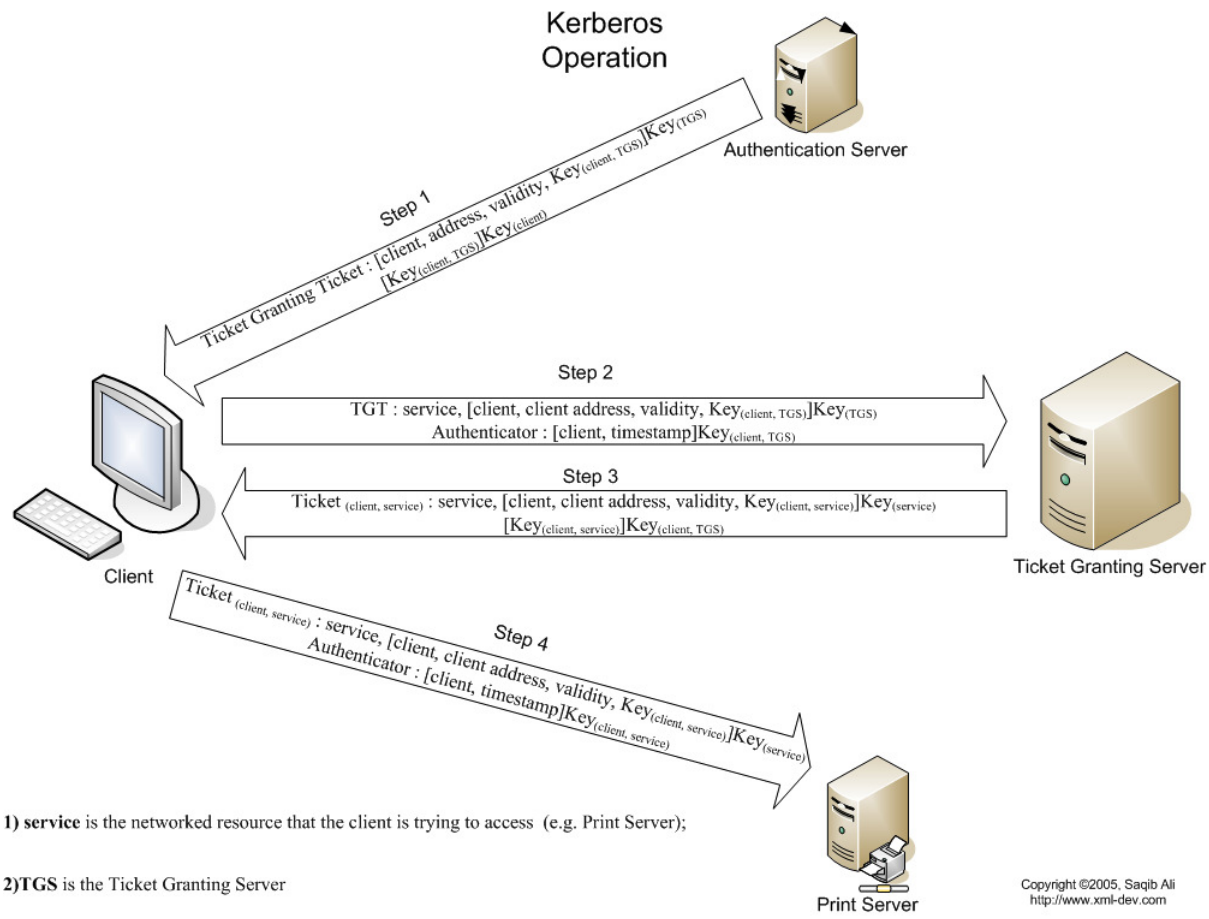
Kerberos



1) service is the networked resource that the client is trying to access (e.g. Print Server);

2)TGS is the Ticket Granting Server

Step 1) Authentication Server to Client
Ticket Granting Ticket : [client, address, validity, $Key_{(client, TGS)}$]$Key_{(TGS)}$
[$Key_{(client, TGS)}$]$Key_{(client)}$

Step 2) Client to Ticket Granting Server
Ticket Granting Ticket : service, [client, client address, validity, $Key_{(client, TGS)}$]$Key_{(TGS)}$
Authenticator : [client, timestamp]$Key_{(client, TGS)}$

Step 3) Ticket Granting Server to Client
Ticket $_{(client, service)}$: service, [client, client address, validity, $Key_{(client, service)}$]$Key_{(service)}$
[$Key_{(client, service)}$]$Key_{(client, TGS)}$

Step 4) Client to Service
Ticket $_{(client, service)}$: service, [client, client address, validity, $Key_{(client, service)}$]$Key_{(service)}$
Authenticator : [client, timestamp]$Key_{(client, service)}$

What follows is a simplified description of the protocol. The following shortcuts will be used: AS = Authentication Server, TGS = Ticket Granting Server, SS = Service Server.

In one sentence: the client authenticates itself to AS, then demonstrates to the TGS that it's authorized to receive a ticket for a service (and receives it), then demonstrates to the SS that it has been approved to receive the service.

In more detail:

1. A user enters a username and password on the client.

2. The client performs a one-way hash on the entered password, and this becomes the secret key of the client.

3. The client sends a clear-text message to the AS requesting services on behalf of the user. Sample Message: "User XYZ would like to request services". Note: Neither the secret key nor the password is sent to the AS.

4. The AS checks to see if the client is in its database. If it is, the AS sends back the following two messages to the client:
* Message A: Client/TGS session key encrypted using the secret key of the user.
* Message B: Ticket-Granting Ticket (which includes the client ID, client network address, ticket validity period, and the client/TGS session key) encrypted using the secret key of the TGS.

5. Once the client receives messages A and B, it decrypts message A to obtain the client/TGS session key. This session key is used for further

communications with TGS. (Note: The client cannot decrypt the Message B, as it is encrypted using TGS's secret key.) At this point, the client has enough information to authenticate itself to the TGS.

6. When requesting services, the client sends the following two messages to the TGS:
* Message C: Composed of the Ticket-Granting Ticket from message B and the ID of the requested service.
* Message D: Authenticator (which is composed of the client ID and the timestamp), encrypted using the client/TGS session key.

7. Upon receiving messages C and D, the TGS decrypts message D (Authenticator) using the client/TGS session key and sends the following two messages to the client:
* Message E: Client-to-server ticket (which includes the client ID, client network address, validity period) encrypted using the service's secret key.
* Message F: Client/server session key encrypted with the client/TGS session key.

8. Upon receiving messages E and F from TGS, the client has enough information to authenticate itself to the SS. The client connects to the SS and sends the following two messages:
* Message G: the client-to-server ticket, encrypted using service's secret key.
* Message H: a new Authenticator, which includes the client ID, timestamp and is encrypted using client/server session key.

9. The server decrypts the ticket using its own secret key and sends the following message to the client to confirm its true identity and willingness to serve the client:
* Message I: the timestamp found in client's recent Authenticator plus 1, encrypted using the client/server session key.

10. The client decrypts the confirmation using its shared key with the server and checks whether the timestamp is correctly updated. If so, then the client can trust the server and can start issuing service requests to the server.

11. The server provides the requested services to the client.

And Now: Do you Code Sign ???