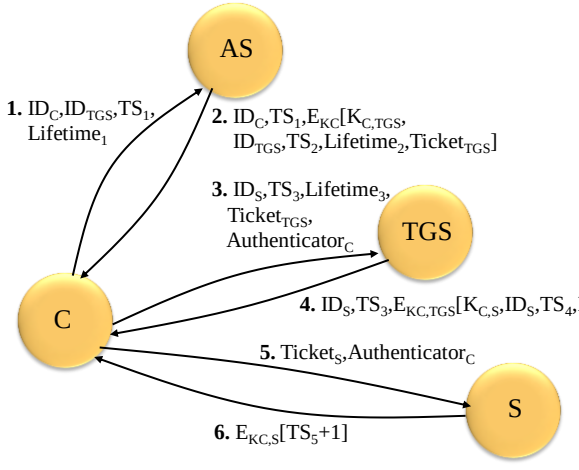
Kerberos Authentication Service Version V4

Ibéria Medeiros

Departamento de Informática
Faculdade de Ciências da Universidade de Lisboa

Kerberos V4



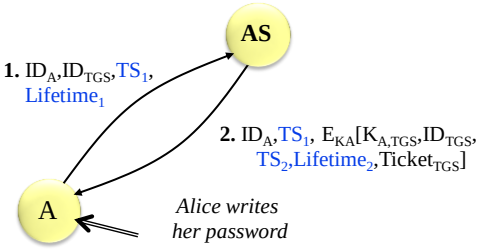
Terminology

AS - *Authentication Server*
TGS - *Ticket-Granting Server*
ID - identifier
AD – network level address
TS - timestamp
K – secret key
$E_K[X]$ - encrypt X with key K

$Ticket_{TGS} = E_{KTGS}[K_{C,TGS}, ID_C, AD_C, ID_{TGS}, TS_2, Lifetime_2]$
$Ticket_S = E_{KS}[K_{C,S}, ID_C, AD_C, ID_S, TS_4, Lifetime_4]$

$Authenticator_C = E_{KC,TGS}[ID_C, TS_3]$
$Authenticator_C = E_{KC,S}[ID_C, TS_5]$

Initial Authentication: User gets the TGT



Terminology

AS - *Authentication Server*
ID_X – identifier of X
K_X – master key of X
$E_K[X]$ – encrypt X with key K
TS - timestamp
$Ticket_{TGS}$ - $E_{KTGS}[K_{A,TGS}$,ID_A, AD_A, ID_{TGS},TS_2,$Lifetime_2]$

Allows Alice to

- start the authentication with Kerberos
- get a secret key for this session (used in all communications with the TGS)
- obtain a ***ticket-granting ticket (TGT)*** (=$Ticket_{TGS}$) that can be used to get other tickets

Is Alice authenticated by Kerberos? Or vice versa?

(Some) of the Messages Fields

- Message 1.

  ID_A, ID_{TGS} : identifiers of Alice and TGS (*Ticket Granting Server*)

  TS_1 : timestamp that allows Alice to match the request with the response (it also allows the AS to verify that Alice's clock is synchronized)

  $Lifetime_1$: suggested lifetime for the ticket (multiple of 5 minutes; with a maximum value of 21 hours, which can be problem …)

- Message 2.

  ID_Ae TS_1 : *see above*

  $K_{A,TGS}$: secret key for this session (generated by the AS) between Alice and TGS

  ID_{TGS} : ensures Alice that this ticket is really for the TGS

  TS_2 : clock value when the ticket was generated

  $Lifetime_2$: interval of time during which the ticket is valid

Fields of the TGT

It is encrypted with the master key of the TGS to prevent changes

$K_{A,TGS}$: secret key for this session between Alice and TGS

ID_A : identifier of the owner of the ticket (i.e., Alice)

AD_A : IP address of Alice's machine (prevents the use of Kerberos v4 in other protocols); ensures that only Alice's machine can use the ticket

ID_{TGS} : identifier of the TGS; can also be used to determine that the ticket was decrypted correctly

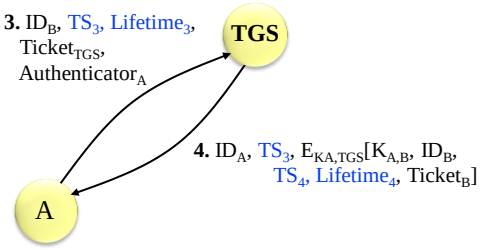
TS_2 e $Lifetime_2$: (*see previous slide*) prevents the ticket from being used beyond its validity period

Notes about Initial Authentication

- Protection of Alice's password
  - in V4, Alice's password is only asked after the AS response arrives
  - the password is deleted as soon the response is decrypted; from then on only the session key is utilized (and an adversary that catches the key can only use it for a limited period)
  - by sending a message to the AS, an adversary can obtain information that allows brute force attacks against Alice's master key

- What is the advantage of having the TGT ?
  - The TGT contains all information that is needed by the TGS to know about the current session of Alice, thus achieving the following
    - » The AS does not have to pass any information to the TGS
    - » It is not necessary to keep any information about the session in the AS or TGS, which simplifies replication and failure recovery mechanisms

Request for Authentication at a Remote Server



Terminology

TGS - *Ticket Granting Server*
ID_X – identifier of X
K_X – master key of X
$E_K[X]$ – encrypt X with key K
TS - timestamp

$Ticket_{TGS} = E_{KTGS}[K_{A,TGS}, ID_A, AD_A, ID_{TGS}, TS_2, Lifetime_2]$
$Ticket_B = E_{KB}[K_{A,B}, ID_A, AD_A, ID_B, TS_4, Lifetime_4]$ $Authenticator_A = E_{KA,TGS}[ID_A, TS_3]$

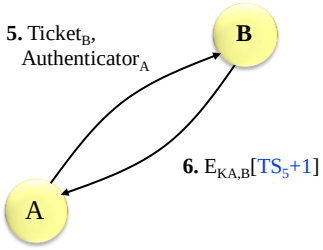
Allows Alice to

- indicate that she wants to authenticate with Bob
- get a session key (to protect the communications with Bob)
- obtain a ticket that will be used while authenticating with Bob

Notes about Authenticating at a Server

- Due to the authenticators it is necessary to have the **clocks of all machines synchronized**
  - it should be possible to define the maximum skew among clocks
  - the timestamp in the authenticator has a higher precision than the other timestamps (to avoid rejection of authentication attempts that are very close in time, in case the server keeps a record of the last observed timestamps)
- In the authentication of Alice at the TGS,
  - one would not need to use an authenticator because the response comes encrypted
  - the authenticator serves to keep the protocol similar to the authentication with other servers

Server Authentication



Terminology

TGS - *Ticket Granting Server*
ID_X – identifier of X
K_X – master key of X
$E_K[X]$ – encrypt X with key K
TS - timestamp

$Ticket_B = E_{KB}[K_{A,B},ID_A,AD_A,ID_B,TS_4,Lifetime_4]$
$Authenticator_A=E_{KA,B}[ID_A,TS_5]$

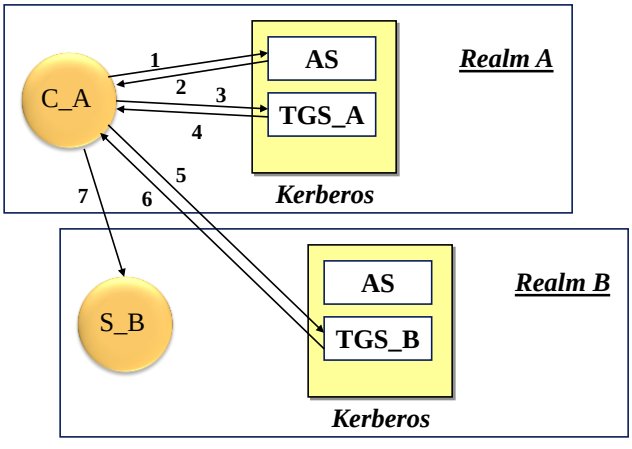
Allows Alice to

- authenticate herself with Bob
- authenticate Bob

Realms

Authentication between Realms

1. $ID_C,ID_{TGS\_A},TS_1,Lifetime_1$
2. $E_{KC}[K_{C,TGS\_A},ID_{TGS\_A},TS_2,$ $Lifetime_2,Ticket_{TGS\_A}]$
3. **ID_{TGS_B}**, TS_3, $Lifetime_3$, $Ticket_{TGS\_A}$, $Authenticator_C$
4. $E_{KC,TGS\_A}[K_{C,TGS\_B},ID_{TGS\_B},$ TS_4, $Lifetime_4,Ticket_{TGS\_B}]$
5. **ID_{S_B}**, TS_5, $Lifetime_5$, $Ticket_{TGS\_B}$, $Authenticator_C$
6. $E_{KC,TGS\_B}[K_{C,S\_B},ID_{S\_B},$ $TS_6,Lifetime_6$, $Ticket_{S\_B}]$
7. $Ticket_{S\_B},Authenticator_C$

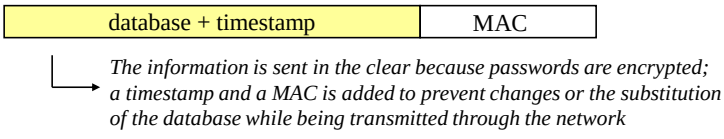


Transitivity in the Authentication Between Realms

- Rule: If realms A and B share a key, and if realms B and C share a key, and realms A and C do NOT share a key, then a user of A can NOT get a ticket for a server in C from B
- Problem: if the above rule was not enforced, then an malicious KDC that shares a key with a realm B, could carry out a personification attack of ANY user to the realm B (and with the realms with whom B shares a key, and then on)
- How does it work?
  - the identifiers are composed of <*name, instance, realm*>
  - If Alice@A attempts to talk with Carolina@C:
    - » Alice gets ticket for TGS_B
    - » Alice uses the ticket at TGS_B to obtain a ticket for TGS_C
    - » Alice uses the ticket at TGS_C to ask for a ticket to Carolina (which is **not provided** because TGS_C compares the realm of Alice with the realm of who generated the ticket to be used in TGS_C)

Replication

- Problem : Since all authentications are based on the KDC
  1. a failure, either in the network or KDC, prevents the users from working
  2. there can be performance problems (if authentication latency is high)
- Solution:
  - replicate the KDC in several nodes (all with the same master key)
  - one of the KDC is picked as the **master**, and the rest as reading replicas
  - all changes are made at the master (operations like add, change or remove)
  - in case of master failure there are no changes, but authentication can still be performed at the reading replicas
  - the database is periodically copied to the replicas



The information is sent in the clear because passwords are encrypted; a timestamp and a MAC is added to prevent changes or the substitution of the database while being transmitted through the network

Version Numbers of Keys

- Kerberos allows changes to the master keys at any instant by either the users or the servers
- On the other hand, a change in the master key invalidates all tickets that are encrypted with the previous master key (e.g., Alice gets a ticket to talk to Bob, and then Bob changes its master key)
- To address this problem in the most transparent way, Kerberos associates with each key a **version number**, and then
  - sends the version number of the key together with the encrypted data
  - requires servers to memorize previous keys while they can be still in use (maximum 21 hours)
  - requires users to use older passwords if data is encrypted with the old key (e.g., the user changes the master key, and then attempts to authenticate with a reading replica before the database is updated)

Bibliography

- W. Stallings, Cryptography and Network Security: Principles and Practice, Six Edition, 2014 (pages 478-495)
- C. Kaufman, R. Perlman, M. Speciner, *Network Security: Private Communication in a Public World (2 Edition)*, Prentice Hall, 2002. pag 307 -318