



# 第六章

## 身份认证与数字签名



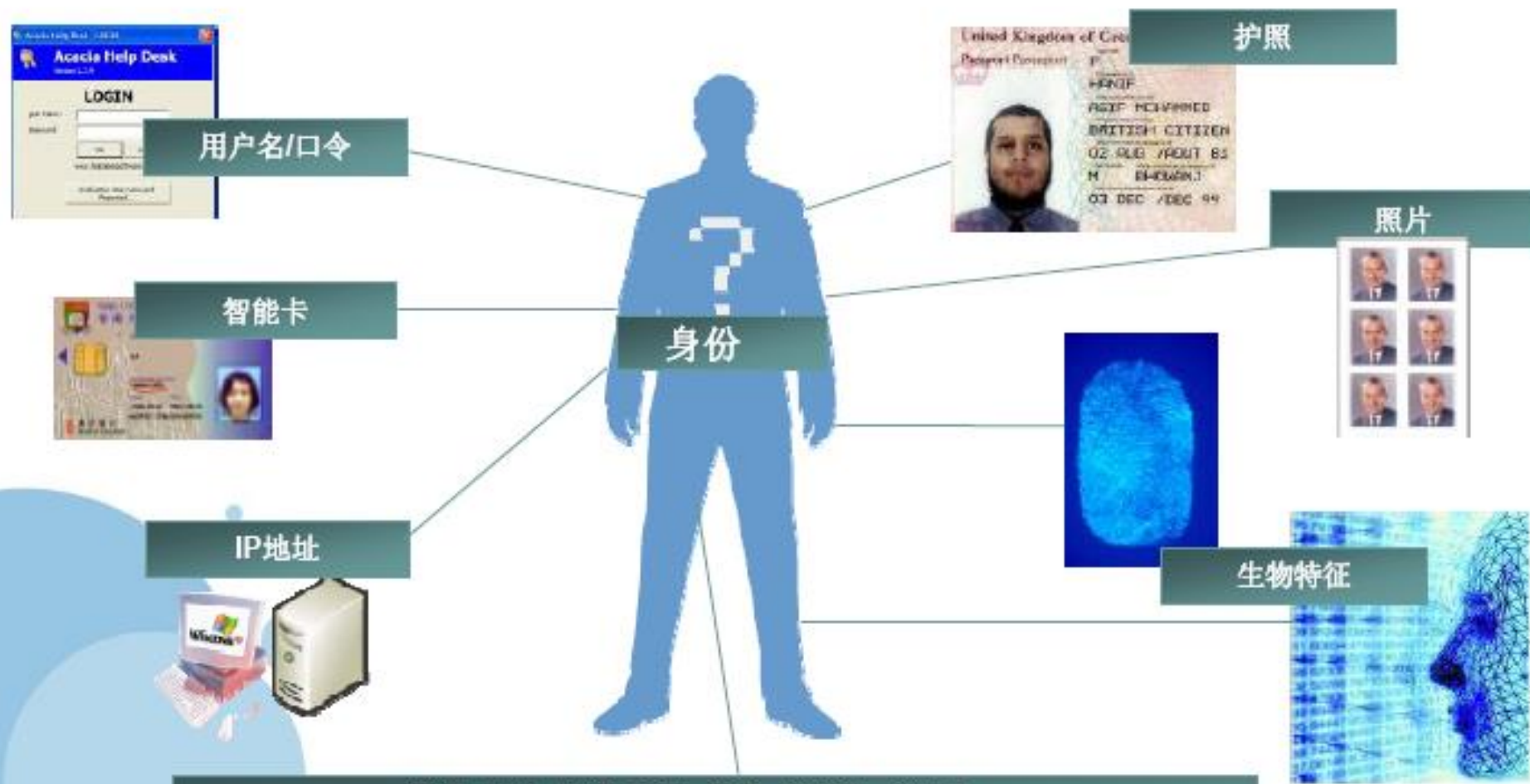
# 主要内容

- ❖ 身份认证
- ❖ 数字签名

# “身份”是什么？

对于身份管理系统的一种定义：

一个通过软件来实现其流程和策略的系统，用来管理数字身份信息的生命周期以及授权信息。



## 6.1 身份认证

- ❖ 身份认证是验证主体的真实身份与其所声称的身份是否符合的过程。
- ❖ 认证的结果只有两个：符合和不符合。
- ❖ 适用于用户、进程、系统、信息等。

# 身份认证的例子

- ❖ 邮件登录
- ❖ **Client**与**Server**之间的鉴别
- ❖ **Telnet**远程登录
- ❖ **Ftp**服务
- ❖ 登录到某台电脑上



# 身份认证系统的组成

- ❖ 出示证件的人，称作示证者P(Prover)，又称声称者(Claimant)。
- ❖ 验证者V (Verifier),检验声称者提出的证件的正确性和合法性，决定是否满足要求。
- ❖ 第三方是可信赖者TP (Trusted third party),参与调解纠纷。在许多应用场合下没有第三方。

# 6.1.1 身份认证的物理基础

## ❖ **Something the user know** (例如口令)

- ❧ 在互联网和计算机领域中最常用的认证方法是口令认证
- ❧ 简单，但不安全。口令有可能被窃取、丢失、复制。

## ❖ 设计依据

- ❧ 安全水平、系统通过率、用户可接受性、成本等

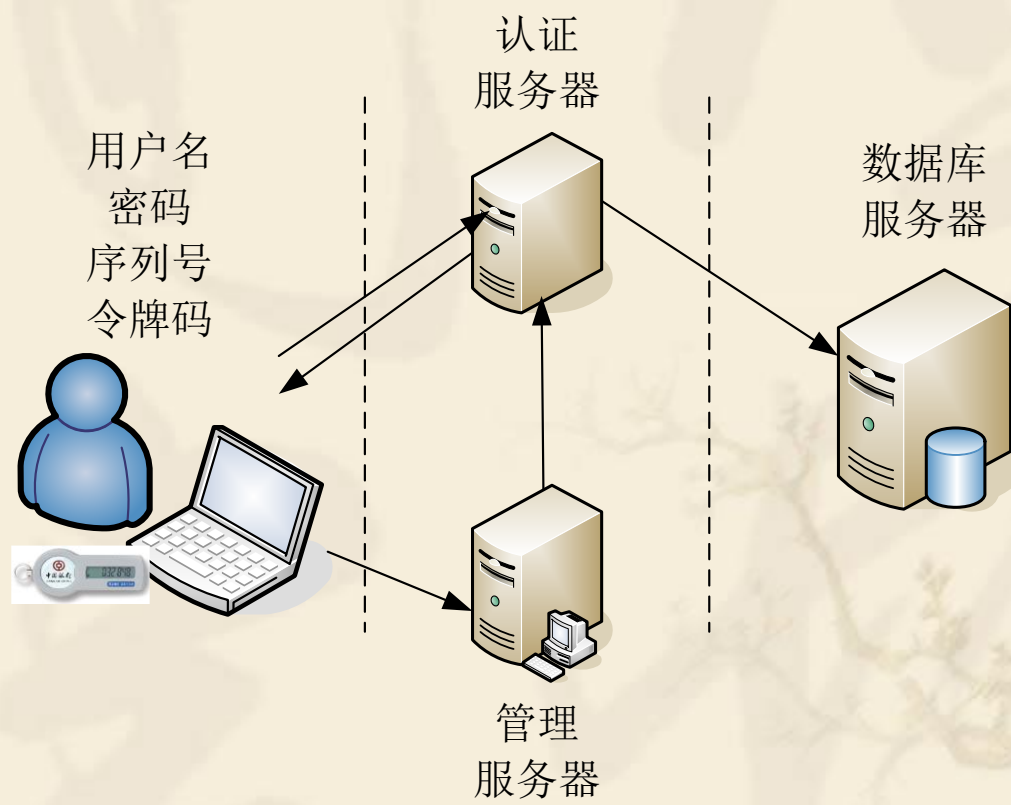
❖ 口令一般并不是以明文的形式存在和使用，而是采用一些加强的处理之后才使用的。

∞ **对口令加密**：对口令的加密算法必须是单向的，即只能加密，不能解密。在验证用户的口令时，验证方用单向函数加密，并与存储的密文相比较，若相等，则确认用户的身份有效，否则确认用户身份无效。

∞ **一次性口令**：使用一次性口令作为身份认证方法，使得中途截获口令变得毫无意义。由于要产生大量的一次性口令，所以必须采用专用的设备来产生口令。







# 身份认证的物理基础

## ❖ Something the user possesses(例如证件)

∞ 认证系统相对复杂





## 用户所拥有的

- ❖ 磁卡或智能卡丢失，那么捡到卡的人就可以假冒真正的用户。
- ❖ 需要一种磁卡和智能卡上不具有的身份信息，这种身份信息通常采用个人识别号**PIN**。持卡人必须自己妥善保存并严格保密。
- ❖ 在验证过程中，验证者不但要验证持卡人的卡是真实的卡，同时还要通过**PIN**来验证持卡人的确是他本人。

# 身份认证的物理基础

❖ **Something the user is**(例如指纹识别)

∞ 更复杂,而且有时会牵涉到本人意愿



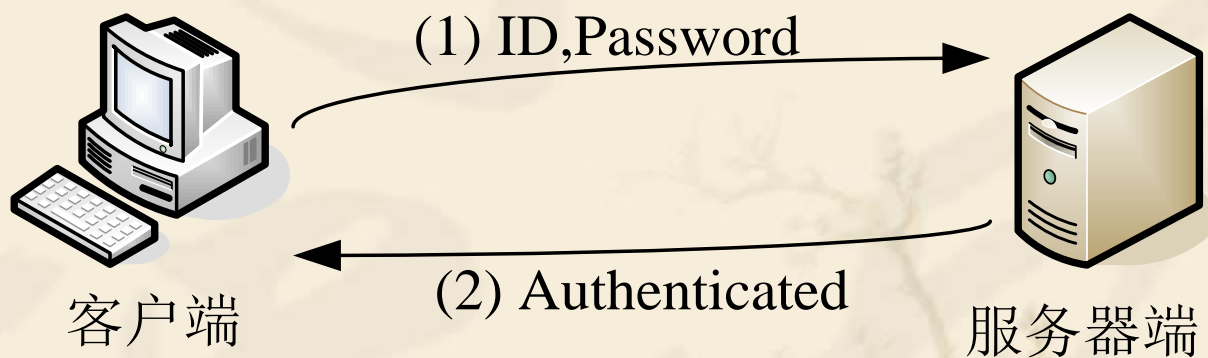


## 6.1.2 身份认证方式

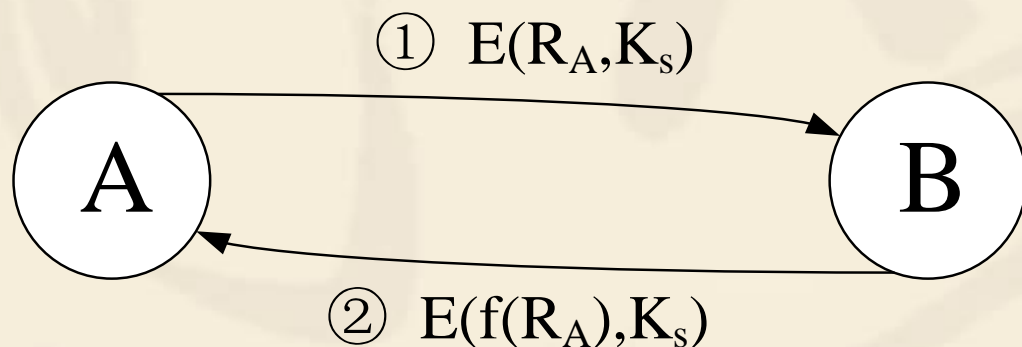
- ❖ 单向认证（One-way Authentication）
- ❖ 双向认证（Two-way Authentication）
- ❖ 信任的第三方认证（Trusted Third-party Authentication）

# 单向认证

❖ 通信的一方认证另一方的身份

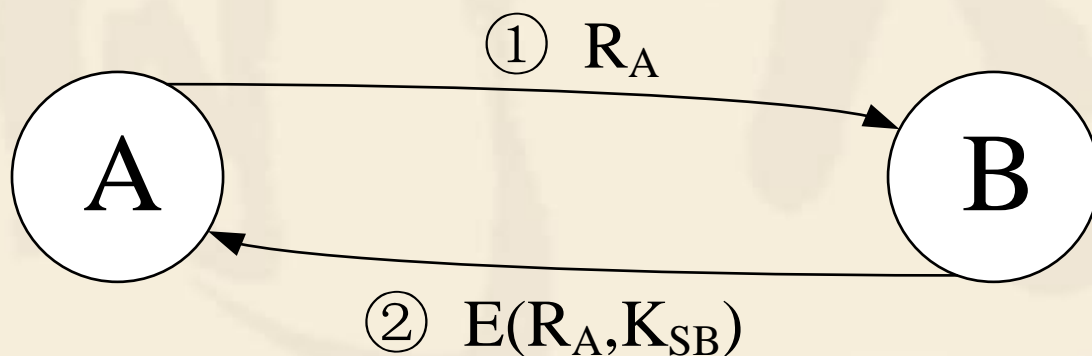


# 用对称密码体制来实现单向认证



- ❖ 某函数变换 $f$
- ❖ 双方共享的密钥 $K_s$
- ❖ 随机数 $R_A$

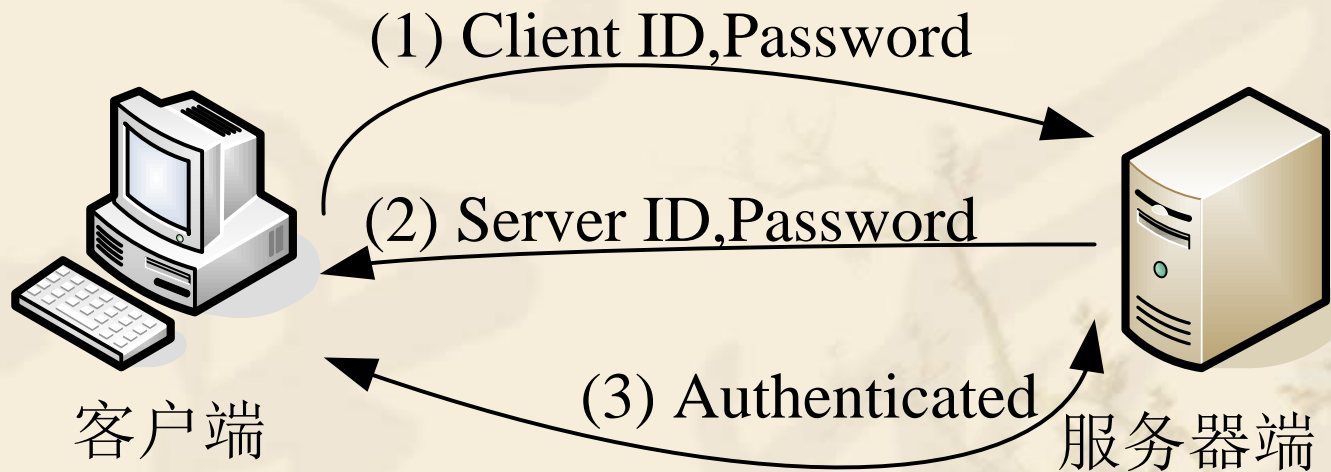
# 用非对称密码体制来实现单向认证



- ❖ 随机数 $R_A$
- ❖ B的私钥 $K_{SB}$

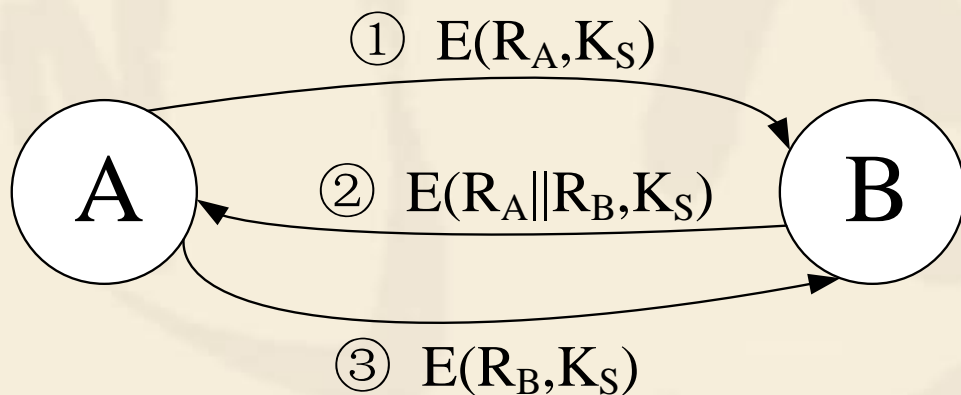
# 双向认证

- ❖ 双方都要提供用户名和密码给对方，才能通过认证。



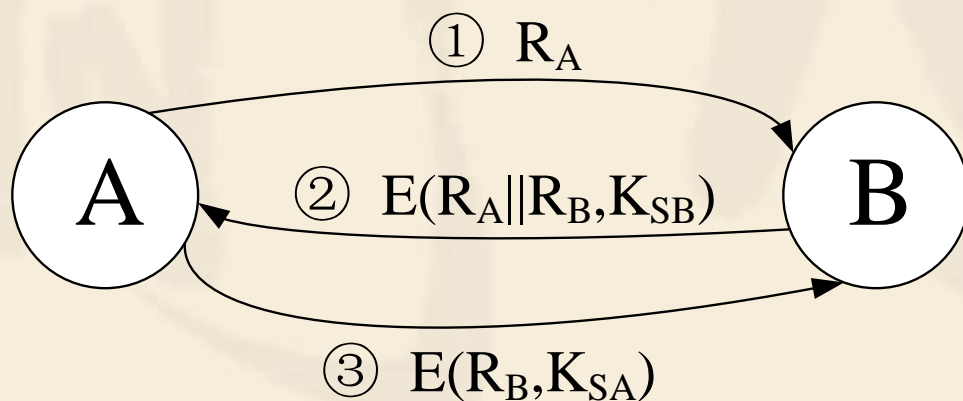


# 用对称密码体制来实现双向认证



- ❖ A产生一个随机数 $R_A$
- ❖ 双方共享的密钥 $K_S$
- ❖ B产生一个随机数 $R_B$

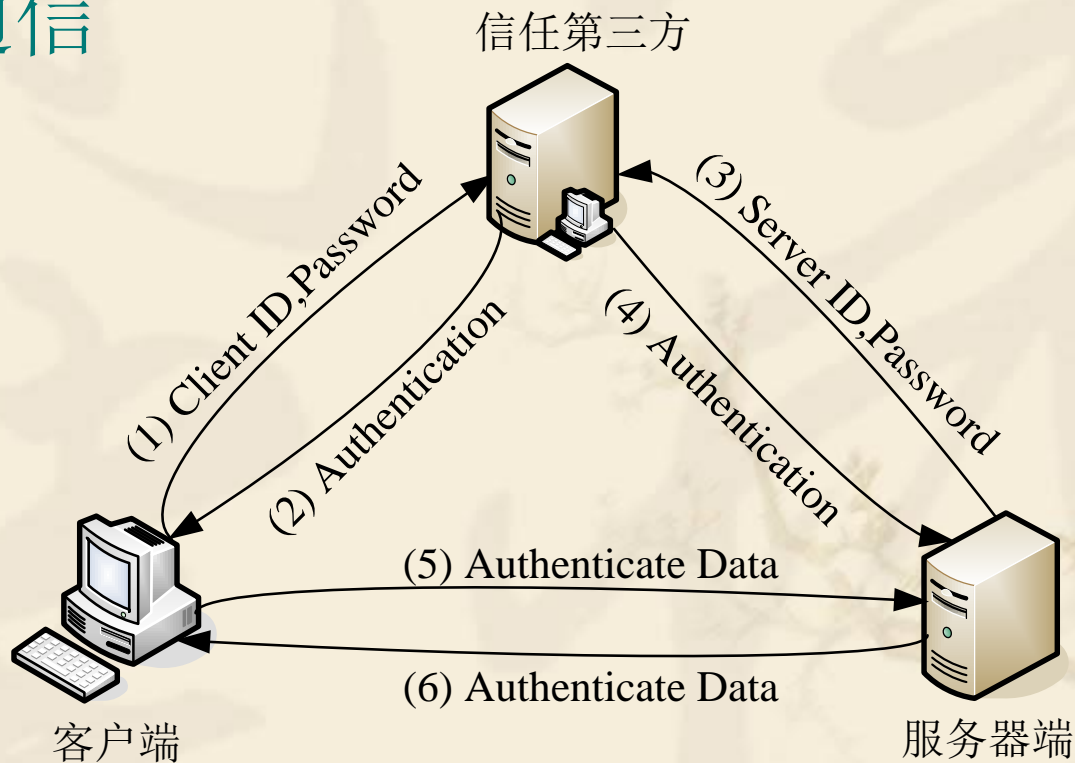
# 用非对称密码体制来实现双向认证



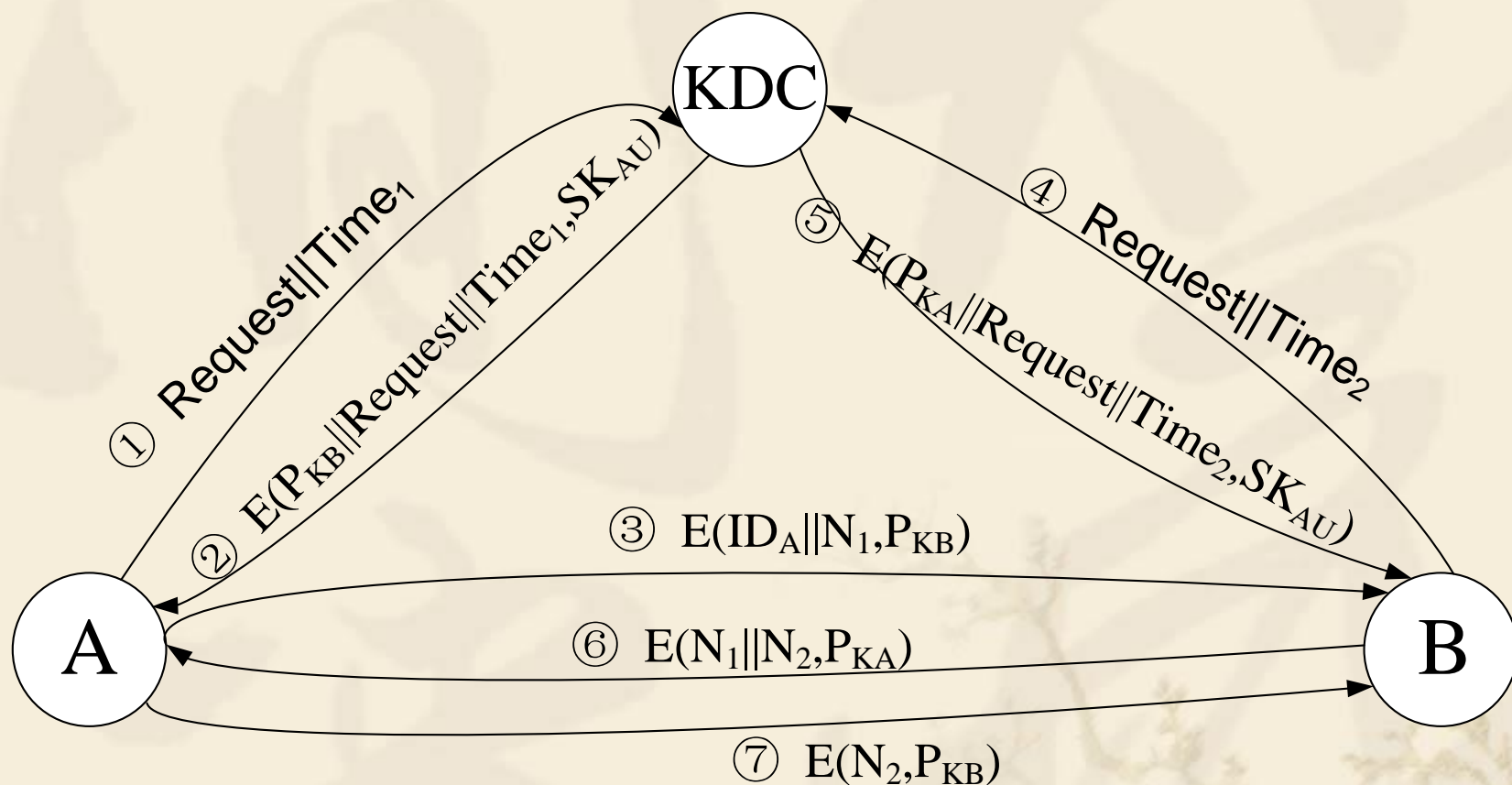
- ❖ A产生一个随机数 $R_A$
- ❖ B产生一个随机数 $R_B$
- ❖ B的私钥 $K_{SB}$
- ❖ A的私钥 $K_{SA}$

# 信任的第三方认证

- ❖ 当两端欲进行连线时，彼此必须先通过信任第三方的认证，然后才能互相交换密钥，而后进行通信



# 一种第三方认证机制



- ❖  $SK_{AU}$ : 管理员的私钥
- ❖  $P_{KB}$ : B的公钥
- ❖  $P_{KA}$ : A的公钥

- ❖  $N_1$ : A的临时交互号
- ❖  $N_2$ : B产生的新临时交互号

## 6.1.3 kerberos协议



# 概述

- ❖ Kerberos是一种计算机网络认证协议，它允许某实体在非安全网络环境下通信，向另一个实体以一种安全的方式证明自己的身份。
- ❖ 它也指由麻省理工实现此协议，并发布的一套免费软件。
- ❖ 它的设计主要针对客户-服务器模型，并提供了一系列交互认证-用户和服务器都能验证对方的身份。
- ❖ Kerberos协议可以保护网络实体免受窃听和重复攻击。
- ❖ Kerberos协议基于对称密码学，并需要一个值得信赖的第三方。

❖ "In Greek mythology, a many headed dog, commonly three, perhaps with a serpent's tail, the guardian of the entrance of Hades."

----- Dictionary of Subjects and Symbols in Art,  
by James Hall, Harper & Row, 1979.



# Key Points

- ❖ Kerberos is an authentication service designed for use in a distributed environment.
- ❖ Kerberos makes use of a trusted third-part authentication service that enables clients and servers to establish authenticated communication.

# Instruction

- ❖ One of the earliest and also one of the most widely used services.
- ❖ Two versions of Kerberos are in common use.
  - ❧ Version 4 implementations still exist.
  - ❧ Version 5 corrects some of the security deficiencies of version 4 and has been issued as a proposed Internet Standard (RFC 1510).



# The problem that Kerberos addresses:

- ❖ Assume an open distributed environment in which users at workstations wish to access services on servers distributed throughout the network.
- ❖ We would like for servers to be able to restrict access to authorized users and to be able to authenticate requests for service.
- ❖ Kerberos provides a centralized authentication server whose function is to authenticate users to servers and servers to users.
- ❖ Kerberos relies exclusively on symmetric encryption, making no use of public-key encryption.

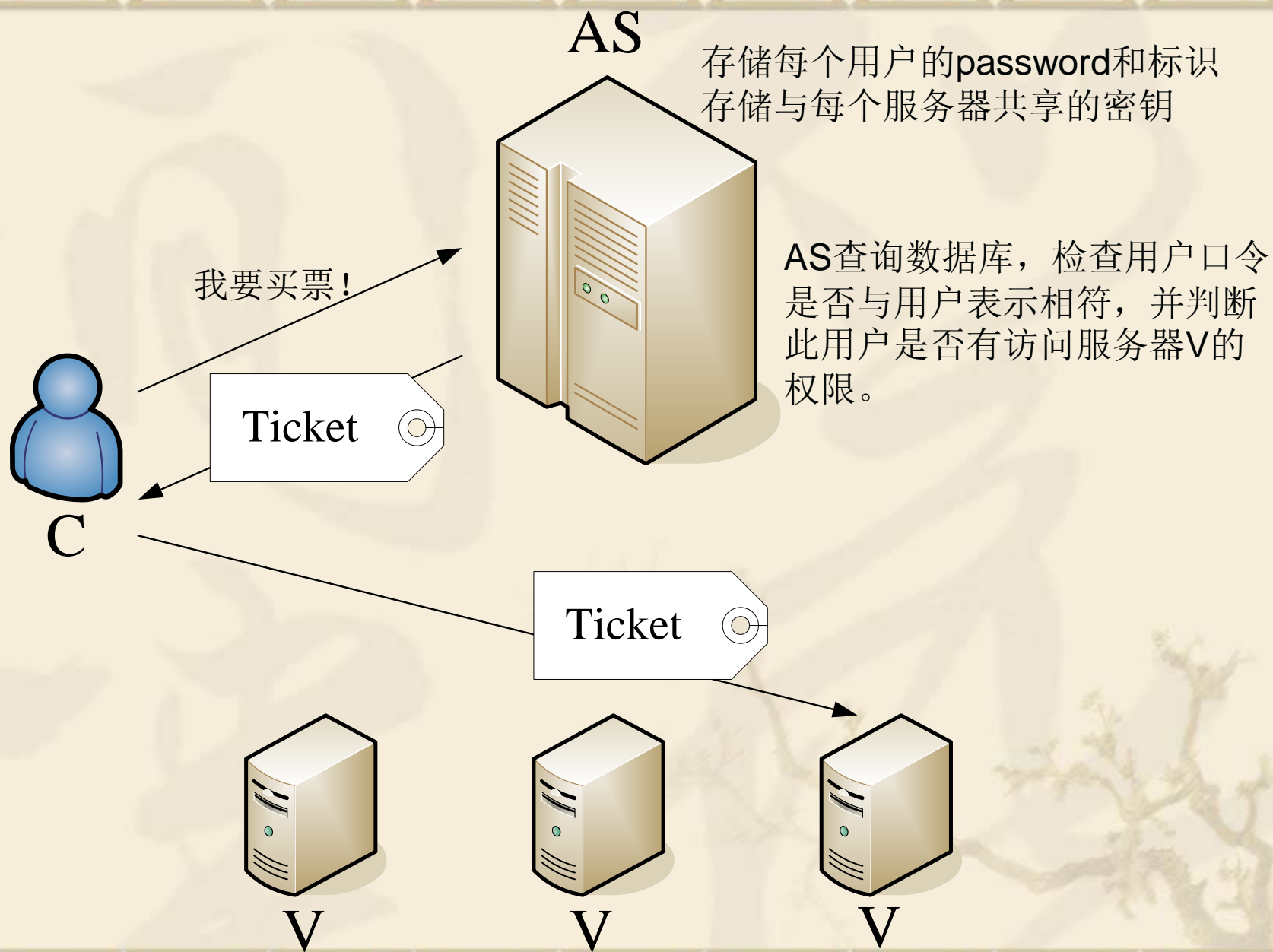


# A Simple Authentication Dialogue

- ❖ In an unprotected network environment, any client can apply to any server for service.
- ❖ servers must be able to confirm the identities of clients who request service. Each server can be required to undertake this task for each client/server interaction.
- ❖ In an open environment, this places a substantial burden on each server.

# Authentication Server (AS)

- ❖ Kerberos认证系统中采用DES作为它的加密算法，每一个使用者都有一个自己的认证密钥（**authentication ,key**），这个认证密钥会存放在认证服务器中。
- ❖ 而认证服务器**AS**负责使用者的身份确认及维护使用者和服务器的资料。



(1)  $C \rightarrow AS: ID_C || P_C || ID_V$

(2)  $AS \rightarrow C: Ticket$

(3)  $C \rightarrow V: ID_C || Ticket$

$Ticket = E(K_v, [ID_C || AD_C || ID_V])$

$C$  = client

$AS$  = authentication Server

$V$  = server

$ID_C$  = identifier of user on  $C$

$ID_V$  = identifier of  $V$

$P_C$  = password of user on  $C$

$AD_C$  = network address of  $C$

$K_v$  = secret encryption key shared by  $AS$  and  $V$



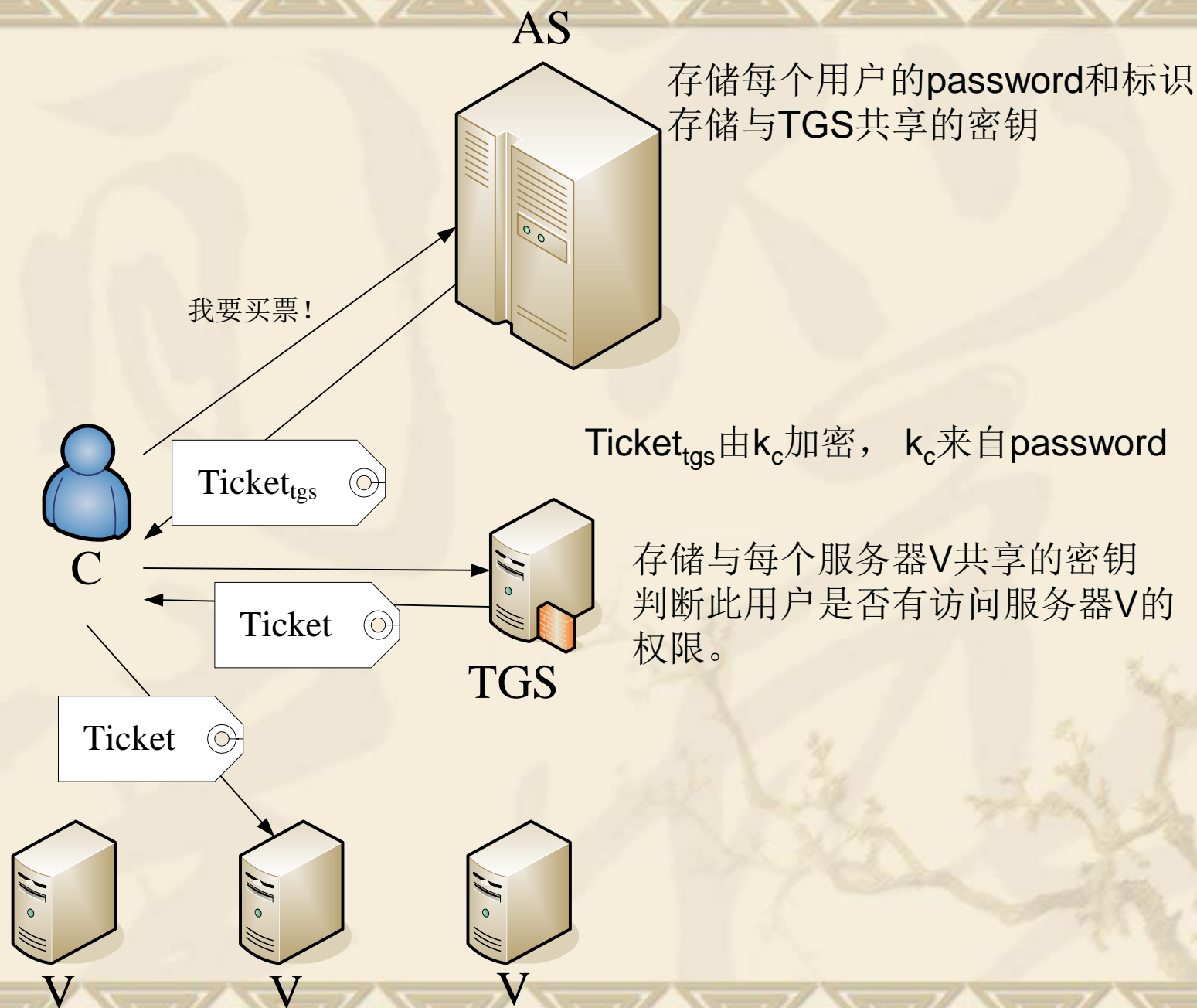
❖ Although the foregoing scenario solves some of the problems of authentication in an open network environment, problems remain. Two in particular stand out.

⌘ we would like to minimize the number of times that a user has to enter a password. Suppose each ticket can be used only once.

⌘ The second problem is that the earlier scenario involved a plaintext transmission of the password.

# A More Secure Authentication Dialogue

- ❖ 把身份认证和访问权限交给两个服务器分别完成
  - ∞ 身份认证，由AS完成
  - ∞ 访问控制，由票据授权服务器（ticket-granting server, TGS)来完成
- ❖ 达到的效果：
  - ∞ 用户口令只需输入一次，且不会在网络上传输。



# Ticket-granting server(TGS)

- ❖ 票据授权服务器TGS专门负责产生客户端与服务器每次通信时所要使用的会话密钥（**session key**）。
- ❖ 每次的认证与会话密钥都会有一个有效期限。。

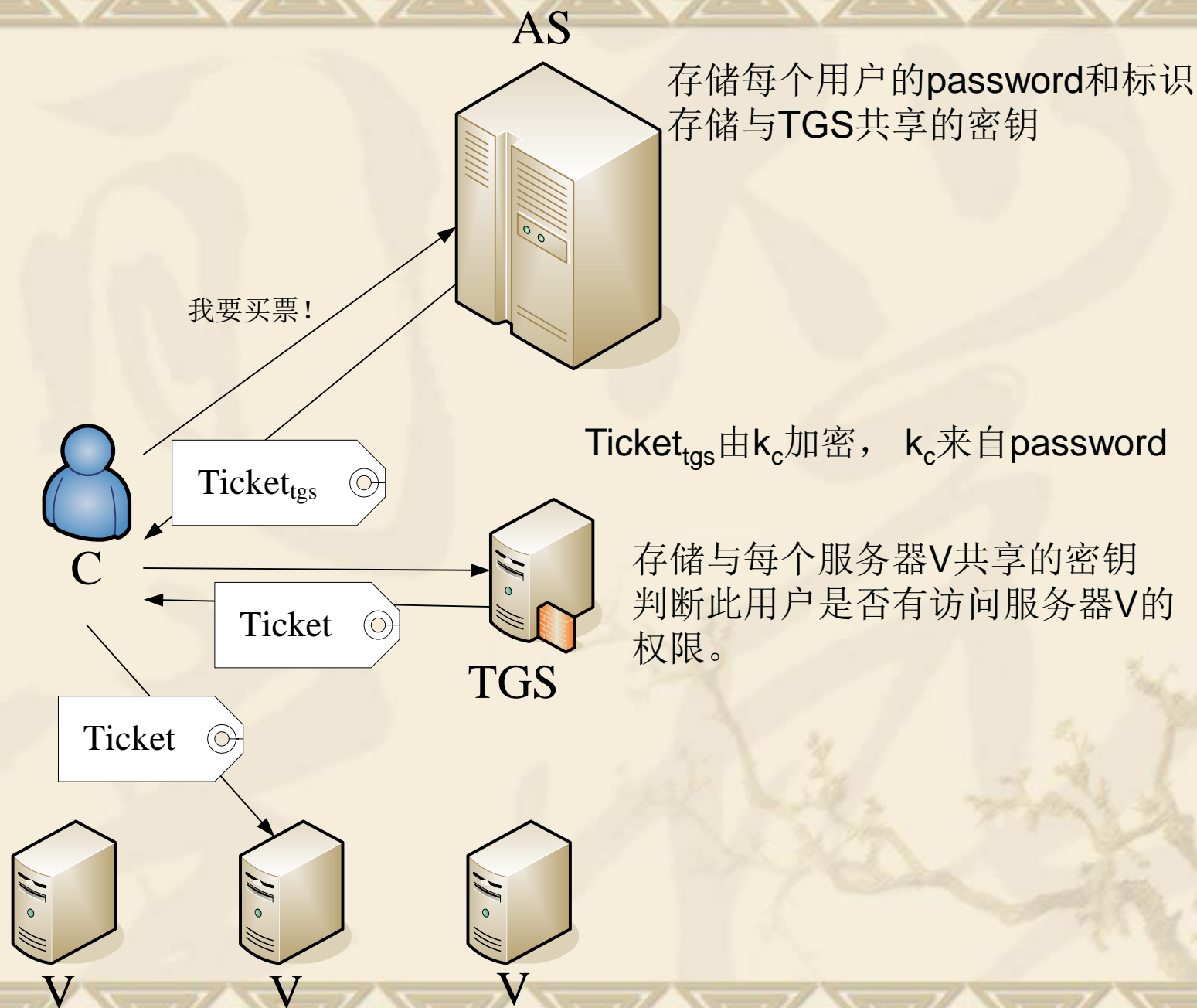


# Kerberos系统架构

- ❖ 在Kerberos系统中，当客户端要和服务器通信时，客户端会先向认证服务器AS请求一张与TGS沟通所需要的票据（通行证）。
- ❖ 该票据中有一个客户端与票据授权服务器进行通信的会话密钥 $K_{c,tgs}$ ，还包含了客户端和服务器的标识、时间戳及有效期限。
- ❖ 当客户端拿到了与TGS通信的票据后，客户端使用此票据再向TGS要一张与服务器连线所需要的票据，此票据中即包含客户端和服务器的会话密钥。

# Kerberos系统架构

- ❖ 客户端利用该会话密钥将它和服务端之间的通信内容加密以在网络上传送。
- ❖ 在与**TGS**通信所使用的票据过期之前，客户端若要再和别的服务器连线的话，就只要通过**TGS**获取服务器的票据即可，不必再通过**AS**。



### Once per user logon session:

- (1)  $C \rightarrow AS: ID_C || ID_{tgs}$
- (2)  $AS \rightarrow C: E(K_c, Ticket_{tgs})$

### Once per type of service:

- (3)  $C \rightarrow TGS: ID_C || ID_v || Ticket_{tgs}$
- (4)  $TGS \rightarrow C: Ticket_v$

### Once per service session:

- (5)  $C \rightarrow V: ID_C || Ticket_v$

$Ticket_{tgs} = E(K_{tgs}, [ID_C || AD_C || ID_{tgs} || TS_1 || Lifetime_1])$

$Ticket_v = E(K_v, [ID_C || AD_C || ID_v || TS_2 || Lifetime_2])$




(3)  $C \rightarrow TGS: ID_C || ID_V || Ticket_{tgs}$




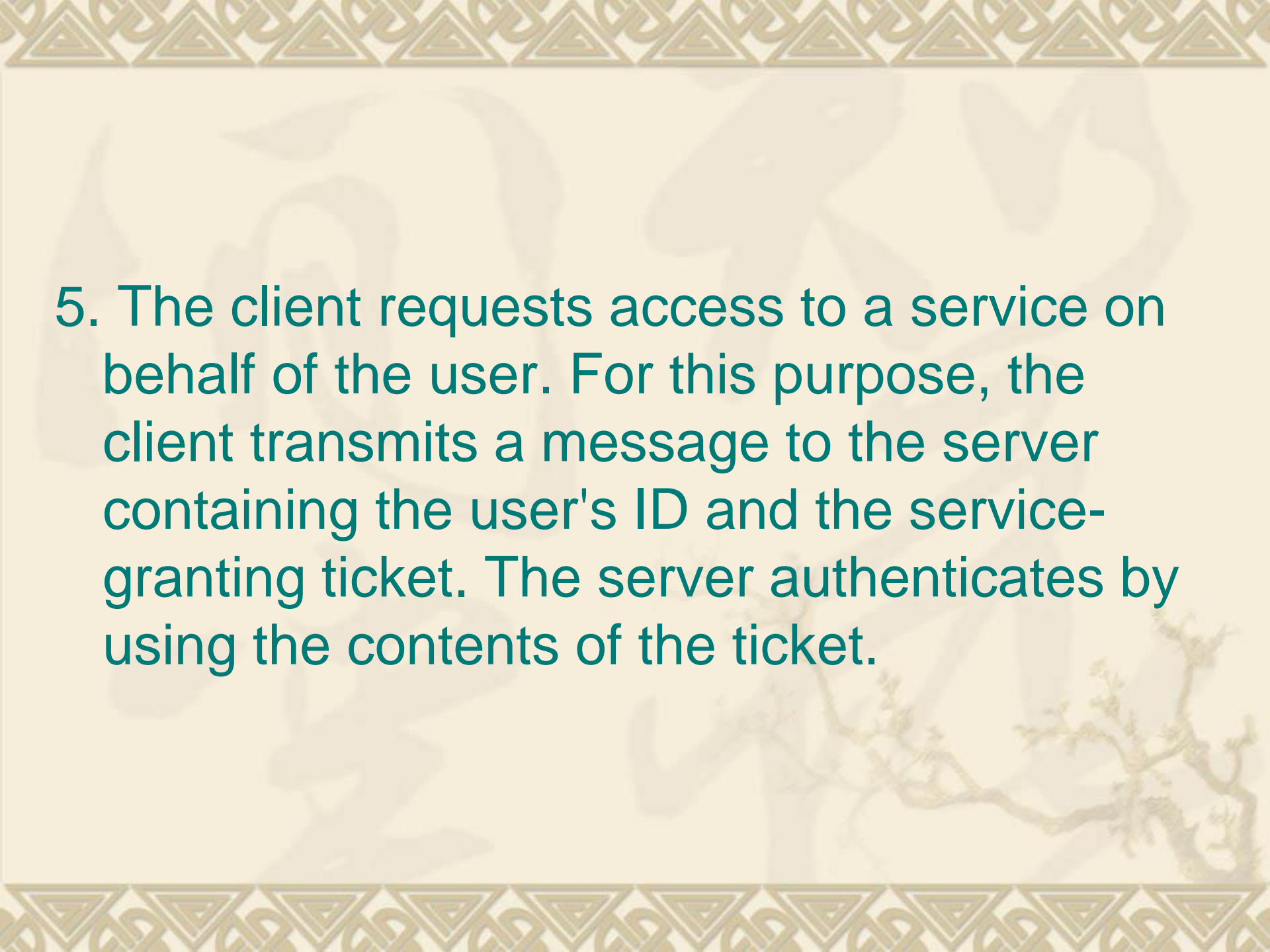
$Ticket_{tgs} = E(K_{tgs}, [ID_C || AD_C || ID_{tgs} || TS_1 || Lifetime_1])$

1. 客户端将用户标识、TGS标识一起送往AS，申请得到**票据授权票据ticket-granting ticket**。
  2. AS用从用户口令推出的密钥 $K_c$ (事先已经存储在AS中)将票据加密，并发送给客户端。由用户在客户端输入口令，并得到 $K_c$ ，将收到的消息解密，得到**票据授权票据ticket-granting ticket**。
- ❖ The client module in the user workstation saves this **ticket-granting ticket**.
  - ❖ Because only the correct user should know the password, only the correct user can recover the ticket. Thus, we have used the password to obtain credentials from Kerberos without having to transmit the password in plaintext.

3. The client requests a **service-granting ticket**(服务授权票据). For this purpose, the client transmits a message to the TGS containing the user's ID, **the ID of the desired service**, and the ticket-granting ticket.
4. The TGS decrypts the incoming ticket and verifies the success of the decryption by the presence of its ID. It checks to make sure that the lifetime has not expired. Then it compares the user ID and network address with the incoming information to authenticate the user. **If the user is permitted access to the server V**, the TGS issues a ticket(**service-granting ticket**,  $\text{Ticket}_v$ ) to grant access to the requested service.



5. The client requests access to a service on behalf of the user. For this purpose, the client transmits a message to the server containing the user's ID and the service-granting ticket. The server authenticates by using the contents of the ticket.





# 使用的密钥

- ❖  $K_c$ : AS和C共享
- ❖  $K_{tgs}$ : AS和TGS共享
- ❖  $K_v$ : TGS和V共享

# Two problems

- ❖ The lifetime associated with the ticket-granting ticket:
  - ⌘ If this lifetime is very short (e.g., minutes), then the user will be repeatedly asked for a password.
  - ⌘ If an opponent captures a service-granting ticket and uses it before it expires, the opponent has access to the corresponding service.
  - ⌘ A network service (the TGS or an application service) must be able to prove that the person using a ticket is the same person to whom that ticket was issued.
- ❖ The second problem is that there may be a requirement for servers to authenticate themselves to users.

# The Kerberos Version 4 Authentication Dialogue(1)

- ❖ Authentication Service Exchange to obtain ticket-granting ticket.

(1)  $C \rightarrow AS: ID_c || ID_{tgs} || TS_1$

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

Where  $Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} || ID_c || AD_c || ID_{tgs} || TS_2 || Lifetime_2])$

$K_{c,tgs}$  : Copy of session key accessible to client created by AS to permit secure exchange between client and TGS without requiring them to share a permanent key.

## The Kerberos Version 4 Authentication Dialogue(2)

- ❖ Ticket-Granting Service Exchange to obtain service-granting ticket.

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

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

$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} || ID_C || AD_C || ID_{tgs} || TS_2 || Lifetime_2])$

$Ticket_v = E(K_v, [K_{c,v} || ID_C || AD_C || ID_v || TS_4 || Lifetime_4])$

$Authenticator_c = E(K_{c,tgs}, [ID_C || AD_C || TS_3])$



## The Kerberos Version 4 Authentication Dialogue(3)

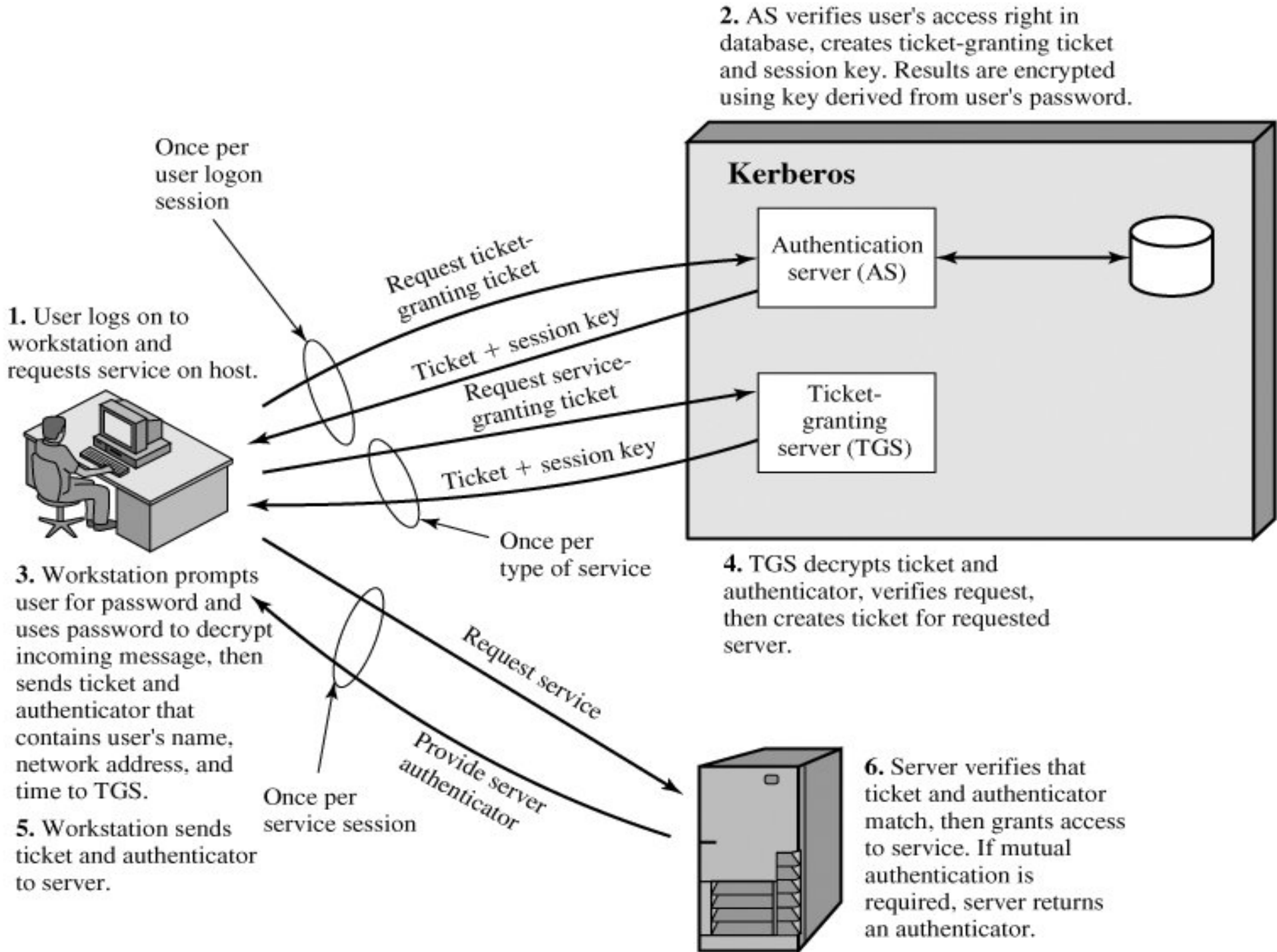
- ❖ Client/Server Authentication Exchange to obtain service.

(5)  $C \rightarrow V$ :  $\text{Ticket}_v \parallel \text{Authenticator}_c$

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

$\text{Ticket}_v = E(K_v, [K_{c,v} \parallel \text{ID}_c \parallel \text{AD}_c \parallel \text{ID}_v \parallel TS_4 \parallel \text{Lifetime}_4])$

$\text{Authenticator}_c = E(K_{c,v}, [\text{ID}_c \parallel \text{AD}_c \parallel TS_5])$



## 6.1.4 零知识证明

- ❖ **Alice:** “我知道联邦储备系统计算的口令”
- ❖ **Bob:** “不，你不知道”
- ❖ **Alice:** 我知道
- ❖ **Bob:** 你不知道
- ❖ **Alice:** 我确实知道
- ❖ **Bob:** 请你的证实这一点
- ❖ **Alice:** 好吧，我告诉你。（她悄悄说出了口令）
- ❖ **Bob:** 太有趣了！现在我也知道了。我要告诉《华盛顿邮报》
- ❖ **Alice:** 啊呀！



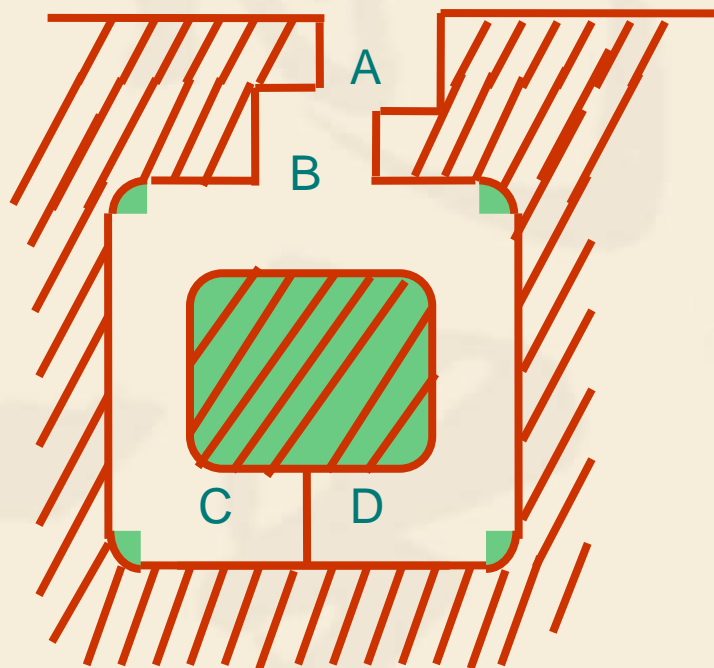
# 零知识证明技术

- ❖ 零知识证明技术可使信息的拥有者无需泄露任何信息就能够向验证者或任何第三方证明它拥有该信息。



# 零知识证明的基本协议

例[Quisquater等1989]。



设P知道咒语，可  
打开C和D之间的秘  
密门，不知道者  
都将走向死胡同中。

# 零知识证明的基本协议

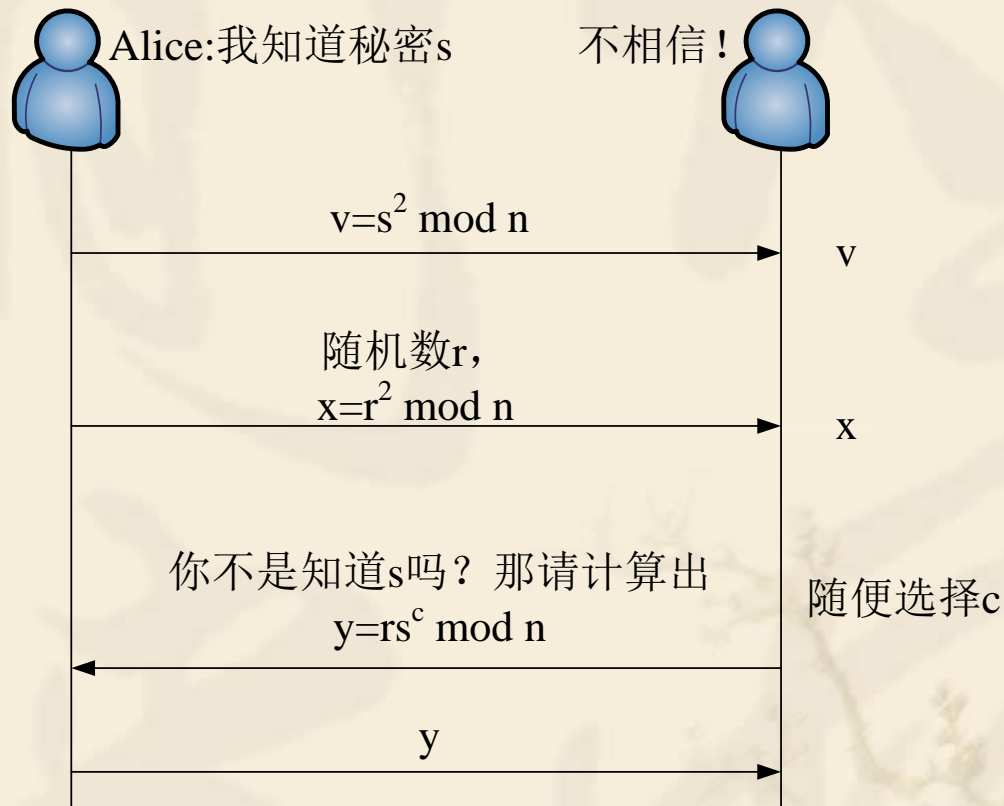
- (1) V站在A点;
- (2) P进入洞中任一点C或D;
- (3) 当P进洞之后, V走到B点;
- (4) V叫P: (a)从左边出来, 或(b)从右边出来;
- (5) P按要求实现(以咒语, 即解数学难题帮助);
- (6) P和V重复执行(1)~(5)共 $n$ 次。

若A不知咒语, 则在B点, 只有50 %的机会猜中B的要求, 协议执行 $n$ 次, 则只有 $2^{-n}$ 的机会完全猜中, 若 $n=16$ , 则若每次均通过B的检验, B受骗机会仅为 $1/65\,536$

# 最简单的零知识证明

- ❖ 问题要求：假如P想说服V，使V相信他确实知道n的因数p和q，但不能告诉V
- ❖ 最简单的步骤：
  - ↻ V随机选择一整数x，计算 $x^4 \bmod n$  的值，并告诉P
  - ↻ P求 $x^2 \bmod n$  并将它告诉V
  - ↻ V验证 $x^4 \bmod n$
- ❖ V知道求 $x^2 \bmod n$  等价于n的因数分解，若不掌握n的因数p和q，求解很困难。

# Fiat-Shamir协议



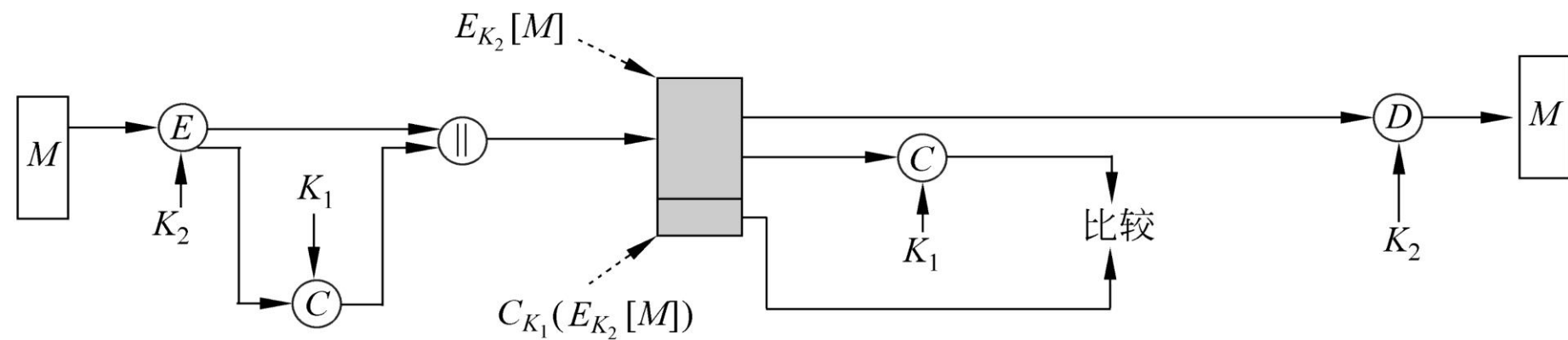
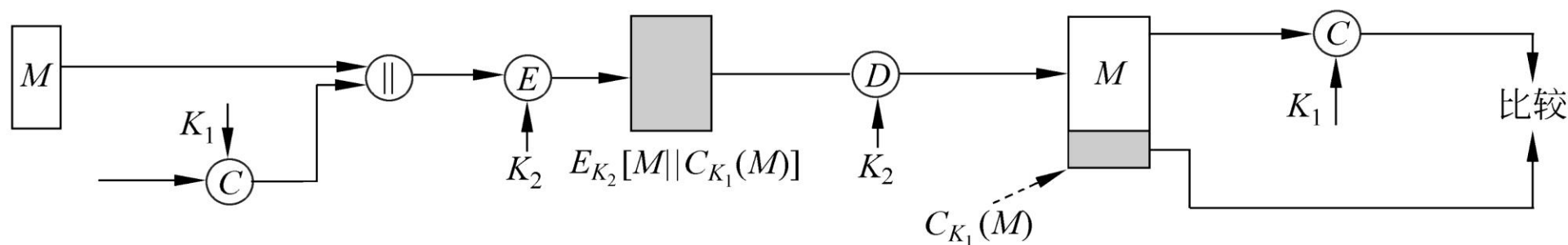
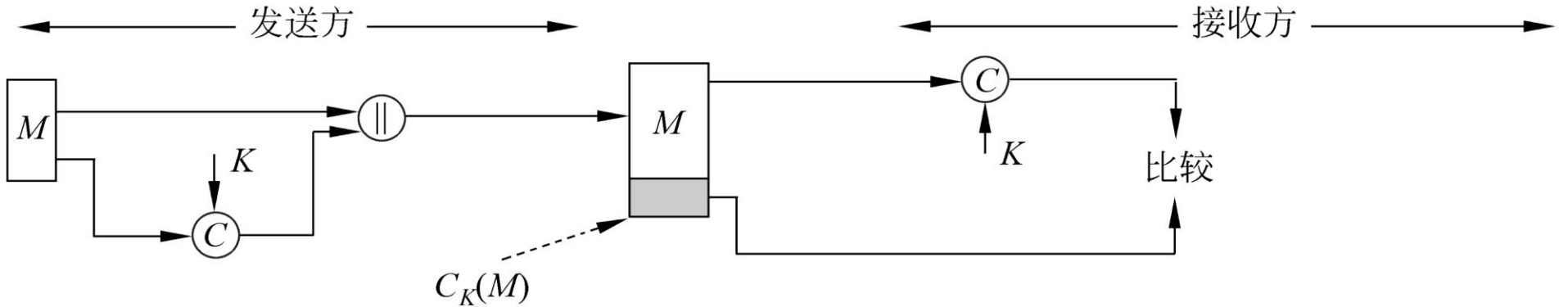
$$xv^c \equiv r^2(s^2)^c \equiv r^2s^{2c} \equiv (rs^c)^2 \equiv y^2 \bmod n$$



## 6.2 数字签名

# 消息认证无法保证抗否认性

- ❖ 可以保护通信双方以防止第3者攻击，不能保护通信双方中一方防止另一方的欺骗和伪造。
  - ❧ B伪造一个消息并使用与A共享的密钥产生该消息的认证码，然后声称该消息来自于A
  - ❧ B有可能伪造A发来的消息，所以A就可以对自己发过的消息予以否认



# 数字签名的基本概念

- ❖ 数字签名由公钥密码发展而来，它在网络安全，包括身份认证、数据完整性、不可否认性以及匿名性等方面有着重要应用。



# 电子签名法

- ❖ 2004年8月28日第十届全国人民代表大会常务委员会第十一次会议通过
- ❖ 第二条 本法所称电子签名，是指数据电文中以电子形式所含、所附用于识别签名人身份并表明签名人认可其中内容的数据。
- ❖ 第十四条 可靠的电子签名与手写签名或者盖章具有同等的法律效力。
- ❖ 第三十二条 伪造、冒用、盗用他人的电子签名，构成犯罪的，依法追究刑事责任；给他人造成损失的，依法承担民事责任。
- ❖ 第三十六条 本法自2005年4月1日起施行。

# 手写签名的特征

- ❖ 签名是可信的
- ❖ 签名是不可伪造的
- ❖ 签名不可重用
- ❖ 签名后的文件是不可变的
- ❖ 签名是不可抵赖的

# 数字签名的功能

- ❖ In situations where there is not complete trust between sender and receiver, something more than authentication is needed. The most attractive solution to this problem is the digital signature.
- ❖ Digital signature must have the following properties:
  - ❧ It must verify the author and the date and time of the signature.
  - ❧ It must to authenticate the contents at the time of the signature.
  - ❧ It must be verifiable by third parties, to resolve disputes.

# 更进一步的要求

- ❖ 依赖性：签名必须是依赖于被签名信息来产生；
- ❖ 唯一性：签名必须使用某些对发送者是唯一的信息，以防止双方的伪造与否认；
- ❖ 可验性：必须相对容易识别和验证该数字签名；
- ❖ 抗伪造：根据一个已有的数字签名来构造消息是不可行的；对一个给定消息伪造数字签名是不可行的；
- ❖ 可用性：在存储器中保存一个数字签名副本是现实可行的。

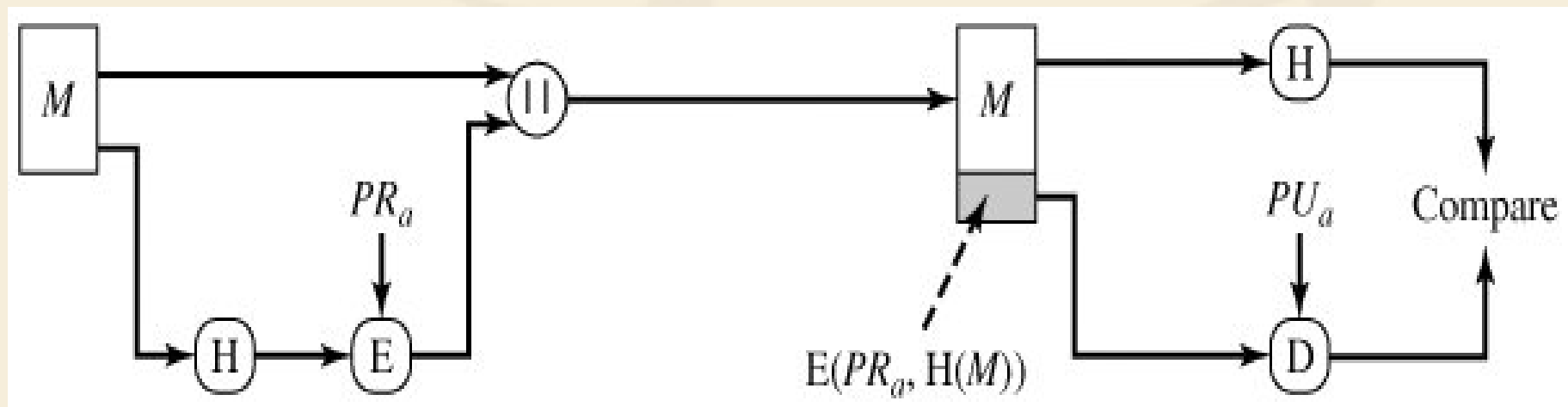
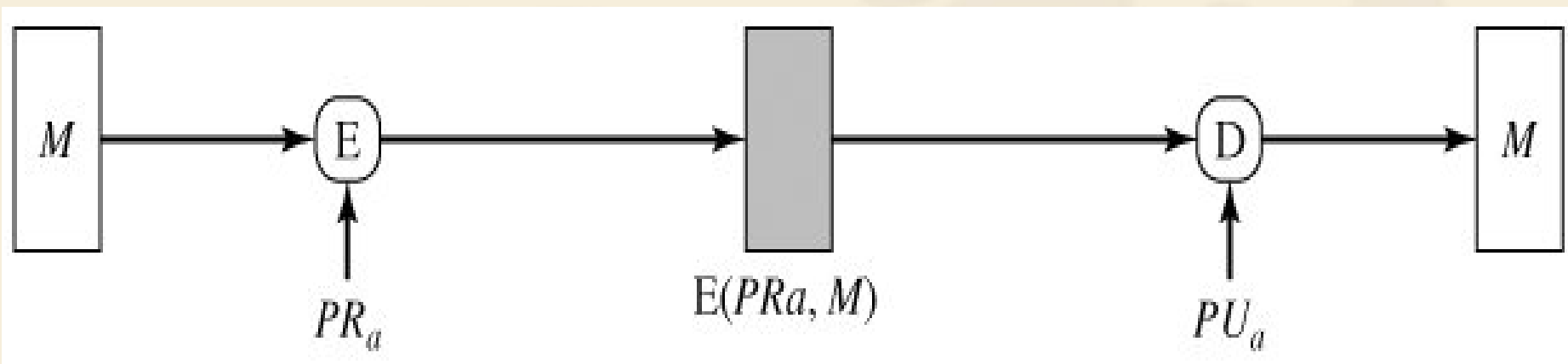


# 签名方法

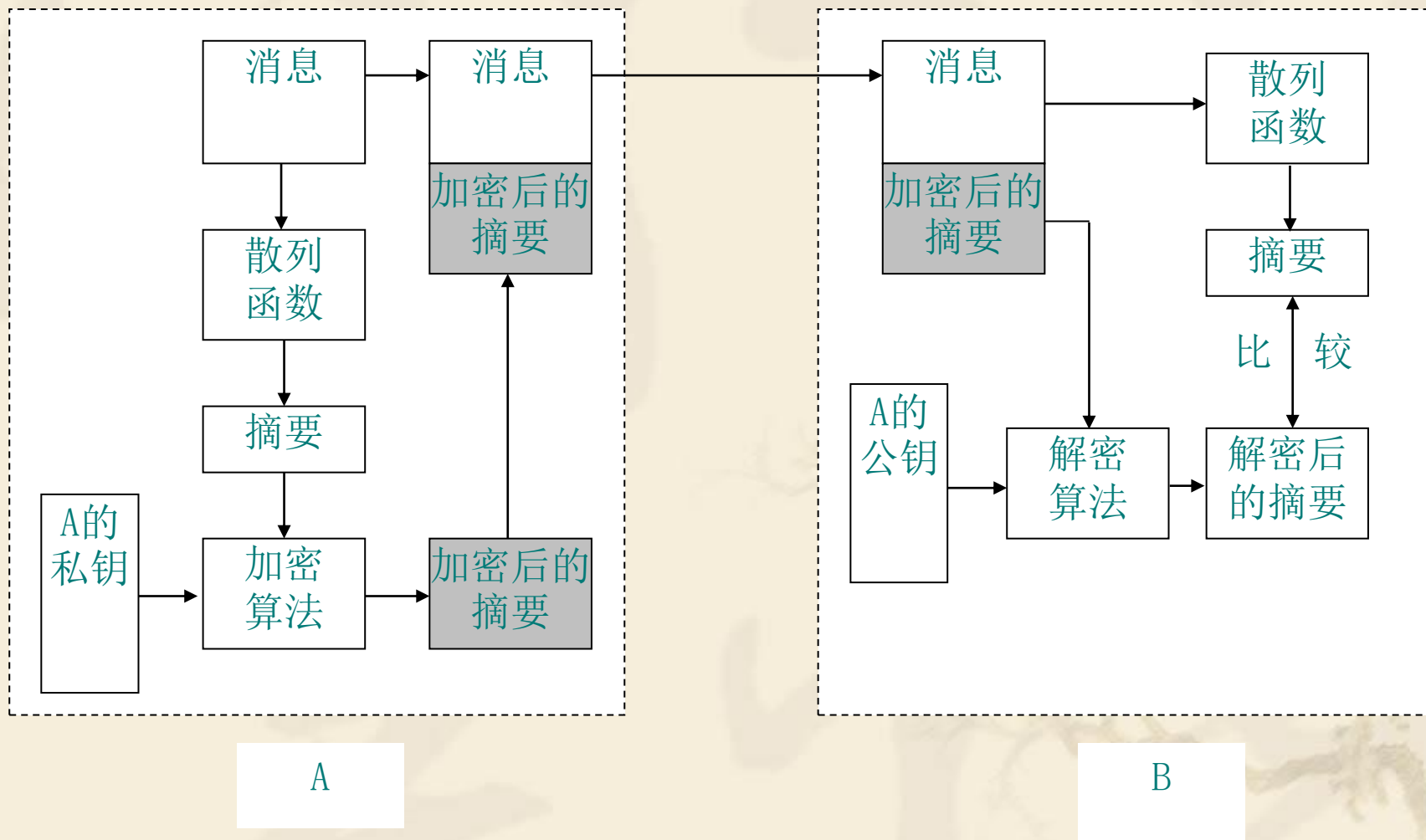
- ❖ **Direct Digital Signature**直接数字签名
- ❖ **Arbitrated Digital Signature**仲裁数字签名

## ❖ Direct Digital Signature 直接方式

- ❧ The direct digital signature involves only the communicating parties (source, destination).
- ❧ It is assumed that the destination knows the public key of the source.
- ❧ A digital signature may be formed by encrypting the entire message with the sender's private key or by encrypting a hash code of the message with the sender's private key.

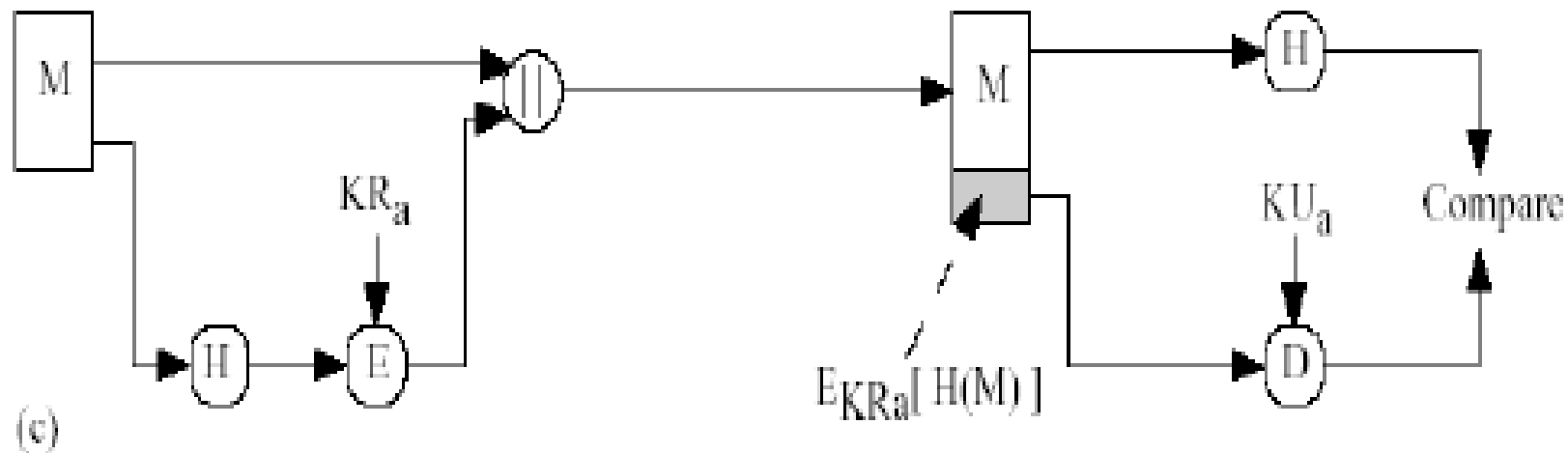


# 直接数字签名





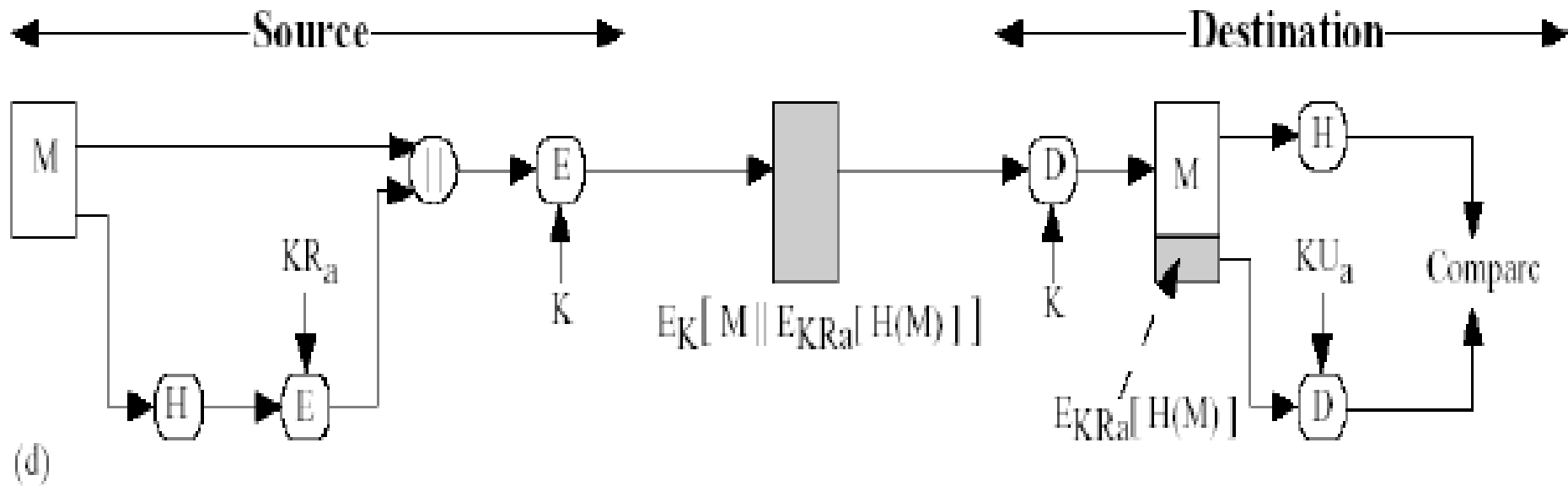
# 散列函数的基本用法 (c)



**Provides authentication and digital signature**

- $H(M)$  is cryptographically protected
- only A could create  $E_{KR_a}[H(M)]$

# 散列函数的基本用法(d)



(d)  $A \rightarrow B: E_K[M \parallel E_{KR_a}[H(M)]]$

Provides authentication and digital signature  
Provides confidentiality

## ❖ All **Direct Digital Signature** described so far share a common weakness:

- ❧ The validity of the scheme depends on the security of the sender's private key.
- ❧ If a sender later wishes to deny sending a particular message, the sender can claim that the private key was lost or stolen and that someone else forged his or her signature.
- ❧ Another threat is that some private key might actually be stolen from X at time T. The opponent can then send a message signed with X's signature and stamped with a time before or equal to T.

# 仲裁数字签名

## ❖ 具有仲裁方式的数字签名

1. 发方X对发往收方Y的消息签名
2. 将消息和签名先发往仲裁者A
3. A对消息和签名验证完后，再连同同一个表示已通过验证的指令一起发给Y.



# 具有仲裁方式的数字签名 ----基于对称密码

## ❖ 例1:

1.  $X \rightarrow A: M || E_{K_{XA}}[ID_X || H(M)]$
2.  $A \rightarrow Y: E_{K_{AY}}[ID_X || M || E_{K_{XA}}[ID_X || H(M)] || T]$

E: 单钥加密算法

$K_{XA}, K_{AY}$ : A与X和Y的共享密钥

M: 消息

T: 时戳

$ID_X$ : X的身份

$H(M)$ : M的杂凑值

此方案不提供对M的保密性

- ❖ 在1中，X以 $E_{K_{XA}}[ID_X \parallel H(M)]$ 作为自己对M的签名，将M及签名发往A。
- ❖ 在2中A将从X收到的内容和 $ID_X$ 、T一起加密后发往Y，其中的T用于向Y表示所发的消息不是旧消息的重放。Y对收到的内容解密后，将解密结果存储起来以备出现争议时使用。
- ❖ 如果出现争议，Y可声称自己收到的M的确来自X，并将 $E_{K_{AY}}[ID_X \parallel M \parallel E_{K_{XA}}[ID_X \parallel H(M)]]$ 发给A，由A仲裁，A由 $K_{AY}$ 解密后，再用 $K_{XA}$ 对 $E_{K_{XA}}[ID_X \parallel H(M)]$ 解密，并对 $H(M)$ 加以验证，从而验证了X的签名。

# 具有仲裁方式的数字签名

## ❖ 例2

X对M的签名

1.  $X \rightarrow A: ID_X \parallel E_{K_{XY}}[M] \parallel E_{K_{XA}}[ID_X \parallel H(E_{K_{XY}}(M))]$

2.  $A \rightarrow Y: E_{K_{AY}}[ID_X \parallel E_{K_{XY}}[M] \parallel E_{K_{XA}}[ID_X \parallel H(E_{K_{XY}}(M))]] \parallel T]$

X和Y的共享密钥

此方案提供了对M的保密性

仲裁者可和发方共谋否认发方曾发过的  
消息，也可和收方共谋产生发方的签名



# 具有仲裁方式的数字签名 ----基于公钥密码

## ❖ 例3



$$1. X \rightarrow A : ID_X \parallel E_{SK_X} [ID_X \parallel E_{PK_Y} [E_{SK_X} [M]]]$$

$$2. A \rightarrow Y : E_{SK_A} [ID_X \parallel E_{PK_Y} [E_{SK_X} [M]] \parallel T]$$

- ❖ 第1步中，X用自己的秘密钥 $SK_X$ 和Y的公开钥 $PK_Y$ 对消息加密后作为对M的签名，以这种方式使得任何第3方（包括A）都不能得到M的明文消息。
- ❖ A收到X发来的内容后，用X的公开钥可对 $E_{SK_X}[ID_X \parallel E_{PK_Y}[E_{SK_X}[M]]]$ 解密，并将解密得到的 $ID_X$ 与收到的 $ID_X$ 加以比较，从而可确信这一消息是来自于X的（因只有X有 $SK_X$ ）。
- ❖ 第2步，A将X的身份 $ID_X$ 和X对M的签名加上一时戳后，再用自己的秘密钥加密发往Y。

❖ 与前两种方案相比，第3种方案有很多优点：

- ❧ 在协议执行以前，各方都不必有共享的信息，从而可防止共谋。
- ❧ 只要仲裁者的秘密钥不被泄露，任何人包括发方就不能发送重放的消息。
- ❧ 对任何第三方（包括A）来说，X发往Y的消息都是保密的。

# RSA签名体制

## ❖ 体制参数

∞ 大素数 $p, q$ ,  $n = p \times q$ ,  $\varphi(n) = (p-1)(q-1)$ 。选整数 $1 < e < \varphi(n)$ , 且 $\gcd(e, \varphi(n)) = 1$ ; 计算 $d$ 满足 $de \equiv 1 \pmod{\varphi(n)}$ .  $\{e, n\}$ 为公开密钥,  $\{d, n\}$ 为秘密密钥。

## ❖ 签名过程

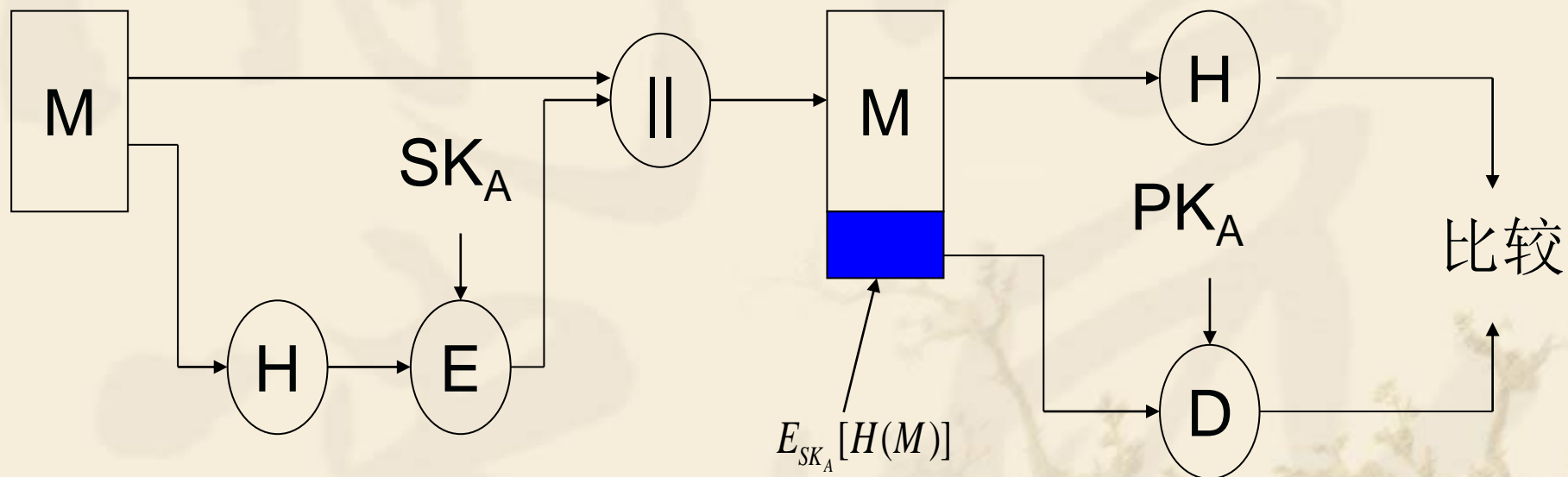
∞  $S = M^d \pmod{n}$

## ❖ 验证过程

∞  $M = S^e \pmod{n}$

❖ 实际应用中加密是对 $H(M)$ 进行的。

# RSA签名方案







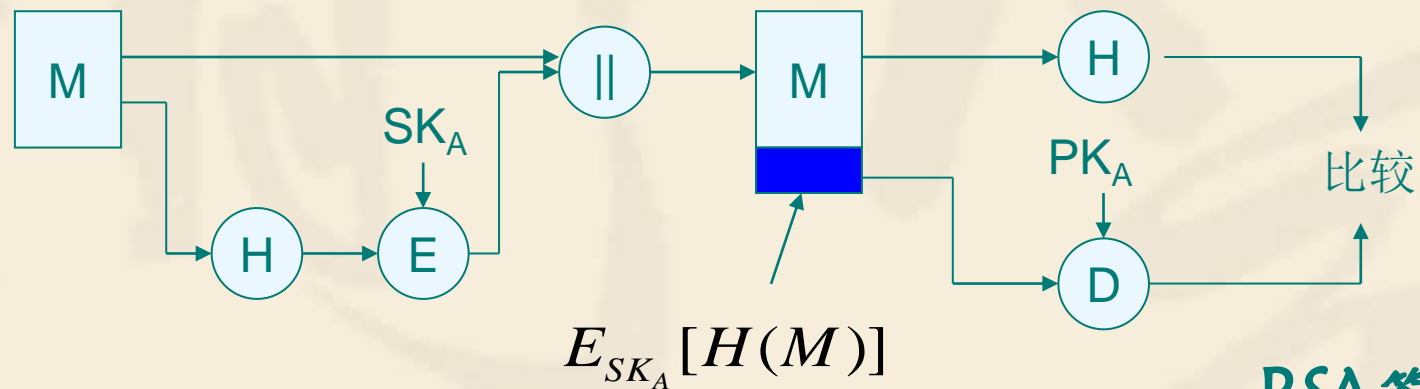
# 数字签名标准

Digital Signature Standard (DSS)

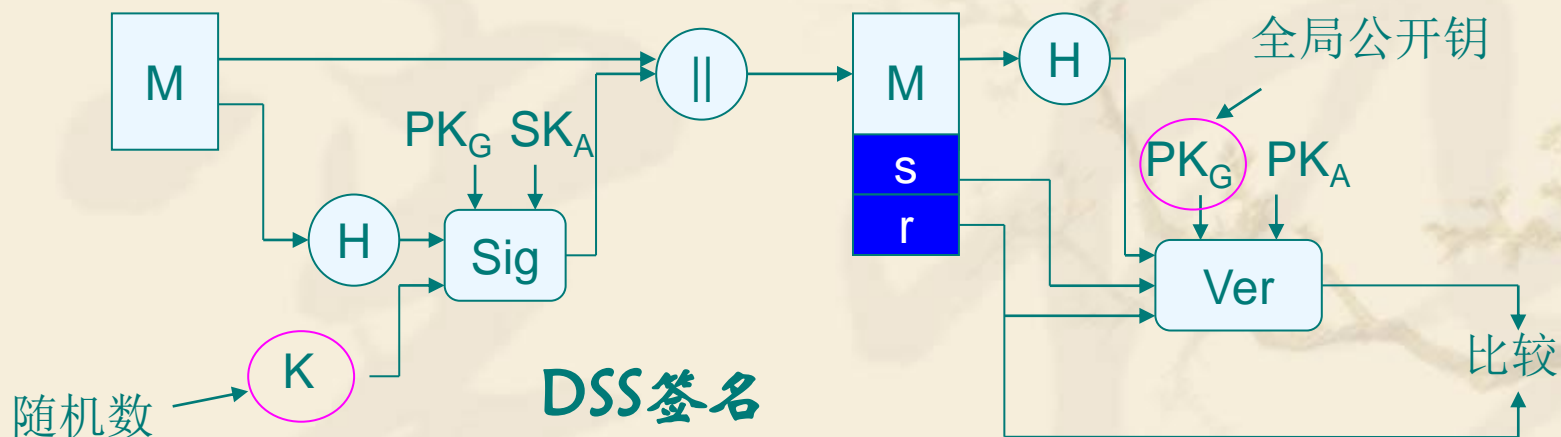
# 概况

- ❖ 由NIST1991年公布
- ❖ 1993年公布修改版
- ❖ 美国联邦信息处理标准FIPS PUB 186
- ❖ 签名长度320bit
- ❖ 只能用于数字签名，不能用于加密

# DSS的基本方式



**RSA签名**



**DSS签名**

# 数字签名算法DSA

## ❖ 算法描述:

(a) 全局公钥  $(p, q, g)$

$p$ : 是  $2^{L-1} < p < 2^L$  中的大素数,  $512 \leq L \leq 1024$ , 按64 bits 递增;

$q$ :  $(p-1)$  的素因子, 且  $2^{159} < q < 2^{160}$ , 即字长160 bits;

$g := h^{p-1} \bmod p$ , 且  $1 < h < (p-1)$ , 使  $h^{(p-1)/q} \bmod p > 1$ 。

(b) 用户私钥  $x$ :  $x$  为在  $0 < x < q$  内的随机数。

(c) 用户公钥  $y$ :  $y = g^x \bmod p$ 。

(d) 用户每个消息用的秘密随机数  $k$ : 在  $0 < k < q$  内的随机数



# 数字签名算法DSA

(e) 签名过程：对消息  $M \in M = Z_p^*$ ，其签名为  $S = \text{Sig}_k(M, k) = (r, s)$ ， $S \in S = Z_q \times Z_q$ ，

$$r \equiv (g^k \bmod p) \bmod q$$

$$s \equiv [k^{-1} (h(M) + xr)] \bmod q$$

(f) 验证过程：计算

$$w = s^{-1} \bmod q; \quad u_1 = [H(M)w] \bmod q;$$

$$u_2 = rw \bmod q; \quad v = [(g^{u_1} y^{u_2}) \bmod p] \bmod q.$$

$$\text{Ver}(M, r, s) = \text{真} \Leftrightarrow v = r$$

$$v \equiv [(g^{H(M)w} g^{xrw}) \bmod p] \bmod q$$

$$\equiv [g^{[H(M) + xr]s^{-1}} \bmod p] \bmod q$$

$$\equiv (g^k \bmod p) \bmod q \equiv r$$



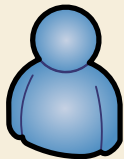
# 不可否认签名

- ❖ 在得不到签名者配合的情况下其他人不能正确进行签名验证，从而可以防止非法复制和扩散签名者所签署的文件。
- ❖ 可用于保护知识产权

$p, q$  是两个素数,  $\alpha$  和  $\beta$  公开

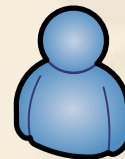
$a$  是签名者的私钥, 公钥是  $\beta = \alpha^a \bmod p$

签名:  $S = M^a \bmod p$



$$b = a^{-1} \bmod q$$

$$d = c^b \bmod p$$



任选  $e_1$  和  $e_2$ , 计算

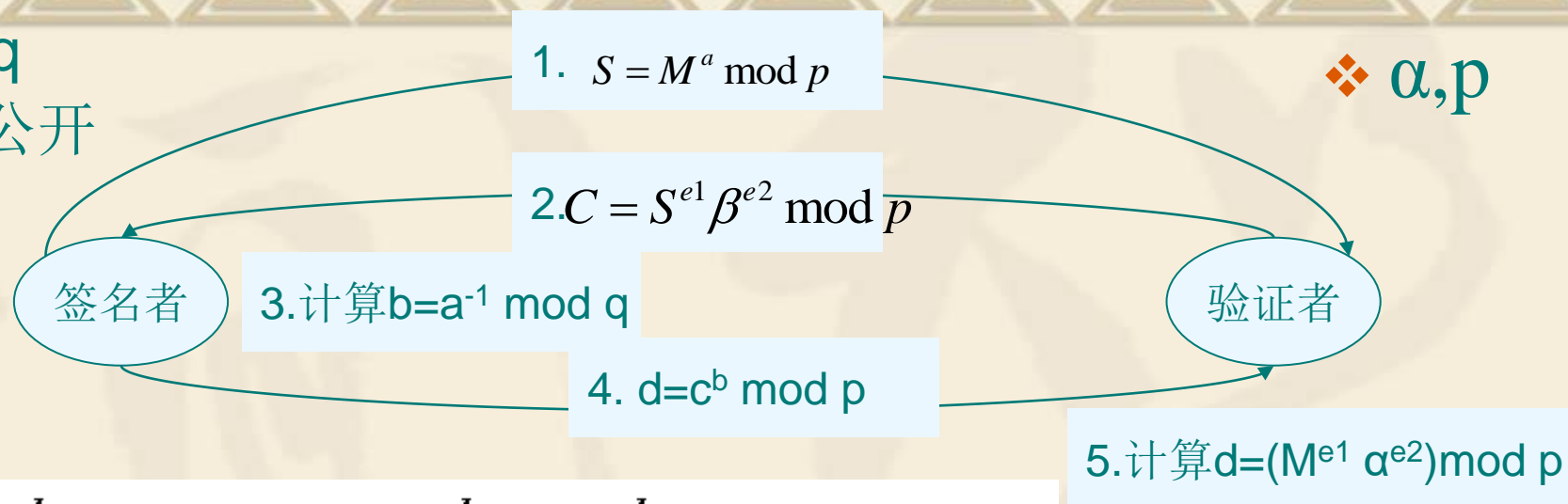
$$C = S^{e_1} \beta^{e_2} \bmod p$$

$$d ? = M^{e_1} \alpha^{e_2} \bmod p$$

❖  $a$ : 签名者的密钥

❖  $p, q$

❖  $\alpha, \beta$  公开



$$d = c^b \bmod p = (S^{e_1})^b (\beta^{e_2})^b \bmod p$$

$$\because \beta = \alpha^a \bmod p, b = a^{-1} \bmod q$$

$$\therefore \beta^b = (\alpha^a \bmod p)^{a^{-1}} = \alpha \bmod p$$

$$\because S = M^a \bmod p, b = a^{-1} \bmod q$$

$$\therefore S^b = (M^a \bmod p)^{a^{-1}} = M \bmod p$$

$$\therefore d = (S^{e_1})^b (\beta^{e_2})^b \bmod p = M^{e_1} \alpha^{e_2} \bmod p$$

# 盲签名

- ❖ Chaum在1983年提出。
- ❖ 需要某人对文件签名，但又不想签名者知道文件内容，称为盲签名。
- ❖ 适应于电子选举、数字货币协议中

# 基于RSA的盲签名

- ❖  $e$  是用户  $B$  的公钥， $d$  是用户  $B$  的私钥
- ❖ 现在用户  $A$  需要得到用户  $B$  对消息  $M$  的签名

$$S = M^d \bmod n \quad \text{但不能让 } B \text{ 得知 } M$$

- ❖ 用户  $A$  随机选择  $k$ ，并计算： $T = Mk^e \bmod n$
- ❖ 把  $T$  发给  $B$
- ❖  $B$  对  $T$  签名： $W = T^d \bmod n$  然后发给  $A$
- ❖  $W/k \equiv T^d/k \equiv (Mk^e)^d/k \equiv M^d k^{ed}/k \equiv M^d \bmod n$