第五讲：数字签名

一、数字签名的基本概念

二、几种常用的数字签名

三、特殊用途的数字签名

四、数字签名体制的安全性

数字签名基本概念

- 在公钥体制中，用接受者的公钥加密消息得到密文，接受者用自己的私钥解密密文得到消息。加密过程任何人都能完成，解密过程只有接受者能够完成。
- 考虑一种相反的过程，发送者用自己的私钥“加密”消息得到“密文”，然后利用发送者的公钥“解密”密文得到消息。很显然，加密只有发送者能够完成，而解密任何人都可以完成。 所以，任何人都可相信是特定的发送者产生了该消息，这就相当于“签名”，证明一个消息的所属。
- **加密用公钥做变换；签名用私钥做变换。**

数字签名基本概念

数字签名：数字签名在信息安全，包括身份认证、数据完整性、不可否认性以及匿名性等方面有重要的应用，特别是在大型网络安全通信中的密钥分配、认证以及电子商务系统中具有重要作用。数字签名是实现认证的重要工具。

随着计算机通信网的发展，人们希望通过电子设备实现快速、远距离的交易，**数字 (或电子) 签名法**应运而生，并开始用于商业通信系统，如电子邮递、电子转账和办公自动化等系统。随着计算机网络的发展，过去依赖于手书签名的各种业务都可用这种电子数字签名代替，它是实现电子贸易、电子支票、电子货币、电子购物、电子出版及知识产权保护等系统安全的重要保证。

数字签名基本概念

数字签名应满足的要求：

1. 收方能够确认或证实发方的签名，但不能伪造，称为 **R1-条件 (unforgeable)**。
2. 发方发出签名的消息给收方后，就不能再否认他所签发的消息，称为 **S-条件 (non-repudiation)**。
3. 收方对已收到的签名消息不能否认，即有收报认证，称为 **R2-条件**。
4. 第三者可以确认收发双方之间的消息传送，但不能伪造这一过程，称为 **T-条件**。

安全的数字签名实现的条件：发方必须向收方提供**足够的非保密信息**，以便使其能验证消息的签名；但又不能泄露用于产生签名的机密信息，以防止他人伪造签名。

数字签名基本概念

- **与手书签名的区别**：手书签名是**模拟的**，且**因人而异**。数字签名是0和1的数字串，**因消息而异**。
- **与消息认证的的区别**：消息认证使收方能验证消息发送者及所发消息内容是否被窜改过。当收发者之间没有利害冲突时，这对于防止第三者的破坏来说是足够了。但当收者和发者之间有利害冲突时，就无法解决他们之间的纠纷，此时须借助满足前述要求的数字签名技术。
- **数字签名的分类**：可以**对整体消息或压缩消息进行签名**。若按明、密文的对应关系划分，每一种中又可分为两个子类：
  - **确定性** (Deterministic) 数字签名，其明文与签名文一一对应，它对一特定消息的签名不变化；
  - **随机化的** (Randomized) 或**概率式**数字签名。

数字签名基本概念

签名体制定义：一个签名体制可由四元组$(\mathcal{M},\mathcal{S},\mathcal{K},\mathcal{V})$组成，其中$\mathcal{M}$是明文空间，$\mathcal{S}$是签名的集合，$\mathcal{K}$是密钥空间，$\mathcal{V}$是证实函数的值域，由真、伪组成。

- **密钥生成**：$(pk,sk)\leftarrow \mathrm{KGen}(1^k)$
- **签名算法**：对每一 $M\in\mathcal{M}$和$K\in\mathcal{K}$，易于计算对M的签名 $S=Sig_k(M)\in\mathcal{S}$。注意签名密钥是秘密的，只有签名人掌握；
- **验证算法**：$Ver_k(M,S)\in\{\text{真},\text{伪}\}=\{0,1\}$

$$Ver_k(M,S)=\begin{cases}\text{真}, & \text{当}S=Sig(M)\\ \text{伪}, & \text{当}S\neq Sig(M)\ \ (\text{显著的概率})\end{cases}$$

验证算法应当公开：已知M，S易于证实S是否为M的签名，以便于他人进行验证。

数字签名基本概念

- **签名体制的安全性**：从 M 和其签名 S 难于推出私钥 K，或伪造一个（M'，S'）使 M'和 S'可被证实为真。
- **签名的长期生命周期**：消息加密和解密可能是一次性的，它要求在解密之前是安全的；而一个签名的消息可能作为一个法律上的文件，如合同等，很可能在对消息签署多年之后才验证其签名，且可能**需要多次验证**此签名。因此，签名的安全性和防伪造的要求更高些，且要求证实速度比签名速度还要快，特别是联机在线实时验证。

几种常用的数字签名

RSA 签名体制

(1) 参数： 令$n = p_1 p_2$，p_1 和 p_2 是大素数，令$M = C = Z_n$，选 e 并计算出 d 使 $ed \equiv 1 \mod \varphi(n)$，公开 n 和 e，将p_1, p_2和 d 保密。$K = (n,p,q,e,d)$。

(2) 签名： 对消息$M \in Z_n$的签名

$$S = Sig_k(M,k) = M^d \bmod n$$

(3) 验证： 对给定的M和S，可按下式验证：

$$Ver_k(M,S) = 真 \Leftrightarrow M \equiv S^e \bmod n$$

几种常用的数字签名

(4) 安全性：由于只有签名者知道d，由 RSA 体制可知，其他人不能伪造签名，但可易于证实所给任意M，S 对是否消息 M 和相应签名构成的合法对。

(5) 标准化：ISO/IEC 9796 和 ANSI X9.30-199X 已将 RSA 作为建议数字签名标准算法。PKCS #1是一种采用杂凑算法 (如MD2或MD5等) 和 RSA相结合的公钥密码标准。

几种常用的数字签名

ElGamal 签名体制（由 ElGamal 在1985年给出）

(1) 体制参数

p：一个大素数，可使 Z_p 中求解离散对数为困难问题；

g：是 Z_p 中乘群 Z_p^* 的一个生成元或本原元素；

$\mathcal{M}$：消息空间，为 Z_p^*；

$\mathcal{S}$：签名空间，为 $Z_p^* \times Z_{p-1}$；

x：用户秘密钥 $x \in Z_p^*$；$y \equiv g^x \bmod p$

$\mathcal{K} = (p, g, x, y)$：其中 p, g, y 为公钥，x为秘密钥。

(2) 签名： 给定消息M，发送者进行下述计算。

(a) 选择秘密随机数 $k \in Z_p^*$；

(b) 计算：$H(M)$, $r = g^k \bmod p$, $s = (H(M) - xr)k^{-1} \bmod (p-1)$

(c) 将 $Sig_k(M, k) = S = (r||s)$ 作为签名，将M, $(r||s)$ 送给对方。

几种常用的数字签名

(3) 验证： **接收者**先计算$H(M)$，并按下式验证

$$Ver_k(H(M), r, s) = 真 \Leftrightarrow y^r r^s \equiv g^{H(M)} \bmod p$$

这是因为$y^r r^s \equiv g^{rx} g^{sk} \equiv g^{(rx+sk)} \bmod p$,

有$(rx + sk) \equiv H(M) \bmod (p-1)$故有$y^r r^s \equiv g^{H(m)} \bmod p$

在此方案中，对同一消息M，由于随机数k不同而有不同的签名值$S = (r||s)$。它是一种**非确定性**的签名体制。

(4) 安全性： 方案的安全性基于求离散对数的困难性。

(5) 应用： 其修正形式已被美国 NIST 作为数字签名标准 DSS 。此体制专门设计作为签名用。ANSI X9.30-199X 已将 ElGamal 签名体制作为签名标准算法。

几种常用的数字签名

Schnorr 签名体制 （C. Schnorr 于1989年提出）

(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$y \equiv g^x \bmod p$。

$\mathcal{M}, \mathcal{S}$：消息空间 $\mathcal{M} = Z_p^*$，签名空间 $\mathcal{S} = Z_p^* \times Z_p$；

密钥空间：$\mathcal{K} = \{(p, q, g, x, y): y \equiv g^x \bmod p\}$

几种常用的数字签名

(2) 签名过程： 令待签消息为M，对给定M做下述运算：

(a) 签名者任选一秘密随机数$k \in Z_q$。

(b) 计算 $r \equiv g^k \bmod p$

$$s \equiv k + xe \bmod q$$

式中 $e = H(r||M)$

(c) 将消息M及其签名$S = Sig_k(M) = (e||s)$送给收信人。

几种常用的数字签名

(3) 验证过程： 接收者收到消息 M 及签名 $S=(e||s)$ 后

(a) 计算 $r' \equiv g^s y^{-e} \bmod p$， 而后计算$H(r'||M)$ 。

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

因为，若 $(e||s)$ 是 M 的合法签名，则有

$r' = g^s y^{-e} \equiv g^{k+xe} g^{-xe} \equiv g^k \equiv r \bmod p$。

几种常用的数字签名

(4) Schnorr 签名与 ElGamal 签名的不同点：

- 在 ElGamal 体制中，g 为 Z_p 的本原元素；而在 Schnorr 体制中，g 为 Z_p^* 中子集 Z_q^* 的本原元素，它不是 Z_p^* 的本原元素。显然 ElGamal 的安全性要高于 Schnorr 。
- Schnorr 的签名较短，由 $|q|$ 及 $|H(M)|$ 决定。
- 在Schnorr 签名中，$r = g^k \bmod p$ 可以预先计算，k 与 M 无关，因而签名只需一次 $\bmod q$ 乘法及减法。所需计算量少，速度快，适用于灵巧卡采用。

几种常用的数字签名

DSS 签名体制

DSS (Digital Signature Standard) 签名标准是1991年8月由美国NIST 公布、1994年5月19日正式公布，1994年12月1日正式采用的美国联邦信息处理标准。其中采用了第6 章中介绍的SHA，其安全性基于解离散对数困难性，它是在 ElGamal 和Schnorr (1991) 两个方案基础上设计的 **DSS** 中所采用的算法简记为 **DSA** (Digital Signature Algorithm) 。此算法由 D. W. Kravitz 设计。这类签名标准具有较大的兼容性和适用性，已成为网中安全体系的基本构件之一。

几种常用的数字签名

(1) 算法描述:

(a) 全局参数 (p,q,g)，p是$2^{L-1}<p<2^{L}$中的大素数，$512\leq L\leq 1024$，按64 bits递增；q 是 $(p-1)$ 的素因子，且$2^{159}<q<2^{160}$，即字长160bits；$g:=h^{p-1}\bmod p$，且$1<h<(p-1)$，使$h^{(p-1)/q}\bmod p>1$。

(b) 用户秘密钥 x，$0<x<q$。

(c) 用户公钥 $y=g^{x}\bmod p$。

(d) 用户每个消息用的秘密随机数k：$0<k<q$。

几种常用的数字签名

(e) 签名过程：对消息$M \in \mathcal{M} = Z_p^*$，其签名为

$$S = Sig_k(M,k) = (r,s)$$

其中$S \in \mathcal{S} = Z_q \times Z_q$,

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

$$s \equiv [k^{-1}(h(M) + xr)] \bmod q$$

(f) 验证过程：计算

$$w = s^{-1} \bmod q \ ; \ u_1 = [H(M)w] \bmod q \ ;$$

$$u_2 = rw \bmod q \ ; \ v = [(g^{u1}y^{u2}) \bmod p] \bmod q。$$

$$Ver(M,r,s) = 真 \Leftrightarrow v = r$$

几种常用的数字签名

(2) 公众反应：

RSA Data Security Inc (DSI) 想以 RSA算法做为标准，因而对此反应强烈。在标准公布之前就指出采用公用模可能使政府能够进行伪造签名。许多大的软件公司早已得到 RSA 的许可证而反对DSS。主要批评意见有：

① DSA不能用于加密或密钥分配；

② DSA 由 NSA 开发的，算法中可能设有陷门；

③ DSA 比 RSA 慢；

④ RSA 已是一个实际上的标准，而DSS 与现行国际标准不相容；

⑤ DSA 未经公开选择过程，还没有足够的时间进行分析证明；

⑥ DSA 可能侵犯了其它专利 (Schnorr签名算法，Diffie-Hellman的公钥密钥分配算法)；

⑦ 由 512 bit 所限定密钥量太小。现已改为512～1024中可被 64 除尽的即可供使用。

几种常用的数字签名

其它签名体制

- **GOST 签名标准：**为俄国采用的数字签名标准，自1995启用，正式称为GOST R34.10-94。算法与 Schnorr 模式下的 ElGamal 签名及 NIST 的 DSA 很相似。算法中也有一个类似于 SHA 的杂凑函数 $H(x)$，其标准号为 GOST R34. 11-94。
- **ESIGN 签名体制：**日本 NTT 的 T. Okamoto等设计的签名方案。宣称在密钥签名长度相同条件下，至少和 RSA/DSA 一样安全且比它们都快。
- **OSS 签名体制：**Ong，Schnorr 和 Shamir[1984] 提出的一种利用*mod n*下多项式的签名算法,方案基于二次多项式。
- **离散对数签名体制：**ElGamal、DSA、GOST、ESIGN、Okamoto等签名体制都是基于离散对数问题。这些体制都可以归结为一般的称之为**离散对数签名体制**之特例。

特殊用途的数字签名

1. 不可否认签名 (Undeniable Signatures)

1989年由 Chaum 和 Antwerpen 引入，这类签名有一些特殊性质，其中最本质的是在无签名者合作条件下不可能验证签名，从而可以防止复制或散布他所签文件的可能性，这一性质使产权拥有者可以控制产品的散发。这在电子出版系统知识产权保护中将有用场。

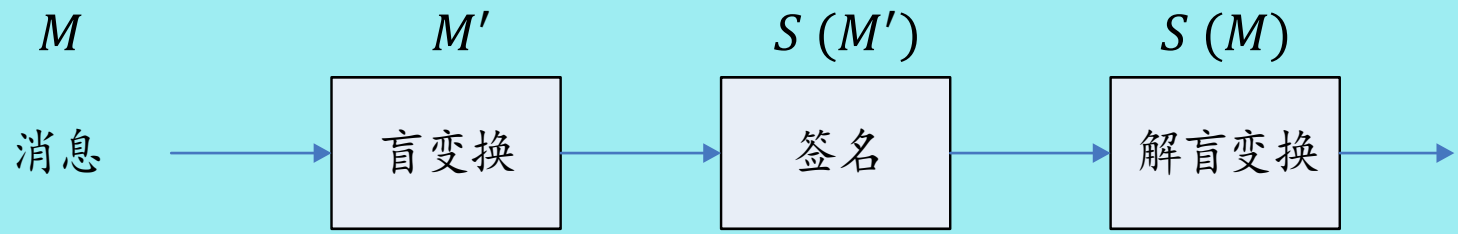
否认协议 (Disavowal Protocol)：在签名者合作下才能验证签名，这会给签名者一种机会，在不利于他时他拒绝合作以达到否认曾签署的文件。为了防止此类事件而引入。

构成签名算法的第三个组成部分，签名者可利用否认协议向法庭或公众证明一个伪造的签名确是假的；如果签名者拒绝参与执行否认协议，就表明签名事实上是真的由他签署的。

特殊用途的数字签名

2. 盲签名 (Blind Signatures)

一般数字签名中，总是要先知道文件内容而后才签署，这正是通常所需要的。但有时需要某人对一个文件签名，但又不让他知道文件内容，称此为**盲签名**(Blind Signature)，它是由Chaum [1983] 最先提出的。选举投票和数字货币协议中将会碰到这类要求。利用盲变换可以实现盲签名。



特殊用途的数字签名

完全盲签名协议:

(a) A取一文件并以一随机值乘之，称此随机值为**盲因子**。

(b) A将此盲文件送给B。

(c) B对盲文件签名。

(d) A以盲因子除之，得到 B 对原文件的签名。

若签名函数和乘法函数是可换的，则上述作法成立。否则要采用其它方法 (而不是乘法) 修改原文件。

安全性讨论： B 可以欺诈吗？是否可以获取有关文件的信息？若盲因子完全随机，则可保证 B 不能由 (b) 中所看到的盲文件得出原文件的信息。即使 B 将 (c) 中所签盲文件复制，他也不能 (对任何人) 证明在此协议中所签的真正文件，而只是知道其签名成立，并可证实其签名。即使他签了100万个文件，也无从得到所签文件的信息。

特殊用途的数字签名

选择分割协议：

每天有许多人进入某国，海关想知道他们是不是贩毒者。他们用概率方法而不是检查每个人来实现这一目的。对入关者抽取1/10进行检查。贩毒者在多数情况下将可逃脱，但有1/10机会被抓获。而法院系统，为了有效惩治贩毒，一旦抓获，其罚金将大于其它9次的获利。要想增加捕获率，必须检查更多人，利用搜查概率值可以成功控制抓获贩毒分子的协议。

选择分割协议在盲签名（电子现金）中有着重要的作用；但是缺点是效率较低。

特殊用途的数字签名

(4) RSA盲签名算法:

D. Chaum 将盲变换看作是信封，盲化文件是对文件加个信封，而去掉盲因子过程是打开信封。文件在信封中时无人可读它，而在盲文件上签名相当于在复写纸信封上签名，从而得到了对真文件 (信封内) 签名。此外他提出第一个实现基于RSA 体制的盲签名的算法。令 B 的公钥为e，秘密钥为d，模为n。

(a) A 要对消息 m 进行盲签名，选$1 < k < m$，作

$$t \equiv mk^e \bmod n \rightarrow B$$

(b) B 对 t 签名，$t^d \equiv (mk^e)^d \bmod n \rightarrow A$

(c) A 计算 $S \equiv \frac{t^d}{k} \bmod n$，得到 $S \equiv m^d \bmod n$

证明： $t^d \equiv (mk^e)^d \equiv m^d k \bmod n \Rightarrow t^d/\mathrm{k} \equiv m^d k/\mathrm{k} \equiv m^d \bmod n$

特殊用途的数字签名

4. 群签名

群体密码学 (Group-Oriented Cryptography) 是1987年由Desmedt 提出。它是研究面向社团或群体中所有成员需要的密码体制。在群体密码中，有一个公用公钥，群体外面的人可以用它向群体发送加密消息，密文收到后要由群体内部成员的子集共同进行解密。本节介绍群体密码学中有关签名的一些内容。

(1) 群签名。

群签名 (Group Signature) 是面向群体密码学中的一个课题，1991 年由 Chaum 和 van Heyst 提出。它有下述几个特点：①只有群中成员能代表群体签名；② 接收到签名的人可用公钥验证群签名，但不可能知道由群体中那个成员所签；③ 发生争议时可由群体中的成员或可信赖机构识别群签名的签名者。

特殊用途的数字签名

例如，这类签名可用于投标中。所有公司应邀参加投标，这些公司组成一个群体，且每个公司都匿名地采用群签名对自己的标书签名。事后当选中了一个满意的标书，就可以识别出签名的公司，而其它标书仍保持匿名。中标者若想反悔已无济于事，因为在没有他参加下仍可以正确识别出他的签名。这类签名还可在其它类似场合使用。

群签名也可以由可信赖中心协助执行，中心掌握各签名人与所签名之间的相关信息，并为签名人匿名签名保密；在有争执时，可以由签名识别出签名人。

Chaum 和 Heyst 曾提出四种群签名方案。有的由可信赖中心协助实现群签名功能，有的采用不可否认并结合否认协议实现。Chaum 所提方案，不仅可由群体中一个成员的子集一起识别签名者，还可允许群体在不改变原有系统各密钥下添加新的成员。群签名目标是对签名者实现无条件匿名保护，而又能防止签名者的抵赖，因此称其为群体内成员的匿名签名更合适些。

特殊用途的数字签名

(2) 面向群体的不可抵赖签名

前面已介绍过不可抵赖签名，这里将介绍在一个群体中由多个人签署文件时能实现不可抵赖特性的签名问题。Desmedt 等所提出的实现方案多依赖于门限公钥体制。

一个面向群体的 (t,n) 不可抵赖签名，其中t 是阈值，n 是群体中成员总数，群体有一公用公钥。签名时也必须有t 人参与才能产生一个合法的签名，而在验证签名时也必须至少有群体内成员合作参与下才能证实签名的合法性。这是一种集体签名共同负责制。

数字签名体制的安全性

- **Forge from scratch:** 给定公钥、签名算法，攻击者伪造出一个合法的消息-签名对。其中，攻击者不能利用已有的消息-签名对，或者和签名者交互作用得到相关消息的签名。
- **Adaptive CMA：** 攻击者在被给定目标消息后，仍然可以询问签名者相关消息 (除了目标消息) 的签名。
- **伪造：** 存在 (existential) 伪造；广义 (universal) 伪造。必须注意的是我们不考虑伪造消息的无意义性。
- **Forge from scratch 的不合理性：** 签名是一个合法的业务。

Textbook RSA 签名的不安全性

- 任何的 (s^e, s) 都可以看作是一个伪造的签名。
- 同态性：由两个合法的消息-签名对(m_1, s_1) (m_2, s_2) 可以得到另一个合法的消息-签名对 $(m_1 m_2, s_1 s_2)$ 。
- 盲签名的性质：$m' = r^e m$, 签名为$m'^d = r m^d$ ，于是可以得到m的签名 $m^d = m'^d / r$。
- 防止存在性伪造的一个途径：消息编码，增加一定的冗余度或使用 hash 函数。
- 如果假定 hash 函数是一个 Random Oracle，那么伪造给定消息的 RSA 签名 (from scratch) 就等价于解 RSA问题。

Textbook ElGamal 签名的不安全性

1. $r < p$

   $u = m'm^{-1} \mod p-1$

   $s' = su \mod p-1$

   $r'\ s.t.\ r' = ru \mod p-1\ and\ r' = r \mod p$

   可以验证 (r', s') 是对消息 m' 的签名。

Textbook ElGamal 签名的不安全性

2. g 必须是群中任意选取的元素

Eve 选取$p-1=bq$，其中b为光滑数，q 为一个大素数。$g=a^t \bmod p$，这里$a=cq, c<b$。首先，在阶为b的群中以g^q为底解y_A^q的离散对数问题，设为z。 很明显，由于b为光滑数，利用 PH 算法可以很快地求出z。而且注意到 $z=x_A \bmod b$，于是可以伪造签名：

$$r=cq<p-1; s=t(m-cqz) \bmod p-1$$

可以验证 (r,s) 是对消息 m 的签名。

Textbook ElGamal 签名的不安全性

3. k 不能重复使用

否则，就有 $k(s_1 - s_2) = m_1 - m_2 \mod p-1$，从而利用$k-1$求解出签名者的私钥。

4. 存在伪造性：

选择任意$u, v < p-1$，满足$\gcd(v, p-1) = 1$,设

$r \leftarrow g^u y_A^v \bmod p$； $s \leftarrow -rv^{-1} \bmod p-1$； $m \leftarrow -ruv^{-1} \bmod p-1$

这时$(m,(r,s))$是合法的消息--签名对；仍然可以使用hash 函数对消息添加一定的冗余度来保证消息的有意义性。

thank you