CS 361S

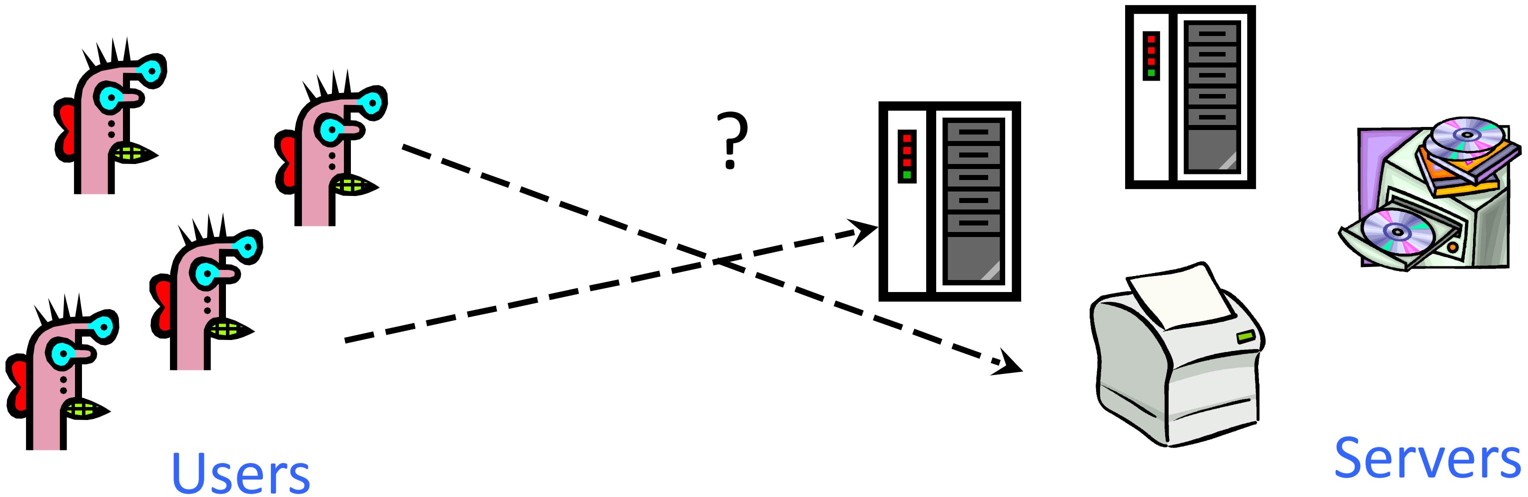
Kerberos

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

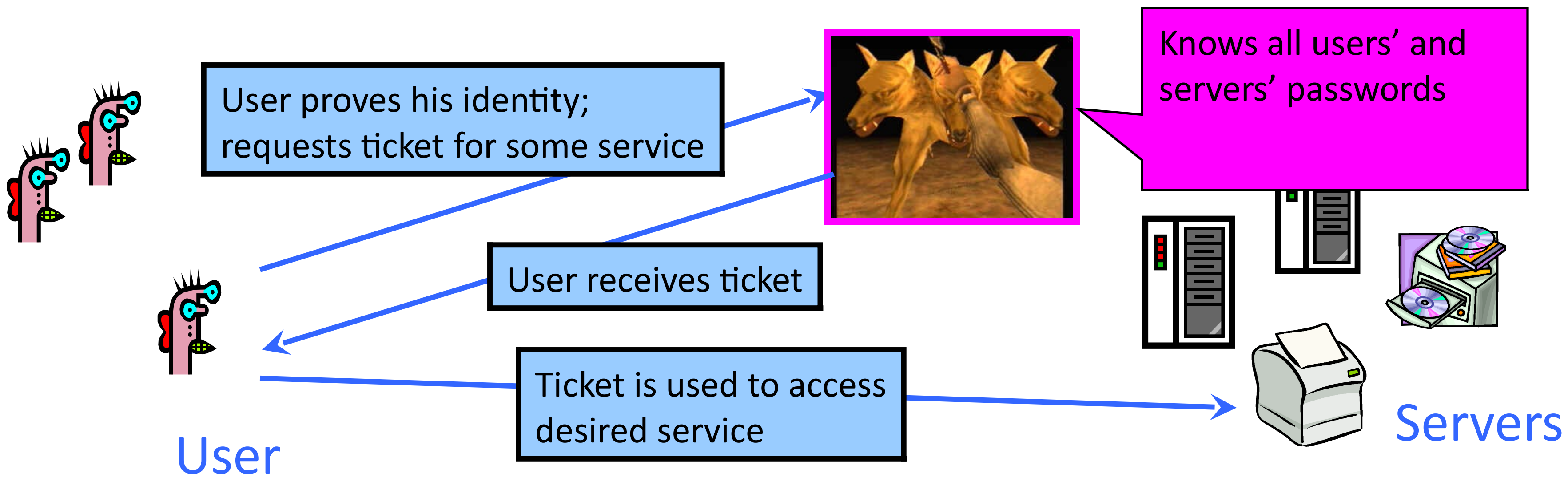
- Security
  - … against attacks by passive eavesdroppers and actively malicious users
- Transparency
  - Users shouldn't notice authentication taking place
  - Entering password is Ok, if done rarely
- Scalability
  - Large number of users and servers

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

Slide contains a diagram

Solution: Trusted Third Party

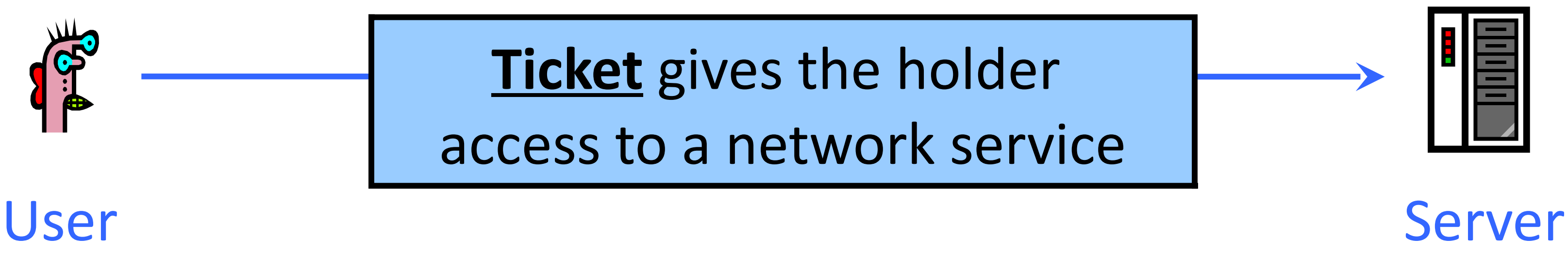


- Trusted authentication service on the network
  - Knows all passwords, can grant access to any server
  - Convenient (but also the single point of failure!)
  - Requires high level of physical security

Diagram transcription

The user proves his identity to a single service that knows all users and their passwords. In return, the user receives a “ticket” which can then be used to access the desired service.

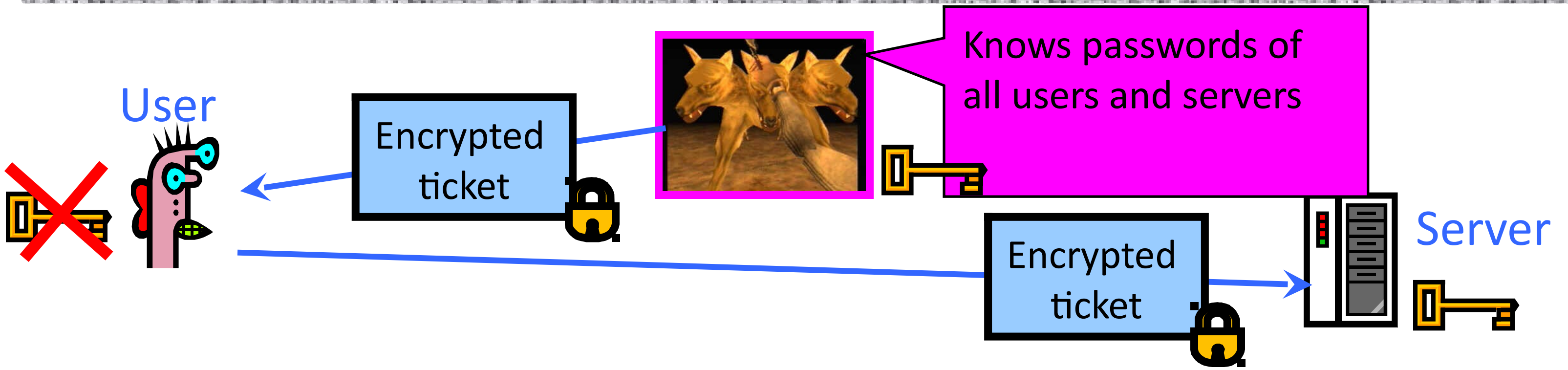
What Should a Ticket Look Like?



- 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

Slide contains a diagram

What Should a Ticket Include?



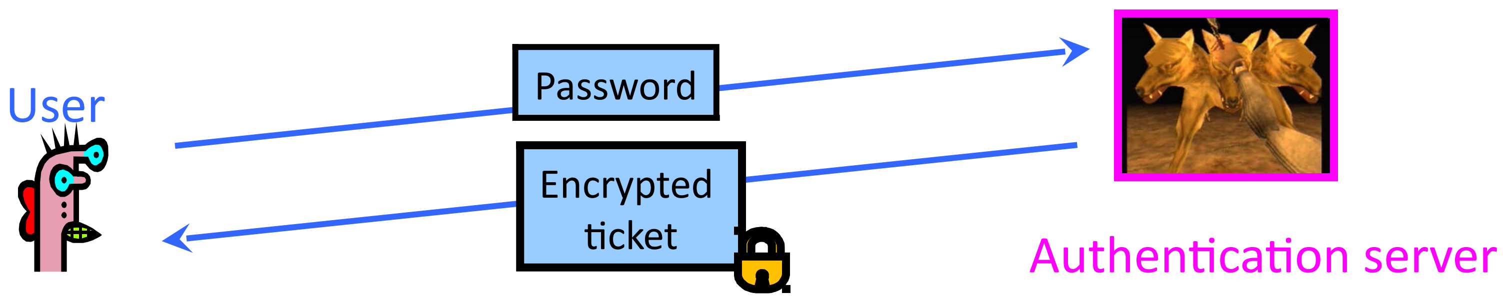
- User name
- Server name
- Address of user's workstation
  - Otherwise, a user on another workstation can steal the ticket and use it to gain access to the server
- Ticket lifetime
- A few other things (session key, etc.)

Diagram transcription

This diagram emphasizes that the ticket provided to the user by the authentication service is encrypted. The key used for this encryption is *not* held by the user, but is held by the service for which this ticket is to be used.

Slide contains a diagram

Naïve Authentication



- 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.

Diagram transcription

The diagram illustrates a naive way for the user to authenticate with the authentication service: the user sends its password, and then the service responds with a ticket

Slide contains a diagram

Two-Step Authentication

- Prove identity <u>once</u> to obtain a special <u>TGS ticket</u>
- Use TGS to get tickets for any network service

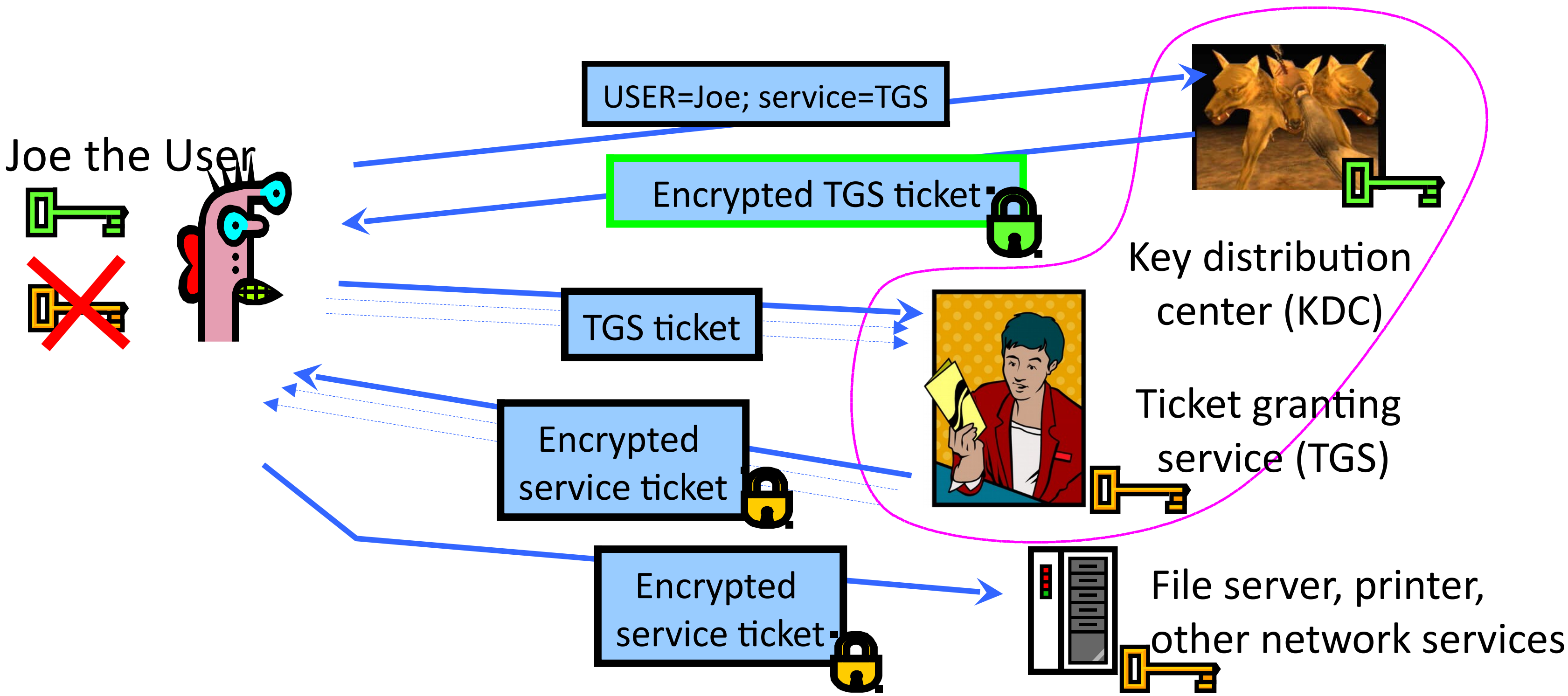


Diagram transcription

The authentication service is now split up into two parties: the key distribution center and the ticket granting service. The user only has to prove its identity *once* to the key distribution center, which gives the user a “ticket granting ticket”. The user can then use this ticket to authenticate to the ticket granting service, which grants tickets for other services on the network.

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

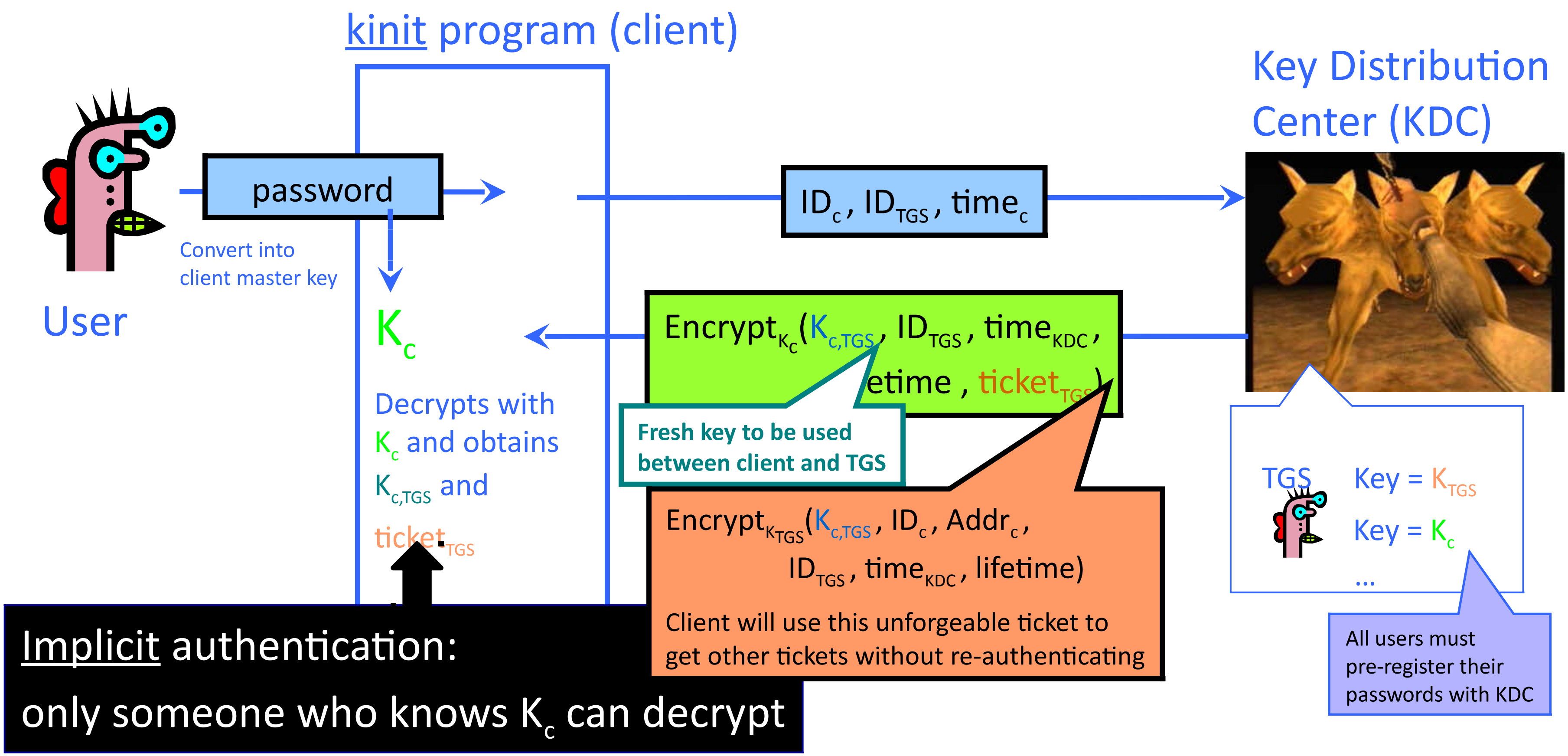
- K_c is <u>long-term</u> key of client C
  - Derived from the user's password
  - Known to the client and the key distribution center (KDC)
- K_{TGS} is <u>long-term</u> key of TGS
  - Known to KDC and the ticket granting service (TGS)
- K_v is <u>long-term</u> key of network service V
  - Known to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

Subsequent diagram transcriptions

The following three slides display the structure of how the user authenticates with the KDC (key distribution center), how the user obtains a service ticket from the TGS (ticket granting service), and how the user obtains a service (e.g. the use of a printer). These are described in detail in the assigned readings.

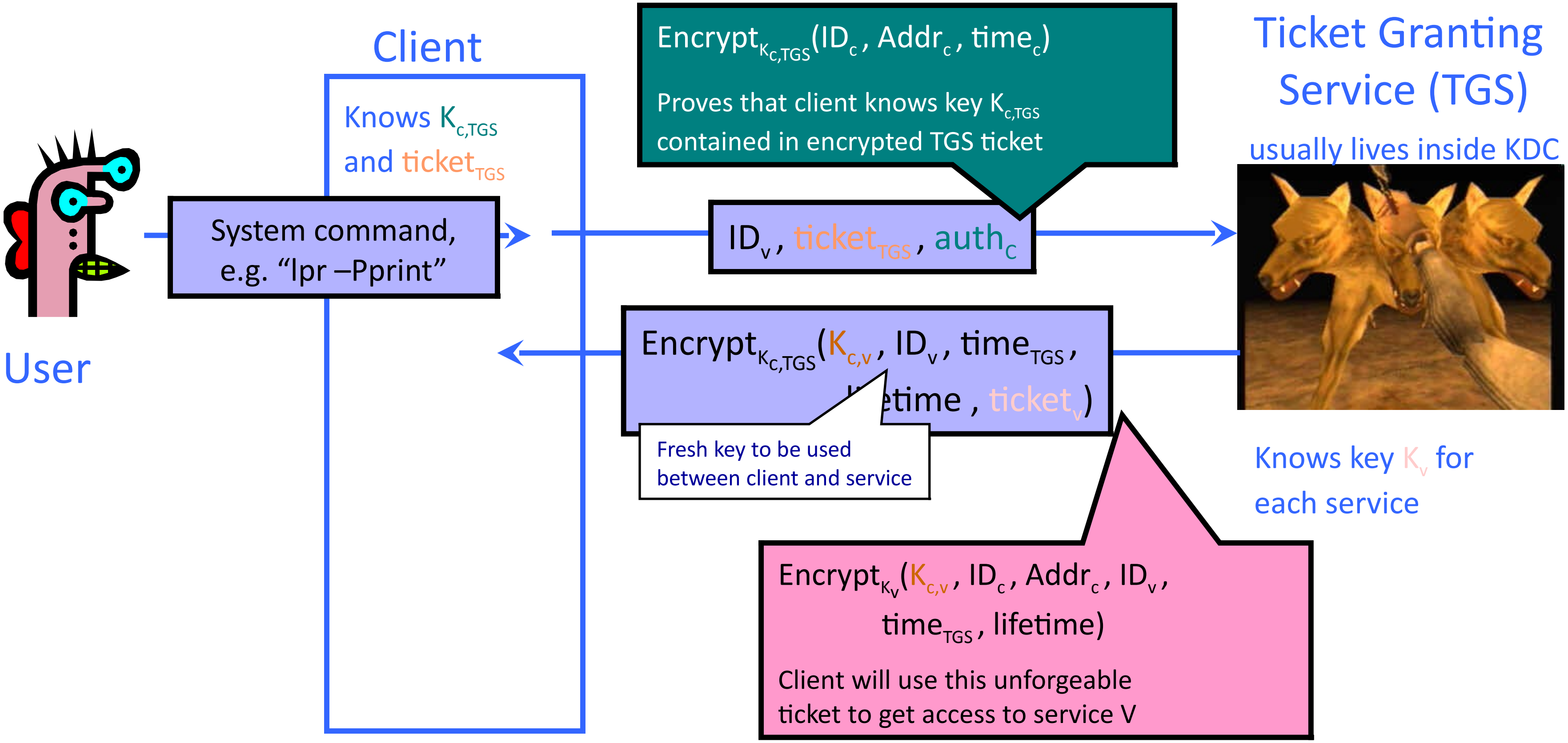
Slide contains a diagram

"Single Logon" Authentication



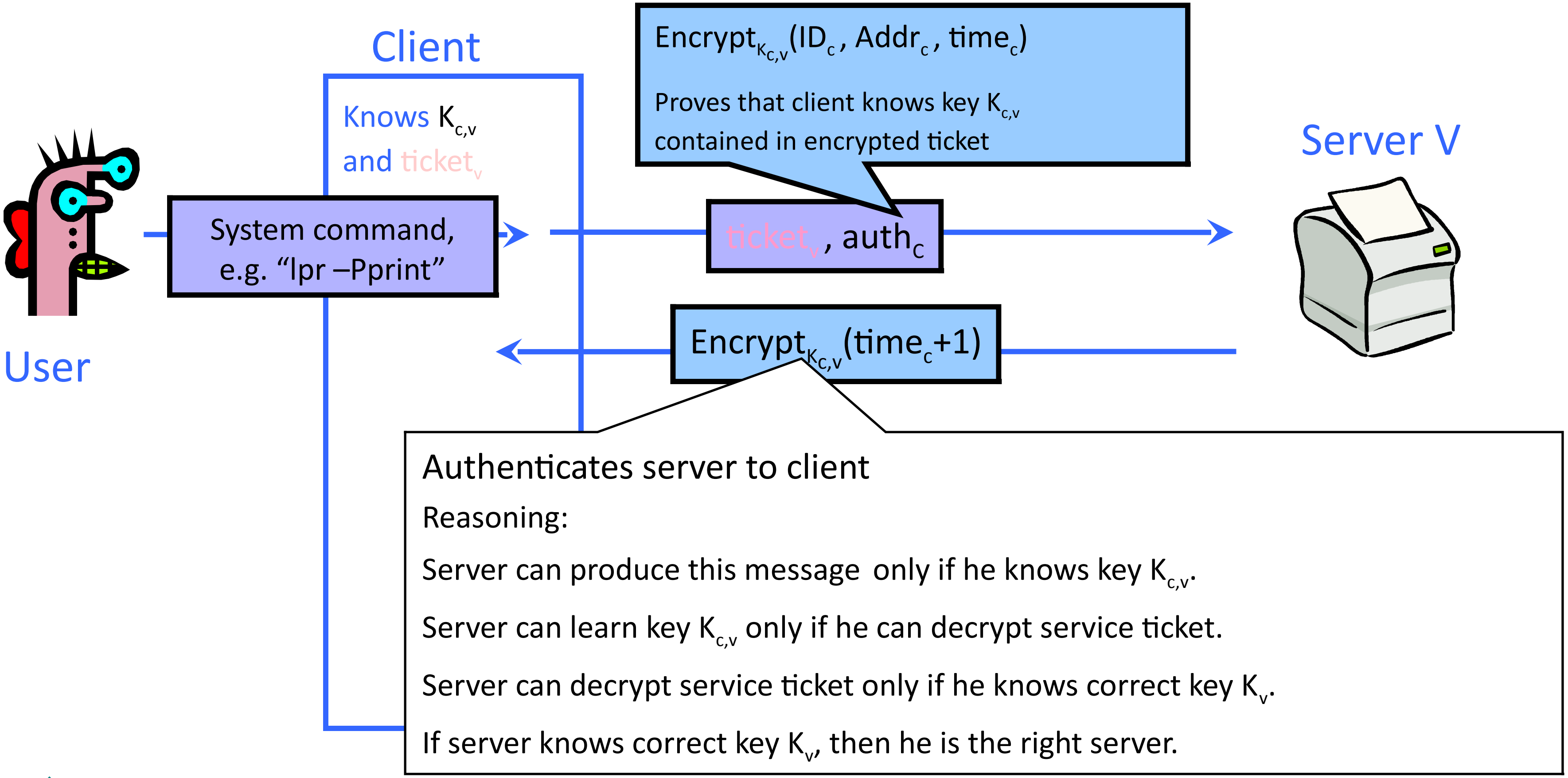
- Client only needs to obtain TGS ticket **once** (say, every morning)
- Ticket is encrypted; client cannot forge it or tamper with it

Obtaining a Service Ticket



- Client uses TGS ticket to obtain a service ticket and a <u>short-term session key</u> for each network 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 interoperation

Summary of Kerberos

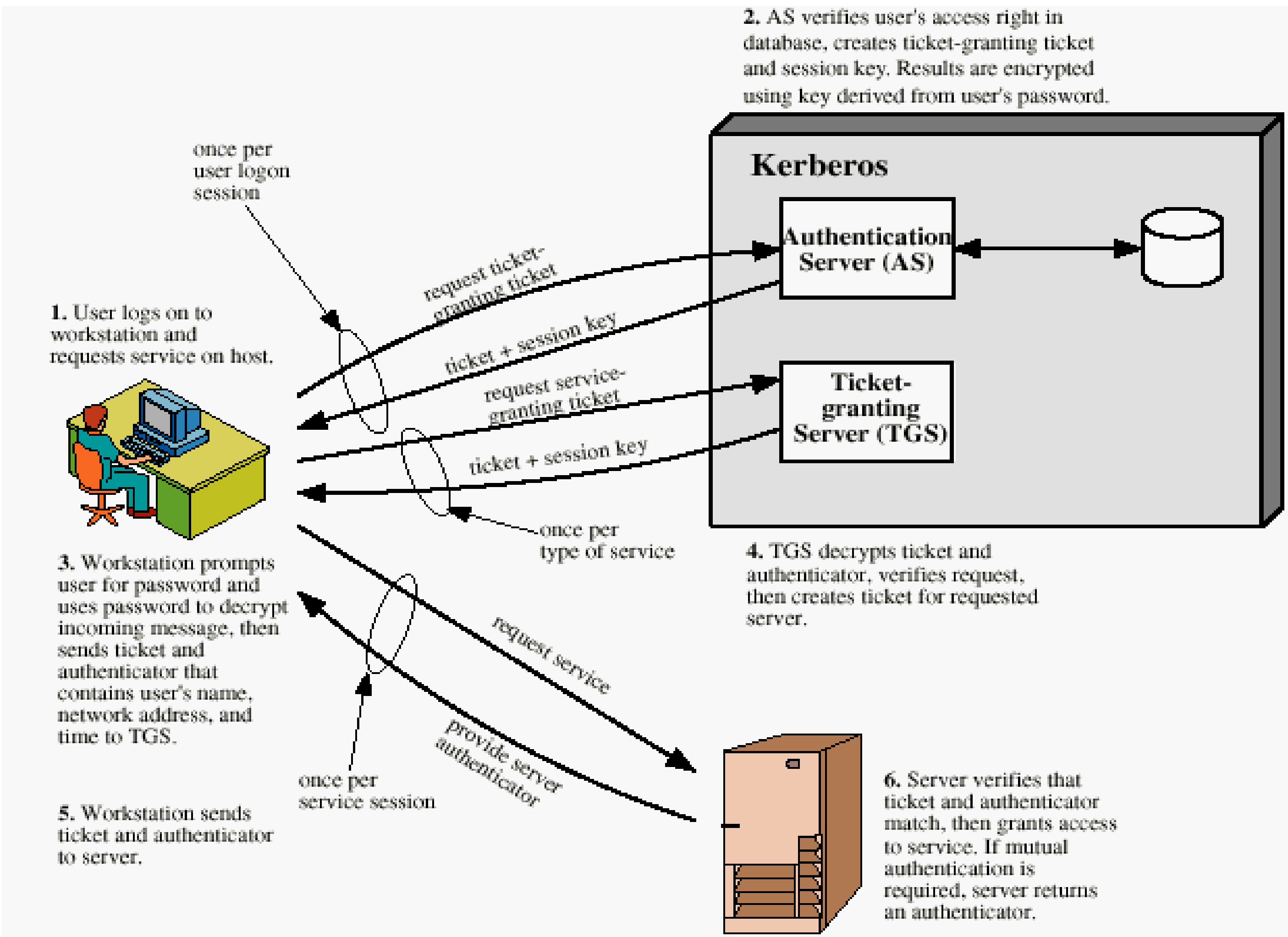


Diagram transcription

The user logs into his workstation and requests a service. The authentication server (KDC) verifies the user's access rights in the database, creates a ticket-granting ticket and a session key, and the results are encrypted using a key derived from the user's password. The workstation prompts the user for a password, and this is used in order to decrypt the response from the KDC. decrypt the incoming message, then sends the ticket and authenticator that contains the user's name, network address, and time, to the TGS. The TGS then decrypts the ticket and authenticator, verifies the request, then creates a ticket for the requested server. The workstation sends this ticket and authenticator to the requested server, and finally the server verifies that the ticket and authenticator match before granting access to the service. If mutual authentication is required, the server may also return an authenticator to the user.

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