**Kerberos 5 Protocol**

**Terminology**

Key Distribution Center, or KDC for short, is an integral part of the Kerberos system. It consists of three components:

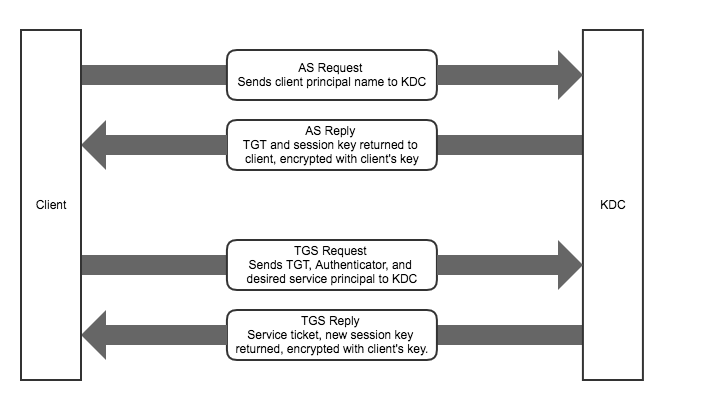
* Database of all principals and their associated encryption keys.
* Authentication Server (AS), which issues an encrypted Ticket Granting Ticket (also known as TGT) to clients who wish to “log in” to Kerberos.
* Ticket Granting Server (TGS), which issues individual service tickets to clients.

Ticket, it is an encrypted, expirable, data structure issued by the KDC that includes a shared encryption key that is unique for each session. Ticket serves two purposes:

* to confirm identity of the end participants
* to establish a short-lived encryption key that both parties can share for secure communication

**Protocol Insight**

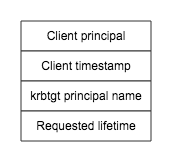
Following is an overview, protocol contasins two phases.

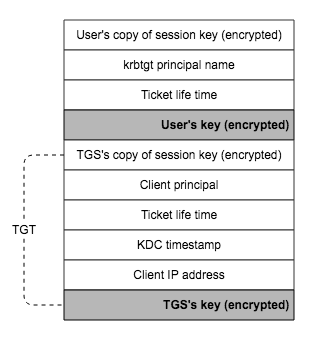
[](https://raw.githubusercontent.com/Reidddddd/reidddddd.github.io/master/assets/images/overview.png)The 1st phase is to prove who you are, a client will send an authentication request (AS\_REQ) to Authentication Server(AS), then AS will reply him with an encrpyted response (AS\_REP) which contains TGT, once the client decrypts the response, he can get the TGT and go to the 2nd phase.

The 2nd phase is to access service, he will send a ticket granting service request (TGS\_REQ), which contains the TGT obtained from AS\_REP, to Ticket Granting Service to tell which service he is going to access, TGS will reply him with an encrpyted response (TGS\_REP) which contains service ticket without which the client can’t access the wanted service. In fact, we can say that the TGT is a special service ticket, it is the required ticket for accessing TGS.

In conclusion, the 1st phase is an authentication process, the 2nd phase is an authorization process, TGS authorizes a client to access specified service.

**AS**

**AS\_REQ** consists of 4 parts, principal name of client, local time when doing this request, the principal of service in requesting (at AS phase, krbtgt should be the pricipal name of TGS by default), the life time of the ticket after which ticket is invalid and can not be used for accessing TGS. AS\_REQ is sent in plaintext.[](https://raw.githubusercontent.com/Reidddddd/reidddddd.github.io/master/assets/images/as_req.png)

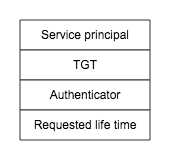
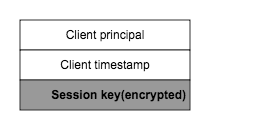
On receiving the AS\_REQ, AS verfies the client principal exists in backend db, and that the client’s timestamp is close to the KDC’s local time (usually with 5 miniutes), this is an early check in the event of a time mismatch between the client and the KDC. If either check fails, an error message is sent back to client and the client is not authenticated. Otherwise, an AS\_REP is sent back.[](https://raw.githubusercontent.com/Reidddddd/reidddddd.github.io/master/assets/images/as_rep.png) **tgt : @pritti**

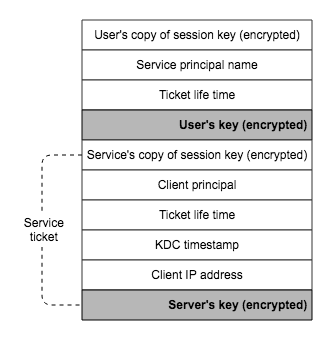
**Funny kura: ktbtgt principal name client lainae firta didinxa**

* The core of **AS\_REP** are two parts, one is the TGT encrypted with TGS’s key so that any client can not encrypt it. The other is the session key which is shared between the client and TGS, both can be obtained only by the user’s key, meaning that only the true user who knows the client’s key can decrpyt the AS\_REP and obtains the session key and the TGT.
* Session key is generated randomly by AS, and it is used for securing the ticket requests that the client makes to the TGS later on for specified kerberized service. KDC has two copies of this session key, one for the client and the other for the TGS.
* TGT can only be decrpyted by TGS, it mainly contains the information about the client’s info needed for double check: the client principal name, ip address of the requesting client, TGS’s copy of the session key, ticket’s lift time.

Try imaging an imediator captured an AS\_REP, but since not knowing the client’s key, he can not obtain the session key and the TGT. Therefore, the imediator is walled from kerberized services, in this way, security is guaranteed.

**TGS @star**

The knowledge of the session key and the possession of TGT allow the client to obtain tickets for further kerberized services without re-entering his password. Now the client can prepare a **TGS\_REQ** to TGS also consisting of four parts: TGT, Authenticator, principal name of service to be accessed, and requested ticket lift time.[](https://raw.githubusercontent.com/Reidddddd/reidddddd.github.io/master/assets/images/tgs_req.png)The authenticator consists of a timestamp and client’s principal, encrypted with the session key acquired from the AS exchange. The authenticator ensures that every ticket requests is unique, and also proves that the client has knowledge of the shared session key established during the AS exchange.[](https://raw.githubusercontent.com/Reidddddd/reidddddd.github.io/master/assets/images/authenticator.png)Upon receipt of the ticket request from the client, same service pricipal check is performed. In addition, TGS will decrypt the TGT using its key and obtain the another copy of session key. If it is from the correct user, both session keys are the same, so TGS can use it to decrypt the Authenticator and compare the client information in it with those in TGT.

If all checks pass, the KDC formulates a **TGS\_REP** that includes a new session key, shared between client and the application server, encrypted with client’s key as well. This ensures that only the true client can read the new session for use with the application server. The service’s copy of session key is embedded inside of the service ticket, encrypted with service’s key. Service ticket is almost the same as TGT except their principal names.[](https://raw.githubusercontent.com/Reidddddd/reidddddd.github.io/master/assets/images/tgs_rep.png)

Service ticket tw server lai pathaune ho teivyerw server’s key le encrypt grne

When the client accesses the application server, the knowledge of the service ticket and the session key will prove to the service that the client is authentic.

**NOTE**

The final step — how the client sends the service ticket to an application server — are different from every application server. Kerberos does not define a standard way for doing this, it leaves these decisions to the individual application servers.

Three cryptographic messages that are passed back and forth in both AS and TGS exchange are the service ticket, tht TGT, and the authenticator, illustrated above, worthy of more attentions.