Tickets

A ticket that client c can use to authenticate itself to s is of the form:

$$T_{c,s} \widehat{=} \{s, c, t, k_{c,s}\}_{key(s)}.$$

t is a timestamp, set to the time at which the ticket is created, and used to verify that a ticket is still valid.

If s is a TGS then this ticket is produced by Kerberos; if s is a normal server then this ticket is produced by a TGS. Note that only s can decrypt $T_{c,s}$.

Getting a ticket granting ticket

A client obtains a TGT by sending its identity and the identity of an appropriate TGS to Kerberos. Kerberos returns a session key and TGT.

Msg 1. $c \rightarrow kerb : c, tgs$

Msg 2. $kerb \rightarrow c \quad : \{T_{c,tgs}, k_{c,tgs}\}_{key(c)}$.

$key(c)$ is formed as a one-way hash of c's password. The user needs to supply the correct password in order for the client to obtain the session key $k_{c,tgs}$.

Getting a ticket

A client can request a ticket for a particular server from a TGS, as follows:

Msg 3. $c \rightarrow tgs : T_{c,tgs}, \{c, t\}_{k_{c,tgs}}$

Msg 4. $tgs \rightarrow c \quad : \{T_{c,s}, k_{c,s}\}_{k_{c,tgs}}$.

The TGS extracts the key $k_{c,tgs}$ from $T_{c,tgs}$.

This step can repeated multiple times (with the same TGT) to obtain tickets for different servers.

Requesting a service

Finally, clients can request a service from a server by sending the ticket:

Msg 5. $c \rightarrow s : T_{c,s}, \{c, t\}_{k_{c,s}}$.

$k_{c,s}$ can then be used to transfer information.

This step can be repeated multiple times, with the same ticket.