



区块链技术：密码学

Blockchain Technology: Cryptography

北京交通大学
计算机与信息技术学院
信息安全系

李超 (li.chao@bjtu.edu.cn)
段莉 (duanli@bjtu.edu.cn)



目录 | CONTENT

■ 引言

■ 区块链中的密码学原理

■ 密码学在区块链中的应用



引言



引言

利用对等网络和密码技术实现的密码货币系统，
交易账单**不可逆**，**不可伪造**，**不可否认**，**可验证**。



区块链用到了哪些密码学原理？

- 哈希算法与数字摘要
- 对称加密
- 非对称加密
- 数字签名
-



哈希算法



- 定义：任意长度的不同消息转化为长度相等但内容不同的二进制数列（由0和1组成）：哈希（hash）值

$$h = H(m)$$

原始数据

哈希函数

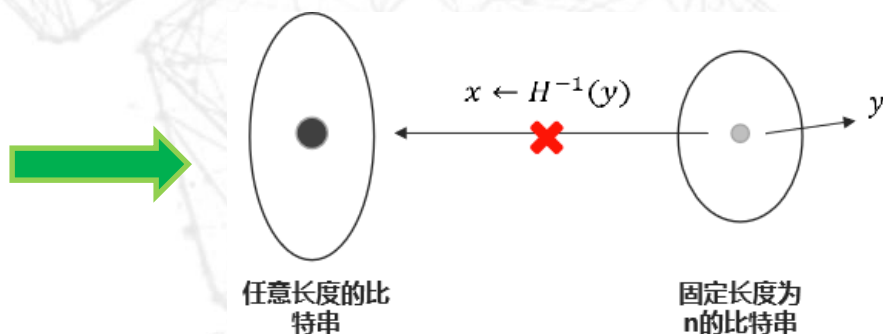
哈希值、摘要
散列、杂凑、指纹



哈希算法

- Hash特性：单向性

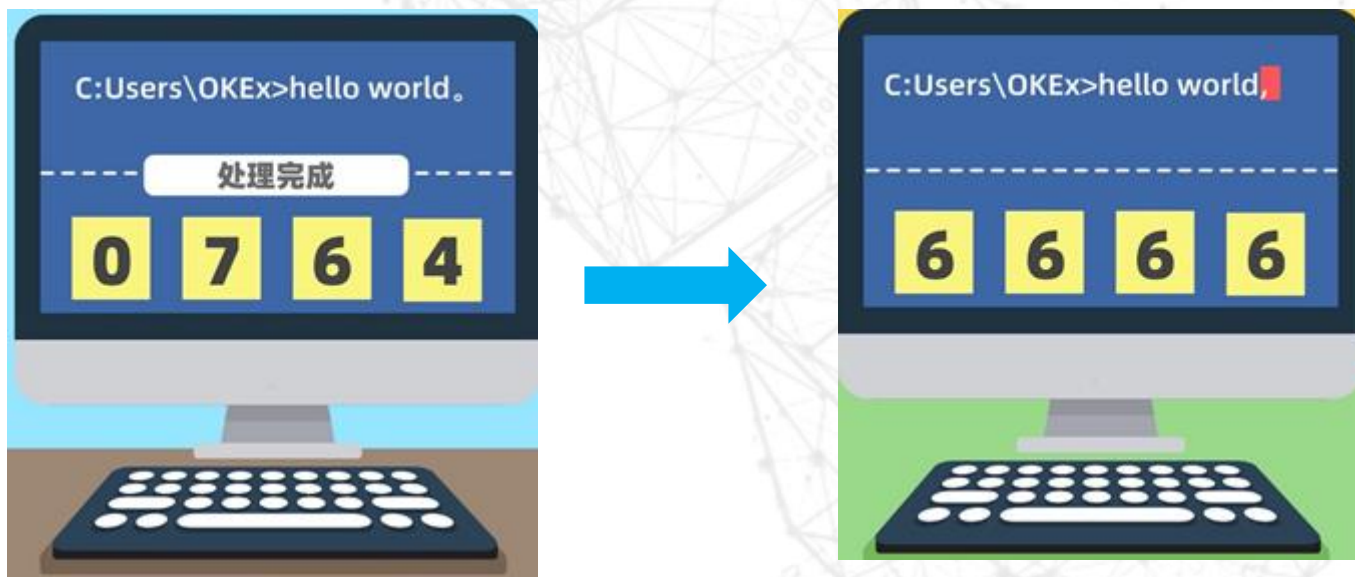
- 可以操作任何大小的报文 m （实际上SHA-1要求不超过 2^{64} ）
- 给定任意长度的 m ，产生的 h 的长度固定
- 给定 m ，计算 $h=H(m)$ 是容易的
- 给定 h 寻找 m ，使得 $H(m)=h$ 是困难的（单向性：one-way）



哈希算法

- Hash特性：抗篡改能力

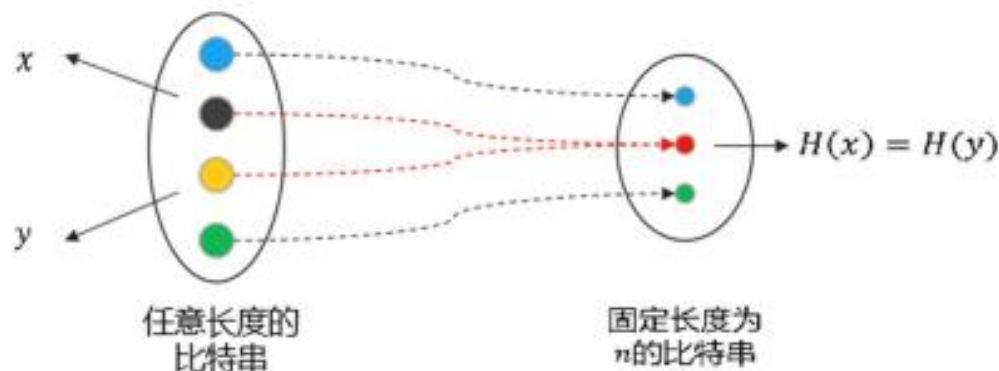
- 改变消息 m 中的任何一个bit或者几个bits都会使 $H(m)$ 发生改变



哈希算法

- Hash特性：抗碰撞能力

- 给定 m ，要找到 m' ， $m' \neq m$ 且 $H(m) = H(m')$ 是计算上不可行的（弱抗碰撞：weakly collision-free）；
- 寻找任何 (x, y) ， $x \neq y$ ，使得 $H(x) = H(y)$ 是计算上不可行的（强抗碰撞：strongly collision-free）



哈希算法

- 纵向的奇偶校验码

	比特1	比特2	比特n
分组1	$\mathbf{b_{11}}$	$\mathbf{b_{21}}$	$\mathbf{\dots}$	$\mathbf{b_{n1}}$
分组2	$\mathbf{b_{12}}$	$\mathbf{b_{22}}$	$\mathbf{\dots}$	$\mathbf{b_{n2}}$
	$\mathbf{\cdot}$	$\mathbf{\cdot}$	$\mathbf{\dots}$	$\mathbf{\cdot}$
分组m	$\mathbf{b_{1m}}$	$\mathbf{b_{2m}}$		$\mathbf{b_{nm}}$
散列码	$\mathbf{C_1}$	$\mathbf{C_2}$		$\mathbf{C_n}$

$$C_i = b_{i1} \oplus b_{i2} \dots \oplus b_{im}$$

哈希算法

- 几种常见的哈希算法:

- MD4/MD5

- ◆ Rivest与1990/1991年设计
 - ◆ 输出: 128比特的哈希值

- SHA算法

- ◆ SHA-1、SHA-256、SHA-384、SHA-512
 - ◆ NIST (美国国家标准技术研究所) 与RSA (美国国家安全局) 设计

- RIPEMD-160

- ◆ 1996年 Hans Dobbertin、Antoon Bosselaers、Bart Pareeneel

<https://tool.oschina.net/encrypt?type=2>

安全性：HASH 碰撞

- 2004 年国际密码学会议（Crypto' 2004）
 - 王小云教授在Crypto' 2004上做破译 **MD5**、**HAVAL-128**、**MD4** 和 **RIPEMD** 算法的报告，令国际顶尖密码学专家都为之震惊
 - 该次会议的总结报告写道：“我们该怎么办？MD5被重创了，它即将从应用中淘汰。SHA-1仍然活着，但也看到了它的末日，现在就得开始更换SHA-1了”



王小云教授

SHA1碰撞的理论计算上界

- 2005年2月7日，美国国家标准技术研究院NIST对外宣称：SHA-1还没有被攻破，并且也没有足够的理由怀疑它会很快被攻破
- **但仅在一周之后，王小云教授再度令世界密码学界大跌眼镜——SHA-1也被她攻破了！王教授提出了复杂度为 2^{63} 的理论攻击**



安全性：HASH 碰撞

- 2017年2月23号，Google在其安全博客上公布了其找到了世界首例的SHA1碰撞，标志着SHA1不再安全了

Collision Attack: Two Different Documents, But Same SHA-1 Hash Fingerprint

SHAttered

The first concrete collision attack against SHA-1
<https://shattered.io>





Marc Stevens
Pierre Karpman

Elie Bursztein
Ange Albertini
Yarik Markov

SHAttered

The first concrete collision attack against SHA-1
<https://shattered.io>



Marc Stevens
Pierre Karpman

Elie Bursztein
Ange Albertini
Yarik Markov

```
sha1sum *.pdf
38762cf7f55934b34d179ae6a4c80cadccbb7f0a 1.pdf
38762cf7f55934b34d179ae6a4c80cadccbb7f0a 2.pdf
/tmp/sha1
sha256sum *.pdf
2bb787a73e37352f92383abe7e2902936d1059ad9f1ba6daaa9c1e58ee6970d0 1.pdf
```

0.64G 8-11h

针对Hash函数的攻击

● 穷举攻击

① 原像攻击和第二原像攻击(Preimage or Second Preimage attack)

- 对于给定的哈希值 h , 试图找到满足 $H(x)=h$ 的 x ;
- 对于 m 位的哈希值, 穷举规模大约是 2^m .

② 碰撞攻击(Collision Resistance)

- 找到两个消息 $x \neq y$, 满足 $H(x) = H(y)$;
- 对于 m 位的哈希值, 预计在 $2^{m/2}$ 次尝试后就将找到两个具有相同哈希值的数据.

③ 因此对于 m 位的哈希值, 抗穷举攻击的强度为 $2^{m/2}$

- 目前128比特已经不够, 需要160比特甚至更多.

比特币使用的哈希算法

- SHA256

- ◆ 输出：256位的哈希值
- ◆ 用途：区块的头部信息、交易数据、工作量证明、比特币地址等

- RIPEMD160

- ◆ 输出：160位的哈希值
- ◆ 用途：比特币地址生成

SHA256简介

	SHA-1	SHA-256	SHA-384	SHA-512
Hash码长度	160	256	384	512
消息长度	$<2^{64}$	$<2^{64}$	$<2^{128}$	$<2^{128}$
分组长度	512	512	1024	1024
字长度	32	32	64	64
迭代步骤数	80	64	80	80
安全性	80	128	192	256

注: 1. 所有的长度以比特为单位.

2. 安全性是指对输出长度为 n 比特哈希函数的生日攻击产生碰撞的工作量大约为 $2^{n/2}$

SHA256简介

□ 算法描述

- 输入数据长度为 l 比特, $1 \leq l \leq 2^{64}-1$;
- 输出哈希值的长度为256比特.

(1) 常量与函数

SHA-256算法使用以下常数与函数:

①常量

初始值 $IV = 0x6a09e667\ bb67ae85\ 3c6ef372\ a54ff53a\ 510e527f\ 9b05688c\ 1f83d9ab\ 5be0cd19$, 这些初值是对自然数中前8个质数(2,3,5,7,11,13,17,19)的平方根的小数部分取前32比特而来.

哈希算法SHA256

信息的预处理：附加填充比特和附加长度。

- Step 1: 附加填充比特
 - ◆ 对消息进行填充，使其长度为： $448 \bmod 512$
 - ◆ 填充比特串的最高位为1，其余位为0

哈希算法SHA256

□ Step 2: 附加长度值

◆ 用64bit表示的初始消息附加在step1的结果之后（低位字节优先）

◆ 如：“abc” 的长度为 $8 \times 3 = 24\text{bit}$ ，“abc” 的

的二进制编码为：01100001 01100010 01100011.

① 补一个“1”：01100001 01100010 01100011 **1**

② 补423个“0”：01100001 01100010 01100011 **10000000 00000000 ... 00000000**

③ 补比特长度24 (64位表示)，得到512比特的数据：

01100001 01100010 01100011 **100** 00 00 ... 011000

423比特 64比特

哈希算法SHA256

□ Step 3: 初始化缓存

- ◆ 8个32 bit的寄存器
- ◆ 存放散列函数的中间值及最终结果

□ Step 4: 处理消息分组序列

- ◆ 将消息M分成N个512 bit的块:
 $M^{(1)}, M^{(2)}, \dots, M^{(i)}, \dots, M^{(N)}$
- ◆ 对于每一个 $M^{(i)}$ 的处理由64步迭代完成
- ◆ 每个消息块分成16个32bit的字段, 标记为
 $M^{(i)0}, M^{(i)1}, M^{(i)2}, \dots, M^{(i)15}$

$$H_1^{(0)} = 6a09e667$$

$$H_2^{(0)} = bb67ae85$$

$$H_3^{(0)} = 3c6ef372$$

$$H_4^{(0)} = a54ff53a$$

$$H_5^{(0)} = 510e527f$$

$$H_6^{(0)} = 9b05688c$$

$$H_7^{(0)} = 1f83d9ab$$

$$H_8^{(0)} = 5be0cd19$$

哈希算法SHA256

□ Step 4: 处理消息分组序列

- ◆ 使用了6种基本逻辑函数
- ◆ 每一步都以256bit缓存作为输入

然后更新缓存内容

- ◆ 每步使用一个32-bit 常数值 K_t 和一个32-bit W_t 初值是消息块,

递推公式为 $W_t = \sigma_1(W_{t-2}) + W_{t-7} + \sigma_0(W_{t-15}) + W_{t-16}$

\oplus	bitwise XOR
\wedge	bitwise AND
\vee	bitwise OR
\neg	bitwise complement
$+$	mod 2^{32} addition
R^n	right shift by n bits
S^n	right rotation by n bits

$$\begin{aligned}Ch(x, y, z) &= (x \wedge y) \oplus (\neg x \wedge z) \\ Maj(x, y, z) &= (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) \\ \Sigma_0(x) &= S^2(x) \oplus S^{13}(x) \oplus S^{22}(x) \\ \Sigma_1(x) &= S^5(x) \oplus S^{11}(x) \oplus S^{25}(x) \\ \sigma_0(x) &= S^7(x) \oplus S^{18}(x) \oplus R^3(x) \\ \sigma_1(x) &= S^{17}(x) \oplus S^{19}(x) \oplus R^{10}(x)\end{aligned}$$

```
const UInt32 K[64] = {
```

```
    0x428a2f98, 0x71374491, 0xb5c0fbcf, 0xe9b5dba5, 0x3956c25b, 0x59f111f1, 0x923f82a4, 0xab1c5ed5,  
    0xd807aa98, 0x12835b01, 0x243185be, 0x550c7dc3, 0x72be5d74, 0x80deb1fe, 0x9bdc06a7, 0xc19bf174,  
    0xe49b69c1, 0xefbe4786, 0x0fc19dc6, 0x240ca1cc, 0x2de92c6f, 0x4a7484aa, 0x5cb0a9dc, 0x76f988da,  
    0x983e5152, 0xa831c66d, 0xb00327c8, 0xbf597fc7, 0xc6e00bf3, 0xd5a79147, 0x06ca6351, 0x14292967,  
    0x27b70a85, 0x2e1b2138, 0x4d2c6dfc, 0x53380d13, 0x650a7354, 0x766a0abb, 0x81c2c92e, 0x92722c85,  
    0xa2bfe8a1, 0xa81a664b, 0xc24b8b70, 0xc76c51a3, 0xd192e819, 0xd6990624, 0xf40e3585, 0x106aa070,  
    0x19a4c116, 0x1e376c08, 0x2748774c, 0x34b0bcb5, 0x391c0cb3, 0x4ed8aa4a, 0x5b9cca4f, 0x682e6ff3,  
    0x748f82ee, 0x78a5636f, 0x84c87814, 0x8cc70208, 0x90beffffa, 0xa4506ceb, 0xbef9a3f7, 0xc67178f2,
```

```
};
```

哈希算法SHA256

□ Step 4: 处理消息分组序列

- ◆ 使用了6种基本逻辑函数
- ◆ 每一步都以256bit缓存作为输入
然后更新缓存内容
- ◆ 每步使用一个32-bit 常数值 K_t 和一个32-bit W_t

□ Step 5: 获得最终结果

- ◆ 所有的512 bit分组处理完毕后
，输出的结果即为最终结果

\oplus bitwise XOR

\wedge bitwise AND

\vee bitwise OR

\neg bitwise complement

$+$ mod 2^{32} addition

R^n right shift by n bits

S^n right rotation by n bits

$$Ch(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z)$$

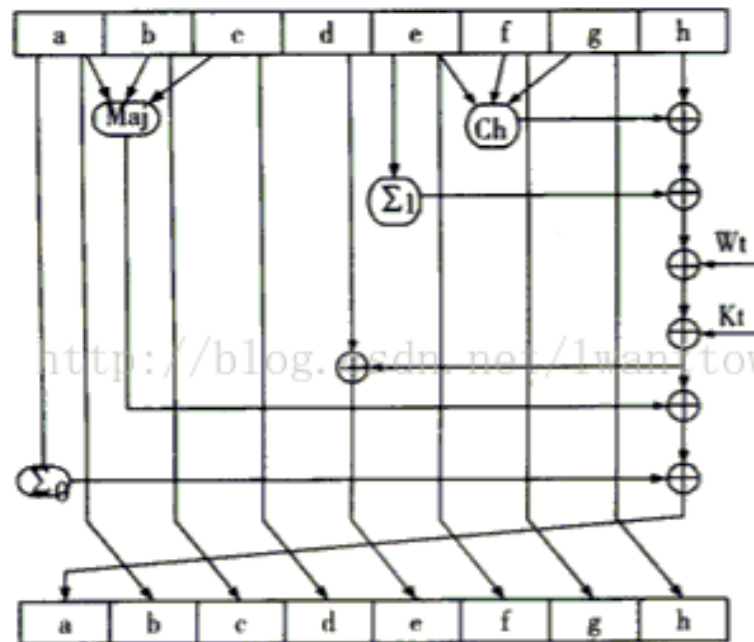
$$Maj(x, y, z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$$

$$\Sigma_0(x) = S^2(x) \oplus S^{13}(x) \oplus S^{22}(x)$$

$$\Sigma_1(x) = S^5(x) \oplus S^{11}(x) \oplus S^{25}(x)$$

$$\sigma_0(x) = S^7(x) \oplus S^{18}(x) \oplus R^3(x)$$

$$\sigma_1(x) = S^{17}(x) \oplus S^{19}(x) \oplus R^{10}(x)$$



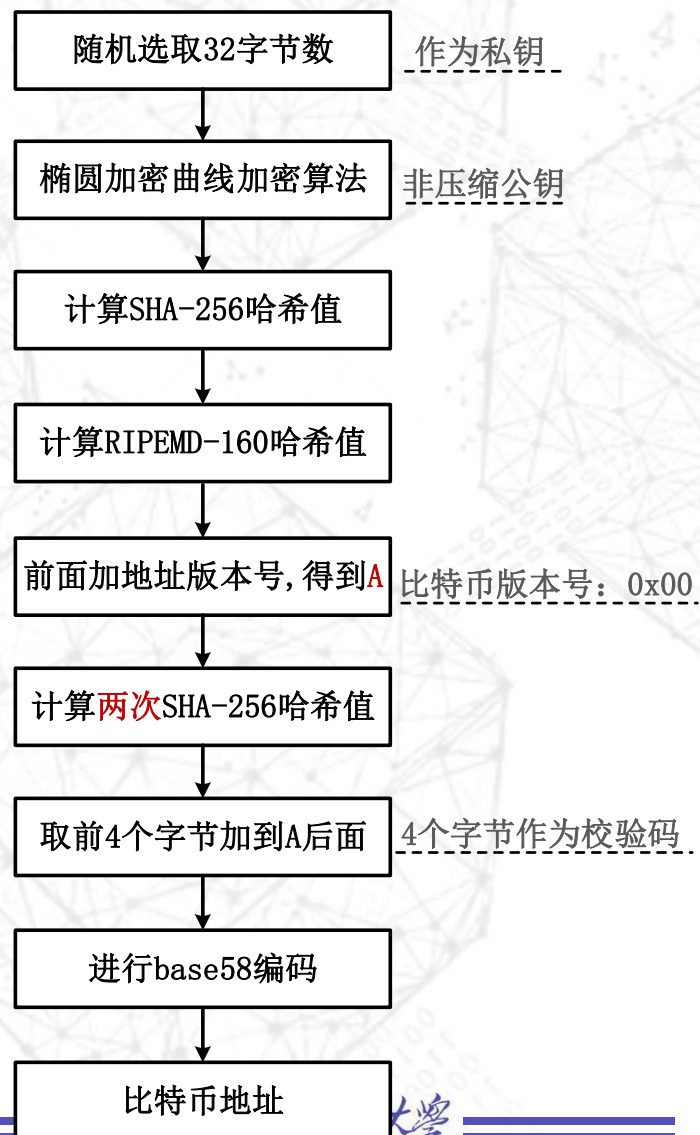
哈希算法SHA256

- 安全性
 - 专业机构设计，经过充分测试和论证；
 - 安全性可满足应用的安全需求；
 - 学者已开展对SHA-256的安全分析(如缩减轮的分析)，尚未发现本质的缺陷；

哈希算法

● 比特币地址的生成:

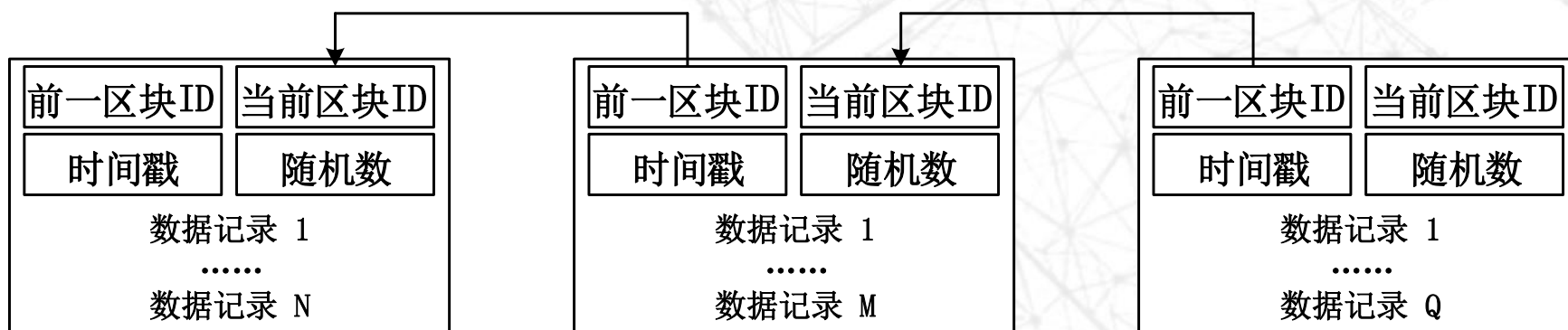
1. 随机选取32字节的数作为私钥
2. 椭圆曲线加密算法得到非压缩公钥
3. 依次计算SHA-256、RIPEMD-160值
4. 上一结果前加地址版本号 (0x00)
5. 计算两次SHA-256
6. 取前4个字节加到第4步结果后面
7. 进行base58编码
8. 获得比特币地址



哈希算法

● 哈希指针链：

- ◆ 基于SHA-256的哈希值是将众多区块连接成链的保障
- ◆ 每个区块的ID其实就是基于SHA-256得到的哈希值
- ◆ 每个区块会记录前一个区块链的ID（SHA-256哈希值），从而连接成链



哈希算法

- 工作量证明 (PoW) :

- ◆ 比特币挖矿：不断尝试随机数，直至符合要求
- ◆ 基于SHA-256的哈希运算
- ◆ 比特币：每10分钟，难度可调整

碰撞数	数据	Hash256
458369	hellobitcoin458369	000009f5450b2839cd7653ddfa4253e67c922bc920c1df8b7263a20be1e7a4ef
2006547	hellobitcoin2006547	00000a6878599eb57fca3fa3b893e3a7a38ff4daff260df6d108efb0679537fb
2392842	hellobitcoin2392842	00000e06b947ba5ccea89d04f0a821750bcb5f04f3b7b3760507195bb7b71cc4
4793265	hellobitcoin4793265	0000022e49c81659bf92244fda4faace18d2628693abc635726b476bb71de562
5747821	hellobitcoin5747821	0000065cdf7084be7278ebe4c452f7e6a56cbd846756e6b103c4d7aa30853a96
6506263	hellobitcoin6506263	00000509861722709f292cb78a1f59ccd3c305b03f737dc02c1a10935bfb6e7c

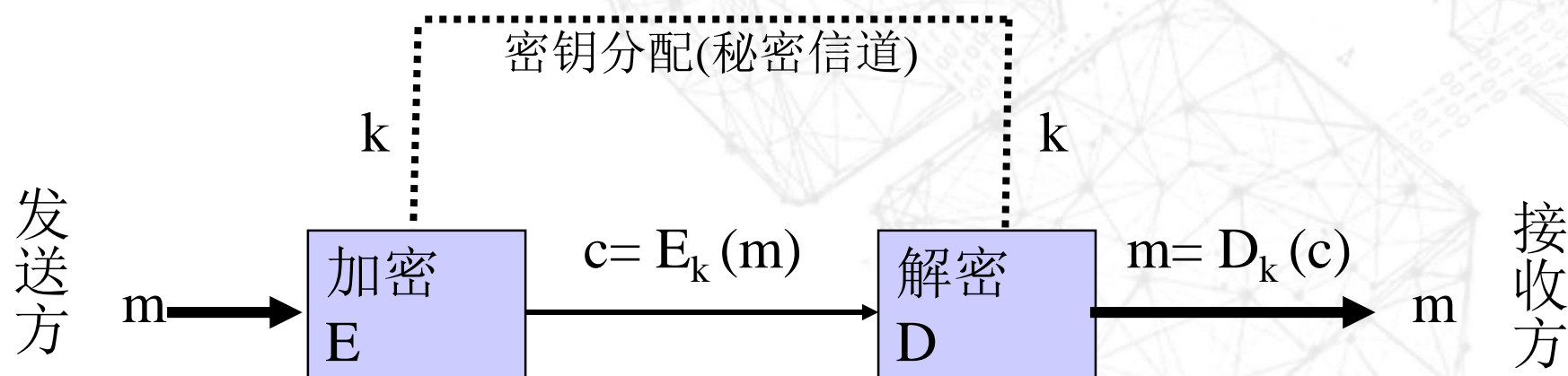
- 对称加密算法
- 非对称加密算法

准备知识

- 冗余函数
- 整数因子分解问题
- 计算两个整数最大公因子的欧几里得算法
- 扩展的欧几里得算法
- 模 n 平方根的困难性
- 求 n 模素数 p 的平方根
- 模 n 的二次剩余集
-

对称密钥算法

- 加密和解密使用**相同的密钥**: $K_E = K_D$
- 密钥必须使用**秘密的信道**分配



对称加密技术

- 常用对称密钥密码算法
 - DES (Data Encryption Standard)及其各种变体
 - IDEA (International Data Encryption Algorithm)
 - RC2, RC4, RC5
 - AES (Advanced Encryption Standard)
 - CAST-128
 - Blowfish
- 应用的最广泛的是**DES**加密算法。

数据加密标准 DES

- 数据加密标准 DES 属于对称密钥密码体制，是一种**分组密码**。
- 在加密前，先对整个明文进行分组，每一个组长长度为 **64** 位。



数据加密标准 DES

DES 是世界上第一个公认的
实用密码算法标准。

DES 的保密性仅取决于对**密钥**的保密，而**算法是公开**的。

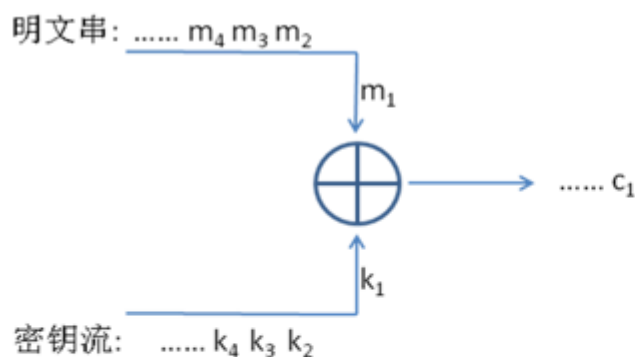
穷举搜索密钥至今仍是破译
DES最有效的方法。

目前较为严重的问题是 DES
的**密钥的长度**。

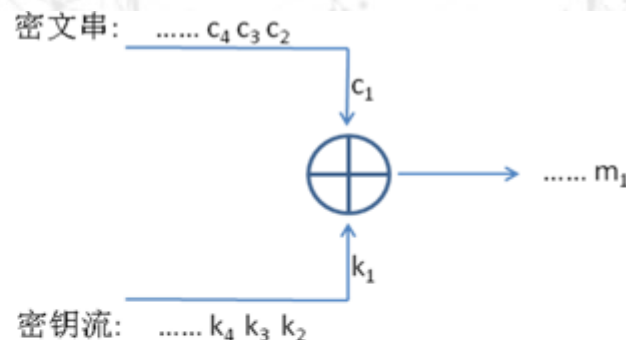
现在已经设计出来搜索 DES
密钥的**专用芯片**。

序列密码

- 主要原理：通过有限状态机产生性能优良的伪随机序列，使用该序列加密信息流，通过逐位加密，得到密文序列。
- 对于序列密码攻击的手段有代数方法和概率统计方法，两者结合可以达到较好的效果。



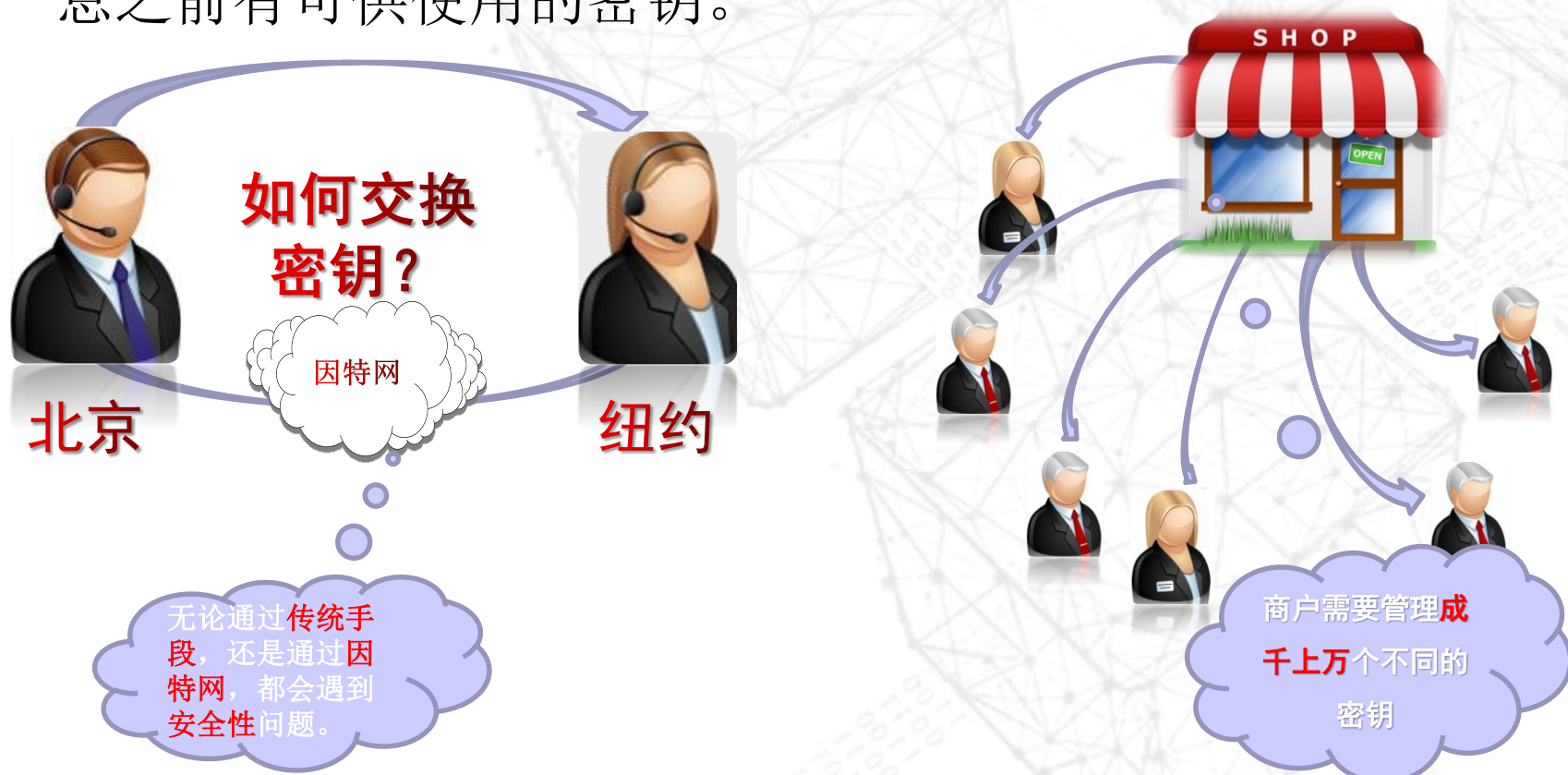
简单流密码
加密结构



解密结构

对称加密技术

- 加密体制的局限性在于，在发送和接收方传输数据时必须先通过安全渠道交流密钥，保证在他们发送或接收加密信息之前有可供使用的密钥。



非对称加密技术

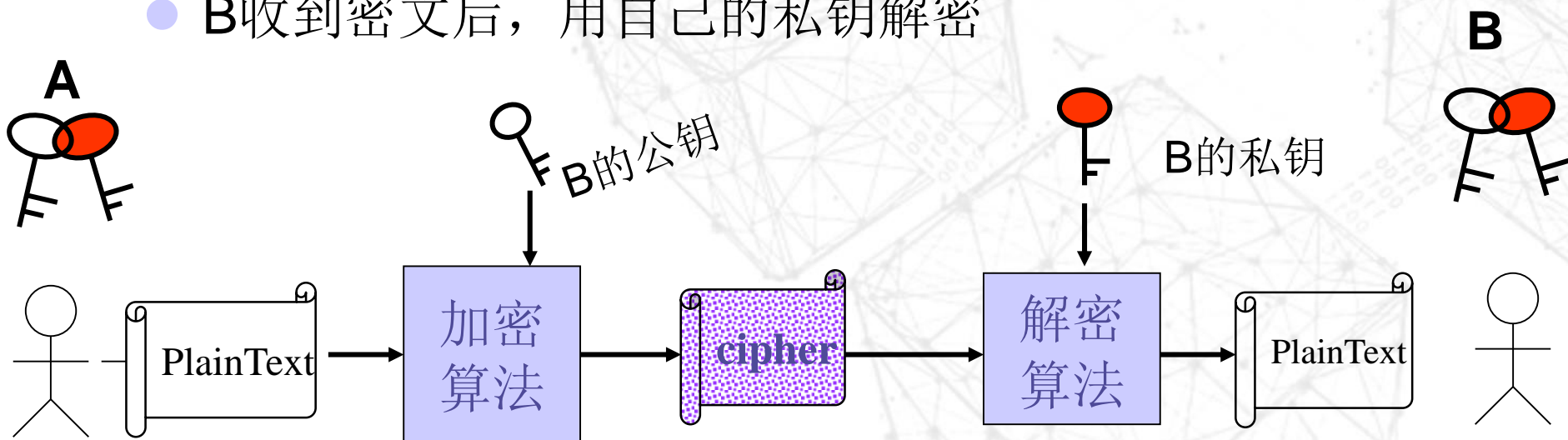
- 非对称密钥体制，也即公钥密钥体制，由Diffie, Hellman, Merkel 1976年提出，获得2015年图灵奖



- 使用**两个**密钥，对于密钥分配、数字签名、认证等有深远影响
- 基于**数学函数**而不是代替和换位，密码学历史上唯一的一次真正的革命

公钥密码系统的加密原理

- 每个通信实体有一对密钥(公钥, 私钥)
 - 公钥公开, 用于加密和验证签名, 私钥保密, 用作解密和签名
- A向B 发送消息, 用B的公钥加密
- B收到密文后, 用自己的私钥解密



任何人向B发送信息都可以使用同一个密钥(B的公钥)加密
没有其他人可以得到B的私钥, 所以只有B可以解密

加解密类型

算法类型	特点	优势	缺陷	代表算法
对称加密	加解密的密钥相同	计算效率高，加密强度高	需提前共享密钥，容易泄露	DES,3DES, AES, IDES
非对称加密	加解密的密钥不相关	无需提前共享密钥	计算效率低，仍存在中间人攻击可能	RSA,ElGamal, 椭圆曲线系列算法

公钥密码算法的表示

- 对称密钥密码

- 密钥：会话密钥(K_s)

- 加密函数： $C = E_{K_s}[P]$

- 对密文 C ，解密函数： $D_{K_s}[C]$

- 公开密钥

- (K_{Ua} , K_{Ra})

- 加密/签名： $C = E_{K_{Ub}}[P], E_{K_{Ra}}[P]$

- 解密/验证： $P = D_{K_{Rb}}[C], D_{K_{Ua}}[C]$

基本思想和要求

- 涉及到各方：发送方、接收方、攻击者
- 涉及到数据：公钥、私钥、明文、密文
- 公钥算法的条件：
 - 产生一对密钥是计算可行的
 - 已知公钥和明文，产生密文是计算可行的
 - 接收方利用私钥来解密密文是计算可行的
 - 对于攻击者，利用公钥来推断私钥是计算不可行的
 - 已知公钥和密文，恢复明文是计算不可行的
 - (可选)加密和解密的顺序可交换

其中最后一条虽然非常有用，但不是对所有的算法都作要求。

公钥密码的理论基础

- 难解问题：没有有效算法，求解所需时间非常长。
- 易解问题：存在有效算法，求解所需时间短
- 一般来说计算一个难解的问题所需要的时间通常是所输入数据长度的一个指数函数，因此计算时间是随着数据长度的增加急剧增加的。

陷门单向函数

- 公钥密码是建立在陷门单向函数上的。
- 单向函数（one way function）：
 - (1) 给定 x ，计算 $y=f_k(x)$ 是容易的；
 - (2) 给定 y ，计算 x 使 $y=f_k^{-1}(x)$ 是不可行的。
- 陷门单向函数（trapdoor one way function）：
 - (1) 给定 x ，计算 $y=f_k(x)$ 是容易的；
 - (2) 给定 y ，计算 x 使 $y=f_k^{-1}(x)$ 是不可行的。
 - (3) 存在 k ，已知 k 时，对给定的任何 y ，若相应的 x 存在，则计算 x 使 $y=f_k^{-1}(x)$ 是容易的。
- 研究公钥密码算法就是要找出合适的陷门单向函数。

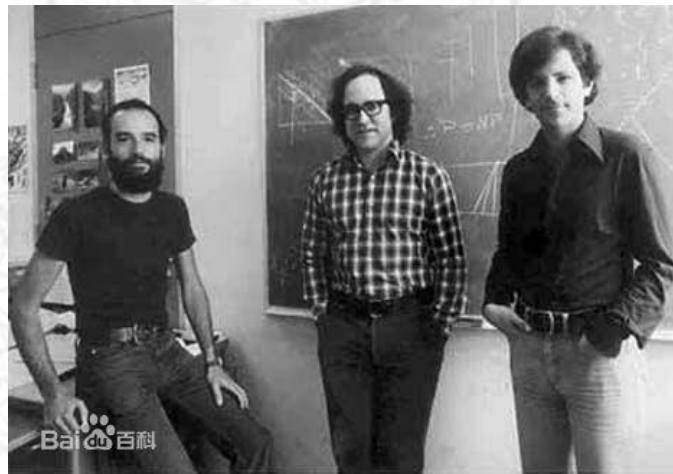
公钥密码的应用范围

- 加密/解密（保密性）
- 数字签名(身份认证)
- 密钥交换（会话密钥）

Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffe-Hellman	No	No	Yes
DSS	No	Yes	No

RSA算法简介

- Ron Rivest, Adi Shamir, Leonard Adleman
 - 获得2002年图灵奖
- RSA的安全性基于大数分解的难度
- RSA在美国申请了专利(已经过期), 在其他国家没有
- RSA已经成了事实上的工业标准, 在美国除外



RSA算法操作过程

● 密钥产生

1. 取两个大素数 p, q , 保密
2. 计算 $n=pq$, 公开 n
3. 计算欧拉函数 $\phi(n) = (p-1)(q-1)$
4. 任意取一个与 $\phi(n)$ 互素的小整数 e , 即
 $\gcd(e, \phi(n))=1; 1 < e < \phi(n)$
5. 寻找 d , $d < \phi(n)$, 使得 $de \equiv 1 \pmod{\phi(n)}$, $de = k\phi(n) + 1$

公开 $(e, n) = (5, 119)$
将 d 保密, 丢弃 p, q

$$p=7, q=17$$

$$n=119$$

$$\phi(n)=96$$

$$\text{选择 } e=5$$

$$5d = k \times 96 + 1$$

$$\text{令 } k=4, \text{ 得到 } d=77$$

RSA 算法加密/解密过程

- 公开密钥: $KU=\{e, n\}$ $\{5, 119\}$
- 秘密密钥: $KR=\{d, n\}$ $\{77, 119\}$

- 加密过程

- 把待加密的内容分成k比特的分组

- $k \leq \log_2 n$, 并写成数字, 设为M, 则 $c = m^e \bmod 119$

$$C = M^e \bmod n$$

- 解密过程

$$m = c^{77} \bmod 119$$

$$M = C^d \bmod n$$

Remember:

$$ed = k\phi(n) + 1$$

$$M^{k\phi(n)+1} \equiv M \bmod n$$



RSA算法的安全性

- 攻击方法
 - 蛮力攻击：对所有密钥都进行尝试
 - 数学攻击：等效于对两个素数乘积(n)的因子分解
- 大数的因子分解是数论中的一个难题

因子分解的进展

十进制数字位数	近似比特数	得到的数据	MIPS 年
100	332	1991	7
110	365	1992	75
120	398	1993	830
129	428	1994	5000
130	431	1996	500

RSA算法的性能

- 速度
 - 软件实现比DES慢100倍
 - 硬件实现比DES慢1000倍

8位公开密钥的 RSA的加密速度	512位	768位	1024位
加密(秒)	0.03	0.05	0.08
解密(秒)	0.16	0.48	0.93
签名(秒)	0.16	0.52	0.97
验证(秒)	0.02	0.07	0.08

公钥密码体制的安全性

- 像对称密码体制一样，密钥穷搜索攻击理论上是有效的；
 - 但是使用的密钥太大 ($>512\text{bits}$)
- 安全性依赖于难解问题；
- 一般难解问题是已知的，但是需要设计的足够难；
- 需要使用很大的数；
- 因此比对称密码算法要慢；

数字签名

- 数字签名的基本概念
- 基于椭圆曲线的数字签名算法
- 数字签名算法在区块链中的应用

数字签名的产生



重要证书、证件采用的防
伪技术是使用特殊材料或
者信息隐藏等

数字签名

1. 否认
2. 伪造
3. 冒充
4. 篡改



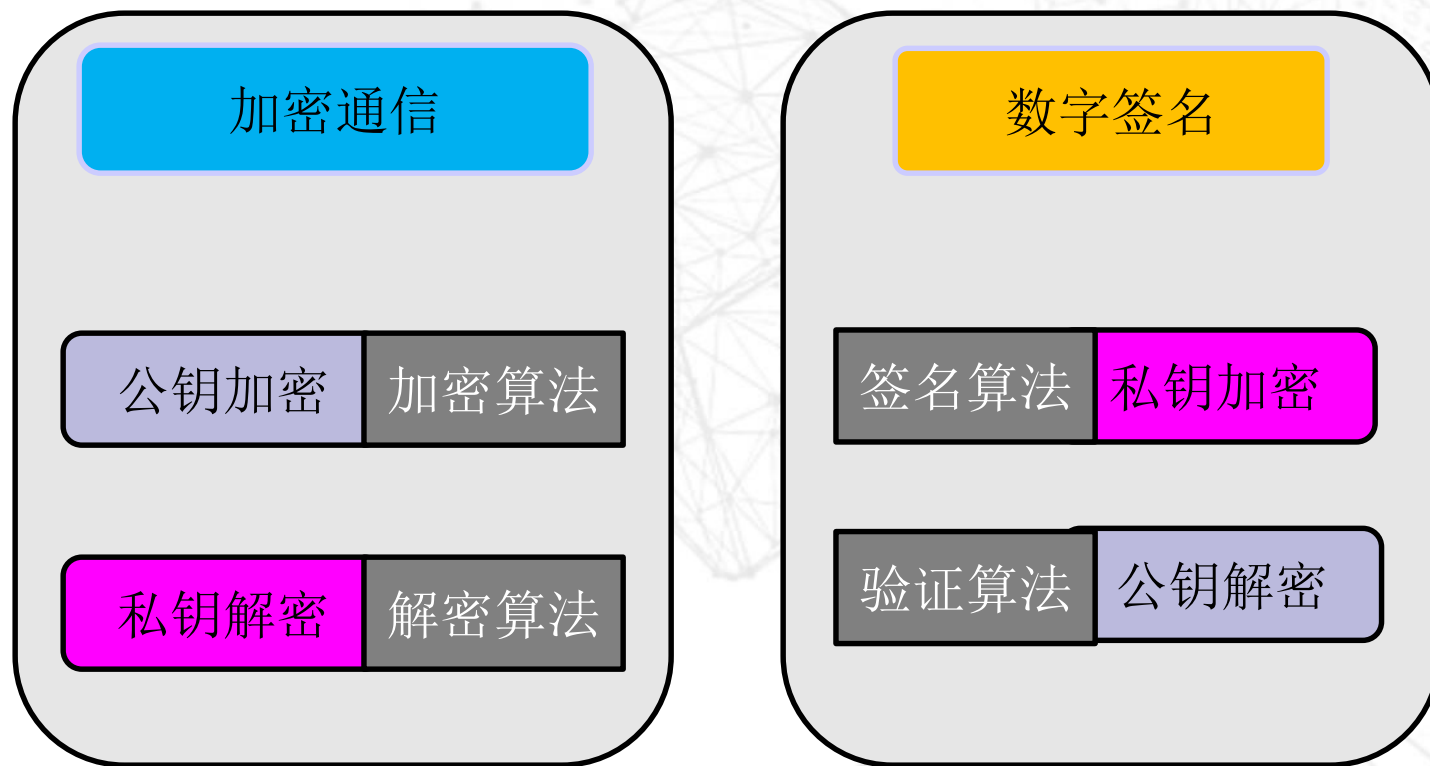
网络

数字签名

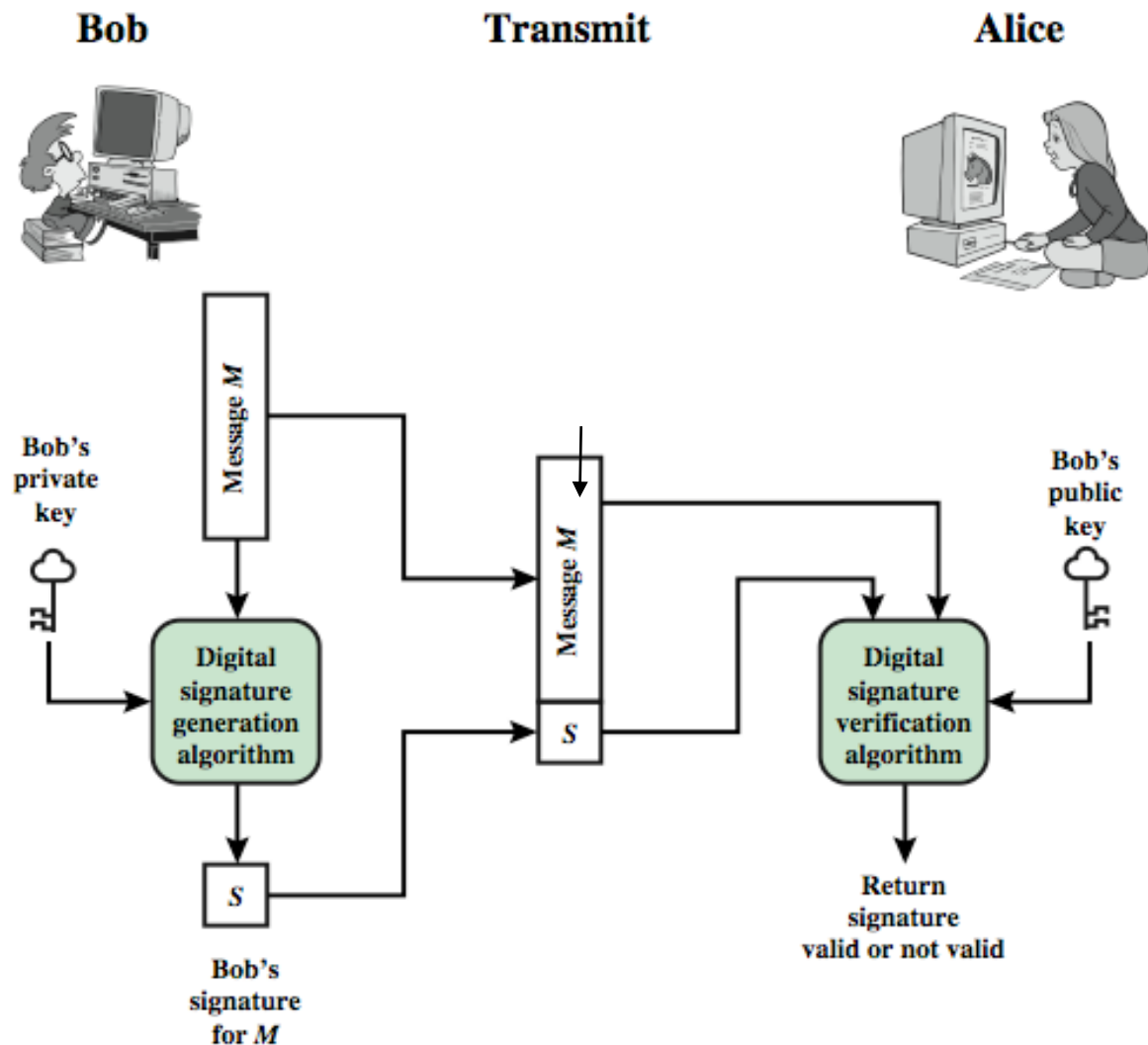
- 公钥和私钥都是Key



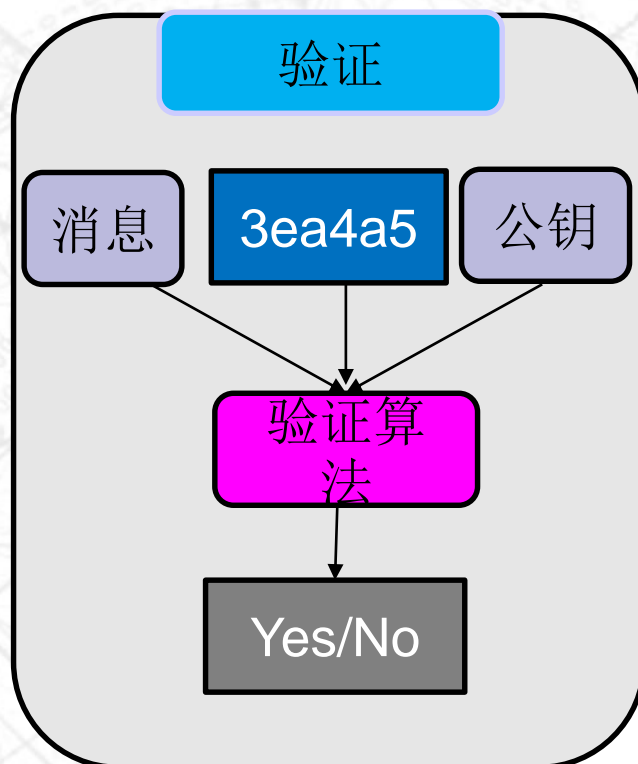
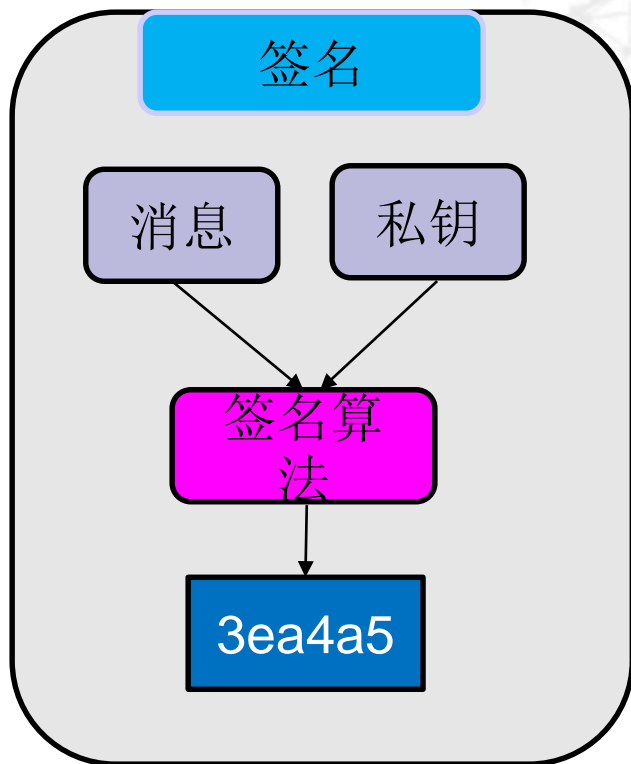
数字签名



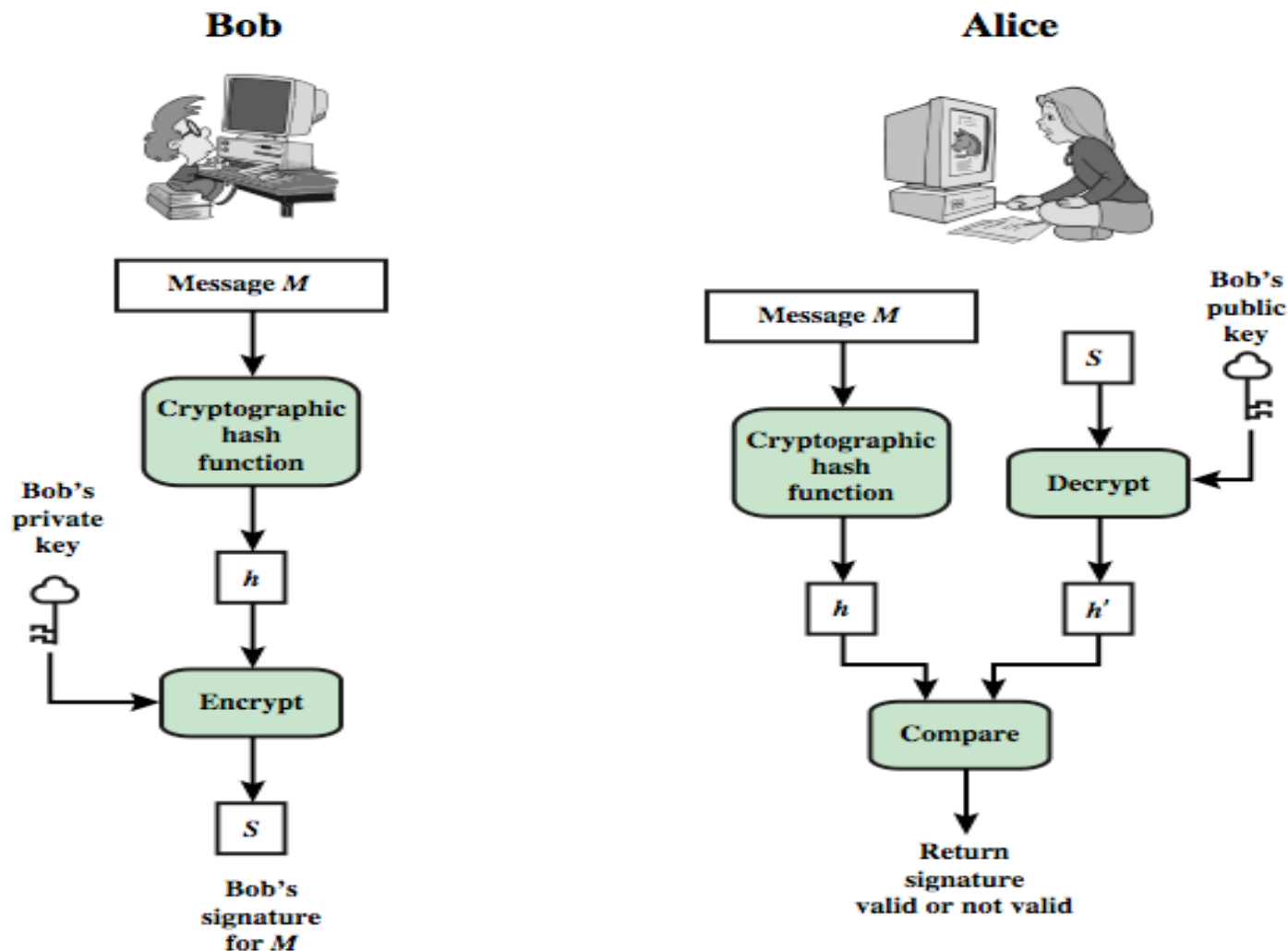
数字签名模型



数字签名原理



数字签名工作流程



数字签名作用

认证

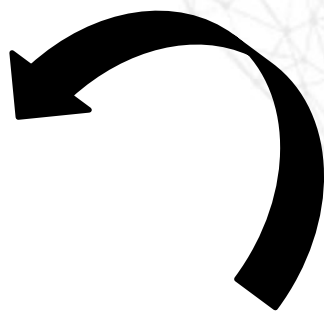
防止抵赖

防篡改

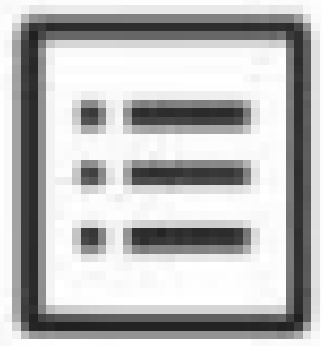
防重放攻击



确认身份



不可反悔



消息完整性



数字签名方案定义

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

① Gen生成密钥对(sk,vk). sk称为签名密钥,vk是验证密钥,用于验证.

② Sig是签名算法. 输入签名密钥sk和要签名的消息m, 得到一个签名

$$\sigma = \text{Sig}_{sk}(m)$$

③ Ver是验证算法. 输入一个签名 σ 和消息m, 输出0或者1.

■ Gen,Sig,Ver应该是多项式时间可计算函数.它们都是公开的函数,vk是公开的,而sk是保密的.

■ 如果没有sk,任何人计算一个满足 $\text{Ver}_{vk}(m,\sigma')=1$ 的消息对(m,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Sig是如何定义的)

签名方案的攻击类型

- 敌手的目标：伪造签名

- ⊙ 如果敌手能够计算满足

$$\text{Ver}_{sk}(m, \sigma) = 1$$

的数据对 (m, σ) , 而 m 没有事先被签名, 则签名 (m, σ) 被称为一个伪造签名

- ⊙ 一个伪造的签名是不使用签名密钥产生的某个消息 m 的一个有效数字签名.

签名方案的攻击类型

下面考虑攻击者可能的几种目的：

- ✧ **完全破译**：攻击者能够确定 Alice 的私钥，即签名函数 sig_K 。因此他能够对任何消息产生有效的签名。
- ✧ **选择性伪造**：攻击者可以对所选择的消息产生有效的签名。换句话说，如果给攻击者一个消息 x ，那么他能（以某种概率）决定签名 y ，使得 $ver_K(x, y) = true$ 。该消息 x 不应该是以前 Alice 曾经签名的消息。
- ✧ **存在性伪造**：攻击者至少能够为一则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 (x, y) ，其中 x 是消息而 $ver_K(x, y) = true$ 。该消息 x 不应该是以前 Alice 曾经签名的消息。

数字签名的设计要求

■ 一个数字签名应满足下列条件：

- 依赖性：签名必须是与消息相关的；
- 唯一性：签名必须使用发送方某些独有的信息
 - ✓ 防止伪造和否认
- 可验证：数字签名必须是在算法上可验证的；
- 抗伪造：伪造数字签名在计算上是不可行的；
 - ✓ 无论是通过以后的数字签名来构造新报文还是对给定的报文构造一个虚假的数字签名（类似笔迹签名不可模仿性）
- 可用性：数字签名的产生识别和证实必须相对简单并且其备份在存储上是可实现的（显然签名不能太长）

数字签名算法

■ 数字签名一般利用公钥密码技术实现，其中其中私钥用来签名，公钥用来验证签名，经典数字签名算法主要有：

- RSA数字签名
- ElGamal数字签名
- Schnorr数字签名
- 数字签名标准DSS
- 基于离散对数问题的一般数字签名体制
- 椭圆曲线数字签名

RSA数字签名方案

④ 定义(RSA签名方案)

由三个算法(Gen,Sig,Ver)构成:

- ① 密钥的生成: Gen选择素数 p, q , 令 $n=pq$. 选择 e 使得 $(e, \phi(n))=1$, 计算 d 使得 $ed \equiv 1 \pmod{\phi(n)}$. $vk:=(n,e), sk:=p,q,d$.
- ② 签名过程 (m, σ) : Sig对于输入 m 和 sk . 计算 $\sigma := m^d \pmod n$, 签名是 (m, σ)
- ③ 验证过程 (e, n) : Ver对于输入 σ, m, vk , $Ver_{vk}(m, \sigma)=1$ 当且仅当 $\sigma^e \equiv m \pmod n$.

椭圆曲线

- 一类特殊的三次代数方程所定义的“平面”曲线
- 属于不定方程：方程的系数为整数，解也要求为整数（或者有理数）
- 费马大定理 当 $n > 2$ 以及 $xyz \neq 0$ 时，方程

$$x^n + y^n = z^n \text{ 没有正整数解}$$

- 安德鲁·怀尔斯1994年最终完成了证明



安德鲁·怀尔斯
Andrew Wiles

椭圆曲线密码系统

● 能量消耗

- 破解228比特的RSA密钥所需能量小于煮沸一茶勺水的能量
- 破解228比特的ECC密钥所需能量足够煮沸地球上所有的水

● 安全

- RSA要达到228比特ECC密钥的安全强度需要2380比特的密钥



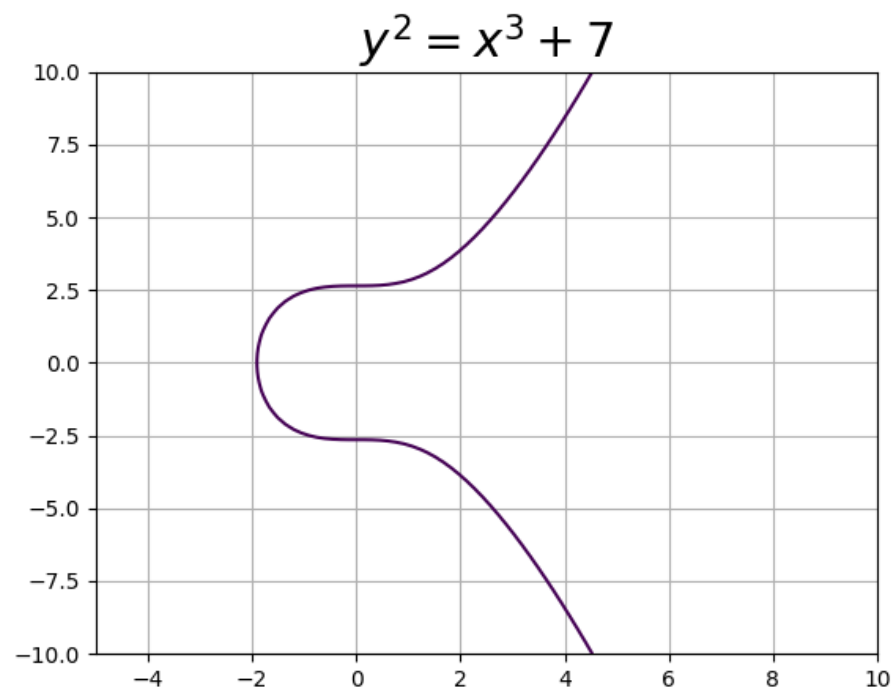
椭圆曲线密码系统

- 1985年由Neal Koblitz和Victor Miller分别独立提出的，2004年开始被广泛使用
- 之前公钥加密算法的问题
 - RSA和Elgamal需要在整数环和域上进行1000bit以上的指数运算
 - 在32位和64位计算机上，耗费大量的CPU时间
 - 密钥长度过大

算法家族	加密系统	安全等级 (bit)			
		80	128	192	256
大整数分解	RSA	1024	3072	7680	15360
离散对数	DH/DSA/Elgamal	1024	3072	7680	15360
椭圆曲线	ECDH/ECDSA	160	256	384	512
对称密码	AES	80	128	192	256

椭圆曲线密码系统

- 特点
 - 有限域 $\text{GF}(2^n)$
 - 运算器容易构造
 - 加密速度快
 - 更小的密钥长度实现同等的安全性
- 椭圆曲线使用160/192/256位的参数
- 使用基于点而不是整数的运算



比特币所使用的曲线

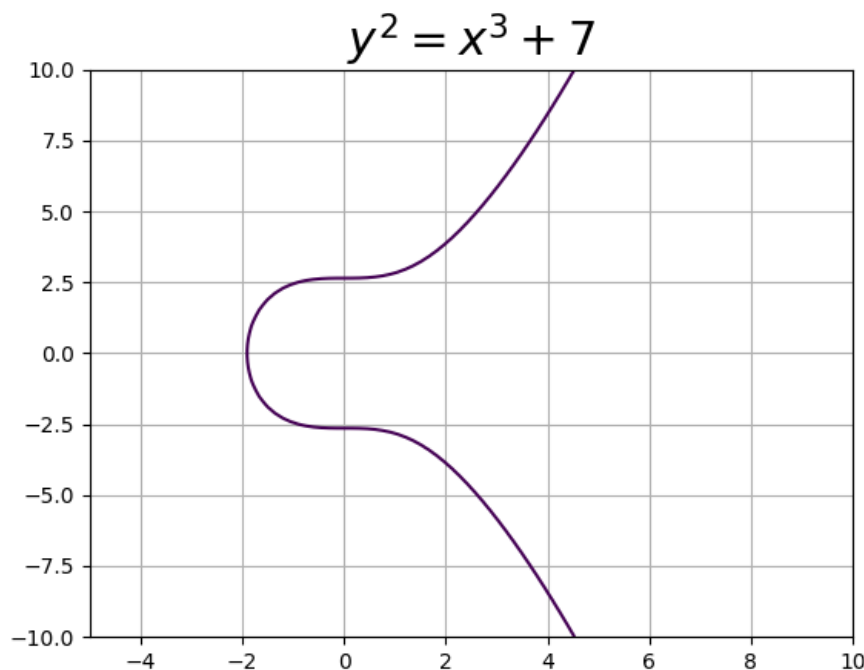
椭圆曲线密码系统

椭圆曲线由简化的Weierstrass方程

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

确定，参数 a, b 决定了曲线的形状

下图就是 $a = 0, b = 7$ 时曲线在实数域上的形状，曲线是关于 x 轴对称的



椭圆曲线密码系统

椭圆曲线密码学中的曲线定义在素数域

定义：

素数域 Z_p ($p > 3$)上的椭圆曲线是所有点 $(x, y) \in Z_p$ 的集合，并且满足

$$y^2 = x^3 + ax + b \bmod p, \\ 4a^3 + 27b^2 \neq 0 \bmod p, a \in Z_p, b \in Z_p$$

曲线还包含一个无穷远处的特殊点：零点 θ

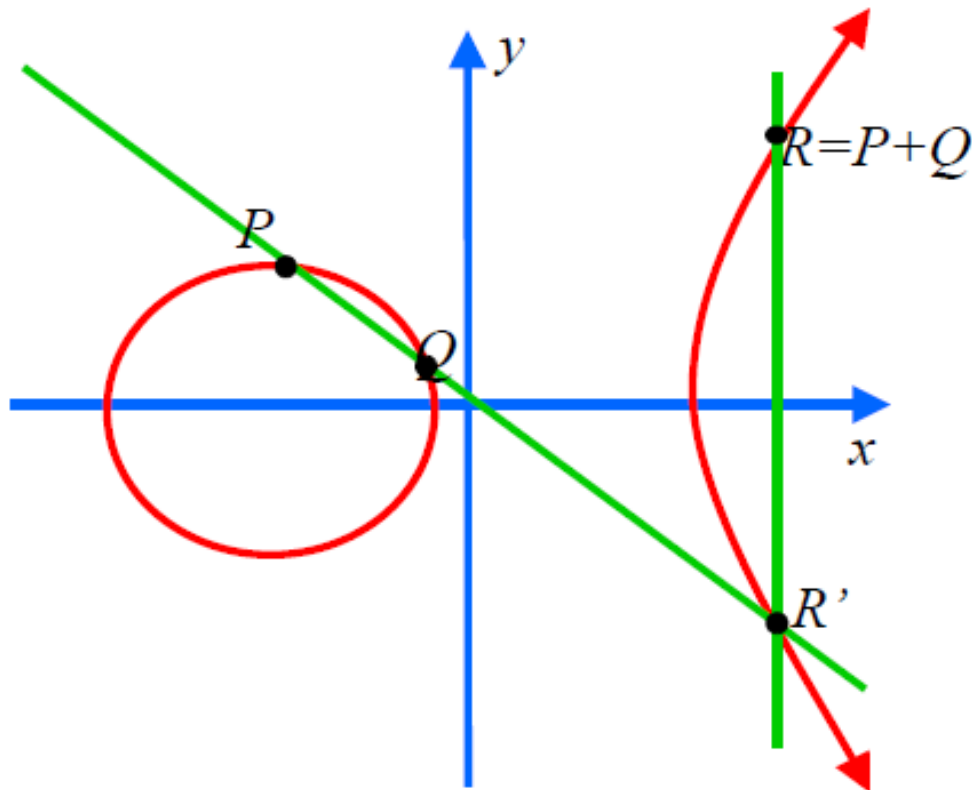
域 $Z_p = \{0, 1, 2, \dots, p - 1\}$

椭圆曲线密码系统

► 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:

$$P + Q = R$$

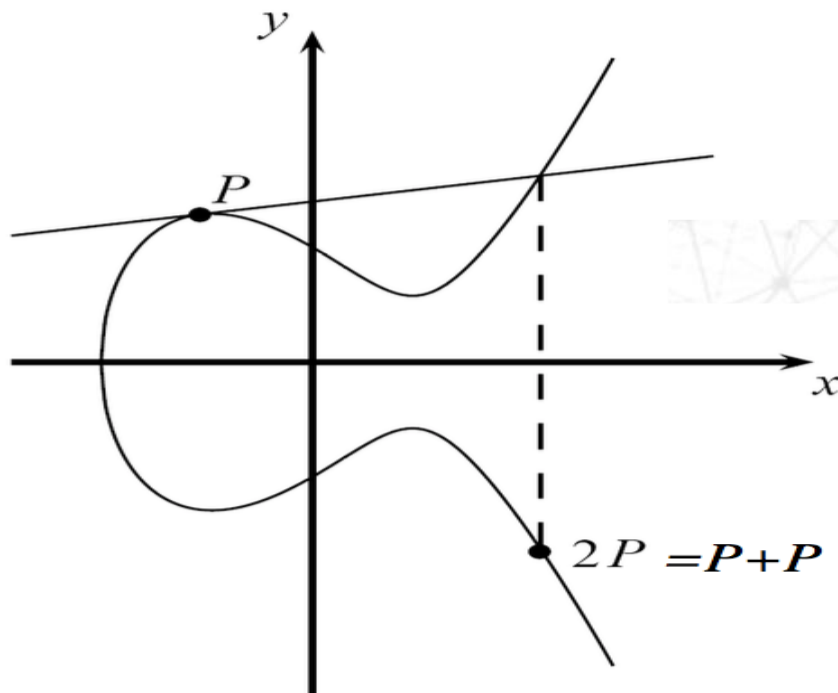


椭圆曲线密码系统

曲线上点的倍乘

$$P + P = 2P$$

过点 P 作切线，与曲线交于一点，该点关于 x 轴的对称点为所求点



椭圆曲线密码系统

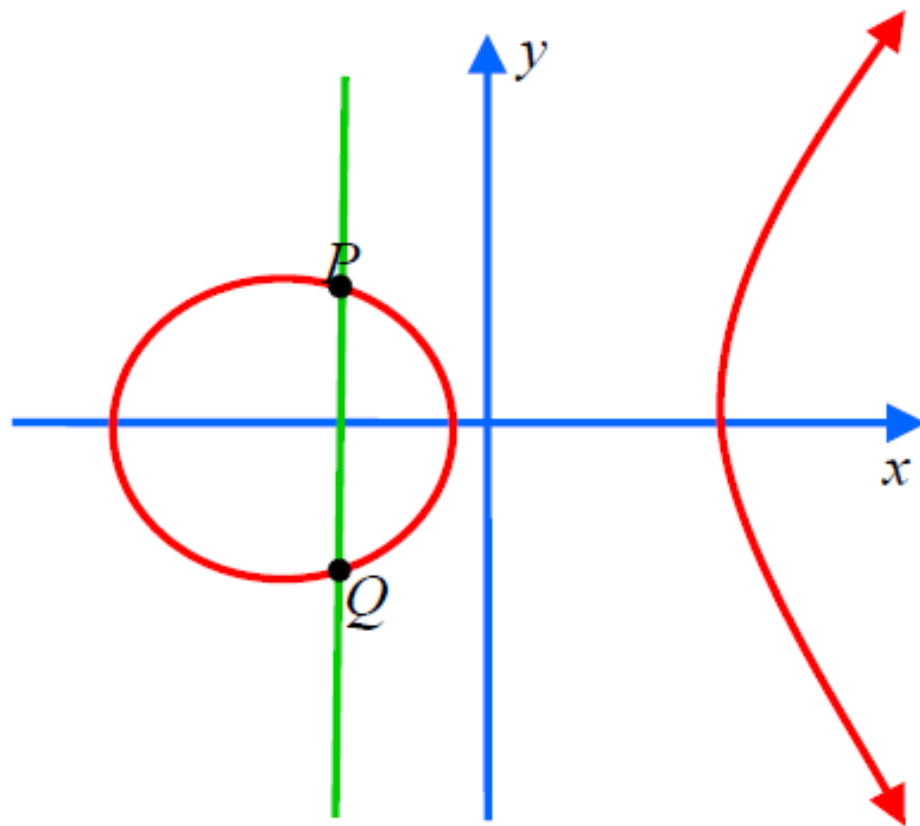
➤ 椭圆曲线上的加法运算

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



➤ 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

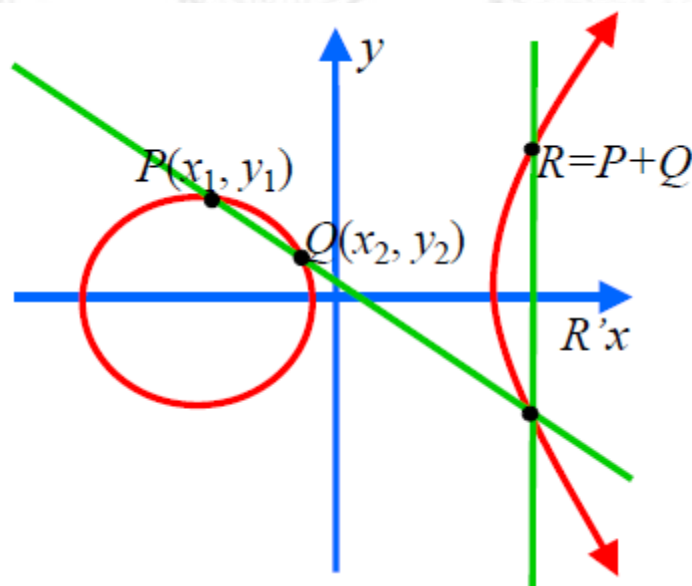
$$\begin{cases} x_3 = \lambda^2 - x_1 - x_2 \\ y_3 = \lambda(x_1 - x_3) - y_1 \end{cases}$$

当 $P \neq Q$ 时,

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

当 $P=Q$ 时,

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

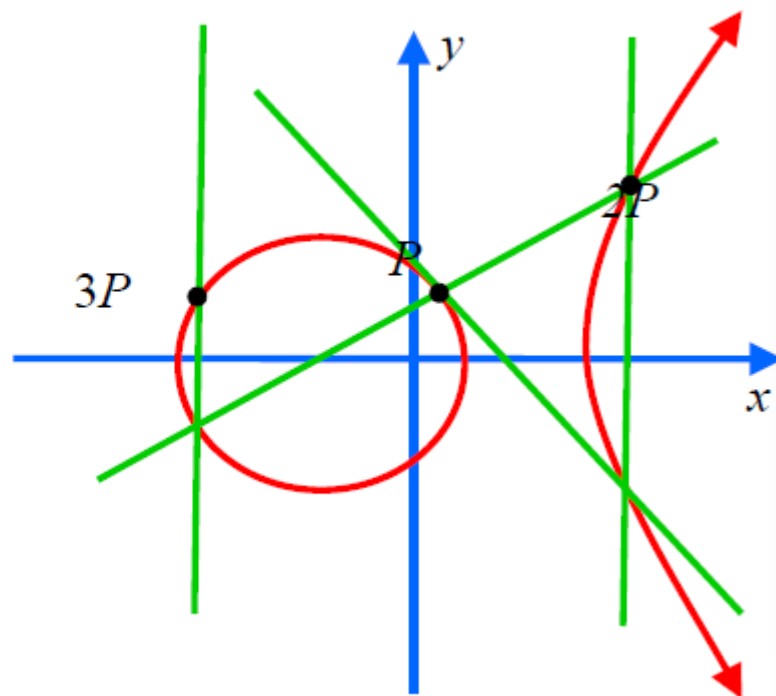


➤ 加法运算的代数表示

$k(k>2)$ 个相同的点 P 相加为

$$\underbrace{P + \dots + P}_k = kP$$

此时称之为点乘运算



➤ 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 $n = \text{ord}(P)$.

由阶为 n 的点 P 在上述加法定义下生成的循环群 $\langle P \rangle$ 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.

椭圆曲线数字签名

- 椭圆曲线密码签名的应用
 - ✓ 2000年美国政府已将椭圆曲线密码引入数字签名标准DSS
 - ✓ 安全，密钥短、软硬件实现节省资源等特点

基于椭圆曲线的数字签名算法

● ECDSA算法

- 椭圆曲线数字签名算法 (EllipticCurveDigitalSignatureAlgorithm, 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(DiscreteLogarithmProblem, DLP)和大数分解问题(IntegerFactorizationProblem, IFP)不同, 椭圆曲线离散对数问题 (EllipticCurveDiscreteLogarithmProblem, ECDLP) 没有亚指数时间的解决方法.

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 ($sk=d, pk=Q$)

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

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.

有特殊性质的签名方案

● 有特殊性质的签名方案

- 盲签名
- 群签名与环签名
- 多重签名
- 聚合签名
- 代理签名
- 不可否认签名
- 一次签名
- 失败即停签名
-

盲签名 (Blind Signature)

- 概念：D.Chaum于1982年首次提出；要求签名者能够在不知道被签名文件内容的情况下对文件进行签名。
另外，即使签名者在以后看到了被签名的文件及其签名，签名者也不能判断出这个签名是他何时为谁生成的。
- 应用场景：电子钱币，匿名选举

盲签名 (Blind Signature)

● 除具有一般数字签名特点外，还具有以下特性：

(1) 签名者对消息的签名是有效的。签名是签名者签署消息的证据，如果把消息给签名者看，签名者确信他签署过这个消息，这个消息具有数字签名的所有性质。

(2) 签名者不能把签署消息的行为与签署的消息相关联，即使他记下了他所作的每一个盲签名，他也不能把某个消息的盲签名与某个消息的内容相关联。

群签名 (Group Signature)

- **概念**：1991年，Chaum和Heyst首次提出群签名 (Group Signature) 原语，允许某个人以集体的名义且以匿名的方式进行签名；允许群管理员打破匿名性以确定实际签名者。
- **应用场景**：某公司董事会做出决定要对某个职员进行处罚，并委托某个董事进行处理。受罚者只知道是董事会集体的决定，不会通过签名联系到具体签署处罚决定的董事，可以避免一些不愉快的事情发生；但是如果某个董事私自签署了有损董事会利益的文件，则董事长能够确认签名董事。

群签名 (Group Signature)

- 群签名特点:

- 1) 只有群中成员才能为消息签名, 产生群签名;
- 2) 签名接收者可以验证签名的有效性, 但不能识别签名者的身份;
- 3) 一旦发生争议, 群管理员可以识别签名者的身份。

数字签名在区块链中的应用

比特币需要利用公钥进行加锁，利用私钥签名进行解锁，从而实现数字货币的交易。解锁过程实际上是利用ECDSA算法的产生数字签名。给定交易信息 m ，签名过程如下：

- ①选择一个随机数 k ;
- ②计算点 $R = k * G = (x_R, y_R)$ ，计算 $r = x_R \bmod n$;
- ③利用私钥 d 计算 $s = k^{-1} * ((H(m) - d * r)) \bmod n$;
- ④输入签名 (r, s) 。

数字签名算法在区块链中的应用

比特币、以太坊等数字货币均采用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，并把签名发给矿工，矿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.

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: $y^2 = x^3 + 7 \bmod p$, 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:

$$n = \text{FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141}$$

为什么会选择这条曲线?

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.

104

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。

本章小结

- 哈希算法与数字摘要
- 对称加密
- 非对称加密
- 数字签名
- 密码算法在区块链中的应用