UnitelmaSapienza
Università degli Studi di Roma

Sapienza
Università di Roma
Dipartimento di Informatica

KERBEROS

T16.2 Herakles, Kerberos, Hekate

Francesco Parisi Presicce

Kerberos Authentication Service

- Developed at MIT under Project Athena in mid 1980s
- Versions 1-3 were for internal use; versions 4 and 5 are being used externally
- Version 4 has a larger installed base, is simpler, and has better performance, but works only with TCP/IP networks
- Version 5 developed in mid 90's (RFC-1510) corrects some of the security deficiencies of Version 4
- krb5-1.18.3 released on 17 November 2020
- Kerberos (intended) Services:
    - Authentication
    - Accounting
    - Audit
  - The last two were never implemented

Objective

To provide a trusted third-party service (based on the Needham/Schroeder authentication protocol), named Kerberos, that can perform authentication between any pair of entities in TCP/IP networks

- primarily used to authenticate user-workstation to server
- Authentication is two-way
- Not meant for high risk operations (e.g., bank transactions, classified government data, student grades)

Needham-Schroeder Protocol

- original third-party key distribution protocol, for session between A and B mediated by KDC
- protocol overview is:
  1. A → KDC: $ID_A \,||\, ID_B \,||\, N_1$
  2. KDC → A: $E_{Ka}[Ks \,||\, ID_B \,||\, N_1 \,||\, E_{Kb}[Ks||ID_A]]$
  3. A → B: $E_{Kb}[Ks||ID_A]$
  4. B → A: $E_{Ks}[N_2]$
  5. A → B: $E_{Ks}[f(N_2)]$

Physical Security

CLIENT WORKSTATIONS

- None, so cannot be trusted

SERVERS

- Moderately secure rooms, with moderately diligent system administration

KERBEROS

- Highly secure room, with extremely diligent system administration

Design Goals

Impeccability

- No cleartext passwords on the network
- No client passwords on servers (server must store secret server key)
- Minimum exposure of client key on workstation (smartcard solution would eliminate this need)

Containment

- Compromise affects only one client (or server)
- Limited authentication lifetime (8 hours, 24 hours, more)

Transparency

- Password required only at login
- Minimum modification to existing applications

Kerberos Model

- Network consists of clients and servers
  - clients may be users, or
  - programs that can, e.g., download files, send messages, access databases and access printers
- Kerberos keeps a database of clients and servers with a secret key for each one (selected at the time of registration)
  - O(n+m) keyspace, instead of O(nm) keyspace with n clients and m servers
- Kerberos provides authentication of one entity to another and issues session key
- Issues tickets for access rights
  - temporary rights issued by authentication server
  - tickets time-stamped to reduce replay attacks

Where to start

- Every principal has a master (secret) key
  - Human user's master key is derived from the password
  - Other resources must have their keys configured in
- Every principal is registered with the Kerberos server AS
- All principals' master keys are stored in the AS database (encrypted using the AS master key)

Encryption and clock

Note:

- Each user has a password which is converted to a DES key
- Client and server do <u>not</u> initially share an encryption key
- Any symmetric key system would work

Clocks

- All machines that use Kerberos are loosely synchronized (within a few minutes) to prevent replays

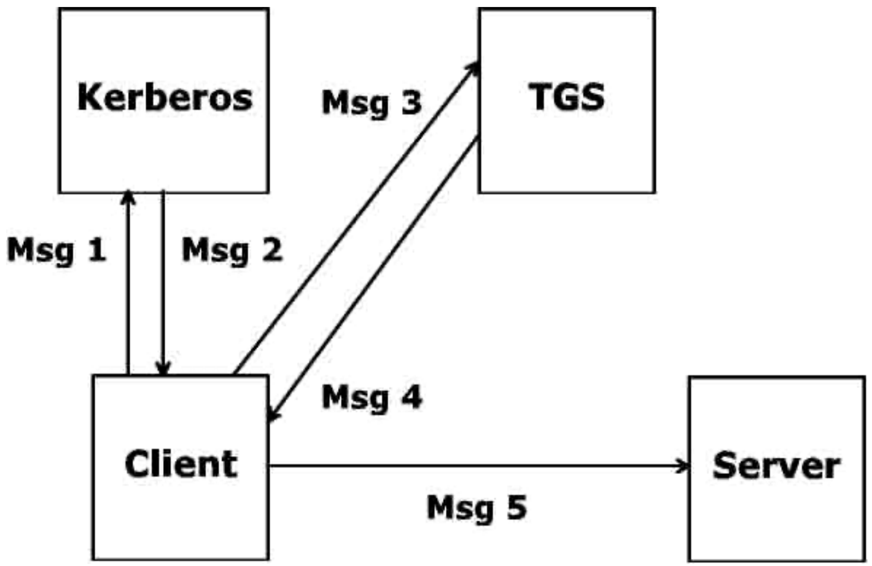
Kerberos Components

- Key Distribution Center (KDC) – consists of two logical components:
  - **Kerberos Database** — with secret key for each principal (user or service)
  - **Authentication Service (AS)** — uses the Kerberos database to verify the identity of users requesting the use of network services
- Ticket Granting Server (TGS) — issues tickets to clients for communicating with network servers after the AS has verified the identity of the client

Kerberos Operation

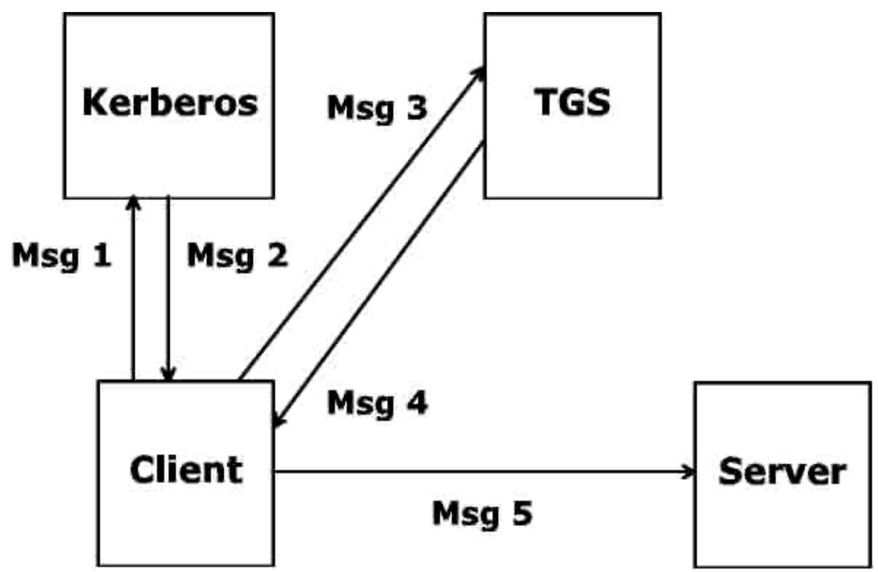
The Kerberos protocol is simple and straightforward.

- First, the Client requests a ticket for a Ticket-Granting Service (TGS) from Kerberos (**Msg 1**).
- This ticket is sent to the client encrypted using the client's secret key (**Msg 2**).
- To use a particular server, the client requests a ticket for that server from the TGS (**Msg 3**).



Kerberos Operation

- If everything is in order, the TGS sends back a ticket to the client for the server (**Msg 4**).
- At this point the client presents this ticket to the server along with an authenticator (**Msg 5**).
- If there is nothing wrong with the client’s credentials, the server permits access to the service.



Getting an Initial Ticket

- When Bob logs into a workstation (WS), WS sends Bob's user id to AS in the clear
- AS returns to the WS, encrypted with Bob's secret key K_{Bob} :
  - A session key $K_{Bob,TGS}$ (a secret key to be used during the current session)
  - A ticket-granting ticket (TGT) containing the session key, the user id, and an expiration time, encrypted with K_{TGS}

Getting an Initial Ticket

- After receiving the message from AS, WS prompts Bob for his password and uses it to derive Bob's secret key K_{Bob}
- Bob's secret key is then used to decipher the session key $K_{Bob,TGS}$ and the TGT
- WS discards both Bob's password and his secret key

Note that

- When Bob requires access to a service (Alice), WS will need to send the TGT to TGS.
- Bob cannot read the contents of the TGT encrypted with TGS secret key.
- Since TGT contains all the information TGS needs about the initial login session, Kerberos can be stateless.

Getting a Server Ticket

- When Bob wants to access a service (Alice), WS sends to TGS the name Alice, and an authenticator which proves that WS knows the session key
- Authenticator consists of the time of day encrypted with the session key (in this case $K_{Bob,TGS}$)
- TGS decrypts the TGT to obtain $K_{Bob,TGS}$, and verifies the timestamp (times can be off by some amount). If so, TGS generates a new session key $K_{Bob, Alice}$ (session key to be shared by Bob and Alice), finds Alice's master key, and sends to WS a "ticket for Alice" and $K_{Bob, Alice}$, encrypted with the session key $K_{Bob,TGS}$
- The "ticket for Alice" consists of Bob's identity, an expiration time, and $K_{Bob, Alice}$ encrypted using Alice's master key

Requesting a Service

- Upon receiving the message from TGS, WS decrypts the message using $K_{Bob,TGS}$
- WS sends the “ticket for Alice” (that it cannot read) and an authenticator to Alice
- Alice uses K_{Alice} to decrypt the ticket to obtain $K_{Bob,Alice}$ and decrypts the authenticator using $K_{Bob,Alice}$ to verify the timestamp
- If everything checks out, Alice knows that the message is from Bob

Use of Session key

Kerberos establishes a session key $K_{Bob,Alice}$ to be used by the applications for

- client to server authentication (no additional step required in the protocol)
- mutual authentication: it requires the additional step of sending another message from server to client { f($A_{Bob, Alice}$) } $K_{Bob, Alice}$, using some known (hash) function f
- message confidentiality using $K_{Bob, Alice}$
- message integrity using $K_{Bob, Alice}$

Kerberos Version 4

Terms:

- C = Client
- AS = authentication server
- V = server
- ID_C = identifier of user on C
- ID_V = identifier of V
- AD_C = network address of C
- K_V = secret encryption key shared by AS and V
- $K_{C,V}$ = secret encryption key shared by C and V
- TS = timestamp
- || indicates concatenation

How Kerberos works

- Kerberos uses two types of credentials
  - tickets (to convey keys and identity)
  - authenticators (to verify ‘identity’)

$Ticket_{tgs} = E_{Ktgs}\,[K_{c,tgs} \,||\, ID_c \,||\, AD_c \,||\, ID_{tgs} \,||\, TS \,||\, Life\,]$

$Authenticator_c = E_{Kc,tgs}\,[\,ID_c \,||\, AD_c \,||\, TS\,]$

- A client uses a ticket (that he/she cannot read or modify) to access a server
  - It can be used multiple times until it expires
- A client generates an authenticator to use a service on the server (once only)

V4 Authentication Dialogue

Authentication Service Exhange: To obtain Ticket-Granting Ticket

- (1) C → AS:

  $ID_c \,||\, ID_{tgs} \,||TS1$

- (2) AS → C:

$E_{Kc}\,[K_{c,tgs} || \, ID_{tgs} \,||\, TS_2 \,||\, Lifetime_2 \,||\, Ticket_{tgs}]$

V4 Authentication Dialogue

Ticket-Granting Service Exchange: To obtain Service-Granting Ticket

- (3) C → TGS:

  $ID_V \,||\, Ticket_{tgs} \,||\, Authenticator_C$

- (4) TGS → C:

  $E_{Kc,tgs}\,[K_{c,v} \,||\, ID_V \,||\, TS4 \,||\, Ticket_V]$

V4 Authentication Dialogue

Client/Server Authentication Exhange: To Obtain Service

- (5) C → V:

  $Ticket_V \,||\, Authenticator_C$

- (6) V → C:

  $E_{Kc,V}[TS5 + 1]$

Replicated Kerberos Servers

- To avoid single point of failure and performance bottleneck, it is possible to replicate Kerberos server
- Mutual consistency of copies of password database could be maintained as follows:
  - All updates are made to a primary (master) copy
  - Other (slave) copies are read only; these copies are replaced periodically by downloading the master copy
  - The database (with encrypted keys) is transferred in the clear
  - To ensure that an attacker has not rearranged data in transit, a cryptographic checksum is also exchanged
  - To ensure that an attacker does not replace a copy by an older copy, a timestamp is also sent

Kerberos V4 Realm

A full-service Kerberos environment consists of the following entities:

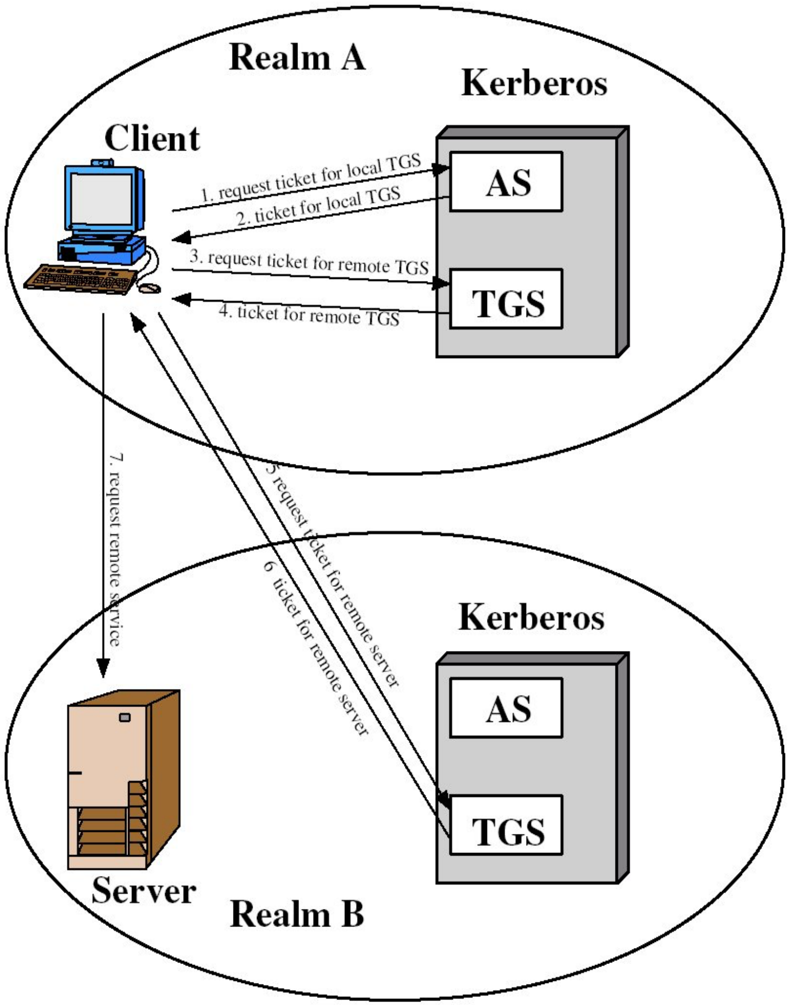
- A Kerberos server
- A set of one, or more, clients
- A set of one, or more, application servers

This environment is known as a realm.

- Networks of clients and servers under different administrative organizations typically constitute different realms.

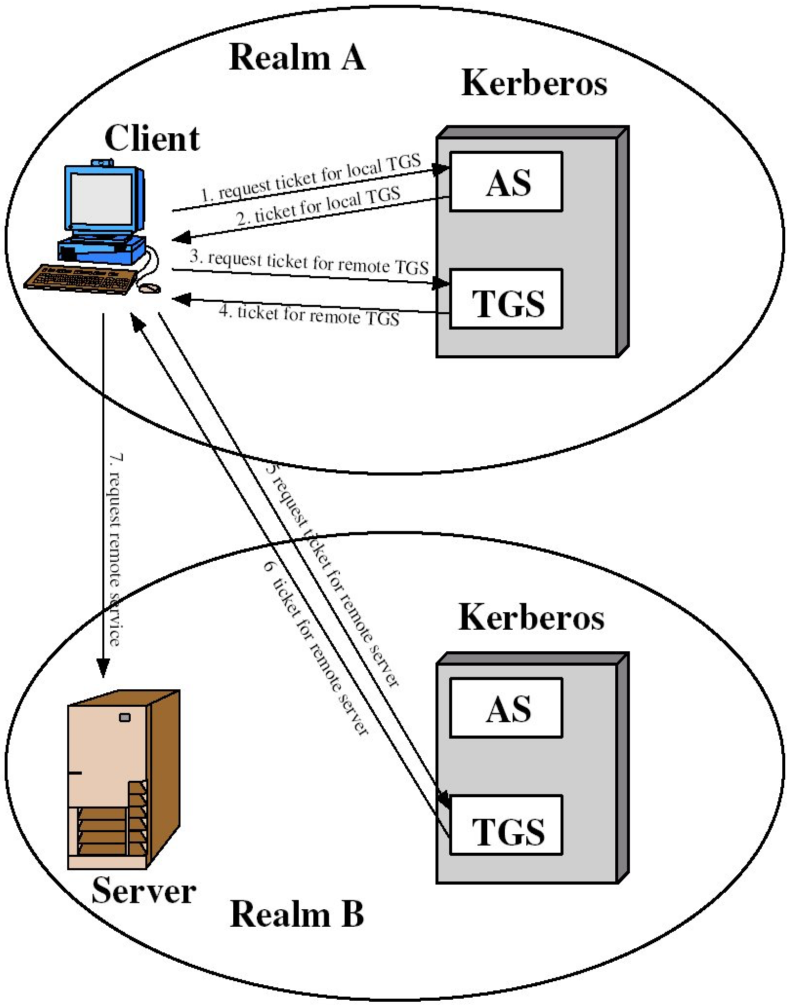
Cross-Realm Operation

- The Kerberos protocol is designed to operate across organizational boundaries: a client in one organization can be authenticated to a server in another.
- Each organization wishing to run a Kerberos server establishes its own "realm".
- The name of the realm in which a client is registered is part of the client's name, and can be used by the end-service to decide whether to honor a request.



Cross-Realm Operation

- By establishing "inter-realm" keys, the administrators of two realms can allow a client authenticated in the local realm to use its authentication remotely.
- With appropriate permissions, a client could arrange registration of a separately-named principal in a remote realm, and engage in normal exchanges with that realm's services.



Cross-Realm Operation: message exchange

- Typically, cross-realm message exchange operates as follows:

  C → AS:

$ID_C \,||\, ID_{tgs} \,||\, TS_1$

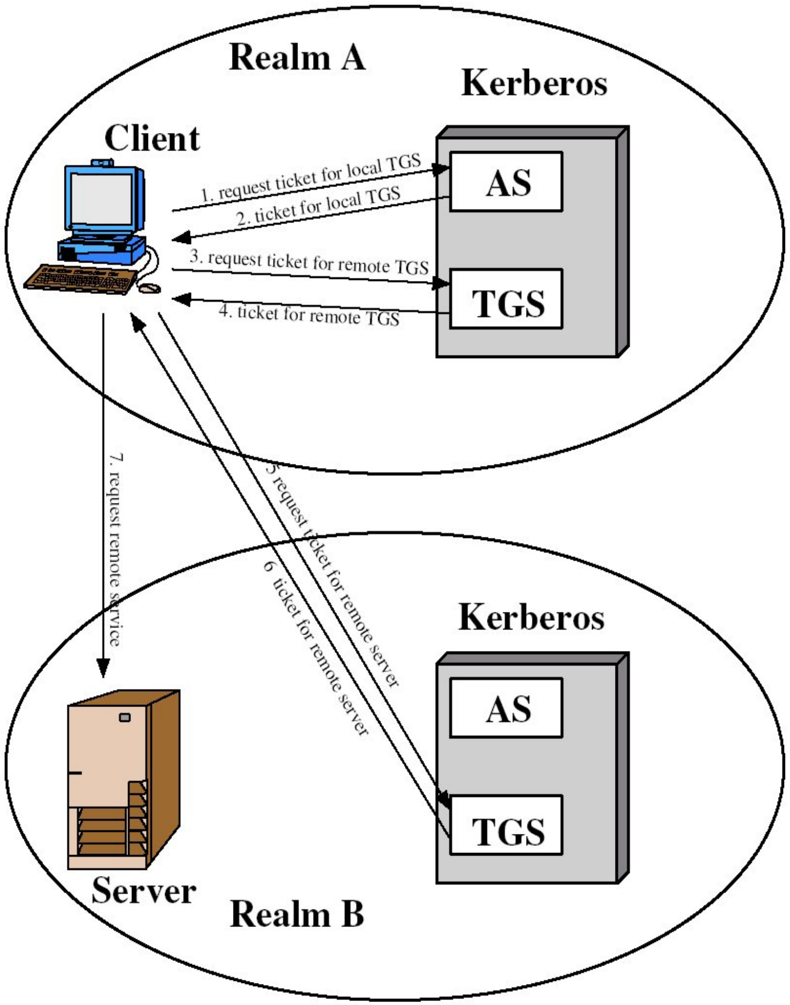
  AS → C:

$E_{KC}\,[K_{C,\,tgs} \,||\, ID_{tgs} \,||$

  $TS_2 \,||\, Lifetime_2 \,||\, Ticket_{tgs}]$

  C → TGS:

$ID_{tgsrem} \,||\, Ticket_{tgs} \,||\, Authenticator_C$



Cross-Realm Operation: message exchange

TGS → C:

$E_{Kc,tgs}$ [$K_{C,\ tgsrem}$ ||
ID_{tgsrem} || TS_4 || $Ticket_{tgsrem}$]

C → TGS_{rem}:

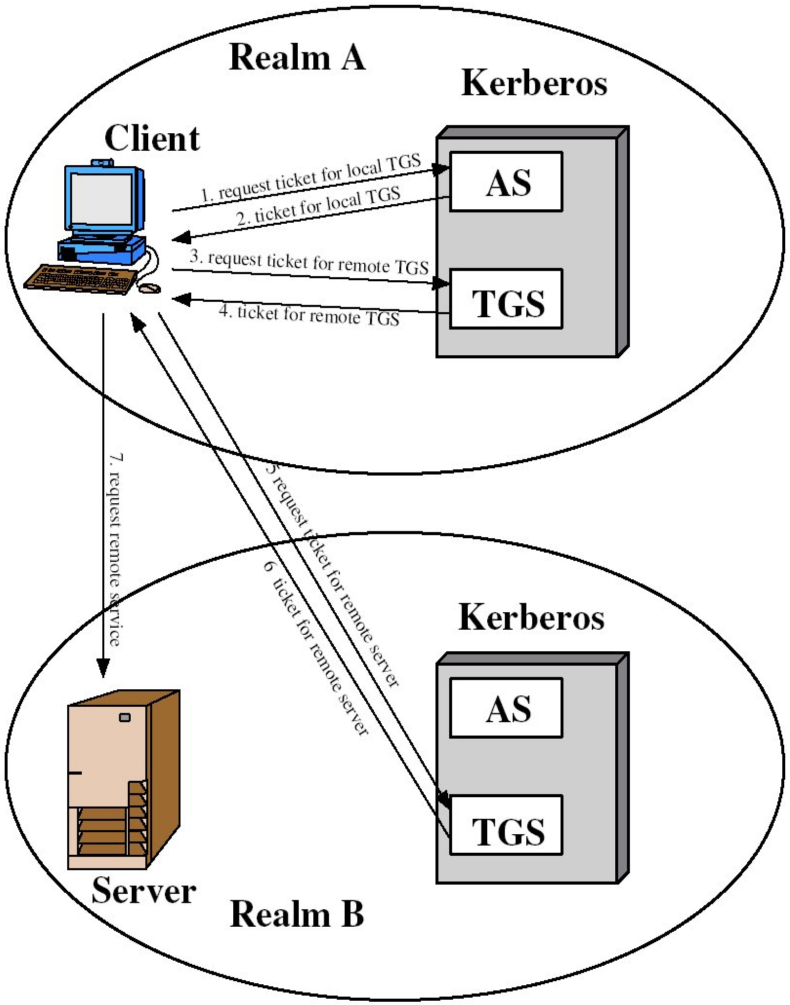
ID_{vrem} || $Ticket_{tgsrem}$ || $Authenticator_C$

TGS_{rem} → C:

$E_{Kc,tgsrem}$ [$K_{c,\ vrem}$ ||
ID_{vrem} || TS_6 || $Ticket_{vrem}$]

C → V_{rem}:

$Ticket_{vrem}$ || $Authenticator_C$



Kerberos V5 vs. V4

addresses environmental shortcomings

- encryption system dependence (only DES)
- internet protocol dependence (only IP addresses)
- byte order (sender’s choosing + tag)
- ticket lifetime (only 8bit of 5 min units = 21 hrs)
- authentication forwarding (not allowed)
- Inter-realm authentication (n^2 relationships in V4, fewer in V5)

Kerberos V5 vs. V4

and technical deficiencies

- double encryption (of ticket= not necessary)
- non-std mode of DES Propagating CBC (now CBC DES for encryption and separate integrity checks)
- session keys (used too often: now sub-session keys)
- password attacks (still possible)

Kerberos V5 Realm

For a realm to function, it requires the following:

- The Kerberos server must have the user ID (UID) and hashed password of all participating users in its database.
  - All users are registered with the Kerberos server.
- The Kerberos server must share a secret key with each server.
  - All servers are registered with the Kerberos server.

Kerberos V5 Multiple Realms

- Kerberos provides a mechanism for support multiple realms and inter-realm authentication.
- Inter-realm authentication adds the following third requirement:
  - The Kerberos server in each inter-operating realm share a secret key with the server in the other realm.
    - The two Kerberos servers are registered with each other.
- This inter-realm scheme requires that the Kerberos server in one realm trust the Kerberos server in the other realm to authenticate its users.
  - In a similar fashion, the participating servers in the second realm must also be willing to trust the Kerberos server in the first realm.

Realms: Hierarchical Organization

- Realms are typically organized hierarchically.
  - Each realm shares a key with its parent and a different key with each child.
- If an inter-realm key is not directly shared by two realms, the hierarchical organization allows an authentication path to be easily constructed.
- If a hierarchical organization is not used, it may be necessary to consult some database in order to construct an authentication path between realms.

Kerberos V5 Credentials: Ticket

- A Kerberos ticket used to pass to server identity of client for whom the ticket was issued.
    - also contains information that server uses to ensure that client using ticket is same client to whom ticket was issued.
- *Some* of the information, encrypted using the server's secret key, in a ticket include
    - Client's name
    - Client's network address
    - Timestamp
    - Session key
- A ticket is good for a single server and a single client; it can, however, be used multiple times to access a server — until the ticket expires.
- Ticket security is assured since its critical elements are encrypted using the server's secret key.

Kerberos V5 Tickets

Kerberos version 5 tickets are renewable, so service can be maintained beyond maximum ticket lifetime.

Ticket can be renewed until minimum of:

- requested end time
- start time + requesting principal's max renewable lifetime
- start time + requested server's max renewable lifetime
- start time + max renewable lifetime of realm

Kerberos V5 Authenticator

- A Kerberos authenticator is generated each time a client wishes to use a service on a server.
- *Some* of the information, encrypted using the key between the client and the server, in an authenticator includes:
  - Client’s name
  - Timestamp
  - Session key
- Unlike a ticket, an authenticator can be used only once.
  - However, a client can create authenticators as needed.

Kerberos V5 Ticket Flags

The flags field was added in Kerberos V5.

- The standard defines 11 flags

- INITIAL
- INVALID
- RENEWABLE
- POSTDATED
- PROXIABLE
- PROXY
- FORWARDABLE

Kerberos in Windows

- Authentication protocol of choice in Windows.
- Windows domains correspond to Kerberos realms; domain controllers act as KDCs.
- Kerberos principals are users and machines.
- Windows authentication is the basis for access control; principals in Windows access control: SID.
  - Note that there are two definitions of principal
- Kerberos ticket contains mandatory field *cname* (client name) and optional field *authorization-data*.
- Windows: *cname* holds principal's name and realm, e.g. d.duck@uniroma1.it, *authorization-data* holds the group SIDs.

Limitations of Kerberos

- It is possible to cache and replay old authenticators during the lifetime (typically 8 hours) of the ticket
- If a server can be fooled about the correct time, old tickets can be reused
- Vulnerable to password guessing attacks (attacker collects tickets and does trial decryptions with guessed passwords)
- Active intruder on the network can cause denial of service by impersonation of Kerberos IP address

Not Addressed by Kerberos V5

- "Denial of service" attacks are not solved with Kerberos.
  - There are places in these protocols where an intruder can prevent an application from participating in the proper authentication steps.
- Principals must keep their passwords (used to generate the secret keys) or secret keys secret.
  - If an intruder steals a principal's key, can masquerade as that principal or impersonate any server to the legitimate principal.

Not Addressed by Kerberos V5

- "Password guessing" attacks are not solved by Kerberos.
  - If a user chooses a poor password, it is possible for an attacker to successfully mount an offline dictionary attack by repeatedly attempting to decrypt, with successive entries from a dictionary, messages obtained which are encrypted under a key derived from the user's password.

Kerberos V5 availability

- Kerberos is not in the public domain, but MIT freely distributes the code.
  - Integrating it into the UNIX environment is another story.
- A number of companies sell versions of Kerberos
- Microsoft has incorporated it into the Windows 2000 Server product line. (http://www.sans.org/rr/win2000/kerberos.php)
- Banned for export by US government until 2000 (due to use of DES), reimplemented at KTH in Sweden
- Now it supports AES as main encryption

Additional references

- S. M. Bellovin and M. Merritt, "Limitations of the Kerberos Authentication System," Proc. USENIX, Winter 1991.
- B. C. Neuman and T. Ts'o, "Kerberos: An authentication service for computer networks," IEEE Communications, September 1994, pp. 33-38.