第六章、数字签名及其在区块链中的应用

何德彪

武汉大学

国家网络安全学院

目　录

目　录

6.1 数字签名的概念

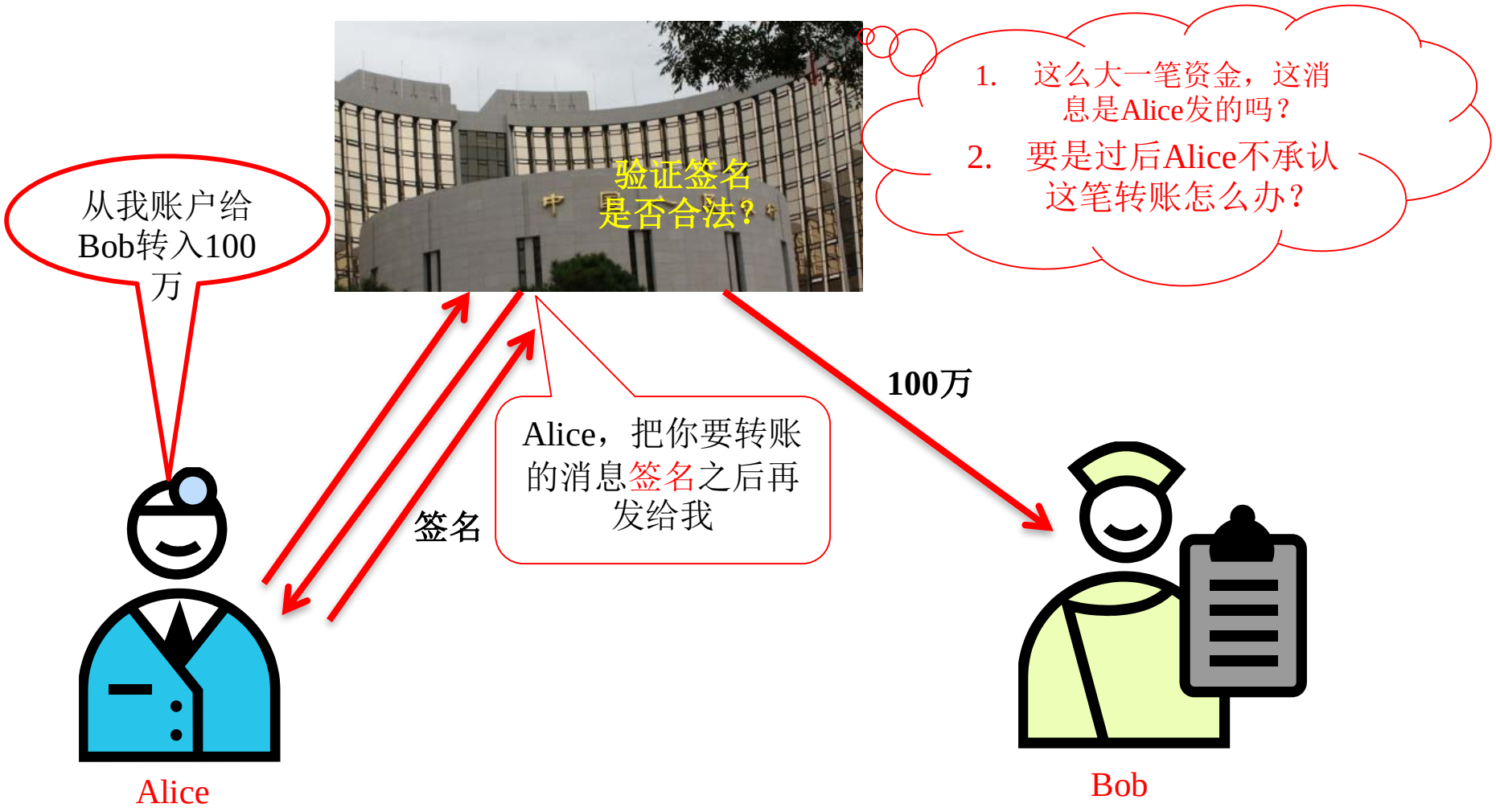
1. 数字签名的定义

- **手写签名**：传统的确认方式，如书信、签约、支付、批复等
- 在网络时代，人们通过网络支付费用、买卖股票，为了保证网上商务活动的安全，需要一个很重要的安全机制——**数字签名**
- 数字签名在信息安全，包括鉴别、数据完整性、抗抵赖性等方面，特别是在大型网络安全通信中的密钥分配、鉴别及电子商务系统中，具有重要作用.

问题的提出

图6.1. 传统合同示意图

6.1 数字签名的概念


从我账户给Bob转入100万
验证签名是否合法？
1. 这么大一笔资金，这消息是Alice发的吗？
2. 要是过后Alice不承认这笔转账怎么办？
Alice，把你要转账的消息签名之后再发给我
签名
100万
Alice
Bob

6.1 数字签名的概念

1. 数字签名的定义与性质

➢ **传统签名及其应用**： 手写签名与印章.

- ✓ 对其所签署的文件进行确认；
- ✓ 如果日后签署文件的双方针对文件的内容发生争执，根据签署文件时留下的签名，第三方可以对签名进行检查以便对争执进行调解.

➢ **数字签名**是一种作用类似于传统的手书签名或印章的电子标记，它**可以达到与手写签名类似的作用**，即使用数字签名.

定义：在ISO7498-2标准中，**数字签名的定义为**“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，防止被人伪造”.

6.1 数字签名的概念

2. 数字签名的性质

数字签名的主要作用：将消息和拥有消息的实体可信地联系起来. 需要具备如下性质：

- **签名是不可伪造的**：除了合法的签名者之外，任何其他人伪造其签名是困 难的.
- **签名是不可抵赖的**：签名者事后不能否认自己的签名.
- **签名是可信的**：任何人都可以验证签名的有效性.
- **签名是不可复制的**：对一个消息的签名不能通过复制变为另一个消息的签名. 如果对一个消息的签名是从别处复制得到的，则任何人都可以发现消息与签名之间的不一致性，从而可以拒绝签名的消息.
- **签名的消息是不可篡改的**：经签名的消息不能被篡改. 一旦签名的消息被篡改，则任何人都可以发现消息与签名之间的不一致性.

6.1 数字签名的定义

3. 数字签名与传统签名的比较

(1). **数字签名**：签名与消息是分开的，需要一种方法将签名与消息绑定在一起.

手写签名：签名认为是被签名消息的一部分.

(2). **数字签名**：在签名验证的方法上，数字签名利用一种公开的方法对签名进行验证，任何人都可以对签名进行验证.

手写签名：验证是由经验丰富的消息接收者通过同以前的签名相比较而进行的.

(3). **数字签名**：在数字世界中很容易被复制(拷贝)，复制后是一模一样的.

手写签名：在物理世界中不易被复制，复制后的签名的容易与原文件区别.

6.1 数字签名的定义

4. 数字签名方案的组成

一个数字签名方案一般包括四个过程：

- **系统初始化过程**：产生数字签名方案中的所有系统参数；
- **密钥对生成过程**：产生用户的私钥(签名密钥)和公钥(验证密钥)；
- **签名过程**：用户利用给定的签名算法对消息签名，签名过程可以公开也可以不公开，但一定包含仅签名者才拥有的私钥；
- **验证过程**：验证者利用公开的系统参数、验证方法和签名者的公钥对给定消息的签名进行验证.

6.1 数字签名的定义

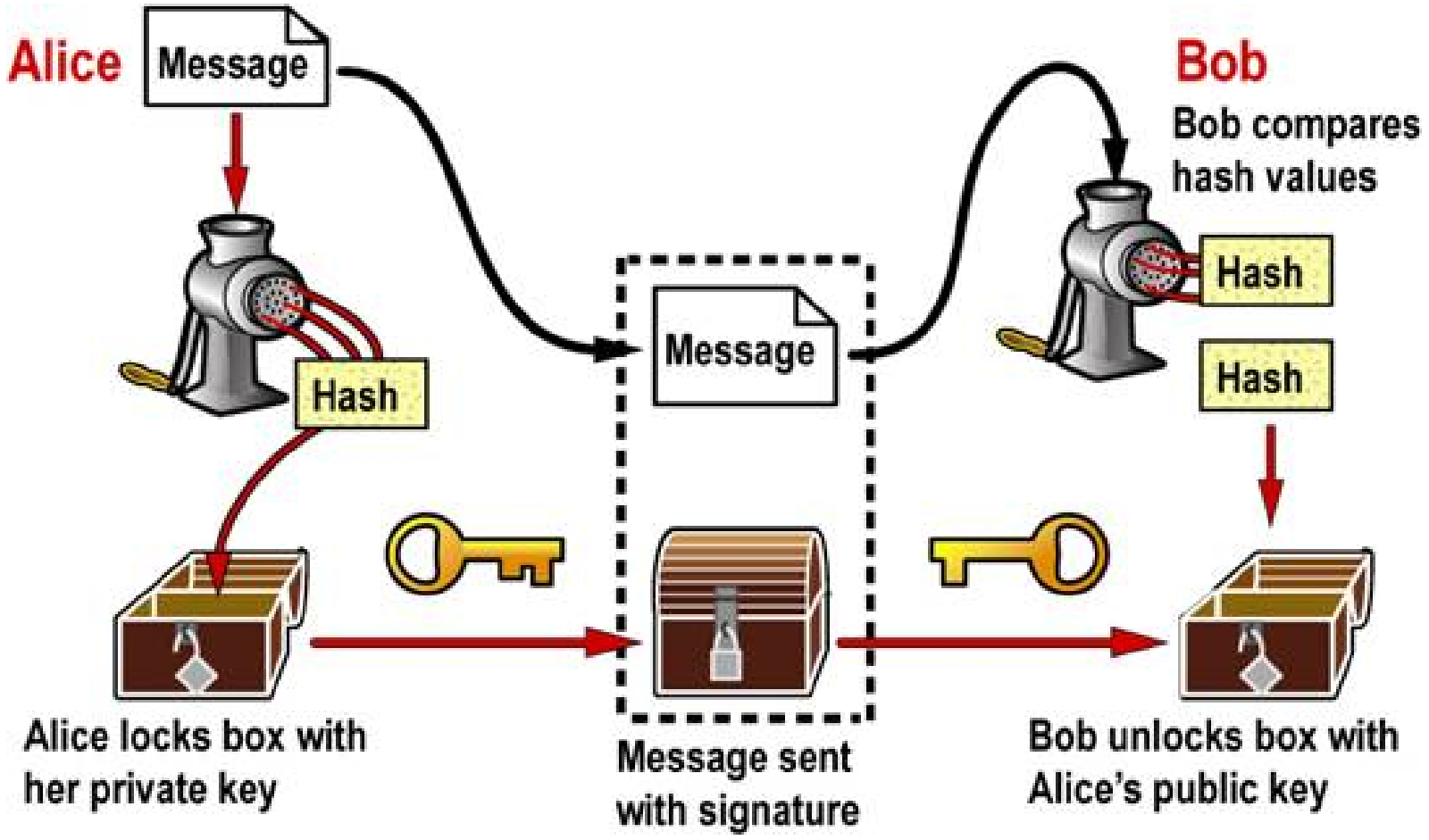


图6.2. 数字签名工作流程示意图

6.1 数字签名的定义

5. 数字签名的设计要求

一个数字签名方案需要满足以下要求：

- 签名必须依赖于被签名消息的比特模式 (签名与消息应该是一个不可分割的整体)；
- 签名必须使用某些对发送者是唯一的信息，以防止伪造和抵赖；
- 数字签名的产生、识别和验证应该相对容易；
- 伪造一个数字签名在计算上是不可行的；(无论是通过对已有的数字签名来构造新报文，还是对给定的报文伪造一个数字签名)

6.1 数字签名的定义

6. 数字签名的攻击类型

敌手(攻击者)的目标是伪造签名，即冒充某个实体产生数字签名，并被验证者接受. 下面提供一套判定准则，用于区分签名方案是在何种意义上被攻克的.

1. 完全攻克。敌手或者能计算签名者的私钥，或者找到一个有效的签名算法在功能上与真正的签名算法等价。

2. 选择性伪造。敌手能够对一个特殊的消息或者预先选定的一类消息构造出正确的签名。签名的构造不直接涉及合法签名者。

3. 存在性伪造。敌手能够伪造至少一个消息的签名。敌手对被伪造签名所对应的消息只有很小的或者根本没有控制能力，而且合法用户可能被卷入该欺骗过程。

6.1 数字签名的定义

对数字签名算法的攻击有两种类型：

1. 惟密钥攻击。此类攻击中，敌手仅知道签名者的公钥。
2. 消息攻击。敌手能够查看与已知消息或者选定消息相应的签名。消息攻击可再细分为三个子类：

   (a) 已知消息攻击。敌手拥有多个消息的签名，这些消息是已知的但不由敌手选取。

   (b) 选择消息攻击。在试图攻克签名方案之前，敌手从一个可选的消息列表中获得有效签名。这种攻击是非自适应的，意思是在获知任何签名之前消息已经被选定。对签名方案的选择消息攻击类似于对公钥加密方案的选择密文攻击 。

   (c) 自适应选择消息攻击。敌手可以将签名者作为一个谕示使用；敌手可以要求获得某些签名，它们所对应的消息依赖于签名者的公钥，他也可以要求获得某些签名，它们所对应的消息依赖于以前得到的签名或消息。

目　录

6.2 经典数字签名算法

数字签名一般利用公钥密码技术来实现，其中私钥用来签名，公钥用来验证签名.比较典型的数字签名方案有：

- RSA签名算法(R. L. Rivest, A. Shamir, and L. M. Adleman, 1978)
- ElGamal 签名算法(T. ElGamal, 1985)
- Schnorr签名算法(C. P. Schnorr, 1989)
- DSS签名算法(NIST, 1991)

目　录

6.2 经典数字签名算法

6.2.1 RSA数字签名算法

基于RSA公钥体制的签名方案通常称为**RSA数字签名方案**。 RSA签名体制的基本算法如下：

1. 密钥的生成（与加密系统一样）：

   公钥$Pk = \{e, n\}$;私钥$Sk = \{d\}$

2. 签名过程(d, n)：

   用户A对消息$M \in Z_n$进行签名，计算

   $$S = Sig(H(M)) = H(M)^d \bmod n;$$

   并将S附在消息M后

3. 验证过程(e, n)：

   给定(M, S)，$Ver(M, S)$为真$\Leftrightarrow$

   $$H(M) = S^e \ (mod\ n)$$成立

6.2 经典数字签名算法

6.2.1 RSA数字签名算法

为什么对消息的Hash函数值签名而不是直接对消息签名？

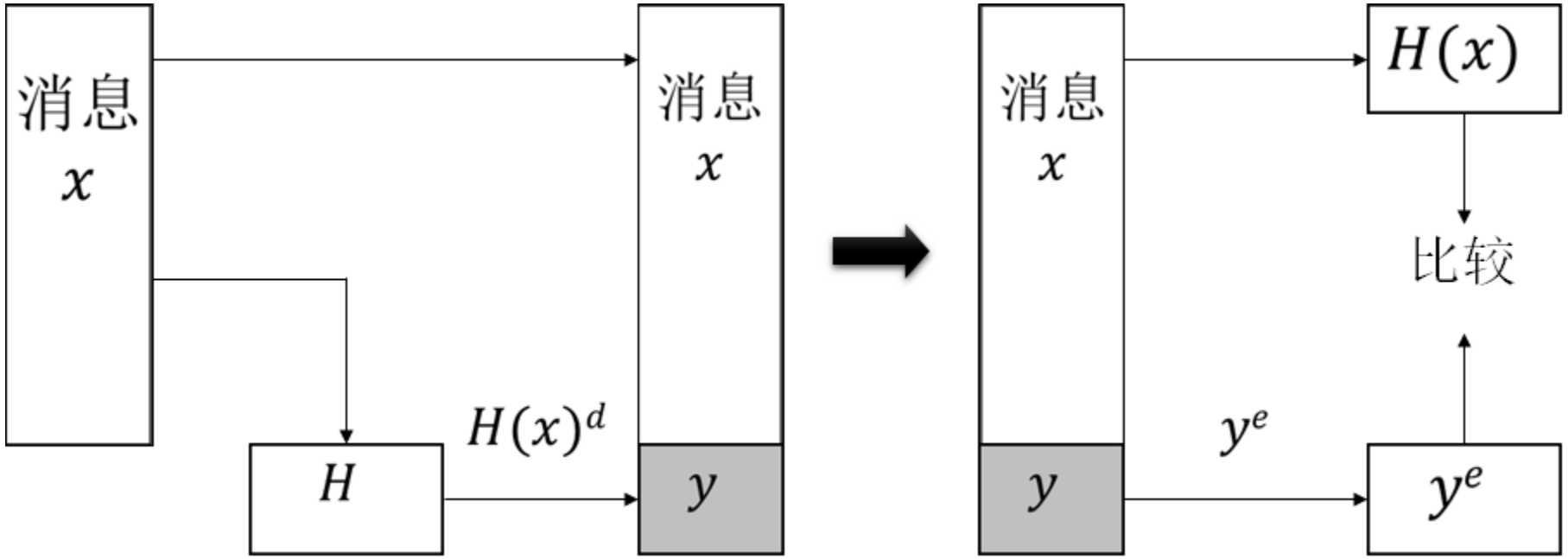


图6.3. RSA数字签名算法工作流程

6.2 经典数字签名算法

6.2.1 RSA数字签名算法

假设RSA直接对消息进行签名

一般攻击：攻击者任选一个数据Y，用A的公钥计算 $X=Y^e \bmod n$，于是便可以用

***Y*伪造*A*对消息*X*的签名**，因 为

$$Y = X^d \bmod n$$

- 实际意义不大：伪造的消息X具有实际意义的概率很小
- Hash函数可以抵御这种攻击

6.2 经典数字签名算法

6.2.1 RSA数字签名算法

利用已有签名进行攻击：如果消息M_1、M_2的签名分别是S_1和S_2，则任何知道M_1，S_1，M_2，S_2的人可以伪造对消息M_1M_2的签名S_1S_2，因为

$$Sig(M_1M_2) = Sig(M_1)Sig(M_2)$$

- 用户不要轻易对其他用户提供的随机数据进行签名
- 更有效的方法：对数据的Hash值签名

6.2 经典数字签名算法

6.2.1 RSA数字签名算法

利用签名获得明文：

- 攻击者截获密文$C=M^e \bmod n$，选择随机数r，并计算

$$x=r^e \bmod n;\quad y=x\cdot C \bmod n;$$

然后攻击者设法让发送者对y签名，获得：

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

攻击者计算：

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

- 用户不要轻易对其他用户提供的随机数据进 行签名
- 更有效的方法：对数据的Hash值签名

6.2 经典数字签名算法

6.2.1 RSA数字签名算法

➢ ***h(M)*的重要性**

***H(M)*的另一个作用—加快签名速度**

- 对整个消息签名，由于公钥体制速度比较慢，当消息比较长时，签名与验证过程都会相当慢
- 对消息的Hash值签名，则无论消息多长，签名都只与 Hash值的长度有关

RSA算法比较慢，用私钥进行签名和公钥进行验证. 因上述RSA签名算法没有加入随机数，当出现重复性的原始资料，**攻击者会通过相同签名信息而猜测出原文.**

怎么办？

6.2 经典数字签名算法

6.2.1 RSA数字签名算法

- PSS (Probabilistic Signature Scheme)私钥签名流程的一种填充模式.
- 目前主流的RSA签名包括RSA-PSS和RSA-PKCS#1 V1.5.
- 后者相对应PKCS (Public Key Cryptography Standards)是一种能够自我恢复签名，而PSS无法从签名中恢复原来的签名.
- OpenSSL-1.1.x以后默认使用更安全的PSS的RSA签名模式.

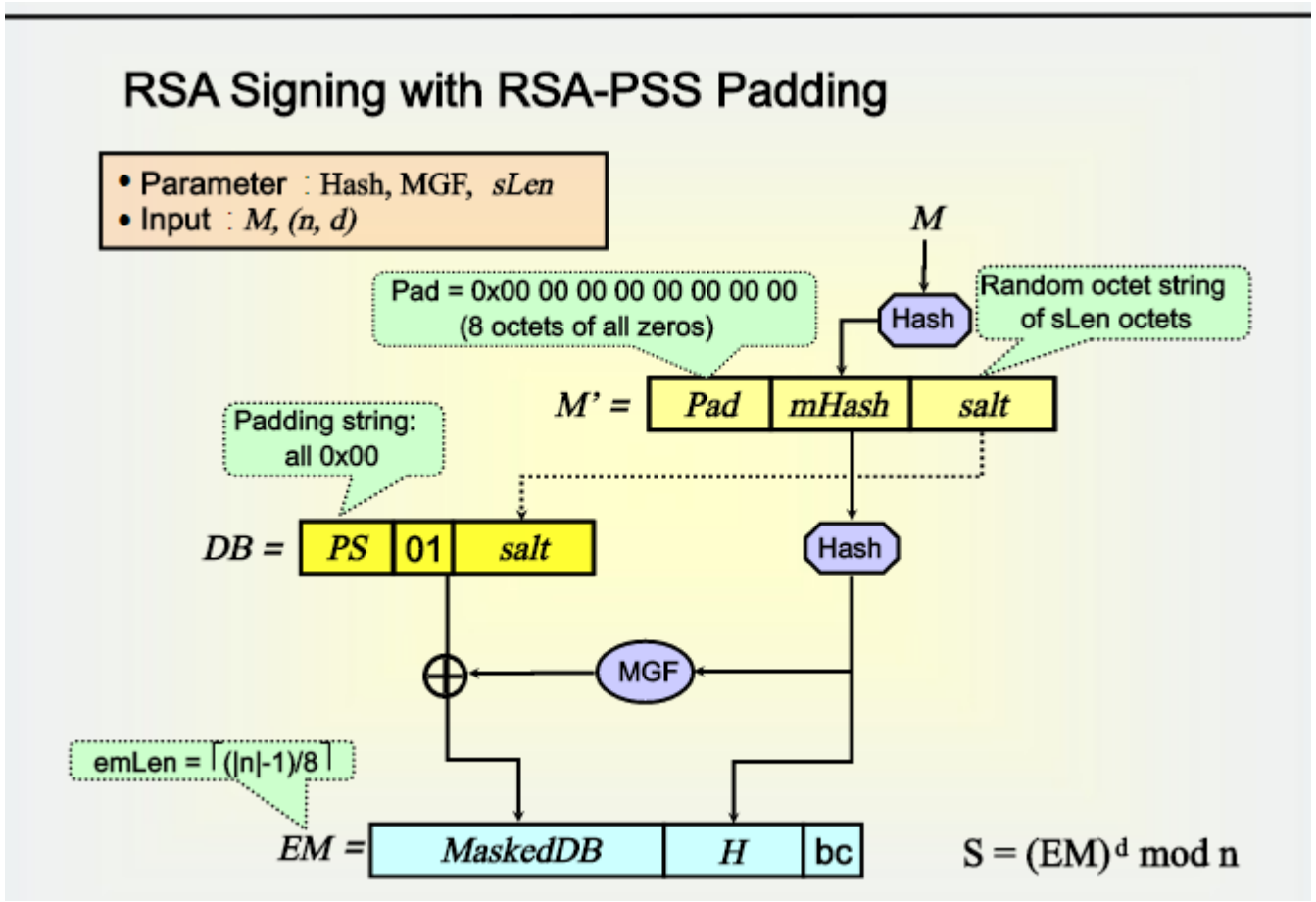


图6.4. PSS算法的编码操作过程

6.2 经典数字签名算法

6.2.1 RSA数字签名算法

RSA-PSS数字签名算法

➢ 密钥生成

① 生成一个模数n，一个公钥e和一个私钥d.

② 假设安全参数为k(n是k比特的数)，我们定义两个整数k_0和k_1并且满足：$k_0+k_1\leq k-1$

③ 然后我们定义两个哈希函数：一个扩展数据$G:\{0,1\}^{k_1}\rightarrow\{0,1\}^{k-k_1-1}$，一个压缩数据$H:\{0,1\}^{*}\rightarrow\{0,1\}^{k_1}$.

6.2 经典数字签名算法

6.2.1 RSA数字签名算法

➢ 签名算法

为了对一个消息m进行签名，签名者执行以下步骤：

①生成一个随机数 $r \in \{0,1\}^{k_0}$

②计算$w=H(m\|r)$

③设置$y=0\|w\|(G_1(w)\oplus r)\|G_2(w)$

④计算$s=y^d \bmod n$

⑤消息m的签名为 s

➢ 验证算法

为了对一个消息m进行的签名s进行验证，验证者者执行以下步骤：

①计算$y=s^e \bmod n$

②将y分解成: $b\|w\|\ \alpha\ \|\ \gamma$，其中$b$的长度为1比特，$w$的长度为$k_1$比特，$\alpha$ 的长度为k_0比特,γ的长度为$k-k_0-k_1-1$比特

③计算$r=\alpha\oplus G_1(w)$

④当且仅当下列等式成立时，接受该签名：

$b=0$, $G_2(w)=\gamma$，且$w=H(m\|r)$

目　录

6.2 经典数字签名算法

6.2.2 ElGamal数字签名

假设p是一个大素数，g是$GF(p)$的生成元.Alice的公钥为$y = g^x \bmod p$, g，p私钥为x.

签名算法：

(1) Alice用H将消息m进行处理，得$h=H(m)$.

(2) Alice选择秘密随机数k，满足

$$0<k< p-1，且(k, p-1)=1，$$

计算

$$r=g^k \ (\mathbf{mod}\ p);$$

$$s=(h - x \cdot r) \cdot k^{-1}(\mathbf{mod}\ (p -1)).$$

(3) Alice将(m，r，s)发送给Bob.

6.2 经典数字签名算法

6.2.2 ElGamal数字签名

验证签名过程:接收方收到M与其签名(r,s)后:

① 计算消息M的Hash值$H(M)$;

② 验证公式

$$y^r r^s = g^{H(M)} \bmod p$$

成立则确认(r,s)为有效签名，否则认为签名是伪造的

6.2 经典数字签名算法

6.2.2 ElGamal数字签名

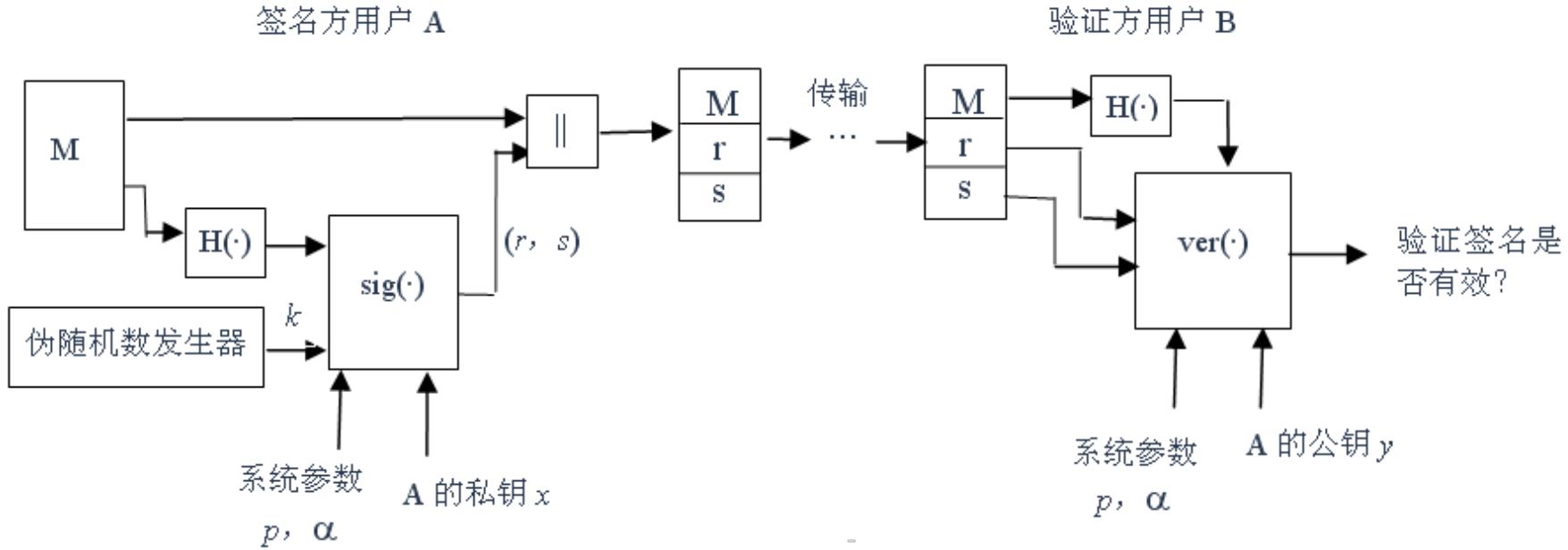


图6.5. PSS算法的编码操作过程

6.2 经典数字签名算法

6.2.2 ElGamal数字签名

选择 $p = 11, g = 2$，Alice选 $x = 8$为私钥，计算$y = 2^8 \bmod 11 = 3$，则公钥为: $y = 3, g = 2, p = 11$

① Alice要对M进行签名，假定$h=H(M)=5$

② 选$k = 9$ ($\gcd(9, 10) = 1$, 9^{-1} (**mod** 10) =9)

③ $r = 2^9$ **mod** 11 = 6，$s=(h - x \cdot r) \cdot k^{-1} =(5-3*6)*9^{-1}$(**mod** 10) =3

很容易检查等式$y^r * r^s \bmod p = g^h \bmod p$是成立的.

6.2 经典数字签名算法

6.2.2 ElGamal数字签名

上述方案的安全性是基于如下**离散对数问题**的：已知大素数p、GF(p)的生成元g和非零元素$y \in GF(p)$，求解唯一的整数k, $0 \le k \le p-2$，使得$y \equiv g^k \pmod{p}$，k称为y对g的离散对数.

在1996年的欧洲密码学会(Proceedings of EUROCRYPT 96)上，David Pointcheval和Jacques Stern给出一个ElGamal签名的变体，并基于所谓分叉技术证明了在**随机预言模型**下所给方案是安全的(在自适应选择消息攻击下能抗击存在性伪造).

目　录

6.2 经典数字签名算法

6.2.3 Schnorr数字签名

Schnorr签名方案是一个短签名方案，它是ElGamal签名方案的变形，其安全性是基于离散对数困难性和哈希函数的单向性的.

假设p和 q是大素数，是q能被p-1整除，q是大于等于160 bit的整数，p是大于等于512 bit的整数，保证GF(p)中求解离散对数困难；g是GF(p)中元素，且$g^q \equiv 1 \bmod p$.

➢ **密钥生成：**

① Alice选择随机数x为私钥，其中$1<x<q$；

② Alice计算公钥$y \equiv g^x \pmod p$；

6.2 经典数字签名算法

6.2.3 Schnorr数字签名

- **签名算法**

① Alice首先随机数k，这里$1<k<q$;

② Alice计算$e = h(M, g^k \bmod p)$

③ Alice计算$s = k - x \cdot e \ (\bmod q)$

④ Alice输出签名(e, s)

- **验证算法**

① Bob计算$g^k \bmod p = g^s \cdot y^e \bmod p$.

② Bob验证$e = h(M, g^k \bmod p)$是否成立，如果成立则输出”Accept”，否则输出" Reject"

6.2 经典数字签名算法

6.2.3 Schnorr数字签名

Schnorr签名与ElGamal签名的不同点：

- **安全性比较：**在ElGamal体制中， g为域$GF(p)$的本原元素；而在Schnorr体制中， g只是域$GF(p)$的阶为q的元素，而非本原元素.因此，虽然两者都是基于离散对数的困难性，然而**ElGamal的离散对数阶为p-1， Schnorr的离散对数阶为q<p-1**. 从这个角度上说， ElGamal的安全性似乎高于Schnorr .
- **签名长度比较：** Schnorr比ElGamal签名长度短.
  - ✓ ElGamal：(m，r，s)，其中r的长度为$|p|$， s的长度为$|p-1|$.
  - ✓ Schnorr： (m，e，s)，其中e的长度为$|q|$， s的长度为$|q|$.

目　录

6.2 经典数字签名算法

6.2.4 DSA算法

- 1991年，美国政府颁布了**数字签名标准**(Digital Signature Standard, DSS)，也称为数字签名算法(Digital Signature Algorithm, DSA)
  - ✓ 和DES一样，DSS也引起了激烈的争论.反对者认为：**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**
  - ✓ 随后，美国政府对其做了一些改进
- 目前DSS的应用已经十分广泛，并被一些国际标准化组织采纳为国际标准
- 2000年，美国政府将RSA和椭圆曲线密码引入到数字签名标准中，进一步丰富了DSA算法

6.2 经典数字签名算法

6.2.4 DSA算法

DSA的主要参数：

- **全局公开密钥分量，可以为用户公用**
  - ✓ p：素数，要求$2^{L-1}<p<2^{L}$,$512<=L<1024$，且L为64的倍数
  - ✓ q: $(p-1)$的素因子，$2^{159}<q<2^{160}$，即比特长度为160位
  - ✓ $g:=h^{(p-1)/q} \bmod p$.其中h是一整数，$1<h<(p-1)$且$h^{(p-1)/q} \bmod p>1$
- **用户私有密钥**
  - ✓ x：随机或伪随机整数，要求$0<x<q$
- **用户公开密钥**
  - ✓ y：$=g^{x} \bmod p$
- **随机数*k***
  - ✓ 随机或伪随机整数，要求$0<k<q$

6.2 经典数字签名算法

6.2.4 DSA算法

DSA签名过程：

① 用户随机选取k

② 计算$e=h(M)$;

③ 计算$r=(g^k \bmod p) \bmod q$

④ 计算$s=k^{-1}(e+x \cdot r) \bmod q$

⑤ 输出(r, s)，即为消息M的数字签名.

DSA验证过程：

① 接收者收到M, r, s后，首先验证$0<r<q$, $0<s<q$；

② 计算$e=h(M)$;

③ 计算$w=(s)^{-1} \bmod q$

④ 计算$u_1=e \cdot w \bmod q$

⑤ 计算$u_2=r \cdot w \bmod q$

⑥ 计算$v=[(g^{u1} \cdot y^{u2}) \bmod p] \bmod q$

⑦ 如果$v=r$，则确认签名正确，否则拒绝.

6.2 经典数字签名算法

6.2.4 DSA算法

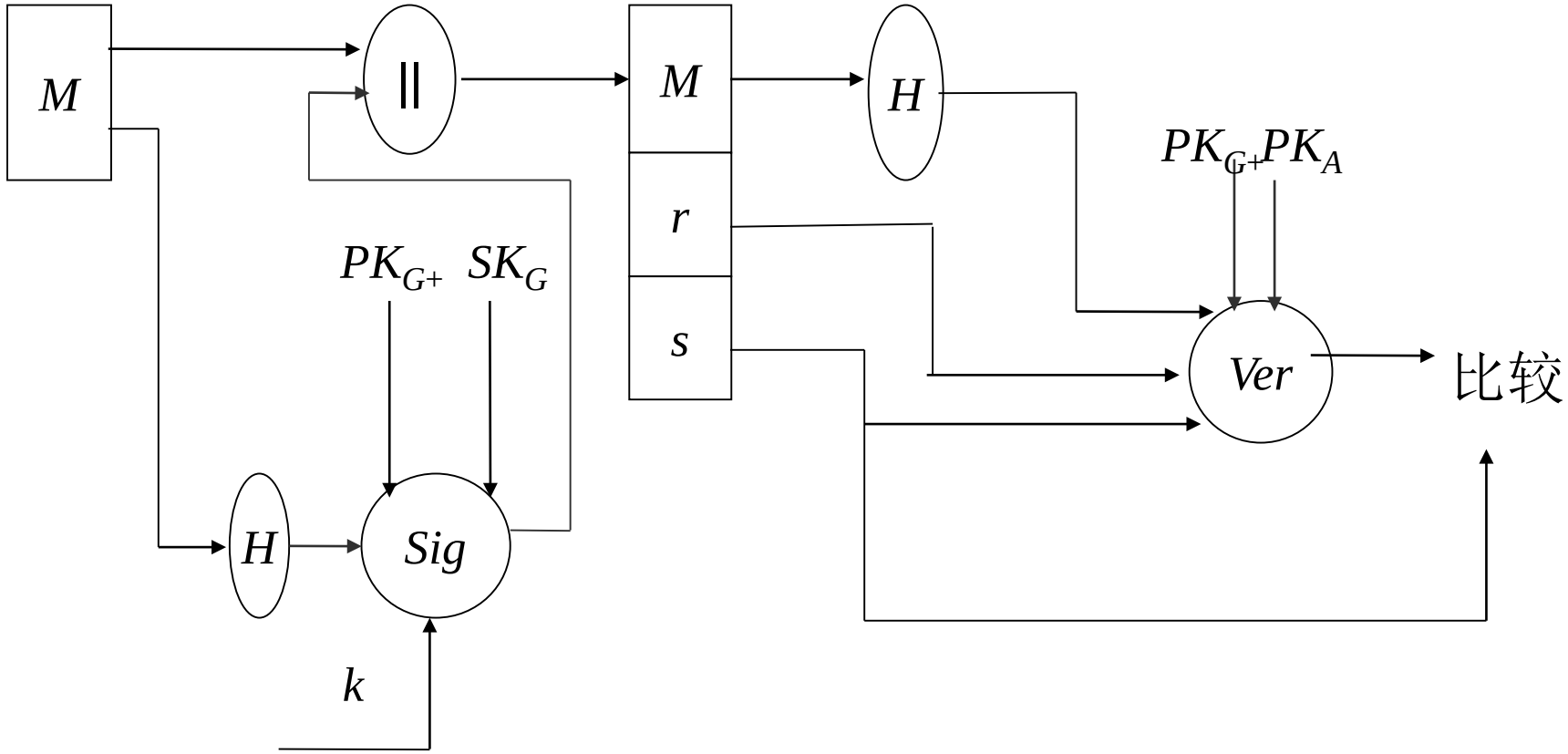


图6.6. DSA算法的工作过程

6.2 经典数字签名算法

6.2.4 DSA算法

例：取q=101，p=78*101+1=7879，h=3, 所以$g=3^{78}(\text{mod}7879) = 170$

取x=75(私钥)，那么$y=g^{x}$(**mod** 7879)=4567 (公钥)

假设Alice想签名一个消息m，**假设**$h(m)$= 1234

签名过程:

选择随机数k=50，计算k^{-1}(**mod** 101)=99

$r=(170^{50}$ **mod** 7879) **mod** 101 =2518 (**mod** 101)=94

s=(1234 + 75*94)*99(**mod** 101)=97

签名为(94，97)

6.2 经典数字签名算法

6.2.4 DSA算法

验证过程：

$w=97^{-1}(\mathbf{mod}\ 101)=25$

$u_1=1234*25(\mathbf{mod}\ 101)=45$

$u_2=94*25(\mathbf{mod}\ 101)=27$

$v=[170^{45}*4567^{27}\ (\mathbf{mod}\ 7879)](\mathbf{mod}\ 101)$

$=2518(\mathbf{mod}\ 101)=94$

$v=r$，该签名是有效的

目　录

6.3 基于椭圆曲线的数字签名算法

6.3.1 椭圆曲线简介

➢ 椭圆曲线的定义

1985年，Koblitz 和Miller独立地提出了椭圆曲线公钥密码体制(ECC)，安全性基于椭圆曲线群上的离散对数问题的难解性，该问题目前最好的解法是指数级时间的算法.

一般认为，RSA和DH密钥交换协议需用1024比特以上的模数才安全，但对ECC，只要160比特的模数就可达到同样级别的安全性.

破解所需时间 (MIPS年)	RSA 密钥大小	ECC 密钥大小	RSA/ECC 密钥大小之比
10^4	512	106	5：1
10^8	768	132	6：1
10^{12}	1024	160	7：1
10^{20}	2048	210	10：1

6.3 基于椭圆曲线的数字签名算法

6.3.1 椭圆曲线简介

➢ **椭圆曲线的定义**

椭圆曲线指的是由Weierstrass方程

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

所确定的曲线.

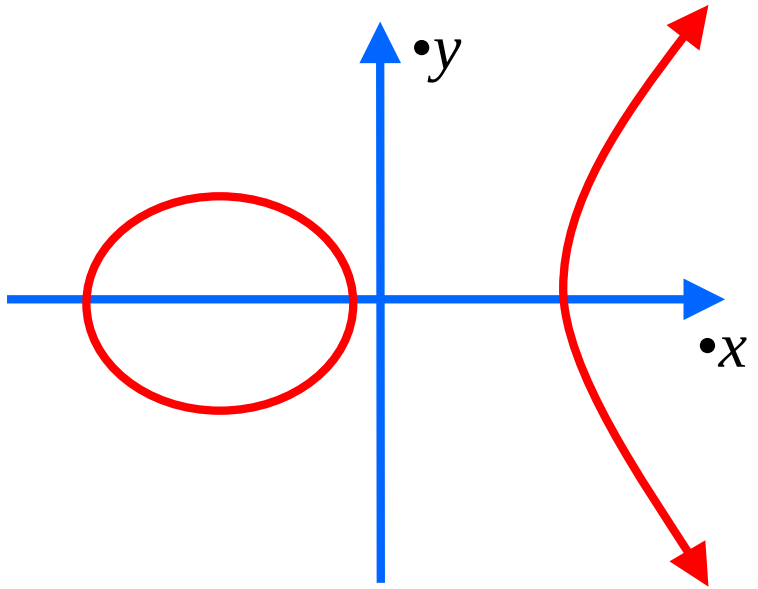


图6.6. 椭圆曲线图形

6.3 基于椭圆曲线的数字签名算法

6.3.1 椭圆曲线简介

➢ **椭圆曲线的定义**

有限域 F_p 上的椭圆曲线 是由满足F_p 上的方程

$$y^2 = x^3 + ax + b$$

的所有点和无穷远点 O 构成的集合

$$E(F) = \{O\} \cup \{(x, y) \in F \times F : y^2 = x^3 + ax + b\}$$

有时也记作E .

6.3 基于椭圆曲线的数字签名算法

6.3.1 椭圆曲线简介

➢ 椭圆曲线上的加法运算

设P，Q是E上的任意两点，连接P，Q交E于R',则称R'关于x轴的对称点R为P与Q的和，记为：

$$P + Q = R$$

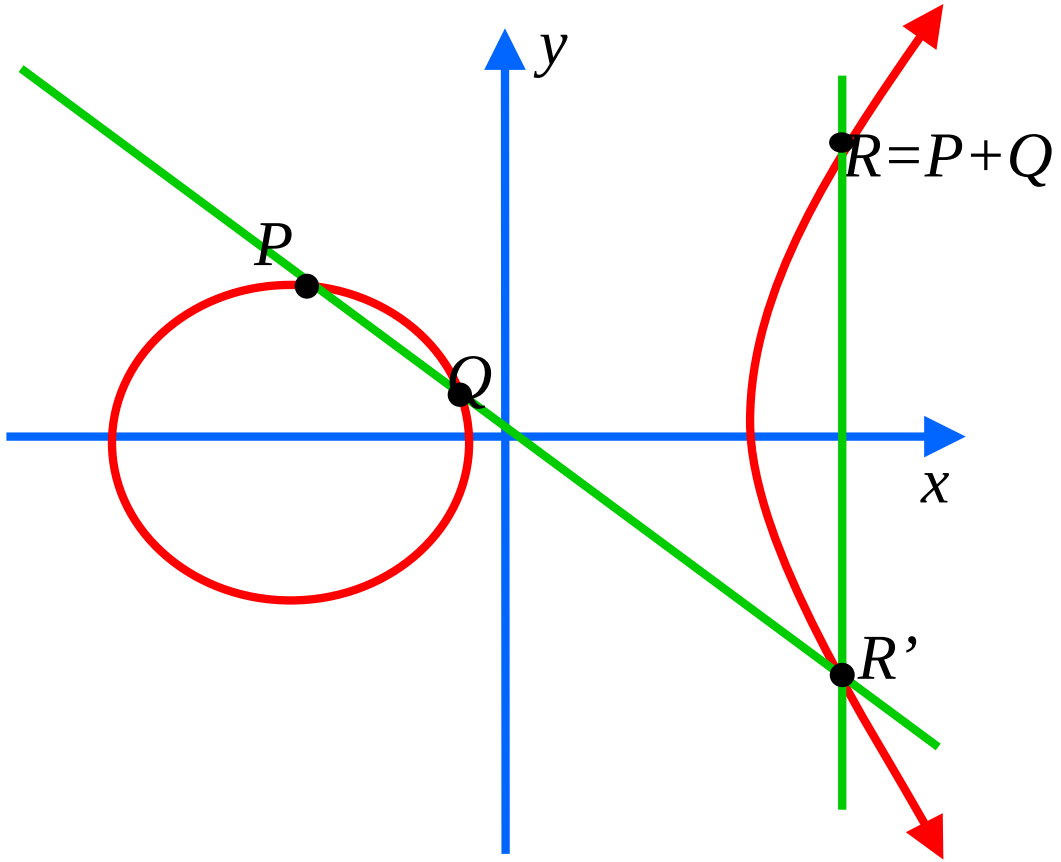


图6.7. 椭圆曲线加法运算示意图

6.3 基于椭圆曲线的数字签名算法

6.3.1 椭圆曲线简介

- 椭圆曲线上的加法运算

当P与Q重合时

$$R = P+Q = P+P = 2P$$

此时称之为点倍运算

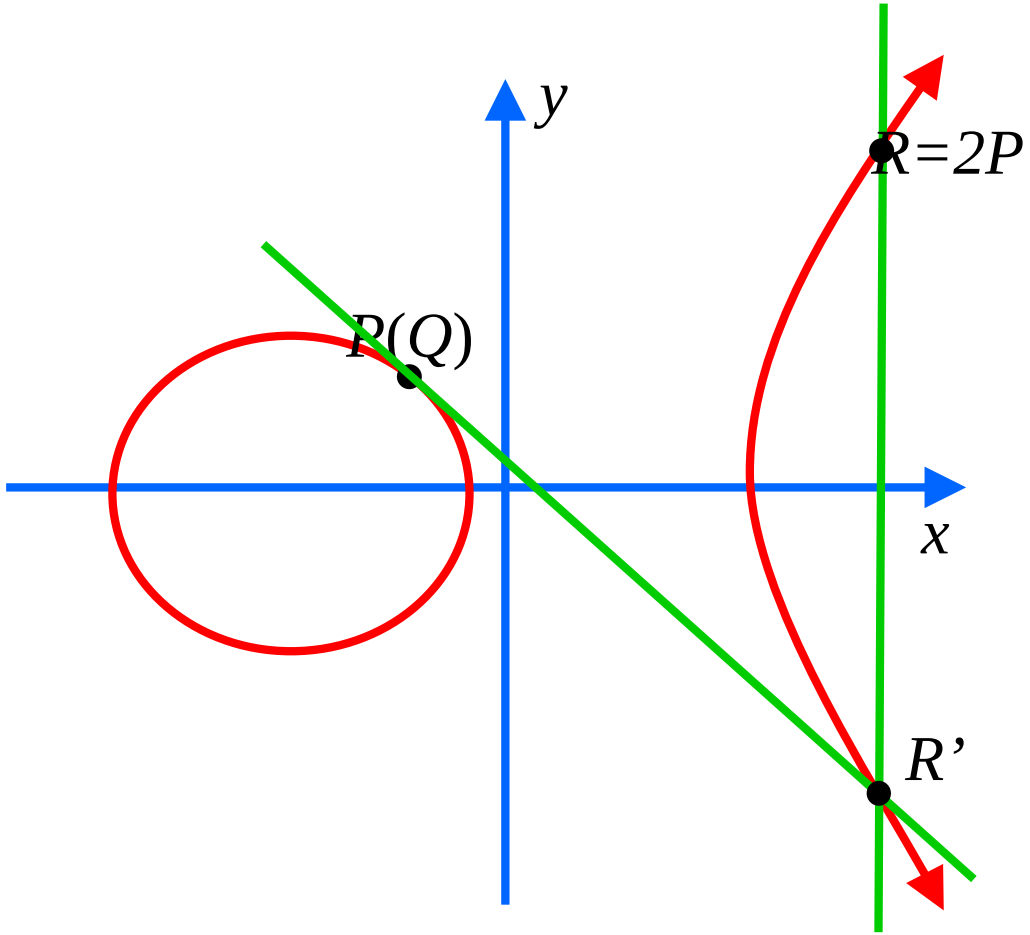


图6.8. 椭圆曲线点倍运算示意图

6.3 基于椭圆曲线的数字签名算法

6.3.1 椭圆曲线简介

- 椭圆曲线上的加法运算

当P与Q关于x轴对称时,

定义P与Q的和为O ,即

$$P + Q = O$$

并称 O 为无穷远点

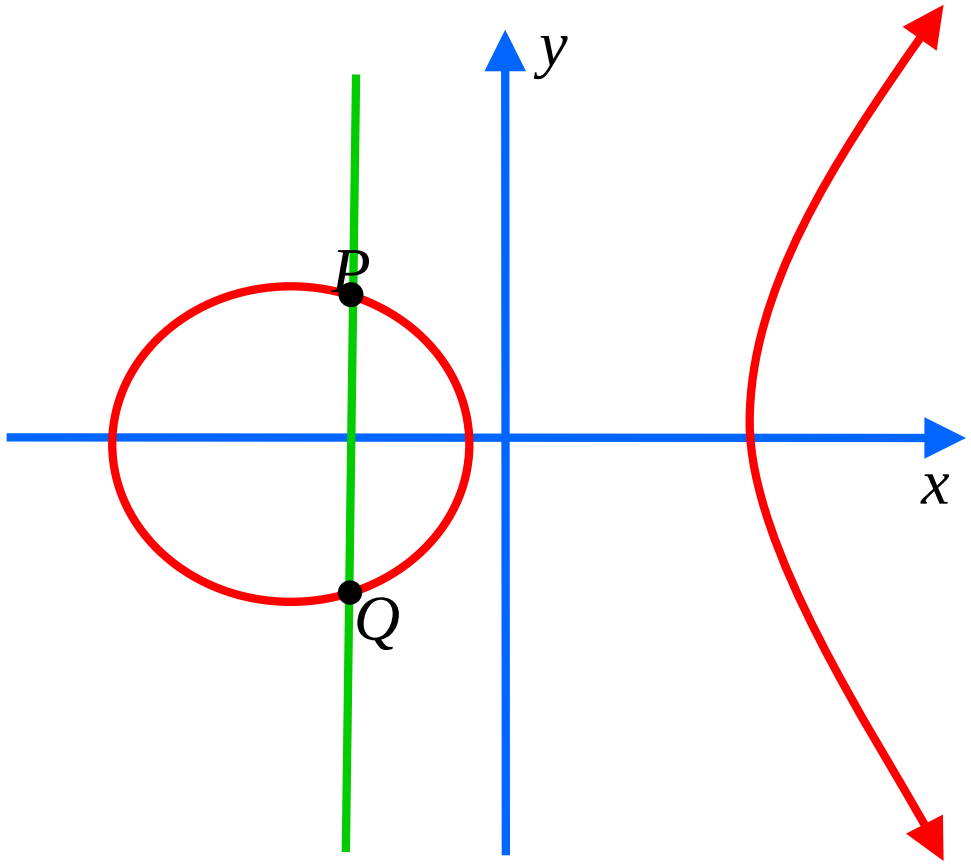


图6.9. 椭圆曲线无穷远点示意图

6.3 基于椭圆曲线的数字签名算法

6.3.1 椭圆曲线简介

➢ 椭圆曲线上的加法运算

可以证明，有限域上的椭圆曲线在我们定义的加法运算下构成群.

既然构成群，就必然有零元和负元，这里的零元就为无穷远点$\boldsymbol{O}$，P的负元就是它关于x轴的对称点，记为$-P$.

显然有

$$P+\boldsymbol{O} = \boldsymbol{O}+P=P$$

若 $P=(x, y)$，则 $-P = (x, -y)$ 且 $P + (-P) = \boldsymbol{O}$

6.3 基于椭圆曲线的数字签名算法

6.3.1 椭圆曲线简介

➢ 加法运算的代数表示

已知$E(F)$上两点$P=(x_1, y_1)$, $Q=(x_2, y_2)$, 求$P+Q$.

解：设$P+Q=R=(x_3, y_3)$,

$$\begin{cases} y^2 = x^3 + ax + b \\ y = \lambda x + c \end{cases}$$

解得

$$\begin{cases} x_3 = \lambda^2 - x_1 - x_2 \\ y_3 = \lambda(x_1 - x_3) - y_1 \end{cases}$$

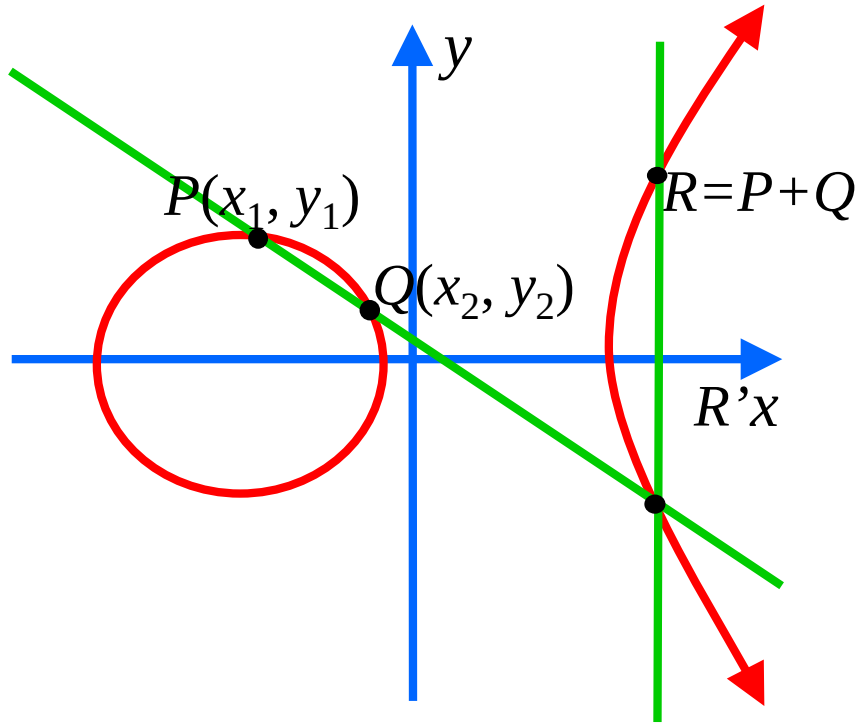


图6.10. 椭圆曲线点加运算代数表示

6.3 基于椭圆曲线的数字签名算法

6.3.1 椭圆曲线简介

➢ 加法运算的代数表示

当P≠Q时，

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1}$$

当P=Q时，

$$\lambda = \frac{3x_1^2 + a}{2y_1}$$

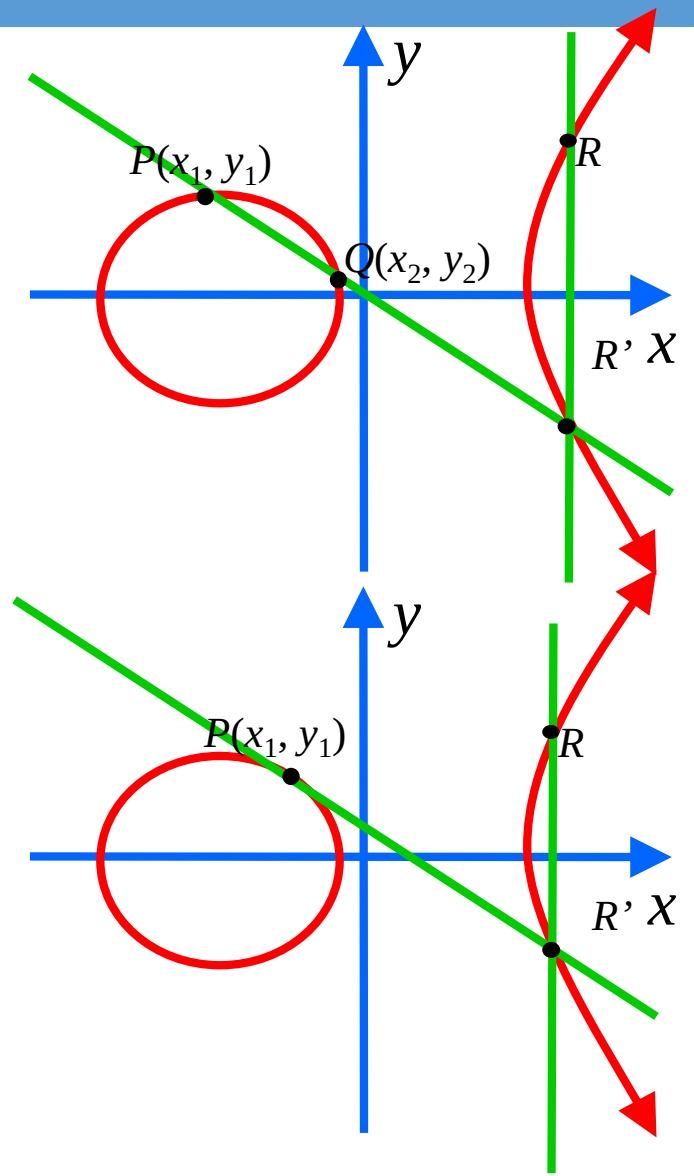


图6.11. 椭圆曲线点加运算代数表示

6.3 基于椭圆曲线的数字签名算法

6.3.1 椭圆曲线简介

- 加法运算的代数表示

$k(k>2)$个相同的点P相加为

$$\underbrace{P+\cdots+P}_{k}=kP$$

此时称之为点乘运算

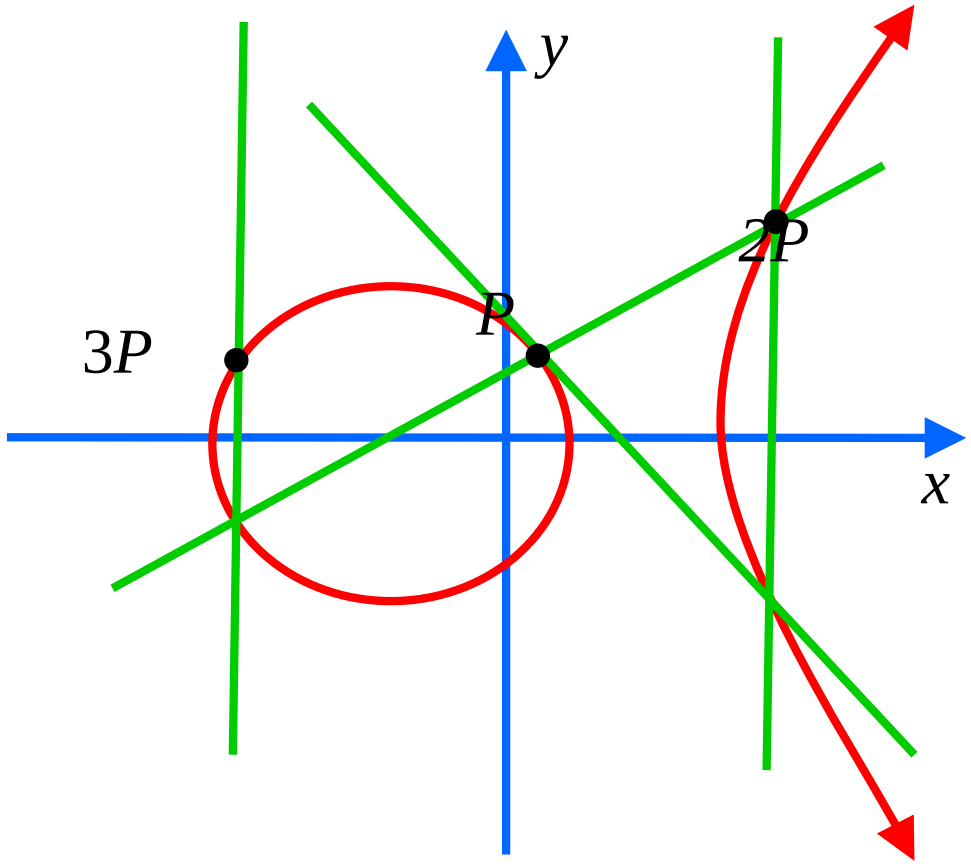


图6.12. 椭圆曲线点乘运算示意图

6.3 基于椭圆曲线的数字签名算法

6.3.1 椭圆曲线简介

➢ **椭圆曲线离散对数问题**

设 $P \in E(F)$, $n = \min\{k \mid k > 0, kP = O\}$ 称 n 为点 P 的阶，记为 $n=ord(P)$.

由阶为 n 的点 P 在上述加法定义下生成的循环群 <P> 是椭圆曲线群 ($E(F)$, +) 的一个 n 阶子群.

设 E 是有限域 F 上的椭圆曲线，G 是E 的一个循环子群，点 P 是 G 的一个生成元，即$G = \{ kP : k \geq 1\}$，在已知 P， Q 的条件下，求解整数n，使得 $nP = Q$ 的问题，称为**椭圆曲线 E 上的离散对数问题**.

目　录

6.3 基于椭圆曲线的数字签名算法

6.3.2 ECDSA算法

- 椭圆曲线数字签名算法(Elliptic Curve Digital Signature Algorithm, ECDSA)是使用椭圆曲线群对数字签名算法(DSA)的模拟.
- ECDSA在1998年被ISO所接受，并且包含它的其他一些标准亦在ISO的考虑之中.
- ECDSA于1999年成为ANSI标准，并于2000年成为IEEE和NIST标准.
- 与普通的离散对数问题(Discrete Logarithm Problem, DLP)和大数分解问题(Integer Factorization Problem, IFP)不同，椭圆曲线离散对数问题(Elliptic Curve Discrete Logarithm Problem, ECDLP)没有亚指数时间的解决方法.

6.3 基于椭圆曲线的数字签名算法

6.3.2 ECDSA算法

1. 系统参数：

- ✓ p, n: 大素数；
- ✓ $E(F_p)$：定义在有限域F_p上的椭圆曲线，由公式$y^2=x^3+a*x+b$
- ✓ G:椭圆曲线$E(F_p)$的基点为阶为q

2. 密钥生成算法

① Alice选择随机数d做为私钥，其中$0< d< n$

② Alice计算公钥$Q = d \cdot G$

③ 输出密钥对 $(sk=d, pk=Q)$

6.3 基于椭圆曲线的数字签名算法

6.3.2 ECDSA算法

给定消息M，Alice执行以下步骤产生消息的签名：

3. 签名算法

① 选择随机数k，其中$0< k< n$

② 计算$K = k \cdot G = (x_1, y_1)$

③ 计算$r= x_1 \bmod n$

④ 计算$e=H(M)$

⑤ 计算$s=k^{-1} \cdot (e+d \cdot r) \bmod n$

⑥ 输出签名(r, s)

收到消息M和签名(r, s)后，Bob执行以下步骤验证签名：

4. 验证算法

① 检验r, s是否满足$0< r, s < n$

② 计算$w= s^{-1} \bmod n$和$e=H(M)$

③ 计算$u_1=e \cdot w \bmod n$和$u_2=r \cdot w \bmod n$

④ 计算$u_1 \cdot G+u_2 \cdot Q= (x_1, y_1)$和$v= x_1 \bmod n$

⑤ 比较v和r是否相等，如果相等则输出1，否则输出0

6.3 基于椭圆曲线的数字签名算法

6.3.2 ECDSA算法

5. 推荐论文

① Hankerson, Darrel, et al. “Software Implementation of Elliptic Curve Cryptography over Binary Fields.” *Proceedings of the Second International Workshop on Cryptographic Hardware and Embedded Systems* (CHES’00), 2000, pp. 1–24.

② Brown, Michael, et al. “Software Implementation of the NIST Elliptic Curves Over Prime Fields.” *The Cryptographers Track at the RSA Conference* (CT-RSA’01), 2001, pp. 250–265.

目　录

6.3 基于椭圆曲线的数字签名算法

6.3.3 SM2数字签名算法

在政府高度重视和市场迫切需求的双向驱动下，国密算法SM1—SM9应时而生. 其中， **SM2是国家密码管理局于2010年12月17日发布的椭圆曲线公钥密码算法**，包含5个部分：

- 总则
- 数字签名算法
- 密钥交换协议
- 公钥加密算法
- 参数定义

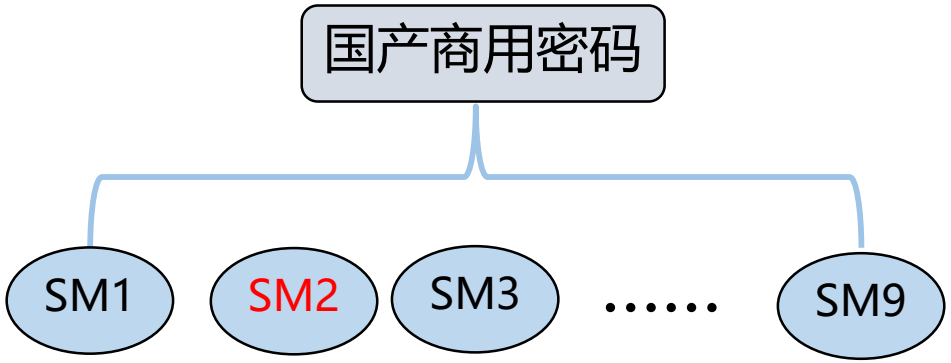


图6.13. 国产商用密码算法示意图

6.3 基于椭圆曲线的数字签名算法

6.3.3 SM2数字签名算法

$$p_{SM2} = 2^{256} - 2^{224} - 2^{96} + 2^{64} - 1$$

1. 曲线参数

SM2标准推荐使用**256位素域F_p上的椭圆曲线$y^2=x^3+ax+b$**，其中：

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

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

G_y=BC3736A2 F4F6779C 59BDCEE3 6B692153 D0A9877C C62A4740 02DF32E5 2139F0A0

2. 密钥生成算法

① Alice选择随机数d_A做为私钥，其中0< d< n

② Alice计算公钥$P_A = dA \cdot G$

③ 输出密钥对 (sk=d_A, pk= P_A)

6.3 基于椭圆曲线的数字签名算法

6.3.3 SM2数字签名算法

3. 签名算法

- 设Alice发签名消息M给Bob，ID_A是Alice的标识符，$ENTL_A$是ID_A的长度，d_A是A的私钥，基点$G=(x_G, y_G)$，A的公钥$P_A=d_A \cdot G=(x_A, y_A)$.
- $Z_A=H(ENTL_A \| ID_A \| a \| b \| x_G \| y_G \| x_A \| y_A)$， H是SM3算法

① 设置$M^*=Z_A \| M$并计算 $e=H(M^*)$;

② 产生随机数 $k \in [1, n-1]$;

③ 计算椭圆曲线点 $G_1=k \cdot G=(x_1, y_1)$;

④ 计算 $r=(e+x_1) \bmod n$，若$r=0$或$r+k=n$则返回②；

⑤ 计算 $s=(1+d_A)^{-1} \cdot (k-r \cdot d_A) \bmod n$，若$s=0$则返回②；

⑥ 以(r, s)作为对消息M的签名.

6.3 基于椭圆曲线的数字签名算法

6.3.3 SM2数字签名算法

4. SM2数字签名算法和ECDSA算法的比较

- **两者的基本思想一致**
  - ① 都是以 r, s为签名
  - ② 以 kG产生r
  - ③ 以 d, r, k 产生s
- **两者有许多不同**
  - ① ECDSA算法计算$r = x_1 \bmod n$，而**SM2算法计算$r = (e+x_1) \bmod n$**，这里$G_1=k \cdot G= (x_1, y_1)$.
  - ② ECDSA算法计算$s=k^{-1}(e-d_A \cdot r) \bmod n$.而SM2数字签名算法计算$s = (1 + d_A)^{-1} \cdot (k - r \cdot d_A) \bmod n$，$M$ 没有直接出现，而是通过r参与其中，私钥d_A作用了两次.
  - ③ ECDSA算法中的k^{-1}不能预计算，而SM2数字签名中的$(1 + d_A)^{-1}$可以预计算，提高了计算性能.
  - ④ ECDSA算法直接使用$e = H(M)$产生签名，而**SM2数字签名算法使用$e = H(M^*)$产生签名**，这里$M^* = Z_A \| M$，$Z_A = H(ENTL_A \| ID_A \| a \| b \| x_G \| y_G \| x_A \| y_A)$包含了用户参数和系统参数，起到一定的认证作用，提高了安全性；

6.3 基于椭圆曲线的数字签名算法

6.3.3 SM2数字签名算法

- 两者有许多不同

④ SM2数字签名算法增加了合理性检查，确保签名正确，提高安全性.例如第④中检查 $r+k=n$是否成立

如果 $r+k=n$ ，则$k=-r \bmod n$，会使得

$$
\begin{aligned}
s &= (1+d_A)^{-1}\cdot(k-r\cdot d_A) \bmod n \\
&= (1+d_A)^{-1}\cdot(-r-r\cdot d_A) \bmod n \\
&= -r\cdot (1+d_A)^{-1}\cdot (1+d_A) \bmod n = -r
\end{aligned}
$$

显然是不合适的.

⑤ SM2数字签名中的s具有线性关系，可以构造盲签名、协同签名、环签名等其它特殊需求的签名.

$$
\begin{aligned}
s &= (1+d_A)^{-1}\cdot(k-r\cdot d_A) \bmod n \\
&= (1+d_A)^{-1}\cdot(k+r-r-r\cdot d_A) \bmod n \\
&= (1+d_A)^{-1}\cdot(k+r)-r \bmod n
\end{aligned}
$$

结论：与ECDSA算法相比， SM2签名算法具有更好的安全性和更低的计算代价.

6.3 基于椭圆曲线的数字签名算法

6.3.3 SM2数字签名算法

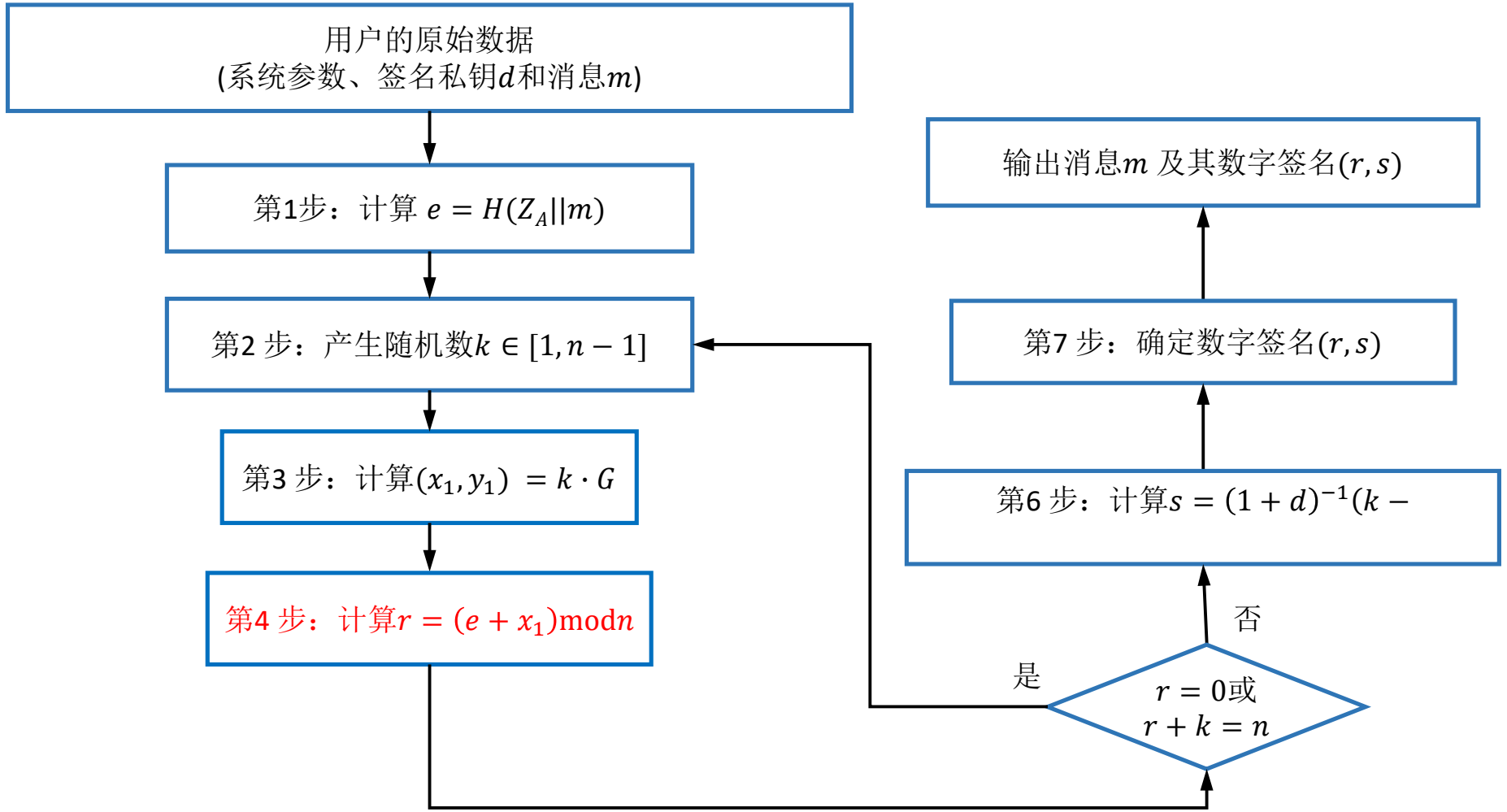


图6.14. SM2数字签名算法签名过程示意图

6.3 基于椭圆曲线的数字签名算法

6.3.3 SM2数字签名算法

5. 验证算法

接收到的消息为M'，签名为(r', s')和发送者Alice的公钥P_A， Bob执行如下步骤验证合法性：

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 设置$M^* = Z_A \| M'$；

④ 计算 $e' = H(M^*)$；

⑤ 计算 $t = (r' + s') \bmod n$，若$t = 0$，则验证不通过；

⑥ 计算椭圆曲线点 $(x_1', y_1') = s' \cdot G + t \cdot P_A$；

⑦ 计算 $v = (e' + x_1') \bmod n$，检验$v = r'$是否成立，若成立则验证通过；否则验证不通过.

6.3 基于椭圆曲线的数字签名算法

6.3.3 SM2数字签名算法

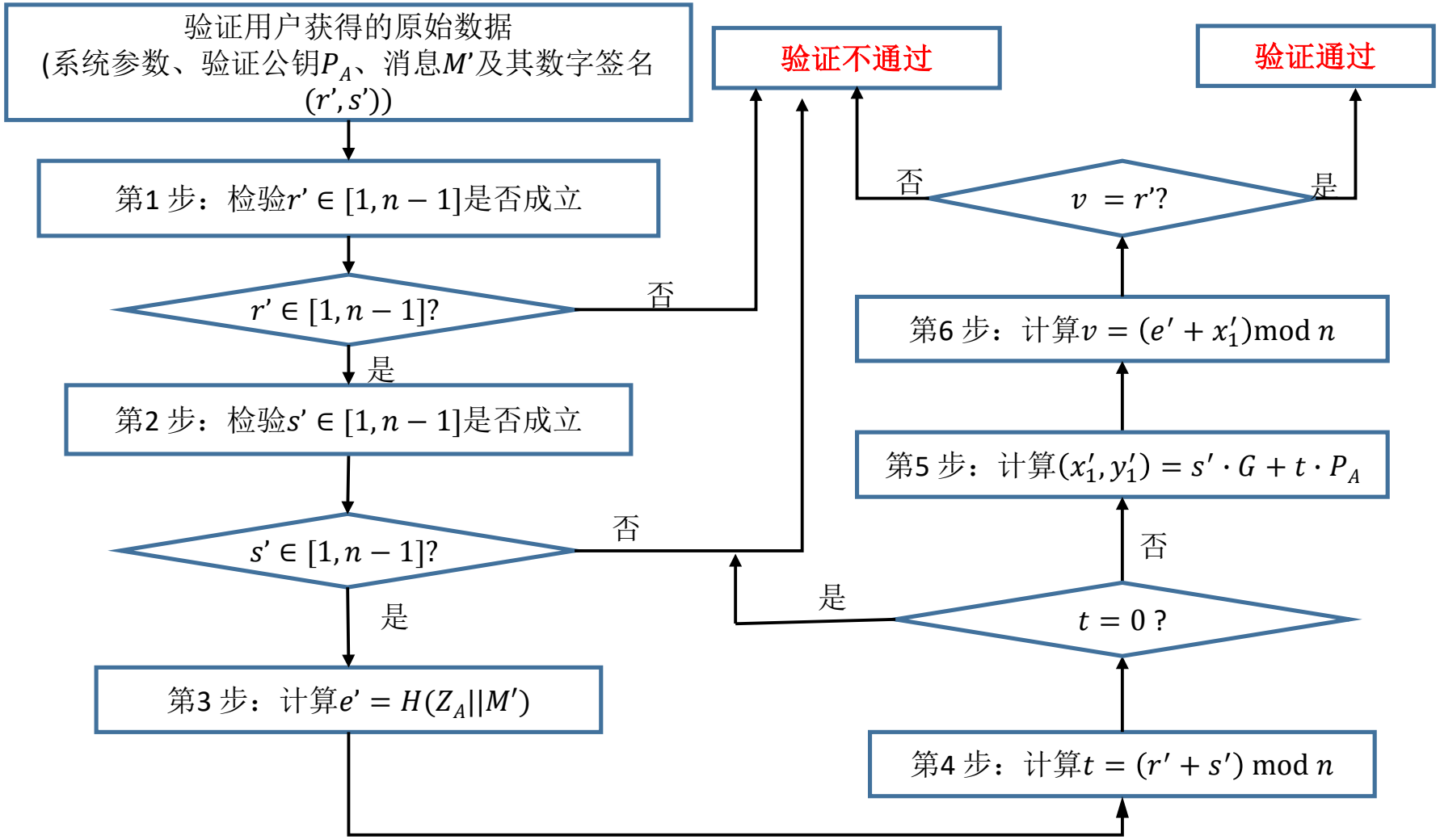


图6.15. SM2数字签名算法验证过程示意图

6.3 基于椭圆曲线的数字签名算法

6.3.3 SM2数字签名算法

➢ 算法的正确性

由于$s=(1+d_A)^{-1}\cdot(k-r\cdot d_A)$，$P_A=d_A\cdot G$，我们可以得到：

$$
\begin{aligned}
&s\cdot G+t\cdot P_A\\
&=s\cdot G+(r+s)\cdot d_A\cdot G\\
&=(s+s\cdot d_A)\cdot G+r\cdot d_A\cdot G\\
&=s\cdot(1+d_A)\cdot G+r\cdot d_A\cdot G\\
&=(1+d_A)^{-1}\cdot(k-r\cdot d_A)\cdot(1+d_A)\cdot G+r\cdot d_A\cdot G\\
&=(k-r\cdot d_A)\cdot G+r\cdot d_A\cdot G\\
&=k\cdot G-r\cdot d_A\cdot G+r\cdot d_A\cdot G\\
&=k\cdot G
\end{aligned}
$$

6.3 基于椭圆曲线的数字签名算法

6.3.3 SM2数字签名算法

➢ **验证算法的特点**

SM2 签名验证算法的一个显著特点是，其中加入了较多的检错功能.

因为这是收信者对收到的签名数据进行验证，而签名数据是经过信道传输过来的，由于信道干扰和对手的篡改，因此，签名数据中含有错误或被篡改的可能性是存在的.

把错误和篡改检测出来，对提高签名验证系统的数据完整性、系统可靠性和安全性是有益的.

- ✓ 验证算法中的①检查签名分量 r'的合理性
- ✓ 验证算法中的②检查签名分量 s'的合理性
- ✓ 验证算法中的⑤检查 t的正确性

6.3 基于椭圆曲线的数字签名算法

6.3.3 SM2数字签名算法

6. SM2签名算法的应用

- **安全：**前尚没有发现求解椭圆曲线离散对数问题的亚指数算法.
- **实现：**软硬件实现规模小，容易实现.160位的椭圆曲线密码的安全性，相当于1024位的RSA密码.
- **实现难点：**倍点运算.
- **应用场景：**目前最大的应用是二代身份证.
- **应用前景：** 2019年10月26日，第十三届全国人民代表大会常务委员会第十四次会议表决通过密码法，于2020年1月1日起施行.

目　录

6.4 基于身份的数字签名算法

6.4.1 基于身份的数字签名简介

PKI 是指由数字证书、证书颁发机构 (CA) 和证书注册机构(RA)组成的一套系统：

- 数字证书公钥和私钥
- 加密算法和摘要算法
- 证书链

  受信任的根证书颁发机构-中级根证书颁发机构-用户证书
- 证书管理

  申请、颁发、吊销、重新颁发、续期等

数字证书主要用途是绑定用户身份和公钥，可以用于公钥加密、密钥写上、数字签名等领域.

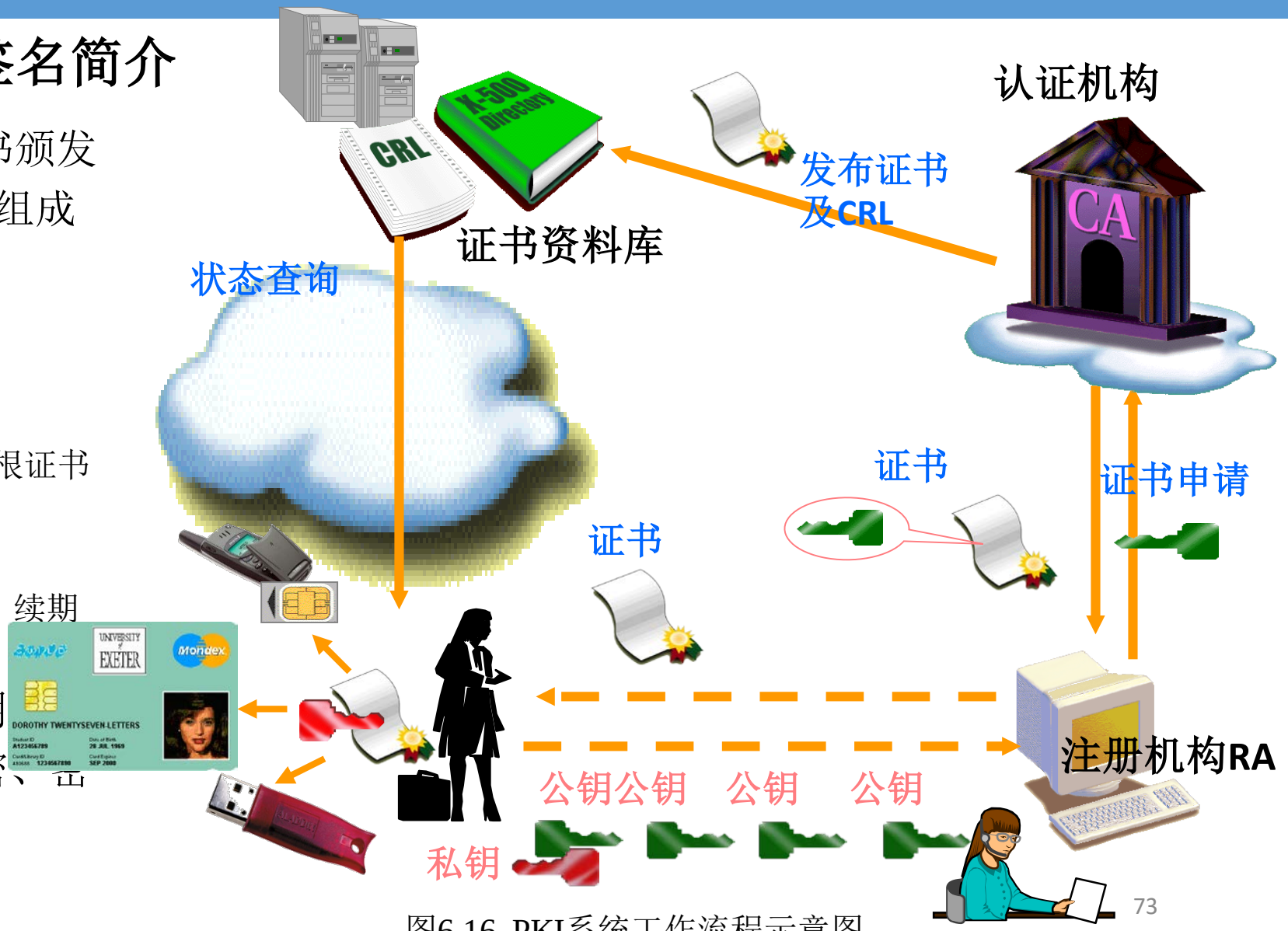


图6.16. PKI系统工作流程示意图

6.4 基于身份的数字签名算法

6.4.1 基于身份的数字签名简介

- **证书的可靠性问题**

从证书可靠性角度来评估中国互联网的安全性，中国互联网面临着三大安全风险：

风险一：大部分重要系统没有部署SSL证书

例如：邮件系统、移动APP中的手机银行、移动社交软件

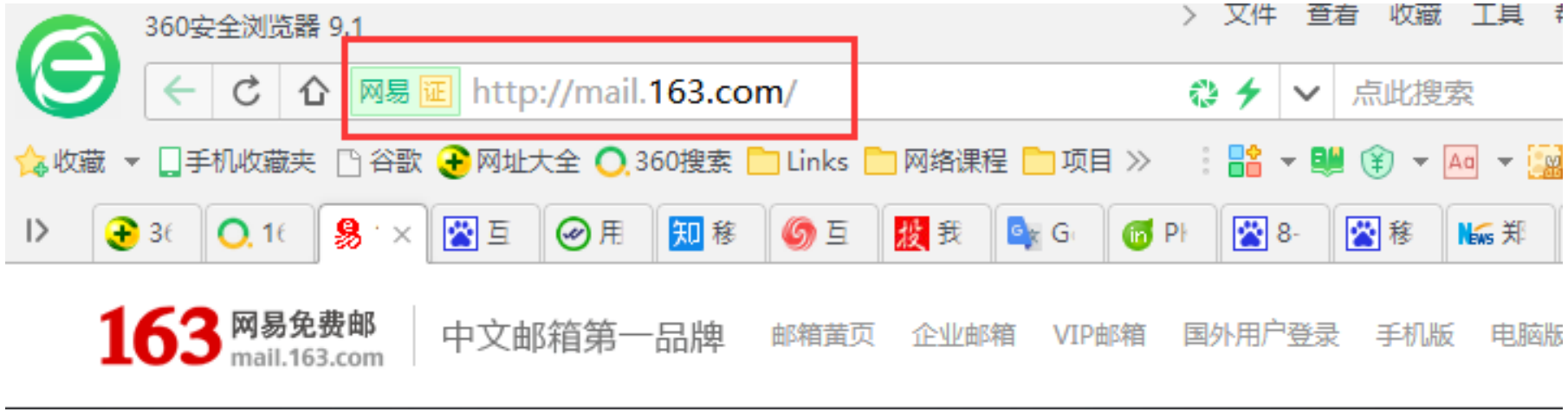


图6.17. 证书可靠性问题示意图1

6.4 基于身份的数字签名算法

6.4.1 基于身份的数字签名简介

- 证书的可靠性问题

从证书可靠性角度[illegible]着三大安全风险：

风险二：部分重要[illegible]行网银证书，电子商务系统

例如：网上银行、

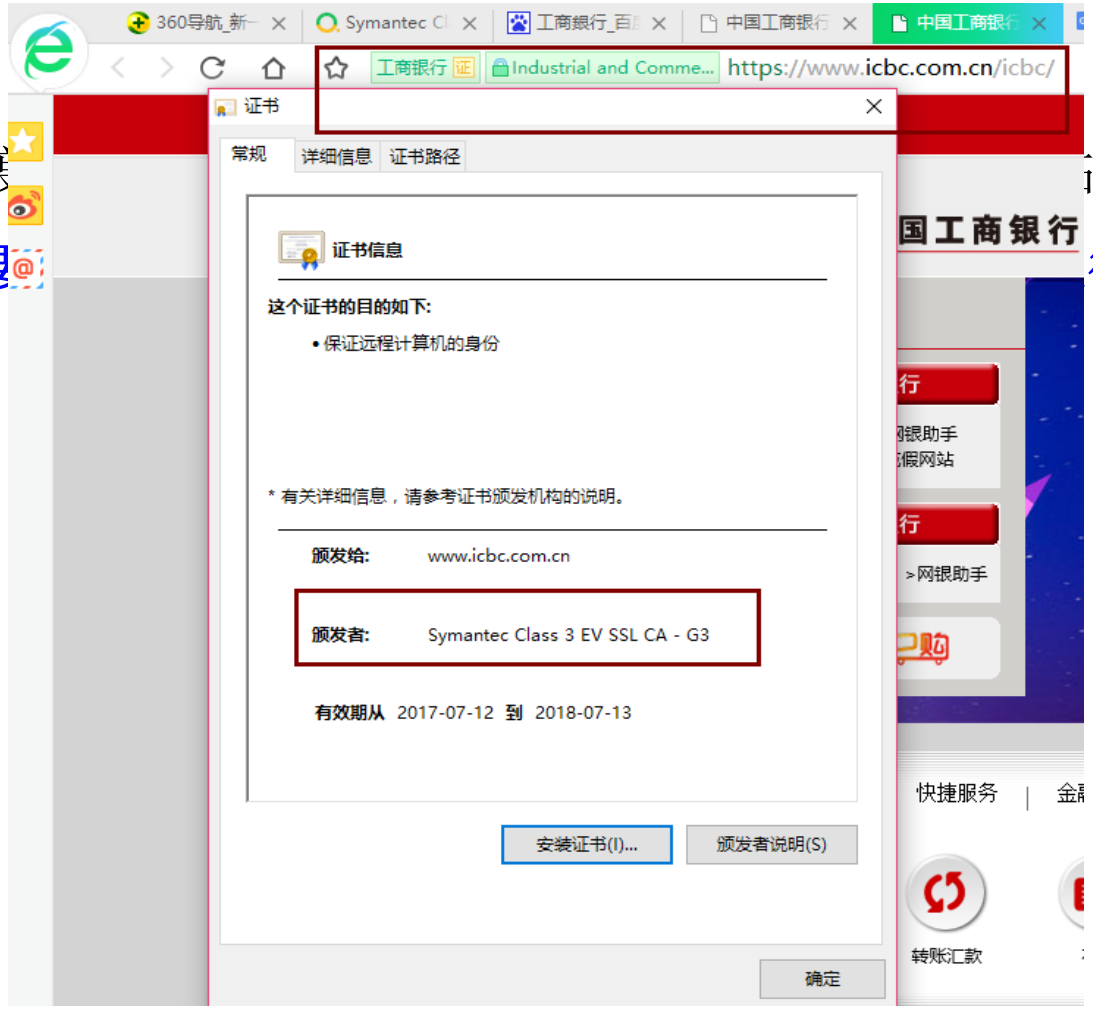


图6.18. 证书可靠性问题示意图2

6.4 基于身份的数字签名算法

6.4.1 基于身份的数字签名简介

- 证书的可靠性问题

从证书可靠性[illegible]大安全风险：

风险三：一些[illegible]

例如：12306[illegible]用自签证书

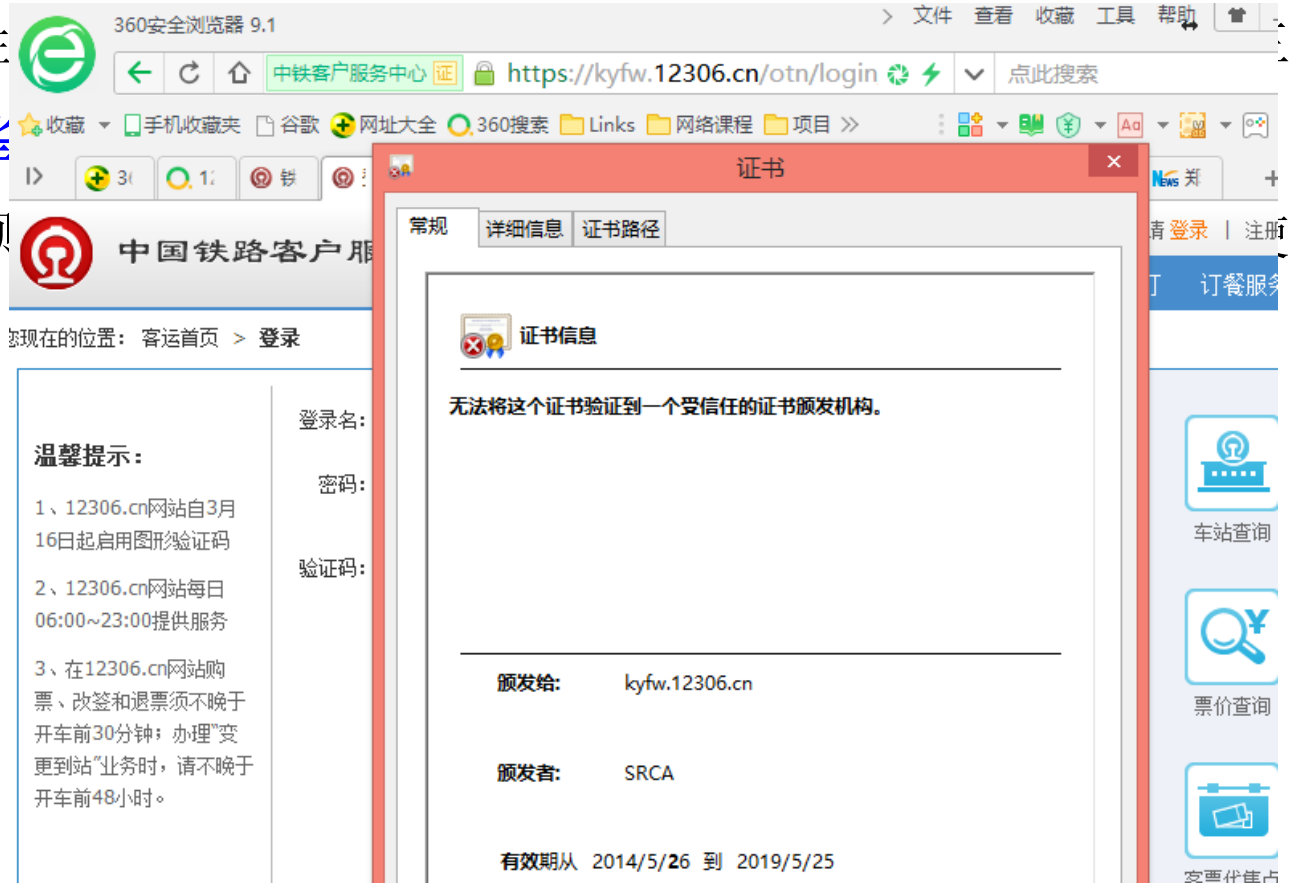


图6.19. 证书可靠性问题示意图3

6.4 基于身份的数字签名算法

6.4.1 基于身份的数字签名简介

➢ 证书的管理问题

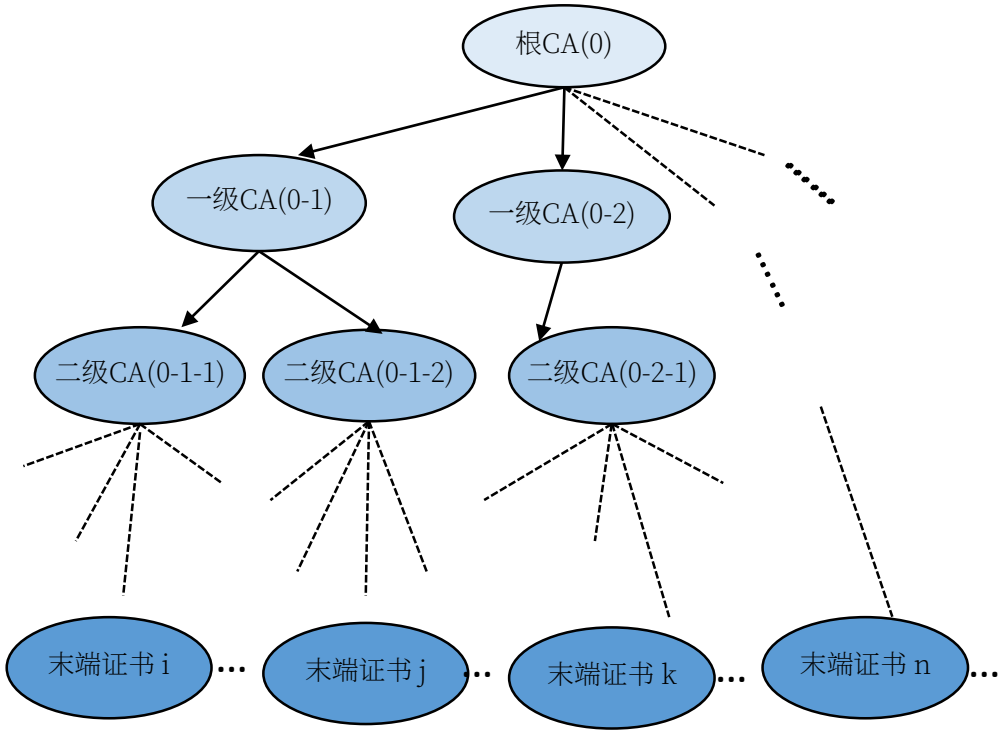


图6.20. 层次化证书结构示意图

CA体系从根证书开始，逐级签发各级证书，从而形成树状信任体系.

弊端：

①难以构造复杂的身份标志体系；

②难以同步数字证书的撤销与数字签名的“作废”；

③数字证书的管理难度以及风险随系统规模的增大而增加；

④数字证书和私钥都需要载体，使用起来不方便；

⑤证书查找与证书验证的开销；

6.4 基于身份的数字签名算法

6.4.1 基于身份的数字签名简介

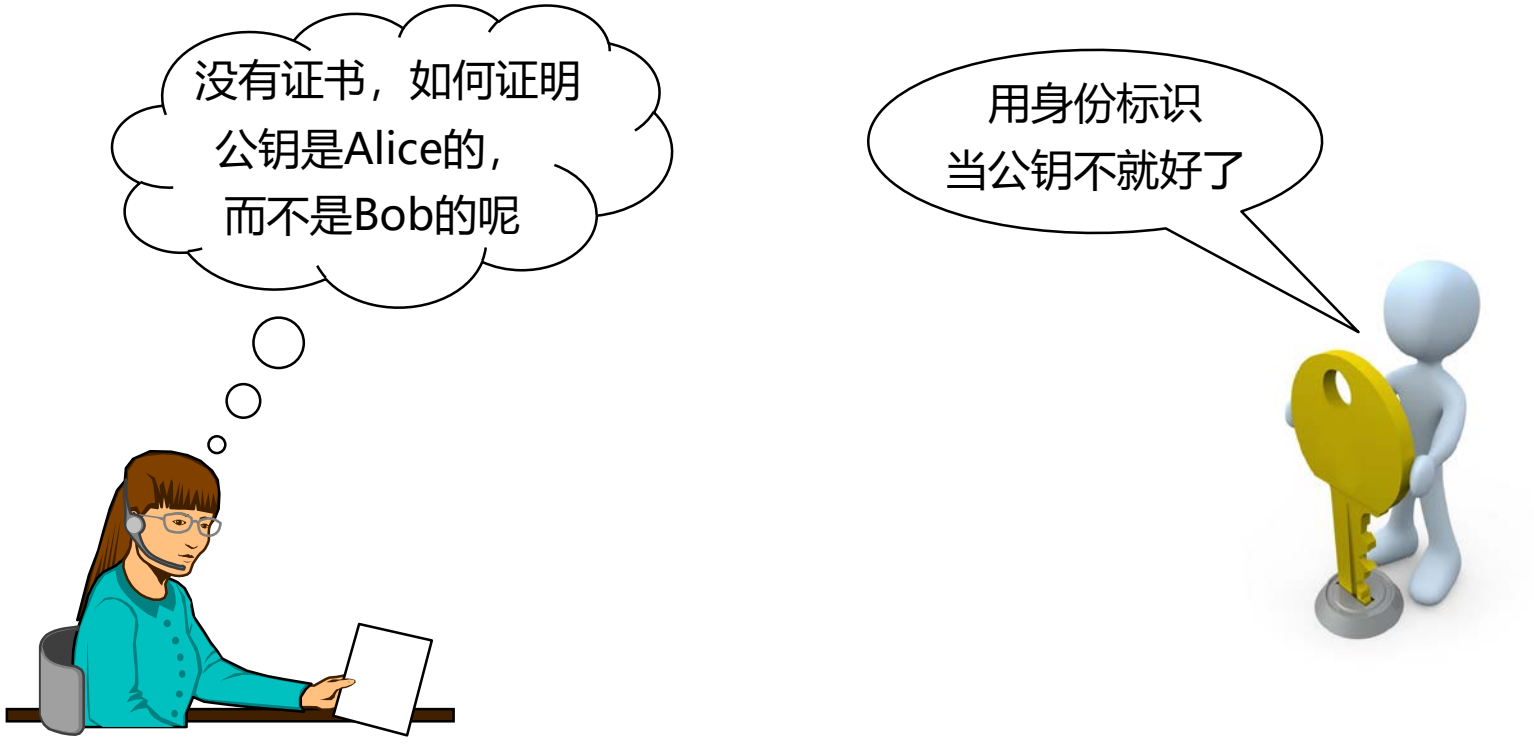


图6.21. 基于身份密码示意图

6.4 基于身份的数字签名算法

6.4.1 基于身份的数字签名简介

- 1984年，以色列密码学家Shamir提出了基于身份的密码系统(Identity-based Cryptography, IBC)的概念，并设计了一个基于RSA算法的基于身份数字签名算法.
- 2001年，D. Boneh和M. Franklin , R. Sakai, K. Ohgishi 和 M. Kasahara利用椭圆曲线上的双线性对设计了基于身份的加密算法.
- 2001年，C. Cocks利用平方剩余难题设计了基于身份的加密算法.
- D. Boneh和M. Franklin提出的IBC (BF-IBC)的安全性可以证明并且有较好的效率，引起了极大的反响.

6.4 基于身份的数字签名算法

6.4.1 基于身份的数字签名简介

一个基于身份的数字签名算法由以下4个子算法构成：

- **初始化**
  - ✓输入: 安全参数k
  - ✓输出: 系统参数params和主密钥master-key
- **用户私钥生成**
  - ✓输入: 系统参数params、主密钥master-key和用户身份 ID$\in\{0,1\}^*$
  - ✓输出: 用户私钥d_{ID}

6.4 基于身份的数字签名算法

6.4.1 基于身份的数字签名简介

➢ 签名

- 输入: 系统参数params，消息M和签名者的私钥d_{ID}
- 输出: 签名σ

➢ 验证

- 输入: 系统参数params, 签名σ，签名者公钥(身份)ID$\in\{0,1\}^*$, 消息M
- 输出: “Accept” 或者 “Reject”.

基于身份的数字签名的工作流程入下图所示：

6.4 基于身份的数字签名算法

6.4.1 基于身份的数字签名简介

签名者

验证者

6.4 基于身份的数字签名算法

6.4.1 基于身份的数字签名简介


公钥

签名者

验证者

6.4 基于身份的数字签名算法

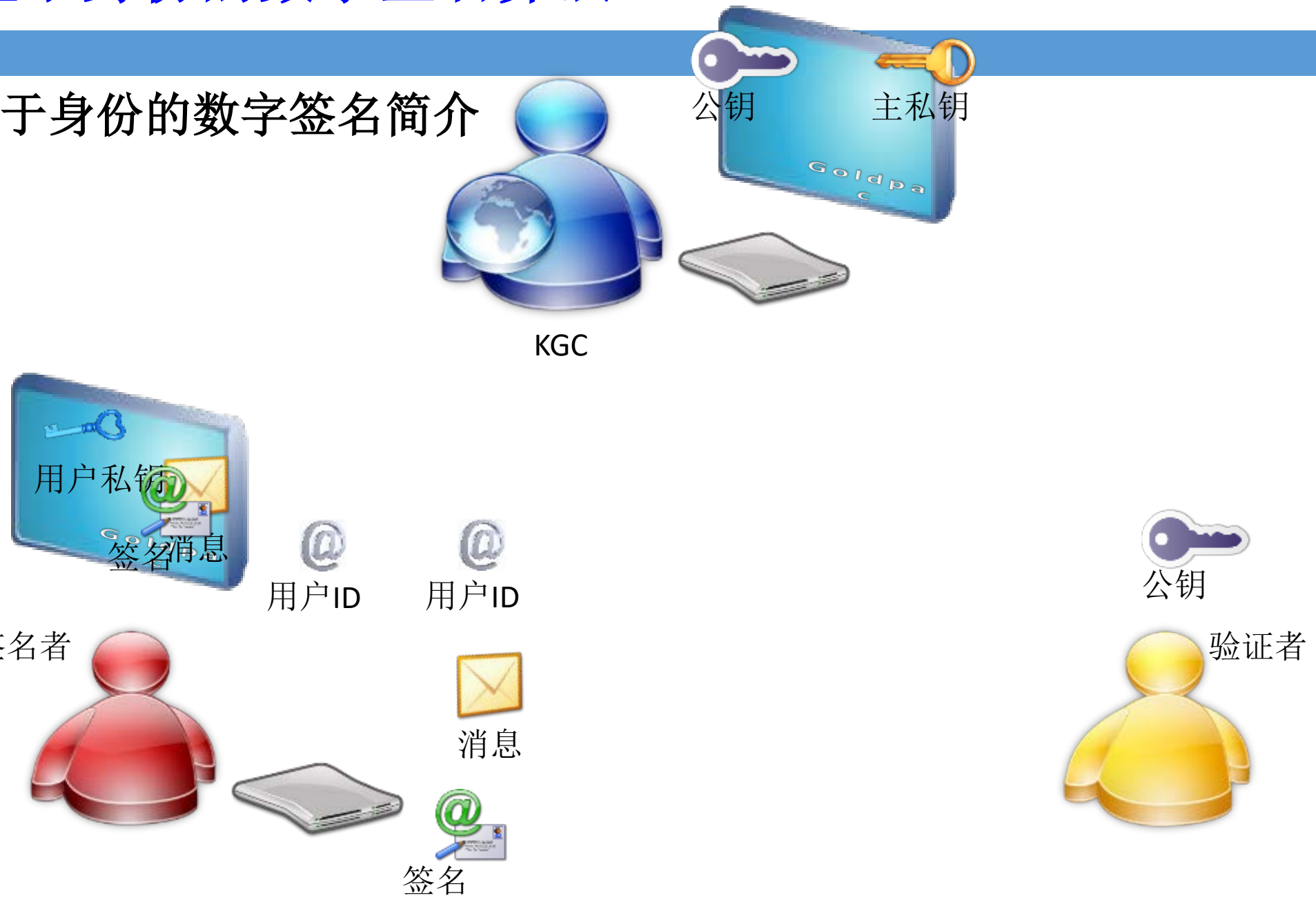
6.4.1 基于身份的数字签名简介

6.4 基于身份的数字签名算法

6.4.1 基于身份的数字签名简介

6.4 基于身份的数字签名算法

6.4.1 基于身份的数字签名简介



6.4 基于身份的数字签名算法

6.4.1 基于身份的数字签名简介

6.4 基于身份的数字签名算法

6.4.1 基于身份的数字签名简介

Shamir提出了一个采用RSA算法的基于身份的数字签名算法。

1. 初始化

① 选取两个大素数p，q，计算它们的乘积n；

② 选取与$\Phi(n)$互素的整数 e ，计算主私钥d满足$e \times d = 1 \bmod \Phi(n)$ ；

③ 选取一个单向函数h；

④ 输出系统参数: n, e, h； 保存主密钥: n 和d.

2. 用户私钥生成:

给定用户的 ID, PKG计算用户的私钥$g_{ID} = h(ID)^d \bmod n$.

6.4 基于身份的数字签名算法

6.4.1 基于身份的数字签名简介

3. 签名:

用户利用私钥 g_{ID} 执行如下步骤，签署消息 m:

① 选取随机整数 r，计算$t = r^{e} \bmod n$；

② 计算$s = g_{ID} \times r^{h(t, m)} \bmod n$；

③ 输出签名 $\sigma = <s, t>$.

4. 验证:

验证者收到$\sigma = <s, t>$，消息m和签名者的公钥(身份ID), 验证下式是否成立

$$s^{e} = h(\mathrm{ID}) \times t^{h(t, m)} \bmod n$$

如果成立则输出“Accept”，否则输出“Reject”.

6.4 基于身份的数字签名算法

6.4.1 基于身份的数字签名简介

- 效率

签名和验证都需要

2 模幂(Modular Exponentiations),

1模乘(Modular Multiplication) 和

1 哈希运算(Hash Operation).

- 安全性

分解因子问题困难Integer Factorization Problem (IFP).

目　录

6.4 基于身份的数字签名算法

6.4.2 SM9数字签名算法

➢ 双线性对的概念

设q是一个大素数，G_1和 G_2是阶为q的**两个加法群**(椭圆曲线上的群)， G_T是阶为q的**乘法群**，我们称映射$e: G_1 \times G_2 \rightarrow G_T$为双线性映射(双线性对)，如果它满足以下条件：

① **双线性性**：任给 $P_1 \in G_1, P_2 \in G_2$和任意整数 $a, b \in Z_q$, $e(a \cdot P_1, b \cdot P_2) = e(P_1, P_2)^{ab}$；

② **非退化性**：存在 $Q_1 \in G_1, Q_2 \in G_2$使得 $e(Q_1, Q_2) \neq 1_{G_T}$；

③ **可计算性**：对于任意给定的 $Q_1 \in G_1, Q_2 \in G_2$，计算 $e(Q_1, Q_2)$是容易的；

如果$G_1 = G_2$，则称e为对称双线性对，否则称e为非对称双线性对.目前在密码学中广泛应用的主要有Weil对、Tate对、Ate对、R-Ate对等.

SM9密码算法选用了安全性能好、运算速率高的R-Ate对.

6. 4 基于身份的数字签名算法

6.4.2 SM9数字签名算法

在政府高度重视和市场迫切需求的双向驱动下，国密算法SM1—SM9应时而生.其中，**SM9算法于2016年3月由国家密码管理局正式公布**.

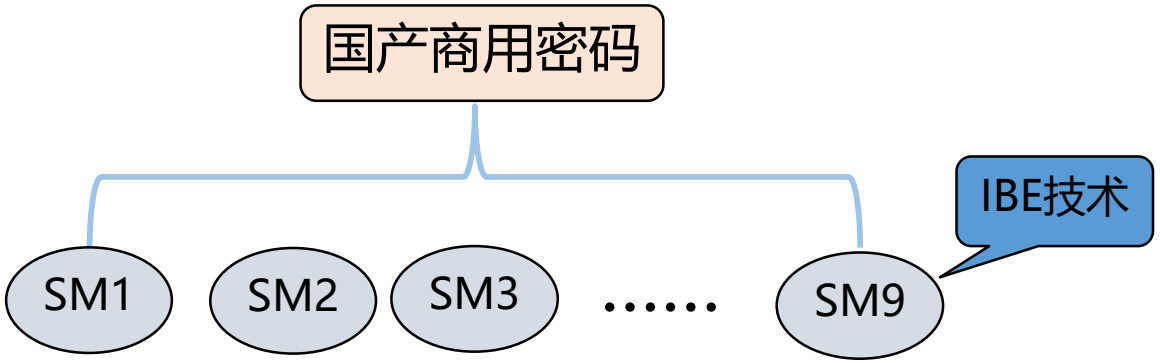


图6.22. 国产商用密码示意图

SM9算法是一种基于双线性对的标识密码算法，它可以把用户的身份标识用以生成用户的公、私密钥对，包含四个部分: (1)总则， (2)数字签名算法， (3)密钥交换协议， (4)密钥封装机制和公钥加密算法.我们主要关注数字签名算法，具体如下：

6.4 基于身份的数字签名算法

6.4.2 SM9数字签名算法

1. 系统参数生成

密钥生成中心(Key Generation Center, KGC)执行以下步骤生成系统参数和主私钥：

① KGC生成随机数sk做为主私钥，这里$0< sk <q-1$；

② KGC计算系统公钥$P_{pub}=sk \cdot P_2$；

③ KGC保存私钥sk, 公布系统公钥.

注意：

① SM9算使用BN曲线，G_1和 G_2分别是椭圆曲线$E(F_p)$和$E(F_{p^2})$的加法群，G_T是乘法群 $F_{p^{12}}$ ，群G_2中元素尺寸是群G_1中元素尺寸的2倍.

② 选择系统公钥为G_2中的元素，那么就可以使得用户私钥和签名中一部分是G_1中元素，降低了用户私钥和签名的尺寸.

6.4 基于身份的数字签名算法

6.4.2 SM9数字签名算法

2. 用户私钥生成

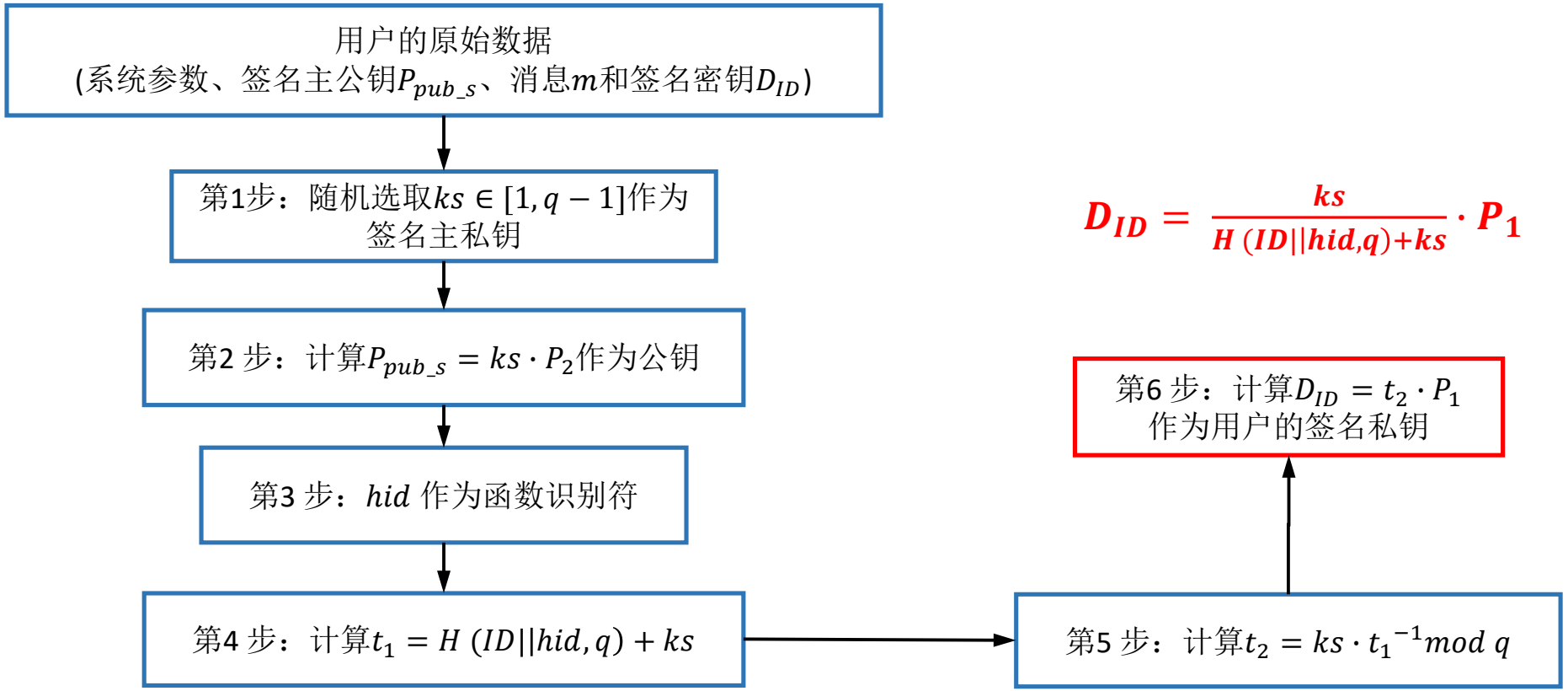


图6.23. SM9数字签名算法用户私钥生成示意图

6.4 基于身份的数字签名算法

6.4.2 SM9数字签名算法

3. 签名

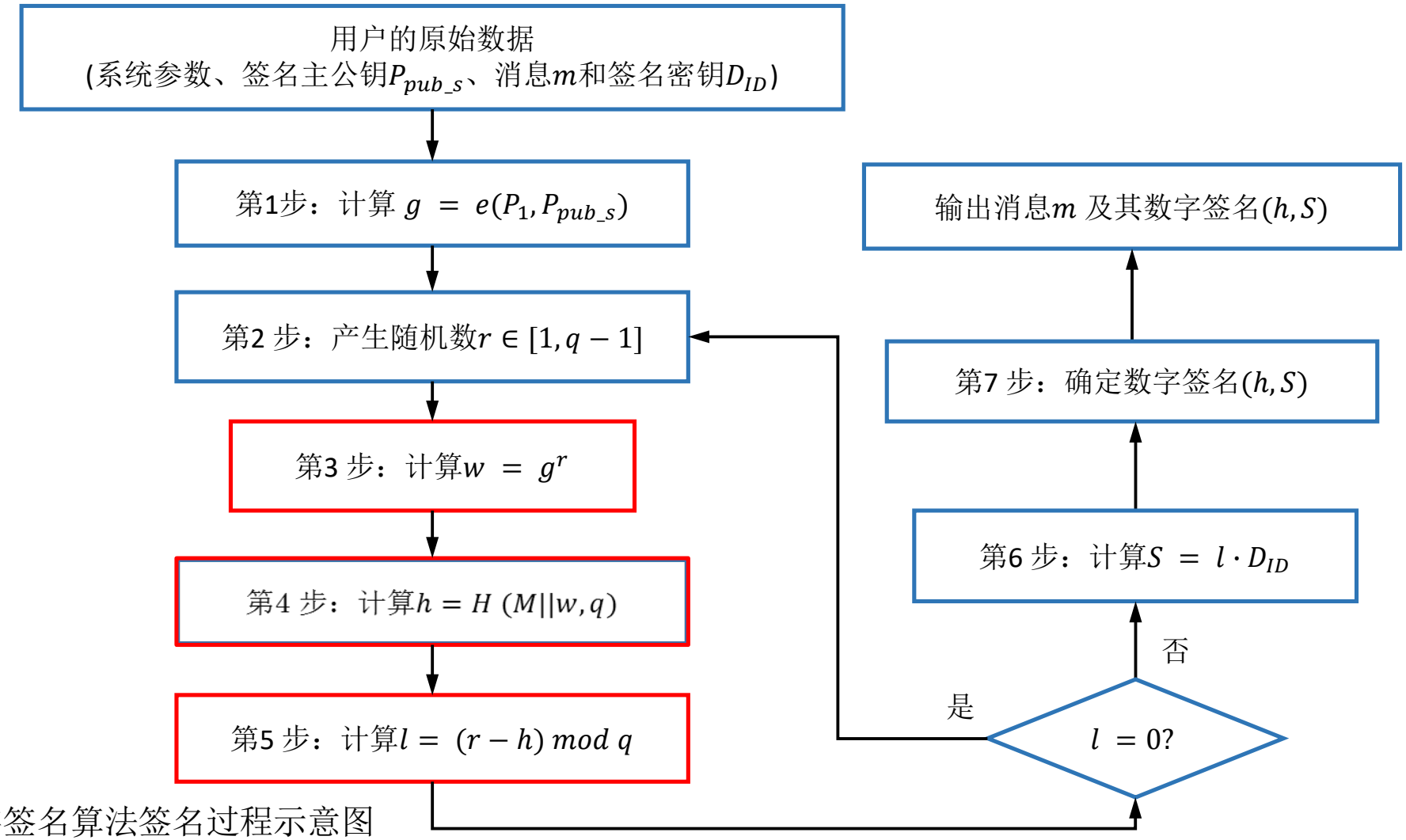


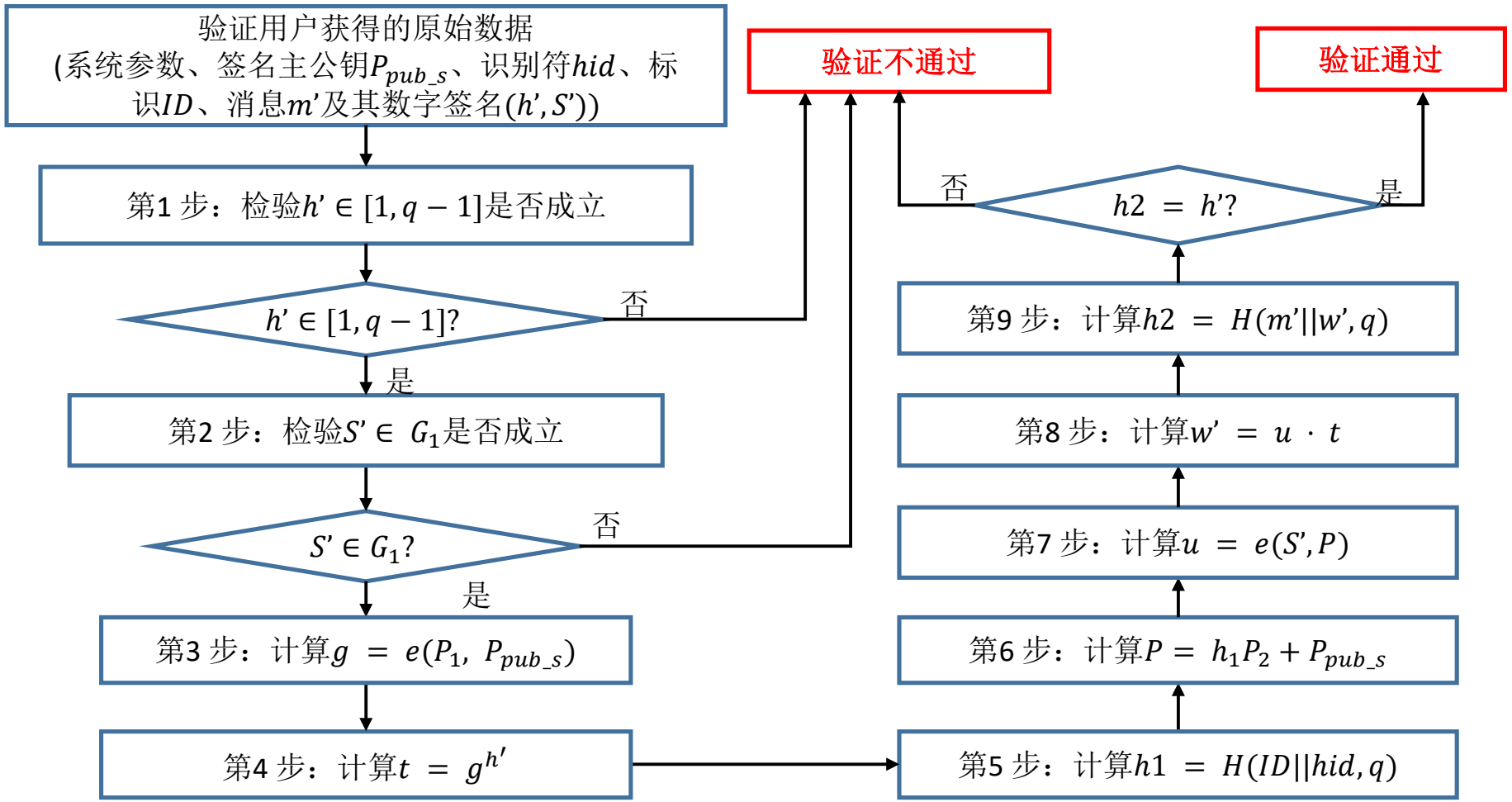
图6.24. SM9数字签名算法签名过程示意图

6.4 基于身份的数字签名算法

6.4.2 SM9数字签名算法

4. 验证

图6.25. SM9数字签名算法验证过程示意图



6.4 基于身份的数字签名算法

6.4.2 SM9数字签名算法

5. 曲线参数

椭圆曲线方程为：$y^2 = x^3 + b$

基域特征 q：B6400000 02A3A6F1 D603AB4F F58EC745 21F2934B 1A7AEEDB E56F9B27 E351457D

方程参数 b：05

群 G_1, G_2 的阶 N：B6400000 02A3A6F1 D603AB4F F58EC744 49F2934B 18EA8BEE E56EE19C D69ECF25

余因子 cf：1

嵌入次数 k：12

扭曲线的参数 β：$\sqrt{-2}$

群 G_1 的生成元 $P_1 = (x_{P1}, y_{P1})$：

坐标 x_{P1}：93DE051D 62BF718F F5ED0704 487D01D6 E1E40869 09DC3280 E8C4E481 7C66DDDD

坐标 y_{P1}：21FE8DDA 4F21E607 63106512 5C395BBC 1C1C00CB FA602435 0C464CD7 0A3EA616

6.4 基于身份的数字签名算法

6.4.2 SM9数字签名算法

5. 曲线参数

群 G_2 的生成元 $P_2 = (x_{P_2}, y_{P_2})$：

坐标 x_{P_2}：(85AEF3D0 78640C98 597B6027 B441A01F F1DD2C19 0F5E93C4 54806C11 D8806141,
37227552 92130B08 D2AAB97F D34EC120 EE265948 D19C17AB F9B7213B AF82D65B)

坐标 y_{P_2}：(17509B09 2E845C12 66BA0D26 2CBEE6ED 0736A96F A347C8BD 856DC76B 84EBEB96,
A7CF28D5 19BE3DA6 5F317015 3D278FF2 47EFBA98 A71A0811 6215BBA5 C999A7C7)

双线性对的识别符 *eid*：0x04

目　录

6.5 数字签名算法在区块链中的应用

比特币、以太币等数字货币均采用ECDSA算法保证交易的安全性.简单的说法是：用户利用私钥对交易信息进行签名，并把签名发给旷工，旷工通过验证签名确认交易的有效性.

- **比特币交易流程**

一笔交易信息的形成有输入和输出，输入是UTXO、解锁脚本(**包含付款人对本次交易的签名(<sig>)和付款人公钥(<PubK(A)>)**)、UTXO序号(来源的)，输出是发送数量、锁定脚本、UTXO序号(生成的).

其实交易的原理，就是使用原有的UTXO生成新的UTXO，所以输入输出都有UTXO序号，别搞混.然后脚本，有解锁脚本和锁定脚本，通常把解锁脚本和锁定脚本串联起来，才能用于验证交易的可行性.

交易的验证目的有两个：**1、输入的UTXO确实是付款人的；2、交易信息没有被篡改过.**

6.5 数字签名算法在区块链中的应用

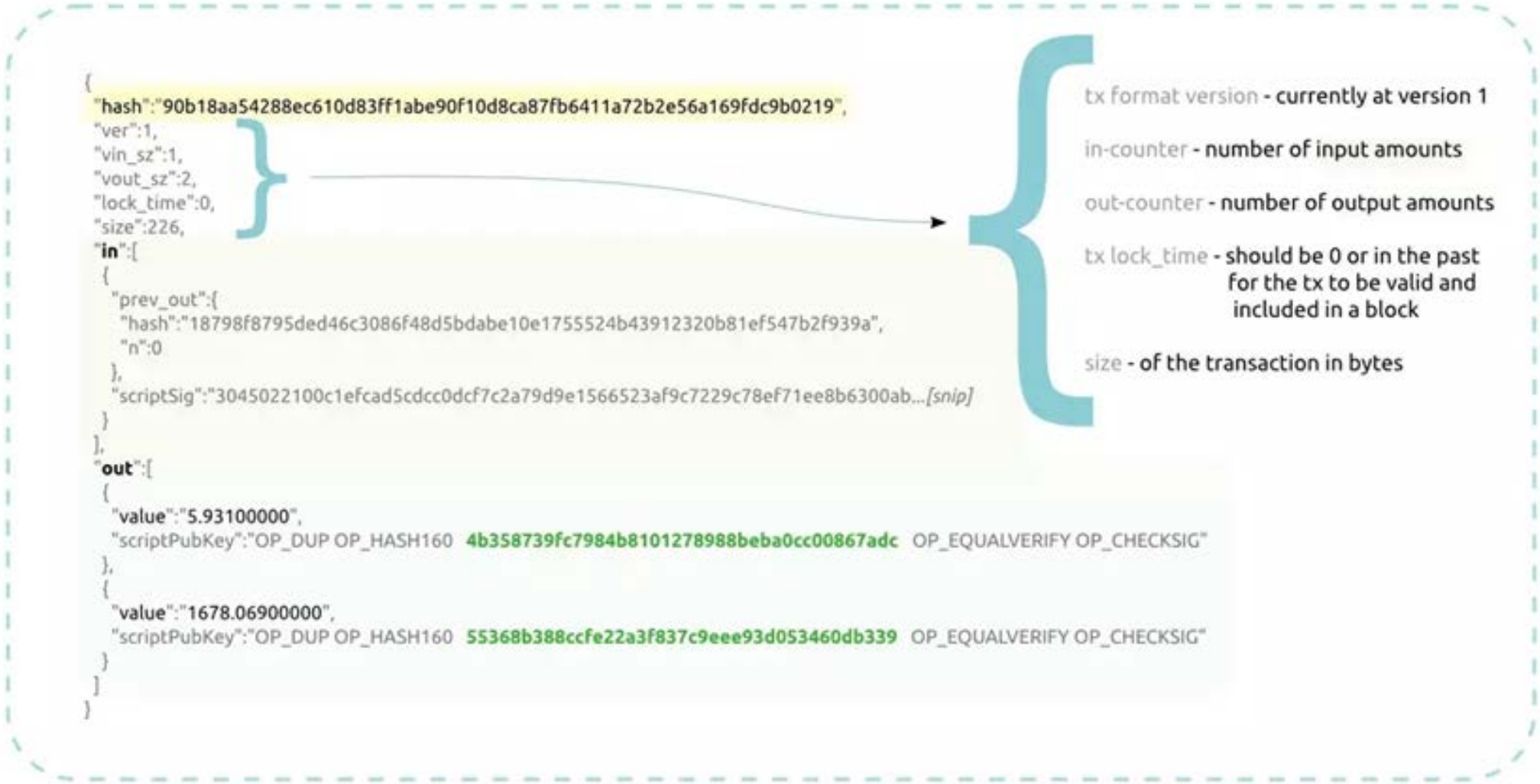


图6.26. 比特币交易信息示意图

6.5 数字签名算法在区块链中的应用

比特币使用基于ECDSA签名算法.选择的椭圆曲线为secp256k1，其中曲线方程为：$y^2 = x^3 + 7 \ \mathbf{mod}\ p$, 这里$p = 2^{256} - 2^{32} - 977$. 曲线的基点为$G$, 其中

x_G = 79BE667EF9DCBBAC55A06295CE870B07029BFCDB2DCE28D959F2815B16F81798

y_G = 483ADA7726A3C4655DA4FBFC0E1108A8FD17B448A68554199C47D08FFB10D4B8

G的阶为：

n = FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141

为什么会选择这条曲线？

6.5 数字签名算法在区块链中的应用

- 一般而言，曲线会被分成两类：“伪随机”曲线以及Koblitz曲线.
- 在一条伪随机曲线里，**参数*a*和*b*是从某个“种子”通过一个特定的伪随机数生成算法**来生成.
- 例如：对于secp256r1(这是标准256位伪随机曲线)来说，它的“种子”是c49d360886e704936a6678e1139d26b7819f7e90，产生的参数是：

  p = 115792089210356248762697446949407573530086143415290314195533631308867097853951

  a = 115792089210356248762697446949407573530086143415290314195533631308867097853948

  b = 41058363725152142129326129780047268409114441015993725554835256314039467401291
- **一个显眼的疑问：这个种子是怎么来的？**为何这个种子不是其他某个看起来更加单纯的数字，比如说15？
- 在斯诺登揭露的关于美国国家安全局(National Security Agency，NSA)密码标准的消息中，一个很重要的点就是说这个种子是以某种方式精心选择的，为了以某种只有NSA知道的方法来弱化这条曲线.

6.5 数字签名算法在区块链中的应用

- 因为哈希函数的特性，NSA不能先找到一条“弱”曲线然后再去确定种子；唯一的攻击途径是尝试不同的种子，直到最后有一个种子产生了一条“弱”曲线.
- 如果国安局只知道一个只能影响一条特定曲线的椭圆曲线的漏洞，那么伪随机数参数的产生流程将阻止他们把那个漏洞标准化推广到其他曲线.然而，**如果他们发现了一个通用的漏洞，那么那个流程也就不能提供保护了**.
- 比特币使用了Koblitz曲线，它不是伪随机曲线.**如果secp256r1是事实上的被NSA破解了**，那么因为比特币是为数不多的几个采用secp256k1而不是secp256r1的程序，比特币真的是躲过了一颗子弹.

谢谢！

补充：数字签名算法的可证明安全性

1. 可证明安全性的定义

可证明安全性是指这样一种“归约”方法：

- 首先确定密码体制的**安全目标**，例如，加密体制的安全目标是信息的机密性，签名体制的安全目标是签名的不可伪造性；
- 然后根据敌手的能力构建一个形式化的**安全模型**；
- 最后指出如果敌手能成功攻破密码体制，则**存在一种算法**在多项式时间内解决一个公认困难的数学问题或公认安全的密码体质.

总之，可证明安全性就是利用**反证法的思想**把密码算法/协议规约到一个数学难题或密码体质，通过后者的困难性或安全性来证明前者的安全.

补充：数字签名算法的可证明安全性

对于数字签名体制，存在以下3种伪造类型：

- **完全攻破**:敌手能够产生与私钥持有者相同的签名，这相当于恢复出了私钥；
- **选择性伪造**:敌手能够伪造一个他选择的消息的签名；
- **存在性伪造**:敌手能够伪造一个消息的签名，这个消息可能仅仅是一个随机比特串.

对于数字签名体制，存在以下2种伪造类型：

- 被动攻击：在被动攻击中，敌手被告知一个公钥，要求产生一个选择性伪造或存在性伪造.这是一种比较弱的攻击模型.
- 主动攻击：积极攻击中最强的攻击是适应性选择消息攻击(Adaptive Chosen Messages Attack)，即敌手可以访问一个签名预言机，它能够产生合法的签名.敌手的目标是产生一个消息的签名，当然这个消息不能是已经询问过签名预言机的消息.

补充：数字签名算法的可证明安全性

对于一个安全参数为k的密码体制，如果敌手成功的概率大于$1/p(k)$，我们就说这个敌手以一个不可忽略的概率成功，这里的p是一个以k为变量的多项式.

可证明安全性的目的在于证明：如果一个敌手能够攻破一个密码体制的某个安全概念，那么我们就可以利用该敌手做一些认为不可能的事情.我们假设敌手A是一个主动攻击敌手，即对于Schnorr签名算法，他能进行签名询问.

- 我们现在希望能够构造一个新算法B_A，它能够在输入一个离散对数问题实例和调用多项式次敌手A的情况下，以一个不可忽略的概率求解离散对数问题.
- 算法B_A说明了如果存在敌手A，就存在一个多项式时间离散对数问题求解算法，能够以一个不可忽略的概率解决离散对数问题.
- 既然我们目前并不相信存在这样的离散对数问题求解算法，我们也可以断定这样的敌手A是不存在的.

补充：数字签名算法的可证明安全性

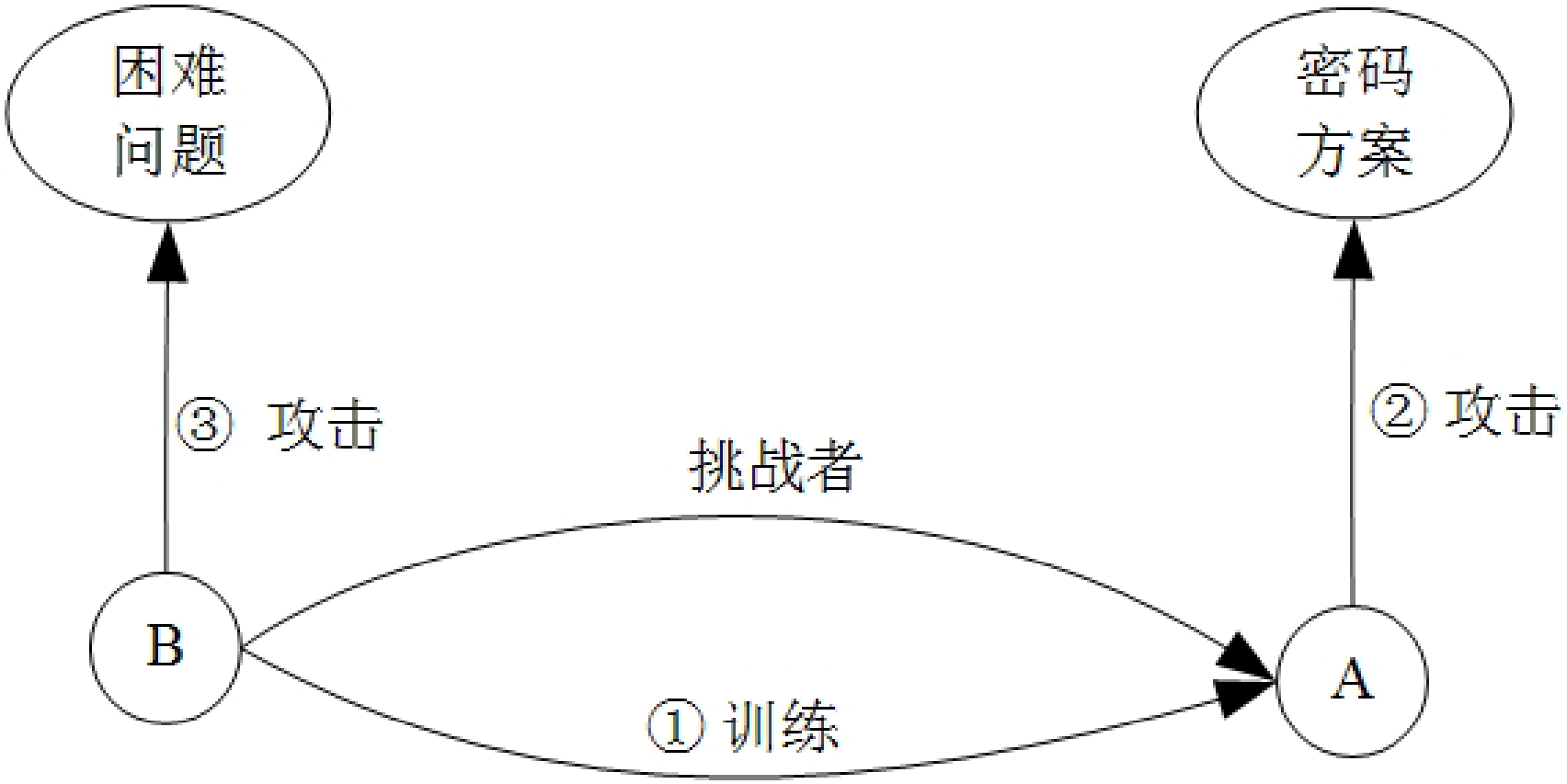


图6.27. 可证明安全流程示意图

补充：数字签名算法的可证明安全性

2. 随机谕言模型

可证明安全的思想就是给定一个算法A，我们提出一个新算法B_A，B_A把A作为子程序.输入给B_A的是我们希望解决的困难问题，输入给A的是某个密码算法. 然而，如果A是一个积极攻击敌手，即A可以对输入的公钥进行签名预言询问. 算法B_A要想使用A作为子程序，就需对A的询问提供回答. 算法B_A需要应对以下几个问题：

- 它的回答应该看起来是合法的. 因为加密应该能够解密，签名应该能够被验证，否则，算法A就知道它的预言机在撒谎.算法B_A就不能再确保算法A是以一个不可忽略的概率成功.
- 它的回答应该与如果预言机是真正的解密/加密预言机时A期望的回答具有相同的概率分布.
- 自始至终，预言机的回答应该是一致的.
- 算法B_A需要在**不知道私钥的情况下提供这些回答**.

补充：数字签名算法的可证明安全性

我们必须让B_A在不知道私钥的情况下能够者签名，但既然我们的体制是安全的，这一点意味着是不可能的. 为了回避这个问题，我们通常使用随机谕言模型.

- 随机谕言是一个理想的哈希函数. 对于每一个新的询问，随机预言产生一个随机值作为回答，如果进行**两次相同的询问，回答一定相同**.
- 在随机谕言模型中，我们假设敌手并**不使用密码算法中定义的那个哈希函数**，而是使用谕言机查询得到哈希运算的结果.
- 对于A的签名谕言询问，算法B_A是通过欺骗随机谕言的回答来适合自己的需要的.

补充：数字签名算法的可证明安全性

3. 分叉引理

如果算法B_A使用随机谕言运行敌手A一次，并成功获得一个消息m的两个签名.当算法B_A使用不同的随机谕言再运行敌手A一次时， B_A能够以一个不可忽略的概率获得这个消息m的另外一个签名.

David Pointcheval and Jacques Stern, "Security Proofs for Signature Schemes", *EUROCRYPT'96*, Saragossa, Spain, May 12–16, 1996, pp. 387–398.

4. Schnorr数字签名算法的安全性证明

假设p和 q是大素数，是q能被p-1整除，q是大于等于160 bit的整数，p是大于等于512 bit的整数，保证GF(p)中求解离散对数困难；g是GF(p)中元素，且$g^q \equiv 1 \bmod p$.

- **密钥生成：**

① Alice选择随机数x为私钥，其中$1<x<p$；

② Alice计算公钥$y \equiv g^x \pmod p$；

补充：数字签名算法的可证明安全性

- **签名算法**

① Alice首先随机数k，并计算$K= g^k \bmod p$，这里$1<k<q$;

② Alice计算$e = h(M, K)$

③ Alice计算$s = k - x \cdot e(\bmod q)$

④ Alice输出签名(e, s)

- **验证算法**

① Bob计算$g^k = g^s \cdot y^e \bmod p$

② Bob验证$e = h(M, g^k \bmod p)$，

如果相等则输出”Accept”，否则输出" Reject"

补充：数字签名算法的可证明安全性

定理. 在随机预言模型中，假设离散对数问题是一个困难问题，Schnorr数字签名算法在主动攻击下是安全的.

证明：假设敌手A能以不可忽略的概率ε伪造一个签名.我们将构造一个算法B_A解决计算理算对数问题.给定离散对数问题$y=g^x$，模拟器把y做为公钥交给敌手A，并回答的敌手A查询，主要包括哈希查询和签名查询.

说明：为了能够在积极攻击下提供证明，我们需要给出算法B_A是如何回答算法A的签名询问的.为了做到这一点，我们利用随机预言模型.我们利用算法B_A能够选择哈希函数输出的能力.

- ***H-查询***: 模拟器保存一个列表L_H，其格式为(M_i, K_i, h_i)，并初始化为空表.当收到一个查询(M_i, K_i)后，模拟器执行如下操作：
  - ✓ 检测L_H是否包含(M_i, K_i, h_i)，如果包含一个项目(M_i, K_i, h_i)，则返回h_i;
  - ✓ 否则，模拟器选择一个随机数h_i，设置$L_H = L_H \cup (M, K, u)$，这里$1< h_i <q$;

补充：数字签名算法的可证明安全性

- ***Sig-查询***: 给定消息M，模拟器执行如下操作来产生签名：

  ① 模拟器选择随机数s和u，并计算$K=g^s y^{-r}$，这里$1< r，s <q$

  ② 如果L_H是否包含$(M, K, *)$， 模拟器返回到步骤①.

  ③ 设置$L=L\cup(M，K, r)$，即当再输入$(M，K)$时，哈希谕言机总是回答u.

  ④ 输出签名(r, s).

若敌手A可以以一个不可忽略的概率伪造一个签名$(m, g^k, r, s= k - x \cdot r)$，则根据分叉引理，$A$可以输出另外一个签名$(m, g^k, r', s'= k - x \cdot r')$.

它们满足以下方程：

$$\begin{cases} g^k \equiv g^s y^r \bmod p \\ g^k \equiv g^{s'} y^{r'} \bmod p \end{cases}$$

补充：数字签名算法的可证明安全性

由于$y=g^x$，我们可以得到

$$\begin{cases} g^k \equiv g^s g^{x\cdot r} \equiv g^{s+xr} \bmod p \\ g^k \equiv g^{s'} g^{x\cdot r'} \equiv g^{s'+x\cdot r'} \bmod p \end{cases}$$

进而，我们可以得到

$$\begin{cases} k \equiv s + x \cdot r \bmod q \\ k \equiv s' + x \cdot r' \bmod q \end{cases}$$

解方程，我们可以得到$x = -\frac{s-s'}{r-r'} \bmod q$

补充：数字签名算法的可证明安全性

5. RSA-PSS数字签名算法的安全性证明

定理. 在随机预言模型中，假设离散对数问题是一个困难问题，RSA-PSS数字签名算法在主动攻击下是安全的.

➢ **密钥生成**

① 生成一个模数n，一个公钥e和一个私钥d.

② 假设安全参数为k(n是k比特的数)，我们定义两个整数k_0和k_1并且满足：$k_0+k_1\leq k-1$

③ 然后我们定义两个哈希函数：一个扩展数据$G:\{0,1\}^{k_1}\rightarrow\{0,1\}^{k-k_1-1}$，一个压缩数据$H:\{0,1\}^{*}\rightarrow\{0,1\}^{k_1}$.

补充：数字签名算法的可证明安全性

➢ 签名算法

为了对一个消息m进行签名，签名者执行以下步骤：

①生成一个随机数 $r \in \{0,1\}^{k_0}$

②计算$w=H(m\|r)$

③设置$y=0\|w\|(G_1(w)\oplus r)\|G_2(w)$

④计算$s=y^d \bmod n$

⑤消息m的签名为 s

➢ 验证算法

为了对一个消息m进行的签名s进行验证，验证者者执行以下步骤：

①计算$y=s^e \bmod n$

②将y分解成: $b\|w\|\ \alpha\ \|\ \gamma$，其中$b$的长度为1比特，$w$的长度为$k_1$比特，$\alpha$ 的长度为k_0比特,γ的长度为$k-k_0-k_1-1$比特

③计算$r=\alpha\oplus G_1(w)$

④当且仅当下列等式成立时，接受该签名：

$b=0$, $G_2(w)=\gamma$，且$w=H(m\|r)$