CIS 6930: IoT Security

Lecture 11

Prof. Kaushal Kafle

Spring 2025

Derived from slides by Adwait Nadkarni, William Enck, Micah Sherr and Patrick McDaniel

Class Notes and Clarifications

- *Quiz aftermath*:

  Static analysis vs Dynamic analysis
  - *Their findings are not mutually exclusive!*
- **Example case**:
  - Think of how you would find input validation gaps *in code vs during runtime*.

```
import os
import sys

def display_name(input_from_user):
    os.system(f"echo Hello, {input_from_user}")

if __name__ == "__main__":
    name= sys.argv[1]
    display_name(name)
```

"Alice"

"Alice; rm -rf /"

User Authentication

Web Authentication

(still based on“something you know”)

Credentials can be

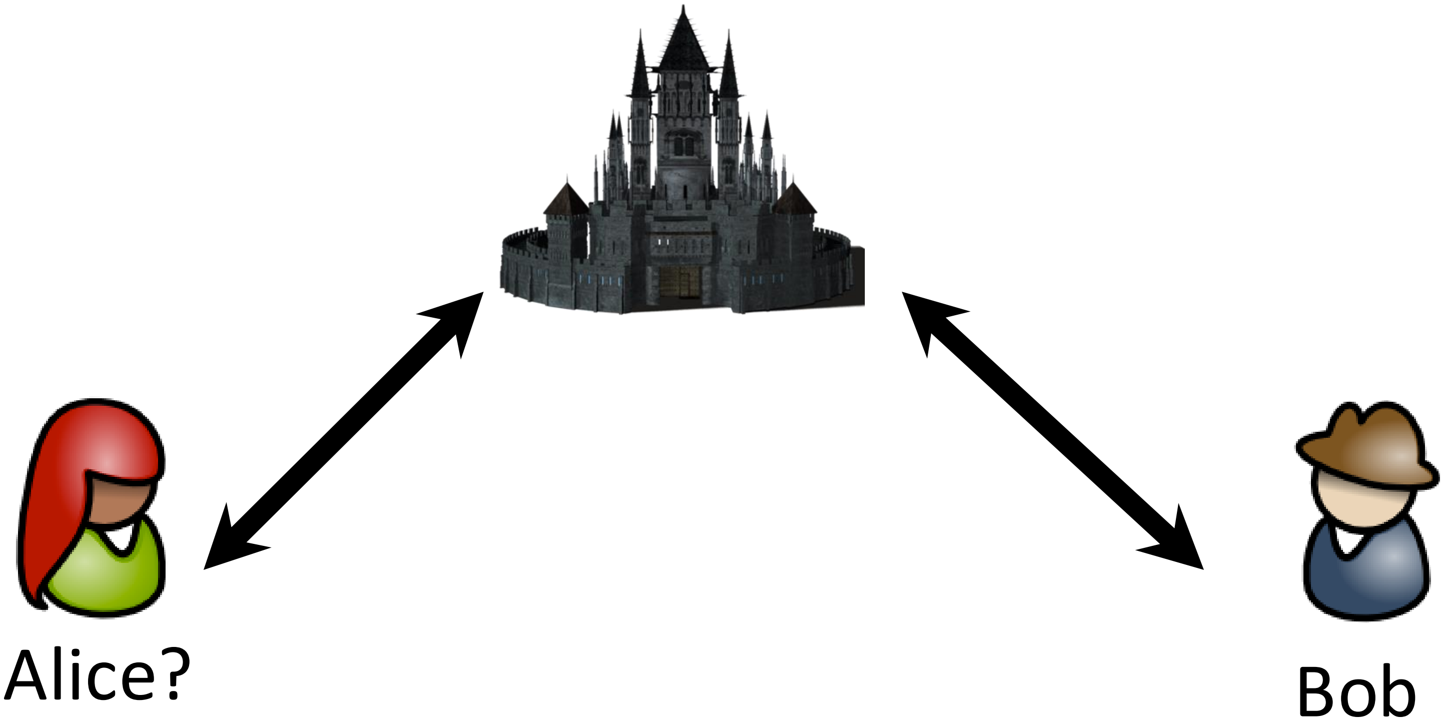
1. Something I am
2. Something I have
3. **Something I know**

Establishment of Session Keys

- D-H is the primary key-exchange protocol.
  - *Exclusive to key-exchange* i.e., does not provide encryption by itself
- Modern system use RSA to authenticate server, and DH for establishing keys.
  - E.g. DH public parameters signed by server's private key to authenticate server.
- Provides *forward secrecy* (private key compromise does not lead to session key compromise!)
  - Think what happens if a server's private keys are compromised in DH based and RSA based authentication..

Establishment of Session Keys

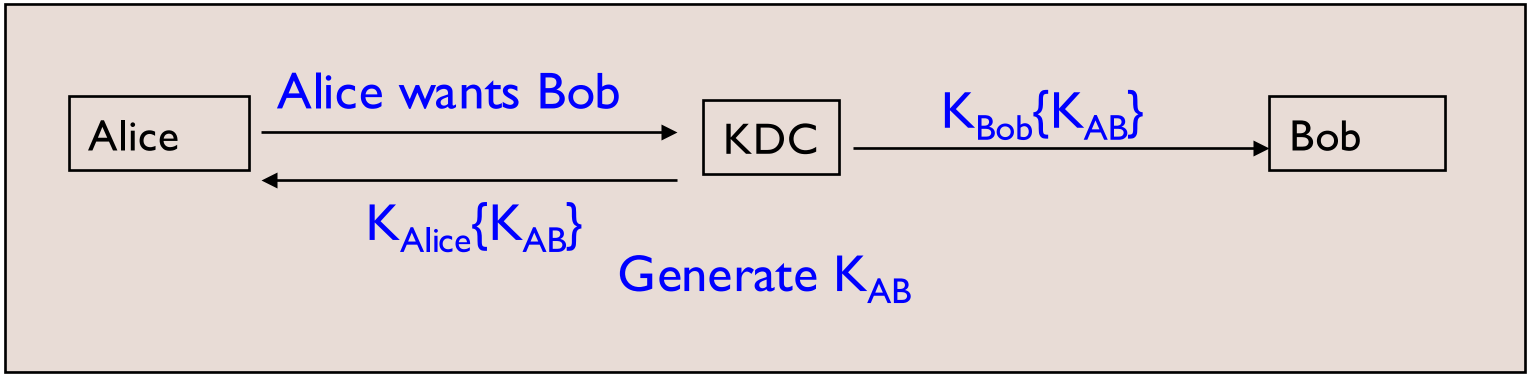
```
~ openssl ciphers -v
TLS_AES_256_GCM_SHA384         TLSv1.3 Kx=any      Au=any    Enc=AESGCM(256)            Mac=AEAD
TLS_CHACHA20_POLY1305_SHA256   TLSv1.3 Kx=any      Au=any    Enc=CHACHA20/POLY1305(256) Mac=AEAD
TLS_AES_128_GCM_SHA256         TLSv1.3 Kx=any      Au=any    Enc=AESGCM(128)            Mac=AEAD
ECDHE-ECDSA-AES256-GCM-SHA384  TLSv1.2 Kx=ECDH     Au=ECDSA  Enc=AESGCM(256)            Mac=AEAD
ECDHE-RSA-AES256-GCM-SHA384    TLSv1.2 Kx=ECDH     Au=RSA    Enc=AESGCM(256)            Mac=AEAD
DHE-RSA-AES256-GCM-SHA384      TLSv1.2 Kx=DH       Au=RSA    Enc=AESGCM(256)            Mac=AEAD
ECDHE-ECDSA-CHACHA20-POLY1305  TLSv1.2 Kx=ECDH     Au=ECDSA  Enc=CHACHA20/POLY1305(256) Mac=AEAD
ECDHE-RSA-CHACHA20-POLY1305    TLSv1.2 Kx=ECDH     Au=RSA    Enc=CHACHA20/POLY1305(256) Mac=AEAD
DHE-RSA-CHACHA20-POLY1305      TLSv1.2 Kx=DH       Au=RSA    Enc=CHACHA20/POLY1305(256) Mac=AEAD
ECDHE-ECDSA-AES128-GCM-SHA256  TLSv1.2 Kx=ECDH     Au=ECDSA  Enc=AESGCM(128)            Mac=AEAD
ECDHE-RSA-AES128-GCM-SHA256    TLSv1.2 Kx=ECDH     Au=RSA    Enc=AESGCM(128)            Mac=AEAD
DHE-RSA-AES128-GCM-SHA256      TLSv1.2 Kx=DH       Au=RSA    Enc=AESGCM(128)            Mac=AEAD
ECDHE-ECDSA-AES256-SHA384      TLSv1.2 Kx=ECDH     Au=ECDSA  Enc=AES(256)               Mac=SHA384
ECDHE-RSA-AES256-SHA384        TLSv1.2 Kx=ECDH     Au=RSA    Enc=AES(256)               Mac=SHA384
DHE-RSA-AES256-SHA256          TLSv1.2 Kx=DH       Au=RSA    Enc=AES(256)               Mac=SHA256
ECDHE-ECDSA-AES128-SHA256      TLSv1.2 Kx=ECDH     Au=ECDSA  Enc=AES(128)               Mac=SHA256
ECDHE-RSA-AES128-SHA256        TLSv1.2 Kx=ECDH     Au=RSA    Enc=AES(128)               Mac=SHA256
DHE-RSA-AES128-SHA256          TLSv1.2 Kx=DH       Au=RSA    Enc=AES(128)               Mac=SHA256
```



Mediated Authentication

Mediated Authentication (With KDC)

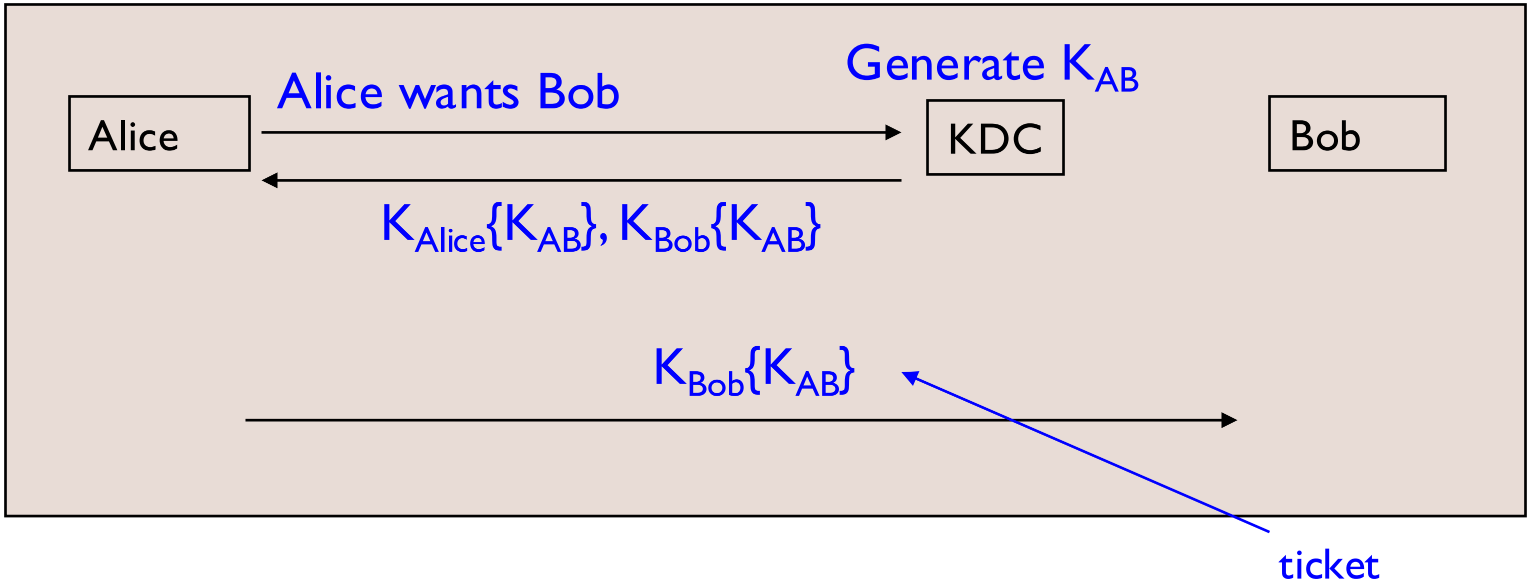
Key Distribution Center (KDC) operation (in principle)



- Some concerns
  - Trudy may claim to be Alice and talk to KDC
    - Trudy cannot get anything useful
  - Messages encrypted by Alice may get to Bob before KDC's message
  - It may be difficult for KDC to connect to Bob

Mediated Authentication (With KDC)

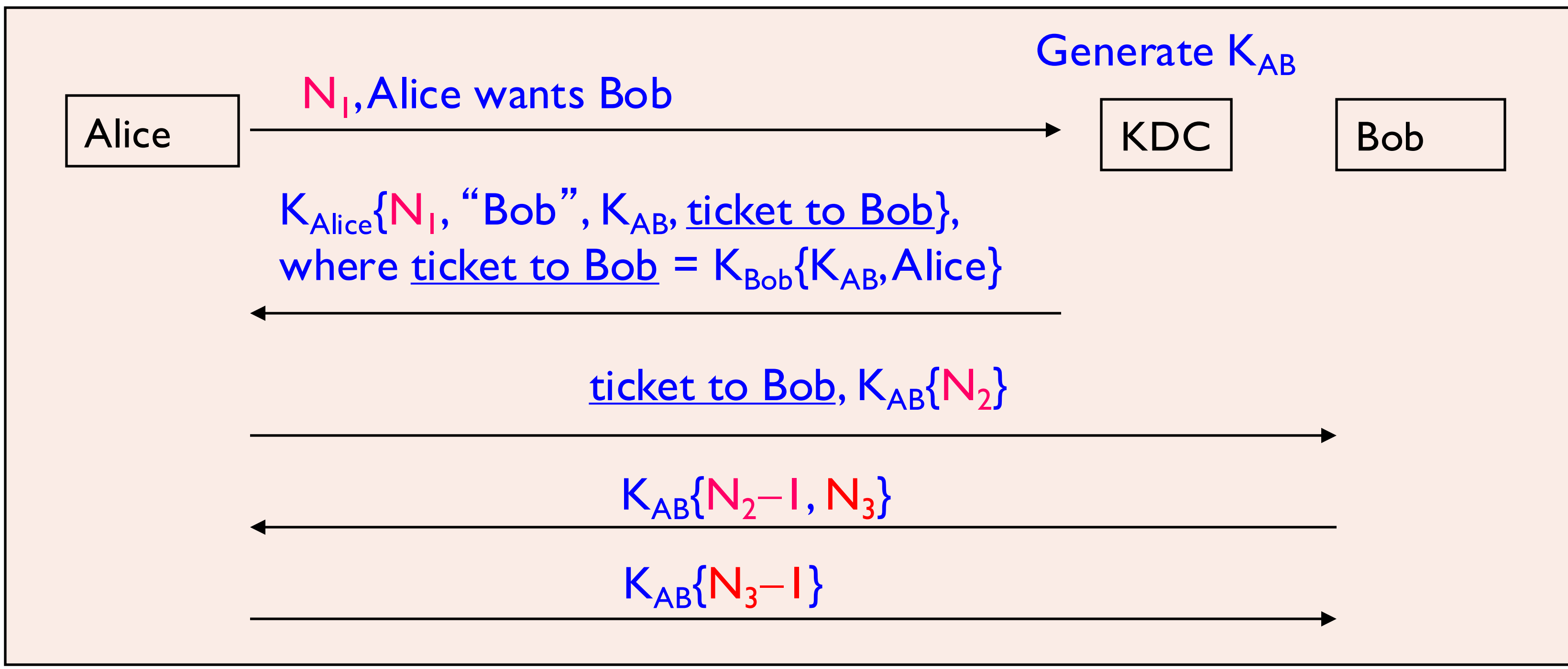
KDC operation (in practice)



- Must be followed by a mutual authentication exchange
  - To confirm that Alice and Bob have the same key

Needham-Schroeder Protocol

- Classic protocol for authentication with KDC
  - Many others have been modeled after it (e.g., Kerberos)
- Nonce: A number that is used only once
  - Deal with replay attacks



Why $\{N_x\text{-}1\}$?

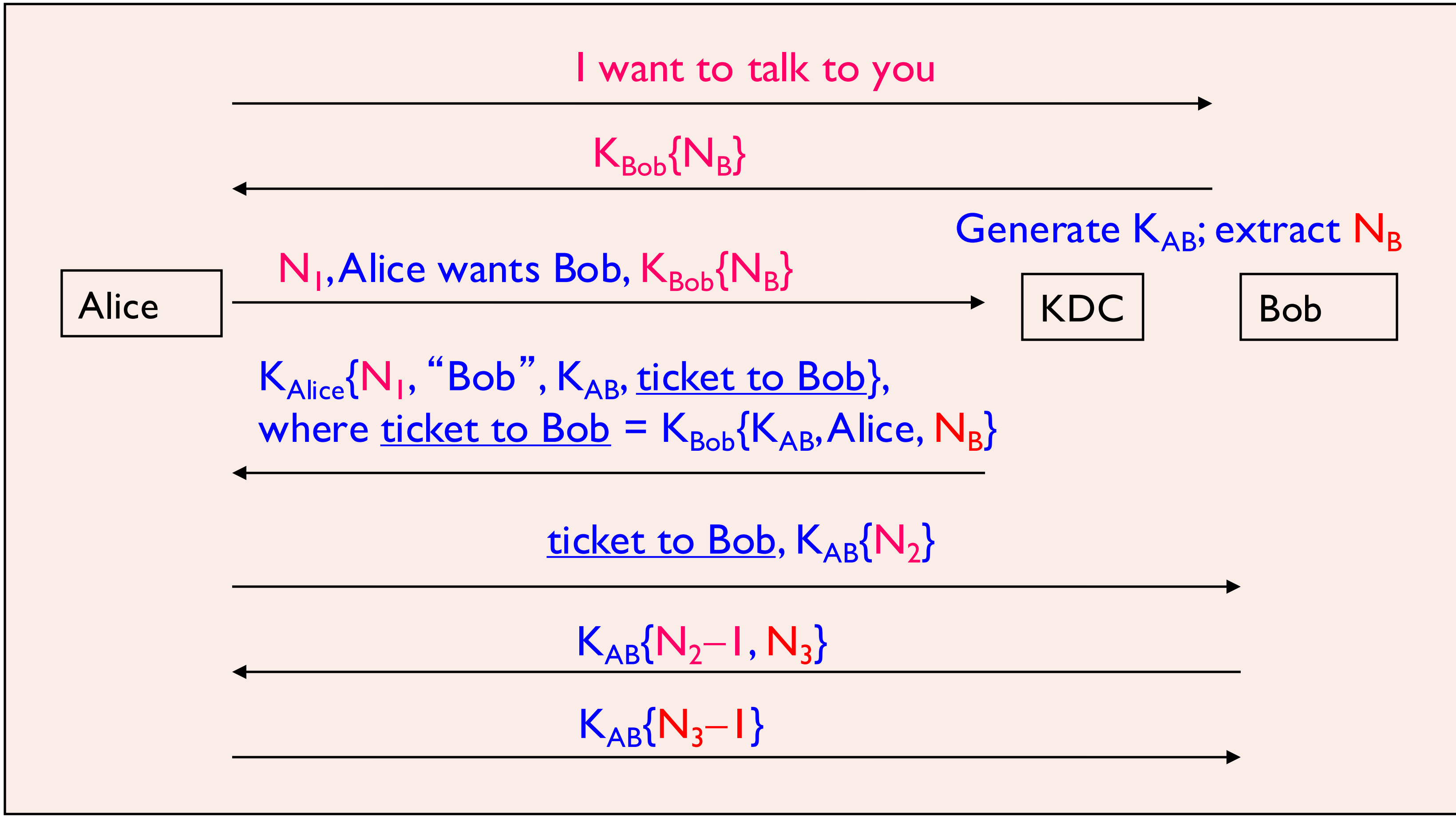
Reflection Attacks (Cont'd)

- Lesson: Don't have Alice and Bob do exactly the same thing
  - Different keys
    - Totally different keys
    - $K_{Alice\text{-}Bob} = K_{Bob\text{-}Alice} + 1$
  - *Different Challenges*
  - The initiator should be the first to prove its identity
    - Assumption: initiator is more likely to be the bad guy

Needham-Schroeder Protocol (Cont' d)

- A vulnerability
  - When Trudy gets a previous key used by Alice, Trudy may reuse a previous ticket issued to Bob for Alice
  - Essential reason
    - The ticket to Bob stays valid even if Alice changes her key

Expanded Needham-Schroeder Protocol



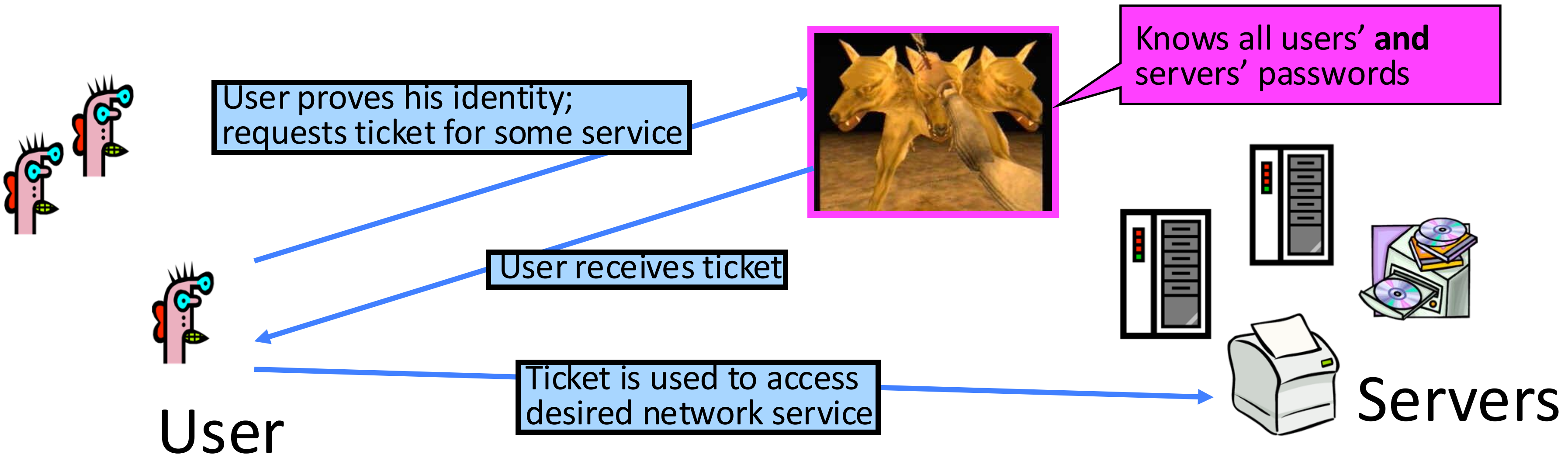
- The additional two messages assure Bob that the initiator has talked to KDC since Bob generates N_B

Kerberos

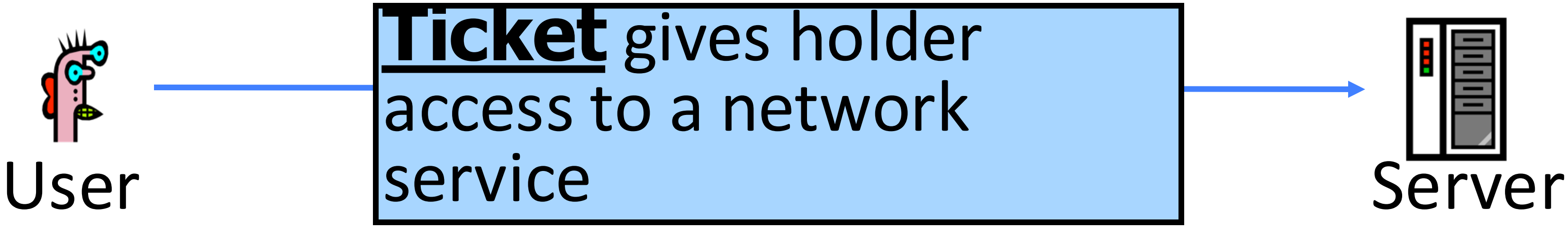
Kerberos

- An online system that resists password eavesdropping and achieves **mutual authentication**
- First single sign-on system (SSO)
- Easy application integration API
- Most widely used (non-web) centralized password system in existence
- Now part of Windows network authentication

Kerberos Overview

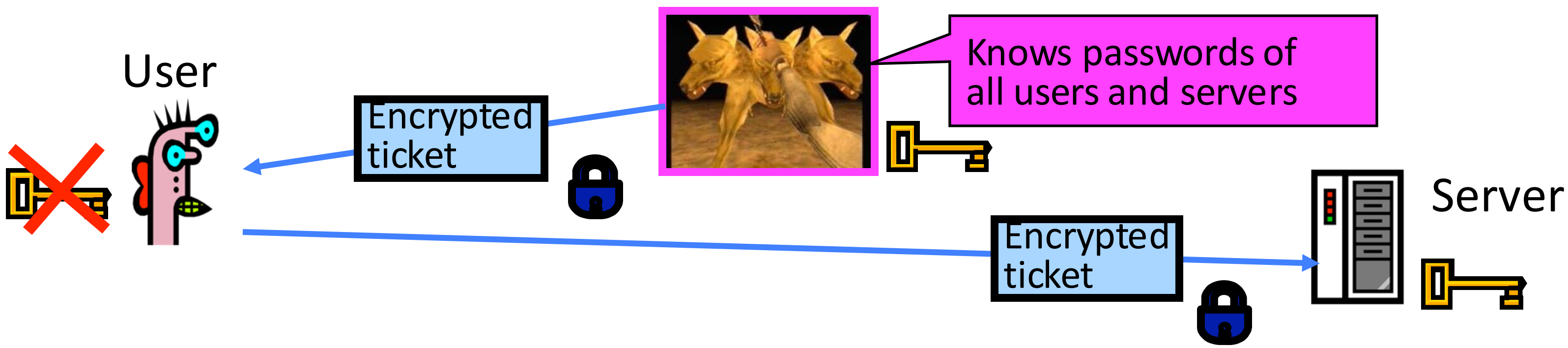


What Should a Ticket Look Like?



- 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

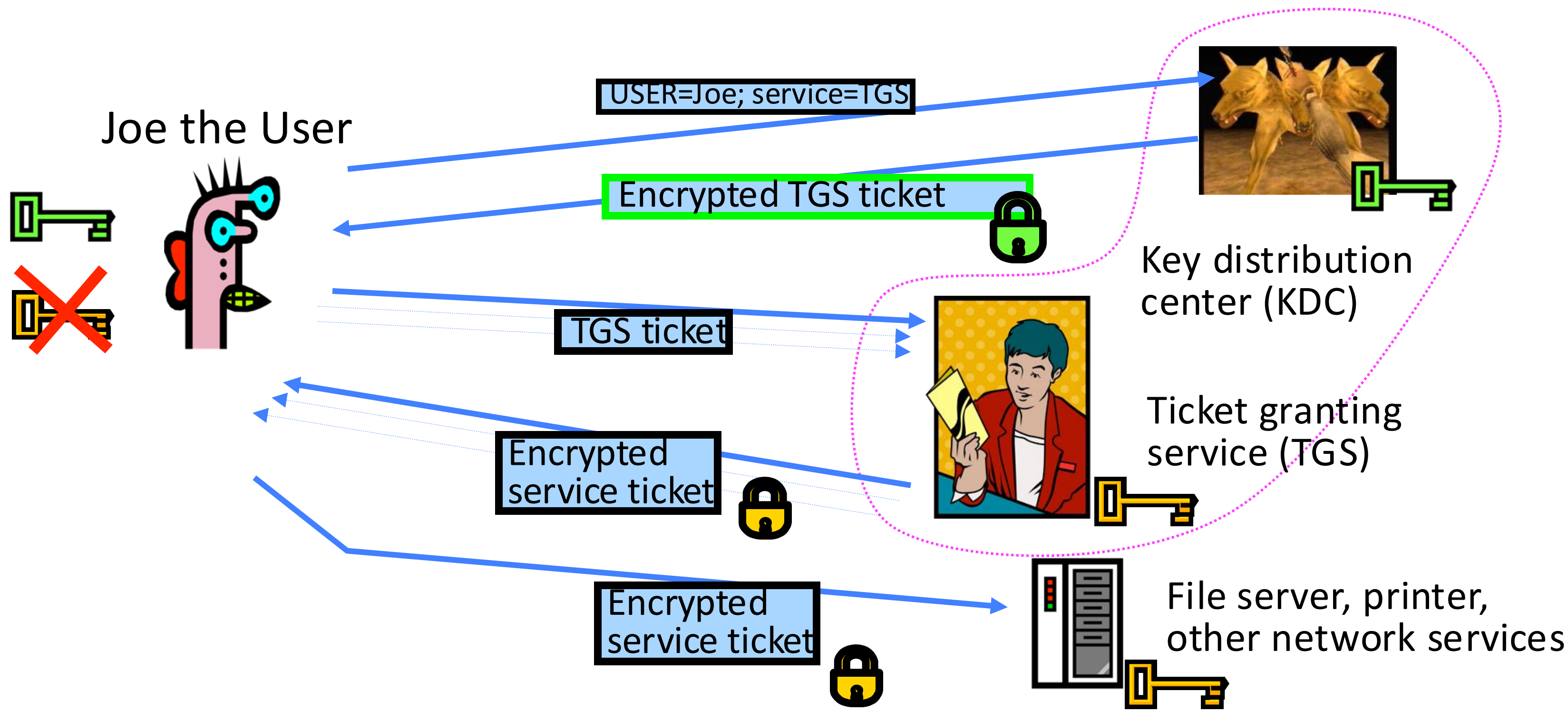
What should a ticket include?



- User name
- Server name
- Address of user's workstation -- **WHY?**
- Ticket lifetime -- **WHY?**
- A few other things (e.g., session key)

Two-Step Authentication

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



Not quite good enuf...

- **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
  - We want mutual authentication

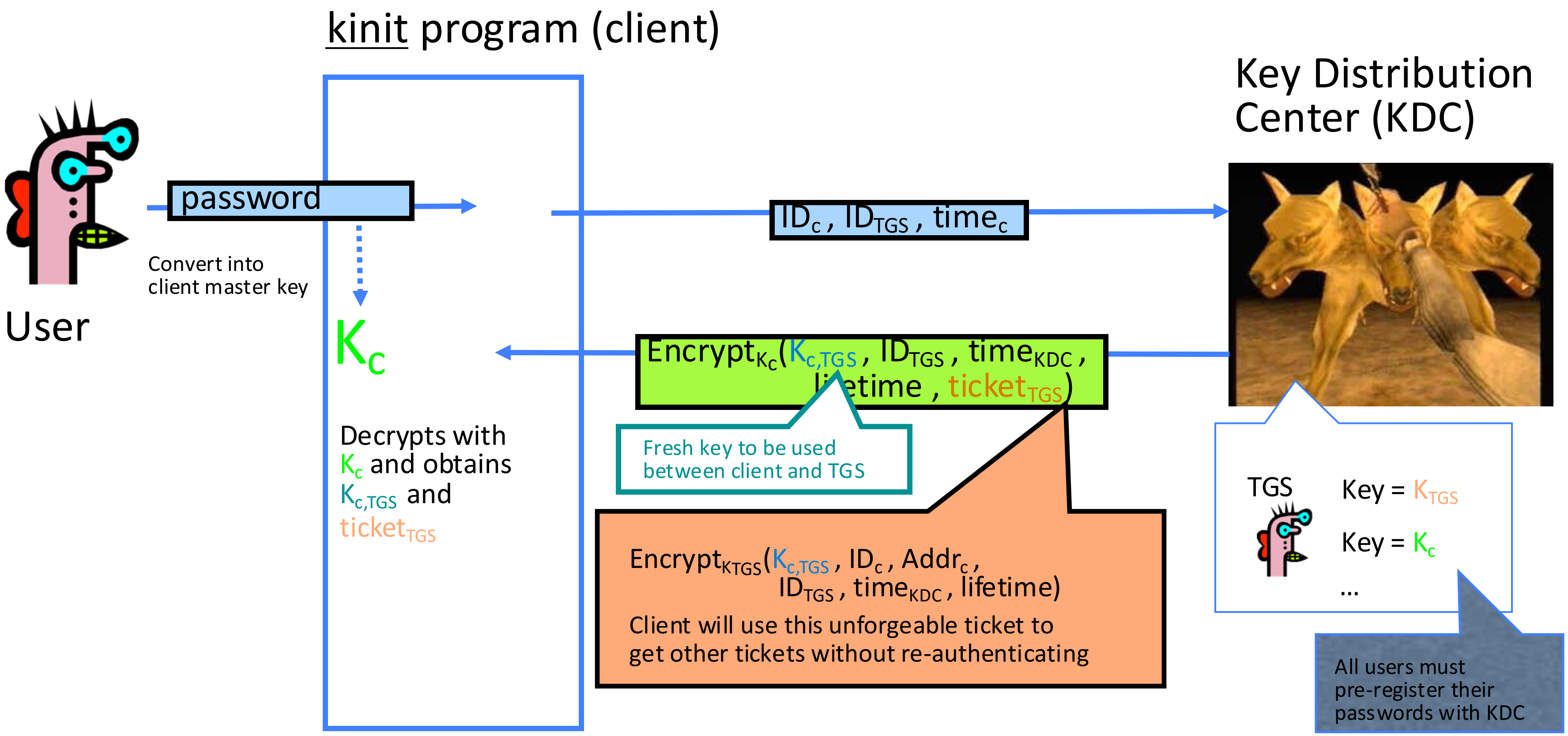
Symmetric Keys in Kerberos

- 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 *session* key between 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

Brace yourself! It’s Kerberos time!

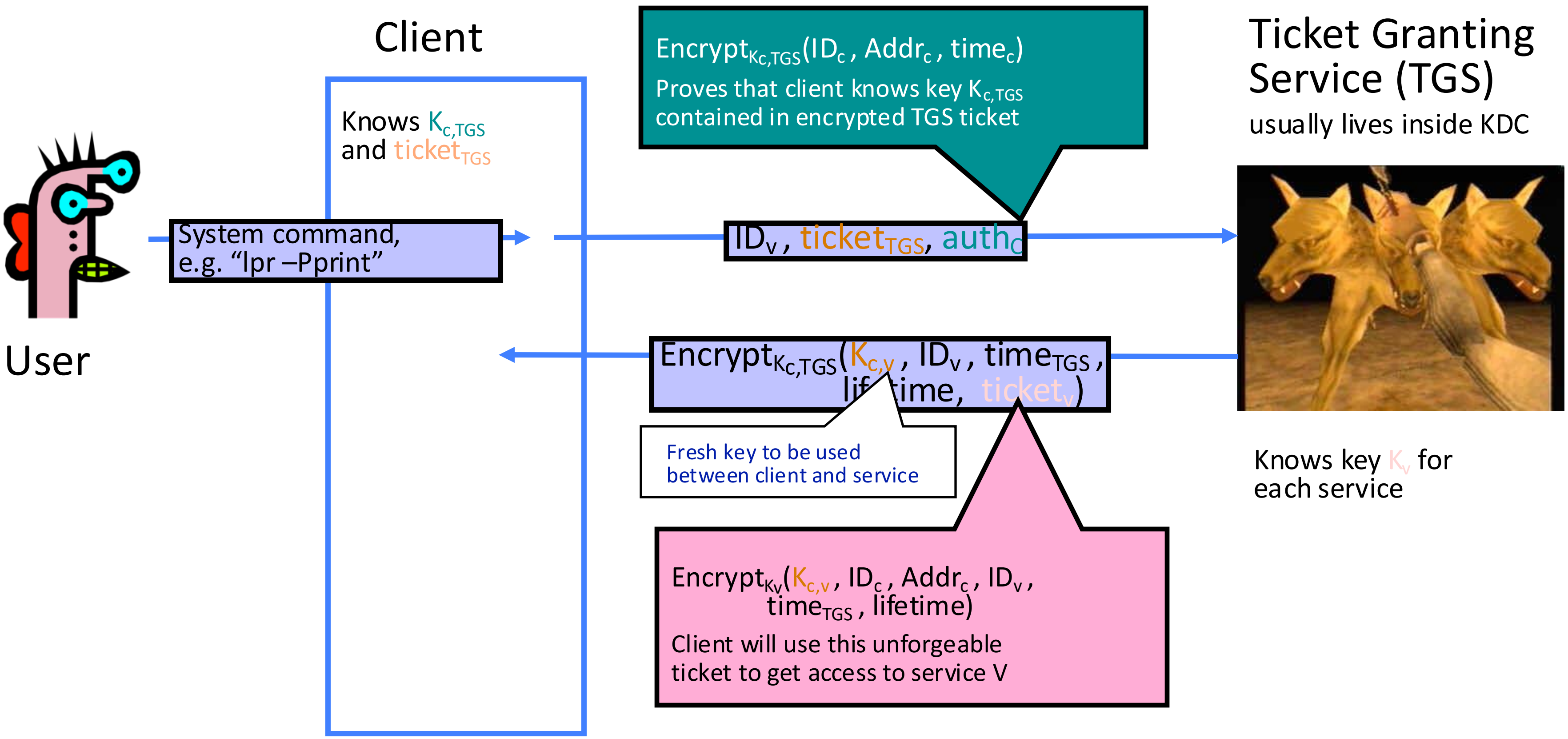
- Three-step process:
  - “Logon” -- obtain TGS ticket from KDC
  - Obtain “service ticket” from TGS
  - Use service

“Single Logon” Authentication



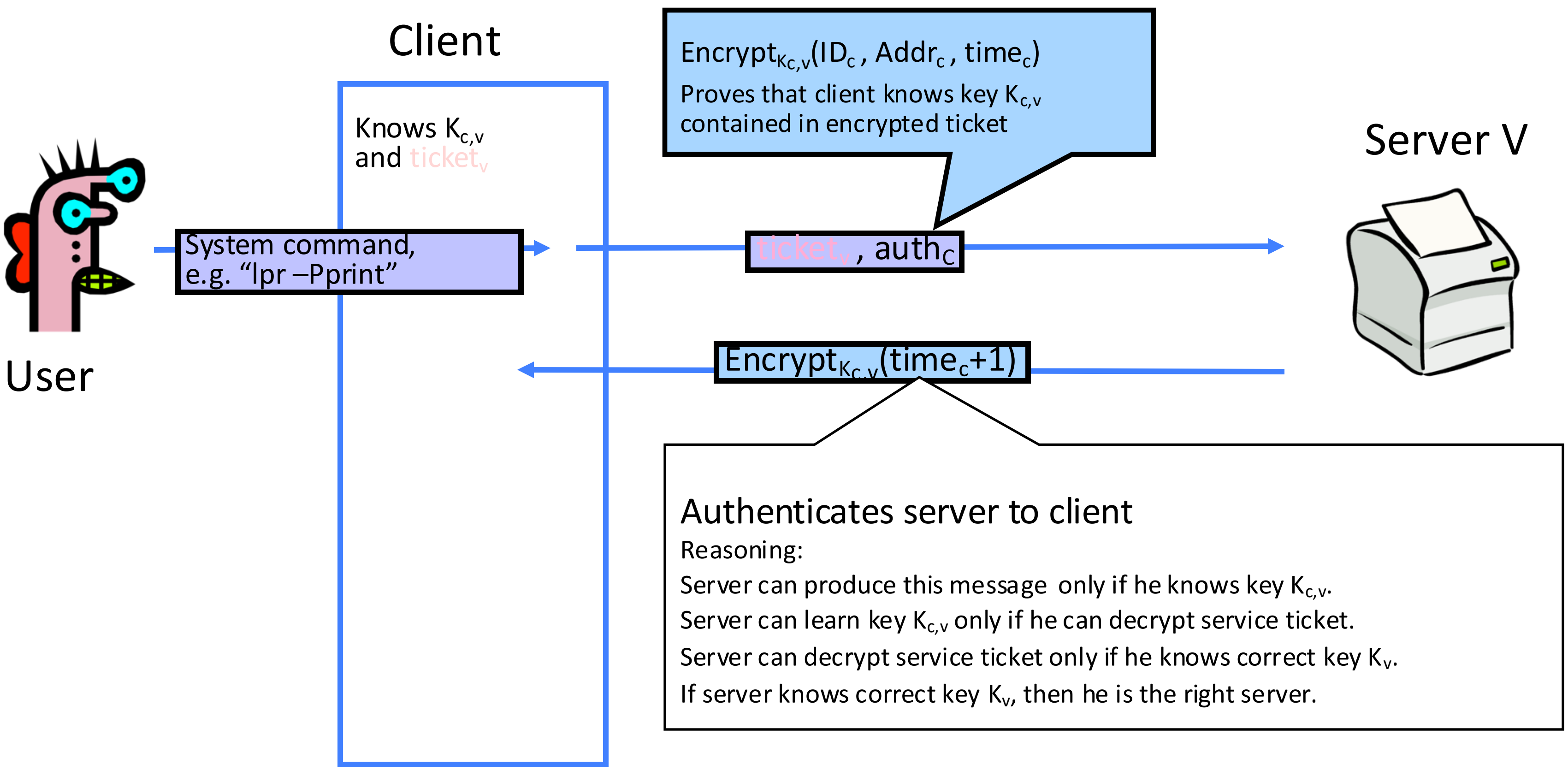
- Client only needs to obtain TGS ticket once (say, every morning)
  - Ticket is encrypted; client cannot forge it or tamper with it

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

Cross-Realm Kerberos

- Extend philosophy to more servers
  - Obtain ticket from TGS for foreign Realm
  - Supply to TGS of foreign Realm
  - Rinse and repeat as necessary

  - “There is no problem so hard in computer science that it cannot be solved by another layer of indirection.”
    - David Wheeler, Cambridge University (circa 1950)