应用密码学（第九讲）
— 数字签名

林东岱

信息安全国家重点实验室

2022年9月

数字签名

在人们的工作和生活中，许多事物的处理需要当事者签名。例如，政府部门的文件、命令、证书、 商业合同和财务凭证等都需要当事者签名。签名起到确认、核准、生效和负责等多种作用。

实际上，签名是证明当事者的身份和数据真实性的一种信息。既然签名是一种信息， 因此签名可以用不同的形式来表示。在传统的以书面文件为基础的事物处理中，采用书面签名的形式，如印章、手印等。书面签名得到司法部门的支持，具有一定的法律意义。在以计算机文件为基础的现代事物处理中，应采用电子形式的签名，即数字签名。

什么是数字签名？

ISO数字签名的定义：

数字签名是指附加在数据单元上的一些数据，或是对数据单元所做的密码变换， 这种数据或变换允许数据单元的接收者用以确认数据单元的来源和数据单元的完整性，并保护数据，防止被伪造。

基本概念

- 数字签名是公钥密码学发展过程中最重要的概念之一, 它可以提供其他方法难以实现的安全性.
- 数字签名: 对数字形式的消息进行所属性证明的一种方法.
- 消息认证 (MAC) 可以保护信息交换双方不受第三方的攻击, 但是它不能处理通信双方自身发生的攻击, 这种攻击可以有多种形式. 例如,
  - Alice 用与 Bob 共享的密钥产生认证消息发送给 Bob. Bob 可以伪造一条消息并称该消息发自 Alice, 因为 Bob 也有相同的密钥.
  - Alice 可以否认曾发送过某条消息. 因为 Bob 可以伪造消息, 所以无法证明 Alice 确实发送过该消息.

  数字签名可以很好地解决这类问题.

基本概念

数字签名必须具有下列特征:

- 它必须能验证签名者、签名日期和时间;
- 它必须能认证被签名的消息内容;
- 签名应能由第三方仲裁，以解决争执.

因此, 数字签名应该满足下列条件:

- 签名必须与被签名的消息相关联;
- 签名必须使用发送方某些独有的信息, 以防止伪造和否认;
- 产生数字签名比较容易;
- 识别和验证签名比较容易;
- 伪造数字签名在计算上是不可行的. 无论是从给定的数字签名伪造消息, 还是从给定的消息伪造数字签名, 在计算上都是不可行的.

基本概念

数字签名与物理签名的差异:

- 签署文件的属性: 在传统签名模式中, 手写签名是所签署文件的物理部分. 然而, 数字签名没有物理地附加在所签文件上, 因此签名算法必须以某种形式将签名“绑定”到所签文件上.
- 签名的验证: 一个传统的签名通过比较其它已认证的签名来验证当前签名的真伪. 数字签名却能通过一个公开的验证算法对它进行确认. 这样, “任何人” 都能验证一个数字签名. 安全数字签名方案的使用能阻止伪造签名的可能性.
- 签名的可复制性: 数字签名文件的"拷贝"与原签名文件相同, 而仿造(如复印)的手写签名文件能与原来的签名文件区分开来. 这意味着我们必须采取措施防止一个数字签名消息被“重复”使用.

数字签名应具有的基本特征

定义 (数字签名方案)

一个签名方案是一个满足下列条件的三元组 $(\mathtt{Gen}, \mathtt{Sig}, \mathtt{Ver})$: 满足

1. $\mathtt{Gen}$ 生成密钥对 (sik, vk). sik 称为签名密钥, vk 是验证密钥, 用于验证.
2. $\mathtt{Sig}$ 是签名算法. 输入签名密钥 sik 和要签名的消息 m, 得到一个签名 $\sigma = \mathtt{Sig}_{sik}(m)$.
3. $\mathtt{Ver}$ 是验证算法. 输入一个签名 σ 和消息 m, 输出 0 或者 1.

方案的可靠性(正确性)要求使用一个签名密钥 sik 对于一个消息 m 的签名, 一定能够通过相应的验证密钥 vk 加上消息 m 的验证. 即

$$\mathtt{Ver}_{vk}(\mathtt{Sig}_{sik}(m), m) = 1$$

数字签名应具有的基本特征

- $\mathtt{Gen}, \mathtt{Sig}, \mathtt{Ver}$ 应该是多项式时间函数. 都是公开的函数, vk 是公开的. 而 sik 是保密的.
- 给定一个消息 m, 除非具有签名密钥的主体, 要计算满足 $\mathtt{Ver}_{vk}(m, \sigma') = 1$ 的数字签名 σ', 应该在计算上不可行. (注意: 对给定的 m, vk, 可能存在不止一个 σ, 这要看函数 $\mathtt{Sig}$ 是如何定义的).
- 如果敌手能够计算出满足 $\mathtt{Ver}_{vk}(m, \sigma) = 1$ 的数据对 (m, σ), 而 m 没有事先被签名, 或者 σ 与已有签名不同, 则签名 (m, σ) 称为一个伪造签名.
- 签名与 MAC 的区别: 1) MAC 使用对称的思想, 不具有专有性; 故只能够对于消息进行认证, 不能对于消息源进行认证. 2) 签名使用公钥方案, 私钥具有专有性, 故可以对于消息源进行认证.

数字签名的攻击模型

对签名方案来讲, 经常考虑下面的攻击模型:

- 唯密钥攻击: 敌手仅仅拥有 Alice 的验证密钥, 即公开的密钥 vk. 企图伪造签名.
- 已知消息攻击: 敌手拥有一系列以前由 Alice 签名的消息.

$$(m_1, \sigma_1), (m_2, \sigma_2), \ldots, (m_\ell, \sigma_\ell)$$

其中 m_i 是消息而 σ_i 是 Alice 对消息 m_i 的签名(因此 $\mathtt{Ver}_{vk}(m_1, \sigma_i) = 1$). 企图伪造签名.

- 选择消息攻击: 敌手请求 Alice 对一个消息列表签名. 因此他选择消息 $m_1, m_2, \ldots, m_s$, 并且 Alice 提供对这些消息的签名 $\sigma_1, \sigma_2, \ldots, \sigma_s$. 敌手企图伪造签名.

数字签名的攻击模型

下面考虑攻击者可能的几种目的:

- 完全攻破: 攻击者能够确定 Alice 的私钥 *sik*. 因此能够对任何消息生成有效的签名.
- 选定消息的伪造: 攻击者能(以某一不可忽略的概率)对自己事先所选定的消息产生一个有效的签名. 即, 攻击者可以选定某个消息 m, (以某种概率)产生签名 σ, 使得 $\mathtt{Ver}_{vk}(m,\sigma)=1$. 该消息 m 不应该是以前 Alice 曾经签过名的消息, 或者 σ 不是 m 已有的签名.
- 存在伪造: 攻击者至少能够为某一消息产生一个有效的签名. 换句话说, 攻击者能创建一个数对 (m,σ), 其中 m 是消息而 σ. 该消息 m 不应该是以前 Alice 曾经签名的消息.

签名方案的构造

1. 将公钥方案用于签名
   Diffie和Hellman的公开密钥系统的思想，对实现数字签名提供了一种简单的实现方法（假定A向B发送一条消息M）：
   (1) A计算出$C = D_A(M)$，对M签名；
   (2) B通过检查$E_A(C)$是否恢复M，验证A的签名；
   (3) 如果A和B之间发生争端，仲裁者可以用(2)中的方法鉴定A的签名。
2. 直接构造签名方案

9.2 RSA 签名

9.2.1 利用 RSA 密码实现数字签名

RSA 签名算法的安全性是建立在大数分解问题的困难性上的，具体过程如：

（1）签名方产生一对公开密钥(n，e)和私有密钥(n，d)，签名过程如下：

（2）用私有密钥(n，d)加密消息得到：$s=M^d \bmod n$。

签名完成之后，签名方将(M，s)发送给接收方。接收方收到签名(M_1，s)之后通过以下步骤进行验证：

（1）用公开密钥(n，e)加密消息摘要得到：$s_1=M_1^e \bmod n$。如果$s_1=s$，那么验证通过，否则拒绝接受签名。因为当且仅当$M_1=M$时，$s=s_1$。

9.2.2 对 RSA 数字签名的攻击

RSA 的数字签名很简单，但要实际应用还要注意许多问题。

（1）一般攻击

由于 RSA 密码的加密运算和解密运算具有相同的形式，都是模幂运算。设 e 和 n 是用户 A 的公开密钥，所以任何人都可以获得并使用 e 和 n。攻击者首先随意选择一个数据 Y，并用 A 的公开密钥计算 $X=Y^e \bmod n$，于是便可以用 Y 伪造 A 的签名。因为 X 是 A 对 Y 的一个有效签名。

这种攻击实际上的成功率是不高的。因为对于随意选择的 Y，通过加密运算后得到的 X 具有正确语义的概率是很小的。

可以通过认真设计数据格式或采用 Hash 函数与数字签名相结合的方法阻止这种攻击。

（2）利用已有的签名进行攻击

假设攻击者想要伪造 A 对 M_3 的签名，他很容易找到另外两个数据 M_1 和 M_2，使得

$$M_3 = M_1 M_2 \bmod n,$$

他设法让 A 分别对 M_1 和 M_2 进行签名：

$$S_1 = (M_1)^d \bmod n,$$
$$S_2 = (M_2)^d \bmod n。$$

于是攻击者就可以用 S_1 和 S_2 计算出 A 对 M_3 的签名 S_3：

$$(S_1 S_2) \bmod n = ((M_1)^d (M_2)^d) \bmod n = (M_3)^d \bmod n = S_3。$$

对付这种攻击的方法是用户不要轻易地对其他人提供的随机数据进行签名。更有效的方法是不直接对数据签名，而是对数据的 Hash 值签名。

（3）利用签名进行攻击获得明文

设攻击者截获了密文 C，$C = M^e \bmod n$，他想求出明文 M。于是，他选择一个小的随机数 r，并计算

$$x = r^e \bmod n,$$
$$y = x C \bmod n,$$
$$t = r^{-1} \bmod n。$$

因为 $x = r^e \bmod n$，所以 $x^d = (r^e)^d \bmod n$，$r = x^d \bmod n$。然后攻击者设法让发送者对 y 签名，于是攻击者又获得

$$S = y^d \bmod n$$

攻击者计算

$$tS \bmod n = r^{-1} y^d \bmod n = r^{-1} x^d C^d \bmod n = C^d \bmod n = M,$$

于是攻击者获得了明文 M。

对付这种攻击的方法也是用户不要轻易地对其他人提供的随机数据进行签名。最好是不直接对数据签名，而是对数据的 Hash 值签名。

（4）对先加密后签名方案的攻击

我们已经介绍了先签名后加密的数字签名方案，这一方案不仅可以同时确保据的真实性和秘密性，而且还可以抵抗对数字签名的攻击。

假设用户 A 采用先加密后签名的方案把 M 发送给用户 B，则他先用 B 的公开密钥 e_B 对 M 加密，然后用自己的私钥 d_A 签名。再设 A 的模为 n_A，B 的模为 n_B。于是 A 发送如下的数据给 B：

$$((M)^{e_B} \bmod n_B)^{d_A} \bmod n_A,$$

如果 B 是不诚实的，则他可以用 M_1 抵赖 M，而 A 无法争辩。因为 n_B 是 B 的模，所以 B 知道 n_B 的因子分解，于是他就能计算模 n_B 的离散对数，即他就能找出满足 $(M_1)^x = M \bmod n_B$ 的 x，然后他公布他的新公开密钥为 xe_B。这时他就可以宣布他收到的是 M_1 而不是 M。

A 无法争辩的原因在于下式成立：

$$((M_1)^{xe_B} \bmod n_B)^{d_A} \bmod n_A = ((M)^{e_B} \bmod n_B)^{d_A} \bmod n_A。$$

为了对付这种攻击，发送者应当在发送的数据中加入时间戳，从而可证明是用 e_B 对 M 加密而不是用新公开密钥 x e_B 对 M_1 加密。另一种对付这种攻击的方法是经过 Hash 处理后再签名。

这里介绍了四种对数字签名的攻击或利用签名进行攻击获得明文的方法，由此可以得出以下结论：

① 不要直接对数据签名，而应对数据的 Hash 值签名。

② 要采用先签名后加密的数字签名方案，而不要采用先加密后签名的数字签名方案。

9.3 数字签名标准 DSS

9.3.1 DSS 的基本方式

1991 年美国政府颁布了数字签名标准 DSS（Digital Signature Standard），这标志着数字签名已得到政府的支持。和当年推出 DES 时一样，DSS 一提出便引起了一场激烈的争论。反对派的代表人物是 MIT 的 Rivest 和 Standford 的 Hellman 教授。反对的意见主要认为，DSS 的密钥太短，效率不如 RSA 高，不能实现数据加密，并怀疑 NIST 在 DSS 中留有“后门”。尽管争论十分激烈，最终美国政府还是颁布了 DSS。针对 DSS 密钥太短的批评，美国政府将 DSS 的密钥从原来的 512 位提高到 512~1024 位，从而使 DSS 的安全性大大增强。从 DSS 颁布至今尚未发现 DSS 有明显缺陷。目前，DSS 的应用已十分广泛，并被一些国际标准化组织采纳作为标准。2000 年 1 月美国政府将 RSA 和椭圆曲线密码引人数字签名标准 DSS，进一步丰富了 DSS 的算法。美国的一些州已经通过了相关法律，正式承认数字签名的法律意义。这是数字签名得到法律支持的一个重要标志。可以预计，从此以后 DSS 数字签名标准将得到广泛的应用。

DSS 使用的算法只能提供数字签名功能，它的整个过程如图 9.3 所示。

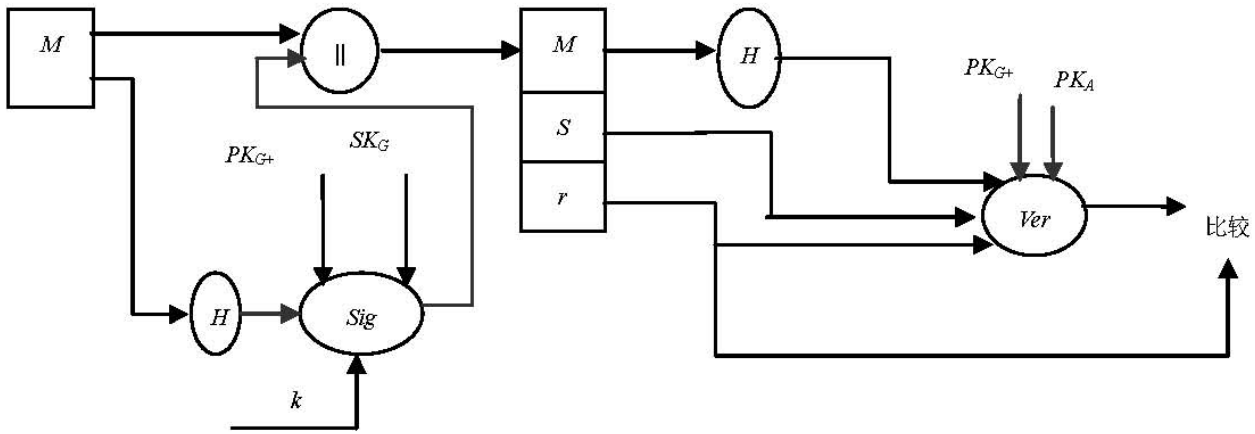

图 9.3　DSS 体制

DSS 签名利用一个单向 Hash 函数产生消息的一个 Hash 值，Hash 值连同随机数 k 一起作为签名函数的输入，签名函数还需使用发方的秘密钥 SK_A 和供所有用户使用的一组参数，称这组参数为全局公开钥 PK_G。签名函数的两个输出 s 和 r 就构成了消息的签名（s，r）。接收方收到消息后再产生出消息的 Hash 值，将 Hash 值与收到的签名一起输入验证函数，此外，验证函数还需输入全局公开钥 PK_G 和发方的公开钥 PK_A。验证函数的输出如果与收到的签名成分 r 相等，则证明了签名是有效的。

9.3.2 数字签名算法 DSA

1. 算法参数

DSS 的签名算法称为 DSA，DSA 使用以下参数：

① p 为素数，要求 $2^{L-1}<p<2^{L}$，其中 $512\leqslant L\leqslant 1024$ 且 L 为 64 的倍数，即

$$L=512+64j \qquad j=0，1，2，\ldots，8$$

② q 为素数，它是$(p-1)$的因子，$2^{159}<q<2^{160}$。

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

④ x 为随机数，$0<x<q$。

⑤ $y=g^{x} \bmod p$。

⑥ k 为随机数，$0<k<g$。

这里参数 p，q，g 可以公开，且可为一组用户公用。x 和 y 分别为一个用户的私钥和公钥。所有这些参数可在一定时间内固定。参数 x 和 k 用于产生签名，必须保密。参数 k 必须对每一签名都重新产生，且每一签名使用不同的 k。

2. 签名的产生

对数据 M 的签名为数 r 和 s，它们分别如下计算产生：

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

$$s=(k^{-1}(\mathrm{SHA}(M)+x r)) \bmod q。$$

其中 k^{-1} 为 k 的乘法逆元，即 $k^{-1}k=1 \bmod q$，且 $0<k^{-1}<q$。SHA 是安全的 Hash 函数，它从数据 M 抽出其摘要 SHA (M)，SHA (M)为一个 160 位的二进制数字串。

应该检验计算所得的 r 和 s 是否为零，若 $r=0$ 或 $s=0$，则重新产生 k，并重新计算产生签名 r 和 s。

最后，把签名 r 和 s 是附在数据 M 后面发给接收者：$(M \| r \| s)$。

3. 验证签名

为了验证签名，要使用参数 p，q，g，用户的公开密钥 y 和其标识符。令 MP，rP，sP 分别为接收到的 M，r 和 s。

① 首先检验是否有 $0<rP<q$，$0<sP<q$，若其中之一不成立，则签名为假。

② 计算：

$$w=(sP^{-1}) \bmod q,$$
$$u_1=(\mathrm{SHA}(MP)\, w) \bmod q,$$
$$u_2=((rP)\, w) \bmod q,$$
$$v=(((g)^{u_1}(y)^{u_2} \bmod p) \bmod q。$$

③ 若 $v=rP$，则签名为真，否则签名为假或数据被篡改。

9.4 其他数字签名方案

9.4.1 离散对数签名体制

1. 离散对数签名体制

（1）体制参数

p：大素数；

q：$(p-1)$或 $p-1$ 的大素因子；

g：$g \in {}_R Z_p^*$，$g^q = 1 \bmod p$；

x：用户秘密，x 为在 $1<x<q$ 内的随机数；

y：用户公钥 $y = g^x \bmod p$。

（2）签名产生过程

1. 计算 m 的杂凑值；

2. 选择随机数 k：$1<k<q$，计算 $r = g^k \bmod p$；

3. 从签字方程 $ak = b + cx \bmod q$ 中解出 s，(r, s)为数字签字。系数 a，b 和 c 根据情况而变，下表中每行给出了六个可能值。

表 9.1　a、b、c 的可能置换

a	b	c
$\pm r$	$\pm s$	$H(m)$
$\pm rH(m)$	$\pm s$	1
$\pm rH(m)$	$\pm H(m)s$	1
$\pm H(m)r$	$\pm rs$	1
$\pm H(m)s$	$\pm rs$	1

（3）签名验证

$$Ver(y,(r,s),m)=True \Leftrightarrow r^a \equiv g^b y^c \pmod p。$$

2. ElGamal 数字签名

选 p 是一个大素数，$p-1$ 是有大素数因子。α是一个模 p 的本原元，将 p 和α公开。用户随机地选择一个整数 x 作为自己的秘密的解密钥，$1\leqslant x\leqslant p-1$，计算 $y\equiv\alpha^x \bmod p$，取 y 为自己的公开的加密钥。公开参数 p 和α可以由一组用户共用。

（1）产生签名

设用户 A 要对明文消息 M 加签名，$0\leqslant m\leqslant p-1$，其签名过程如下：

① 用户 A 随机地选择一个整数 k，$1<k\leqslant p-1$，且$(k,p-1)=1$；

② 计算 $r=\alpha^k \bmod p$；

③ 计算 $s=(m-x_A r)k^{-1} \pmod{p-1}$；

④ 取(r,s)作为 M 的签名，并以$<m,r,s>$的形式发给用户 B。

（2）验证签名

用户 B 验证$\alpha^m=y_A{}^r r^s$，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$s=(m-x_A r)k^{-1}\pmod{p-1}$，所以$m=x_A r+k s\pmod{p-1}$，从而$\alpha^m=\alpha^{x_A r+k s}=y_A{}^r r^s \bmod p$，故签名可验证。

对于上述 ElGamal 数字签名，为了安全，随机数 k 应当是一次性的。否则，可用过去的签名冒充现在的签名。

注意，由于取(r, s)作为M的签名，所以 ElGamal 数字签名的数据长度是明文的两倍，即数据扩展一倍。

例：取$p=11$，生成元$\alpha=2$，私钥$x=8$。计算公钥

$$y=\alpha^x \bmod p=2^8 \bmod 11=3。$$

取明文$m=5$，随机数$k=9$，因为$(9, 11)=1$，所以$k=9$是合理的。计算

$$r=\alpha^k \bmod p=2^9 \bmod 11=6。$$

再利用 Euclidean 算法从下式求出s，过程如下：

$$M=(k s+x_A r)\bmod p-1$$
$$5=(9 s+8\times 6)\bmod 10$$
$$s=3$$

于是，签名$(r, s)=(6, 3)$。

为了验证签名，需要验证$\alpha^M=y_A{}^r r^s$是否成立。为此计算

$$\alpha^M=2^5 \bmod 11=32 \bmod 11=10,$$
$$y_A{}^r r^s \bmod p=3^6\times 6^3 \bmod 11=729\times 216 \bmod 11=157464 \bmod 11=10,$$

因为 10=10，通过签名验证，这说明签名是真的。

（3）安全性分析

以下探讨有关 ElGamal 签名体制安全性的几个结果。

1）在签名验证过程中，检验$r<p$的重要性。Bleichenbacher 发现了下面的攻击，条件是在 Bob 接受的签名中有$r>p$成立。设(r, s)表示消息m的签名，Malice 可以通过下面的方法伪造一个消息m'的一个新签名：

① $u=m'm^{-1}(\bmod p-1)$；

② $s'=s\,u(\bmod p-1)$；

③ 计算 r'，使之满足 $r'=r\,u(\bmod p-1)$和 $r'=r(\bmod p)$；这可以应用中国剩余定理来完成。

然后，例行的做法是检验下面的同余等式成立：

$$y_A{}^{r'}r^{s'}=y_A{}^{ru}r^{su}=(y_A{}^{r}r^{s})^{u}=\alpha^{mu}=\alpha^{m'}(\bmod p)$$

如果 Bob 对 $r<p$ 进行验证，就会阻止这样的攻击。这是因为在上面第 3 步中由中国剩余定理所计算的 r'是一个 $p(p-1)$量级的量。

2）这也是由 Bleichenbacher 发现的：Alice 应该从 F_p^*中随机地选取公开参数α。如果该参数不是由 Alice 选取地，那么必须有一个所有用户都知道地公共过程来检验α选择的随机性。

现在假设公共参数α，p 是由 Malice 选择的。参数 p 可以按如下方法来建立：设 $p-1=b\,q$，其中 q 可以是一个足够大的素数，而 b 可以是平滑的（也就是说，b 仅有一个小的素因子，因此在 b 阶群上计算离散对数是容易的）。

Malice 可以如下产生α：

$$\alpha=\beta^{t}(\bmod p)$$

对某个$\beta=c\,q$，有 $c<b$。

对 Alice 的公钥y_A，我们知道，当底为α时，求y_A的离散对数是困难的。但是，当底为α^q时，求取$y_A{}^q$的离散对数是容易的。该离散对数为$z=x_A(\bmod b)$，也就是说，下面的同余等式成立：

$$y_A{}^{q}=(g^{q})^{z}(\bmod p)。$$

有了 z，Malice 可以伪造 Alice 的签名如下：

$$r=\beta=c\,q,$$

$$s=t\,(m-c\,q\,z)(\bmod p-1)。$$

然后，例行的做法是检验下面的同余等式成立：

$$y_A{}^{r}r^{s}=y_A{}^{cq}(\beta^{t})^{(m-cqr)}=\alpha^{cqr}\alpha^{rq-cqr}=g^{m}(\bmod p)。$$

因此，(r, s)实际上是 m 的一个有效签名，其生成过程没有使用 x_A，而是使用 $x_A \pmod b$。

在这个签名伪造攻击中，r 是一个可以被 q 整除的值。而在 p 的标准参数建立阶段，p 满足 $p=b\,p$，其中 q 是一个大的素数。因此，在验证过程中，如果 Bob 验证$(q\times r)$，那么就可以防范 Bleichenbacher 所发现的攻击。

3）最后一个关于短暂密钥 k。类似于 ElGamal 加密：ElGamal 签名的生成过程也是一个随机的算法。它的随机性是由于这个短暂密钥 k 的随机性。

Alice 永远不要在不同的签名过程中重复使用同一个短暂密钥。如果重复使用一个短暂密钥 k 对两个消息 $m_1 \neq m_2 \pmod{p-1}$进行签名，那么可有

$$K(s_1-s_2)=m_1-m_2 \pmod{p-1}$$

因为 $k^{-1} \pmod{p-1}$存在，$m_1 \neq m_2 \pmod{p-1}$意味着

$$k^{-1}=(s_1-s_2)/(m_1-m_2) \pmod{p-1}$$

即得到了 k^{-1}。依次地，可以计算出 Alice 地私钥 x_A：

$$x_A=(m_1-k\,s_1)/r \pmod{p-1}$$

还应该注意，这个短暂地密钥必须从空间 Z_{p-1}^*中随机均匀地选取。当一个签名是由小型计算机生成时，比如智能卡或掌上设备等，应该特别注意：必须要确保这些设备配足了足够的、可依赖的随机资源。

只要保证 k 在每次签名中仅使用一次，并且它是随机均匀产生的，那么签名过程的第二个等式说明它实质上是为签名者的私钥 x 提供了一次性的乘法加密。因此，这两个密钥在信息论安全的意义上彼此互相保护。

3. Schnorr 签名体制

（1）体制参数

p，q：大素数，$q \mid p-1$。q是大于等于160 bits的整数，p是大于等于512 bits的整数，保证Z_p中求解离散对数困难；

g：Z_p^*中元素，且$g^q \equiv 1 \bmod p$；

x：用户密钥$1<x<q$；

y：用户公钥$y \equiv g^x \bmod p$。

空间$M = Z_p^*$，签字名空间$S = Z_p^* \times Z_q$；密钥空间$K = \{(p, q, g, x, y): y \equiv g^x \bmod p\}$。

（2）签名过程

令待签消息为M，对M做下述运算：

（a）发送者任选一秘密随机数$k \in Z_q$。

（b）计算

$$r \equiv g^k \bmod p,$$
$$s \equiv k + x\,e \bmod p。$$

式中$e = H(r \| M)$。

（c）签字$S = Sig_k(M) = (e, s)$。

（3）验证过程

收信人收到消息M及签字$S = (e, s)$后，执行以下操作：

（a）计算$r' \equiv g^s y^e \bmod p$而后计算$H(r' \| M)$；

（b）验证$Ver(M, r, s) \Leftrightarrow H(r' \| M) = e$。

因为，若$(e \| s)$是M的合法签字，则有

$$g^s y^e \equiv g^{k - x\,e}\, g^{x\,e} \equiv g^k \equiv r \bmod p。$$

Thank you!

(To be continued....)