Kerberos

Gianluca Dini
Dept. of Ingegneria dell'Informazione
University of Pisa
Email: gianluca.dini@unipi.it
Version: 12/05/2025

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
- Adopted in the Unix world

Roadmap

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

Kerberos requirements (→)

- Security
  - Eavesdropping and spoofing must be not 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)

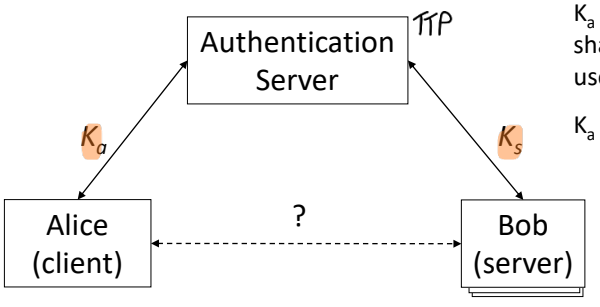
Kerberos requirements

- 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

7

Was conceived for Symmetric encryption

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

Users' workstation

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

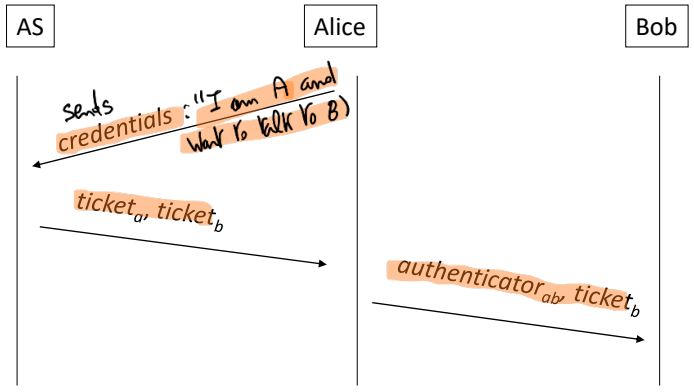
K_a, K_s are long term (master) keys. Alice's key derived from pwd, Server's key is just a key.

Obj. 1: Alice wants proof she's talking to Bob and they also want to establish a shared key.

Session can last a varying amount of time

8

The basic idea



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

↳ Forwards Bob ticket key + an authenticator,

Kerberos May-25 9

9

What allows Alice to prove Bob she has ticket in her hands and so does Bob.

Tickets ensure key authentication, authenticators for key confirmation

Kerberos V

Protocol can be split in two parts: ticket exchange for session key, then as long as session key is valid I just need to send authenticator

KRB_AS_REQ
KRB_AS_RESP
KRB_AP_REQ
KRB_AP_RESP

KRB assumes clocks are synchronised → timestamps → validity period of session key

Conceived for Unix campus

M1 $A \rightarrow AS$: A,B,t,L,N_a,WS → address (ERit, IP adr.) & user's workstation of Alice; nonce (why both t_a N_a?)

M2 $AS \rightarrow A$: $\{A,B,t,L,K_{ab},WS\}_{K_b}, \{B,t,L,N_a,K_{ab}\}_{K_a}$

M3 $A \rightarrow B$: $\{A,B,t,L,K_{ab},WS\}_{K_b}, \{A,t_a,\cdots\}_{K_{ab}}$

M4 $B \rightarrow A$: $\{t_a,\cdots\}_{K_{ab}}$

① Proof of freshness is a nonce but stronger for Alice's N_a.

ticket has timestamp to avoid Needham Schroeder protocol (Message for Bob doesn't contain any proof of freshness)

$ticket_b$ | $authenticator_{ab}$ | $ticket_a$

NOTE: Workstation is in the receiving TICKET for server, not user

Kerberos May-25 10

10

① t_a is another timestamp to inform Bob she has the key in her hands. Bob sends a very similar message back to prove knowledge

Kerberos V

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

- During the ***validity interval L*** 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 Importance will be evident when it comes to delegation (no personal computers)
- The ***$subkey_a$*** e ***$subkey_b$*** can be used for the service fulfillment.
  - instead of using K_{AB}, you can perform other transactions with subkeys. Reduce the amount of CT generated by some key at minimum, especially because K_{AB} is used for authentication

Analysis

Assumptions

clock synchronization

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

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

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

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

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

They consider time stamps as fresh

⇒ Consider AS as an authority on session keys

Idealized protocol

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

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

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

Analysis

After message M2

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

Because of timestamp is fresh, and since AS is trusted over generate

key authentication

After message M3

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

Same here, in M3 we have K_{AB} encrypting message in the current execution in t_A.

Proof really straight forward

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

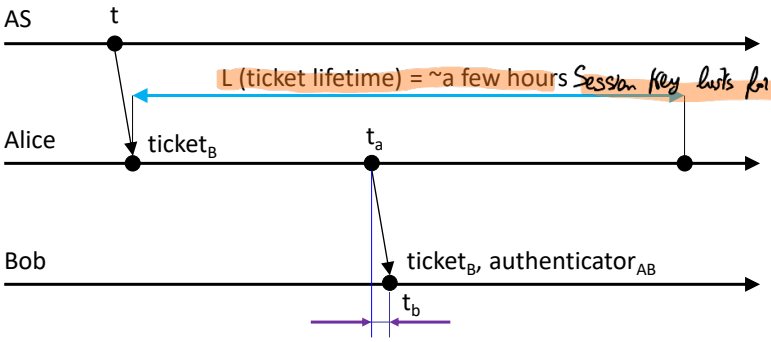
After message M4

key confirmation

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

Lifetime and authenticator

Timeline of protocol



Authenticator is timestamped by means of Alice's timestamp. Message will arrive at Bob's at t_B (network delay + slight difference in computing speed, slight desynchronization)

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

So you will have a difference: $t_b - t_a$. λ is auth lifetime that describes when an auth is fresh. You could have longer lifetimes, but more possible replays.

If you reduce λ, probability of denial of service increases.

While ticket lifetime is in the order of hours, authenticator lifetime in order of minutes.

NOTE: If AS is low, sys is low. If AS is compromised, machine is compromised.

Comments

- Security
  - Kerberos requires synchronized clocks
    - λ= 5 minutes ➔ authenticator can be replayed in that window from the same WS
  - 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

more complex because of:

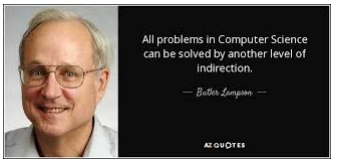
- A user uses quite a few services
- The user must authenticate to each service
- Two approaches
  - The WS deletes Ka after every execution of the protocol ➔ the user must input the pwd for each new session (authentication) and then delete it soon ➔ not usable
  - The WS caches (in the clear) K_a for the user session (potentially a long period) ➔ insecure

User must establish a session key with several servers

User would need to authenticate once for each server

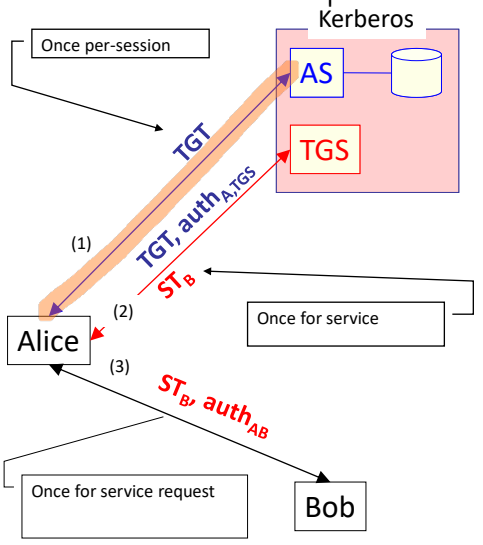
the user must input the pwd for each new session (authentication) and then delete it soon → not usable

Famous quote



All problems in computer science can be solved by another level of indirection.
- David Wheeler

TGS and TGT



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

20

Ticket Granting Service

- Phase 1
  - Alice interacts with AS and receives a TGT, *ticket granting ticket*, a ticket for server TGS
    - TGS is yet another server
- 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

21

The 1st two messages remain the same. They involve AS.
K_{AB} is now the key to talk to TGS now.

Interacting with TGS

Phase 2: msg KRB_TGS_REQ

Alice asks TGS a service ticket for B

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

AUTHENTICATOR

Phase 2: msg KRB_TGS_RESP

TGS releases Alice a service ticket for B

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

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

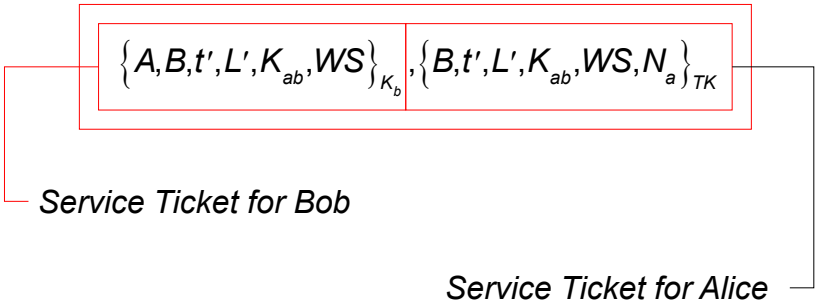
- TK: Ticketing Key (temporary key)
- (t, L) durata del TGT

Service Ticket Request

Alice asks TGS a service ticket for B

Messaggio KRB_TGS_RESP

Service Ticket for Bob released by TGS to Alice

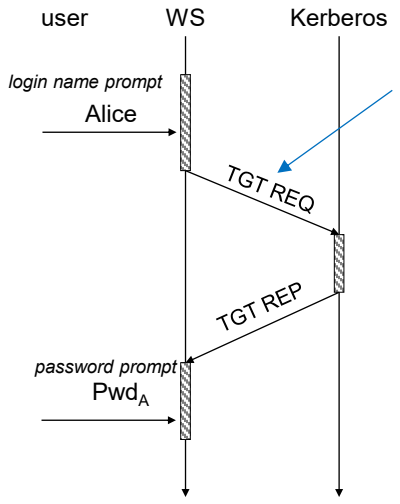


Kerberos

PRE-AUTHENTICATION

At the beginning Kerberos was structured like this: Alice introduces her name, the Kerberos protocol starts. When the workstation obtains 1st message from AS or TGT, user introduces Pwd and workstation tries to decrypt message. Problem: anyone could pretend to be Alice so Kerberos would send a message encrypted with K_A so adversary could implement offline attack.

Normal authentication



- Kerberos does not help WS to authenticate Alice
- TGT is encrypted
- TGT can be used to mount an off-line attack against PWD_A

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 a ticket
  - An adversary may exploit this to launch a pwd attack
  - Guess a pwd and verify the guess by decrypting a ticket

Authentication [↓]

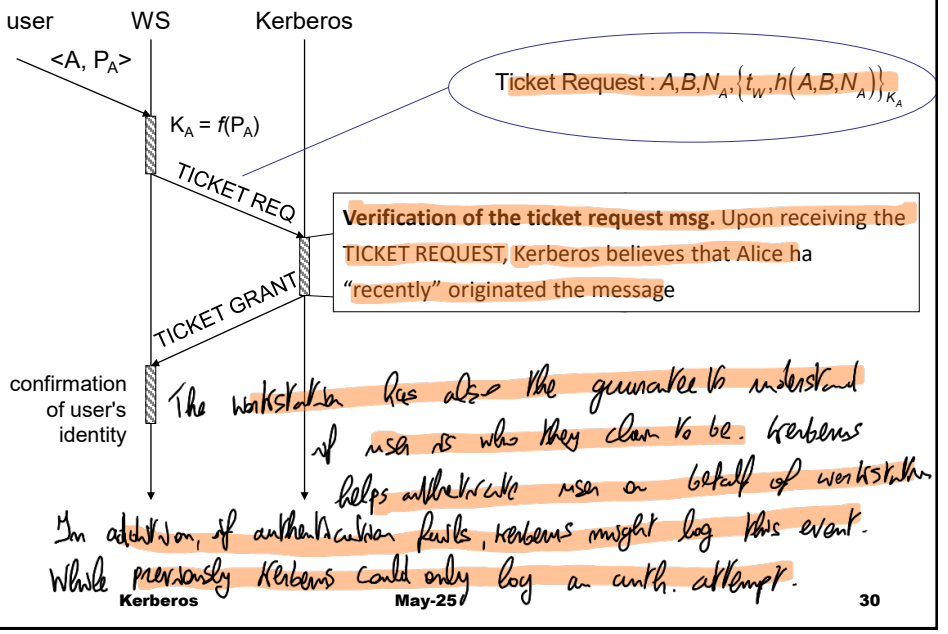
- Basic Kerberos **does not** help WS to authenticate users (indirect authentication)
  - WS is just a means through which users access services
  - WS are uniform, interchangeable, thin client;
  - However, PC are a reality

Kerberos 5 – 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 TGT and use it to offline launch a pwd attack

Protocol modified this way: user inserts username + pwd. Key is generated and then the request is now authenticated by means of Alice's key.

Kerberos 5 pre-authentication

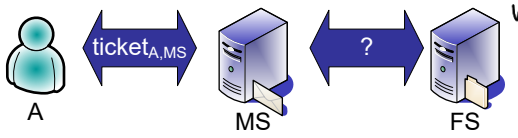


Kv5 Pre-Authentication

- **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

User Alice that wants to check mail and store it in mailbox (a file) so mailserver can write in a file.



File server

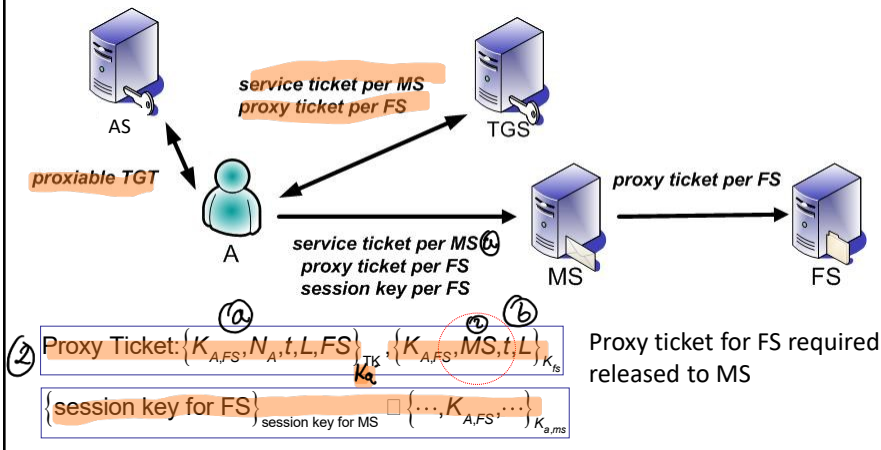
- Example. Mail Server MS has to interact with File System FS *on the iser behalf* according to the *minimum privilege principle* ①
- Kerberos provides two mechanisms that allow Alice to delegate MS
  - proxy tickets
  - forwardable TGT

① You must have access rights that allow you to do your work, no more.

32

Proxy ticket

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



Proxy ticket for FS required by A, released to MS

33 Alice requests a proxiable TGT: allows Alice to ask TGS a service ticket on behalf of someone else: "this TGT will be used not by me but someone else". Problem: when mailserver wants to write on file server on Alice Mailbox, either mail server can write on any file, or it can write on Alice Mailbox when it is operating on Alice's behalf.

1 would be too dangerous.

② One ticket is to talk to the mail server, but for Mail Server TGS_A talk to File Server which is needed to talk to File server. Second ticket will allow Mail server to talk to File Server on behalf of Alice.

TGS_B

- $K_{A,MS}$: session key to talk to the mail server.
- Proxy ticket contains session key $K_{A,FS}$ that TGS releases to Alice in a way that Alice can handle to the MS (both tickets come in pairs, one encrypted by means of MS and one by means of A).

- Service ticket contains session key to talk to server (tickets are two, in which session key is encrypted by means of K_A and K_{MS}). Alice retains K_A encrypted one, obtains $K_{A,MS}$ and forwards other ticket to MS when she wants to perform mail transaction (SERVICE TICKET PER MS)
- Mail server needs to talk to FS on behalf of Alice.
  Alice has proxy ticket (ticket given by TGS to let Alice interact with FS. It contains $K_{A,FS}$ to interact with FS.). This comes in pairs, you have two tickets. Alice forwards this proxy ticket for FS to the mail server (the part for the server, Alice retains the part for herself). Since this is a ticket to talk to the FS, it contains key $K_{A,FS}$; ⑥ will be sent that will be forwarded to the FS in case MS needs transactions.
  But MS will need that key, how will it receive it? It will be received in encrypted form with $K_{A,MS}$.

NOTICE: Proxy ticket sent to Alice doesn't have Alice's address, it has the MS address.

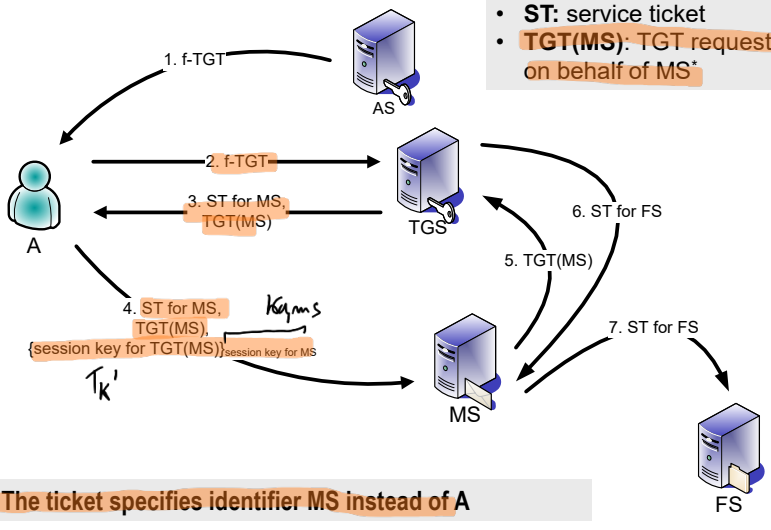
Proxy ticket: cons

- Problem. This solution requires that
  - Alice knows in advance all the proxy tickets she needs or, Alice knows how MS's service is implemented: violates encapsulation
  - 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

Tickets have short time duration, so if MS is compromised, you only see MS proxy tickets you have (can only intercept with FS, and only in lifetime of ticket).

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

Alice requires a forwardable TGT to AS. By means of FTGT A requires from TGS a service ticket for MS (to interact with MS) and she asks for a TGT(MS), that Alice requires for the MS. Alice, instead of having to know how MS is implemented (no need to know which service is needed), she handles MS the TGT(MS): "use this to talk to any server you need on behalf of me". It is used to ask service tickets on behalf of Alice. The ticket specifies identifier MS instead of A. Ticket limits identifier and key together.

Alice transmits the usual Service Ticket and the TGT(MS), that allows MS to request tickets.

And Alice sends encrypted TK' by means of $K_{A,MS}$

Forwardable ticket

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

The ticketing key TK' is associated to MS instead of A

Proxy vs forwardable ticket

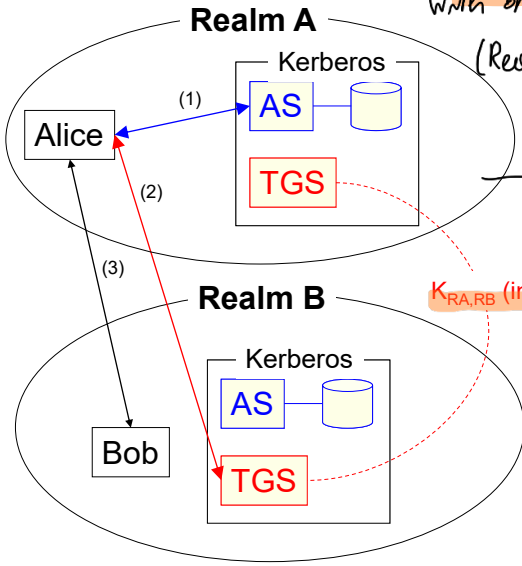
- Proxy ticket
  - (PRO) The user controls which rights to delegate the server
  - (CON) The user needs to know which tickets will be necessary
- Forwardable ticket
  - (PRO) The server determines which ticket it needs
  - (CON) A compromised servers can abuse of all rights

Limitations to delegations

- A ticket has a maximum lifetime
- A ticket may specify a maximum number of access rights (capability) You can specify access rights at a finer granularity in the tickets.
  Ex: I only want to read mailbox, not necessary to write

For the moment we assume Alice, Bob, AS work in the same organization (or realm). What if I want to interact with other departments or realms? (Realm: Kerberos Terminology)

Realms e referral tickets



→ Different Auth servers.

$K_{RA,RB}$ (inter-realm key)

↓ There exists an inter realm key shared by two TGSs of realms

Realms e referral tickets

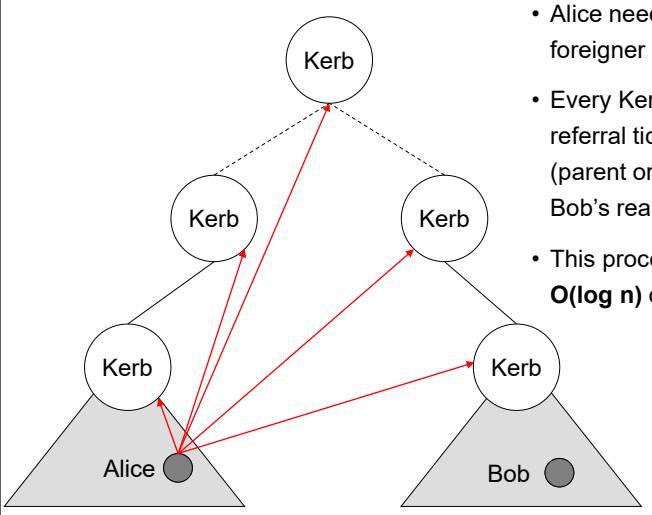
- THE PROBLEM
- Users and servers may belong to different administrative domains (realms)
- Kerberos authenticates principals in its own realm
- Problem. Alice has to authenticate a *foreign* server Bob (in another realm)

Realms e referral tickets

- THE PROTOCOL
- Alice asks Kerberos A a *referral* TGT for realm B (1). (1)
- Kerberos A generates a referral TGT using an *inter-realm key* ($K_{RA,RB}$) TGT won't be encrypted by means of T_K but by means of $K_{RA,RB}$ to be sent to Kerberos B.
- 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)

↓
Not (A) but this $K_{RA,RB}$

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

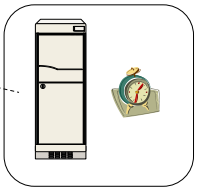
Clock synchronization

Time server

Adversary has an old key and the related ticket

NTP

Possible objectives



Server

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

Public-key encryption

Certificates remove the need of shared secrets based on reusable passwords

Procedure PKINIT

$$M1.\quad A \rightarrow T \quad S_A(A,B,N_A), certificate_A$$
$$M2.\quad T \rightarrow A \quad ticket_B, E_{e_A}(S_T(K,N_A,L,B))$$

Alice holds *certificate*$_T$

W2K encapsulates PKINIT in its Kerberos-based authentication environment

47

48