

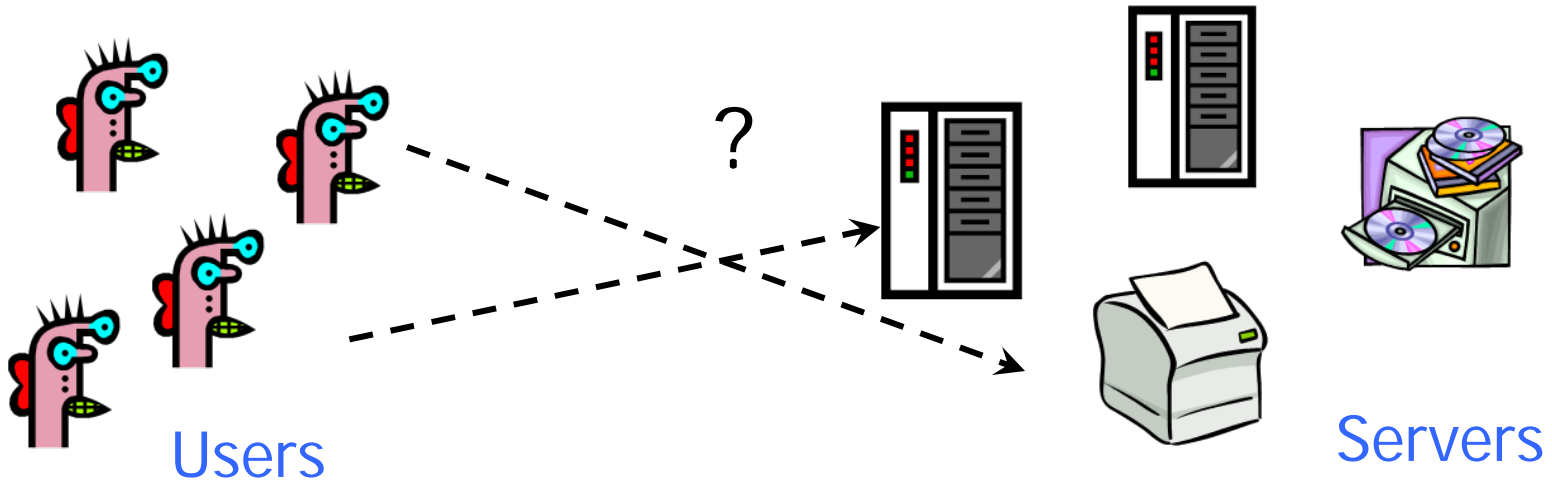
Kerberos

Vitaly Shmatikov

Reading Assignment

- ◆ Kaufman Chapters 13 and 14
- ◆ “Designing an Authentication System: A Dialogue in Four Scenes”
 - Nice 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 \Rightarrow compromise all users
- **Inefficient**: to change password, user must contact every server

Requirements

◆ Security

- ... against attacks by passive eavesdroppers and actively malicious users

◆ Reliability

◆ 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
 - Can't trust workstations to verify users' identities

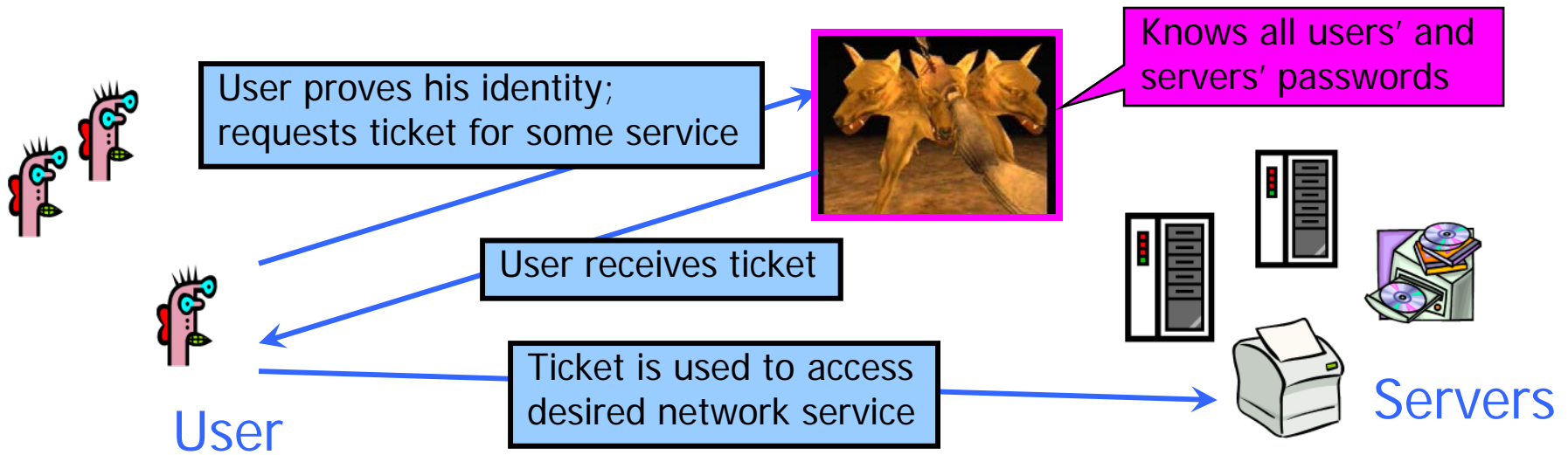
◆ Network address impersonation

- Malicious user changes network address of his workstation to impersonate another workstation

◆ Eavesdropping, tampering and replay

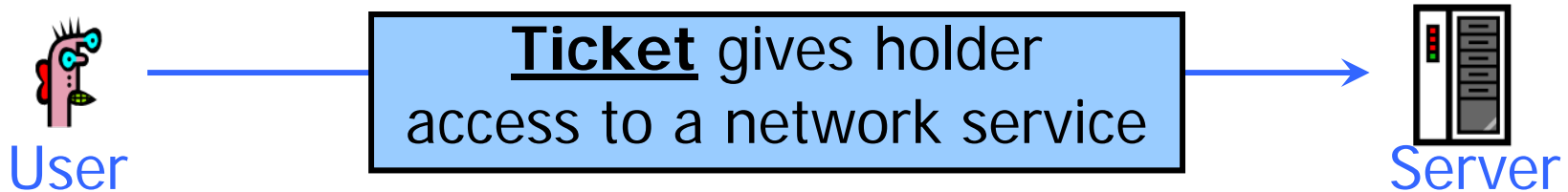
- Malicious user eavesdrops, tampers or replays other users' conversations to gain unauthorized access

Solution: Trusted Third Party



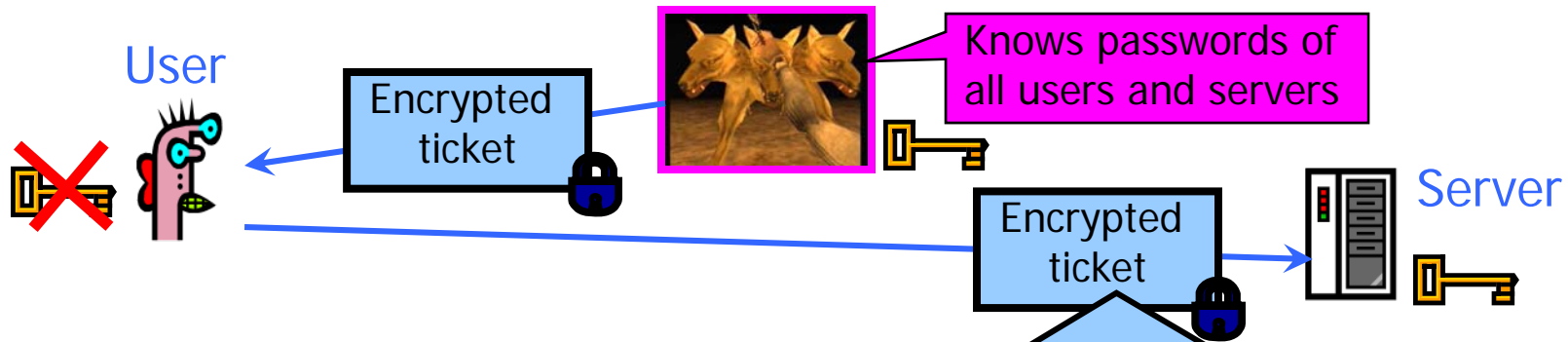
- ◆ Trusted **authentication service** on the network
 - Knows all passwords, can grant access to any server
 - Convenient, but also the single point of failure
 - Requires high level of physical security

What Should a Ticket Look Like?



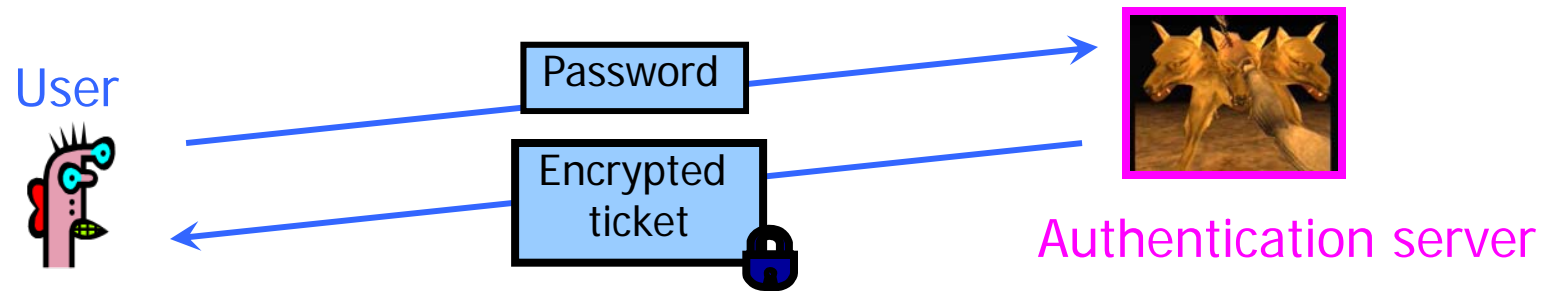
- ◆ Ticket cannot include server's plaintext password
 - Otherwise, next time user will access server directly without proving his identity to authentication service
- ◆ Solution: **encrypt** some information with a key known to the server (but not the user!)
 - Server can decrypt ticket and verify information
 - User does not learn server's key

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 (e.g., session key)

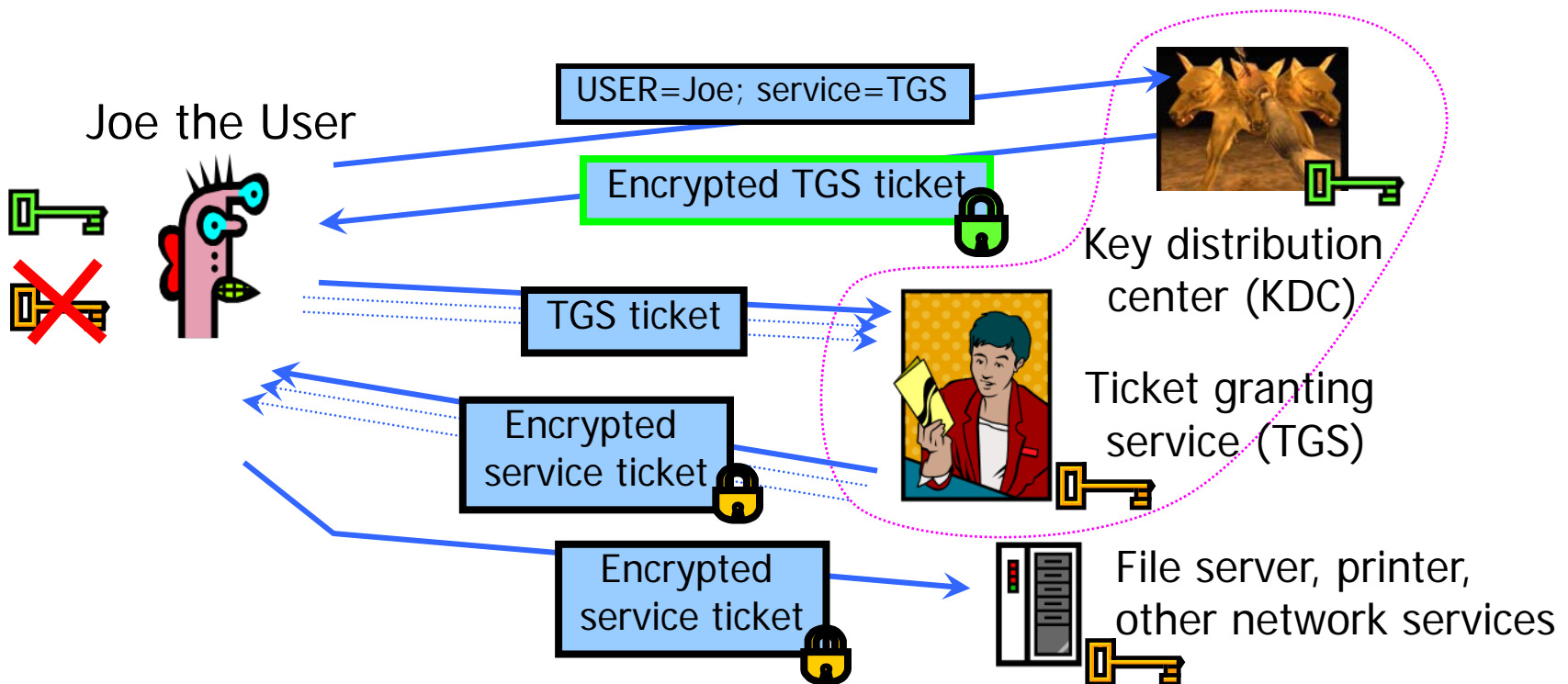
How Is Authentication Done?



- ◆ **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 special TGS ticket
- ◆ Use TGS to get tickets for any network service



Still Not Good Enough

◆ Ticket hijacking

- Malicious user may steal the service ticket of another user on the same workstation and use it
 - IP 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

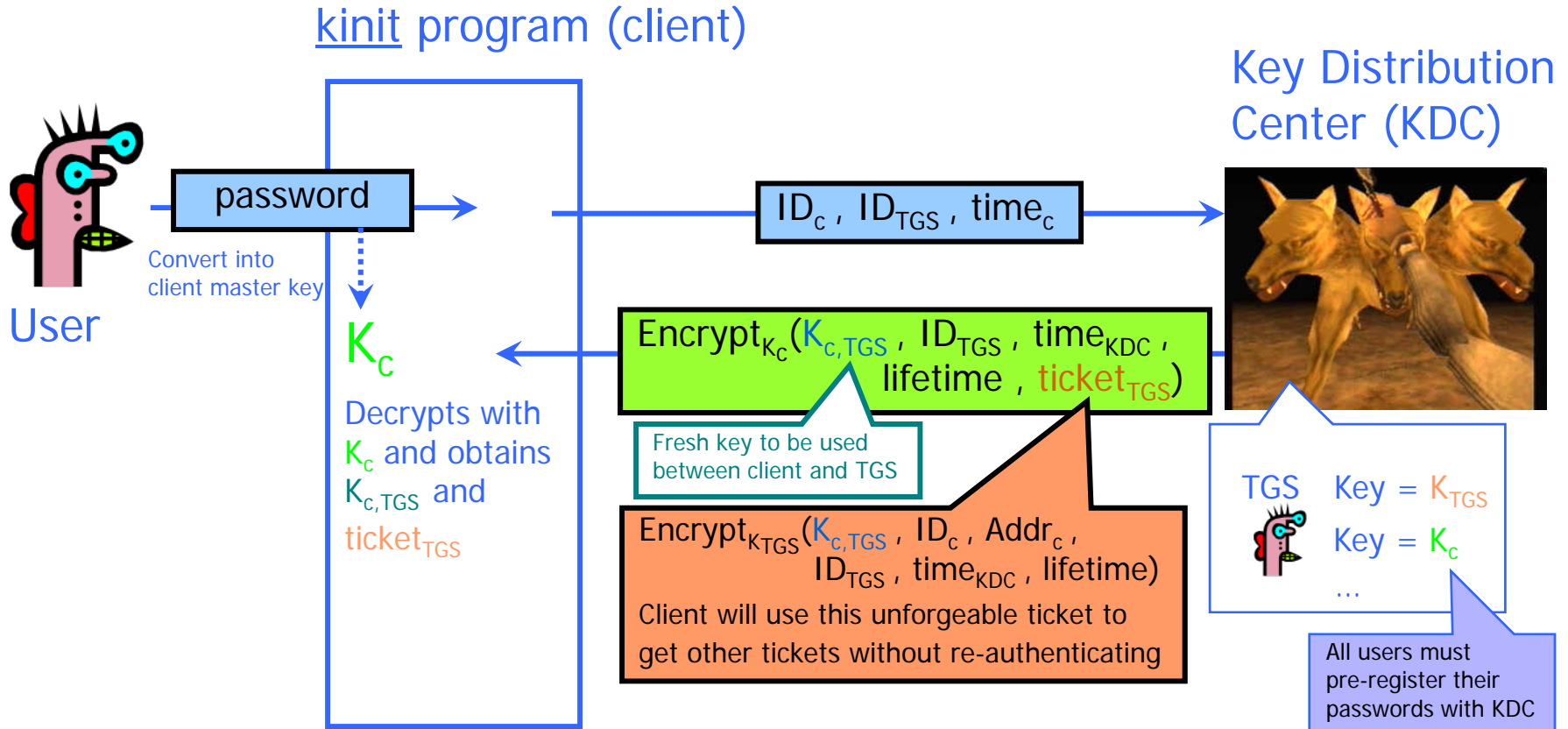
◆ No server authentication

- 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

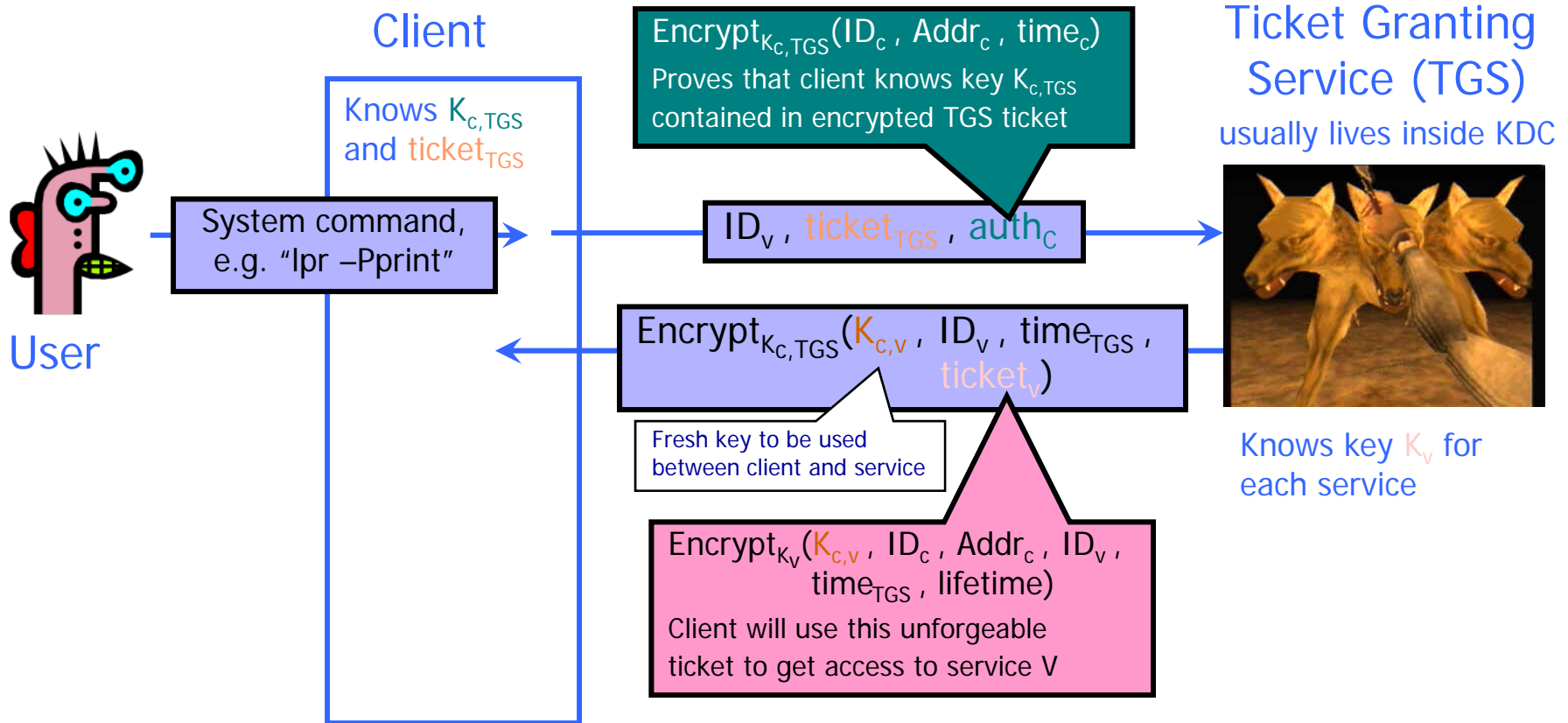
- ◆ K_C is long-term key of client C
 - Derived from user's password
 - Known to client and key distribution center (KDC)
- ◆ K_{TGS} is long-term key of TGS
 - Known to KDC and ticket granting service (TGS)
- ◆ K_V is long-term key of network service V
 - Known to V and TGS; separate key for each service
- ◆ $K_{C,TGS}$ is short-term key between C and TGS
 - Created by KDC, known to C and TGS
- ◆ $K_{C,V}$ is short-term key between C and V
 - Created by TGS, known to C and V

"Single Logon" Authentication



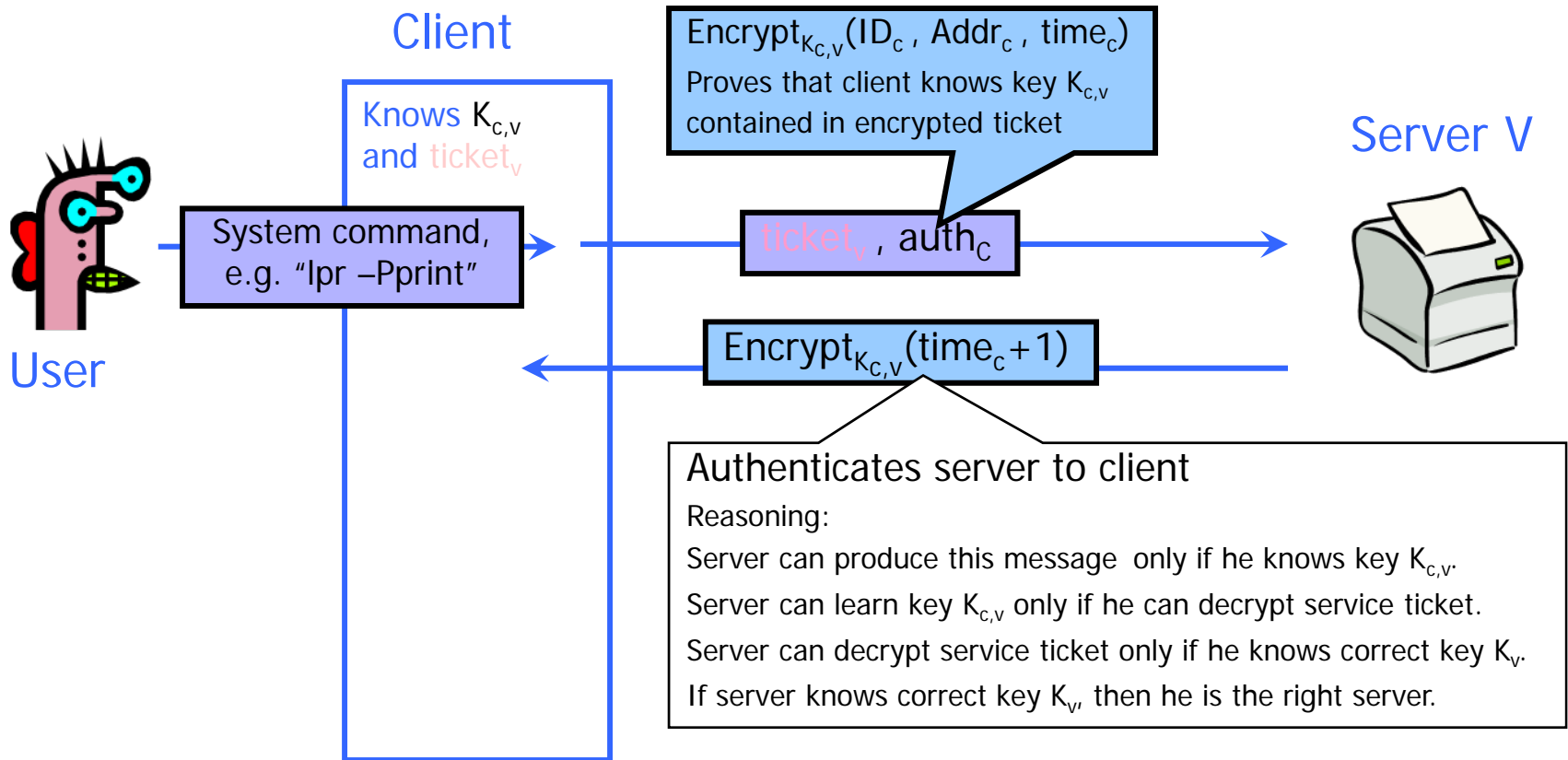
- ◆ Client only needs to obtain TGS ticket **once** (say, every morning)
 - Ticket is encrypted; client cannot forge it or tamper with it

Obtaining a Service Ticket



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

Obtaining Service

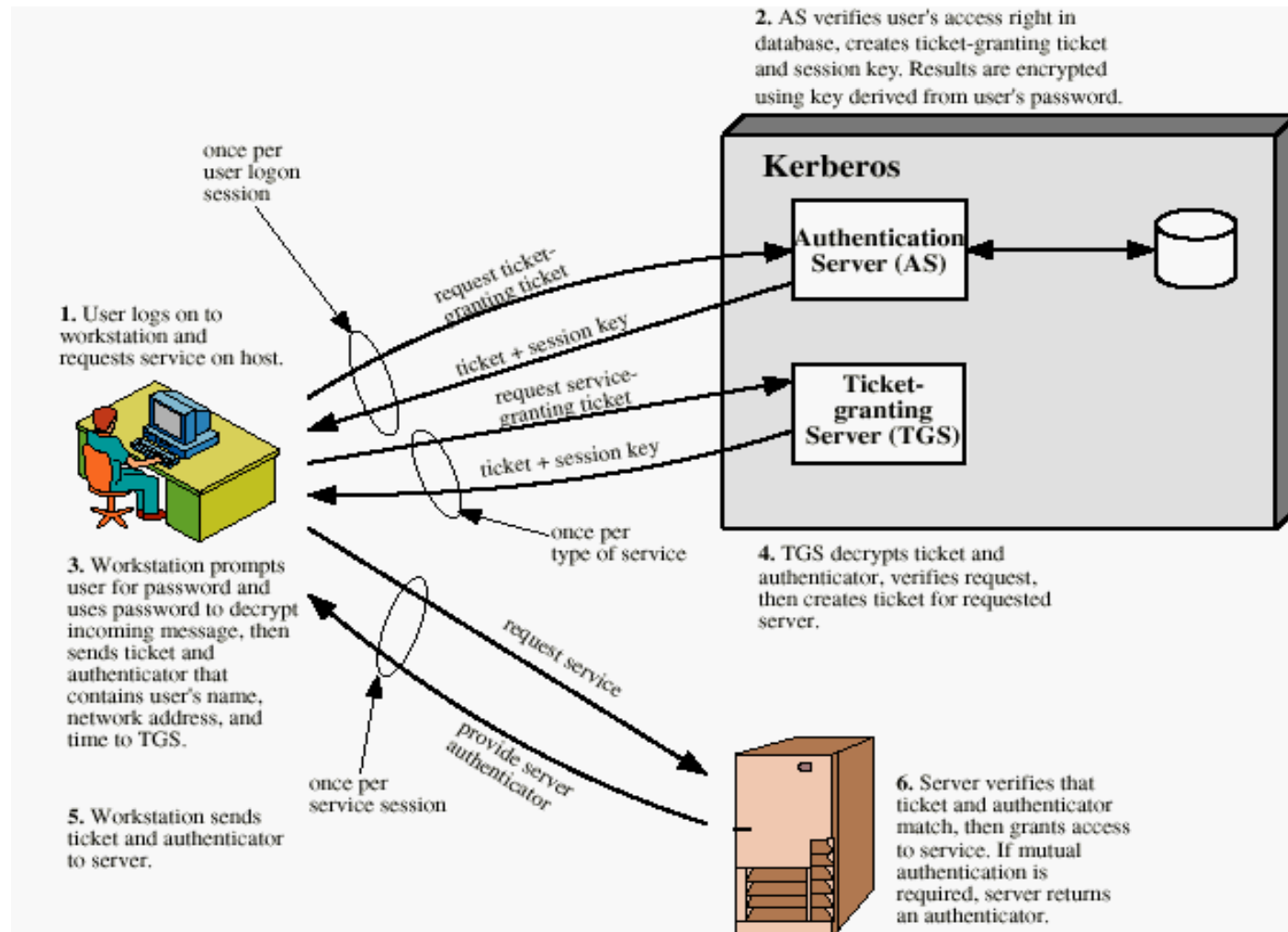


- ◆ For each service request, client uses the short-term key for that service and the ticket he received from TGS

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 interoperoperation

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

◆ Symmetric cryptography only

Problematic Issues

- ◆ Password dictionary attacks on client master keys
- ◆ Replay of authenticators
 - 5-minute lifetimes long enough for replay
 - Timestamps assume global, secure synchronized clocks
 - Challenge-response would have been better
- ◆ Same user-server key used for all sessions
- ◆ Homebrewed PCBC mode of encryption
- ◆ Extraneous double encryption of tickets
- ◆ No ticket delegation
 - Printer can't fetch email from server on your behalf

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
 - Servers can access other servers on user's behalf
- ◆ Realm hierarchies for inter-realm authentication
- ◆ Richer ticket functionality
- ◆ Explicit integrity checking + standard CBC mode
- ◆ Multiple encryption schemes, not just DES

Practical Uses of Kerberos

- ◆ Email, FTP, network file systems and 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, telnet
- ◆ Secure windowing systems
 - xdm, kx