UNIT III

Key Management and Distribution:

Public Announcement of Public Keys:

The idea of public-key encryption is that the public key is made available to everyone, allowing anyone to send encrypted messages to the key owner.

- However, this approach has a weakness, anyone can forge a public announcement and pretend to be the key owner.
- This can lead to a situation where the forger can read all encrypted messages until the key owner discovers the forgery and alerts other participants.
- Therefore, it is important to be cautious when sharing public keys and to verify their authenticity.

Publicly Available Directory:

A dynamic directory of public keys is maintained by a trusted entity or organization.

- Each participant registers a public key with the directory authority.
- Registration requires secure authenticated communication or in-person verification.
- Participants can replace their public keys with new ones at any time.
- Secure authenticated communication is mandatory for accessing the directory electronically.
- The scheme is more secure than individual public announcements, but still vulnerable to attacks if the private key of the directory authority is compromised or if records are tampered with.

Public-Key Authority:

A scenario which involves a central authority maintaining a directory of public keys of all participants, with each participant knowing the authority's public key is considered.

- A sends a message to the authority requesting the current public key of B.
- The authority responds with a message encrypted with its private key, containing B's public key, the original request, and a timestamp.
- A stores B's public key and sends a message to B containing an identifier and a nonce.
- B retrieves A's public key from the authority.
- B sends a message to A encrypted with A's public key, containing A's nonce and a new nonce generated by B.
- A returns a message encrypted with B's public key, containing B's nonce.

These steps securely deliver public keys to A and B, and they can begin their protected exchange. Caching can be used to avoid frequent use of the initial four messages.

Public-Key Certificates:

Public key distribution can have issues with a bottleneck public-key authority and tampering with directories of names and public keys.

- Certificates are an alternative approach that include a public key, key owner identifier, and trusted third-party signature.
- Certificate authorities, like government agencies or financial institutions, are trusted by the user community to create and sign certificates.
- Users can obtain a certificate from the authority, and anyone needing their public key can obtain the certificate and verify its validity.

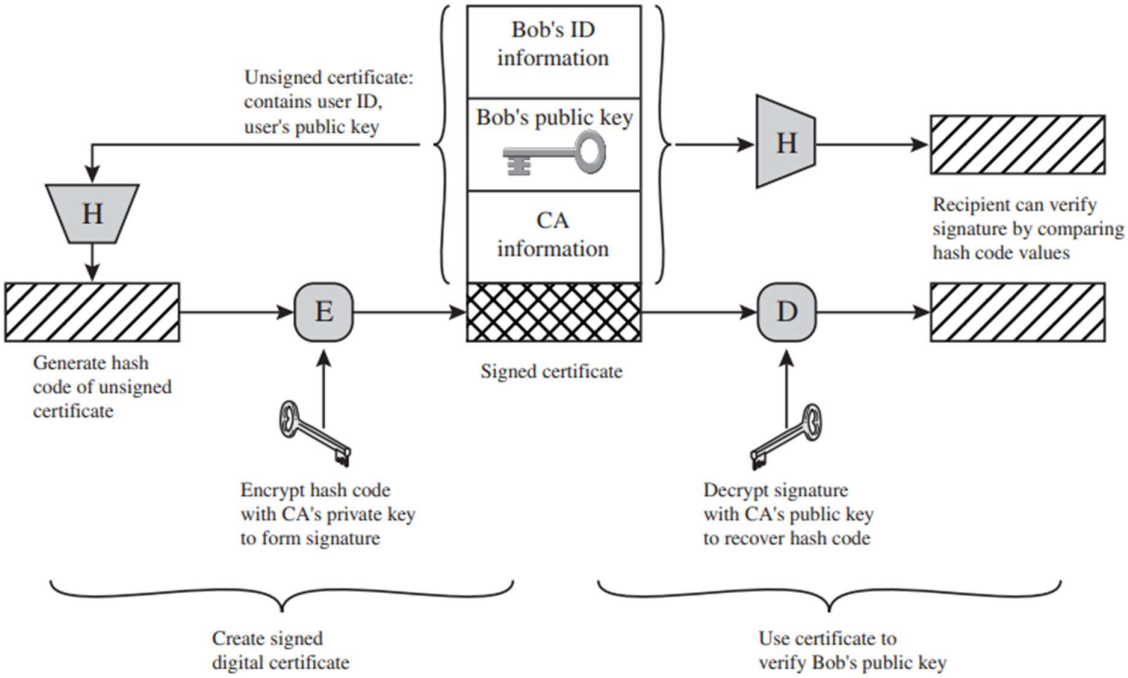
- Requirements for a certificate scheme include reading a certificate to determine owner name and public key, verifying its origin and not being counterfeit, and only the certificate authority can create and update certificates.
- The compromise of a private key is similar to the loss of a credit card, and timestamps serve as expiration dates.
- X.509 certificates are widely used in network security applications.

X. 509 Certificates:

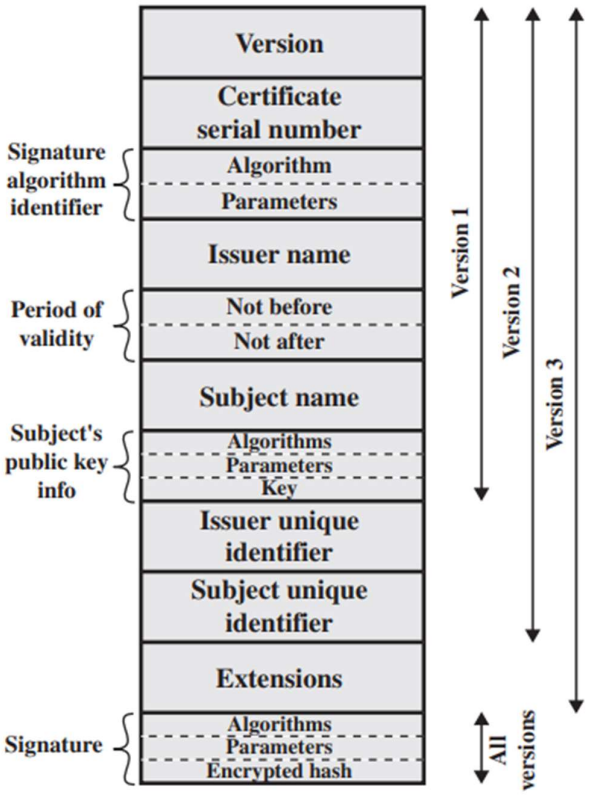
X.509 is part of the X.500 series that defines a directory service which maintains a database of information about users.

- X.509 defines a framework for authentication services and may serve as a repository of public-key certificates.
- The X.509 certificate contains the public key of a user and is signed by a trusted certification authority.
- X.509 also defines alternative authentication protocols based on the use of public-key certificates.
- X.509 certificate format is used in S/MIME, IP Security, and SSL/TLS.
- X.509 is based on the use of public-key cryptography and digital signatures, and recommends RSA for the digital signature scheme.

Certificates:



- X.509 scheme uses a public-key certificate for each user.
- The certificates are created by a trusted certification authority (CA).
- The certificates are stored in a directory server, which provides an easily accessible location for users to obtain certificates.
- The directory server is not responsible for creating public keys or certification.



(a) X.509 certificate

- **Version:** Differentiates certificate format versions.
- **Serial number:** Unique integer value associated with the certificate.
- **Signature algorithm identifier**: Algorithm used to sign the certificate and associated parameters.
- **Issuer name:** X.500 name of the CA that created and signed the certificate.
- **Validity period:** Dates the certificate is valid.
- **Subject name:** Name of the user to whom the certificate refers.
- **Subject's public-key information:** Public key and identifier of the algorithm used.
- **Issuer unique identifier:** Optional identifier used to uniquely identify the issuing CA.
- **Subject unique identifier:** Optional identifier used to uniquely identify the subject.
- **Extensions:** Set of one or more extension fields (version 3 only).
- **Signature:** Hash code of all other fields encrypted with the CA's private key.

  $Y\ll X\gg$ = the certificate of user X issued by certification authority Y
- User certificates generated by a CA have the following characteristics:
  1. Any user with access to the public key of the CA can verify the user public key that was certified.
  2. No party other than the certification authority can modify the certificate without this being detected.
- The X.509 hierarchy is a structure of digital certificates that are issued by CA
- The hierarchy consists of a root CA, intermediate CAs, and end-entity certificates.

X.509 Version 3:
Version 2 of X.509 had some limitations that needed to be addressed, such as inadequate subject field, lack of security policy information, and inability to identify different keys used by the same owner at different times.

- To address these issues, version 3 includes optional extensions that can be added to the certificate format.
- The extensions are divided into three categories:
    1. key and policy information
    2. subject and issuer attributes
    3. Certification path constraints.
- The criticality indicator in each extension indicates whether it can be safely ignored, and if it is set to TRUE and the extension is not recognized, the certificate must be treated as invalid.
- Key and Policy Information:
    1. **Authority key identifier**: identifies the public key to be used to verify the signature on this certificate or CRL and enables distinct keys of the same CA to be differentiated.
    2. **Subject key identifier**: identifies the public key being certified, useful for subject key pair updating and for cases where the subject has multiple key pairs.
    3. **Key usage:** indicates the purposes for which, and the policies under which, the certified public key may be used.
    4. **Private-key usage period**: indicates the period of use of the private key corresponding to the public key.
    5. **Certificate policies**: lists policies that the certificate is recognized as supporting, together with optional qualifier information.
    6. **Policy mappings:** used only in certificates for CAs issued by other CAs and allow an issuing CA to indicate that one or more of its policies can be considered equivalent to another policy used in the subject CA's domain.
- Certificate subject and Issuer Attributes:
    1. **Subject name alternative:** provides additional names that can be used to identify the subject of the certificate, including email addresses and DNS names.
    2. **Issuer name alternative:** similar to subject name alternative, but provides additional names that can be used to identify the issuer of the certificate.
    3. **Subject directory attributes:** Conveys any desired X.500 directory attribute values for the subject of this certificate.
- Certification path constraints
    1. **Basic constraints:** Indicates if the subject may act as a CA. If so, a certification path length constraint may be specified.
    2. **Name constraints:** Indicates a name space within which all subject names in subsequent certificates in a certification path must be located.
    3. **Policy constraints:** Specifies constraints that may require explicit certificate policy identification or inhibit policy mapping for the remainder of the certification path.

User Authentication:

Kerberos:

Kerberos is an authentication service that helps to restrict access to authorized users and authenticate requests for services in an open distributed environment where users at workstations want to access services on servers distributed throughout the network.

- Kerberos addresses three threats that exist in such an environment: users pretending to be other users, altering the network address of a workstation, and eavesdropping on exchanges.
- Kerberos relies exclusively on symmetric encryption and has two versions in common use: Version 4 and Version 5.
- Version 5 corrects some security deficiencies of Version 4 and has been issued as a proposed Internet Standard.
- Kerberos provides a centralized authentication server whose function is to authenticate users to servers and servers to users.
- This helps to prevent unauthorized users from gaining access to services and data that they are not authorized to access.

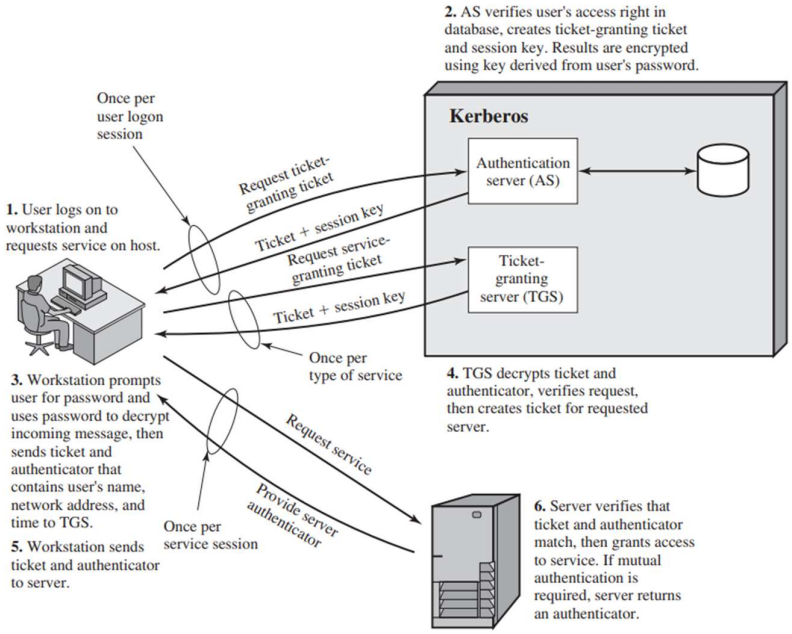
Motivation:

Kerberos is an authentication service that provides a centralized server to authenticate users and servers to each other.

- This approach relies on symmetric encryption, unlike public-key encryption used in other authentication schemes.
- Kerberos was developed to address the limitations of relying on individual workstations or trusting client systems concerning user identity.
- In an open environment, the user must prove their identity for each service invoked, and servers must prove their identity to clients.
- Kerberos assumes a distributed client/server architecture and employs one or more Kerberos servers to provide an authentication service.
- The requirements for Kerberos include security, reliability, transparency, and scalability.
- It uses a trusted third-party authentication service that relies on a protocol.

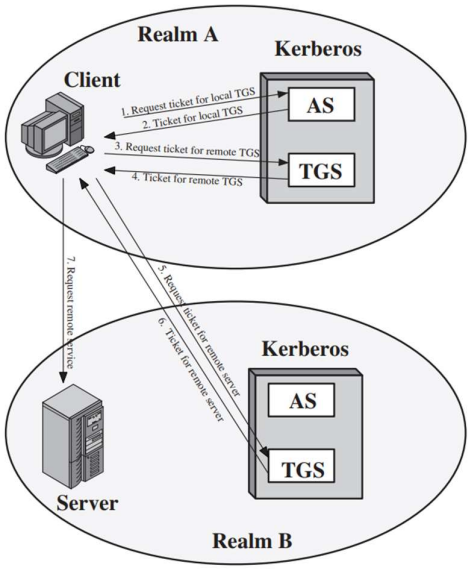
Kerberos Version 4:

- The user enters their username and password into the client machine.
- The client machine sends a request to the Kerberos authentication server (AS) for a ticket-granting ticket (TGT).
- The Kerberos AS verifies the user's credentials and sends back the TGT encrypted with a secret key known only by the user and the AS.
- The client machine decrypts the TGT using the user's secret key and sends a request for a service ticket to the ticket-granting server (TGS), along with the TGT.
- The TGS verifies the TGT and issues a service ticket encrypted with a secret key shared by the TGS and the requested service.
- The client machine decrypts the service ticket using the secret key shared between the TGS and the requested service and sends the decrypted ticket to the service to prove its identity.



Kerberos Version 5:

- The client sends an initial request to the authentication server (AS) that includes the client's identity.
- The AS responds with a pre-authentication token encrypted with the client's password.
- The client sends a TGT request to the ticket granting server (TGS) that includes the pre-authentication token and a request for a TGT for the desired service.
- The TGS verifies the client's identity and issues a TGT encrypted with the TGS secret key, which is sent to the client.
- The client sends a request for a service ticket to the TGS that includes the TGT and the desired service name.
- The TGS verifies the TGT and issues a service ticket encrypted with the service's secret key.
- The client presents the service ticket to the service, which decrypts it using its secret key and grants the client access to the requested service.



Transport-Level Security:

Web Security Threats:

	Threats	Consequences	Countermeasures
Integrity	• Modification of user data • Trojan horse browser • Modification of memory • Modification of message traffic in transit	• Loss of information • Compromise of machine • Vulnerabilty to all other threats	Cryptographic checksums
Confidentiality	• Eavesdropping on the net • Theft of info from server • Theft of data from client • Info about network configuration • Info about which client talks to server	• Loss of information • Loss of privacy	Encryption, Web proxies
Denial of Service	• Killing of user threads • Flooding machine with bogus requests • Filling up disk or memory • Isolating machine by DNS attacks	• Disruptive • Annoying • Prevent user from getting work done	Difficult to prevent
Authentication	• Impersonation of legitimate users • Data forgery	• Misrepresentation of user • Belief that false information is valid	Cryptographic techniques

Web Traffic Security Approaches:

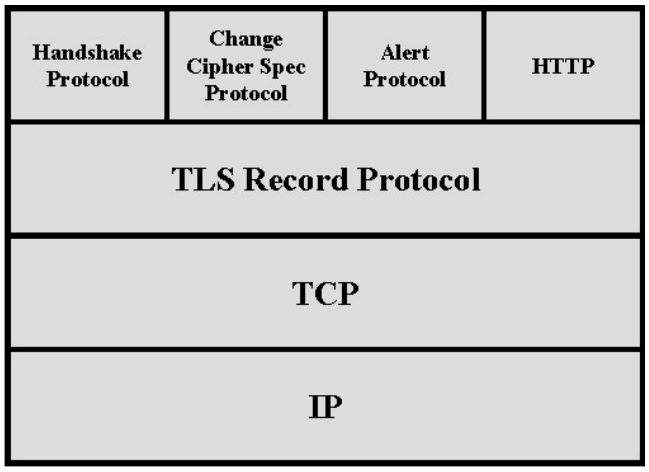
- **SSL/TLS Encryption:** This encrypts data in transit between the web server and client browser to prevent eavesdropping and data tampering.
- **Web Application Firewall (WAF):** This filters incoming traffic to web applications to block attacks such as SQL injection and cross-site scripting (XSS).
- **Intrusion Detection/Prevention Systems (IDS/IPS**): These systems monitor network traffic and alert or block suspicious activity that could indicate an attack.
- **Content Security Policy (CSP**): This is a set of directives that web developers can use to prevent cross-site scripting (XSS) attacks and other code injection attacks.
- **Two-Factor Authentication (2FA):** This adds an extra layer of authentication beyond a username and password to verify the identity of the user.
- **DDoS Protection:** This detects and mitigates Distributed Denial of Service (DDoS) attacks, which flood a web server with traffic to overwhelm it and make it inaccessible.

Transport Layer Security:

TLS Architecture:

Transport Layer Security (TLS) is a protocol that provides secure communication over a network.

- It is designed to prevent eavesdropping, tampering, and message forgery.
- TLS operates at the transport layer of the OSI model and sits between the application layer and the network layer.
- TLS uses a public-key infrastructure (PKI) to authenticate the identity of the communicating parties.



Handshake Protocol:
This protocol is a process that takes place between a client and a server to establish a secure connection for data transmission.

- The handshake protocol consists of a series of messages that are exchanged between the client and server to negotiate the cryptographic parameters and authenticate each other.

Change Cipher Spec Protocol:
This protocol is responsible for changing the encryption keys and algorithms in use during a TLS session.

- The CCS protocol consists of a ChangeCipherSpec message, which is a one-byte message that is sent by the sender to indicate that it is ready to switch to the new encryption parameters.
- The receiver must acknowledge the change by sending a ChangeCipherSpec message of its own.

Alert Protocol:
This protocol is responsible for sending messages between the client and server to indicate the occurrence of errors or abnormal conditions during the TLS session.

- The Alert protocol is used to transmit warning and fatal alerts between the client and server.

TLS Record Protocol:
The Record Protocol is responsible for the fragmentation and reassembly of higher-level protocol messages into manageable blocks, adding necessary TLS headers, applying encryption and integrity protection, and transmitting these blocks to the peer entity.

Cryptographic Computations:
Cryptographic computations refer to the mathematical operations and algorithms used to secure and protect sensitive data by encoding it in a way that only authorized parties can access it.

- Cryptographic computations typically involve complex mathematical operations, such as encryption, decryption, key generation, digital signatures, and hashing.
- These operations are designed to prevent unauthorized access, tampering, and interception of sensitive information by third parties.

HTTPS:

HTTPS is a combination of HTTP and SSL/TLS used for secure communication between web browsers and servers.

- URLs begin with https:// and use port 443.
- When HTTPS is used, the URL, document contents, form contents, cookies, and HTTP header are all encrypted.
- Both SSL and TLS implementations are referred to as HTTPS

Connection Initiation:

In HTTPS, the client sends a request to the server to establish a connection.

- The client also initiates the TLS (Transport Layer Security) handshake by sending a ClientHello message to the server.
- Once the handshake is complete, the client can then send HTTP requests to the server.
- All data sent over the connection is encrypted using TLS.
- There are three levels of awareness in HTTPS:
    1. **HTTP level:** The HTTP level is responsible for sending and receiving HTTP requests and responses over the TLS connection.
    2. **TLS level:** The TLS level establishes a secure session between the client and the server.
    3. **TCP level:** The TCP level establishes a connection between them.

Connection Closure:

When using HTTP, a client or server can indicate that the connection should be closed after the current request/response is completed.

- In HTTPS, closing the connection requires using the TLS protocol to close the connection with the other end, which involves closing the underlying TCP connection.
- To close the connection, each side must use the TLS alert protocol to send a close_notify alert.
- An HTTPS client must also be able to handle situations where the underlying TCP connection is terminated without notice.
- This could be due to a server error or a communication problem, but it could also indicate a security attack, so the client should issue a security warning when this occurs.