CS 361S

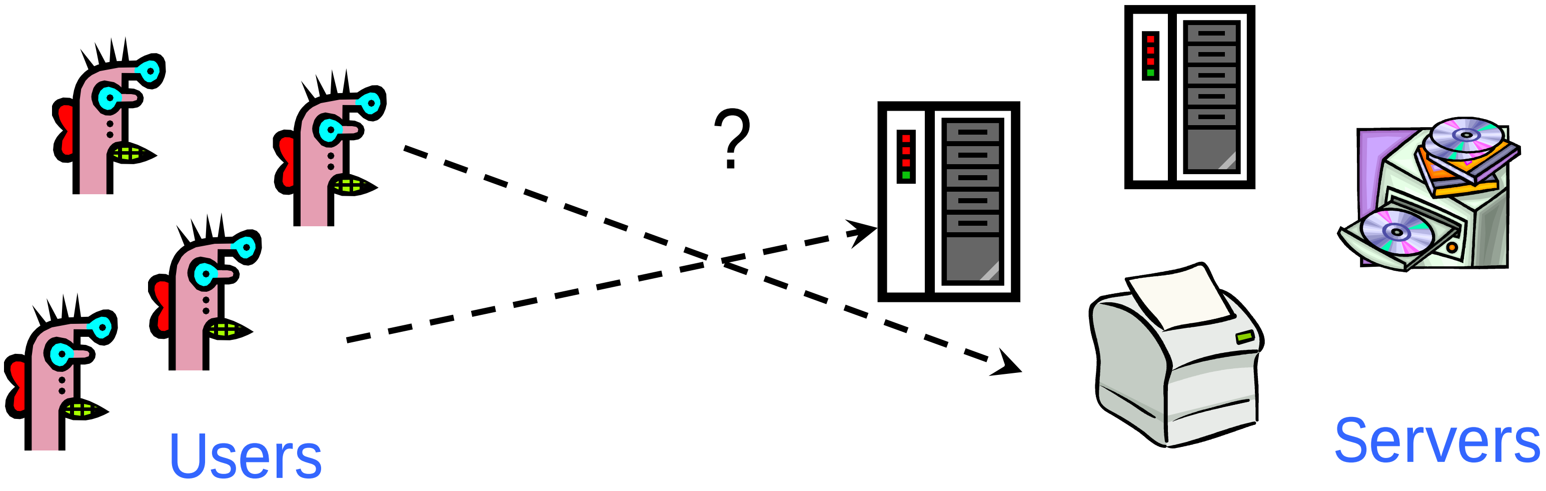
Kerberos

Vitaly Shmatikov

Reading Assignment

- Kaufman Chapters 13 and 14
- “Designing an Authentication System: A Dialogue in Four Scenes”
  - A high-level survey of network threats and design principles behind Kerberos

Many-to-Many Authentication



How do users prove their identities when requesting services from machines on the network?

Naïve solution: every server knows every user's password

- Insecure: break into one server ⇒ compromise all users
- Inefficient: to change password, user must contact every server

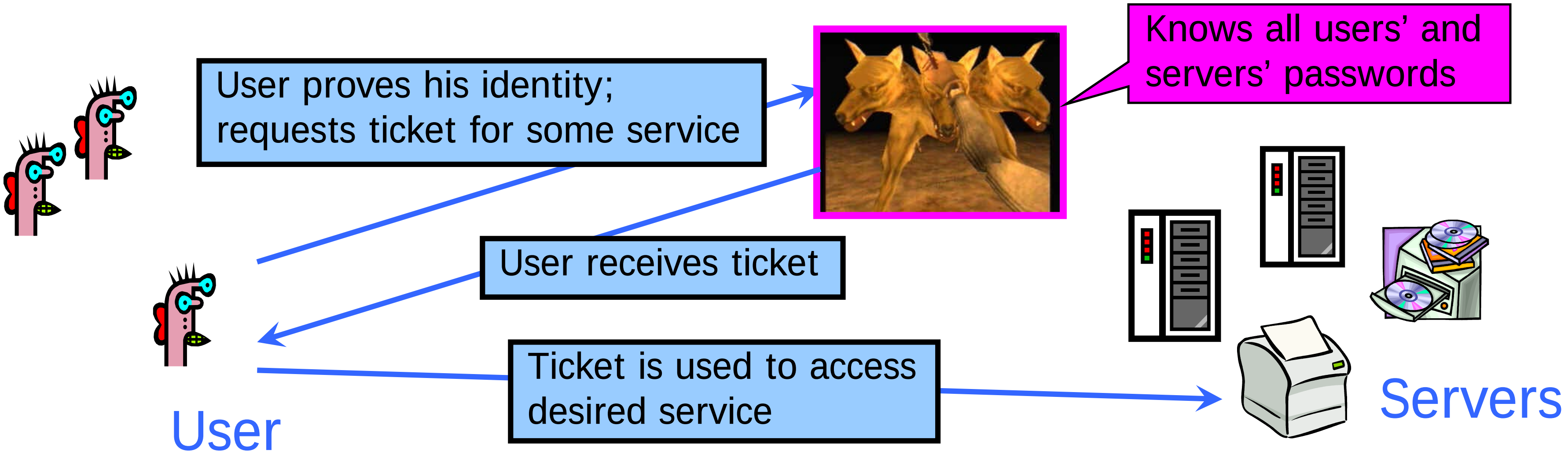
Requirements

- Security
  - ... against attacks by passive eavesdroppers and actively malicious users
- Transparency
  - Users shouldn’t notice authentication taking place
  - Entering password is Ok, if done rarely
- Scalability
  - Large number of users and servers

Threats

- User impersonation
  - Malicious user with access to a workstation pretends to be another user from the same workstation
- Network address impersonation
  - Malicious user changes network address of his workstation to impersonate another workstation
- Eavesdropping, tampering, replay
  - Malicious user eavesdrops, tampers, or replays other users' conversations to gain unauthorized access

Solution: Trusted Third Party



User proves his identity; requests ticket for some service

Knows all users' and servers' passwords

User receives ticket

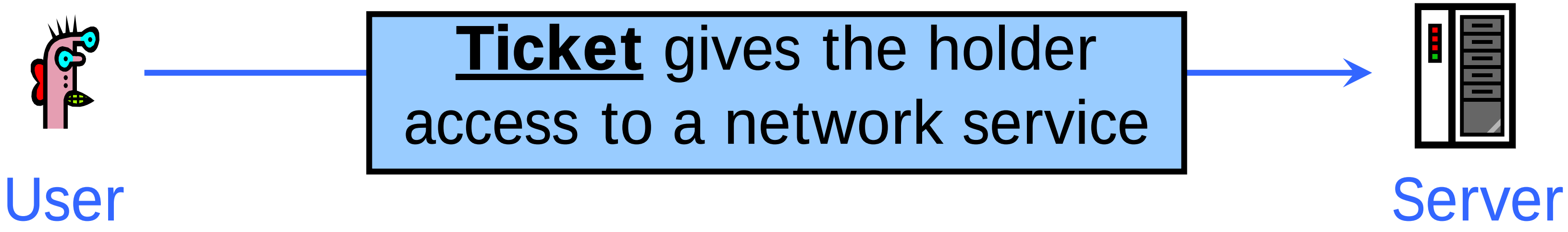
Ticket is used to access desired service

User

Servers

- Trusted authentication service on the network
  - Knows all passwords, can grant access to any server
  - Convenient (but also the single point of failure!)
  - Requires high level of physical security

What Should a Ticket Look Like?



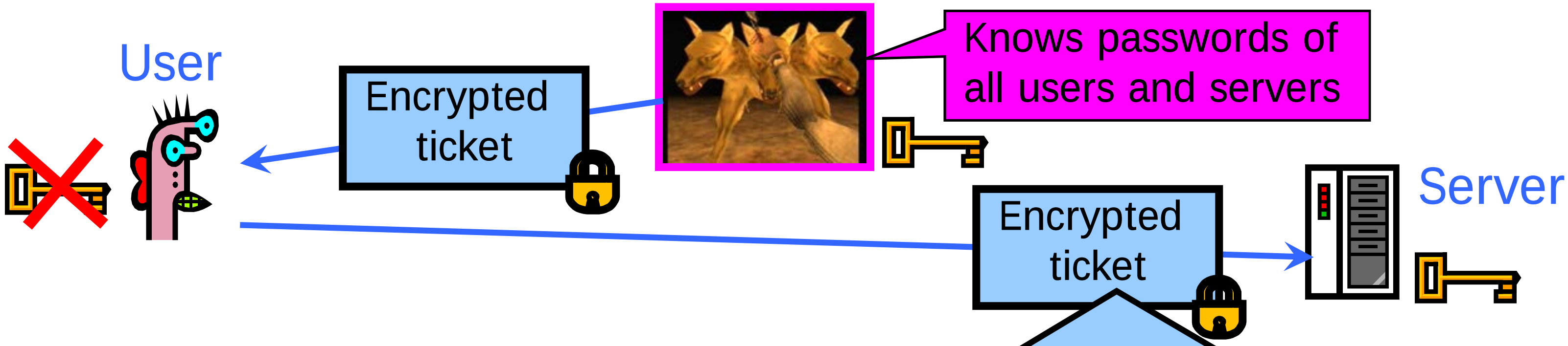
Ticket gives the holder access to a network service

User

Server

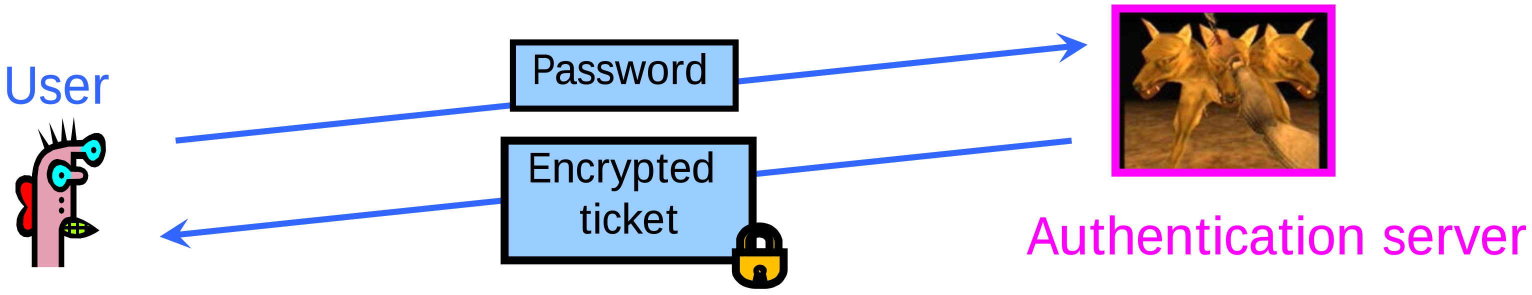
- User should not be able to access server without first proving his identity to authentication service
- Ticket proves that user has authenticated
  - Authentication service encrypts some information with a key known to the server (but not the user!)
    - The only thing the user can do is pass the ticket to the server
    - Hash functions would’ve worked well, but this is 1980s design
  - Server decrypts the ticket and verifies information

What Should a Ticket Include?



- 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
- A few other things (session key, etc.)

Naïve Authentication



User

Password

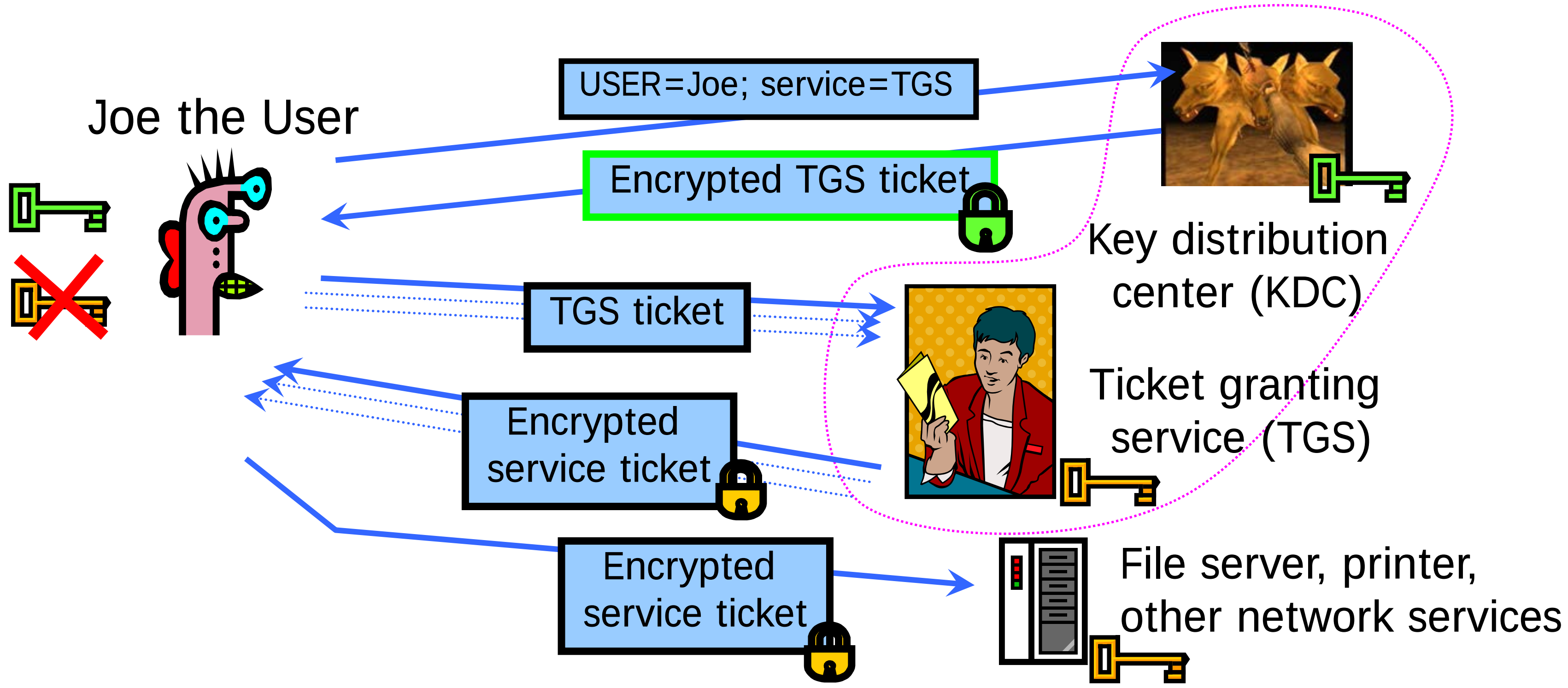
Encrypted ticket

Authentication server

- **Insecure:** passwords are sent in plaintext
  - 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 a special TGS ticket
- Use TGS to get tickets for any network service



Joe the User

USER=Joe; service=TGS

Encrypted TGS ticket

Key distribution center (KDC)

TGS ticket

Ticket granting service (TGS)

Encrypted service ticket

Encrypted service ticket

File server, printer, other network services

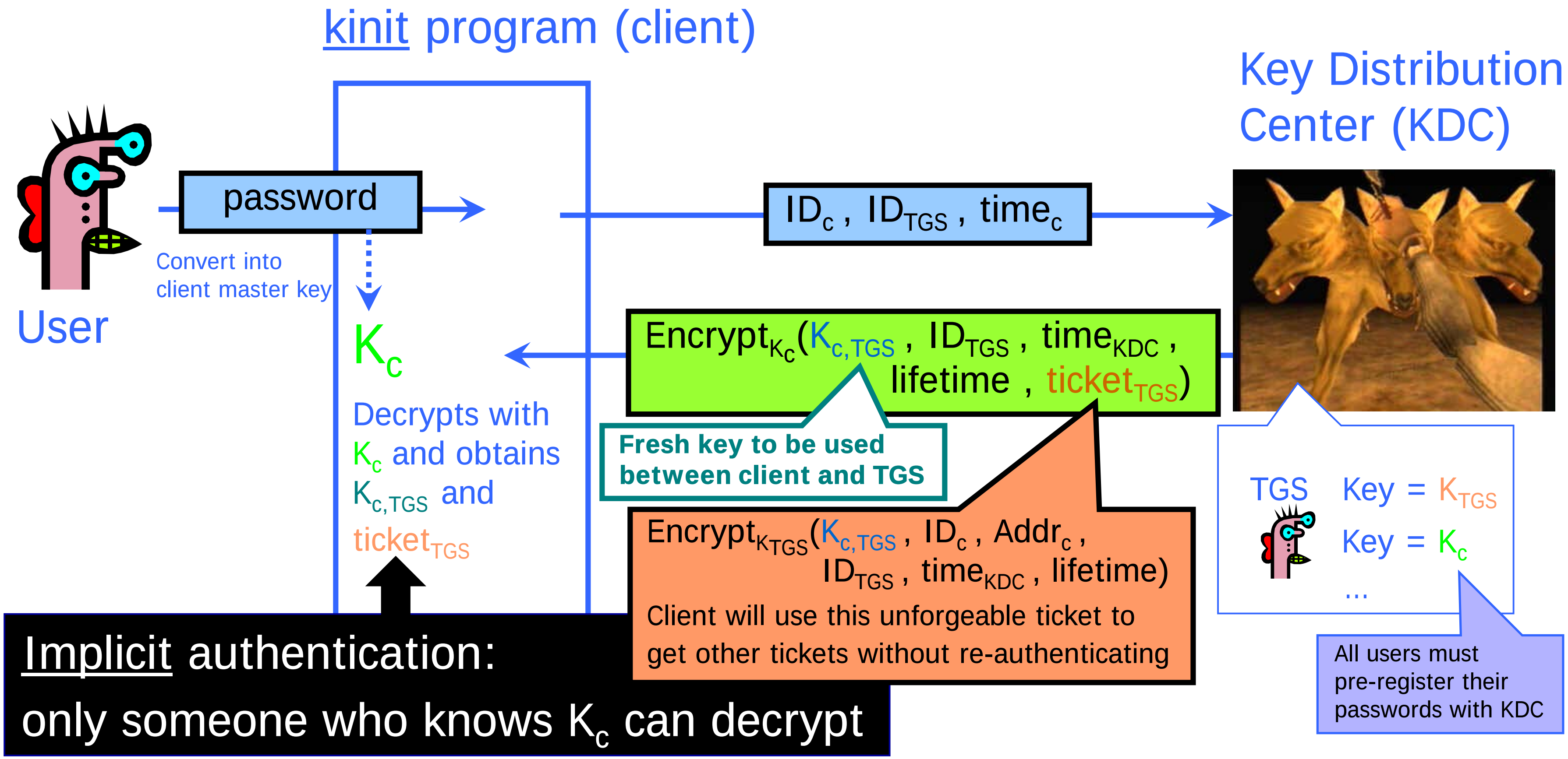
Threats

- Ticket hijacking
  - Malicious user may steal the service ticket of another user on the same workstation and try to use it
    - Network address verification does not help
  - Servers must verify that the user who is presenting the ticket is the same user to whom the ticket was issued
- No server authentication
  - Attacker may misconfigure the network so that he receives messages addressed to a legitimate server
    - Capture private information from users and/or deny service
  - Servers must prove their identity to users

Symmetric Keys in Kerberos

- K_c is long-term key of client C
  - Derived from the user's password
  - Known to the client and the key distribution center (KDC)
- K_{TGS} is long-term key of TGS
  - Known to KDC and the ticket granting service (TGS)
- K_v is long-term key of network service V
  - Known to V and TGS; each service V has its own long-term key
- $K_{c,TGS}$ is short-term session key betw. C and TGS
  - Created by KDC, known to C and TGS
- $K_{c,v}$ is short-term session key between C and V
  - Created by TGS, known to C and V

“Single Logon” Authentication



kinit program (client)

User

password

Convert into client master key

K_c

Decrypts with K_c and obtains $K_{c,TGS}$ and $ticket_{TGS}$

ID_c , ID_{TGS} , $time_c$

Key Distribution Center (KDC)

$Encrypt_{K_c}(K_{c,TGS}$, ID_{TGS} , $time_{KDC}$, lifetime , $ticket_{TGS})$

Fresh key to be used between client and TGS

$Encrypt_{K_{TGS}}(K_{c,TGS}$, ID_c , $Addr_c$, ID_{TGS} , $time_{KDC}$, lifetime)

Client will use this unforgeable ticket to get other tickets without re-authenticating

TGS Key = K_{TGS}

Key = K_c

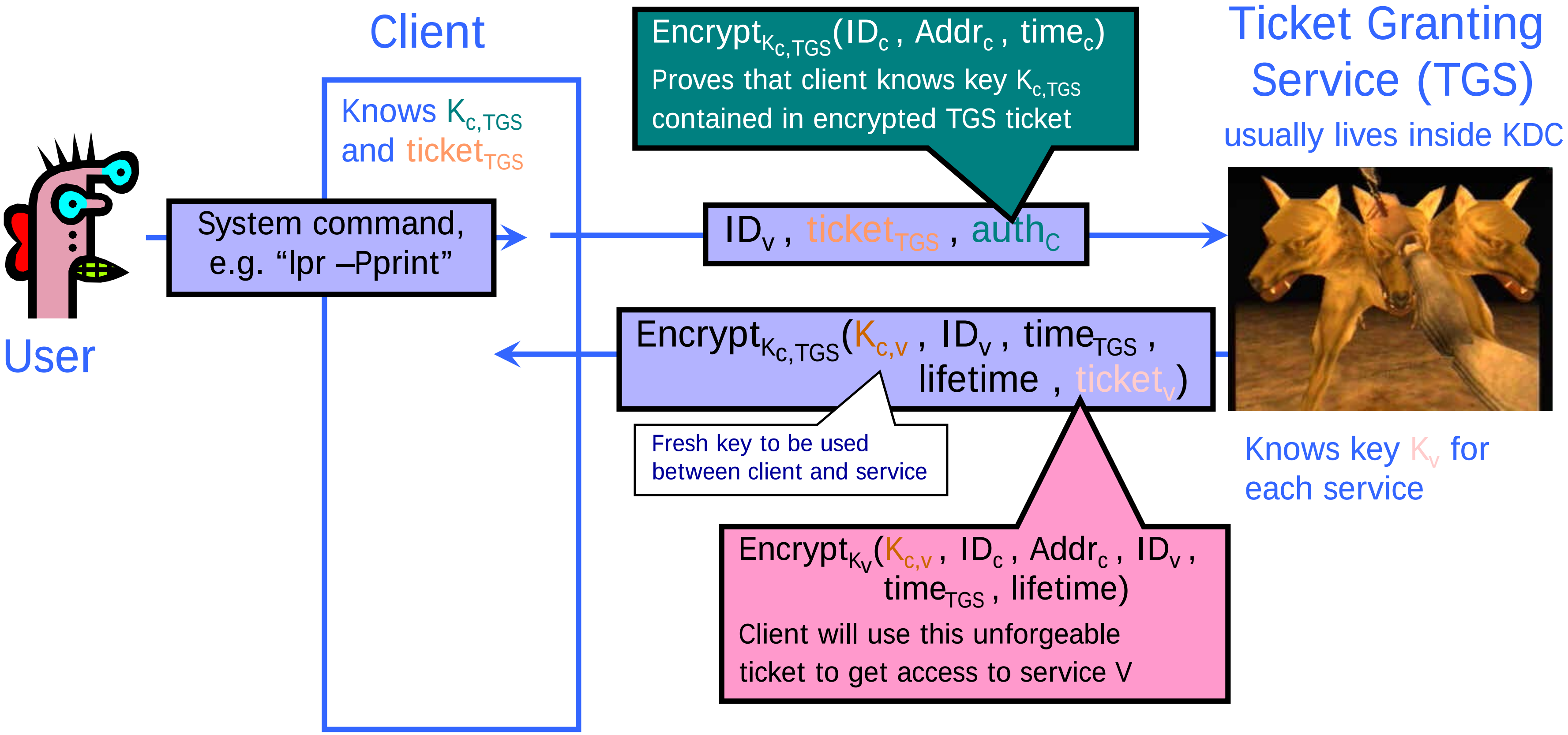
…

All users must pre-register their passwords with KDC

Implicit authentication: only someone who knows K_c can decrypt

- Client only needs to obtain TGS ticket **once** (say, every morning)
- Ticket is encrypted; client cannot forge it or tamper with it

Obtaining a Service Ticket



Client

Knows $K_{c,TGS}$ and $ticket_{TGS}$

User

System command, e.g. “lpr –Pprint”

$Encrypt_{K_{c,TGS}}(ID_c , Addr_c , time_c)$

Proves that client knows key $K_{c,TGS}$ contained in encrypted TGS ticket

ID_v , $ticket_{TGS}$, $auth_C$

Ticket Granting Service (TGS)

usually lives inside KDC

$Encrypt_{K_{c,TGS}}(K_{c,v}$, ID_v , $time_{TGS}$, lifetime , $ticket_v)$

Fresh key to be used between client and service

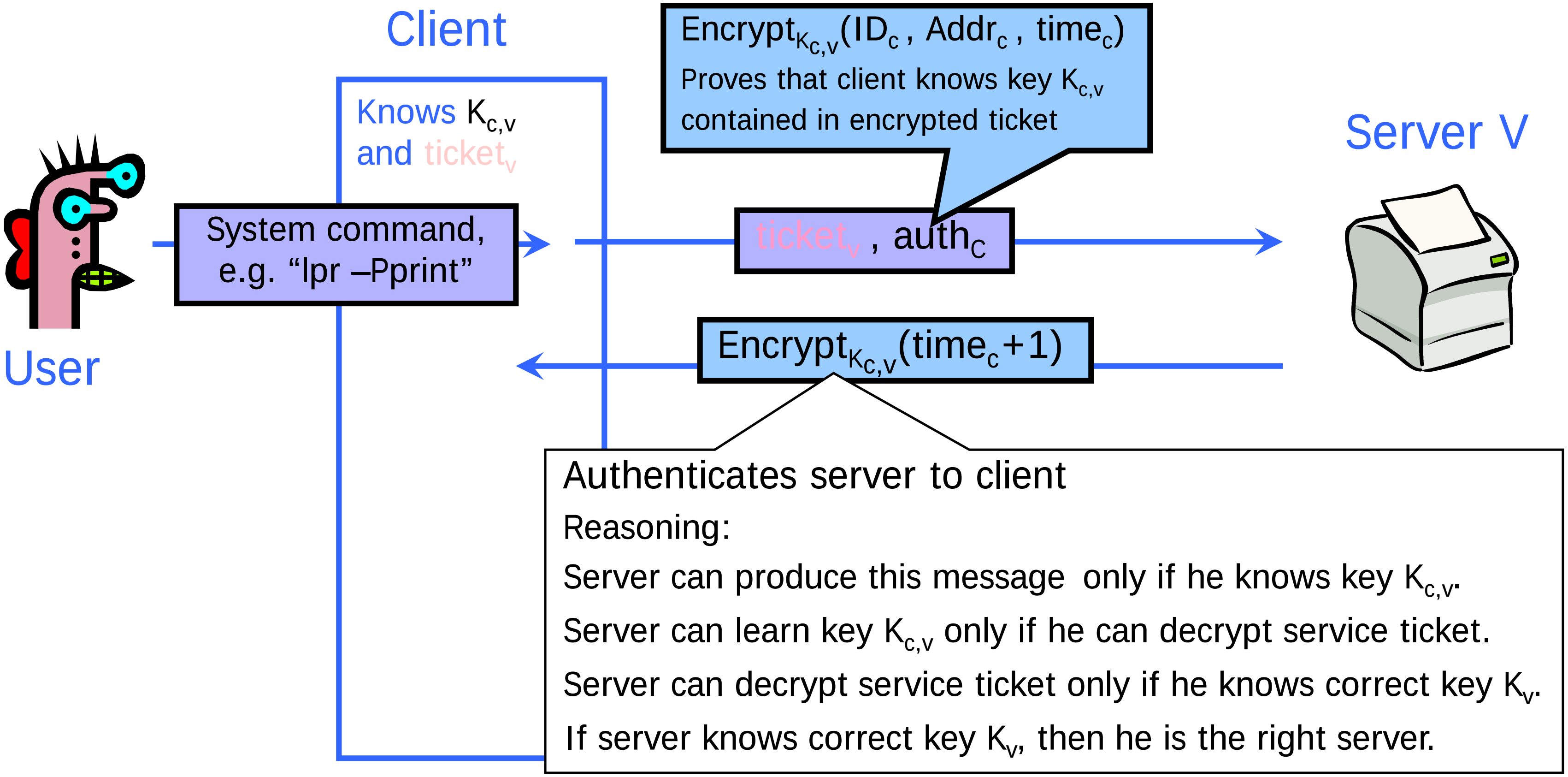
Knows key K_v for each service

$Encrypt_{K_v}(K_{c,v}$, ID_c , $Addr_c$, ID_v , $time_{TGS}$, lifetime)

Client will use this unforgeable ticket to get access to service V

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

Obtaining Service



Client

Knows $K_{c,v}$
and $ticket_v$

User

System command,
e.g. “lpr –Pprint”

$Encrypt_{K_{c,v}}(ID_c , Addr_c , time_c)$

Proves that client knows key $K_{c,v}$ contained in encrypted ticket

Server V

$ticket_v$, $auth_C$

$Encrypt_{K_{c,v}}(time_c+1)$

Authenticates server to client

Reasoning:

Server can produce this message only if he knows key $K_{c,v}$.

Server can learn key $K_{c,v}$ only if he can decrypt service ticket.

Server can decrypt service ticket only if he knows correct key K_v.

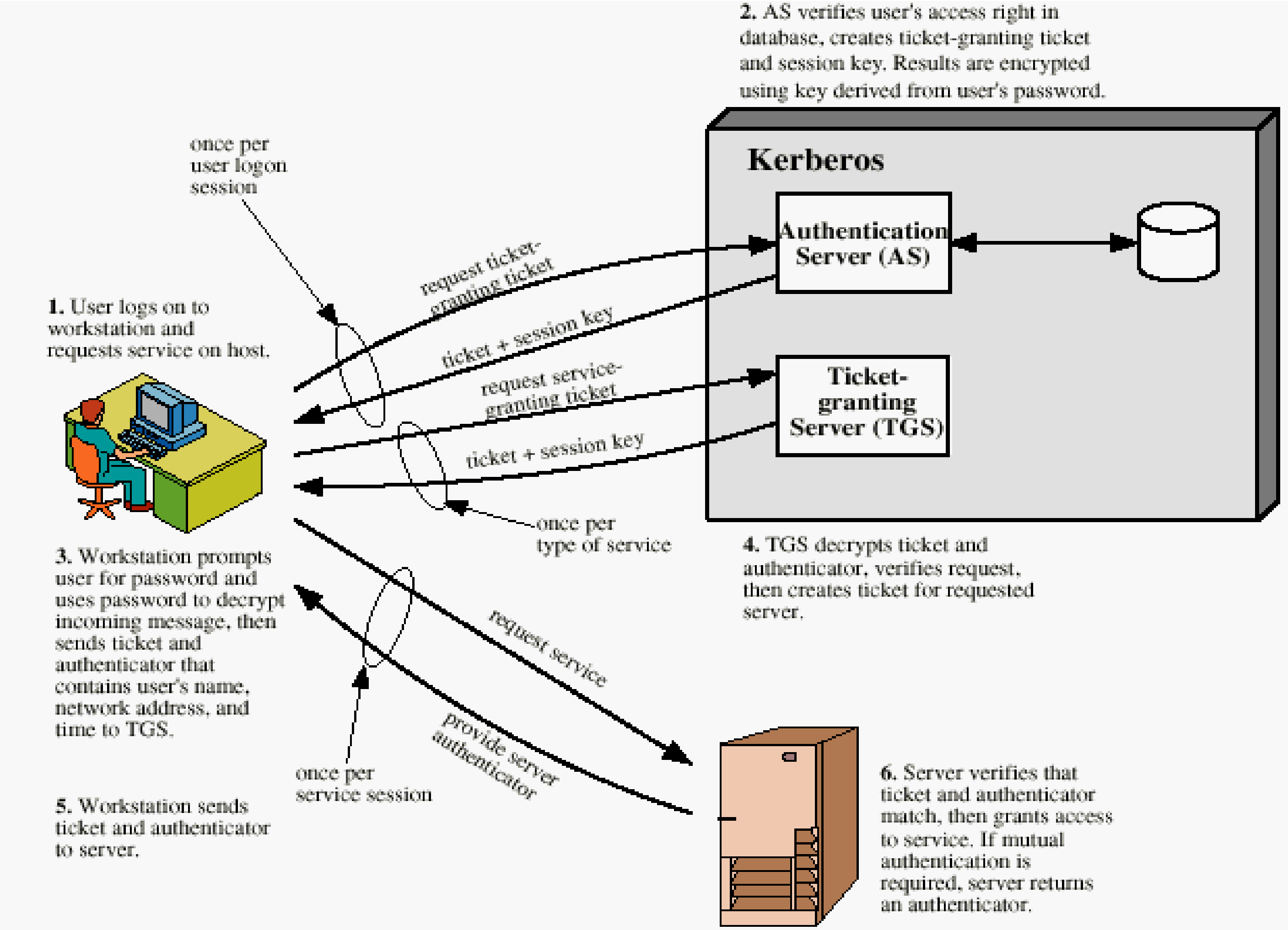
If server knows correct key K_v, then he is the right server.

- For each service request, client uses the short-term key for that service and the ticket he received from TGS

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...
    - 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

Summary of Kerberos



1. User logs on to workstation and requests service on host.

2. AS verifies user's access right in database, creates ticket-granting ticket and session key. Results are encrypted using key derived from user's password.

3. Workstation prompts user for password and uses password to decrypt incoming message, then sends ticket and authenticator that contains user's name, network address, and time to TGS.

4. TGS decrypts ticket and authenticator, verifies request, then creates ticket for requested server.

5. Workstation sends ticket and authenticator to server.

6. Server verifies that ticket and authenticator match, then grants access to service. If mutual authentication is required, server returns an authenticator.

Kerberos
Authentication Server (AS)
Ticket-granting Server (TGS)
once per user logon session
request ticket-granting ticket
ticket + session key
request service-granting ticket
ticket + session key
once per type of service
request service
provide server authenticator
once per service session

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
    - Re-used by multiple sessions between same user and server
- Proofs of identity based on authenticators
  - Client encrypts his identity, addr, time with session key; knowledge of key proves client has authenticated to KDC
    - Also prevents replays (if clocks are globally synchronized)
  - Server learns this key separately (via encrypted ticket that client can't decrypt), verifies client's authenticator
- Symmetric cryptography only

Kerberos Version 5

- Better user-server authentication
  - Separate subkey for each user-server session instead of re-using the session key contained in the ticket
  - Authentication via subkeys, not timestamp increments
- Authentication forwarding (delegation)
  - Servers can access other servers on user's behalf, eg, can tell printer to fetch email
- Realm hierarchies for inter-realm authentication
- Explicit integrity checking + standard CBC mode
- Multiple encryption schemes, not just DES

Practical Uses of Kerberos

- Microsoft Windows
- Email, FTP, network file systems, many other applications have been kerberized
  - Use of Kerberos is transparent for the end user
  - Transparency is important for usability!
- Local authentication
  - login and su in OpenBSD
- Authentication for network protocols
  - rlogin, rsh
- Secure windowing systems