

# Authentication Protocol

---

Yuan Xue

Nov 7 2006



# Authentication Basics

## ◆ Terms

- Authentication vs. integrity
- Message authentication = data integrity
- **Source authentication**
- non-repudiation

## ◆ **Authentication** is the process of reliably verifying the identity of someone (or something)

- A computer authenticates another computer
- A computer is authenticates a person
  - ◆ User's secret must be remembered by the user

## ◆ Secure communication

- Initial authentication handshake →  
Integrity protection and/or encryption of the data



# Authentication Approaches

---

- ◆ Password-based
- ◆ Address-based
- ◆ Cryptographic
- ◆ Other approaches

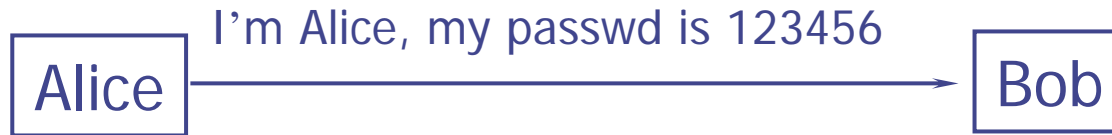
## Basic Guideline

- What you know
- What you have
- What you are



# Password-Based Authentication

- ◆ Authentication based on **what you know**



- ◆ Problem

- Eavesdropping
- Solution → cryptography-based

- ◆ Storing user passwords

- Password can not be stored in cleartext
- Store hashes of the password
- Store encrypted version

- ◆ Password guessing

- Online vs. offline



# Address-based Authentication

- ◆ Authentication based on **where you are**
  - Infer the identity of the source based on the network address
- ◆ UNIX Berkeley rtools
  - Computer B has a list of network addresses of “equivalent machines”
    - ◆ If A is listed, then any account on A is equivalent to the same account name on B
  - Computer B has a list of <address, remote account, local account>
    - ◆ E.g. <A, Alice, Bob>, then request from A with name Alice will be authorized with account Bob



# Address-based Authentication

---

- ◆ In UNIX, `hosts.equiv` and `.rhosts` files list hosts and users that are trusted by the local host when a connection is made using the [rshd](#) service
- ◆ A global file `/etc/hosts.equiv` contains trusted remote hosts.
- ◆ In each user's home directory, a per-user `.rhosts` file contains host-user pairs.



# Address-based Authentication

## ◆ File format

- ***hostname [username]***
- + anyhost/user

## ◆ Example hosts.equiv entries ( Local computer A )

- ++
  - ◆ Allows any user from any host to connect to A
- B +
  - ◆ Allows any user from the remote hosts B to connect to A.
- + Alice
  - ◆ Allows the user Alice to connect to A from any remote host.

## ◆ Example .rhosts entries

- In these examples, the .rhosts file is in the home directory of the user Alice on computer A.
- ++
  - ◆ Allows any user from any host to connect to this host (A) as the user Alice.
- B Bob
  - ◆ Allows the user Bob from the remote host B to connect to A as the user Alice.

## ◆ Example of how the hosts.equiv and the .rhosts file combine

- Rule: **the most restrictive combination of the entries applies**
- hosts.equiv file with <+ Alice>
- .rhosts in the home directory of the user Alice with the following entry: B +
- In this case, these entries combine to mean that only the user Alice from the remote host B can connect to A as Alice.



# Address-based Authentication

- ◆ Entry `<+ +>` → severe security hazards
  - It allows any user on any machine to connect to the local host as the same user name.
  - If it is specified in the `/etc/hosts.equiv` file, it allows any user on any machine to connect to the local host as any user.
- ◆ If A trusts B and B is hacked, then the attacker could gain access to A.
- ◆ Network address impersonation (address spoofing)
  - Ingress filtering



# Cryptographic Authentication

## ◆ Basic idea

- Alice proves her identity to Bob by performing a cryptographic operation on a quantity Bob supplies.
- The cryptographic operation performed by Alice is based on Alice's key.
- Cryptographic operations include
  - ◆ Symmetric key, asymmetric key, hash operations.

## ◆ Where to get the key

- Computer
- Person: password → key
  - ◆ Doing a hash of the password
  - ◆ Using the password to encrypt/decrypt a key which is stored in a computer/dir service



# Cryptographic Authentication

---

## ◆ Two types of authentication

- One-way authentication (login only)
- Mutual authentication

## ◆ One-way authentication

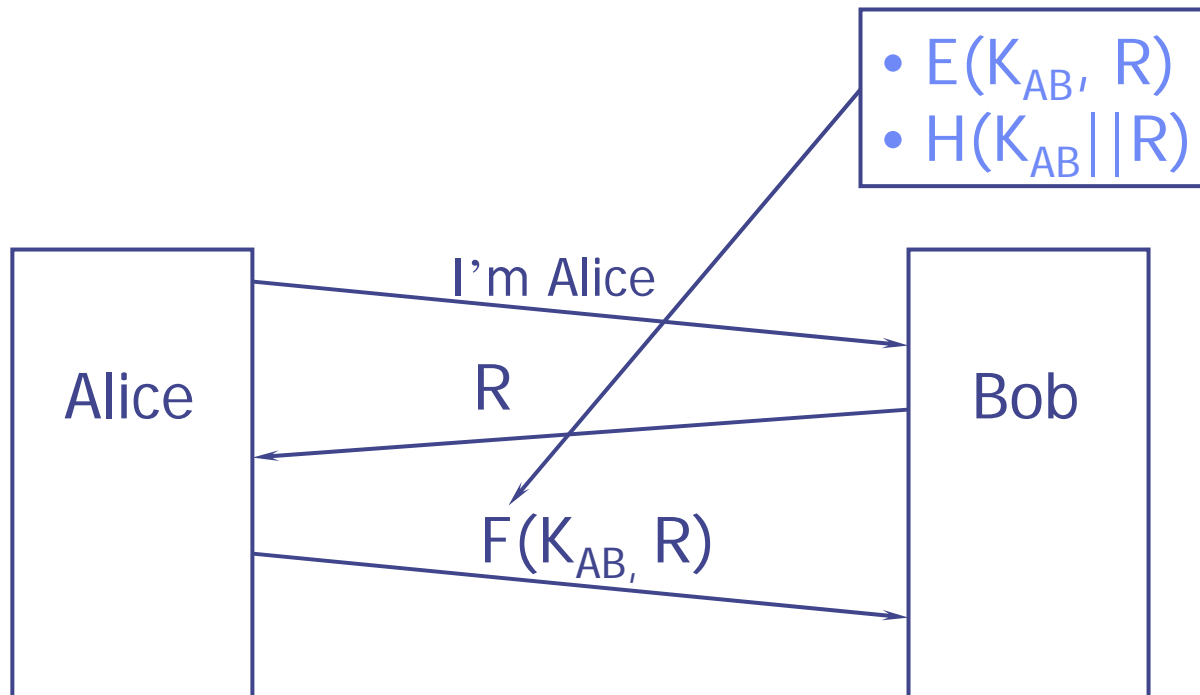
- Alice sends  $\langle \text{name} + \text{Passwd} \rangle \rightarrow \text{Bob}$
- Replace the transmission of cleartext passwd with a cryptographic challenge/response
  - ◆ Symmetric/Asymmetric/Hash operations



# Cryptographic Authentication

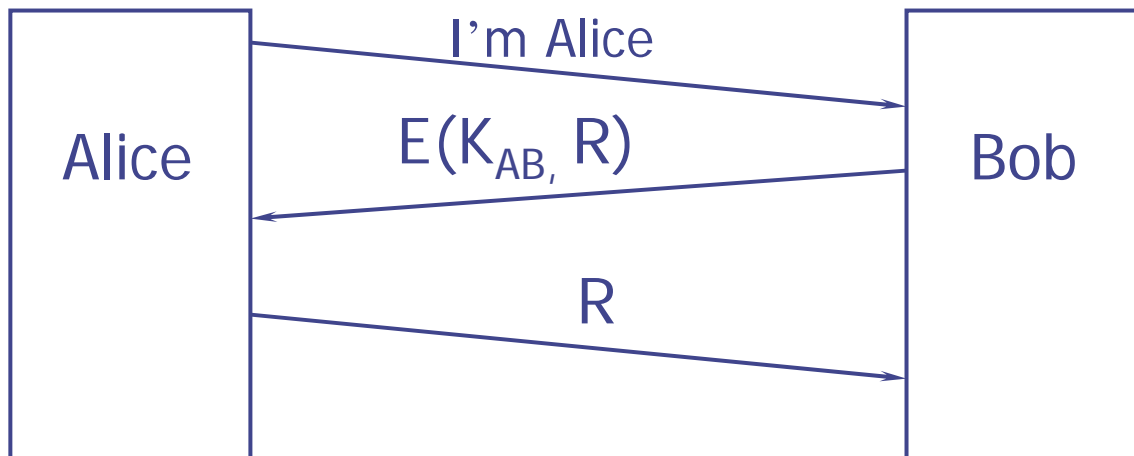
## ◆ Symmetric Key Based Authentication

- Not a mutual authentication
- Offline-password guessing attack
- If the database at Bob is hacked, attacker could impersonate Alice



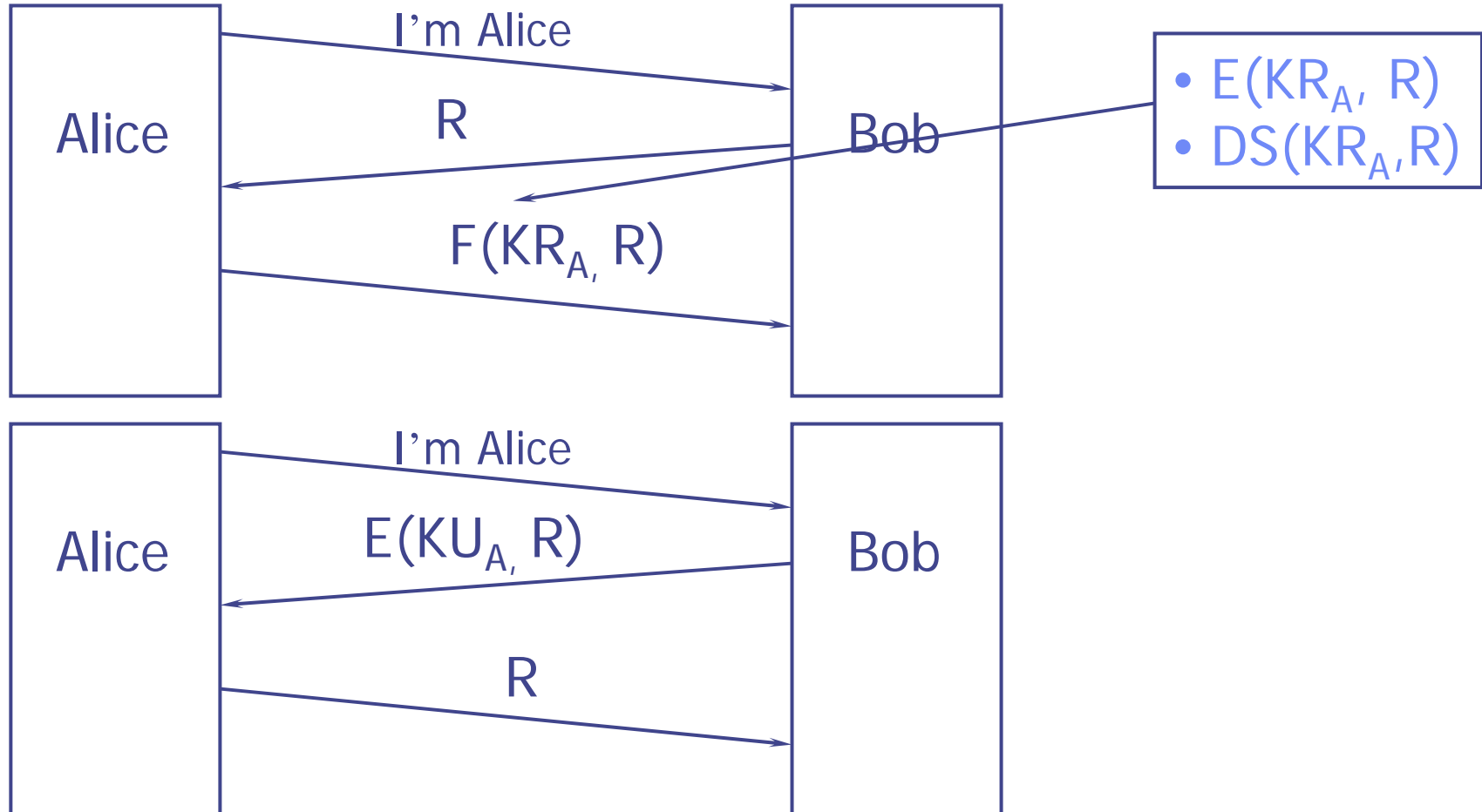
# Cryptographic Authentication

- ◆ Symmetric Key Based Authentication
  - Requires reversible cryptography
  - Vulnerability to dictionary attack
  - Some support for mutual authentication



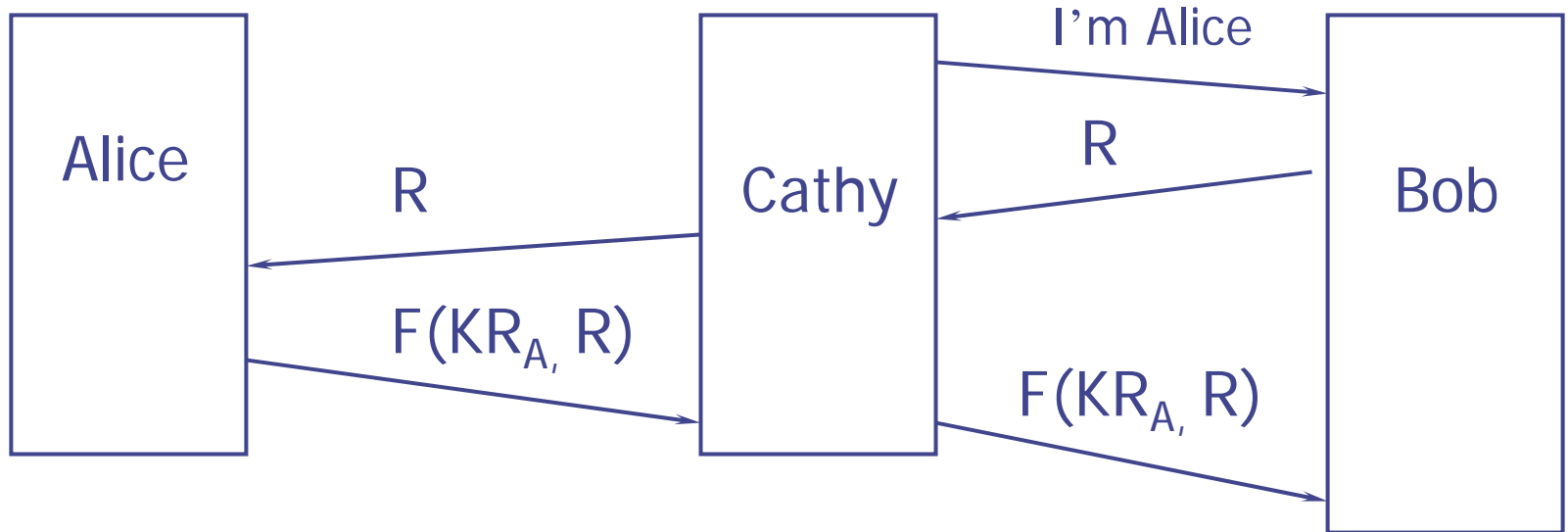
# Cryptographic Authentication

## ◆ Asymmetric Key Based Authentication



# Cryptographic Authentication

- ◆ A serious problem -- Reflection attack

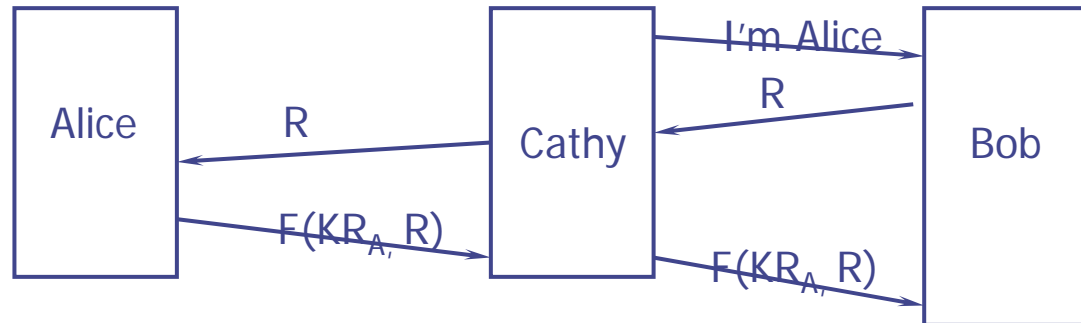


- ◆ Reflection attack
  - A way of attacking a challenge-response authentication system which uses the same protocol in both directions.
  - The basic idea is to trick the target into providing the answer to its own challenge.

# Reflection Attack

## ◆ Attack procedure

- The attacker initiates a connection to a target.
- The target attempts to authenticate the attacker by sending it a challenge.
- The attacker opens another connection to the target, and sends the target this challenge as its own.
- The target responds to that challenge.
- The attacker sends that response back to the target ("reflects" it) on the first connection.
- If the authentication protocol is not carefully designed, the target will accept that response as valid, thereby leaving the attacker with one fully-authenticated channel connection (the other one is simply abandoned).



## ◆ Solutions

- Require the initiating party to first respond to challenges before the target party responds to its challenges.
- Require the key/protocol to be different between the two directions.

# Authentication vs. Key Distribution

## ◆ Cryptographic Authentication

- Based on prior knowledge/ownership of key
- Dependably verify the knowledge of key
- Distribution of session keys

## ◆ Key Distribution

- Prior knowledge/ownership of master keys
- Distribution of session keys after verification of master keys

## ◆ Symmetric key

- A, B share a master key with KDC
- A and B share a master key

## ◆ Asymmetric key

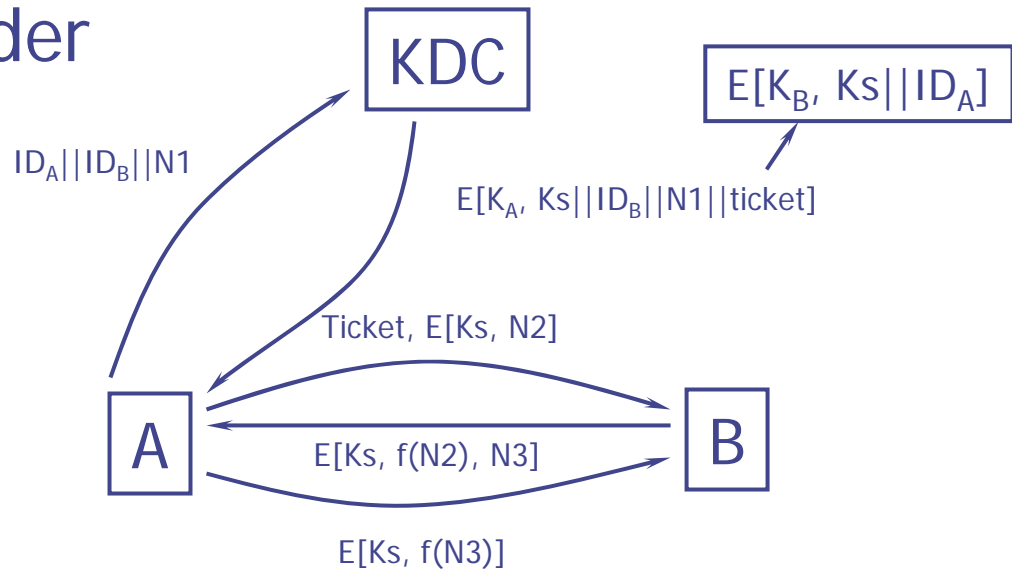
- A and B both have a pair of public and private keys
- The public keys of A and B are reliably distributed to each other



# Authentication

-- KDC Based Symmetric Key

## ◆ Needham-Schroeder



## ◆ Limitation

Replay Attack!

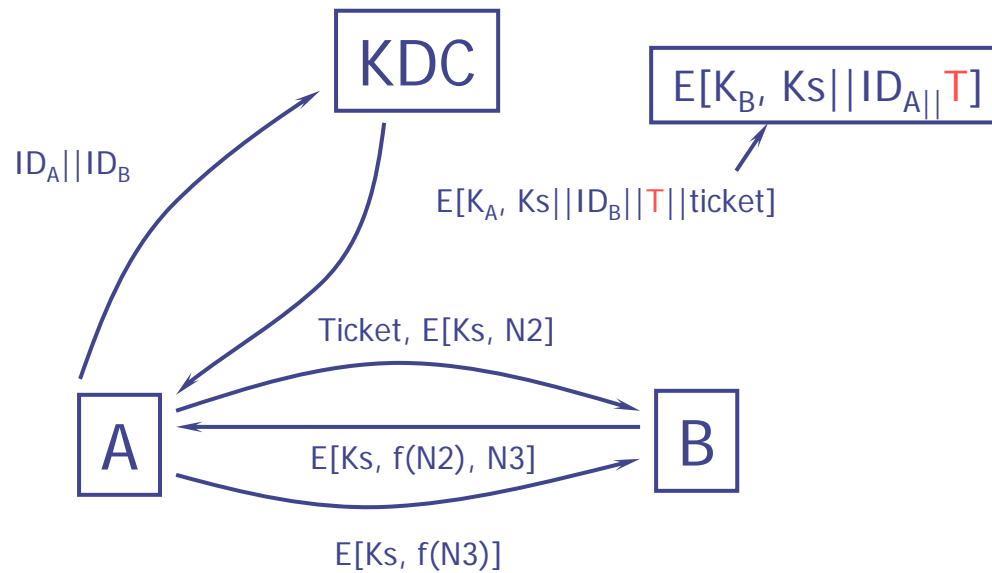
## ◆ Solution

- Add a timestamp



# Authentication

-- KDC Based Symmetric Key



# Kerberos

---

## ◆ An Authentication Service

- Based on client-server model (user and server provider)
- Mutual authentication support: between user and server

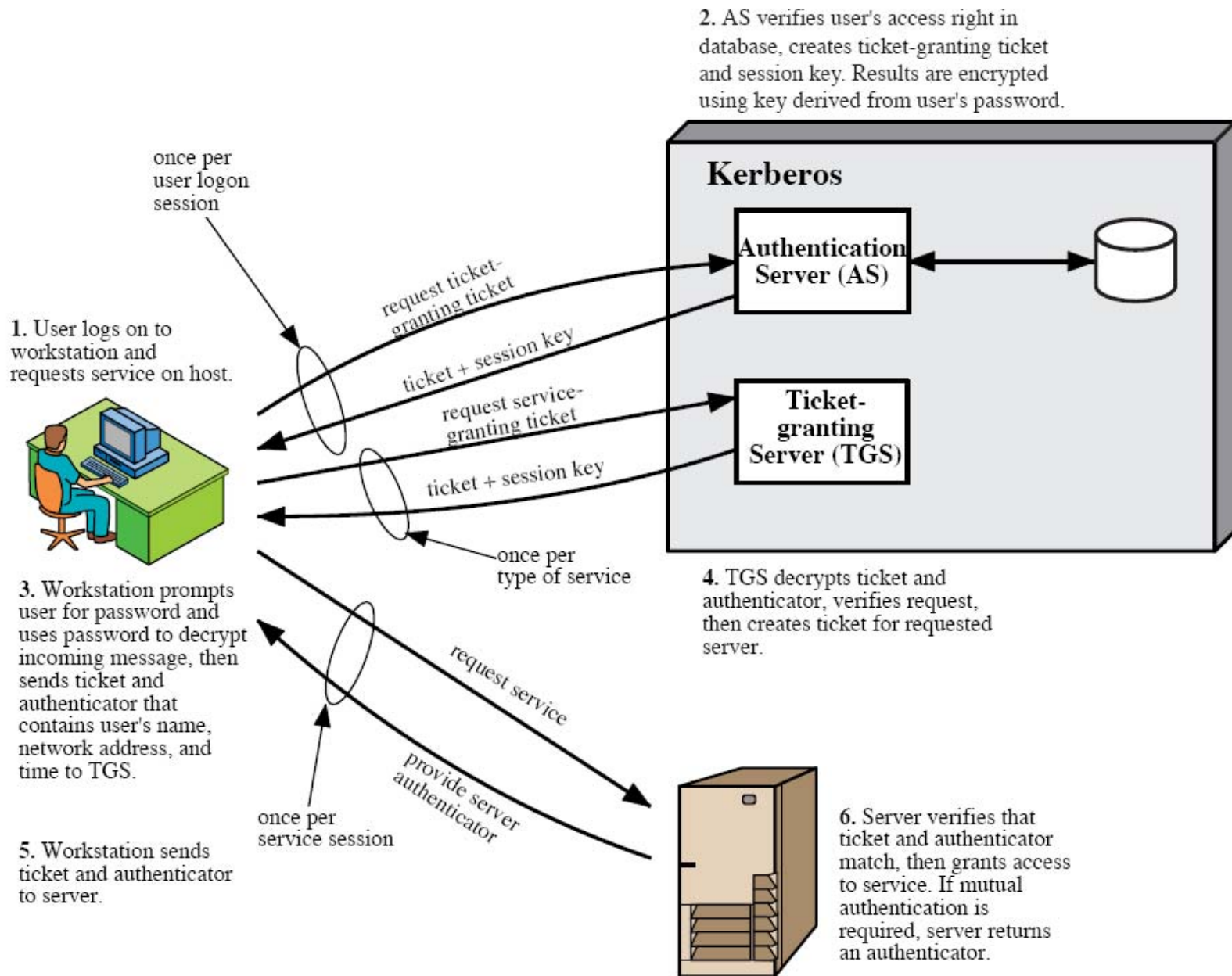
## ◆ Key assumption

- The server can not rely on the client host to authenticate user

## ◆ Basics

- Based on KDC-based symmetric key
- Based on Needham-Schroeder protocol
- Use "tickets" to prove the identity of a user
- Main entities
  - ◆ Authentication Server (AS)
  - ◆ Ticket Granting Server (TGS)





**Figure 14.1 Overview of Kerberos**

**(a) Authentication Service Exchange: to obtain ticket-granting ticket**

(1)  $C \rightarrow AS: ID_C \parallel ID_{tgs} \parallel TS_1$

(2)  $AS \rightarrow C: E_{K_c} [K_{c,tgs} \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2 \parallel Ticket_{tgs}]$

$$Ticket_{tgs} = E_{K_{tgs}} [K_{c,tgs} \parallel ID_C \parallel AD_C \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2]$$

**(b) Ticket-Granting Service Exchange: to obtain service-granting ticket**

(3)  $C \rightarrow TGS: ID_v \parallel Ticket_{tgs} \parallel Authenticator_c$

(4)  $TGS \rightarrow C: E_{K_{c,tgs}} [K_{c,v} \parallel ID_v \parallel TS_4 \parallel Ticket_v]$

$$Ticket_{tgs} = E_{K_{tgs}} [K_{c,tgs} \parallel ID_C \parallel AD_C \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2]$$

$$Ticket_v = E_{K_v} [K_{c,v} \parallel ID_C \parallel AD_C \parallel ID_v \parallel TS_4 \parallel Lifetime_4]$$

$$Authenticator_c = E_{K_{tgs}} [ID_C \parallel AD_C \parallel TS_3]$$

**(c) Client/Server Authentication Exchange: to obtain service**

(5)  $C \rightarrow V: Ticket_v \parallel Authenticator_c$

(6)  $V \rightarrow C: E_{K_{c,v}} [TS_5 + 1]$  (for mutual authentication)

$$Ticket_v = E_{K_v} [K_{c,v} \parallel ID_C \parallel AD_C \parallel ID_v \parallel TS_4 \parallel Lifetime_4]$$

$$Authenticator_c = E_{K_{c,v}} [ID_C \parallel AD_C \parallel TS_5]$$

# Authentication Summary

---

- ◆ Security in communication
  - An initial authentication handshake
  - Distribution of session keys
  - Then integrity protection and/or encryption of the data
- ◆ Types
  - One-way
  - Mutual
- ◆ Approaches
  - Symmetric key (KDC, distributed), Asymmetric key
- ◆ Authentication Service
  - Kerberos (composed **Needham-Schroeder** protocol)
- ◆ Attacks to Authentication
  - Replay Attack
  - Reflection Attack

