerberos MUTUAL TRUST: Key management and distribution – Symmetric key distribution using symmetric and asymmetric encryption – Distribution of public keys – X.509 Certificates.

AUTHENTICATION REQUIREMENTS

- In the context of communication across a network, the following attacks can be identified:
- **Disclosure** – releases of message contents to any person or process not possessing the appropriate cryptographic key.
- **Traffic analysis** – discovery of the pattern of traffic between parties.
- **Masquerade** – insertion of messages into the network fraudulent source.
- **Content modification** – changes to the content of the message, including insertion deletion, transposition and modification.
- **Sequence modification** – any modification to a sequence of messages between parties, including insertion, deletion and reordering.
- **Timing modification** – delay or replay of messages.
- **Source repudiation** – denial of transmission of message by source.
- **Destination repudiation** – denial of transmission of message by destination.
- Measures to deal with first two attacks are in the realm of message confidentiality.

  Measures to deal with 3 through 6 are regarded as message authentication. Item 7 comes under digital signature and dealing with item 8 may require a combination of digital signature and a protocol to counter this attack.

AUTHENTICATION FUNCTIONS

- Any message authentication or digital signature mechanism can be viewed as having fundamentally two levels. At the lower level, there may be some sort of function that produces an authenticator: a value to be used to authenticate a message. This lower layer function is then used as primitive in a higher-layer authentication protocol that enables a receiver to verify the authenticity of a message.
- **The different types of functions** that may be used to produce an **authenticator** are as follows:

- **Message encryption** – the cipher text of the entire message serves as its authenticator.
- **Message authentication code (MAC)** – a public function of the message and a secret key that produces a fixed length value serves as the authenticator.
- **Hash function** – a public function that maps a message of any length into a fixed length hash value, which serves as the authenticator

MESSAGE ENCRYPTION

- Message encryption by itself can provide a measure of authentication. The analysis differs from symmetric and public key encryption schemes.
- Suppose the message can be any arbitrary bit pattern. In that case, there is no way to determine automatically, at the destination whether an incoming message is the ciphertext of a legitimate message (Figure -4.1). One solution to this problem is to force the plaintext to have some structure that is easily recognized but that cannot be replicated without recourse to the encryption function. We could, for example, append an error detecting code, also known as Frame Check Sequence (FCS) or checksum to each message before encryption
- 'A' prepares a plaintext message M and then provides this as input to a function F that produces an FCS. The FCS is appended to M and the entire block is then encrypted. At the destination, B decrypts the incoming block and treats the result as a message with an appended FCS. B applies the same function F to attempt to reproduce the FCS. If the calculated FCS is equal to the incoming FCS, then the message is considered authentic.

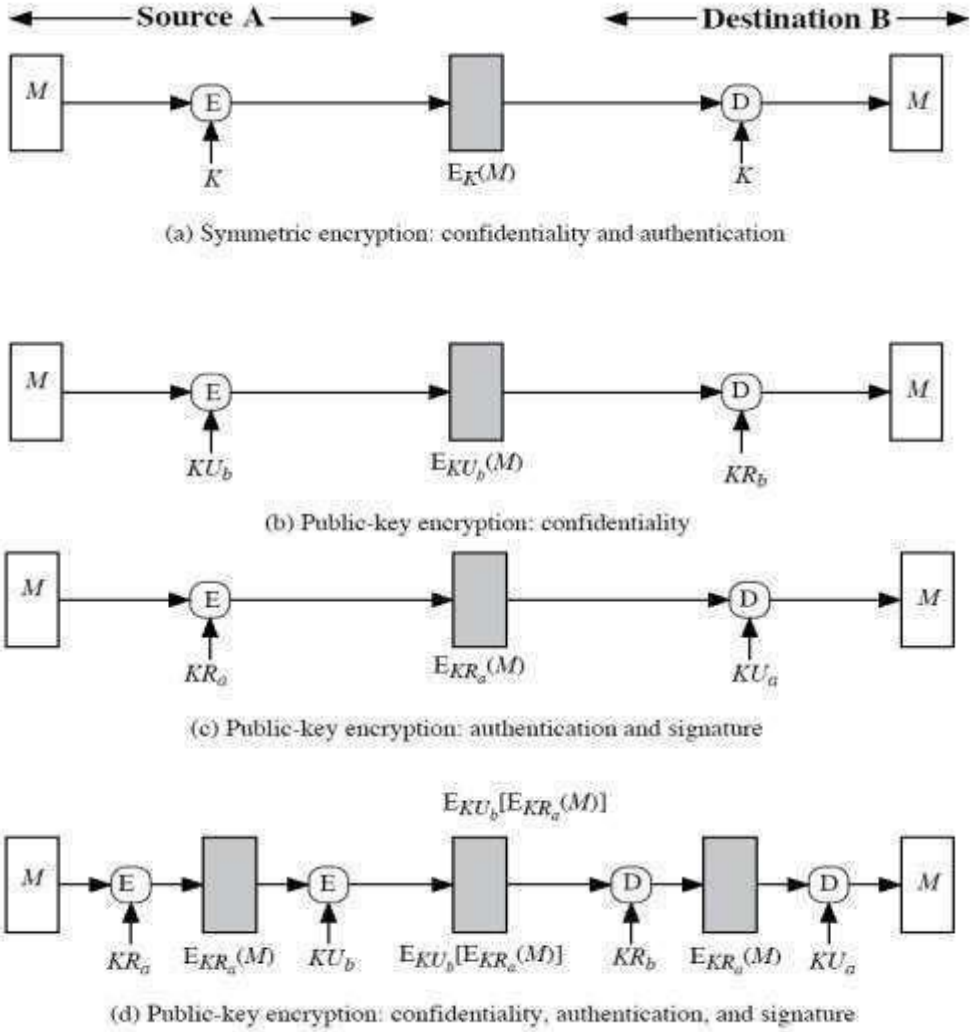


Figure 4.1 . Basic Uses of Message Encryption

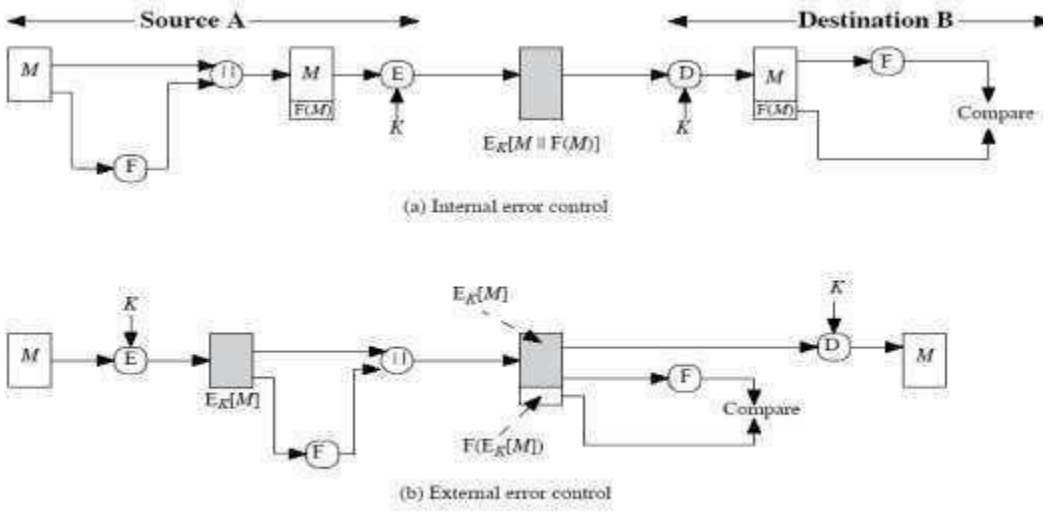


Figure 4.2. Internal and External Error Control

As shown in Figure 4.2, In the internal error control, the function F is applied to the plaintext, whereas in external error control, F is applied to the ciphertext (encrypted message).

MESSAGE AUTHENTICATION CODE (MAC)

- An alternative authentication technique involves the use of secret key to generate a small fixed size block of data, known as cryptographic checksum or MAC that is appended to the message. This technique assumes that two communication parties say A and B, share a common secret key 'k'. When A has to send a message 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

- The message plus MAC are transmitted to the intended recipient. The recipient performs the same calculation on the received message, using the shared secret key, to generate a new MAC. The received MAC is compared to the calculated MAC. If it is equal, then the message is considered authentic.
- A MAC function is similar to encryption. One difference is that MAC algorithm need not be reversible, as it must for decryption. In general, the MAC function is a many-to-one function.

Requirements for MAC:

- When an entire message is encrypted for confidentiality, using either symmetric or asymmetric encryption, the security of the scheme generally depends on the bit length of the key. Barring some weakness in the algorithm, the opponent must resort to a brute-force attack using all possible keys. On average, such an attack will require $2^{(k-1)}$ attempts for a k-bit key.

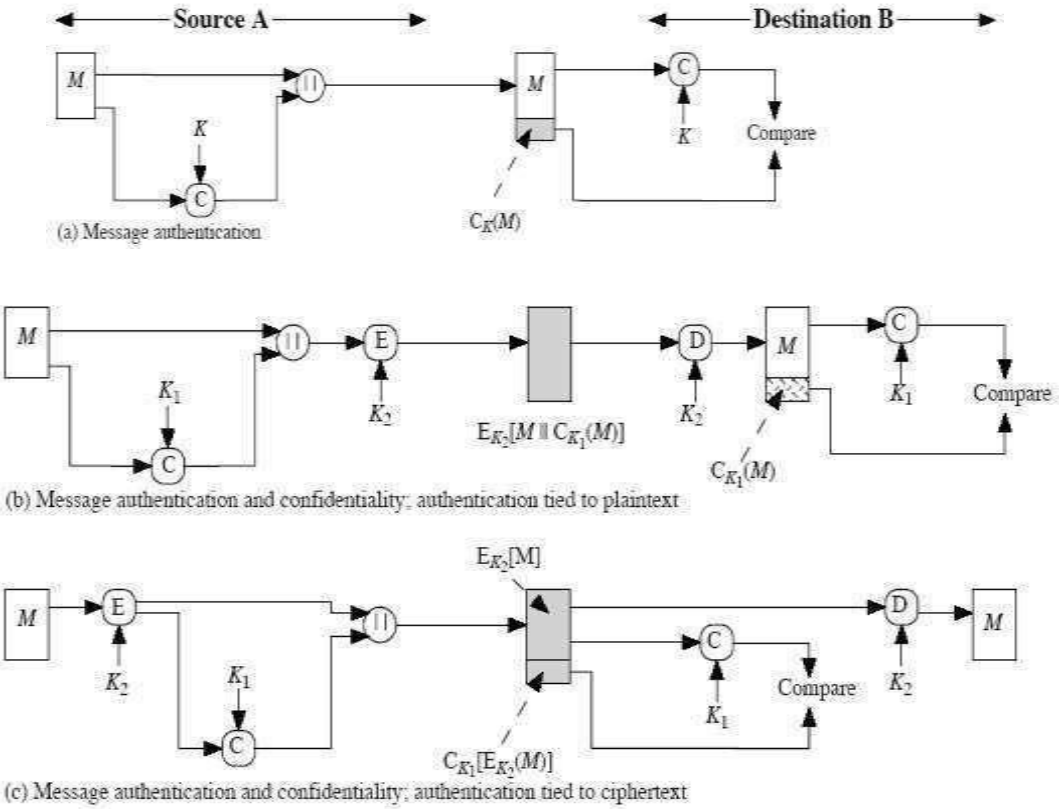


Figure 4.3 Basic Uses of Message Authentication Code (MAC

- In the case of a MAC, the considerations are entirely different. Using brute-force methods, how would an opponent attempt to discover a key?
- If confidentiality is not employed, the opponent has access to plaintext messages and their associated MACs. Suppose k>n; that is, suppose that the key size is greater than the MAC size. Then, given a known M_1 and MAC_1, with $MAC_1 = C(K,M_1)$, the cryptanalyst can perform $MAC_i = C(K_i,M_1)$ for all possible key values K_i.
- At least one key is guaranteed to produce a match of $MAC_i = MAC_1$.
- Note that a total of 2^k MACs will be produced, but there are only $2^n < 2^k$ different MAC values. Thus, a number of keys will produce the correct MAC and the opponent has no way of knowing which is the correct key. On average, a total of $2^k/2^n = 2^{(k-n)}$ keys will produce a match. Thus, the opponent must iterate the attack:
- **Round 1**

  Given: M_1, $MAC_1 = C(K,M_1)$

  Compute $MAC_i = C(K_i,M_1)$ for all 2^k keys

  Number of matches $\approx 2^{(k-n)}$

Round 2

- Given: M_2, $MAC_2 = C(K, M_2)$
- Compute $MAC_i = C(K_i, M_2)$ for the $2^{(k-n)}$ keys resulting from Round 1
- Number of matches $\approx 2^{(k-2xn)}$
- and so on. On average, a rounds will be needed if k = a x n. For example, if an 80-bit key is used and the MAC is 32 bits long, then the first round will produce about 2^{48} possible keys. The second round will narrow the possible keys to about 2^{16} possibilities. The third round should produce only a single key, which must be the one used by the sender.
- If the key length is less than or equal to the MAC length, then it is likely that a first round will produce a single match.
- Thus, a brute-force attempt to discover the authentication key is no less effort and may be more effort than that required to discover a decryption key of the same length. However, other attacks that do not require the discovery of the key are possible.
- Consider the following MAC algorithm. Let $M = (X_1||X_2||...||X_m)$ be a message that is treated as a concatenation of 64-bit blocks X_i. Then define

$$\Delta(M) = X1 \oplus X2 \oplus ... \oplus Xm$$

$$C(k,M) = E(k, \Delta(M))$$

- Where $\oplus$ is the exclusive-OR (XOR) operation and the encryption algorithm is DES in electronic codebook mode. Thus, the key length is 56 bits and the MAC length is 64 bits. If an opponent observes {M||C(K, M)}, a brute-force attempt to determine K will require at least 2^{56} encryptions. But the opponent can attack the system by replacing X_1 through X_{m-1} with any desired values Y_1 through Y_{m-1} and replacing X_m with Y_m where Y_m is calculated as follows

$$Y_m = Y_1 \oplus Y_2 \oplus \quad ... \quad \oplus Y_{m1} \oplus \Delta(M)$$

- The opponent can now concatenate the new message, which consists of Y_1 through Y_m, with the original MAC to form a message that will be accepted as authentic by the receiver. With this tactic, any message of length 64 x (m-1) bits can be fraudulently inserted.
- Then the MAC function should satisfy the following requirements:
- The MAC function should have the following properties:
- If an opponent observes M and C(K,M), it should be computationally infeasible for the opponent to construct a message M' such that C(K,M') = C(K,M)
- $C_K(M)$ should be uniformly distributed in the sense that for randomly chosen messages, M and M', the probability that C(K,M) = C(K,M') is 2^{-n} where n is the number of bits in the MAC.
- Let M' be equal to some known transformation on M. i.e., M' = f(M).

HASH FUNCTIONS:

- A **hash function** H accepts a variable-length block of data as input and produces a fixed-size hash value h=H(M). A "good" hash function has the property that the results of applying the function to a large set of inputs will produce outputs that are evenly distributed and apparently random.
- The kind of hash function needed for security applications is referred to as a **cryptographic hash function**. A cryptographic hash function is an algorithm for which it is computationally infeasible (because no attack is significantly more efficient than brute force) to find either (a) a data object that maps to a pre-specified hash result (the one-way property) or (b) two data objects that map to the same hash result(the collision-free property). Because of these characteristics, hash functions are often used to determine whether or not data has changed.
- the input is padded out to an integer multiple of some fixed length (e.g., 1024bits), and the padding includes the value of the length of the original message in bits. The length field is a security measure to increase the difficulty for an attacker to produce an alternative message with the same hash value.

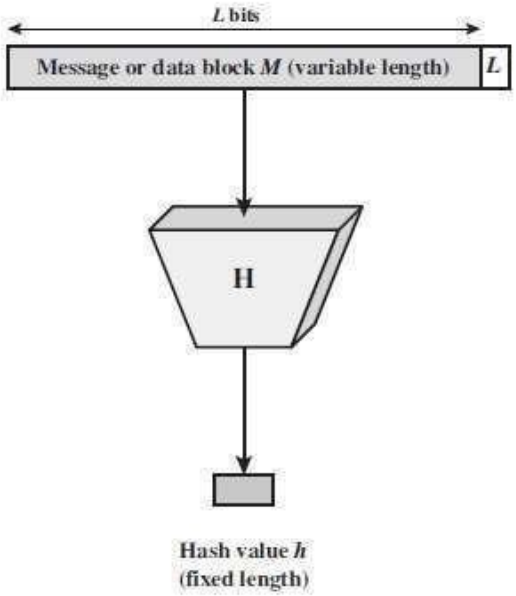


Figure 4.4 – Hash function

Applications of Cryptographic Hash functions

1. **Message Authentication :** When a hash function is used to provide message authentication, the hash function value is often referred to as a **message digest**. More commonly, message authentication is achieved using a **message authentication code (MAC)**, also known as a **keyed hash function**. Typically, MACs are used between two parties that share a secret key to authenticate information exchanged between those parties.

2. **Digital Signature** - The hash value of a message is encrypted with a user's private key. Anyone who knows the user's public key can verify the integrity of the message that is associated with the digital signature

3. Hash functions are commonly used to create a **one-way password file**. 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.

4. Hash functions can be used for **intrusion detection** and **virus detection**. StoreH(F) for each file on a system and secure the hash values (e.g., on a CD-R that is kept secure). One can later determine if a file has been modified by re computingH(F). An intruder would need to change F without changing H(F).

- 5.A cryptographic hash function can be used to construct a **pseudorandom function(PRF)** or a **pseudorandom number generator (PRNG)**.A common application for a hash-based PRF is for the generation of symmetric keys.

Simple Hash Functions

- All hash functions operate using the following general principles. The input (message, file, etc.) is viewed as a sequence of n-bit blocks. The input is processed one block at a time in an iterative fashion to produce an n-bit hash function.

- One of the simplest hash functions is the bit-by-bit exclusive-OR (XOR) of every block. This can be expressed as follows:

$$C_i = b_{i1} \oplus b_{i1} \oplus \ldots \oplus b_{im}$$

Where

C_i = ith bit of the hash code, $1 = i = n$

m = number of n-bit blocks in the input

b_{ij} = ith bit in jth block

$\oplus$ = XOR operation

- Thus, the probability that a data error will result in an unchanged hash value is 2^n. With more predictably formatted data, the function is less effective. For example, in most normal text files, the high-order bit of each octet is always zero. So if a 128-bit hash value is used, instead of an effectiveness of 2^{128}, the hash function on this type of data has an effectiveness of 2^{112}.

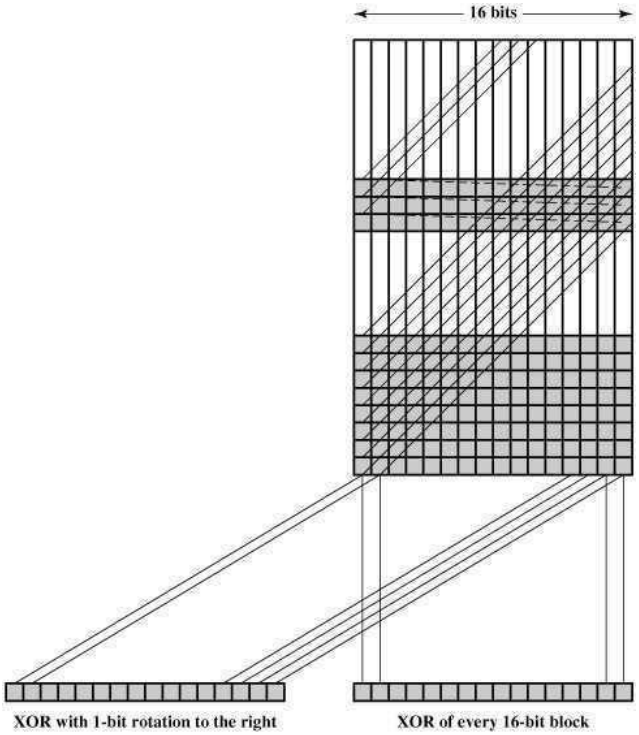


Figure 4.5. Two Simple Hash Functions

- **A simple way to improve matters** is to perform a one-bit circular shift, or rotation, on the hash value after each block is processed. The procedure can be summarized as follows:

- Initially set the n-bit hash value to zero.
- Process each successive n-bit block of data as follows:
  - Rotate the current hash value to the left by one bit.
  - XOR the block into the hash value.

- Figure 4.5 illustrates these two types of hash functions for 16-bit hash values.
- Given a message consisting
- of a sequence of 64-bit blocks $X_1, X_2, X_3 \ldots X_N$, define the hash code as h=H(M) as the block-by-block XOR of all blocks and append the hash code as the final block:

$$h = XN+1 = X1 \oplus X2 \oplus \ldots \oplus XN$$

4.4.3 REQUIREMENTS AND SECURITY

- Before proceeding, we need to define two terms. For a hash value h=H(x), we say that is the **preimage** of h. That is, is a data block whose hash function, using the function H, is h. Because H is a many-to-one mapping, for any given hash value h, there will in general be multiple preimages.

- A **collision** occurs if we have x ≠ y and H(x) = H(y). Because we are using hash functions for data integrity, collisions are clearly undesirable

- **Attacks on Hash functions**

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

- **Brute-Force Attacks**

- 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.

- **Cryptanalysis**

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

Table 11.1 Requirements for a Cryptographic Hash Function H

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.

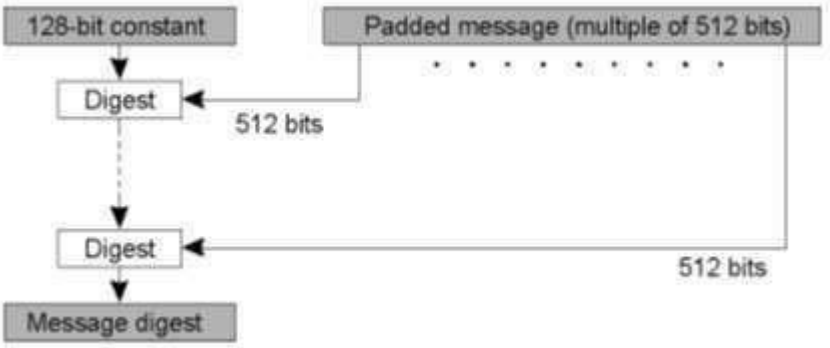


Figure 4. 6 – MD5 algorithm

MD 5

- MD5 algorithm was developed by Professor Ronald L. Rivest in 1991. According to RFC 1321, "MD5 message-digest algorithm takes as input a message of arbitrary length and produces as output a 128-bit "fingerprint" or "message digest" of the input …The MD5 algorithm is intended for digital signature applications, where a large file must be "compressed" in a secure manner before being encrypted with a private (secret) key under a public-key cryptosystem such as RSA

- MD5 algorithm consists of 5 steps (Figure 4.6):
- Step **1. Appending Padding Bits.** The original message is "padded" (extended) so that its length (in bits) is congruent to 448, modulo 512. The padding rules are: The original message is always padded with one bit "1" first. Then zero or more bits "0" are padded to bring the length of the message up to 64 bits fewer than a multiple of 512.
- Step 2. **Appending Length**. 64 bits are appended to the end of the padded message to indicate the length of the original message in bytes. The rules of appending length are: The length of the original message in bytes is converted to its binary format of 64 bits. If overflow happens, only the low-order 64 bits are used. Break the 64-bit length into 2 words (32 bits each). The low-order word is appended first and followed by the high-order word.

- Step 3**. Initializing MD Buffer**. MD5 algorithm requires a 128-bit buffer with a specific initial value. The rules of initializing buffer are:

  The buffer is divided into 4 words (32 bits each), named as A, B, C, and D.

  Word A is initialized to: 0x67452301.

  Word B is initialized to: 0xEFCDAB89.

  Word C is initialized to: 0x98BADCFE.

  Word D is initialized to: 0x10325476.

- Step 4. **Processing Message in 512-bit Blocks**. This is the main step of MD 5 algorithm, which loops through the padded and appended message in blocks of 512 bits each. For each input block, 4 rounds of operations are performed with 16 operations in each round

- F (X, Y, Z) = XY or not (X) Z
- G (X, Y, Z) = XZ or Y not (Z)
- H (X, Y, Z) = X xor Y xor Z
- I (X, Y, Z) = Y xor (X or not (Z))

Secure Hash Algorithm (SHA-1)

- SHA was designed by NIST & NSA in 1993, revised 1995 as SHA-1
- US standard for use with DSA signature scheme
  - standard is FIPS 180-1 1995, also Internet **RFC3174**
  - the algorithm is SHA, the standard is SHS
- produces 160-bit hash values
- now the generally preferred hash algorithm
- based on design of MD4 with key differences

	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Message Digest Size	160	224	256	384	512
Message Size	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block Size	512	512	512	1024	1024
Word Size	32	32	32	64	64
Number of Steps	80	64	64	80	80

Note: All sizes are measured in bits.

Comparison of SHA Parameters

SHA – 512 ALGORITHM

- The Secure Hash Algorithm (SHA) was developed by the National Institute of Standards and Technology (NIST) and published as a federal information processing standard (FIPS 180) in 1993; a revised version was issued as FIPS 180-1 in 1995 and is generally referred to as SHA-1. The actual standards document is entitled ***Secure Hash Standard***.
- SHA is based on the hash function MD4 and its design closely models MD4.
- SHA-1 is also specified in RFC 3174, which essentially duplicates the material in FIPS 180-1, but adds a C code implementation.
- The algorithm takes as input a message with a maximum length of less than 2^{128} bits and produces as output a 512-bit message digest (Figure 4.7). The input is processed in 1024-bit blocks. Below figure depicts the overall processing of a message to produce a digest.

1 . Append padding bits. The message is padded so that its length is congruent to 896 modulo 1024 [length ≡ 896 (mod 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.

1. **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. In Figure 4.7, the expanded message is represented as the sequence of 1024-bit blocks M_1, M_2,..., M_N, so that the total length of the expanded message is N x 1024 bits.

2. **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 the following 64-bit integers (hexadecimal values):

a = 6A09E667F3BCC908

b = BB67AE8584CAA73B

c = 3C6EF372FE94F82B

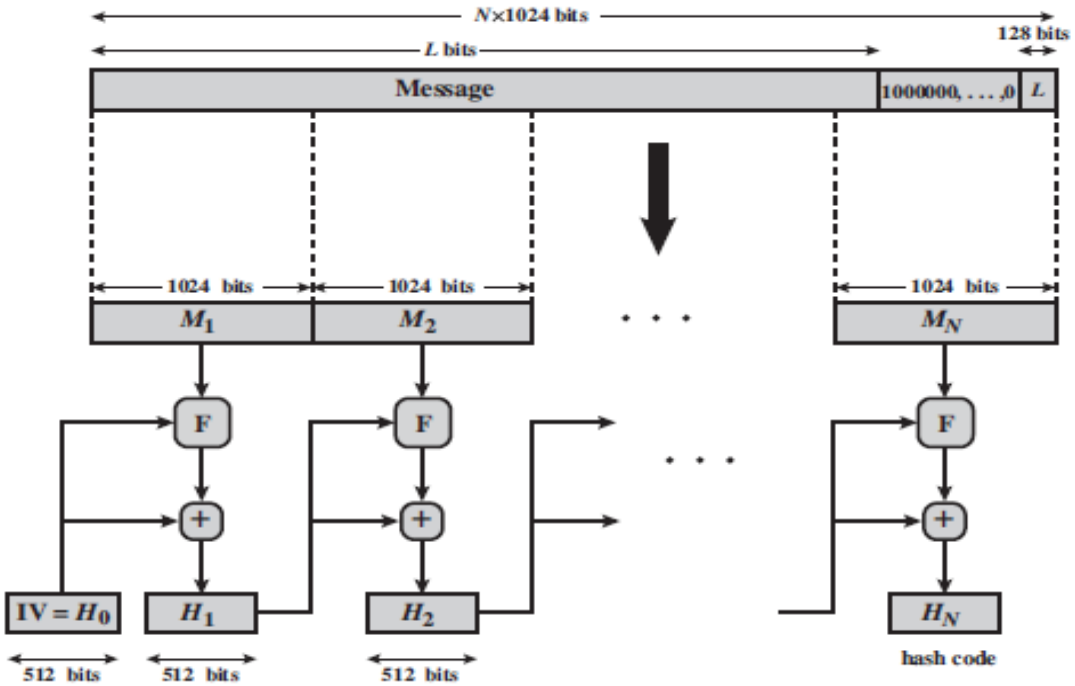
d = A54FF53A5F1D36F1

e = 510E527FADE682D1

f = 9B05688C2B3E6C1F

g = 1F83D9ABFB41BD6B

h = 5BE0CDI9137E2179



- **Process** message in 1024-bit (128-word) blocks. The heart of the algorithm is a module that consists of 80 rounds; this module is labeled F in Figure 3.9. The logic is illustrated in Figure 4.8. Each round takes as input the 512-bit buffer value abcdefgh, 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_t derived from the current 1024-bit block being processed (M_i).

  These values are derived using a message schedule. Each round also makes use of an additive constant K_t, where $0 \leq t \leq 79$ indicates one of the 80 rounds. These words represent the first sixty-four bits of the fractional parts of the cube roots of the first eighty prime numbers. The constants provide a "randomized" set of 64-bit patterns, which should eliminate any regularities in the input data. The output of the eightieth round is added to the input to the first round (H_{i-1}) to produce H_i. The addition is done independently for each of the eight words in the buffer with each of the corresponding words in H_{i-1} using addition modulo 2^{64}.

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

The behavior of SHA-512 can be summarized as follows:

$H_0 = IV$

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

$MD = H_N$

where

IV = initial value of the abcdefgh buffer, defined in step 3

$abcdefgh_i$ = the output of the last round of processing of the i^{th} message block

N = the number of blocks in the message (including padding and length fields)

SUM_{64} = Addition modulo 2^{64} performed separately on each word of the pair of inputs.

MD = final message digest value

- **Each round can be defined mathematically as (Figure 4.9):**

$$T_1 = h + Ch(e,f,g) + (\sum_1^{512} e) + W_t + K_t$$

$$T_2 = (\sum_0^{512} a) + Maj(a,b,c)$$

h = g

g = f

f = e

e = d + T_1

d = c

c = b

b = a

a = T_1 +T_2

where t = step number; $0 \leq t \leq 79$

Ch(e, f, g) = (e AND f) ⊕ (NOT e AND g)

Maj(a, b, c) = (a AND b) ⊕ (a AND c) ⊕ (b AND c)

$$(\sum_0^{512} a) = ROTR^{28}(a) \oplus ROTR^{34}(a) \oplus ROTR^{39}(a)\text{i}$$

$$(\sum_1^{512} e) = ROTR^{14}(e) \oplus ROTR^{18}(e) \oplus ROTR^{41}(e)$$

$ROTR^n(x)$ = circular right shift of x by n bits

W_t = word derived from the current input block

K_t = additive constant

+ = addition modulo 2^{64}

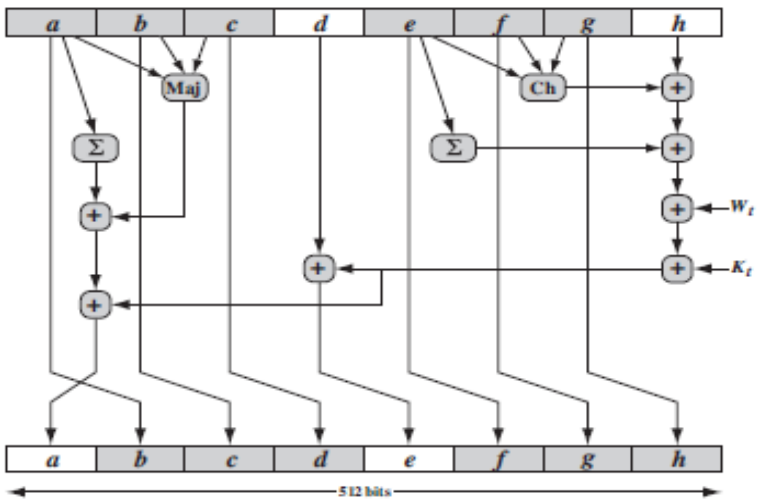


Figure 4.9: Elementary Operations (SHA-512)

- **Important Observations about round function**

- It could be observed that six of the eight words are derived out of simple permutation by means of rotation.

- Only two output words ***a***, and ***e*** are generated by substitution. Word ***e*** is a function of the input variables (d, e, f, g, h), W_t and K_t. Word ***a*** is a function of all the input variables except for d, W_t and K_t.

- Figure 4.10 illustrates how the 64-bit word values W_t are derived from the 1024-bit message. The first 16 values of W_t are taken directly from the 16 words of the current block. The remaining values are defined as

$$W_t = \sigma_1^{512}(W_{t-2}) + W_{t-7} + \sigma_0^{512}(W_{t-15}) + W_{t-16}$$

Where

$$\sigma_0^{512}(x) = ROTR^1(x) \oplus ROTR^8(x) \oplus SHR^7(x),$$
$$\sigma_1^{512}(x) = ROTR^{19}(x) \oplus ROTR^{61}(x) \oplus SHR^6(x)$$

$ROTR^n(x)$ = circular right shift of x by n bits, $SHR^n(x)$ = left shift of x by n bits with padding by zeros on the right and + = addition modulo 2^{64}

In the first 16 steps of processing, the value of W_t is equal to the corresponding word in the message block. For the remaining 64 steps, the value of W_t consists of the circular left shift by one bit of the XOR of four of the preceding values of W_t, with two of those values subjected to shift and rotate operations. This introduces a great deal of redundancy and interdependence into the message blocks that are compressed, which complicates the task of finding a different message block that maps to the same compression function output.

The SHA-512 algorithm has the property that every bit of the hash code is a function of every bit of the input. The complex repetition of the basic function F produces results that are well mixed; that is, it is unlikely that two messages chosen at random, even if they exhibit similar regularities, will have the same hash code. Unless there is some hidden weakness in SHA-512, which has not so far been published, the difficulty of coming up with two messages having the same message digest is on the order of 2^{256} operations, while the difficulty of finding message with a given digest is on the order of 2^{512} operations.

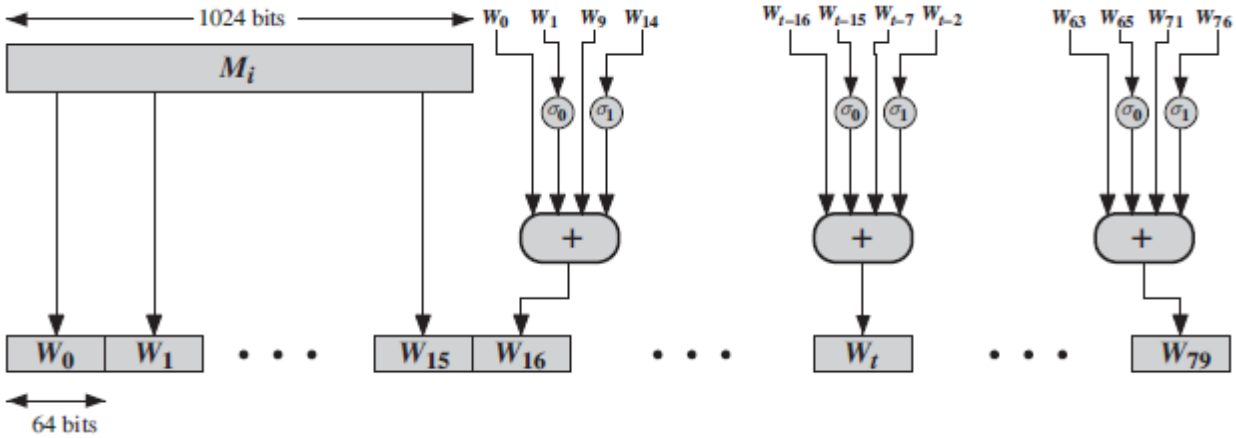


Figure 4.10: Input Sequence Creation (SHA-512)

Keys For Comparison	MD5	SHA
Security	Less Secure than SHA	High Secure than MD5
Message Digest Length	128 Bits	160 Bits
Attacks required to find out original Message	2^{128} bit operations required to break	2^{160} bit operations required to break
Attacks to try and find two messages producing the same MD	2^{64} bit operations required to break	2^{80} bit operations required to break
Speed	Faster, only 64 iterations	Slower than MD5, Required 80 iterations
Successful attacks so far	Attacks reported to some extents	No such attach report yet

HMAC

- MAC derived from a cryptographic hash function. The motivations for this interest are

1. Cryptographic hash functions such as MD5 and SHA generally execute faster in software than symmetric block ciphers such as DES.
2. Library code for cryptographic hash functions is widely available.

HMAC Design Objectives

- RFC 2104 lists the following design objectives for HMAC.

• To use, without modifications, available hash functions. In particular, to use hash functions that perform well in software and for which code is freely and widely available.

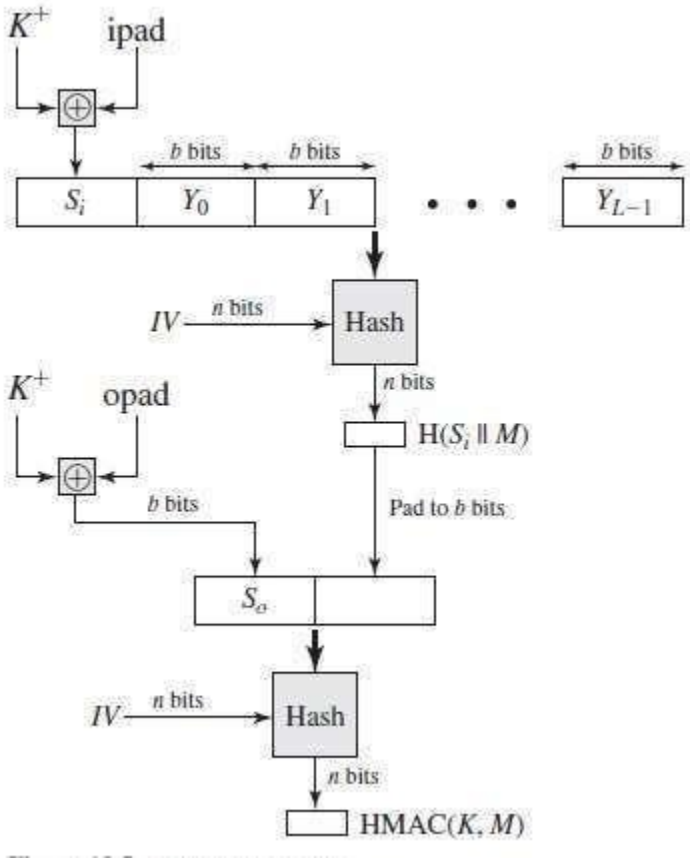


Figure 4.11- HMAC Structure

• To allow for easy replaceability of the embedded hash function in case faster or more secure hash functions are found or required.

• To preserve the original performance of the hash function without incurring a significant degradation.

• To use and handle keys in a simple way.

• To have a well understood cryptographic analysis of the strength of the authentication mechanism based on reasonable assumptions about the embedded hash function.

HMAC Algorithm

- Figure 4.11 illustrates the overall operation of HMAC. Define the following terms.

H = embedded hash function (e.g., MD5, SHA-1, RIPEMD-160)

IV = initial value input to hash function

M = message input to HMAC (including the padding specified in the embedded hash function)

Yi _ i th block of M, 0 ≤ i ≤ (L – 1)

L _ number of blocks in M

b _ number of bits in a block

n _ length of hash code produced by embedded hash function

K _ secret key; recommended length is ≥ n; if key length is greater than b,

the key is input to the hash function to produce an n-bit key

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

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

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

Then HMAC can be expressed as

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

The algorithm can be explained as the following sequence of steps.

1. 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 zero bytes 0 x 00).

2. XOR (bitwise exclusive-OR) K+ with ipad to produce the b-bit block Si.

3.Append M to Si.

4.Apply H to the stream generated in step 3.

5.XOR K+ with opad to produce the b-bit block So

6.Append the hash result from step 4 to So

7.Apply H to the stream generated in step 6 and output the result.

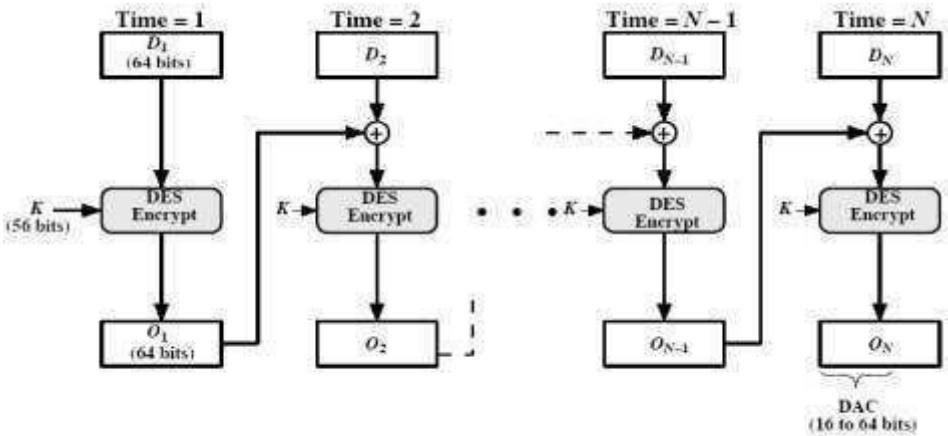


Figure 4.13 - Data Authentication Algorithm (DAA)

MAC BASED ON BLOCK CIPHERS: DAA and CMAC

- MAC based on DES
- One of the most widely used MACs, referred to as Data Authentication Algorithm (DAA) is based on DES.
- The algorithm can be defined as using cipher block chaining (CBC) mode of operation of DES with an initialization vector of zero (figure 4.13). The data to be authenticated are grouped into contiguous 64-bit blocks: D1,D2 … Dn. if necessary, the final block is padded on the right with zeros to form a full 64-bit block. Using the DES encryption algorithm and a secret key, a data authentication code (DAC) is calculated.

CMAC – Cipher Based Message Authentication Code

- First, let us consider the operation of CMAC when the message is an integer multiple n of the cipher block length b. For AES, b = 128 and for triple DES, b = 64. The message is divided into n blocks, M1,M2,...,Mn. The algorithm makes use of a k-bit encryption key K and an n-bit constant K1. For AES, the key size k is 128, 192, or 256 bits; for triple DES, the key size is 112 or 168 bits. CMAC is calculated as follows:

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

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:

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

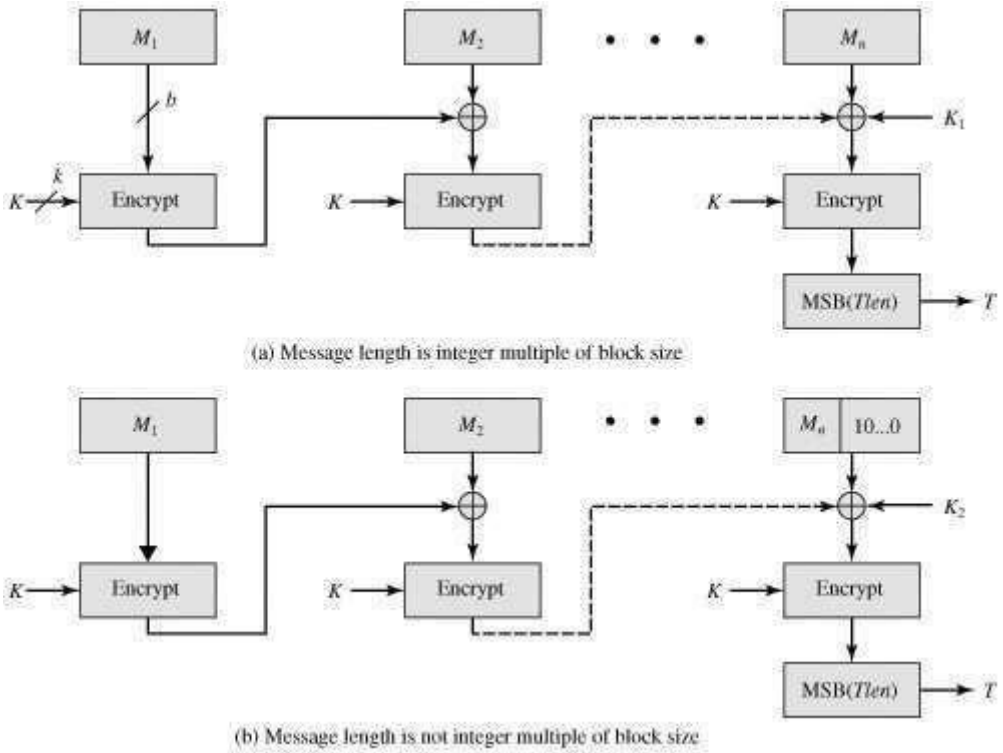


Figure 4.14 - Cipher-Based Message Authentication Code (CMAC)

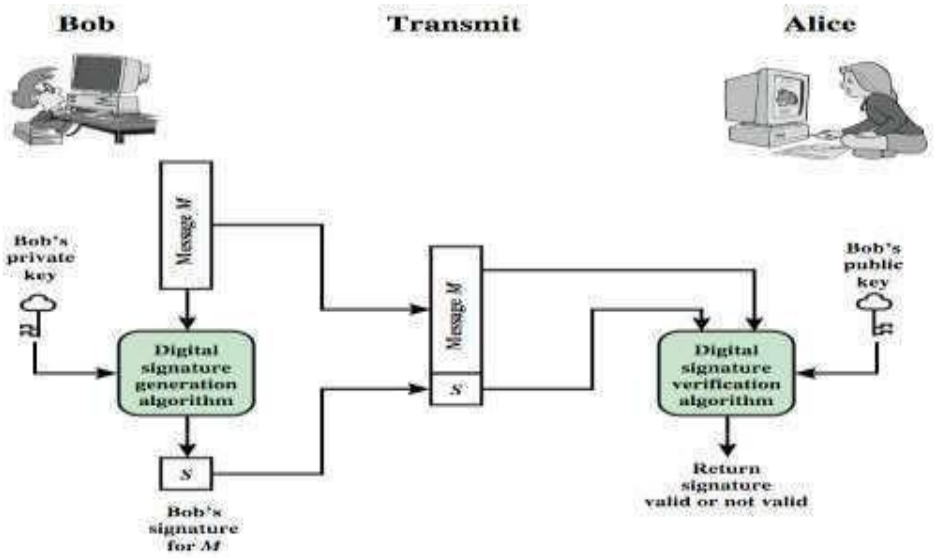
- where multiplication (·) is done in the finite field (2n) and x and x2 are first and second order polynomials that are elements of GF(2n) Thus the binary representation of x consists of n - 2 zeros followed by 10; the binary representation of x2 consists of n - 3 zeros followed by 100.
- The finite field is defined with respect to an irreducible polynomial that is lexicographically first among all such polynomials with the minimum possible number of nonzero terms.
- For the two approved block sizes, the polynomials are and $x64 + x4 + x3 + x + 1$ and $x128 + x7 + x2 + x + 1$.
- To generate K_1 and K_2, the block cipher is applied to the block that consists entirely of 0 bits. The first sub key is derived from the resulting ciphertext by a left shift of one bit and, conditionally, by XORing a constant that depends on the block size. The second sub key is derived in the same manner from the first sub key.

DIGITAL SIGNATURE STANDARD

- The most important development from the work on public-key cryptography is the digital signature. Message authentication protects two parties who exchange messages from any third party. However, it does not protect the two parties against each other either fraudulently creating, or denying creation, of a message. A digital signature is analogous to the handwritten signature, and provides a set of security capabilities that would be difficult to implement in any other way. It must have the following properties:

  - It must verify the author and the date and time of the signature
  - It must to authenticate the contents at the time of the signature
  - It must be verifiable by third parties, to resolve disputes

Thus, the digital signature function includes the authentication function.



Bob can sign a message using a digital signature generation algorithm. The inputs to the algorithm are the message and Bob's private key. Any other user, say Alice, can verify the signature using a verification algorithm, whose inputs are the message, the signature, and Bob's public key.

The following lists the following types of attacks, in order of increasing severity. Here A denotes the user whose signature is being attacked and C denotes the attacker.

- **Key-only attack**: C only knows A's public key.
- **Known message attack**: C is given access to a set of messages and 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 is chosen after C knows A's public key but before 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.
- [GOLD88] then defines success as breaking a signature scheme as an outcome in which C can do any of the following with a non-negligible probability:
- **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.

- The DSS uses an algorithm that is designed to provide only the digital signature function. Unlike RSA, it cannot be used for encryption or key exchange .Nevertheless, it is a public-key technique. Figure below contrasts the DSS approach for generating digital signatures to that used with RSA.

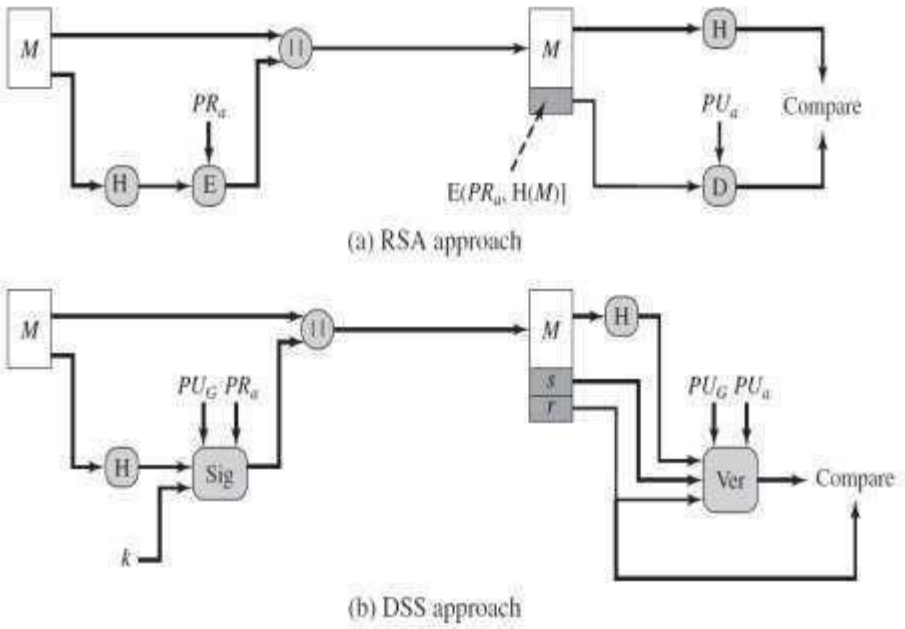


Figure : Two Approaches to Digital Signatures

- In the RSA approach, 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 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. 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.

- The DSS approach also makes use of a hash function. The hash code is provided as input to a signature function along with a random number 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 (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_G) , 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 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

There are three parameters that are public and can be common to a group of users. A 160-bit prime number is chosen. Next, a prime number is selected with a length between 512 and 1024 bits such that q divides (p - 1). Finally, g is chosen to be of the form $h^{(p-1)/q}$ mod p, where h is an integer between 1 and (p-1) with a restriction that g must be greater than 1.

With these numbers in hand, each user selects a private key and generates a public key. The private key x must be a number from 1 to (q-1) and should be chosen randomly or pseudorandomly

- The Digital Signature Algorithm (DSA) is shown below

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

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

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

Signature = (r, s)

Verifying

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

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

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

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

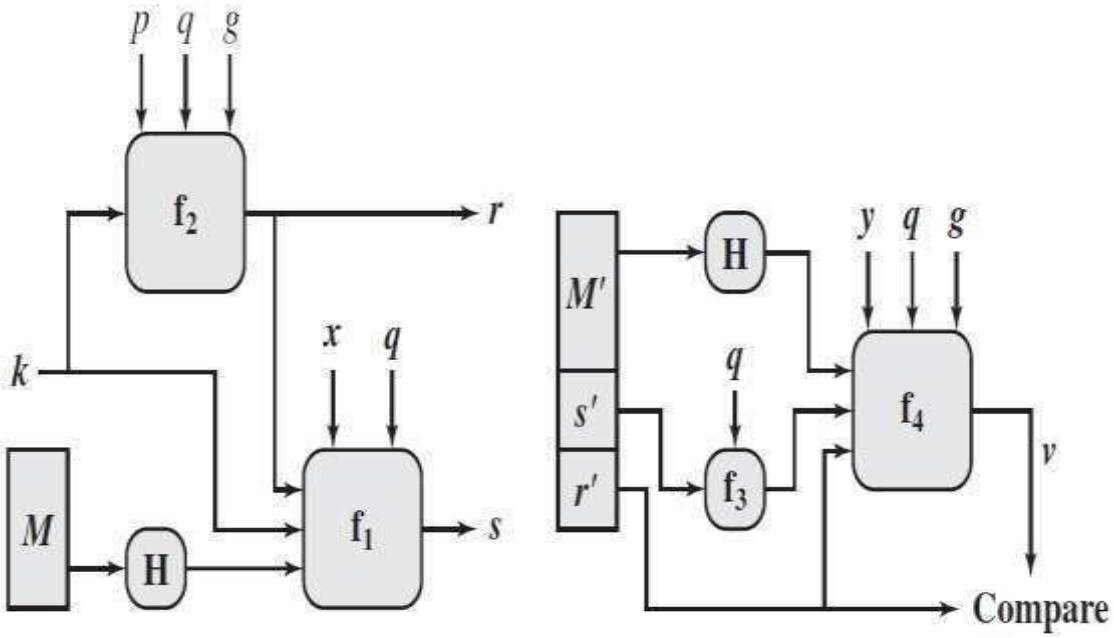
TEST: $v = r'$

M = message to be signed

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

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

- To create a signature, a user calculates two quantities, and , 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 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 component of the signature, then the signature is validated. Figure below depicts the functions of signing and verifying.



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

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

(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

- Indeed, a user could precalculate a number of values of r to be used to sign documents as needed

ElGamal cryptosystem

In 1984, T. ElGamal announced a public-key scheme based on discrete logarithms, closely related to the Diffie-Hellman technique. The ElGamal cryptosystem is used in some form in a number of standards including the digital signature standard (DSS) and the S/MIME email standard. As with Diffie-Hellman, the global elements of ElGamal are a prime number q and a, which is a primitive root of q.

Key Generation

User A generates a private/public key pair as shown.

1. chooses a private -secret key (number): $1 < x_A < q-1$
2. compute their **public key**: $y_A = a^{x_A} \bmod q$.
3. A's private key is X_A and A's public key is $\{q, a, Y_A\}$.

Encryption

B encrypts a message to send to A computing

1. Represent message M in range $0 <= M <= q-1$
   - longer messages must be sent as blocks

2. Chose random integer k with $1 <= k <= q-1$ one-time key $K = y_A^k \bmod q$
3. Encrypt M as a pair of integers (C_1, C_2) where
   $C_1 = a^k \bmod q$; $C_2 = KM \bmod q$

Decryption

A then recovers message by recovering key K as $K = C_{1\ xA} \bmod q$ and computing M as $M = C_2\ K^{-1} \bmod q$

- Note: a unique k must be used each time otherwise result is insecure.

Global Public Elements	
q	prime number
α	$\alpha < q$ and α a primitive root of q

Key Generation by Alice	
Select private X_A	$X_A < q - 1$
Calculate Y_A	$Y_A = \alpha^{X_A} \bmod q$
Public key	$\{q, \alpha, Y_A\}$
Private key	X_A

Encryption by Bob with Alice's Public Key	
Plaintext:	$M < q$
Select random integer k	$k < q$
Calculate K	$K = (Y_A)^k \bmod q$
Calculate C_1	$C_1 = \alpha^k \bmod q$
Calculate C_2	$C_2 = KM \bmod q$
Ciphertext:	(C_1, C_2)

Decryption by Alice with Alice's Private Key	
Ciphertext:	(C_1, C_2)
Calculate K	$K = (C_1)^{X_A} \bmod q$
Plaintext:	$M = (C_2 K^{-1}) \bmod q$

Figure The ElGamal Cryptosystem

For example, let us start with the prime field GF(19); that is, q = 19. It has primitive roots {2, 3, 10, 13, 14, 15}, We choose a = 10.

Use field GF(19) q=19 and a=10

Alice computes her key:

A chooses x_A=5 & computes y_A=10^5 mod 19 = 3

- Bob send message m=17 as (11,5)
  - bychosing random k=6
  - computing $K = y_A^k \bmod q = 3^6 \bmod 19 = 7$
  - computing $C_1 = a^k \bmod q = 10^6 \bmod 19 = 11$;

    $C_2 = KM \bmod q = 7.17 \bmod 19 = 5$
- Alice recovers original message by
  - computing:recover $K = C_1^{xA} \bmod q =$

    $11 \bmod 19 = 7$
  - compute inverse $K^{-1} = 7^{-1} = 11$
  - recover $M = C_2 K^{-1} \bmod q = 5.11 \bmod 19 = 17$
- If a message must be broken up into blocks and sent as a sequence of encrypted blocks, a unique value of k should be used for each block.

ENTITY AUTHENTICATION

- Entity authentication is a technique designed to let one party prove the identity of another party. An entity can be a person, a process, a client, or a server. The entity whose identity needs to be proved is called the claimant; the party that tries to prove the identity of the claimant is called the verifier
- There are two differences between message authentication (data-origin authentication) and entity authentication.
- Message authentication might not happen in real time; entity authentication does.
- Message authentication simply authenticates one message; the process needs to be repeated for each new message. Entity authentication authenticates the claimant for the entire duration of a session

Verification Categories

- **Something Known** – This is a secret known only by the claimant that can be checked by the verifier. Examples are a password, a PIN, a secret key and a private key.
- **Something possessed** – This is something that can prove the claimant's identity.Examples are a passport, a drivers license, a credit card and a smart card.
- **Something inherent** – This is an inherent characteristics of the claimant .Examples are conventional signature, fingerprints, Voice,facial characteristics, retinal pattern and handwriting.

PASSWORDS

- The simplest and oldest method of entity authentication is the password-based authentication, where the password is something that the claimant knows
- Fixed Password
- One-Time Password

Fixed Password

- First approach : The system keeps a table (a file) that is sorted by user identification. To access the system resources ,the user sends the user identification and password, in plaintext, to the system. The system uses the identification to find the password in the table. If it matches, access is granted.

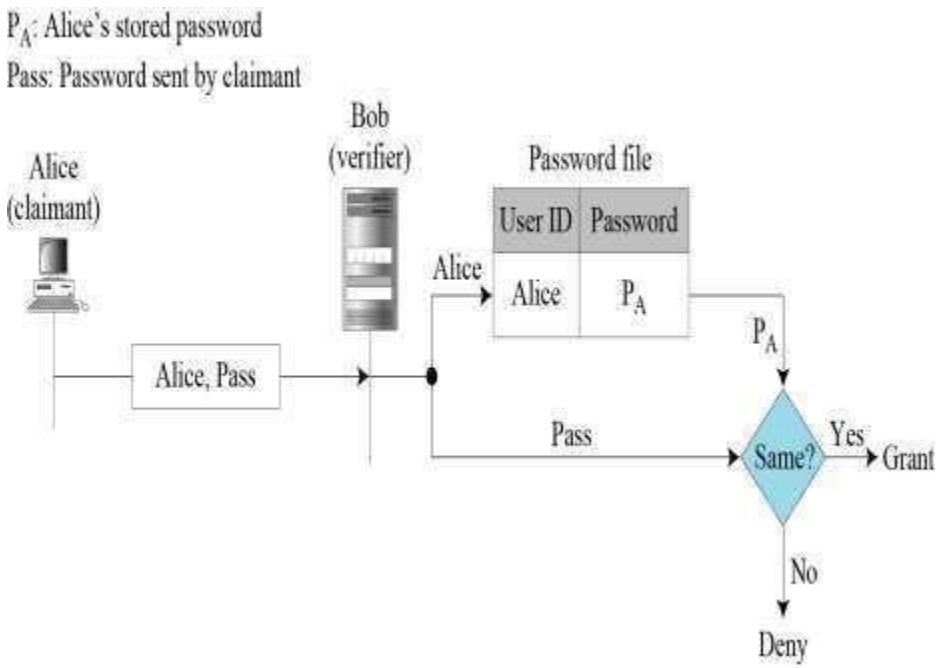


Figure : User ID and password file

- **Attacks on the first approach**
- Eavesdropping
- Stealing the password
- Accessing a password file
- Guessing

- **Second approach** : A more secure approach is to store the hash of the password (instead of plaintext password) in the password file. Any user can read the contents of the file, but, because the hash function is a one-way function. It is almost impossible to guess the value of the password

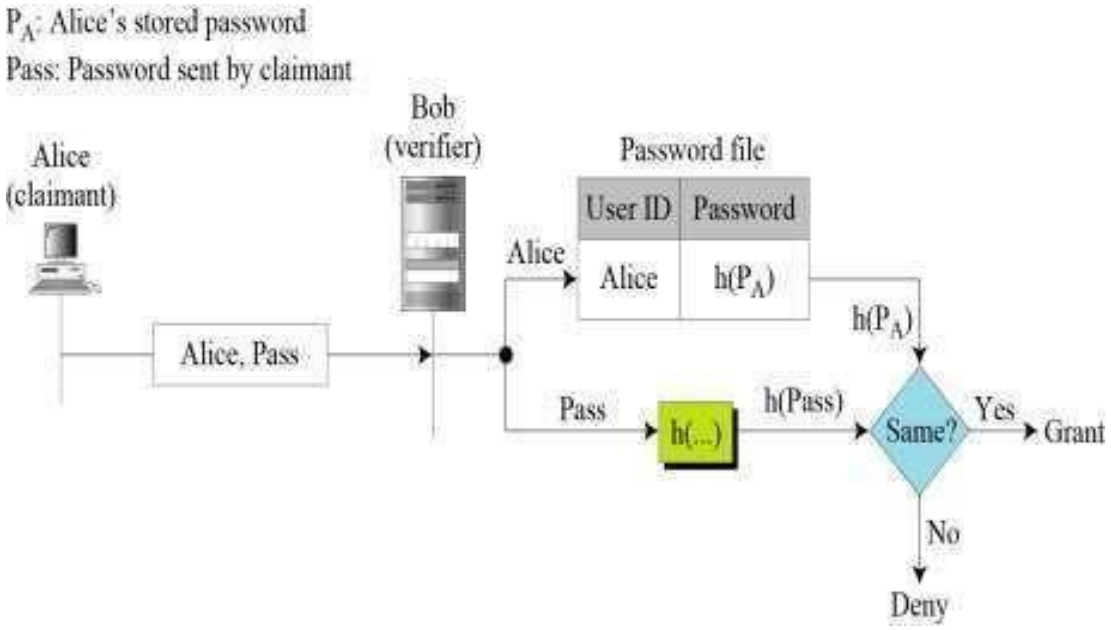


Figure : Hashing the code

Attacks on the second approach

- Eavesdropping

- **Third Approach**: This approach is called salting the password. When the password string is created, a random string, called the salt, is concatenated to the password. The salted password is then hashed. The ID the salt and the hash are then stored in the file. Now, when a user asks for access, the system extracts the salt, concatenates it with the received password, makes a hash out of the result, and compares it with the hash stored in the file.

- The third approach schematic diagram is shown below.

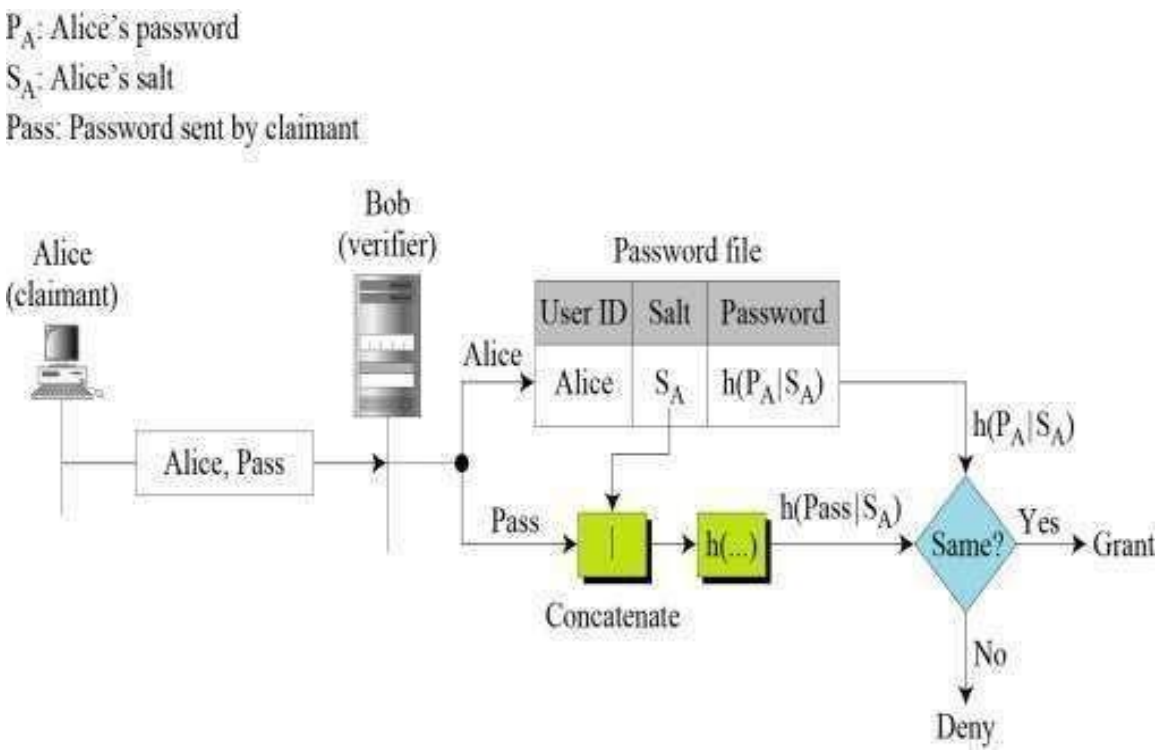


Figure : Salting the password

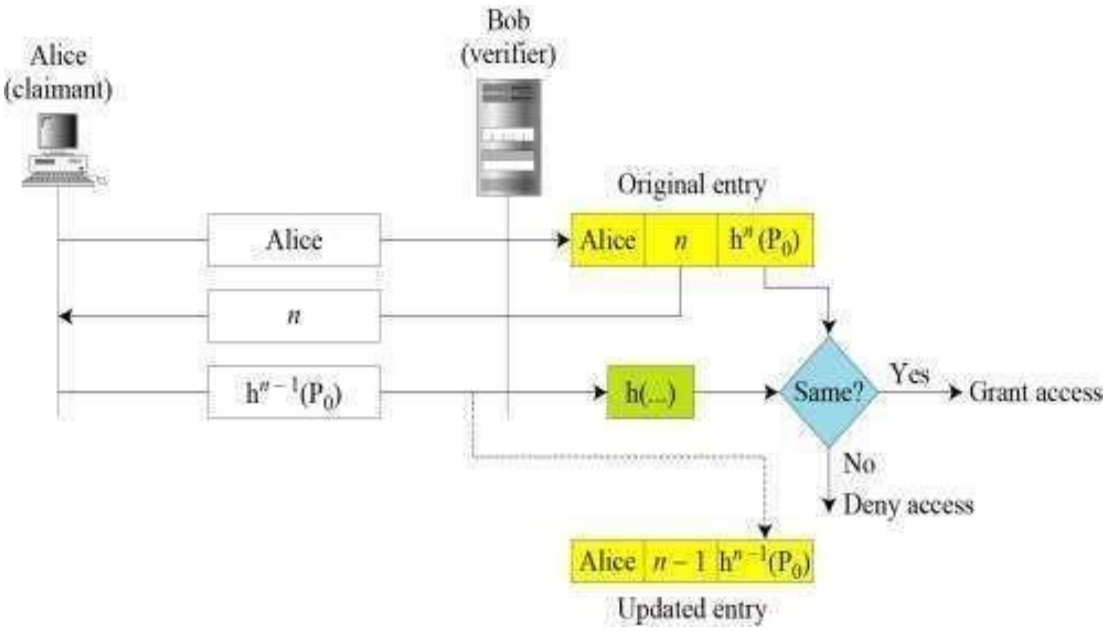
- **Fourth approach** : In the fourth approach, two identification techniques are combined. A good example of this type of authentication is the use of an ATM card with a PIN (personal identification number). The card belongs to the category "something possesses" and the PIN belongs to the category "something known". The PIN is a password that enhamces the security of the card. If the card is stolen,it cannot be used unless the PIN is known.The PIN number, however, is very short so it is easily remembered by the owner. This makes it vulnerable to the guessing type of attack

One-Time Password

- In the first approach, the user and the system agree upon a list of passwords.
- In the second approach, the user and the system agree to sequentially update the password.
- In the third approach, the user and the system create a sequentially updated password using a hash function.

$$h^n(x) = h(h^{n-1}(x)) \quad h^{n-1}(x) = h(h^{n-2}(x)) \quad \ldots \quad h^2(x) = h(h(x)) \quad h^1(x) = h(x)$$

Lamport one-time password



CHALLENGE-RESPONSE

- In password authentication, the claimant proves her identity by demonstrating that she knows a secret, the password. In challenge-response authentication, the claimant proves that she knows a secret without sending it. In challenge-response authentication, the claimant proves that she knows a secret without sending it to the verifier. The challenge is a time-varying value sent by the verifier; the response is the result of a function applied on the challenge.
- Four ways in Challenge Response

1) Using a Symmetric-Key Cipher
2) Using Keyed-Hash Functions
3) Using an Asymmetric-Key Cipher
4) Using Digital Signature

- **Using a Symmetric-Key Cipher**

First Approach:

In the first approach, the verifier sends a nonce, a random numberused only once, to challenge the claimant. A nonce must be time-varying every time it is created, it is different. The claimant responds to the challenge using the secret key shared between the claimant and the verifier.

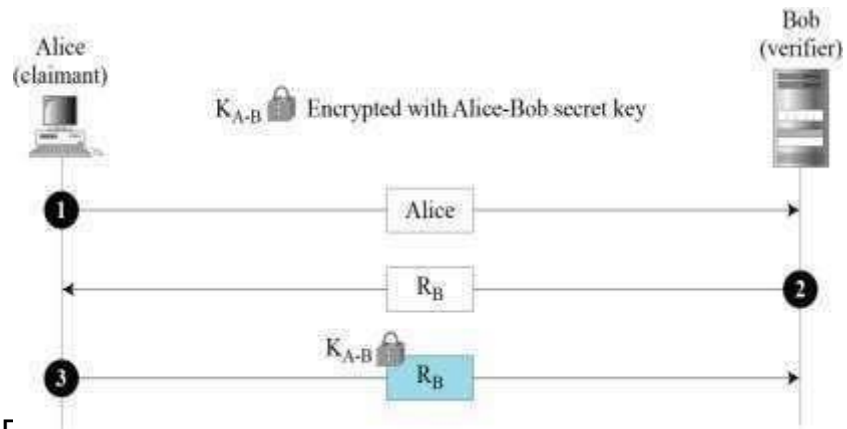


Figure : Nonce challenge

- **Second Approach**
- In the second approach, the time-varying value is a timestamp, which changes with time. In this approach the challenge message is the current time sent from the verifier to the claimant. If the client and server clocks are synchronized, the claimant knows the current time. This means that there is no need for the challenge message . The first and the third message can be combined. The result is that authentication can be done using one message.

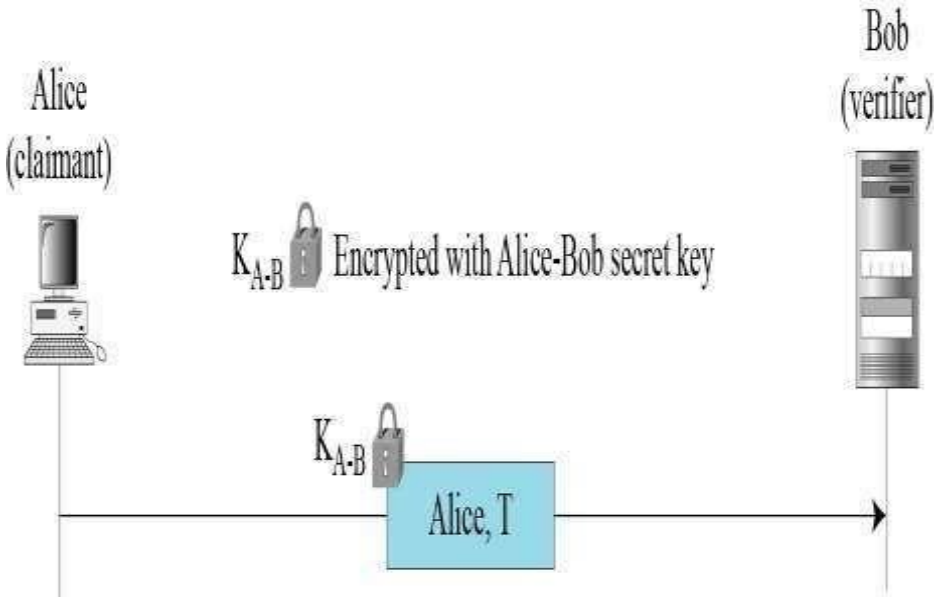


Figure : Timestamp Challenge

- **Third Approach**
- In the third approach, bidirectional authentication is used. The first and second approaches are for unidirectional authentication. Alice is authenticated to Bob but not the other way around. If Alice also needs to be sure about Bob's identity, we need bidirectional authentication.

- The third Approach is shown below

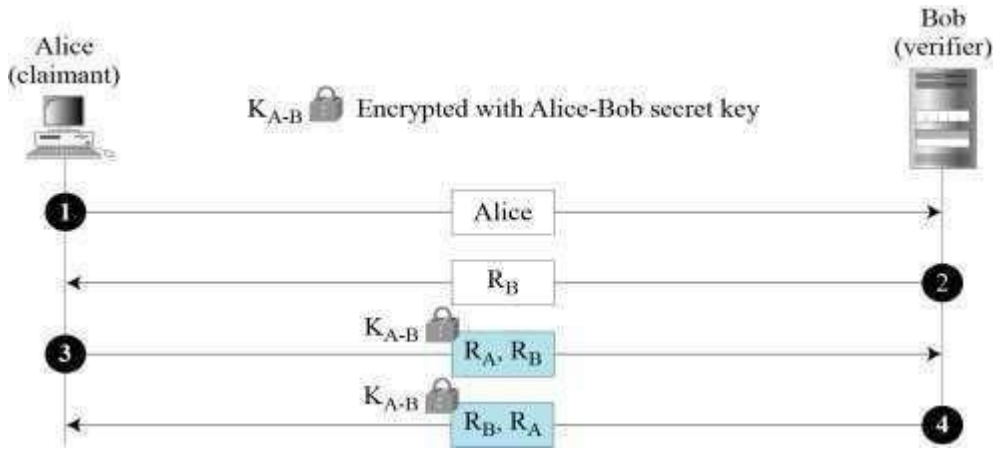


Figure : Bidirectional authentication

Using Keyed-Hash Functions

- Instead of using encryption/decryption for entity authentication, we can also use a keyed-hash function (MAC). The advantage is that it preserves the integrity of challenge and response messages and at the time uses a secret key. The timestamp is sent both as plaintext and as text scrambled by the keyed-hash function. When Bob receives the message, he takes the plaintext T, applies the keyed-hash function and then compares his calculation with what he received to determine the authenticity of Alice.

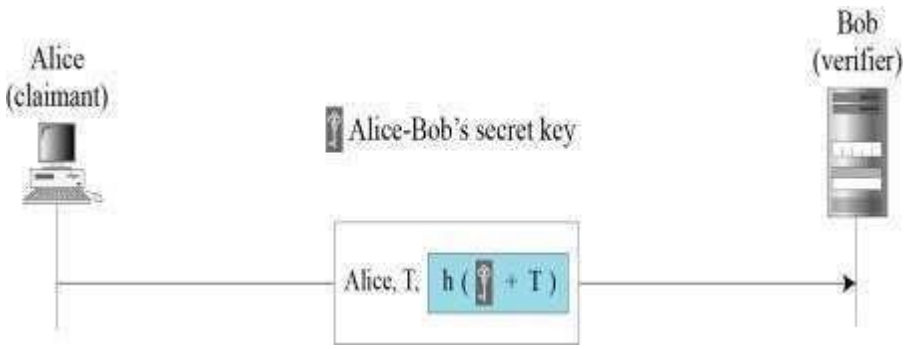
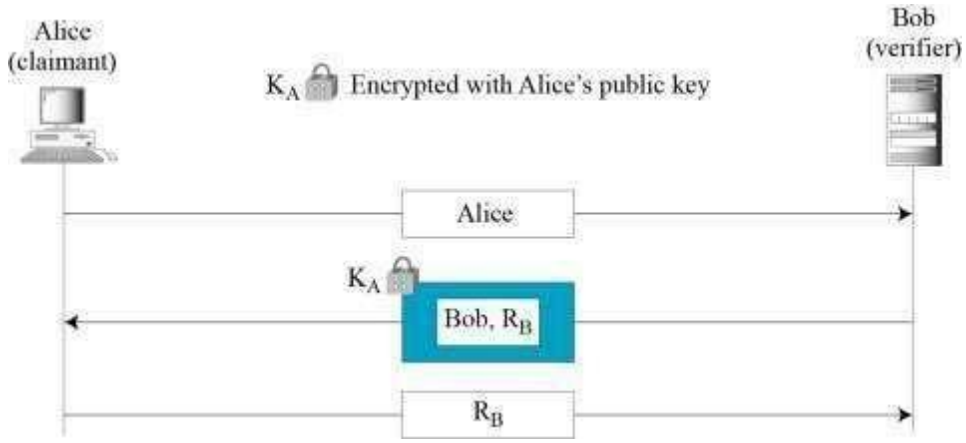


Figure : Keyed-hash function

Using an Asymmetric-Key Cipher

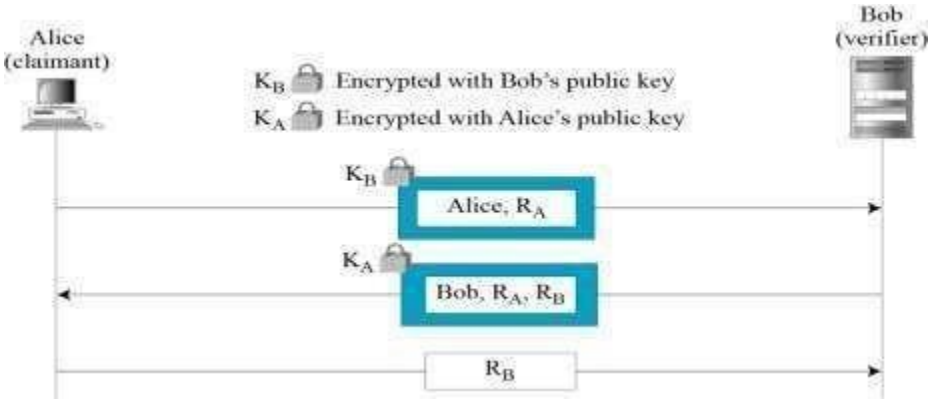
- Here the secret key must be the private key of the claimant. The claimant must show that she owns the private key related to the public key that is available to everyone. This means that the verifier must encrypt the challenge using the public key of the claimant. The claimant then decrypts the message using her private key. The response to the challenge is the decyypted challenge. Following are two approaches: one for unidirectional authentication and one for bidirectional authentication.
- In the first approach, Bob encrypts the challenge using Alice's public key. Alice decrypts the message with her private key and sends the nonce to Bob.



- Figure : Unidirectional, asymmetric-key authentication

- In the second approach, two public keys are used, one in each direction. Alice sends her identity and nonce encrypted with Bob's public key. Bob responds with his nonce encrypted with Alice's public key. Finally, Alice responds with Bob's decrypted nonce.

- The second approach using bidirectional, asymmetric-key is shown below.



- Figure : Bidirectional, asymmetric-key

Using Digital Signature

- Entity authentication can also be achieved using a digital signature. When a digital signature is used for entity authentication, the claimant uses her private key for signing. Two approaches are shown here.
- First Approach:

  In the first approach, Bob uses a plaintext challenge and Alice signs the response.

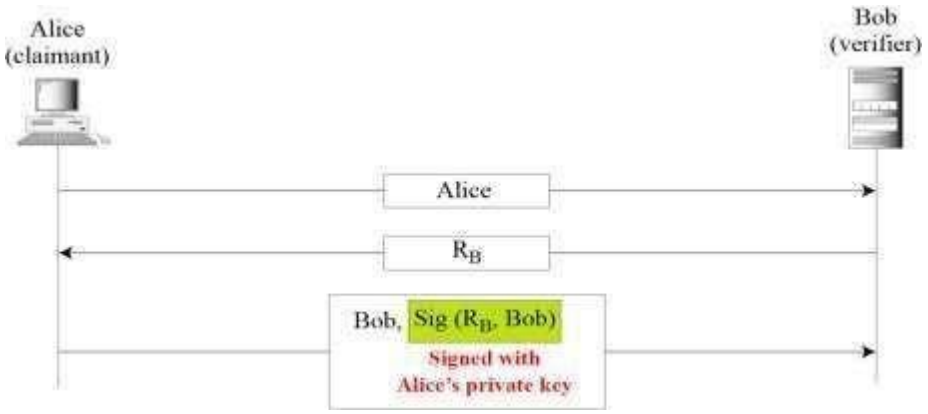


Figure : Digital signature, unidirectional

- Second Approach:

  In the second approach, Alice and Bob authenticate each other.

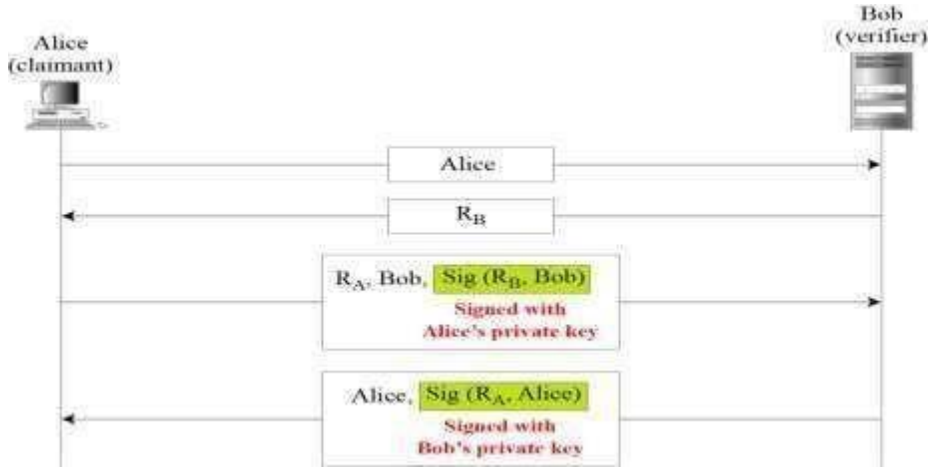


Figure : Digital signature, bidirectional authentication

BIOMETRICS

- Biometrics is the measurement of physiological or behavioral features that identify a person (authentication by something inherent). Biometrics measures features that cannot be guessed, stolen, or shared. Several components are needed for biometrics, including capturing devices, processors, and storage devices..
- **Enrollment** : Before using any biometric techniques for authentication, the corresponding feature of each person in the community should be available in the database. This is referred to as enrollment.
- **Authentication:** Authentication is done by verification or identification
- **Verification** : In verification, a person's feature is matched against a single record in the database (one-to-one matching) to find if she is who she is claiming to be. This is useful when a bank needs to verify a customer's signature on a check.

- **Identification :** In identification, a person's feature is matched against all records in the database(one-to-many matching) to find if she has a record in the database. This is useful when a company needs to allow access to the building only to employees.

- **BIOMETRIC TECHNIQUES**
- Biometric techniques are divided into two main categories : Physiological and Behaviorial.

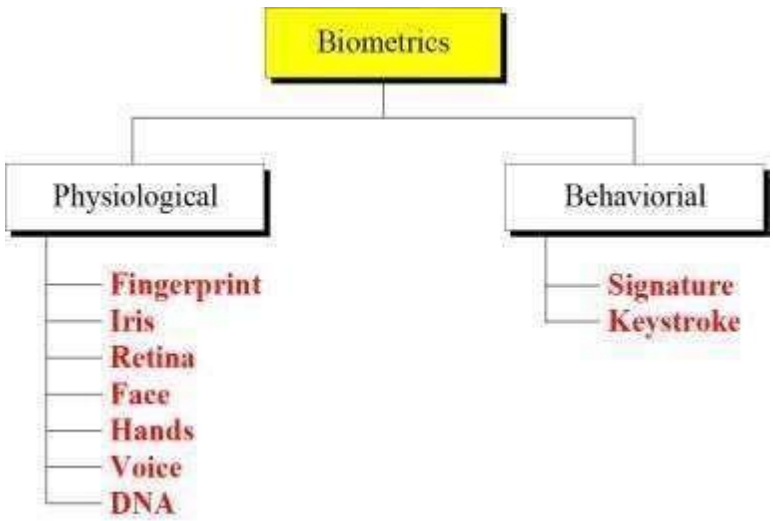


Figure : Biometric Techniques

Requirements for an Ideal Biometric Identifier

1. Universality – Every person should have the biometric characteristic
2. Uniqueness - No two persons should be the same in terms of the biometric characteristic
3. Performance - The biometric characteristic should be invariant over time

4. Collectability - The biometric characteristic should be measurable with some (practical) sensing device
5. Acceptability - One would want to minimize the objections of the users to the measuring/collection of the biometric

- DNA (Deoxyribo Nucleic Acid)
  One-dimensional unique code for one's individuality, but identical twins have identical DNA patterns
- Issues limiting the utility of DNA
  - Contamination
  - Access
  - Automatic real-time recognition issues
  - Privacy issues: information about susceptibilities of a person to certain diseases could be gained from the DNA pattern

- **FINGERPRINT RECOGNITION**

There are three levels on which Fingerprint Recognition carry out.

- Level 1 : Identify the pattern of Fingerprint
- Level 2 : Based on ridge characteristics i.e. ridge minutiae
- Level 3 : Based on shape, size of ridges and pores

- **Advantages of Fingerprint Biometrics**

Fingerprint pattern stable through out the lifetime
Fingerprints are unique in nature. It is easily analyzed and compare
Inexpensive device, Oldest form of biometrics

Limitations of Fingerprint Biometrics

It is not right tool for those persons who working in chemical labs

- **FACE RECOGNITION**
  - Capture Image
  - Find Face in Image
  - Features Extract (store template)
  - Compare Template
  - Declare Match

- **HAND GEOMETRY**

  Hand or fingers geometry is an automated measurement of many dimension of hand and fingers

- **IRIS RECOGNITION**

  Iris scanning measures the iris pattern in the colored part of the eye.

- **RETINA RECOGNITION**

  Images back of the eye and compare blood vessels with existing date.

- **VOICE/SPEAKER RECOGNITION**

  It is a Behavioral Trait. Voice or speaker recognition uses vocal characteristics to recognize individual. A telephone or microphone can act as a sensor.

- **SIGNATURE VERIFICATION**

  It is a Behavioral Trait. An automated method of measuring an individual signature. This technology examine speed, direction, pressure of stylus while writing, the time that the stylus is in and out of contact with the paper/tablet

KEYSTROKES DYNAMICS

It is a Behavioral Trait . Keystrokes dynamics is an automated method of examining an individual's keystrokes on a 'keyboard'. This technology examine such as speed, pressure, total time taken to type particular words and time elapsed between hitting certain keys

ADVANTAGES OF BIOMETRICS

- Effective technique to enhance security.
- User friendly.
- Does not use any password, PIN or secret code that are compromised.
- It used physical and behavioral traits for identification and authentication that are difficult to compromised.
- Long lasting performance, because biometrics traits are permanence in nature.
- Difficult to fool biometrics system.

DIS-ADVANTAGES OF BIOMETRICS

- Retina recognition required closed physical contact of the scanning
- device which may not be generally accepted by public.
- Costly
- Voice recognition requires large amount of computer storage, peoples voice can change, background noise can interfere.
- Biometric features may changes over time.
- Signature recognition has poor long term reliability, accuracy difficulty to ensure.

Kerberos

- trusted key server system from MIT
- provides centralised private-key third-party authentication in a distributed network
  - allows users access to services distributed through out the network
  - without needing to trust all workstations
  - rather all trust a central authentication server
- two versions in use: Kerberos 4 & Kerberos 5

Kerberos Requirements

- **Security-** A network eavesdropper should not be able to obtain the necessary information to impersonate a user. More generally,Kerberos should be strong enough that a potential opponent does not find it to be the weak link.
- **Reliability** - : For all services that rely on Kerberos for access control, lack of availability of the Kerberos service means lack of availability of the supported services. Hence, Kerberos should be highly reliable and should employ a distributed server architecture with one system able to back up another.
- **Transparency** - Ideally, the user should not be aware that authentication is taking place beyond the requirement to enter a password
- **Scalability** - The system should be capable of supporting large numbers of client and servers.This suggests a modular, distributed architecture.

Kerberos 4 Overview

- a basic third-party authentication scheme
- have an Authentication Server (AS)
  - users initially negotiate with AS to identify themselves
  - AS provides a non-corruptible authentication credential (ticket granting ticket TGT)
- have a Ticket Granting server (TGS)
  - users subsequently request access to other services from TGS on basis of users TGT

A Simple Authentication Dialogue

(1) C -> AS : ID_C || P_C || ID_V

C = client

AS = authentication server

ID_C = identifier of user on C

P_C = password of user on C

ID_V = identifier of server V

C asks user for the password

AS checks that user supplied the right password

An authentication server (AS) that knows the passwords of all users and stores these in a centralized database. In addition, the AS shares a unique secret key with each server

Message 2

(2) AS -> C : Ticket

Ticket = $E_{K(V)}$ [ID_C || AD_C || ID_V]

K(V) = secret encryption key shared by AS and V

AD_C = network address of C

Ticket cannot be altered by C or an adversary

Message 3

(3) C -> V: ID_C | | Ticket

Server V decrypts the ticket and checks various fields

AD_C in the ticket binds the ticket to the network address of C

- The user logs on to a workstation and requests access to server V. The client module C in the user's workstation requests the user's password and then sends a message to the AS that includes the user's ID, the server's ID, and the user's password. The AS checks its database to see if the user has supplied the proper password for this user ID and whether this user is permitted access to server V. If both tests are passed, the AS accepts the user as authentic an must now convince the server that this user is authentic.To do so, the AS creates a **ticket** that contains the user's ID and network address and the server's ID. This ticket is encrypted using the secret key shared by the AS and this server.This ticket is then sent back to C. Because the ticket is encrypted, it cannot be altered by C or by an opponent. With this ticket, C can now apply to V for service. C sends a message to V containing C's ID and the ticket. V decrypts the ticket and verifies that the user ID in the ticket is the same as the unencrypted user ID in the message. If these two match, the server considers the user authenticated and grants the requested service

Problems:

- Each time a user needs to access a different service he/she needs to enter their password
  - Read email several times
  - Print, mail, or file server
  - Assume that each ticket can be used only once (otherwise open to replay attacks)
- Password sent in the clear

Authentication Dialogue II

- **Once per user logon session**

  (1) C -> AS: ID_C || ID_{TGS}

  (2)AS -> C: $E_{K(C)}$ [$Ticket_{TGS}$]

  $Ticket_{TGS}$ is equal to

  $E_{K(TGS)}$ [ID_C || AD_C || ID_{TGS} || TS_1 || $Lifetime_1$]

  where,

  TGS = Ticket-granting server

  ID_{TGS} = Identifier of the TGS

  $Ticket_{TGS}$ = Ticket-granting ticket or TGT

  TS_1 = timestamp

  $Lifetime_1$ = lifetime for the TGT

  $K_{(C)}$ = key derived from user's password

- **Once per type of service**

  (3) C -> TGS: ID_C || ID_V || $Ticket_{TGS}$

  (4)TGS -> C : $Ticket_V$

  $Ticket_V$ is equal to

  $E_{K(V)}$ [ID_C || AD_C || ID_V || TS_2 || $Lifetime_2$]

  Where,

  K(V): key shared between V and

  TGS is called the service-granting ticket (SGT)

- **Once per service session**

  (5) C -> V: ID_C || $Ticket_V$

  C says to V "I am ID_C and have a ticket from the TGS" . Let me in!

- The details of the scheme is as follows:

1. The client requests a ticket-granting ticket on behalf of the user by sending its user's ID to the AS, together with the TGS ID, indicating a request to use the TGS service.
2. The AS responds with a ticket that is encrypted with a key that is derived from the user's password ( ), which is already stored at the AS.When this response arrives at the client, the client prompts the user for his or her password, generates the key, and attempts to decrypt the incoming message. If the correct password is supplied, the ticket is successfully recovered.

- Now that the client has a ticket-granting ticket, access to any server can be obtained with steps 3 and 4.

3. The client requests a service-granting ticket on behalf of the user. For this purpose, the client transmits a message to the TGS containing the user's ID, the ID of the desired service, and the ticket-granting ticket.

- 4.The TGS decrypts the incoming ticket using a key shared only by the AS and the TGS (K_{tgs}) and verifies the success of the decryption by the presence of its ID. It checks to make sure that the lifetime has not expired.Then it compares the user ID and network address with the incoming information to authenticate the user. If the user is permitted access to the server V, the TGS issues a ticket to grant access to the requested service.

- Finally, with a particular service-granting ticket, the client can gain access to the corresponding service with step 5.

5. The client requests access to a service on behalf of the user. For this purpose, the client transmits a message to the server containing the user's ID and the service-granting ticket. The server authenticates by using the contents of th ticket.

This new scenario satisfies the two requirements of only one password query per user session and protection of the user password.

KERBEROS REALMS AND MULTIPLE KERBERI

- A full-service Kerberos environment consisting of a Kerberos server, a number of clients, and a number of application servers requires the following:

1. The Kerberos server must have the user ID and hashed passwords of all participating users in its database. All users are registered with the Kerberos server.
2. The Kerberos server must share a secret key with each server. All servers are registered with the Kerberos server.

Such an environment is referred to as a Kerberos realm.

3. The Kerberos server in each interoperating realm shares a secret key with the server in the other realm.The two Kerberos servers are registered with each other.

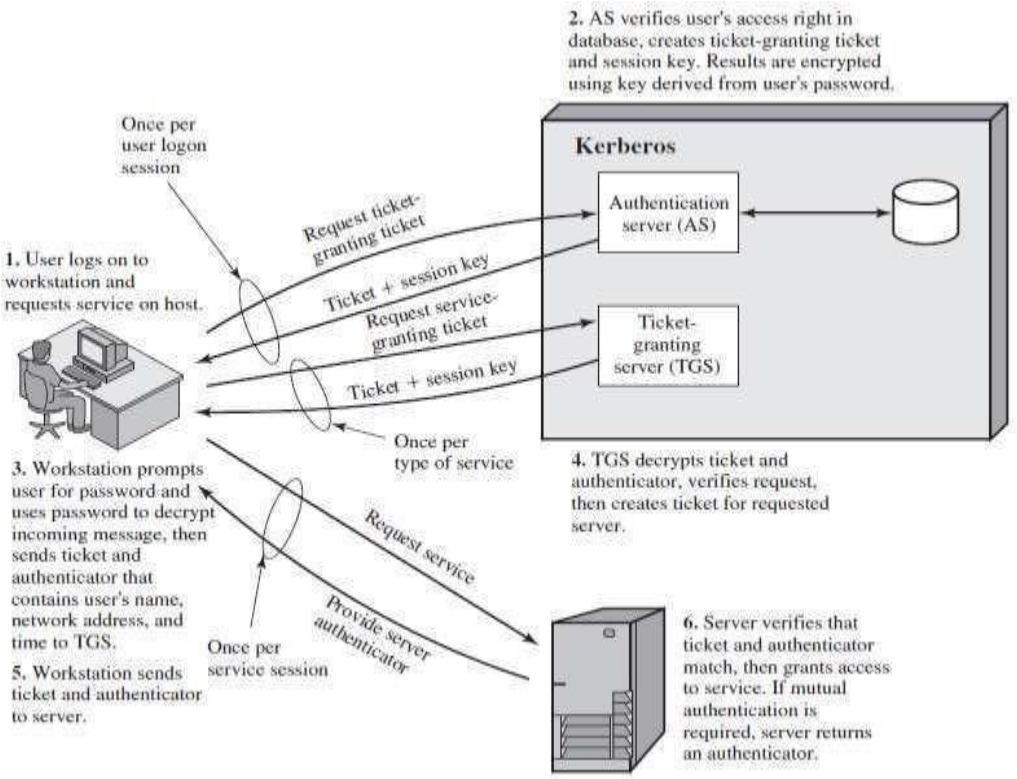


Figure: Overview of Kerberos

- The concept of realm can be explained as follows. A Kerberos realm is a set of managed nodes that share the same Kerberos database. The Kerberos database resides on the Kerberos master computer system, which should be kept in a physically secure room. A read-only copy of the Kerberos database might also reside on other Kerberos computer systems. However, all changes to the database must be made on the master computer system. Changing or accessing the contents of a Kerberos database requires the Kerberos master password. A related concept is that of a Kerberos principal, which is a service or user that is known to the Kerberos system. Each Kerberos principal is identified by its principal name. Principal names consist of three parts: a service or user name, an instance name, and a realm name
- A user wishing service on a server in another realm needs a ticket for that server. The user's client follows the usual procedures to gain access to the local TGS and then requests a ticket-granting ticket for a remote TGS (TGS in another realm).The client can then apply to the remote TGS for a service-granting ticket for the desired server in the realm of the remote TGS. The details of the exchanges illustrated below:

$$
\begin{aligned}
&\textbf{(1)}\ \mathrm{C} \rightarrow \mathrm{AS}: && ID_c \,\|\, ID_{tgs} \,\|\, TS_1 \\
&\textbf{(2)}\ \mathrm{AS} \rightarrow \mathrm{C}: && \mathrm{E}(K_c, [K_{c,tgs} \,\|\, ID_{tgs} \,\|\, TS_2 \,\|\, Lifetime_2 \,\|\, Ticket_{tgs}]) \\
&\textbf{(3)}\ \mathrm{C} \rightarrow \mathrm{TGS}: && ID_{tgsrem} \,\|\, Ticket_{tgs} \,\|\, Authenticator_c \\
&\textbf{(4)}\ \mathrm{TGS} \rightarrow \mathrm{C}: && \mathrm{E}(K_{c,tgs}, [K_{c,tgsrem} \,\|\, ID_{tgsrem} \,\|\, TS_4 \,\|\, Ticket_{tgsrem}]) \\
&\textbf{(5)}\ \mathrm{C} \rightarrow \mathrm{TGS}_{\mathrm{rem}}: && ID_{vrem} \,\|\, Ticket_{tgsrem} \,\|\, Authenticator_c \\
&\textbf{(6)}\ \mathrm{TGS}_{\mathrm{rem}} \rightarrow \mathrm{C}: && \mathrm{E}(K_{c,tgsrem}, [K_{c,vrem} \,\|\, ID_{vrem} \,\|\, TS_6 \,\|\, Ticket_{vrem}]) \\
&\textbf{(7)}\ \mathrm{C} \rightarrow \mathrm{V}_{\mathrm{rem}}: && Ticket_{vrem} \,\|\, Authenticator_c
\end{aligned}
$$

- The ticket presented to the remote server (Vrem:) indicates the realm in which the user was originally authenticated. The server chooses whether to honor the remote request.

Kerberos Version 5

- Kerberos version 5 is specified in RFC 4120 and provides a number of improvements over version 4 .
- THE VERSION 5 AUTHENTICATION DIALOGUE

(a) Authentication Service Exchange to obtain ticket-granting ticket

(1) C: AS Options || IDc || Realmc || IDtgs || Times || Nonce1

(2) AS: C RealmC || IDC || Tickettgs || E(Kc, [Kc,tgs || Times || Nonce1 || Realmtgs || IDtgs])

Tickettgs = E1Ktgs, [Flags || Kc,tgs || Realmc || IDC || ADC || Times]

(b) Ticket-Granting Service Exchange to obtain service-granting ticket

(3) C: TGS Options || ID_V || *Times* || $Nonce_2$ || $Ticket_{tgs}$ || $Authenticator_C$

(4) TGS: C $Realm_C$ || ID_C || $Ticket_V$ || $E(K_{C,tgs}, [K_{C,V}$ || *Times* || $Nonce_2$ || $Realm_V$ || $ID_V])$

$Ticket_{tgs} = E1K_{tgs}$, [*Flags* || $K_{C,tgs}$ || $Realm_C$ || ID_C || AD_C || *Times*]2

$Ticket_V = E1K_V$, [*Flags* || $K_{C,V}$ || $Realm_C$ || ID_C || AD_C || *Times*]2

$Authenticator_C = E(K_{C,tgs}$, [$ID_C$ || $Realm_C$ || TS_1])

(c) Client/Server Authentication Exchange to obtain service

(5) C: V Options **|| $Ticket_V$ || *$Authenticator_C$***

(6) V: C **$E_{Kc,V}$ [TS_2 || *Subkey* || *SeqZ*]**

$Ticket_V = E(K_V$, [*Flag* || $K_{C,V}$ || $Realm_C$ || ID_C || AD_C || *Times*])

$Authenticator_C = E1K_{C,V}$, [$ID_C$ || $Relam_C$ || TS_2 || *Subkey* || *SeqZ*]

- First, consider the authentication service exchange. Message (1) is a client request for a ticket-granting ticket. As before, it includes the ID of the user and the TGS.

- The following new elements are added:
  - Realm: Indicates realm of user
  - Options: Used to request that certain flags be set in the returned ticket
  - Times: Used by the client to request the following time settings in the ticket:
    —from: the desired start time for the requested ticket
    —till: the requested expiration time for the requested ticket
    —rtime: requested renew-till time
  - Nonce: A random value to be repeated in message (2) to assure that the response is fresh and has not been replayed by an opponent
- Message (2) returns a ticket-granting ticket, identifying information for the client, and a block encrypted using the encryption key based on the user's password. This block includes the session key to be used between the client and the TGS, times specified in message (1), the nonce from message (1), and TGS identifying information. The ticket itself includes the session key, identifying information for the client, the requested time values, and flags that reflect the status of this ticket and the requested options. These flags introduce significant new functionality to version 5.

- **Kerberos Version 5 Flags**

INITIAL - This ticket was issued using the AS protocol and not issued based on a ticket granting ticket.

PRE-AUTHENT - During initial authentication, the client was authenticated by the KDC before a ticket was issued.

HW-AUTHENT - The protocol employed for initial authentication required the use of hardware expected to be possessed solely by the named client.

RENEWABLE - Tells TGS that this ticket can be used to obtain a replacement ticket that expires at a later date.

MAY-POSTDATE - Tells TGS that a postdated ticket may be issued based on this ticket-granting ticket.

POSTDATED - Indicates that this ticket has been postdated; the end server can check the authtime field to see when the original authentication occurred.

INVALID - This ticket is invalid and must be validated by the KDC before use.

PROXIABLE - Tells TGS that a new service-granting ticket with a different network Address may be issued based on the presented ticket.

PROXY - Indicates that this ticket is a proxy.

FORWARDABLE - Tells TGS that a new ticket-granting ticket with a different network . Address may be issued based on this ticket-granting ticket.

FORWARDED - Indicates that this ticket has either been forwarded or was issued based on authentication involving a forwarded ticket-granting ticket.

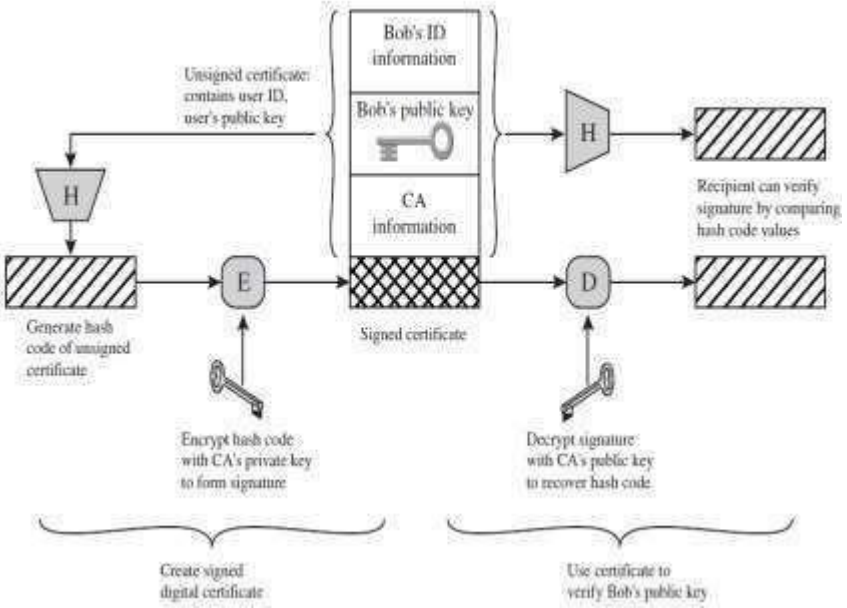
X.509 CERTIFICATES

- X.509 defines a framework for the provision of authentication services by the X.500 directory to its users. The directory may serve as a repository of public-key certificates. X.509 is based on the use of public-key cryptography and digital signatures.

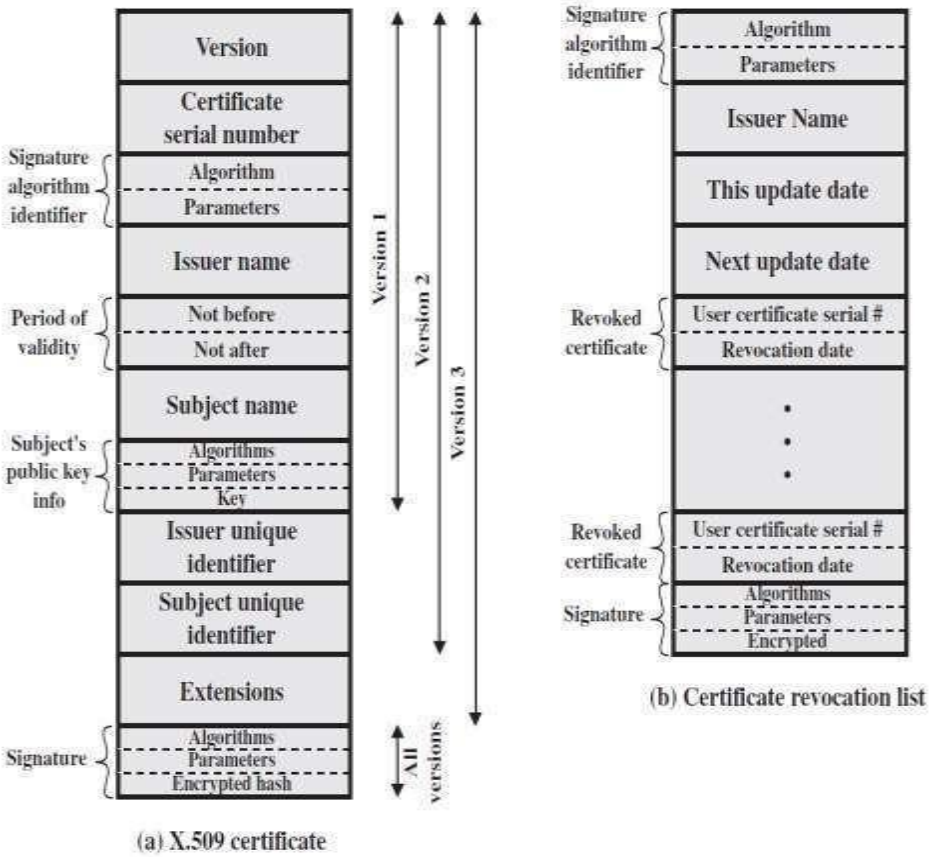
Certificates

- The heart of the X.509 scheme is the public-key certificate associated with each user. These user certificates are assumed to be created by some trusted certification authority (CA) and placed in the directory by the CA or by the user.

- Figure below illustrates the generation of a public-key certificate.



- Figure below illustrates the X.509 Formats



- Version: Differentiates among successive versions of the certificate format; the default is version 1. If the issuer unique identifier or subject unique identifier are present, the value must be version 2. If one or more extensions are present, the version must be version 3.
- Serial number: An integer value unique within the issuing CA that is unambiguously associated with this certificate.
- Signature algorithm identifier: The algorithm used to sign the certificate together with any associated parameters. Because this information is repeated in the signature field at the end of the certificate, this field has little, if any, utility.
- Issuer name: X.500 is the name of the CA that created and signed this certificate.
- Period of validity: Consists of two dates: the first and last on which the certificate is valid.
- Subject name: The name of the user to whom this certificate refers.That is, this certificate certifies the public key of the subject who holds the corresponding private key.
- Subject's public-key information: The public key of the subject, plus an identifier of the algorithm for which this key is to be used, together with any associated parameters.
- Issuer unique identifier: An optional-bit string field used to identify uniquely the issuing CA in the event the X.500 name has been reused for different entities
- Subject unique identifier: An optional-bit string field used to identify uniquelythe subject in the event the X.500 name has been reused for different entities.
- Extensions: A set of one or more extension fields. Extensions were added in version 3 and are discussed later in this section.
- Signature: Covers all of the other fields of the certificate; it contains the hash code of the other fields encrypted with the CA's private key.This field includes the signature algorithm identifier.

- The standard uses the following notation to define a certificate:

$$CA \ll A \gg = CA\{V, SN, AI, CA, UCA, A, UA, Ap, T^A\}$$

where

$Y \ll X \gg$ = the certificate of user X issued by certification authority Y

$Y\{I\}$ = the signing of I by Y. It consists of I with an encrypted hash code appended

V = version of the certificate

SN = serial number of the certificate

AI = identifier of the algorithm used to sign the certificate

CA = name of certificate authority

UCA = optional unique identifier of the CA

A = name of user A

UA = optional unique identifier of the user A

Ap = public key of user A

T^A = period of validity of the certificate

The CA signs the certificate with its private key. If the corresponding public key is known to a user, then that user can verify that a certificate signed by the CA is valid.

OBTAINING A USER'S CERTIFICATE

User certificates generated by a CA have the following characteristics:

- Any user with access to the public key of the CA can verify the user public key that was certified.
- No party other than the certification authority can modify the certificate without this being detected.

- Now suppose that A has obtained a certificate from certification authority and B has obtained a certificate from CA X2. If A does not securely know the public key of X2, then B's certificate, issued by X2, is useless to A.A can read B's certificate, but A cannot verify the signature. However, if the two CAs have securely exchanged their own public keys, the following procedure will enable A to obtain B's public key.
- Step 1 A obtains from the directory the certificate of X2 signed by X1. Because A securely knows X1's public key,A can obtain X2's public key from its certificate and verify it by means of X1's signature on the certificate.
- Step 2 A then goes back to the directory and obtains the certificate of B signed by X2. Because A now has a trusted copy of X2's public key,A can verify the signature and securely obtain B's public key. A has used a chain of certificates to obtain B's public key. In the notation of
- X.509, this chain is expressed as

$$X_1 \ll X_2 \gg X_2 \ll B \gg$$

- In the same fashion, B can obtain A's public key with the reverse chain:

$$X_2 \ll X_1 \gg X_1 \ll A \gg$$

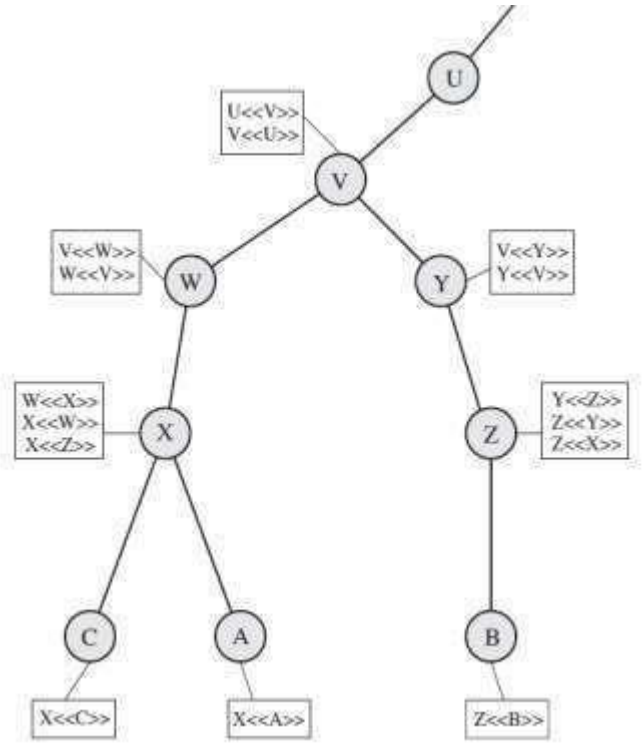
- This scheme need not be limited to a chain of two certificates. An arbitrarily long path of CAs can be followed to produce a chain. A chain with N elements would be expressed as

$$X_1 \ll X_2 \gg X_2 \ll X_3 \gg \ldots X_N \ll B \gg$$

The directory entry for each CA includes two types of certificates:

- **Forward certificates**: Certificates of X generated by other CAs
- **Reverse certificates**: Certificates generated by X that are the certificates of other CAs

- X.509 suggests that CAs be arranged in a hierarchy as shown below



- In the above example, user A can acquire the following certificates from the directory to establish a certification path to B.

- **REVOCATION OF CERTIFICATES**

It may be desirable on occasion to revoke a certificate before it expires, for one of the following reasons.

1. The user's private key is assumed to be compromised.
2. The user is no longer certified by this CA. Reasons for this include that the subject's name has changed, the certificate is superseded, or the certificate was not issued in conformance with the CA's policies.
3. The CA's certificate is assumed to be compromised.

Each CA must maintain a list consisting of all revoked but not expired certificates issued by that CA, including both those issued to users and to other CAs. These lists should also be posted on the directory.

PART -A

1. Define Authentication (or) What is message Authentication?

It is a procedure that verifies whether the received message comes from assigned source has not been altered.It uses message authentication codes, hash algorithms to authenticate the message.

2. List the authenticationrequirements.

Disclosure – releases of message contents to any person or process not possessing the appropriate cryptographic key.

Traffic analysis – discovery of the pattern of traffic between parties.

Masquerade – insertion of messages into the network fraudulent source.

Timing modification – delay or replay of messages.

Source repudiation – denial of transmission of message by source.

Destination repudiation – denial of transmission of message by destination.

3. List the three types of authentication functions.

The different types of functions that may be used to produce an **authenticator** are as follows:

Message encryption – the cipher text of the entire message serves as its authenticator.

Message authentication code (MAC) – a public function of the message and a secret key that produces a fixed length value serves as the authenticator.

Hash function – a public function that maps a message of any length into a fixed length hash value, which serves as the authenticator

4. What is confidentiality? How it differs from authentication .

- Confidentiality is a set of rules or a promise that limits access or places restrictions on certain types of information.

- Authentication is the process of ensuring or confirming your own identity.

5. Define MAC (or) What is message authentication?

MAC stands for Message Authentication Code.

MAC involves the use of secret key to generate a small fixed size block of data, known as cryptographic checksum or MAC that is appended to the message. This technique assumes that two communication parties say A and B, share a common secret key 'k'. When A has to send a message 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

6. Define hash function

A **hash function** H accepts a variable-length block of data as input and producesa fixed-size hash value h=H(M) .A "good" hash function has the property that the results of applying the function to a large set of inputs will produce outputs that are evenly distributed and apparently random.

The kind of hash function needed for security applications is referred to as a **cryptographic hash function.**

7. Define Weak-collision resistance.

For any given block x, it is computationally infeasible to find y!=x(not equal) with H(y)=H(x).

It is proportional to 2^n

8. Define Weak-collision resistance.

It is computationally infeasible to find any pair (x,Y) such that H(y)=H(x).

It is proportional to $2^{n/2}$

9. Define one way property of hash function.

The one way property of a hash function states that for any given hash code h, , it is computationally infeasible to find y= x with H(y)=H(x).

10. Specify the properties of hash function.

Hash function H have the following properties:

H can be applied to a block of data of any size

H produces a fixed length output

H(x) is relatively easy to compute for any given x,making both hardware and software implementations practical.

11. Difference between MAC and one way hash function.

MAC involves the use of secret key to generate a small fixed size block of data, known as cryptographic checksum or MAC that is appended to the message. This technique assumes that two communication parties say A and B, share a common secret key 'k'. When A has to send a message 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

The one way property of a hash function states that for any given hash code h, , it is computationally infeasible to find y= x with H(y)=H(x).

- **12. Difference between direct digital signature and arbitrated digital signature.**
- The direct digital signature involves only the communicating parties (source, destination). It is assumed that the destination knows the public key of the source. 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.
- 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 sent to Y with an indication that it has been verified to the satisfaction of the arbiter. The presence of A solves the problem faced by direct signature schemes: that X might disown the message.The arbiter plays a sensitive and crucial role in this sort of scheme, and all parties must have a great deal of trust that the arbitration mechanism is working properly

13. What is block size of MD5 and message digest size of SHA-1?

block size of MD5 = 128 bit

message digest size of SHA-1 = 160 bits.

14. what is message digest ?

A message digest is a cryptographic hash function cntainig a string of digits created by a one way hashing formula.

Message digest hash numbers represent specific files containing the protected works.

15. Mention the importance of compression function MD5 algorithm.

- Compression function mixes two fixed length inputs and produces a single fixed length output of the same size as one of the inputs.

16. Mention any two message digest algorithms and its features.

1. MD5 algorithm

It produces 128 bit message digests.

Maximum message size = infinity

Its quite fast.

2. SHA algorithm

Basic unit of processing = 512 bits.

Maximum message size= 2^{64}-1 bits.

Digest length = 160 its.

17. Mention the importance of compression function SHA algorithm.

It takes a fixed length input and returns a shorter and fixed length output.

18. Define Digital signature.

The digital signature is an authentication mechanism that enables the creator of message to attach a code that acts as signature.

It can be formed by taking the hash value of the message and encrypting the message with creator's private key.

19. How encryption differs from digital signature.

Digital signature is an authentication mechanism that enables the creator of message to attach a code that acts as signature.

Encryption is in which a message is encrypted with a recipient's public key. It provides confidentiality.

- **20. Specify the parameters of Elgamal Digital signature algorithm.**
- H-collision resistance hash function.
- p-a large prime such that computing discrete logarithms modulo p is difficult
- g randomly chosen generator of the multiplicative group of integers modulo p(g<p)

21. Specify the parameters of Schnorr Digital signature algorithm

Two prime numbers p and q such that q is a prime factor ofp-1.

An integer a,such that $a^{q} = 1 \bmod p$

a,p,q comprises the global public key.

A random integer s with 0<s<q represents the user's private key.

User's public key ,$v=a^{-s} \bmod p$.

22.specify the global components of DSS algorithm.

Global Public-Key Components: p, q and g are the parameters that are public and can be common to a group of users.

q: Prime divisor of p-1 where, $2^{159} < q < 2^{160}$ i.e. 160-bit number.

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 between 512 and 1024 bits in increment of 64 bits.

g: Chose to be of the form $h^{(p-1)/q} \bmod p$ where h is any integer with $1 < h < (p-1)$ such that $h^{(p-1)/q} \bmod p > 1$

23 Define Kerberos.

Kerberos is an authentication service developed as part of project Athena at MIT. The problem that Kerberos address is, assume an open distributed environment in which users at work stations wish to access services on servers distributed throughout the network.

24. What is Kerberos? What are the uses?

Kerberos is an authentication service developed as a part of project Athena at MIT.Kerberos provide a centralized authentication server whose functions is to authenticate servers

.

25. What 4 requirements were defined by Kerberos?

- Secure
- Reliable
- Transparent
- Scalable

26. In the content of Kerberos, what is realm?

A full service Kerberos environment consisting of a Kerberos server, a no. of clients, no.of application server requires the following:

- The Kerberos server must have user ID and hashed password of all participating users in its database.
- The Kerberos server must share a secret key with each server. Such an environment is referred to as "Realm".

27. What is the purpose of X.509 standard?

X.509 defines framework for authentication services by the X.500 directory to its users.X.509 defines authentication protocols based on public key certificates

PART B

- Explain in detail about three different types of Authentication function.
- Explain about secure hash algorithm (SHA) in detail.
- Explain the process of deriving eighty 64-bit words from the 1024 bits for processing of a single block and also discuss single round function in SHA – 512 algorithm. Show the values of W16, W17, W18 and W19.
- Explain about HMAC and CMAC in detail.
- Illustrate the digital Signature algorithms in detail.
- Highlight and explain the steps in Digital Signature Algorithm (DSA).
- Alice chooses Q=101 and P=7879. Assume (q, p, g and y): Alice's Public key. Alice selects h=3 and calculates g. Alice chooses x=75 as the private key and calculates y. Now, Alice can send a message to Bob. Assume that H (M) = 22 and Alice choose secret no K=50. Verify the Signature.
- Explain Entity Authentication in detail.
- Explain Kerberos
- Explain X.509 in detail.

Supportive online Certification courses

NPTEL

Cryptography and Network Security

Introduction to Cryptology

Foundations of Cryptography

Computational number theory and cryptography

COURSERA

Cryptography

Applied Cryptography

Number theory and cryptography

Cryptography and Information theory

Asymmetric cryptography and key management

Symmetric Cryptography

UDEMY

Introduction to Cryptography

Cryptography with python

Complete Cryptography master class

Real Time Applications

- Kerberos used for
  - client/server applications,
  - multi-tier applications,
  - Web-based application authentication

- X. 509 certificates are used in most network security applications, including
  - IP security,
  - secure sockets layer (SSL),
  - secure electronic transactions (SET), and
  - S/MIME

- The Applications of Biometrics include
  - Logical Access Control
  - Physical Access Control
  - Time and Attendance
  - Law Enforcement
  - Surveillance.