密码学

第七章 数字签名

网络空间安全学院
朱丹 戚明平
zhudan/mpqi@nwpu.edu.cn

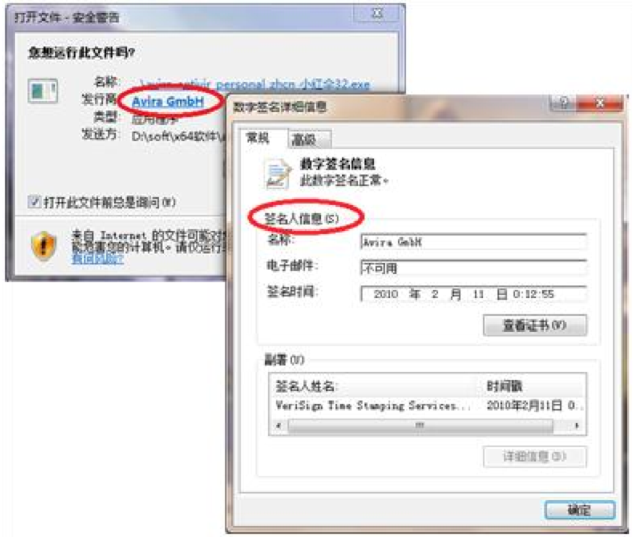
知识回顾-数字签名的基本概念

- 在人们的工作和生活中，许多事物的处理需要当事人签名。
- 签名起到确认、核准、生效和负责任等多种作用
- 签名是证明当事者的身份和数据真实性的一种信息
- 签名可以用不同的形式来表示
  - 书面签名：手签、印章、手印等，已得到司法部门的支持
  - 数字签名：电子数字形式的签名，已得到中国和其它一些国家的法律支持

知识回顾-数字签名的基本概念

Basic Concepts of Digital Signatures

- 一种完善的签名应满足以下四个条件：
  - 签名与文件具有绑定性
  - 签名者事后不能否认自己的签名
  - 任何其他人不能伪造签名
  - 如果当事双方关于签名真伪发生争执，能够在仲裁者面前通过验证确认其真伪



知识回顾-数字签名的模型

- 一个数字签名体制包括两个方面的处理：
  - 施加签名
  - 验证签名
- 设施加签名的算法为$\boldsymbol{SIG}$，产生签名的密钥为K_d，被签名的数据为M，产生的签名信息为S，则有

$$S = \boldsymbol{SIG}(M, K_d)$$

- 设验证签名的算法为$\boldsymbol{VER}$，验证签名的密钥为K_e，用$\boldsymbol{VER}$对签名S进行验证，可鉴别S的真假。即

$$\boldsymbol{VER}(S, K_e) = \begin{cases} \text{真, 当验证结果符合判定准则} \\ \text{假, 当验证结果不符合判定准则} \end{cases}$$

知识回顾-数字签名的模型

- 签名函数必须满足以下条件，否则文件内容及签名被篡改或冒充时均无法发现：

  ① 当$M \neq M'$时，有$\boldsymbol{SIG}(M, K_d) \neq \boldsymbol{SIG}(M', K_d)$

  ✓ 条件①要求签名S至少和被签名的数据M一样长（——映射关系）。当M太长时，应用很不方便

  ✓ 将条件①改为：虽然当$M \neq M'$时，存在$S = S'$，但对于给定的M或S，要找出具有相同签名的M'在计算上是不可能的

知识回顾-数字签名的模型

- 签名函数必须满足以下条件，否则文件内容及签名被篡改或冒充时均无法发现：

  ② 签名S只能由签名者产生，否则别人便可伪造，于是签名者也就可以抵赖

  ③ 收信者可以验证签名S的真伪。这使得当签名为假是收信者不必上当

  ④ 签名者也应有办法鉴别收信者所出示的签名是否是自己的签名。这就给签名者以自卫的能力

知识回顾-利用公钥密码实现数字签名

- **利用公钥密码实现数字签名的一般方法：**
- 数字签名—消息验证过程：

签名通信协议：$A \xrightarrow{M} B$

① A用自己的私钥K_{d_A}对数据M进行签名：

$$S_A = \boldsymbol{SIG}(M, K_{d_A})$$

② 如果不需要保密，则A直接将S_A发送给用户B

③ 如果需要保密，则A用B的公钥K_{e_B}对S_A加密，得到密文C:

$$C = E(S_A, K_{e_B})$$

④ 最后，A将C发送给B，并将S_A或C留底

知识回顾-利用公钥密码实现数字签名

⑤ B收到后，若是不保密通信，则用A的公钥K_{e_A}对签名进行验证：

$$\boldsymbol{VER}(S_A, K_{e_A}) = \boldsymbol{VER}(\boldsymbol{SIG}(M, K_{d_A}), K_{e_A}) \in \{0,1\}$$

⑥ 若是保密通信，则B先用自己的私钥K_{d_B}对C解密，然后再用A的公钥K_{e_A}对签名进行验证：

$$D(C, K_{d_B}) = D(E(S_A, K_{e_B}), K_{d_B}) = S_A$$

$$\boldsymbol{VER}(S_A, K_{e_A}) = \boldsymbol{VER}(\boldsymbol{SIG}(M, K_{d_A}), K_{e_A}) \in \{0,1\}$$

⑦ 如果验证结果为真，则说明S_A是A的签名，否则S_A不是A的签名

⑧ B对收到的C或S_A留底

知识回顾-利用公钥密码实现数字签名

- **利用公钥密码实现数字签名的一般方法：**
- 签名通信协议安全性分析：
  - 因为只有A才拥有K_{d_A}，而且由公开的K_{e_A}在计算上不能求出保密的私钥K_{d_A}
  - 签名的操作只有A才能进行，其他任何人都不能进行
  - K_{d_A}就相当于A的印章或指纹，而S_A就是A对M的签名
  - 对此A不能抵赖，其他任何人不能伪造
  - 事后如果A和B关于签名的真伪发生争执，则他们应向公正的仲裁者出示留底的签名数据，由仲裁者当众验证签名，解决纠纷

知识回顾-利用公钥密码实现数字签名

- **利用RSA密码实现数字签名：**
  - 设M为明文，$K_{e_A}=<e,n>$是A的公钥
  - $K_{d_A}=<p,q,\varphi(n),d>$是A的私钥
- 则A对M的签名过程是：
  - $S_A=D(M,K_{d_A})=M^d \bmod n$
  - S_A便是A对M的签名
- 验证签名的过程是：
  - $E(S_A,K_{e_A})=(M^d)^e \bmod n=M$

知识回顾-利用公钥密码实现数字签名

- **对RSA数字签名的攻击**
- 一般攻击

  这种攻击在实际上有效性不大

- 利用已有的签名进行攻击；攻击签名获得**明文**

  对策：**A**不直接对数据*M*签名，而是对**HASH(*M*)**签名

- 对先加密后签名方案的攻击

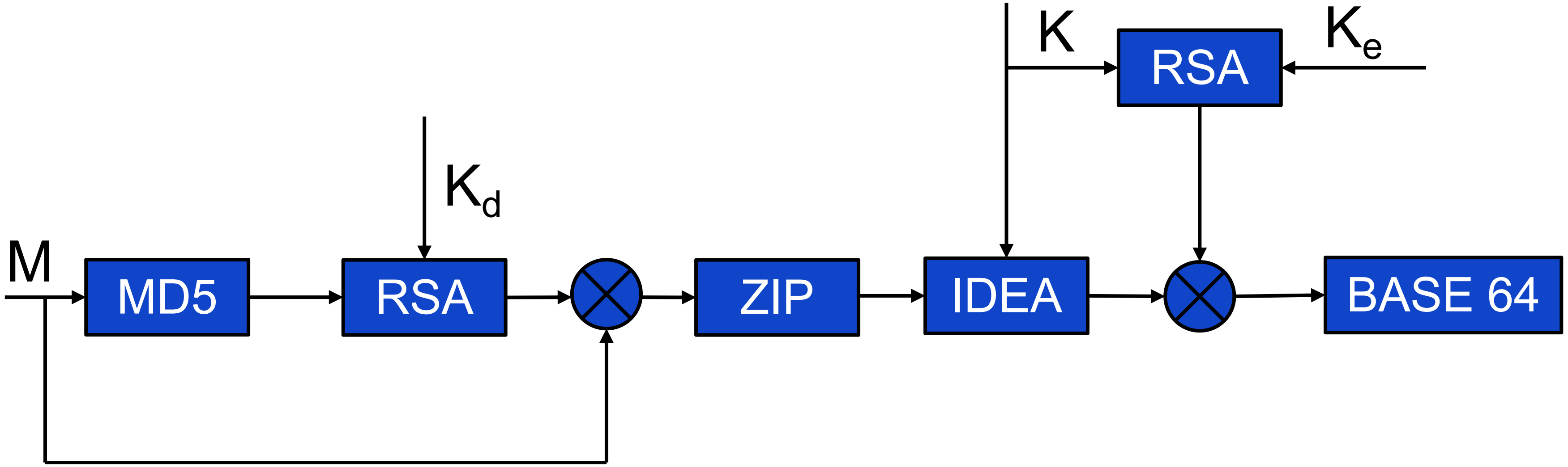
  对策：(1) A在发送数据中加入时间戳；(2)对**HASH(*M*)**签名; (3) 先签名再加密

知识回顾-利用公钥密码实现数字签名

- **对RSA数字签名的攻击**
- 结论：
  - 不直接对数据M签名，而是对HASH(M)签名
  - 使用时间戳
  - 对于同时确保机密性和真实性的通信，应当**先签名后加密**

知识回顾-利用公钥密码实现数字签名

- **RSA数字签名的应用**
- RSA数字签名的应用：PGP (Pretty Good Privacy)



知识回顾-利用公钥密码实现数字签名

- **利用ElGamal密码实现数字签名：**
  - 密钥选择：选p是一个大素数，$p-1$有大素因子，a是模p的一个本原元，将p和a公开作为密码基础参数；用户随机地选择一个整数x（$1 \le x \le p-1$）作为私有的解密密钥；计算$y = a^x \bmod p$，取y作为公开的加密密钥。
  - 产生签名：设明文为M，$0 \le M \le p-1$，签名过程如下
    ① 用户A随机地选择一个整数k，$1 < k < p-1$，且$(k, p-1) = 1$
    ② 计算$r = a^k \bmod p$
    ③ 计算$s = (M - xr)k^{-1} \bmod p-1$
    ④ 取(r, s)作为M的签名，并以$< M, r, s >$的形式发给用户B

知识回顾-利用公钥密码实现数字签名

- **利用ElGamal密码实现数字签名：**
  - 用户B接收$< M, r, s >$，用A的公钥验证

$$a^M = y^r r^s \bmod p$$

是否成立，若成立则签名为真，否则签名为假。

  - 签名的可验证性证明如下：

因为$s = (M - xr)k^{-1} \bmod p - 1$

所以$M = sk + xr \bmod p - 1$

所以$a^M = a^{sk+xr} = a^{sk}a^{xr} = y^r r^s \bmod p$，故签名可以验证。

章节安排

Outline

- 数字签名基本概念
- 数字签名的模型
- 利用公钥密码实现数字签名
- 盲签名

章节安排

Outline

- 数字签名基本概念
- 数字签名的模型
- **利用公钥密码实现数字签名**
- 盲签名

7.3(2) 利用公钥密码实现数字签名

利用ElGamal密码实现数字签名

安全性

- 从公开密钥求私钥是离散对数问题。
- $p-1$要有大素数因子，否则易受攻击。
- 为了安全，随机数k应当是一次性的。否则，时间一长，k将可能泄露。因为

$$x=(M-ks)r^{-1}\ mod\ p-1$$

而且，$<r,s>$是攻击者可以获得的，如果攻击者知道M，便可求出保密密钥x

利用ElGamal密码实现数字签名

安全性

- 如果k重复使用，如用k签名M_1和M_2。于是，有

$$M_1 = s_1 k + xr \bmod p-1$$

$$M_2 = s_2 k + xr \bmod p-1$$

故有，

$$M_1 - M_2 = (s_1 - s_2)k \bmod p-1$$

如果攻击者知道了M_1和M_2，便可求出k，进而求出保密的解密密钥x。

- 由此可知，不要随便给别人签名。不要直接对M签名，而是对HASH(M)签名。

利用ElGamal密码实现数字签名

应用

- 优点：安全，方便
- 缺点：由于取$< r, s >$作为M的签名，所以数字签名的长度是明文的两倍，数据扩张一倍；随机数k，要求实际应用时要有高质量的随机数产生器，增加了工程实现的工作量。
- 美国数字签名标准（DSS）的签名算法DSA是它的一种变形
- 俄罗斯数字签名标准（GOST）也是采用一种ElGamal密码签名变种

美国数字签名标准的签名算法DSA

密钥选择：

① 选取一个素数p，要求$2^{L-1} < p < 2^{L}$，其中$L = 512 + 64j,\ j = 0,1,2, \ldots, 8$;

② 选取一个素数q，它是$p-1$的因子，$2^{159} < q < 2^{160}$;

③ 计算$g = h^{(p-1)/q} \bmod p$，h满足$1 < h < p-1$，且满足使$h^{(p-1)/q} \bmod p > 1$;

④ 随机选择一个数x作为用户的私钥，$0 < x < q$;

⑤ 计算$y = g^{x} \bmod p$作为用户的公钥。

参数p、q和g可以公开。

7.3(3) 利用公钥密码实现数字签名

美国数字签名标准的签名算法DSA

- 产生签名：设明文为M，签名过程如下

① 用户A随机地选择一个整数k, $0 < k < q$;

② 计算$r = (g^k \bmod p) \bmod q$;

③ 计算$s = k^{-1}(\mathrm{SHA}(M) + xr) \bmod q$;

④ 检验r和s是否为0，若其中一个为0，则重新选取k，重新计算r和s;

⑤ 取(r, s)作为M的签名，并以$< M, r, s >$的形式发给用户B

7.3(3) 利用公钥密码实现数字签名

美国数字签名标准的签名算法DSA

- 用户B接收到$< M, r, s >$，根据$< p, q, g >$，验证过程如下

① 检验$0 < r < q$及$0 < s < q$是否成立，若其中之一不成立，签名为假；

② 计算

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

$$u_1 = \mathrm{SHA}(M) w \bmod q$$

$$u_2 = rw \bmod q$$

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

若$v = r$，则签名为真；否则签名为假或数据被篡改。

美国数字签名标准的签名算法DSA

- 签名的可验证性证明如下：

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

$$= \left(g^{\mathrm{SHA}(M)w\ mod\ q} y^{rw\ mod\ q}\ mod\ p\right)\ mod\ q$$

$$= \left(g^{\mathrm{SHA}(M)w+xrw\ mod\ q}\ mod\ p\right)\ mod\ q$$

$$= \left(g^{(\mathrm{SHA}(M)\ +xr)w\ mod\ q}\ mod\ p\right)\ mod\ q$$

因为$w = s^{-1}\ mod\ q = k(\mathrm{SHA}(M) + xr)^{-1}\ mod\ q$，于是

$$v = \left(g^{k\ mod\ q}\ mod\ p\right)\ mod\ q$$

7.3(3) 利用公钥密码实现数字签名

美国数字签名标准的签名算法DSA

- 签名的可验证性证明如下：

$$v = \left(g^{k \bmod q} \bmod p\right) \bmod q$$

$$= \left(g^{k+tq} \bmod p\right) \bmod q$$

$$= \left(g^k g^{tq} \bmod p\right) \bmod q$$

$$= \left(g^k \left(h^{(p-1)/q}\right)^{tq} \bmod p\right) \bmod q$$

$$= \left(g^k h^{t(p-1)} \bmod p\right) \bmod q$$

$$= \left(g^k \bmod p\right) \bmod q = r$$

利用椭圆曲线实现数字签名（ECDSA）

- 一个椭圆曲线密码由下面的六元组描述：

$$T = \ < p,\ a,\ b,\ G,\ n,\ h >$$

其中，p为大于3素数，p确定了有限域$GF(p)$；元素$a,\ b \in GF(p)$，a和b确定了椭圆曲线；G为循环子群E_1的生成元，n为素数且为生成元G的阶，G和n确定了循环子群E_1。

$$y^2 = x^3 + ax + b \bmod p$$

7.3(4) 利用公钥密码实现数字签名

- **利用椭圆曲线实现数字签名（ECDSA）**
  - 密钥选择：
    ① 随机选择一个数$d \in \{1,2,\ldots,n-1\}$作为用户的私钥；
    ② 计算$Q = dG$作为用户的公钥。
  - 生成签名：设明文为M，签名过程如下
    ① 随机选择一个数$k \in \{1,2,\ldots,n-1\}$；
    ② 计算$R(x_R, y_R) = kG$，并记$r = x_R \bmod n$；
    ③ 计算$s = (\mathrm{HASH}(M) + rd)k^{-1} \bmod n$；
    ④ 取(r, s)作为M的签名，并以$< M, r, s >$的形式传输或存储

7.3(4) 利用公钥密码实现数字签名

- **利用椭圆曲线实现数字签名（ECDSA）**
  - 接收到消息$< M, r, s >$，验证过程如下

① 计算

$$e = \mathrm{HASH}(M)$$

$$u = es^{-1} \, mod \, n$$

$$v = rs^{-1} \, mod \, n$$

② 利用公钥Q计算$R(x_R, y_R) = uG + vQ$；

③ 如果$r = x_R \, mod \, n$，则签名为真；否则签名为假或数据被篡改。

7.3(4) 利用公钥密码实现数字签名

利用椭圆曲线实现数字签名（ECDSA）

- 签名的可验证性证明如下：

因为$s=(\mathrm{HASH}(M)+rd)k^{-1} \bmod n=(e+rd)k^{-1} \bmod n$,

所以$s^{-1}=k(e+rd)^{-1} \bmod n$

所以$uG+vQ=ek(e+rd)^{-1}G+rk(e+rd)^{-1}Q$

$$=(e+rd)^{-1}(ekG+rkQ)$$

$$=(e+rd)^{-1}(ekG+rkdG)$$

$$=(e+rd)^{-1}(e+rd)kG=R$$

7.3(4) 利用公钥密码实现数字签名

利用椭圆曲线实现数字签名（ECDSA）

应用

- 特点： 安全、密钥短、软硬件实现节省等
- 2000年美国政府已将椭圆曲线密码引入数字签名标准DSS
- 中国也采用椭圆曲线密码签名

7.3(5) 利用公钥密码实现数字签名

利用SM2算法实现数字签名

- 密钥推荐使用256位素域$GF(p)$上的椭圆曲线 $y^2 = x^3 + ax + b$，参数如下：

$p = FFFFFFFE\ FFFFFFFF\ FFFFFFFF\ FFFFFFFF\ FFFFFFFF\ 00000000\ FFFFFFFF\ FFFFFFFF$

$a = FFFFFFFE\ FFFFFFFF\ FFFFFFFF\ FFFFFFFF\ FFFFFFFF\ 00000000\ FFFFFFFF\ FFFFFFFC$

$b = 28E9FA9E\ 9D9F5E34\ 4D5A9E4B\ CF6509A7\ F39789F5\ 15AB8F92\ DDBCBD41\ 4D940E93$

$n = FFFFFFFE\ FFFFFFFF\ FFFFFFFF\ FFFFFFFF\ 7203DF6B\ 21C6052B\ 53BBF409\ 39D54123$

$x_G = 32C4AE2C\ 1F198119\ 5F990446\ 6A39C994\ 8FE30BBF\ F2660BE1\ 715A4589\ 334C74C7$

$y_G = BC3736A2\ F4F6779C\ 59BDCEE3\ 6B692153\ D0A9877C\ C62A4740\ 02DF32E5\ 2139F0A0$

- 密钥选择：

① 私钥是随机数d，$d \in [1, n-2]$

② 公钥$P = dG = d(x_G, y_G) = (x_P, y_P)$

7.3(5) 利用公钥密码实现数字签名

利用SM2算法实现数字签名

- 在利用SM2算法实现数字签名时，签名者和验证者都需要用密码学杂凑函数求得用户A的杂凑值

$$Z_A = H_{256}(ENTL_A \parallel ID_A \parallel a \parallel b \parallel x_G \parallel y_G \parallel x_P \parallel y_P)$$

其中ID_A是用户的标识，$ENTL_A$是由标识长度转换而成的两个字节；

$H_{256}(\ )$选用的SM3；

SM2 密码算法使用规范（GM/T 0009-2012）规定，在无特殊约定的情况下，用户身份标识ID默认值从左至右依次为：

0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38。

7.3(5) 利用公钥密码实现数字签名

利用SM2算法实现数字签名

- 生成签名：设明文为M，签名过程如下

① 置$\overline{M} = Z_A || M$;

② 计算$e = H_v(\overline{M})$ 并将e的数据表示为整数;

③ 用随机数发生器产生随机数$k \in [1, n-1]$；

④ 计算椭圆曲线点$kG = (x_1, y_1)$，并将x_1的数据表示为整数;

⑤ 计算$r = e + x_1 \bmod n$，若$r = 0$或$r + k = n$，则返回③;

⑥ 计算$s = (1+d)^{-1}(k - rd) \bmod n$，若$s = 0$则返回③;

⑦ 取(r, s)作为M的签名，并以$< M, r, s >$的形式传输或存储。

利用SM2算法实现数字签名

- 接收到消息$< M', r', s' >$，验证过程如下

① 检验$r' \in [1, \ n-1]$是否成立，若不成立则验证不通过；

② 检验$s' \in [1, \ n-1]$是否成立，若不成立则验证不通过；

③ 置 $\overline{M'} = Z_A \,||M'$；

④ 计算$e' = H_v(\overline{M'})$ 并将e'的数据表示为整数；

⑤ 将r', s'的数据表示为整数，计算$t = (r' + s') \, mod \, n$，若$t = 0$，则验证不通过；

⑥ 计算椭圆曲线点$(x_1{}', y_1{}') = s'G + tP$；

⑦ 计算$R = e' + x_1{}' \, mod \, n$，检验$R = r'$是否成立，若成立，则验证通过；否则，验证不通过。

利用SM2算法实现数字签名

- 签名的可验证性证明如下：

① 因为签名时确保了$r \in [1, n-1]$和$s \in [1, n-1]$，如果没有篡改和错误，则有$r' \in [1, n-1]$和$s' \in [1, n-1]$。对此进行检验，可发现其是否有错误或被篡改，确保其完整性。

② 因为签名时确保了$r \neq 0$且$s \neq 0$，如果$t = (r+s) = 0 \bmod n$，则说明$r+s$是n的整数倍。而签名时确保了$r+k$不是n的整数倍，$r+s = r+(1+d)^{-1}(k-rd) = \frac{r+k}{1+d}$也不是$n$的整数倍。否则因为$d$是正整数，将导致$r+s$是$n$的整数倍，产生矛盾。这说明，如果$r'$和$s'$是正确的，则$t = (r'+s') \bmod n = t \neq 0$。

利用SM2算法实现数字签名

- 签名的可验证性证明如下：

③ 一方面

$$sG + tP = sG + (r + s)(dG) = (s + rd + sd)G$$

另一方面

$$s + rd + sd = s(1 + d) + rd = \frac{k - rd}{1 + d}(1 + d) + rd = k$$

所以$sG + tP = kG = (x_1, y_1)$。这说明，如果$x_1{}'$和$e'$没有被篡改或错误，则有$x_1' = x_1$， $e' = e$，因而有$R = r'$，签名可验证。

章节安排

Outline

盲签名的概念和需求

- 在普通数字签名中，签名者总是先知道数据的内容后才实施签名，这是通常的办公事务所需要的。但有时却需要某个人对某数据签名，而又不能让他知道数据的内容。称这种签名为盲签名（Blind Signature）；
- 在无记名投票选举和数字货币系统中往往需要这种盲签名；
- 盲签名在电子商务和电子政务系统中有着广泛的应用。

7.6 盲签名

- **盲签名与普通签名相比有两个显著的特点**
  - 签名者不知道所签署的数据内容；
  - 在签名被接收者泄露后，签名者不能追踪签名。即：如果把签名的数据给签名者看，他确信是自己的签名，但他无法知道什么时候对什么样的盲数据施加签名而得到此签名数据。

7.6 盲签名

盲签名的技术思想

- 接收者首先将待签数据进行盲变换，把变换后的盲数据发给签名者；
- 经签名者签名后再发给接收者；
- 接收者对签名再作去盲变换，得出的便是签名者对原数据的盲签名。

这样便满足了条件①。要满足条件②，必须使签名者事后看到盲签名时不能与盲数据联系起来，这通常是依靠某种协议来实现的。

数据 → 盲变换 → 签名 → 去盲变换 → 盲签名

感谢聆听!

T H A N K Y O U F O R Y O U R A T T E N T I O N !