cshao@gdut.edu.cn

第3章 数字签名与身份认证

认证（Authentication）

- *认证含义*
  - 认证在很多场合都使用，是一个常被误用的术语.它只是表示采用提供的某些手段可以保证实体是所说的那个，或信息没有被非授权人改变
    - 实体认证 **entity authentication (identification)**
    - 数据源认证 **data origin authentication**
  - 上世纪**70**年中期时还认为认证与保密本质是相同
    - **Hash,**数据签名出现

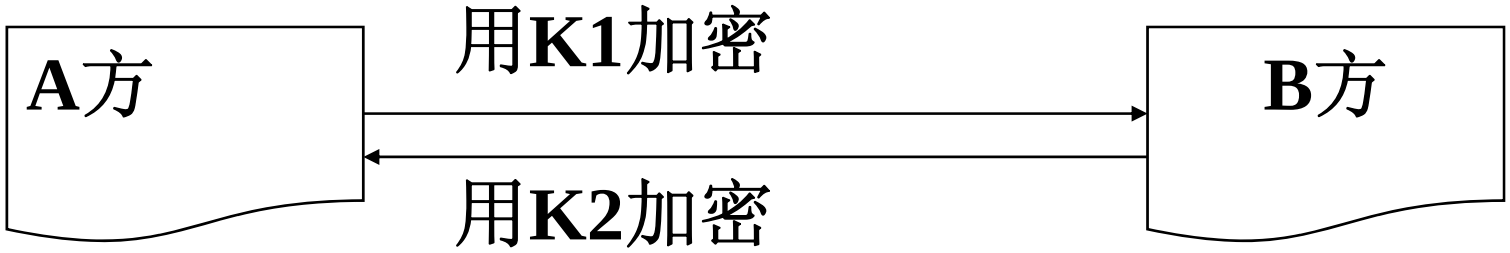
第3章 数字签名与身份认证

- 报文鉴别
- 散列函数
- 数字签名
- 身份认证

一 报文鉴别

- 报文鉴别是这样一种过程
- 它是通信的接收方能够鉴别验证所收到的报文(包括发送者、报文内容、发送时间和序列等)的真伪。

- *报文源的鉴别*
  - 收方使用约定的密钥（由发方决定）对收到的密文进行解密，并且检验还原的明文是否正确，根据检验结果就可以验证对方的身份。其原理如图所示。

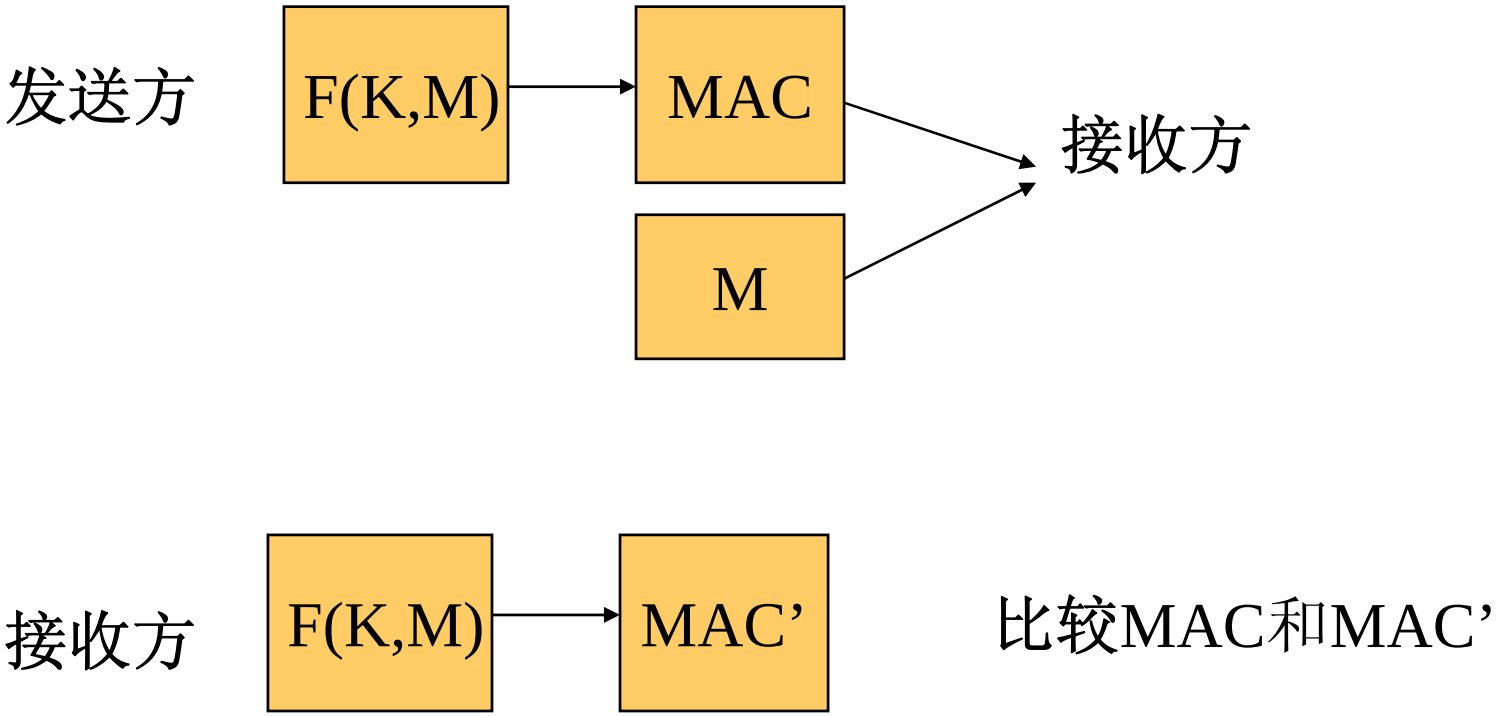


利用报文加密对身份进行鉴别

- *报文宿的鉴别*
  - 在以密钥为基础的鉴别方案的每一报文中，同时加入收方标识符**IDB**；在以通行字为基础的鉴别方案中，每一报文加入收方通行字**PWB**。
  - 若采用公开密钥密码，报文宿的认证只要发方**A**对报文用**B**的公开密钥进行加密即可，因为只有**B**才能用自己保密的解密密钥还原报文，若还原的报文是正确的，**B**便确认自己是指定的收方。

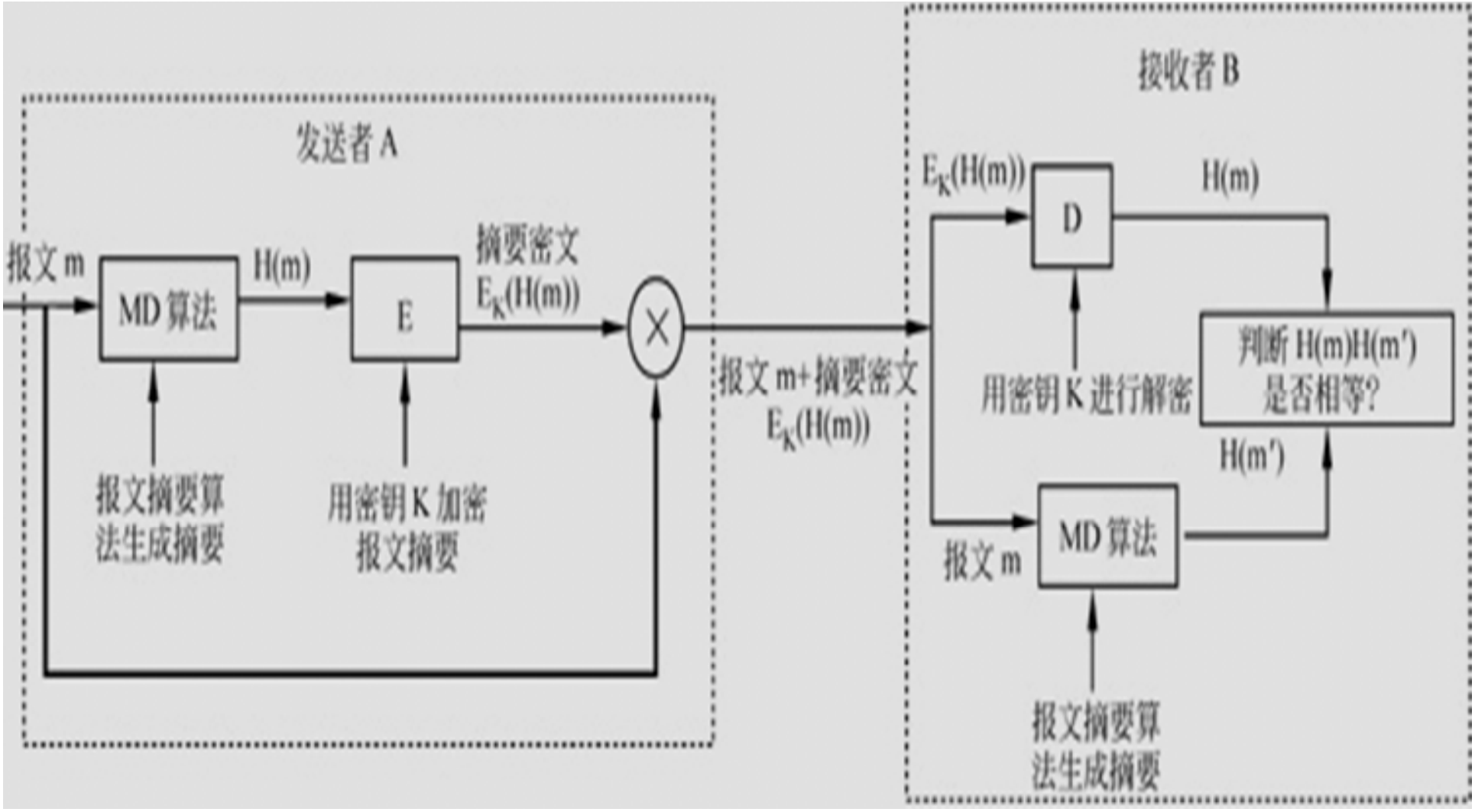
- *报文时间性的鉴别*
  - 初始向量法
  - 时间参数法
  - 随机数法
  - 挫败重播攻击。指攻击者并不试图从截获的数据中恢复出有价值的信息，而是直接将截获的数据重发，以达到非法的目的。

- *报文内容的鉴别*
- *报文鉴别码(Message Authentication Code,MAC)：*
  - 报文鉴别码是用一个密钥生成的一个小的数据块，追加在报文的后面。



- *报文内容的鉴别*
- *报文加密:*用完整的报文的密文作为对报文的认证
  - 对称密钥体系报文加密
  - 公钥密码体系报文加密
  - 但在特定的网络应用中，许多报文并不需要加密，而要求发送的报文应该是完整和不是伪造的。
  - 例如，通知网络上所有的用户有关上网的注意事项。对于不需要加密的报文进行加密和解密，将对计算机增加很多不必要的开销。因此，可使用单独的相对简单的报文鉴别算法来达到目的。

- *报文内容的鉴别*
- *报文摘要*
  - 报文鉴别码的一个变种，将可变长度的报文**M**作为单向散列函数的输入，然后得出一个固定长度的标志**H（M）**，这个**H（M）**就称为报文摘要（**MD**）。
  - 单向散列函数的特点是从一个报文生成一个**MD**代码是容易的，但反过来从一个代码生成一个报文则实际上是不可能的。另外，它保证不同的报文不会得出同样的**MD**代码。如果没有这个特性，攻击者就可能用一个伪造报文替代真报文，只要该伪造报文与真报文能够生成同样的**MD**即可。
-



报文摘要MD算法进行报文鉴别原理示意图

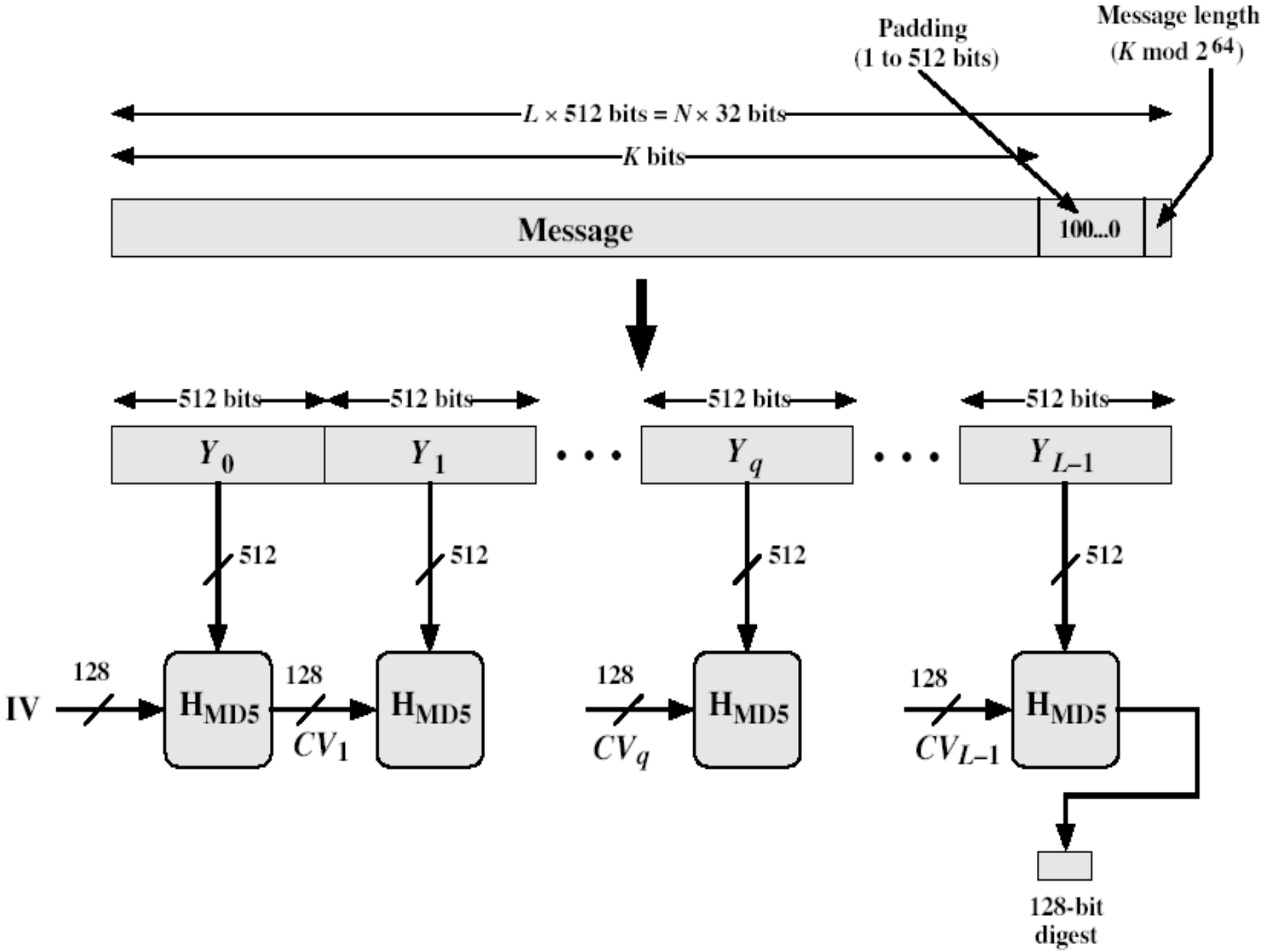
二 散列函数

- 散列函数（**Hash**）又称哈希函数，是把任意长度的报文（消息）**M**，通过函数**H**，将其变换为一个固定长度的散列码**h**，散列函数表示为 **h=H(M)**，它生成报文所独有的“指纹”。惟一地对应原始报文 。
- *用途一验证完整性*
- 如果原始报文改变并且再次通过散列函数，它将生成不同的报文摘要，因此，散列函数能用来检测报文的完整性，保证报文从建立开始到收到始终没有被改变和破坏。运行相同算法的接收者应该收到相同的报文摘要，否则报文是不可信的。

- *用途—密钥认证*
- 大部分操作系统的密码都是以**hash**版本的形式存储，而不是密码的原始文本。
- 当有人登录时，输入的密码先做**hash**处理，然后与存储的数据比较。
- 这意味着机器上没有任何地方保存原始密码，别人很难偷到它。
- 同时摘要算法是单向函数，不可能通过函数值解出原始密码。

- *当前国际通行的两大密码标准*
  - **MD5：**常用的**128**位的消息摘要，大量用于口令存储机制。由国际著名密码学家图灵奖获得者兼公钥加密算法**RSA**的创始人**Rivest**设计
  - **SHA**和**SHA－1：160**位的消息摘要。由美国专门制定密码算法的标准机构—美国国家标准技术研究院（**NIST**）与美国国家安全局（**NSA**）设计。
  - 两大算法是目前国际电子签名及许多其它密码应用领域的关键技术，广泛应用于金融、证券等电子商务领域。其中，**SHA－1**早在**1994**年便为美国政府采纳，目前是美国政府广泛应用的计算机密码系统。

- *摘要算法(MD5)*
- 对**MD5**算法简要的叙述可以为：**MD5**以**512**位分组来处理输入的信息，且每一分组又被划分为**16**个**32**位子分组，经过了一系列的处理后，算法的输出由四个**32**位分组组成，将这四个**32**位分组级联后将生成一个**128**位散列值。


Padding
(1 to 512 bits)
Message length
(K mod 2^{64})
$L \times 512$ bits = $N \times 32$ bits
K bits
Message
100...0
512 bits
512 bits
512 bits
512 bits
Y_0
Y_1
Y_q
Y_{L-1}
512
512
512
512
128
128
128
128
IV
H_{MD5}
H_{MD5}
H_{MD5}
H_{MD5}
CV_1
CV_q
CV_{L-1}
128-bit
digest

- ***1 填充使信息长度为512的整数倍***
- 首先需要对信息进行填充，使信息长度扩展至**N*512+448 Bits**，即**N*64+56**个字节（**Bytes**），**N**为一个正整数。
- 填充的方法如下
  - 在信息的后面填充一个**1**和无数个**0**，直到满足上面的条件时才停止用**0**对信息的填充。
- 然后，在这个结果后面附加一个以**64**位二进制表示的原信息**(**填充前**)**长度。
- 经过这两步的处理，现在的信息字节长度**=N*512+448+64=(N+1)*512**，即长度恰好是**512**的整数倍。这样做的原因是为满足后面处理中对信息长度的要求。

对字符串“**abc**”进行填充

- **(1)**二进制表示：**01100001 01100010 01100011**
- **(2)**消息长**24**，先填充一位**1**，然后填充**423**位**0**
- 此时长度为**0*512+448 Bits**
- **(3)**使用**64**位二进制表示消息的长度**24**
- **0X00000000 00000018**
- 这时总长度为**512**位,分成**16**组,每组**32**位以**16**进制表示如下:
- **M[0]=61626380 M[1]=00000000 M[2]=00000000**
- … **M[14]=00000000 M[15]=00000018**

- ***2 初始化MD缓冲区—记录128位消息摘要***
- 由四个32位寄存器A,B,C,D表示。初始化为
- **A=0x01234567，B=0x89abcdef，C=0xfedcba98，D=0x76543210，**
- ***3 按512位的分组处理输入消息***
- 将上面四个链接变量复制到另外四个变量中：**A到a，B到b，C到c，D到d。**

- 主循环有四轮，每轮循环都很相似。
- 每一轮进行**16**次操作。每次操作对**a**、**b**、**c**和**d**中的其中三个作一次非线性函数运算，然后将所得结果加上第四个变量，一个子分组和一个常数。再将所得结果向右环移一个不定的数，并加上**a**、**b**、**c**或**d**中之一。最后用该结果取代**a**、**b**、**c**或**d**中之一。
- 以一下是每轮中用到的四个非线性函数。

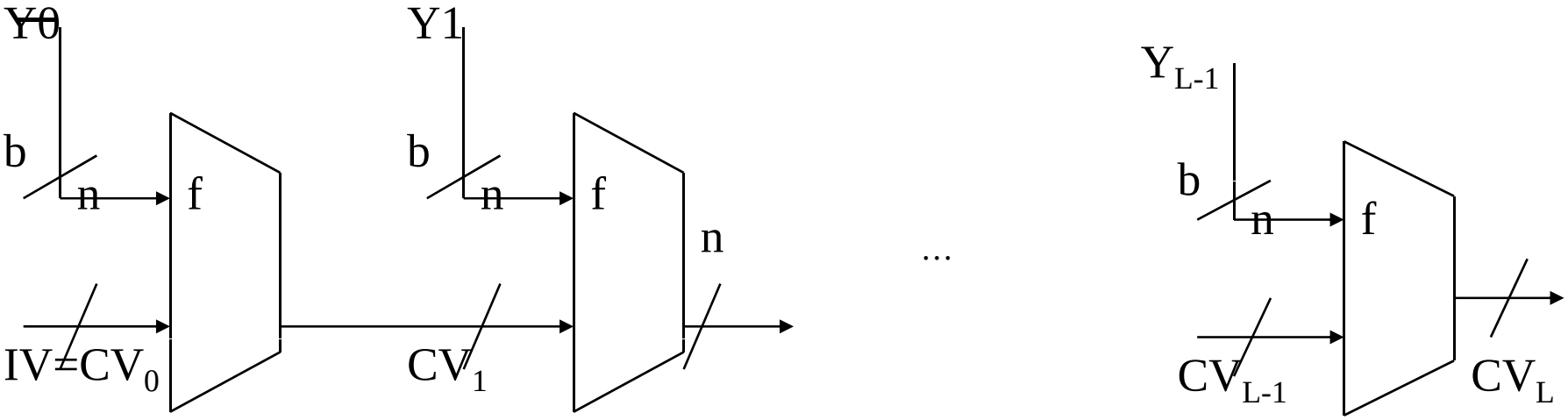
  F(X,Y,Z) =(X&Y)|((~X)&Z)
  G(X,Y,Z) =(X&Z)|(Y&(~Z))
  H(X,Y,Z) =X^Y^Z
  I(X,Y,Z)=Y^(X|(~Z))
  （**&**与，**|**或，**~**非，**^**异或）
- 常数**ti**可以如下选择：

  在第**i**步中，**ti**是2^{32}***abs(sin(i))**的整数部分，**i**的单位是弧度。

- 四轮完成后，说明对一个**512**分组的处理完成，同时新的**ABCD**产生。
- 下一个处理基于**ABCD**和新的**512**分组
- 直到所有的分组全部处理完，**ABCD**的级联输出为散列值。

- *单向散列函数的性质*
  - 散列函数的目的是为文件、报文或其他数据产生一个“指纹”，所以要求具备如下性质：
  - (1)广泛适用性　函数H适用于任何大小的数据分组；
  - (2)码长固定性　函数H产生定长输出，一个短报文的散列与百科全书报文的散列将产生相同长度的散列码；
  - (3)易计算性　对于任何数据M，计算H（M）是容易的；
  - (4)单向不可逆性　无法根据散列码倒推报文，这就是上面提到的单向函数性质；
  - (5)弱单向性　对于任意给定的数据X，要计算出另一个数据Y，使H（X）=H（Y），这在计算上是不可行的，这就是弱单向散列函数性质；
  - (6)强单向性　要寻找任何一对数据（X，Y），使得H（X）=H（Y），这在计算上是不可行的，这就是强单向散列函数性质，即对于不同的报文不能产生相同的散列码。

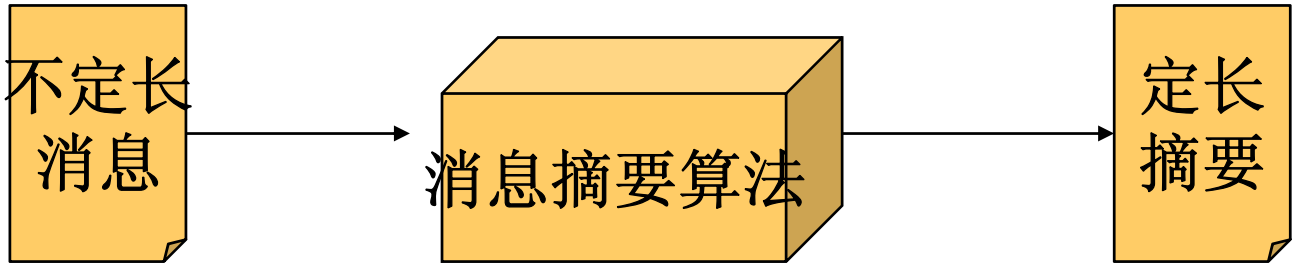
- *散列函数的一般结构*
  - 散列函数是建立在压缩函数基础之上的，它通过对消息分组的反复迭代压缩，生成一个长度固定的散列值。



IV=初始向量**Cvi**=链接变量 **Yi**=第**i**个输入分组 **f**=压缩函数
L=输入的分组数 **n**=散列值长度 **b**=输入分组长度
图**3-4** 散列函数总体结构

- 如果压缩函数是抗冲突的，那么迭代函数的合成值也是抗冲突的，也就是说该散列函数也是抗冲突的。
- 设计安全的散列函数的关键就是设计安全的、抗冲突的压缩函数。
- 总的来说，任何散列函数必定存在冲突，因为将长度至少等于分组长度b的数据映射为长度为n的散列值，其中b＞n，所以冲突是肯定存在的。而需要做的是要把寻找冲突在计算上变为不可能。

- 对这类算法的攻击方式
- *一个安全的摘要算法在设计时必须满足两个要求*
  - 其一是寻找两个输入得到相同的输出值在计算上是不可行的，这就是我们通常所说的抗冲突的；
  - 其二是对一个输出，能得到给定的输入在计算上是不可行的，即不可从结果推导出它的初始状态。
- *其实主要还是前一个条件，因为从理论上很容易证明后面一个条件基本上都是可以满足的。*



- 摘要算法对任意长的原文产生定长的摘要，按照香农的信息论，当原文的长度超过一定的程度的时候，摘要中就无法记录原文中的所有信息，这意味着存在着信息的丢失，所以说理论上不可能从签名中恢复原文。
- 一般的攻击是碰撞攻击：寻找两个输入得到相同的输出值(注意计算上的可行)

- 最主要的一点是：摘要算法的用途决定了，它只要能找到碰撞就足以让它失效，并不需要找到原文：就意味着两个不同的文件可以产生相同的“指纹”，这样就可以伪造签名。
- 比如操作系统的用户安全机制，只要得到用户密码文件（其中记录了密码的**MD5**），然后随便生成一个碰撞的原文（不一定要跟原密码相同），就可以用这个密码登录了。

•一条消息

•**2004年8月，在美国加州圣芭芭拉召开的国际密码大会上，山东大学的王小云教授宣布了她及她的研究小组近年来的研究成果MD5等四个著名密码算法的破译结果。**

•**2005年2月7日，美国国家标准技术研究院发表申明，SHA-1没有被攻破，并且没有足够的理由怀疑它会很快被攻破。而仅仅在一周之后，王小云就宣布了破译SHA-1的消息。**

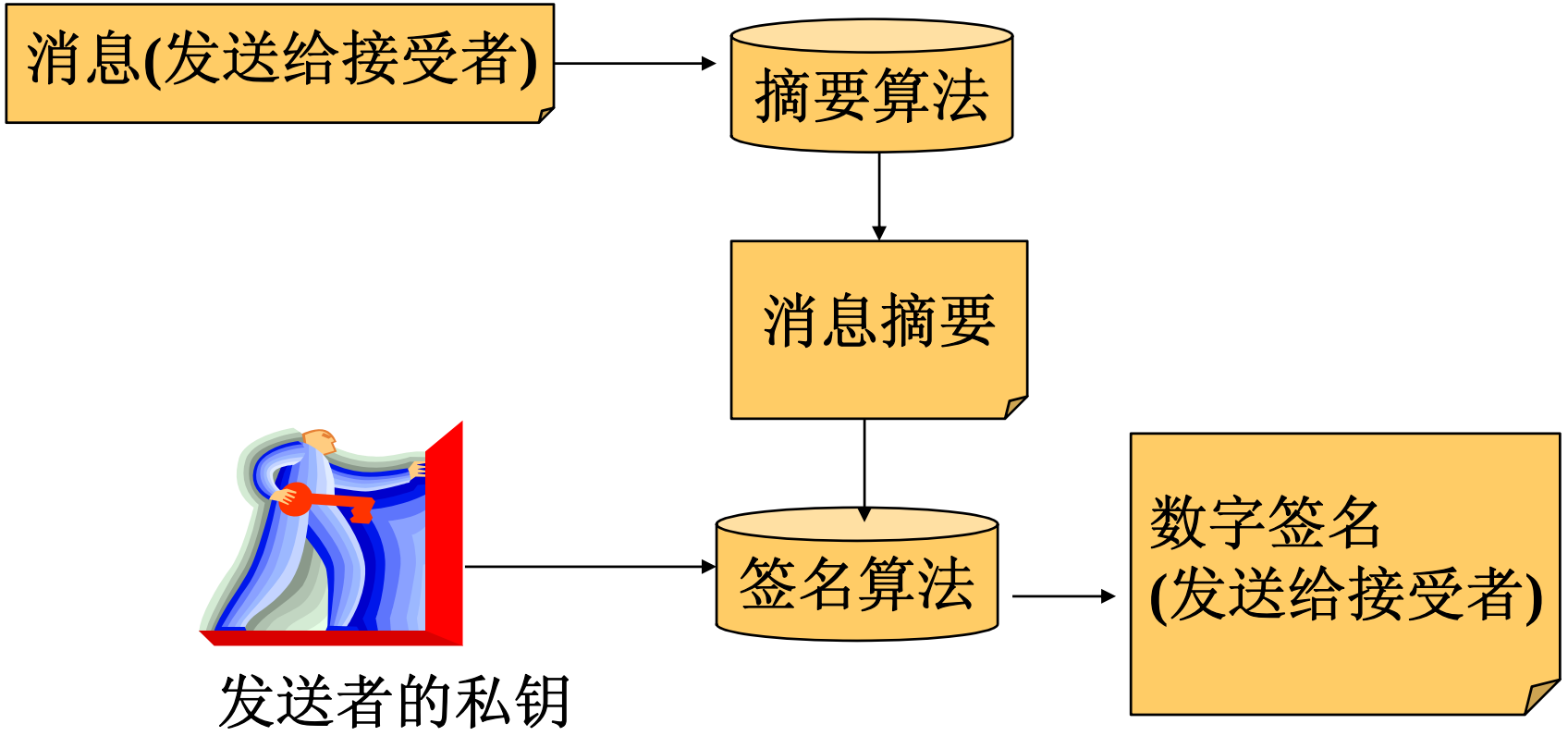
•**其破解报告中附了两对1024原文的碰撞例子，计算量在一小时左右。**

三 数字签名

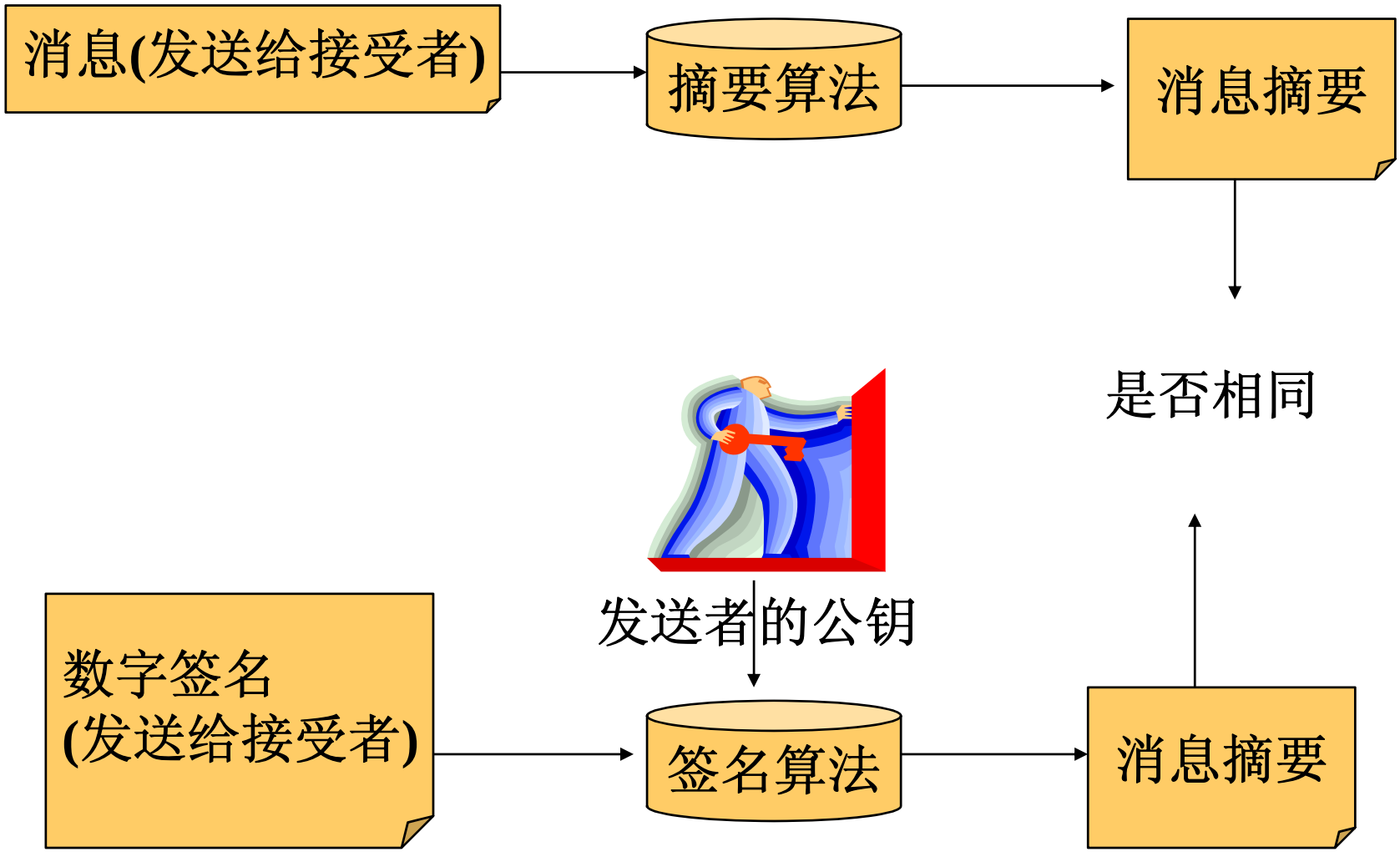
- 证明消息确实是由发送者签发的
- 并且可以用于验证数据或程序的完整性
- 它和传统的手写签名类似，满足以下条件：
  - 收方可以确认或证实签名确实是由发方签名的
  - 签名不可伪造
  - 签名不可重用，签名是消息的一部分，不能把签名移到其它消息上
  - 签名不可抵赖
  - 第三方可以确认收发双方之间的消息但不能篡改

- 目前已有大量的数字签名算法，如RSA数字签名算法、ElGamal数字签名算法、Fiat-Shamir 数 字 签 名 算 法 、 Guillou-Quisquarter数字签名算法、Schnorr数字签名算法、Ong-Schnorr-Shamir数字签名算法、美国的数字签名标准/算法（DSS/DSA）、椭圆曲线数字签名算法和有限自动机数字签名算法等。

- 数字签名实质就是把一个特定的数据与某个人相关联，该数据代表这个人。
- 它是一种重要的消息摘要。
- 包括两个部分签名和验证。



- 消息和数字签名是一起发给接受者。接受者通过签名来确定发送者的身份以及数据的完整性。



- ***1 RSA数字签名体制***

公开密钥n：两素数p和q的乘积(p,q必须保密)

e：与(p-1)(q-1)互素

私钥d：$e \times d \bmod [(p-1)(q-1)]=1$(辗转相除法)

等价表示为$d=e^{-1} \bmod [(p-1)(q-1)]$

签名：$r=m^d \bmod n$

验证：$m=r^e \bmod n$

缺点：

产生密钥很麻烦，受到素数产生技术的限制，很难做到一次一密

为保证安全性，n要600bit以上，运算代价高，速度慢。

•2 离散对数签名体制

•ElGamal数字签名算法、Schnorr数字签名算法非常相似。基于离散对数问题的一般数字签名的两个例子。

•相关参数

选择大素数**p**，**q**
q是**p-1**或**p-1**的大素数因子
选择**g**，值在**1**和**p**之间，且满足$g^q \equiv 1 \bmod p$
私钥**x**小于**q**
公钥**y**= $g^x \bmod p$ (g,p也公开)

- *2 离散对数签名体制—签名*
- 给定要签名的消息为**M**
- 首先选择一个小于**q**并互素的随机数**k**
- 计算**r= g^kmod p**
- 计算**s=(H(M)-xr)k^{-1} mod(p-1)**。
- 把消息和签名结果（M，r ，s ）发送给接收者。
- 由于r 和 s中都引入了随机数k的影响，所以即使是对于同一个消息，由于k的不同，也会导致签名结果（ r ， s ）的变化。所以称其为随机化的数字签名。

•2 离散对数签名体制—认证过程

•接收方收到消息和签名结果（M，r ，s ）后，可以按下面的步骤验证签名的有效性。

•(1) 取得发送方的公钥 y;

•(2) 预查合法性：如果p-1 ≥r≥1,那么继续后续步骤；否则，签名是不合法。

•(3) 计算 u_1：$u_1=y^r r^s \bmod p$;

•(4) 计算 u_2：$u_2=g^{H(M)} \bmod p$;

•(5) 比较u_1和u_2：如果$u_1=u_2$,表示签名有效；否则，签名无效。

•通过简单的证明，可以证明上述的签名过程与认证过程算法是成立的。

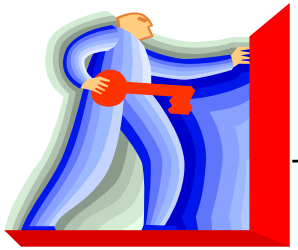
M

消息(发送给接受者)

摘要算法

消息摘要

H(M)



签名算法

数字签名
(发送给接受者)

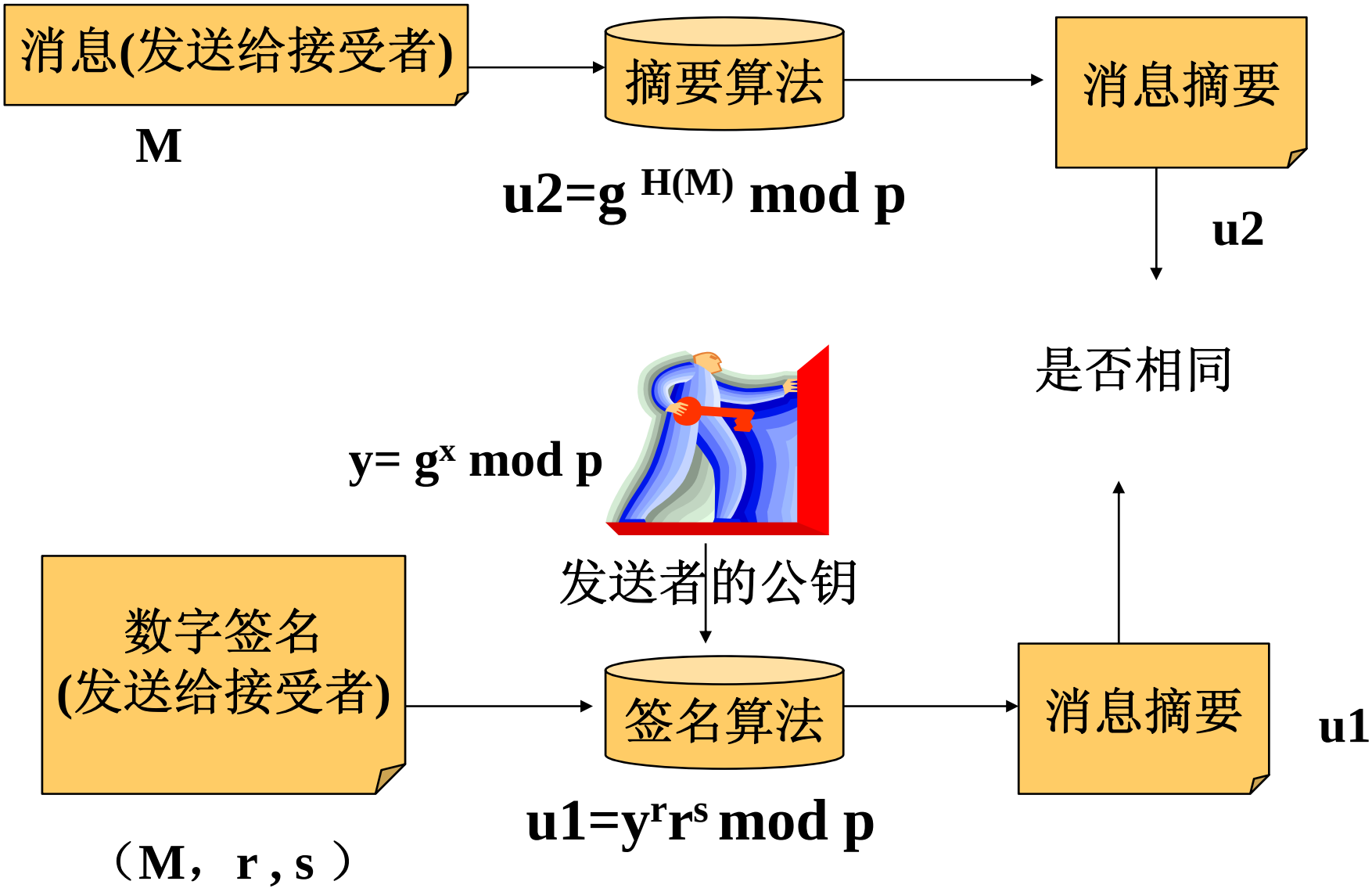
发送者的私钥

x

$r = g^k \bmod p$

$s = (H(M) - xr)^{k-1} \bmod (p-1)$。

（M，r，s）


消息(发送给接受者)
M
摘要算法
u2=g H(M) mod p
消息摘要
u2
是否相同
y= g^x mod p
发送者的公钥
数字签名
(发送给接受者)
（M，r，s）
签名算法
u1=y^r r^s mod p
消息摘要
u1

- ***2 ElGamal数字签名体制的变形***
- 把签名过程中的等式
  - $s=(H(M)-xr)k^{-1} \bmod (p-1)$
  - $H(M)=(xr+ks) \bmod (p-1)$
- 修改为$u=x\times v+k\times w \bmod (p-1)$称为签名方程
- 其中，如果u=H(M)，v=r，w=s，那么签名方程就是ElGamal签名过程中使用的等式。

•如果把u、v、w分别对应不同次序的H（M）、r、s，就可得到其他的ElGamal数字签名变体体制。表中列举6种不同的变形。

序号	u	v	w	签名方程	认证等式
1	H（M）	r	s	$H(M)=xr+ks \bmod (p-1)$	$g^{H(M)}=(g^x)^r r^s \bmod p$
2	H（M）	s	r	$H(M)=xs+kr \bmod (p-1)$	$g^{H(M)}=(g^x)^s r^r \bmod p$
3	s	r	H(M)	$s=xr+kH(M) \bmod (p-1)$	$g^s=(g^x)^r r^{H(M)} \bmod p$
4	s	H(M)	r	$s=xH(M)+kr \bmod (p-1)$	$g^s=(g^x)^{H(M)} r^r \bmod p$
5	r	s	H(M)	$r=xs+kH(M) \bmod (p-1)$	$g^r=(g^x)^s r^{H(M)} \bmod p$
6	r	H(M)	s	$r=xH(M)+ks \bmod (p-1)$	$g^r=(g^x)^{H(M)} r^s \bmod p$

- ***2 ElGamal数字签名体制的安全性讨论***
  - (1) 面对不知道明文密文对的攻击
  - 由于攻击者不知道用户私钥x，所以要伪造用户的签名，需要先选定一个r(或s)，然后实验求取另外一个值s(或r)。都是属于求解离散对数问题。
  - 目前还没有找到计算离散对数问题的多项式时间算法。为了抗击已知的攻击，p应该至少是150位以上的十进制整数，并且p-1至少有一个大的素因子。

- ***2 ElGamal数字签名体制的安全性讨论***
  - (2) 面对已经知道明文密文对的攻击
  - 攻击者知道（r，s）是消息M的合法签字。在这种情况下，如果用户泄漏常常使用的K（而不是每次随机产生的K），攻击者就有可能求解出用户的私钥x。否则，要求解出用户的私钥x也是难以得逞的。

- ***3 DSS数字签名体制***
  - 美国国家标准技术研究所(NIST)1994年5月19日公布了数字签名标准(DSS, Digital SignatureStandard)，标准采用的算法便是DSA，密钥长度为512~1024位。密钥长度愈长，签名速度愈慢，制约运算速度的主要因素是大数的模指数运算。
  - Digital Signature Algorithm(DSA)是Schnorr和ElGamal签名算法的变种。DSS是由美国国家标准化研究院和国家安全局共同开发的。由于它是由美国政府颁布实施的，主要用于与美国政府做生意的公司，其他公司则较少使用，它只是一个签名系统，而且美国政府不提倡使用任何削弱政府窃听能力的加密软件，认为这才符合美国的国家利益。

- *3 DSS数字签名体制—DSA基本过程*

公开密钥

p L位长的素数，L从512到1024且是64的倍数

q 160 位长且是 p-1 的素因子 (可在一组用户中共享)

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

$y = g^{x} \bmod p$ (一个p位的数)

私人密钥

$x < q$ （一个 160 位的数）

签名

k 选取一个小于 q 的随机数

r (签名) $= (g^{k} \bmod p) \bmod q$

s (签名) $= (k^{-1} (H(m) + x\, r) \bmod q$

验证

$w = s^{-1} \bmod q$

$u_1 = (H(m) \times w) \bmod q$

$u_2 = (r\, w) \bmod q$

$v = ((g^{u_1} \times y^{u_2}) \bmod p) \bmod q$

如果 $v = r$， 则签名被验证。

实际计算DSA时可通过预计算来加快速度。
r的值与消息无关，因此可产生一串随机的k，并且预先计算出与之对应的r值，k^{-1}。

• 3 DSS数字签名体制—DSA的变形

公开密钥

p L位长的素数，L从512到1024且是64的倍数

q 160 位长且是 p-1 的素因子 (可在一组用户中共享)

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

$y = g^{x} \bmod p$ (一个p位的数)

私人密钥

$x < q$ （一个 160 位的数）

签名

k 选取一个小于 q 的随机数

r (签名) $= (g^{k} \bmod p) \bmod q$

s (签名) $= (k^{-1} (H(m) + x\,r)) \bmod q$

验证

$w = s^{-1} \bmod q$

$u_1 = (H(m) \times w) \bmod q$

$u_2 = (r\,w) \bmod q$

$v = ((g^{u_1} \times y^{u_2}) \bmod p) \bmod q$

如果 $v = r$， 则签名被验证。

不用计算$\mathbf{k^{-1}}$。

签名时，产生小于**q**的随机数**k**，**d**。

$$r = (g^{k} \bmod p) \bmod q$$

$$s = (H(m) + x\,r) \times d \bmod q$$

$$t = k\,d \mod q,$$

通过计算来验证签名：

$$w = t/s \bmod q$$

$$u_1 = (H(m) \times w) \bmod q$$

$$u_2 = (r\,w) \bmod q$$

$$v = ((g^{u_1} \times y^{u_2}) \bmod p) \bmod q$$

：$v = r$，则签名被验证有效。

- *4 安全性*
- 绝大多数公开密钥算法都基于下面3个疑难问题：
  - 背包问题：给定一个互不相同的数组成的集合，找出一个子集其和为N
  - 离散对数：若p是素数，g和M是整数，找出x满足$g^x \equiv M \pmod p$
  - 因子分解：设N是两个素数的乘积，则有
    - 分解N
    - 给定整数M和C，寻找d满足$M^d \equiv C \pmod N$
    - 给定整数e和C，寻找M满足$M^e \equiv C \pmod N$
    - 给定整数x，判定是否存在整数y满足$x \equiv y^2 \pmod N$

- 公开密码学如此狭窄的数学基础令人担忧，因子分解或者离散对数问题的突破将使所有公开密钥算法不安全。

- *5 数字签名的应用*
  - 最早的应用之一用来禁止对核试验条约的验证。
  - 美国和前苏联互相允许把地震测试仪放入另一个国家，以便对核试验进行监控。
  - 问题是每个国家需要确信监控国没有篡改从被监控国的地震仪传来的数据
  - 同时监控国要确信监测器只发送规定的需要监测的消息。

- *6 各种形式的数字签名－多重签名*
- 数字签名能实现用户对消息的认证，然而在实际应用中往往需要多个用户对同一消息进行数字签名认证，即多重数字签名。多重数字签名以密钥共享为基础，结合密码算法实现。
- 密钥共享是指，将一个密钥s分割成由N个参与者掌握的部分密钥$m_1,m_2,...m_n$，参与者的某些制定组合可以通过出示其掌握的秘密份额恢复密钥s，而其他组合则不能得到s的任何信息。

- 优点：
  - 攻击者要想得到数字签名密钥必须得到t个部分的密钥
  - 即使某个或某些部分密钥丢失，也不会影响到整个密钥
  - 实现权利的分配，避免滥用职权。
- 缺点
  - 简单使用密钥共享方案管理签名密钥是满足不了实际的应用，一旦使用该密钥进行数字签名，该密钥即被解密并会被签名的执行者所掌握，因而可在随后进行单独的签名。
- 必须有这样一种机制，使参与签名的密钥共享者可用他们的部分密钥计算部分签名，再由部分签名最终生成对消息的真正签名，而在这一系列过程中并不泄漏任何有关密钥和部分密钥的信息。

- ***6 各种形式的数字签名—***

不可抵赖的数字签名

- 一般的数字签名能够被准确复制。这个性质有时是有用的，比如公开宣传品的发布。
- 在其它时间，它可能有问题。
- 想象一下数字签名的私人或商业信件。如果到处散布那个文件的许多拷贝，而每个拷贝又能够被任何人验证，这样可能会导致窘迫或勒索。
- 最好的解决方案是数字签名能够被证明是有效的，但没有签名者的同意，接收者不能把它给第三方看。

- *6 各种形式的数字签名—*

不可抵赖的数字签名

- 需求描述：

Alice软件公司发布DEW软件。为了确信软件中不带病毒，他们在每个拷贝中包括一个数字签名。然而，他们只想软件的合法买主能够验证数字签名，而使用盗版者则不能。同时，如果DEW拷贝中发现有病毒，Alice软件公司应该不可能否认。

- 不可抵赖签名适合于这类任务。

- *6 各种形式的数字签名－*

不可抵赖的数字签名

- 数学描述是复杂的，但其基本思想是简单的：
- （1）Alice向Bob出示一个签名；
- （2）Bob产生一个随机数并送给Alice；
- （3）Alice利用随机数和其私钥进行计算，将计算结果送给Bob。Alice只能计算该签名是否有效。
- （4）Bob确认这个结果。

•6 各种形式的数字签名—

不可抵赖的数字签名

•Bob不能转而让Carol确信Alice的签名是有效的，因为Carol不知道Bob的数字是随机数。

•Carol只有在她与Alice本人完成这个协议后才能确信Alice的签名是有效的。

•6 各种形式的数字签名－

不可抵赖的数字签名

•这个解决方法并不完美，在某些情况下，**Bob**让**Carol**确信**Alice**的签名有效是可能的。

- *例如，**Bob**买了**DEW**的一个合法拷贝，他能在任何时候验证软件包的签名。*
- *然后，**Bob**使**Carol**相信他是来自于**Alice**软件公司的销售商。他卖给她一个**DEW**的盗版。*
- *当**Carol**试图验证**Bob**的签名时，他同时要验证**Alice**的签名。当**Carol**发给**Bob**随机数时，**Bob**然后把它送给**Alice**。当**Alice**响应后，**Bob**就将响应送给**Carol**。于是**Carol**相信她是该软件的合法买主，尽管她并不是。*

- *6 各种形式的数字签名－*

不可抵赖的数字签名

- 即使如此，不可抵赖的签名仍有许多应用
- 在很多情况中，**Alice**不想任何人能够验证她的签名。她不想她的个人通信被媒体核实、展示并从文中查对，或者甚至在事情已经改变后被验证。
- 如果她对她卖出的信息签名，她不希望没有对信息付钱的那些人能够验证它的真实性。控制谁验证她的签名是**Alice**保护她的个人隐私的一种方法。

下面是不可抵赖数字签名的算法实现：

首先，公开一个大素数p和一个本原元g，p和g可由一组签名者公用。Alice有一个私人密钥x和一个公开密钥$h = g^x \bmod p$。

对消息m签名时， Alice计算：$z = m^x \bmod p$ 即可。验证稍微复杂一些：

(1) Bob 选择两个小于 p 的随机数 a 和 b，并将下式计算结果 c 发送给 Alice：

$$c = z^a h^b \bmod p$$

(2) Alice先计算$t = x^{-1} \bmod (p\text{-}1)$，再将下式计算结果$d$发送给Bob：

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

(3) Bob 进一步确认下式是否成立：

$$d \equiv m^a g^b \pmod p$$

如果此式成立，Bob 就认为该签名是真的。

- *6 各种形式的数字签名—*

  不可抵赖的数字签名
- 相关的概念是“受托不可抵赖签名”。
- 设想**Alice** 为**Toxins**公司工作，并使用不可抵赖的签名协议发送控告文件给报纸。
- **Alice**能够对报社记者验证她的签名，但不向其它任何人验证签名。
- 然而执行总裁**Bob**怀疑文件是**Alice**提供的，他要求**Alice**执行否认协议来澄清她的名字，**Alice**拒绝了。
- 如果**Trent**是法院系统，否认协议只能由**Trent**执行外，受托不可抵赖签名类似不可抵赖签名。**Bob**不能要求**Alice**执行否认协议，只有**Trent**能够。

- *6 各种形式的数字签名—*

  指定的确认者签名

- 需求描述：
  - **Alice**公司销售**DEW**软件的生意非常兴隆，事实上，**Alice**验证不可抵赖签名的时间比编写新的功能部件的时间更多。
  - **Alice**很希望有一种办法可以在指定一个特殊的人或团体负责对整个公司的签名验证。
  - 结果表明，用指定的确认人签名是可行的。
  - **Alice**能够对文件签名，而**Bob**相信签名是有效的，但他不能使第三方相信。同时，**Alice**能够指定**Carol**作为她签名后的确认人。**Alice**甚至事先不需要得到**Carol**的同意，她只需要**Carol**的公开密钥。如果**Alice**不在家，已经离开公司，或者突然死亡了，**Carol**仍然能够验证**Alice**的签名。

下面叙述了 Alice 怎样签署消息，而 Bob 又怎样验证消息，才能使得 Carol 在以后某个时刻向 Dave 验证 Alice 的签名。

首先，公开一个大素数p和一个本原元素g，p和g由一组用户公用，两个素数的积n也是公开的。Carol拥有一个私人密钥z和一个公开密钥$h = g^x \bmod p$。

在该协议中，Alice 能够签署消息 m，使得 Bob 确信此签署是合法的，但是，不能使第三方确信。

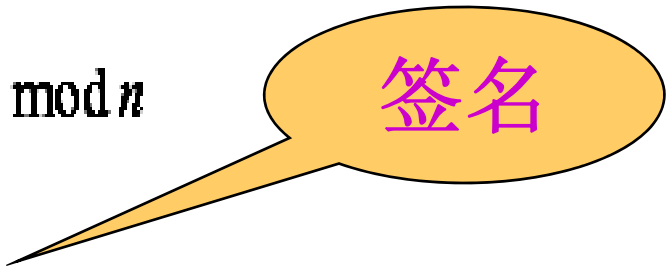
(1) Alice 选择一个随机数 x，并计算：

$$a = g^x \bmod p$$

$$b = h^x \bmod p$$

她计算 m 的散列值 $H(m)$，以及 a 和 b 并置起来的散列值 $H(a, b)$。然后计算：

$$j = (H(m) \oplus H(a,b))^{1/3} \bmod n$$



并将 a、b 和 j 发送给 Bob。

(2) Bob 选择两个小于 p 的随机数 s 和 t，并发送给 Alice：

$$c = g^s h^t \bmod p$$

(3) Alice 选择一个小于 p 的随机数 q,并发送给 Bob：

$$d = g^q \bmod p$$

$$e = (cd)^x \bmod p$$

(4) Bob 把 s 和 t 发送给 Alice。

(5) Alice 确认：

$$g^s h^t \equiv c \pmod p$$

然后将 q 发送给 Bob。

(6) Bob 确认：

$$d \equiv g^q \pmod p$$

$$e / a^q \equiv a^s b^t \pmod p$$

如果 Alice 和 Bob 全都验算完毕，Bob 就认为这个签名是真实的。

Bob 不能用这个证明的副本使 Dave 相信这个签名是真实的，但是，Dave 能够和 Alice 指定的确认者 Carol 一起构造一个协议。下面叙述 Carol 怎样使 Dave 相信 a 和 b 构成一个有效签名。

(1) Dave 选择两个小于 p 的随机数 u 和 v，并发送给 Carol：

$$k = g^u a^v \bmod p$$

(2) Carol 选择一个小于 p 的随机数 w，并发送给 Dave：

$$l = g^w \bmod p$$

$$y = (kl)^z \bmod p$$

(3) Dave 将 u 和 v 发送给 Carol。

(4) Carol 确认：

$$g^u a^v \equiv k \pmod{p}$$

然后将 w 发送给Dave

(5) Dave 确认：

$$g^w \equiv l \pmod{p}$$

$$y / h^w \equiv h^u b^v \pmod{p}$$

如果 Carol 和 Dave 全都验算完毕，Dave 就认为这个签名是真实的。

- *6 各种形式的数字签名—*

  指定的确认者签名

- **Carol**可能是版权事务所、政府机构、其它的很多事物。这个协议允许组织机构把签署文件的人同帮助验证签名的人分开。

•6 各种形式的数字签名—团体签名

- 需求描述：
  - 一个公司有几台计算机，每台都连在局域网上。公司的每个部门有它自己的打印机（也连在局域网上），并且只有本部门的人员才被允许使用他们部门的打印机。
  - 因此，打印前，必须使打印机确信用户是在那个部门工作的。
  - 同时，公司想保密，不可以暴露用户的姓名。
  - 然而，如果有人在当天结束时发现打印机用得太频繁，主管者必须能够找出谁滥用了那台打印机，并给他一个帐单。

•6 各种形式的数字签名—

团体签名

•对这个问题的解决方法称为团体签名。它具有以下特性：

- *只有该团体内的成员能对消息签名；*
- *签名的接收者能够证实消息是该团体的有效签名。*
- *签名的接收者不能决定是该团体内哪一个成员签的名；*
- *在出现争议时，签名能够被“打开”， 以揭示签名者的身份。*

- 具有可信仲裁者的团体签名
- 本协议使用可信仲裁者**Trent**：
  - （1）**Trent**生成一大批公开密钥/私钥密钥对，并且给团体内每个成员一个不同的唯一私钥表。在任何表中密钥都是不同的（如果团体内有**n**个成员，每个成员得到**m**个密钥对，那么总共有**n*m**个密钥对）。
  - （2）**Trent**以随机顺序公开该团体所用的公开密钥主表。**Trent**保持一个哪些密钥属于谁的秘密记录。
  - （3）当团体内成员想对一个文件签名时，他从自己的密钥表中随机选取一个密钥。
  - （4）当有人想验证签名是否属于该团体时，只需查找对应公开钥主表并验证签名。
  - （5）当争议发生时，**Trent**知道哪个公钥对应于哪个成员。

- 这个协议的问题在于需要可信的一方。**Trent**知道每个人的私钥因而能够伪造签名。而且，**m**必须足够长以避免试图分析出每个成员用的哪些密钥。

•6 各种形式的数字签名—

盲签名

•数字签名协议的一个基本特征是文件的签署者知道他们在签署什么。

•有时候我们想要别人签署一个他们从未看过其内容的文件。

- *完全盲签名*
  - **Bob**是一个公证员，**Alice**要他签一个文件，但又不想让他知道他在签什么。**Bob**不关心文件中说些什么，他只是证明他在某一时刻公证过这个文件。他愿意这样进行：
  - （1）**Alice**取出文件并将它乘以一个随机值，这个随机值称为盲因子；
  - （2）**Alice**送这份隐蔽好的文件给**Bob**；
  - （3）**Bob**在这个隐蔽好的文件上签名；
  - （4）**Alice**将其除以隐蔽因子，留下**Bob**签过的原始文件。
  - 只有当签名函数和乘法函数是可交换时，这个协议才能有效。如果不是，有别的方法可代替乘法来修改文件。

- *完全盲签名的性质是：*
  - **Bob**在文件上的签名是有效的。签名就是**Bob**签署这份文件的证据。如果把文件给**Bob**看，**Bob**确信他签署过这份文件。
  - **Bob**不能把签署文件的行为与签署了的文件相关联。他没有办法把已签了名的文件和他在协议中收到的任何信息相关联。如果他用这个协议签了一百万份文件，他照样没有办法知道在哪种情况下他签了哪一份文件。

- *盲签名*
  - 用完全盲签名协议，**Alice**能让**Bob**签任何东西：“**Bob**欠**Alice**一百万美元”，“**Bob**欠**Alice**一袋软糖。”可能的事远远不止于此。这个协议在许多场合都无用。

- *盲签名*
  - 有一个办法可以让**Bob**知道他在签什么，同时仍保持盲签名的有用性质。这个协议的核心是分割选择技术 。
  - 如，移民局要确信有没有走私可卡因。可以搜查每一个人，但他们换用了一种概率解决办法。他们检查入境人中的十分之一。
  - 盲签名协议以类似的方式发挥作用。**Bob**将得到一大堆不同的隐蔽好的文件，他打开即检查除一个文件以外的所有文件，然后对最后一个文件签名。

- *盲签名*
  - 应用需求
  - 反间谍人员，他们的身份是秘密的，甚至反间谍机构也不知道他们是谁。
  - 这个机构的上司想给每个特工一份签名的文件，文件上写有：“这个签名文件的持有人（这里插入特工的化名）享有完全的外交豁免权。”
  - 所有的特工有他们自己的化名名单。特工们不想也不能把他们的化名送给签名机构。
  - 另一方面，签名机构也不想盲目地签特工送来的文件。聪明的特工可能会代之一条消息，象：“特工（名字）已经退休并获得一年一百万美元的养老金。签名：总统先生”。
  - 在这种情况下，盲签名可能是有用的。

- *盲签名*
  - 假设所有特工都有十个可能的化名，这些化名都是他们自己选的，别人不知道。
  - 同时假设特工们并不关心他们将在哪个化名下得到外交豁免权。
  - 再假设这个授权签名机构ALICE，特工BOB。

- （1）BOB准备了10份文件，每一个使用不同的化名，并给予那个特工外交豁免权。
- （2）BOB用不同的盲因子隐蔽每个文件。
- （3）BOB把这10份隐蔽好的文件给ALICE。
- （4）ALICE随机选择9份文件并向BOB索要每份文件的盲因子。
- （5）BOB向ALICE发送适当的盲因子。
- （6）ALICE打开（即去掉盲因子）9份文件，并确信它们是正确的——而不是退休授权。
- （7）ALICE在第十个文件上签名并把它送给BOB。
- （8）BOB去掉盲因子并读出他的新化名：“The Crimson Streek”。签署的文件在那个名字下给予他外交豁免权。

- *盲签名*
  - 分析
    - 这个协议能防止**BOB**欺骗。他要欺骗，他必须准确地预测**ALICE**不会检查哪一份文件。他这样做的机会是**n**分之一。这个协议就和先前的盲签名协议一样，并保持了它所有的匿名性质。

身份认证

- 身份认证是通信双方在实质性数据传输之前进行审查和证实对方身份的操作。
- 是网络安全的基础，是对访问者进行授权的前提。
- 身份验证(Identification)是用户向系统出示自己身份证明的过程。
- 身份认证(Authentification)是系统查核用户身份证明的过程。
- 这两个过程是判明和确认通信双方真实身份的两个重要环节，人们常把这两项工作统称为身份验证(或身份鉴别)。

身份认证—认证的方法

- 基于用户知道什么的身份认证(口令核对)
  - 基本做法：每一个合法用户都有系统给的一个用户名/口令对，用户进入时，系统要求输入用户名、口令，如果正确，则该用户的身份得到了验证。
  - 优点：方法简单。
  - 缺点：用户取的密码一般较短，且容易猜测，容易受到口令猜测攻击；口令的明文传输使得攻击者可以通过窃听通信信道等手段获得用户口令；加密口令还存在加密密钥的交换问题。

身份认证—认证的方法

- 基于用户拥有什么的身份认证
  - IC卡认证
    - *基本做法*：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，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。
    - *优点*：IC卡由专门的厂商通过专门的设备生产，通过IC卡硬件不可复制来保证用户身份不会被仿冒。
    - *缺点*：每次从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。因此，静态验证的方式还是存在根本的安全隐患。

身份认证—认证的方法

- 基于用户拥有什么的身份认证
  - 动态口令
  - *基本做法*：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。
  - *优点*：动态口令技术采用一次一密的方法，有效地保证了用户身份的安全性。
  - *缺点*：若如果客户端硬件与服务器端程序的时间或次数不能保持良好的同步，就可能发生合法用户无法登陆的问题。并且用户每次登录时还需要通过键盘输入一长串无规律的密码，一旦看错或输错就要重新来过，用户的使用非常不方便。

身份认证—认证的方法

- *基于用户是谁身份认证*
  - 生物特征认证
  - *基本做法*：采用每个人独一无二的生物特征来验证用户身份的技术。常见的有指纹识别、虹膜识别等。
  - *缺点：*
  - 生物特征识别的准确性和稳定性还有待提高
  - 由于研发投入较大和产量较小的原因，生物特征认证系统的成本非常高，目前只适合于一些安全性要求非常高的场合如银行、部队等使用，还无法做到大面积推广。

身份认证—认证的方法

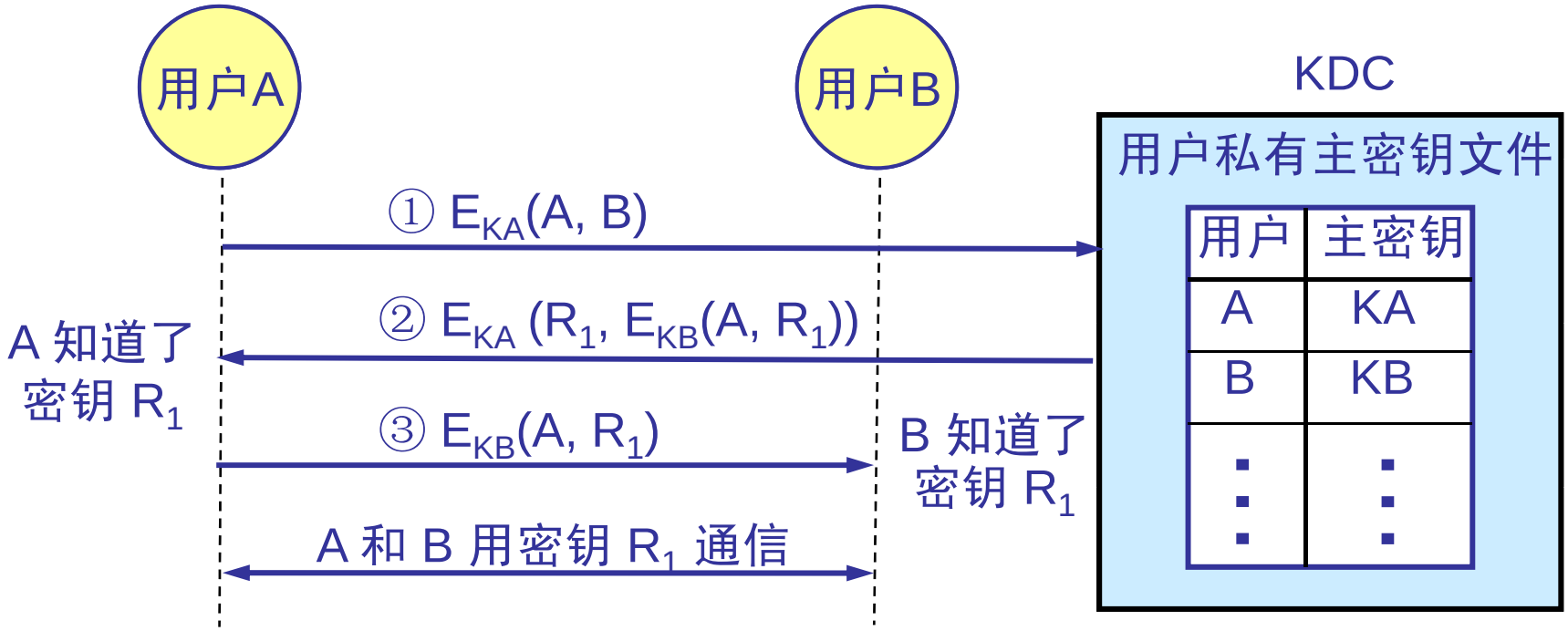
- 认证服务是运行在一个安全机器上的一个程序，通过发出一个需求、给出一个响应这样一个过程在网络上进行认证。
- 软件认证和硬件认证
- 从认证需要验证的条件来看，可以分为单因子认证和双因子认证
- 从认证信息来看，可以分为静态认证和动态认证

身份认证—Kerberos认证协议

- **Kerberos系统简介**
  - 在希腊神话中，**Kerberos**是守护地狱之门的三头狗。
  - **(MIT)在80年代首先提出并实现的,是该校Athena计划的一部分。**
  - Kerberos 是一种基于私钥加密算法的，需要可信任的第三方作为认证服务器的网络认证系统。Kerberos设计的目的是解决在分布网络环境下，用户访问网络资源时的安全问题。

- ***Kerberos系统简介***
  - Kerberos协议中有三个通信参与方，需要验证身份的通信双方和一个双方都信任的第三方KDC。
  - 将发起认证服务的一方称为客户方，客户方需要访问的对象称为服务器方。
  - 在Kerberos中客户方是通过向服务器方递交自己的“凭据”（Ticket）来证明自己的身份的，该凭据是由密匙分配中心（KDC）的第三方服务中心专门为客户方和服务器方在某一阶段内通信而生成的。

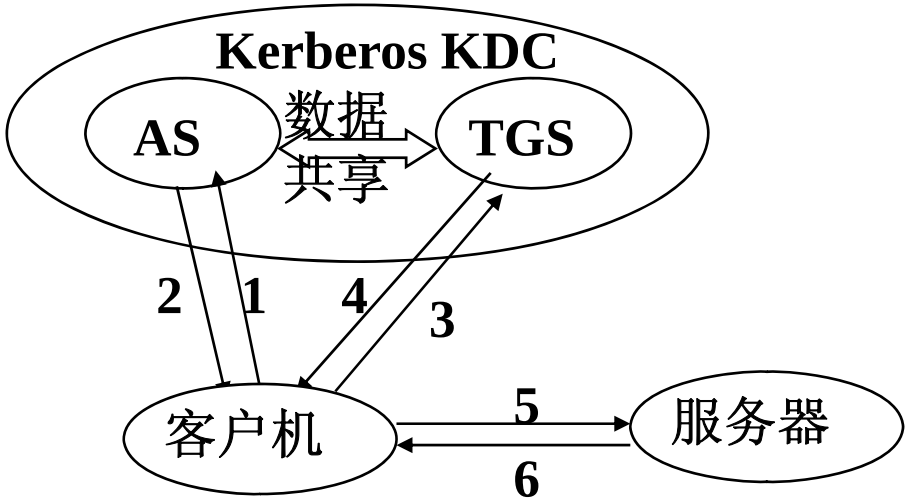
常规密钥分配协议



密钥 **R1**又称为会话密钥，用于这一次通信中

- ***Kerberos的组成***
  - • Kerberos 应 用 程 序 库 (Kerberos Applicaton library)
  - •加密/解密库(encryption/decryption library)
  - •数据库程序库(database library)
  - • 数 据 库 管 理 程 序 (database administration programs)
  - •KDBM服务器(KDBM server)
  - •认证服务器(authentication server)
  - •数据库复制软件(db propagation software)
  - •用户程序(user programs)
  - •应用程序**(applications)**

- *Kerberos的认证方案*
  - 以第四版说明处理过程



Kerberos协议运作步骤

认证服务器（AS），用于在登陆时验证用户的身份；与每个用户共享一个秘密口令。

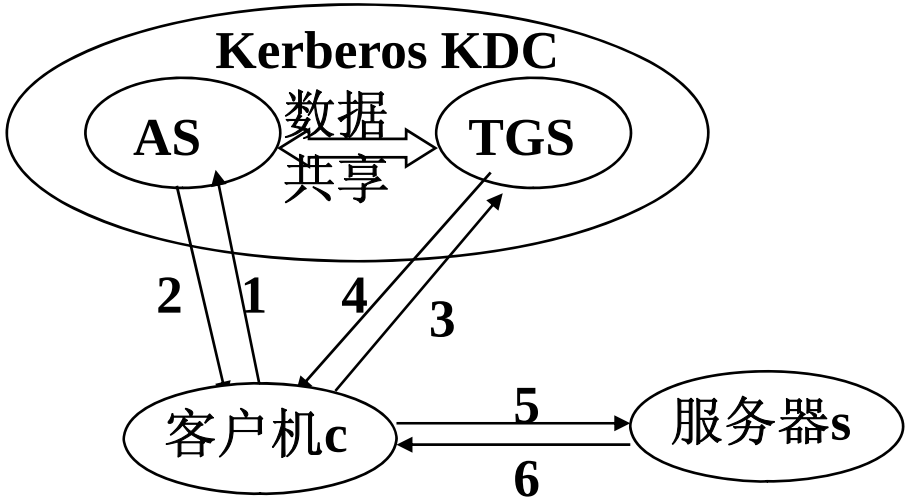
授予许可证服务器（TGS），发放“身份证明许可证”；与AS共享一个秘密口令K_{tgs}

服务器（Server），客户请求工作的实际执行者。

Kerberos有一个所有客户和自己安全通信所需的秘密密钥数据库**(KDC).**

Kerberos知道每个人的秘密密钥，所以能产生消息向每个实体证实另一个实体的身份。

同时还能产生会话密钥，只供一个客户机和一个服务器使用，会话密钥用来加密双方的通信消息，通信完毕，即被销毁。



Kerberos协议运作步骤

相关符号

c：用户C身份码

s：应用服务器S身份码

ADx：x的网络地址

tgs：TGS身份码

Kx：x的密钥

Kx,y：x与y的会话密钥

{...}Kx：以Kx加密扩号中的信息

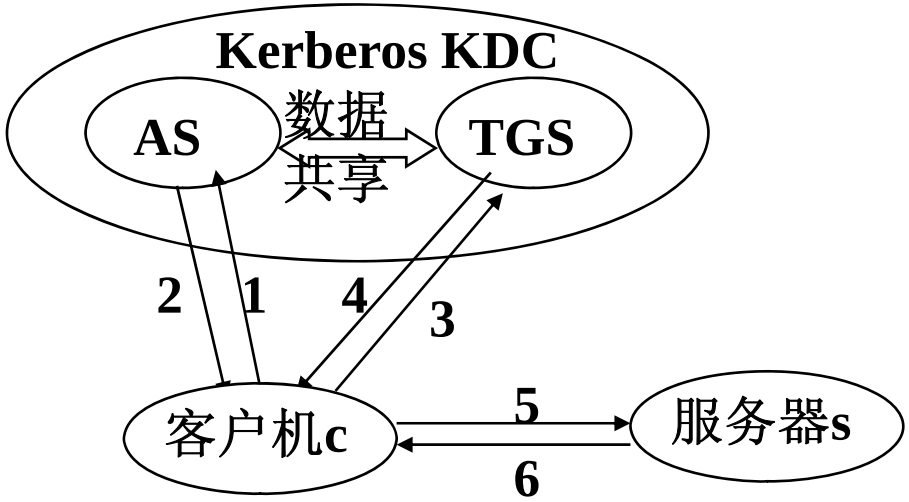
life：生存周期

TS:时间戳

Ac:身份验证器(自己生成)

$T_{x,y}$:x与y间的凭据

阶段1



Kerberos协议运作步骤

客户C从AS处申请(步骤1)及取得初始凭据(步骤2)；

认证业务交换过程：完成客户向KDC请求与TGS通信时使用的凭据以及会话密钥的过程。

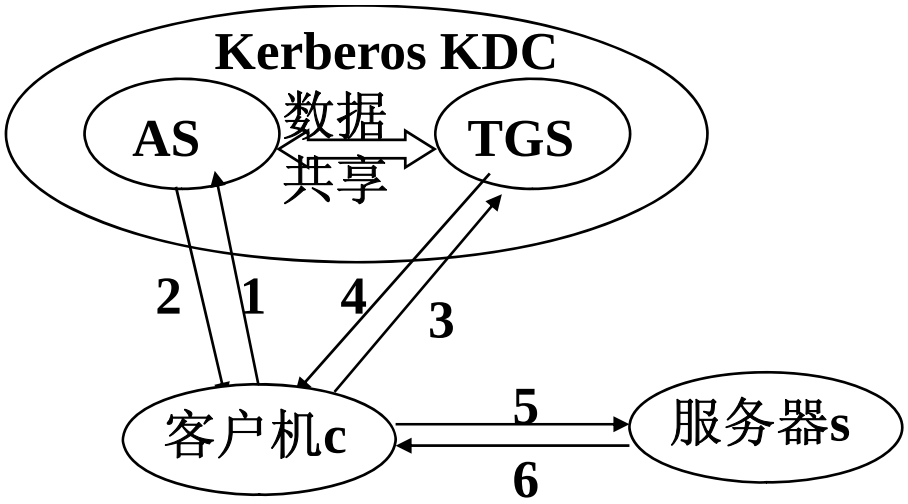
(步骤1):C->AS:{c, TS1}

(步骤2):AS->C:{$K_{c,tgs}$, tgs, TS2, life1, $T_{c,tgs}$}K_c

初始凭据$T_{c,tgs}$＝{$K_{c,tgs}$, c, ADc, tgs, TS2, life1}K_{tgs}

“凭据”类似常规密钥分配协议中的$E_{KB}(A, R1)$

阶段2：



Kerberos协议运作步骤

客户C从TGS处申请(步骤3)及取得应用服务器S的票(步骤4)；*授权凭据业务交换过程*

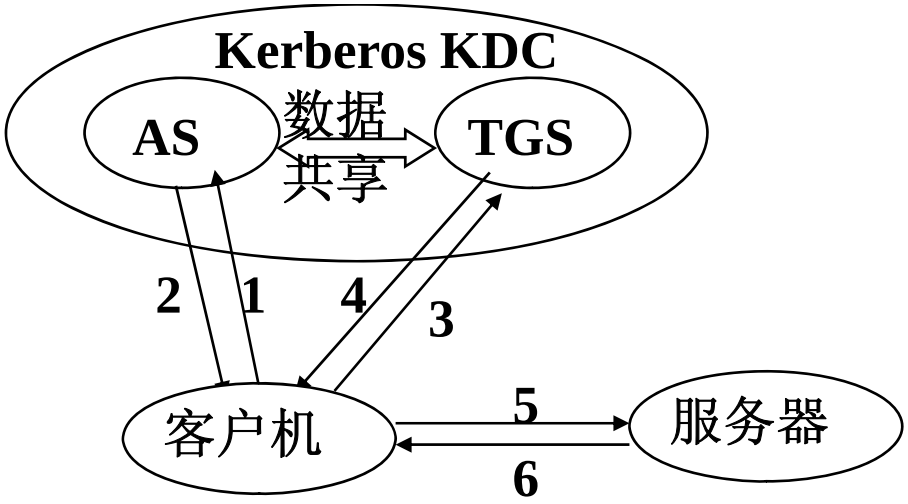
(步骤3)C->TGS:{s, $T_{c,tgs}$, Ac}

Ac＝{c, AD_c, TS3}$K_{c, tgs}$

初始票$T_{c,tgs}$＝{$K_{c,tgs}$, c, ADc, tgs, TS2, life1}K_{tgs}

TGS可以检查身份验证器Ac来证明客户的名称和地址是否与凭据中的名称和接受消息的地址一致。

阶段2：



Kerberos协议运作步骤

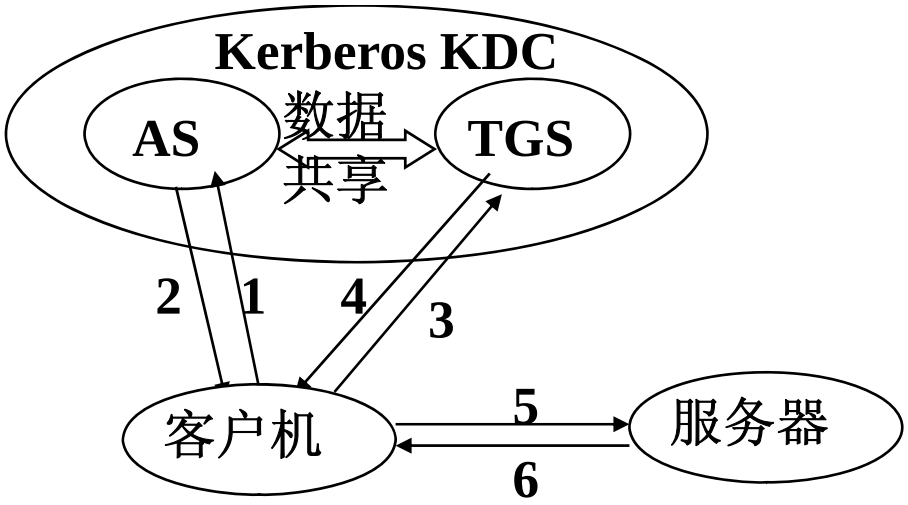
客户C从TGS处申请(步骤3)及取得应用服务器S的票(步骤4)；

(步骤4)TGS->C:$\{s, \mathbf{K_{c,s}}, TS4, T_{c,s}\}K_{c,tgs}$

$T_{c,s} = \{K_{c,s}, c, AD_c, s, TS4, life2\}K_s$

授权凭据业务交换过程，是客户方C向TGS请求与最终的应用服务器进行通信所需要的凭据和会话密钥的过程。

阶段3：



Kerberos协议运作步骤

客户C向应用服务器S请求服务(步骤5,6)。

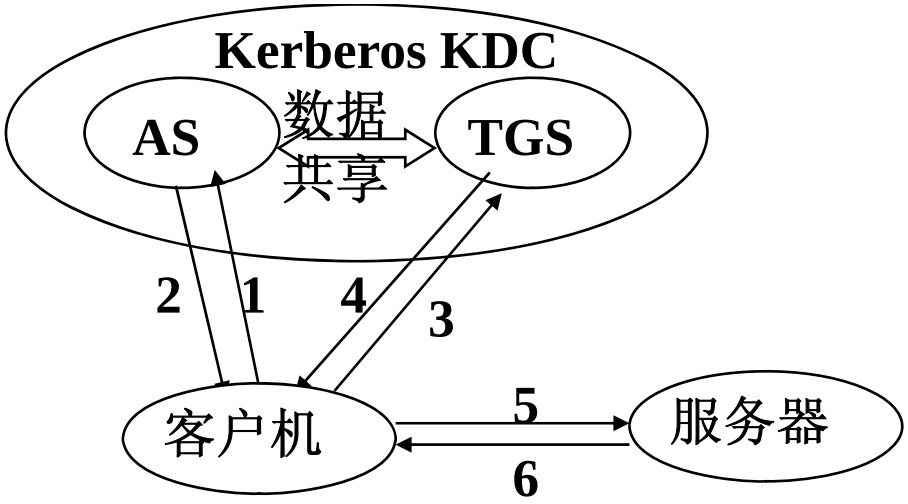
*用户/服务器双向认证交换过程，*客户方通过递交服务器凭据证明自己的身份的同时，通过一个典型的挑战/响应消息交换服务器向客户证明自己的身份。

(步骤5)C->S:$\{Ac, T_{c,s}\}$

$T_{c,s}=\{K_{c,s}, c, AD_c, s, TS4, life2\}K_s$

$Ac=\{c, AD_c, TS5\}K_{c,s}$

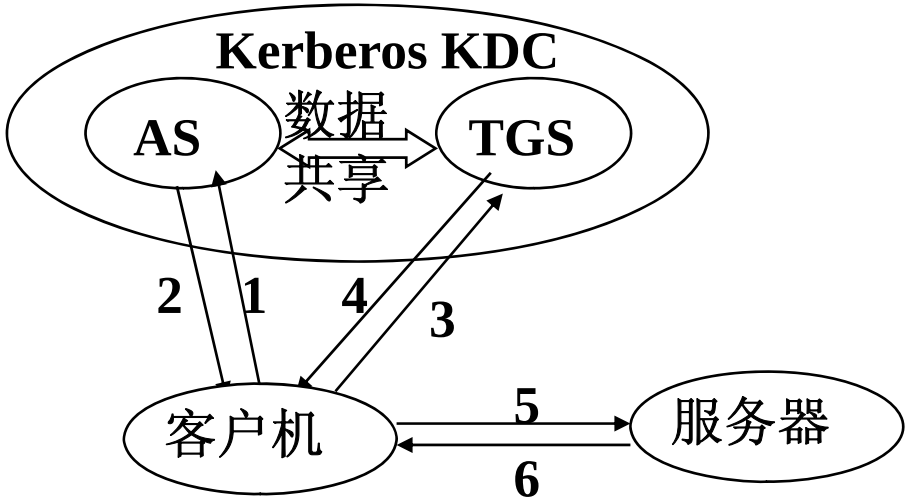
阶段3：



Kerberos协议运作步骤

客户C向应用服务器S请求服务(步骤5,6)。
若要互相验证身份，服务器可执行6
(步骤6)S->C:{TS5+1}$K_{c,s}$
服务器返回身份验证码中的时间戳加1，再用会话密钥加密，C可以解密，说明C可以保证只有s才能创建它。消息的内容向C保证它不是以前的应答。

- *Kerberos的认证方案*
  - 1和2在用户首次登录系统时使用
  - 3和4在用户每次申请某个特定的应用服务器的服务使用
  - 5用于特定的服务器中每一个服务的认证
  - 6可选，只用于相互认证

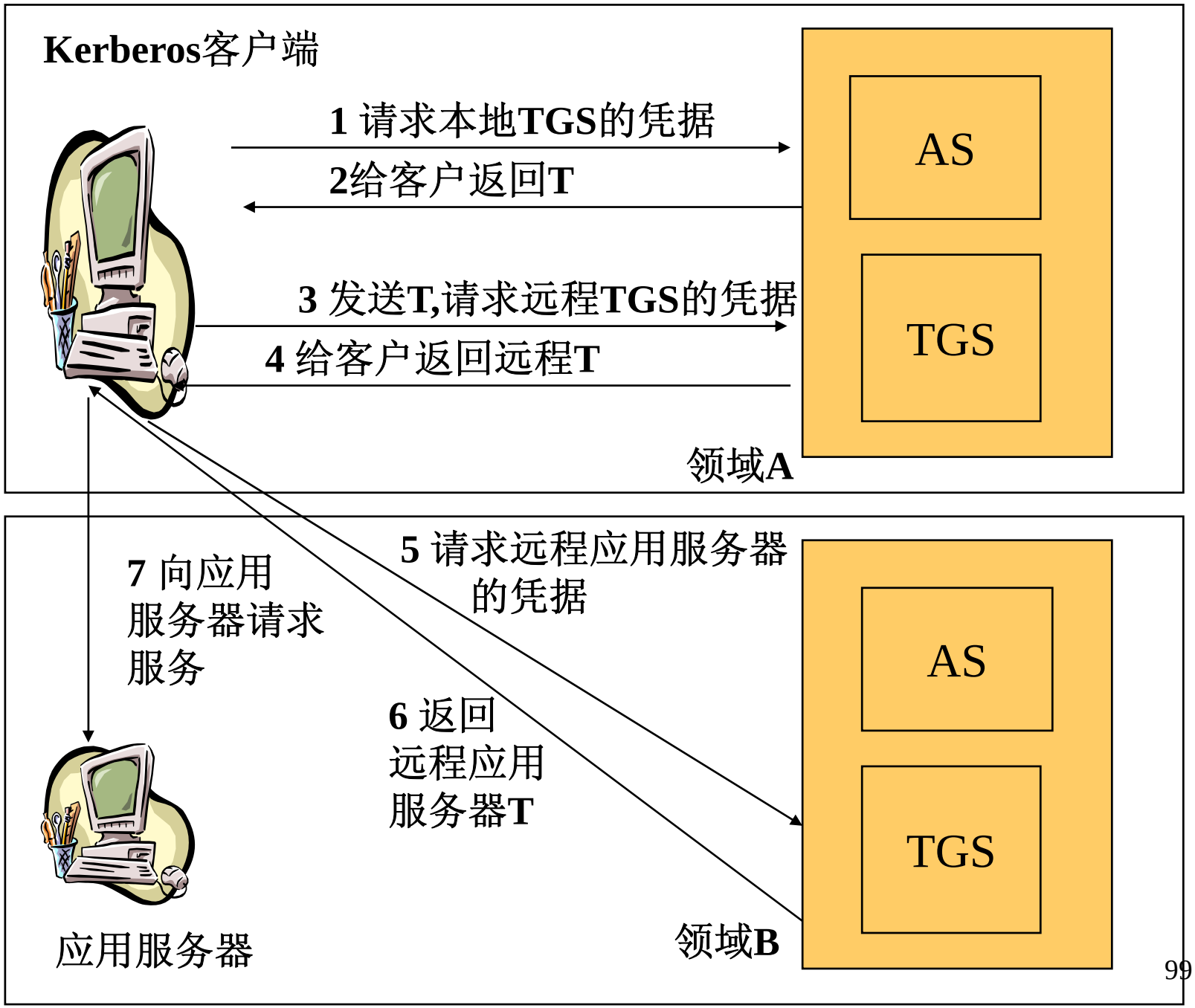


Kerberos协议运作步骤

- ***Kerberos提供三种安全等级***
  - 1)只在网络开始连接时进行认证,认为连接建立起来后的通信是可靠的.认证式网络文件系统(Authenticated network file system) 使用此种安全等级。
  - 2)安全消息传递:对每次消息都进行认证工作,但是不保证每条消息不被泄露。
  - **3)私有消息(private messages)传递:**不仅对每条消息进行认证,而且对每条消息进行加密.**Kerberos**在发送密码时就采用私有消息模式。

- ***Kerberos基础结构和交叉领域认证***
  - 当一个系统跨多个组织时，就不可能用单个认证服务器实现所有的用户注册，需要多个认证服务器各自负责系统中部分用户和服务器的认证。
  - 领域：某个特定认证服务器所注册的用户和服务器的全体称为一个领域。

- 要支持交叉领域， **Kerberos**必须满足三个条件
  - **Kerberos**服务器在数据库中必须拥有所有用户**ID**和所有参与用户口令哈希后的密钥。所有用户都已经注册到**Kerberos**服务器
  - **Kerberos**服务器必须与每个应用服务器共享保密密钥。所有服务器已经注册到**Kerberos**服务器。
  - 不同领域的**Kerberos**服务器之间共享一个保密密钥。相互之间要注册。


Kerberos客户端
1 请求本地TGS的凭据
2给客户返回T
AS
TGS
3 发送T,请求远程TGS的凭据
4 给客户返回远程T
领域A
5 请求远程应用服务器
的凭据
7 向应用
服务器请求
服务
6 返回
远程应用
服务器T
AS
TGS
应用服务器
领域B

- ***Kerberos的特点***
  - ①与授权机制相结合。
  - ②实现了一次性签放的机制，并且签放的票据都有一个有效期。
  - ③支持双向的身份认证，即服务器可以通过身份认证确认客户方的身份，而客户如果需要也可以反向认证服务方的身份。
  - ④支持分布式网络环境下的认证机制，通过交换“跨域密钥”来实现。
  - ⑤Kerberos在分布式网络环境中具有比较强的安全性，能防止攻击和窃听，能提供高可靠性和高效的服务，具有透明性（用户除了发送Password外，不会觉察出认证过程），可扩充性好。

- ***Kerberos的优点***
  - （1）较高的安全性。Kerberos系统对用户的口令进行加密后作为用户的私钥，从而避免了用户的口令在网络上显示传输，使得窃听者难以在网络上取得相应的口令信息。
  - （2）用户透明性。用户在使用过程中，仅在登录时要求输入口令与平常的操作完全一样，Kerberos 的存在对于合法用户来说是透明的。
  - （3）扩展性好。Kerberos为每个服务提供认证，Kerberos可以较方便地实现用户数的动态改变。

- ***Kerberos的缺点***
  - （1）Kerberos服务器与用户共享的秘密是用户的口令字，服务器在回应时不验证用户的真实性，因假设只有合法用户拥有口令字。如攻击者记录申请回答报文，就易形成代码攻击。
  - （2）随用户数增加，密钥管理较复杂。Kerberos拥有每个用户的口令字的散列值，AS与TGS负责户间通信密钥的分配。当N个用户想同时通信时，仍需要N（N-1）/2个密钥。
  - （3）AS和TGS是集中式管理，容易形成瓶颈，系统的性能和安全也严重依赖于AS和TGS的性能和安全。在AS和TGS前应该有访问控制，以增加AS和TGS的安全。

身份认证—X.509认证协议

- 概述
  - X.509基于公开密钥和数字签名。虽然这个协议并没有指定公钥加密算法 ，但是推荐使用RSA。
  - 在X.509 中规定了几个可选的认证过程 。所有过程都假设双方互相知道对方的公钥。

身份认证—X.509认证协议

- 主要内容
  - 简单认证程序:建议了安全度较低的身份认证
  - 强认证程序:高安全度,使用公开密钥密码学.可分为“单向的”、“双向的”、“三向的”
  - 密钥及证书管理
  - 证书扩充及证书吊销列表扩充

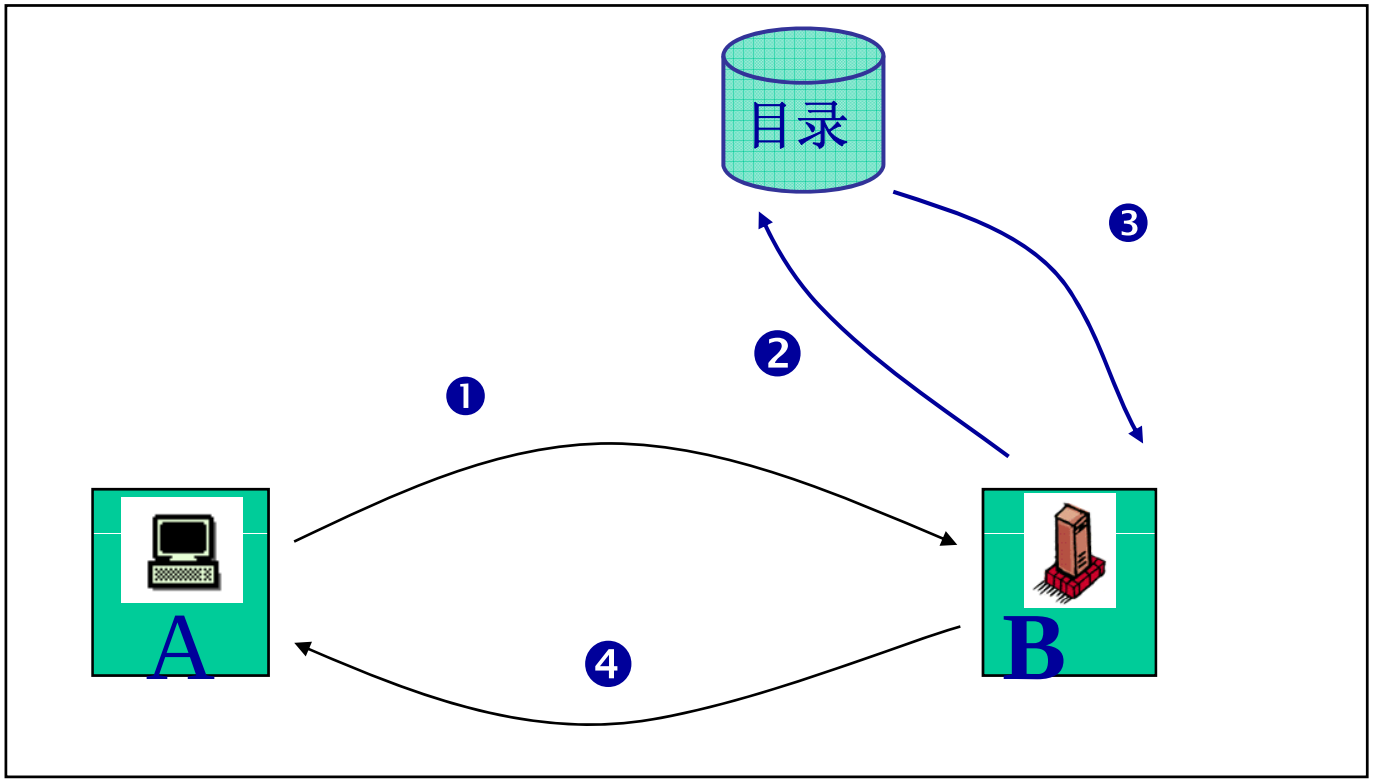
- ***X.509认证协议——简单认证过程***
  - 在**X.509**中提供的简单认证的运行方式有下列三种方式

1.用户将其口令及用户代号，未做任何加密保护，直接以明文方式传送至接收端。

2.用户将其个人之口令、用户代号、一随机数和/或时间戳，在经过一单向函数保护后，传送至接收瑞。

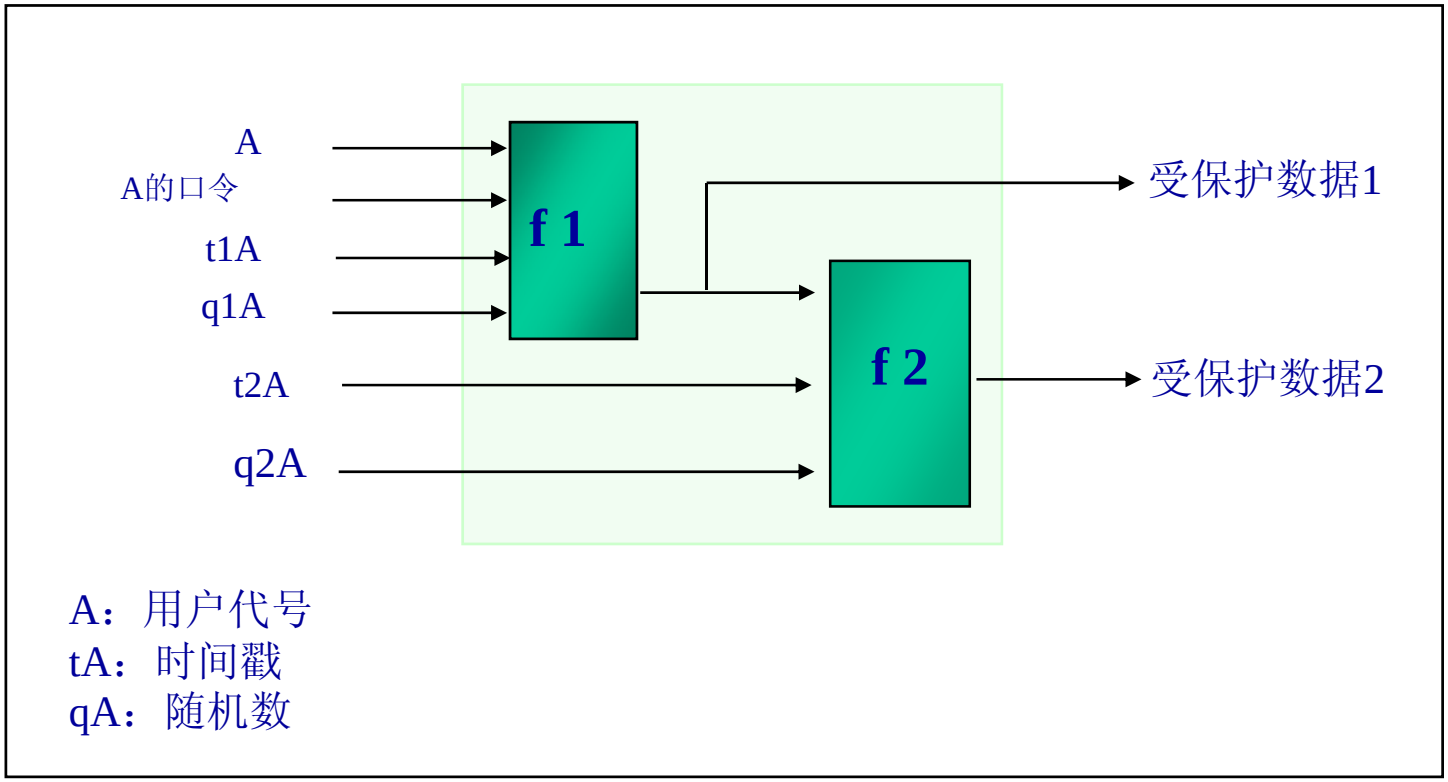
3.用户用上面第**2**种方式所述的方法，先经一次单向函数保护所有数据，然后再连同另一组随机数和/或时间戳，再经过第二次的单向函数保护后，传送至接收端。

- ***X.509认证协议——简单认证过程方式1***



– 由用户**A**将其用户代号及口令送至接收端**B**。
– 接收端**B**将**A**送来的口令及用户代号送至目录服务器，目录服务器比对先前**A**储存在此的口令。
– 目录服务器响应**B**是否接受**A**为合法的用户。
– **B**回复**A**是否为合法的用户。

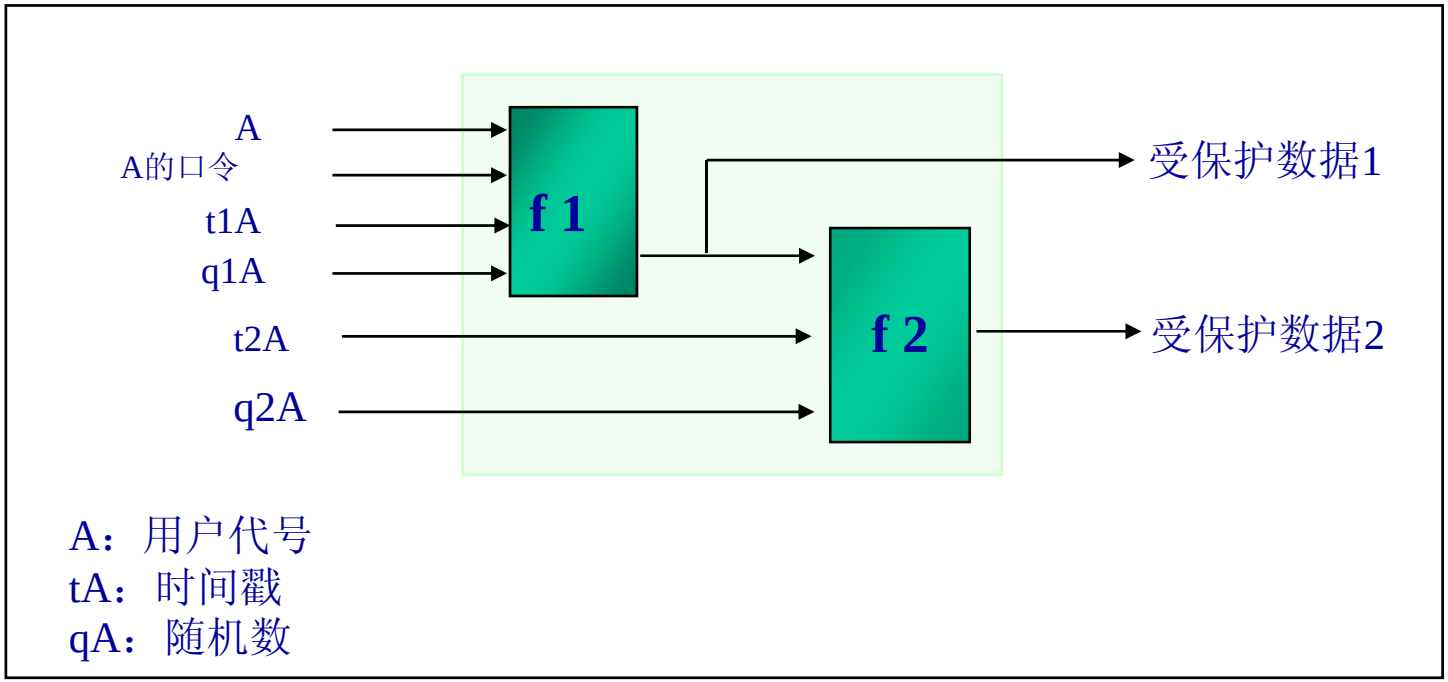
- ***X.509认证协议——简单认证过程方式2***



– 第二种方法使用图中的**f1**哈希运算
– 受保护数据**1=f1**（**t1A,q1A,A,A**的口令）
– 发送方**A**所送出的数据=（**t1A,q1A,A,**受保护数据**1**）

- ***X.509认证协议——简单认证过程方式2***
  - 口令以单向函数保护起来，且包含了时间戳、随机数，可降低重放攻击。
  - 实施过程如下
    - 当**A**要获得**B**认证时，产生“发送方**A**所送出的数据”=（**t1A,q1A,A,**受保护数据**1**）
    - **B**找到**A**的口令，将**t1A,q1A**执行**f1**（**t1A,q1A,A,A**的口令）
    - 若结果和“受保护数据**1**”相同，则承认**A**是合法用户
    - **B**回复**A**为合法用户

- ***X.509认证协议——简单认证过程方式3***



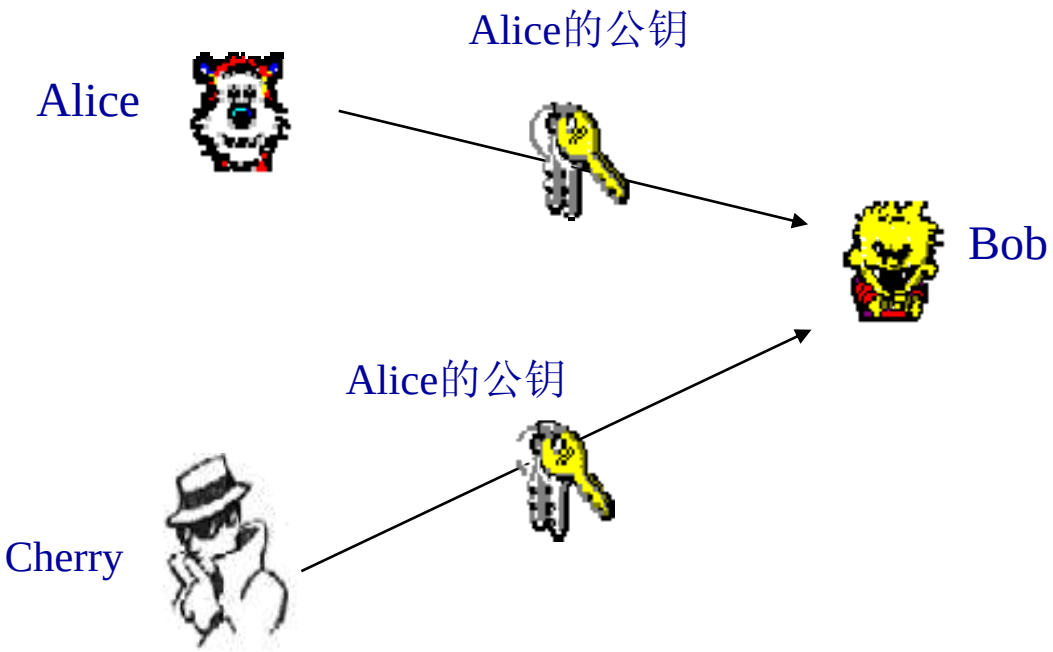
– **受保护数据1=f1（t1A,q1A,A,A的口令）**
– **受保护数据2=f2（t2A,q2A,受保护数据1）**
– **发送方A所送出的数据=（t1A,q1A, t2A,q2A,A，受保护数据2）**

- ***X.509认证协议——简单认证过程方式3***
  - 实施过程
    - 当**A**要获得**B**认证时，产生“发送方**A**所送出的数据”＝（**t1A,q1A, t2A,q2A,**受保护数据**2**）
    - **B**找到**A**的口令，将**t1A,q1A , t2A,q2A**执行

      f2（**t2A,q2A,**受保护数据**1**）
    - 若结果和“受保护数据**2**”相同，则承认**A**是合法用户
    - **B**回复**A**为合法用户

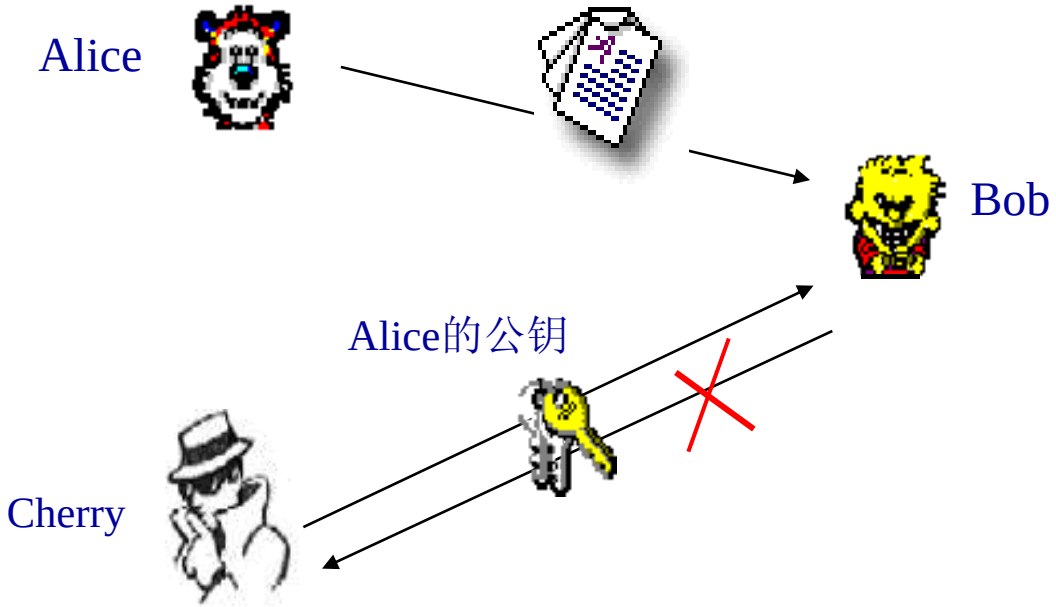
- ***X.509认证协议——密钥及证书管理***
  - “简单认证”，若是使用封闭的系统，极容易实现，成本较低，但是使用在因特网上，则“简单认证”安全性就不足以保障认证的安全性了。
  - 在**X.509**中以公开密钥密码的技术能够让通信双方容易共享密钥的特点，并利用公钥密码系统中数字签名的功能，强化网络上远程认证的能力，定义出强认证程序，以达到所谓“强认证”的目的 。

- ***X.509认证协议——密钥及证书管理***
  - 公钥系统中的问题—网络上公钥的确认



  - 防止上述攻击的机制是大家信任的**认证中心(CA)**以数字签名技术，将每个用户的公钥与个人的身份数据签成电子证书。

- ***X.509认证协议——密钥及证书管理***



– 用户收到证书后通过验证程序，确信证书无误，所含公钥、身份数据及其他内容确是证书上声称的主体的。

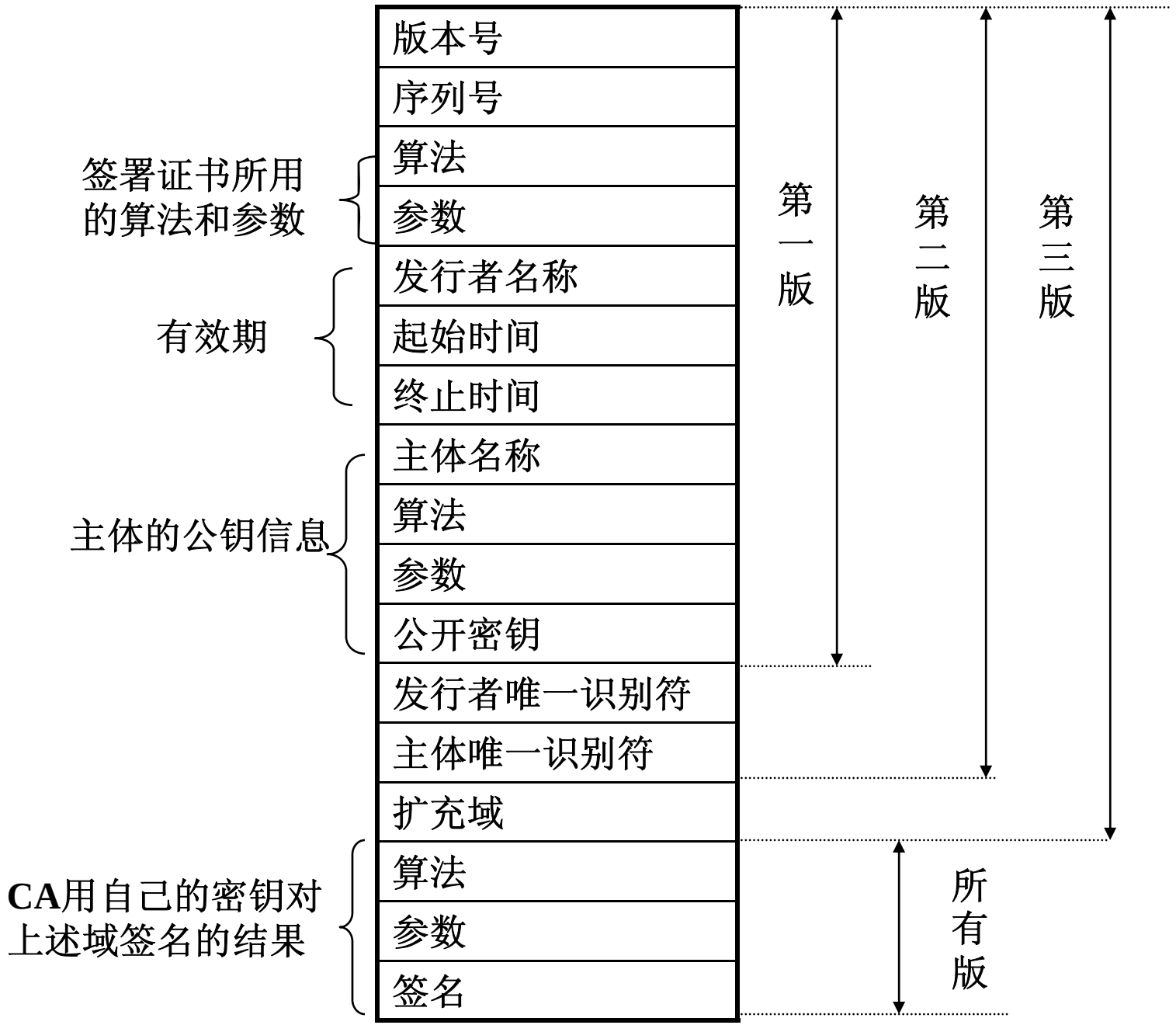
- ***X.509认证协议——密钥及证书管理***
  - 在**X.509**中提到证书必须符合有下列两点特性：

1.所有可取得认证中心公钥的用户，可以找到任何已经经过该认证中心认证的公钥。

2.除了认证中心本身以外，其它任何人修改证书的动作都会被察觉、检测出来。

认证中心以自己的私钥为用户签发证书，用户可以使用认证中心的公钥验证所获证书的正确性。

- ***X.509认证协议——密钥及证书管理***
  - 用户的数字证书是**X.509**的核心
  - 证书由某个可信的证书发放机构**CA**建立，并由用户自己或**CA**放入公共目录，以供其他用户访问。
  - **X.509**中数字证书的格式如后图所示


版本号
序列号
算法
参数
发行者名称
起始时间
终止时间
主体名称
算法
参数
公开密钥
发行者唯一识别符
主体唯一识别符
扩充域
算法
参数
签名
签署证书所用的算法和参数
有效期
主体的公钥信息
CA用自己的密钥对上述域签名的结果
第一版
第二版
第三版
所有版

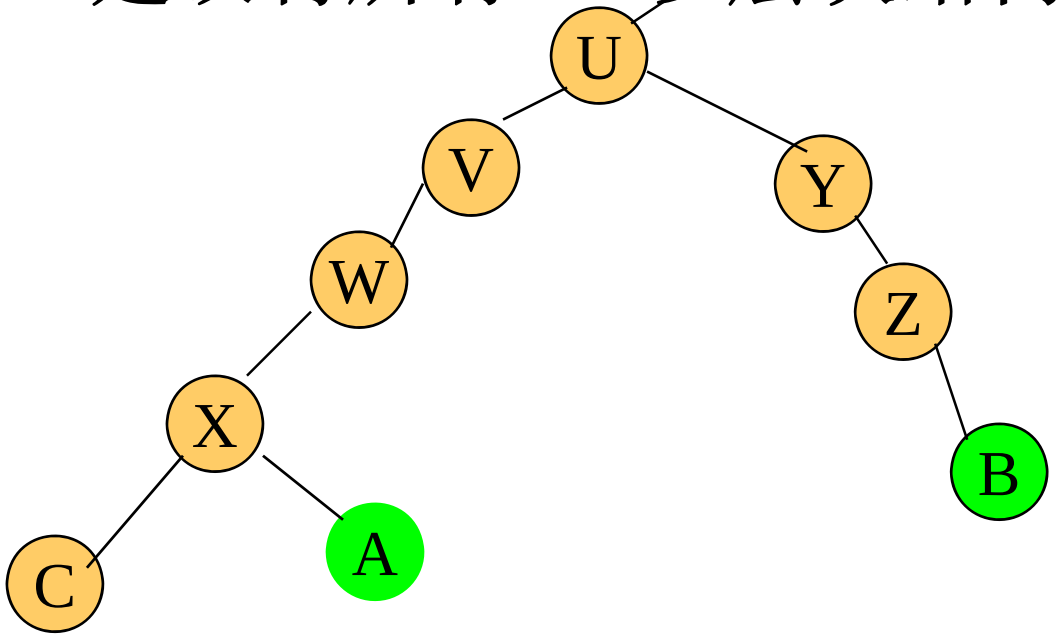
- ***X.509认证协议——密钥及证书管理***
  - **X.509使用以下表示法**
  - **CA《A》=CA{V,SN,AI,CA,T_A,A,A_P}**
  - **Y《X》表示证书发放机构Y向用户X发放的证书**
  - **Y{I}表示Y对I的哈希值签名**

- ***X.509认证协议——密钥及证书管理***
  - 证书的获取
  - 用户证书除了放在公共目录供他人访问，还可以由用户直接把证书发给其它用户
  - 用户**B**得到**A**的证书后，可相信用**A**的公开密钥加密的消息不会被他人获悉，还相信用**A**的秘密密钥签署的消息是不可伪造的。

- *证书的获取*
  - 通常用户数量极多，应由多个CA，每个CA为一部分用户发行，签署证书。
  - 多CA的证书获取
  - 前提：两个CA：X1和X2相互安全的交换公钥,互相签署证书。
  - 则A可通过下述过程获得B的公钥
    - （1）A从目录中获取由X1签署的X2的证书X1《X2》，因A知道X1的公钥，可验证X2的证书，并从中获得X2的公钥
    - （2）A从目录中获得X2签署的B的证书X2《B》，因A知道X2的公钥，可验证B的证书，并从中获得B的公钥

- *证书的获取*
  - 上述过程表示为一个证书链
  - **X1《X2》 X2《B》**
  - 反过来**B**也可以获得**A**的公钥
  - **X2《X1》 X1《A》**
  - **N**个证书的证书链
  - **X1《X2》 X2《X3》 …… XN《B》**
  - 任意两个相邻的**CA Xi-1**和**Xi**彼此间为对方建立了证书，对每一个**CA**来说，由其他**CA**为这一**CA**建立的所有证书都要存放于目录中，并使用户知道所有证书之间的连接关系。

- *证书的获取*
  - **X.509建议将所有CA以层次结构组织起来如图**



  - **X《W》 W《V》 V《U》 U《Y》 Y《Z》 Z《B》**

- *证书的吊销*
  - 有些证书未到截至日期被吊销
  - 每个**CA**要维护一个证书吊销列表
  - 每一用户收到他人消息中的证书时，都必须通过目录检查是否被吊销

- ***X.509认证协议——强认证***
  - 强认证是借助用户所拥有的公钥来证明他的身份，先决条件是前述的认证中心体系已建立。
  - 基于公钥密码系统的三种不同信赖程度的认证
    - 单向认证
    - 双向认证
    - 三向认证

•X.509认证协议——强认证

–单向认证

– 单向认证只需要一次通信。用户A向用户B发送一条消息，证明自己的身份。

–A→B: $A\{t_A, r_A, B, sgnData, E_{kub}[K_{ab}]\}$

- 时间戳t_A包含起始时间和终止时间，用来防止报文的延迟传送；
- r_A是一个现时，在一个有效期内是唯一的，B需要保存r_A直到有效期结束；
- B是B的身份信息；
- sgnData是签名数据；
- 最后是用B的公钥加密的会话密钥。
- 整个报文还要由A签名，以保证完整性和抗抵赖性。

- *单向认证*
  - B收到$A \rightarrow B: A\{t_A, r_A, B, sgnData, E_{kub}[K_{ab}]\}$后
    - 从认证中心获得A的公钥，检查证书是否过期或注销
    - 验证签名，确定完整性
    - 检查此文件的识别数据，B是否为收方
    - 检查t_A是否在有限期内
    - 检查r_A是否重复出现
  - 达到的认证功能
    - 确认发送方、确认接受方、数据完整性

- ***X.509认证协议——强认证***
  - 双向认证
  - 增加了一次通信过程如下：
  - **1）$A \rightarrow B$: $A\{t_A, r_A, B, sgnData, E_{kub}[K_{ab}]\}$**
  - **2）$B \rightarrow A$: $B\{t_B, r_B, A, r_A, sgnData, E_{kua}[K_{ba}]\}$**
  - 这种方式允许双方相互认证，但是两条消息中有冗余信息。

- ***X.509认证协议——强认证***
  - 三向认证
  - 在最后增加了一个**A**向**B**的回答。
  - **1）A→B: A{t_A,r_A,B,sgnData,E_{kub}[K_{ab}]}**
  - **2）B→A: B{t_B,r_B,A, r_A,sgnData,E_{kua}[K_{ba}]}**
  - **3）A→B: A{ r_B }**
  - 最后一条报文包含现时r_B的签名备份，这样就无需检查时间戳，因为每一端都可以通过检查返回的现时来探测重放攻击。

- ***X.509认证协议—小结***
  - 简单认证程序:建议了安全度较低的身份认证
  - 密钥及证书管理
    - 原因
    - 数字证书
    - 证书的格式、获取、吊销
    - 证书链
  - 强认证程序:高安全度,使用公开密钥密码学.可分为“单向的”、“双向的”、“三向的”

身份认证—CA系统结构

- 公钥基础设施PKI (Public Key Infrastructure)
  - 是一种遵循既定标准的密钥管理平台,它能够为所有网络应用提供加密和数字签名等密码服务及所必需的密钥和证书管理体系。
  - 完整的PKI系统必须具有
    - 权威认证机构(CA)、
    - 数字证书库、
    - 密钥备份及恢复系统、
    - 证书作废系统、
    - 应用接口（API）

身份认证—CA系统结构

- CA的主要功能
  - 证书的颁发
    - 接收、验证用户的数字证书的申请
    - 将申请的内容进行备案
    - 并根据申请的内容确定是否受理该数字证书申请。
    - 如果中心接受该数字证书申请，则进一步确定给用户颁发何种类型的证书。
    - 新证书用认证中心的私钥签名以后，发送到目录服务器供用户下载和查询。
    - 为了保证消息的完整性，返回给用户的所有应答信息都要使用认证中心的签名。

– 证书的更新

- 认证中心可以定期更新所有用户的证书，或者根据用户的请求来更新用户的证书。

– 证书的查询

- 其一是证书申请的查询，认证中心根据用户的查询请求返回当前用户证书申请的处理过程；
- 其二是用户证书的查询，这类查询由目录服务器来完成，目录服务器根据用户的请求返回适当的证书。

– 证书的作废

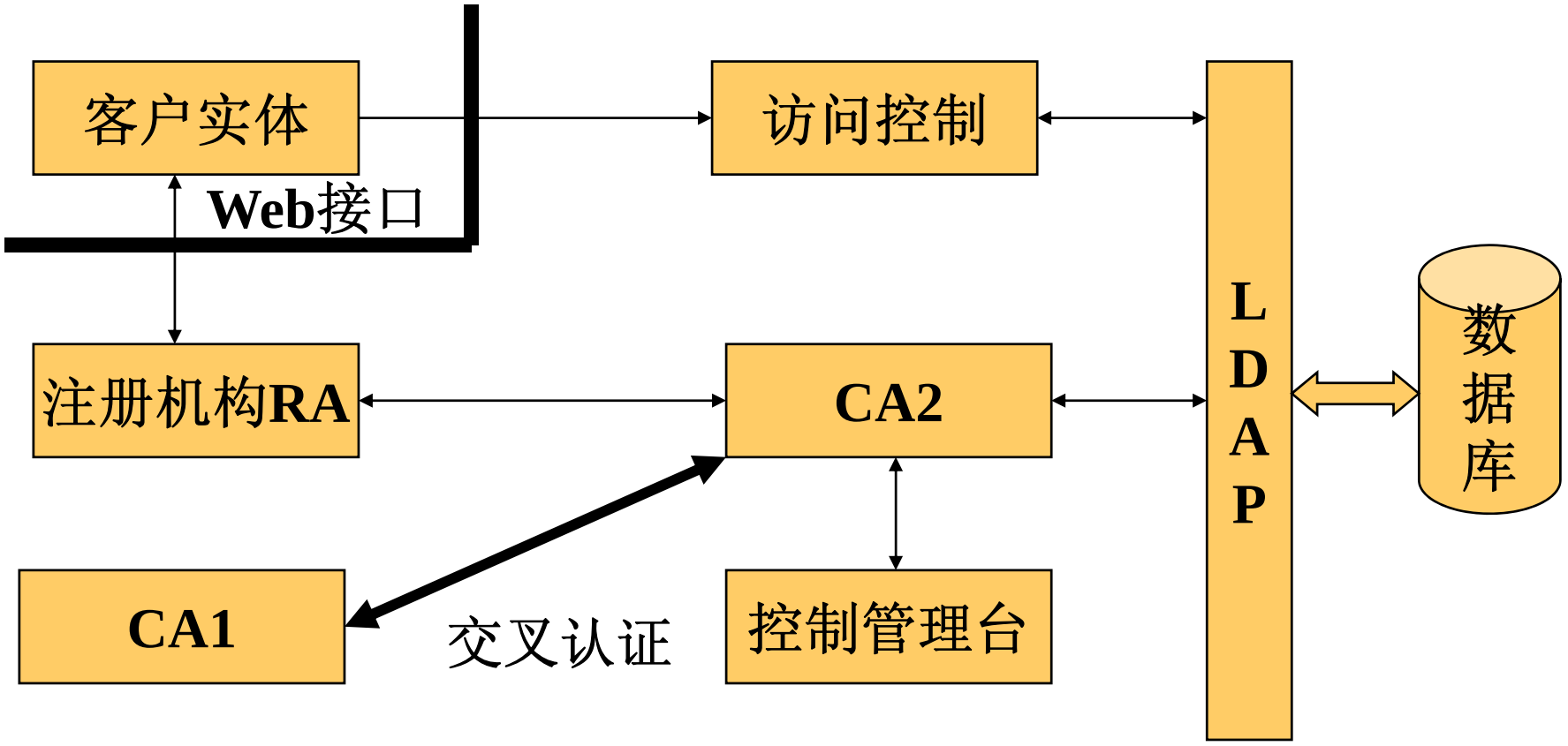
- 当用户的私钥由于泄密等原因造成用户证书需要申请作废时，用户需要向认证中心提出证书作废的请求，认证中心根据用户的请求确定是否将该证书作废。
- 另外一种证书作废的情况是证书已经过了有效期，认证中心自动将该证书作废。认证中心通过维护证书作废列表来完成上述功能。

– 证书的归档

- 证书具有一定的有效期，证书过了有效期之后就将作废，但是我们不能将作废的证书简单地丢弃，因为有时我们可能需要验证以前的某个交易过程中产生的数字签名，这时我们就需要查询作废的证书。基于此类考虑，认证中心还应当具备管理作废证书和作废私钥的功能。

身份认证—CA系统结构

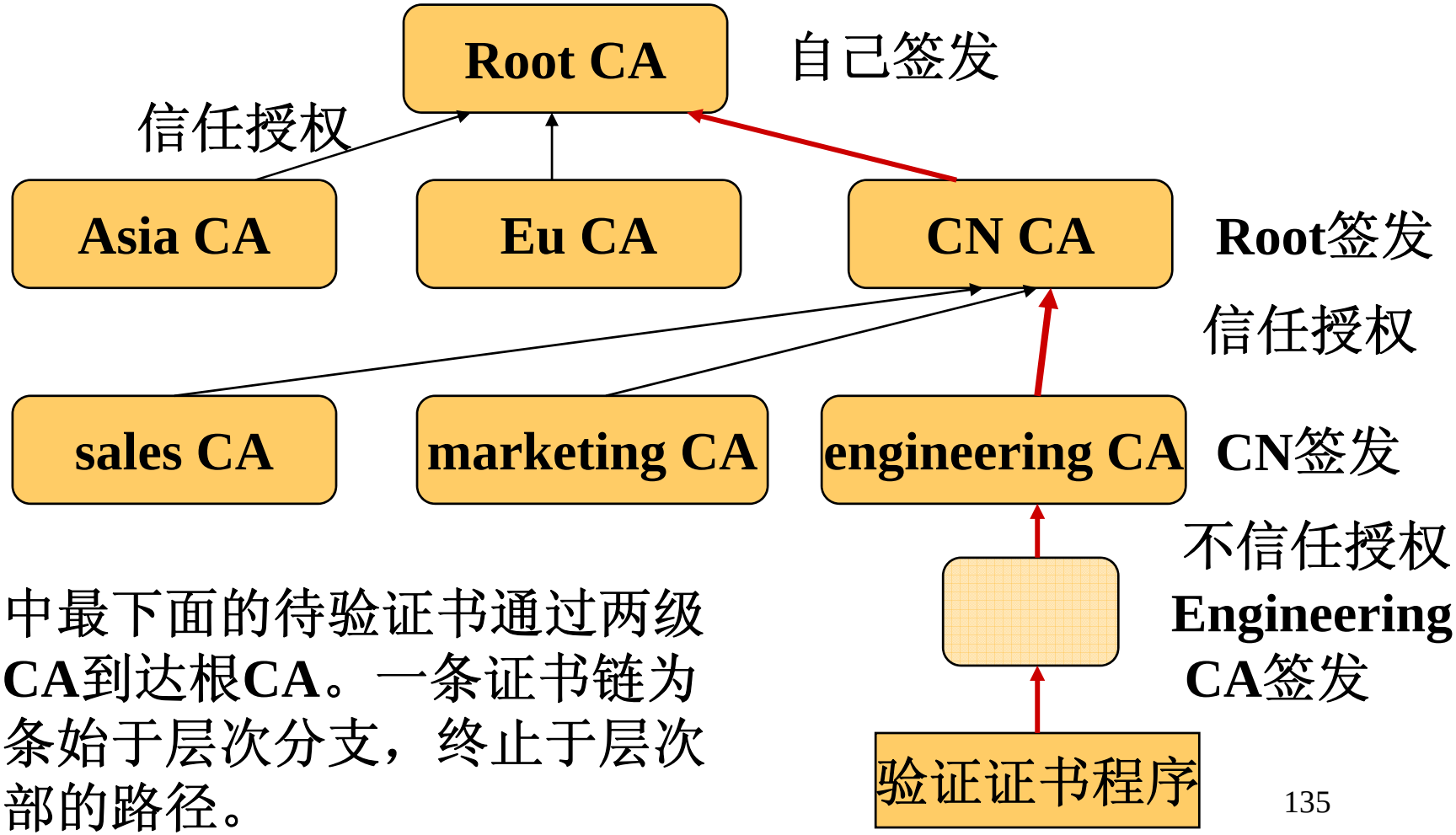
- 认证机构(CA)是核心，信任的发源地



以OpenCA为例说明CA的系统结构

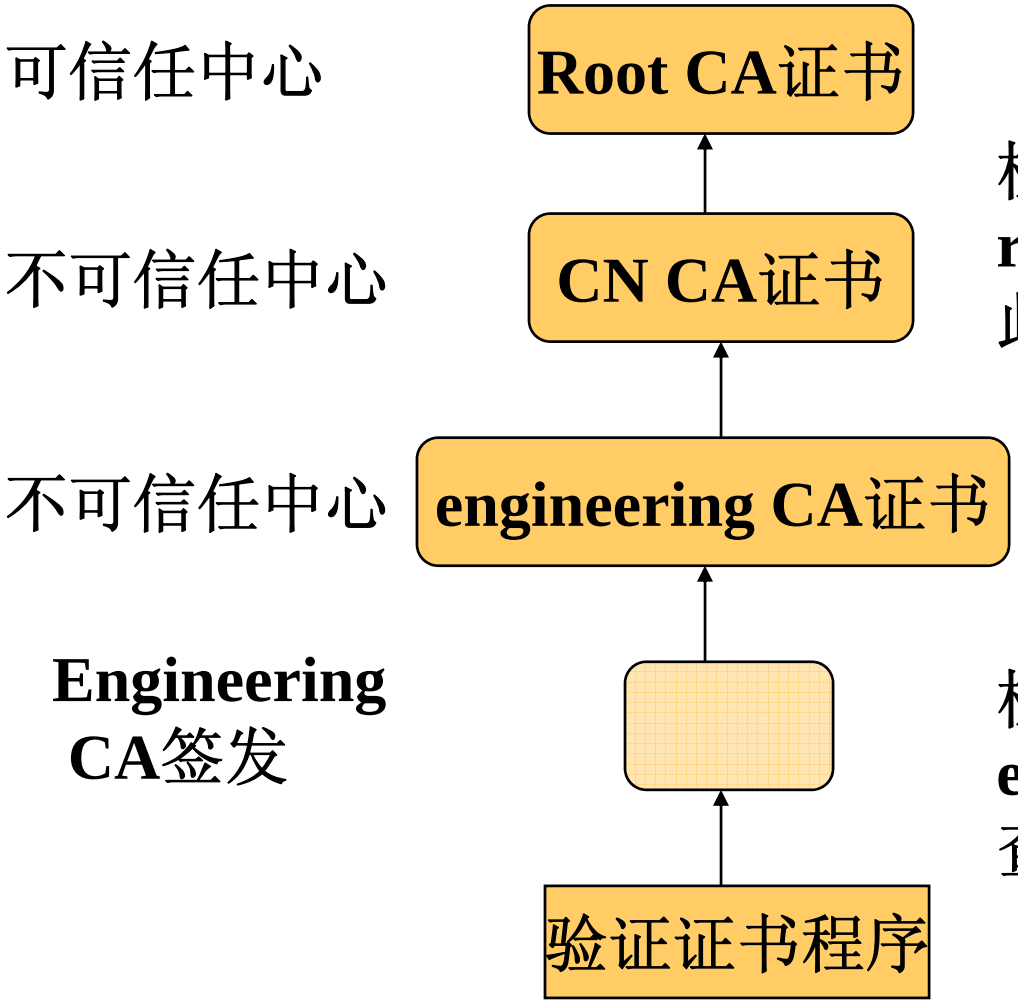
- ***CA系统结构－CA服务器***
  - 整个认证中心的核心，保存了根**CA**的私钥
  - 产生证书、密钥备份、证书管理、交叉认证
- ***CA系统结构－RA服务器***
  - 注册机构，实现用户的注册
  - 分成**RA**操作员、**RA**服务器
- ***CA系统结构－ 证书目录服务器***
  - 认证中心颁发的证书只是捆绑了特定的实体和身份和公钥，但是没有提供如何找到该实体证书的方法。
  - **LDAP**轻量级目录访问协议。

- *CA系统结构－CA操作步骤*
- *CA系统结构－ 证书链构造*
  - CA的层次结构可被映射为证书链



图中最下面的待验证书通过两级子CA到达根CA。一条证书链为一条始于层次分支，终止于层次顶部的路径。

- ***CA系统结构－证书验证过程***



检查有效期，验证该证书由**root**签发。因其可信，验证到此为止

检查有效期，验证该证书由**cn ca**签发。因其不可信，查下一个证书

检查有效期，验证该证书由**engineering**签发。因其不可信，查下一个证书

身份认证—授权管理基础设施PMI

- **PMI(Privilege Management Infrastructure)**
  - 目标是向用户和应用程序提供授权管理服务，提供用户身份到应用授权的映射功能。
  - 授权管理基础设施**PMI**以资源管理为核心，对资源的访问控制权统一交由授权机构统一处理，即由资源的所有者来进行访问控制。
  - 同公钥基础设施**PKI**相比，两者主要区别在于：**PKI**证明用户是谁，而**PMI**证明这个用户有什么权限，能干什么，而且授权管理基础设施**PMI**需要公钥基础设施**PKI**为其提供身份认证。

身份认证—授权管理基础设施PMI

- 授权管理基础设施**PMI** 在体系上可分为三级，分别是**SOA** 中心、**AA** 中心和**AA** 代理点。
  - *SOA 中心*
    - 信任源点（SOA 中心）是整个授权管理体系的中心业务节点，也是整个授权管理基础设施PMI 的最终信任源和最高管理机构。SOA 中心的职责主要包括授权管理策略的管理、应用授权受理、AA 中心的设立审核及管理、授权管理体系业务的规范化等。

身份认证—授权管理基础设施PMI

- AA 中心
- AA中心是授权管理基础设施PMI 的核心服务节点，是对应于具体应用系统的授权管理分系统，由具有设立AA 中心业务需求的各应用单位负责建设，并与SOA 中心通过业务协议达成相互信任关系。AA 中心的职责主要包括应用授权受理、属性证书的发放和管理以及AA代理点的设立审核和管理等。AA 中心需要为其所发放的所有属性证书维持一个历史记录和更新记录

身份认证—授权管理基础设施PMI

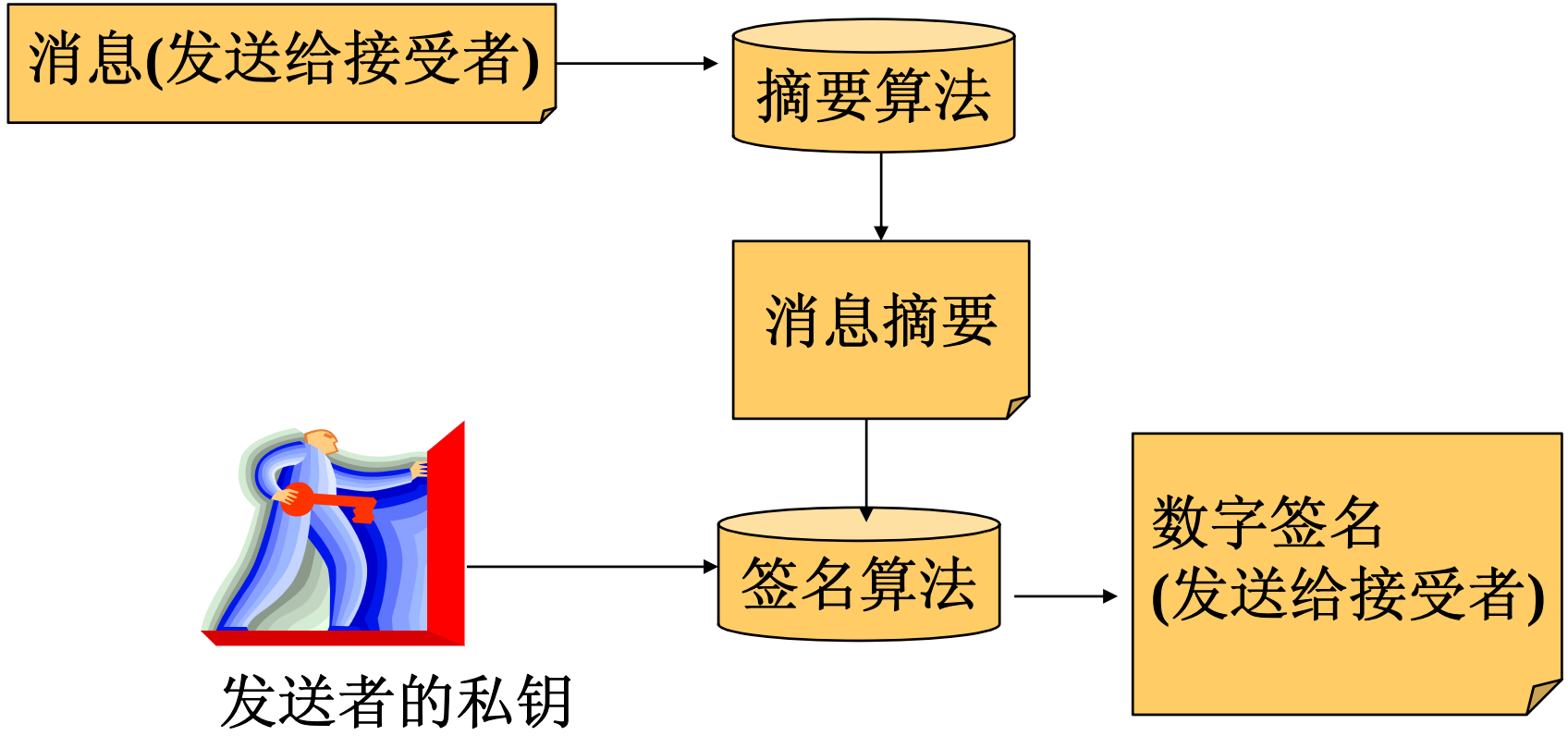
- AA 代理点
- AA 代理点是授权管理基础设施PMI 的用户代理节点，也称为资源管理中心，是与具体应用用户的接口，是对应AA 中心的附属机构，接受AA 中心的直接管理，由各AA 中心负责建设，并报经主管的SOA 中心同意并签发相应的证书。AA 代理点的设立和数目由各AA 中心根据自身的业务发展需求而定。AA 代理点的职责主要包括应用授权服务代理和应用授权审核代理等，负责对具体的用户应用资源进行授权审核，并将属性证书的操作请求提交到授权服务中心进行处理。

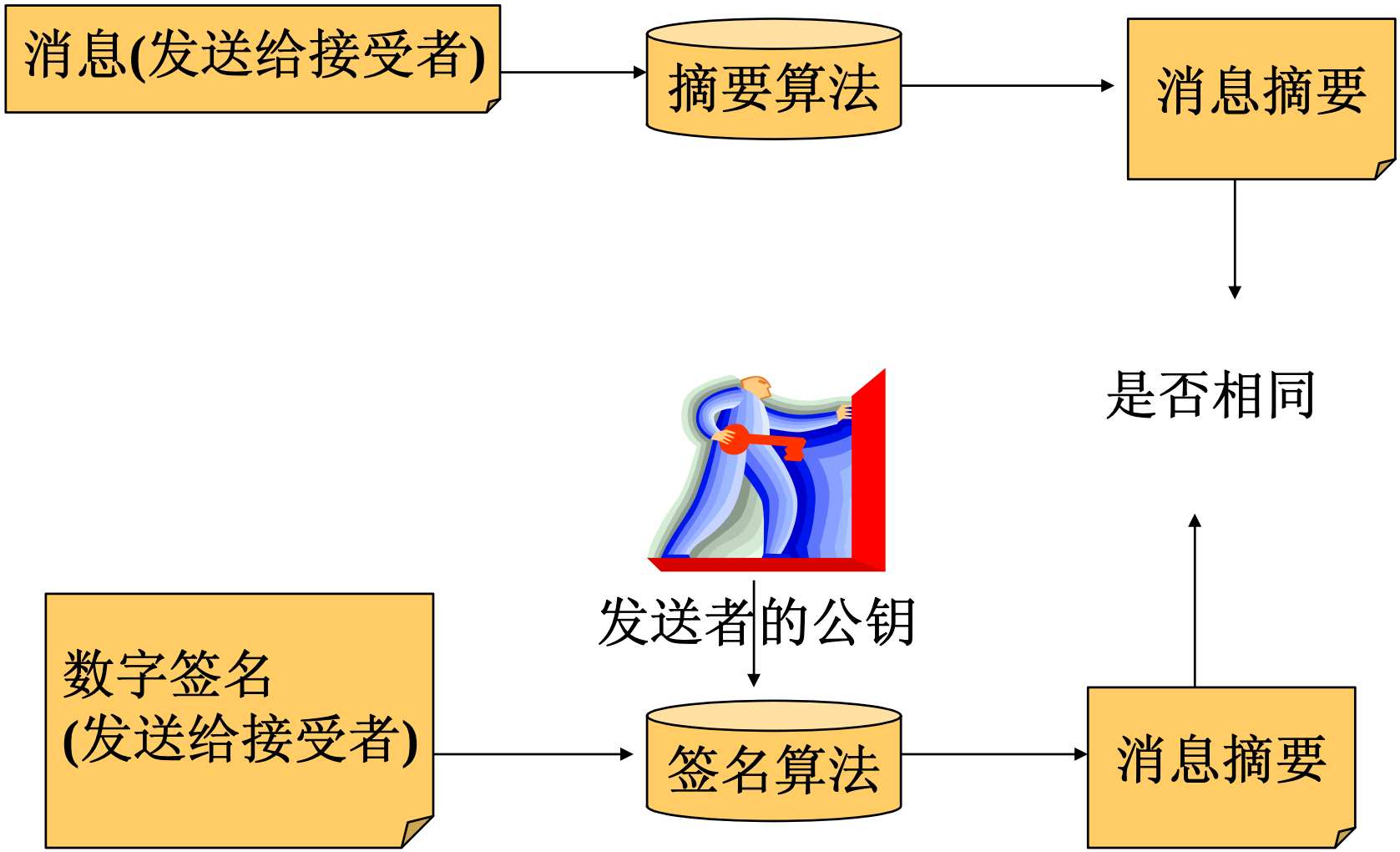
总结

- 报文鉴别—完整性
  - 报文源鉴别
  - 报文宿鉴别
  - 时间性鉴别
  - 报文内容鉴别：报文鉴别码、报文加密、报文摘要
- 散列函数
  - 特点：单向性(两种)
  - 攻击的方式的有效性来自于应用

总结

- 数字签名
  - 实质和两个阶段



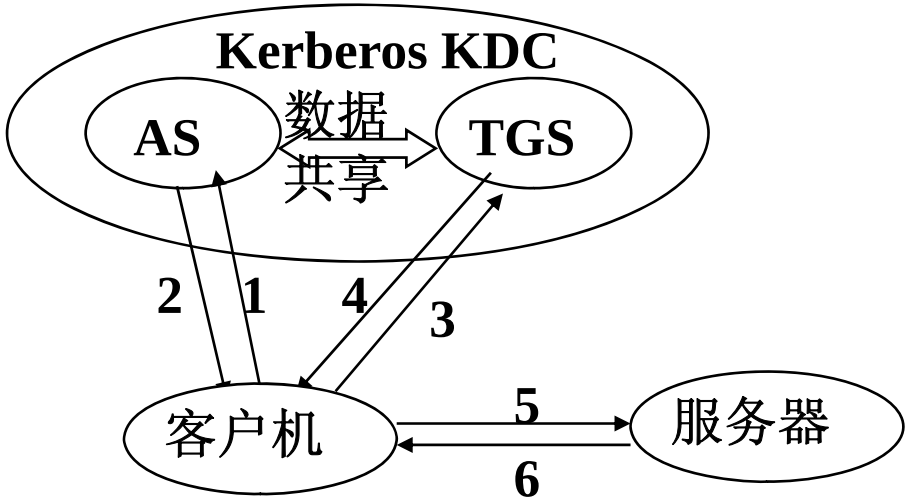

消息(发送给接受者)
摘要算法
消息摘要
是否相同
发送者的公钥
数字签名
(发送给接受者)
签名算法
消息摘要

总结

- 数字签名
  - 具体算法
  - 数字签名的形式
    - 多重签名
    - 不可抵赖的签名
    - 制定确定人的签名
    - 团体签名
    - 盲签名

总结

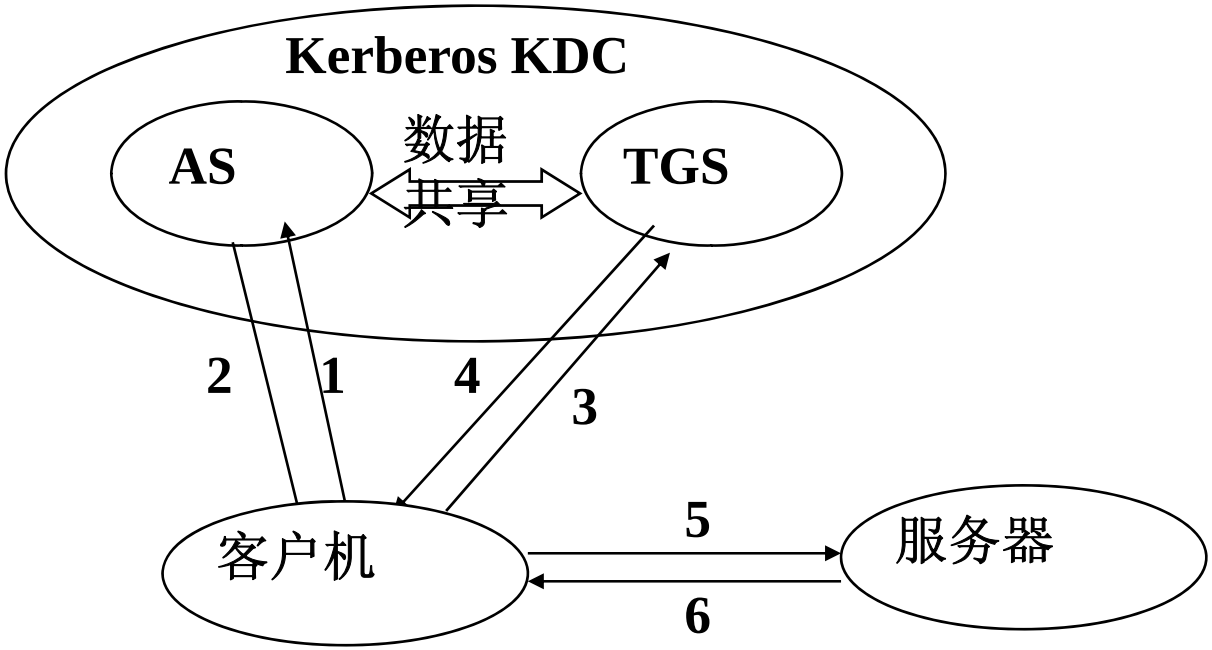
- 身份认证
  - 两个环节：I，A
  - 3种身份认证的方法：
  - 协议
    - **Kerberos**
    - **X.509**
    - **CA**
    - **PMI**



Kerberos协议运作步骤

总结

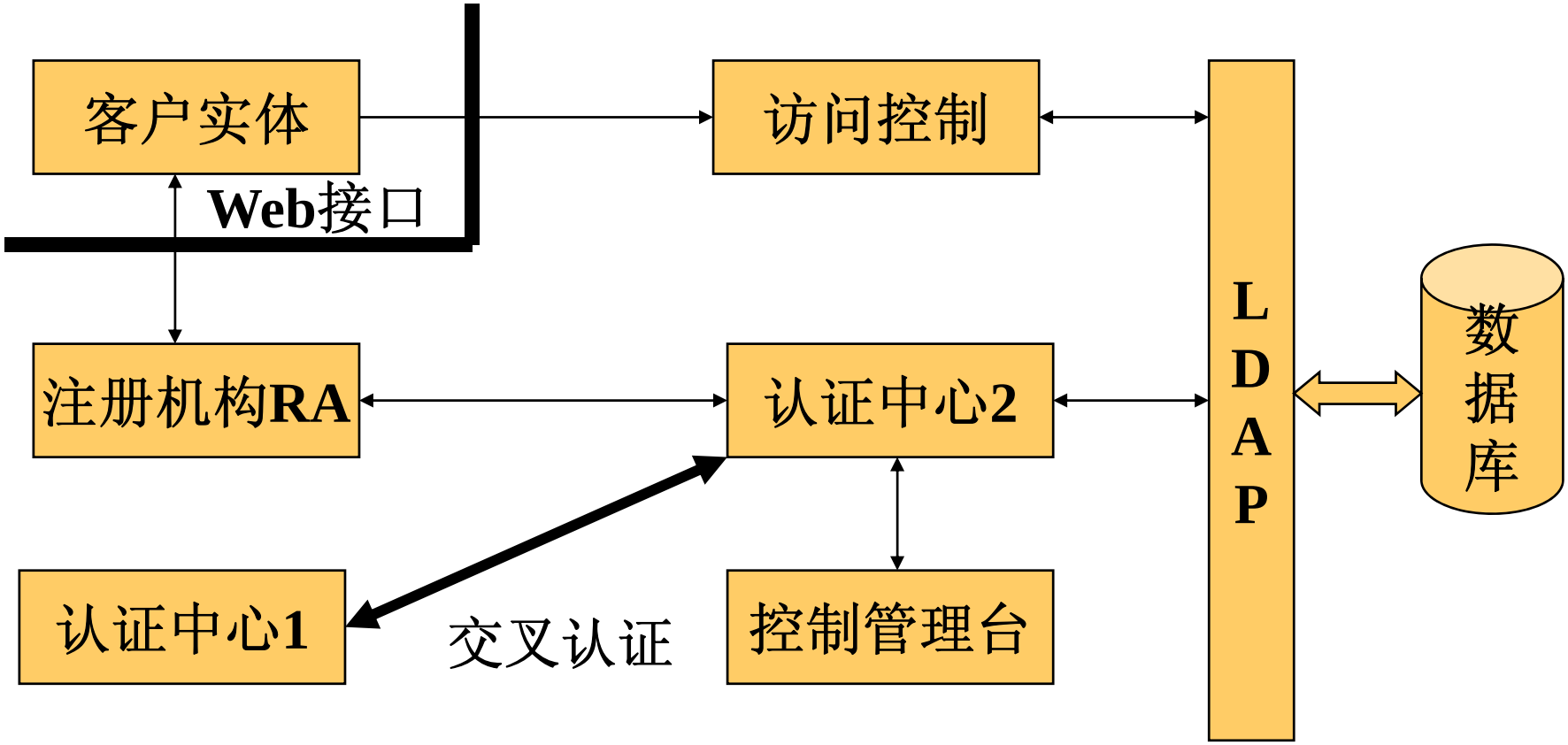
- 身份认证
  - **Kerberos**



Kerberos协议运作步骤

总结

- 身份认证
  - **X.509**
  - 简单认证程序:建议了安全度较低的身份认证
  - 密钥及证书管理
    - 原因
    - 数字证书
    - 证书的格式、获取、吊销
    - 证书链
  - 强认证程序:高安全度,使用公开密钥密码学.可分为“单向的”、“双向的”、“三向的”



以OpenCA为例说明CA的系统结构

CA服务器、RA服务器、证书目录服务器、CA操作步骤、证书链、验证过程