# Kerberos

## Kerberos

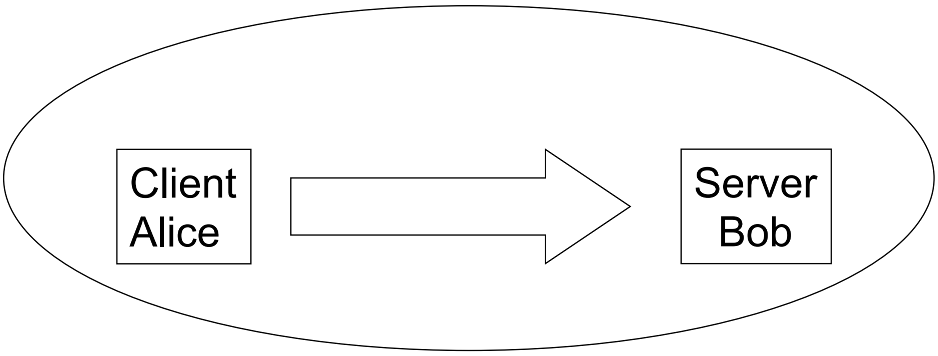
- Kerberos is based on the Needham-Schroeder protocol (1978)
- Kerberos was developed at MIT in1980
- Kerberos V4 and Kerberos V5 (RFC 1510)
- Kerberos if part of OSF DCE and Windows 2K (e later)
- In Windows 2000, Kerberos has replaced the Windows NT domain authentication mechanism

# Roadmap

- The simplified architecture
- The complete architecture
  - Pre-authentication
  - Delegation
    - proxiable tickets
    - forwardable tickets
  - Realms



# Scenario




- Server only allows authorized accesses
- Server authenticates service requests

# Client authentication

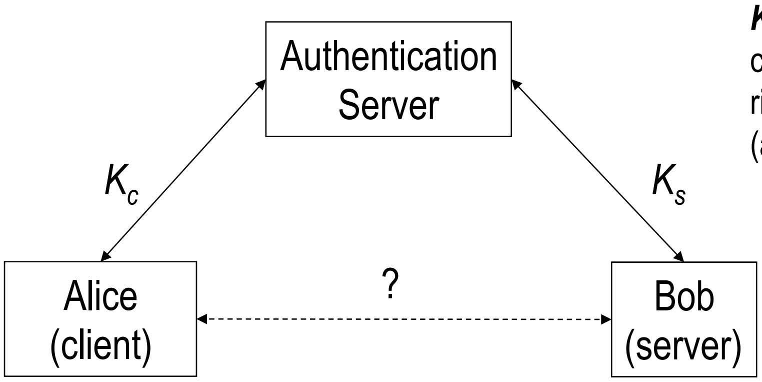
1. Server trust workstation for user authentication. The server applies a policy base on UID (closed system)
2. Similar to point 1 but the server authenticates the WSs (closed system)
3. Server requires the user to provide a proof of idenity for each service request, and vice versa (open system)



# Objectives

- **Security**
  - Eavesdropping and spoofing must be non possible for an outsider
- **Availability**
  - If Kerberos is unavailable all the services become unavailable
- **Transparency**
  - The authentication process must be transparent but password typing
- **Scalability**
  - Kerberos must handle a large number of servers and users

# Kerberos: architettura di base




***$K_c$*** e ***$K_s$*** (***master key***) sono segreti condivisi tra AS e client e server, rispettivamente
(ad esempio derivati da password)

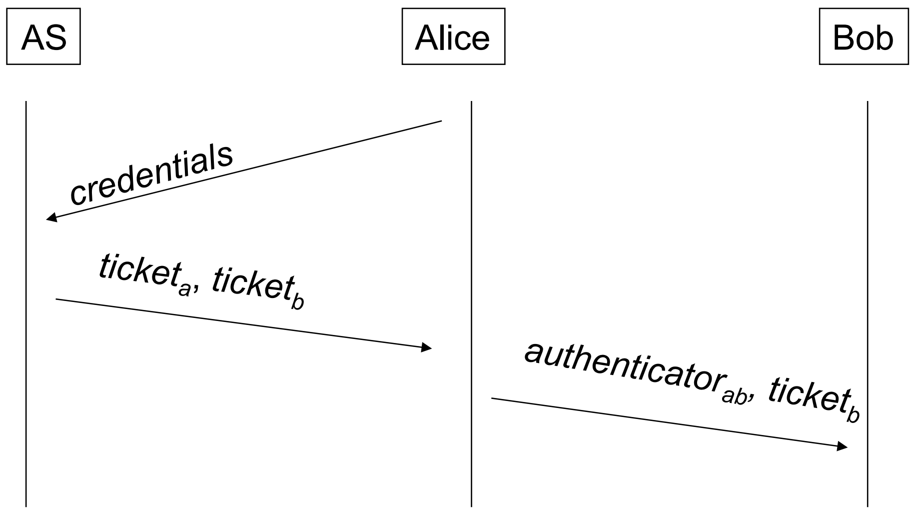
**Obiettivo primario**: autenticazione mutua di client e server

**Obiettivi secondari**:

- Stabilire una chiave condivisa tra client e server
- Provare al server che il client è attivo e viceversa
- …



# The basic idea




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

# Kerberos V (simpl.)

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



# Kerberos V

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

- ***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

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

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

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

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

$$\boxed{\begin{array}{l} A \mid\equiv \#(t) \\ B \mid\equiv \#(t) \quad B \mid\equiv \#(t_a) \end{array}}$$

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

dopo il messaggio M2

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

dopo il messaggio M3

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

*key authentication*

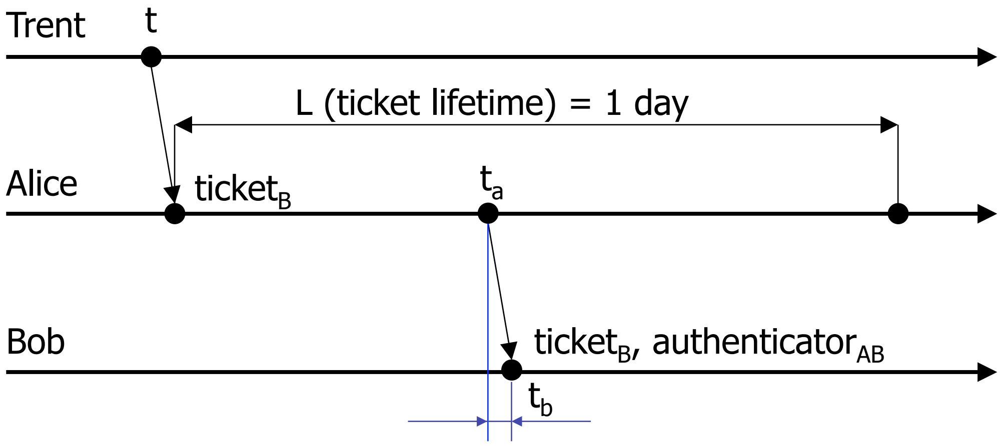
$$B \mid\equiv A \mid\equiv A \overset{K_{ab}}{\leftrightarrow} B$$

dopo il messaggio M4

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

*key confirmation*

# Lifetime and authenticator




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



# Comments

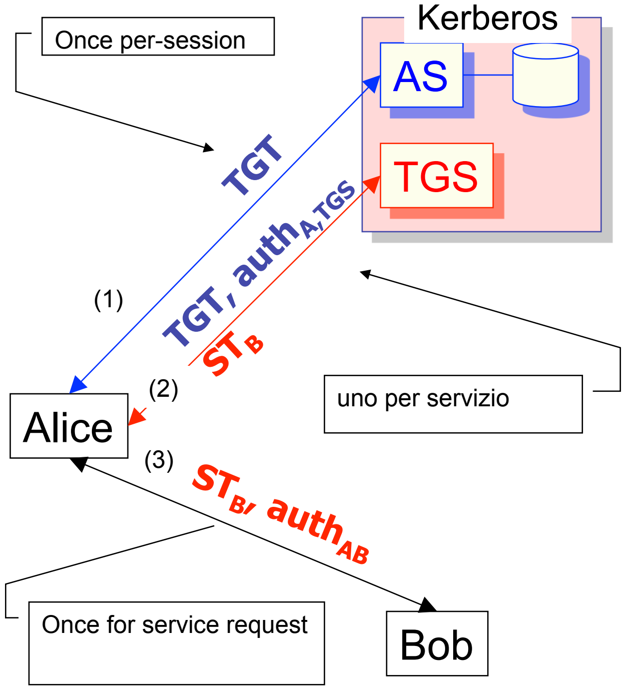
- Kerberos requires synchronized clocks
- In Kerberos 5, λ= 5 minutes:
  - Authenticator may be replayed in that window
- If Ka and Kb derive from a pwd, then they are as secure as the pawd

# 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, Cambridge)

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

**Phase 2**

Alice interacts with **TGS** and receives $\mathbf{ST_b}$, ***service ticket,*** a ticket for server **B**

**Phase 3**

Alice uses $\mathbf{ST_b}$ 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

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

# 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*

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

*Service Ticket for Bob*

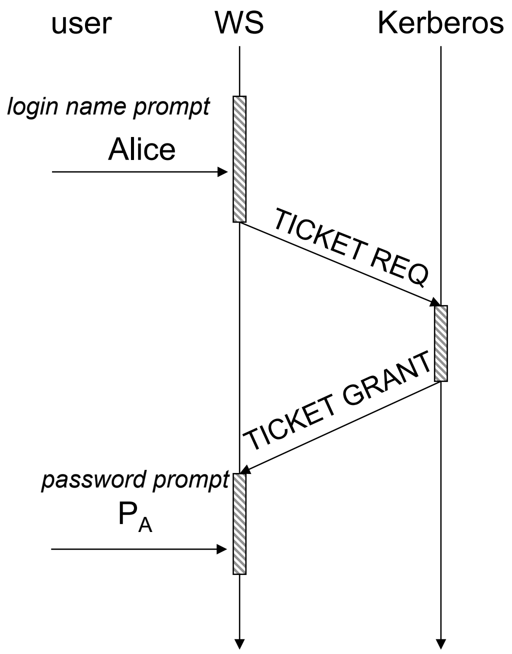
*Service Ticket for Alice*

# Authentication

- Kerberos authenticates users w.r.t. network services
- Kerberos **does not** authenticate users w.r.t. AS
  - Anyone may ask 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
- 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;

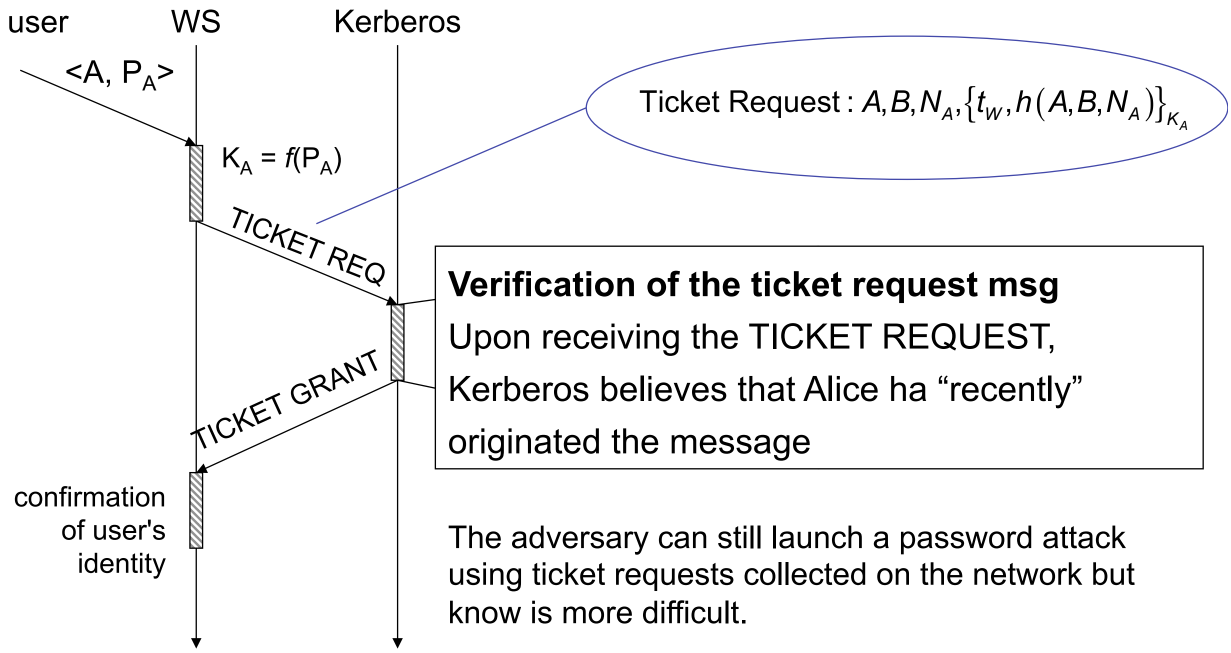


# Normal authentication




- WS does not authenticate Alice but the ticket is encrypted.
- However, an adversary can collect tickets (on demand) and use them to launch a pwd attack (known plaintext attack)

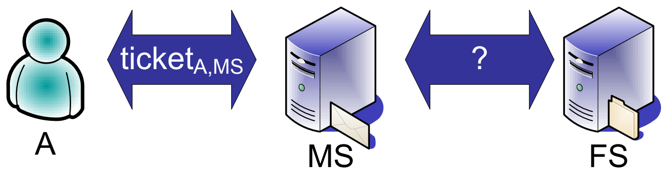
# Kerberos 5 pre-authentication




The adversary can still launch a password attack using ticket requests collected on the network but know is more difficult.



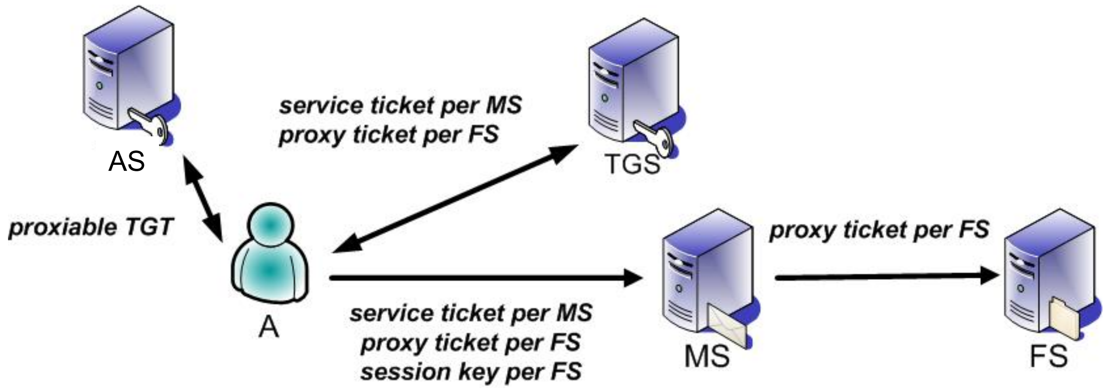
# Delegation




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

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

Proxy ticket for FS required by A, released to MS

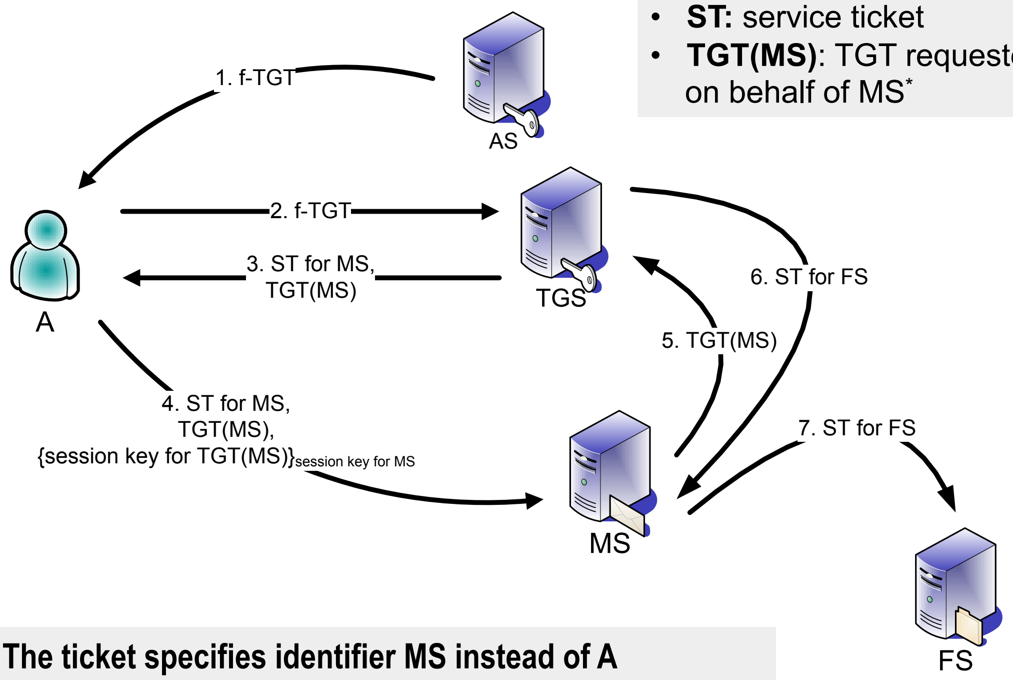


# Proxy ticket: cons

- 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
     $\{\ldots, TK', \ldots\}_{Ka}, \{\ldots, TK', MS, \ldots\}_{Ktgs}$.
- **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

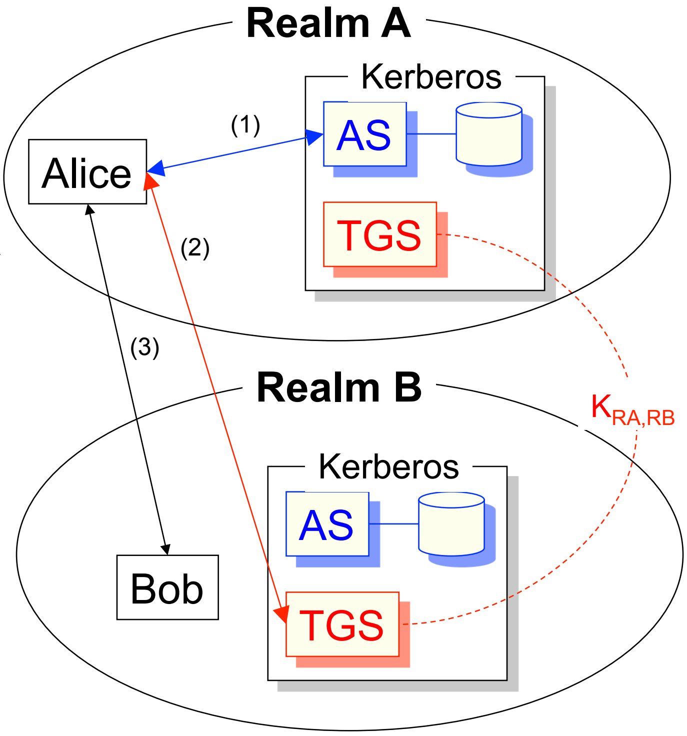


# Limitations to delegations

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

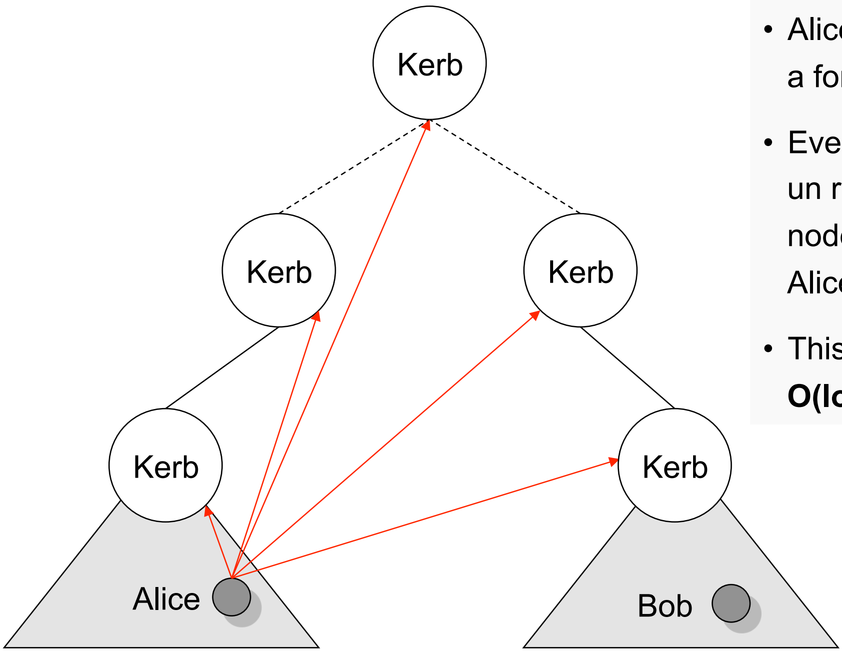
# Realms e referral tickets

- Users and servers may belong to different administrative domains (**realms**)
- Kerberos authenticates principals in its realm
- Problem. Alice has to authenticate a **foreign** server Bob (in another realm)
  - Alice asks Kerberos A a ***referral TGT*** for realm B (1).
  - Kerberos A generates a **referral TGT** using an ***inter-realm key*** ($K_{RA,RB}$)
  - 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 foreign 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

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



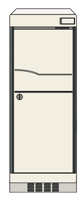
# Clock synchronization

Adversary has an old key and the related ticket

Time server


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

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

Alice holds *certificate*$_T$

W2K encapsulates PKINIT in its Kerberos-based authentication environment