Key distribution and certification

- In the case of public key encryption model the authenticity of the public key of each partner in the communication is ensured by a **Certification Authority (CA)**

- In the case of symmetric key encryption the authenticity of the common key is ensured by a **Key Distribution Center (KDC)**

- Problem solution**: trusting autority**.

•.

Kerberos

- The Kerberos protocol is designed to provide reliable authentication over open and insecure networks where communications between the hosts belonging to it may be intercepted .

- The technique was developed at MIT in the 1980. It represents an authentication service based on the symmetric key encryption and on a Key Distribution Center (KDC) which is a trusted third part.

- KDC consists of two parts logically separeted **Authentication Server (AS)** and **Ticket Granting Server (TGS).**

- **AS is responsible for handling a login request from a user.** The AS maintains a database of secret keys; each entity on the network — whether a client or a server — shares a secret key known only to itself and to the AS. Knowledge of this key serves to prove the entity's identity.

- Setting up secure channels is handled by **TGS.** TGS hands out special messages, known as **tickets**, that are used to convince a server that the client is really who he claims to be.

- A ticket is an unforgeable, non replayable, authenticated object. It is an encrypted data structure naming a user and a service that the user is allowed to obtain. It also contain a time value and some control information.

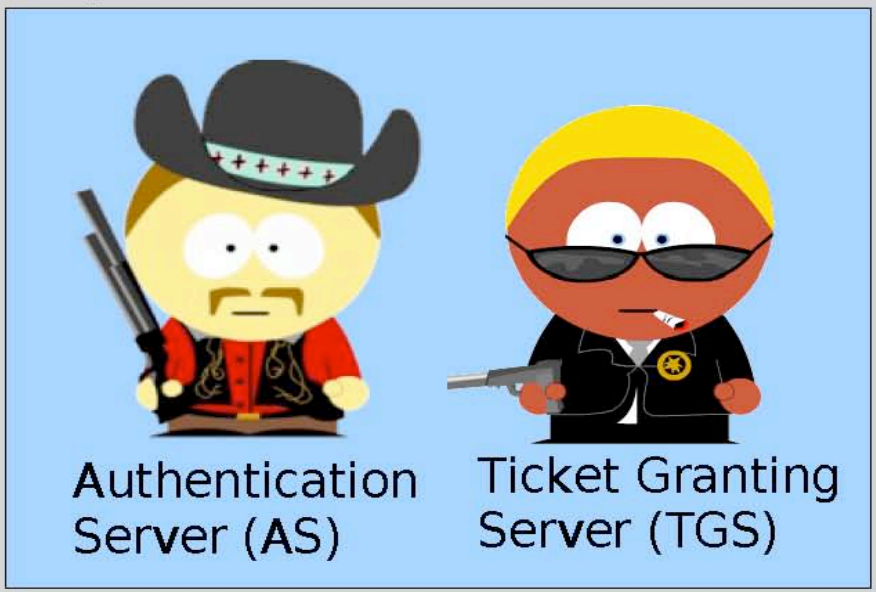
- The following is an *intuitive description*. The client (Alice) authenticates itself to the Authentication Server and receives a ticket. (All tickets are time-stamped.)

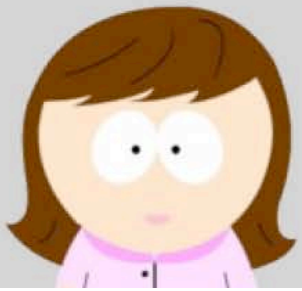
- It then contacts the Ticket Granting Server, and using the ticket it demonstrates its identity and asks for a service. If the client is eligible for the service, then the Ticket Granting Server sends another ticket to the client.

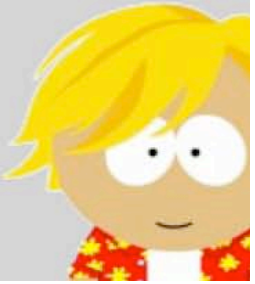
- The client then contacts the Service Server, and using this ticket it proves that it has been approved to receive the service.

Single Sign On & Software Libero

Key Distribution Center (KDC)

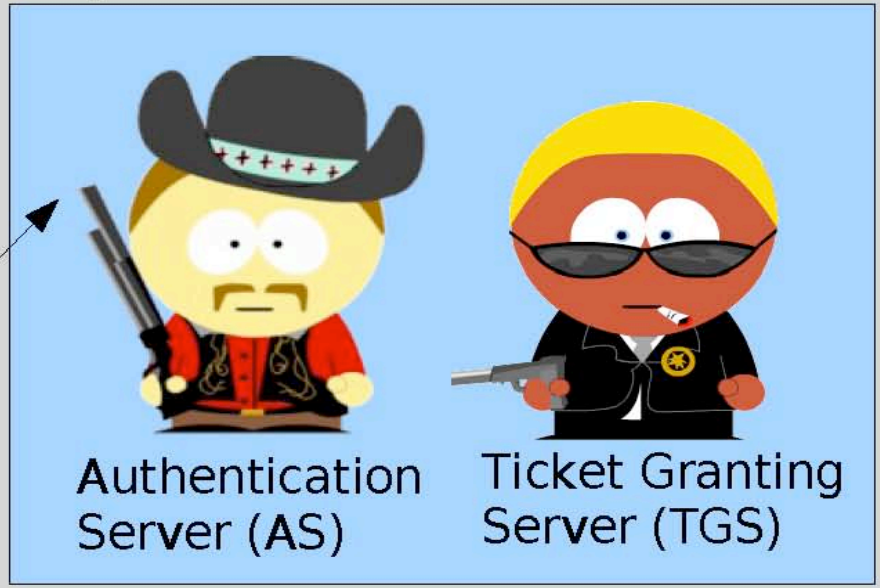






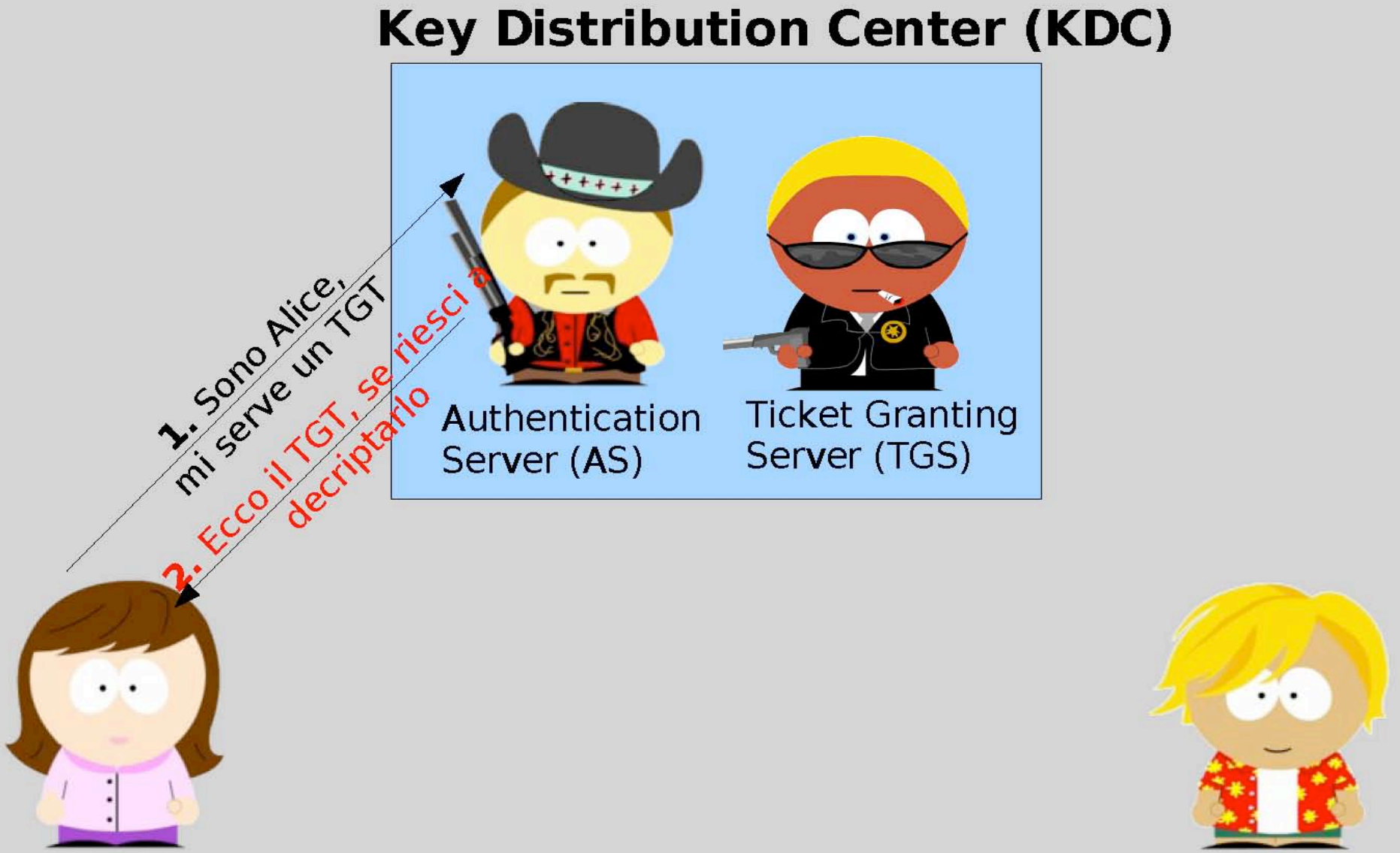
Single Sign On & Software Libero

Key Distribution Center (KDC)

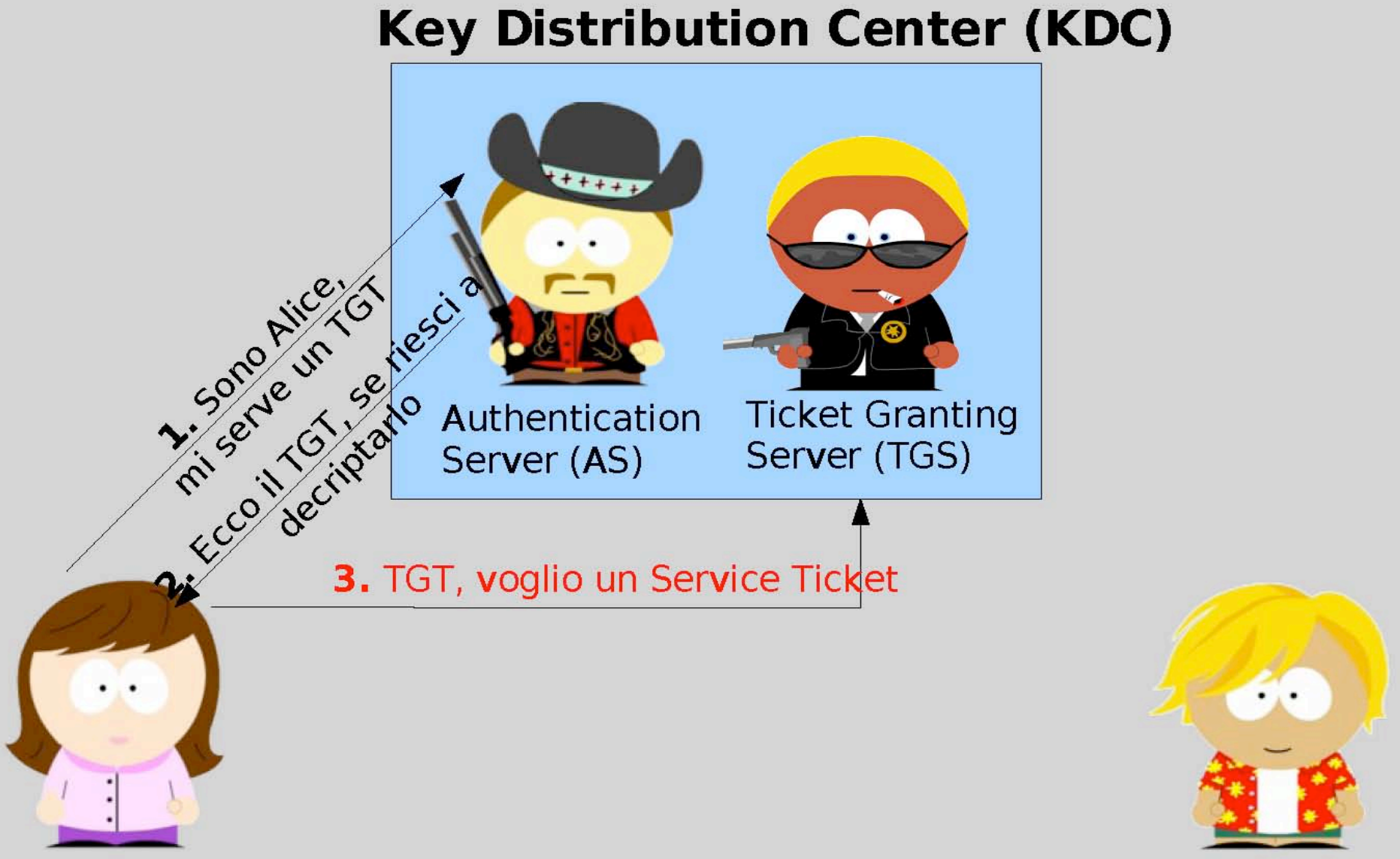


1. Sono Alice, mi serve un TGT

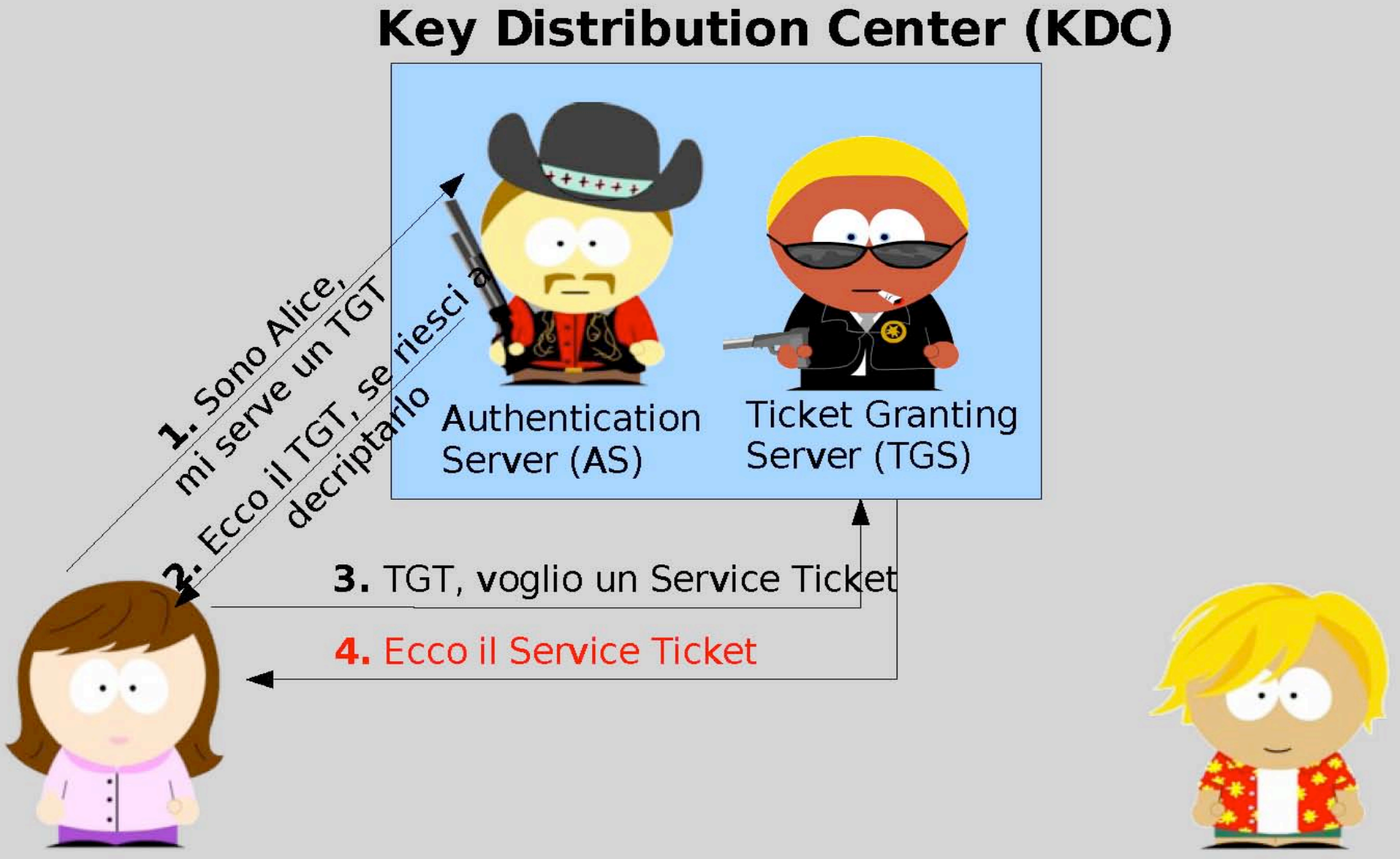
Single Sign On & Software Libero



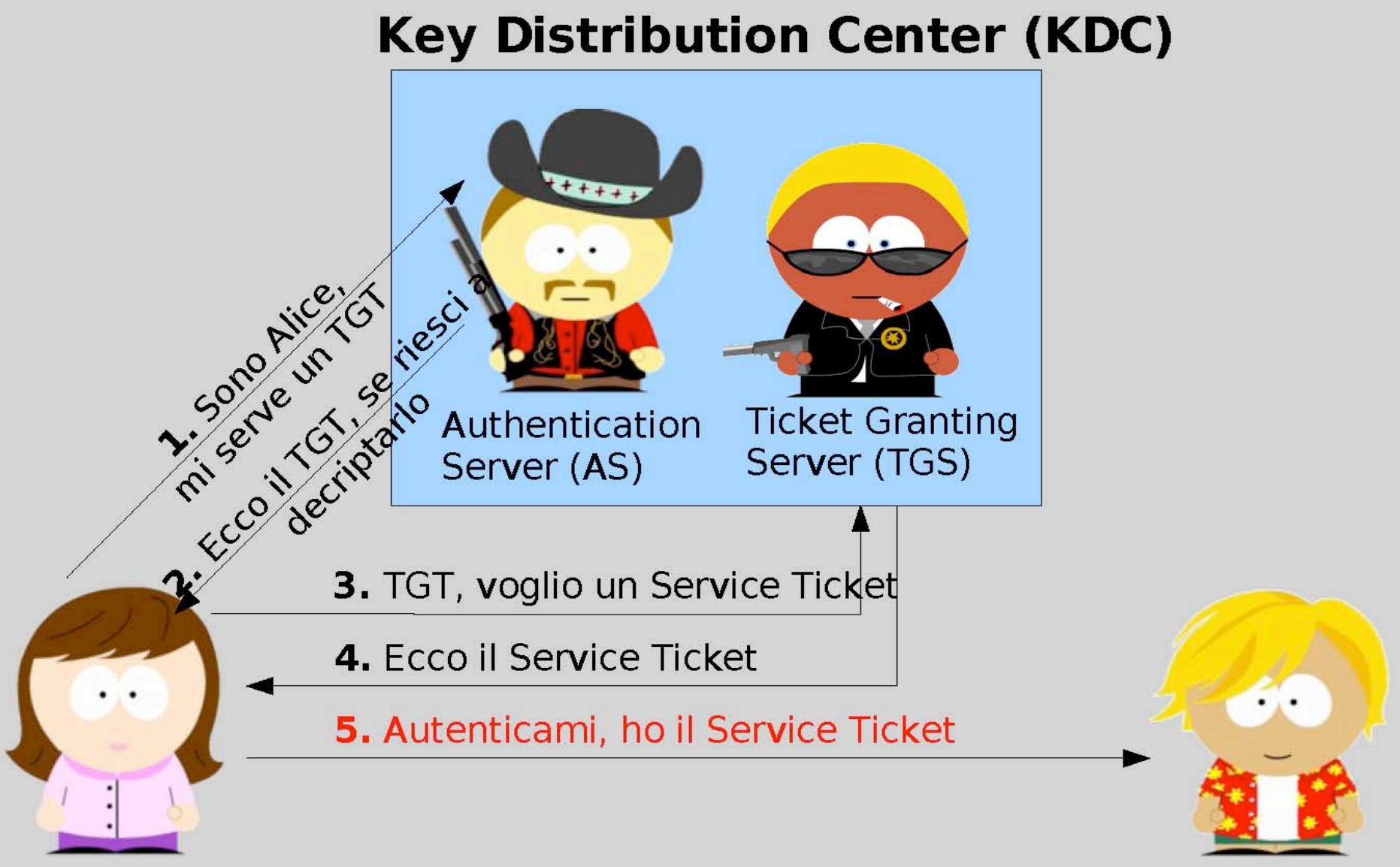
Single Sign On & Software Libero



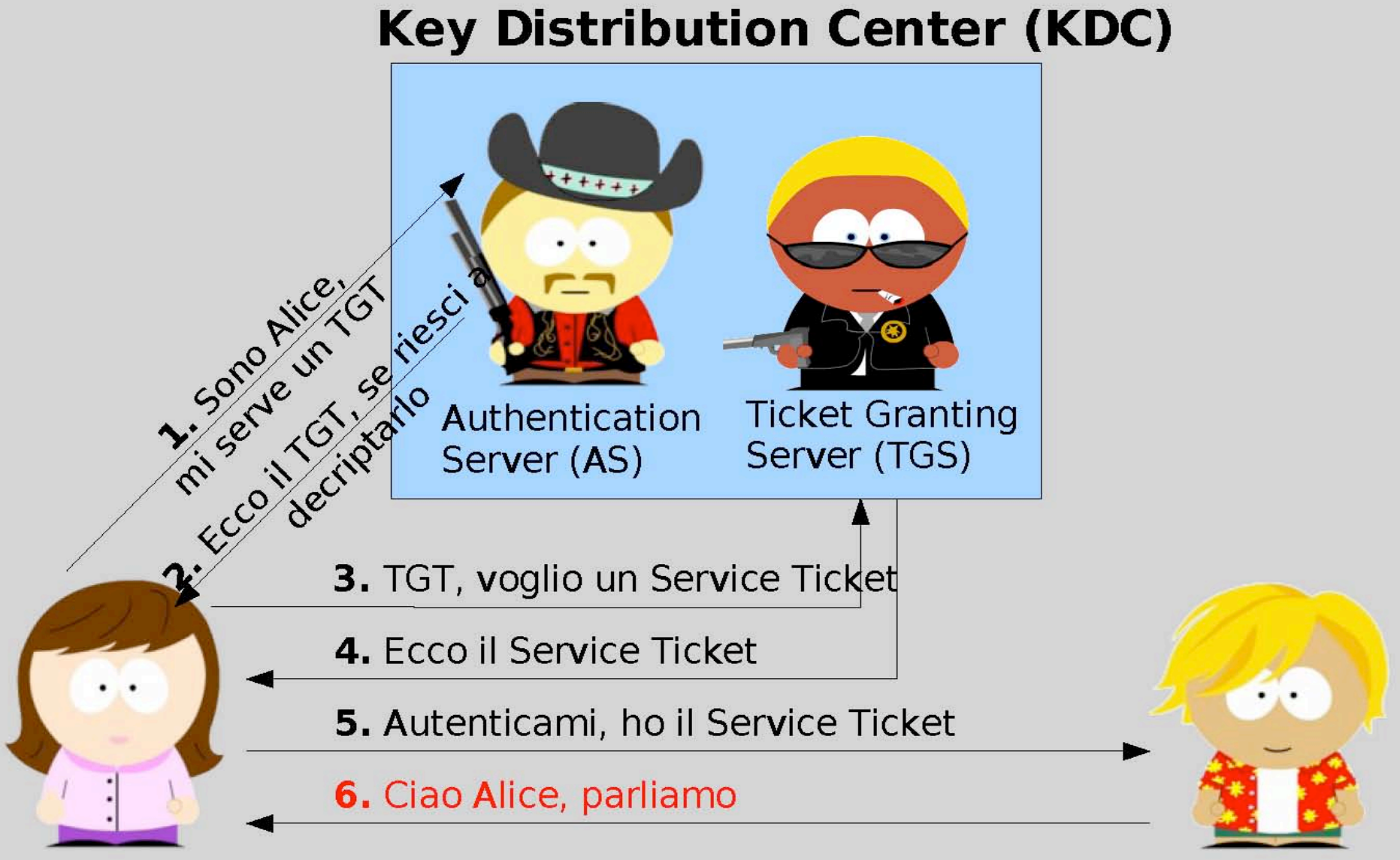
Single Sign On & Software Libero



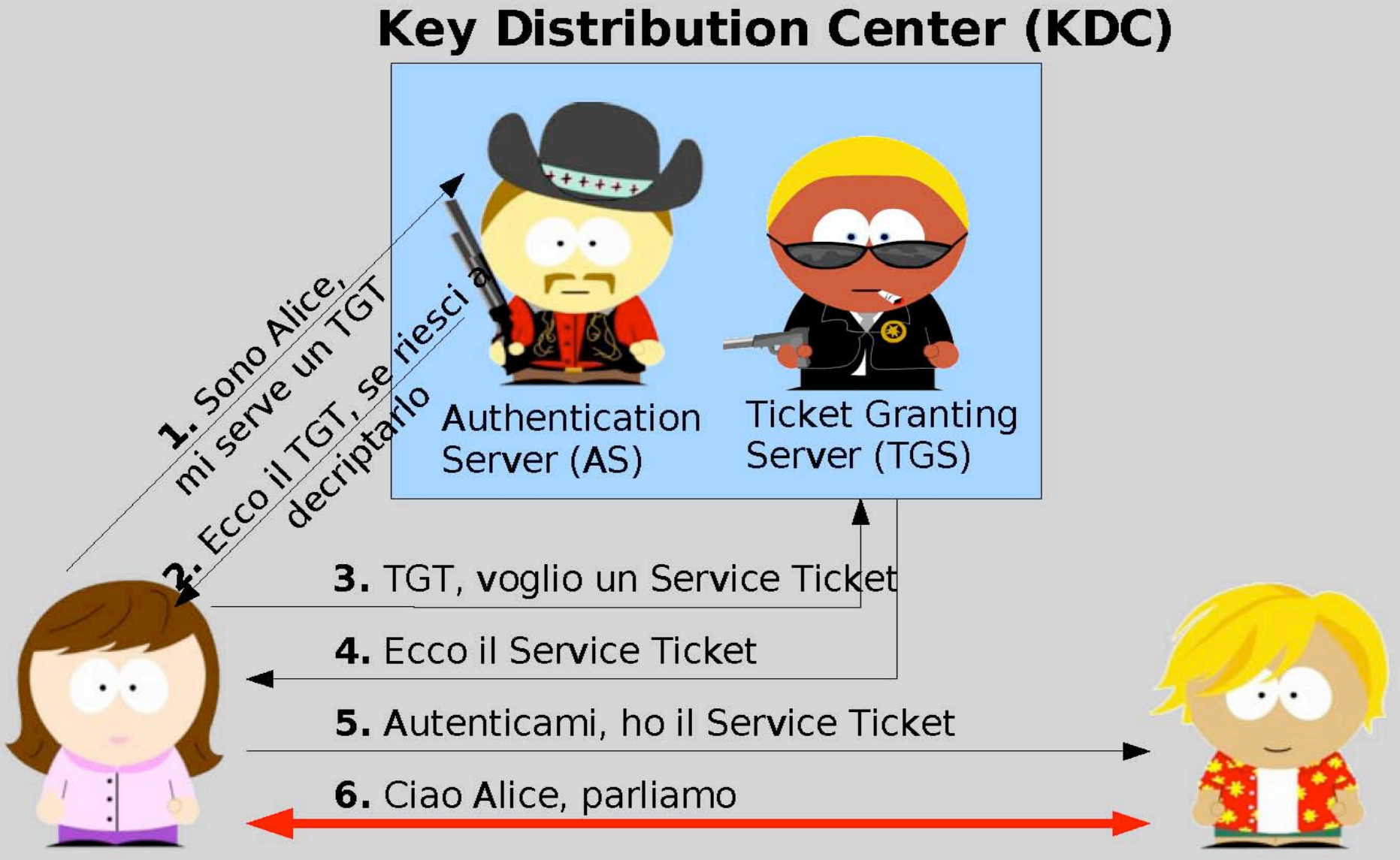
Single Sign On & Software Libero



Single Sign On & Software Libero



Single Sign On & Software Libero



- AS = Authentication Server
- SS = Service Server
- TGS = Ticket-Granting Server
- TGT = Ticket-Granting Ticket

User Client-based Logon

1) A user client sends the user identity to AS when the user logs on.

2) The AS verify that the user is authorized. The AS generates the secret key,**Ka**, by hashing the password of the user found at the database.

Client Authentication

AS sends back to the client a messages encrypted with Ka (the secret key of the client/user) containing:

- **Ks,** for use in communication with Ticket Granting Server (*Client/TGS Session Key)*.
- **Ktgs(A,Ks)** *Ticket-Granting Ticket,* which includes the client ID (A), the *client/TGS session key (KS),* encrypted using the *secret key* **Ktgs ,** shared between AS and TGS

.

- When the message from AS has been received the Alice's client asks to Alice the password . The password is used to generate Ka. If using Ka the message can be decrypted, then the user is authenticated by AS and Alice obtains the **KS** key and the *ticket Ktgs(A,Ks*).

- The client deletes the Alice's password. So, the password is present in the client only a few millisecond.

- Note that the passwords are stored at the Kerberos server, not at the client, and that the passwords did not have to be passed across the network, even in encrypted form (security advantage).

3)Once the client receives the messages, it decrypts the message to obtain **Ks,** the *Client/TGS Session Key*. This session key is used for further communications with the TGS. (Note: The client cannot decrypt **Ktgs(A,Ks)** , as it is encrypted using TGS's secret key.)

Client Service Authorization

1) When requesting services, the client sends the following two messages to the TGS:
   - Message C: Composed of **Ktgs(A,Ks),** and the ID of the requested server, **Bob**.
   - Message D: Authenticator (which is composed of the client ID and the timestamp), encrypted using the *Client/TGS Session Key,* **Ks(A,T)**

2) Upon receiving messages C and D, the TGS retrieves **Ktgs(A,Ks)** out of message C. It decrypts **Ktgs(A,Ks)** using **Ktgs,** the TGS secret key. This gives it **Ks** the "client/TGS session Key". Using this key, the TGS decrypts message D (Authenticator) and sends the following two messages to the client:

message E: client-to-server-ticket (wich includes the client ID and **Kab**, Client/Server Session Key) encrypted using **Kb**, the server secret key.

message F: **Kab,** Client/Server Session Key encrypted with **Ks**, the Client/TGS Session Key

Client Service Request

1) Upon receiving messages E and F from TGS, the client has enough information to authenticate itself to the Bob.

   The client connects to the SS and sends the following two messages:

   - message E from the previous step (the client-to-server-ticket, encrypted using **Kb**, Bob secret key)

   - message G: a new authenticator, wich incluses the client ID, time stamp and is encrypted using **Kab**, the client/server session key.

2) Bob decrypts the ticket using **Kb,** its own secret Key, to retrieve **Kab.** Using **Kab**, Bob decrypts the Autenticator and sends the following message to the client to confirm its true identity and willingness to serve the client:

Message H: the time stamp found in client'sAuthenticator plus 1, encrypted using **Kab**.

3) the client decrypts the confirmation using **Kab**, and checks whether the timestamp is correctly updated. If so,then the client can trust the server and can start issuing service requests to the server.
4) the server provides the requested services to the client.

Authentication servers

- The servers offering services may belong to different domains, each of them with own AS and TGS.
- If a client wishes to access a server belonging to a different domain it is necessary to require to the local TGS a ticket that is accepted by the remote TGS.
- In order to achieve this result, the remote **TGS** must be registered on the local TGS **as a local server**.
- In this way, the local TGS can give to Alice a valid ticket for the remote TGS and Alice is able to obtain a ticket for the remote server.

Kerberos was carefully designed to withstand attacks in distributed environment.

- **No password communicated on the network**
- **Cryptographic protection (**against spoofing).
- **Limited period of validity**. Each ticket is issued for a limited period of time. The ticket contains a time stamp with wich a receiving server determines the ticket validity.(long attacks, such brute force cryptanalysis, are usually neutralized because the attacker does not have time to complete the attack).
- **Time stamps to prevent reply attacks**. Kerberos requires reliable access to a universal clock. Each user request to a server is stamped with the time of the request. This time is compared to the current time. The request is accepted only if the time is reasonably close to the current time

DRAWBACKS

• **Single point of failure**: Il requires continous availability of the central server. When the Kerberos server is down,no one can log on (multiple Kerberos servers).

•Kerberos requires the clocks of the involved hosts to be **synchronized**. The tickets have a time availability period and if the host clock is not synchronized with the Kerberos server clock, the authentication will fail with the Kerberos server clock.

•Since all authentication is controlled by a centralized KDC, compromise of this authentication infrastructure will allow an attacker to impersonate any user.