Kerberos

Gianluca Dini
Dept. of Ingegneria dell'Informazione
University of Pisa
Email: gianluca.dini@unipi.it
Version: 2022-05-25

Kerberos

- Based on the Needham-Schroeder protocol (1978)
- Developed at MIT in1980
- Kerberos V4 and Kerberos V5 (RFC 1510)
- Part of OSF DCE and Windows 2K (and later)
- Replaced the Windows NT domain authentication mechanism since W2K

Roadmap

- The simplified architecture
- The complete architecture
- Pre-authentication
- Delegation: Proxiable tickets and Forwardable tickets
- Realms

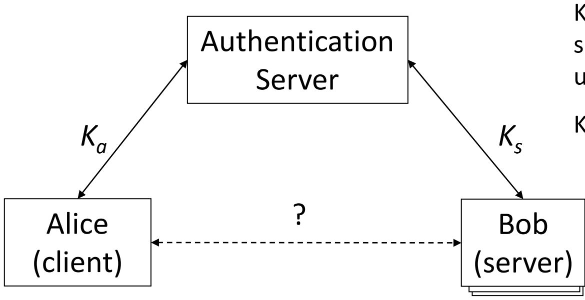
Kerberos requirements (→)

- Security
  - Eavesdropping and spoofing must be non possible for an outsider
- Availability
  - If Kerberos is unavailable all the services become unavailable
  - Kerberos is stateless so it can be replicated (but the pwd db must be replicated as well)

Kerberos requirements

- Transparency
  - The authentication process must be transparent but password typing
  - User experiments the customary `login:-password:` interface
- Scalability
  - Kerberos must handle a large number of servers and users

Kerberos: base architecture

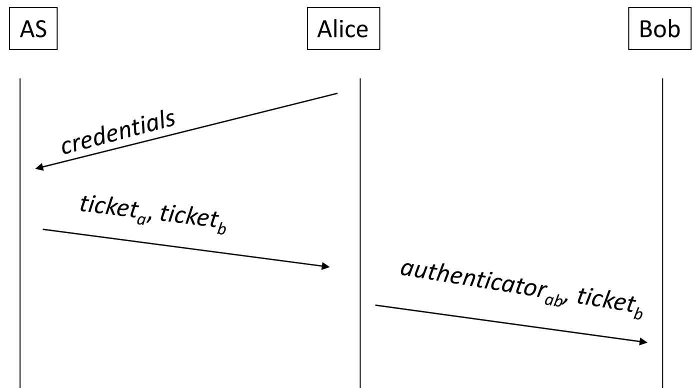


K_a e K_s (master keys) are shared secrets betwee AS and user and server, respectively

K_a is derived from a password

- Primary objective: mutual authentication
- Secondary objectives: establish a shared key between client and server

The basic idea



- $ticket_a = E_{Ka}(\text{"B"} || K_{ab} || \ldots)$;
- $ticket_b = E_{Kb}(\text{"A, B"} || K_{ab} || \ldots)$;
- $authenticator_{ab} = E_{Kab}(t_a || \ldots)$

Kerberos V (simplified)

KRB_AS_REQ	$M1 \quad A \to AS:$	A, B, t, L, N_a, WS
KRB_AS_RESP	$M2 \quad AS \to A:$	$\{A, B, t, L, K_{ab}, WS\}_{K_b}, \{B, t, L, N_a, K_{ab}\}_{K_a}$
KRB_AP_REQ	$M3 \quad A \to B:$	$\{A, B, t, L, K_{ab}, WS\}_{K_b}, \{A, t_a, \cdots\}_{K_{ab}}$
KRB_AP_RESP	$M4 \quad B \to A:$	$\{t_a, \cdots\}_{K_{ab}}$

$ticket_b$ $\quad$ $authenticator_{ab}$ $\quad$ $ticket_a$

Kerberos V

KRB_AS_REQ	$M1 \quad A \to AS:$	A, B, t, L, N_a, WS
KRB_AS_RESP	$M2 \quad AS \to A:$	$\{A, B, t, L, K_{ab}, WS\}_{K_b}, \{B, t, L, N_a, K_{ab}\}_{K_a}$
KRB_AP_REQ	$M3 \quad A \to B:$	$\{A, B, t, L, K_{ab}, WS\}_{K_b}, \{A, t_a, subkey_a\}_{K_{ab}}$
KRB_AP_RESP	$M4 \quad B \to A:$	$\{t_a, subkey_b\}_{K_{ab}}$

- *L* Validity interval of the ticket. Alice reuses the ticket for multiple authentications to Bob without interacting with AS so avoiding messages M1 and M2
- The timestamp t_a is generated by Alice. *Bob* verifies the freshness. For each authentication, Alice generates a new authenticator using the same K_{ab} but a different t_a
- The work station identifier *WS* allows the server to control which computers can use the ticket
- The $subkey_a$ *e* $subkey_b$ can be used for the service fulfillment.

Analysis

Assumptions

$$A \mid\equiv A \overset{K_a}{\leftrightarrow} AS \qquad B \mid\equiv B \overset{K_b}{\leftrightarrow} AS$$

$$AS \mid\equiv A \overset{K_a}{\leftrightarrow} AS \qquad AS \mid\equiv B \overset{K_b}{\leftrightarrow} AS$$

$$AS \mid\equiv A \overset{K_{ab}}{\leftrightarrow} B$$

$$A \mid\equiv \left(AS \Rightarrow A \overset{K_{ab}}{\leftrightarrow} B \right) \qquad B \mid\equiv \left(AS \Rightarrow A \overset{K_{ab}}{\leftrightarrow} B \right)$$

clock synchronization

$$A \mid\equiv \#(t)$$

$$B \mid\equiv \#(t) \qquad B \mid\equiv \#(t_a)$$

Idealized protocol

$$M2 \quad AS \to A: \quad \left\{ t, N_a, \left(A \overset{K_{ab}}{\leftrightarrow} B \right) \right\}_{K_a}, \left\{ t, \left(A \overset{K_{ab}}{\leftrightarrow} B \right) \right\}_{K_b}$$

$$M3 \quad A \to B: \quad \left\{ t, \left(A \overset{K_{ab}}{\leftrightarrow} B \right) \right\}_{K_b}, \left\{ t_a, \left(A \overset{K_{ab}}{\leftrightarrow} B \right) \right\}_{K_{ab}} \text{ from } A$$

$$M4 \quad B \to A: \quad \left\{ t_a, \left(A \overset{K_{ab}}{\leftrightarrow} B \right) \right\}_{K_{ab}} \text{ from } B$$

Analysis

After message M2

$$A \mid\equiv A \overset{K_{ab}}{\leftrightarrow} B$$

After message M3

$$B \mid\equiv A \overset{K_{ab}}{\leftrightarrow} B$$

key authentication

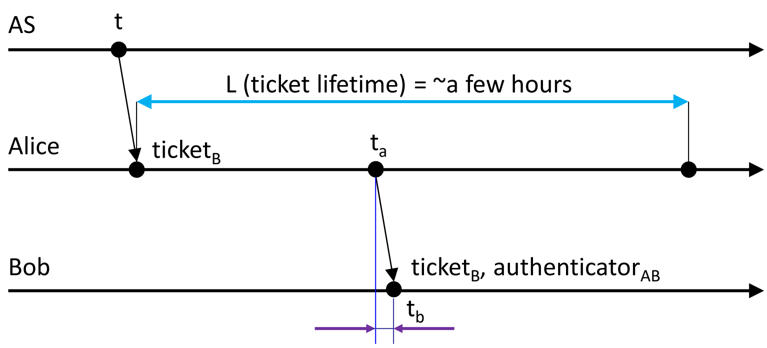
$$B \mid\equiv A \mid\equiv A \overset{K_{ab}}{\leftrightarrow} B$$

After message M4

$$A \mid\equiv B \mid\equiv A \overset{K_{ab}}{\leftrightarrow} B$$

key confirmation

Lifetime and authenticator



- $|t_b - t_a| <= \lambda$ (authenticator lifetime)
- authenticator lifetime ≈ clock skew (λ= 5 minuti)

Comments

- Security
  - Kerberos requires synchronized clocks
    - λ= 5 minutes => authenticator can be replayed in that window
  - K_a derives from a pwd
    - Ka is as secure as the pwd
- Transparency
  - After L, the user is logged out
- Availability
  - Kerberos is stateless so it can be replicated (but the pwd db must be replicated as well)

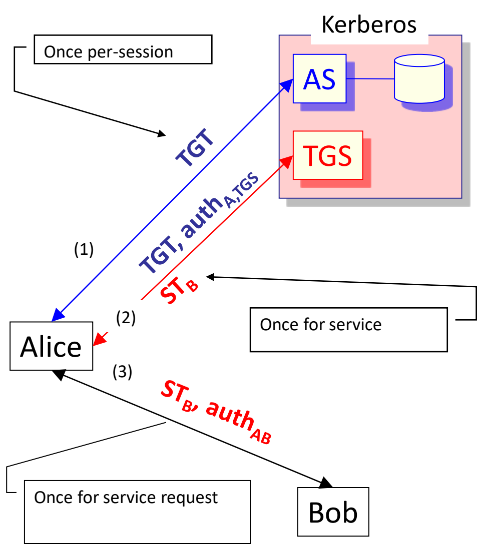
Kerberos

THE COMPLETE ARCHITECTURE

Complete architecture

- A user uses quite a few services
- The user has to authenticate to each service
- Two approaches
  - The user inputs the pwd for each new authentication and then deleted soon (little usable)
  - The WS stores the pwd for a long period (little secure)

TGS and TGT



"Every problem in computer science can be solved by adding a level of indirection" – D. Wheelar

1. AS releases the ticket granting ticket (TGT) to interact with the Ticket Granting Server (TGS)
2. TGS releases a service ticket (ST_B) to interact with Bob

Ticket Granting Service

Phase 1

Alice interacts with AS and receives a TGT, *ticket granting ticket*, a ticket for server TGS

Phase 2

Alice uses TGT to request TGS a service ticket ST_b to interact with the B

Phase 3

Alice uses ST_b to authenticate and to get authenticated to Bob

Interacting with TGS

Phase 2: msg KRB_TGS_REQ

Alice asks TGS a service ticket for B

$$\{A,t_a\}_{TK},B,t',L',N_a,\{A,TGS,t,L,TK,WS\}_{K_{TGS}}$$

Phase 2: msg KRB_TGS_RESP

TGS releases Alice a service ticket for B

$$\left\{A,B,t',L',K_{ab},WS\right\}_{K_b},\left\{B,t',L',K_{ab},WS,N_a\right\}_{TK}$$

Messaggio KRB_TGS_REQ

$$\{A,t_a\}_{TK},B,t',L',N_a,\{A,TGS,t,L,TK,WS\}_{K_{TGS}}$$

Authenticator

Sent by Alice to TGS

Ticket Granting Ticket

- TK: Ticketing Key (temporary key)
- (t, L) durata del TGT

Service Ticket Request

Alice asks TGS a service ticket for B

Messaggio KRB_TGS_RESP

Service Ticket for Bob released by TGS to Alice

$$\left\{A,B,t¢L¢K_{ab},WS\right\}_{K_b},\left\{B,t¢L¢K_{ab},WS,N_a\right\}_{TK}$$

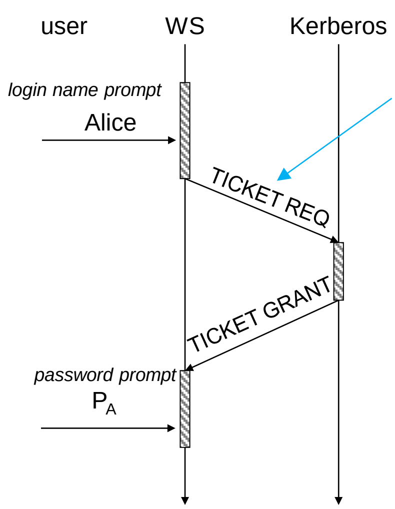
Service Ticket for Bob

Service Ticket for Alice

Authentication

- Basic Kerberos authenticates users w.r.t. network services
- Basic Kerberos **does not** authenticate users w.r.t. AS
  - Anyone may ask AS for a ticket on Alice behalf
  - Kerberos guarantees that nobody, but Alice, can use that ticket
  - An adversary may use this to launch a pwd attack
    - Guess a pwd and verify the guess by decrypting a ticket
- Basic Kerberos **does not** help WS to authenticate users (indirect authentication)
  - WS is just a means through which users access services
  - WS are uniform, interchangeable, thin client;

Normal authentication

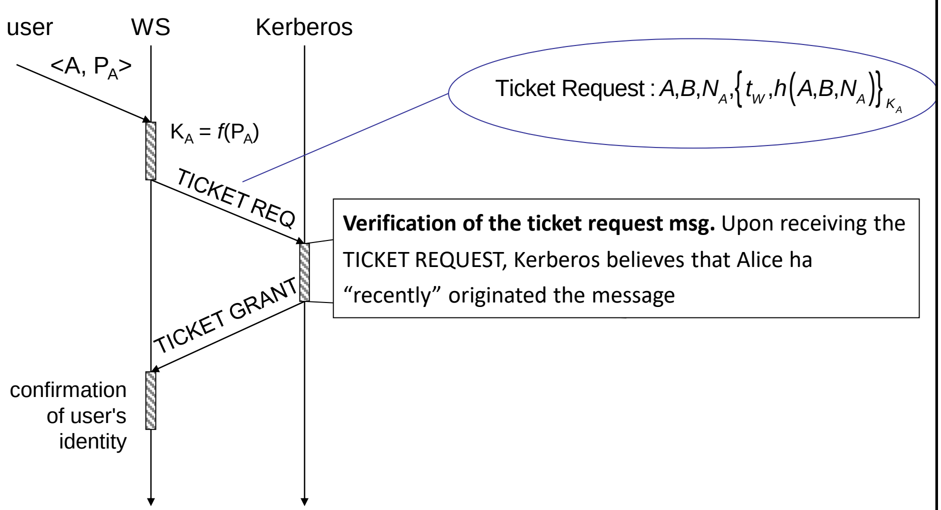


WS does not authenticate Alice but the **TGT** is encrypted.

Kv5 – normal authentication

- ACTIVE ATTACK - an adversary may sit at WS and try to guess the pwd
  - In the Kerberos' log you only see the ticket request
- PASSIVE ATTACK - an adversary can collect a ticket granting ticket and use it to offline launch a pwd attack

Kerberos 5 pre-authentication

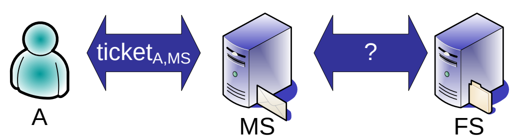


Verification of the ticket request msg. Upon receiving the TICKET REQUEST, Kerberos believes that Alice ha "recently" originated the message

Kv5 Pre-Authentication

- **Active attack** - Kerberos verifies ticket request and logs failed requests => active attacks become detectable
- **Passive attack** - the adversary can still launch a (passive) password attack using ticket requests collected on the network, but the attack becomes more difficult because the ticket request contains a timestamp and a pseudo-random number

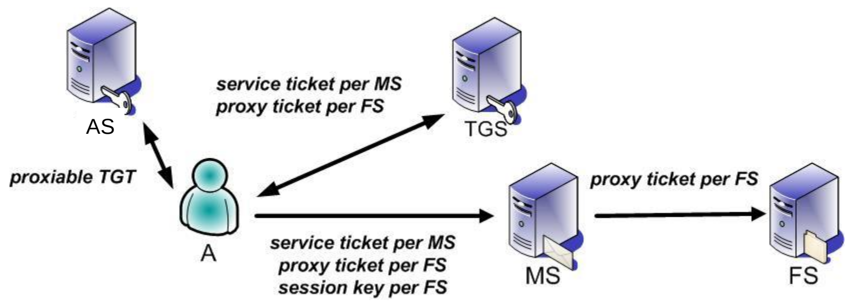
Delegation



- Example. Mail Server MS has to interact with File System FS *on the iser behalf* according to the *minimum privilege principle*
- Kerberos provides two mechanisms that allow Alice to delegate MS
  - proxy tickets
  - forwardable TGT

Proxy ticket

PT allows us to request a service ticket linked to an address (WS) different from the requesting one



Proxy Ticket: $\{K_{A,FS}, N_A, t, L, FS\}_{TK}, \{K_{A,FS}, MS, t, L\}_{K_{fs}}$

$\{\text{session key for FS}\}_{\text{session key for MS}} \circ \{\ldots, K_{A,FS}, \ldots\}_{K_{a,ms}}$

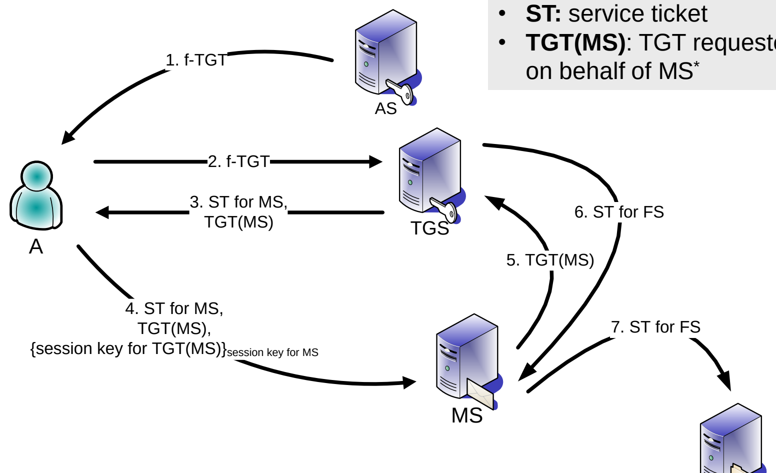
Proxy ticket for FS required by A, released to MS

Proxy ticket: cons

- Problem. This solution requires that
  - Alice knows in advance all the proxy tickets she needs or,
  - She is able to negotiate them with MS as needed
- Forwardable tickets make it possible to solve this problem and allow the delegated server MS to ask the necessary tickets

Forwardable ticket

- **f-TGT:** ***forwardable*** TGT
- **ST:** service ticket
- **TGT(MS)**: TGT requested by A on behalf of MS*



* The ticket specifies identifier MS instead of A

Forwardable ticket

- **[…]**
- **Step 3**. TGS returns Alice
  1. A ***service ticket*** for MS
     $\{\ldots, K_{a,ms}, \ldots\}_{TK}, \{\ldots, K_{a,ms}, \ldots\}_{Kms}$
  2. A ***TGT*** containing the ***ticketing key*** associated to MS instead of Alice
     $\{\ldots, TK', \ldots\}_{Ka}, \{\ldots, TK', \underline{MS}, \ldots\}_{Ktgs}$.
- **Step 4.** Alice forwards the two tickets to MS together with the ticketing key **TK'** encrypted by means of con **$K_{a,ms}$**
- **[…]**

The ticketing key is associated to MS instead of A

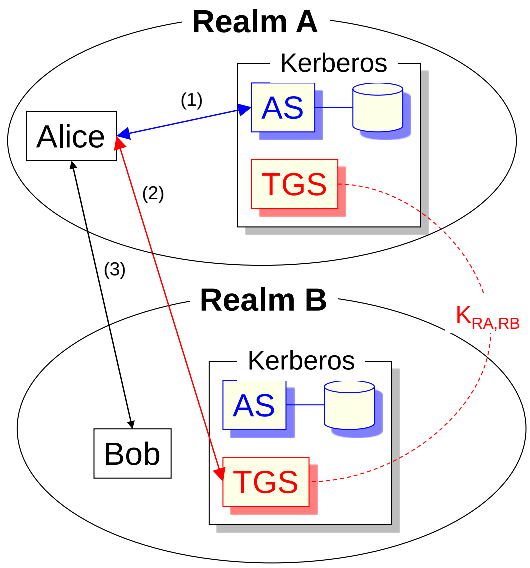
Proxy vs forwardable ticket

- Proxy ticket
  - (PRO) The user controls which rights to delegate the server
  - (CON) The user needs to know which tickets will be necessary
- Forwardable ticket
  - (PRO) The server determines which ticket it needs
  - (CON) A compromised servers can abuse of all rights

Limitations to delegations

- A ticket has a maximum lifetime
- A ticket may specify a maximum number of access rights (capability)

Realms e referral tickets



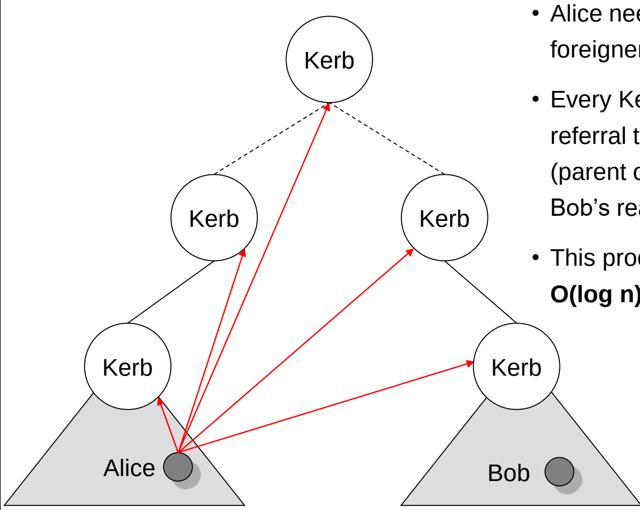
Realms e referral tickets

- THE PROBLEM
- Users and servers may belong to different administrative domains (realms)
- Kerberos authenticates principals in its own realm
- Problem. Alice has to authenticate a *foreign* server Bob (in another realm)

Realms e referral tickets

- THE PROTOCOL
- Alice asks Kerberos A a referral TGT for realm B (1).
- Kerberos A generates a referral TGT using an *inter-realm key* (KRA,RB)
- Alice uses the referral TGT to request Kerberos B a service ticket to Bob (2).
- Alice uses the service ticket to interact to Bob (3)

Hierarchy of realms



- Alice needs a service ticket for a foreigner server Bob
- Every Kerberos gives Alice a un referral ticket for the next node (parent or child) until Alice gets Bob's realm
- This process requires **O(log n)** operations

Intrusion tolerance

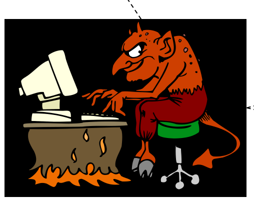
- A pragmatic approach: Kerberos is subject to intrusions but limits their effects
  - Workstation. Damages are limited to the workstation and its users
  - Server. Damages are extended to all server's users
    - A good practice is to distribute servers over multiple machines
  - Kerberos. The system is completely broken

Clock synchronization

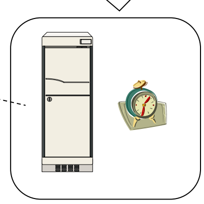
Time server

Adversary has an old key and the related ticket

NTP



Possible objectives



If the adversary succeeds in turning back the clock, then it can reuse the key

Server

Public-key encryption

Certificates remove the need of shared secrets based on reusable passwords

Procedure PKINIT

$$\boxed{\begin{array}{lll} M1. & A \rightarrow T & S_A(A, B, N_A), certificate_A \\ M2. & T \rightarrow A & ticket_B, E_{e_A}\left(S_T(K, N_A, L, B)\right) \end{array}}$$

Alice holds $certificate_T$

W2K encapsulates PKINIT in its Kerberos-based authentication environment