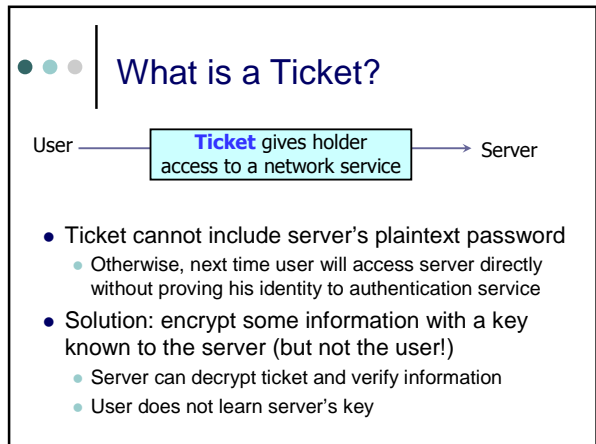
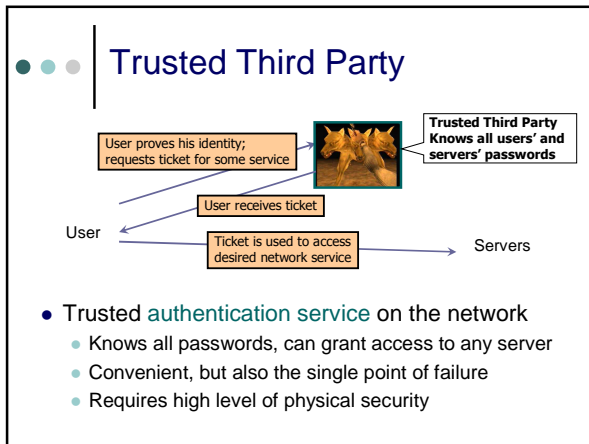


# Kerberos

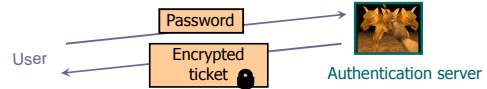
- ## Authenticating to Multiple Servers
- Consider a set of user that needs to access different services on the net
    - Need to authenticate to each of them
    - Naïve solution: every server knows every user's password
      - **Insecure**: breaking into one server can compromise all users
      - **Inefficient**: to change password, a user must contact every server



## Contents of a Ticket

- User name
- Server name
- Address of user's workstation
  - Otherwise, a user on another workstation can steal the ticket and use it to gain access to the server
- Ticket lifetime (duration for which valid)
- A few other things (e.g., session key)

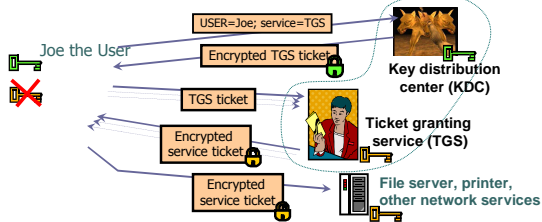
## User Authentication to Third Party



- **Insecure:**
  - Eavesdropper can steal the password and later impersonate the user to the authentication server
- **Inconvenient:** need to send the password each time to obtain the ticket for any network service
  - Separate authentication for email, printing, etc.

## Two-Step Authentication

- Prove identity **once** to obtain special **TGS ticket**
- Use TGS to get tickets for any network service



## Symmetric Keys in Kerberos

- $K_C$  : private key of client C
  - Derived from user's password
  - Known to client and key distribution center (KDC)
- $K_{TGS}$  : private key of TGS
  - Known to KDC and ticket granting service (TGS)
- $K_V$  : private key of network service V
  - Known to V and TGS; separate key for each service
- $K_{C,TGS}$  : session key between C and TGS
  - Created by KDC, known to C and TGS, valid only for one session (some lifetime) between C and TGS
- $K_{C,V}$  : session key between C and V
  - Created by TGS, known to C and V, valid only for one session (some lifetime) between C and TGS



## “Single Logon” Authentication

- Client C types in password once
- Converted to client key  $K_C$
- C sends to KDC :  $(ID_C, ID_{TGS}, time_C)$
- KDC sends to C :  $(K_{C,TGS}, ID_{TGS}, time_{KDC}, lifetime, ticket_{TGS})$  encrypted with  $K_C$ 
  - $ticket_{TGS} = (K_{C,TGS}, ID_C, Addr_C, ID_{TGS}, time_{KDC}, lifetime)$  encrypted with  $K_{TGS}$
  - Client will use this ticket to get other tickets without re-authenticating



- $K_{C,TGS}$  : short term session key
  - used for communication between C and TGS during lifetime
- Typical validity of TGS ticket – 1 day
  - Client only needs to obtain TGS ticket once a day (say, every morning)
  - Password is entered once and then deleted from the client machine after obtaining the TGS ticket
  - Password is never sent over the network
  - Ticket is encrypted; client cannot forge it or tamper with it



## Obtaining a Service Ticket

- C sends to TGS:  $(ID_V, ticket_{TGS}, auth_C)$ 
  - $auth_C = (ID_C, Addr_C, time_C)$  encrypted with  $K_{C,TGS}$
  - **authenticator** to ensure it is the same client that got the ticket
- TGS sends to C:  $(K_{C,V}, ID_V, time_{TGS}, ticket_V)$  encrypted with  $K_{C,TGS}$ 
  - $ticket_V = (K_{C,V}, ID_C, Addr_C, ID_V, time_{TGS}, lifetime)$  encrypted with  $K_V$

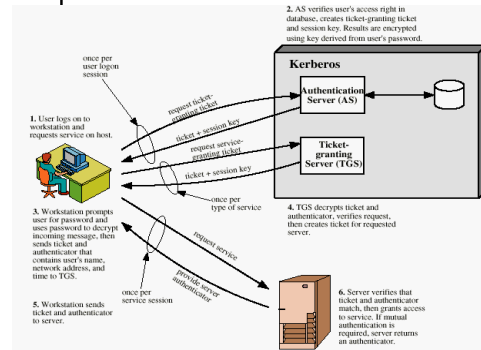


- Client uses TGS ticket to obtain a service ticket and a short-term session key for each network service
  - One encrypted, unforgeable ticket per service (printer, email, etc.)

## Obtaining Service

- C sends to V:  $(\text{ticket}_V, \text{auth}_C)$
- $\text{auth}_C = (\text{ID}_C, \text{Addr}_C, \text{time}_C)$  encrypted with  $K_{C,V}$
- V sends to C:  $(\text{time}_C+1)$  encrypted with  $K_{C,V}$ 
  - Authenticates server to client
- For each service request, client uses the short-term session key for that service and the ticket he received from TGS

## Summary of Kerberos



## Important Ideas in Kerberos

- Short-term session keys
  - Long-term secrets used only to derive short-term keys
  - Separate session key for each user-server pair
    - ... but multiple user-server sessions re-use the same key
- Proofs of identity are based on **authenticators**
  - Client encrypts his identity, address and current time using a short-term session key
    - Also prevents replays (if clocks are globally synchronized)
  - Server learns this key separately (via encrypted ticket that client can't decrypt) and verifies user's identity

## Kerberos in Large Networks

- One KDC isn't enough for large networks (why?)
- Network is divided into **realms**
  - KDCs in different realms have different key databases
- To access a service in another realm, users must do **cross-realm authentication**
  - Get ticket for home-realm TGS from home-realm KDC
  - Get ticket for remote-realm TGS from home-realm TGS
    - As if remote-realm TGS were just another network service
  - Get ticket for remote service from that realm's TGS
  - Use remote-realm ticket to access service
  - **$N(N-1)/2$  key exchanges for full  $N$ -realm interoperation (NOT SCALABLE)**
- Use **Hierarchical cross-realm authentication**



## Hierarchical Cross-realm Authentication

- Organize realms as trees