UNIT-III

CryptographicHashFunctions: MessageAuthentication, SecureHashAlgorithm(SHA-512), **Message authentication codes:** Authentication requirements, HMAC, CMAC, Digital signatures, Elgamal Digital Signature Scheme.
Key Management and Distribution: Symmetric Key Distribution Using Symmetric & Asymmetric Encryption, Distribution Public Keys, Kerberos, X.509 Authentication Service, Public–Key Infrastructure.

CryptographicHashFunctions:

MessageAuthentication:

1. Message authentication is a mechanism or service used to verify the integrity of a message.
2. Message authentication assures that data received is exactly same as sent by the transmitter (i.e contains no modification, insertion, detection or replay)
3. The following attacks can be identifies :

   a) **Disclosure** :Release of message contents to any person or process not processing the appropriate cryptographic key.

   b) **Traffic Analysis**: Discovery of pattern of traffic between parties.

   c) **Masquerede:** Insertion of messages into the network from a fraudulent source. This includes the creation of messages by an opponent that are purported to come from an authorized entity.

   d) **Content Modification** : Changes made to the content of a message including insertion , detection etc.

   e) **Sequence Modification:** Any modification to the sequence of messages.

   f) **Timing Modification:** Delay or replay of a message.

   g) **Source Reproduction:** Denial of transmission of message by source.

   h) **Destination Reputation:** Denial of receipt of message by destination
4. **Measures to deal with:**
   - **Disclosure, Traffic Analysis :** Message confidentiality
   - **Masquerade, Content Modification, Sequence Modification, Timing Modification :** Message Authentication
   - **Source Reproduction:** Need of digital signature.
   - **Destination Reputation:** Protocol designed to counter the attack.
5. **Authentication Function:**

   A message authentication has two levels of functionality

   At lower level: Authenticator (A value to be used to authenticate the message)

   At higher level: Authentication Protocol

   (Enables receiver to verify the authenticity of a message)
6. **The types of functions that may be used to produce an authentication are:**
   - **Message encryption:** The ciphertext of the entire message serves as authentication.
   - **Message Authentication Code (MAC):** A secret key of fixed length value that serves as the authenticator.

- **Hash Function:** A function that maps the message of any length into a fixed length hash value which serves as the authenticator.
- **Message Encryption:**

  a) **Symmetric Encryption:** Confidentiality and authentication

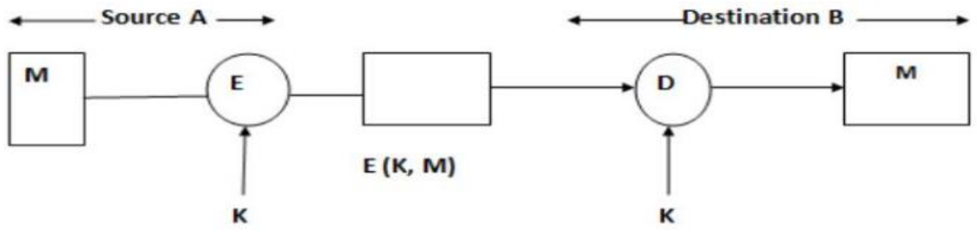


Figure 5.12.a Symmetric Encryption

- Key k is shared by A and B.
- No other party knows the key
- B is assured that message has come from A -> A is the only other party that is having k.

b) **Public key encryption:** Confidentiality

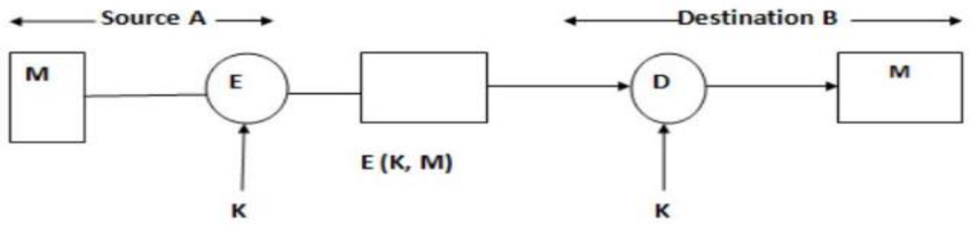


Figure 5.12.bPublic Key encryption

- No authentication because any opponent could also use B's public key to encrypt a message claiming to be from A.

c) **Public Key encryption (Authentication and Signature)**

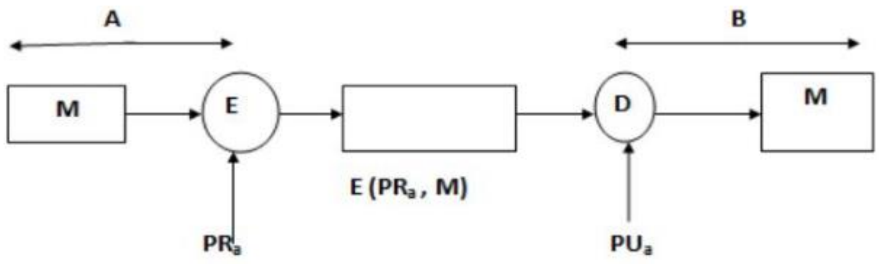


Figure 5.12.c Public Key encryption

- Does not provide confidentiality because anyone in possession of A's public key can decrypt the ciphertext.

d) **Public key encryption :** Confidentiality , authentication and signature

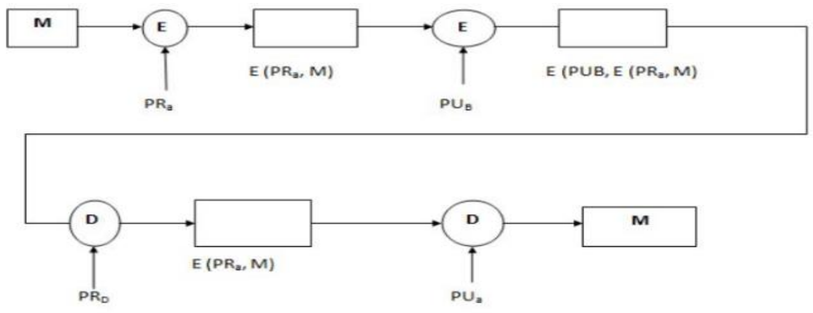
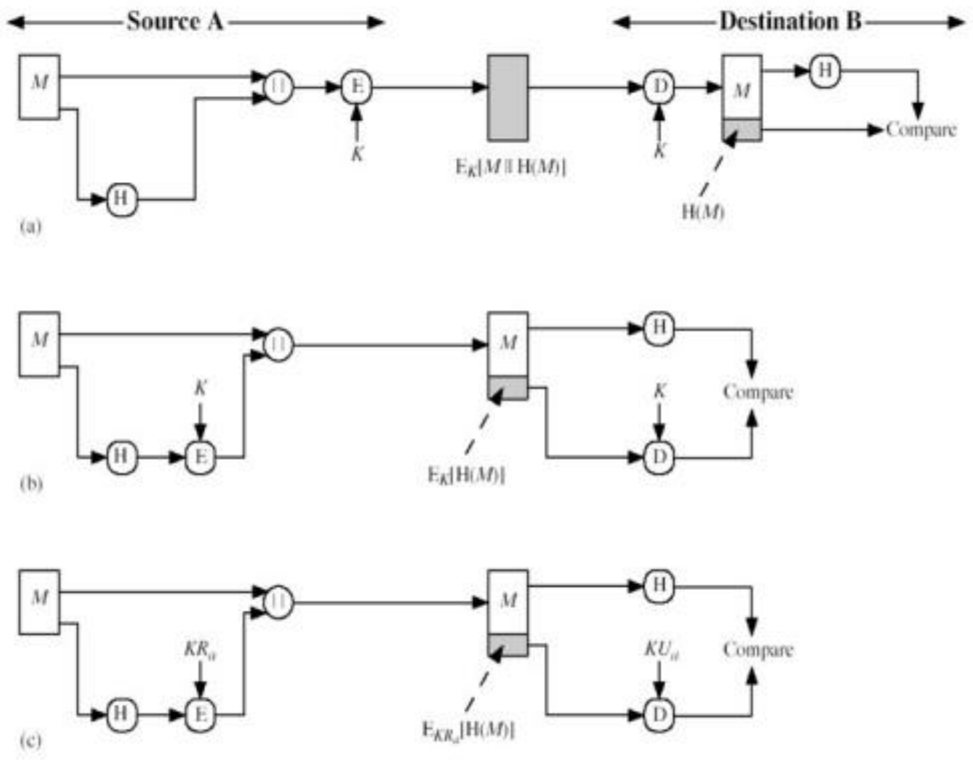


Figure 5.12.d Public key encryption

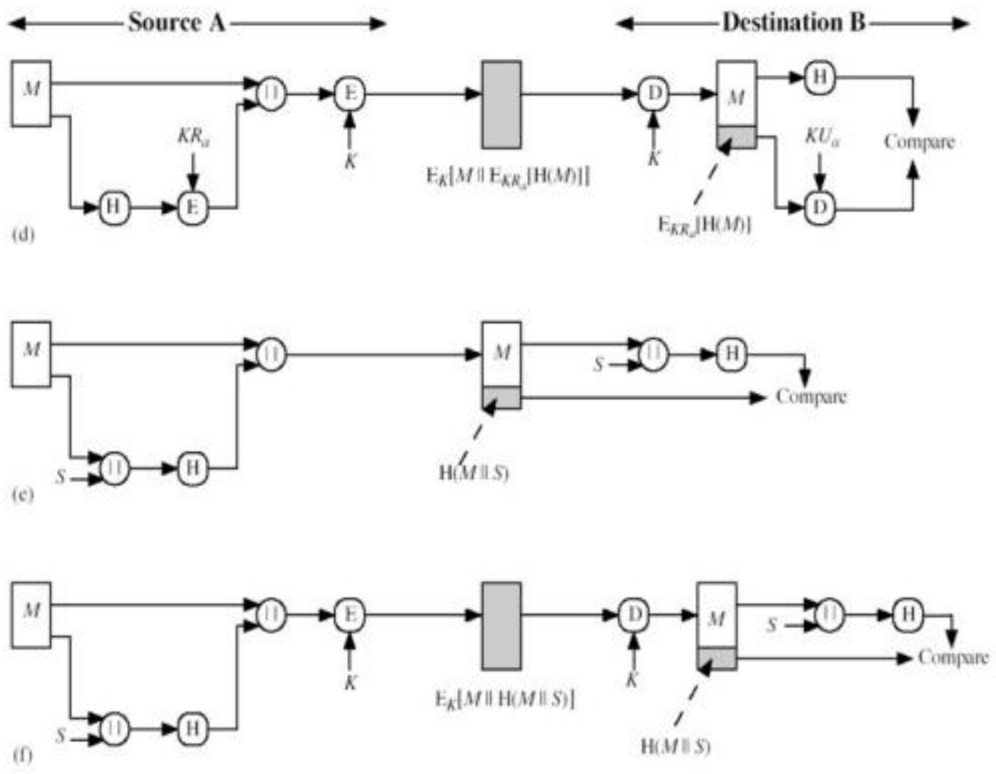
Another type of threat that exist for data is the lack of **message authentication**. In this threat, the user is not sure about the originator of the message. Message authentication can be provided using the cryptographic techniques that use secret keys as done in case of encryption.

Hash Functions :

A hash value is generated by a function H of the form: h = H(M) where M is a variable-length message, and H(M) is the fixed length hash value (also referred to as a message digest or hash code)



The purpose of a hash function is to produce a "fingerprint" of a file, message, or other block of data. To be useful for message authentication, a hash function H must have the following properties: 1. H can be applied to a block of data of any size. 2. H produces a fixed-length output. 3. H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical. 4. For any given code h, it is computationally infeasible to find x such that H(x) = h. 5. For any given block x, it is computationally infeasible to find y 6= x with H(y) = H(x) (sometimes referred to as weak collision property). 6. It is computationally infeasible to find any pair (x, y) such that H(x) = H(y) (sometimes referred to as strong collision property).

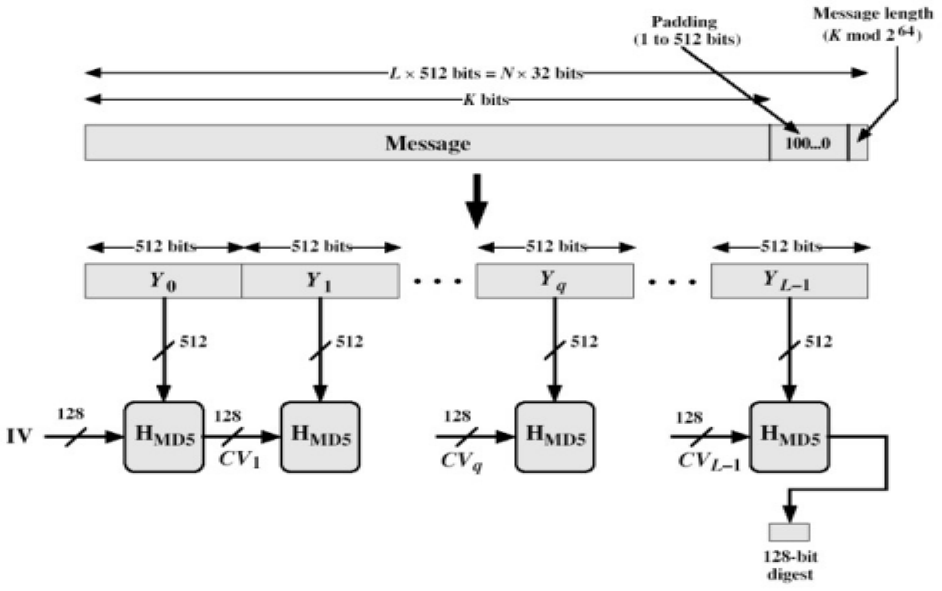


SHA-512;

<u>SecureHashAlgorithm(SHA-512)</u>

- The Secure Hash Algorithm (SHA) was invented by the National Security Agency (NSA) and published in 1993 through the National Institute of Standard and Technology (NIST) as a U.S. Federal Information Processing Standard (FIPS PUB 180).
- SHA depends on and shares the similar building blocks as the MD4 algorithm. The design of SHA introduced a new process which develop the 16-word message block input to the compression function to an 80-word block between other things.
- The processing of SHA works as follows −
- **Step 1 − Append padding bits** − The original message is padded and its duration is congruent to 448 modulo 512. Padding is continually inserted although the message already has the desired length. Padding includes a single 1 followed by the essential number of 0 bits.
- **Step 2 − Append length** − A 64-bit block considered as an unsigned 64-bit integer (most essential byte first), and defining the length of the original message (before padding in step 1), is added to the message. The complete message's length is a multiple of 512.
- **Step 3 −Initialize the buffer** − The buffer includes five (5) registers of 32 bits each indicated by A, B, C, D, and E. This 160-bit buffer can be used to influence temporary and final outcomes of the compression function. These five registers are initialized to the following 32-bit integers (in hexadecimal notation).

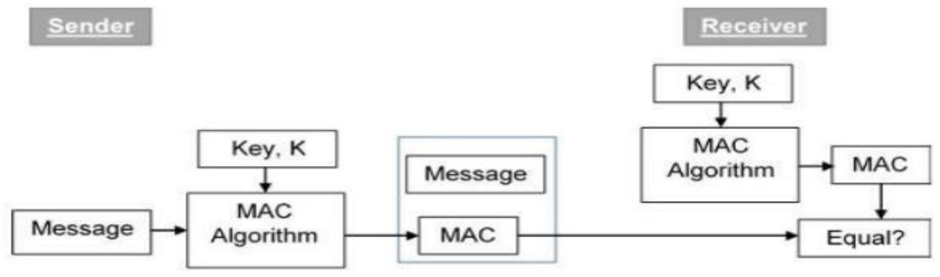
- A = 67 45 23 01
- B = ef cd ab 89
- C = 98 ba dc fe
- D = 10 32 54 76
- E = c3 d2 e1 f0
- The registers A, B, C, and D are actually the same as the four registers used in MD5 algorithm. But in SHA-1, these values are saved in big-endian format, which define that the most essential byte of the word is located in the low-address byte position. Therefore the initialization values (in hexadecimal notation) occurs as follows −
- word A = 67 45 23 01
- word B = ef cd ab 89
- word C = 98 ba dc fe
- word D = 10 32 54 76
- word E = c3 d2 e1 f0
- **Step 4 − Process message in 512-bit blocks** − The compression function is divided into 20 sequential steps includes four rounds of processing where each round is made up of 20 steps.
- The four rounds are structurally same as one another with the only difference that each round need a different Boolean function, which it can define as f1, f2, f3, f4 and one of four multiple additive constants Kt (0 ≤t ≤79) which is based on the step under consideration.
- **Step 5 − Output** − After processing the final 512-bit message block t (considering that the message is divided into t 512-bit blocks), and it can obtain a 160-bit message digest.



<u>Message Authentication Code (MAC)</u>
MAC algorithm is a symmetric key cryptographic technique to provide message authentication. For establishing MAC process, the sender and receiver share a symmetric key K.

Essentially, a MAC is an encrypted checksum generated on the underlying message that is sent along with a message to ensure message authentication.

The process of using MAC for authentication is depicted in the following illustration



Steps to understand the entire process in detail −

- The sender uses some publicly known MAC algorithm, inputs the message and the secret key K and produces a MAC value.
- Similar to hash, MAC function also compresses an arbitrary long input into a fixed length output. The major difference between hash and MAC is that MAC uses secret key during the compression.
- The sender forwards the message along with the MAC. Here, we assume that the message is sent in the clear, as we are concerned of providing message origin authentication, not confidentiality. If confidentiality is required then the message needs encryption.
- On receipt of the message and the MAC, the receiver feeds the received message and the shared secret key K into the MAC algorithm and re-computes the MAC value.
- The receiver now checks equality of freshly computed MAC with the MAC received from the sender. If they match, then the receiver accepts the message and assures himself that the message has been sent by the intended sender.
- If the computed MAC does not match the MAC sent by the sender, the receiver cannot determine whether it is the message that has been altered or it is the origin that has been falsified. As a bottom-line, a receiver safely assumes that the message is not the genuine.

<u>Limitations of MAC</u>
There are two major limitations of MAC, both due to its symmetric nature of operation −

- **Establishment of Shared Secret.**
  - It can provide message authentication among pre-decided legitimate users who have shared key.
  - This requires establishment of shared secret prior to use of MAC.
- **Inability to Provide Non-Repudiation**
  - Non-repudiation is the assurance that a message originator cannot deny any previously sent messages and commitments or actions.
- MAC technique does not provide a non-repudiation service. If the sender and receiver get involved in a dispute over message origination, MACs cannot provide a proof that a message was indeed sent by the sender.

- Though no third party can compute the MAC, still sender could deny having sent the message and claim that the receiver forged it, as it is impossible to determine which of the two parties computed the MAC.

<u>Authentication Requirements</u>

In the context of communications across a network, the following attacks can be identified:

1. Disclosure: Release of message contents to any person or process not possessing the appropriate cryptographic key.

2. Traffic analysis: Discovery of the pattern of traffic between parties. In a connectionoriented application, the frequency and duration of connections could be determined. In either a connection-oriented or connectionless environment, the number and length of messages between parties could be determined.

3. Masquerade: Insertion of messages into the network from a fraudulent source.This includes the creation of messages by an opponent that are purported to come from an authorized entity. Also included are fraudulent acknowledgments of message receipt or nonreceipt by someone other than the message recipient.

4. Content Modification: Changes to the contents of a message, including insertion, deletion, transposition, or modification.

5. Sequence modification: Any modification to a sequence of messages between parties, including insertion, deletion, and reordering.

6. Timing modification: Delay or replay of messages. In a connection-orientated application, an entire session or sequence of messages could be a replay of some previous valid session, or individual messages in the sequence could be delayed or replayed.

7. Repudiation: Denial of receipt of message by destination or denial of transmission of message by source.

<u>HMAC:</u>

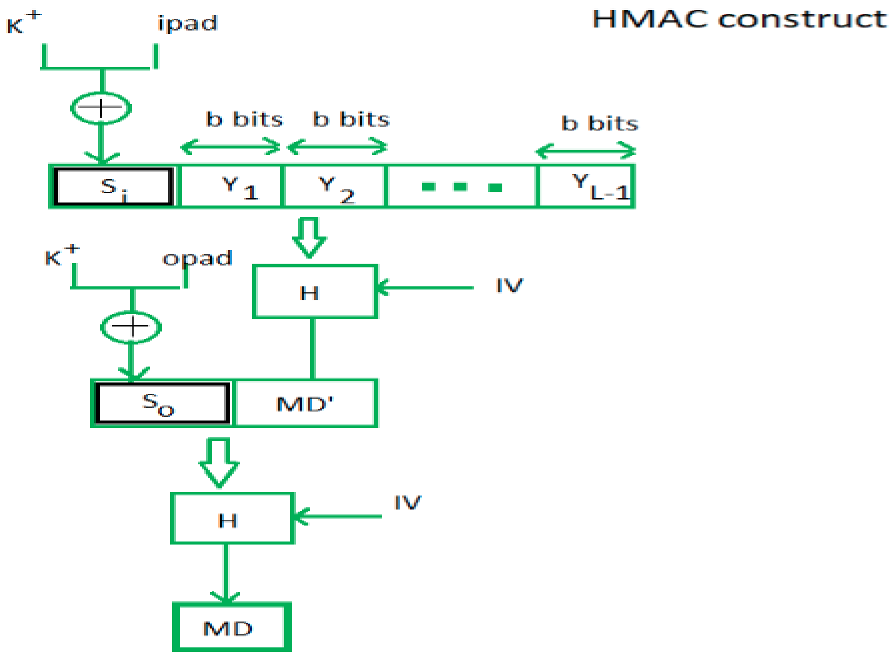
HMAC algorithm stands for Hashed or Hash-based <u>Message Authentication Code</u>. It is a result of work done on developing a MAC derived from cryptographic hash functions. HMAC is a great resistance towards cryptanalysis attacks as it uses the Hashing concept twice. HMAC consists of twin benefits of Hashing and MAC and thus is more secure than any other authentication code.

Objectives –

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

HMAC algorithm –

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



Here, H stands for Hashing function,
M is the original message
Si and So are input and output signatures respectively,
Yi is the ith block in original message M, where I ranges from [1, L)
L = the count of blocks in M
K is the secret key used for hashing
IV is an initial vector (some constant)
The generation of input signature and output signature Si and So respectively.

$S_i = K^+ \oplus ipad$ where K^+ is nothing but K padded with zeros on the left so that the result is b bits in length

$S_o = K^+ \oplus opad$ where ipad and opad are 00110110 and 01011100 respectively taken b/8 times repeatedly.

$MD' = H(S_i \,||\, M)$

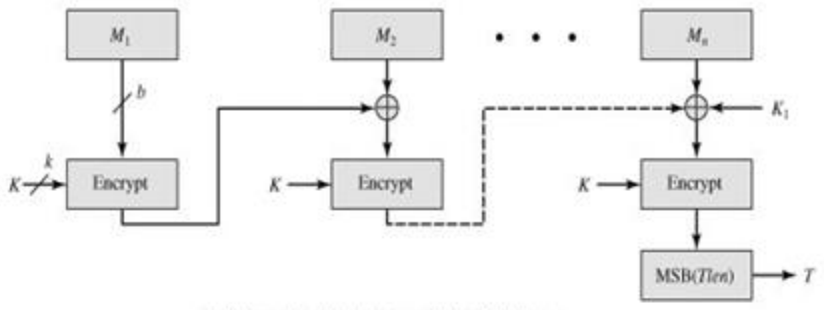
$MD = H(S_o \,||\, MD')$ or $MD = H(S_o \,||\, H(S_i \,||\, M))$

CMAC

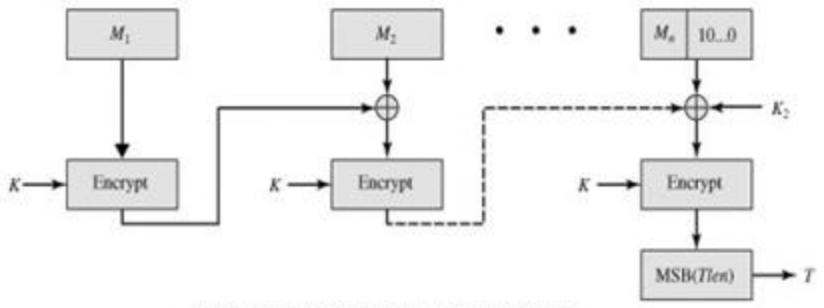
The Data Authentication Algorithm defined in FIPS PUB 113, also known as the CBC-MAC (cipher block chaining message authentication code). This cipher-based MAC has been widely adopted in government and industry.MAC is secure under a reasonable set of security criteria, with the following restriction. Only messages of one fixed length of mn bits are processed, where n is the cipher block size and m is a fixed positive integer. Black and Rogaway demonstrated that this limitation could be overcome using three keys: one key of length k to be used at each step of the cipher block chaining and two keys of length n, where k is the key length and n is the cipher block length

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, M_1, M_2... Mn. The algorithm makes use of a k-bit encryption key K and an n-bit constant K_1. 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:

Cipher-Based Message Authentication Code (CMAC)



(a) Message length is integer multiple of block size



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

C1= E (K, M1)

C2 = E (K, [M2 $\oplus$C1])

C3 = E (K, [M3 $\oplus$C2])

.

.

.

Cn= E (K, [Mn $\oplus$Cn1$\oplus$K1])

T = MSB_{Tlen}(Cn)

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

Tlen = bit length of T

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 precedes 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 = E (K, 0n)

K1 = L · x

K2 = L · x2 = (L · x) · x

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.

<u>Digital Signatures:</u>

The digital signature is analogous to the handwritten signature. It must have the following properties:

• It must verify the author and the date and time of the signature.

• It must authenticate the contents at the time of the signature.

• It must be verifiable by third parties, to resolve disputes. Thus, the digital signature function includes the authentication function.

On the basis of these properties, we can formulate the following requirements for a digital signature:

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

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

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

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

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

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

ELGAMAL DIGITAL SIGNATURE SCHEME

The ElGamal signature scheme involves the use of the private key for encryption and the public key for decryption .

Before proceeding, we need a result from number theory,that for a prime number q, if a is a primit ive root of q, then

$\alpha, \alpha^2, \ldots, \alpha^{q-1}$ are distinct (mod q). It can be shown that, if a is a primitive root of q, then

1. For any integer m, $\alpha^m \equiv 1 \pmod{q}$ if and only if $m \equiv 0 \pmod{q-1}$.
2. For any integers, i, j, $\alpha^i \equiv \alpha^j \pmod{q}$ if and only if $i \equiv j \pmod{q-1}$.

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

1. Generate a random integer XA, such that 1 6 XA<q - 1.
2. Compute YA = aXA mod q.
3. A's private key is XA; A's pubic key is {q, a, YA}.
To sign a message M, user A first computes the hash m = H(M), such that m is an integer in the range 0 <= m <= q - 1. A then forms a digital signature as follows.

1.Choose a random integer K such that 1 <= K <= q - 1 and gcd(K, q - 1) = 1. That is, K is relatively prime to q - 1.
2.Compute S1 = aKmod q. Note that this is the same as the computation of C1 for ElGamal encryption.

3. ComputeK-1mod(q-1).Thatis,computetheinverseofKmoduloq-1.
4. Compute S2 = K-1(m - XAS1) mod (q - 1).
5.The signature consists of the pair (S1, S2). Any user B can verify the signature as follows.
1. Compute V1 = am mod q.
Compute V2 = (YA) 1(S1) mod q. The signature is valid if V1 = V2. Let us demonstrate that this is so. Assume that the equality is true. Then we have

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

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}, as shown in Table 8.3. We choose a = 10.

Alice generates a key pair as follows:

1.Alice chooses XA = 16.
2. Then YA = aXA mod q = a16 mod 19 = 4.
3. Alice’s private key is 16; Alice’s pubic key is {q, a, YA} = {19, 10, 4}. Suppose Alice wants to sign a message with hash value m = 14.
1.Alice chooses K = 5, which is relatively prime to q - 1 = 18.
2. S1 = aKmod q = 105mod 19 = 3

3. $K^{-1} \bmod (q - 1) = 5^{-1} \bmod 18 = 11$.
4. $S_2 = K^{-1}(m - X_A S_1) \bmod (q - 1) = 11(14 - (16)(3)) \bmod 18 = -374 \bmod 18 = 4$.

Bob can verify the signature as follows.

1. $V_1 = \alpha^m \bmod q = 10^{14} \bmod 19 = 16$.
2. $V_2 = (Y_A)^{S_1}(S_1)^{S_2} \bmod q = (4^3)(3^4) \bmod 19 = 5184 \bmod 19 = 16$.

Thus, the signature is valid.

Key Management and Distribution:

SYMMETRIC KEY DISTRIBUTION USING SYMMETRIC ENCRYPTION

For symmetric encryption to work, the two parties to an exchange must share the same key, and that key must be protected from access by others. Furthermore, fre- quent key changes are usually desirable to limit the amount of data compromised if an attacker learns the key.

Therefore, the strength of any cryptographic system rests with the key distribution technique, a term that refers to t he means of deliver- ing a key to two parties who wish to exchange data without allowing others to see the key. For two parties A and B, key distribution can be achieved in a number of ways, as follows:

1. A can select a key and physically deliver it to B.
2. A third party can select the key and physically deliver it to A and B.
3. If A and B have previously and recently used a key, one party can transmit
the new key to the other, encrypted using the old key.
4.If A and B each has an encrypted connection to a third party C, C can deliver
a key on the encrypted links to A and B.
Options 1 and 2 call for manual delivery of a key. For link encryption, this is a
reasonable requirement, because each link encryption device is going to be exchanging data onl y with its partner on the other end of the link.

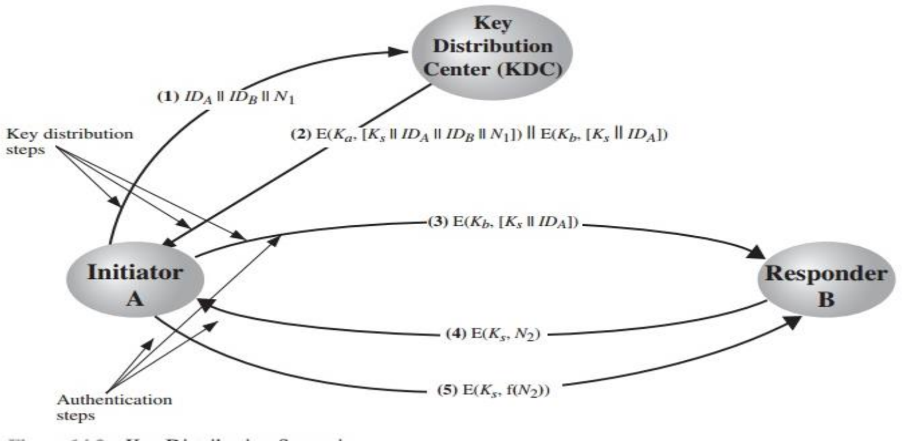
However, for **end-to-end encryption** over a network , manual delivery is awkward
. In a distributed system, any
given host or terminal may need to engage in exchanges with many other hosts and terminals over time.

Thus, each device needs a number of keys supplied dynamically. The problem is especially diffic ult in a wide-area distributed system.

The scale of the problem depends on the number of communicating pairs thatmust be supporte d. If end-to-end encryption is done at a network or IP level, then a
key is needed for each of hosts on the network that wish to communicate. Thus, if there are N hosts, the number of required keys is [N(N - 1)]/2 .

A Key Distribution Scenario

The key distribution concept can be deployed in a number of ways. A typical sce- nario is illustrated in Figure 14.3, which is based on a figure in [POPE79]. The scenario assumes that each user shares a unique master key with the key distribution center (KDC).


Key Distribution Center (KDC)
(1) $ID_A \| ID_B \| N_1$
(2) $E(K_a, [K_s \| ID_A \| ID_B \| N_1]) \| E(K_b, [K_s \| ID_A])$
Key distribution steps
(3) $E(K_b, [K_s \| ID_A])$
Initiator A
Responder B
(4) $E(K_s, N_2)$
(5) $E(K_s, f(N_2))$
Authentication steps

Let us assume that user A wishes to establish a logical connection with B and requires a one-time session key to protect the data transmitted over the connection. A has a master key, Ka, known only to itself and the KDC;

similarly, B shares the master key Kb with the KDC. The following steps occur.

1.A issues a request to the KDC for a session key to protect a logical connectionto B. The messa ge includes the identity of A and B and a unique identifier, N1, for this transaction, which we refer to as a **nonce**.The nonce may be a

- timestamp, a counter, or a random number; the minimum requirement is that it d iffers with each request.

Also, to prevent masquerade, it should be difficult foran opponent to guess the nonce. Thus, a r andom number is a good choice for a nonce.

2.The KDC responds with a message encrypted using Ka. Thus, A is the only one who can succe ssfully read the message, and A knows that it originated at the KDC. The message includes two items intended for A:

• The one-time session key, Ks, to be used for the session
• The original request message, including the nonce, to enable A to match this response with the appropriate request.
Thus, A can verify that its original request was not altered before reception by
the KDC and, because of the nonce, that this is not a replay of some previous request.

In addition, the message includes two items intended for B:

• The one-time session key, Ks, to be used for the session
• An identifier of A (e.g., its network address), IDA
These last two items are encrypted with Kb (the master key that the KDC shares with B).They are to be sent to B to establish the connection and prove A’s identity.

3.A stores the session key for use in the upcoming session and forwards to B the information that originated at the KDC
for B, namely, E(Kb,[Ks || IDA]). Because this information is encrypted with Kb, it is protecte d from eavesdropping. B now knows the session key (Ks), knows that the other party is A (from IDA), and knows that the information originated at the KDC (because it is encrypted using Kb).
At this point, a session key has been securely delivered to A and B, and they may begin their protected exchange. However, two additional steps are desirable:

4.Using the newly minted session key for encryption, B sends a nonce, N2, to A.
5.Also, using Ks, A responds with f(N2), where f is a function that performs some transformation on N2 (e.g., adding one).
These steps assure B that the original message it received (step 3) was not a replay.

SYMMETRIC KEY DISTRIBUTION USING ASYMMETRIC ENCRYPTION

If A wishes to communicate with B, the following procedure is employed:

1.A generates a public/private key pair {PUa, PRa} and transmits a message to B consisting of PUa and an identifier of A, IDA.
2.B generates a secret key, Ks, and transmits it to A, which is encrypted with A's public key.
3.A computes D(PRa, E(PUa, Ks)) to recover the secret key. Because only A can decrypt the message, only A and B will know the identity of Ks.
4.A discards PUa and PRa and B discards PUa.
A and B can now securely communicate using conventional encryption and the session key Ks. At the completion of the exchange, both A and B discard Ks.

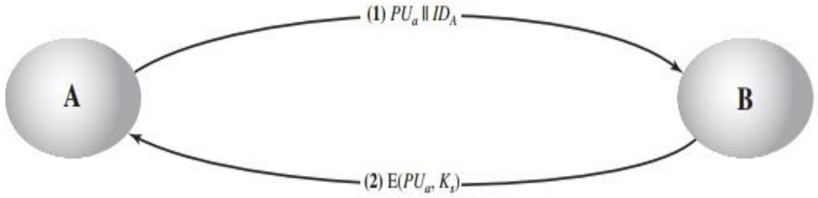
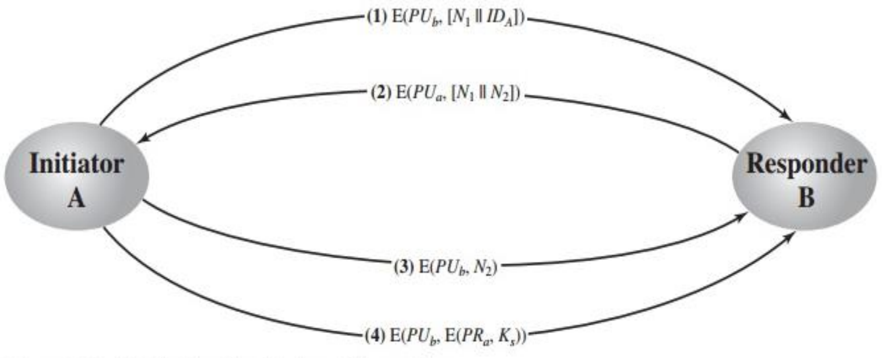


Figure 14.7 Simple Use of Public-Key Encryption to Establish a Session Key

Secret Key Distribution with Confidentiality and Authentication

provides protection against both active and passive attacks. We begin at a point when it is assu med that A and B have exchanged public keys by one of the schemes described subsequently in this chapter. Then the following steps occur.


(1) E(PU_b, [$N_1 \| ID_A$])
(2) E(PU_a, [$N_1 \| N_2$])
Initiator A
Responder B
(3) E(PU_b, N_2)
(4) E(PU_b, E(PR_a, K_s))
Figure 14.8 Public-Key Distribution of Secret Keys

A uses B’s public key to encrypt a message to B containing an identifier of A(IDA) and a nonce (N1), which is used to identify this transaction uniquely.
1.B sends a message to A encrypted with PUa and containing A’s nonce (N1) as ell as a new nonce generated by B (N2). Because only B could have (N2). Because only B could have decrypted message (1), the presence of N1 in message (2) assures A that the correspondent is B.
2. A returns N2, encrypted using B’s public key, to assure B that its correspondent is A.
A selects a secret key Ks and sends M = E(PUb, E(PRa, Ks)) to B. Encryption of this message with B’s public key ensures that only B can read it; encryption

with A’s private key ensures that only A could have sent it.

3. B computes D(PUa, D(PRb, M)) to recover the secret key.
4.The result is that this scheme ensures both confidentiality and authentication in the exchange of a secret key.

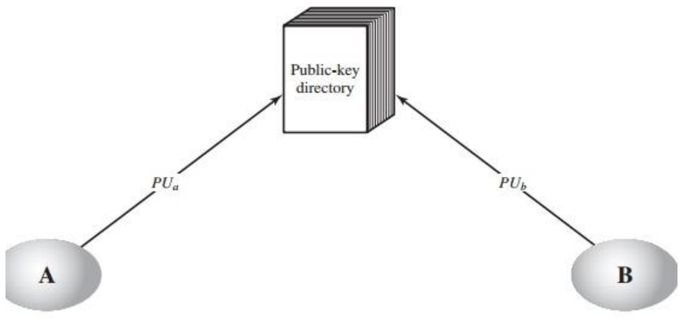
DISTRIBUTION OF PUBLIC KEYS
Several techniques have been proposed for the distribution of public keys. Virtually all these proposals can be grouped into the following general schemes:

- Public announcement
- Publicly available directory
- Public-key authority
- Public-key certificates

A
PU_a
PU_a
PU_a
PU_a
B
PU_b
PU_b
PU_b
PU_b

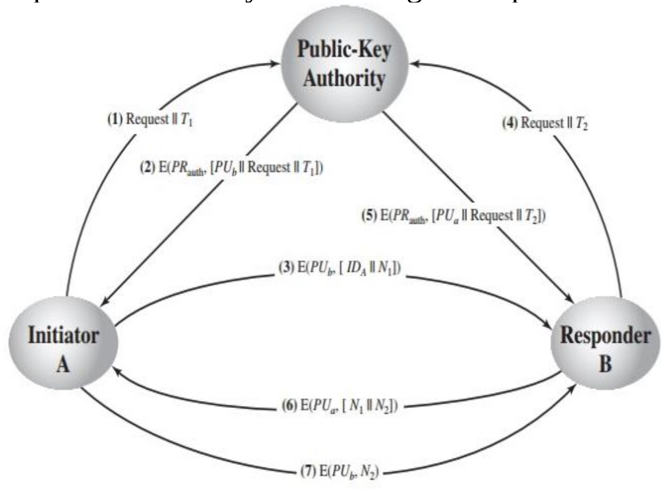
'igure 14.9 Uncontrolled Public-Key Distribution

Public Announcement of Public Keys
The point of publickey encryption is that the public key is public. Thus, if there is some broadl y accepted publickey algorithm, such as RSA, any participant can send his or her public key to any other participant or broadcast the key to the community at large. Although this approach is convenient, it has a major weakness. Anyone can forge such a public announcement. That is, some user could pretend to be user A and send a public key to another participant or broadcast such a public key.
Publicly Available Directory
A greater degree of security can be achieved by maintaining a publicly available dynamic directory of public keys.Such a scheme would include the following elements:
1. The authority maintains a directory with a {name, public key} entry for each participant.
2. Each participant registers a public key with the directory authority. Registration would have to be in person or by some form of secure authenticated communication.
3. A participant may replace the existing key with a new one at any time, either because of the desire to replace a public key that has already been used for a large amount of data, or because the corresponding private key has been com- promised in some way.



Participants could also access the directory electronically. For this purpose, secure, authenticated communication from the authority to the participant is mandatory.
Public-Key Authority
The following steps occur.

1.A sends a timestamped message to the public-key authority containing a
request for the current public key of B.
2.The authority responds with a message that is encrypted using the authority's private key, PRaut h.Thus,A is able to decrypt the message using the authority's public key.Therefore,A is assured tha t the message originated with the authority.The message includes the following:
• B's public key, PUb, which A can use to encrypt messages destined for B
• The original request used to enable A to match this response with the cor- responding earlier request and to verify that the original request was not altered before reception by the authority.


Public-Key Authority
(1) Request ‖ T_1
(2) $E(PR_{auth}, [PU_b \| Request \| T_1])$
(4) Request ‖ T_2
(5) $E(PR_{auth}, [PU_a \| Request \| T_2])$
(3) $E(PU_b, [ID_A \| N_1])$
Initiator A
Responder B
(6) $E(PU_a, [N_1 \| N_2])$
(7) $E(PU_b, N_2)$

The original timestamp given so A can determine that
this is not an old message from the authority containing a key other than B's current public key
3.A stores B's public key and also uses it to encrypt a message to B containing an
identifier of A (IDA) and a nonce (N1), which is used to identify this transaction uniquely.
4. B retrieves A's public key from the authority in the same manner as A retrieved B's public key.

At this point, public keys have been securely delivered to A and B, and they may begin their protected exchange. However, two additional steps are desirable:

5. B sends a message to A encrypted with PUa and containing A's nonce (N_1) as well as a new nonce generated by B (N_2). Because only B could have decrypted message (3), the presence of N_1 in message (6) assures A that the correspondent is B.

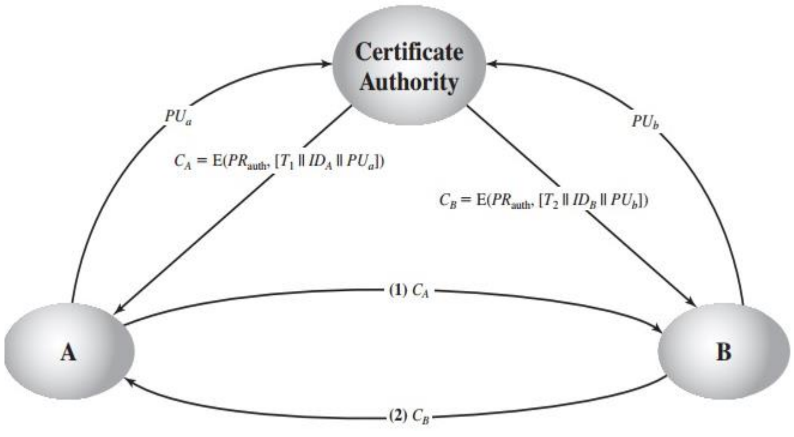
6. A returns N2, which is encrypted using B's public key, to assure B that its cor- respondent is A.

<u>Public-Key Certificates</u>

A user can present his or her public key to the authority in a secure manner and obtain a cer- tificate. The user can then publish the certificate. Anyone needing this user's pub- lic key can obtain the certificate and verify that it is valid by way of the attached trusted signature. requirements on this scheme:

1.Any participant can read a certificate to determine the name and public key of the certificate's owner.
2. Any participant can verify that the certificate originated from the certificate authority and is not counterfeit.
3. Only the certificate authority can create and update certificates.
These requirements are satisfied by the original proposal the following additional requirement.

4.Any participant can verify the currency of the certificate.



Certificate Authority
PU_a
PU_b
$C_A = E(PR_{auth}, [T_1 \| ID_A \| PU_a])$
$C_B = E(PR_{auth}, [T_2 \| ID_B \| PU_b])$
(1) C_A
A
B
(2) C_B

Kerberos:

Kerberos provides a centralized authentication server whose function is to authenticate users to servers and servers to users. In Kerberos Authentication server and database is used for client authentication. Kerberos runs as a third-party trusted server known as the Key Distribution Center (KDC). Each user and service on the network is a principal.
The main components of Kerberos are:

Authentication Server (AS):

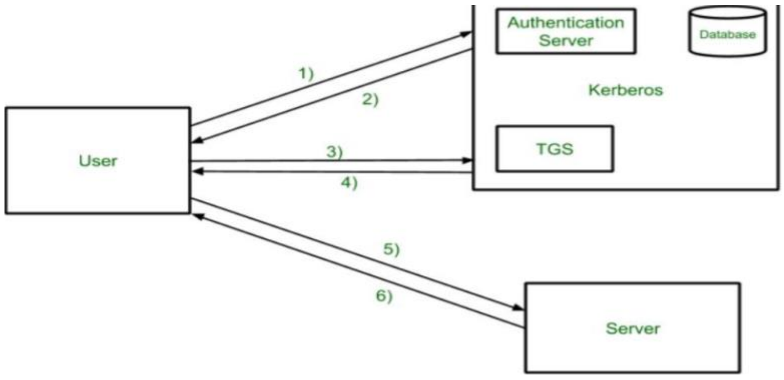
The Authentication Server performs the initial authentication and ticket for Ticket Granting Service.

Database:

The Authentication Server verifies the access rights of users in the database.

Ticket Granting Server (TGS):

The Ticket Granting Server issues the ticket for the Server



- **Step-1:**
  User login and request services on the host. Thus user requests for ticket-granting service.
  Step-2:
  Authentication Server verifies user’s access right using database and then gives ticket-granting-ticket and session key. Results are encrypted using the Password of the user.
  Step-3:
  The decryption of the message is done using the password then send the ticket to Ticket Granting Server. The Ticket contains authenticators like user names and network addresses.
  Step-4:
  Ticket Granting Server decrypts the ticket sent by User and authenticator verifies the request then creates the ticket for requesting services from the Server.
  Step-5:
  The user sends the Ticket and Authenticator to the Server.
  Step-6:
  The server verifies the Ticket and authenticators then generate access to the service. After this User can access the services.

Kerberos Limitations

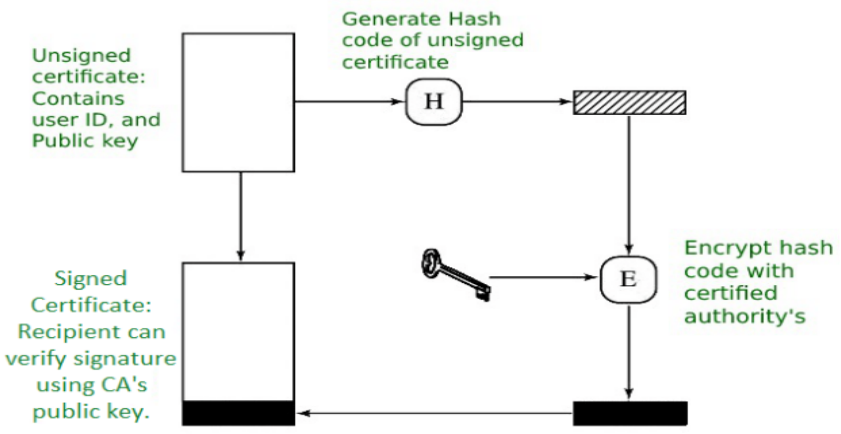
- Each network service must be modified individually for use with Kerberos
- It doesn’t work well in a timeshare environment
- Secured Kerberos Server
- Requires an always-on Kerberos server
- Stores all passwords are encrypted with a single key
- Assumes workstations are secure
- May result in cascading loss of trust.
- Scalability

X.509 Authentication Service

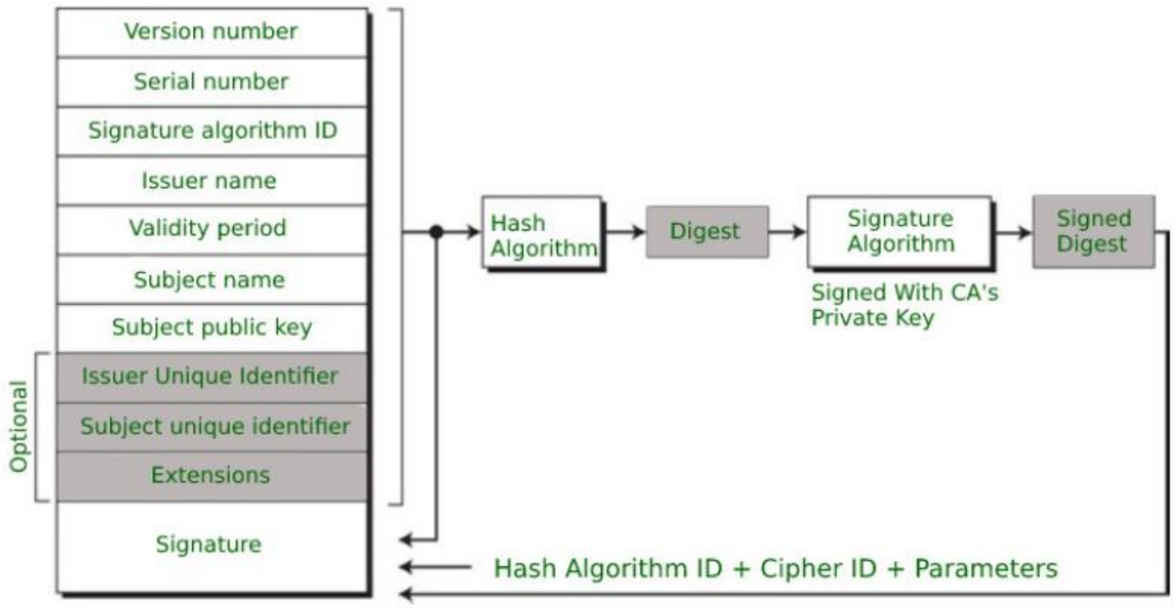
X.509 is a digital certificate that is built on top of a widely trusted standard known as ITU or International Telecommunication Union X.509 standard, in which the format of PKI certificates is defined. X.509 digital certificate is a certificate-based authentication security framework that can be used for providing secure transaction processing and private information.

Working of X.509 Authentication Service Certificate:

The core of the X.509 authentication service is the public key certificate connected to each user. These user certificates are assumed to be produced by some trusted certification authority and positioned in the directory by the user or the certified authority.


Unsigned certificate: Contains user ID, and Public key
Generate Hash code of unsigned certificate
H
Encrypt hash code with certified authority's
E
Signed Certificate: Recipient can verify signature using CA's public key.

Format of X.509 Authentication Service Certificate:



Generally, the certificate includes the elements given below:

- **Version number:** It defines the X.509 version that concerns the certificate.
- **Serial number:** It is the unique number that the certified authority issues.
- **Signature Algorithm Identifier:** This is the algorithm that is used for signing the certificate.
- **Issuer name:** Tells about the X.500 name of the certified authority which signed and created the certificate.
- **Period of Validity:** It defines the period for which the certificate is valid.
- **Subject Name:** Tells about the name of the user to whom this certificate has been issued.
- **Subject's public key information:** It defines the subject's public key along with an identifier of the algorithm for which this key is supposed to be used.
- **Extension block:** This field contains additional standard information.
- **Signature:** This field contains the hash code of all other fields which is encrypted by the certified authority private key.

Applications of X.509 Authentication Service Certificate:

Many protocols depend on X.509 and it has many applications, some of them are given below:

- Document signing and Digital signature
- Web server security with the help of Transport Layer Security (TLS)/Secure Sockets Layer (SSL) certificates
- Email certificates
- Code signing
- Secure Shell Protocol (SSH) keys
- Digital Identities

Public Key Infrastructure

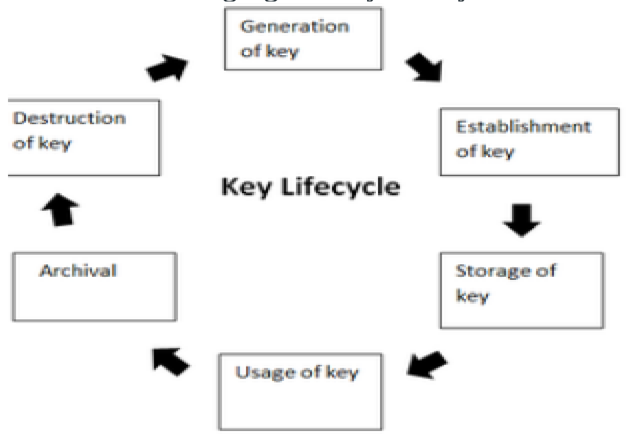
Public key infrastructure or PKI is the governing body behind issuing digital certificates. It helps to protect confidential data and gives unique identities to users and systems. Thus, it ensures security in communications.

The public key infrastructure uses a pair of keys: the public key and the private key to achieve security. The public keys are prone to attacks and thus an intact infrastructure is needed to maintain them.

Managing Keys in the Cryptosystem:

The security of a cryptosystem relies on its keys. Thus, it is important that we have a solid key management system in place. The 3 main areas of key management are as follows:

- A cryptographic key is a piece of data that must be managed by secure administration.
- It involves managing the key life cycle which is as follows:



- Public key management further requires:
  - **Keeping the private key secret:** Only the owner of a private key is authorized to use a private key. It should thus remain out of reach of any other person.
  - **Assuring the public key:** Public keys are in the open domain and can be publicly accessed. When this extent of public accessibility, it becomes hard to know if a key is correct and what it will be used for. The purpose of a public key must be explicitly defined.

PKI or public key infrastructure aims at achieving the assurance of public key.

Public Key Infrastructure:

Public key infrastructure affirms the usage of a public key. PKI identifies a public key along with its purpose. It usually consists of the following components:

- A digital certificate also called a public key certificate
- Private Key tokens
- Registration authority
- Certification authority
- CMS or Certification management system

Working on a PKI:

Let us understand the working of PKI in steps.

- **PKI and Encryption:** The root of PKI involves the use of cryptography and encryption techniques. Both symmetric and asymmetric encryption uses a public key.
- **Public Key Certificate or Digital Certificate:** Digital certificates are issued to people and electronic systems to uniquely identify them in the digital world.

Digital certificates are also called X.509 certificates. This is because they are based on the ITU standard X.509.

- The Certification Authority (CA) stores the public key of a user along with other information about the client in the digital certificate. The information is signed and a digital signature is also included in the certificate.
- The affirmation for the public key then thus be retrieved by validating the signature using the public key of the Certification Authority.

Certifying Authorities: A CA issues and verifies certificates. This authority makes sure that the information in a certificate is real and correct and it also digitally signs the certificate. A CA or **Certifying Authority performs these basic roles**:

- Generates the key pairs – This key pair generated by the CA can be either independent or in collaboration with the client.
- Issuing of the digital certificates – When the client successfully provides the right details about his identity, the CA issues a certificate to the client. Then CA further signs this certificate digitally so that no changes can be made to the information.
- Publishing of certificates – The CA publishes the certificates so that the users can find them.
- Verification of certificate – CA gives a public key that helps in verifying if the access attempt is authorized or not.
- Revocation – In case of suspicious behavior of a client or loss of trust in them, the CA has the power to revoke the digital certificate.

Classes of a Digital Certificate:

A digital certificate can be divided into four broad categories. These are :

- Class 1: These can be obtained by only providing the email address.
- Class 2: These need more personal information.
- Class 3: This first checks the identity of the person making a request.
- Class 4: They are used by organizations and governments.

Process of creation of certificate:

The creation of a certificate takes place as follows:

- Private and public keys are created.
- CA requests identifying attributes of the owner of a private key.

- Public key and attributes are encoded into a CSR or Certificate Signing Request.
- Key owner signs that CSR to prove the possession of a private key.
- CA signs the certificate after validation.

Creation of Trust layers among CA Hierarchies:

Each CA has its own certificate. Thus, trust is built hierarchically where one CA issues certificates to other CAs. Moreover, there is a root certificate that is self-signed. For a root CA, the issuer and the subject are not two separate parties but a single party.

Security of Root CA:

As you saw above, the ultimate authority is the root CA. Hence, the security of root CA is of huge importance. If the private key of a root CA is not taken care of, then it might turn into a catastrophe.