

# Authentication

# Outline

- Concepts and basic authentication
- Strong authentication
- Authentication and Key Exchange

# Authentication

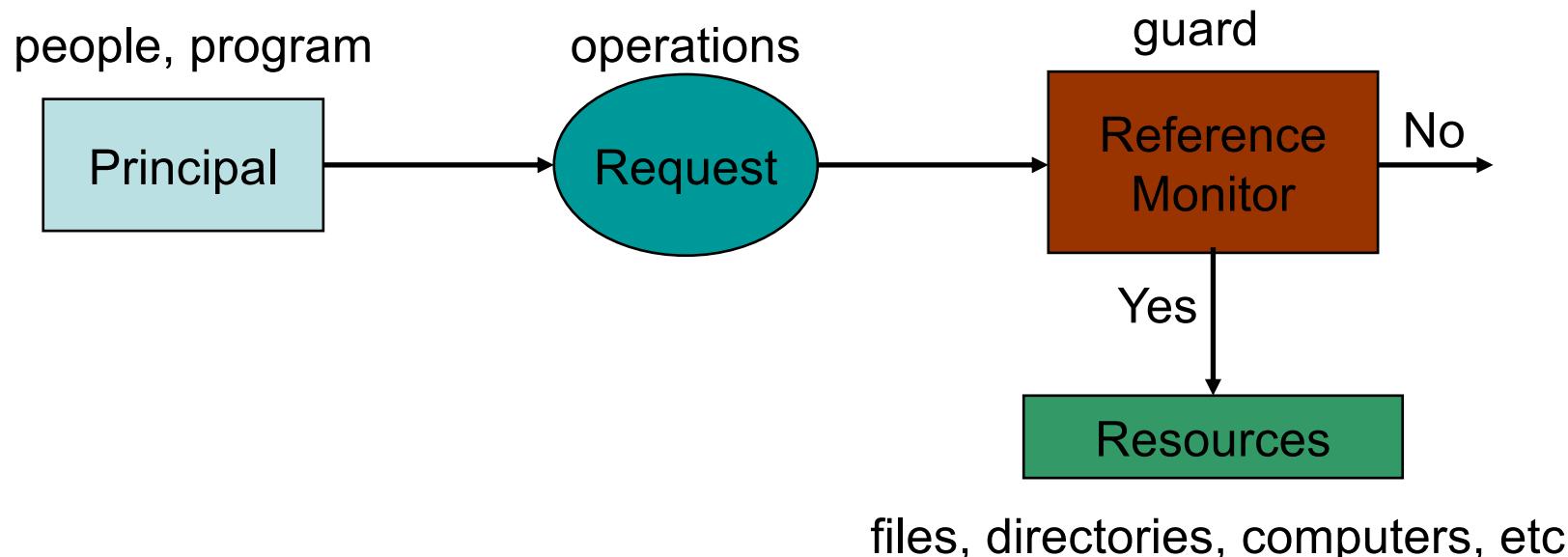
Concepts and basic authentication

# Authentication (from Wikipedia)

**Authentication** (Greek: *αυθεντικός*, from 'authentes'='author') is the process by which a computer, computer program, or another user attempts to confirm that the computer, computer program, or user from whom the second party has received some communication is, or is not, the claimed first party.

# Authentication, authorization and access control

- Authentication: determining who makes the request
- Authorization: determining who is allowed to access resources
- Access control: how to make the decision and control the access



# Authentication model

From the textbook (computer security: principles and practice)

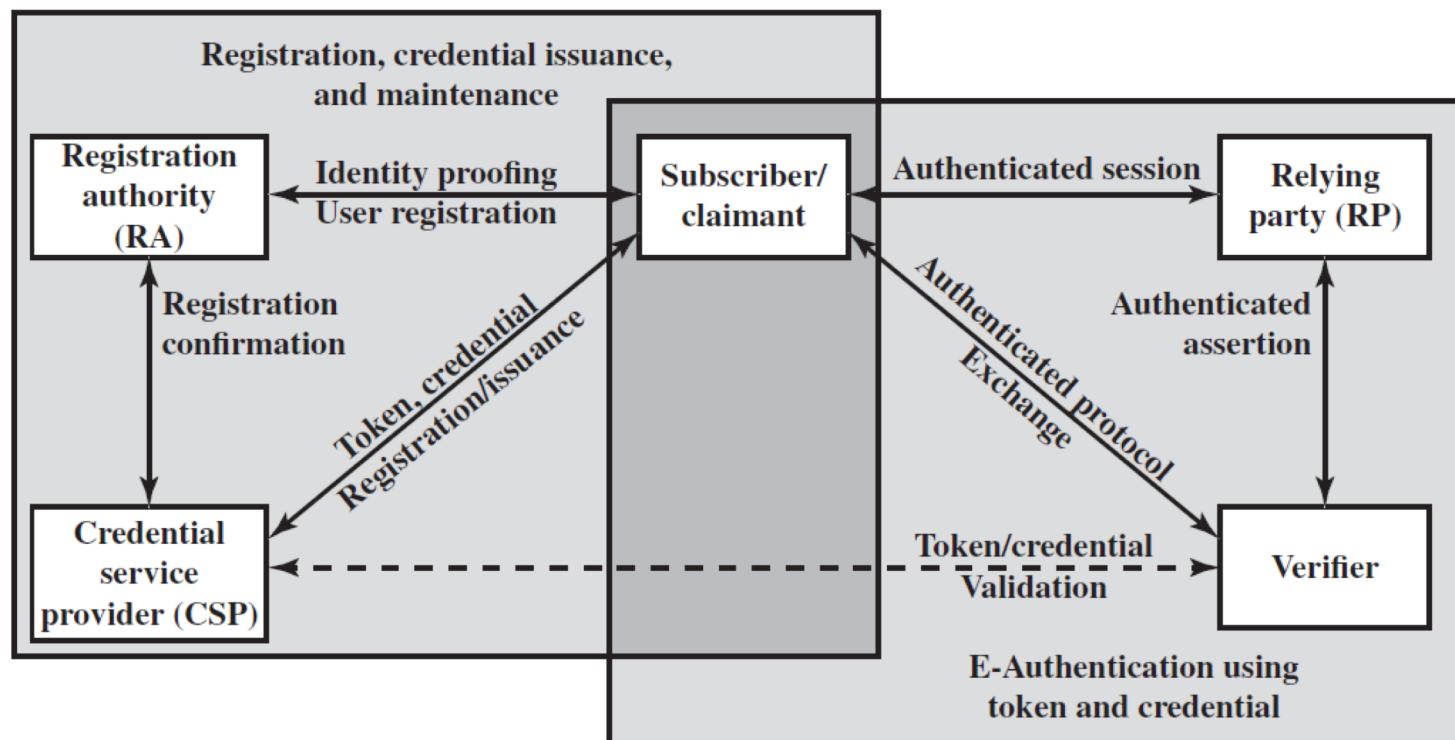


Figure 3.1 The NIST SP 800-63-2 E-Authentication Architectural Model

# Means of user authentication

- User authentication is based on knowledge shared by a computer and a user
- Three types of knowledge
  - Something the user knows: password, PIN, etc.
  - Something the user has: credit card, debit card, etc.
  - Something the user is: fingerprint, DNA, etc
  - Something the user does: voice pattern, handwriting, typing rhythm, etc.

# Password authentication

- Password: a mutually agreed-upon secret code between a computing system and a user
- Authenticate a user on the basis of  $\langle \text{user}, \text{password} \rangle$  pair

# Attacks on passwords

- Guess: online or offline (obtaining password file)
  - Try common passwords
  - Personalized guesses
  - Password reuse
- Steal: examples
  - Workstation hijacking
  - User mistakes
  - Electronic monitoring

# Weak passwords

- Morris and Thompson studied the password distribution in 79
  - 2 letters: 2%; 3 letters: 14%; 4 letters 14%; 5 letters: 22%; words in dictionaries: 15%
- British online bank Egg found users still choosing weak passwords in 02
  - Family member names: 50%

# How to crack a weak password

- No password
- The same as the user ID
- Derivable from user ID
- Common word lists (“password”...) plus patterns (“asdfg”...)
- Look up words in online dictionary
  - One contains 80,000 words, and trying all takes only 80s
- Add capitalization (“PaSsWorD”...)
- Add substitution (0 for o ...)

# More password cracking

- Use more power hardware
  - AMD can try XX passwords every second
- Use better algorithms
  - Model relations between letters with hidden Markov chain
- Leverage exposed passwords
  - SQL injection on RockYou.com exposes 32 million passwords

# Reasons for using password authentication

- Easy to deploy
  - Alternatives are more expensive and more complicated to deploy
- Robust
  - Approaches like single-sign-on create a single point of failure.
- Flexible
  - Automatic password managers are hard to synchronize across devices

# Choose a strong password

- Use characters other than A-Z
- Choose long passwords
- Avoid actual names or words
- Choose an unlikely password
- Change the password regularly
- Do not write it down
- Do not tell anyone else

# Password selection strategies

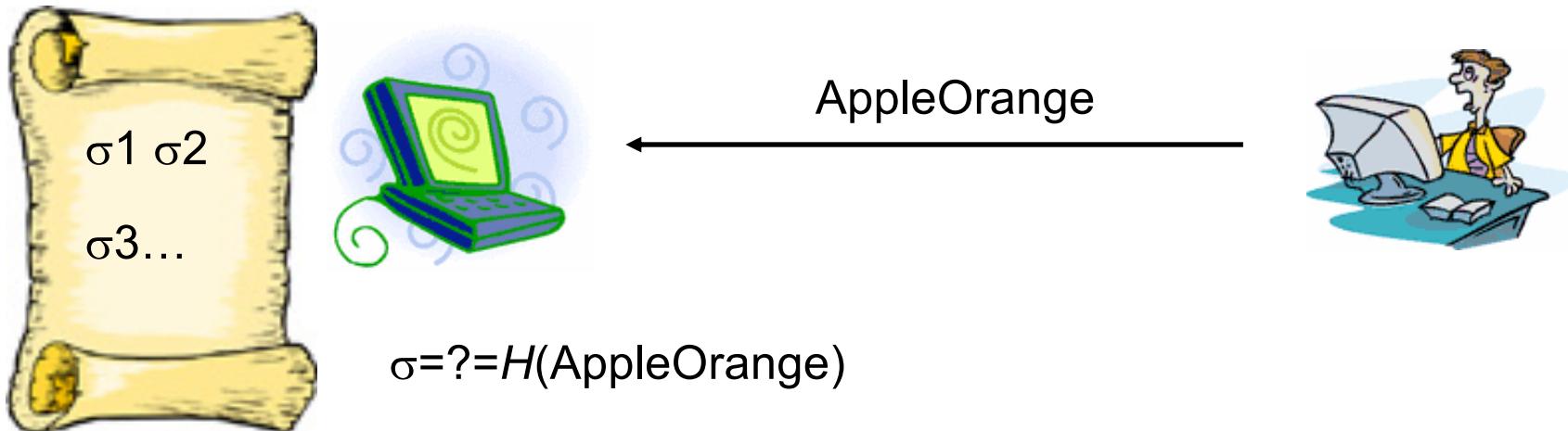
- User education
- Computer-generated passwords
- Password checking
- Password policy

# How to store the password list?

- Obvious solution: keeping it on the server in plaintext
- But, if cracker gets into the server, somehow...

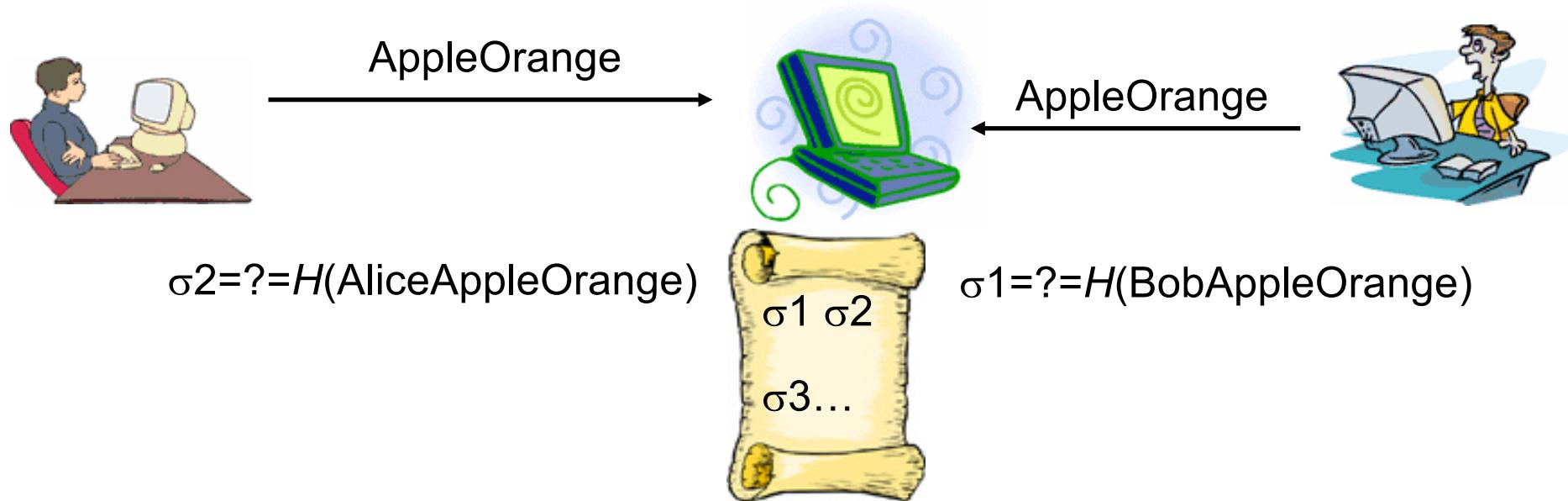
# Store password list

- Keep only the fingerprints of the passwords



# Store password list (cont'd)

- What happens if two choose the same password?



- In Unix,  $\sigma = H(\gamma, \text{passwd})$  and the computer keeps  $(\gamma, \sigma)$

# UNIX password storage and protection

- Stored in shadow password file
  - Protected by system privilege
- Threats to the password file
  - Hacking
  - Breaking in less protected systems of the same user
  - Backup devices carry password file
  - Network sniffing

# Authentication

Strong user authentication

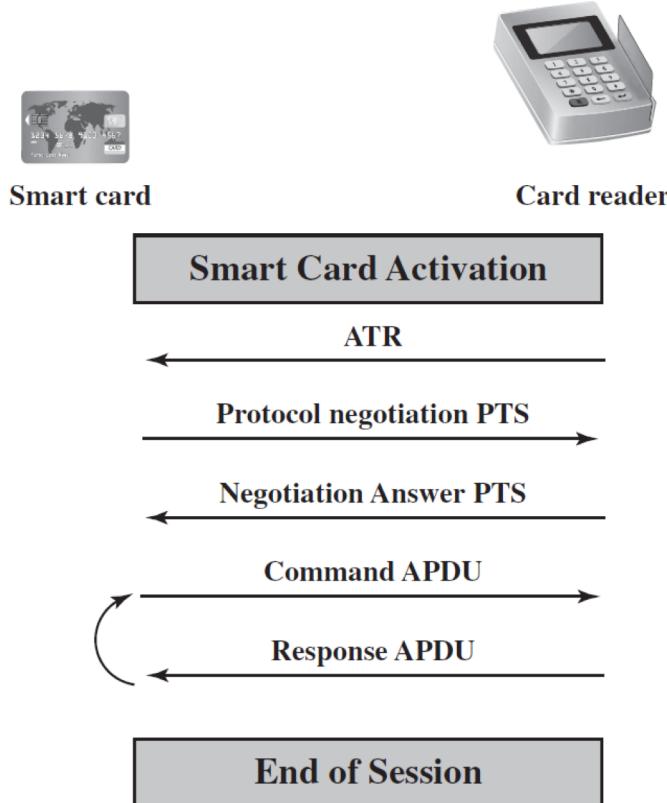
# Problems of password authentication

- An adversary may tap the network wire to steal your password
- Your passwords might get lost or stolen

# Memory card and smart card

- Memory card
  - Credit card, bank card with magnetic stripe
  - Some further protected by PIN
  - Weaknesses: special reader, risk of token loss, usability
- Smart card
  - Include a microprocessor
  - User interface: keyboard and display
  - Electronic interface: contact or NFC
- Authentication protocol
  - User => card => computer
  - One-time password
  - Challenge-response

# Smart card/reader exchange



APDU = Application protocol data unit  
ATR = Answer to reset  
PTS = Protocol type selection

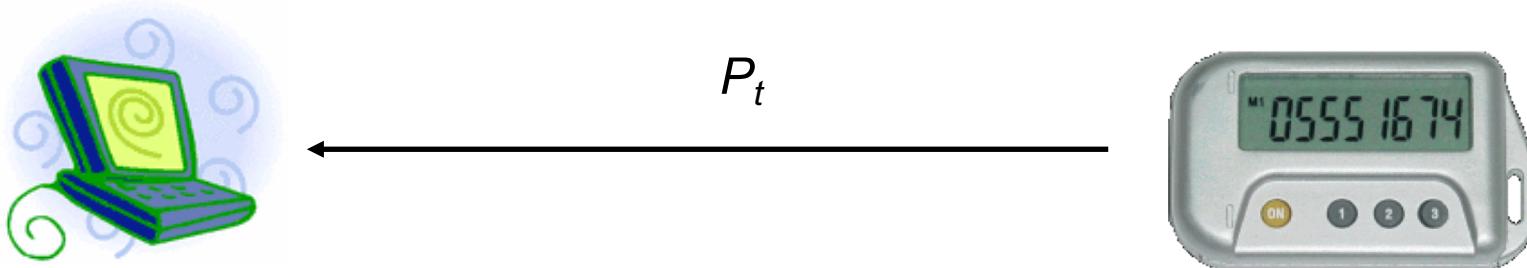
Figure 3.5 Smart Card/Reader Exchange

# One-time password

- Password good for one-use only
- Password token
  - A device that generates unpredictable password
  - Synchronous token
    - Change password every minute
    - Time alignment between the token and the computer
    - Need to adjust alignment periodically



# How to design a one-time password token

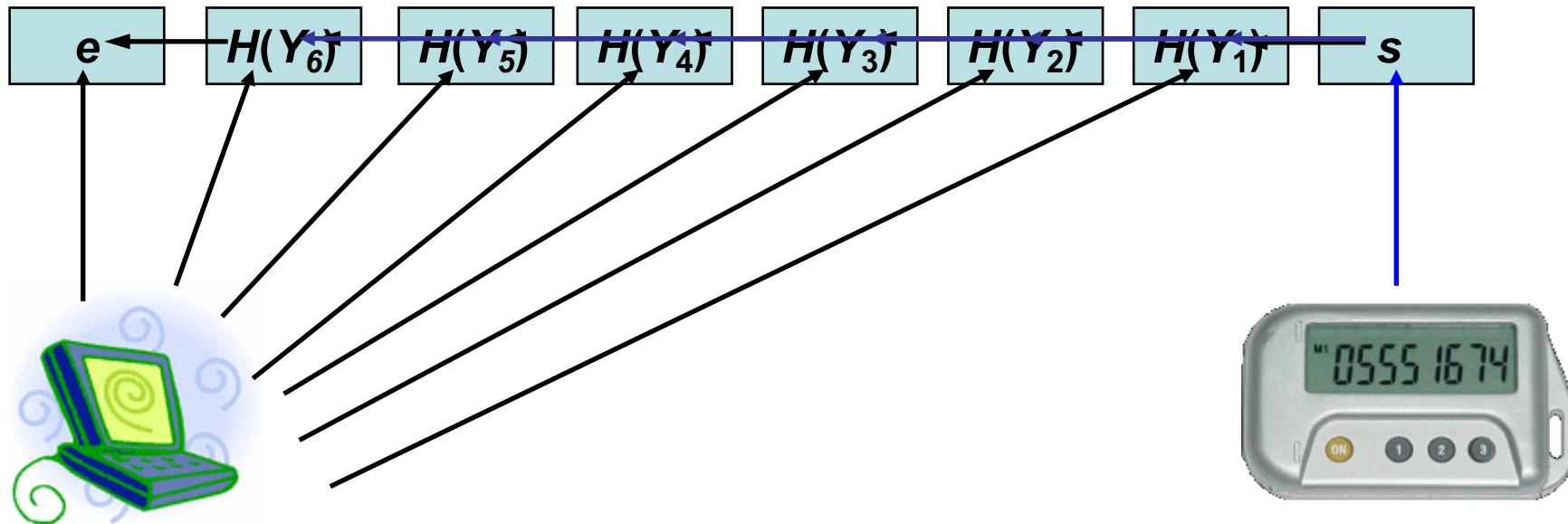


$$P_t = ?= H(K, \text{time } t)$$

$$P_t = H(K, \text{time } t)$$

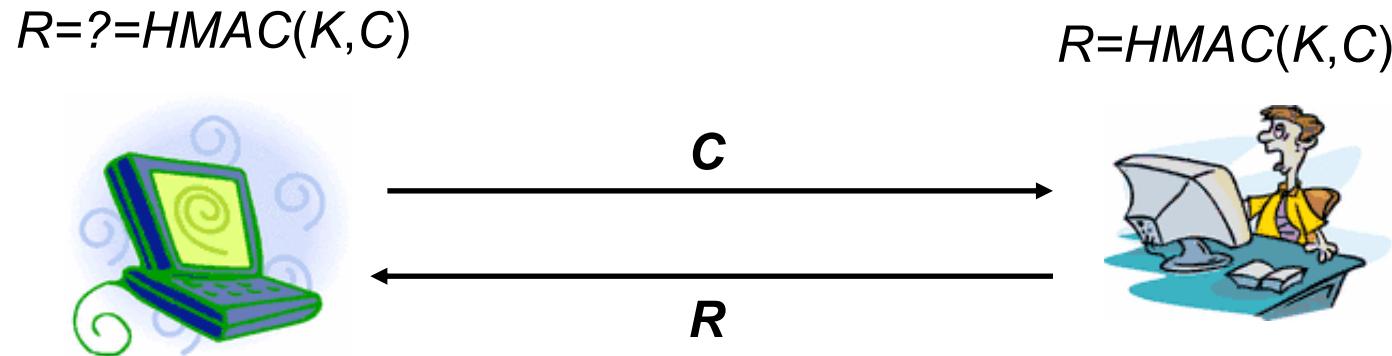
- Token only calculates hash once
- But, it needs to keep synchronization with the computer

# How to design a password token (cont'd)



- Hash-chain approach
- This approach works against eavesdropper, why?
- This approach does not need synchronization between token and computer
- But token has to perform many hash operations

# Challenge and response



- Use one-way function and a shared secret, a client can authenticate a server even in the presence of eavesdropper

# Tell human and computers apart

- Sometimes, one only needs to prove that he is a human
  - Many security problems caused by “zombies”, computers controlled by virus, worms
- How to tell human and computer apart
  - Human being can understand some fuzzy concepts while computers cannot
  - Can we design a trapdoor one-way function with human cognition capability as key?

# CAPTCHA

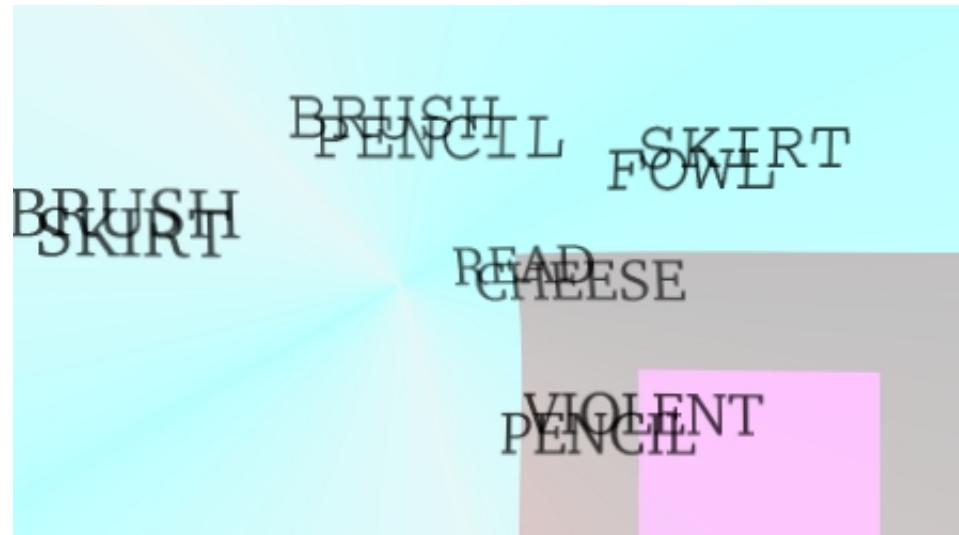
- Turing test: a test of a machine's capability to perform human-like conversation.
- CAPTCHA: **completely automated public Turing test to tell computers and humans apart**
  - CAPTCHA is a one-way function to computer
  - CAPTCHA is not a one-way function to human being
  - <http://www.captcha.net/captchas/>

# CAPTCHA used by gmail

A distorted red word "sking" is centered on the slide. The letters are slightly blurred and have a red glow, typical of a CAPTCHA image.

- This approach is no longer secure: some pattern-recognition problem can automatically detect the characters

# A more secure example



# Another example



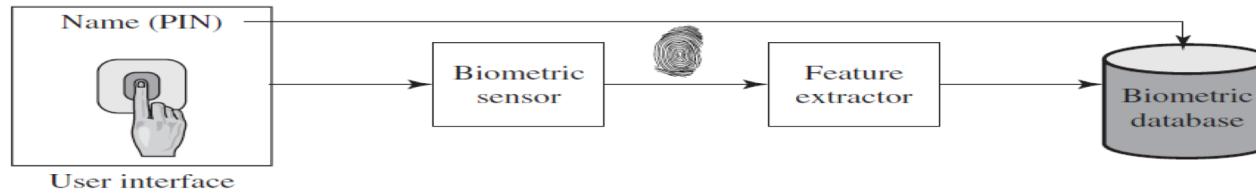
Choose a word that relates to all the images.



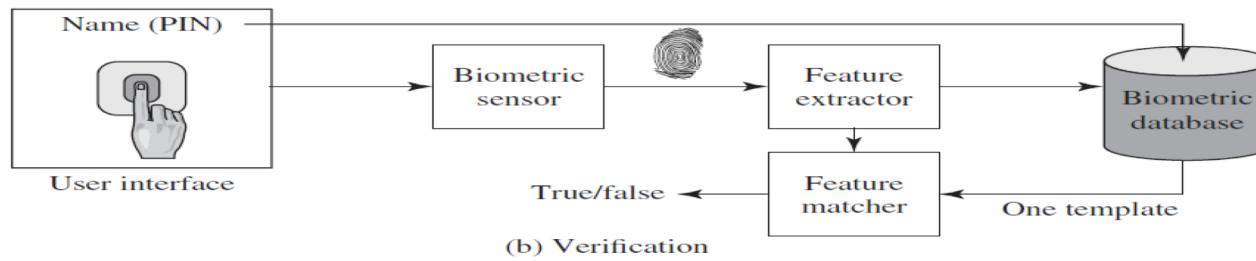
# Biometric authentication

- Handwritten signatures
- Keystroke dynamics
- Face recognition
- Hand geometry
- DNA
- Fingerprints
- Iris code
- Retina pattern
- Voice

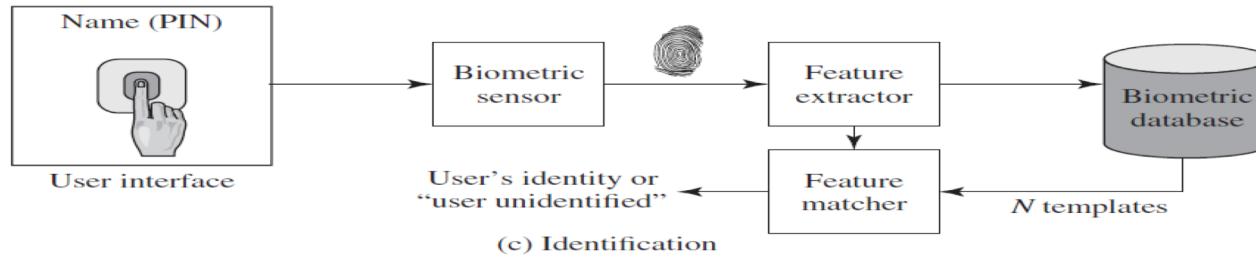
# Biometric system: operations



(a) Enrollment

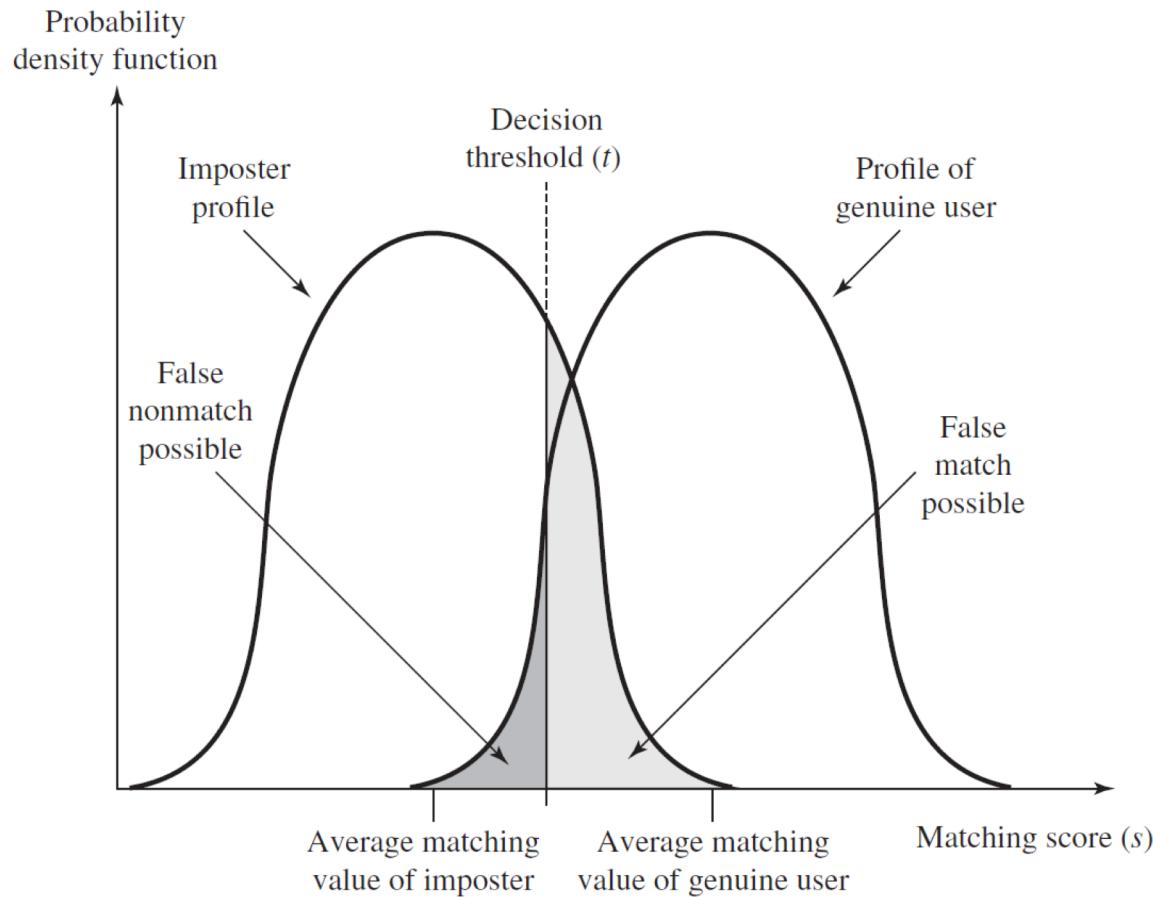


(b) Verification



(c) Identification

# Biometric accuracy



# Problems of biometrics

- Limitation of technologies
  - DNA typing has a high rate of false positives
- Environmental factors
  - Noise, dirt, vibration, and unreliable lighting conditions
- Forgery
  - Fingerprints planted by villains
  - The age of a fingerprints
- Challenge: more usable biometrics technologies
  - E.g, meet bank's goal of 1% fraud rate and 0.01% insult rate

# Mutual authentication

- Man-in-the-middle attack
- Server also needs to authenticate itself to user
  - Simple authentication: last login time
  - Mutual Challenge-and-Response authentication

# Other Authentication

- Two factor authentication
- Single Sign On

# Authentication

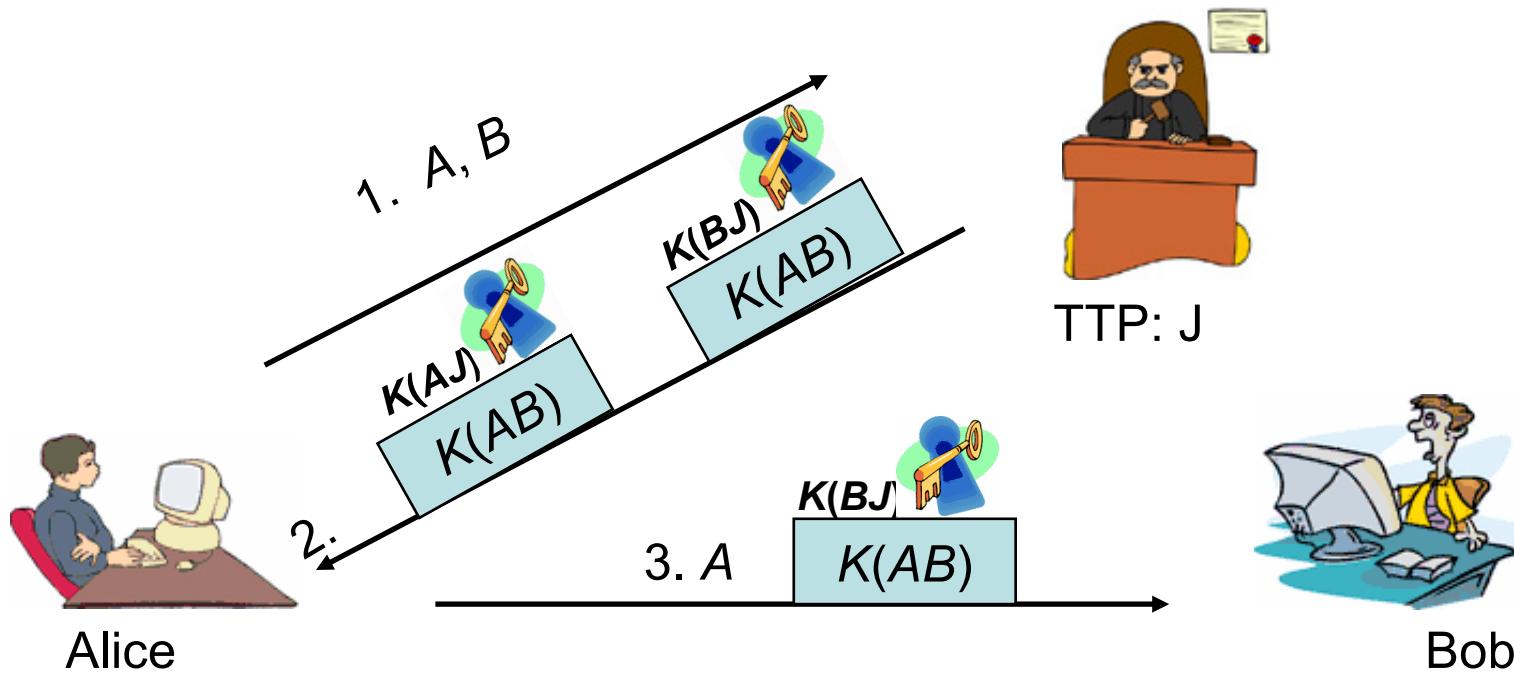
Authentication and key exchange protocols

# Key exchange

- Authentication between two agents
  - One agent gains confidence in the other's identity
  - E.g., computer to computer, software to software
- Establish session key
  - Authentication is coupled with the distribution of a session key
  - Session key is used to protect authenticity, confidentiality and integrity of later communication

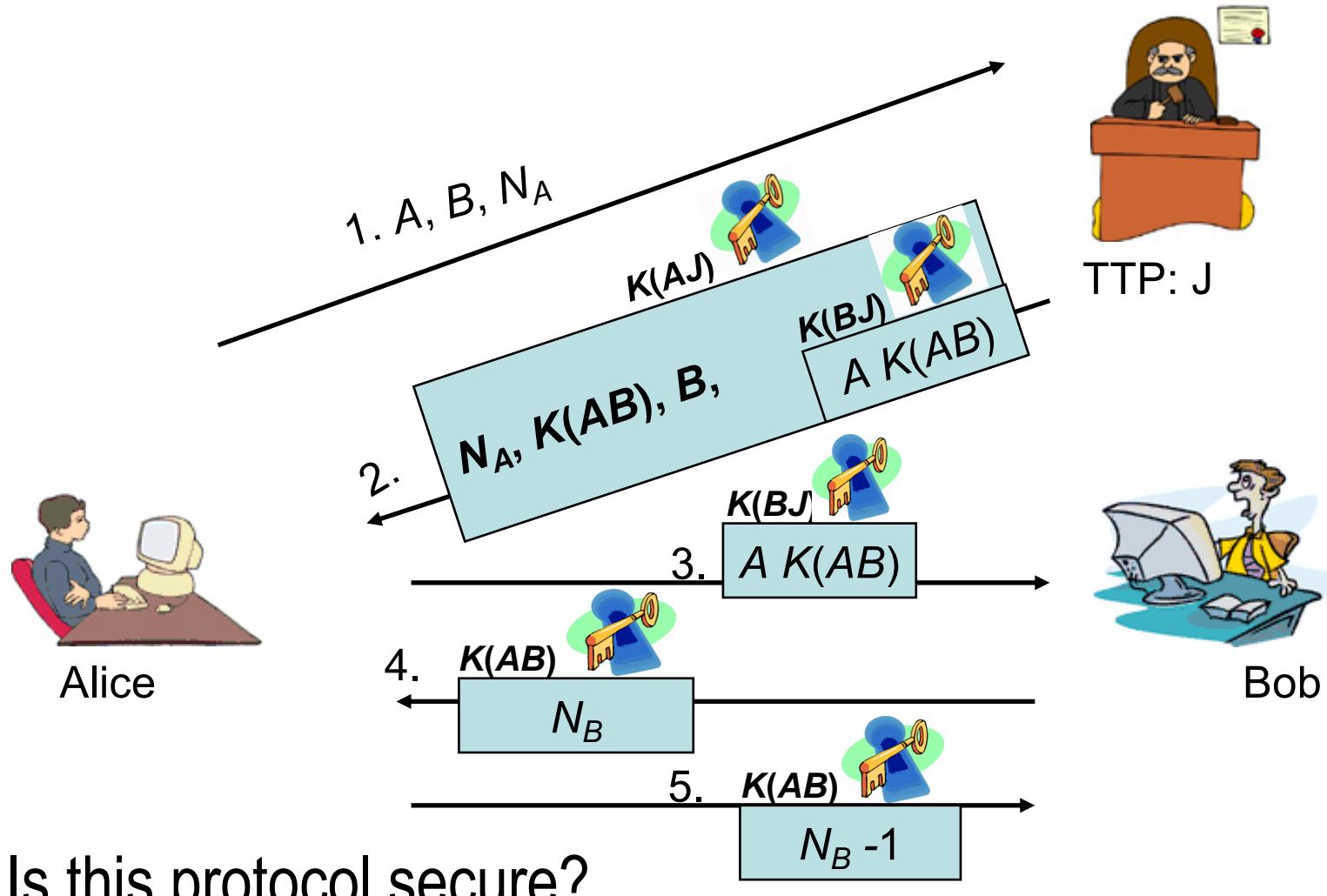
# A simple key exchange protocol

$K(AB)$ : a secret key of  $A$  and  $B$   
 $K(BJ)$ : a secret key of  $B$  and  $J$   
 $K(AJ)$ : a secret key of  $A$  and  $J$



- Is this protocol secure?
  - How could Bob know whom he is talking to?

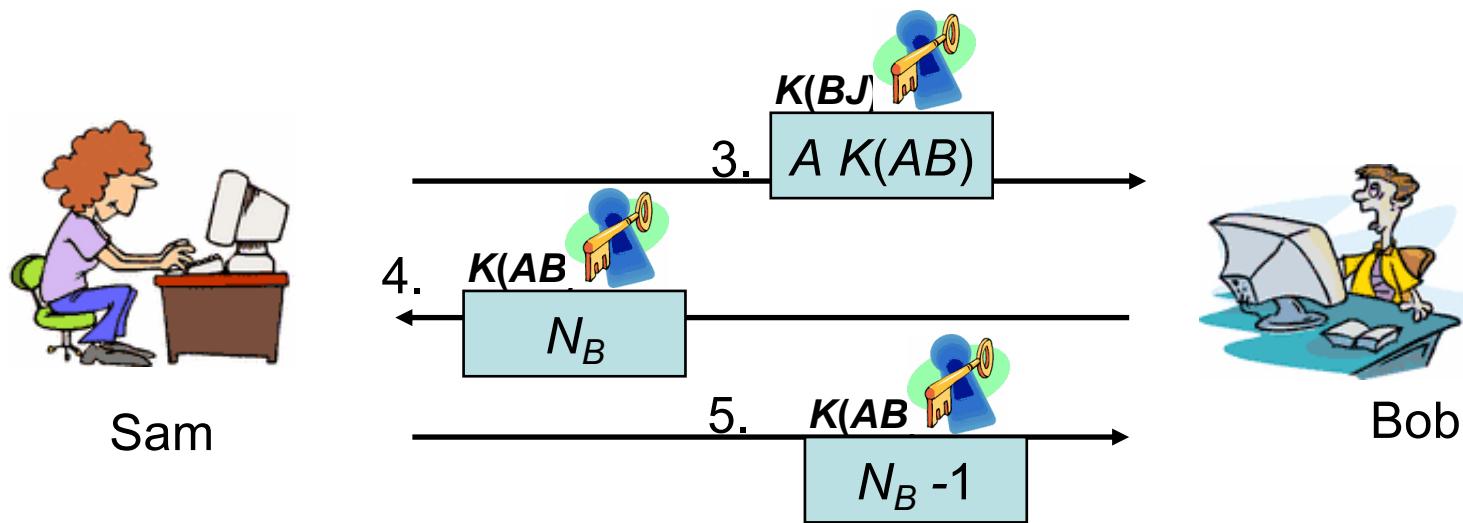
# The Needham-Schroeder protocol



- Is this protocol secure?

# Denning-Sacco replay attack

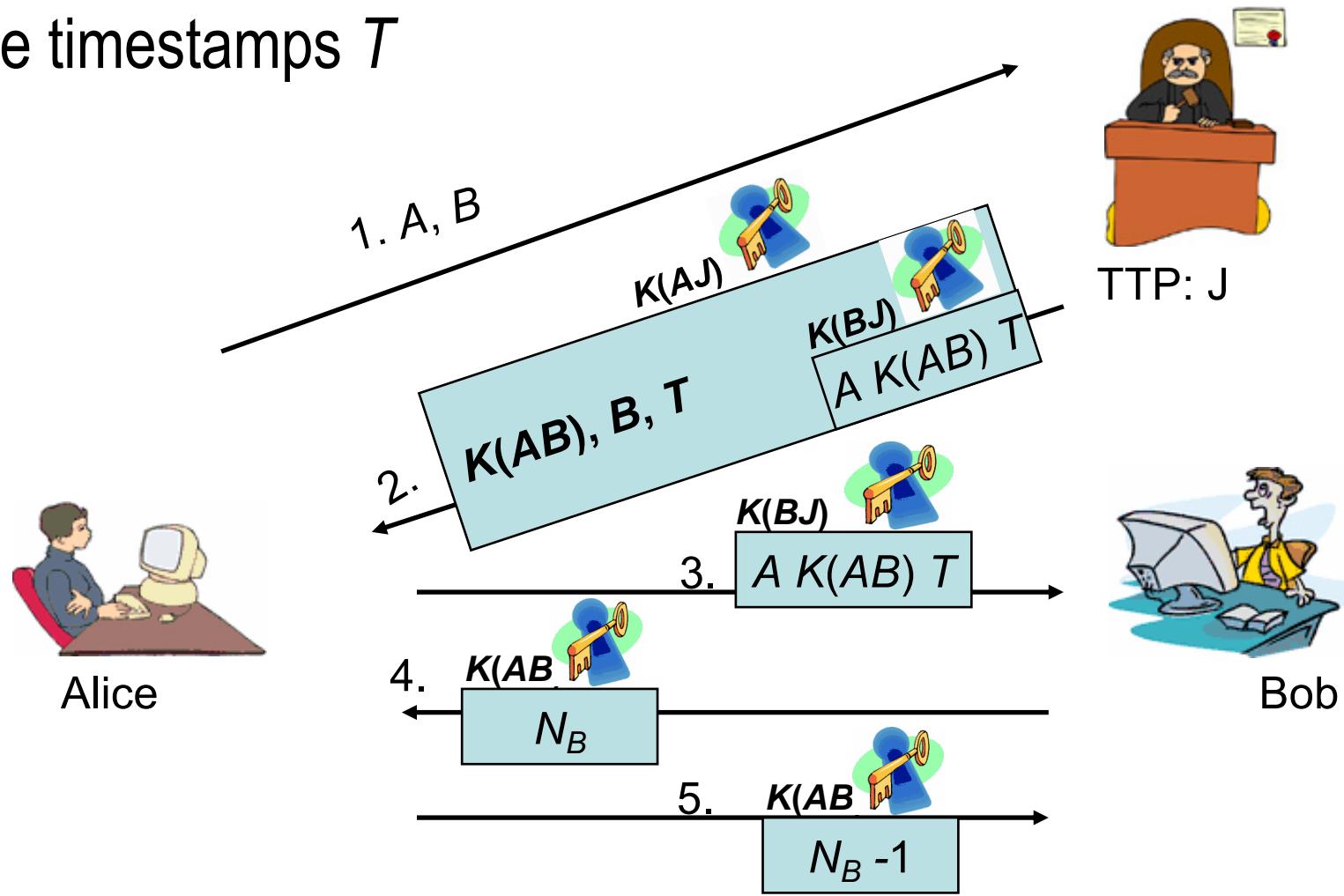
- If Sam stole the session key of last session,  $K(AB)$ , he can play the following game



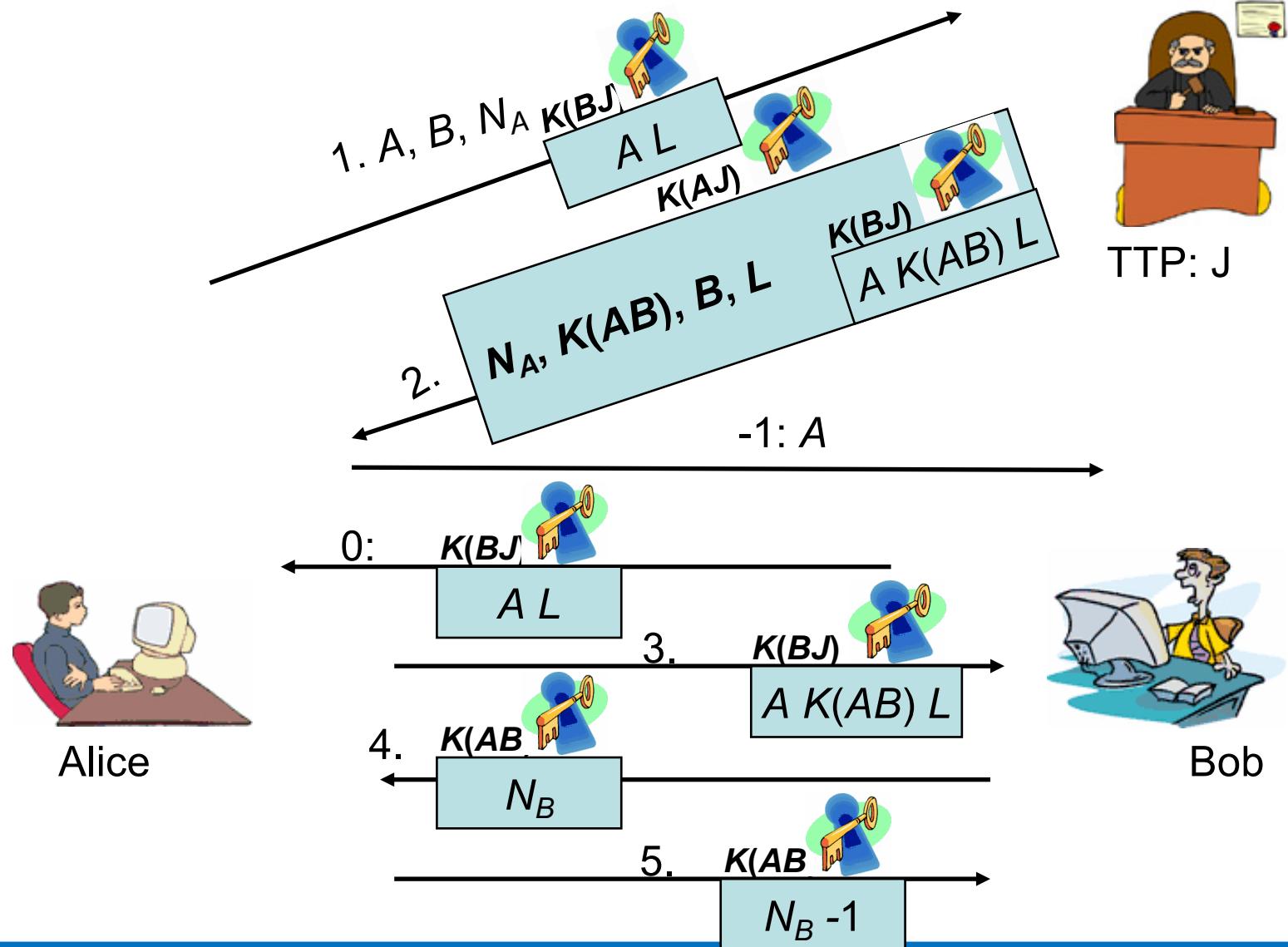
- Sam can replay 3 to 5, to impersonate Alice

# Denning-Sacco fix

- Use timestamps  $T$

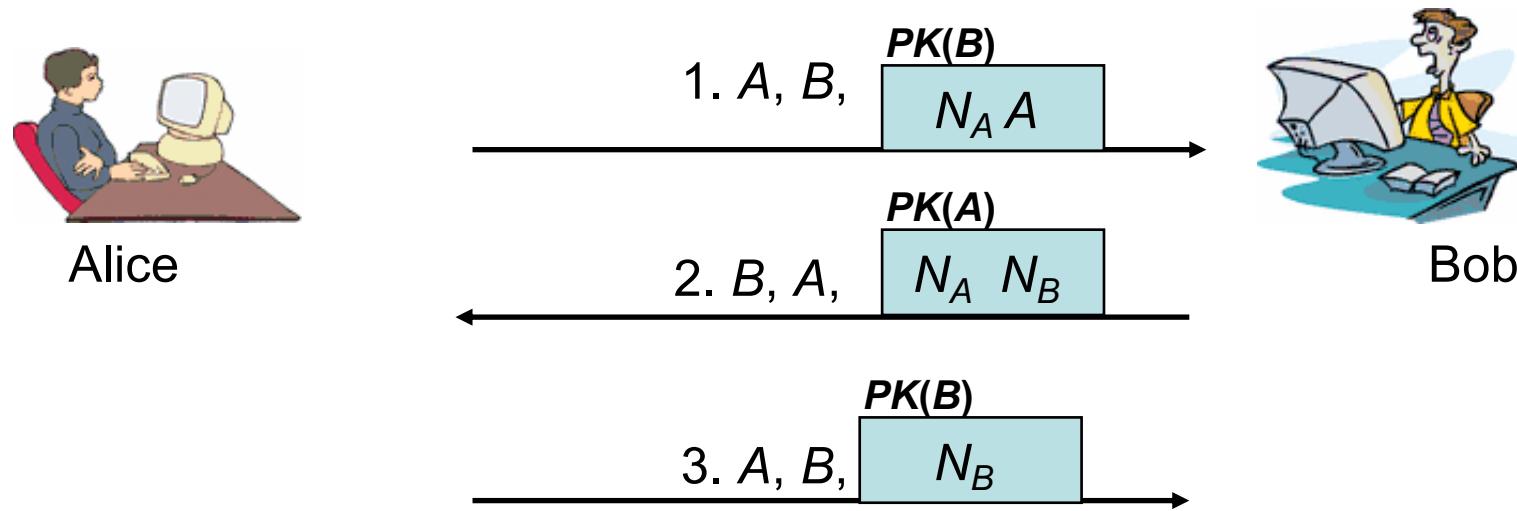


# Needham-Schroeder fix



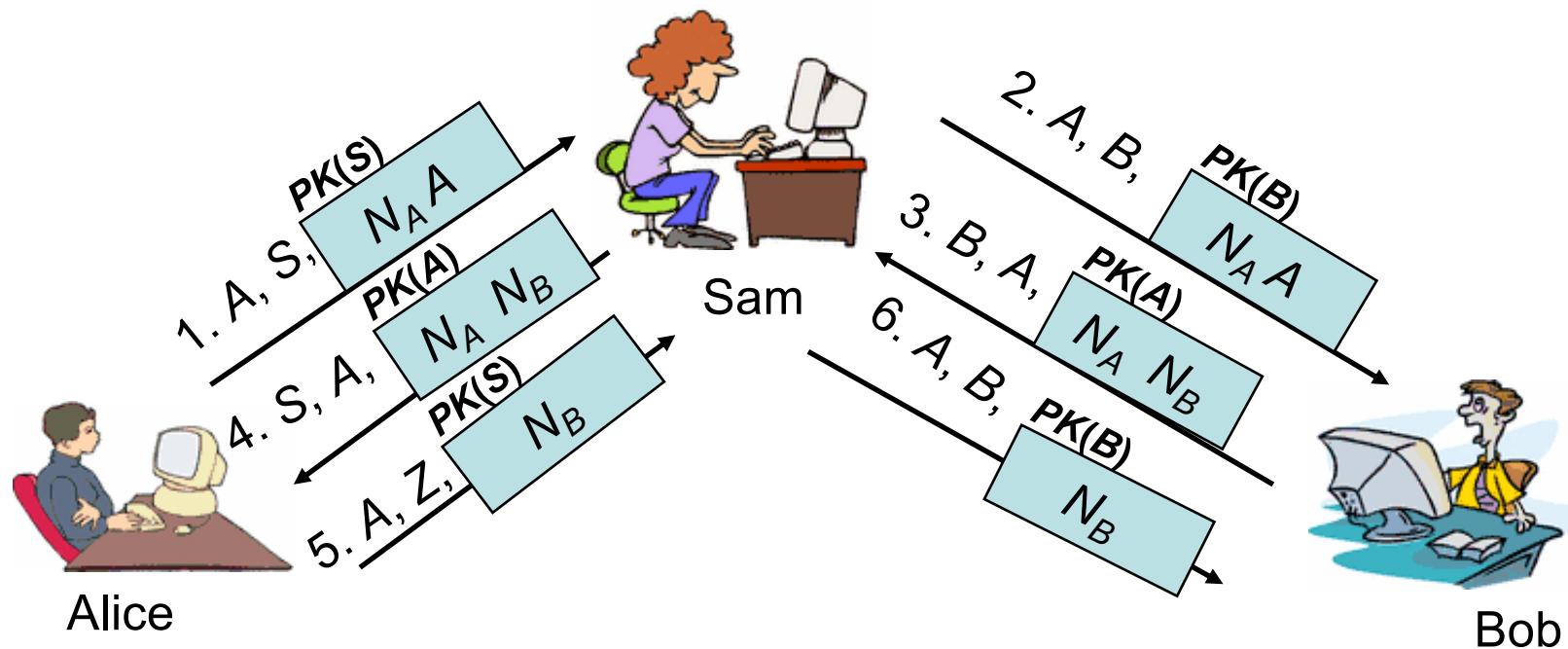
# Needham-Schroeder public key authentication

$PK(A)$ : a public key of A  
 $PK(B)$ : a public key of B

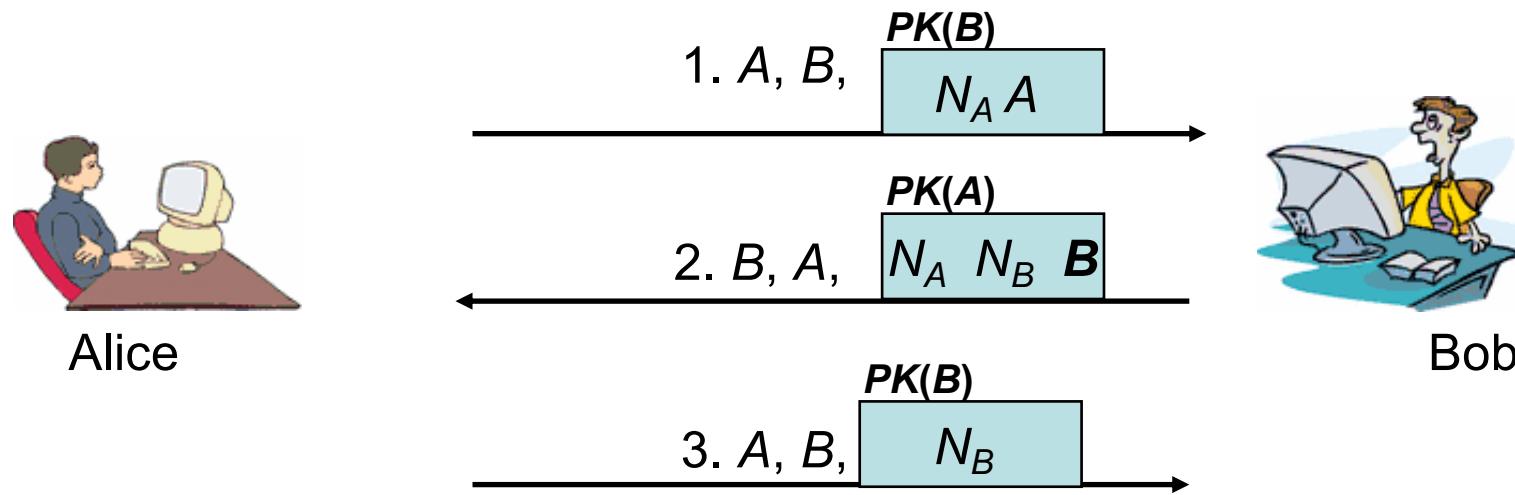


- Any problem in this protocol?

# Man-in-the-middle attack



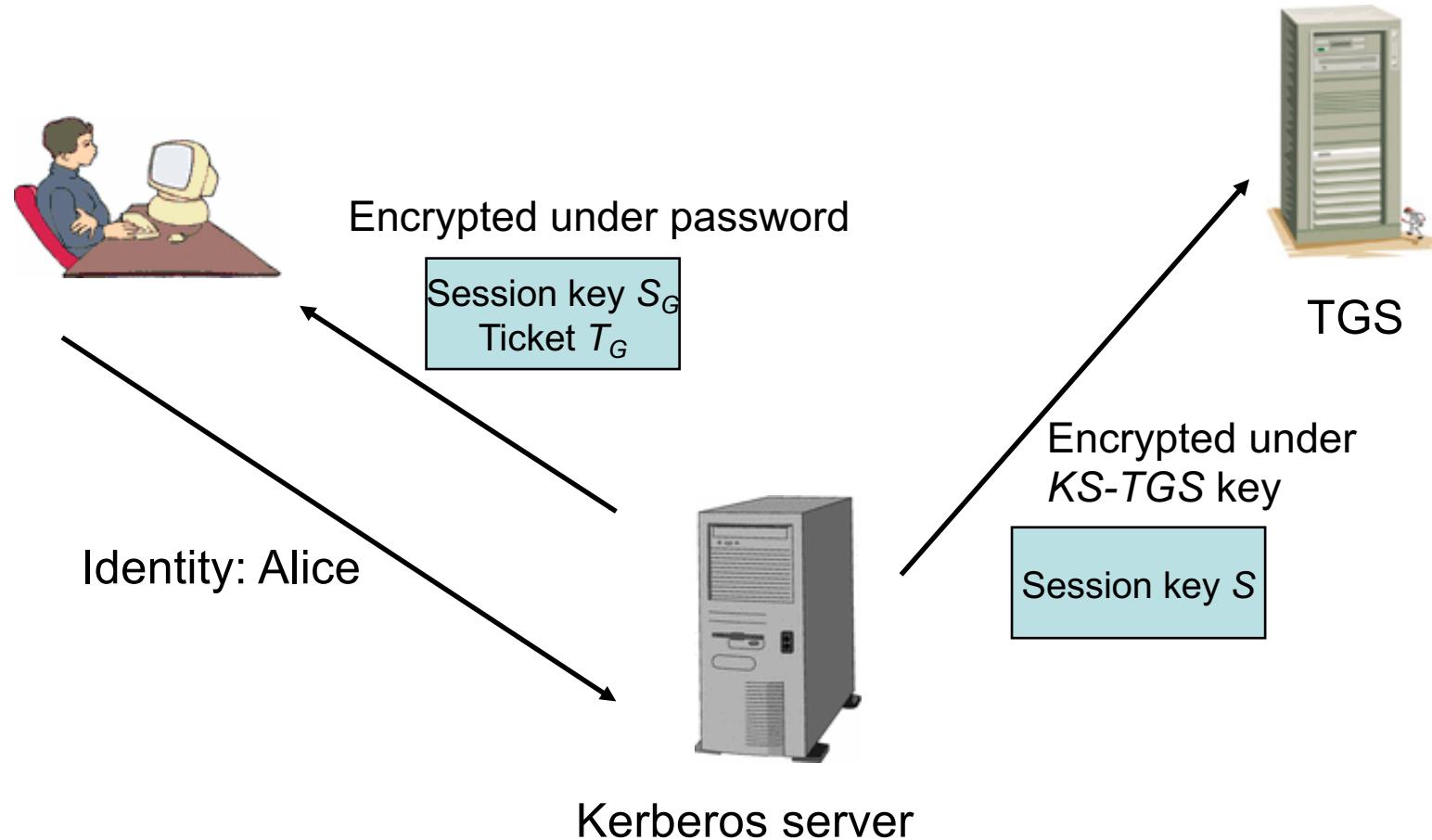
# A fix



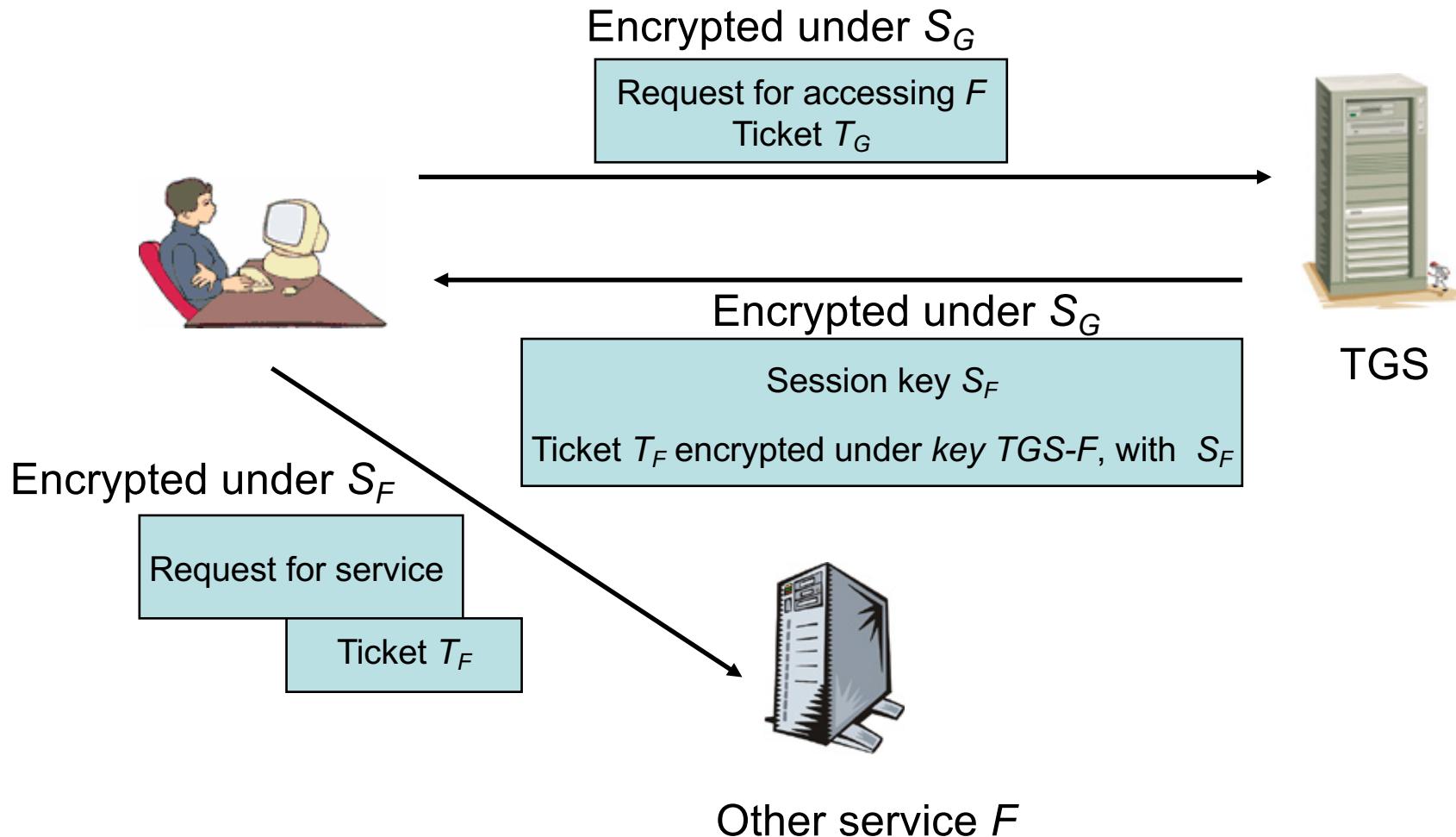
# Kerberos

- Trusted 3rd party authentication protocol designed for TCP/IP networks
  - Based on Needham-Schroeder + timestamp
- Kerberos ticket
  - An authentication/access control token with username, service, time and control information
- Actors in Kerberos system
  - Clients
  - Kerberos server
  - Ticket-Granting Server (TGS)
  - Other servers

# Initiating a Kerberos session



# Obtain a ticket to access a service



# Security strength of Kerberos

- No passwords passing on the network
- Protection against spoofing
- Limited period of validity
- Timestamps to prevent replay attacks
- Mutual authentication
  - 1+user's timestamp

# Kerberos Vulnerabilities

- Require continuous availability of a trusted TGS
- Require a trusted relation between TGS and  $F$
- Require timely transactions
- A subverted workstation saves passwords
- Password guessing works
- Kerberos does not scale well