جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

كلية العلوم

قسم الأمن السيبراني

Lecture: (5)

X.509 Standard & Kerberos

Subject: authentication and access control

second Stage

Lecturer: Asst. Lecturer. Suha Alhussieny

X.509 Standard

In cryptography, **X.509** is an International Telecommunication Union (ITU) standard defining the format of public key certificates. X.509 is a standard format for public key certificates, X.509 certificates are used in many Internet protocols, including TLS/SSL, which is the basis for HTTPS, the secure protocol for browsing the web. They are also used in offline applications, like electronic signatures.

X.509 is part of the X.500 series of recommendations that define a directory service. The directory is, in effect, a server or distributed set of servers that maintains a database of information about users. The information includes a mapping from user name to network address, as well as other attributes and information about the users.

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 of the type. Each certificate contains the public key of a user and is signed with the private key of a trusted certification authority. In addition, X.509 defines alternative authentication protocols based on the use of public-key certificates. X.509 is an important standard because the certificate structure

and authentication protocols defined in X.509 are used in a variety of contexts.

X.509 is based on the use of public-key cryptography and digital signatures. The standard does not dictate the use of a specific algorithm but recommends

RSA. The digital signature scheme is assumed to require the use of a hash function.

One scheme has become universally accepted for formatting public-key certificates: the X.509 standard. X.509 certificates are used in most network security applications, including IP security, transport layer security (TLS), and S/MIME,

Typically, public-key infrastructure (PKI) implementations make use of X.509 certificates.

X.509 Certificates

X.509 certificates are digital documents that are used to verify the identity of individuals, organizations, or devices over the internet. They are widely used in various applications like secure email, web browsing, online banking, and electronic transactions.

X.509 certificates are a standard format for public key certificates which are used in various security protocols to establish a secure and authenticated connection between parties over networks .

An X.509 certificate binds an identity to a public key using a digital signature. A certificate contains an identity (a hostname, or an organization, or an individual) and a public key (RSA, DSA, ECDSA, ed25519, etc.), and is either signed by a certificate authority or is self-signed. When a certificate is signed by a trusted certificate authority, or validated by other means, someone holding that

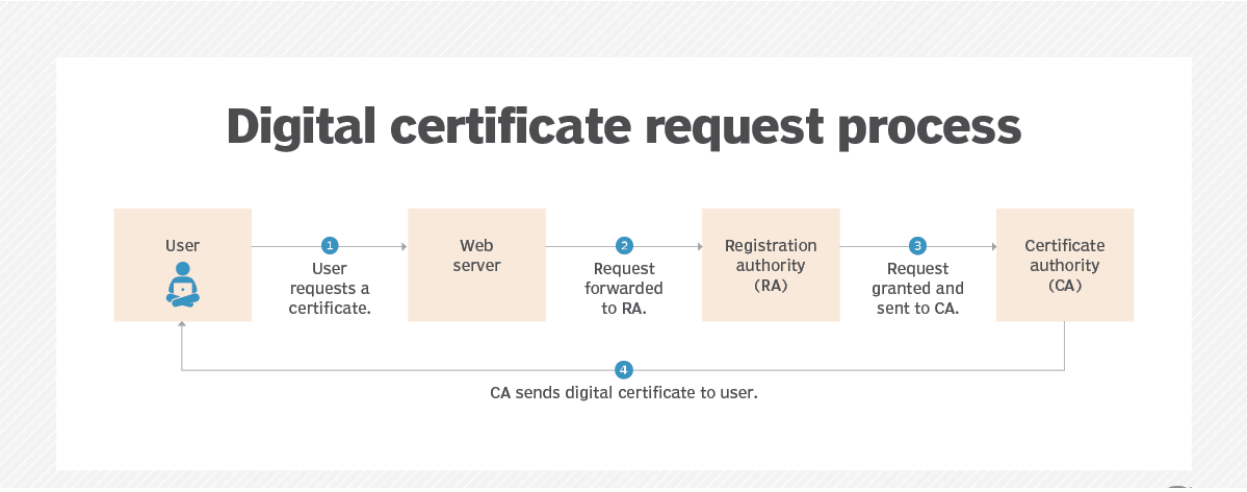
certificate can use the public key it contains to establish secure communications with another party, or validate documents digitally signed by the corresponding private key.

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. The directory server itself is not responsible for the creation of public keys or for the certification function; it merely provides an easily accessible location for users to obtain certificates.

X.509 certificates bind an identity to a public key using a digital signature. In the X.509 system, there are two types of certificates. The first is a CA certificate. The second is an end-entity certificate. A CA certificate can issue other certificates. The top level, self-signed CA certificate is sometimes called the Root CA certificate. Other CA certificates are called intermediate CA or subordinate CA certificates. An end-entity certificate identifies the user, like a person, organization or business. An end-entity certificate *cannot* issue other certificates. An end-entity certificate is sometimes called a leaf certificate since no other certificates can be issued below it.

The X.509 certificate is a safeguard against malicious network impersonators. When a certificate is signed by a trusted authority, or is otherwise validated, the device holding the certificate can validate documents. It can also use a public key certificate to secure communications with a second party.

The **use of X.509 certificates** ensures that the communication is encrypted and authenticated, thereby providing a high level of security for online transactions.



The elements of X.509 certificates include the following elements; -

- **Version:** Differentiates among successive versions of the certificate format;

1. the default is version 1.
2. If the issuer unique identifier or subject unique identifier are present, the value must be version 2.
3. 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.

Kerberos

Kerberos is an authentication service designed for use in a distributed environment. Kerberos provides a trusted third-party authentication service that enables clients and servers to establish authenticated communication.

Kerberos is an authentication service developed as part of Project Athena at MIT. The problem that Kerberos addresses is this: Assume an open distributed

environment in which users at workstations wish to access services on servers distributed throughout the network. We would like for servers to be able to restrict access to authorized users and to be able to authenticate requests for service. In this environment, a workstation cannot be trusted to identify its users correctly to network services.

In particular, the following three ***threats*** exist:

1. A user may gain access to a particular workstation and pretend to be another user operating from that workstation.
2. A user may alter the network address of a workstation so that the requests sent from the altered workstation appear to come from the impersonated workstation.
3. A user may eavesdrop on exchanges and use a replay attack to gain entrance to a server or to disrupt operations.

In any of these cases, an unauthorized user may be able to gain access to services and data that he or she is not authorized to access. Rather than building in elaborate authentication protocols at each server, Kerberos provides a centralized authentication server whose function is to authenticate users to servers and servers to users. Unlike most other authentication schemes described, Kerberos relies exclusively on symmetric encryption, making no use of public-key encryption.

Kerberos Motivation

If a set of users is provided with dedicated personal computers that have no network connections, then a user's resources and files can be protected by physically securing each personal computer. When these users instead are served by a centralized timesharing system, the time-sharing operating system must provide the security. The operating system can enforce access-control policies based on user identity and use the logon procedure to identify users.

Today, neither of these scenarios is typical. More common is a distributed architecture consisting of dedicated user workstations (clients) and distributed or centralized servers. In this environment, **three approaches to security can be envisioned.**

1. Rely on each individual client workstation to assure the identity of its user or users and rely on each server to enforce a security policy based on user identification (ID).

2. Require that client systems authenticate themselves to servers, but trust the client system concerning the identity of its user.

3. Require the user to prove his or her identity for each service invoked. Also require that servers prove their identity to clients.

Instead of using traditional passwords, Kerberos employs strong, time-limited secret-key cryptography, multiple secret keys, and a third-party service to authenticate client-server applications and user identities. In creating Kerberos, MIT developers wanted to support both authentication and authorization so that once a user is authenticated, they're also authorized.

EXAMPLE

- Josh sits down at his desk exactly at 9 a.m. He's concerned about being late for his 9 a.m. Zoom call, so he quickly logs onto his computer.

- Because his company uses Kerberos, he doesn't have to sign into Zoom. He can just click on the Zoom link, and he is automatically authenticated and can enter the meeting.

- During the meeting, Josh needs to access his project management software to present some information. He clicks on the application, and it opens instantly without making the other meeting guests wait while he signs in.

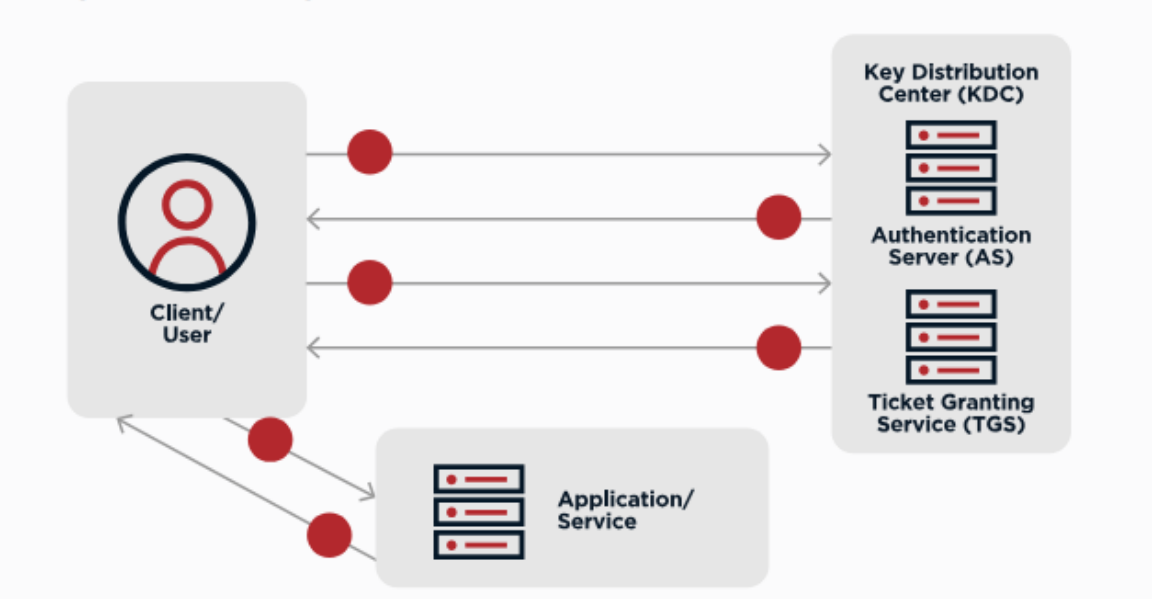
- After the meeting, he has to access sensitive information in a secure database. When he clicks on the link to enter, he is immediately authenticated through Kerberos and does not have to enter separate credentials.

Here are the principal entities involved in the typical Kerberos workflow:

- Client: The client acts on behalf of the user and initiates communication for a service request
- Server: The server hosts the service the user wants to access
- Authentication Server (AS): The AS performs the desired client authentication. If the authentication happens successfully, the AS issues the client a ticket called TGT (Ticket Granting Ticket). This ticket assures the other servers that the client is authenticated

- Key Distribution Center (KDC): In a Kerberos environment, the authentication server logically separated into three parts: A database (db), the Authentication Server (AS), and the Ticket Granting Server (TGS). These three parts, in turn, exist in a single server called the Key Distribution Center
- Ticket Granting Server (TGS): The TGS is an application server that issues service tickets as a service

How Entities Communicate Within Kerberos



Client/User and Authentication Server (AS)

1. **Query:** An access request is sent from the user via a secret key to the authentication server (AS) in the key distribution center (KDC).
2. **Response:** The AS uses the secret key to authenticate the user (by comparing their credentials in the database). Once authenticated, the AS

sends its own secret key along with a time-limited token back to the user. The user accepts the key and the time-limited token from the AS.

Client/User and Ticket-granting Service (TGS)

3. **Query:** The user takes the key and time-limited token received from the AS and sends them to the TGS along with a request to access the application.
4. **Response:** When the TGS gets the request, it decrypts it with the secret key that was shared with the AS and issues the client a token, which is encrypted with another secret key.

Client/User and Application/Server

5. **Query:** The client takes the token they got from the TGS and sends it with a secret key to the application they want to access. (Note: Once the client possesses the token from the TGS, the other services can be assured that the client is authenticated.
6. **Response:** When the application receives the request, it encrypts the token with a secret key and shares it back to the client and the TGS.
    - **Access Granted to User:** The application allows the user access for a certain period of time according to the limitations of the token.

Once the checks are met, the target server sends the client a message verifying that the client and the server have authenticated each other. The user can now engage in a secure session.