CIS 4930: Secure IoT

Prof. Kaushal Kafle

Lecture 22

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

Class Notes

- **Few notifications**

  1. The first project grades are live.

  - If you have any questions, you can let me know via canvas message or during office hours.

  2. Last week to submit your bug bounties!

  3. Format of the final exam is the same as the midterm exam.

  4. I will talk about the final project report and the exam details in the next class.

  5. Student Assessment of Instruction

  Respond to the course assessment survey.

Web Authentication

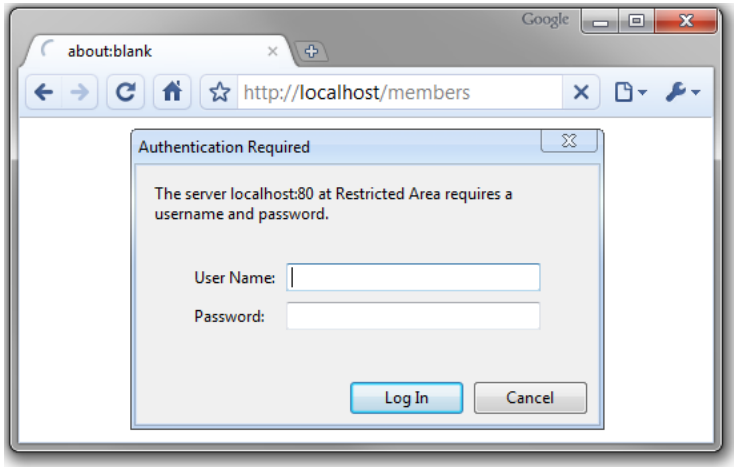
(still based on“something you know”)

Credentials can be

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

Web Authentication

- Authentication is a bi-directional process
  - Client
  - Server
  - Mutual authentication
- Several standard authentication tools
  - Basic (client)
  - Digest (client)
  - Secure Socket Layer (server, mutual)



Basic Authentication

Basic Authentication -- is this secure?

- Encoded ! = Encrypted
  - Passwords easy to intercept (base-64 encoded; not encrypted)
- Passwords:
  - easy to guess
  - easy to share
- No server authentication - easy to fool client into sending password to malicious server

Digest Authentication

Challenge/Response

- Challenge nonce is a one time random string/value

$$\text{nonce} = H(\text{IPaddress} : \text{timestamp} : \text{server secret})$$

  - more generally, a **nonce** is number or string (often randomly or pseudorandomly chosen) that is **only used once**
    - Cannot be forged by anyone else
- Response: challenge hashed with username and password

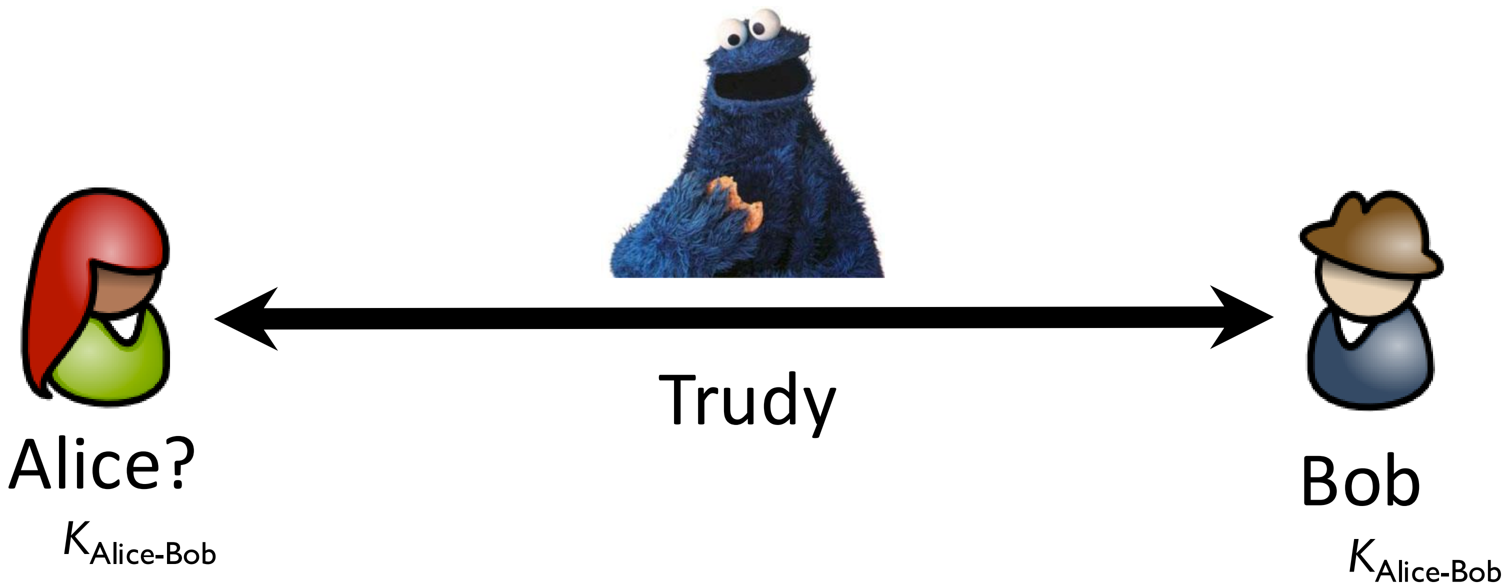
$$response = H(H(name : realm : password) : nonce : H(request))$$

Advantages of Digest over Basic

- Cleartext password never transmitted across network
- Cleartext password never stored on server
- **Replay attacks** difficult
- Intercepted response only valid for a single URL
- Shared disadvantages
  - Vulnerable to man-in-the-middle attacks (no server-side auth)
  - Document itself can be sniffed

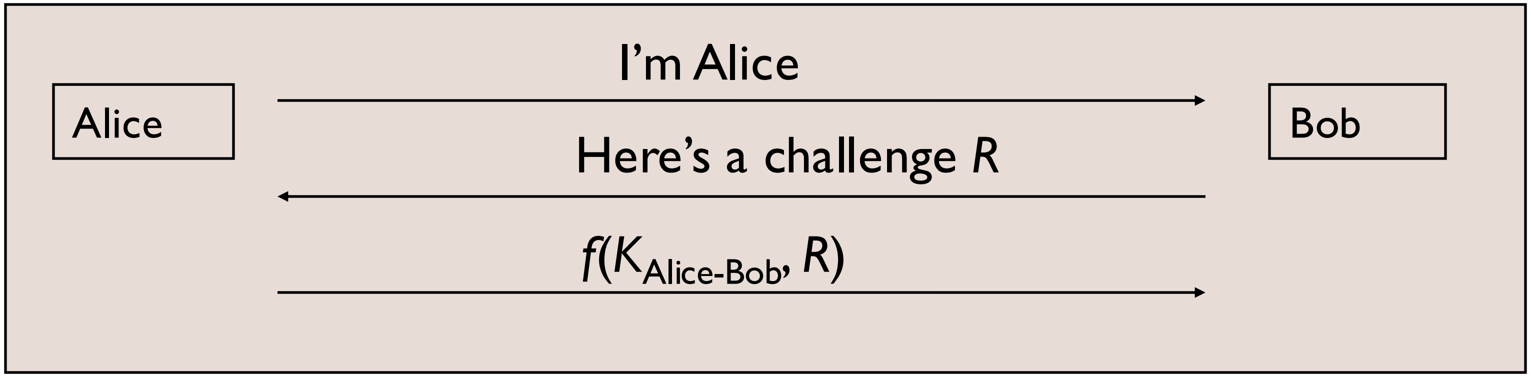
Authentication Handshakes

- Secure communication almost always includes an initial authentication handshake.
  - Authenticate each other
  - Establish session keys
  - *This process is not trivial; flaws in this process undermine secure communication*



Authentication

Authentication with Shared Secret



- Weaknesses
  - Authentication is not mutual; Trudy can convince Alice that she is Bob
  - Trudy can hijack the conversation after the initial exchange
  - If the shared key is derived from a password, Trudy can mount an off-line password guessing attack (*R is known*)
  - Trudy may compromise Bob's database and later impersonate Alice

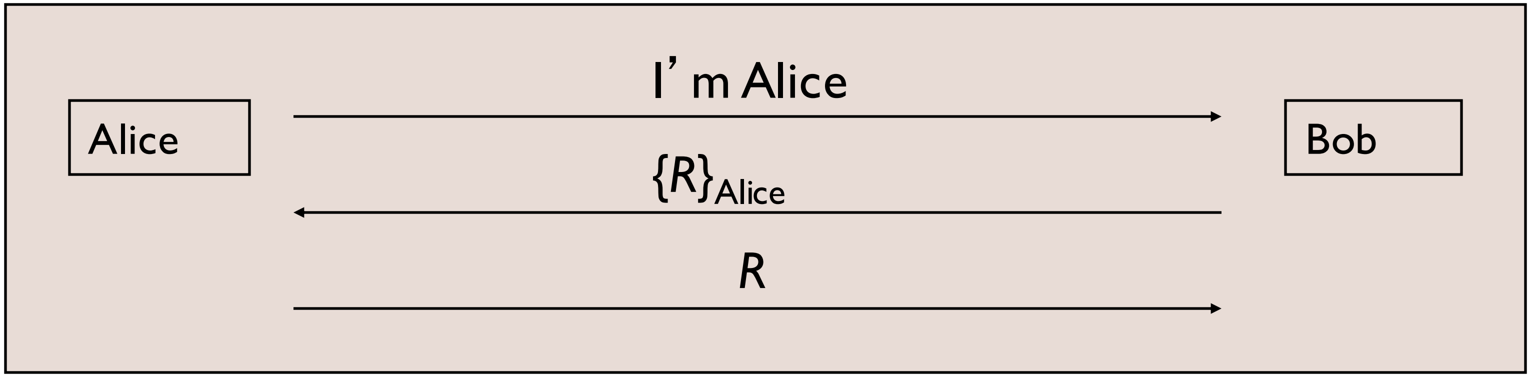
Authentication with Shared Secret (Cont'd)

- A variation
  - Requires reversible cryptography
  - Other variations are possible
- Weaknesses
  - All the previous weaknesses remain
  - Trudy doesn't have to see R to mount off-line password guessing if R has certain patterns (e.g., concatenated with a timestamp)
    - Trudy sends a message to Bob, pretending to be Alice

Authentication with Public Key

- Bob's database is less risky
- Weaknesses
  - Authentication is not mutual; Trudy can convince Alice that she is Bob
  - Trudy can hijack the conversation after the initial exchange
  - Trudy can trick Alice into signing something
    - Mitigation: Use different private key for authentication

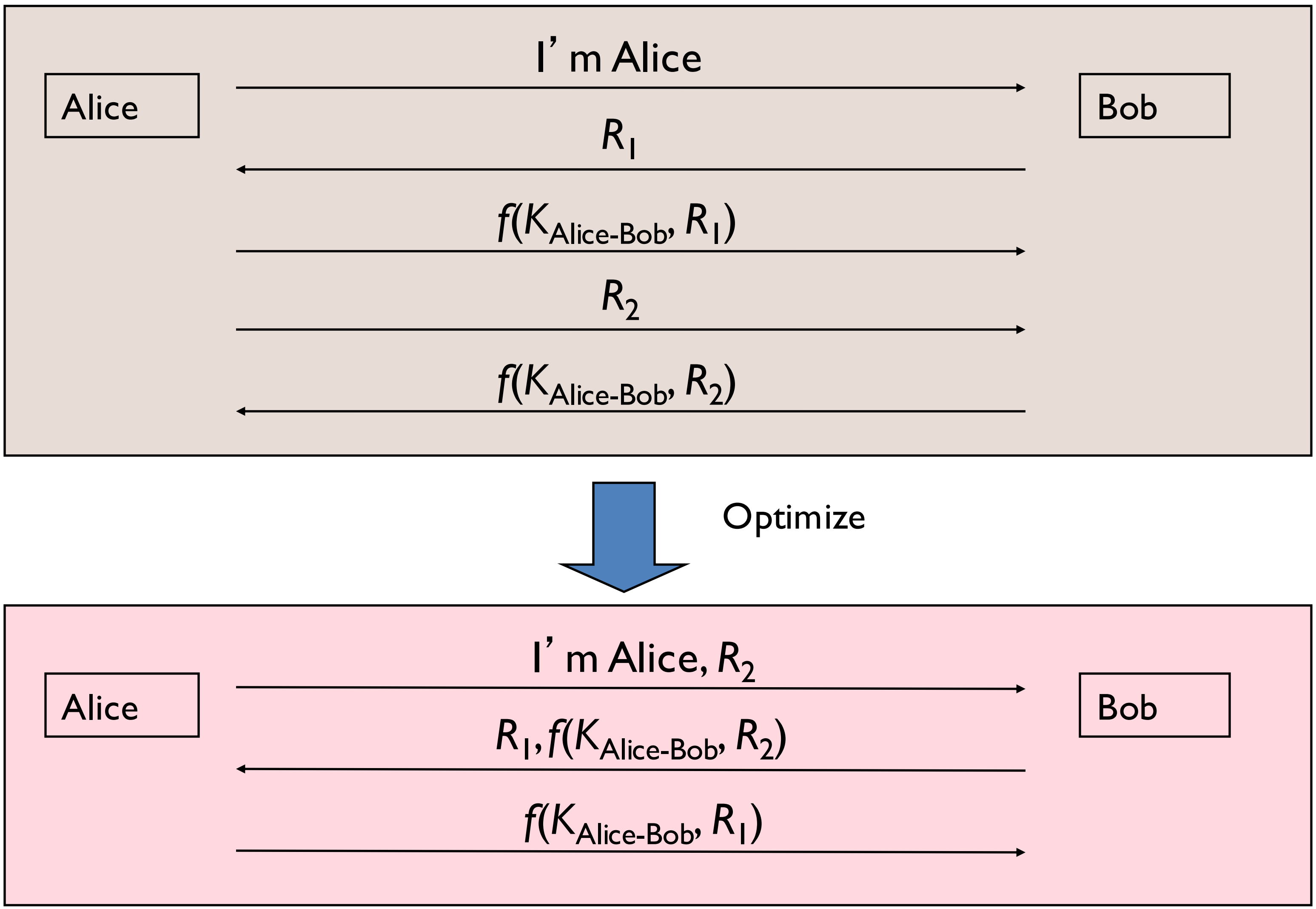
Authentication with Public Key (Cont'd)



A variation

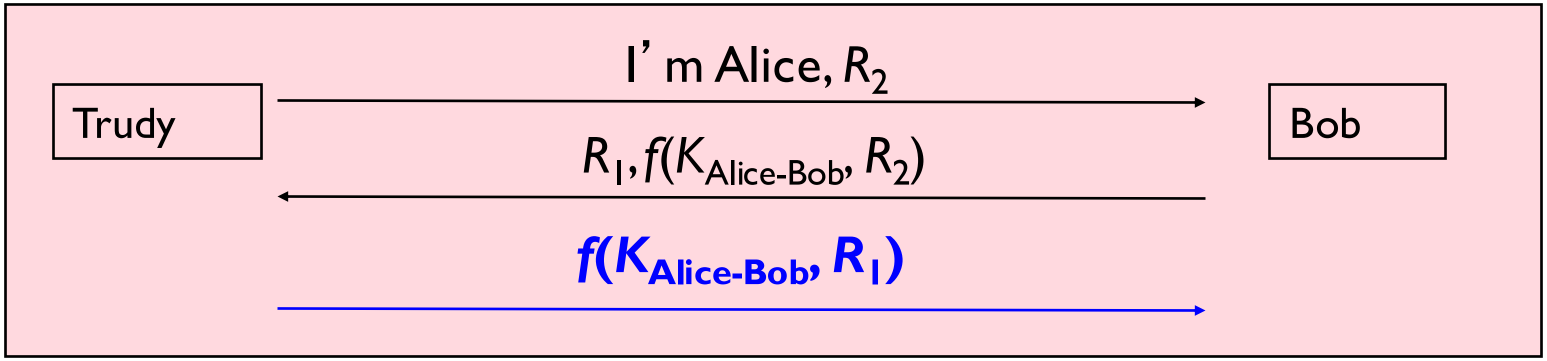
What happens if Trudy could get Alice to decrypt things arbitrarily?

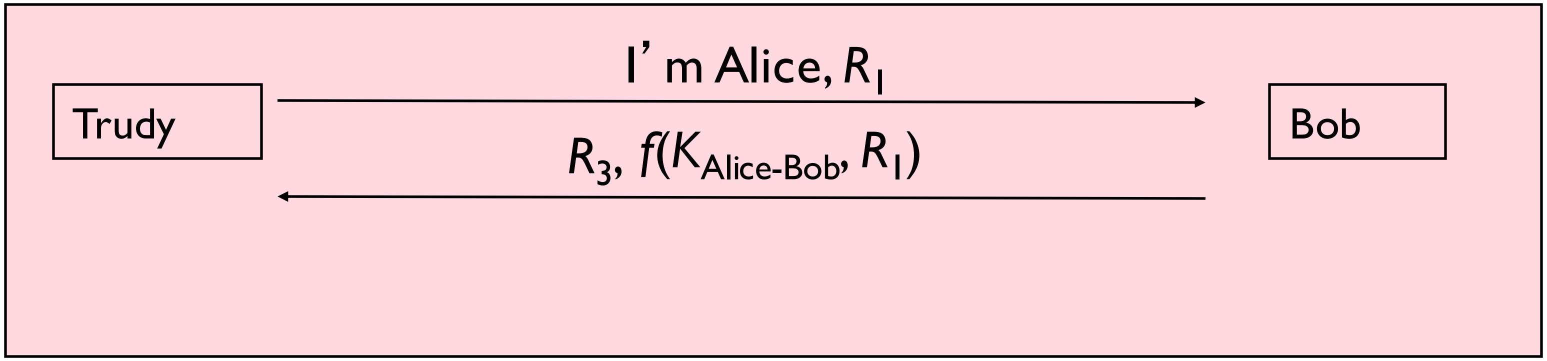
Mutual Authentication



Mutual Authentication (Cont'd)

- Reflection attack



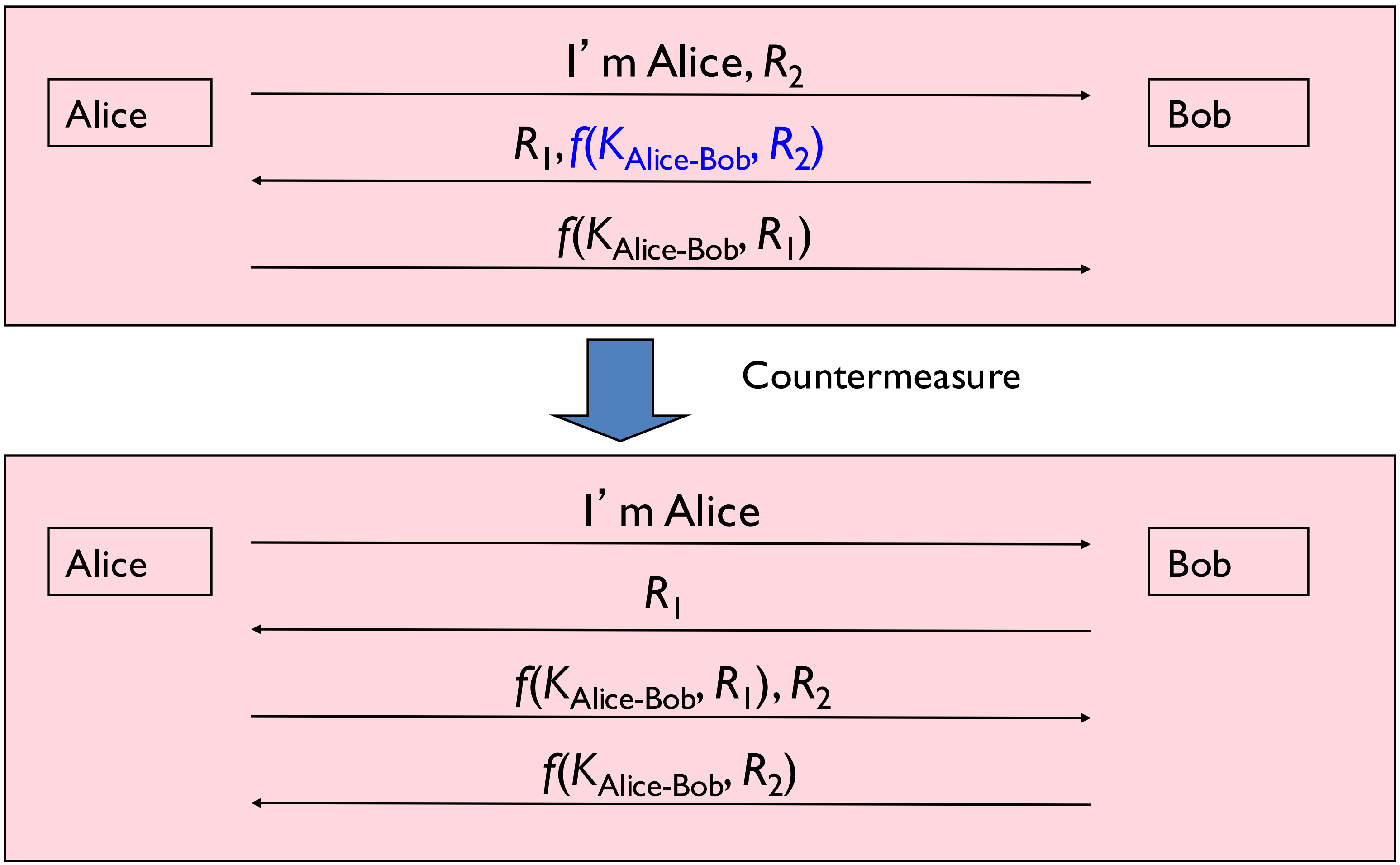


Reflection Attacks (Cont'd)

- 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

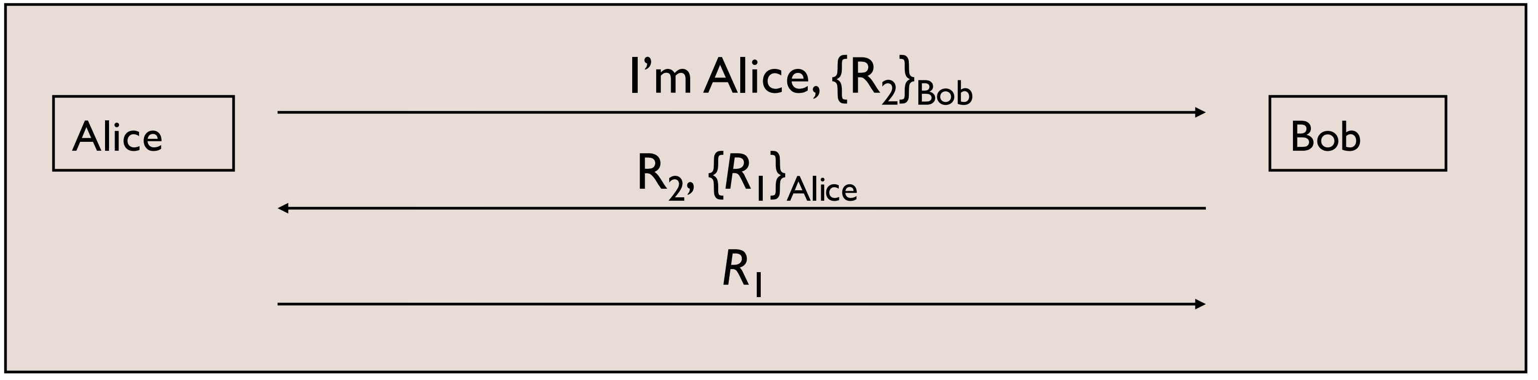
Mutual Authentication (Cont'd)

- Password guessing



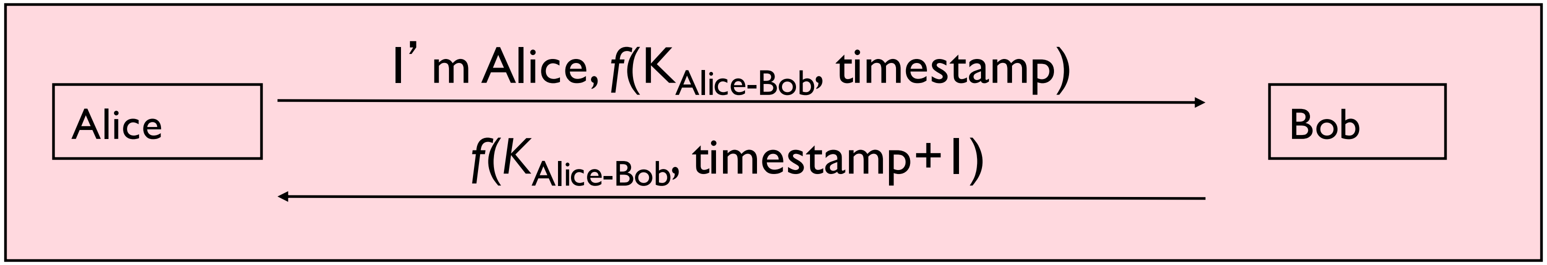
Mutual Authentication (Cont'd)

- Public keys
  - Authentication of public keys is a critical issue



Mutual Authentication (Cont’d)

- Mutual authentication with timestamps
  - Require synchronized clocks
  - Alice and Bob have to encrypt different timestamps

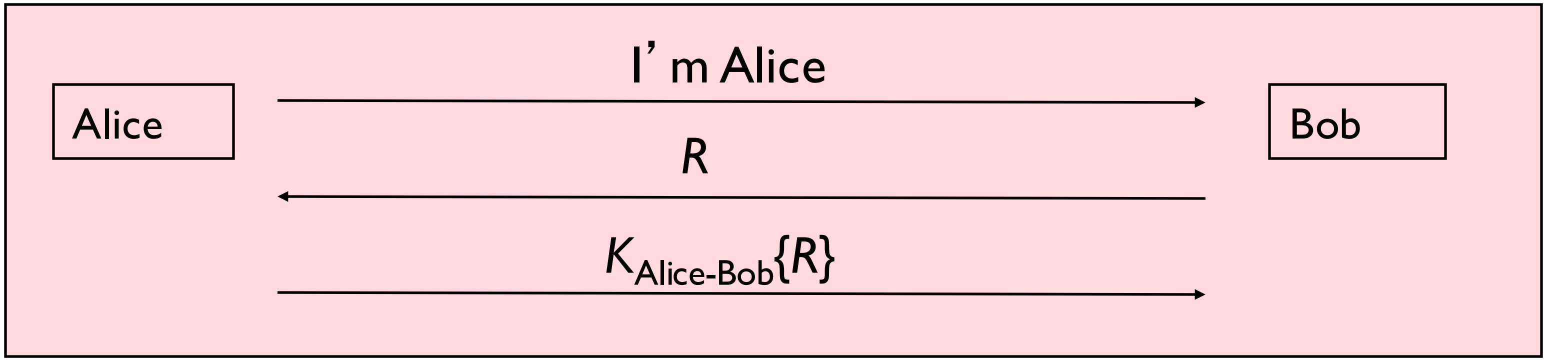


Integrity/Encryption for Data

- Communication after mutual authentication should be cryptographically protected as well
  - Require a *session key* established during mutual authentication

Establishment of Session Keys

- Secret key based authentication
  - Assume the following authentication happened.
  - Can we use $K_{Alice\text{-}Bob}\{R\}$ as the session key?
  - Can we use $K_{Alice\text{-}Bob}\{R+1\}$ as the session key?
  - In general, modify $K_{Alice\text{-}Bob}$ and encrypt R. Use the result as the session key.



Establishment of Session Keys (Cont’d)

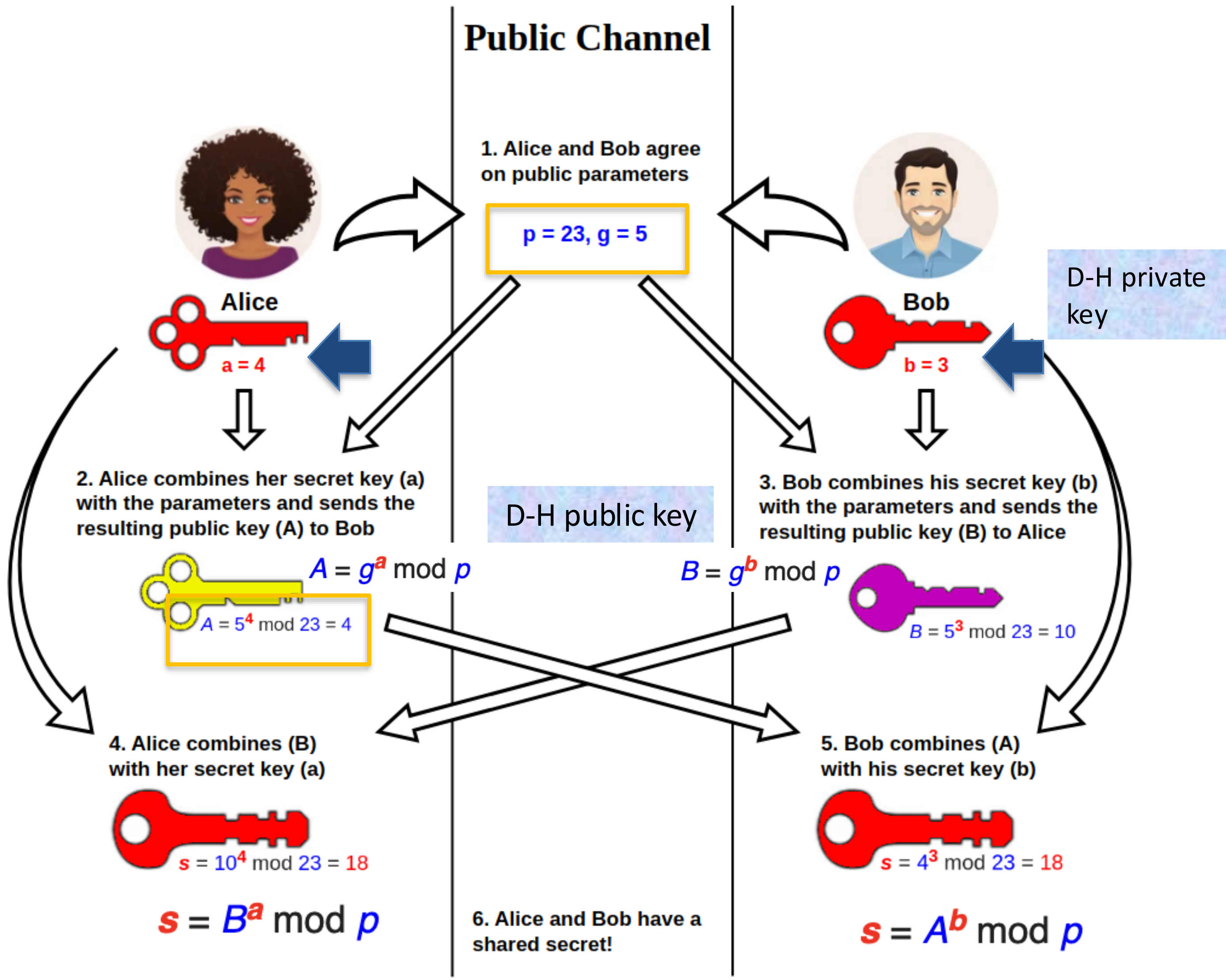
- Two-way *public key* based authentication
  1. Alice chooses a random number R, encrypts it with Bob’s public key, result used as session key.
     - Trudy may hijack the conversation
  2. Alice encrypts and signs R
     - Trudy may save all the traffic, and decrypt all the encrypted traffic when she is able to compromise Bob
     - *Less severe threat*

Two-Way Public Key Based Authentication (Cont'd)

- A better approach
  - Alice chooses and encrypts R_1 with Bob's public key
  - Bob chooses and encrypts R_2 with Alice's public key
  - Session key is $R_1 \oplus R_2$
  - Trudy will have to compromise both Alice and Bob
- An even better approach
  - Alice and Bob establish the session key with *Diffie-Hellman key exchange*
  - Alice and Bob sign the quantity they send
  - Trudy can't learn anything about the session key even if she compromises both Alice and Bob

Diffie-Hellman Key Exchange

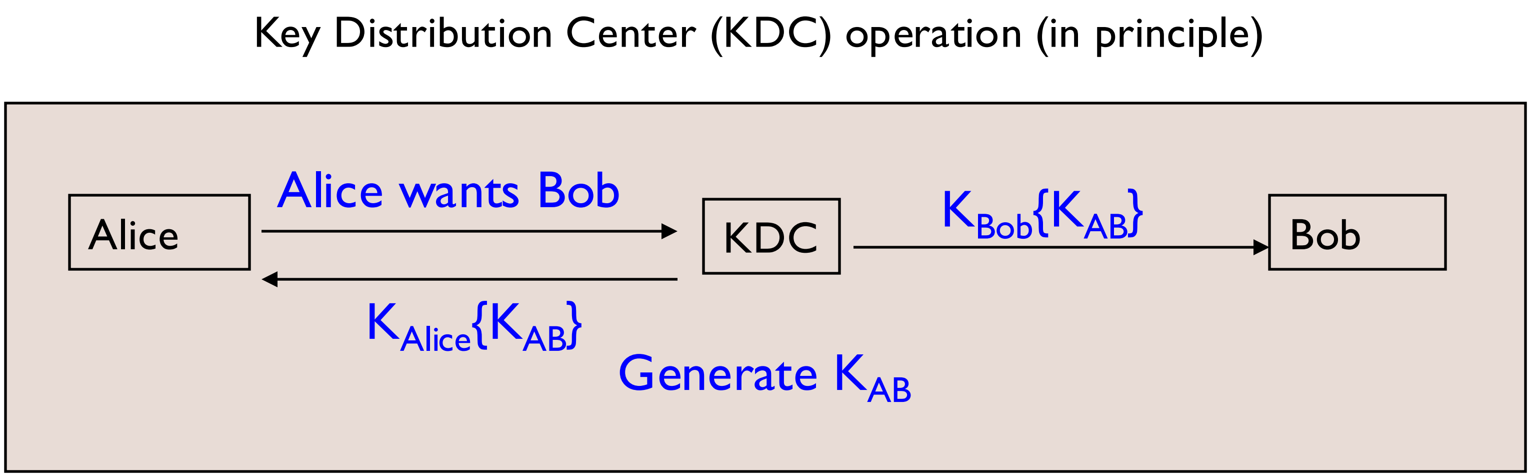
- Used to establish session keys
- Preferred over RSA as it provides *forward secrecy*.



Establishment of Session Keys (Cont’d)

- One-way public key based authentication
  - It’s only necessary to authenticate the server
    - Example: SSL
  - Encrypt R with Bob’s public key
  - Diffie-Hellman key exchange
    - Bob signs the D-H public key

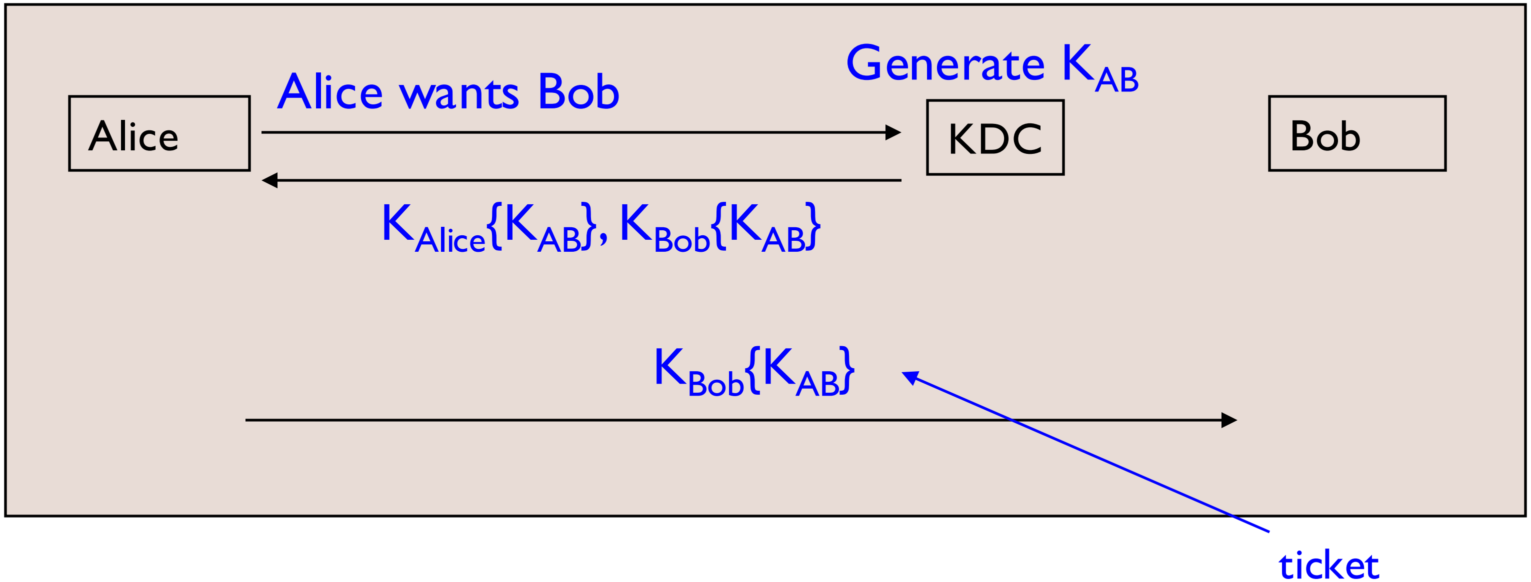
Mediated Authentication (With KDC)



- 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)

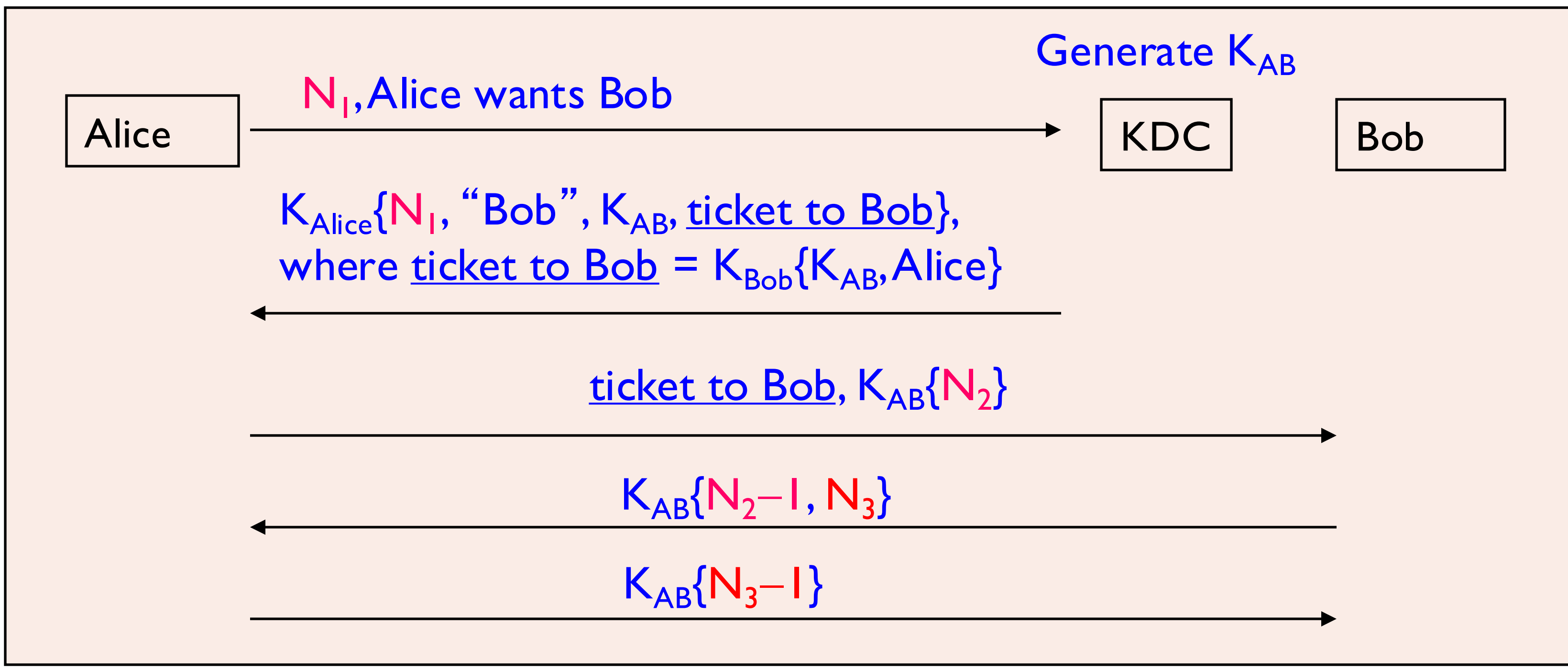
KDC operation (in practice)



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

Needham-Schroeder Protocol

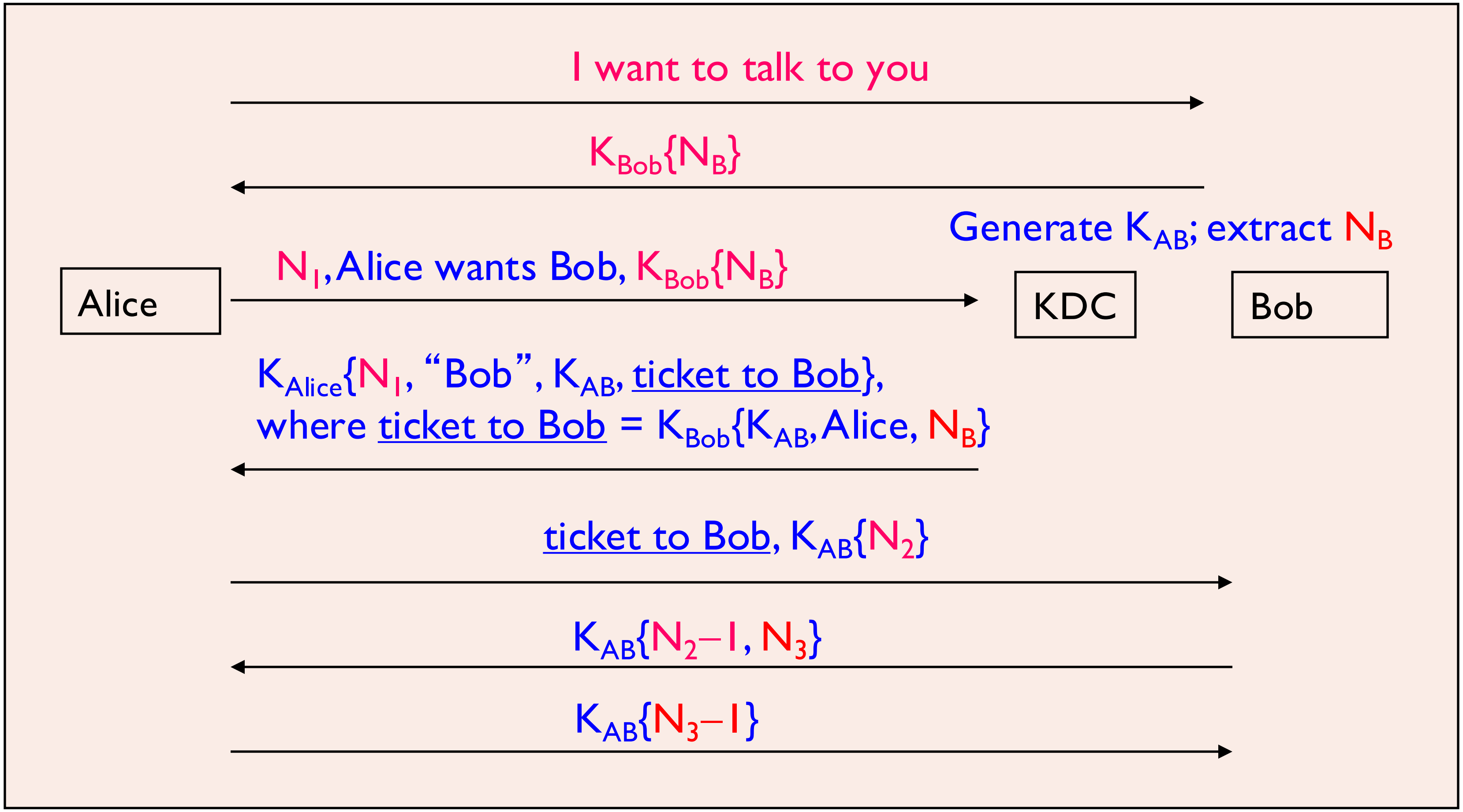
- Classic protocol for authentication with KDC
  - Many others have been modeled after it (e.g., Kerberos)
- Nonce: A number that is used only once
  - Deal with replay attacks



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

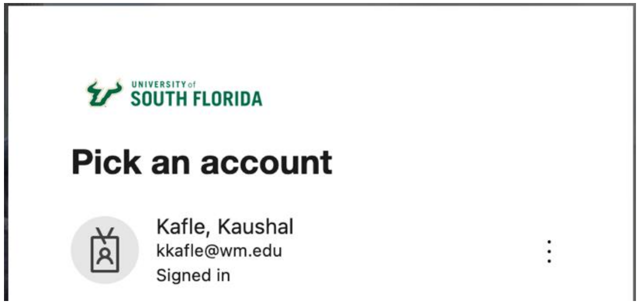


- The additional two messages assure Bob that the initiator has talked to KDC since Bob generates N_B

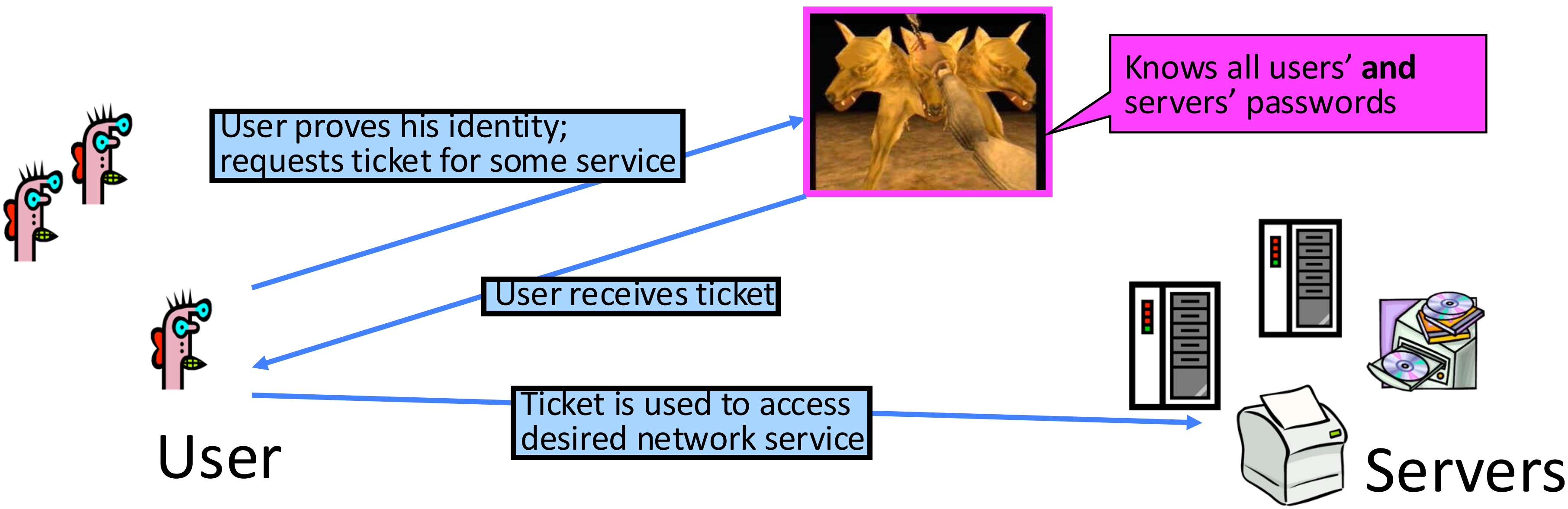
Kerberos

Kerberos

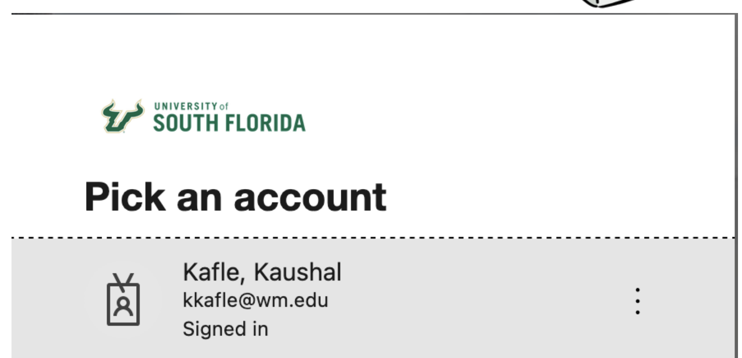
- An online system that resists password eavesdropping and achieves **mutual authentication**
- First single sign-on system (SSO)
- Easy application integration API
- Most widely used (non-web) centralized password system in existence
- Now part of Windows network authentication



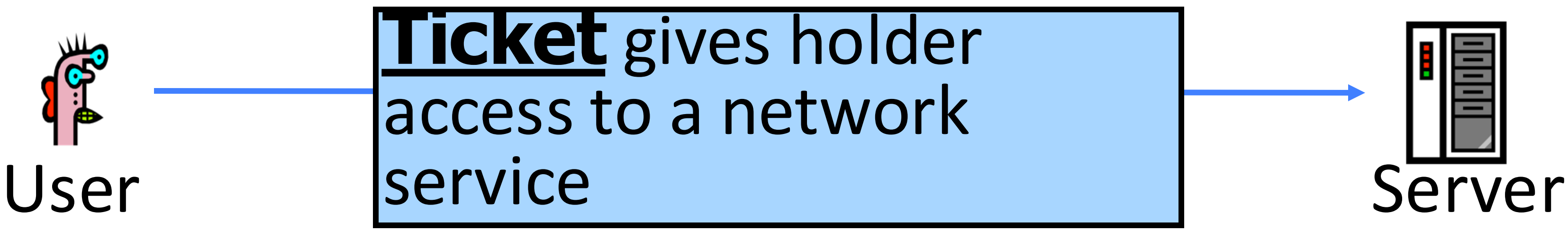
Kerberos Overview



E.g. SSOs

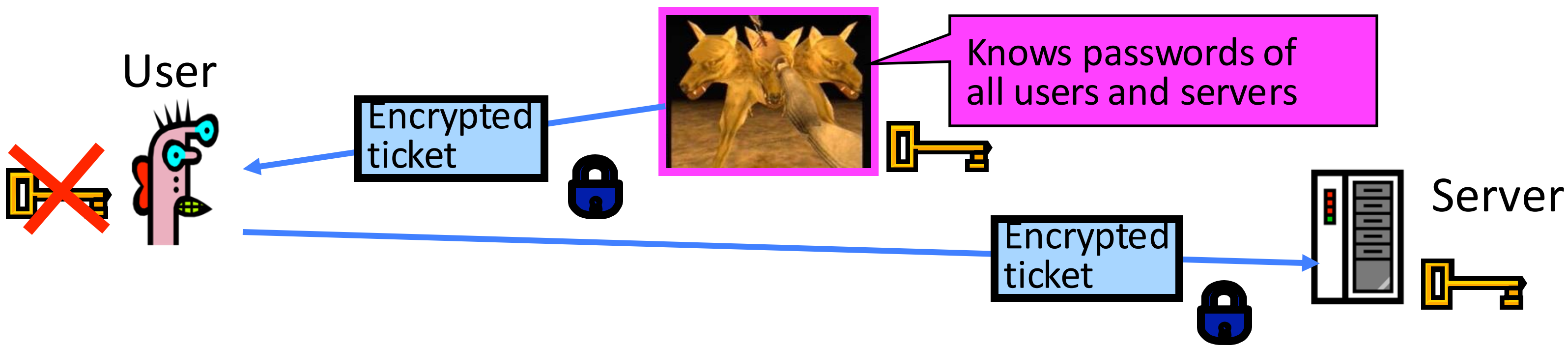


What Should a Ticket Look Like?



- 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?



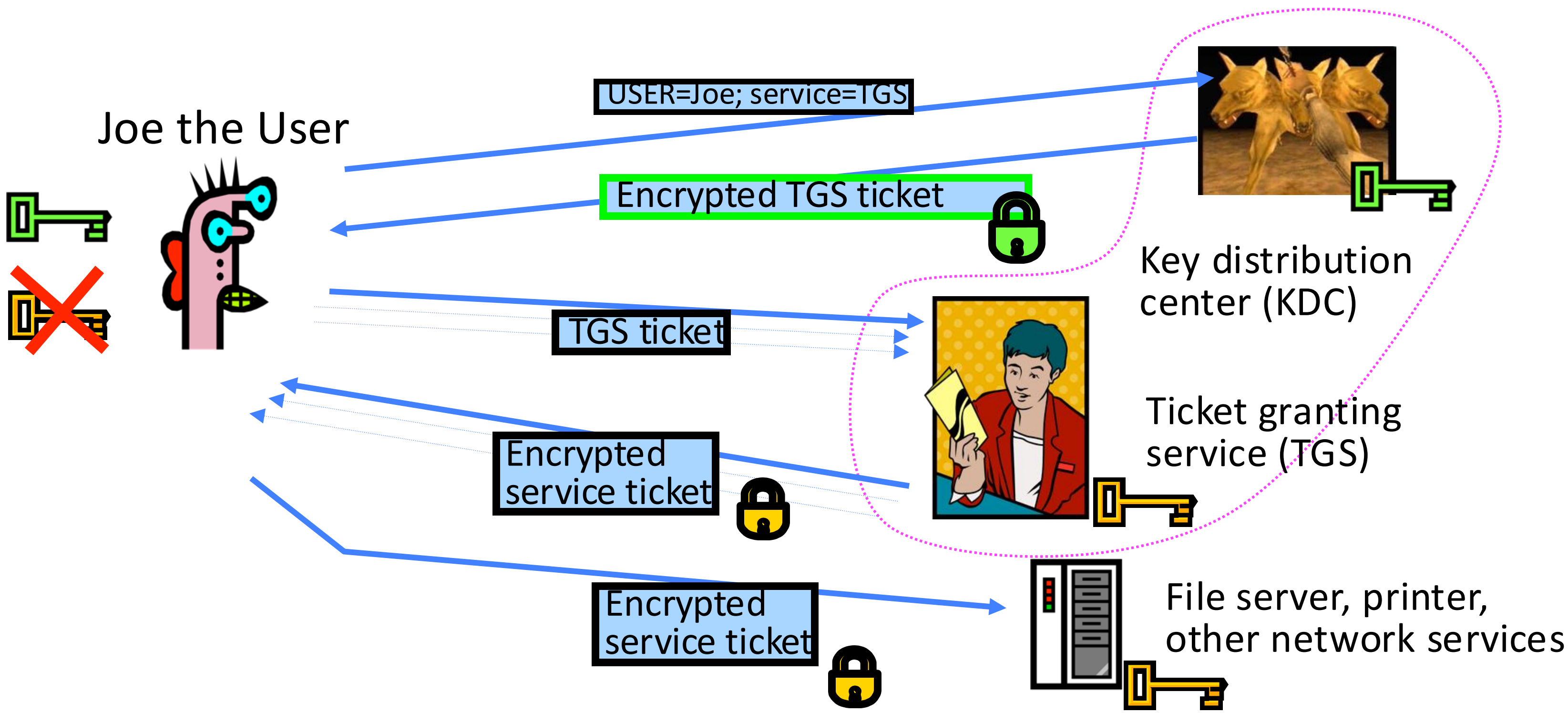
- User name
- Server name
- Address of user's workstation -- **WHY?**
- Ticket lifetime -- **WHY?**
- A few other things (e.g., session key)

So that ticket expires, prevents reuse

No ticket reuse by other user.

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

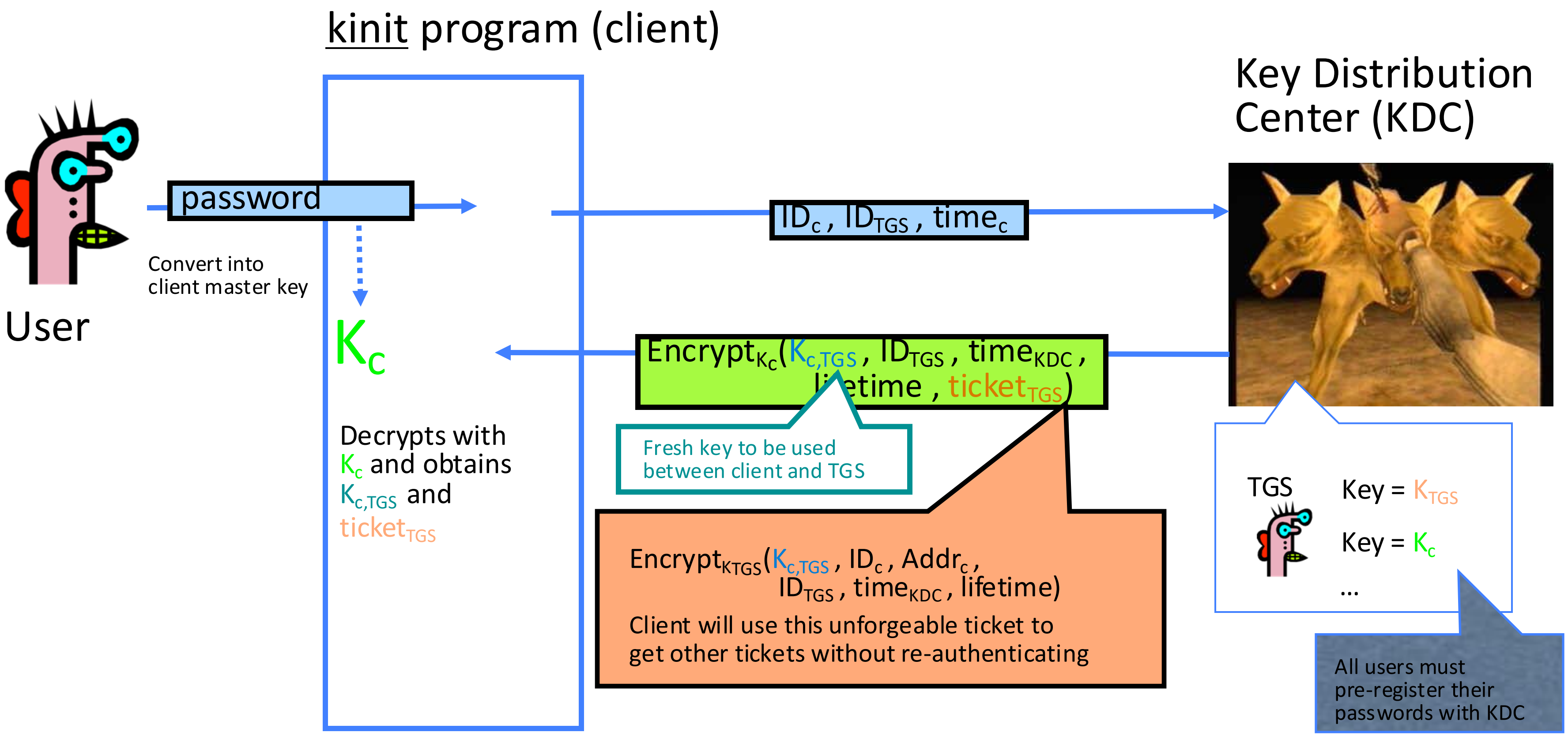
Password-based key derivation function 2 (PBKDF 2)

- 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!

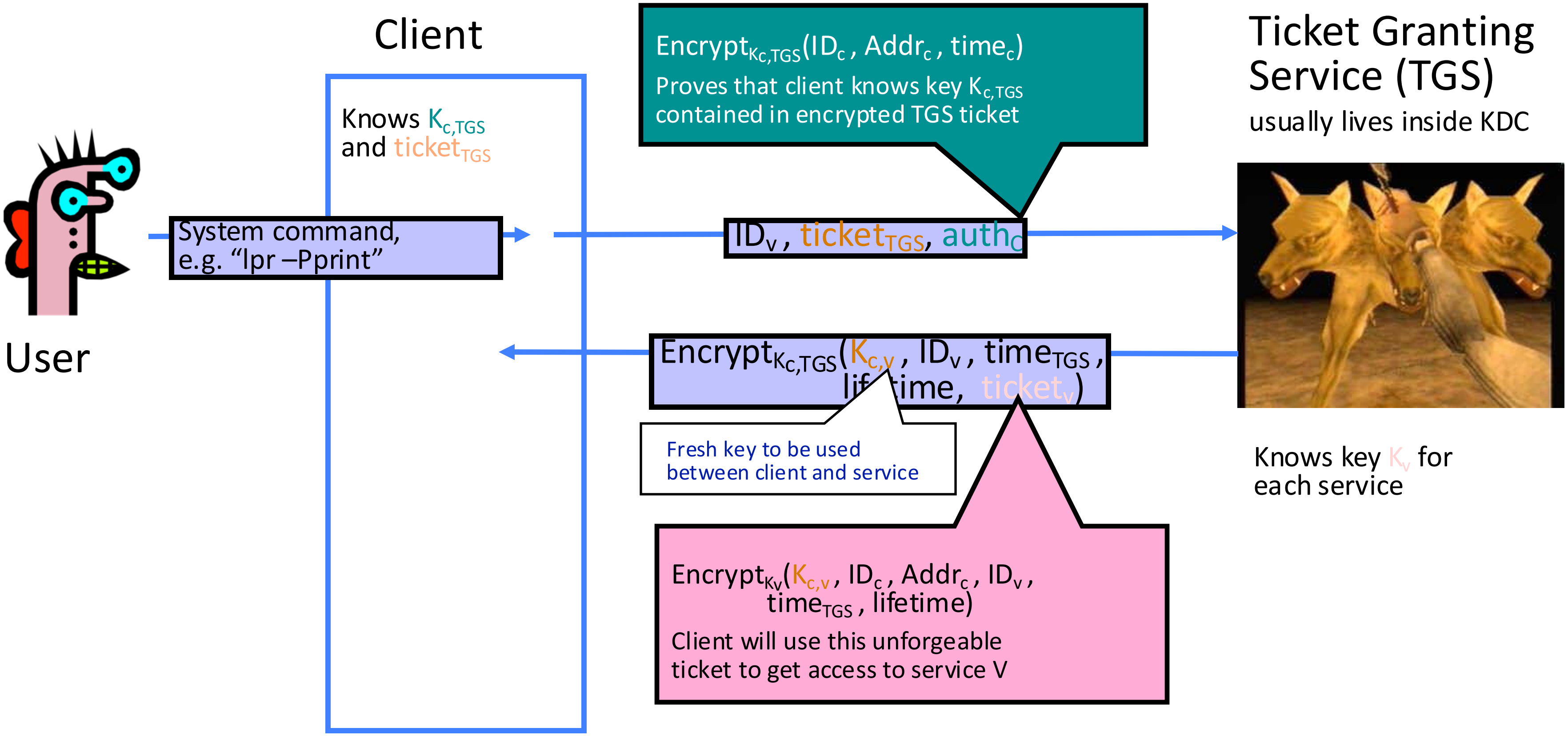
- Three-step process:
  - “Logon” -- obtain TGS ticket from KDC
  - Obtain “service ticket” from TGS
  - Use service

“Single Logon” Authentication



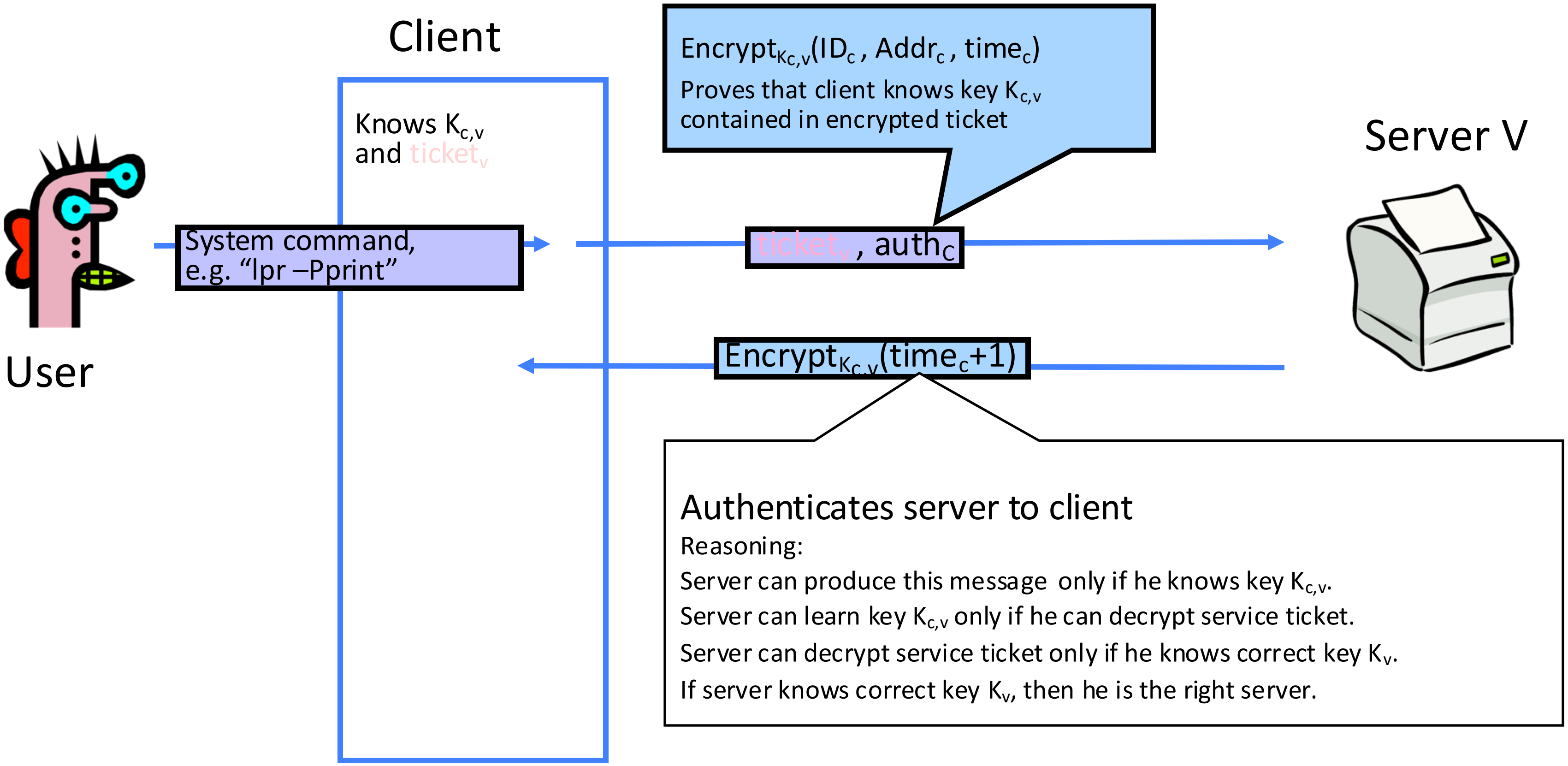
- Client only needs to obtain TGS ticket once (say, every morning)
  - Ticket is encrypted; client cannot forge it or tamper with it

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
- Meant for users/services in one Kerberos realm to access resources in another Kerberos realm
  - 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)