Unit-9 –Remote User Authentication

Remote user authentication is a mechanism in which the **remote** server verifies the legitimacy of a **user** over an insecure communication channel.

Remote User-Authentication using Symmetric Encryption

Mutual Authentication

- Two-level hierarchy of symmetric encryption keys can be used to provide confidentiality for communication in a distributed environment.
- In general, this strategy involves the use of a trusted key distribution center (KDC).
- Each party in the network shares a secret key, known as a master key, with the KDC.
- The KDC is responsible for generating keys to be used for a short time over a connection between two parties, known as session keys, and for distributing those keys using the master keys to protect the distribution.
- Proposal initially put forth by Needham and Schroeder for secret key distribution using a KDC includes authentication features.
- The protocol can be summarized as follows.
  1. $A \rightarrow KDC$: $ID_A||ID_B||N_1$
  2. $KDC \rightarrow A$: $E(K_a, [K_s||ID_B||N_1||E(K_b, [K_s||ID_A])])$
  3. $A \rightarrow B$: $E(K_b, [K_s||ID_A])$
  4. $B \rightarrow A$: $E(K_s, N_2)$
  5. $A \rightarrow B$: $E(K_s, f(N_2))$
- The protocol is still vulnerable to a form of replay attack.
- Suppose that an opponent, X, has been able to compromise an old session key.
- X can impersonate A and trick B into using the old key by simply replaying step 3.
- Unless B remembers indefinitely all previous session keys used with A, B will be unable to determine that this is a replay.
- If X can intercept the handshake message in step 4, then it can impersonate A's response in step 5.
- From this point on, X can send bogus messages to B that appear to B to come from A using an authenticated session key.
- Denning proposes to overcome this weakness by a modification to the Needham/Schroeder protocol that includes the addition of a timestamp to steps 2 and 3.
- Her proposal assumes that the master keys, K_a and K_b, are secure, and it consists of the following steps.
  1. $A \rightarrow KDC$: $ID_A||ID_B$
  2. $KDC \rightarrow A$: $E(K_a, [K_s||ID_B||T||E(K_b, [K_s||ID_A||T])])$
  3. $A \rightarrow B$: $E(K_b, [K_s||ID_A||T])$
  4. $B \rightarrow A$: $E(K_s, N_1)$
  5. $A \rightarrow B$: $E(K_s, f(N_1))$
- T is a timestamp that assures A and B that the session key has only just been generated.
- Thus, both A and B know that the key distribution is a fresh exchange.
- A and B can verify timeliness by checking that

$$|Clock - T| < \Delta t_1 + \Delta t_2$$

- Where Δt_1, is the estimated normal discrepancy between the KDC's clock and the local clock (at A or B) and Δt_2 is the expected network delay time.
- A new concern is raised: namely, that this new scheme requires reliance on clocks that are synchronized throughout the network points out a risk involved.
- The risk is based on the fact that the distributed clocks can become unsynchronized as a result of sabotage on or faults in the clocks or the synchronization mechanism.
- The problem occurs when a sender's clock is ahead of the intended recipient's clock. In this case, an opponent can intercept a message from the sender and replay it later when the timestamp in the message becomes current at the recipient's site.
- This replay could cause unexpected results.

- Gong refers to such attacks as **suppress-replay attacks**.
- One way to counter suppress-replay attacks is to enforce the requirement that parties regularly check their clocks against the KDC's clock.
- The other alternative, which avoids the need for clock synchronization, is to rely on handshaking protocols using nonces.
- This latter alternative is not vulnerable to a suppress-replay attack, because the nonces the recipient will choose in the future are unpredictable to the sender.
- The Needham/Schroeder protocol relies on nonces only but, as we have seen, has other vulnerabilities.
- Improved strategy was presented in this protocol is
  1. $A \rightarrow B$: $ID_A||N_a$
  2. $B \rightarrow KDC$: $ID_B||N_b||E(K_b, [ID_A||N_a||T_b])$
  3. $KDC \rightarrow A$: $E(K_a, [ID_B||N_a||K_s||T_b])||E(K_b, [ID_A||K_s||T_b])||N_b$
  4. $A \rightarrow B$: $E(K_b, [ID_A||K_s||T_b])||E(K_s, N_b)$
- This protocol provides an effective, secure means for A and B to establish a session with a secure session key.
- Furthermore, the protocol leaves A in possession of a key that can be used for subsequent authentication to B, avoiding the need to contact the authentication server repeatedly.
- Suppose that A and B establish a session using the aforementioned protocol and then conclude that session.
- Subsequently, but within the time limit established by the protocol, A desires a new session with B.
- The following protocol ensues:
  1. $A \rightarrow B$: $E(K_b, [ID_A||K_s||T_b])||N_a{}'$
  2. $B \rightarrow A$: $N_b{}'||E(K_s, N_a{}')$
  3. $A \rightarrow B$: $E(K_s, N_b{}')$
- When B receives the message in step 1, it verifies that the ticket has not expired.
- The newly generated nonces and assure each party that there is no replay attack.

One-Way Authentication

- With some refinement, the KDC strategy is a candidate for encrypted electronic mail.
- Because we wish to avoid requiring that the recipient (B) be on line at the same time as the sender (A), steps 4 and 5 must be eliminated.
- For a message with content , the sequence is as follows:
  1. $A \rightarrow KDC$: $ID_A||ID_B||N_1$
  2. $KDC \rightarrow A$: $E(K_a, [K_s||ID_B||N_1||E(K_b, [K_s||ID_A])])$
  3. $A \rightarrow B$: $E(K_b, [K_s||ID_A])||E(K_s, M)$
- This approach guarantees that only the intended recipient of a message will be able to read it.
- It also provides a level of authentication that the sender is A.
- The protocol does not protect against replays.

Kerberos

- Kerberos is an authentication protocol.
- It provides a way to authenticate clients to services to each other through a trusted third party.

Requirements of Kerberos

- **Secure**: Kerberos should be strong enough that a potential opponent does not find it to be the weak link.
- **Reliable**: 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 distributed server architecture, with one system able to back up another.

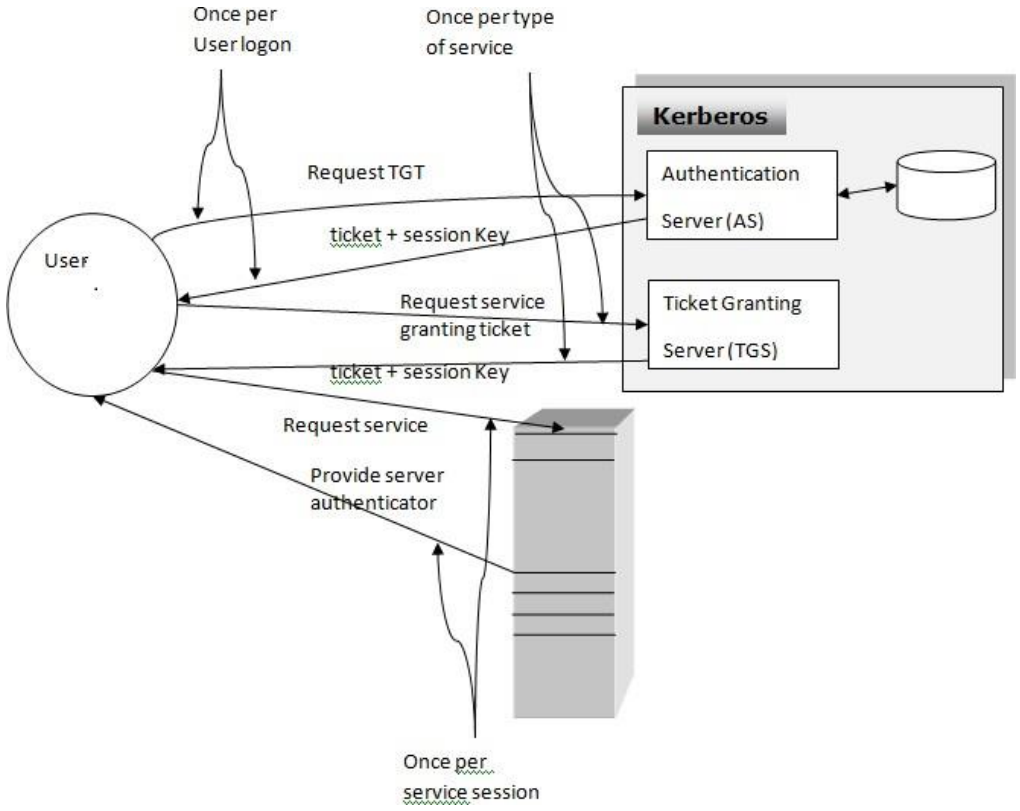
- **Transparent**: Ideally, the user should not be aware that authentication is taking place, beyond the requirement to enter a password.
- **Scalable**: The system should be capable of supporting large numbers of clients and servers. This suggests a modular, distributed architecture.

Kerberos protocol Terminology

1. **Authentication Server (AS)**: A server that issues tickets for a desired service which are in turn given to users for access to the service.
2. **Client**: An entity on the network that can receive a ticket from Kerberos.
3. **Credentials**: A temporary set of electronic credentials that verify the identity of a client for a particular service. It also called a ticket.
4. **Credential cache or ticket file**: A file which contains the keys for encrypting communications between a user and various network services.
5. **Crypt hash**: A one-way hash used to authenticate users.
6. **Key**: Data used when encrypting or decrypting other data.
7. **Key distribution center (KDC)**: A service that issue Kerberos tickets and which usually run on the same host as the ticket-granting server (TGS).
8. **Realm**: A network that uses Kerberos composed of one or more servers called KDCs and a potentially large number of clients.
9. **Ticket-granting server (TGS)**: A server that issues tickets for a desired service which are in turn given to users for access to the service. The TGS usually runs on the same host as the KDC.
10. **Ticket-granting ticket (TGT)**: A special ticket that allows the client to obtain additional tickets without applying for them from the KDC.

Overview of Kerberos

- The overview of Kerberos is shown and described below:



- User logs onto workstation and request on host for TGT.
- AS verifies user's access right in database, creates ticket-granting ticket and session key. Results are encrypted using key derived from user's password.
- Workstation prompts the user for password and uses password to decrypt incoming message, then sends ticket and authentication that contain user's name, network address and time to TGS.
- TGS decrypts the ticket and authenticator, verifies request, then creates ticket for requested server.
- Workstation sends ticket and authentication to server.
- Server verifies that ticket and if the authenticator matches, then grants access to service. If mutual authentication is required, server returns an authenticator.
- The figure shows the above exchange.

Kerberos Authentication Dialogue

- The following message exchanges take place for authentication through Kerberos: **Authentication service exchange to obtain Ticket granting ticket (TGT)**
- This exchange takes place only once per a user logon session.
- User obtains a **Ticket Granting Ticket** from the authentication server. This ticket is sent to the **Ticket Granting Server** to obtain service tickets.

C → AS: IDc || IDtgs || TS1

AS → C: E(Kc, [Kc, tgs || IDtgs || TS2 || Lifetime2 || Tickettgs])

Tickettgs = E(Ktgs, [Kc, tgs || IDC || ADC ||IDtgs || TS2 || Lifetime2])

Ticket-Granting service exchange to obtain Service Granting Ticket

- This exchange takes place for each type of service.
- Here, the user presents the TGT to the ticket granting server. The TGS return a **Service Granting Ticket** to the user after proper authentication.
- An authenticator is added in the message which is encrypted using the key shared by the user and TGS

C → TGS: IDv || Tickettgs || Authenticatorc

TGS → C: E(Kc, tgs, [Kc, v || IDV || TS4 || Lifetime4 || Ticketv])

Ticketv = E(Kv, [Kc, v || IDC || ADC ||IDV || TS4 || Lifetime4])

Authenticatorc = E(Kc, tgs, [IDC ||ADC || TS3])

Client-Server authentication exchange to obtain service

- The user sends the Service Granting Ticket to the application server (of which the service is needed).
- The message also contains authenticator which proves the sender's identity to the server. Moreover, the server replies with the timestamp present in the authenticator. This authenticates the server to the user.

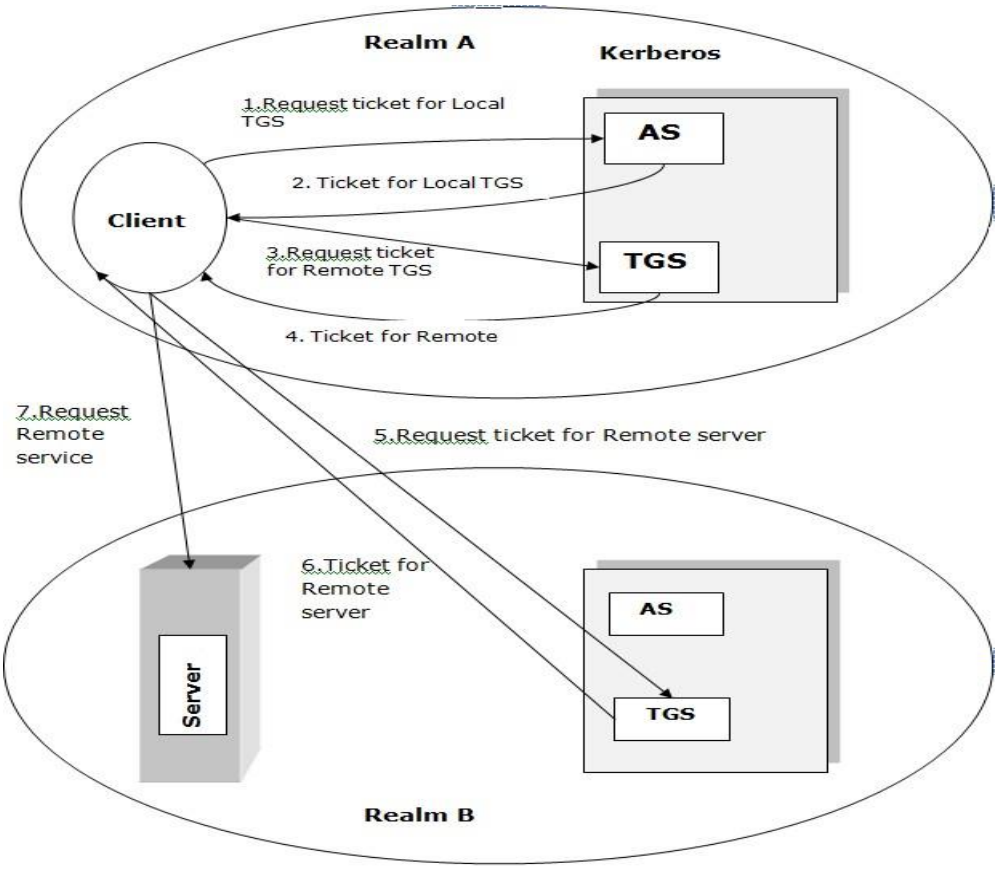
C → V: Ticketv || Authenticatorc

V → C: E(Kc, v, TS5 +1)

Authenticatorc = E(Kc, v, [IDC ||ADC || TS5])

Kerberos Realm

- 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 using Kerberos master password.
- A Kerberos principal, is a service or user that is known to the Kerberos system. Each Kerberos principal is identified by its principal name.
- Networks of clients and servers under different administrative organizations constitute different realms.
- For inter realm communication, the Kerberos servers in the two realms must be authenticated and registered to each other.



Request for Service in Another Realm

- A user wishing service on a server in another realm obtains a ticket for that server as given below:
  1. **C →AS: IDc||IDtgs||TS1**
  2. **AS→ C: E(Kc, [Kc,tgs||IDtgs||TS2||Lifetime2||Tickettgs]**
  3. **C→ TGS: IDtgsrem||Tickettgs||Authenticatorc**
  4. **TGS→ C: E(Kc,tgs, [Kc,tgsrem||IDtgsrem||TS4||Tickettgsrem])**
  5. **C→ TGSrem: IDvrem||Tickettgsrem||Authenticatorc**
  6. **TGSrem→ C: E(Kc,tgsrem , [Kc,vrem||IDvrem||TS6||Ticketvrem])**
  7. **C→ Vvrem: Ticketvrem||Authenticatorc**

  where **IDtgsrem** is the identity of remote TGS,
  Tickettgsrem is the TGT for remote TGS,
  IDvrem is the identity of remote server and
  Ticketvrem is the Service granting ticket for remote server.

Comparison between Kerberos Version 4 And Version 5

- The environmental shortcomings of Kerberos version 4 and their corresponding improvements in version 5 are listed below:
  1. **Encryption system dependence**: Version 4 requires the use of DES whereas version 5 includes a ciphertext tag with an encryption type identifier so that any encryption technique may be used.
  2. **Internet protocol dependence**: Version 4 requires the use of Internet Protocol (IP) addresses. Version 5 allows any network address type to be used.
  3. **Message byte ordering**: In version 4, the sender of a message employs a byte ordering of its own choosing but in version 5, all message structures are defined using Abstract Syntax Notation One (ASN.1) and Basic Encoding Rules (BER), which provide an unambiguous byte ordering.
  4. **Ticket lifetime**: Lifetime values in version 4 are encoded in an 8-bit, each unit of 5 minutes. Thus, the maximum lifetime that can be expressed is 28 x 5 = 1280 minutes, or a little over 21 hours. In version 5, tickets include an explicit start time and end time, allowing tickets with arbitrary lifetimes.
  5. **Authentication forwarding**: Version 4 does not allow credentials issued to one client to be forwarded to some other host and used by some other client. For example, a client issues a request to a print server that then, cannot access the client's file from a file server, using the client's credentials for access. Version 5 provides this capability.
  6. **Inter realm authentication**: In version 4, interoperability among realms requires many Kerberos-to-Kerberos relationships but version 5 supports a method that requires fewer relationships.
- Technical deficiencies of version 4 and its alternate in version 5 are:
  1. **Double encryption**: The tickets provided to clients are encrypted twice, once with the secret key of the target server and then again with a secret key known to the client. The second encryption is not necessary and is computationally wasteful.
  2. **PCBC encryption**: Encryption in version 4 makes use of a nonstandard mode of DES known as propagating cipher block chaining (PCBC) which is vulnerable to attack. Version 5 allows the standard CBC mode to be used for encryption.
  3. **Session keys**: Each ticket includes a session key used for encrypting messages. However, because the same ticket may be used repeatedly, replay attack is possible. In version 5, it is possible for a client and server to negotiate a subsession key, which is to be used only for that one connection.
  4. **Password attacks**: Both versions are vulnerable to a password attack. The message from the AS to the client includes material encrypted with a key based on the client's password. An opponent can capture this message and attempt to decrypt it by trying various passwords. Thus the opponent can discover the client's password and may subsequently use it to gain authentication credentials from Kerberos.

Remote User Authentication using Asymmetric Encryption

Mutual Authentication

- This protocol assumes that each of the two parties is in possession of the current public key of the other.
- A protocol using timestamps is:
  1. $A \rightarrow AS$: $ID_A||ID_B$
  2. $AS \rightarrow A$: $E(PR_{as}, [ID_A||PU_a||T])||E(PR_{as}, [ID_B||PU_b||T])$
  3. $A \rightarrow b$: $E(PR_{as}, [ID_A||PU_a||T])||E(PR_{as}, [ID_B||PU_b||T])||E(PU_b, E(PR_a, [K_s||T]))$
- In this case, the central system is referred to as an authentication server (AS), because it is not actually responsible for secret-key distribution.
- AS provides public-key certificates.
- The session key is chosen and encrypted by A; hence, there is no risk of exposure by the AS.
- The timestamps protect against replays of compromised keys.

- This protocol is compact but, as before, requires the synchronization of clocks.
- Another approach, proposed by Woo and Lam, makes use of nonces. The protocol consists of the following steps.
  1. $A \rightarrow KDC$: $ID_A||ID_B$
  2. $KDC \rightarrow A$: $E(PR_{auth}, [ID_B||PU_b])$
  3. $A \rightarrow B$: $E(PU_b, [N_a||ID_A])$
  4. $B \rightarrow KDC$: $ID_A||ID_B||E(PU_{auth}, N_a)$
  5. $KDC \rightarrow B$: $E(PR_{auth}, [ID_A||PU_a])||E(PU_b, E(PR_{auth}, [N_a||K_s||ID_B]))$
  6. $B \rightarrow A$: $E(PU_a, [E(PR_{auth}, [N_a||K_s||ID_B])||N_b])$
  7. $A \rightarrow B$: $E(K_s, N_b)$
- This seems to be a secure protocol that takes into account the various attacks.
- However, the authors themselves spotted a flaw and submitted a revised version of the algorithm:
  1. $A \rightarrow KDC$: $ID_A||ID_B$
  2. $KDC \rightarrow A$: $E(PR_{auth}, [ID_B||PU_b])$
  3. $A \rightarrow B$: $E(PU_b, [N_a||ID_A])$
  4. $B \rightarrow KDC$: $ID_A||ID_B||E(PU_{auth}, N_a)$
  5. $KDC \rightarrow B$: $E(PR_{auth}, [ID_A||PU_a])||E(PU_b, E(PR_{auth}, [N_a||K_s||ID_A||ID_B]))$
  6. $B \rightarrow A$: $E(PU_a, [E(PR_{auth}, [N_a||K_s||ID_A||ID_B])||N_b])$
  7. $A \rightarrow B$: $E(K_s, N_b)$
- The identifier of A, ID_A, is added to the set of items encrypted with the KDC's private key in steps 5 and 6.
- This binds the session key to the identities of the two parties that will be engaged in the session.
- This inclusion of accounts for the fact that the nonce value is considered unique only among all nonces generated by A, not among all nonces generated by all parties.

One-Way Authentication

- We have already presented public-key encryption approaches that are suited to electronic mail.
- These approaches require that either the sender know the recipient's public key (confidentiality), the recipient know the sender's public key (authentication), or both (confidentiality plus authentication).
- In addition, the public-key algorithm must be applied once or twice to what may be a long message.
- If confidentiality is the primary concern, then the following may be more efficient:

$$A \rightarrow B: \quad E(PU_b, K_s)||E(K_s, M)$$

- In this case, the message is encrypted with a one-time secret key.
- A also encrypts this one-time key with B's public key.
- Only B will be able to use the corresponding private key to recover the one-time key and then use that key to decrypt the message.
- This scheme is more efficient than simply encrypting the entire message with B's public key.
- If authentication is the primary concern, then a digital signature may suffice:

$$A \rightarrow B: \quad M||E(PR_a, H(M))$$

- This method guarantees that A cannot later deny having sent the message.
- However, this technique is not provide confidentiality.
- To counter such a problem, both the message and signature can be encrypted with the recipient's public key:

$$A \rightarrow B: \quad E(PU_b\ , [M||E(PR_a, H(M))])$$

- The latter two schemes require that B know A's public key and be convinced that it is timely.
- An effective way to provide this assurance is the digital certificate. Now we have

$$A \rightarrow B: \quad M||E(PR_a, H(M))||E(PR_{as}, [T||ID_A||PU_a])$$