Kerberos | Kerberos Terminology | Kerberos Working | Characteristics of Kerberos

Kerberos Protocol

What is Kerberos?

Kerberos: Kerberos is a network authentication protocol that works on the basis of tickets to allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner.

In Greek mythology, a many headed dog, the guardian of the entrance of Hades

What do the three heads of Kerberos represent?

Kerberos is a three-step security process used for authorization and authentication. The three-heads of Kerberos are:

1-User,

2-KDC-Key Distribution Service (security server) and

3-Services (servers).

Kerberos is a standard feature of Windows software.

Why Kerberos?

Kerberos is an authentication protocol that is used to verify the identity of a user or host. The authentication is based on tickets used as credentials, allowing communication and proving identity in a secure manner even over a non-secure network.

Characteristics of Kerberos

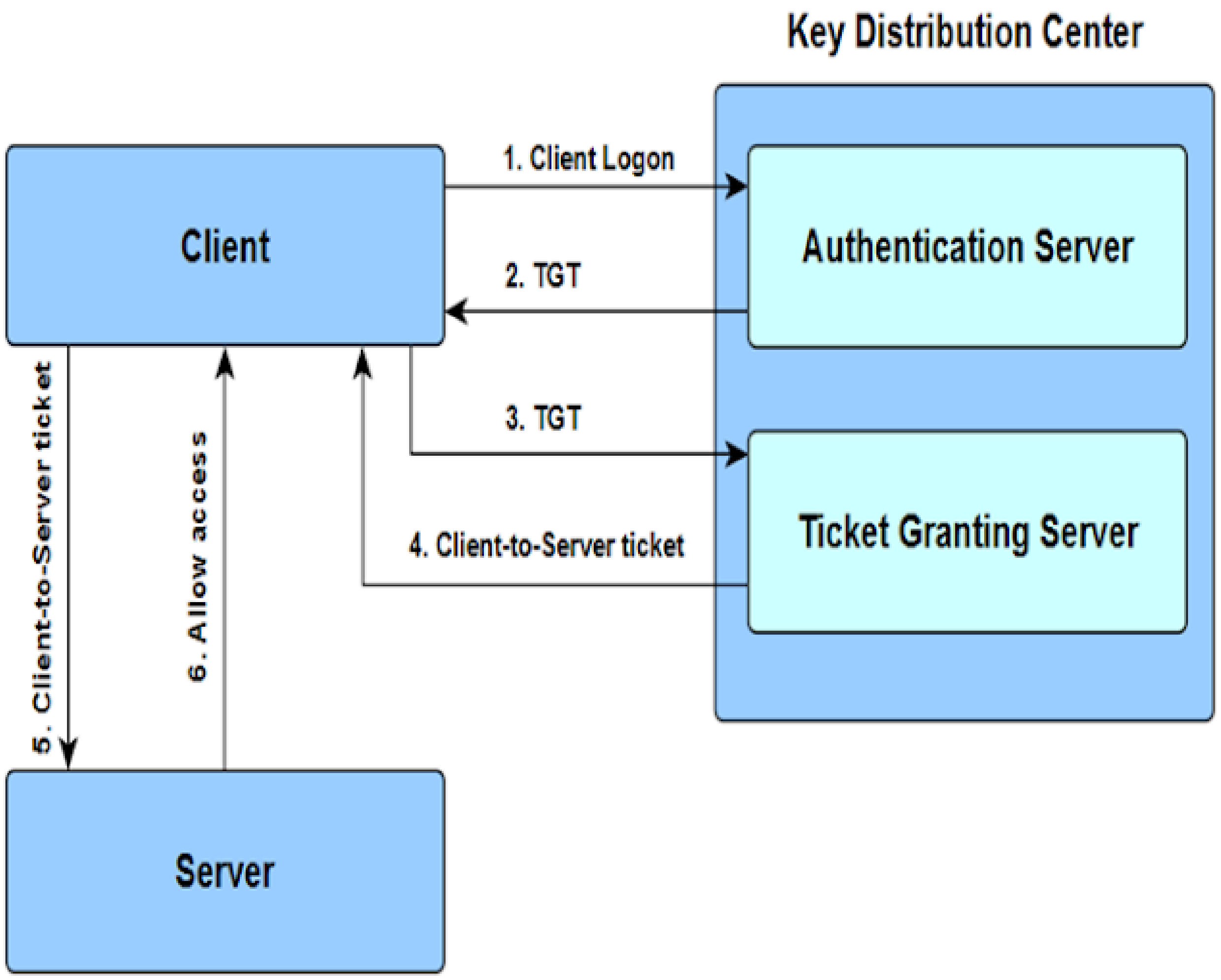
Secure: Kerberos should be strong enough that a potential opponent does not find it to be the weak link.

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



Authentication Server (AS): A server that issues tickets for a desired service which are in turn given to users for access to the service.

Client: An entity on the network that can receive a ticket from Kerberos.

Credentials: A temporary set of electronic credentials that verify the identity of a client for a particular service. It also called a ticket.

Credential cache or ticket file: A file which contains the keys for encrypting communications between a user and various network services.

Crypt hash: A one-way hash used to authenticate users.

Key: Data used when encrypting or decrypting other data.

Key distribution centre (KDC): A service that issue Kerberos tickets and which usually run on the same host as the ticket-granting server (TGS).

Realm: A network that uses Kerberos composed of one or more servers called KDCs and a potentially large number of clients.

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.

Ticket-granting ticket (TGT): A special ticket that allows the client to obtain additional tickets without applying for them from the KDC.

Working of Kerberos

Step 1: (Fig 1)

The AS, receives the request by the client and verifies that the client.

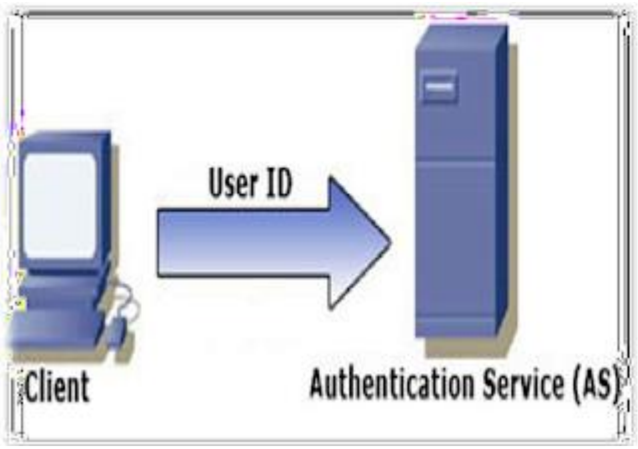


Figure : Authentication Service verifies the User ID

Step 2:

Upon verification, a timestamp is created with current time in a user session with expiration date. The timestamp ensures that when 8 hours is up, the encryption key is useless.

Step 3: (Fig 2)

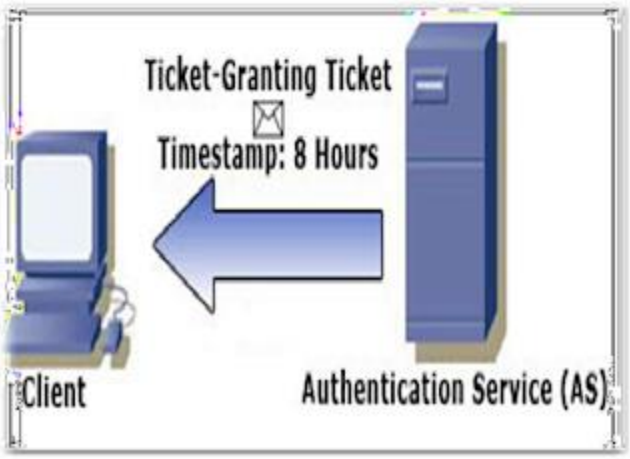
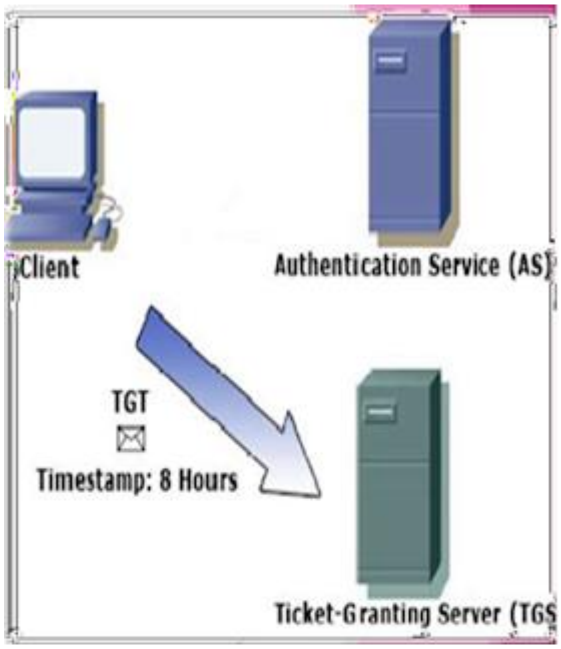


Figure : Authentication Service issues TGT

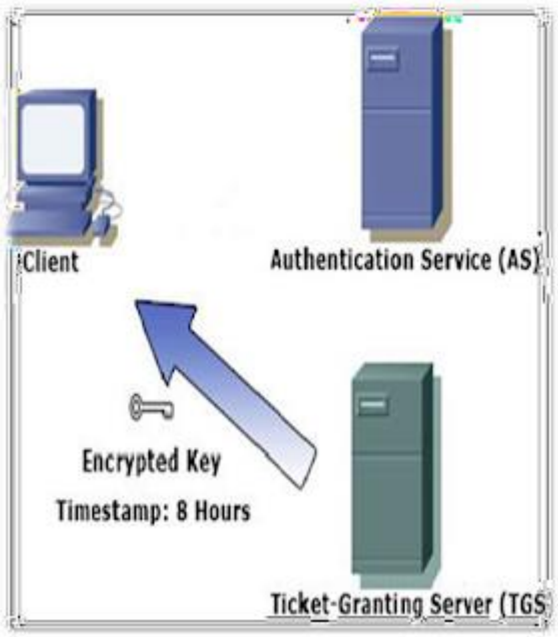
The key is sent back to the client in the form of a TGT.

Step 4: (Fig 3)



The client submits the TGT to the TGS, to get authenticated.

Step 5: (Fig. 4)



Step 6:

The client decrypts the ticket & send ACK to TGS.

Step 7 (Fig. 5)

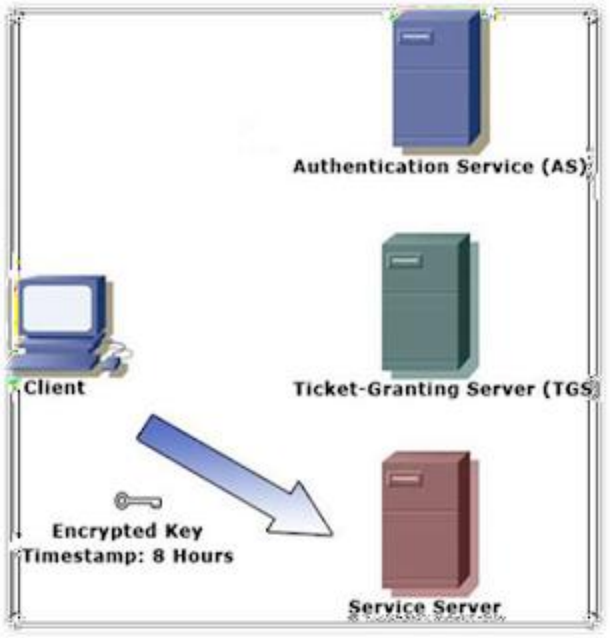


Figure : Service server decrypt key and check the time stamp

Client sends its own encrypted key to the service server.

The server decrypts the key and check timestamp is still valid or not.

Step 8: (Fig. 6)

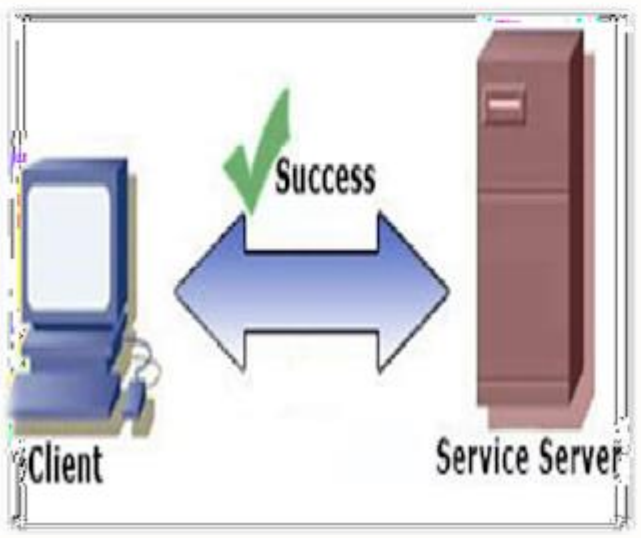


Figure : For secret keys communication initiated

The client decrypts the ticket. If the keys are still valid, communication is initiated between client and server. Now the client is authenticated until the session expires.

Is Kerberos symmetric or asymmetric?

Kerberos is capable of both symmetric and asymmetric cryptography.

Is Kerberos safe?

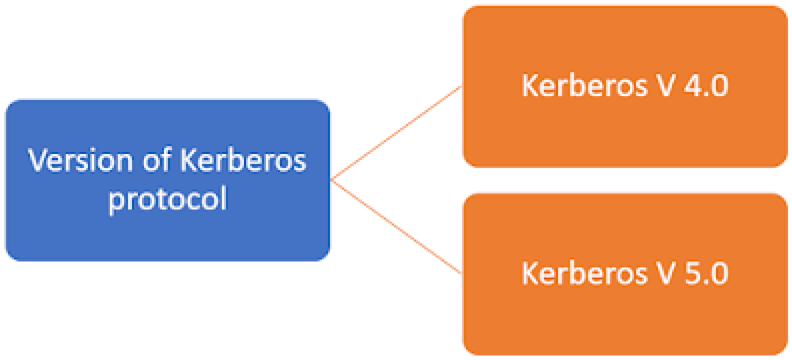
Kerberos is more secure than other authentication methods because it does not send plain text pass- words over the network and instead of password uses encrypted tickets.

Kerberos Version 4 | Kerberos version 4 using authentication and ticket granting server

What is Kerberos?

Kerberos: Kerberos is a network authentication protocol that works on the basis of tickets to allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner.

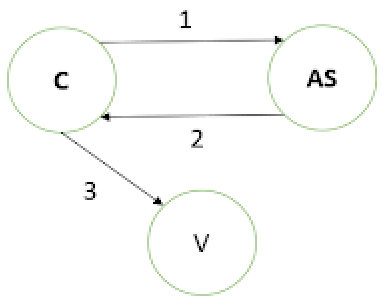
Different Version of Kerberos Protocols



Using Authentication Server (AS)

$$
\begin{aligned}
&\textbf{(1)}\ \ \mathrm{C} \rightarrow \mathrm{AS}: && ID_C \| P_C \| ID_V \\
&\textbf{(2)}\ \ \mathrm{AS} \rightarrow \mathrm{C}: && Ticket \\
&\textbf{(3)}\ \ \mathrm{C} \rightarrow \mathrm{V}: && ID_C \| Ticket
\end{aligned}
$$

$$Ticket = E(K_v, [ID_C \| AD_C \| ID_V])$$



where

C	= client	ID_V	= identifier of V
AS	= authentication server	P_C	= password of user on C
V	= server	AD_C	= network address of C
ID_C	= identifier of user on C	K_v	= secret encryption key shared by AS and V

Step – 1: In this scenario, the user logs on to a workstation and requests access to server V. The client module C in the user's workstation requests the user's password and then sends a message to the AS that includes the user's ID, the server's ID, and the user's password. The AS checks its database to see if the user has supplied the proper password for this user ID and whether this user is permitted access to server V. If both tests are passed, the AS accepts the user as authentic and must now convince the server that this user is authentic.

Step – 2: To do so, the AS creates a ticket that contains the user's ID and network address and the server's ID. This ticket is encrypted using the secret key shared by the AS and this server. This ticket is then sent back to C. Because the ticket is encrypted, it cannot be altered by C or by an opponent.

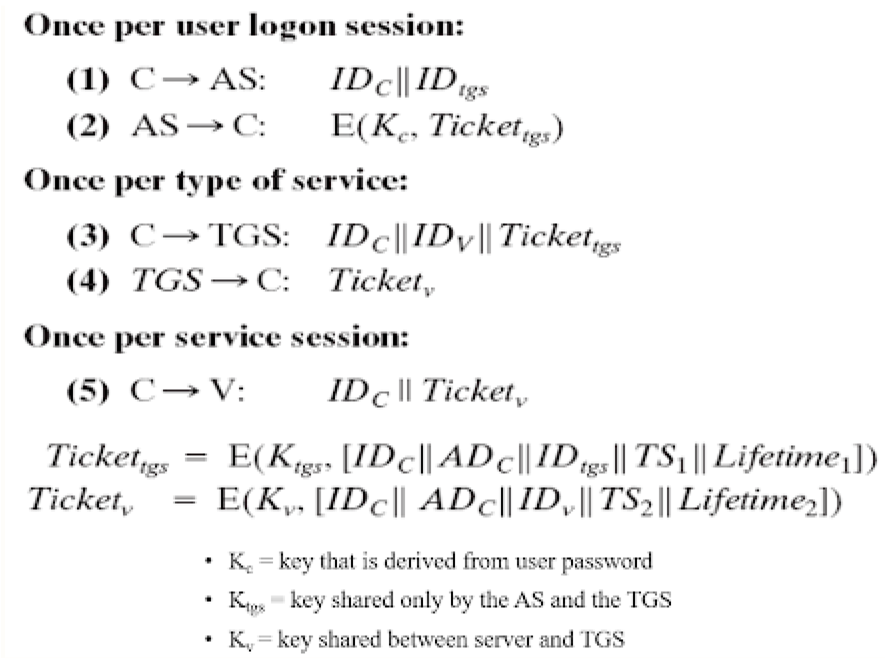
Step – 3: With this ticket, C can now apply to V (Server) for service. C sends a message to V containing C's ID and the ticket. V decrypts the ticket and verifies that the user ID in the ticket is the same as the unencrypted user ID in the message. If these two matches, the server considers the user authenticated and grants the requested service.

Problems:

Problem – 1: Under this scheme, a user would need a new ticket for every different service. If a user wants to access a print server, a mail server, a file server, and so on, the first instance of each access would require a new ticket.

Problem – 2: In this scheme, password is transmitted without encryption. An eavesdropper could capture the password and use any service accessible to the victim.

Using Ticket Granting Server (TGS)



Once per user logon session:

(1) $C \rightarrow AS$: $ID_C \| ID_{tgs}$

(2) $AS \rightarrow C$: $E(K_c, Ticket_{tgs})$

Once per type of service:

(3) $C \rightarrow TGS$: $ID_C \| ID_V \| Ticket_{tgs}$

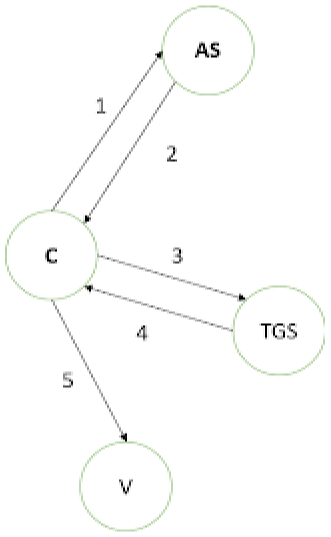
(4) $TGS \rightarrow C$: $Ticket_v$

Once per service session:

(5) $C \rightarrow V$: $ID_C \| Ticket_v$

$$Ticket_{tgs} = E(K_{tgs}, [ID_C \| AD_C \| ID_{tgs} \| TS_1 \| Lifetime_1])$$
$$Ticket_v = E(K_v, [ID_C \| AD_C \| ID_v \| TS_2 \| Lifetime_2])$$

- K_c = key that is derived from user password
- K_{tgs} = key shared only by the AS and the TGS
- K_v = key shared between server and TGS



Step – 1: The client requests a ticket-granting ticket on behalf of the user by sending its user's ID to the AS, together with the TGS ID, indicating a request to use the TGS service.

Step – 2: The AS responds with a ticket that is encrypted with a key that is derived from the user's password (KC), which is already stored at the AS. When this response arrives at the client, the client prompts the user for his or her password, generates the key, and attempts to decrypt the incoming message. If the correct password is supplied, the ticket is successfully recovered. Thus, we have used the password to obtain credentials from Kerberos without having to transmit the password in plaintext. Here, the opponent may be able to reuse the ticket to spoof the TGS. To counter this, the ticket includes a timestamp, indicating the date and time at which the ticket was issued, and a lifetime, indicating the length of time for which the ticket is valid.

Step – 3: The client requests a service-granting ticket on behalf of the user. For this purpose, the client transmits a message to the TGS containing the user's ID, the ID of the desired service, and the ticket-granting ticket.

Step – 4: The TGS decrypts the incoming ticket using Ktgs and verifies the success of the decryption by the presence of its ID. It checks to make sure that the lifetime has not expired. Then it compares the user ID and network address with the incoming information to authenticate the user. If the user is permitted access to the server V, the TGS issues a ticket to grant access to the requested service.

Step – 5: The client requests access to a service on behalf of the user. For this purpose, the client transmits a message to the server containing the user's ID and the service-granting ticket. The server authenticates by using the contents of the ticket.

Problems

Problem – 1: A network service (the TGS or an application service) must be able to prove that the person using a ticket is the same person to whom that ticket was issued.

Problem – 2: There may be a requirement for servers to authenticate themselves to users. Without such authentication, the false server would then be in a position to act as a real server and capture any information from the user and deny the true service to the user.

Solution

AS provides both the client and the TGS with a secret piece of information in a secure manner. Then the client can prove its identity to the TGS by revealing the secret information—again in a secure manner. An efficient way of accomplishing this is to use an encryption key as the secure information; this is referred to as a session key in Kerberos.

Kerberos Version 4 Message Exchange

Kerberos Version 4 Message Exchange Scenario

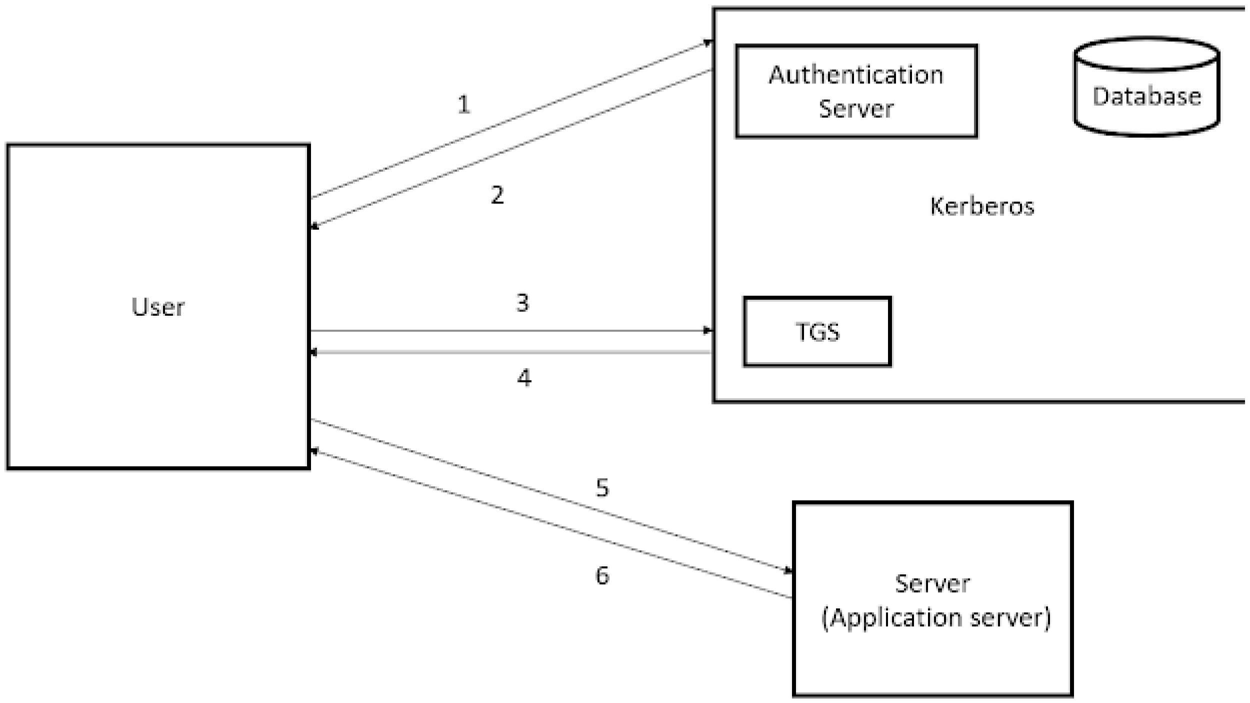
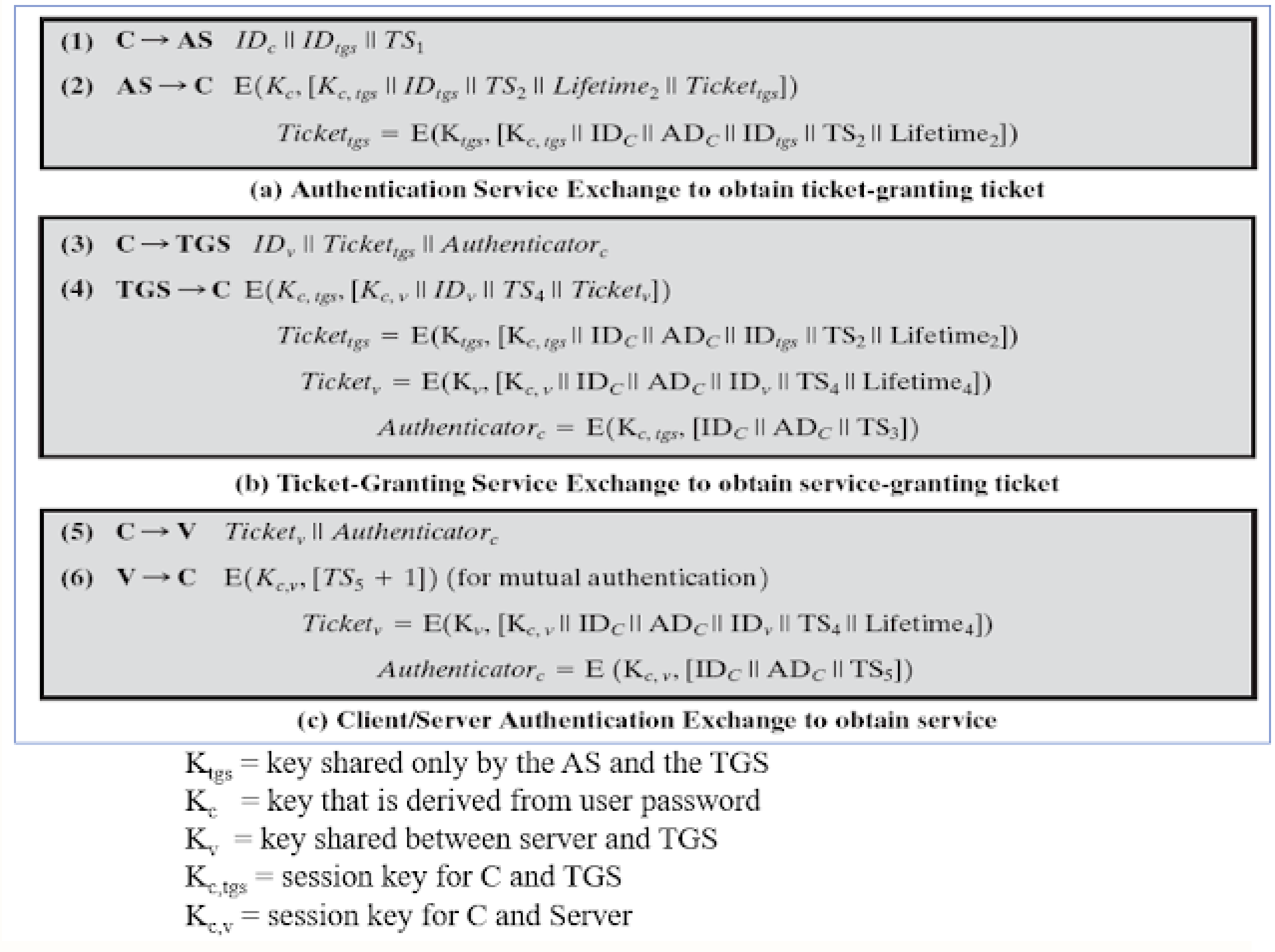


Figure : Kerberos Version 4 Message Exchange Scenario



(1) $\mathbf{C \rightarrow AS}$ $ID_c \,||\, ID_{tgs} \,||\, TS_1$

(2) $\mathbf{AS \rightarrow C}$ $E(K_c, [K_{c,tgs} \,||\, ID_{tgs} \,||\, TS_2 \,||\, Lifetime_2 \,||\, Ticket_{tgs}])$

$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \,||\, ID_C \,||\, AD_C \,||\, ID_{tgs} \,||\, TS_2 \,||\, Lifetime_2])$

(a) Authentication Service Exchange to obtain ticket-granting ticket

(3) $\mathbf{C \rightarrow TGS}$ $ID_v \,||\, Ticket_{tgs} \,||\, Authenticator_c$

(4) $\mathbf{TGS \rightarrow C}$ $E(K_{c,tgs}, [K_{c,v} \,||\, ID_v \,||\, TS_4 \,||\, Ticket_v])$

$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \,||\, ID_C \,||\, AD_C \,||\, ID_{tgs} \,||\, TS_2 \,||\, Lifetime_2])$

$Ticket_v = E(K_v, [K_{c,v} \,||\, ID_C \,||\, AD_C \,||\, ID_v \,||\, TS_4 \,||\, Lifetime_4])$

$Authenticator_c = E(K_{c,tgs}, [ID_C \,||\, AD_C \,||\, TS_3])$

(b) Ticket-Granting Service Exchange to obtain service-granting ticket

(5) $\mathbf{C \rightarrow V}$ $Ticket_v \,||\, Authenticator_c$

(6) $\mathbf{V \rightarrow C}$ $E(K_{c,v}, [TS_5 + 1])$ (for mutual authentication)

$Ticket_v = E(K_v, [K_{c,v} \,||\, ID_C \,||\, AD_C \,||\, ID_v \,||\, TS_4 \,||\, Lifetime_4])$

$Authenticator_c = E\,(K_{c,v}, [ID_C \,||\, AD_C \,||\, TS_5])$

(c) Client/Server Authentication Exchange to obtain service

K_{tgs} = key shared only by the AS and the TGS
K_c = key that is derived from user password
K_v = key shared between server and TGS
$K_{c,tgs}$ = session key for C and TGS
$K_{c,v}$ = session key for C and Server

Step – 1: The client sends a message to the AS requesting access to the TGS. It includes a timestamp, so that the AS knows that the message is timely.

Step – 2: The AS responds with a message, encrypted with a key derived from the user's password (K_C), that contains the ticket. The encrypted message also contains a copy of the session key, $K_{C,\,tgs}$, where the subscripts indicate that this is a session key for C and TGS. Because this session key is inside the message encrypted with K_C, only the user's client can read it. The same session key is included in the ticket, which can be read only by the TGS. Thus, the session key has been securely delivered to both C and the TGS.

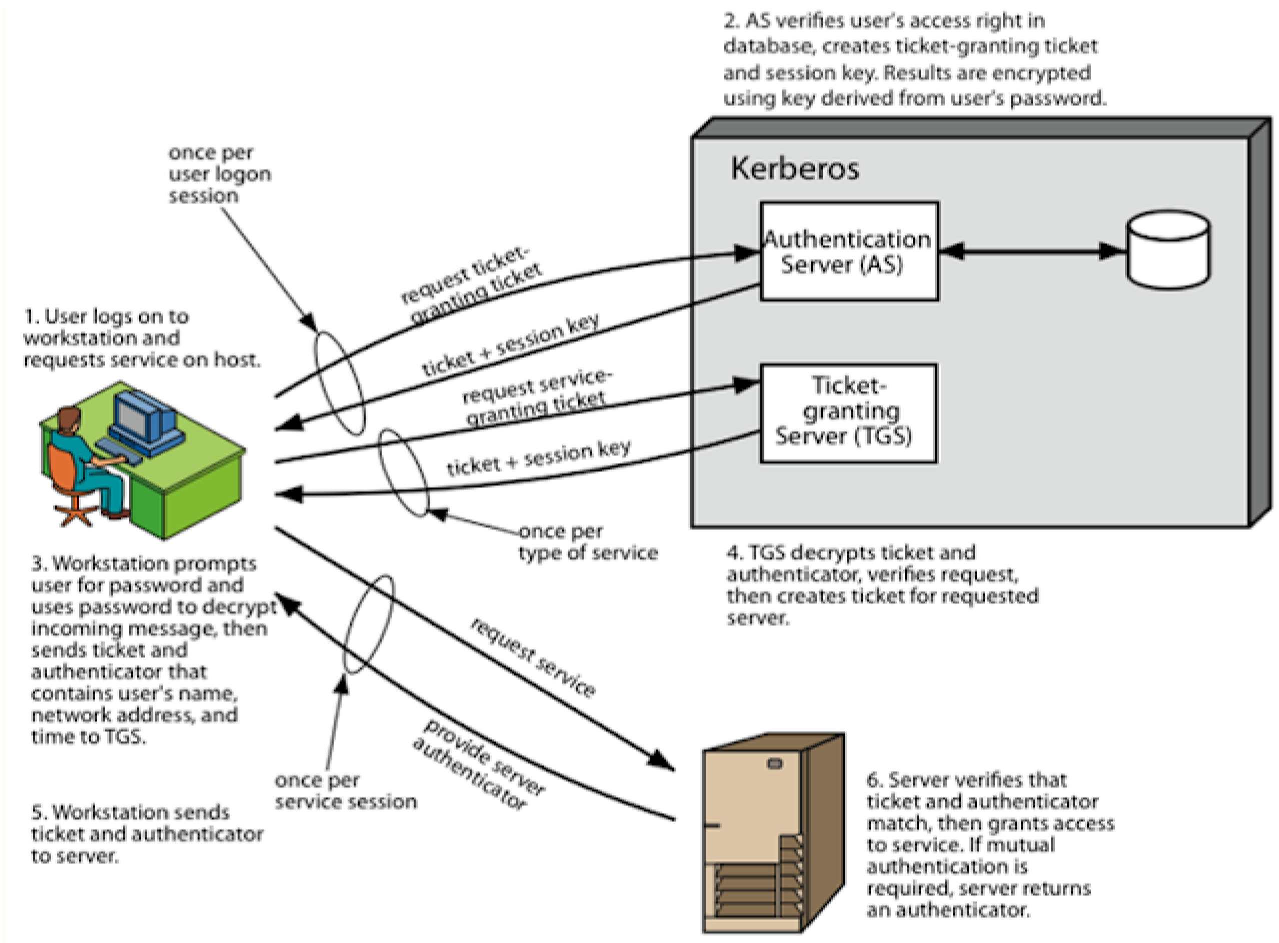
Step – 3: C sends TGS a message that includes the ticket plus the ID of the requested service. In addition, C transmits an authenticator, which includes the ID and address of C's user and a timestamp. The TGS uses the session key to decrypt the authenticator. The TGS can then check the name and address from the authenticator with that of the ticket and with the network address of the incoming message. If all match, then the TGS is assured that the sender of the ticket is indeed the ticket's real owner.

Step – 4: Reply message from TGS is encrypted with $K_{C,\ tgs}$ and includes a session key to be shared between C and the server V, the ID of V, and the timestamp of the ticket. The ticket itself includes the same session key.

Step – 5: When C sends ticket and an authenticator. The server can decrypt the ticket, recover the session key, and decrypt the authenticator.

Step – 6: The server returns the value of the timestamp from the authenticator, incremented by 1, and encrypted in the session key. C can decrypt this message to recover the incremented timestamp. Because the message was encrypted by the session key, C is assured that it could have been created only by V. The contents of the message assure C that this is not a replay of an old reply.

Summery of Kerberos version 4 message exchange scenario


2. 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.
Kerberos
Authentication Server (AS)
Ticket-granting Server (TGS)
once per user logon session
request ticket-granting ticket
ticket + session key
request service-granting ticket
ticket + session key
once per type of service
1. User logs on to workstation and requests service on host.
3. Workstation prompts user for password and uses password to decrypt incoming message, then sends ticket and authenticator that contains user's name, network address, and time to TGS.
4. TGS decrypts ticket and authenticator, verifies request, then creates ticket for requested server.
request service
provide server authenticator
once per service session
5. Workstation sends ticket and authenticator to server.
6. Server verifies that ticket and authenticator match, then grants access to service. If mutual authentication is required, server returns an authenticator.

Kerberos Realm | Inter-realm Authentication

What is Kerberos Realm

A full-service Kerberos environment consists of a Kerberos server, a number of clients, all are registered with Kerberos server, a number of application servers, all are sharing keys with Kerberos server. Such an environment is referred to as a ***Kerberos realm***.

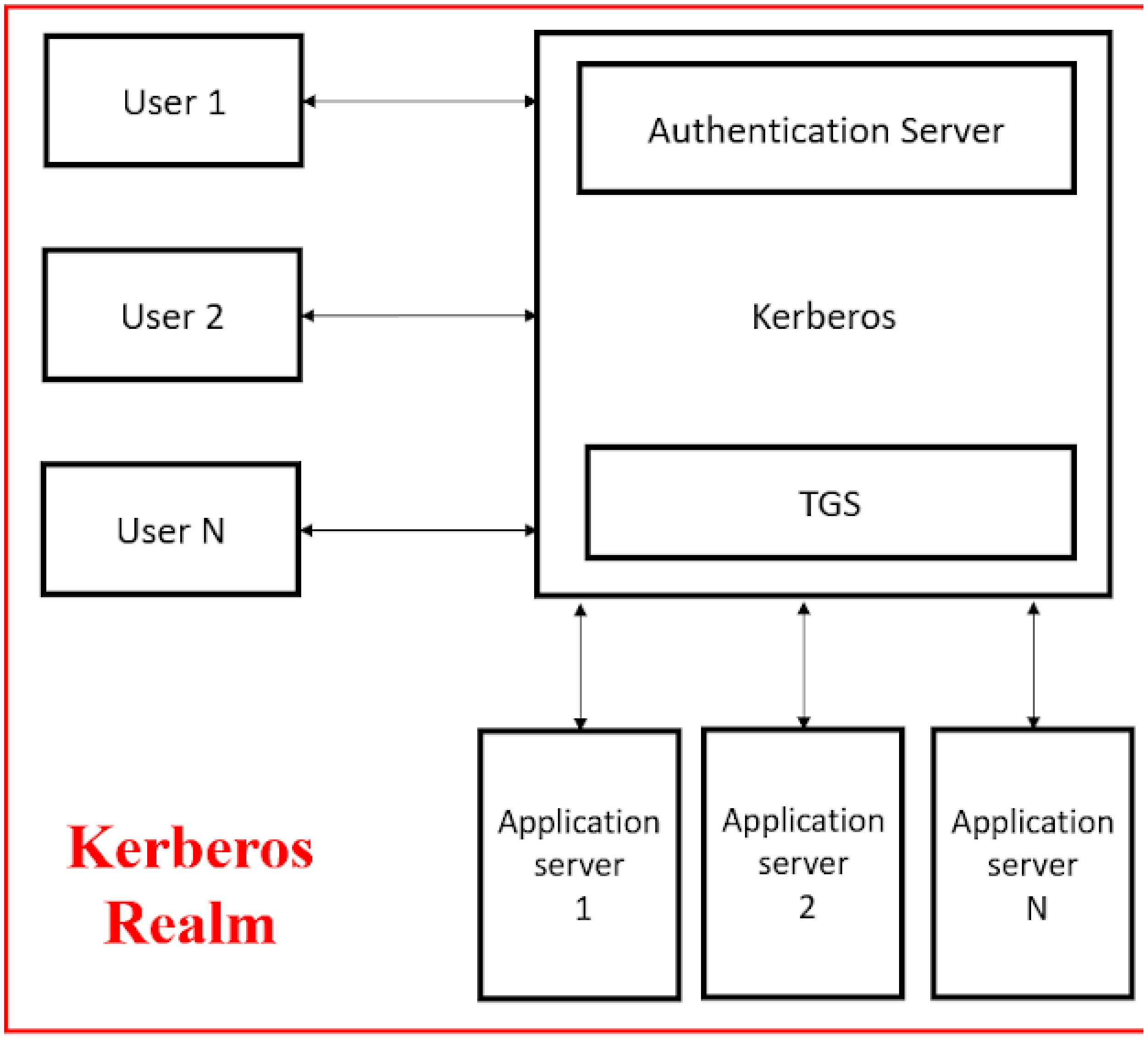


Figure: Kerberos Realm

Inter Realm Authentication

(1) $\mathrm{C} \rightarrow \mathrm{AS}$: $ID_c \,\|\, ID_{tgs} \,\|\, TS_1$

(2) $\mathrm{AS} \rightarrow \mathrm{C}$: $\mathrm{E}(K_c, [K_{c,\,tgs} \,\|\, ID_{tgs} \,\|\, TS_2 \,\|\, Lifetime_2 \,\|\, Ticket_{tgs}])$

(3) $\mathrm{C} \rightarrow \mathrm{TGS}$: $ID_{tgsrem} \,\|\, Ticket_{tgs} \,\|\, Authenticator_c$

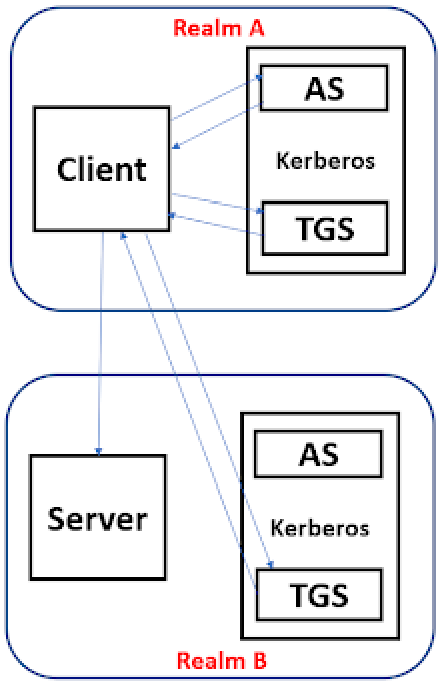
(4) $\mathrm{TGS} \rightarrow \mathrm{C}$: $\mathrm{E}(K_{c,tgs}, [K_{c,\,tgsrem} \,\|\, ID_{tgsrem} \,\|\, TS_4 \,\|\, Ticket_{tgsrem}])$

(5) $\mathrm{C} \rightarrow \mathrm{TGS}_{\mathrm{rem}}$: $ID_{vrem} \,\|\, Ticket_{tgsrem} \,\|\, Authenticator_c$

(6) $\mathrm{TGS}_{\mathrm{rem}} \rightarrow \mathrm{C}$: $\mathrm{E}(K_{c,tgsrem}, [K_{c,\,vrem} \,\|\, ID_{vrem} \,\|\, TS_6 \,\|\, Ticket_{vrem}])$

(7) $\mathrm{C} \rightarrow \mathrm{V}_{\mathrm{rem}}$: $Ticket_{vrem} \,\|\, Authenticator_c$

1. C → AS : Request ticket for local TGS
2. AS → C : Ticket for local TGS
3. C → TGS : Request ticket for remote TGS
4. TGS → C : Ticket for remote TGS
5. C → $\mathrm{TGS}_{\mathrm{rem}}$: Request ticket for remote Server
6. $\mathrm{TGS}_{\mathrm{rem}}$ → C : Ticket for remote Server
7. C → $\mathrm{V}_{\mathrm{rem}}$: Request for remote service


Realm A
Client
AS
Kerberos
TGS
Server
AS
Kerberos
TGS
Realm B