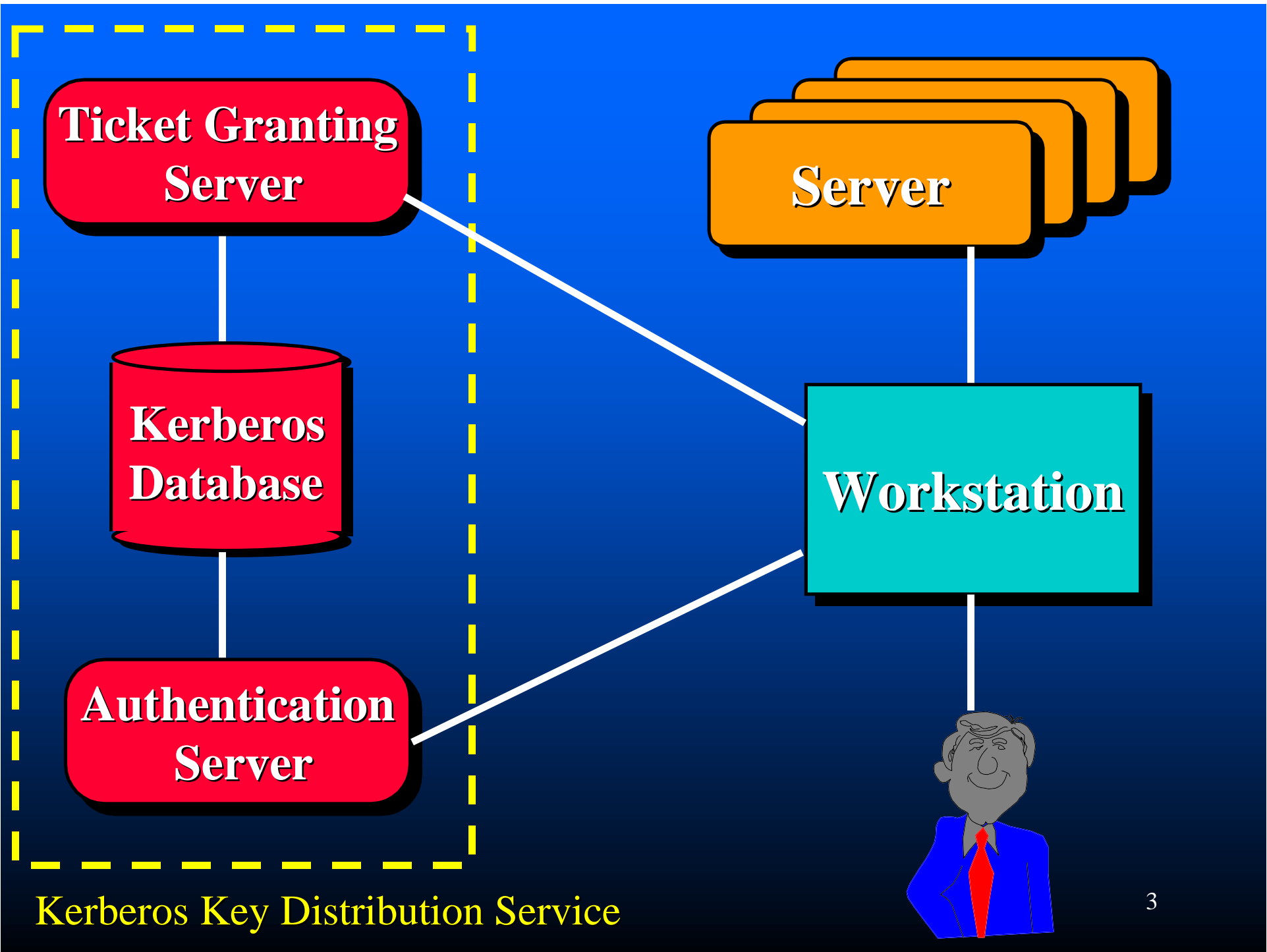


# Kerberos

- Part of project Athena (MIT).
- Trusted 3rd party authentication scheme.
- Assumes that hosts are not trustworthy.
- Requires that each client (each request for service) prove it's identity.
- Does not require user to enter password every time a service is requested!

# Kerberos Design

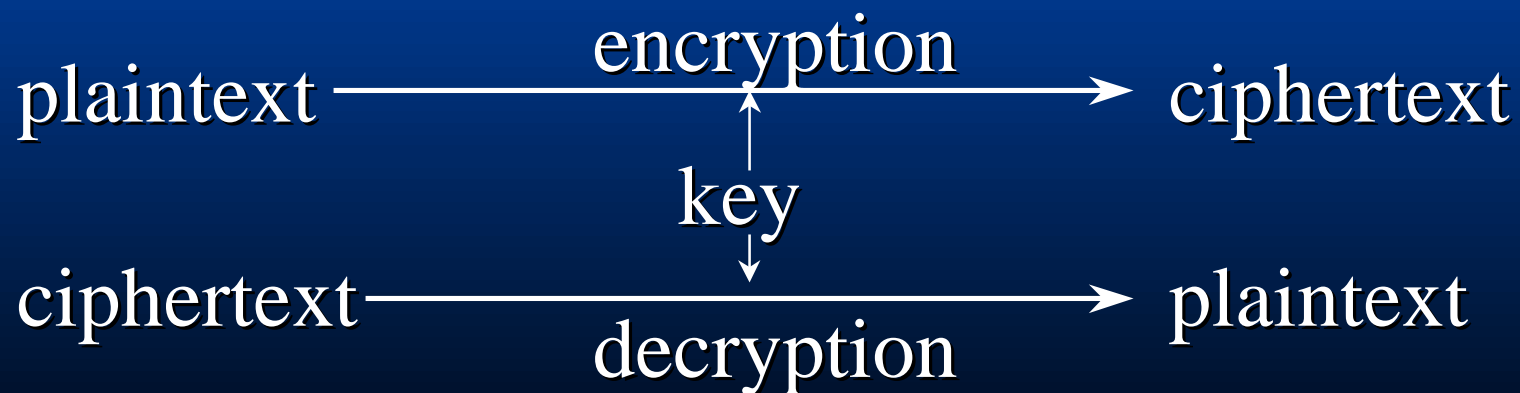
- User must identify itself once at the beginning of a workstation session (login session).
- Passwords are never sent across the network in cleartext (or stored in memory)
- Every user has a password and every service has a password.
- The only entity that knows all the passwords is the Authentication Server.



Kerberos Key Distribution Service

# Secret Key Cryptography

- The encryption used by current Kerberos implementations is DES, although Kerberos V5 has hooks so that other algorithms can be used.



# Tickets

- Each request for a service requires a ticket.
- A ticket provides a single client with access to a single server.
- Tickets are dispensed by the “Ticket Granting Server” (*TGS*), which has knowledge of all the encryption keys.
- Tickets are meaningless to clients, they simply use them to gain access to servers.

## Tickets (cont.)

- The **TGS** seals (encrypts) each ticket with the secret encryption key of the server.
- Sealed tickets can be sent safely over a network - only the server can make sense out of it.
- Each ticket has a limited lifetime (a few hours).

# Ticket Contents

- Client name (user login name)
- Server name
- Client Host network address
- Session Key for C $\leftrightarrow$ S
- Ticket lifetime
- Creation timestamp

# Session Key

- Random number that is specific to a session.
- Session Key is used to seal client requests to server.
- Session Key can be used to seal responses (application specific usage).

# Authenticators

- Authenticators prove a client's identity.
- Includes:
  - Client user name.
  - Client network address.
  - Timestamp.
- Authenticators are sealed with a session key.

# Bootstrap

- Each time a client wants to contact a server, it must first ask the 3rd party (*TGS*) for a ticket and session key.
- In order to request a ticket from the *TGS*, the client must already have a TG ticket and a session key for communicating with the *TGS*!

# Authentication Server

- The client sends a plaintext request to the *AS* asking for a ticket it can use to talk to the *TGS*.
- REQUEST:
  - login name
  - *TGS* name

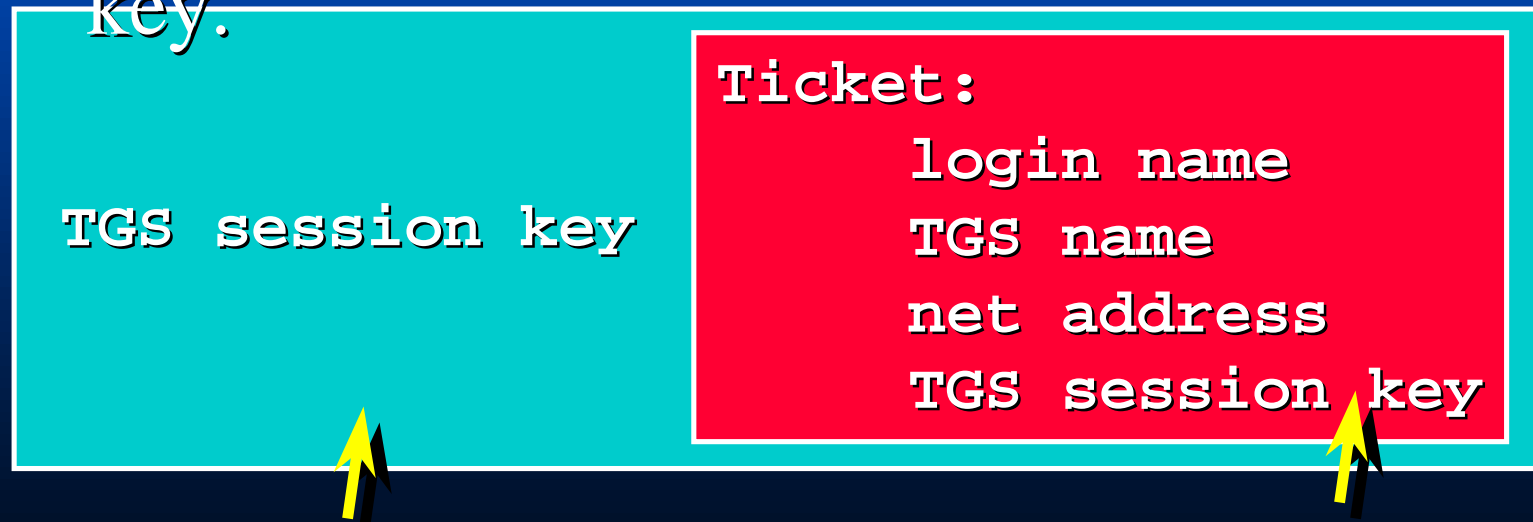
Since this request contains only well-known names, it does not need to be sealed.

# Authentication Server

- The *AS* finds the keys corresponding to the login name and the *TGS* name.
- The *AS* creates a ticket:
  - login name
  - *TGS* name
  - client network address
  - *TGS* session key
- The *AS* seals the ticket with the *TGS* secret key.

# Authentication Server Response

- The **AS** also creates a random session key for the client and the **TGS** to use.
- The session key and the sealed ticket are sealed with the user (login name) secret key.



Sealed with user key

Sealed with TGS key

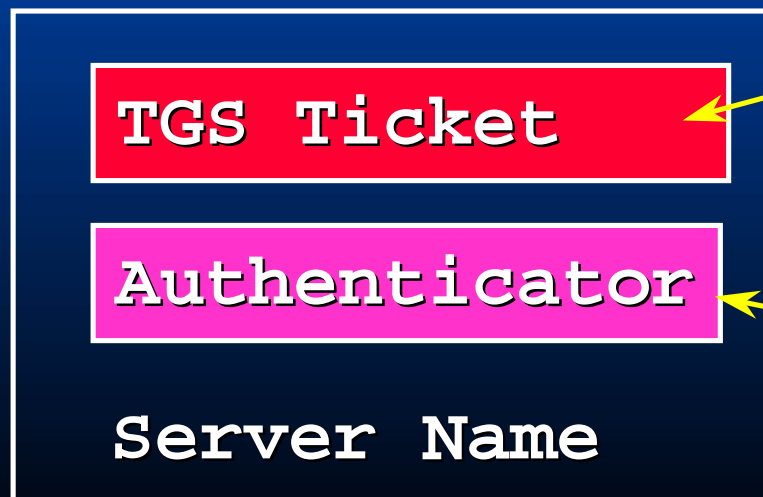
# Accessing the *TGS*

- The client decrypts the message using the user's password as the secret key.
- The client now has a session key and ticket that can be used to contact the *TGS*.
- The client cannot see inside the ticket, since the client does not know the *TGS* secret key.

# Accessing a Server

- When a client wants to start using a server (service), the client must first obtain a ticket.
- The client composes a request to send to the

*TGS:*



sealed with  
TGS key

sealed with  
session key

# *TGS* response

- The *TGS* decrypts the ticket using it's secret key. Inside is the TGS session key.
- The *TGS* decrypts the Authenticator using the session key.
- The *TGS* check to make sure login names, client addresses and *TGS* server name are all OK.
- *TGS* makes sure the Authenticator is recent.

# *TGS* Response

- Once everything checks out - the TGS:
  - builds a ticket for the client and requested server. The ticket is sealed with the server key.
  - creates a session key
  - seals the entire message with the TGS session key and sends it to the client.

# Client accesses Server

- The client now decrypts the *TGS* response using the TGS session key.
- The client now has a session key for use with the new server, and a ticket to use with that server.
- The client can contact the new server using the same format used to access the *TGS*.

# Kerberos Summary

- Every service request needs a ticket.
- Tickets come from the TGS (except the ticket for the TGS!).
- Workstations cannot understand tickets, they are encrypted using the server key.
- Every ticket has an associated session key.
- Tickets are reusable.

# Kerberos Summary (cont.)

- Tickets have a finite lifetime.
- Authenticators are only used once (new connection to a server).
- Authenticators expire fast !
- Server maintains list of authenticators (prevent stolen authenticators).
- There is a lot more to Kerberos!!!