

# 信息安全概论

身份认证

刘亚维



认证协议

**Authentication Protocols**

# 认证协议

- 安全可靠的通信除需进行消息认证外，还需建立一些规范的协议对数据来源的可靠性、通信实体的真实性加以认证，以防止欺骗、伪装等攻击，一般分为三个层次：
  - 数据完整性认证
  - 数据源认证：使得通信业务与具体实体捆绑认证，数据由哪个实体而来
  - 实体认证：发起通信或访问的实体是否具有合法身份和相应权限
- 这些认证常常一起进行，也有时单独进行
- 实体认证(身份认证)包括身份证实和身份识别：

# 身份证实

- 你是否是你宣称的你
- 验证方首先知道要验证的身份是谁，进一步证实来访或与之通信的人是否具有该身份
  - 一般用于A和B确定通信时所用，通常的网络认证协议都是身份证实
- 具体技术：输入个人信息，经公式或算法运算所得结果与卡中或数据库中存储信息经公式运算所得结果比较

# 身份识别

- 我是否知道你是谁
- 验证方不知道来访人是否为合法身份，没有比较确定的目标，只要满足某个条件就可判定身份的特点。验证者一般为权威机构
  - 一般在实体认证中需要，比如判断来访者是否是在逃犯，是否为密码开启者，是否为本公司员工。通常用指纹、虹膜技术
- 具体技术：输入个人信息，经过处理提取模版信息，试着在存储数据库中找出一个与之匹配的模版而后得出结论



- 本节以网络通信的一个基本问题的解决引出认证协议的基本意义，这一基本问题陈述如下：
  - A和B是网络的两个用户，他们想通过网络先建立安全的共享密钥再进行保密通信。A(B)如何确信自己正在和B(A)通信而不是和C通信呢？
  - 这种通信方式为双向通信，此时的认证称为相互认证
  - 类似地，对于单向通信来说，认证称为单向认证

# 相互认证

- A, B双方在建立共享密钥时需要考虑保密性和实时性。
- 保密性：会话密钥应以密文传送，因此双方应事先共享密钥或者使用公钥
- 实时性：防止重放
  - 序列号方法
  - 时戳
  - 询问-应答

# 序列号方法

- 对交换的每一条消息加上序列号，序列号正确才被接收
- 要求每个用户分别记录与其他每一用户交互的序列号，增加用户负担，因而很少使用



# 时戳法

- A收到消息中包含**时戳**，且A看来这一时戳充分接近自己的当前时刻，A才认为收到的消息是新的并接收
- 要求各方时间同步

## 询问—应答

- 用户A向B发出一个**一次性随机数**作为询问，如果收到B发来的应答消息也包含一正确的一次性随机数，A就认为消息是新的并接受之。

# 各种方法的比较

- 时戳法不适用于面向连接的应用过程
  - 要求不同的处理器之间时间同步，所用的协议必须是容错的以处理网络错误
  - 协议中任何一方时钟出现错误失去同步，则敌手攻击的可能性增加
  - 网络中存在延迟，不能期待保持精确同步，必须允许误差范围

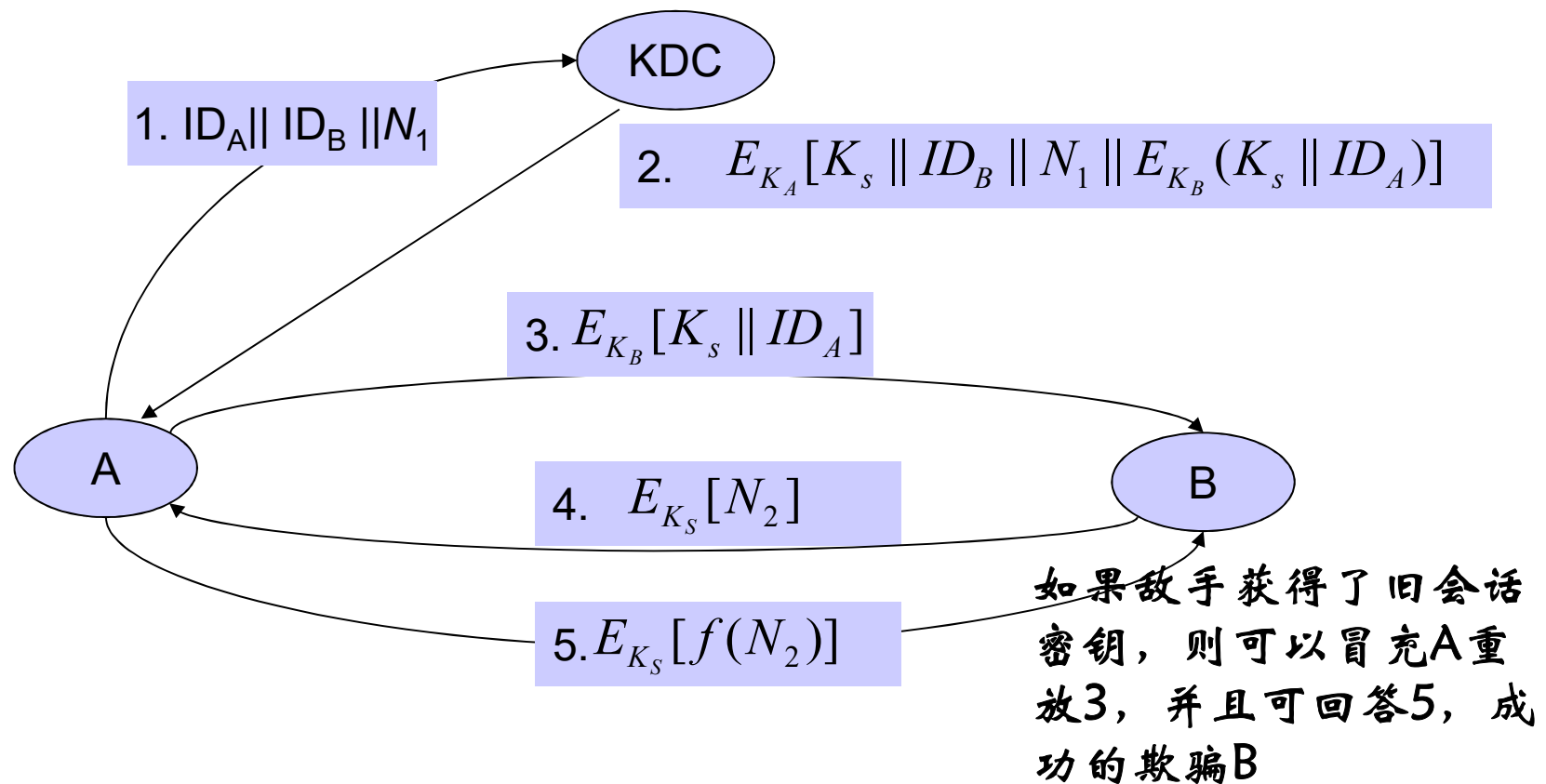
# 各种方法的比较

- 询问－应答不适合于无连接的应用过程
  - 在传输前需要经过询问－应答这一额外的握手过程，与无连接应用过程的本质特性不符。
  - 无连接应用最好使用安全时间服务器提供同步

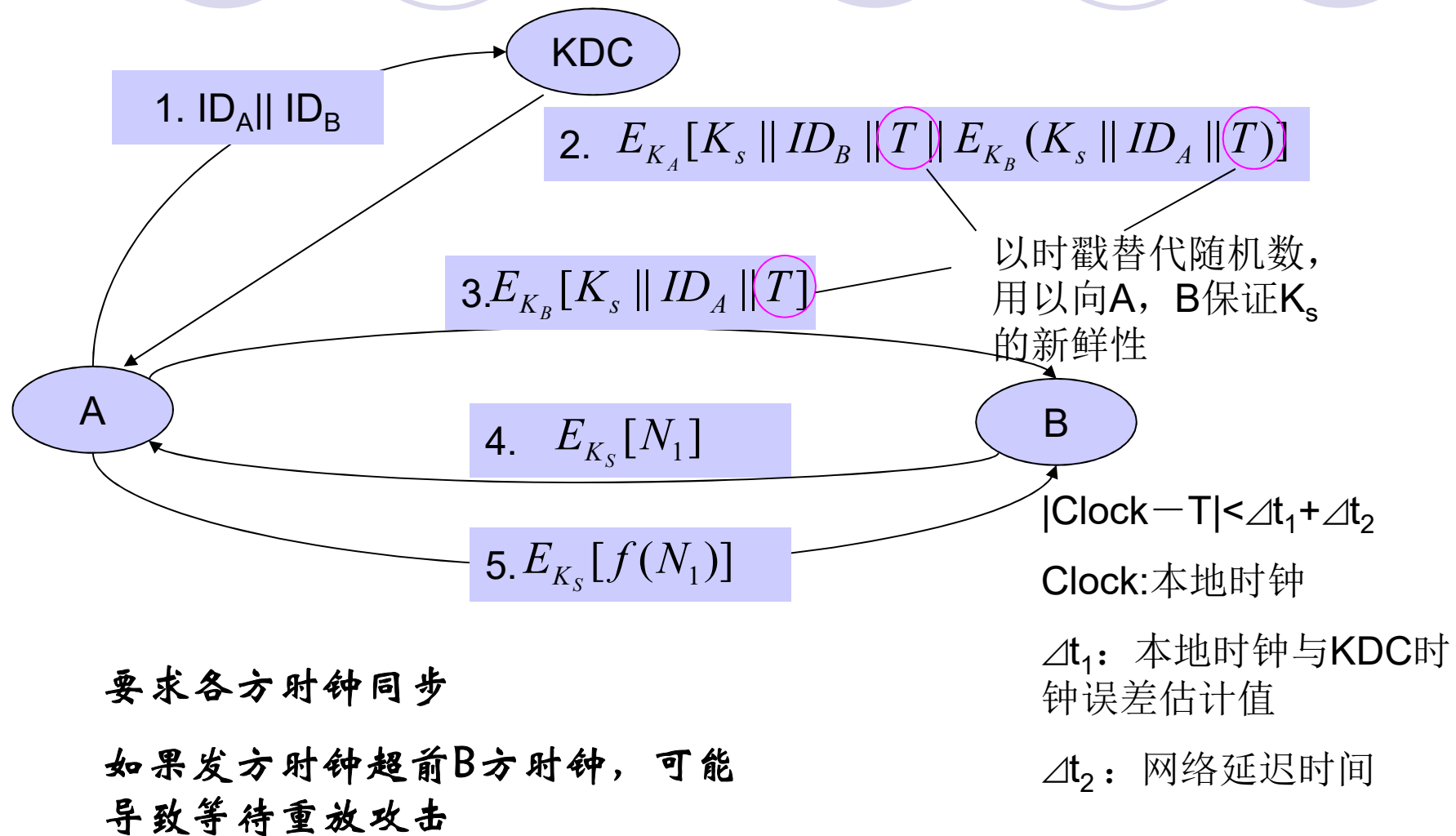


# 基于对称加密的远程用户认证

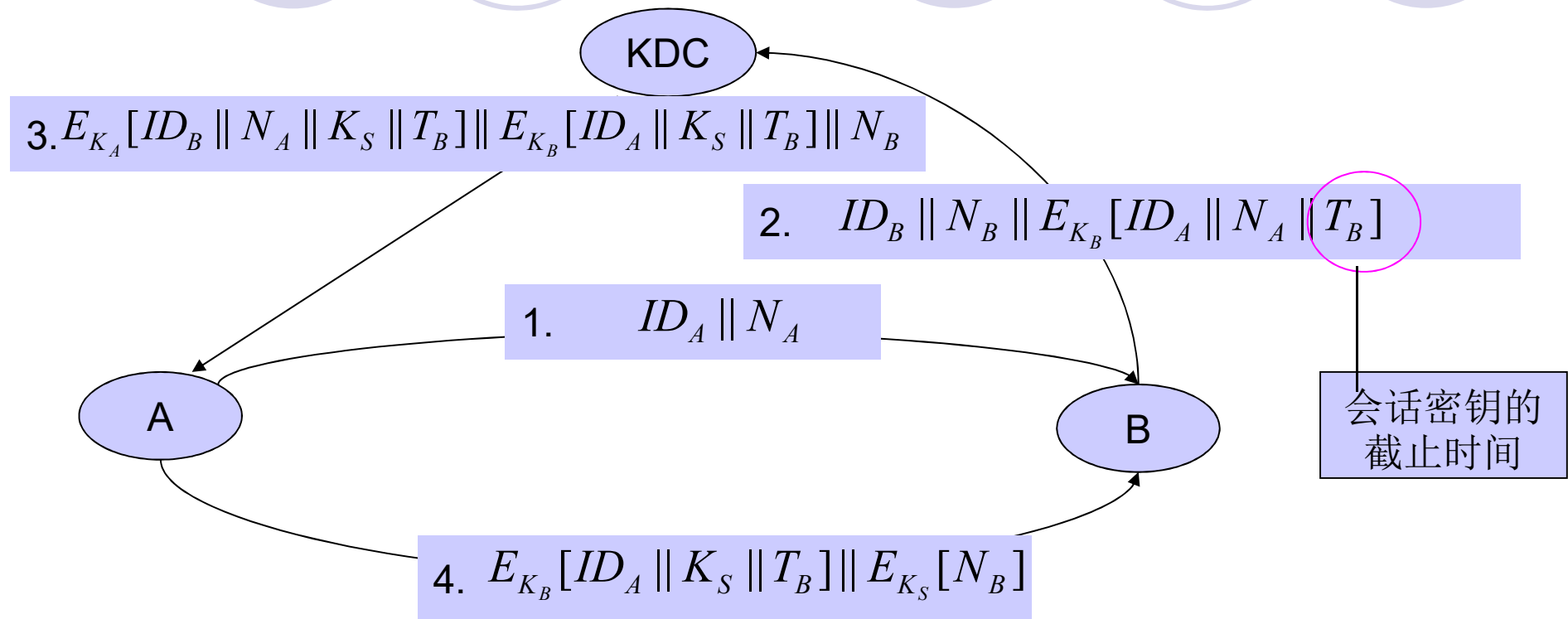
# Needham—Schroeder协议



# Needham—Schroeder改进协议 (1)



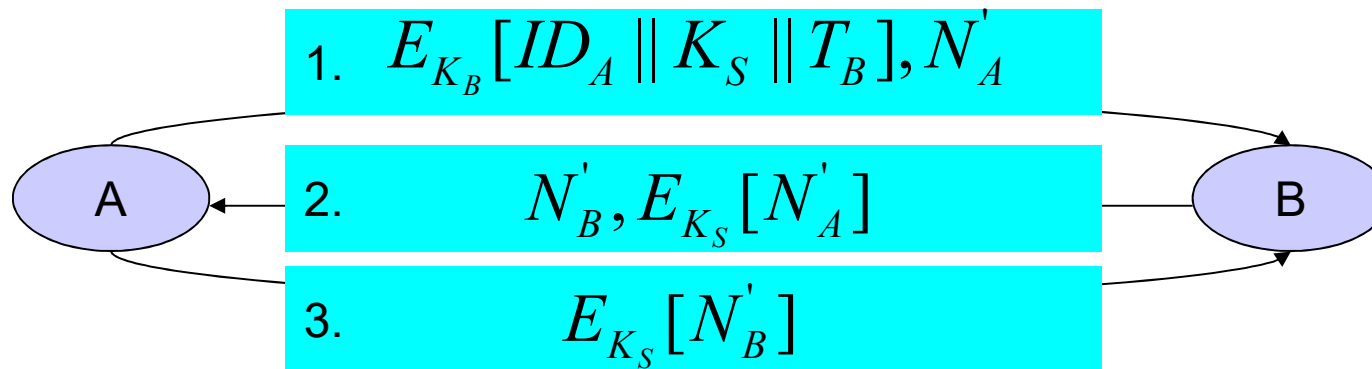
# Needham—Schroeder改进协议 (2)





# Needham—Schroeder改进协议 (2)

有效期内可不通过KDC直接认证

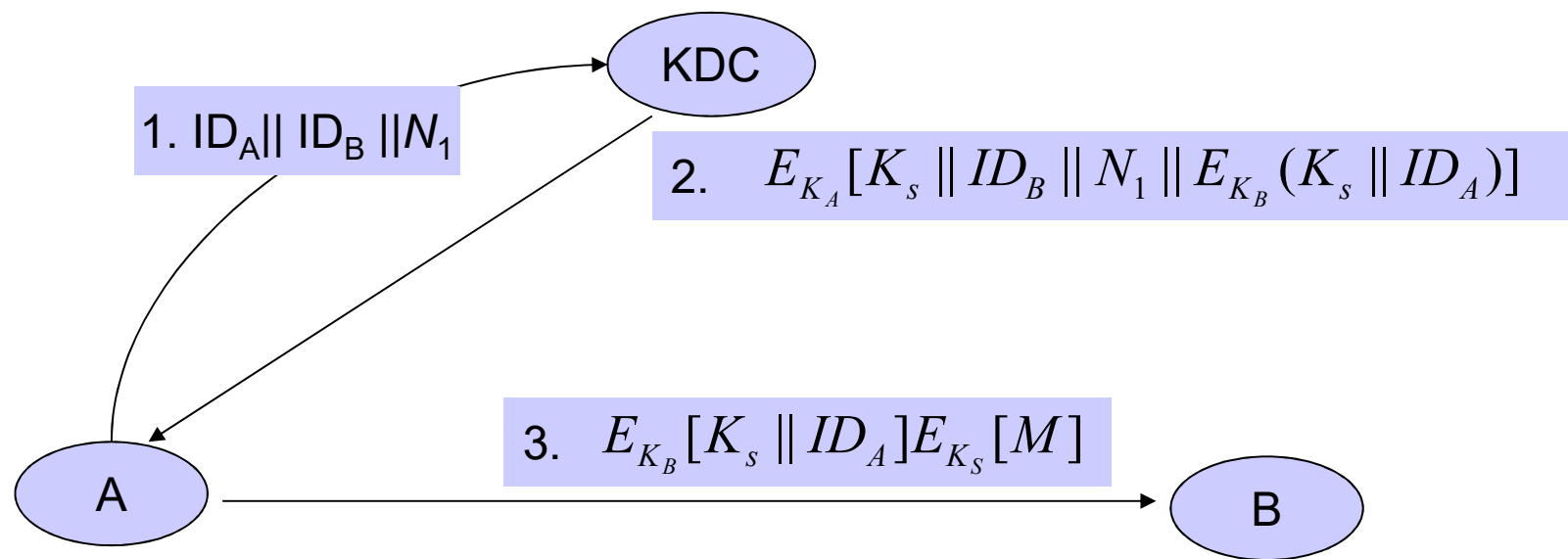


# 单向认证



- 不需要双方同时在线（电子邮件）
- 邮件接收者希望认证邮件的来源以防假冒
- 分为单钥加密方法和公钥加密方法

# 单钥加密



不要求B同时在线，保证只有B能解读消息，提供对A的认证。不能防止重放攻击。



# Kerberos认证系统

# Kerberos

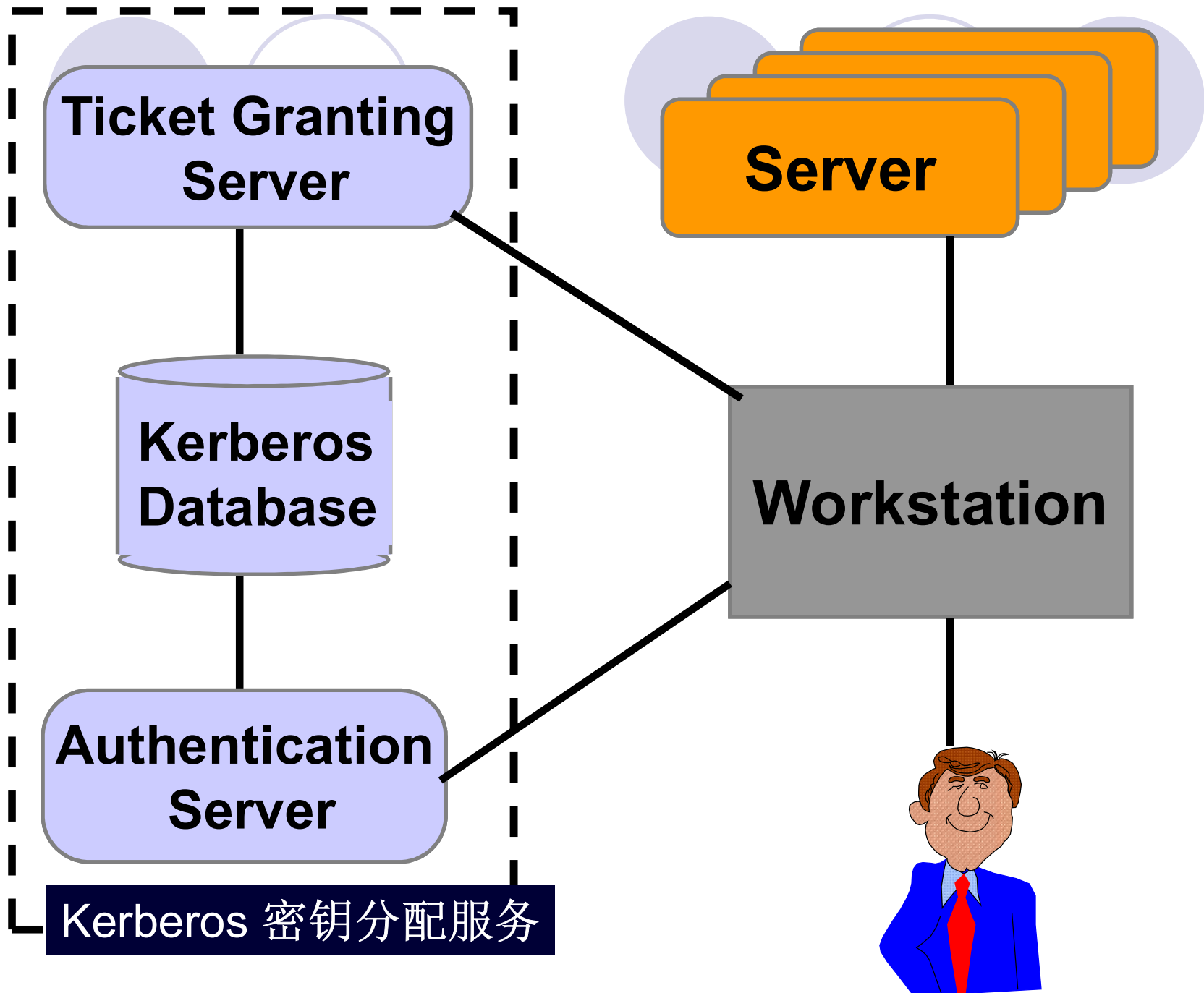
- 雅典娜计划的一部分 (MIT)。
- 可信的第三方认证方案。
- 假定主机是不可信的
- 要求每个client (对每次业务请求) 证明其身份。
- 不要求用户每次业务请求都输入密码!
- 密码设计基于Needham – Schroeder协议

# Kerberos 设计

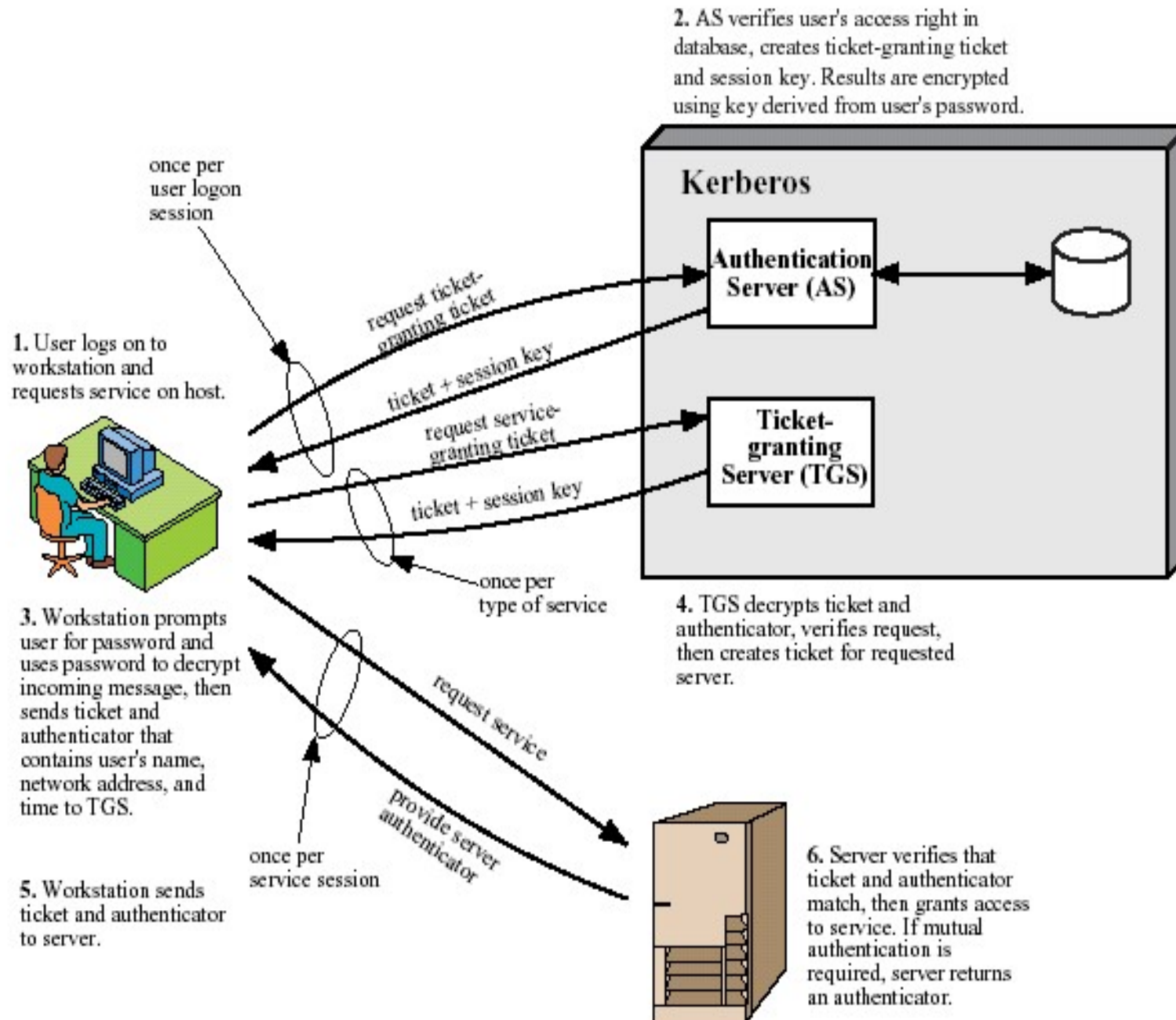
- 用户必须在工作站会话开始的时候验证自己(登录会话).
- 密码永远不在网络中明文传输(或在存储器中存储)

# Kerberos设计 (续)

- 每个用户有一个口令.
- 每个业务有一个口令.
- 知道所有口令的唯一实体是认证服务器.







# 票据 (Tickets)

- 每个业务请求需要一个 ticket.
- 一个只能用于单个用户访问单个服务器.
- Tickets 由 “Ticket Granting Server” (*TGS*) 分发, *TGS* 有所有的加密密钥.
- Tickets 对 clients 是无意义的, clients 只是用他们接入服务器.

## 票据 (Tickets) (续)

- *TGS* 用服务器的加密密钥加密每个ticket
- 加密的tickets可在网上安全传输，只有服务器可以解密。
- 每一 ticket有有限的生存期 (几个小时)。

# Ticket 内容

- Client名 (用户登陆名)
- Server 名
- Client 主机网络地址
- Client/Server会话密钥，用于加密 client和server间的请求和响应
- Ticket 生存期
- 产生时戳

# 认证符 (Authenticators)

- 认证符证明client身份.
- 包括
  - Client 用户名.
  - Client 网络地址.
  - 时戳.
- 认证符以 session key加密.

# 问题



- 每次client要访问要申请新的ticket.
- 为了从 *TGS* 获得ticket, client 必须已经有 TG ticket 和一个会话密钥与 *TGS* 通信!

# 认证服务器(AS)

- Client向AS发送明文请求以获得一个ticket用于和TGS通信。
- 请求:
  - 登录名
  - TGS名称

由于这个请求只包括公开的名字，不需要加密。

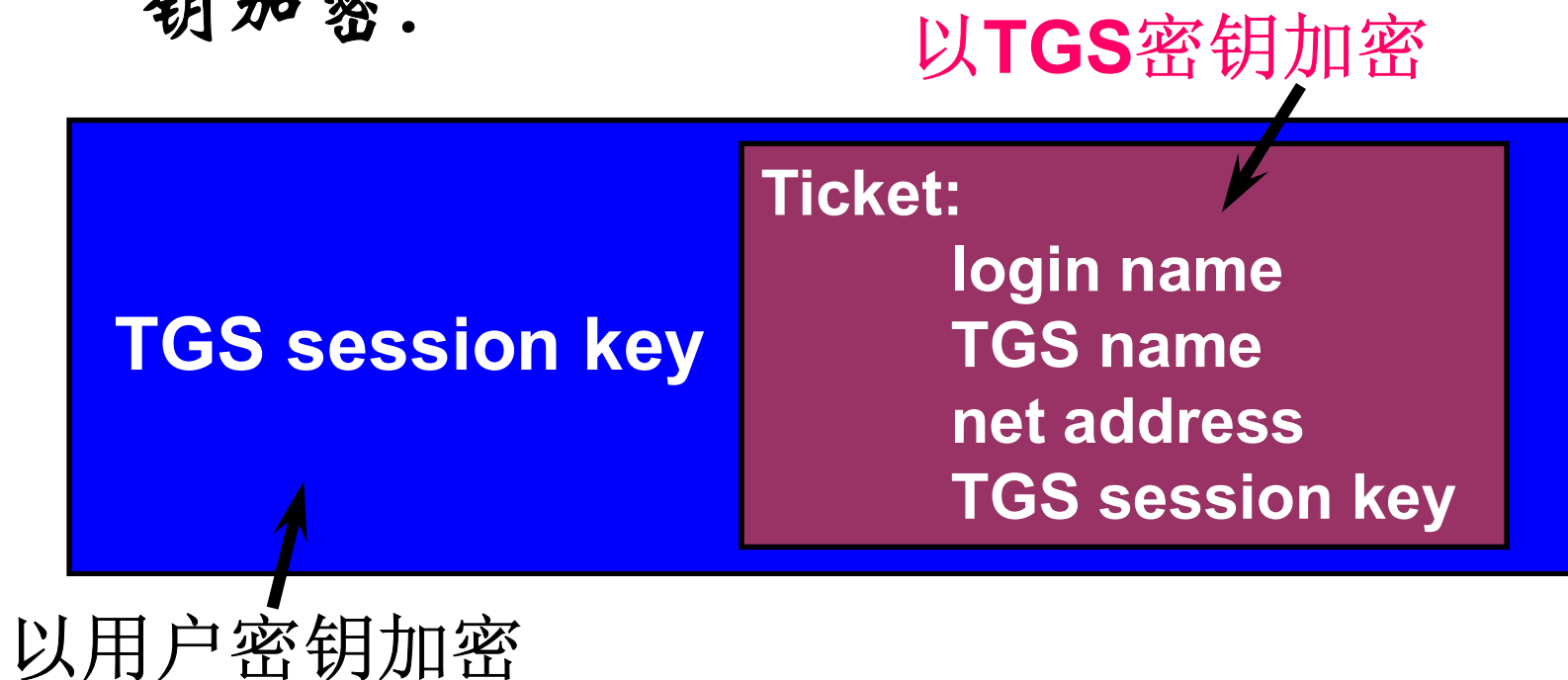
## 认证服务器(AS)

- *AS* 找到与登陆名和 *TGS* 名对应的密钥。
- *AS* 生成一个ticket:
  - login name
  - *TGS* name
  - Client网络地址
  - *TGS* 会话密钥
- *AS* 以 *TGS* 秘密密钥加密此ticket.



# AS响应

- AS同时产生一个随机会话密钥供client和TGS使用.
- 会话密钥和加密的ticket 以用户的秘密密钥加密.

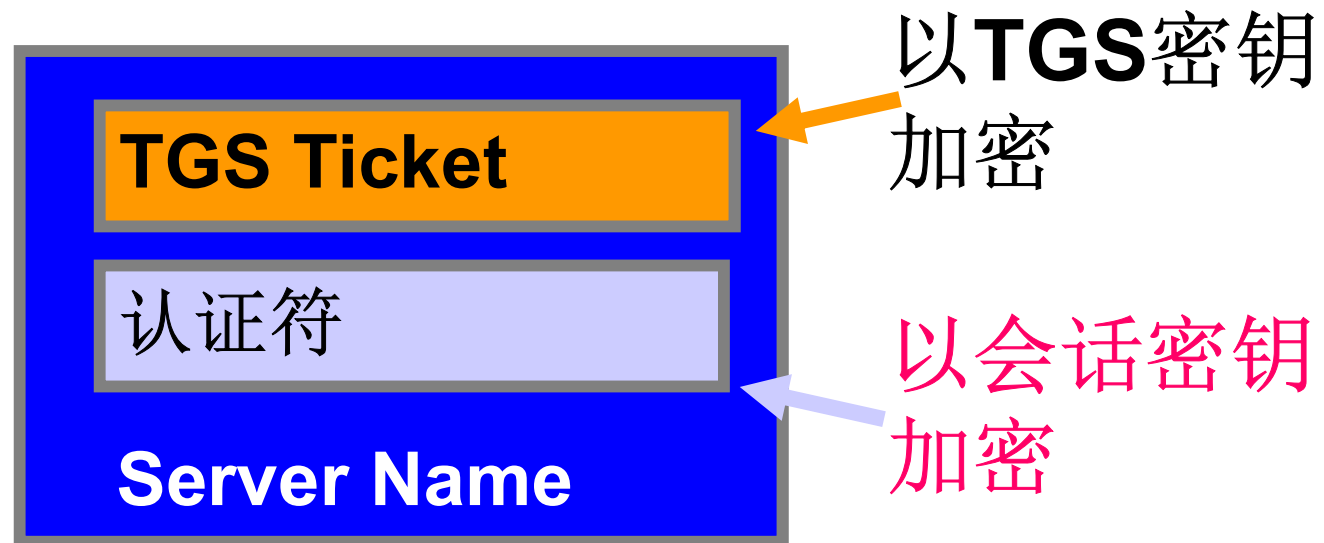


## 访问 *TGS*

- Client以用户口令作为秘密密钥解密消息.
- Client这时获得了会话密钥和与 *TGS* 通信的 ticket.
- Client看不到ticket内部的内容,因为client 不知道 *TGS* 的秘密密钥.

# 访问服务器

- 当client想开始使用服务,client必须首先获得一个 ticket.
- Client构造一个请求发送给 *TGS*:



## TGS 响应

- *TGS* 用其秘密密钥解密 ticket 获得 *TGS* 会话密钥.
- *TGS* 用会话密钥解密信任状.
- *TGS* 检查验证登陆名, client 地址和 *TGS* server 名都正确.
- *TGS* 验证信任状是最新的.

# TGS 响应



一旦所有验证通过，TGS:

- 产生一个ticket使client用于请求的server.  
Ticket用server密钥加密.
- 产生一个会话密钥
- 以TGS会话密钥加密整个消息发回给client.

# 用户访问服务器

- Client以TGS会话密钥解密 *TGS*响应.
- Client现在获得了与服务器通信的会话密钥和ticket.
- client 用与和访问 *TGS*一样格式的数据访问服务器。

# Kerberos 总结

- 每业务请求需要一个 ticket.
- Tickets 来源于TGS (除了访问TGS的ticket).
- 工作站不能解密用服务器密钥加密的tickets.
- 每个 ticket有个相应的session key.
- Tickets 不能重用.

# Kerberos 总结（续）

- Tickets有有限生命期.
- 信任状只能使用一次.
- 信任状失效的更快!
- 服务器维护一个信任状列表



# 协议内容

## 第一阶段 身份验证服务交换

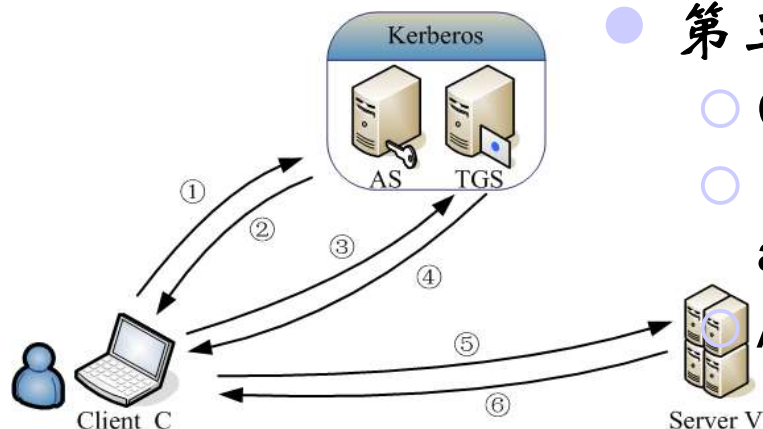
- $C \rightarrow AS: ID_C \parallel ID_{tgs} \parallel TS_1$
- $AS \rightarrow C: E_{K_C}[K_{c,tgs} \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2 \parallel Ticket_{tgs}]$
- $Ticket_{tgs} = E_{K_{tgs}}[K_{c,tgs} \parallel ID_C \parallel AD_C \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2]$

## 第二阶段 票据授予服务交换

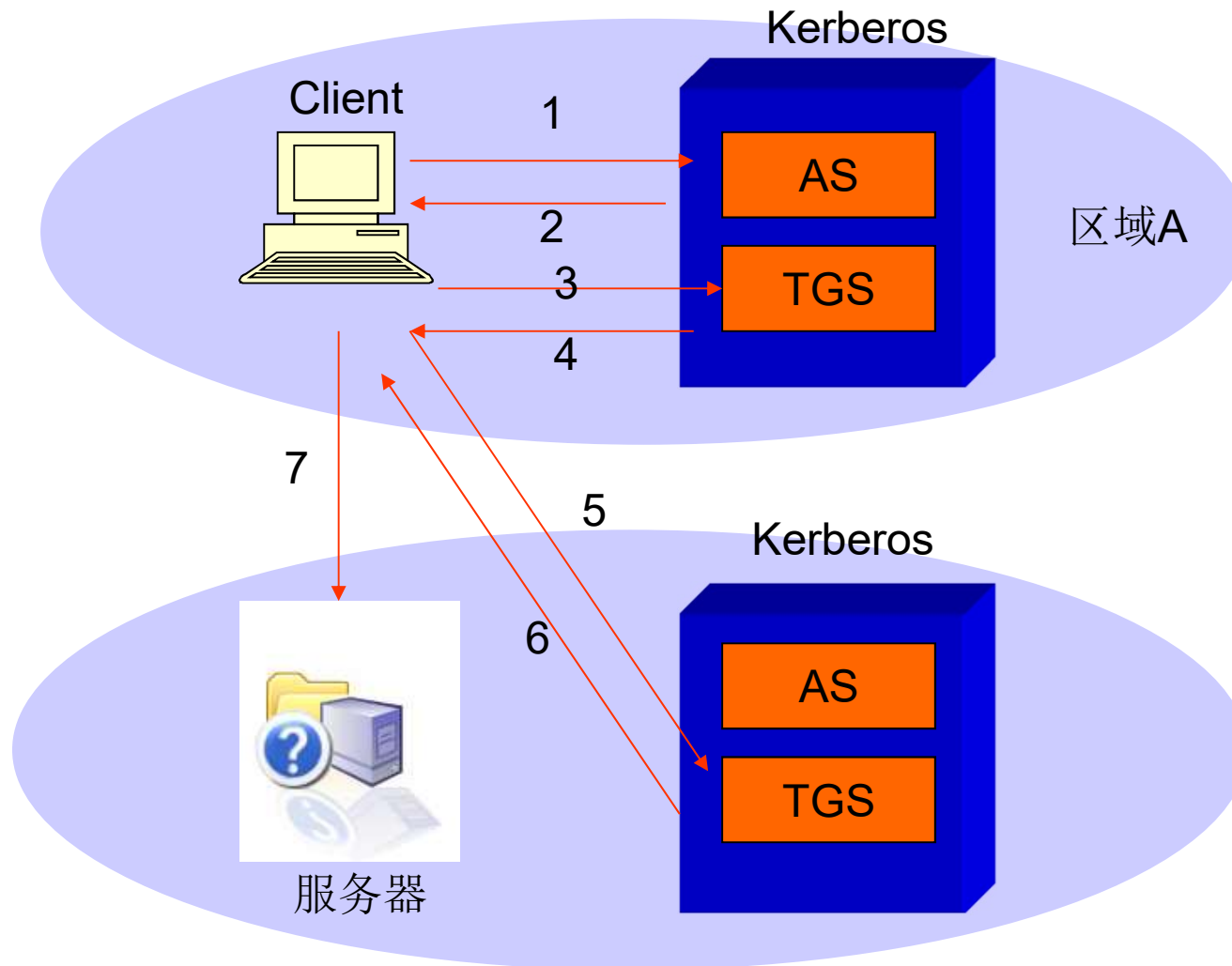
- $C \rightarrow TGS: ID_V \parallel Ticket_{tgs} \parallel Authenticator_c$
- $TGS \rightarrow C: E_{K_C}[K_{c,v} \parallel ID_V \parallel TS_4 \parallel Ticket_v]$
- $Authenticator_c = E_{K_C}[ID_C \parallel AD_C \parallel TS_3]$
- $Ticket_v = E_{K_V}[K_{c,v} \parallel ID_C \parallel AD_C \parallel ID_V \parallel TS_4 \parallel Lifetime_4];$

## 第三阶段 客户与服务器身份验证交换

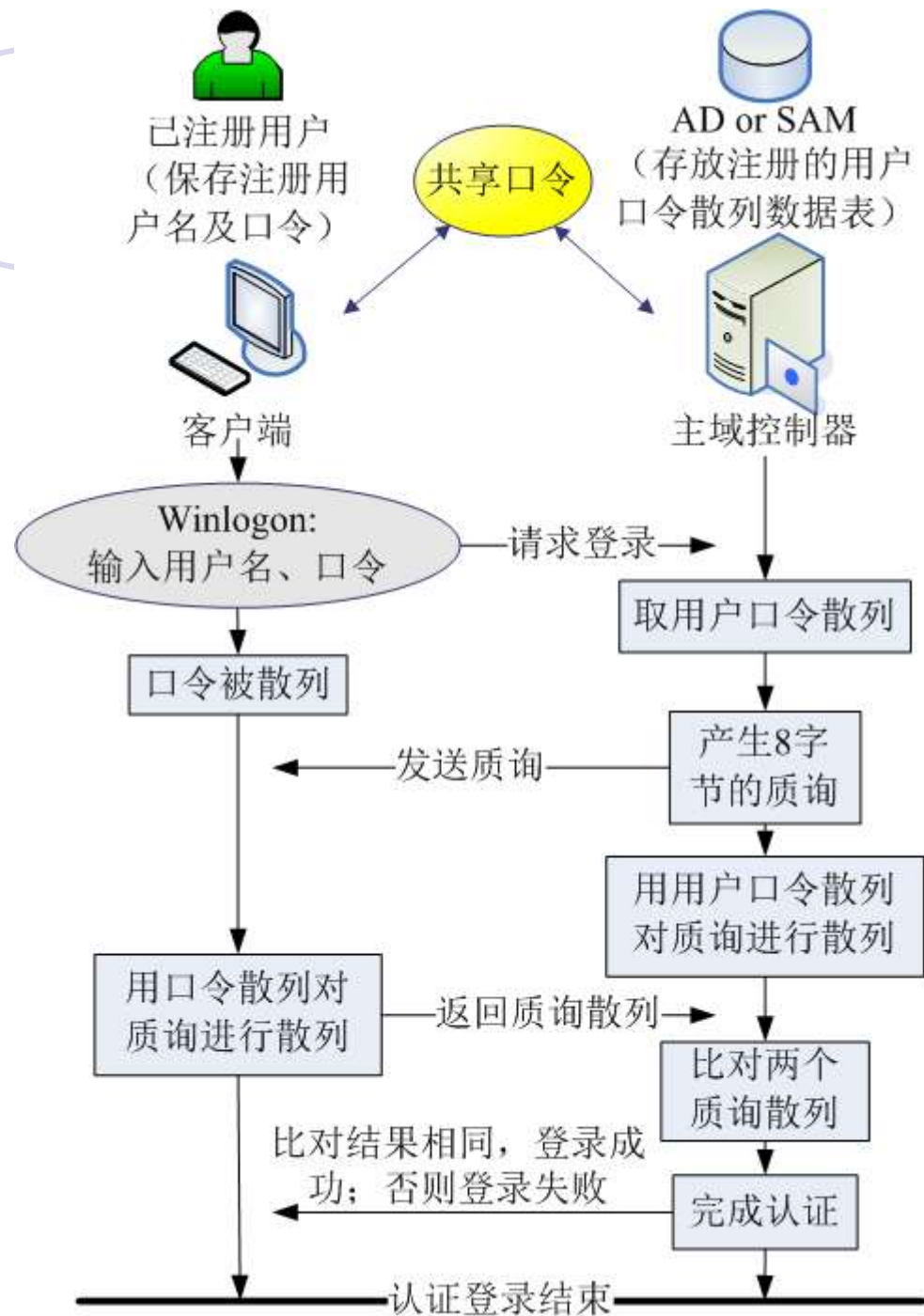
- $C \rightarrow V: Ticket_v \parallel Authenticator_c$
- $V \rightarrow C: E_{K_C}[TS_5 + 1] \text{ (for mutual authentication)}$
- $Authenticator_c = E_{K_C}[ID_C \parallel AD_C \parallel TS_5]$



# Kerberos 区域和多区域的Kerberos



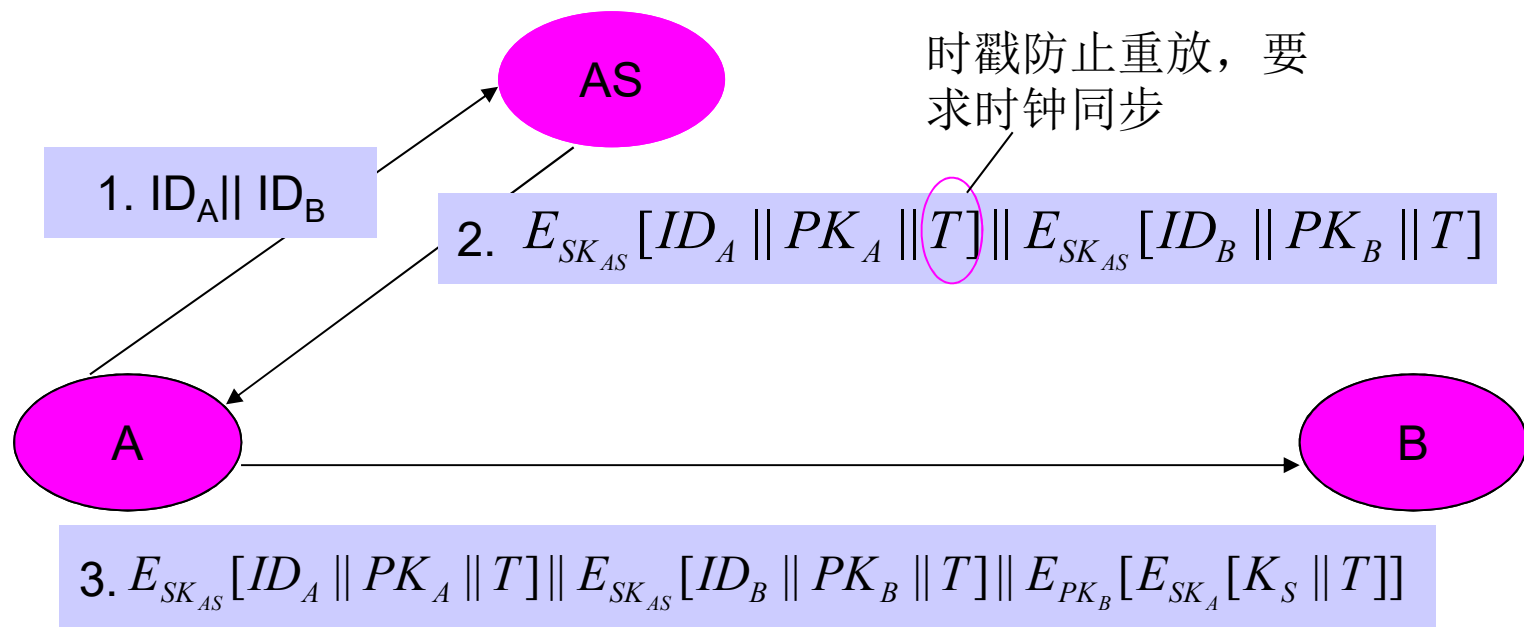
# Windows系统的安全认证



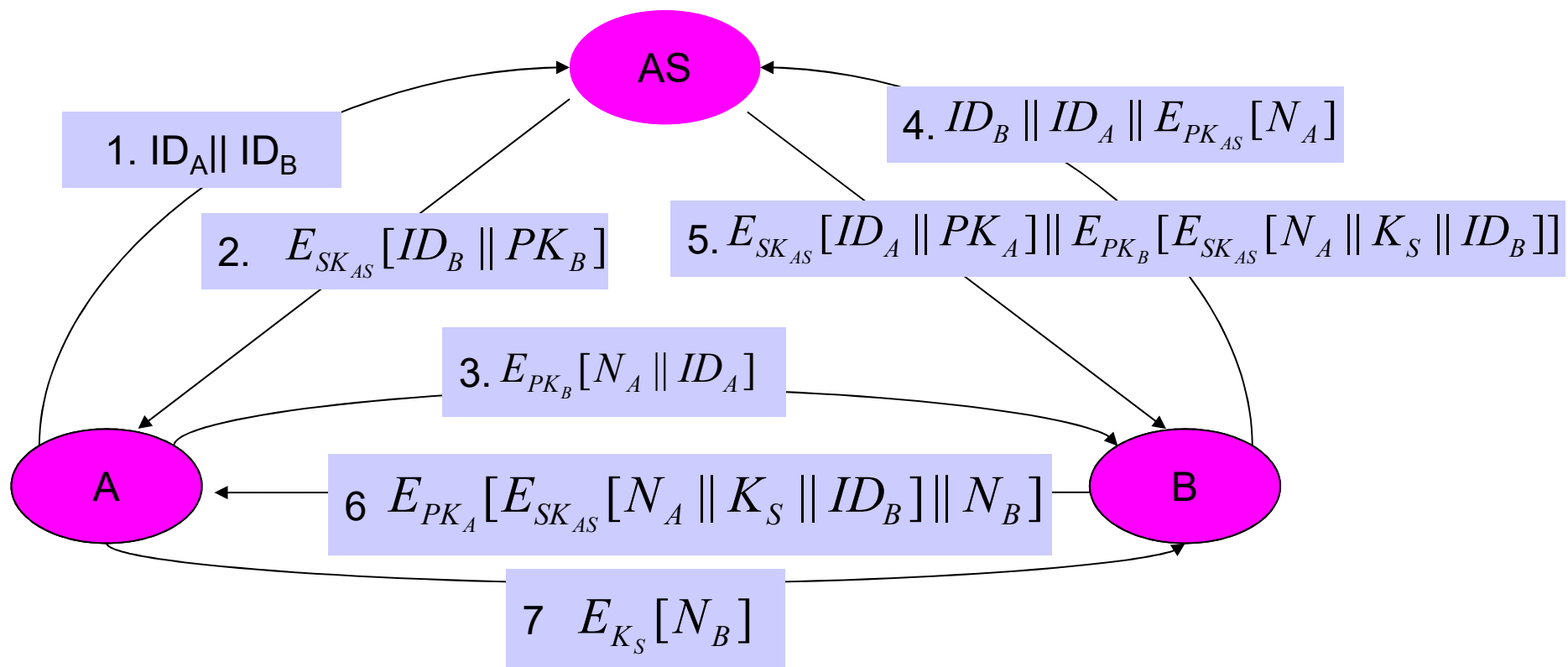


# 基于非对称加密的远程用户认证

# 公钥加密体制



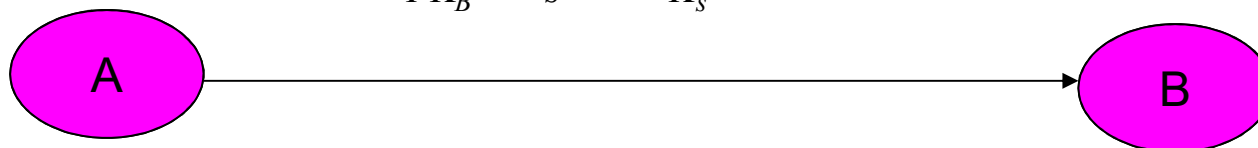
# 公钥加密体制



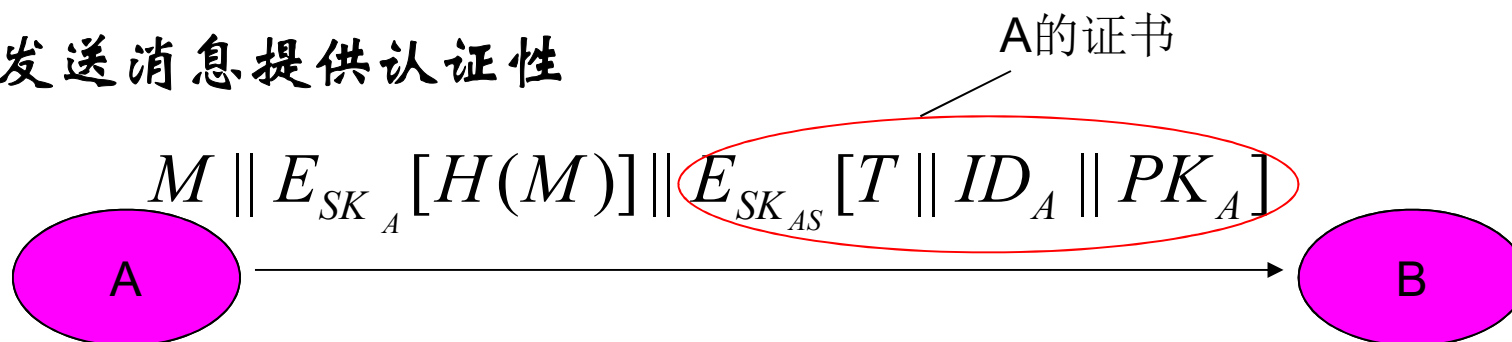
# 公钥加密单向认证

对发送消息提供保密性

$$E_{PK_B}[K_s] \parallel E_{K_s}[M]$$



对发送消息提供认证性



对发送消息提供保密和认证性





X.509和PKI

Public Key Infrastructure



# X.509认证交换协议



- OSI目录检索服务标准X.500首先公布于1988年
- 该标准中包括了一部分陈述认证的标准，即ISO/IEC 9594-8或ITU-T X.509建议。
- 1993年和1995年分别对X.509建议作了微小修改。

# X.509认证交换协议

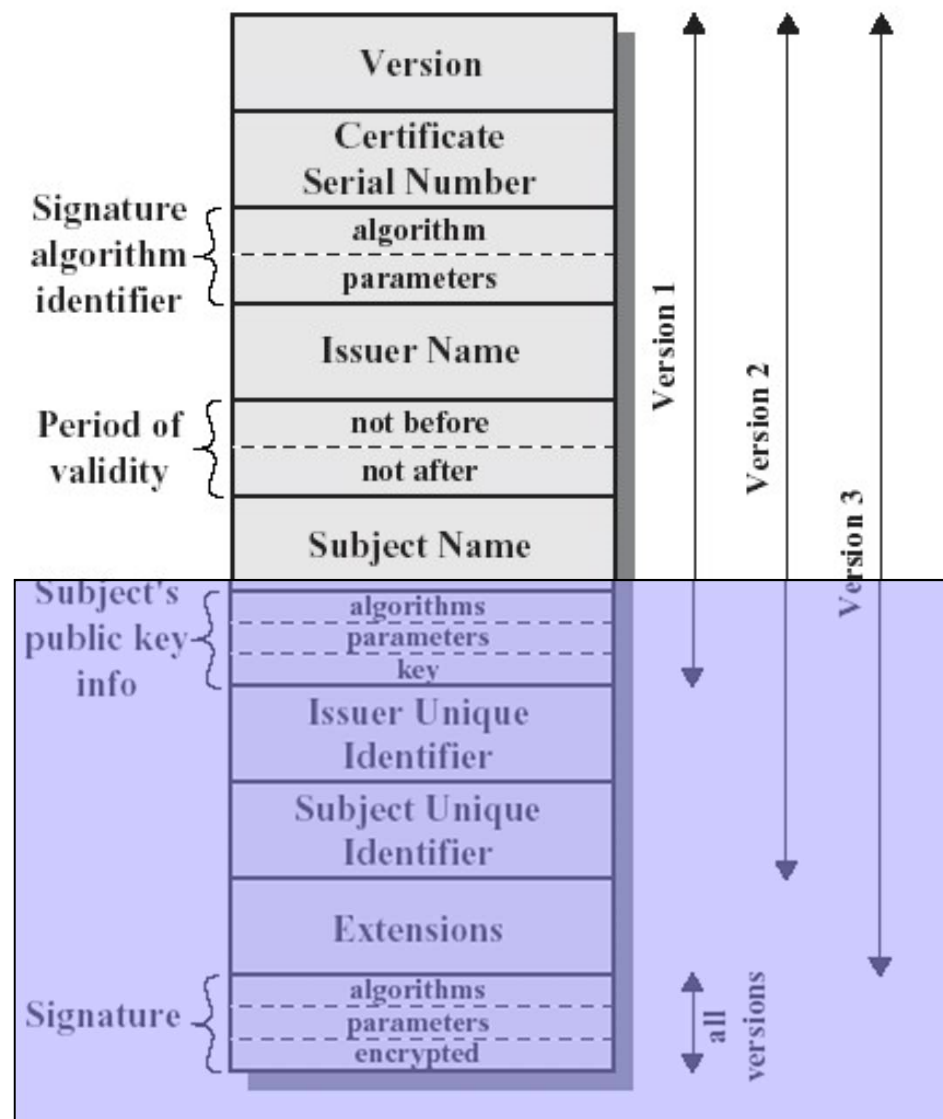
- X.509 基于 **公开密钥加密** 和 **数字签名**。
  - 这个标准没有专门指定使用的加密算法，但推荐使用RSA。
  - 数字签名则假定要求使用散列函数，同样，标准没有专门指定使用的散列算法。
    - 1988的建议书包括一个推荐散列算法的描述，这个算法已被证明是不安全的，1993年的推荐书中已经去掉了这个算法。

# X.509认证交换协议

- X. 509方案的核心是与每个用户联系的**公开密钥证书**。
  - 这些用户证书采取由某些可信**证书权威机构(CA)**创建，由CA或用户放在目录中。
  - **目录服务器**本身不负责公开密钥的生成或证书函数，它仅提供一个易于访问的位置以使用户获得证书。

# X.509证书格式

- 版本1、2、3
- 序列号
  - 在CA内部唯一
- 签名算法标识符
  - 用来签名证书的算法以及一些相关的参数。由于这个信息与在证书尾部签名字段的内容重复，因此这个字段几乎没有作用。
- 签发人名字
  - CA的名字
- 有效时间
  - 起始和终止时间
- 主体名
  - 这张证书提及的用户名。也就是这张证书证实了持有相应私人密钥主体的公开密钥。



# X.509证书格式(续)

- 个体的公钥信息

- 算法
- 参数
- 密钥

- 签发人唯一标识符

- 个体唯一标识符

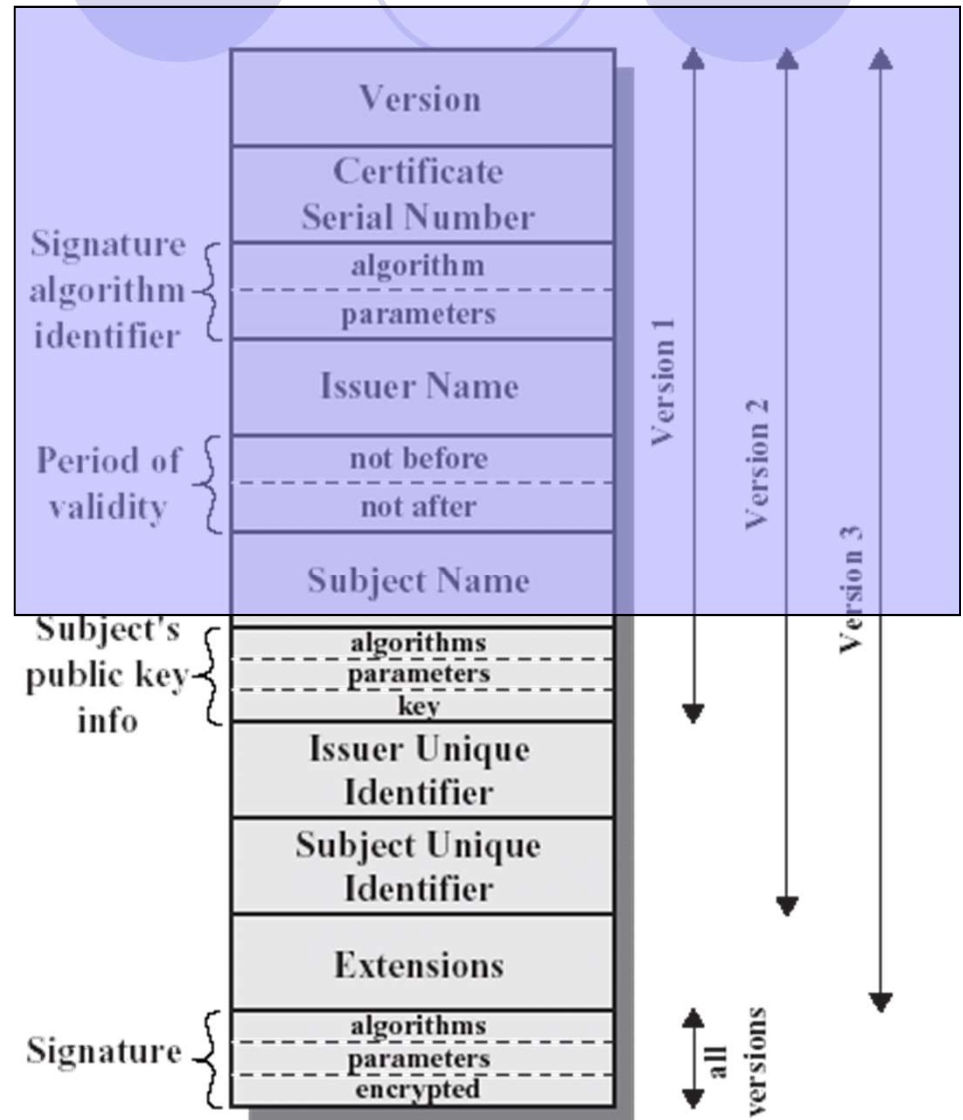
- 扩展域

- 第3版中加入

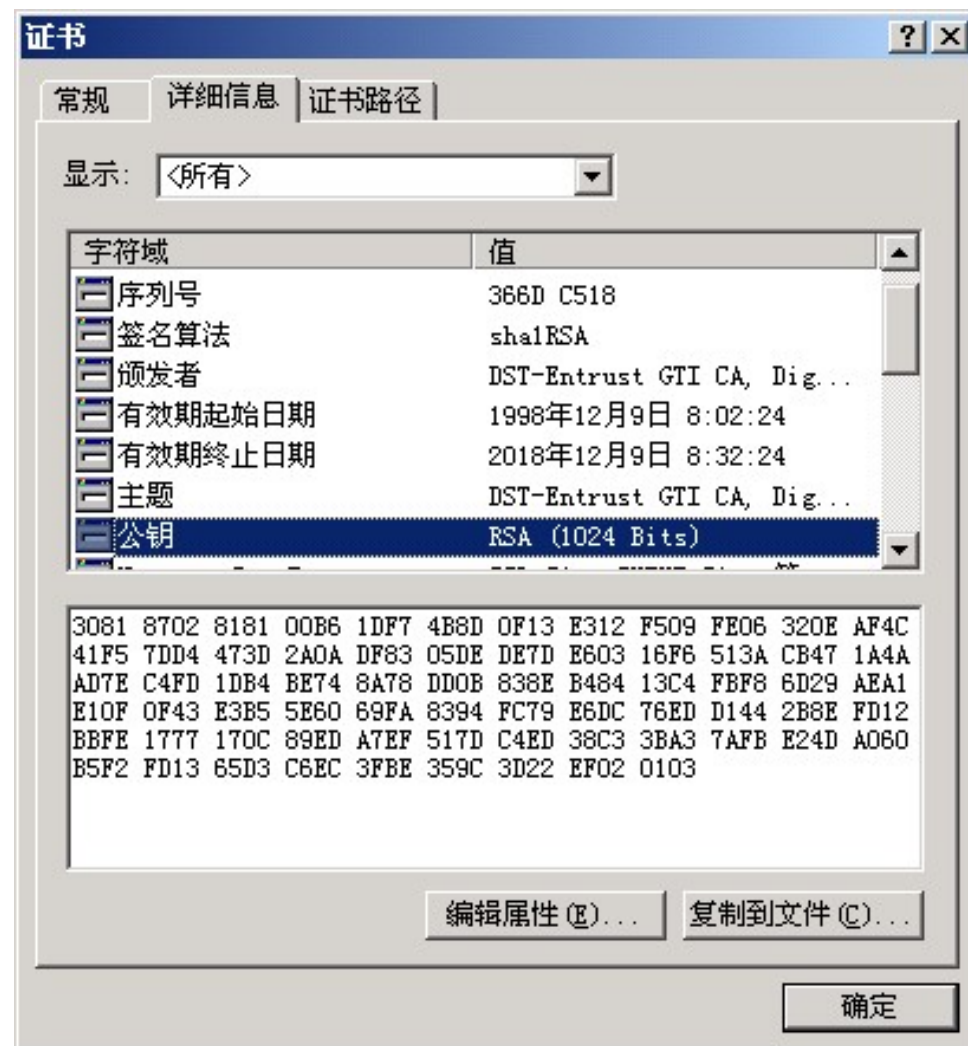
- 签名

- 包括证书的其他所有字段；它包含用CA私有密钥加密的其他字段的散列码。这个字段还包括签名算法标识符。

第2版中  
加入，  
甚少用  
到



# X.509证书示意图



# X.509证书的符号定义

- 这个标准使用下面的符号定义一个证书：

$CA\langle\langle A \rangle\rangle = CA\{V, SN, AI, CA, T_A, A, A_P\}$

其中

$Y\langle\langle X \rangle\rangle$  = 由证书权威机构Y颁发的用户X的证书

$Y\{I\}$  = Y对I的签名。它由I加上一个加密的散列码组成。

- CA用它的私有密钥对证书签名。

- 如果一个用户知道对应公开密钥，那么这个用户可以验证由CA签名的证书是有效的。

# 获得一个用户证书

- 证书可以放在一个目录内。而无需目录提供特殊的保护措施。
  - 因为证书是不可伪造的
  - 如果所有用户都预订了一个CA的证书，那么这是对那个CA的共同信任，所有用户的证书能被放在所有用户都能访问的目录内。
  - 用户可以直接将他或她的证书传递给其他的用户。



# 证书链

## ● 动机

- 如果用户数目巨大，让所有用户间向一个CA预订可能是不现实的。
- 当用户多了，有许多CA可能更切合实际
- 每个CA可以安全地向一小部分用户提供它的公开密钥。

# 证书链

- 假定A已经从证书权威机构 $X_1$ 处获得一证书，B从证书权威机构 $X_2$ 获得一证书。
  - 如果A无法安全地获得 $X_2$ 的公开密钥，那么 $X_2$ 颁发给B的证书对A是没有用的；A可以阅读B的证书但不能验证签名。
- 如何使A安全可信的获得B的公开密钥？
  - 两个CA能够安全地交换各自的公开密钥

# 证书链

- 如果两个CA能够安全地交换各自的公开密钥，下面的过程可以使A获得B的公开密钥：
  1. A从目录获得 $X_1$ 签名的 $X_2$ 的证书。
    1. 因为A能安全地知道 $X_1$ 的公开密钥，A从它的证书中能获得 $X_2$ 的公开密钥，并通过证书中 $X_1$ 的签名来证实它。
  2. 然后A能回到目录中得到由 $X_2$ 签名的B的证书。
    1. 因为现在A已经拥有一个可信的 $X_2$ 的公开密钥备份，因此A能验证这个签名并安全地获得B的公开密钥。

# 证书链

- A使用一个证书链获得了B的公开密钥。这个链用X.509的符号可表示如下

$$X_1 \ll X_2 \gg X_2 \ll B \gg$$

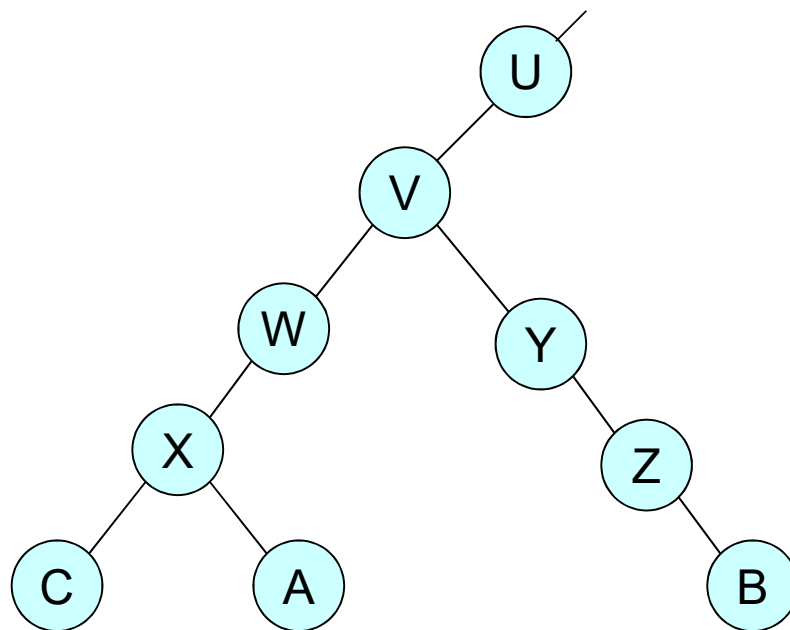
- 用同样的方式，B使用相反的链可获得A的公开密钥：

$$X_2 \ll X_1 \gg X_1 \ll A \gg$$

证书链并不局限于两个CA的情况下，可以用任意长度的CA来产生一个链。在这种情况下，链中的每对CA必须已经相互创建了证书。

# 证书链

- X.509 建议 CA 组织成层次结构。



A如何获得B的证书上?

# 证书的注销

- 由于各种原因，证书需要被注销
  - 比如，私钥泄漏、密钥更换、用户变化
- 注销的方法
  - CA维护一个CRL(Certificate Revocation List)
- 当用户收到报文中的证书时，必须确定证书是否已被取消。
  - 用户每次收到一个证书时可以检查目录；
  - 为了避免相关目录检查带来的时延(和可能的开销)，用户可能会保持一个证书和表的高速缓存或者被撤销的证书。



# PKI的用途

## 机密性

只有指定的接收者才能访问信息。

## 数据完整性

交易中的信息不可篡改。

## 技术上的不可抵赖性

交易中一方事后不能抵赖。

## 认证

通信中一方可以确定和谁在通信。



# PKI 还可用于 ....

## 授权

用户可以有效的访问拥有权限的受控资源。

## 访问控制

用户可以方便的控制需要进行控制的资源



## PKI包含什么？

一个PKI是一系列安全业务的集合，  
采用公钥密码和 X.509 数字证书，  
应用于分布式计算系统。

# PKI包含什么？

- 密钥对：公开和秘密密钥
- X.509数字证书
- CA (Certificate Authority)
- RA (Register Authority)
- 公共目录服务 (PKI)
- 相关CA间的信任关系
- 一组PKI政策：注销，维护，更新等等。
- PKI 应用
- 集成PKI的目录服务 (企业)

# 专用还是通用PKI

- 目前许多消息和群组产品提供自己的PKI，专用PKI支持专门应用。
- 也需要通用PKI支持多种应用，管理多个应用产生的密钥 集中管理证书政策，与企业目录更紧密的集成，降低管理的复杂性。

# 密钥对交换

The title is positioned to the left of a series of five circles. The first two circles are partially overlapping and partially behind the title text. The remaining three circles are spaced out horizontally to the right of the first two.

问题:

1. 接收者如何确定的知道发送者的公开密钥 (用于验证数字签名)
2. 发送者如何确定的知道接收者的公开密钥 (用于加密消息)

# 数字证书



解决方案：数字证书。

数字证书将一个人或其他实体的身份和公开密钥绑定，以确保密钥确实属于该用户或实体。

# 数字证书

包含下列信息的数据结构:

- 所有者公钥
- 用户或实体的身份
- 发布者身份
- 由第三方信任机构签名 (Certificate Authority)
- 有限的有效期.
- X.509 定义该结构 (ITU, IETF PKIX standards). 现在为 v3.

# 数字证书分类



## 3. 中级

注册过程需要个人当面认定，高强度的密码模块和访问控制

典型应用：敏感的公司间邮件或 EDI

## 4. 高级

需要个人当面以及详细调查最强的计算安全控制

# 数字证书

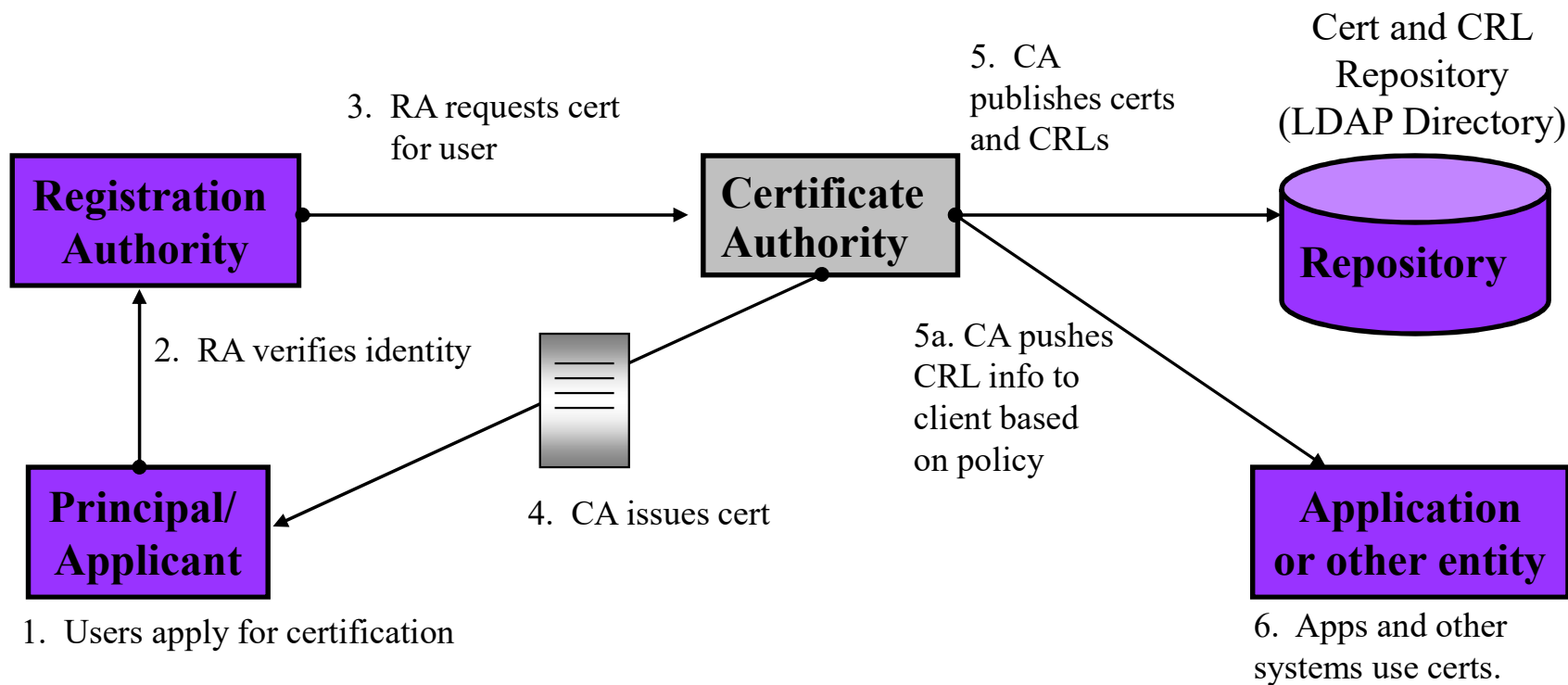


Certificate Authority (CA) 对给定的安全域产生和发布证书。

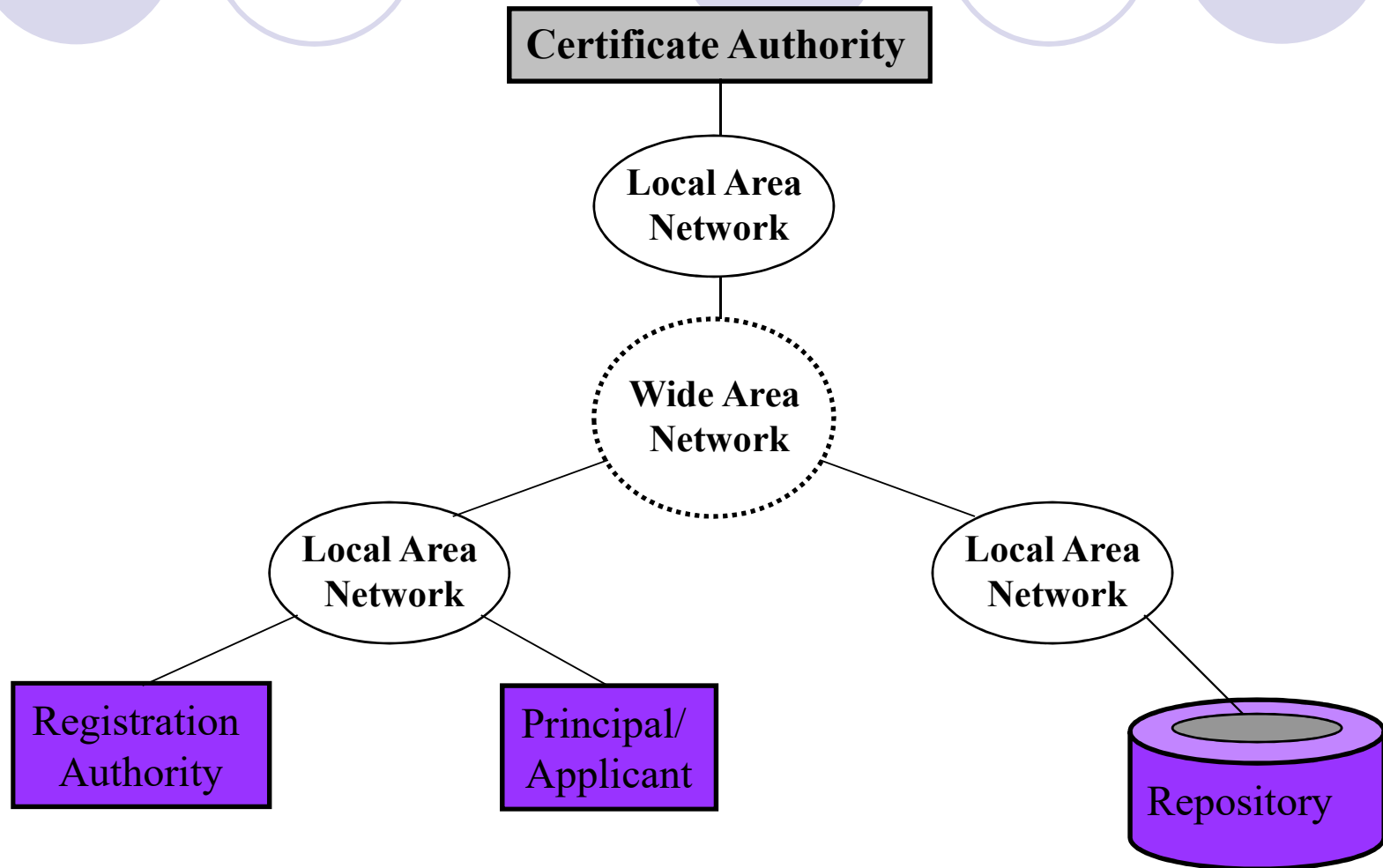
控制政策，相对其他用户具有绝对的权威。



# 数字证书



# 数字证书



# CA 信任模型

任何支持超出单个安全域的PKI必须考虑：

如何建立对 CA本身的信任??

问： 谁是CA的可信第三方？  
谁签发它的数字证书？（分层）  
它和谁共享信任？（交叉证书）

# CA 信任模型



三种主要的 CA 信任模型:

1. 交叉认证
2. 证书链
3. hub/桥 CA

# CA 信任模型

## 1. 交叉认证:

### A. 证书信任列表

CTLs 是信任的证书或 CA 的列表.

Web 浏览器可以在client端存储 CTLs , 或在中心目录或信任server上存储。管理员可指定信任CA的集合.

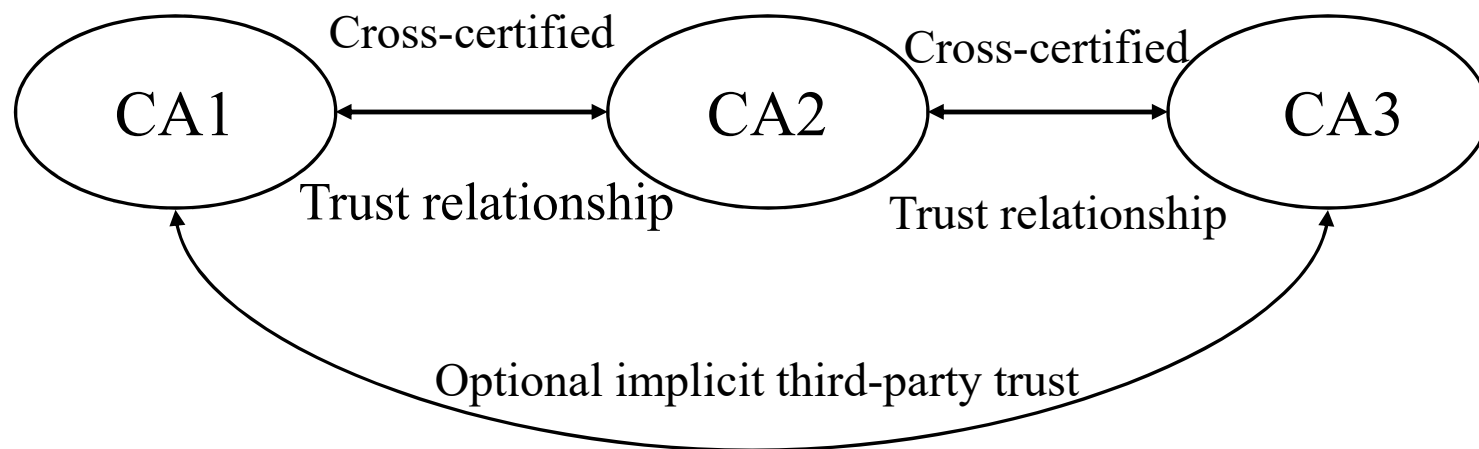
在验证证书有效性的同时,client检查发布证书的CA是否在证书列表中。

# CA 信任模型

## 1. 交叉认证:

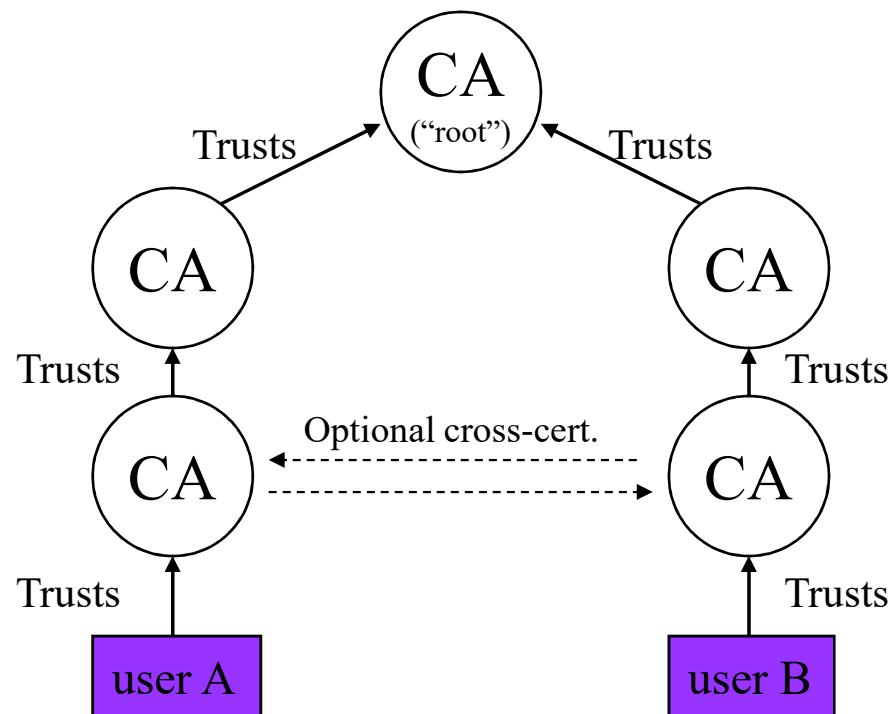
### B. 双侧交叉认证

CAs 通过公证彼此的公钥彼此交叉认证。



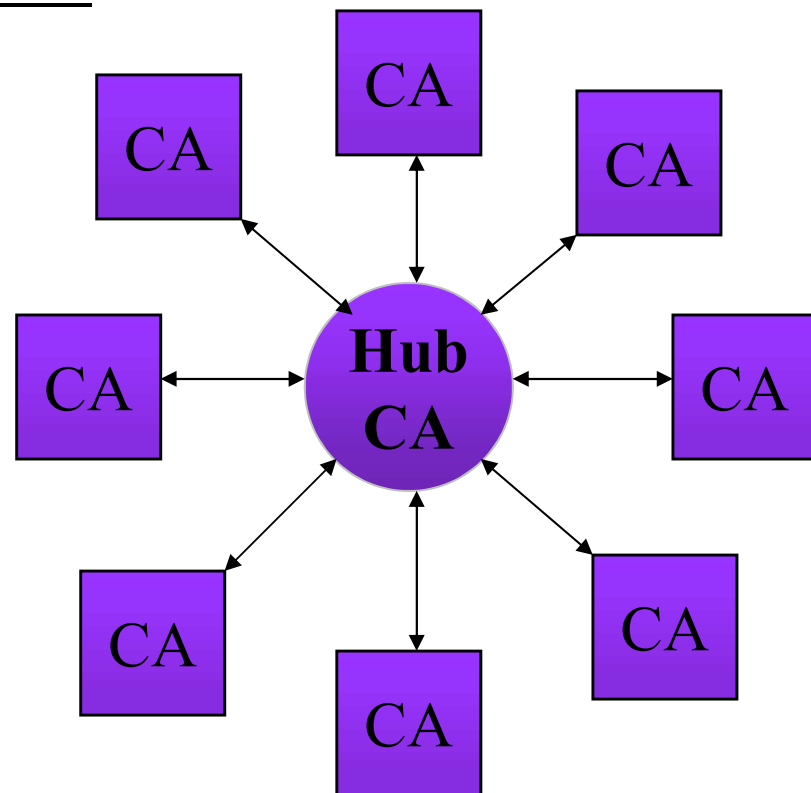
# CA 信任模型

## 2. 证书链:



# CA 信任模型

## 3. Hub/桥 CA





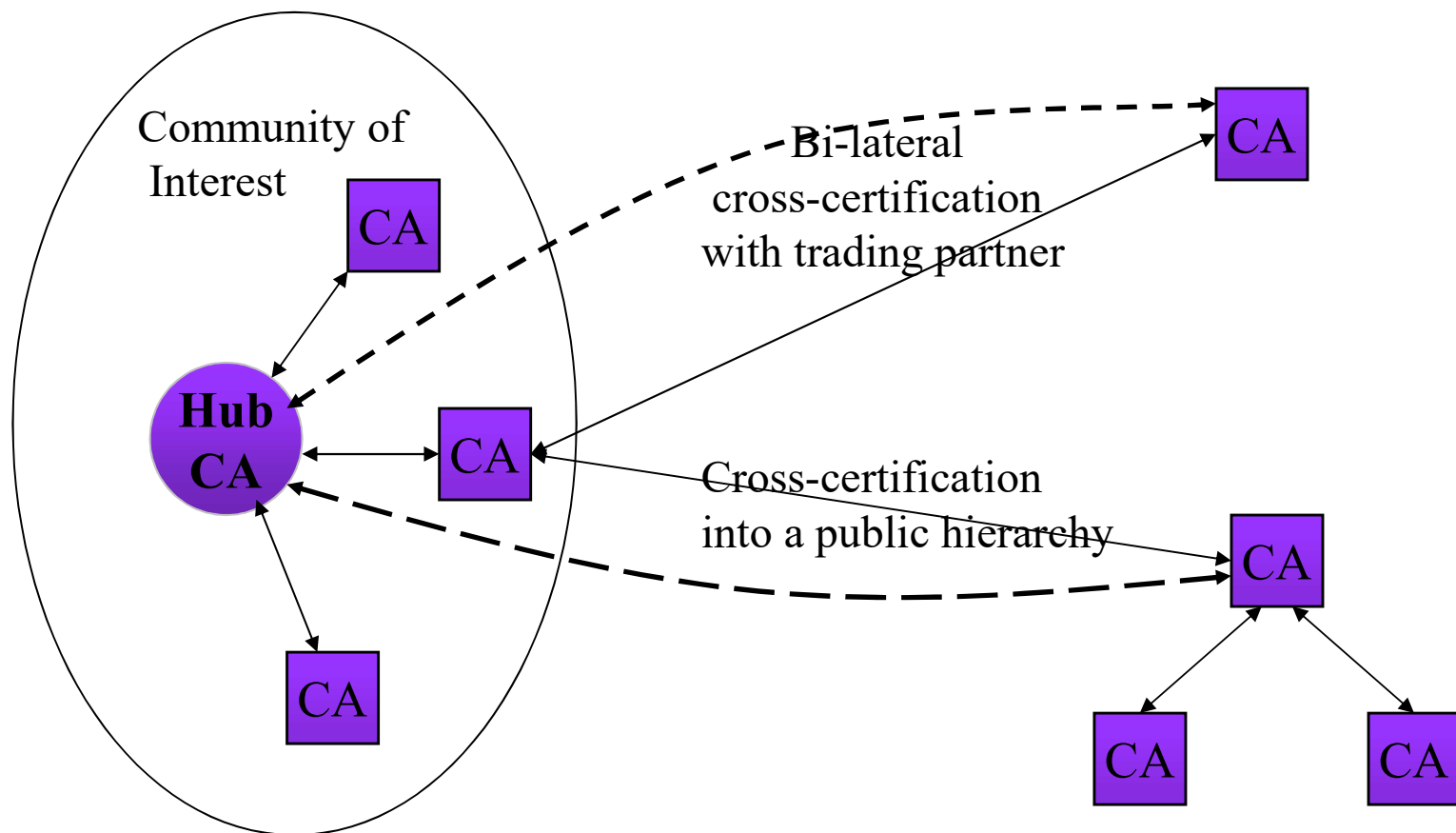
# CA 信任模型

## 3. Hub/桥CA

- Hub/桥 CA 为成员提供中心化的信任管理/和政策映射。
- 建立确信级别和维护政策。
- Hub CA 会检查成员CA 的政策以保证一致性和完整性。
- 处理交叉认证任务和维护目录仓库。

# CA 信任模型

## Hybrid





# PKI 政策

“证书政策”：

“A named set of rules that indicates the applicability of a certificate to a particular community and/or class of application with common security requirements.”

-- *RFC 2527*

可指定证书应用的限制

也用于授权 CAs:

# 秘密密钥管理



## 1. Client-based 密钥存储:

- 当前最常用的密钥存储. (Microsoft “Wallet”, Entrust “Profile”)
- PKI client 使用软件存储机制在客户PC的硬盘上加密用户的credentials/私钥. 用户通常使用口令. .

# 秘密密钥管理

## 2. Smart Cards:

- 私钥的理想存储机制.
- 存储私钥, 证书, 其他 credentials, 通常以 PIN 保护.
- 便携!

# 秘密密钥管理

## 3. 秘密密钥的目录存储:

- 访问秘密信息, 用户登录进入目录.
- 应用为本地应用从目录下载秘密签名和加密密钥。
- 提供位置无关性.
- 管理员可以集中管理安全信息和政策。



# 授权和访问控制

PKI, 密钥对管理, 数字证书 均用于认证....

但是数字证书也可用于第二个目的--授权:

## 授权

用户可以有效的访问其拥有合法权限的资源

## 访问控制

用户/管理员可以容易地对其负责的资料进行访问控制。.

# 授权和访问控制

集成 PKI 的授权基础设施使用一些和 PKI 一样的构件，主要有：

X.509 数字证书

LDAP 目录

- 但是应当认为是一个不同的基础设施。
- 在证书的一个域中提供一个指向动态目录 (通常是 LDAP) 的指针，其中包括授权的属性，使得证书是相对静态的。





# 身份证明技术

# 身份证明技术

传统的身份证明：

一般是通过检验“物”的有效性来确认持该物的身份。徽章、工作证、信用卡、驾驶执照、身份证、护照等，卡上含有个人照片(易于换成指纹、视网膜图样、牙齿的X适用的射像等)。

信息系统常用方式：

用户名和口令

# 交互式证明



## 两方参与

示证者P(Prover), 知道某一秘密, 使V相信自己掌握这一秘密;

验证者V(Verifier), 验证P掌握秘密; 每轮V向P发出一询问, P向V做应答。V检查P是否每一轮都能正确应答。

# 交互证明与数学证明的区别

- 数学证明的证明者可自己独立的完成证明
- 交互证明由P产生证明，V验证证明的有效性来实现，双方之间要有通信
- 交互系统应满足
  - 完备性：如果P知道某一秘密，V将接收P的证明
  - 正确性：如果P能以一定的概率使V相信P的证明，则P知道相应的秘密

# Fiat-Shamir身份识别方案

参数:

选定一个随机模  $m = p \times q$ 。产生随机数  $v$ ，且使  $s^2 = v$ ，即  $v$  为模  $m$  的平方剩余。  $m$  和  $v$  是公开的， $s$  作为  $P$  的秘密

# Fiat-Shamir身份识别方案

- (1) P取随机数 $r(<m)$ , 计算 $x = r^2 \bmod m$ , 送给V;
- (2) V将一随机bit  $b$ 送给P;
- (3) 若 $b=0$ , 则P将 $r$ 送给V; 若 $b=1$ , 则P将 $y = rs$ 送给V;
- (4) 若 $b=0$ , 则V证实 $x = r^2 \bmod m$ , 从而证明P知道, 若 $b=1$ , 则V证实 $xv = y^2 \bmod m$ , 从而证明p知道。

这是一次证明, p和v可将此协议重复 $t$ 次, 直到v相信p知道 $s$ 为止。

# Fiat-Shamir身份识别方案

- 完备性
  - 如果P和V遵守协议，且P知道 $s$ ，则应答 $rs$ 是模 $m$ 下 $xv$ 的平方根，V接收P的证明，所以协议是完备的。
- 正确性
  - P不知道 $s$ ，他也可取 $r$ ，送 $x=r^2 \bmod m$ 给V，V送 $b$ 给P。P可将 $r$ 送出，当 $b=0$ 时则V可通过检验而受骗，当 $b=1$ 时，则V可发现P不知 $s$ ，B受骗概率为 $1/2$ ，但连续 $t$ 次受骗的概率将仅为 $2^{-t}$
  - V无法知道P的秘密，因为V没有机会产生 $(0,1)$ 以外的信息，P送给V的消息中仅为P知道 $v$ 的平方根这一事实。

# 零知识证明

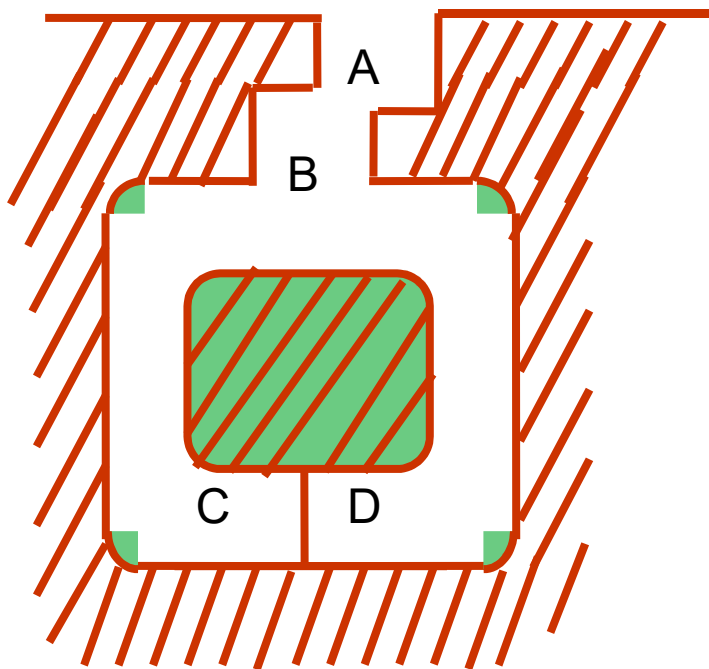
最小泄露证明和零知识证明：

以一种有效的数学方法，使 $V$ 可以检验每一步成立，最终确信 $P$ 知道其秘密，而又能保证不泄露 $P$ 所知道的信息。



# 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P不知咒语, 则在B点, 只有50 %的机会猜中V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 $1/65\,536$


# 零知识证明的基本协议

## 哈密尔顿回路

图论中有一个著名问题，对有 $n$ 个顶点的全连通图 $G$ ，若有一条通路可通过且仅通过各顶点一次，则称其为哈密尔顿回路。Blum[1986] 最早将其用于零知识证明。当 $n$ 大时，要想找到一条Hamilton回路，用计算机做也要好多年，它是一种单向函数问题。若A知道一条回路，如何使B相信他知道，且不告诉他具体回路？



- (1) A将 $G$ 进行随机置换，对其顶点进行移动，并改变其标号得到一个新的有限图 $H$ 。因 $G \cong H$ 故 $G$ 上的Hamilton回路与 $H$ 上的Hamilton回路一一对应。已知 $G$ 上的Hamilton回路易于找出 $H$ 上的相应回路；
- (2) A将 $H$ 的副本给B；
- (3) B向A提出下述问题之一：(a) 出示证明 $G$ 和 $H$ 同构，(b) 出示 $H$ 上的Hamilton回路；
- (4) A执行下述任务之一：(a) 证明 $G$ 和 $H$ 同构，但不出示 $H$ 上的Hamilton回路，(b) 出示 $H$ 上的Hamilton回路但不证明 $G$ 和 $H$ 同构；
- (5) A和B重复执行(1)~(4)共 $n$ 次。

Five circles are arranged in a horizontal row at the top of the slide. From left to right, the colors are: solid purple, white with a purple outline, solid purple, white with a purple outline, and solid purple.

*Any Questions?*