Section 2.5 Digital Signature

本讲的主要内容

- 数字签名的简介
- 基于**RSA**数字签名
- 基于离散对数数字签名
  - **ElGamal**数字签名
  - **Schnorr**数字签名
  - DSA数字签名
- 基于**ECC**数字签名

引言

手写签名是一种传统的确认方式，如写信、签订协议、支付确认、批复文件等。在数字系统中同样有签名应用的需求，如假定A发送一个认证的信息给B，如果没有签名确认的措施，B可能伪造一个不同的消息，但声称是从A收到的；或者为了某种目的，A也可能否认发送过该消息。很显然，数字系统的特点决定了不可能沿用原先的手写签名方法来实现防伪造或抵赖，这就是提出了如何实现数字签名的问题。

数字签名是电子信息技术发展的产物，是针对电子文档的一种签名确认方法，所要达到的目的是：对数字对象的合法化、真实性进行标记，并提供签名者的承诺。随着信息技术的广泛使用，特别是电子商务、电子政务等快速发展，数字签名的应用需求越来越大。

数字签名的简介

数字签名体制是以电子签名形式存储消息的方法，所签名的消息能够在通信网络中传输。

在当今数字化的信息世界里，数字化文档的认证性、完整性和不可否认性是实现信息化的基本要求，也决定信息化的普及和推广。数字签名是满足上述要求的主要手段之一，也是现代密码学的主要研究内容之一。

数字签名是日常生活中手写签名的电子对应物。

数字签名与手写签名的不同

- 签名：手写签名是被签文件的物理组成部分；数字签名是连接到被签消息上的数字串。
- 传输方式：数字签名和所签名的消息能够在通信网络中传输。手写签名使用传统的安全方式传输。
- 验证：手写签名是通过将它与真实的签名进行比较来验证；而数字签名是利用已经公开的验证算法来验证。
- 数字签名的复制是有效的；而手写签名的复制品是无效的。
- 手书签字是模拟的，且因人而异。数字签字是0和1的数字串，因消息而异。

数字签名的概念

所谓数字签名(Digital Signature)，也称电子签名，是指附加在某一电子文档中的一组特定的符号或代码，它是利用数学方法和密码算法对该电子文档进行关键信息提取并进行加密而形成的，用于标识签发者的身份以及签发者对电子文档的认可，并能被接收者用来验证该电子文档在传输过程中是否被篡改或伪造。

*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字技术。

数字签名的理解

- 数字签名由公钥密码发展而来，作为密码学基本元语，它在网络安全，包括身份认证、数据完整性、不可否认性等方面有着重要应用。
- 数字签名的目的是提供一种手段，使得一个实体把他的身份与某个信息捆绑在一起。一个消息的数字签名实际上是一个数，它仅仅依赖于签名者知道的某个秘密，也依赖于被签名信息的本身。
- 数字签名基于两条基本的假设：一是私钥是安全的，只有其拥有者才能获得；二是产生数字签名的惟一途径是使用私钥。
- *与消息加密不同点：*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；而一个签字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字，且可能需要多次验证此签字。

数字签名的安全要求

- 签名是可以被验证的；

  接受者能够核实签名者对消息的签名。

- 签名是不可伪造的；

  除了签名者，任何人(包括接受者)不能伪造消息的签名。

- 签名是不可重用的；

  同一消息不同时刻其签名是有区别的。

- 签名是不可抵赖的；

  签名者事后不能抵赖对消息的签名，出现争议时，第三方可解决争端；

数字签名算法应满足的要求

- 签名的产生必须使用签名者独有的一些信息以防伪造和否认，同时，要求保证独有的一些信息的安全性。
- 签名的产生应较为容易。
- 签名的识别和验证应较为容易。
- 对已知的数字签名构造一新的消息或对已知的消息构造一假冒的数字签名在计算上都是不可行的。

数字签名方案的组成

数字签名方案是由五元组组成的，

即**{P,S,K,Sign,Ver}**。

P:明文空间；

S:签名空间；

K:密钥空间；

Sign:签名算法；

Ver:验证算法；

数字签名的过程

- 系统初始化过程

  生成数字签名方案用到的所有参数。

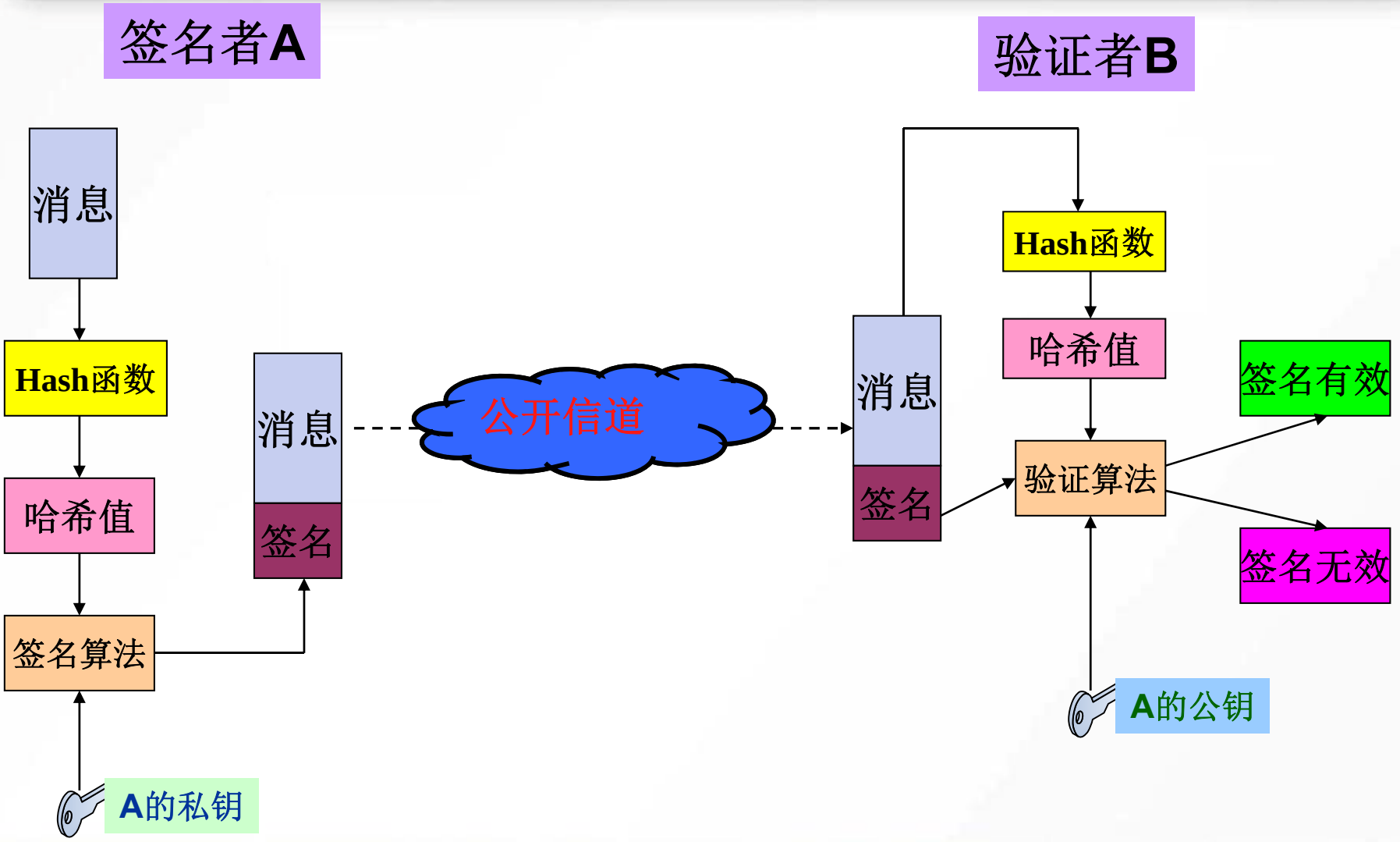
- 签名生成过程

  用户利用给定的算法对消息产生签名s=Sign(m)。

- 签名验证过程

  验证者利用公开的验证方法对给定消息的签名进行验证，得出签名的有效性。Ver(s,m)=0或1。

数字签名描述


签名者A
验证者B
消息
Hash函数
哈希值
签名算法
A的私钥
消息
签名
公开信道
消息
签名
Hash函数
哈希值
验证算法
签名有效
签名无效
A的公钥

基于对称密码体制的数字签名

发方X对发往收方Y的消息签名后，将消息及其签名先发给仲裁者A，A对消息及其签名验证完后，再连同一个表示已通过验证的指令一起发往收方Y。此时由于A的存在，X无法对自己发出的消息予以否认。在这种方式中，仲裁者起着重要的作用并应取得所有用户的信任。以下是实现方法：

① X→A：$M \| E_{XA}[ID_X \| H(M)]$。

② A→Y：$E_{AY}[ID_X \| M \| E_{XA}[ID_X \| H(M)] \| T]$。

其中E是单钥加密算法，K_{XA}和K_{AY}分别是X与A共享的密钥和A与Y共享的密钥，H(M)是M的哈希值，T是时戳，ID_X是X的身份。

基于对称密码体制的数字签名(分析)

- 在①中，X以$E_{XA}[ID_X||H(M)]$作为自己对M的签名，将M及签名发往A。在②中A将从X收到的内容和ID_X、T一起加密后发往Y，其中的T用于向Y表示所发的消息不是旧消息的重放。Y对收到的内容解密后,将解密结果存储起来以备出现争议时使用。
- 如果出现争议，Y可声称自己收到的M的确来自X，并将$E_{AY}[ID_X||M||E_{XA}[ID_X||H(M)]||T]$
- 发给A，由A仲裁，A由KAY解密后，再用K_{XA}对$E_{XA}[ID_X||H(M)]$解密，并对H(M)加以验证，从而验证了X的签名。

基于对称密码体制的数字签名(安全性)

- 由于Y不知K_{XA}，因此不能直接检查X的签名，但Y认为消息来自于A因而是可信的。所以在整个过程中，A必须取得X和Y的高度信任，即这个方案的前提是A是公正的、可信的，且只有A能够检查签名。
- 安全的问题：即仲裁者可和发方共谋以否认发方曾发过的消息，也可和收方共谋以伪造发方的签名。这一问题可采用公钥加密技术的方法得以解决。

RSA数字签名方案(初始化)

1。选取两个大素数p和q，两个数长度接近，1024位

2。计算n=p*q, ψ(n)=(p-1)(q-1)

3. 随机选取整数e(1<e<ψ(n))，满足gcd(e,ψ(n))=1

4.计算d，满足d*e=1（modψ(n)）。

注：n公开， p和q保密。

e为公钥，d为私钥。

RSA数字签名方案(签名和验证)

签名算法

1.利用一个安全的Hash函数h来产生消息摘要h(m)。

2.用签名算法计算签名$s=Sign_k(m)=h(m)^d \bmod n$。

验证算法

1.首先利用一个安全的Hash函数h计算消息摘要h(m)。

2.用检验等式$h(m) \bmod n=s^e \bmod n$ 是否成立，若相等签名有效，否则，签名无效。

RSA数字签名方案(正确性)

由于 $s=h(m)^d \bmod n$

$d*e=1$（$\bmod \psi(n)$）

所以 $s^e \bmod n=h(m)^{ed} \bmod n$

$=h(m)^{k\psi(n)+1} \bmod n$

$= h(m) *h(m)^{k\psi(n)} \bmod n$

$=h(m)*(h(m)^{\psi(n)})^k \bmod n$

$= h(m)$

RSA数字签名方案(举例)

- **初始化：**

  假设A选取p = 13，q = 11，e = 13，则有n = pq = 143，φ(n) = (p-1)(q-1) = 12×10 = 120。求解ed = 13d ≡ 1(mod 120) 得 d = 37。因此A的公钥为 （n = 143,e = 13）；私钥为d = 37。

- **签名过程：**

  假定消息m的Hash值h(m) = 16，则计算m签名$s = h(m)^d \bmod n = 16^{37} \bmod 143 = 3$。

- **验证过程：**

  接受者B收到签名后，计算$s^e \bmod n = 3^{13} \bmod 143 = 16$，h(m) mod n = 16 mod 143 = 16，等式 h(m) mod n = s^e mod n成立，因此，B验证此签名有效。

RSA数字签名方案(安全性)

- 如果不使用Hash函数，则对消息m_1, m_2的签名分别为$s_1 = m_1^d \bmod n$，$s_2 = m_2^d \bmod n$，假设攻击者获得了这两个签名，就可以伪造消息$m_1 * m_2$ 的有效签名$s_1 * s_2$。
- 对于大消息而言，对其Hash值的签名不仅不失数字签名特征，而且大大提高其签名和验证的效率。
- RSA签名方案还存在签名可重用的问题，即对同一消息在不同时刻签名是相同的。

基于离散对数数字签名

- ElGamal数字签名
- Schnorr数字签名
- DSA数字签名

Elgamal签名算法(初始化和签名)

- 初始化：

  首先选择一个大素数p，使在Zp中求解离散对数困难。然后选择一个生成元$g \in Zp^*$，计算$y \equiv g^x \bmod p$，则公开密钥y, g, p，私钥x。

- 签名过程：

  设待签消息为m，签名者选择随机数$k \in_R Zp^*$ ，计算：

  $$r \equiv (g^k \bmod p)$$

  $$s \equiv (h(m) - xr) k^{-1} \bmod (p-1)$$

  则数字签名为(s, r)，其中h()为Hash函数。

Elgamal签名算法(验证)

- 验证过程：

签名接受者在收到消息m和签名值(r, s)后，首先计算h(m)，然后验证等式：

$$y^r r^s \equiv g^{h(m)} \bmod p$$

如等式成立，则数字签名有效；否则签名无效。

Elgamal签名算法(正确性)

因为：

$$r \equiv (g^k \bmod p)$$

$$s \equiv (h(m) - xr) k^{-1} \bmod (p-1)$$

所以：

$$ks \equiv h(m) - xr \bmod (p-1)$$

$$g^{ks} \equiv g^{h(m)-xr} \bmod p$$

$$g^{ks} g^{xr} \equiv g^{h(m)} \bmod p$$

$$y^r r^s \equiv g^{h(m)} \bmod p$$

Elgamal签名算法(举例)

- **初始化：**

  假设A选取素数p = 19，Zp* 的生成元g = 2。选取私钥 x = 15，计算y = g^x mod p = 2^{15}mod 19 ≡12，则A的公钥是（p = 19,g = 2,y = 12）。

- **签名过程：**

  设消息m的Hash值h(m) = 16，则A选取随机数k = 11， 计算r = g^k mod p = 2^{11} mod 19 ≡15，k^{-1} mod (p-1) ≡5。最后计算签名s = [h(m)-xr]k^{-1} mod (p-1) = 5(16-15×15)mod 18 ≡17。得到A对m的签名为(15, 17)。

- **验证过程：**

  接受者B得到签名（15,17）后计算$y^r r^s$ mod p = $12^{15}15^{17}$mod 19 ≡ 5，$g^{h(m)}$ mod p = 2^{16} mod 19 ≡5 。验证等式$y^r r^s$ ≡$g^{h(m)}$ (mod p)相等，因此B接受签名。

Elgamal签名算法(安全性)

- 不能泄露随机数k。
- 不能使用相同的k对两个不同消息进行签名。
- 若不用Hash函数，可能会受到攻击。

例如攻击者可以选取任一整数对(u, v)，gcd(v，p-1) = 1。计算$r = g^u y^v \bmod p$ 和 $s = -rv^{-1} \bmod (p-1)$，则(r, s)就是对消息 $m = su \bmod p$的一个有效签名。这是因为$y^r r^s = y^r g^{us} y^{vs} = y^r g^{su} y^{-r} \equiv g^{su} \bmod p$。可见，使用Hash函数能够有效的提高ElGamal数字签名方案的安全性。

Schnorr签名算法(初始化和签名)

- 初始化：

首先选择一个大素数p，使在Zp中求解离散对数困难。然后选择一个生成元$g \in Z_p^*$，$g^q \equiv 1 \pmod p$，最后选取随机数$1<x<q$，计算$y \equiv g^x \bmod p$，则公钥为(p,q,g,y) ，私钥为x。

- 签名：

用户选择随机数k，对消息进行如下计算得签名值(e, s).

$r \equiv g^k \bmod p$

$e=H(r,m)$

$s \equiv xe+k \bmod q$

Schnorr签名算法(验证)

- 验证：

接受方在收到消息m和签名值(e, s)后，进行以下步骤：

计算$r_1 \equiv g^s y^{-e} \bmod p$

计算$H(r_1, m)$

验证等式：$H(r_1, m)=e$

如果等式成立，接受签名，否则签名无效。

Schnorr签名算法(正确性)

签名体制的正确性证明：

$$r_1 \equiv g^s y^{-e} \bmod p$$

$$\equiv g^s g^{-xe} \bmod p$$

$$\equiv g^{s-xe} \bmod p$$

$$\equiv g^{xe+k-xe} \bmod p$$

$$\equiv g^k \bmod p$$

$$\equiv r$$

$H(r_1, m)=e$

$r \equiv g^k \bmod p$； $e=H(r,m)$; $s \equiv (xe+k) \bmod q$；

Schnorr签名算法(举例)

➢初始化：

假设A选取素数p = 23和q = 11，其中(p-1) / q = 2。选取生成元h = 11 ∈ Zp* ，并计算$g = h^{(p-1)/q} \bmod p = 11^2 \bmod 23 \equiv 6$，则有 $g^q \bmod p = [h^{(p-1)/q} \bmod p]^q \bmod p = h^{(p-1)} \bmod p \equiv 1$ 。既然g≠1，那么g生成Zp* 中一个11阶循环子群。然后选取私钥x = 10 ,计算$y = g^x \bmod p = 6^{10} \bmod 23 \equiv 4$，则A的公钥是（p = 23,q = 11,g = 6,y = 4），私钥为（x = 10）。

Schnorr签名算法(举例)

签名过程：

选取随机数$k = 9$，$1 \le k \le 10$，计算$r = g^k \bmod p = 69 \bmod 23 = 16$。设有$e = h(m||r) = 13$，计算$s = (xe+k) \bmod q = (10 \times 13 + 9) \bmod 11 \equiv 7$。因此消息m的签名为（$e = 13, s = 7$）。

验证过程：

签名接收者B计算$r_1 = g^s y^{-e} \bmod p = 6^7 \times 4^{-13} \bmod 23 = 16 \equiv r$。则有 $h(m \,||\, r_1) = h(m \,||\, r) = e = 13$，因此B接受签名。

DSA数字签名

1994年12月美国国家标准和技术研究所(NIST，National Institute of Standard and Technology)正式颁布了数字签名标准DSS(Digital Signature Standard)，它是在ElGamal和Schorr数字签名方案的基础上设计的。DSS最初建议使用p为512比特的素数，q为160比特的素数，后来在众多的批评下，NIST将DSS的密钥p从原来的512比特增加到介于512比特到1024比特之间。当p选为512比特的素数时，ElGamal签名的长度为1024比特，而DSS中通过160比特的素数q可将签名的长度降低为320比特，这就大大地减少了存储空间和传输带宽。由于DSS具有较大的兼容性和适用性，因此DSS将得到广泛的应用。数字签名标准DSS中的算法常称为DSA（Digital Signature Algorithm）。

DSA数字签名算法（初始化）

首先选择一个160位比特的素数q，接着选择一个长度在512到1024比特之间的素数p，使得p-1能被q整除，最后选择$g = h^{(p-1)/q} \bmod p$，其中h是整数，满足$1< h < p-1$，且$g > 1$。

用户A选择1到q之间的随机数x作为用户的私钥，计算$y = g^x \bmod p$，用户的公钥为（p,q,g,y）。

DSA数字签名算法（签名）

用户选择随机数k，对消息m，计算两个分量r和s产生签名值(r,s)：

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

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

其中h为Hash函数，DSS标准中规定了Hash算法为SHA-1算法。

DSA数字签名算法（验证）

接受方在收到消息**m**和签名值**(r,s)**后，进行以下计算步骤：

$w \equiv s^{-1} \bmod q$；

$u_1 \equiv [h(m)w] \bmod q$；

$u_2 \equiv rw \bmod q$；

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

将**v**和**r**进行比较，若**v = r**，则签名有效；否则，签名无效。

DSA数字签名算法（正确性）

因为：$r \equiv (g^k \bmod p) \bmod q$；

$s \equiv [h(m)+xr]k^{-1} \bmod q$；

$w \equiv s^{-1} \bmod q$ ；

$u_1 \equiv [h(m)w] \bmod q$；

$u_2 \equiv rw \bmod q$ ；

$$v \equiv \left(g^{u_1} y^{u_2} \bmod p \right) \bmod q$$

所以：$v \equiv \left(g^{u_1} y^{u_2} \bmod p \right) \bmod q$ $\equiv [(g^{h(m)w} y^{rw}) \bmod p] \bmod q$

$\equiv [(g^{h(m)w} g^{xrw}) \bmod p] \bmod q \equiv [g^{(h(m)+xr)w} \bmod p] \bmod q \equiv (g^{skw} \bmod p) \bmod q \equiv (g^k \bmod p) \bmod q = r$

DSA数字签名算法（举例）

- 初始化：

  假设A选取素数p = 23，q = 11 ，其中(p-1) / q = 2。选择随机数h = 12$\in_R$Zp* ，计算g = h $^{(p-1)/q}$ mod p = 12^2 mod 23 = 6。既然g≠1，那么g生成Zp* 中唯一的q阶循环子群。接着选择随机数x = 10 满足1≤x≤q-1，并计算y = g^x mod p = 6^{10} mod 23 = 4。则公钥为（p = 23,q = 11,g = 6,y = 4），私钥为（x = 10）。

- 签名过程：

  选取随机数k = 9，计算r = (g^k mod p) mod q = (6^9 mod 23) mod 11= 5。然后计算k^{-1}mod q = 5。假设h(m) = 13，计算s = [h(m)+xr] k^{-1}mod q = 5×(13+10×5) mod 11 = 7。因此消息m的签名为（r = 5,s = 7）。

- 验证过程：

  签名接收者B计算w = s^{-1} mod q = 8，u_1 = [h(m)w]mod q = 13×8 mod 11 = 5， u_2 = rw mod q = 5×8 mod 11 = 7。然后计算 $v=\left(g^{u_1}y^{u_2} \bmod p\right) \bmod q$ = ($6^5 4^7$ mod 23) mod 11 = 5 = r。所以B接受签名。

Elgmal与Schnorr性能对比

签名体制	签名	验证	签名值长度
ElGamal	$T_E+T_H+2T_M$	$3T_E+T_H+T_M$	|p|+|p-1|
Schnorr	$T_E+T_H+T_M$	$2T_E+T_H+T_M$	|q|+|h(m)|
DSA	$T_E+T_H+2T_M$	$2T_E+T_H+3T_M$	2|q|

T_E：幂运算的计算量

T_H：哈希计算的计算量

T_M：乘积运算的计算量

其余的运算与上述三种运算相比可忽略不计

ECC签名体制

- 初始化：

  首先选择一个椭圆曲线，接着构造椭圆群Ep(a,b)，选择一个生成元G。

  选择用户公私钥对：选择1到p-1之间的随机数n_A作为用户的私钥，计算P= n_AG作为用户的公钥。

- 签名：

  用户选择随机数k，对消息进行如下计算得签名值(r,s).

  kG=(x,y),r ≡x mod p

  s≡k-H(m)*n_A mod p

  如果r=0或s=0，则另选随机数k，重新执行上面的过程。

ECC签名体制(验证)

接受方在收到消息m和签名值(r,s)后，进行以下步骤：

计算：$sG+H(m)P=(x_1,y_1)$，$r_1 \equiv x_1 \bmod p$。

验证等式：$r_1 \equiv r \bmod p$。

如果等式成立，接受签名，否则签名无效。

ECC签名体制(正确性)

签名体制的正确性证明：

$sG+H(m)P$

$=(k-H(m)n_A)G+H(m)P$

$=kG-H(m)n_AG+H(m)P$

$=kG-H(m)P+H(m)P$

$=kG$

所以，$r_1 \equiv r \bmod p$。

$R=kG$； $P=n_AG$； $s=k-H(m)*n_A \bmod p$

ECC签名体制(举例)

- **初始化：**

假设椭圆曲线为Z_{23}上的$y^2 = x^3+x+4$。参数分别为：p = 23，G = (0,2) d = 9，P= dG = (4,7)。公钥为(4，7) ，私钥为9。

- **签名过程：**

选取随机数k = 3，假设h(m) = 4 ，则计算(x,y) = kG = 3(0,2)= (11,9)，r = x mod n = 11 mod 23 = 11，$s=k^{-1}(H(m)+n_A \bmod p) = 3^{-1}(4+9 \bmod 23) = 13$。因此对m的签名为(11，13)。

- **验证过程：**

签名接收者B 得到签名后计算：$sG+H(m)P= 13G+ 4P = (11,9)$，$r_1 = x_1 \bmod n = 11 \bmod 29 = 11 = r$。因此B接受签名。

特殊数字签名

在数字签名的实际应用当中，一些特殊的场合往往有特殊的需求，因此，需要在基本数字签名技术的基础上进行扩展，以满足这些特殊的需求。譬如，为保护信息拥有者的隐私，产生了盲签名；为了实现签名者的匿名性和可跟踪性，产生群签名；为了实现签名权的安全传递，产生代理签名；为了实现多人对同一个消息的签名，产生了多重签名等等。正是这些实际应用的特殊需求，使得各种各样的特殊数字签名的研究一直是数字签名研究领域非常活跃的部分并产生很多分支。

本讲主要内容

- 数字签名的简介
- 基于**RSA**数字签名
- 基于离散对数数字签名
  - **ElGamal**数字签名
  - **Schnorr**数字签名
  - DSA数字签名
- 基于**ECC**数字签名

谢　谢！