Kerberos Stepwise

Step 1 : Login

The **user** enters the username and password. In some cases you only have to enter the password in **step 5**. The client will then transform the password into a client secret key.

Client	Server	Service
Client secret key		

Step 2 : Request for Ticket Granting Ticket – TGT, Client to Server

The **client** sends a plaintext message to the authentication server. This message contains

- username;
- the name of the requested service (in this case this is the Ticket Granting Server – TGS);
- the network address;
- the requested lifetime of the TGT.

Note that no secret information (client secret key or password) is sent).

Client	Server	Service
Client secret key		
Request for a TGT		

Step 3 : Server checks if the user exists

The **server** receives the message and will check if the username exists in the Key Distribution Center – KDC. Again this is not a credential check but only a check to verify that the user is defined. If all is OK the server proceeds.

Step 4 : Server sends TGT back to the client

The **server** generates a random key called the session key that is to be used between the client and the TGS.

The authentication server then sends back two messages to the client

- Message A is encrypted with the client secret key. The client secret key is not transferred but is retrieved from the password (more to speak the hash) found in the user database. This happens all on the server side. The message contains
  - TGS name;
  - timestamp;
  - lifetime;
  - the TGS session key (the key generated in the beginning of this step).
- Message B is the **Ticket Granting Ticket**, encrypted with the TGS secret key, that contains
  - your name;
  - the TGS name;
  - timestamp;
  - your network address;
  - lifetime;
  - the TGS session key (same as in message A).

Client	Server	Service
Client secret key	Client secret key (locally created)	
Request for a TGT	TGT	
	TGS session key	
	TGS secret key	

Step 5 : Enter your password

The **client** receives both messages and then requests the user for the password. In some cases this is already done in the first step. The password is then converted (hash) to the client secret key. Note that this key was also generated on the server side in the previous step.

Step 6 : Client obtains the TGS Session Key

The **client** now uses the client secret key to decrypt message A. This gives the client the TGS Session key.

The client can not do anything with message B (the TGT) for the moment as this is encrypted with the TGS secret key (which is only available at the server side). This encrypted TGT is stored locally in the credential cache.

Client	Server	Service

Client secret key	Client secret key (locally created)	
Request for a TGT	TGT	
Encrypted TGT	TGT	
TGS session key	TGS session key	
	TGS secret key	

Step 7 : Client requests server to access a service

The **client** now prepares two messages to be send to the server

- Message C is an unencrypted message that contains
  - the service that the client wants to access;
  - the lifetime;
  - message B or the TGT (this TGT itself is encrypted and included in the unencrypted message send to the server).
- Message D is a so called Authenticator encrypted with the TGS session key and contains
  - your name;
  - timestamp.

Step 8 : Server verifies if service exist

The **server** first verifies if the requested service exists in the KDC. If this is the case, it will proceed.

Step 9 : Server verifies request

The **server** now extracts the content of message B (the TGT) from message C and then decrypts that message B (the TGT) with its TGS secret key. This will give the server the TGS session key. This is a shared key between the client and the server.

With this TGS session key, the server is now able to also decrypt the message D.

The server now has your name and a timestamp from message D and a name and timestamp from message B. The server will then

- Compare both name and timestamp in both values;
- Check if the TGT is expired (the lifetime field in the TGT);
- Check that the Authenticator is not in the cache (to prevent replay).

If all checks turn out OK, the server continues.

Client	Server	Service
Client secret key	Client secret key (locally created)	
Request for a TGT	TGT	
Encrypted TGT	TGT	
TGS session key	TGS session key	
	TGS secret key	

Step 10 : Server generates service session key

The **server** generates a random service session key. It will then send two messages to the client.

- Message E : the service ticket that is encrypted with the service secret key and contains
    - your name;
    - the service name;
    - timestamp;
    - your network address;
    - lifetime;
    - the service session key.
- Message F : encrypted with the TGS session key containing
    - service name;
    - timestamp;
    - lifetime;
    - service session key.

Client	Server	Service
Client secret key	Client secret key (locally created)	
Request for a TGT	TGT	
Encrypted TGT	TGT	
TGS session key	TGS session key	
	TGS secret key	
	Service ticket	
	Service session key	

Step 11 : Client receives service session key

Because the **client** has the TGS session key cached from previous steps it can now decrypt message F to obtain the service session key. It is however not possible to decrypt the service ticket (message E) because that one is encrypted with the service secret key.

Client	Server	Service
Client secret key	Client secret key (locally created)	
Request for a TGT	TGT	
Encrypted TGT	TGT	
TGS session key	TGS session key	
	TGS secret key	
Encrypted Service ticket	Service ticket	
Service session key	Service session key	

Step 12 : Client contacts service

Now it’s time for the **client** to contact the service. Again two messages are send

- Message G : a new authenticator message encrypted with the service session key that contains
  - your name;
  - timestamp.
- Message H : the previously received message E, that is still encrypted with the service secret key

Step 13 : Service receives the request

The **service** then decrypts the message H (which is the same as message E) with its service secret key. It then uses the service session key (that was stored in message H/E) to obtain the service session key.

The service will then use that newly obtain service session key to decrypt the authenticator message G.

Client	Server	Service
Client secret key	Client secret key (locally created)	
Request for a TGT	TGT	
Encrypted TGT	TGT	
TGS session key	TGS session key	
	TGS secret key	
Encrypted Service ticket	Service ticket	
Encrypted Service ticket	Service ticket	Service ticket
Service session key	Service session key	Service session key

Step 14 : Service verifies request

Similar to step 9, the service then does some verification

- Compare the user name from the authenticator (message G) to the one in the ticket (comparing message H/E);
- Compare the timestamp in message G with the timestamp in the ticket (H/E);
- Check if the lifetime (message H/E) is expired;
- Check that the authenticator (message G) is not already in the cache, to prevent replay attacks.

If all checks turn out OK, the service continues.

Step 15 : Service confirms identity to the client

The **service** will then confirm its identity to the client

- Message I : an authenticator message encrypted with the service session key that contains
    - the id of the service;
    - timestamp.

Step 16 : Client receives confirmation

The **client** then receives the authenticator message (I) and decrypts its with the cache service session key (obtained in step 11). This allows the client to know the id of the service and if the timestamp is valid. If everything is OK the client can proceed.

Step 17 : Client communicates with the service

The authentication and verification is finished. The client can now talk to the service. Note that this only involves the authentication and verification of a service. This process will not decide if the client is actually allowed to do the requested service. This is something that is decided by the ACLs within the server that provides the service. This is not part of Kerberos.