Unit V: Authentication

Er. Kobid Karkee
Himalaya College of Engineering

Introduction

- Authentication is the process of recognizing a user's identity. It is the mechanism of associating an incoming request with a set of identifying credentials.
- The credentials provided are compared to those on a file in a database of the authorized user's information on a local operating system or within an authentication server.
- Authentication is the process of determining whether a user (or other entity) should be allowed access to a system.
- Only Authenticated users are allowed access to system resources.

Introduction

- Note that authentication is a binary decision— access is granted or it is not—while authorization is all about a more fine-grained set of restrictions on access to various system resources
  - Authentication: Are you who you say you are?
  - Authorization: Are you allowed to do that?

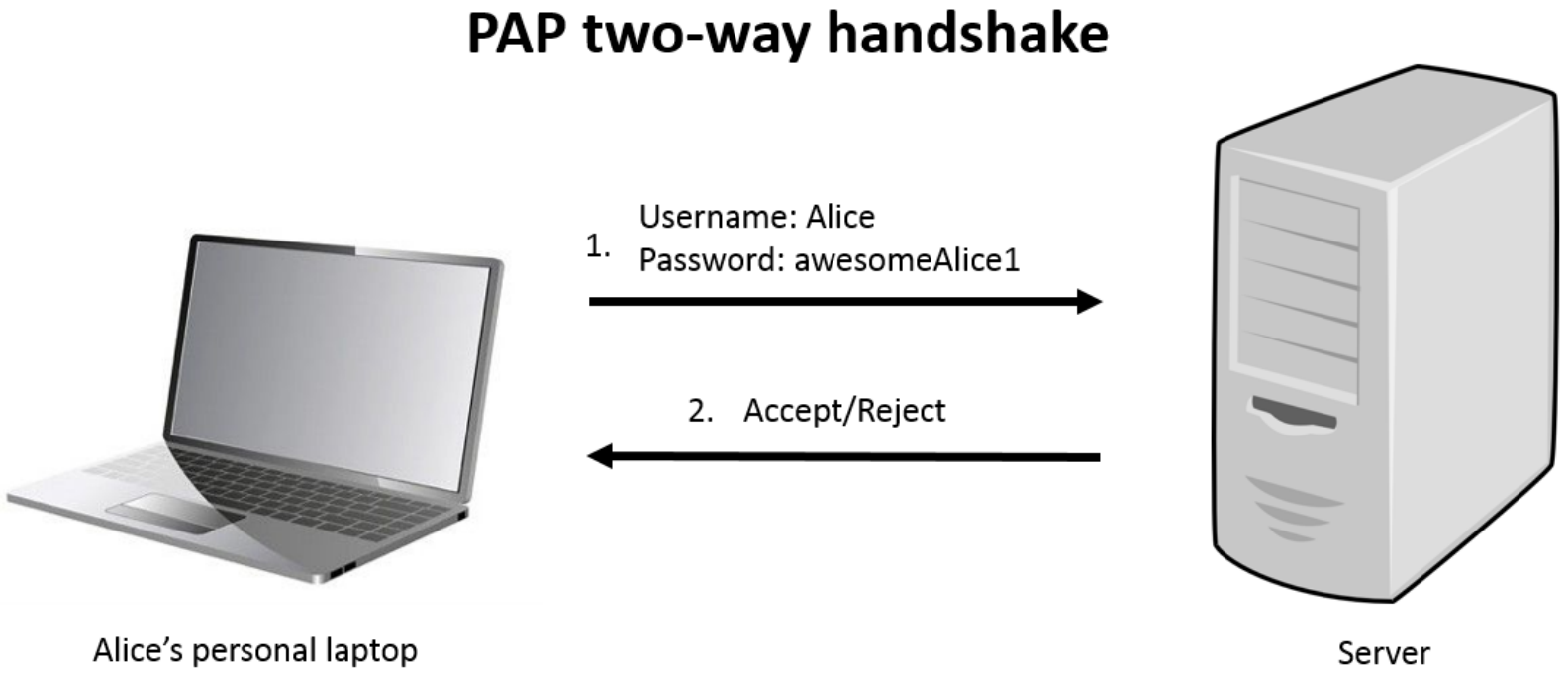
Authentication System

- Technique that provides access control for systems by checking to see if a user's credentials match the credentials in a database of authorized users or in a data authentication server.
- Authentication is any process by which a system verifies the identity of a user who wishes to access it.
- Three Factors in Authentication System:
  1. Something you know
     - Passwords/Secret key
  2. Something you have
     - Secure tokens/smart card/ ATM card
  3. Something you are
     - Biometrics (eg: fingerprint)

Password-based Authentication

- A password is a string of alphabets, numbers and special characters, which is supposed to be known only to the entity (usually person) that is being authenticated.
  - Prompt for user ID and Password
  - User enters user ID and Password
  - User ID and Password Validation
  - Authentication Result
  - Inform user accordingly
- Passwords are often stored as hash value of original password.

Password-based Authentication



Dictionary Attack

- A type of brute force attack where an intruder attempts to crack a password- protected security system with a "dictionary list" of common words and phrases used by businesses and individuals.
- A dictionary attack is a password attack that attempts to determine a password by trying words from a predefined list, or dictionary, of likely passwords.
- Dictionary attacks often succeed because many people tend to choose short passwords that are ordinary words or common passwords, or simple variants obtained, for example, by appending a digit or punctuation character.

Dictionary Attack

- Dictionary attacks are relatively easy to defeat, e.g. by using a passphrase or otherwise choosing a password that is not a simple variant of a word found in any dictionary or listing of commonly used passwords.
- Using a strong, uncommon password will make an attacker's job more difficult, but not impossible.

Defence against Dictionary and Brute-force Attacks

Slow down repeated logins:

- This is the simplest countermeasure available.
- An end user is unlikely to notice a 0.1 second delay while logging in, but that delay would accumulate quickly for an attacker, especially if they cannot parallelize their attempts.

Force captchas after multiple failed logins:

- While a user could have simply forgotten which password they used for the account, this will help slow down an attacker significantly.
- This is a great deterrent method as for modern captchas are difficult to defeat with computers.
- Many captchas need manual inputs in order to be solved.

Defence against Dictionary and Brute-force Attacks

Lock accounts:

- Even better, a system can be configured to lock an account after a specified number of attempted logins.
- Many websites will trigger additional protections for accounts with repeated bad password attempts.
- In the extreme case, for example, an iPhone will self-destruct (wipe all data) after10 tries.

Defence against Dictionary and Brute-force Attacks

Refresh passwords:

- Modern systems typically require users to cycle passwords regularly.
- Some corporate environments require users to change passwords every 90 days, or maybe even every 30 days.
- The rationale behind this is that an attacker who is attempting a brute- force attack against a complex password would need weeks to succeed.
- If the password changes during that time frame, the attacker will need to start over.
- However, as many users would confess, these strict password requirements can backfire, with users choosing weaker, sequential passwords ('Crypto2018,' 'Crypto2019,' and so on).
- An attacker would quickly try incrementing the password.

Challenge Response System

- Challenge Response Authentication Mechanism (CRAM) is the most often used way to authenticate actions.
- They are a group of protocols in which one side presents a challenge (to be answered) and the other side must present a correct answer (to be checked/validated) to the challenge in order to get authenticated.
- Following Protocol is a challenge-and-response protocol.
- In it, we assume that Alice is identifying herself to Bob, and their common secret key is denoted by K. (Bob can also identify himself to Alice, by interchanging the roles of Alice and Bob in the scheme.)
- In this scheme, Bob sends a challenge to Alice, and then Alice sends Bob her response.(Message authentication code)

Challenge Response System

1. Bob chooses a random challenge, denoted by r, which he sends to Alice.
2. Alice computes her response

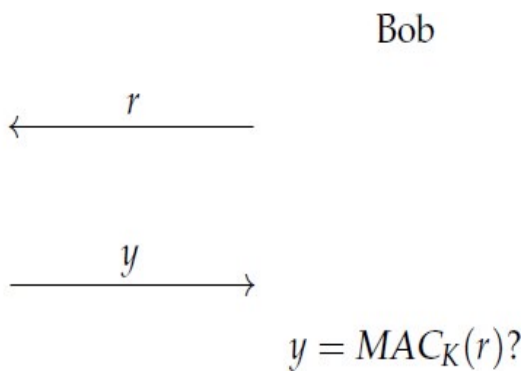
$$y = MAC_K(r)$$

and she sends y to Bob.

3. Bob computes

$$y' = MAC_K(r).$$

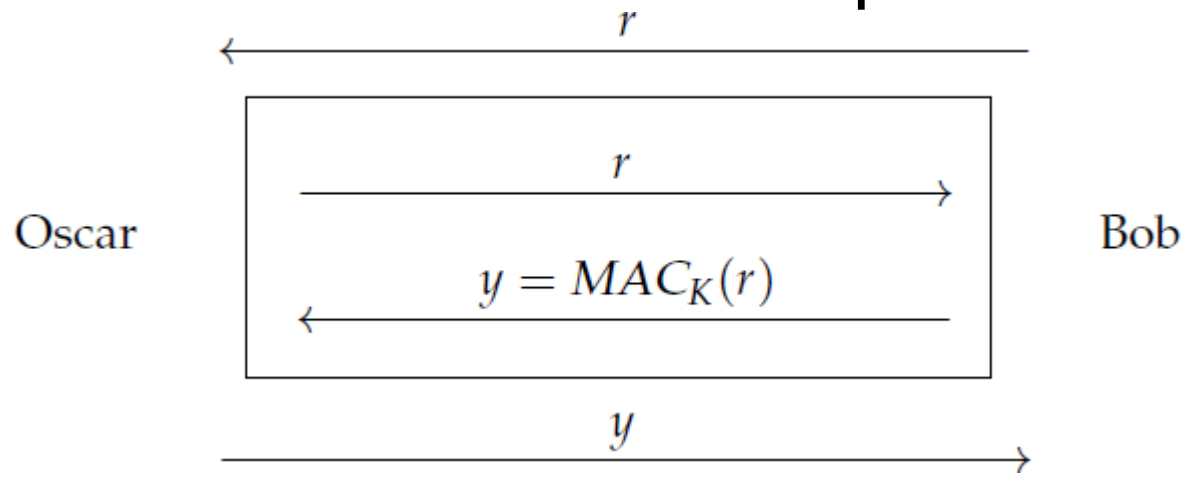
If $y' = y$, then Bob "accepts"; otherwise, Bob "rejects."



But this protocol is insecure (because of parallel session attack)

Parallel Session Attack

- In parallel session attack Oscar impersonates Alice.



- Within the first session (in which it is supposed that Oscar is impersonating Alice to Bob), Oscar initiates a second session in which he asks Bob to identify himself.
- This second session is boxed in above figure.

Parallel Session Attack

- In this second session, Oscar gives Bob the same challenge that he received from Bob in the first session.
- Once he receives Bob's response, Oscar resumes the first session, in which he relays Bob's response back to him.
- Thus, Oscar is able to successfully complete the first session.

Parallel Session Attack

- Following is the **secure challenge response protocol:**

1. Bob chooses a random challenge, r, which he sends to Alic
2. Alice computes
$$y = MAC_K(ID(Alice) \parallel r)$$
and sends y to Bob.
3. Bob computes
$$y' = MAC_K(ID(Alice) \parallel r).$$
If $y' = y$, then Bob "accepts"; otherwise, Bob "rejects."

- Note: A scheme in which Alice and Bob are both proving their identities to each other is called mutual authentication or mutual identification. Both participants are required to "accept" if a session of the scheme is to be considered a successfully completed session.

Biometric Systems

(Fingerprint, face recognition, eye retina, iris recognition etc)

- Biometrics represent the "something you are" method of authentication.
- “You are your key”
- There are many different types of biometrics, including such long-established methods as fingerprints.
- Recently, biometrics based on speech recognition, gait (walking) recognition, and even a digital doggie (odor recognition) have been developed.
- Biometrics are currently a very active topic for research.

Biometric Systems

- In the information security arena, biometrics are seen as a more secure alternative to passwords.
- For biometrics to be a practical replacement for passwords, cheap and reliable systems are needed.
- Today, usable biometric systems exist, including laptops/smartphones using thumbprint authentication, palm print systems for secure entry into restricted facilities, the use of fingerprints to unlock car doors, and so on.

Biometric Systems

An ideal biometric would satisfy all the following:

Universal:

- A biometric should apply to virtually everyone.
- In reality, no biometric applies to everyone.
- For example, a small percentage of people do not have readable fingerprints.

Distinguishing:

- A biometric should distinguish with virtual certainty.
- In reality, we can't hope for 100% certainty, although, in theory, some methods can distinguish with very low error rates.

Biometric Systems

Permanent:

- Ideally, the physical characteristic being measured should never change.
- In practice, it's sufficient if the characteristic remains stable over a reasonably long period of time.

Collectable:

- The physical characteristic should be easy to collect without any potential to cause harm to the subject.
- In practice, collectability often depends heavily on whether the subject is cooperative or not.

Biometric Systems

Reliable, robust, and user-friendly:

- These are just some of the additional real-world considerations for a practical biometric system.
- Some biometrics that have shown promise in laboratory conditions have subsequently failed to deliver similar performance in practice.

Biometric Systems

- There are two phases to a biometric system which are:
  1. Enrollment phase
  2. Recognition phase

Enrollment Phase:

- First, there is an enrollment phase, where subjects have their biometric information gathered and entered into a database.
- Typically, during this phase very careful measurement of the pertinent physical information is required.
- Since this is one-time work (per subject), it's acceptable if the process is slow and multiple measurements are required.

Biometric Systems

- In some fielded systems, enrollment has proven to be a weak point since it may be difficult to obtain results that are comparable to those obtained under laboratory conditions.

Recognition Phase:

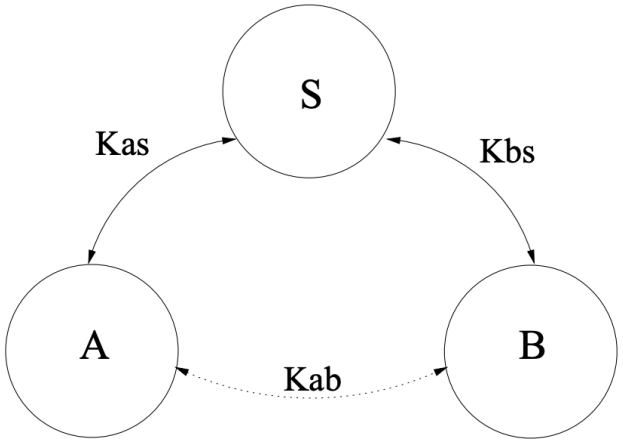
- This occurs when the biometric detection system is used in practice to determine whether (for the authentication problem) to authenticate the user or not.
- This phase must be quick, simple, and accurate.

The Needham-Schroeder Scheme

- Many existing protocols are derived from one proposed by Needham and Schroeder (1978), including the widely used Kerberos authentication protocol suite.
- N-S is a shared-key authentication protocol designed to generate and propagate a session key, i.e., a shared key for subsequent symmetrically encrypted communication.
- There are three principals: A and B, two principals desiring mutual communication, and S, a trusted key server.
- It is assumed that A and B already have secure symmetric communication with S using keys K_{AS} and K_{BS}, respectively.

The Needham-Schroeder Scheme

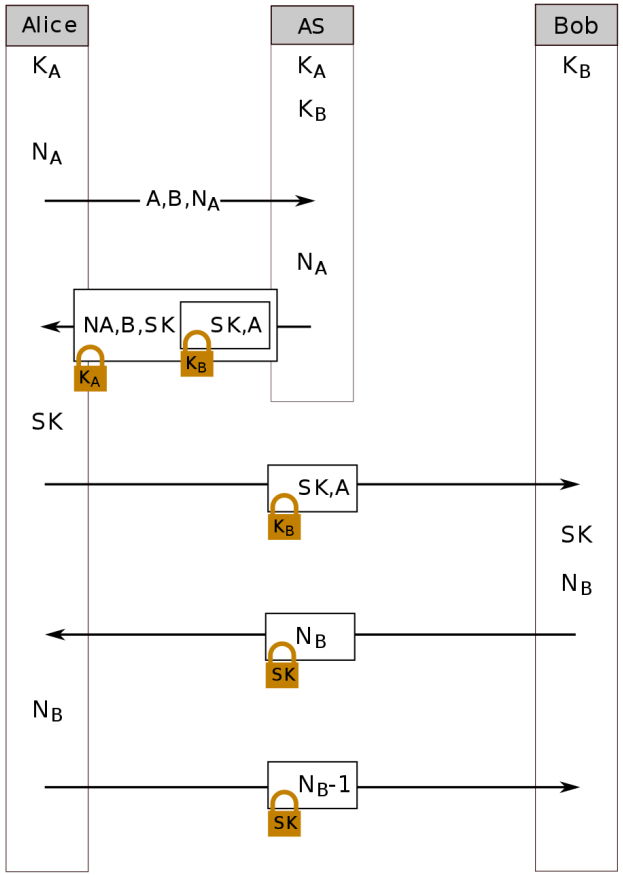
- N-S uses nonces (short for "numbers used once"), randomly generated values included in messages.
- If a nonce is generated and sent by A in one step and returned by B in a later step, A knows that B's message is fresh and not a replay from an earlier exchange.
- Note that a nonce is not a timestamp. The only assumption is that it has not been used in any earlier interchange, with high probability.



Needham-Schroeder Symmetric-Key Protocol

- Here, Alice (A) initiates the communication to Bob (B). S is a server trusted by both parties. In the communication:
  - A and B are identities of Alice and Bob respectively
  - K_{AS} is a symmetric key known only to A and S
  - K_{BS} is a symmetric key known only to B and S
  - N_A and N_B are nonces (number used once) generated by A and B respectively
  - K_{AB} is a symmetric, generated key, which will be the session key of the session between A and B.
- The protocol can be specified as follows in security protocol notation:

Needham-Schroeder Symmetric-Key Protocol



Symmetric Needham–Schroeder protocol scheme

Needham-Schroeder Symmetric-Key Protocol

1. $A \rightarrow S : A, B, N_A$

   Alice sends a message to the server identifying herself and Bob, telling the server she wants to communicate with Bob.

2. $S \rightarrow A : \{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$

   The server generates K_{AB} and sends back to Alice a copy encrypted under K_{BS} for Alice to forward to Bob and also a copy for Alice.

   Since Alice may be requesting keys for several different people, the nonce assures Alice that the message is fresh and that the server is replying to that particular message and the inclusion of Bob's name tells Alice who she is to share this key with.

3. $A \rightarrow B : \{K_{AB}, A\}_{K_{BS}}$

   Alice forwards the key to Bob who can decrypt it with the key he shares with the server, thus authenticating the data.

Needham-Schroeder Symmetric-Key Protocol

4. $B \rightarrow A : \{N_B\}_{K_{AB}}$
   - Bob sends Alice a nonce encrypted under K_{AB} to show that he has the key.
5. $A \rightarrow B : \{N_B - 1\}_{K_{AB}}$
   - Alice performs a simple operation on the nonce, re-encrypts it and sends it back verifying that she is still alive and that she holds the key.

Attacks on the protocol:

- The protocol is vulnerable to a **replay attack** (as identified by Denning and Sacco).
- If an attacker Oscar uses an older, compromised value for K_{AB}, he can then replay the message $\{K_{AB}, A\}_{K_{BS}}$ to Bob, who will accept it, being unable to tell that the key is not fresh.

Needham-Schroeder Symmetric-Key Protocol

- There are two ways in which Bob is deceived by this attack:
  1. The key K that is distributed in the session is not known to Bob's intended peer, Alice.
  2. The key K for the session is known to someone other than Bob's intended peer (namely, it is known to Oscar).

Fixing the Attack:

- This flaw can be fixed by the inclusion of a timestamp. (Denning 1981)
- This flaw can also be fixed by using an extra nonce.(Neumann 1993)

Needham-Schroeder Symmetric-Key Protocol

- Fixing the attack using an extra nonce:

$$A \rightarrow B : A$$

Alice sends to Bob a request.

$$B \rightarrow A : \{A, \mathbf{N'_B}\}_{K_{BS}}$$

Bob responds with a nonce encrypted under his key with the Server.

$$A \rightarrow S : A, B, N_A, \{A, \mathbf{N'_B}\}_{K_{BS}}$$

Alice sends a message to the server identifying herself and Bob, telling the server she wants to communicate with Bob.

$$S \rightarrow A : \{N_A, K_{AB}, B, \{K_{AB}, A, \mathbf{N'_B}\}_{K_{BS}}\}_{K_{AS}}$$

Note the inclusion of the nonce.

- The protocol then continues as described through the final three steps as described in the original protocol above. Note that N'_B is a different nonce from N_B.
- The inclusion of this new nonce prevents the replaying of a compromised version of $\{K_{AB},A\}_{K_{BS}}$ since such a message would need to be of the form $\{K_{AB},A,N'_B\}_{K_{BS}}$ which the attacker can't forge since s/he does not have K_{BS}.

Needham-Schroeder Public-Key Protocol

- This assumes the use of a public-key encryption algorithm.
- Here, Alice (A) and Bob (B) use a trusted server (S) to distribute public keys on request. These keys are:
  1. KP_A and K_{SA}, respectively public and private halves of an encryption key- pair belonging to A (S stands for "secret key" here)
  2. K_{PB} and K_{SB}, similar belonging to B
  3. K_{PS} and K_{SS}, similar belonging to S. (Note that this key-pair will be used for digital signatures, i.e., K_{SS} used for signing the message (encrypt) and K_{PS} used for verification (decrypt)).
- The protocol runs as follows:

1. A requests B's public keys from S.

$$A \rightarrow S : A, B$$

Needham-Schroeder Public-Key Protocol

2. S responds with public key K_{PB} alongside B's identity, signed by the server for authentication purposes.

$$S \rightarrow A : \{K_{PB}, B\}_{K_{SS}}$$

3. A chooses a random number N_A and sends it to B.

$$A \rightarrow B : \{N_A, A\}_{K_{PB}}$$

4. B now knows A wants to communicate, so B requests A's public keys.

$$B \rightarrow S : B, A$$

5. Server with public key K_{PA} alongside A's identity, signed by the server for authentication purposes.

$$S \rightarrow B : \{K_{PA}, A\}_{K_{SS}}$$

Needham-Schroeder Public-Key Protocol

6. B chooses a random number N_B and sends it to A along with N_A to prove ability to decrypt with K_{SB}.

$$B \rightarrow A : \{N_A, N_B\}_{K_{PA}}$$

7. A confirms N_B to B, to prove ability to decrypt with K_{SA}.

$$A \rightarrow B : \{N_B\}_{K_{PB}}$$

- At the end of the protocol, A and B know each other's identities, and know both N_A and N_B. These nonces are not known to eavesdroppers.

Needham-Schroeder Public-Key Protocol

Attack on the protocol:

- This protocol is vulnerable to the man-in-the-middle attack.
- If an attacker I can persuade A, to initiate a session with them, they can relay the messages to B and convince B that he is communicating with A.
- Ignoring the traffic to and from S, which is unchanged, the attack runs as follows:

1. A sends N_A to I who decrypts the message using K_{SI}.

$$A \rightarrow I : \{N_A, A\}_{K_{PI}}$$

2. I relays the message to B, pretending that A is communicating.

$$I \rightarrow B : \{N_A, A\}_{K_{PB}}$$

Needham-Schroeder Public-Key Protocol

3. B sends N_B.
   $$B \rightarrow I : \{N_A, N_B\}_{K_{PA}}$$
4. I relays it to A.
   $$I \rightarrow A : \{N_A, N_B\}_{K_{PA}}$$
5. A decrypts N_B and confirms it to I, who learns it.
   $$A \rightarrow I : \{N_B\}_{K_{PI}}$$
6. I re-encrypts N_B and convinces B that she has decrypted it.
   $$I \rightarrow B : \{N_B\}_{K_{PB}}$$

- At the end of the attack, B falsely believes A is communicating with him, and that N_A and N_B are only known to A and B.
- The attack was first described in a 1995 paper by Gavin Lowe.

Needham-Schroeder Public-Key Protocol

- The paper also describes a fixed version of the scheme, referred to as the Needham–Schroeder–Lowe protocol.
- The fix involves the modification of message six to include the responder's identity, that is we replace:

$$B \rightarrow A : \{N_A, N_B\}_{K_{PA}}$$

with the fixed version:

$$B \rightarrow A : \{N_A, N_B, B\}_{K_{PA}}$$

- The intruder cannot successfully replay the message because A is expecting a message containing the identity of I whereas the message will have identity of B.

Kerberos Protocol

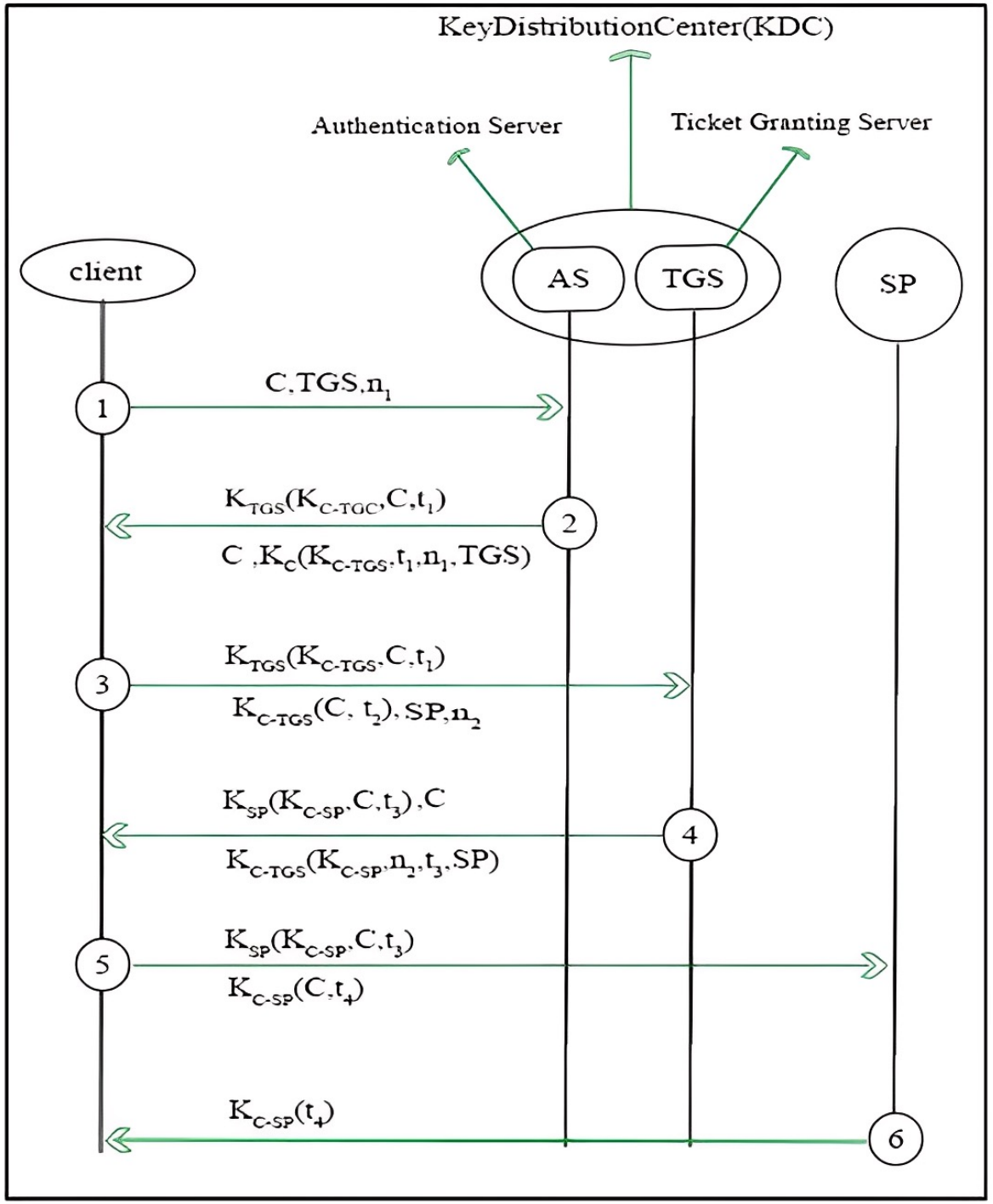
- Kerberos is an authentication protocol used by processes/hosts communicating over an insecure network to verify each other's identity in a secure manner.
- It is based on the idea that a central server provides authenticated tokens called "tickets" to requesting applications.
- A ticket is an unforgeable, non-replayable, authenticated object.
- The security of the protocol depends on the assumption that the participating machines maintain loosely synchronized time.
- The entities involved in Kerberos are:
  1. Key Distribution Center (KDC)
     - Authentication Server (AS)
     - Ticket Granting Server (TGS)
  2. Service Server (SS) or Service Provider (SP)
  3. Ticket Granting Ticket (TGT)

Kerberos Protocol

- The core of the Kerberos is the Key Distribution Center (KDC), a trusted third party which contains two logical parts, i.e. an Authentication Server (AS) and a Ticket Granting Server (TGS).
- AS maintains a database to store the credentials of users and issues users with Ticket-Granting Ticket (TGT) upon successful authentication.
- TGS is used to issue users with Client-to-Server ticket upon receiving valid TGT.
- Kerberos also enables mutual authentication (i.e. both the user and the server verify each other's identity).
- A client authenticates itself to the AS once and obtains a ticket that can be used to obtain additional tickets from the SS without requiring the client to re-authenticate itself for every service requested.

Kerberos Protocol Steps

- Alongside figure is a simplifed description of the steps the client follow to get service from a service provider using Kerberos 5 Authentication Protocol.
- Step 1: Client sends his name (C), and the name of Ticket Granting Server (TGS) along with a Nonce (n_1) to AS.



Kerberos Protocol Steps

Step 2: AS checks the client's identity and if exist a user with this identity and access-rights, it will send these messages to C:

$$C, K_C (K_{C\text{-}TGS}, n_1, t_1, TGS), K_{TGS} (K_{C\text{-}TGS}, C, t_1)$$

- Where $K_{TGS} (K_{C\text{-}TGS}, C, t_1)$ is the ticket for the client to access the TGS and it contains the session key between client and TGS. This ticket is encrypted by a secret key K_{TGS}, known only to the TGS and the AS, which prevents alteration.
- AS uses the message $K_C (K_{C\text{-}TGS}, n_1, t_1, TGS)$ to send the $K_{C\text{-}TGS}$ to the client. This message is encrypted by using the key K_C which is derived from client's password.
- n_1 is a random number that indicates message freshness and prevents replay
- attacks.
- t_1 is a timestamp that is used to inform client of the times these messages were issued.

Kerberos Protocol Steps

Step 3: After obtaining the session key ($K_{C\text{-}TGS}$), Client sends these messages to TGS:

$$n_2, SP, K_{C\text{-}TGS}(C, t_2), K_{TGS}(K_{C\text{-}TGS}, C, t_1)$$

- Where SP is the name of the service provider and $K_{TGS}(K_{C\text{-}TGS}, C, t_1)$ is the ticket that was issued in the previous step.
- $K_{C\text{-}TGS}(C, t_2)$ is an authenticator to validate the client and ticket to TGS.

Kerberos Protocol Steps

Step 4: TGS decrypts the ticket and authenticator, verifies the request, and then sends these messages to the client:

$$C\ ,\ K_{SP}(K_{C\text{-}SP}\ ,\ C,\ t_3)\ ,\ K_{C\text{-}TGS}\ (K_{C\text{-}SP}\ ,\ n_2,\ t_3,\ SP)$$

- Similar to the second step, $K_{SP}(K_{C\text{-}SP}\ ,\ C,\ t_3)$ is a ticket for client access to the service provider and contains the session key between client and SP.
- This ticket is encrypted using K_{SP}.
- The TGS also sends message $K_{C\text{-}TGS}\ (K_{C\text{-}SP}\ ,\ n_2,\ t_3,\ SP)$ to the client that contains $K_{C\text{-}SP}$ and is encrypted using $K_{C\text{-}TGS}$.

Kerberos Protocol Steps

Step 5: User obtains the session key $K_{C\text{-}SP}$ and sends the ticket $K_{SP}(K_{C\text{-}SP}, C, t_3)$ alongside an authenticator $K_{C\text{-}SP}(C, t_4)$ to the service provider.

Step 6: SP checks the validity of the ticket and authenticator, then if everything is correct, SP will grant service access to the client, and if mutual authentication is required, SP will send the authenticator $K_{C\text{-}SP}(t_4)$ to the client.

Kerberos Advantages

- A user's password is not sent on the wire (either in plaintext or ciphertext) during session initiation.
- Kerberos provides cryptographic protection against spoofing.

  Each service access request is mediated by the TGS, which knows that the identity of the user/client is authenticated by the Kerberos AS and processes the user/client request encrypted with the Client/TGS session key.
- As each ticket has a limited validity period, long-term cryptanalytic attacks cannot be launched.
- A host responds back only if the request messages have timestamp value close to the current time at the host.
- Kerberos provides mutual authentication.

  The TGS and SS can respectively get access to the Client/TGS session key and the Client/Server session key only after they can decrypt the messages containing these keys with their appropriate secret keys. The client uses this approach to indirectly authenticate the servers.

Kerberos Weakness

- Single point of failure as Kerberos requires continuous availability of a trusted ticket-granting server for all access control and authentication checks.
- Authenticity of servers requires a trusted relationship between the TGS and every service server.
- Timely transactions are required to reduce chances of a user with genuine ticket being denied service. Hence, the clocks of the involved hosts must be synchronized within configured limits.
- Password guessing could still work to get the valid secret key for a user. The whole system is still dependent on the user password.
- Network services cannot be accessed without obtaining Kerberos authentication. All applications run by the users in the network need to go through Kerberos authentication.