

# Lecture Kerberos

Riccardo Torre

29 novembre 2025

**Cos'è** Servizio di autenticazione che fornisce un server di autenticazione centralizzato la cui funzione è di autenticare gli utenti ai server e i server agli utenti. Anche i server sono autenticati tra di loro.

**Nota** Kerberos fa uso solo ed esclusivamente di crittografia simmetrica!!! (DES a 56 bit)

**Requisiti Kerberos** Sicuro, trasparente, scalabile, affidabile.

## Attori

1. **AS = Authentication Server:** conosce gli hash delle password di tutti gli utenti e li memorizza in un database centralizzato e crea ogni volta una chiave simmetrica con il client sfruttando l'hash della password come seme da passare a funzioni pseudo random; condivide un'unica chiave con tutti i server;
2. **TGS = Ticket Granting Server;** emette i ticket agli utenti autenticati dall'AS necessari per ottenere un particolare servizio. Il ticket ha una durata (*Lifetime*) ben precisa e può essere riutilizzato. Condivide le chiavi con tutte le risorse (gli application server).
3. **Application server:** forniscono un determinato servizio.

## Ticket

- Ticket per entrare nel gioco che hanno una validità  $Time_1$  ore  $\rightarrow$  crittografia più forte;
- Ticket per utilizzare un servizio che hanno una validità  $Time_2$  minuti  $\rightarrow$  crittografia leggera;

$$Time_1 > Time_2$$

e

$$Lifetime_1 > Lifetime_2$$

ovvero la durata del ticket per entrare nel gioco è maggiore di quella del ticket per utilizzare un servizio. Man mano si usano ticket che durano sempre di meno.

**Authenticator** Serve per verificare che chi sta operando e chi si dichiara sono la stessa persona. Può essere usato una sola volta.

## Legenda

- **C:** Client
- **AS:** Authentication Server
- **TGS:** Ticket Granting Server
- **V:** server di un particolare servizio (ad esempio, il server di stampa).

**Note sui messaggi** I messaggi sono 6 in totale. Di seguito vengono riportati alcune note relative ai messaggi (1 a 6)

- (1) C fa autenticazione con il server AS; non trasmette la password perché non deve viaggiare in chiaro (per prevenire da un attacco man-in-the-middle); deve esserci un meccanismo per aggiornare la password
- (2) AS restituisce un Ticket; la chiave di sessione non viene trasmessa ma generata da C e da AS utilizzando una funzione basata su DES a cui viene passato  $H(\text{password})^1$  come seme.
- (3) il ticket è firmato da AS; TGS riceve il ticket, verifica se C ha i permessi per usufruire del servizio fornito da V.

## Il protocollo

- (1)  $C \rightarrow AS: ID_c \parallel ID_{tgs} \parallel TS_1$
- (2)  $AS \rightarrow C: E(K_c, [K_{c,tgs} \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2 \parallel Ticket_{tgs}])$   
 $Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \parallel ID_c \parallel AD_c \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2])$
- (3)  $C \rightarrow TGS: ID_v \parallel Ticket_{tgs} \parallel Authenticator_c$
- (4)  $TGS \rightarrow C: E(K_{c,tgs}, [K_{c,v} \parallel ID_v \parallel TS_4 \parallel Ticket_v])$   
 $Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \parallel ID_c \parallel AD_c \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2])$   
 $Ticket_v = E(K_v, [K_{c,v} \parallel ID_c \parallel AD_c \parallel ID_v \parallel TS_4 \parallel Lifetime_4])$   
 $Authenticator_c = E(K_{c,tgs}, [ID_c \parallel AD_c \parallel TS_3])$
- (5)  $C \rightarrow V: Ticket_v \parallel Authenticator_c$
- (6)  $V \rightarrow C: E(K_{c,v}, [TS_5 + 1])$  (per fare la mutua autenticazione)  
 $Ticket_v = E(K_v, [K_{c,v} \parallel ID_c \parallel AD_c \parallel ID_v \parallel TS_4 \parallel Lifetime_4])$   
 $Authenticator_c = E(K_{c,v}, [ID_c \parallel AD_c \parallel TS_5])$

---

<sup>1</sup>hash della password.

**Note sul protocollo** Nel messaggio (2) la chiave  $K_c$  viene generata usando gli hash della password locali come seme da passare una funzione pseudorandom separatamente sia da  $K_c$  che da AS. La chiave  $K_c$  viene cambiata ogni volta che si ripete il protocollo da capo. Le identità vengono legate crittograficamente alle chiavi per evitare **binding attacks**. Il ticket è qualcosa di codificato con la chiave condivisa fra AS e TGS. In Kerberos v4 la crittografia è annidata. La versione di Needham-Schroeder simmetrica è molto simile. Nel ticket viene inserito anche l'indirizzo di rete per identificare il client nella rete. Nel messaggio (3) viene presentato il ticket assieme ad  $ID_v$  perché non viene specificato all'interno della codifica di  $Ticket_v$  (dunque serve ad identificare il server che offre il servizio di cui C vuole usufruire). L' $Authenticator_c$  serve al TGS per verificare che di  $Ticket_{tgs}$ :

- $TS_2 < TS_3$ ;
- $TS_3 < Lifetime_2$ ;

dunque è necessario per garantire la sicurezza del protocollo che i clock siano sincronizzati. In generale deve valere che  $TS_i < TS_{i+1}$  e  $TS_{i+1} < Lifetime_i$ .

**Realms** Kerberos è **scalabile**, è possibile mettere insieme dei realm, sistemi kerberizzati (AS, TGS e i vari servizi). Si fa uno sharing del database centralizzato di tutte le risorse disponibili localizzato sul Kerberos master computer system. C'è una confederazione di realms. I vari realm hanno una copia del database centralizzato. C nel realm A si autentica con l'AS e ottiene un ticket per presentarlo al TGS di A specificando di voler utilizzare il servizio S del realm B. Dunque il TGS crea un apposito ticket e lo restituisce a C. Questo ticket viene presentato al TGS di B perché è lui che amministra e gestisce i servizi nel realm B (e non il TGS nel realm A). C'è una relazione di fiducia tra i due TGS (TGS realm A con TGS realm B). Quindi il TGS di B restituisce il ticket da presentare al server che offre il servizio S.

**Numero di chiavi nell'autenticazione cross-realms** Supponendo che ci siano  $n$  realm, il realm A per poter parlare con gli altri  $n - 1$  realm ha bisogno di  $\frac{n(n-1)}{2}$  chiavi. Questa formula è derivata dal calcolo degli archi di un grafo completo. Se si modellano i realms come i nodi di un grafo completo, e gli archi che collegano i TGS dei vari realms, per ciascun nodo si contano il numero di archi uscenti e i nodi sono  $n$ . Dato che ogni arco verrà contato due volte (si pensi al caso di un grafo completo composto da due nodi  $A \rightarrow B$  e  $B \rightarrow A$ ) è necessario effettuare la divisione per 2.

**v4 vs v5** AES al posto di DES, ticket lifetime con tuning al posto di ticket da 24 h, qualsiasi net address e non solo IP address, ordinamento dei byte dei messaggi, authentication forwarding che permette di evitare di chiedere un ticket per risorse correlate (ad esempio chiedere il ticket per la stampante e chiedere il ticket per il file da stampare diventa chiedere il ticket per stampare il file), autenticazione interrealm meno chiavi (meno di  $O(n^2)$ ), non c'è più un encoding annidato del ticket, cioè il ticket è stato separato dal messaggio che lo conteneva.