志存高远 责任为先

用户认证

地址：赣州市红旗大道86号 信息工程学院
网址：www.jxust.edu.cn
邮编：341000

目录/Contents

01
Part

身份认证概念

身份认证的概念

- **身份认证是计算机及网络系统识别操作者身份的过程**
  - **计算机网络是一个虚拟的数字世界，用户的身份信息是用一组特定的数据来表示的，计算机只能识别用户的数字身份，所有对用户的授权也是针对用户数字身份的授权**
  - **现实世界是一个真实的物理世界，每个人都拥有独一无二的物理身份**
  - **保证操作者的物理身份与数字身份相对应**

身份认证的功能

- **信息安全体系的目的是保证系统中的数据只能被有权限的“人”访问，未经授权的“人”无法访问**
- **身份认证是整个信息安全体系的基础**
  - **用于解决访问者的物理身份和数字身份的一致性问题，给其他安全技术提供权限管理的依据**
- **防火墙、入侵检测、VPN、安全网关等安全技术建立在身份认证之上**
  - **针对数字身份进行权限管理，解决数字身份能干什么的问题**

身份认证的分类

- **用户与主机之间的认证 – 认证人的身份**
  - **单机状态下的身份认证**
  - **计算机验证人的身份：你是否是你声称的那个人？**
  - **人的存储和计算能力有限**
    - **记忆高数量的密码密钥困难**
    - **执行密码运算能力有限**

身份认证的分类

- **主机与主机之间的认证 – 通信的初始认证握手**
  - **网络环境下的身份认证**
  - **计算机验证计算机**
  - **计算机存储和计算能力强大**
    - **能存储高数量的密码和密钥**
    - **能够快速地进行密码运算**

用户认证

- 由计算机对用户身份进行识别的过程
- 用户向计算机系统出示自己的身份证明，以便计算机系统验证确实是所声称的用户，允许该用户访问系统资源
- 用户认证是对访问者授权的前提，即用户获得访问系统权限的第一步。

用户认证的依据

- 所知 (what you know)
  - 密码、口令
- 所有 (what you have)
  - 身份证、护照、智能卡等
- 所是 (who you are)
  - 指纹、DNA等

静态口令

- **用户设定密码，计算机验证**
- **易泄露**
  - **用户经常用有意义的字符串作为密码**
  - **用户经常把密码抄在一个自己认为安全的地方**
  - **密码是静态数据，每次验证过程使用的验证信息都是相同的，易被监听设备截获**
- **用户名/密码方式一种是极不安全的身份认证方式**
  - **可以说基本上没有任何安全性可言**

两个问题

- **口令存储**
  - **通常经过加密后存储在计算机中**
- **口令传输**
  - **一般采用双方协商好的加密算法或单向散列函数对口令进行处理后传输**

动态口令 1

- 是一种让用户的密码按照**时间或使用次数**不断动态变化，每个密码只使用一次的技术，有效地改进口令认证的安全性
- 专用硬件：动态令牌
  - 内置电源、密码生成芯片和显示屏
  - 密码生成芯片运行专门的密码算法，根据当前时间或使用次数生成当前密码并显示在显示屏上
  - 认证服务器采用相同的算法计算当前的有效密码
  - 用户使用时只需要将动态令牌上显示的当前密码输入客户端计算机，即可实现身份的确认

动态口令 2

- **优点**
  - **每次使用的密码必须由动态令牌来产生，只有合法用户才持有该硬件**
  - **一次一密，每次登录过程中传送的信息都不相同，以提高登录过程安全性**
- **缺点**
  - **动态令牌与服务器端程序的时间或次数必须保持良好的同步**
  - **使用不便**

IC卡认证

- IC卡是一种内置集成电路的卡片，卡片中存有与**用户身份相关的数据**，可以认为是不可复制的硬件
- IC卡由合法用户随身携带，登录时必须将IC卡插入专用的**读卡器**读取其中的信息，以验证用户的身份
- IC卡硬件的不可复制可以保证用户身份不会被仿冒
- IC卡中读取的数据还是**静态**的
  - 通过内存扫描或网络监听等技术还是很容易截取到用户的身份验证信息

USB Key认证

- 近几年发展起来的一种方便、安全、经济的身份认证技术
- 软硬件相结合
- 一次一密
- USB Key是一种USB接口的硬件设备，它内置单片机或智能卡芯片，可以存储用户的密钥或数字证书，利用USB Key内置的密码学算法实现对用户身份的认证
- 可使用以上几种技术保护USB Key本身的安全

生物特征认证

- 采用每个人独一无二的**生物特征**来验证用户身份
  - 指纹识别、虹膜识别等
- 生物特征认证是**最可靠**的身份认证方式，因为它直接使用人的物理特征来表示每一个人的数字身份
- 受到现在的生物特征识别技术成熟度的影响

特点比较

	特点	应用	主要产品
用户名/密码方式	简单易行	保护非关键性的系统，不能保护敏感信息	嵌入在各种应用软件中
IC卡认证	简单易行	很容易被内存扫描或网络监听等黑客技术窃取	IC加密卡等
动态口令	一次一密，较高安全性	使用烦琐，有可能造成新的安全漏洞	动态令牌等
生物特征认证	安全性最高	技术不成熟，准确性和稳定性有待提高	指纹认证系统等
USB Key认证	安全可靠，成本低廉	依赖硬件的安全性	USB接口的设备

网络环境中的认证

- **分布式的网络环境**
  - **大量客户工作站和分布在网络中的服务器**
  - **服务器向用户提供各种网络应用和服务**
  - **用户需要访问分布在网络不同位置上的服务**
- **服务器的安全**
  - **服务器需要通过授权来限制用户对资源的访问**
  - **授权和访问控制是建立在用户身份认证基础上的**
- **通信安全都要求有初始认证握手的要求**

02
Part

网络用户认证

02.1
Part

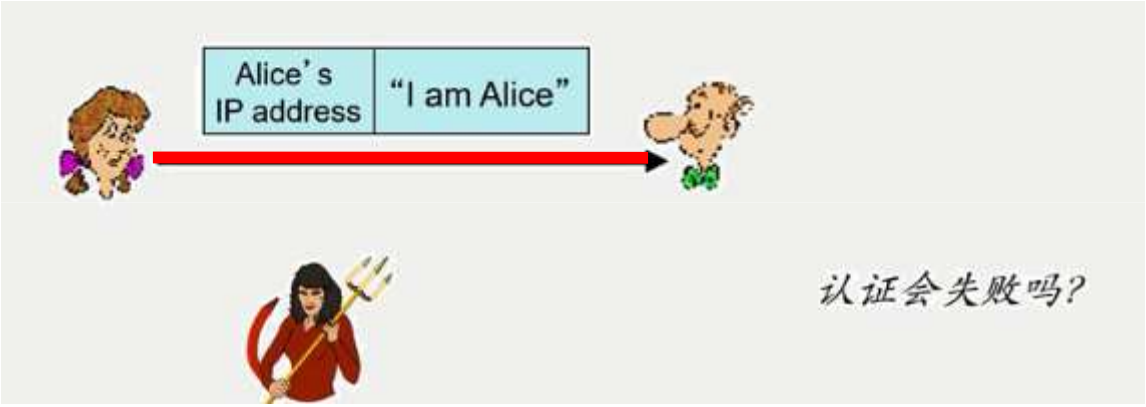
网络用户认证实例

证明我是我

- **认证1.0：I am alice**

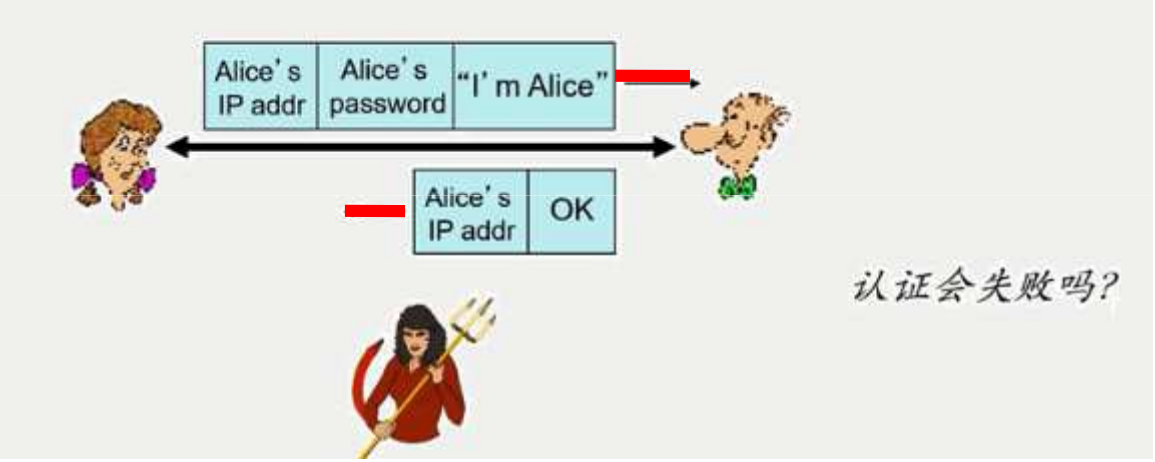
改进版

- **认证2.0：Alice把自己的IP地址放在报文里**



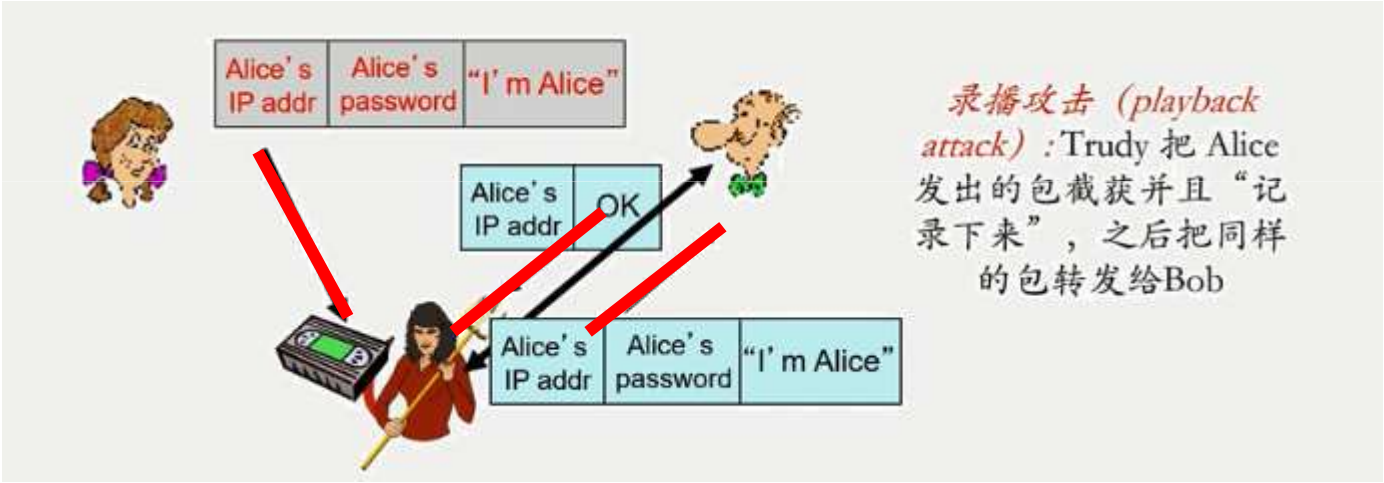
继续改进

- **认证3.0：Alice发送一段密码证明自己说的是真的**



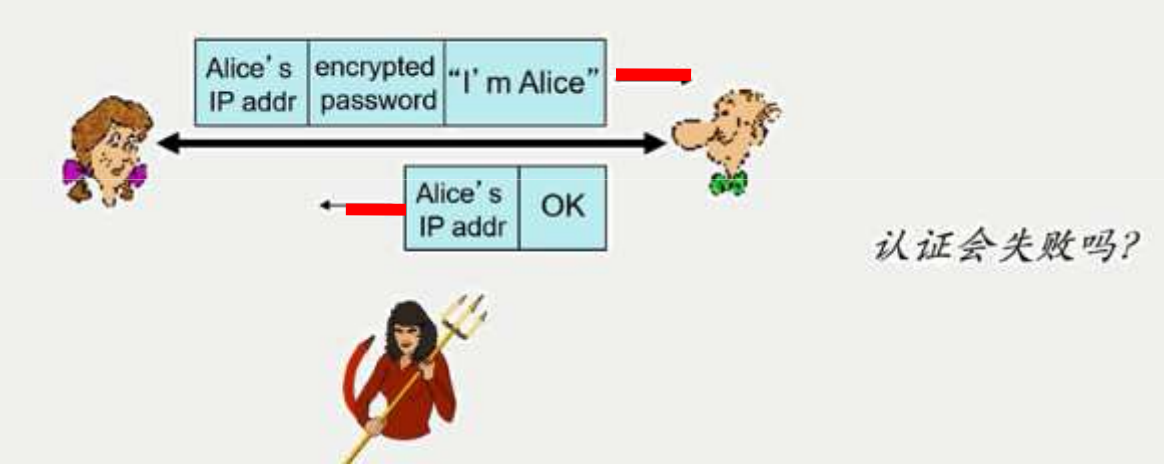
继续改进

- **认证3.0：Alice发送一段密码证明自己说的是真的**



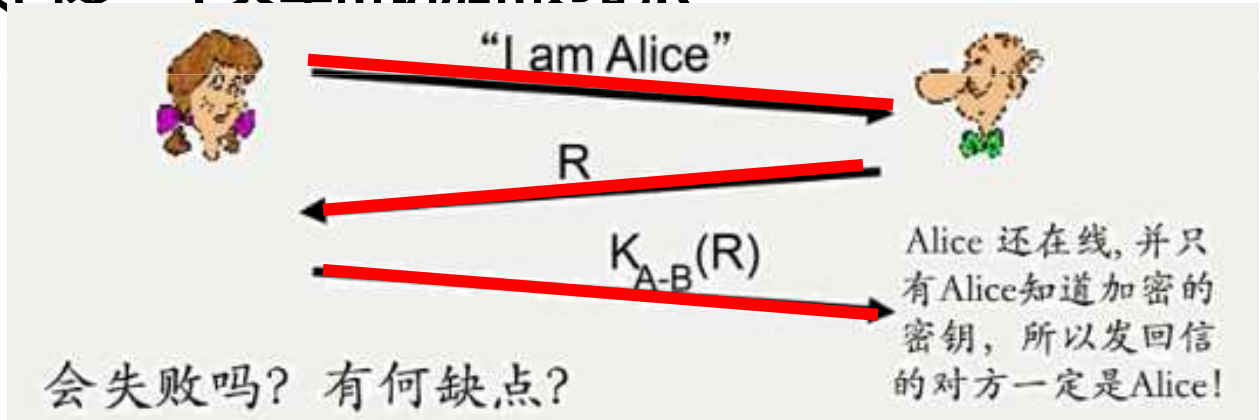
再试一次

- **认证3.1：Alice发送一段加密密码证明自己说的是真的**



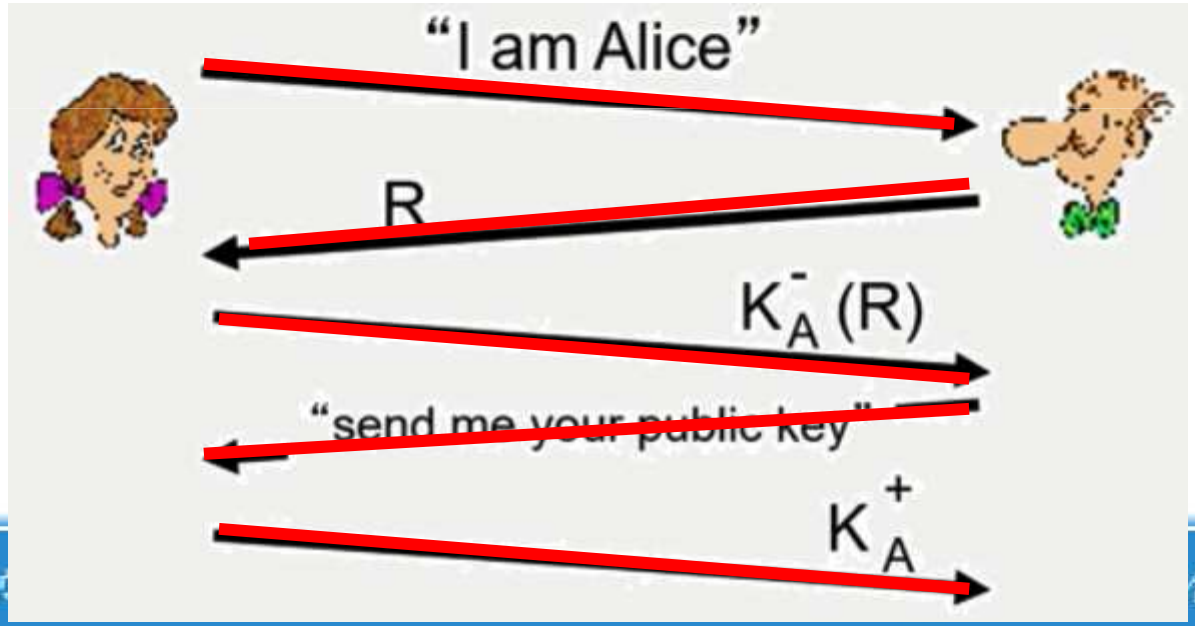
认证算法4.0

- **目标：避免录播攻击**
- **nonce:一个临时生成一次性使用的信息**
- **为了证明Alice还“在线”，Bob发给Alice一个nonce，R。此时Alice必须回复一个共享密钥加密过的R**



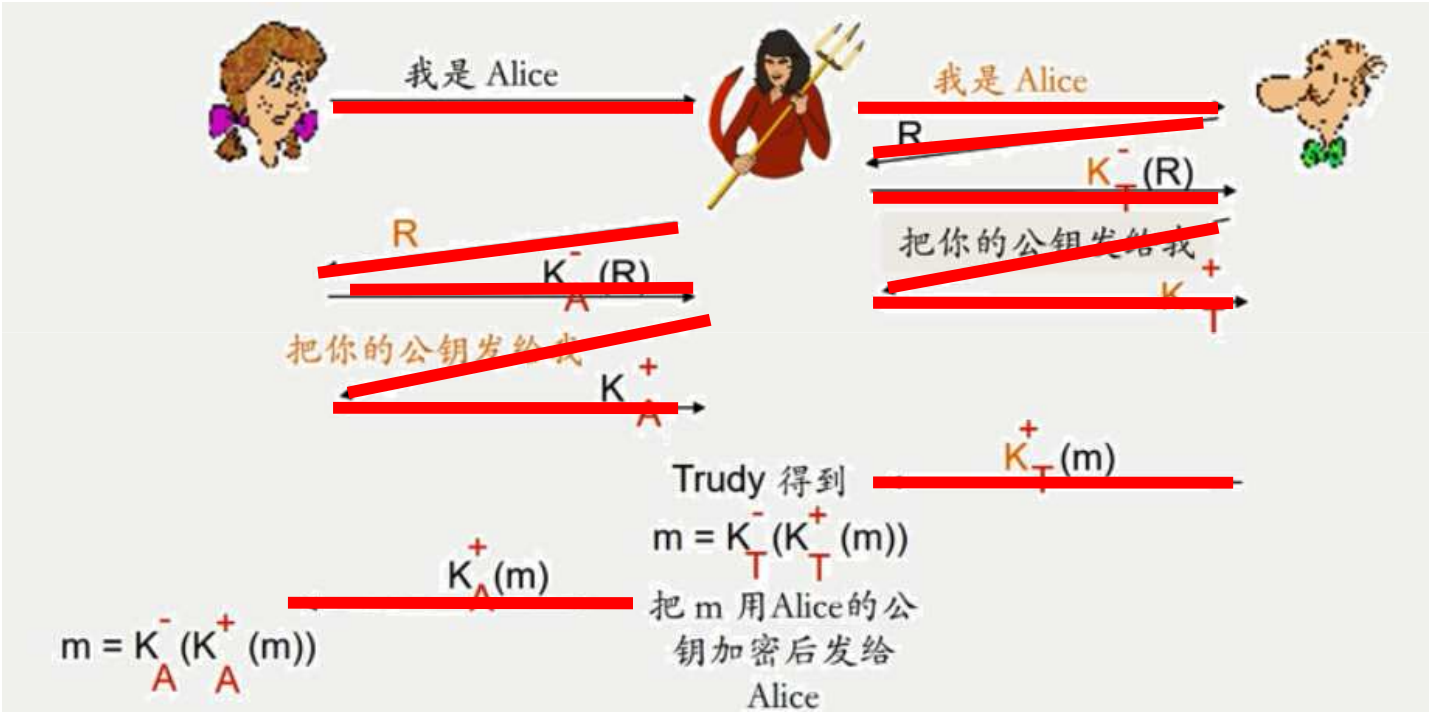
认证算法5.0

- 算法4.0要求共享密钥
- 我们是否可以设计出利用公共密钥结构的算法?
- **认证算法5.0:**用nonce和公共密钥结构做认证



认证算法5.0：安全漏洞

- ***中间人攻击***



02.2
Part

单向认证

单向认证

- 只有通信的一方认证另一方的身份，而没有反向的认证过程
- 电子邮件
  - 不要求发送方和接收方同时在线
  - 邮件接收方对发送方进行认证
- 单向认证不是完善的安全措施，可以非常容易地冒充验证方，以欺骗被验证方

原始的单向认证技术

1、发送方的用户名和密码通过明文方式传送，易窃听

2、接收方检验用户名和密码，然后进行通信，通信中没有保密

基于密码技术的单向认证

- 不再发送明文的用户名和密码，而是基于“挑战 – 响应”方式
- 符号
  - $f(K_{Alice-Bob}, R)$：使用Alice和Bob的共享秘密、按照某种规则对R做密码变换
    - $K_{Alice-Bob}\{R\}$：使用$K_{Alice-Bob}$作为共享密钥，基于秘密密钥算法对R进行加密
    - $h(K_{Alice-Bob}, R)$：计算R和$K_{Alice-Bob}$的哈希值
  - $[R]_{Alice}$：Alice使用私钥对数据R签名
  - $\{R\}_{Alice}$：使用Alice的公钥对数据R加密

基于共享秘密的单向认证技术 - 1

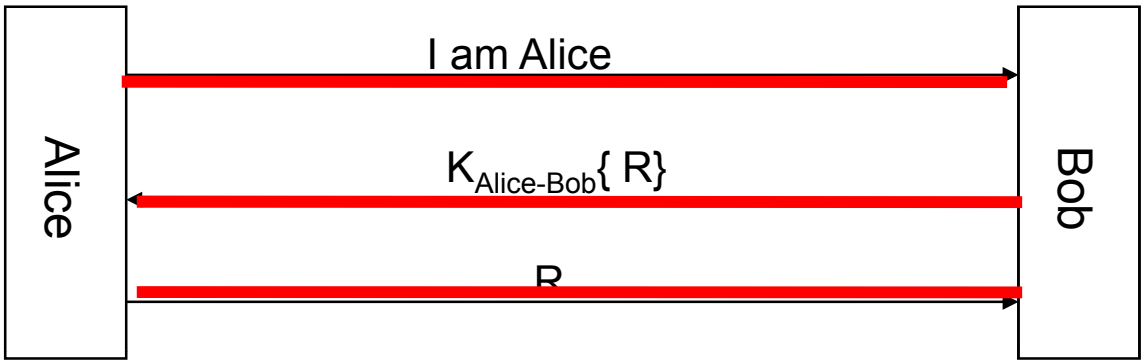
1、侦听者可以看到R和$f(K_{Alice-Bob}, R)$，但是不能计算出$K_{Alice-Bob}$

2、不要求 f 可逆，因此 f 可以是一个哈希函数

3、侦听者掌握R和$f(K_{Alice-Bob}, R)$后，可以进行离线口令猜解

4、攻取Bob的数据库，则可以冒充Alice

基于共享秘密的单向认证技术 - 2



1、侦听者可以看到R和$f(K_{Alice-Bob}, R)$，但是不能计算出$K_{Alice-Bob}$

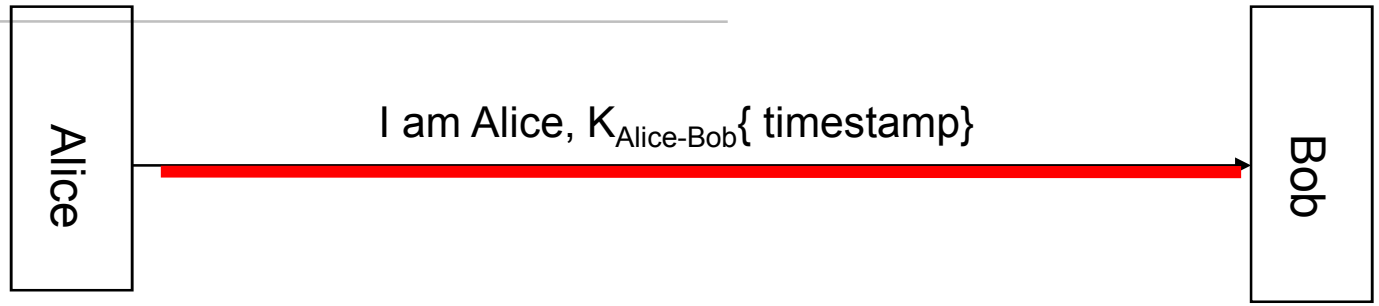
2、如果R是一个有格式的数据且具有有限时效，则Bob可以认证Alice ，防止假冒

3、要求可逆的密码算法

4、侦听者掌握R和$f(K_{Alice-Bob}, R)$后，可以进行离线口令猜解

5、攻取Bob的数据库，则可以冒充Alice

基于共享秘密的单向认证技术 - 3



1、timestamp是时戳（当前时间）。Bob解密结果，当结果在一定时差内，则Alice通过认证

2、高效：只需要一条消息

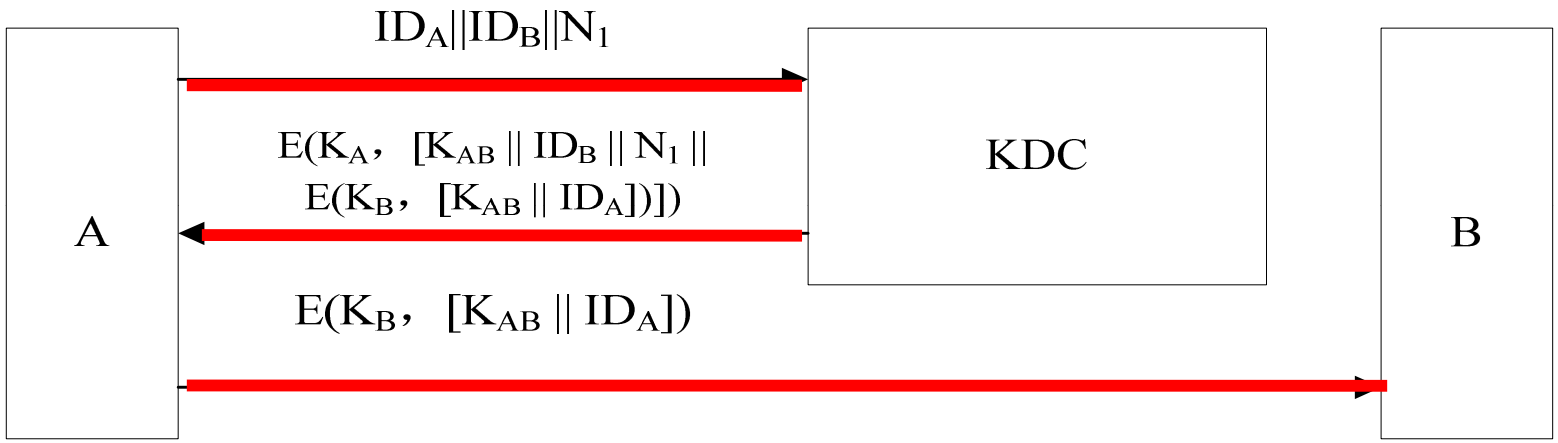
3、要求Alice和Bob有同步的时钟

4、攻击者若能够调整Bob的时间，则可以重新使用以前侦听到的加密时戳

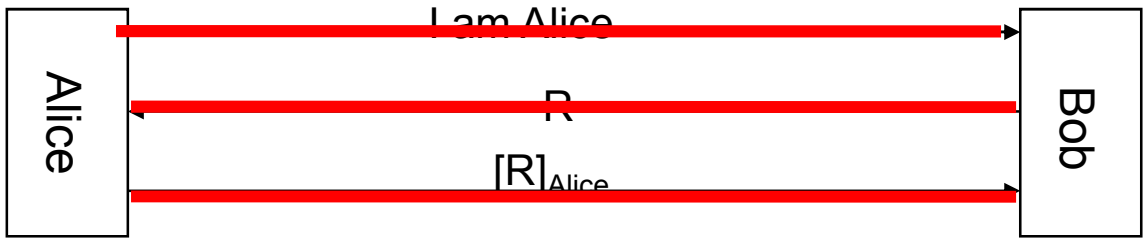
5、动作迅速的攻击者可以利用侦听的加密时戳在另一台机器上冒充Alice

6、攻取Bob的数据库，则可以冒充Alice

基于共享秘密的单向认证技术 - 4



基于公钥体制的单向认证技术 - 1

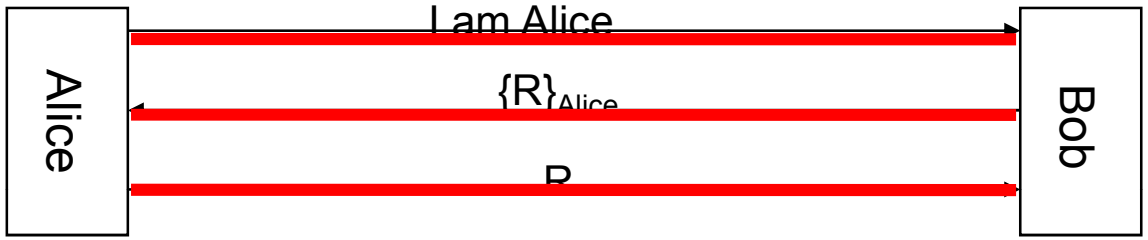


1、Alice对数据R用自己的私钥签名，Bob用Alice的公钥检验签名

2、侦听者无法冒充Alice，即使他攻取了Bob的数据库

3、可以诱骗Alice对特定数据的签名

基于公钥体制的单向认证技术 - 2



1、Bob用Alice的公钥加密数据R，并要求Alice解密

2、侦听者无法冒充Alice，即使他攻取了Bob的数据库

3、如果想知道其他人发送给Alice的加密信息，可以诱骗Alice解密

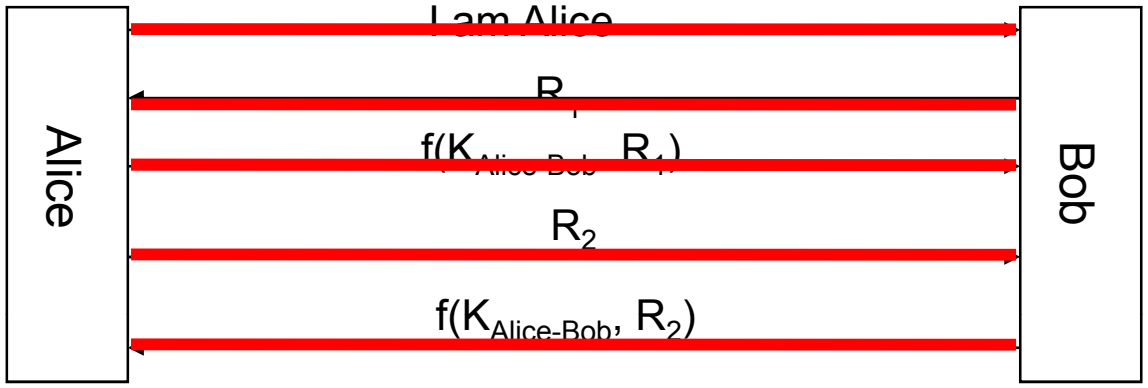
02.3
Part

双向认证

双向认证

- **用于通信双方的相互认证**
- **认证的同时可以协商会话密钥**

基于共享秘密的双向认证 - 1

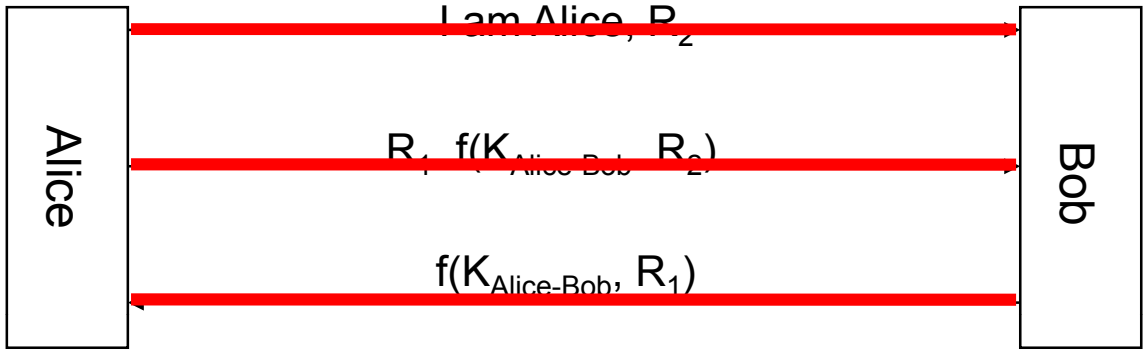


1、基于“挑战 – 响应”方式

2、双方使用共享的秘密对数据进行密码变换，实现对通信对方的认证

3、效率不高：消息过多

基于共享秘密的双向认证 - 2

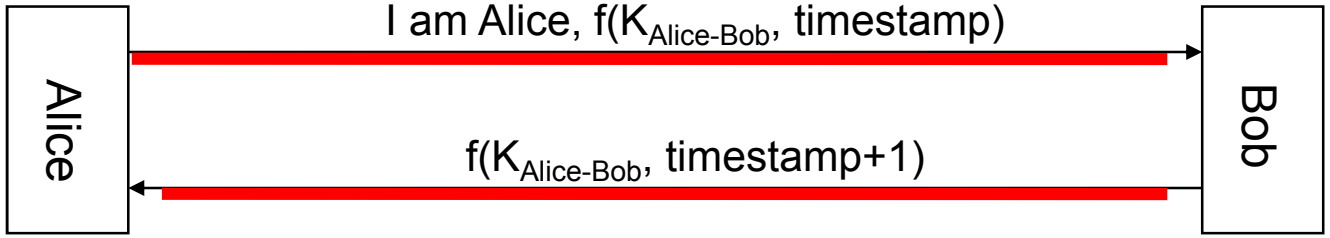


1、基于“挑战 – 响应”方式

2、双方使用共享的秘密对数据进行密码变换，实现对通信对方的认证

3、效率提高：消息减少为三条

基于共享秘密的双向认证 - 3



1、基于“挑战 – 响应”方式，使用时戳（timestamp）作为挑战

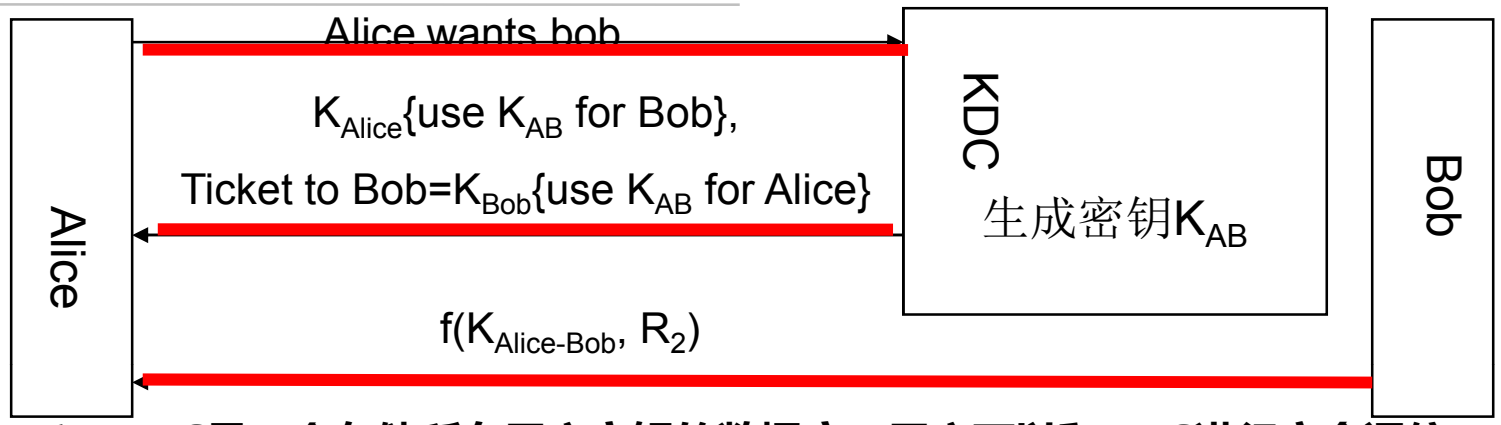
2、双方使用共享的秘密对挑战进行密码变换，实现对通信对方的认证

3、效率提高：消息减少为二条

使用KDC的认证 - 1

- **问题：假设基于秘密密钥技术实现认证**
  - **如果网络上有n个节点，则每个节点都必须知道n-1个密钥。**
  - **每增加一个节点，就必须生成n个新的密钥。密钥分发非常困难**
- **解决方法：使用密钥分发中心KDC**
  - **KDC是一个可靠的节点，知道每个节点的密钥，即KDC和每一个节点都共享密钥**
  - **节点希望和其它节点通信，则首先和KDC联系，由KDC分配一个临时的会话密钥**

使用KDC的认证 - 2



1、KDC是一个存储所有用户密钥的数据库。用户可以和KDC进行安全通信

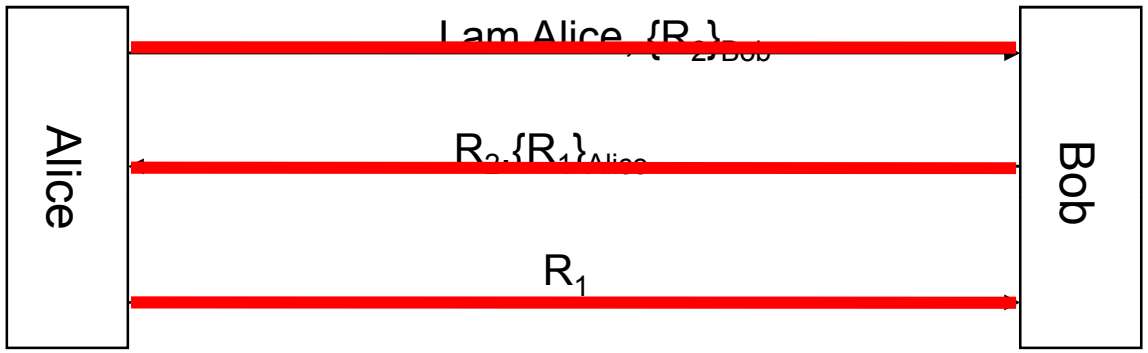
2、Alice请求KDC分配一个会话密钥，以与Bob通信

3、KDC生成一个会话密钥K_{AB}，并用Alice的密钥加密发给Alice。同时，KDC还用Bob的密钥加密K_{AB}，交给Alice转发。

4、Alice解密得到K_{AB}，并把用Bob的密钥加密的K_{AB}发送给Bob

5、Alice和Bob可以基于K_{AB}实现双向认证

基于公钥技术的双向认证



1、基于“挑战 – 响应”方式

2、双方使用对方的公钥加密数据，要求对方加密，实现对通信对方的认证

03
Part

基于对称密钥的认证系统 Kerberos

Kerberos背景

- 在MIT的Athena项目中开发的一种认证服务
- 试图解决如下问题
  - 在公用网络中，用户通过工作站访问网络服务，这些服务是由分布在网络中的服务器提供的。
  - 服务器能够对用户的每一项服务请求进行认证
    - 仅仅依赖工作站对用户的认证是不够的
    - 用户访问每一种网络服务，都需要向服务器证明其身份

Kerberos背景

- **安全威胁：工作站无法保证用户的身份真实性**
  - **非法用户访问某个工作站，并假冒另一个合法用户**
  - **非法用户更改工作站网络地址，假冒另一个工作站**
  - **非法用户窃听消息交换过程，并实施重放攻击**
- **目标：在各种情况下，都能防止用户对服务的非授权访问**

Kerberos认证服务

- Kerberos：一种基于对称密钥、在网络上实施身份认证的服务
  - 身份认证作为网络上一种标准的安全服务来提供
  - 能够实现用户和服务器之间的双向认证
  - 集中式的认证服务
    - 通过运行在网络中某个安全节点的密钥分发中心（KDC，又称为认证服务器）提供认证服务。
    - 用户能够用用户名和口令登录工作站，工作站使用用户名和密码与KDC联系，代替用户获得使用远程资源所需要的信息
- Kerberos v4 and Kerberos v5
  - 相互竞争市场，v4用户量大

Kerberos特征

- **提供一种基于可信第三方的认证服务**
  - KDC作为第三方，若用户与服务器都信任KDC，则Kerberos就可以实现用户与服务器之间的双向鉴别。如果KDC是安全的，并且协议没有漏洞，则认证是安全的
- **安全性**
  - 能够有效防止攻击者假冒合法用户
- **可靠性**
  - Kerberos服务自身可采用分布式结构，KDC之间互相备份
- **透明性**
  - 用户只需要提供用户名和口令，工作站代替用户实施认证的过程
- **可伸缩性**
  - 能够支持大量用户和服务器

4.2.1 Kerberos版本4

- 基本的第三方认证方案
- 认证服务器（AS）
  - 用户最初与AS协商以识别自己
  - AS提供不可破坏的身份认证凭据（票据授权票据TGT）
- 票据授权服务器（TGS）
  - 用户随后基于用户TGT请求从TGS访问其他服务
- 使用DES的复杂协议

4.2.1 Kerberos版本4

- **一个简单的认证会话1.0**
  - **消息（1）C→AS：$ID_c||P_c||ID_v$**
  - **消息（2）AS→C：Ticket**
  - **消息（3）C→V：$ID_c||$ Ticket ，**

    Ticket =E(K_v，[$ID_c||AD_c||ID_v$])

其中：

C=客户端；

AS=认证服务器；

V=服务器；

ID_c =客户端上用户的身份标识；

ID_v =服务器的身份标识；

P_c=客户端上用户的口令；

V=服务器；

ID_c =客户端的网络地址；

K_v=认证服务器和服务器间共享的加密密钥。

Ticket=票据。

认证会话1.0存在问题

- 希望用户输入口令次数最少
  - 访问同一服务器
  - 访问不同不同服务器
- 口令明文传输
- 解决方案
  - 避免口令明文传输
  - 票据授权服务器

4.2.1 Kerberos版本4

- **一个更安全的认证会话2.0**
  - 为避免明文传输密钥的方案和一个称为票据授权服务器（TGS）的新服务
  - 每次用户登录会话就执行一次：
    - 消息（1）C→AS：$IDc||ID_{tgs}$
    - 消息（2）AS→C：$E(K_c, Ticket_{tgs})$
  - 每种类型的服务执行一次：
    - 消息（3）C→TGS：$ID_c||AD_v||Ticket_{tgs}$
    - 消息（4）TGS → C：$Ticket_v$
  - 每个服务会话执行一次：
    - 消息（5）C→V：$ID_c||Ticket_v$
  - $Ticket_{tgs}=E(K_{tgs}, [ID_c||AD_c||ID_{tgs}||TS_1||Lifetime_1])$
  - $Ticket_v=E(K_v, [ID_c||AD_c||ID_v||TS_2||Lifetime_2])$

其中：

ID_{tgs}=票据授权服务器的身份标识；

K_c=用户口令生成的密钥；

$Ticket_{tgs}$=票据授权票据；

$Ticket_v$=服务授权票据；

认证会话2.0存在问题

- 授权票据的生命周期
  - 太短：多次输入口令
  - 太长：攻击者等待用户退出
- 伪造合法的网络地址
- 票据使用者和所有者一致
- 验证服务器身份(发给加密的服务器）

认证会话2.0改进

- **改进方案**
  - **增加时间戳TS**
  - **增加会话密钥（$k_{c,tgs}$客户端与TGS、客户端与服务器$k_{c,v}$）**
  - **双向认证**
- **思路**
  - **AS分配$K_{c,tgs}$会话密钥，由客户端转交给TGS**
  - **TGS分配$K_{c,v}$会话密钥，由客户端转交给服务器**

Kereros 认证服务交换

- **Step 1:获取票据授权票据**
  - (1) C → AS $ID_c \| ID_{tgs} \| TS_1$
  - (2) AS → C $E(K_c, [K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}])$

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

Kereros 票据授权服务交换

- **Step 2:获得服务授权票据**
  - (3) C → TGS $ID_v \| Ticket_{tgs} \| Authenticator_c$
  - (4) TGS → C $E(Kc,tgs, [Kc,v \| IDv \| TS_4 \| Ticket_v])$

$Ticket_{tgs} = E(Kt_{gs}, [K_{c,tgs} \| ID_C \| ADC \| ID_{tgs} \| TS_2 \| Lifetime_2])$

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

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

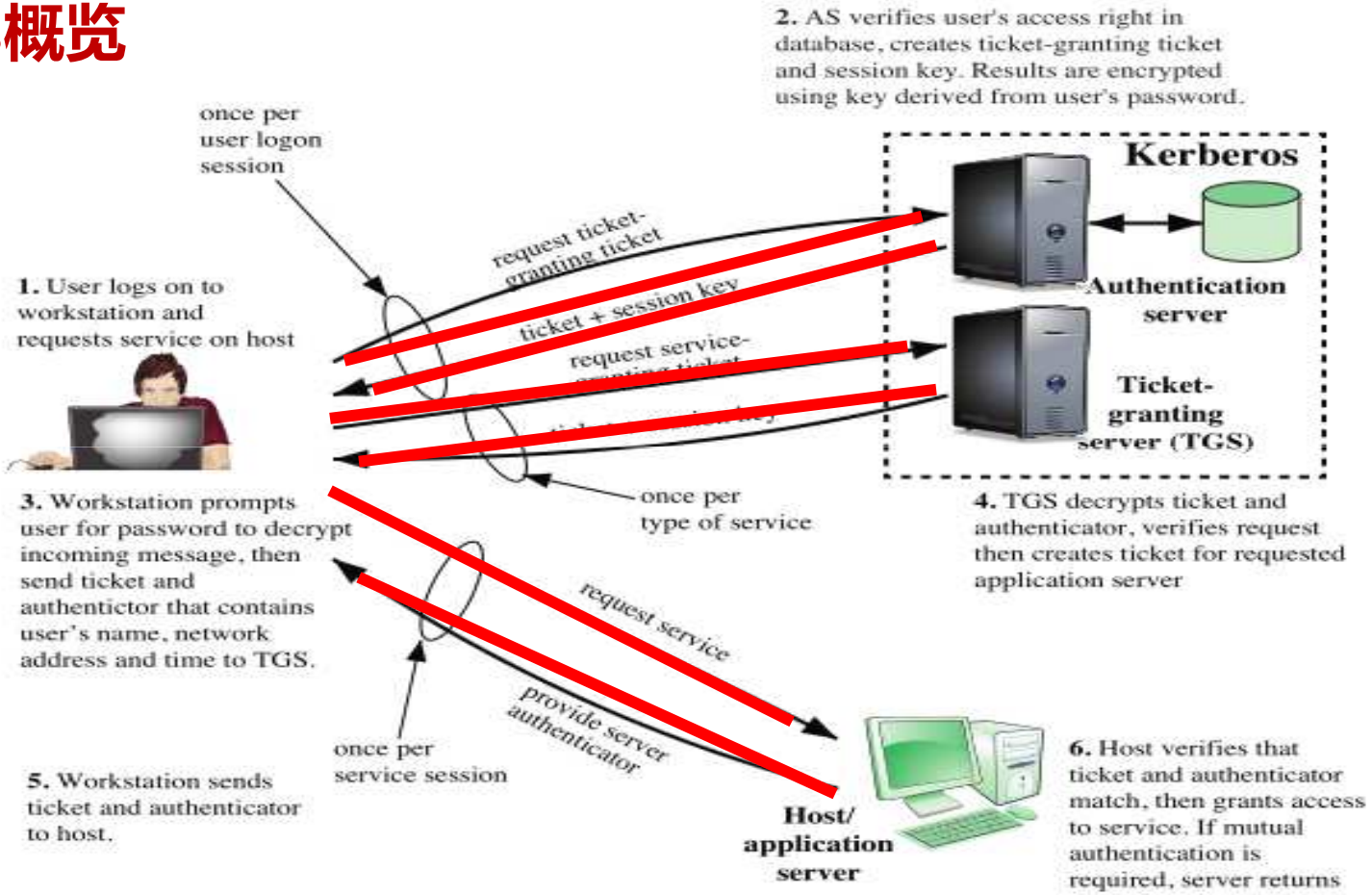
Kereros 客户端/服务器认证交换

- **Step 3:获得服务**
  - (5) C → V $Ticket_v \| Authenticator_c$
  - (6) V → C $E(K_{c,v}, [TS_5 + 1])$ (for mutual authentication)

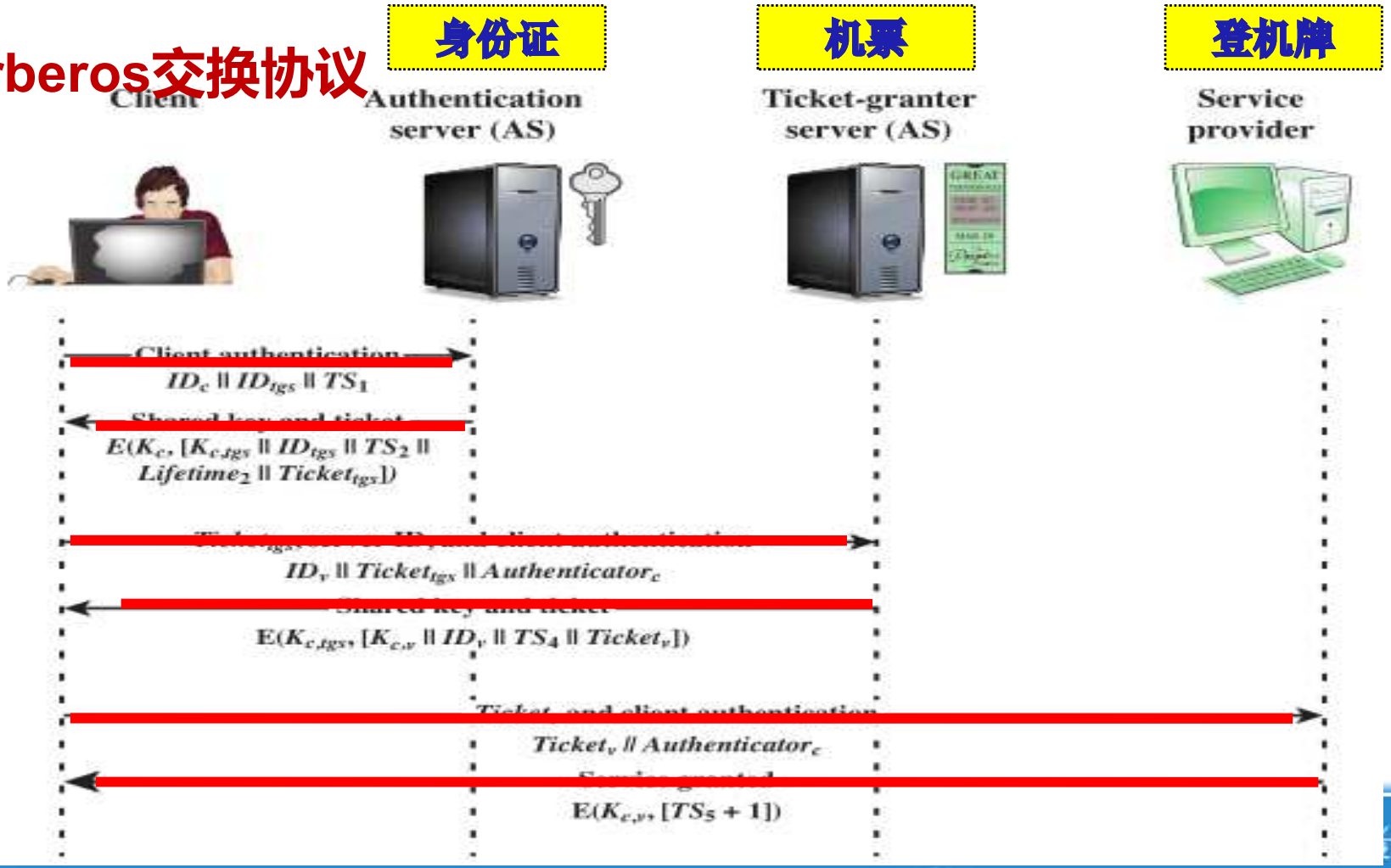
$Ticket_v = E(K_v, [K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4])$

$Authenticator_c = E(K_{c,v}, [ID_C \| AD_C \| TS_5])$

Kerberos概览


2. AS verifies user's access right in database, creates ticket-granting ticket and session key. Results are encrypted using key derived from user's password.
once per user logon session
Kerberos
request ticket-granting ticket
ticket + session key
request service-
Authentication server
Ticket-granting server (TGS)
1. User logs on to workstation and requests service on host
once per type of service
4. TGS decrypts ticket and authenticator, verifies request then creates ticket for requested application server
3. Workstation prompts user for password to decrypt incoming message, then send ticket and authentictor that contains user's name, network address and time to TGS.
request service
provide server authenticator
once per service session
5. Workstation sends ticket and authenticator to host.
Host/ application server
6. Host verifies that ticket and authenticator match, then grants access to service. If mutual authentication is required, server returns an authenticator.

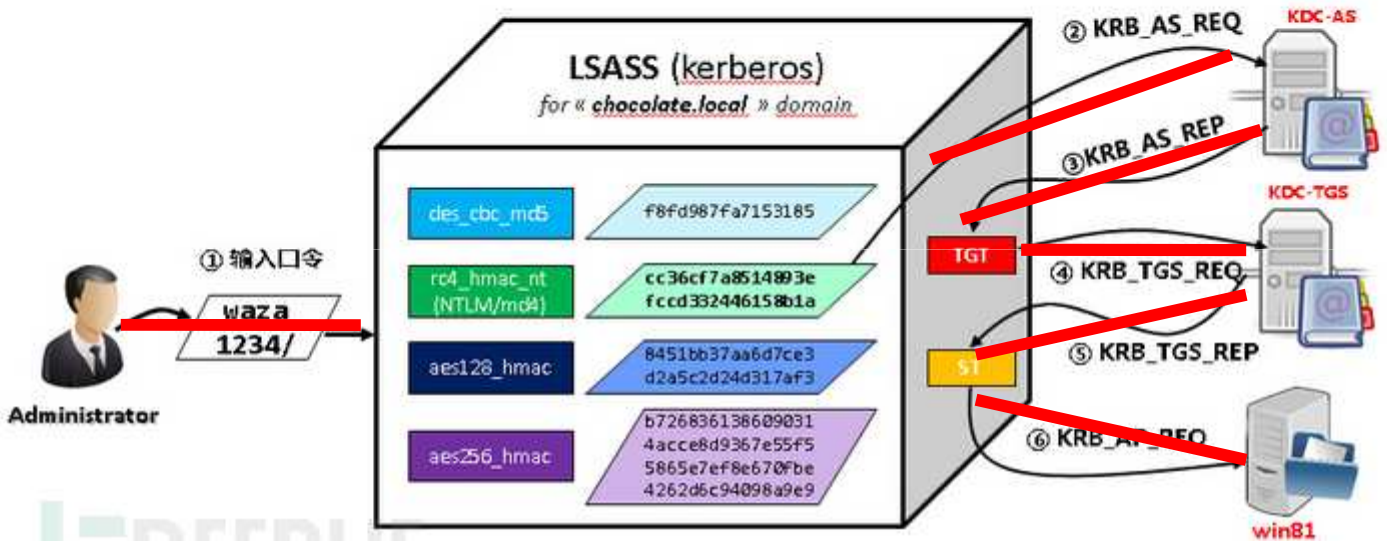
Kerberos交换协议



Kerberos配置

- Kerberos服务器称为KDC
- 每个实体都和KDC共享一个秘密密钥，称为该实体的主密钥。
  - KDC有一个记录实体名字和对应主密钥的数据库。这个数据库用KDC的主机密钥K_{KDC}进行加密。
- 用户使用用户名和口令登录工作站，工作站代替用户向KDC申请访问网络服务的票据。用户主密钥由口令生成。
- 用户只需记住口令，其它网络设备需要记住自己的主密钥。
- 算法：DES

Windows Kerberos实例



认证步骤

- 1.客户端对用户口令执行散列运算。此散列值（即用户密钥）成为客户端和KDC共享的长期密钥（**long term key**）
- **2.KRB_AS_REQ-**客户端加密一个时间戳，然后发送给身份验证服务AS

认证步骤

- **3.KRB_AS_REP-**身份验证服务会解密时间戳，若解密成功，表明了客户端获得某个特定用户的口令（即验证了用户的身份）。AS向客户端回复两条信息：
- ①短期会话密钥，用于客户端向KDC发起后续的请求，该消息经客户端的长期密钥加密。（此短期会话密钥仅适用于该客户端和KDC之间）
- ②票据授权票据（Ticket Granting Ticket，简称**TGT**），包含有关用户名、域名、时间和组成员资格等信息。该消息经仅可知KDC的密钥加密（在Windows环境中为krbtgt账户的NT-Hash）。**记住KDC不记录状态：客户端每次请求访问一项服务时，TGT都会被转发。**

认证步骤

- 4.**KRB_TGS_REQ**-客户端使用AS返回的会话密钥构建访问特定服务的请求。客户端把TGT连同请求一起发送到票据授予服务。
- 5.**KRB_TGS_REP**-票据授予服务解密TGT和服务请求，然后如果请求被允许，票据授予服务向客户端发送一个服务票据（Service Ticket，简称ST），包括两个部分：
- ①远程服务器的部分-包含请求用户的组成员资格、时间戳、用于客户端和远程服务器之间通信的会话密钥。使用远程服务器和KDC共享的长期密钥加密这部分消息。
- ②客户端的部分-包含用于客户端和远程服务器之间通信的会话密钥。使用步骤3中AS回复的短期会话密钥加密这部分消息。

认证步骤

- 6.**KRB_AP_REQ**-客户端把服务票据中的服务器部分和请求一起发送到远程服务器。远程服务器将直接接受该服务器票据，并不需要和KDC直接通信，因为该票据是用远程服务器和KDC共享的长期密钥加密过的。解密成功即表明KDC已经允许了此次通信。

03.2
Part

Kerberos扩展

Kerberos域

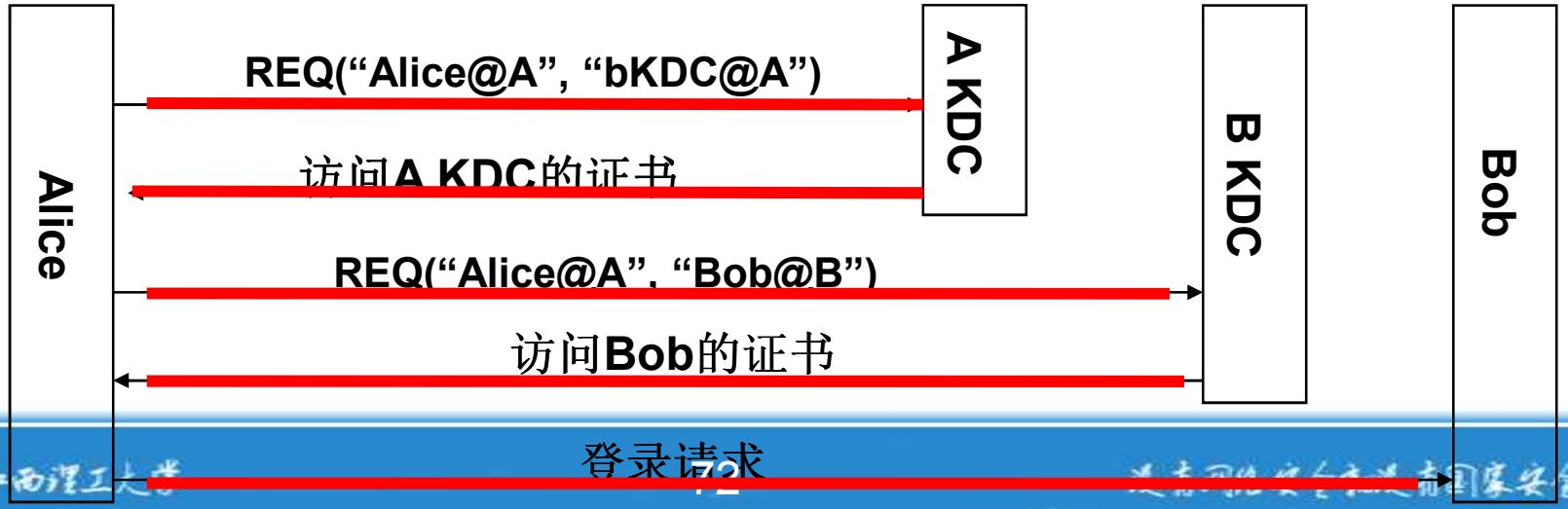
- 一组共享相同Kerberos数据库的受管节点
- Kerberos数据库驻留在Kerberos主计算机系统上，该系统应保存在物理安全的房间中
- Kerberos数据库的只读副本也可能驻留在其他Kerberos计算机系统上
- 必须在主计算机系统上对数据库进行所有更改
- 更改或访问Kerberos数据库的内容需要Kerberos主密码

Kerberos域

- **网络上的实体被划分成不同的域**
  - **每个域都有自己的KDC服务器**
  - **所有用户和服务/资源都在KDC上注册**
  - **如果域中有多个KDC，则它们是等价的：拥有相同的KDC主密钥，拥有相同的数据库**
- **不同域的KDC拥有不同的KDC主密钥以及不同的数据库，因为它们是为不同的实体集合服务的**

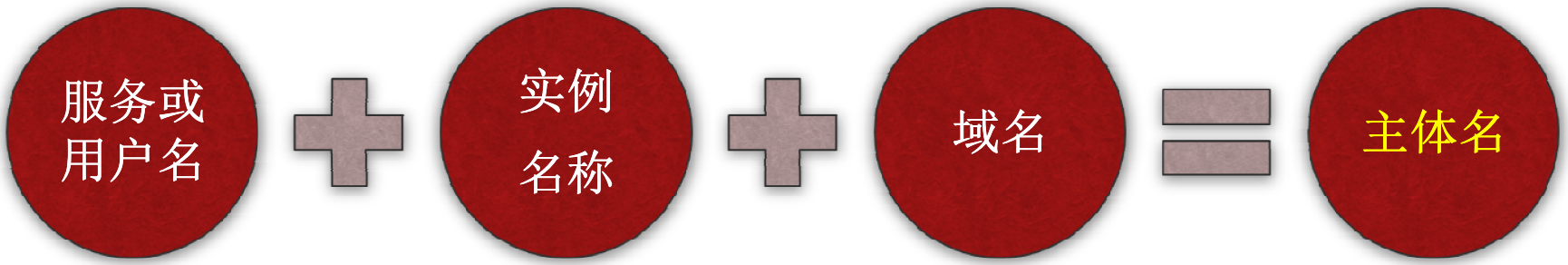
域间认证

- 问题：某个域中的实体需要认证另外一个域中的实体的身份
- 方法：域B的KDC可以在域A中注册成为一个实体，这使得域A中的用户可以访问域B的KDC，就像访问域A中其它资源一样。



Kerberos主体

- Kerberos系统已知的服务或用户
- 每个Kerberos主体都由其主体名标识



主要名称由三部分组成

Kerberos v4的缺陷

- 依赖性
  - 对IP协议的依赖性和对时间依赖性
- 门票有效期一般为5分钟到21小时，往往不能满足需求
- 域间的鉴别、管理困难
- 非标准的DES加密，易受攻击
- 未对用户口令进行额外保护，易受攻击

Kerberos v5 的改进

- 支持任何加密技术
- 除IP协议以外，还支持其它通信协议
- 票据有效期：任意长度
- 具有鉴别转发能力
- 更有效的方法来解决域间认证问题
- 提供一种预鉴别机制，使得口令攻击更困难

4.2.2 Kerberos版本5

表 4.3 Kerberos版本5消息交换总结

(1) C → AS $Options \| IDc \| Realmc \| IDtgs \| Times \| Nonce1$

(2) AS → C $Realmc \| IDC \| Tickettgs \| E(Kc, [Kc,tgs \| Times \| Nonce1 \| Realmtgs \| IDtgs])$

$Ticket_{tgs} = E(K_{tgs}, [Flags \| Kc,tgs \| Realmc \| IDC \| ADC \| Times])$

(a) 用于获取票据授权票据的认证服务交换

表 4.3 Kerberos版本5消息交换总结

(3) C → TGS *Options* || *IDv* || *Times* || || *Nonce2* || *Tickettgs* || *Authenticatorc*

(4) TGS → C *Realmc* || *IDC* || *Ticketv* || E(*Kc,tgs*, [*Kc,v* || *Times* || *Nonce2* || *Realmv* || *IDv*])

Tickettgs = E(*Ktgs*, [*Flags* || *Kc,tgs* || *Realmc* || *IDC* || *ADC* || *Times*])

Ticketv = E(*Kv*, [*Flags* || *Kc,v* || *Realmc* || *IDC* || ADC || *Times*])

Authenticatorc = E(*Kc,tgs*, [*IDC* || *Realmc* || *TS*1])

(b)用于获得服务授权票据的票据授权服务交换

表 4.3 Kerberos版本5消息交换总结

(5) C → V　*Options* || *Ticketv* || *Authenticatorc*

(6) V → C　E*Kc,v* [*TS2* || *Subkey* || *Seq#*]

Ticket*v* = E(*Kv*, [Flags || *Kc,v* || *Realmc* || *IDC* || *ADC* || *Times*])

Authenticatorc = E(*Kc,v*, [*IDC* || *Realmc* || *TS2* || *Subkey* || *Seq#*])

(c)为获得服务而进行的客户端/服务器认证交换

04
Part

公钥证书X.509

背景

- A公司的张三先生要到B公司去拜访, 但是B公司的所有人都不认识他
- 常用的办法是带公司开的一封介绍信, 并在信上敲上A公司的公章
- 张三先生到了B公司后, 把介绍信给B公司的前台李四
- 李四一看介绍信上有A公司的公章, 就相信张三先生不是歹人了


考籍专用章

江苏省高等教育自学考试

转考介绍信

山东省 自学考试办公室:

[redacted] 同志因工作调动，现转往你处（淄博市），

继续参加自学考试，请接洽。

涂州自考网 xzzkw.com

江苏省高等教育自学考试办公室

考籍专用章

二〇一二年七月二十五日

注：考生合格成绩另转

背景

- 和B公司有业务往来的公司很多, 每个公司的公章不同
- 李四要能分辨各种公章, 非常麻烦
- 中介公司C, 专门开设了一项"代理公章"的业务
  - 张三先生去B公司, 需要带2封介绍信
  - 介绍信1: 含有C公司的公章和A公司的公章, 且注明C公司信任A公司
  - 介绍信2:含有A公司的公章, 并包含一些具体内容
- 李四只要记住中介公司C的公章即可
  - 先检查介绍信1的C公章, 验明正身; 确认无误后,
  - 再对比介绍信1和介绍信2的公章是否一致, 如果一致, 说明介绍信2是可信任的

专业术语

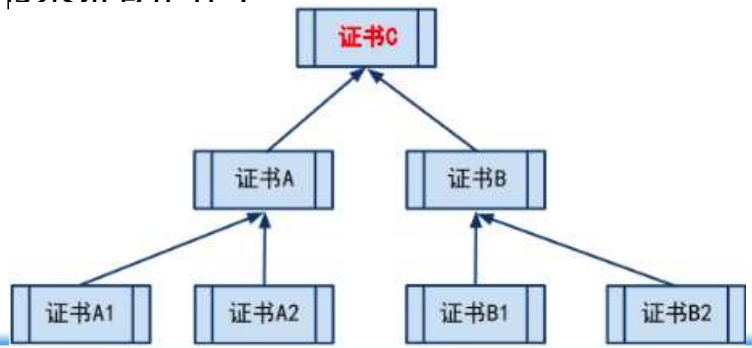
- **证书**
  - 相当于例子中的公章，它证明介绍信确实是A公司发出的
  - 在实际网络通信过程中，一般是用来证明某请求确实是某用户发出的
- **CA**
  - 负责管理和签发证书的第三方机构，相当于例子中的C公司
  - 一般来说，CA必须是所有行业和所有公众都信任的，认可的，因此它必须具有足够的权威性

专业术语

- **CA证书**
  - CA颁发的证书
  - 在例子中, 它用来证明介绍信1确实是CA发出的
  - CA证书中, 包含要CA证书来证明的实体的公钥
  - 如例子中, 介绍信1中包含公司A的公章，接收者可以用介绍信1中的公章来对比介绍信2中的公章是否相同.
- **证书之间的信任关系**
  - 例子中, 介绍信1注明了公章C信任公章A
  - 就是用一个证书来证明另一个证书是真实可信的

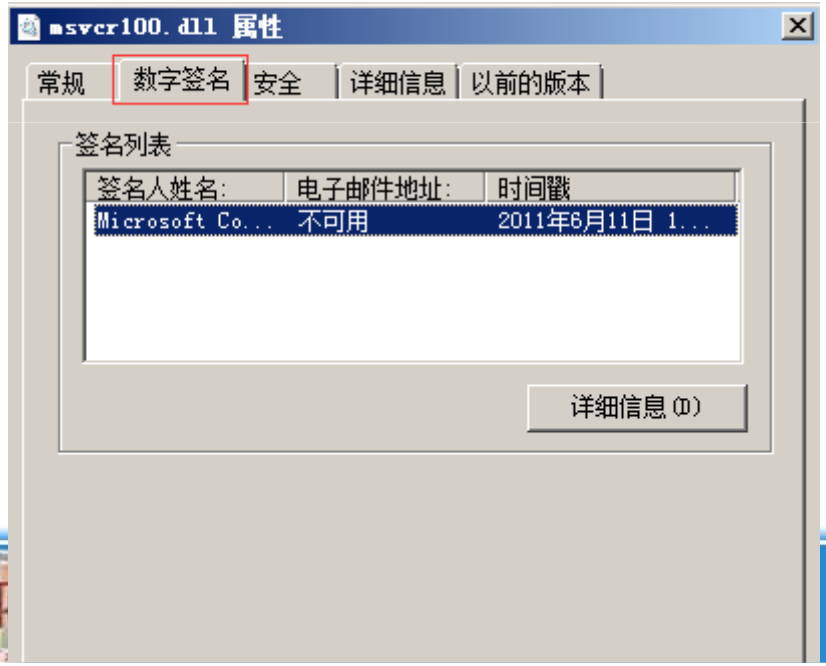
专业术语

- **根证书**
  - 假设C证书信任A证书和B证书, A证书信任A1证书和A2证书, B证书信任B1证书和B2证书. 则它们之间, 构成了一个树形关系
  - 处于树根位置的证书, 就是根证书
  - 除了根证书, 其他证书都要依靠上一级的证书来证明自己.
  - 根证书, 是自己证明自己可靠



证书的作用

- **验证网站是否可信（针对HTTPS）**
- **验证某文件是否可信（是否被篡改）**



专业术语

- **数字签名**
  - 数字签名是将摘要(digest1)用A(发送者)的私钥加密, 与原文(Context)一起传送给B(接收者)
  - B只有用A的公钥才能获得摘要(digest1), 然后用Hash函数对收到的原文(Context)进行计算, 得到另一个摘要(digest2)
  - 对比digest1和digest2. 如果相同, 说明收到的信息未被修改, 否则被修改过

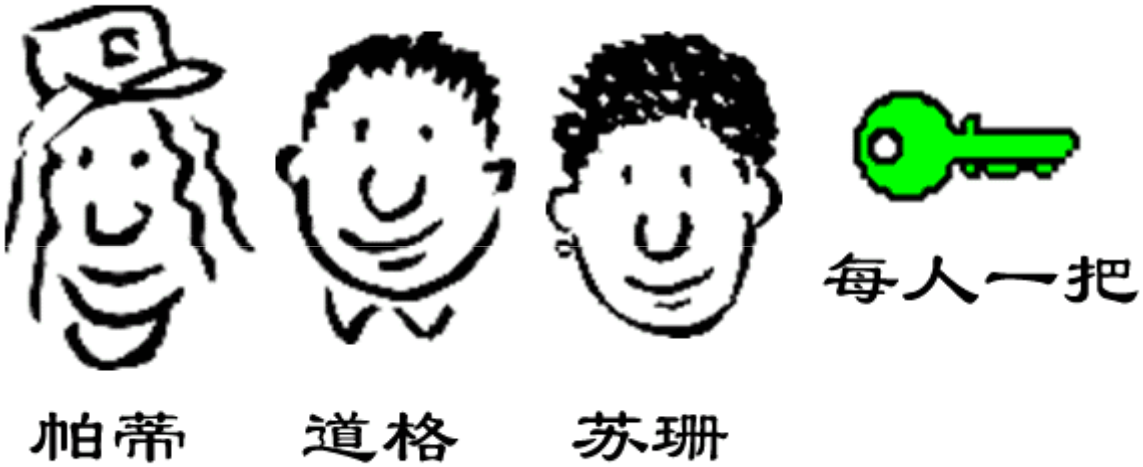
白话数字签名

鲍勃

鲍勃的公钥

鲍勃的私钥

每人一把



鲍勃的公钥加密

苏珊

"Hey Bob, how about lunch at Taco Bell. I hear they have free refills!"

公钥加密

HNFmsEm6Un
BejhhyCGKO
KJUxhiygSBC
EiC0QYIh/Hn
3xgiKBcyLK1
UcYiYlxx2lCF
HDC/A

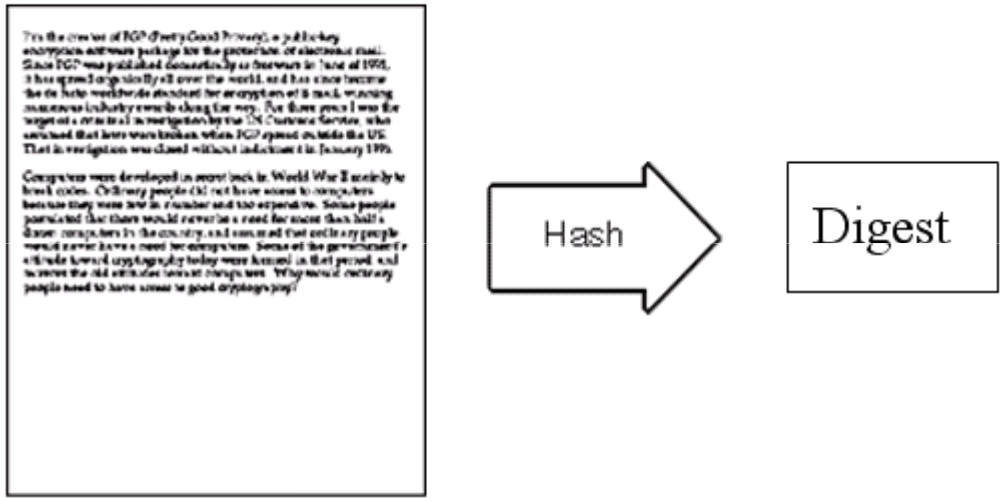
私钥解密

HNFmsEm6Un
BejhhyCGKO
KJUxhiygSBC
EiC0QYIh/Hn
3xgiKBcyLK1
UcYiYlxx2lCF
HDC/A

私钥解密

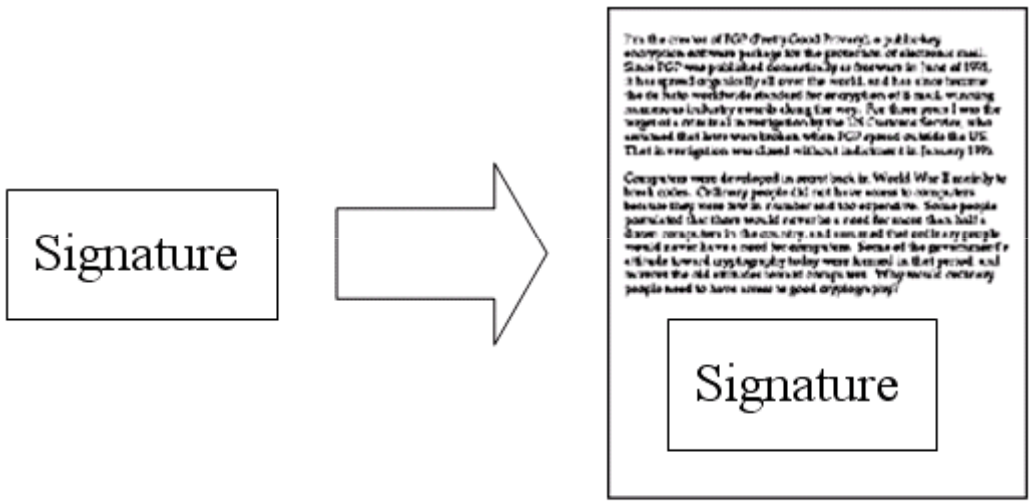
"Hey Bob, how about lunch at Taco Bell. I hear they have free refills!"

回信 **数字签名**



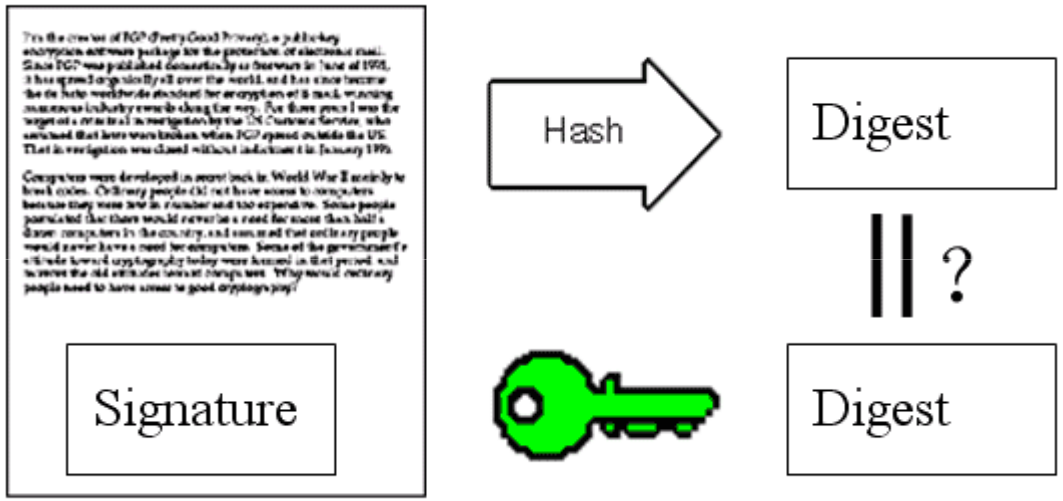
摘要加密

一起发给苏珊



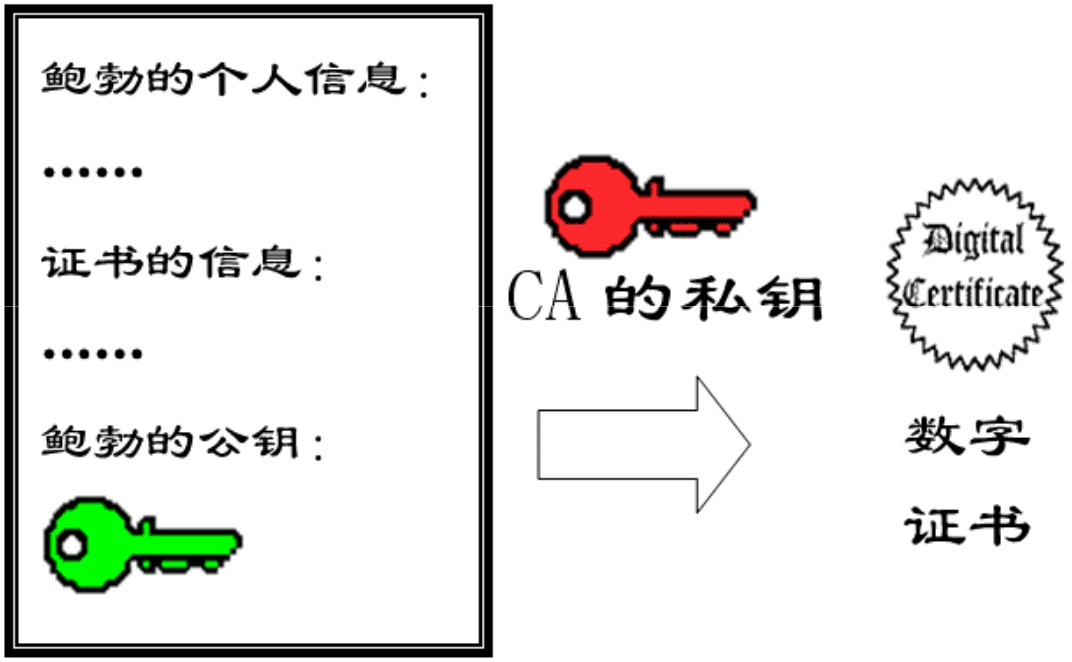
公钥解密

摘要进行对比



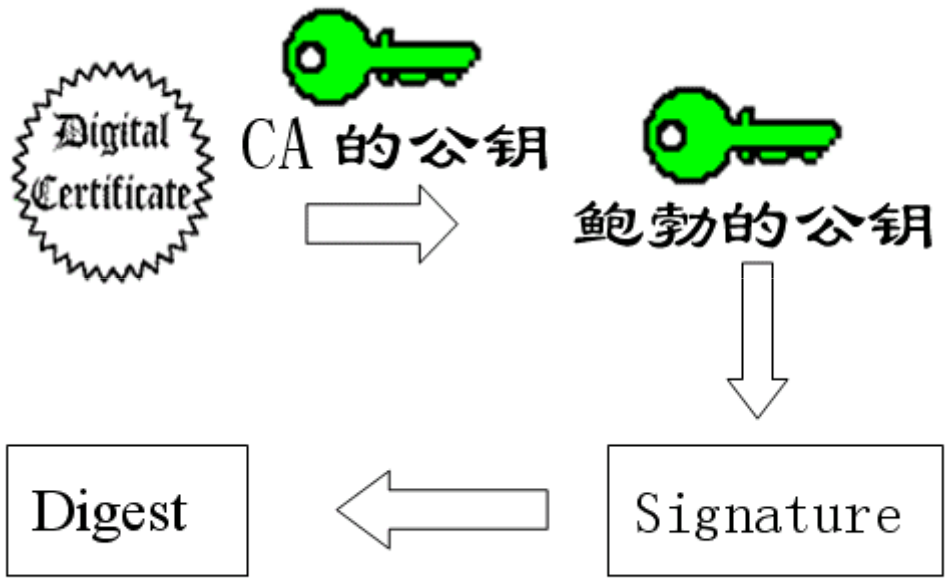
道格就冒充鲍勃

公钥做认证



附上数字证书

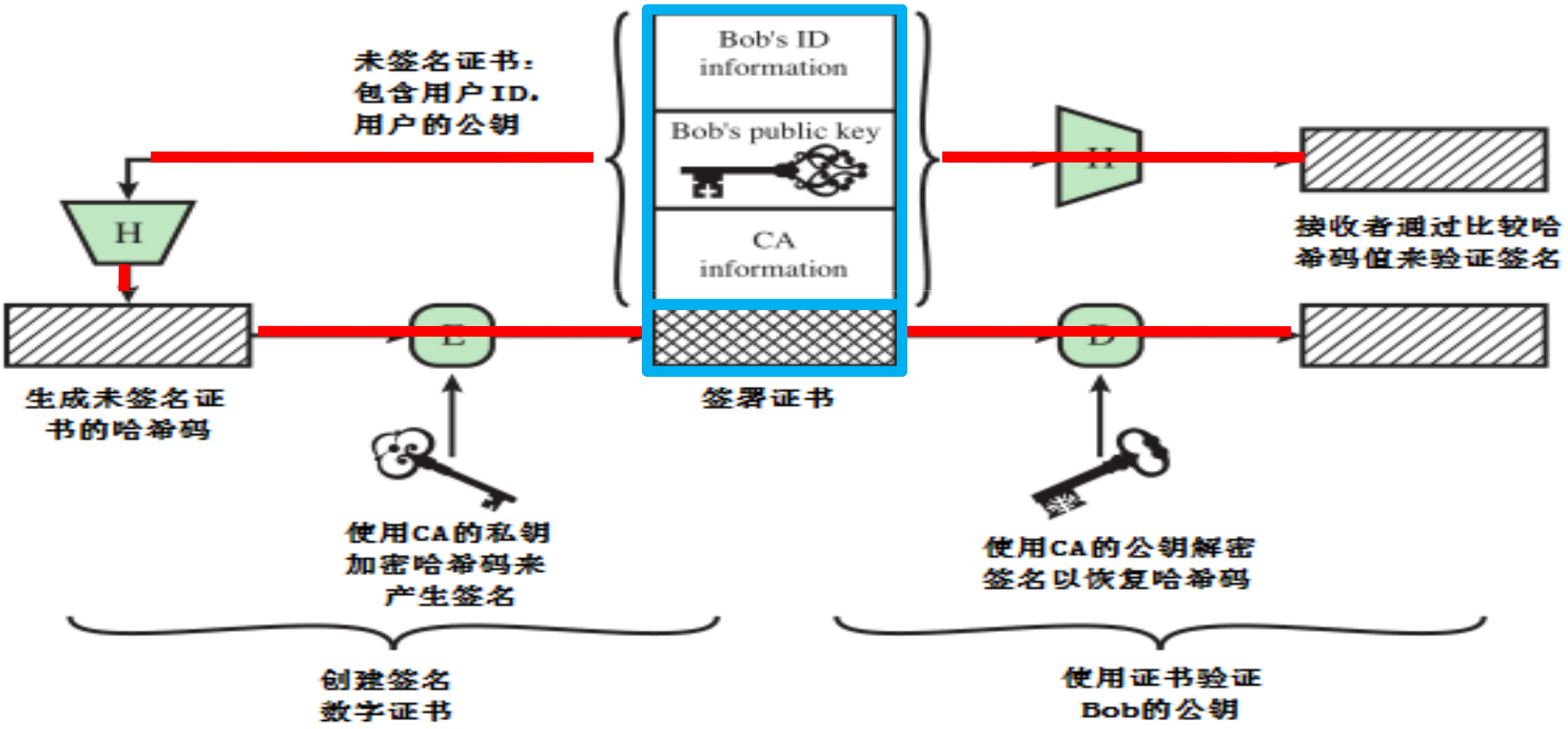
证明”数字签名”



4.3.1 公钥证书

- **公钥证书由公钥加上公钥所有者的用户ID以及可信任的第三方签名的整个数据块组成。通常，第三方是由用户团体所信任的认证中心（CA），如政府机构或金融机构。**
  - **用户可以通过安全的渠道把他的公钥提交给这个CA，获得证书**
  - **然后，用户可以发布证书**
  - **需要用户的公钥的任何人都可以获取证书，并通过附加可信签名验证它是否有效**
- **人们广泛接受的公钥证书是X.509标准**
  - **X.509证书应用于大多数的网络安全设施，包括IP安全、安全套接字层（SSL）、安全电子交易（SET）和S/MIME**

公钥证书的使用


未签名证书：
包含用户ID，
用户的公钥
Bob's ID information
Bob's public key
CA information
H
H
接收者通过比较哈希码值来验证签名
生成未签名证书的哈希码
E
D
签署证书
使用CA的私钥加密哈希码来产生签名
使用CA的公钥解密签名以恢复哈希码
创建签名数字证书
使用证书验证Bob的公钥

4.4 X.509证书

- ITU-T建议X.509是定义目录服务的X.500系列建议的一部分
- 定义X.500目录向其用户提供身份认证服务的框架
- 该目录可以充当公钥证书的存储库
- X.509证书定义了另一个使用公钥证书的认证协议。最初发布于1988年；基于公钥加密和数字签名的使用。

X.509简介

- X.509是由国际电信联盟（ITU-T）制定的关于数字证书结构和认证协议的一种重要标准
- 解决“在公用网络中提供用户目录信息服务”的问题
  - 目录是指管理用户信息数据库的服务器或一组分布服务器，用户信息包括用户名到网络地址的映射等用户信息或其他属性
- X.509是基于公钥密码体制和数字签名的服务。其标准中并未规定使用某个特定的算法，但推荐使用RSA；其数字签名需要用到散列函数，但并没有规定具体的散列算法。

4.4.1 证书

- X.509的核心是与每个用户相关的公钥证书。
  - 所谓证书就是一种经过签名的消息，用来确定某个名字和某个公钥的绑定关系。
- 这些用户证书由一些可信的认证中心(CA)创建并被CA或用户放入目录服务器中
- 目录服务器本身不创建公钥和证书，仅仅为用户获得证书提供一种简单的存取方式

数字证书的概念

- 数字证书就是互联网通讯中标识通讯各方身份信息的一系列数据，提供了一种在Internet上验证身份的方式，其作用类似于驾驶执照或身份证。
- 数字证书是一个经数字证书授权中心数字签名的包含公开密钥拥有者信息以及公开密钥的文件。最简单的证书包含一个公开密钥、名称以及证书授权中心的数字签名。
- 获得证书的人只要信任这个证书授权中心，就可相信他所获得的证书。

数字证书的安全性

- 证书是公开的，可复制的
- 任何具有证书授权中心（certificate authority, CA）公钥的用户都可以验证证书有效性
- 除了CA以外，任何人都无法伪造、修改证书
- 证书的安全性依赖于CA的私钥

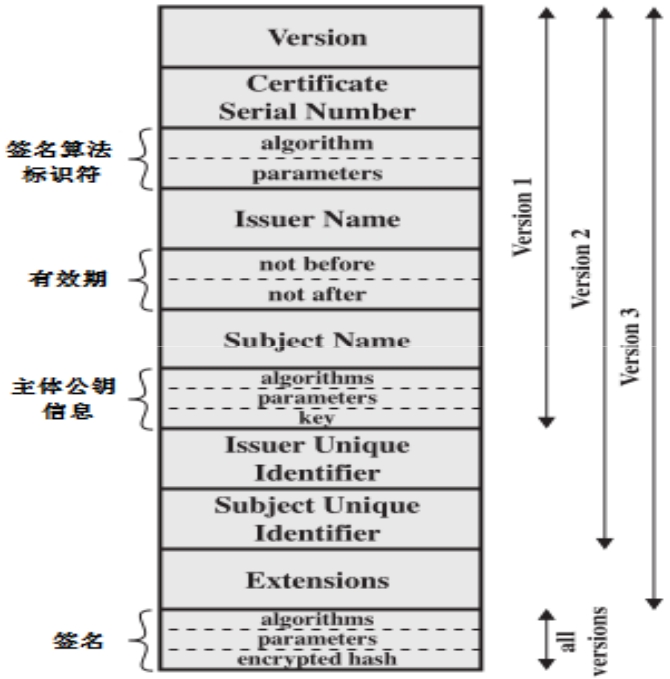
证书格式

- **X.509证书包含以下信息：**
  - **版本号(Version)：目前定义了三个版本，版本1的编号为0，版本2的编号为1，版本3的编号为2。**
  - **序列号(Serial number)：一个整数，和签发该证书的CA名称一起惟一标识该证书。**
  - **签名算法标识(Signature algorithm identifier)：指定证书中计算签名的算法。**
  - **签发者(Issuer name)：创建、签名该证书的CA的X.500格式名字。**
  - **有效期(Period of validity)：包含两个日期，即证书的生效日期和终止日期。**
  - **证书主体名(Subject name)：持有证书的主体的X.500格式名字，证明此主体是公钥的所有者。**

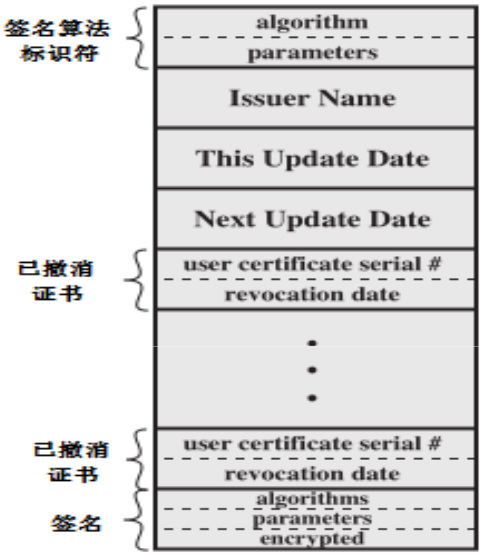
证书格式（续）

- **证书主体的公钥信息(Subject's public-key information)：**主体的公钥以及将被使用的算法标识，带有相关的参数。
- **签发者惟一标识(Issuer unique identifier)：**版本2和版本3中可选的域，用于惟一标识认证中心CA。
- **证书主体惟一标识(Subject unique identifier)：**版本2和版本3中可选的域，用于惟一标识证书主体。
- **扩展(Extensions)：**仅仅出现在版本3中，一个或多个扩展域集。
- **签名(Signature)：**覆盖证书的所有其他域，以及其他域被CA私钥加密后的散列代码

X.509格式



(a) X.509 证书

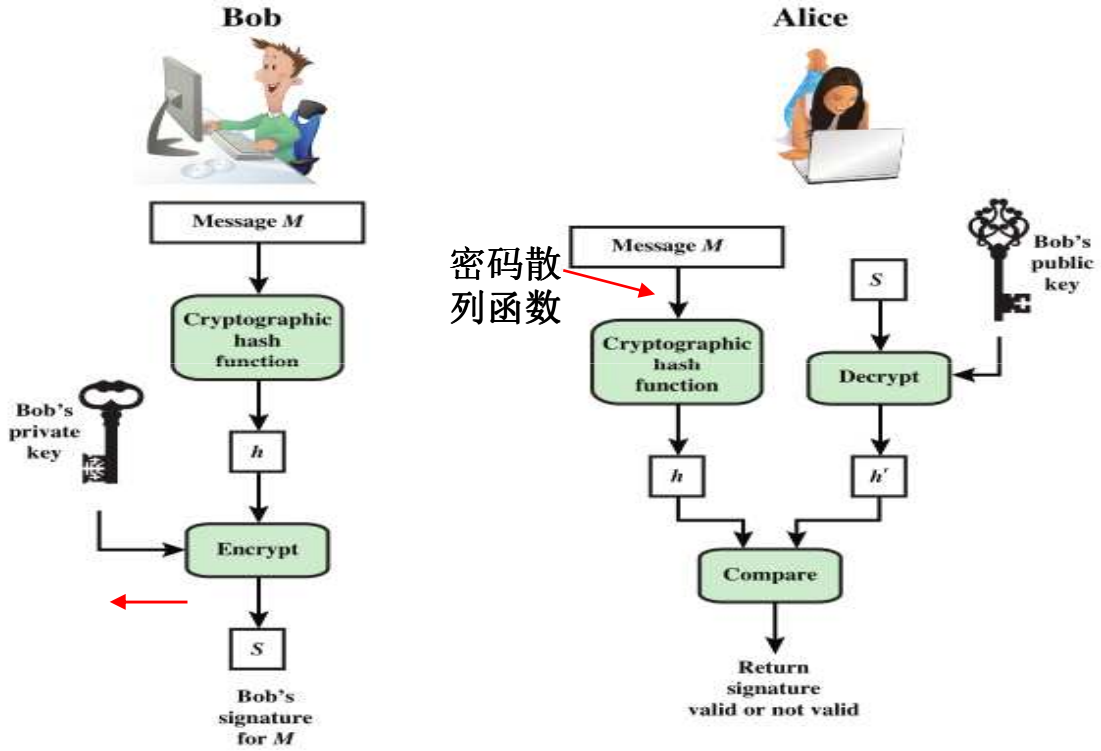


(b) 撤消证书列表

X.509格式

- 证书
  - 版本号
  - 序列号
  - 签名算法
  - 颁发者
  - 证书有效期
    - 此日期前无效
    - 此日期后无效
  - 主题
  - 主题公钥信息
    - 公钥算法
    - 主题公钥
  - 颁发者唯一身份信息（可选项）
  - 主题唯一身份信息（可选项）
  - 扩展信息（可选项）
    - ...
- 证书签名算法
- 数字签名

证书用户数字签名


Bob
Alice
Message M
Cryptographic hash function
Bob's private key
h
Encrypt
S
Bob's signature for M
密码散列函数
Message M
Cryptographic hash function
h
S
Bob's public key
Decrypt
h'
Compare
Return signature valid or not valid

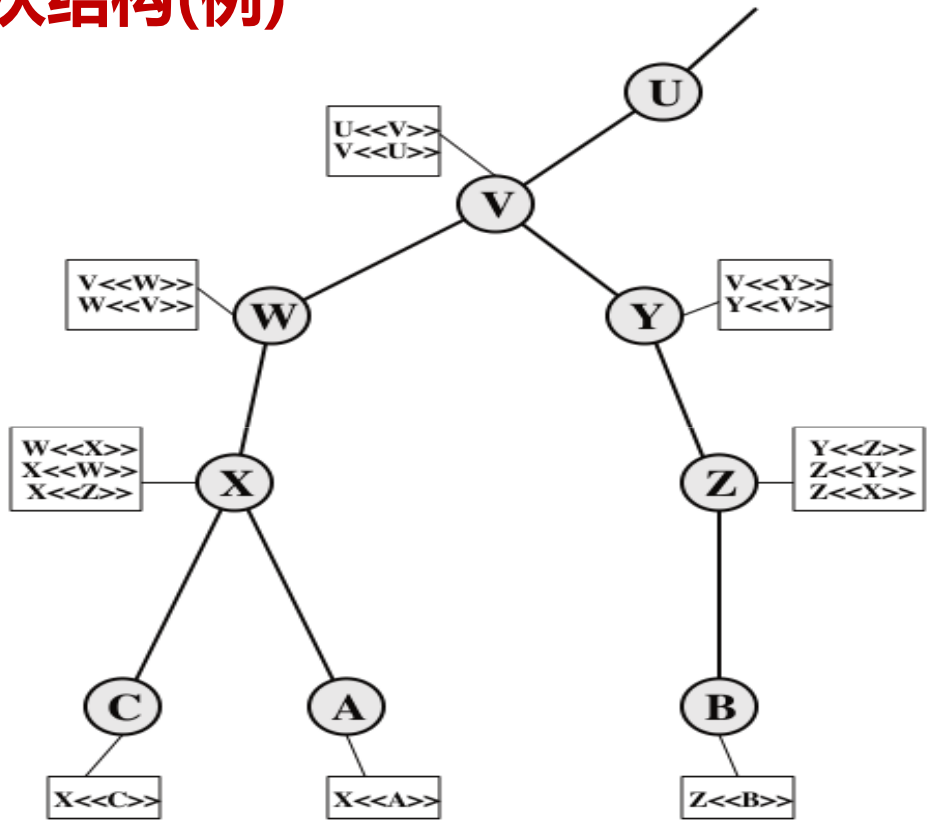
证书获取

- CA生成的用户证书具有以下特点：
  - 任何可以访问CA公钥的用户均可获得证书中的用户公钥。
  - 只有CA可以修改证书。
- 由于证书不可伪造，因此证书可以存放在目录中而不需要对目录进行特别保护

证书获取（续）

- 此标准采取以下表示法来定义一个证书：
  - $CA<<A>>=CA\{V,SN,AI,CA,UCA,A,UA,A_p,T^A\}$
  - 其中：
    - $Y<<X>>$=由认证中心Y发放的用户X的证书；
    - $Y\{I\}$=Y对I的签名，包括I，并在其后追加一个加密过的散列码。
- A使用了一个证书链来得到B的公钥。使用X.509中的表示方法
  - $X_1<<X_2>>X_2<<B>>$
- 通过同样的方式，B可以通过反向链来得到A的公钥：
  - $X_2<<X_1>>X_1<<A>>$

4.6 X.509层次结构(例)



证书获取（续）

- 用户A可以从目录中得到后续证书，进而得到一个通向B的认证路径
  - X<<W>>W<<V>>V<<Y>>Y<<Z>>Z<<B>>
  - 当A已经了这些证书时，它可以依次展开认证路径来恢复可信的B的公钥副本。使用这个公钥B可以发送加密过的消息。
- B可以通过如下路径得到A的公钥副本
  - Z<<Y>>Y<<V>>V<<W>>W<<X>>X<<A>>

证书撤销

- 可能由于以下原因提前撤回证书
  - 用户密钥被认为不安全
  - 用户不再信任该CA
  - CA证书被认为不安全
- 每个CA必须存储一张证书撤销列表（Certificate Revocation List，CRL），用于列出所有被CA撤销但还未到期的证书，包括发给用户和其他CA的证书
- CRL也应被放在目录中

4.4.2 X.509 版本 3

包含许多可添加到版本2格式的可选扩展

每个扩展包括：

- 扩展标识符
- 关键性指标
- 扩展值

证书扩展分为三大类：

- 密钥和政策信息
- 证书主体和证书发放者属性
- 认证路径约束

openssl生成证书的流程

- **安装openssl,配置环境（openssl.cfg）**
- **步骤**
  - **生成CA根证书**
    - 生成CA私钥（.key）
    - 生成CA证书请求（.csr）
    - 自签名得到根证书（.crt）
  - **生成用户证书**
    - 生成私钥（.key）
    - 生成证书请求（.csr）
    - 用CA根证书签名得到证书（.crt）

证书文件常见后缀名

- 后缀
  - .key格式：私有的密钥
  - .csr格式：证书签名请求（证书请求文件），含有公钥信息，certificate signing request的缩写
  - .crt格式：证书文件，certificate的缩写
  - .crl格式：证书吊销列表，Certificate Revocation List的缩写
  - .pem格式：用于导出，导入证书时候的证书的格式，有证书开头，结尾的格式
  - .pfx证书： Personal Information Exchange, IIS

CA根证书的生成步骤

- # Generate CA private key
  - openssl genrsa -out ca.key 2048
- # Generate CSR
  - openssl req -new -key ca.key -out ca.csr
- # Generate Self Signed certificate（CA 根证书）
  - openssl x509 -req -days 365 -in ca.csr -signkey ca.key -out ca.crt

用户证书的生成步骤

- # private key
  - openssl genrsa -des3 -out server.key 1024
- # generate csr
  - openssl req -new -key server.key -out server.csr
- # generate certificate
  - openssl ca -in server.csr -out server.crt -cert ca.crt -keyfile ca.key

生成pem和pfx格式证书

- **生成PEM(Privacy Enhanced Mail)格式证书**
  - copy server.crt+server.key server.pem
- **合成.pfx证书(Personal Information Exchange,_IIS)**
  - 将私钥和服务器crt证书合成.pfx证书
  - openssl pkcs12 -export -out server.pfx -inkey server.key -in server.crt

X.509证书实例

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 286 (0x11e)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=cn, ST=jx, L=gz, O=jxust, OU=is,
CN=jxust/emailAddress=s@qq.com
        Validity
            Not Before: Nov 26 23:56:11 2018 GMT
            Not After : Nov 26 23:56:11 2019 GMT
        Subject: C=cn, ST=jx, O=jxust, OU=it, CN=a/emailAddress=abc@qq.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (1024 bit)
                Modulus:
                    00:dc:60:81:f5:df:99:db:c2:37:12:6c:d6:28:c7:
                    fd:40:98:d0:33:89:c1:6f:d1:85:46:96:4b:f9:a8:
                    78:1f:13:0a:cf:ab:be:03:d5:4b:a0:5a:6e:5a:c3:
                    df:59:70:59:7b:c9:ba:a5:da:84:35:56:12:b2:38:
                    ec:0e:7f:52:b3:6f:a1:37:4a:9d:fd:64:19:73:36:
                    c6:02:9e:1f:b5:c7:15:fc:10:08:bd:55:0b:07:c6:
                    c8:93:f8:1c:2e:10:07:45:d3:90:12:c9:50:9f:3c:
                    d9:60:b5:72:c4:b3:af:68:50:1b:74:b2:18:cd:ee:
                    d0:24:47:72:16:cc:fe:0e:bf
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                7B:AA:EF:E6:A1:52:D6:39:B6:2F:F3:4B:66:EA:57:88:E2:D4:A3:0B
            X509v3 Authority Key Identifier:

DirName:/C=cn/ST=jx/L=gz/O=jxust/OU=is/CN=jxust/emailAddress=s@qq.com
                serial:CD:08:C5:05:51:FE:01:2D

    Signature Algorithm: sha256WithRSAEncryption
         58:1d:63:35:24:1f:f1:3c:a3:2d:a2:bb:91:26:23:c8:0e:6e:
         72:cd:06:c8:a8:4d:fb:5f:35:2a:3f:5c:b0:f9:da:78:11:a5:
         a7:6d:9d:73:1f:1d:e5:4e:27:98:2d:5a:7c:d2:2f:da:4d:53:
         f6:bb:14:33:38:3f:d2:97:ff:9e:3f:09:4c:cd:58:81:6b:d6:
         de:3b:12:6a:82:ee:64:cf:aa:e0:99:05:52:82:4e:b7:0b:d4:
         06:09:e4:42:cf:b3:28:56:5d:a0:8e:66:16:e8:b5:09:e9:0f:
         5c:a1:f1:94:e4:8e:45:dc:ac:26:36:70:48:b8:f0:d9:eb:4e:
         e3:34:10:39:9c:88:a8:29:02:26:aa:d3:9b:05:1a:58:5f:50:
         3d:63:ec:fc:c4:1c:b1:e5:eb:ec:ba:46:57:89:a8:86:0b:75:
         f0:09:90:54:d3:b6:4a:b1:28:70:07:c0:64:eb:41:24:83:db:
         ff:d3:f3:1b:b4:a9:cd:d9:e6:6f:e9:b0:b1:8b:45:9b:a2:82:
         33:f8:fa:d9:aa:2f:23:2d:46:82:50:15:25:a1:4b:40:ea:6e:
         4e:f7:88:16:15:dc:91:1e:6e:12:af:54:0b:35:dc:98:0c:27:
         40:99:04:3f:90:de:c6:96:43:4c:de:ec:b3:39:57:1d:43:0c:
         42:44:15:e2
```

X.509证书实例

```
-----BEGIN CERTIFICATE-----
MIIDqjCCApKgAwIBAgICAR4wDQYJKoZIhvcNAQELBQAwbTELMAkGA1UEBhMCY24x
CzAJBgNVBAgMAmp4MQswCQYDVQQHDAJnejEOMAwGA1UECgwFanh1c3QxCzA
BAsMAmlzMQ4wDAYDVQQDDAVqeHVzdDEXMBUGCSqGSIb3DQEJARYIc0BxcS5
HhcNMTgxMTI2MjM1NjExWhcNMTkxMTI2MjM1NjExWjBeMQswCQYDVQQGEwJ
MAkGA1UECAwCangxDjAMBgNVBAoMBWp4dXN0MQswCQYDVQQLDAJpdDEKMAg
AwwBYTEZMBcGCSqGSIb3DQEJARYKYWJjQHFxLmNvbTCBnzANBgkqhkiG9w0
AAOBjQAwgYkCgYEA3GCB9d+Z28I3EmzWKMf9QJjQM4nBb9GFRpZL+ah4HxM
A9VLoFpuWsPfWXBZe8m6pdqENVYSsjjsDn9Ss2+hN0qd/WQZczbGAp4ftcc
vVULB8bIk/gcLhAHRdOQEslQnzzZYLVyxLOvaFAbdLIYze7QJEdyFsz+Dr8
AaOB5jCB4zAJBgNVHRMEAjAAMCwGCWCGSAGG+EIBDQQfFh1PcGVuU1NMIEd
YXRlZCBDZXJ0aWZpY2F0ZTAdBgNVHQ4EFgQUe6rv5qFS1jm2L/NLZupXiOL
gYgGA1UdIwSBgDB+oXGkbzBtMQswCQYDVQQGEwJjbjELMAkGA1UECAwCang
BgNVBAcMAmd6MQ4wDAYDVQQKDAVqeHVzdDELMAkGA1UECwwCaXMxDjAMBgN
BWp4dXN0MRcwFQYJKoZIhvcNAQkBFghzQHFxLmNvbYIJAM0IxQVR/gEtMA0
SIb3DQEBCwUAA4IBAQBYHWM1JB/xPKMtoruRJiPIDm5yzQbIqE37XzUqP1y
EaWnbZ1zHx3lTieYLVp80i/aTVP2uxQzOD/Sl/+ePwlMzViBa9beOxJqgu5
mQVSgk63C9QGCeRCz7MoVl2gjmYW6LUJ6Q9cofGU5I5F3KwmNnBIuPDZ607
nIioKQImqtObBRpYX1A9Y+z8xByx5evsukZXiaiGC3XwCZBU07ZKsShwB8B
g9v/0/MbtKnN2eZv6bCxi0WbooIz+PrZqi8jLUaCUBUloUtA6m5O94gWFdyRHm4S
r1QLNdyYDCdAmQQ/kN7GlkNM3uyzOVcdQwxCRBXi
-----END CERTIFICATE-----
```

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,6A055F96577E21CB

5ECmrrlvRovtH1C/136YhmIidyCrXhs7TStJnM8S6XD6ZhPZCBPS9ELhKgquy
5g/BFn5j4fH+/UevEb8L/fxPqXA7haLkloy5QTteVcEg6rZ5k2gvW6lgKWC1v
TEAhD/MeMRspw4w4sg9uv9oaPDpMZ0Q17aMvzuiaeMdsjgoiuKodF0fNKf8JD
hVT2JejumeW1fswsrTi2hgXVqLiaoz9c4JkpxunhkzFixycGbdXJendJimtfW
EkkKy6naZ1LPBsq2sbIbuDsu8msOMT11VuSQ8aVAK4gBFjRQAFn+6IcBb97U8
5yrTkVRE3a8l8nUBuiDi7CvkdLQcIqHlTujmosfAdtFf+d/YXYA59r7VHNCkA
8pg5wwYpSF0hVEVu++9I0m1WVdjl9mmZOskbSpDZIgJ4C0oIsIbogbC4JcA+i
A9pVhPZ8GCRBrF9HEflvc2APQlNNGc+DtWJEhmxZ8Yls9NzG8L9ebskWxM6rv
CFVCdyOOC/51/+QAlAsrMcqJADdKOyF7H7NWHOAna2ducCBMgv0CxRHRADnJq
wGHsJtAolDRwgrU5HESPTM42akAaQYbXKC4lEP2116ad3WA5pRKe8Qr+gjFU
bs27jL0EXxwQGPZjkkUEKX9hGdYUd5uq6EhsqSepC8+NZculsU3AsU4SCxxwC
vcucCP2julpshbLsTQePPYTCffDSY7tpiWF5IJ8SEehVK5OxElnlbbIBHw1Yl
RxqhjwOFurDP5talzy2aI/Fdcpj067edvQvSxqX3oNrvWT5brzxJnA==
-----END RSA PRIVATE KEY-----
```

X.509证书的ASN.1数据结构（部分）

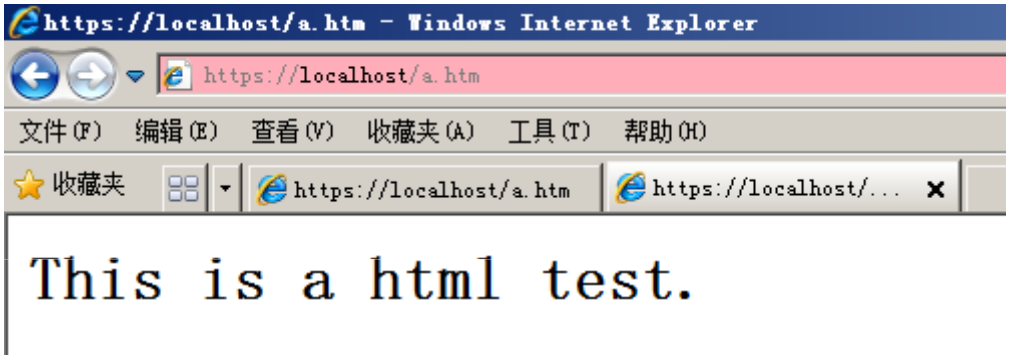
```
Certificate ::= SEQUENCE {
        tbsCertificate       TBSCertificate, -- 证书主体
        signatureAlgorithm   AlgorithmIdentifier, -- 证书签
        signatureValue       BIT STRING --证书签名值,是使用sigr
部分指定的签名算法对tbsCertificate证书主题部分签名后的值.
         }
```

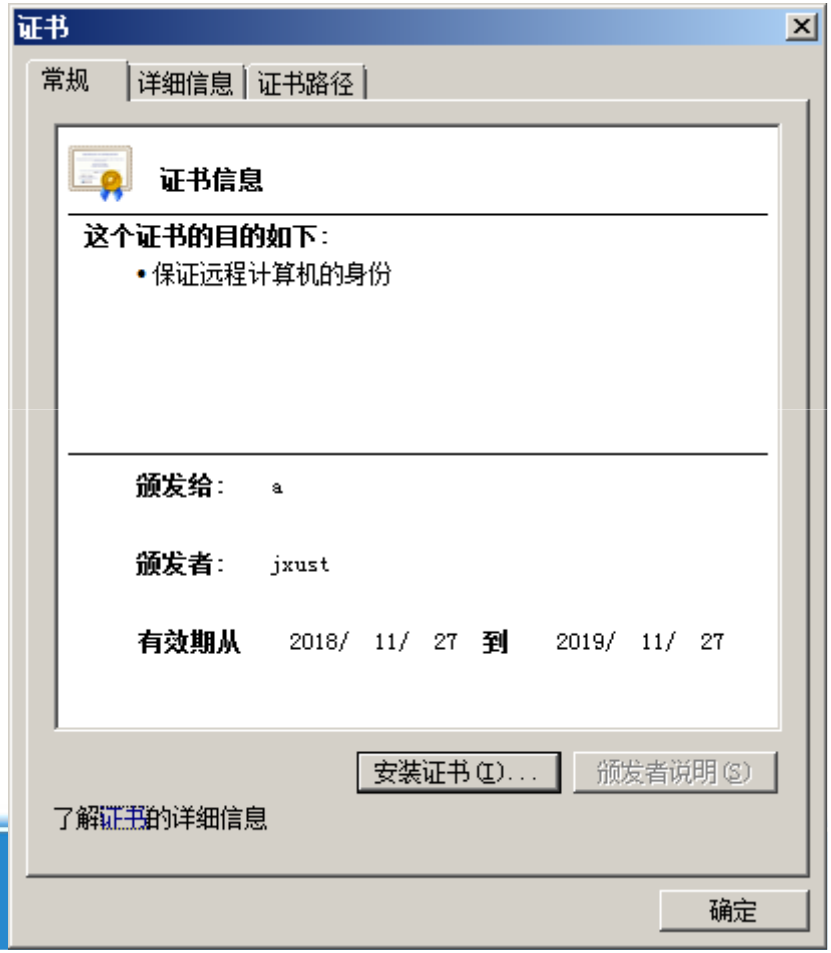
```
TBSCertificate ::= SEQUENCE {
     version         [0] EXPLICIT Version DEFAULT v1, -- 证书版本号
     serialNumber       CertificateSerialNumber, -- 证书序列号，对同一CA
所颁发的证书，序列号唯一标识证书
     signature           AlgorithmIdentifier, --证书签名算法标识
     issuer              Name,                --证书发行者名称
     validity            Validity,            --证书有效期
     subject             Name,                --证书主体名称
     subjectPublicKeyInfo SubjectPublicKeyInfo,--证书公钥
     issuerUniqueID [1] IMPLICIT UniqueIdentifier OPTIONAL,
                         -- 证书发行者ID(可选)，只在证书版本2、3中才有
     subjectUniqueID [2] IMPLICIT UniqueIdentifier OPTIONAL,
                         -- 证书主体ID(可选)，只在证书版本2、3中才有
     extensions      [3] EXPLICIT Extensions OPTIONAL
                         -- 证书扩展段（可选），只在证书版本3中才有
     }
```

X.509证书的对应的结构体（部分）

```
struct x509_st
{
        X509_CINF *cert_info;
        X509_ALGOR *sig_alg;
        ASN1_BIT_STRING *signature;
        int valid;
        int references;
        char *name;
        CRYPTO_EX_DATA ex_data;
        /* These contain copies of various extension values */
        long ex_pathlen;
        long ex_pcpathlen;
        unsigned long ex_flags;
        unsigned long ex_kusage;
        unsigned long ex_xkusage;
        unsigned long ex_nscert;
        ASN1_OCTET_STRING *skid;
        AUTHORITY_KEYID *akid;
        X509_POLICY_CACHE *policy_cache;
        STACK_OF(DIST_POINT) *crldp;
        STACK_OF(GENERAL_NAME) *altname;
        NAME_CONSTRAINTS *nc;
#ifndef OPENSSL_NO_RFC3779
        STACK_OF(IPAddressFamily) *rfc3779_addr;
        struct ASIdentifiers_st *rfc3779_asid;
#endif
#ifndef OPENSSL_NO_SHA
        unsigned char sha1_hash[SHA_DIGEST_LENGTH];
#endif
        X509_CERT_AUX *aux;
} /* X509 */;
```

浏览器运行


https://localhost/a.htm - Windows Internet Explorer
https://localhost/a.htm
文件(F) 编辑(E) 查看(V) 收藏夹(A) 工具(T) 帮助(H)
收藏夹
https://localhost/a.htm
https://localhost/...
This is a html test.


证书
常规
详细信息
证书路径
证书信息
这个证书的目的如下:
• 保证远程计算机的身份
颁发给: a
颁发者: jxust
有效期从 2018/ 11/ 27 到 2019/ 11/ 27
安装证书(I)...
颁发者说明(S)
了解证书的详细信息
确定

Server主要代码

- **载入用户的数字证书， 此证书用来发送给客户端。证书里包含有公钥**
  - **SSL_CTX_use_certificate_file(ctx, "cacert.pem", SSL_FILETYPE_PEM)**
- **载入用户私钥**
  - **SSL_CTX_use_PrivateKey_file(ctx, "privkey.pem", SSL_FILETYPE_PEM)**
- **接收客户端连接**
  - **正常收发数据**

Client主要代码

- 建立正常的socket连接
- 将新连接的socket 加入到SSL
  - SSL_set_fd(ssl, sockfd);
- 建立SSL 连接
  - SSL_connect(ssl)
- 正常收发数据

Server端运行结果

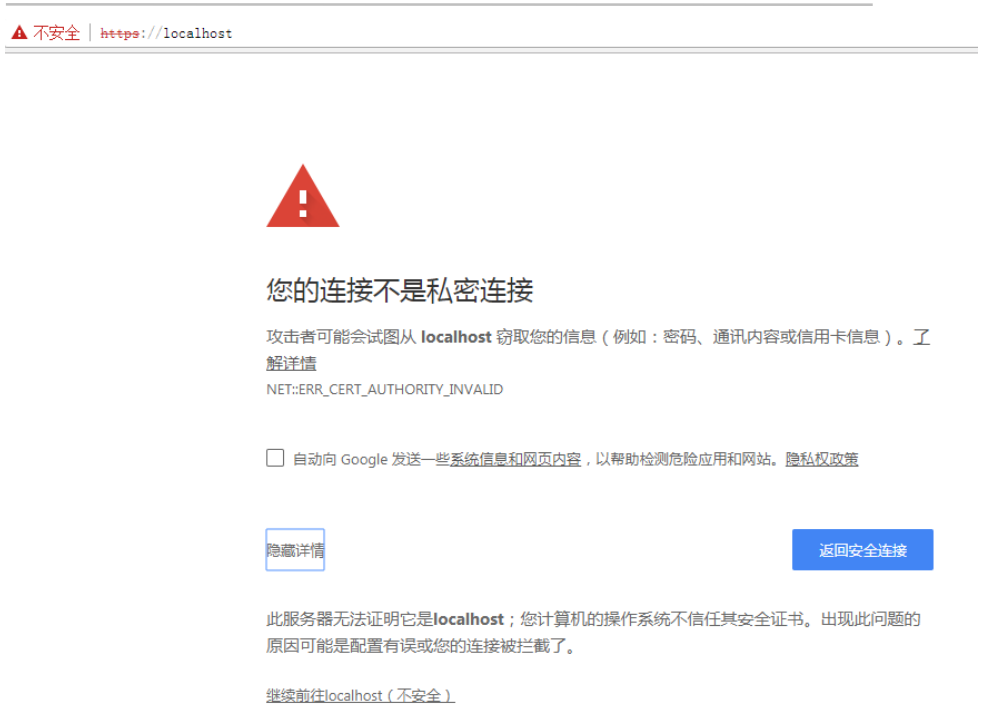
```
socket created
binded
begin listen
server: got connection from 127.0.0.1, port 32362, socket 296
消息接收失败！错误代码是0，错误信息是'No error'
消息'HTTP/1.1 200 OK

<html><head><title>abc</title></head><body>who is a dog</body></html>'发送成功，
共发送了88 个字节！
```

Client运行结果

```
socket created
server connected
Connected with AES256-GCM-SHA384 encryption
数字证书信息:
证书: /C=CN/ST=JX/L=JXUST/O=JXUST/OU=XX/CN=JXUST/emailAddress=abc@def.com
颁发者: /C=CN/ST=JX/L=JXUST/O=JXUST/OU=XX/CN=JXUST/emailAddress=abc@def.com
```

浏览器执行结果


不安全 | https://localhost
您的连接不是私密连接
攻击者可能会试图从 localhost 窃取您的信息（例如：密码、通讯内容或信用卡信息）。了解详情
NET::ERR_CERT_AUTHORITY_INVALID
自动向 Google 发送一些系统信息和网页内容，以帮助检测危险应用和网站。隐私权政策
隐藏详情
返回安全连接
此服务器无法证明它是localhost；您计算机的操作系统不信任其安全证书。出现此问题的原因可能是配置有误或您的连接被拦截了。
继续前往localhost（不安全）


不安全 | https://localhost
who is a dog

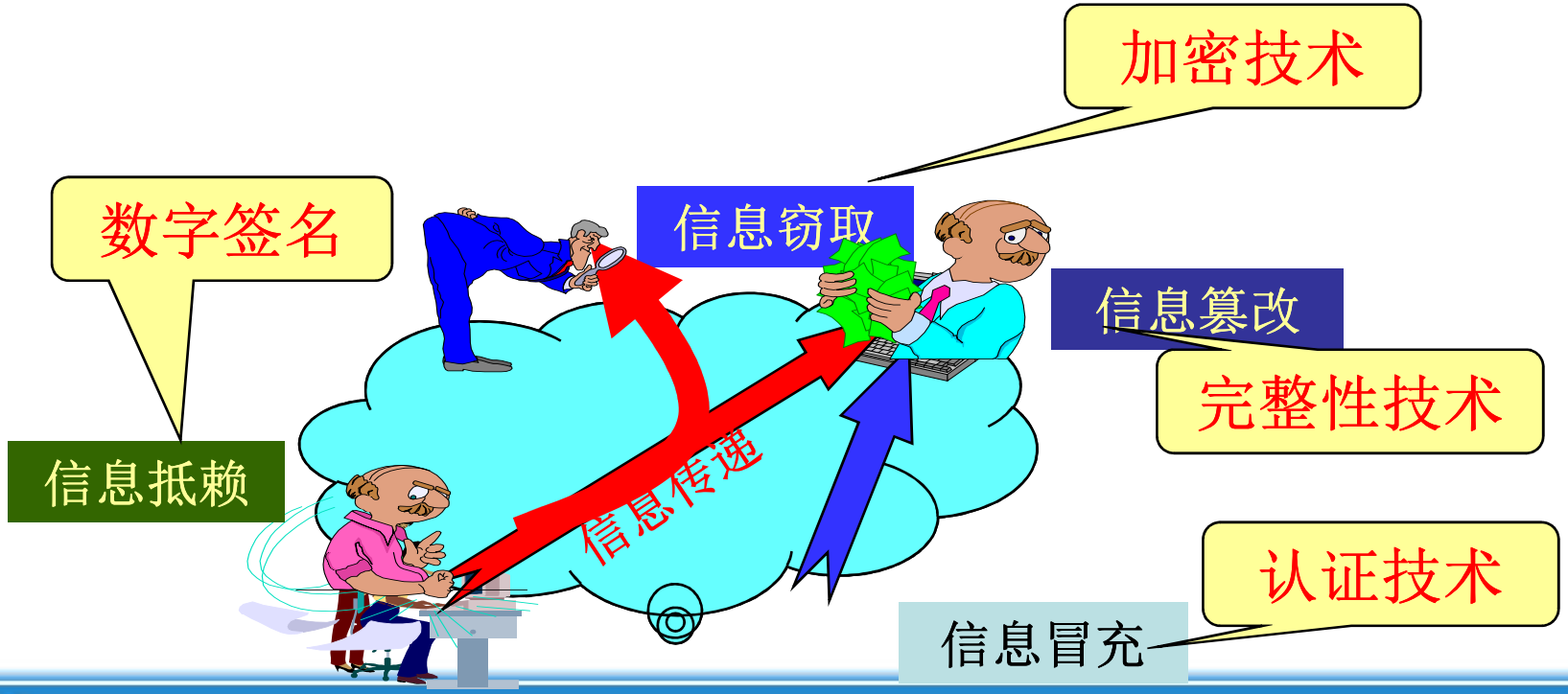
05
Part

公钥基础设施（PKI）

安全的主要威胁

- **假冒**：指非法用户假冒合法用户身份获取敏感信息；
- **截取**：指非法用户截获通信网络的数据；
- **篡改**：指非法用户改动所截获的信息和数据；
- **否认**：指通信的单方或多方事后否认曾经参与某次活动；

常见的解决方法



公钥基础设施（PKI）的含义

- PKI(Public Key Infrastructure，公钥基础设施)，是一个基于公钥概念和技术实现的、具有通用性的安全基础设施。
- PKI公钥基础设施的主要任务是在开放环境中为开放性业务提供公钥加密和数字签名服务。
- PKI是生成、管理、存储、分发和吊销基于公钥密码学的公钥证书所需要的硬件、软件、人员、策略和规程的总和。

PKI的内容和目标

- PKI技术以公钥技术为基础，以数字证书为媒介，结合对称加密和非对称加密技术，将个人、组织、设备的标识信息与各自的公钥捆绑在一起。
- 其主要目的是通过自动管理密钥和证书，为用户建立起一个安全、可信的网络运行环境，使用户可以在多种应用环境下方便地使用加密和数字签名技术，在互联网上验证用户的身份，从而保证了互联网上所传输信息的真实性、完整性、机密性和不可否认性。
- PKI是目前为止既能实现用户身份认证，又能保证互联网上所传输数据安全的惟一技术。

PKI的理解

- 基于公开密钥理论和技术建立起来的安全体系。
- 一个被广泛认识并且被普遍接受的相当标准化的结构。
- 提供信息安全服务的具有普适性的安全基础设施。
- CA认证、数字证书、目录服务以及相关安全应用组件模块的集合 。
- 核心是要解决信息网络空间中的信任问题，确定可信赖的数字身份。
- 似乎可以解决绝大多数网络安全问题，并初步形成了一套完整的解决方案。

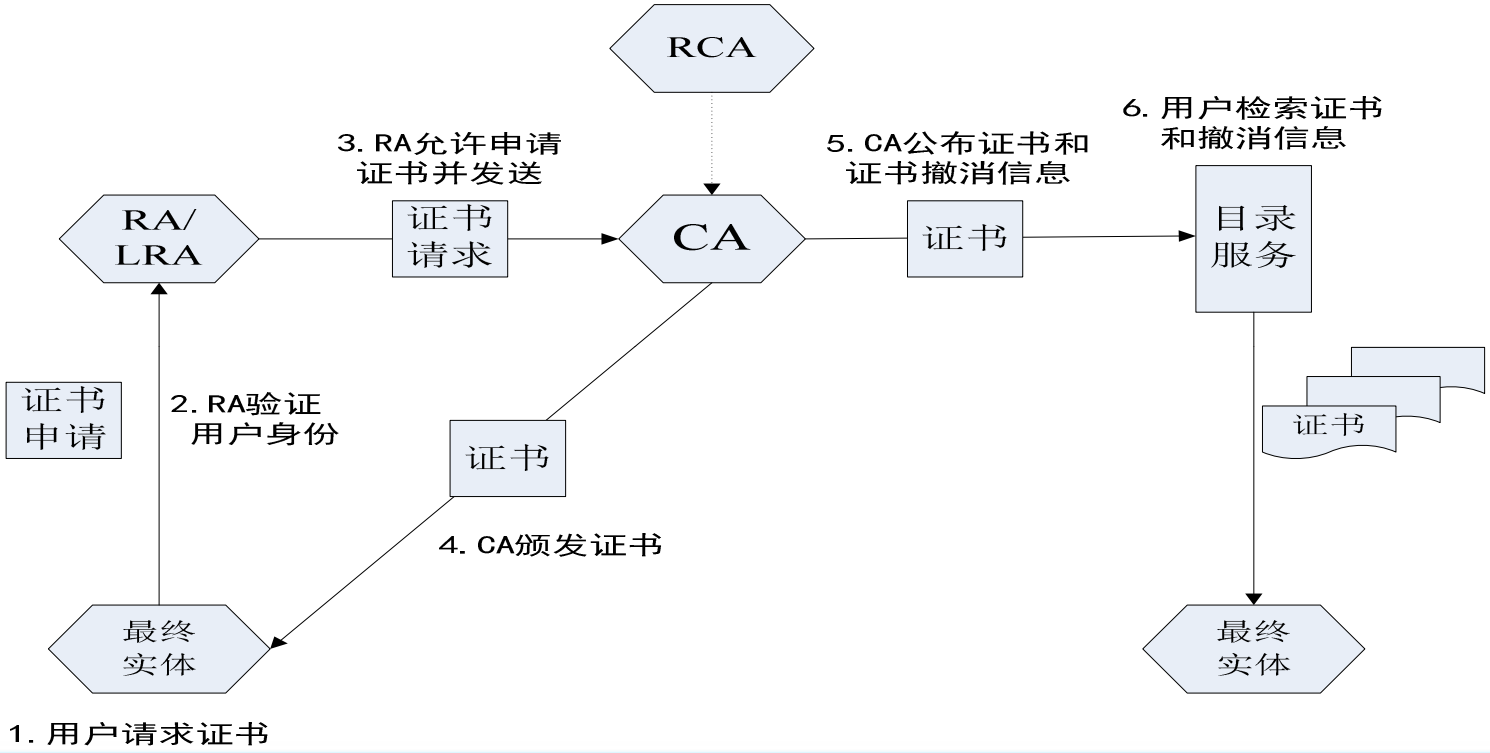
PKI的优势

- 节省费用
- 透明性和易用性
- 互操作性
- 可扩展性
- 多用性
- 支持多平台

PKI的主要组件

- **证书授权中心**
  - 证书签发机构，是PKI的核心，使PKI 应用中权威的、可信任的、公正的第三方机构。主要负责产生、分配并管理所有参与网上交易的实体所需身份认证数字证书。
- **证书库**
  - 证书的集中存放地，提供公众查询。
- **密钥备份及恢复系统**
  - 对用户的解密密钥进行备份，当丢失时进行恢复，而签名密钥不用备份和恢复。
- **证书撤销处理系统**
  - 证书由于某种原因需要作废，终止使用，将通过证书撤销列表CRL来实现。
- **PKI应用接口系统**
  - 为各种各样的应用提供安全、一致、可信任的方式与PKI交互，确保所建立起来的网络环境安全可靠，并降低管理成本。

PKI主要组件


RCA
3. RA允许申请
证书并发送
5. CA公布证书和
证书撤消信息
6. 用户检索证书
和撤消信息
RA/
LRA
证书
请求
CA
证书
目录
服务
证书
申请
2. RA验证
用户身份
证书
证书
4. CA颁发证书
最终
实体
最终
实体
1. 用户请求证书

PKI主要组件的简介

- **公钥证书：**由可信实体签名的电子记录，记录将**公钥**和密钥（公私钥对）**所有者**的身份捆绑在一起。公钥证书是PKI的基本部件。
- **根CA**：一个单独的、可信任的根CA是PKI的基础，生成一个自签名证书，亦称CA证书或根证书。
- **注册机构和本地注册机构RA**：证书发放审核部件，接受个人申请，对证书申请者进行资格审查并发送给CA。本质上， RA 是CA系统的一个**功能组件**，可设计成CA的代理处，分担CA的一定功能以增强可扩展性。
- **目录服务（资料库）**：PKI的一个重要组成部分，主要用于发布用户的证书和证书作废列表（黑名单）。

PKI主要组件的简介（续）

- **管理协议：用于管理证书的注册、生效、发布和撤销。**
  - PKI管理协议包括PKI CMP、消息格式、CMMF、PKCS#10。
- **操作协议：允许用户方便地通过证书库检索和验证证书和CRL。**
  - 在大多数情况下，操作协议与现有协议（如Ftp、Http、Ldap等）共同合作。
- **最终用户（End User）：也称最终实体（End-Entity）,可以是人，也可以是机器，如路由器，或计算机中运行的进程，如应用系统。**

CA的简介

- **CA机构，又称为证书认证(Certificate Authority)中心，作为电子交易中受信任的第三方，承担公钥体系中公钥的合法性检验的责任。核心功能就是发放和管理数字证书**
- **CA中心为每个使用公开密钥的用户发放一个数字证书，数字证书的作用是证明证书中列出的用户合法拥有证书中列出的公开密钥。**
- **CA机构的数字签名使得攻击者不能伪造和篡改证书。它负责产生、分配并管理所有参与网上交易的个体所需的数字证书，因此是安全电子交易的核心环节。**

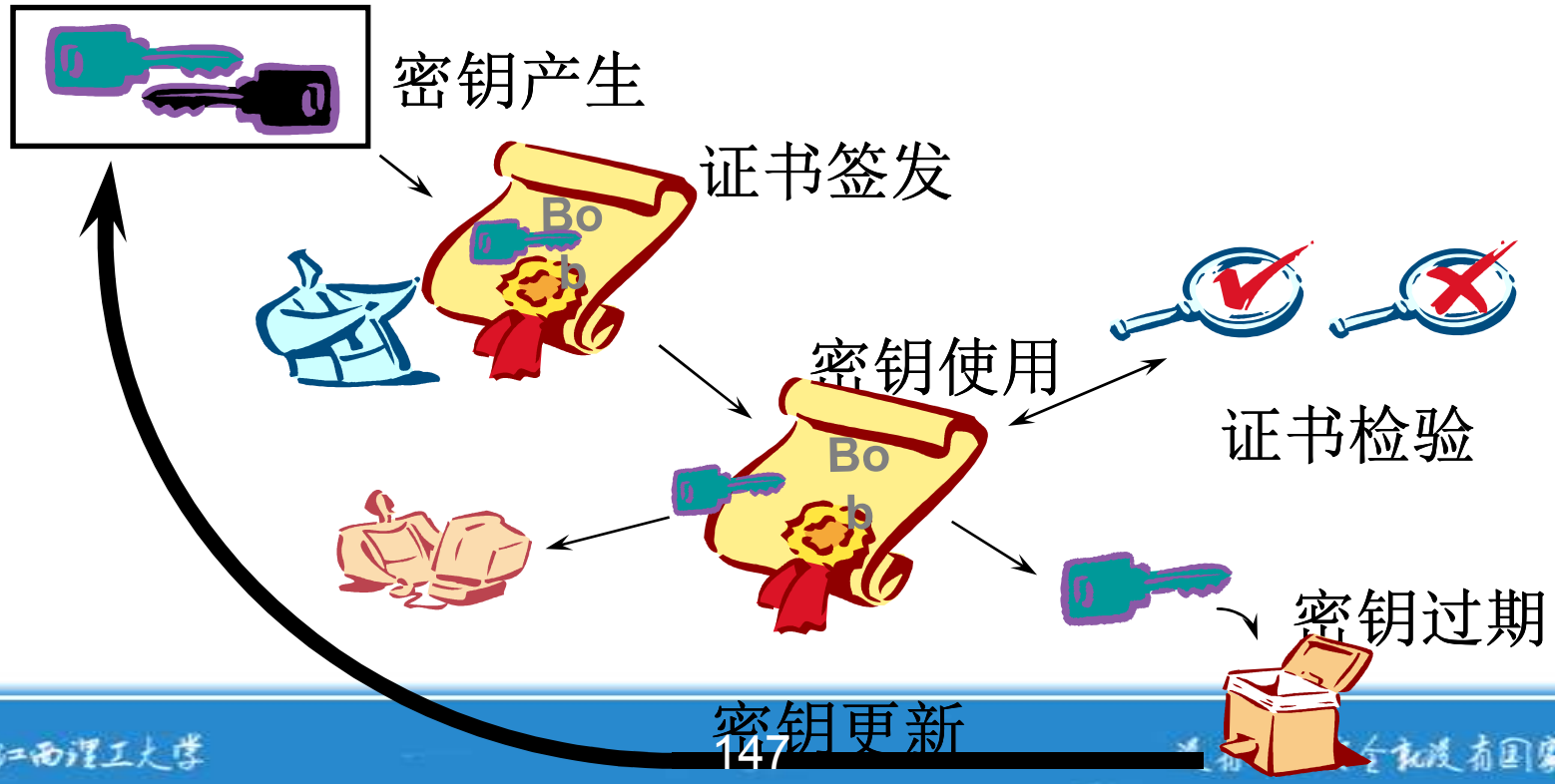
CA组成

- **签名和加密服务器**
  - 对于数字证书和被撤销的数字证书，应有认证机构的数字签名。
- **密钥管理服务器**
  - 与签名加密服务器连接，按配置生成密钥、撤销密钥、恢复密钥和查询密钥。
- **证书管理服务器**
  - 主要完成证书的生成、作废等操作控制。
- **证书发布和CRL发布服务器**
  - 用于将证书信息按一定时间间隔对外发布，为客户提供证书下载和CRL下载等服务。
- **在线证书状态查询服务器**
  - 证书用户随时都知道某个证书的最新状态。
- **Web服务器**
  - 用于证书发布和有关数据认证系统政策的发布。

注册中心（RA）

- **注册中心**是数字证书注册审批机构。
- **RA 系统**是CA系统的证书发放、管理的延伸，是整个CA中心得以正常运营不可缺少的一部分。本质上是CA的功能组件。
- 但有的系统中，将RA合并在CA中。
- 一般而言，注册中心控制注册、证书传递、其他密钥和证书生命周期管理过程中主体和PKI间的交换。
- 在任何环境下，RA都不发起关于主体的可信声明。

密钥生命周期

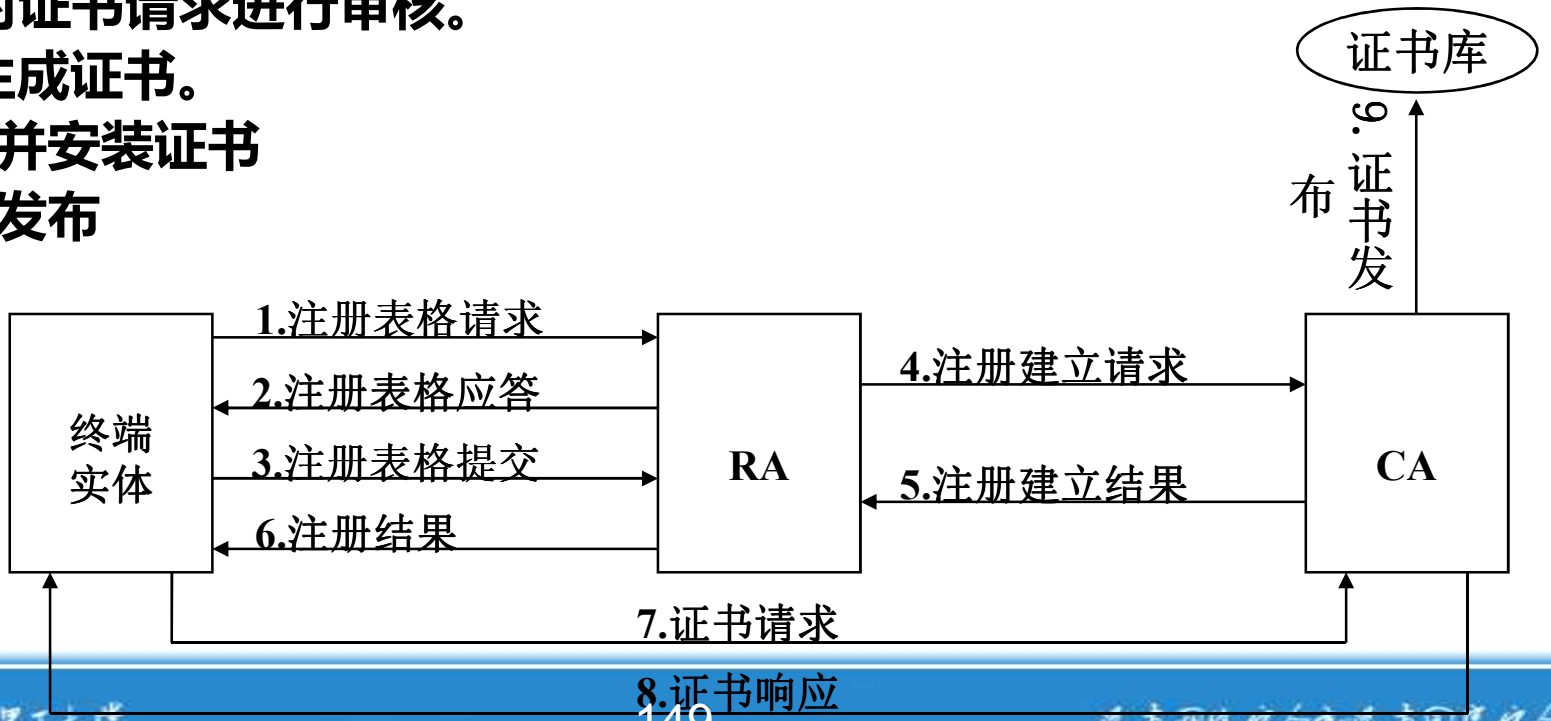


证书管理

- 证书注册
- 证书存放
- 证书撤销
- 证书验证
- 证书状态查询

证书注册

- 申请人提交证书请求。
- RA对证书请求进行审核。
- CA生成证书。
- 下载并安装证书
- 证书发布



证书的更新

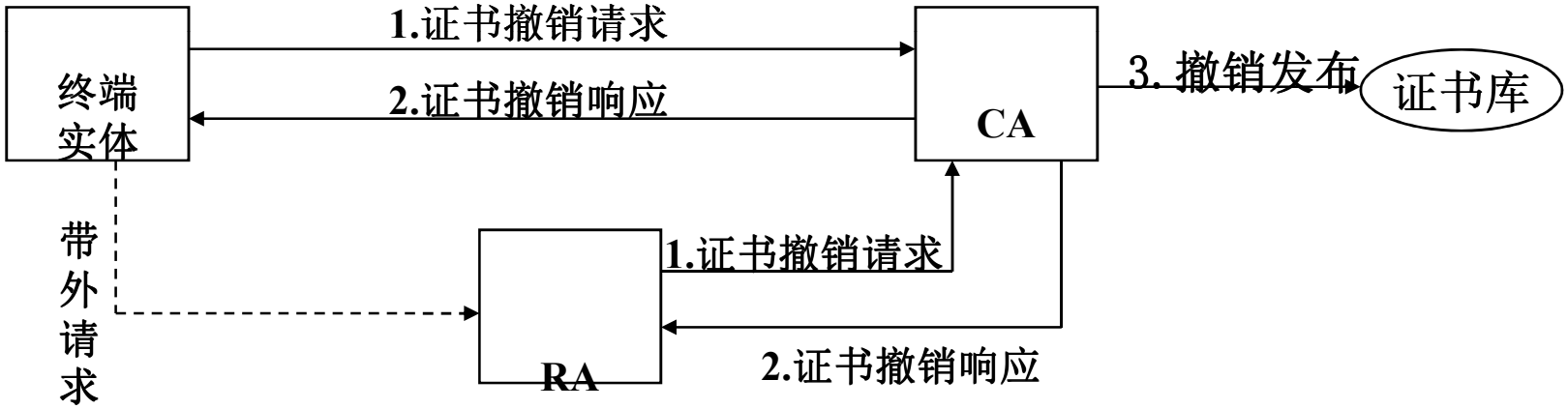
- 最终实体证书更新
  - 证书过期
  - 一些属性的改变
  - 发放新证书
- CA证书更新

证书存放

- 使用IC卡存放
- 直接存放在磁盘或自己的终端上
- USB Key
- 证书库

证书撤销

- 当条件（雇佣关系结束、证书中信息修改等）要求证书的有效期在证书结束日期之前终止，或者要求用户与私钥分离时（私钥可能以某种方式泄露），证书被撤销。



证书验证

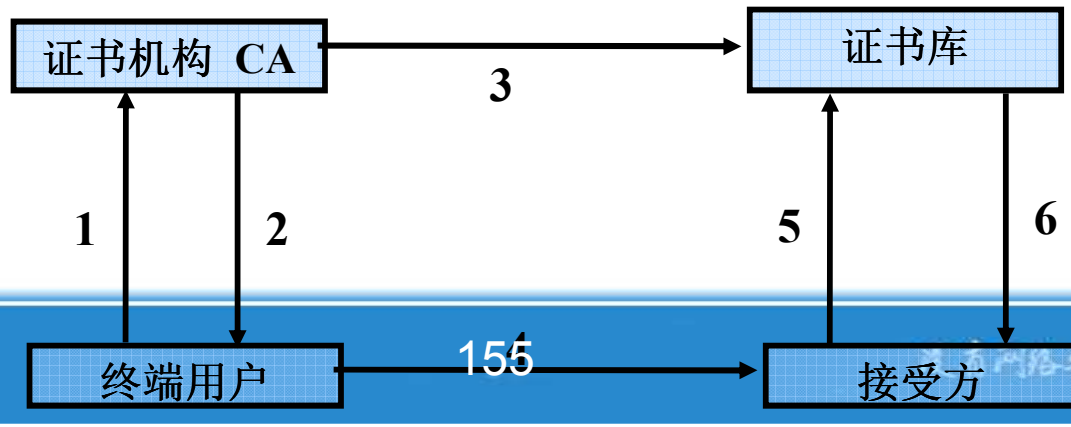
- 使用CA证书验证其他终端实体证书有效性。
- 检查证书的有效期，确保该证书是否有效。
- 检查该证书的预期用途是否符合CA在该证书中指定的所有策略限制。
- 在证书撤销列表（CRL）中查询确认该证书是否被CA撤销。

证书状态查询

- 定期下载证书撤销列表（CRL）。
- 在线证书状态协议OCSP（Online Certificate Status Protocol），其目的为了克服基于CRL的撤销方案的局限性，为证书状态查询提供即时的最新响应。OCSP使用证书序列号、CA名称和公开密钥的散列值作为关键字查询目标的证书。

PKI的运行举例

- 1）终端用户向证明机构（CA）提出数字证书申请；
- 2）CA验明终端用户身份，并签发数字证书；
- 3）CA将证书公布到证书库中；
- 4）终端用户对电子信件数字签名作为发送认证，确保信件完整性，不可否认性，并发送给接受方。
- 5）接受方接收信件，根据终端用户标识向证书库查询证书的状态；
- 6）下载终端用户证书并用其公钥验证数字签名，确定电子信件数字签名的有效性；



数字证书的应用

- 现有持证人A向持证人B传送数字信息，为了保证信息传送的真实性、完整性和不可否认性，需要对要传送的信息进行数据加密和数字签名，其传送过程如下：
  - A准备好要传送的数字信息（明文）。
  - A对数字信息进行哈希（hash）运算，得到一个信息摘要。
  - A用自己的私钥对信息摘要进行加密得到A的数字签名，并将其附在数字信息上。
  - A随机产生一个加密密钥（DES密钥），并用此密钥对要发送的信息进行加密，形成密文。

数字证书的应用（续）

- A获得B的证书，并用CA证书验证及检查证书撤销列表验证B证书是否有效。
- A用B的公钥（来自证书）对刚才随机产生的加密密钥进行加密，将加密后的DES密钥连同密文一起传送给B。
- B收到A传送过来的密文和加过密的DES密钥，先用自己的私钥对加密的DES密钥进行解密，得到DES密钥。
- B将收到的信息摘要和新产生的信息摘要进行比较，如果一致，说明收到的信息没有被修改过。

数字证书的应用（续）

- B然后用DES密钥对收到的密文进行解密，得到明文的数字信息，然后将DES密钥抛弃（即DES密钥作废）。
- B获得A的证书，并用CA证书验证及检查证书撤销列表验证A证书是否有效。
- B用A的公钥（来自证书）对A的数字签名进行解密，得到信息摘要。
- B用相同的hash算法对收到的明文再进行一次hash运算，得到一个新的信息摘要。

06
Part

联合身份管理

4.6 联合身份管理

- **一种集中的自动化方法，可以为员工和其他授权个人提供企业范围的资源访问。**
  - **重点是为每个用户（人或过程）定义身份，将属性与身份相关联，并制订一种方法使得用户可以证明其身份。**
  - **身份管理系统的中心思想是使用单点登录（SSO），使用户能够在单次身份验证后访问所有网络资源。**

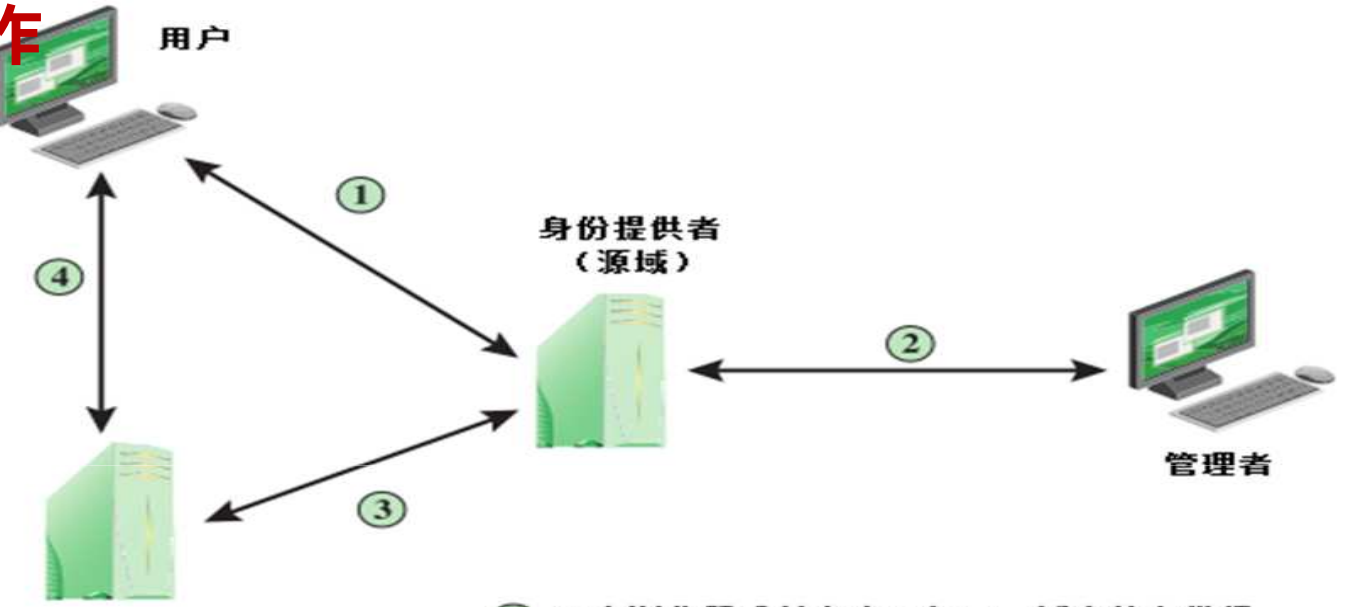
4.6.1 身份管理

- **身份管理系统的主要内容：**
  - **认证：确认与提供的用户名相关联的用户。**
  - **授权：基于认证，授权到特殊服务和/或者资源的接口。**
  - **审计：进行登录和授权的过程。**
  - **供给：系统中用户的登记。**
  - **工作流程自动化：事务处理中的数据移动。**
  - **委派管理：使用基于角色的接入控制来授权许可。**
  - **口令同步：为单点登录（SSO）或减少登录（RSO）建立进程。**
  - **自助口令密码重置：使用户可以修改密码。**
  - **联合：一个使认证和许可从一个系统（或企业）转移到另一个系统（或企业）的进程。以减少用户认证的次数。**

4.6.2 身份联合

- **身份联合就是将身份扩展到多个安全域。**
- **身份联合的目的就是共享数字身份，使得用户只需要认证一次就可以接入多个域的应用及其资源。**
- **这是合作组织在协商的标准和分享数字身份的相互信任的基础上形成的一种联合。**
- **标准：联合身份管理使用一系列标准作为构件，以便 在不同的域或不同系统之间进行安全身份交换。**
- **主要标准：可扩展标记语言（XML）、简单对象访问协议（SOAP）、WS安全、安全断言标记语言（SAML）。**

联合身份操作



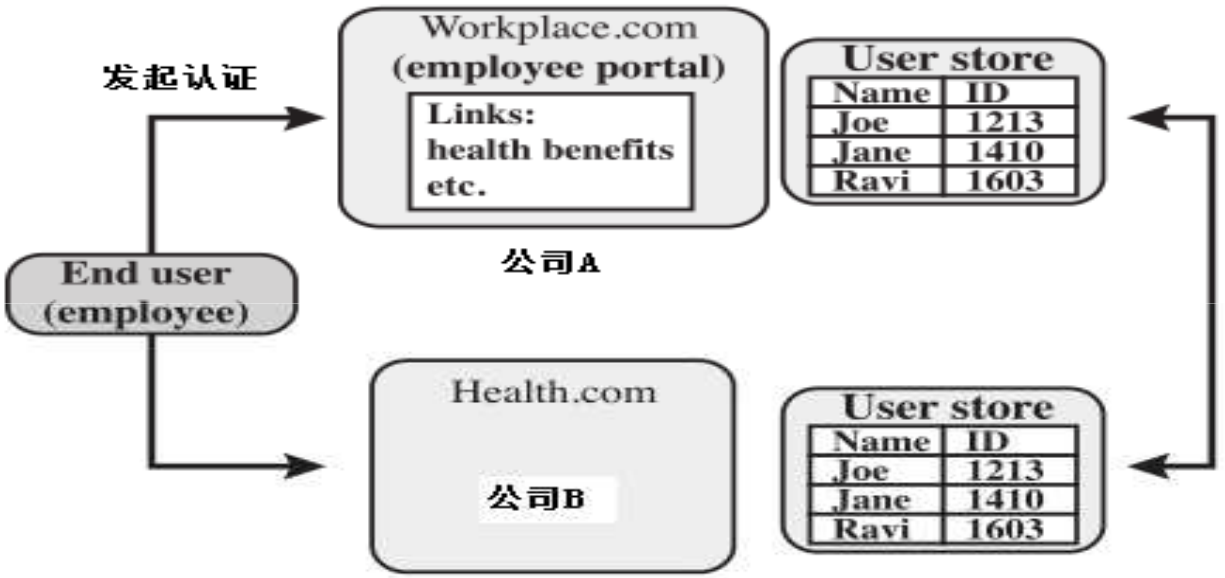
① 用户浏览器或其它应用与同一域中的身份提供者进行认证对话。

② 一些与身份相关的属性（如允许角色），可能由同一域的管理者提供。

③ 用户相要接入的远程域的服务提供者，需从源域的身份认证提供者获得身份信息、认证信息和相关属性。

④ 服务提供者与远程登录用户展开对话，并基于用户的身份和属性来确定接入控制限制。

身份联合



(a) 基于帐号链接的联合

小结

- **身份认证概念**
- **网络用户认证**
- **基于对称密钥的认证系统Kerberos**
- **基于公钥的认证系统X.509**
- **公钥基础设施PKI**
- **联合身份管理**

志存高远 责任为先

感谢聆听

网址：www.jxust. edu.cn
邮编：341000