区块链基础及应用

Chapter 1 密码学及加密货币概述

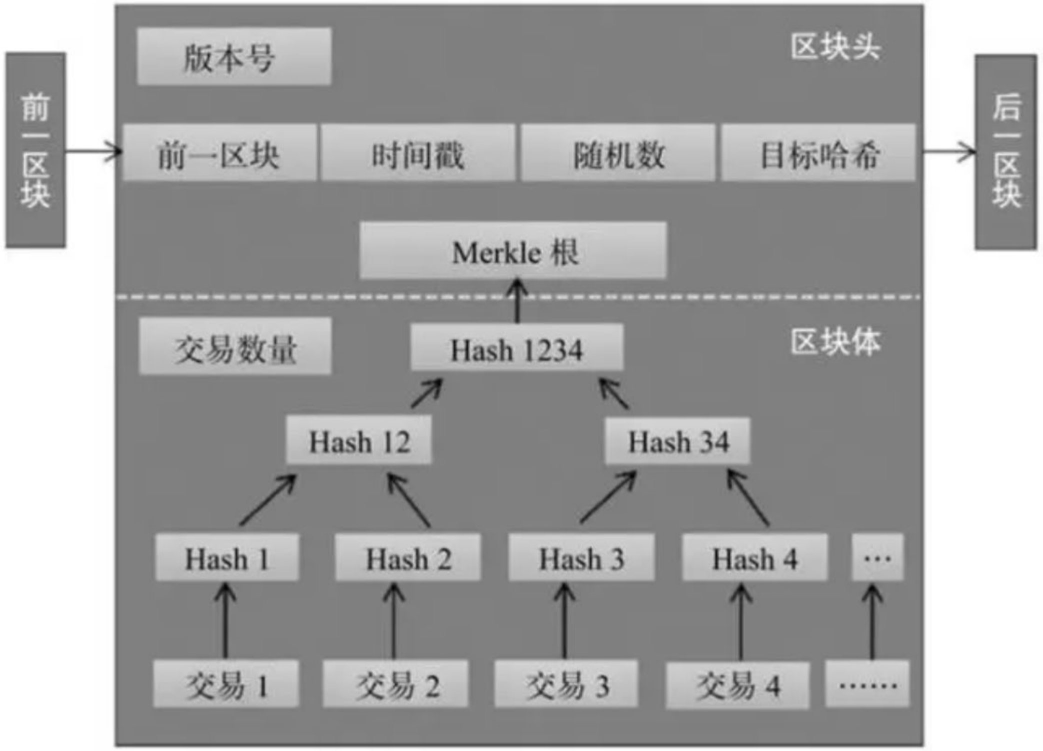
苏 明

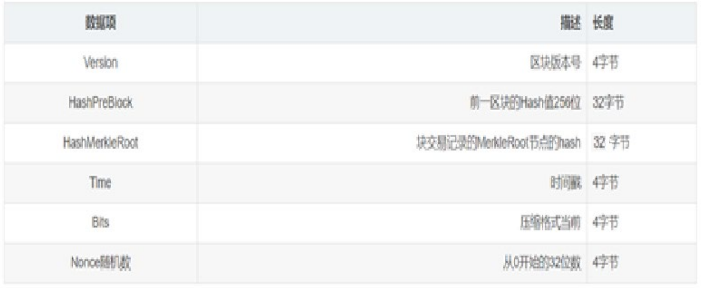
比特币白皮书概览

- Satoshi Nakamato

- Bitcoin: A Peer-to-Peer Electronic Cash System

区块链的组织结构

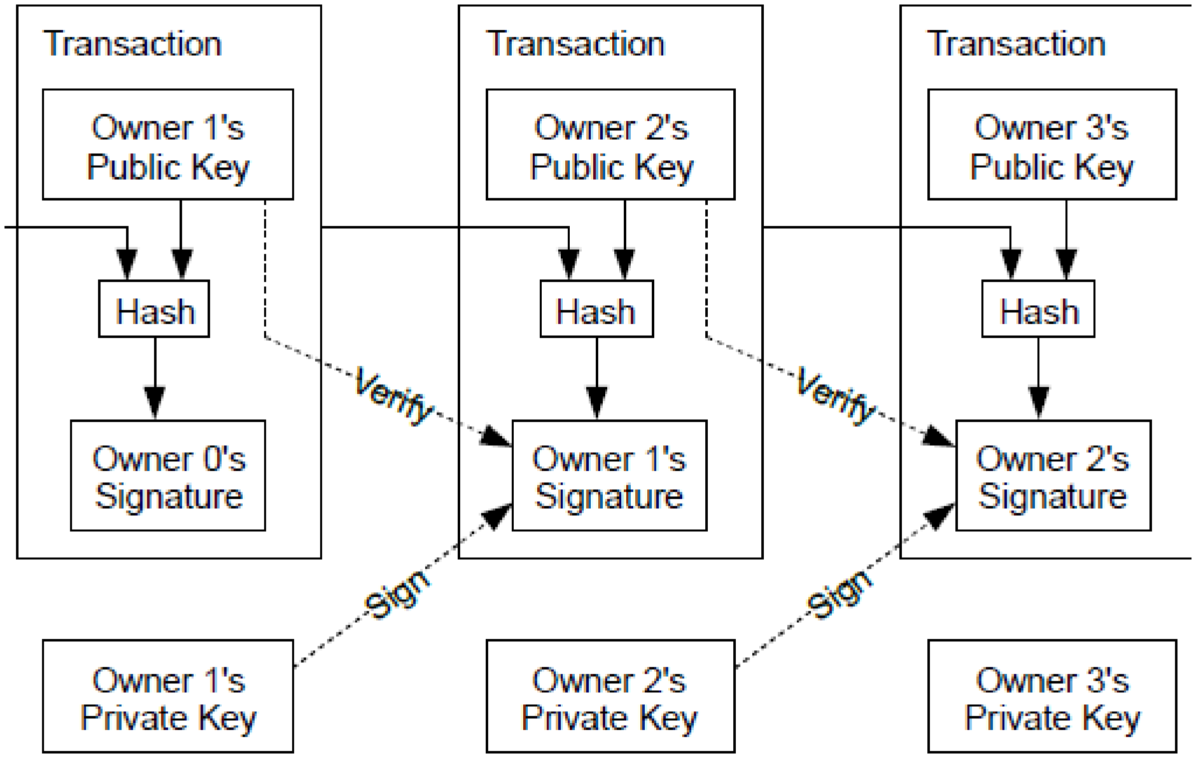




数据项	描述	长度
Version	区块版本号	4字节
HashPreBlock	前一区块的Hash值256位	32字节
HashMerkleRoot	块交易记录的MerkleRoot节点的hash	32 字节
Time	时间戳	4字节
Bits	压缩格式当前	4字节
Nonce随机数	从0开始的32位数	4字节

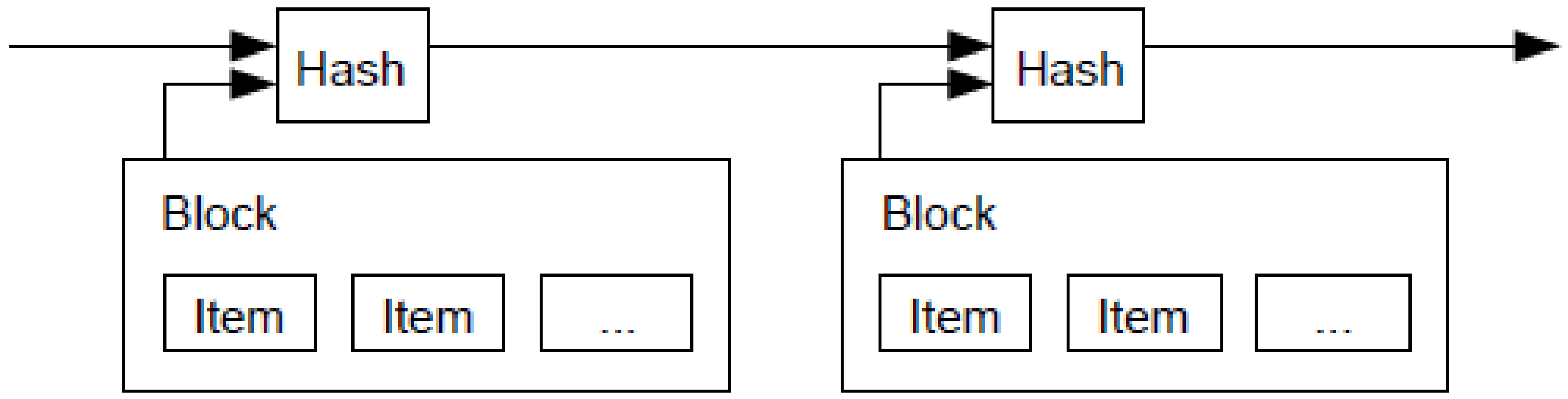
BlockChains & Bitcoin (1/9)

- 交易(Transaction)



BlockChains & Bitcoin (2/9)

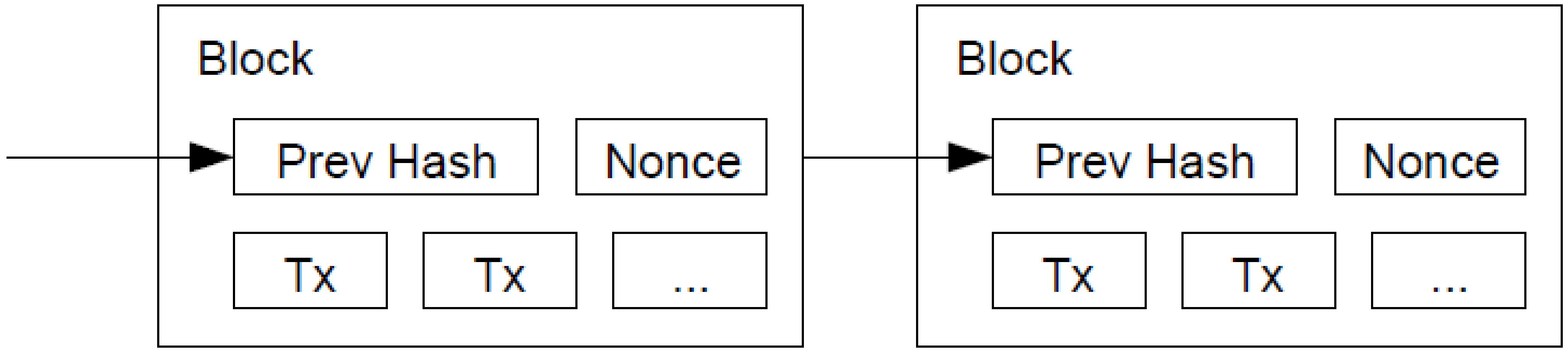
- 时间标签服务 (Timestamp Server)



时间对于整个比特币系统是很重要的。解决Double-spending的核心在于确认这笔交易在之前未发起过，因此如何在没有第三方的情况下明确时间顺序的先后成为重要问题。

BlockChains & Bitcoin (3/9)

- 工作量证明(Proof-of-Work)

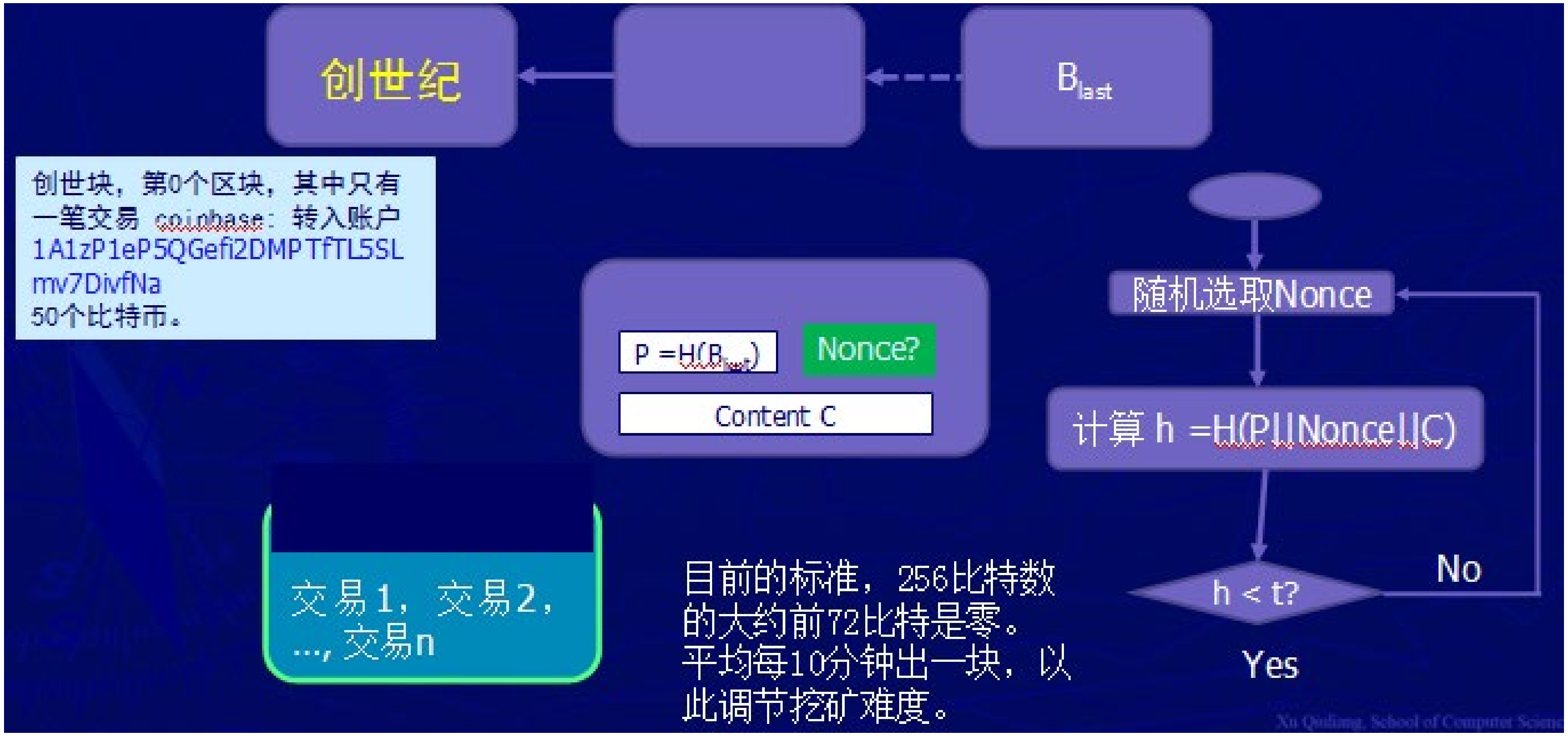


实现工作量证明的方式：逐次修改区块中的nonce直到**满足要求**的区块哈希值产生。

BlockChains & Bitcoin (4/9)

- 系统参数
  - 选取一个hash函数 H() = SHA256(SHA256())
  - 取一个门限值 t = 00000000FF…FF/d = $2^{224}/d$.
  - 选取一个签名体制 EC-DSA
- 交易
  - 建立一个签名算法(sk, pk),
  - A = RIPEMD160(SHA256(pk))便形成一个账户。
  - 交易: (A, 2, B, sig_{skA}(A, 2, B), pk_A)

BlockChains & Bitcoin (5/9)

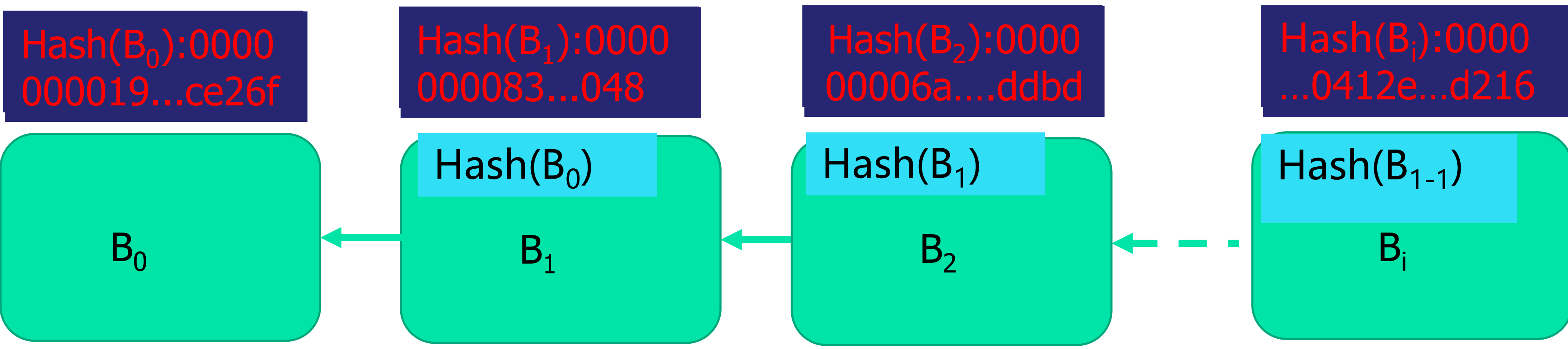


创世块，第0个区块，其中只有一笔交易 coinbase：转入账户 1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa 50个比特币。

交易1，交易2，…，交易n

目前的标准，256比特数的大约前72比特是零。平均每10分钟出一块，以此调节挖矿难度。

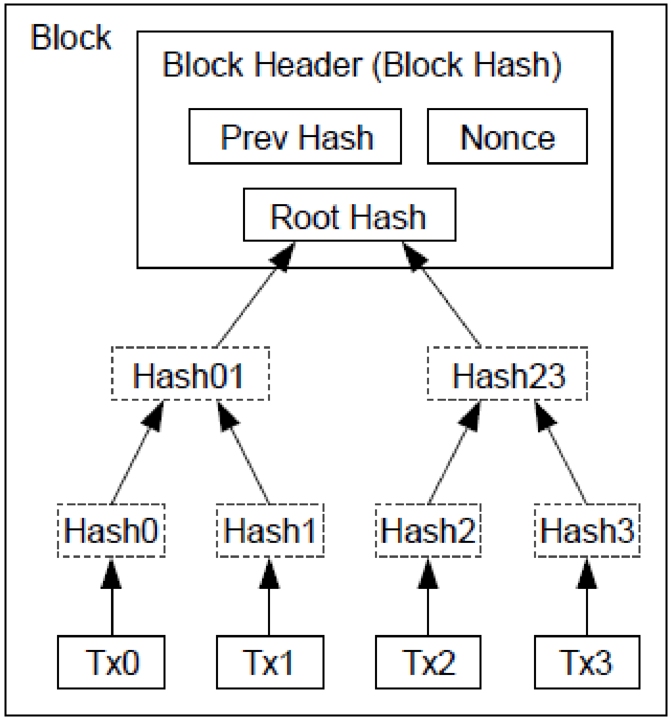
BlockChains & Bitcoin (6/9)



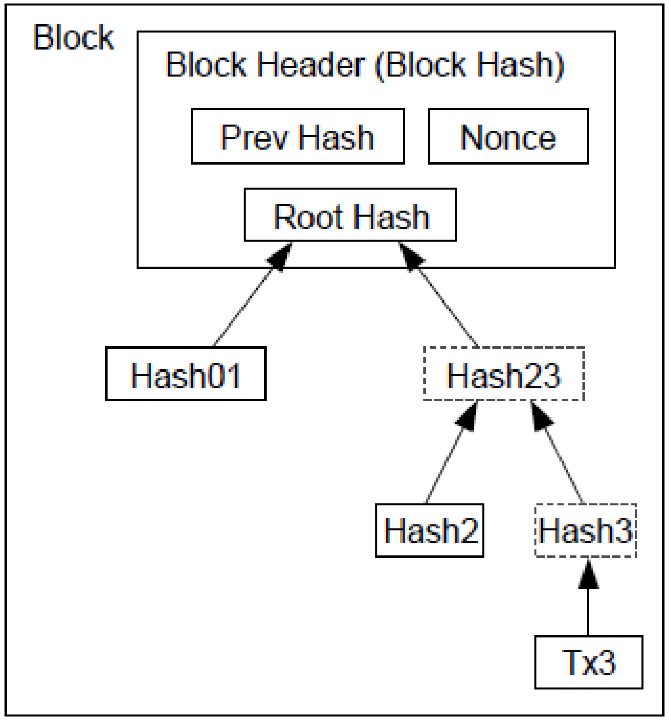
$Hash(B_0)$:0000 000019...ce26f
$Hash(B_1)$:0000 000083...048
$Hash(B_2)$:0000 00006a....ddbd
$Hash(B_i)$:0000 ...0412e...d216
B_0
$Hash(B_0)$
B_1
$Hash(B_1)$
B_2
$Hash(B_{1-1})$
B_i

BlockChains & Bitcoin (7/9)

- 空间存储回收(Reclaiming Disk Space)



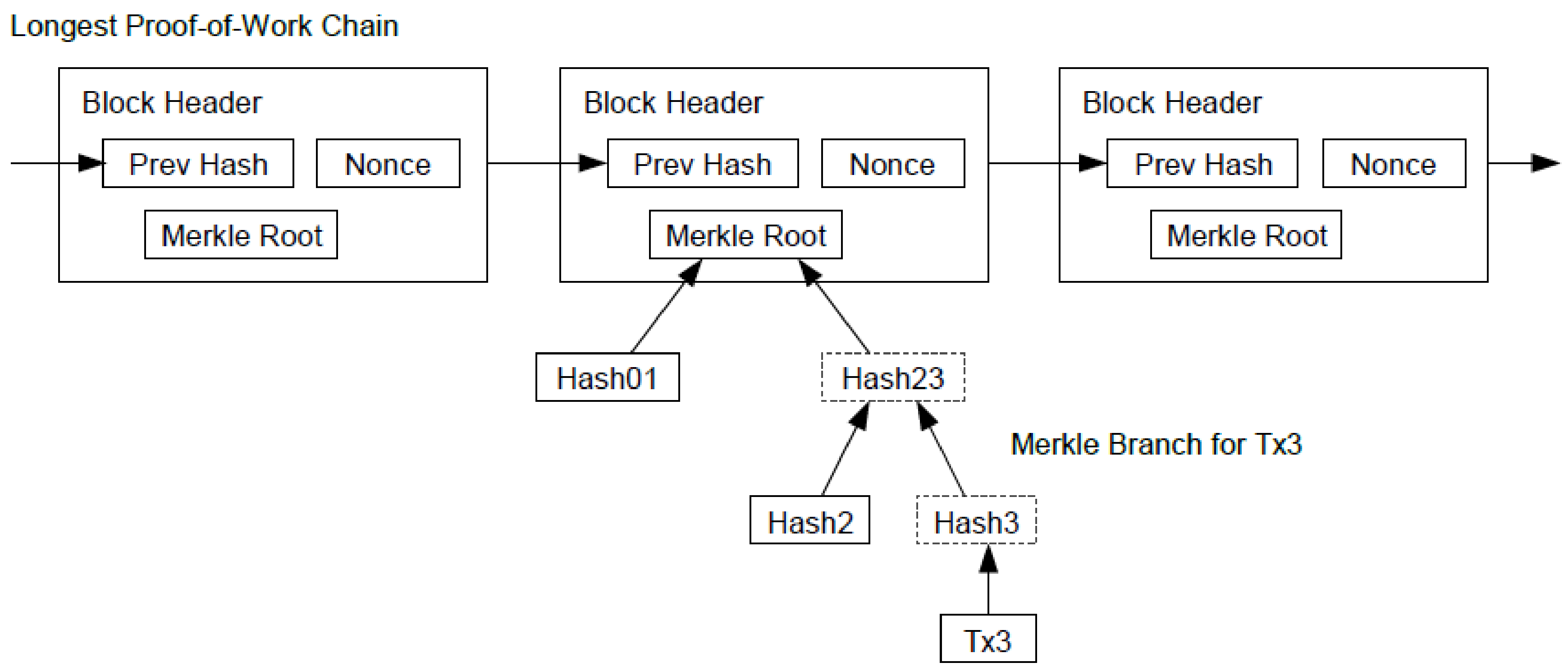
Transactions Hashed in a Merkle Tree



After Pruning Tx0-2 from the Block

BlockChains & Bitcoin (8/9)

- 交易验证 (Simplified Payment)



BlockChains & Bitcoin (9/9)

诚实大多数原理

- 算力主要消耗在挖矿、区块链的生成、交易确认中
- 系统稳定性的缺省信任基础：算力掌握在大多数诚实的用户手中，出于自身利益的考虑，这些用户也愿意维护区块链系统
- 比特币51%攻击理论：**攻击者要创造一条新链条，然后长度超越旧链条，覆盖旧链条**。如果现在只有1次确认，被覆盖的概率稍高，而到了6次确认，被覆盖的概率下降为接近"0"。

第一章概览

- 密码学哈希函数
- 哈希指针及数据结构
- 数字签名
- 公钥即身份
- 两种简单的加密货币

Hash

- Crypto **Hash**

- 消息摘要

Hash Function

- A hash function h(x) is a map from n bits to m bits where m < n.
- An output of a hash function is called a message digest or a hash value.
- If two n bits streams (or messages) have identical hashing values (message digests-> collision

Hash

- Its input can be any string of any size.
- It produces a fixed size output.
- It is efficiently computable.

computing the hash of ann-bit string should have a running time that is O(n)

Properties of Crypto Hash

Three additional properties:

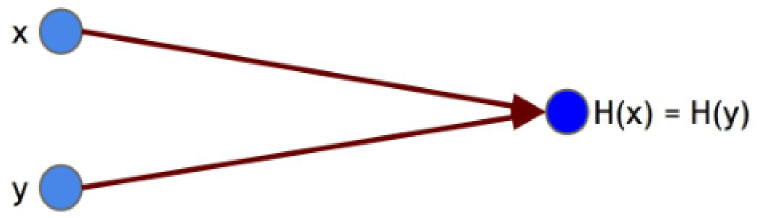
(1) collision-resistance,

(2) hiding,

(3) puzzle-friendliness.

Property 1: Collision-resistance

Collision-resistance: A hash function *H* is said to be collision resistant if it is infeasible to find two values, *x* and *y*, such that *x* ≠ *y*, yet *H(x)=H(y)*.



Property 1: Collision-resistance

- Birthday paradox Problem

- A bad design

$$H(x) = x \ mod \ 2^{256}$$

Property 1: Collision-resistance

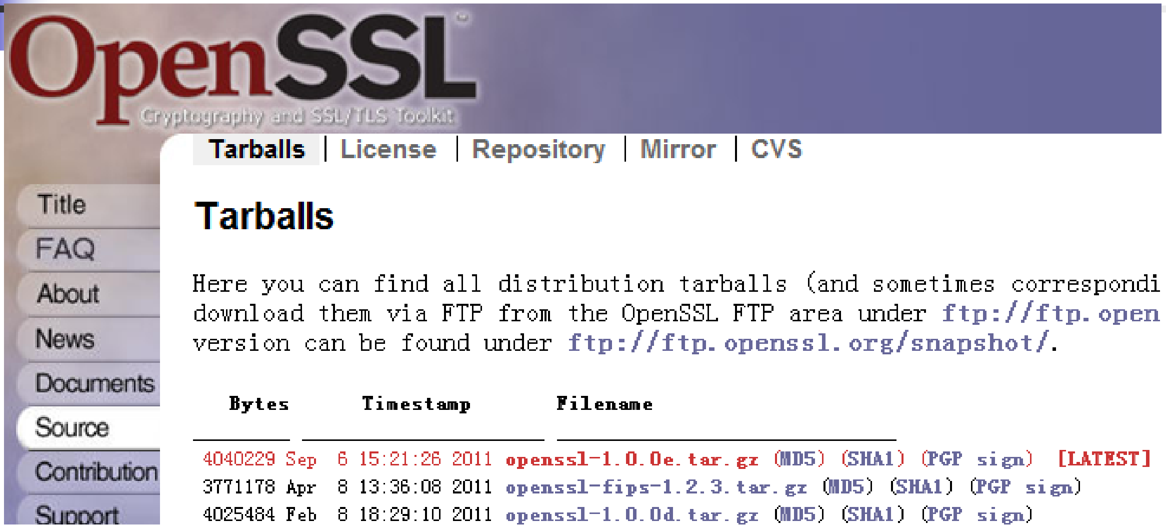
- Message Digest

Collision-free hashes provide an elegant and e fficient solution

Example: SecureBox, an authenticated online file storage system

- ✓ Save storage
- ✓ Ensure its integrity

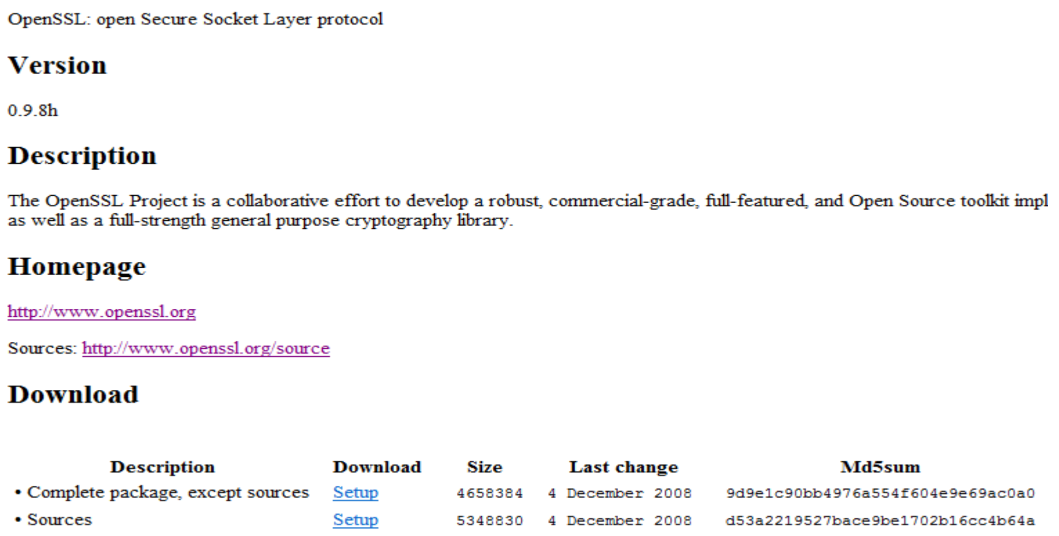
Property 1: Collision-resistance


OpenSSL
Cryptography and SSL/TLS Toolkit
Tarballs | License | Repository | Mirror | CVS
Title
FAQ
About
News
Documents
Source
Contribution
Support

Tarballs

Here you can find all distribution tarballs (and sometimes correspondi
download them via FTP from the OpenSSL FTP area under ftp://ftp.open
version can be found under ftp://ftp.openssl.org/snapshot/.

```
  Bytes      Timestamp          Filename
 4040229 Sep  6 15:21:26 2011 openssl-1.0.0e.tar.gz (MD5) (SHA1) (PGP sign)  [LATEST]
 3771178 Apr  8 13:36:08 2011 openssl-fips-1.2.3.tar.gz (MD5) (SHA1) (PGP sign)
 4025484 Feb  8 18:29:10 2011 openssl-1.0.0d.tar.gz (MD5) (SHA1) (PGP sign)
```



OpenSSL: open Secure Socket Layer protocol

Version

0.9.8h

Description

The OpenSSL Project is a collaborative effort to develop a robust, commercial-grade, full-featured, and Open Source toolkit impl
as well as a full-strength general purpose cryptography library.

Homepage

http://www.openssl.org

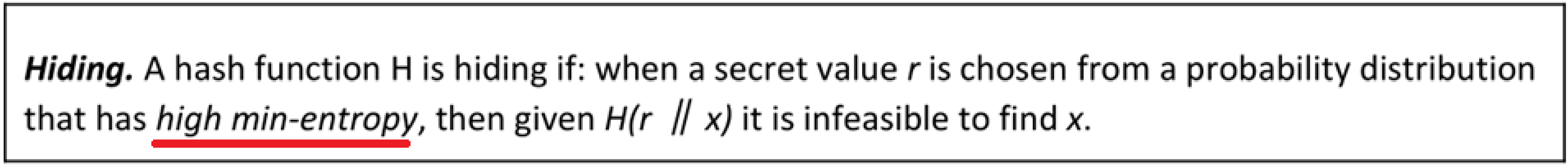
Sources: http://www.openssl.org/source

Download

Description	Download	Size	Last change	Md5sum
• Complete package, except sources	Setup	4658384	4 December 2008	9d9e1c90bb4976a554f604e9e69ac0a0
• Sources	Setup	5348830	4 December 2008	d53a2219527bace9be1702b16cc4b64a

Property 2: Hiding

Given the output of the hash function y=H(x)， there's no feasible way to figure out what the input x is

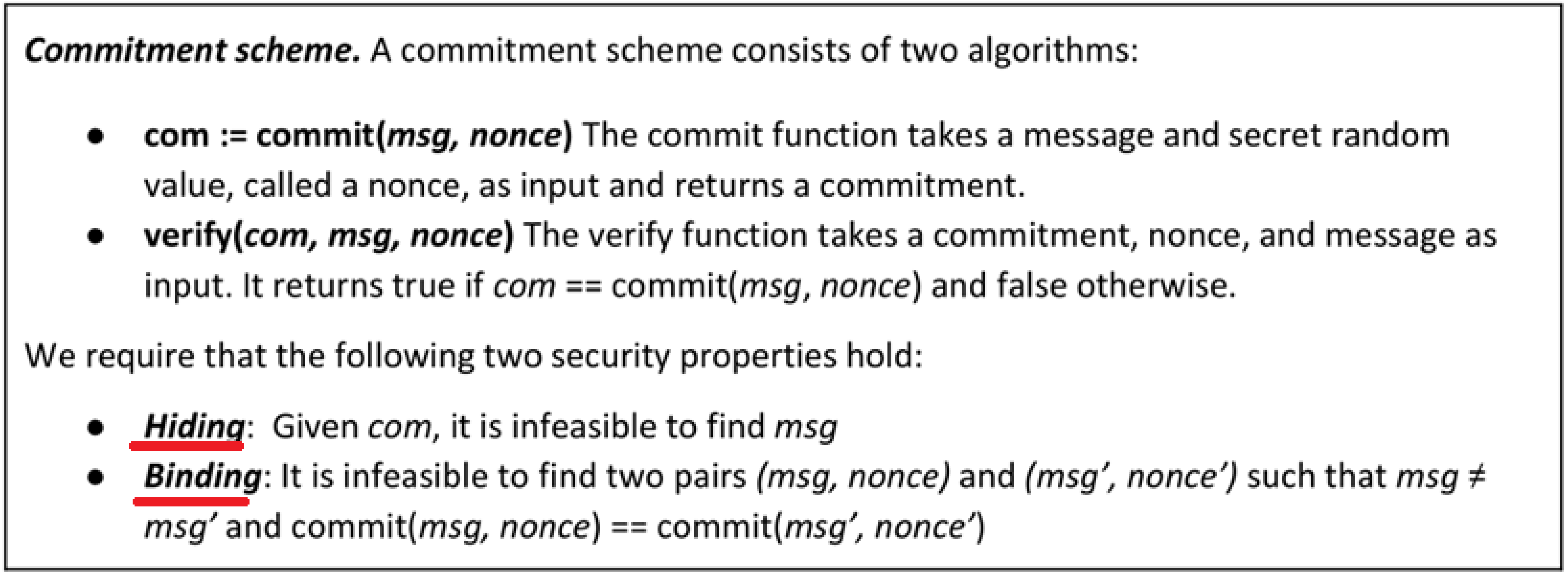


Hiding. A hash function H is hiding if: when a secret value *r* is chosen from a probability distribution that has *high min-entropy*, then given *H(r ∥ x)* it is infeasible to find *x*.

Property 2: Hiding

Application: Commitments

A commitment is the digital analog of taking a
value, sealing it in an envelope, and putting that envelope out on the table where everyone ca
n see it. Later, you can open the envelope and reveal the value that you committed to earlier.



Commitment scheme. A commitment scheme consists of two algorithms:

- **com := commit(*msg, nonce*)** The commit function takes a message and secret random value, called a nonce, as input and returns a commitment.
- **verify(*com, msg, nonce*)** The verify function takes a commitment, nonce, and message as input. It returns true if *com* == commit(*msg*, *nonce*) and false otherwise.

We require that the following two security properties hold:

- ***Hiding***: Given *com*, it is infeasible to find *msg*
- ***Binding***: It is infeasible to find two pairs *(msg, nonce)* and *(msg', nonce')* such that *msg* ≠ *msg'* and commit(*msg*, *nonce*) == commit(*msg'*, *nonce'*)

Property 2: Hiding

commit(*msg, nonce)* := H(*nonce* // *msg*) where *nonce* is a random 256-bit value

Hiding?
Binding?

Property 3: Puzzle friendliness

Puzzle friendliness. A hash function *H* is said to be puzzle-friendly if for every possible n-bit output value *y*, if k is chosen from a distribution with high min-entropy, then it is infeasible to find *x* such that H(k || x) = y in time significantly less than 2^n.

Application: Search puzzle.

Search puzzle. A search puzzle consists of

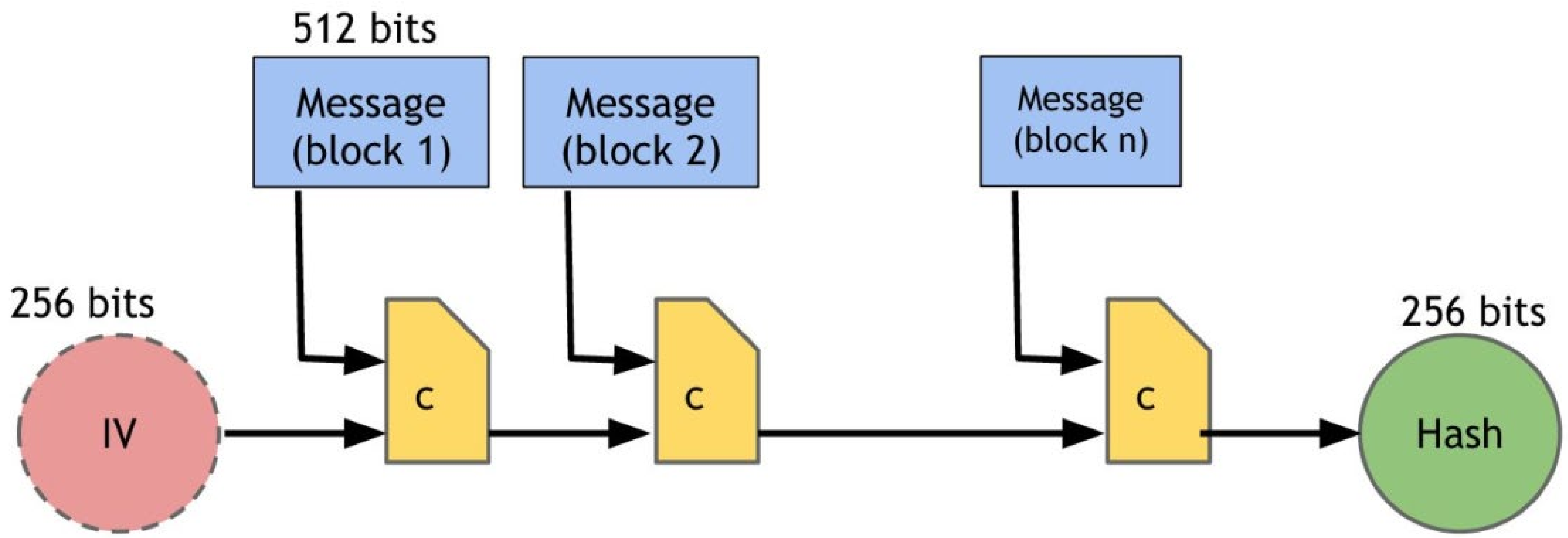
- a hash function, *H*,
- a value, *id* (which we call the ***puzzle-ID***), chosen from a high min-entropy distribution
- and a target set *Y*

A solution to this puzzle is a value, *x*, such that

$$H(id \,\|\, x) \in \boldsymbol{Y}.$$

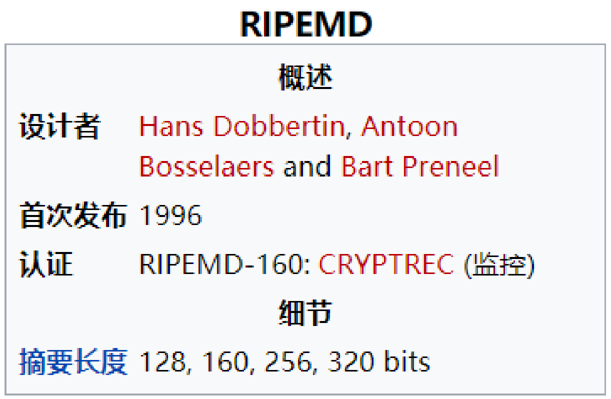
SHA-256

- SHA-256



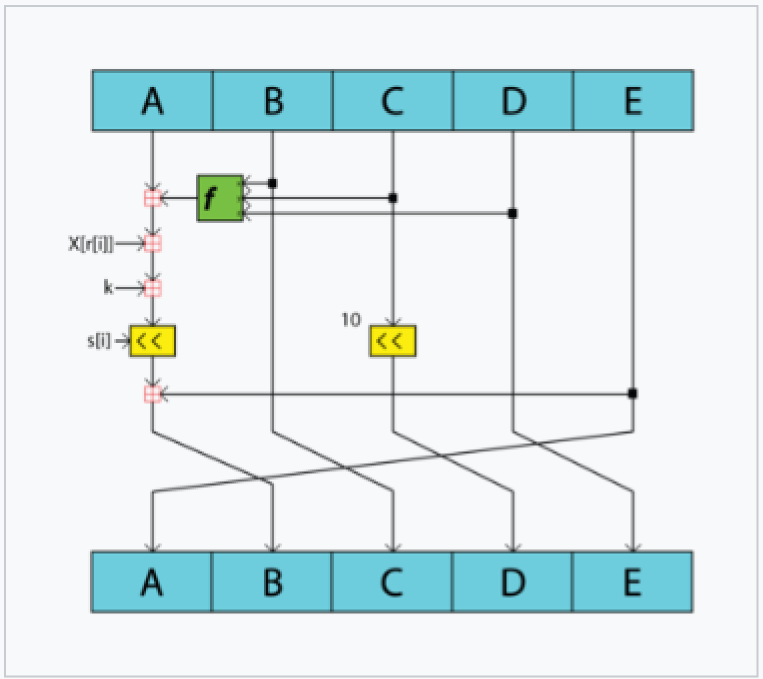
Hash functions-Bitcoin

- **RIPEMD160：** 以MD4为基础原则所设计的，其表现与SHA-1类似.



RIPEMD

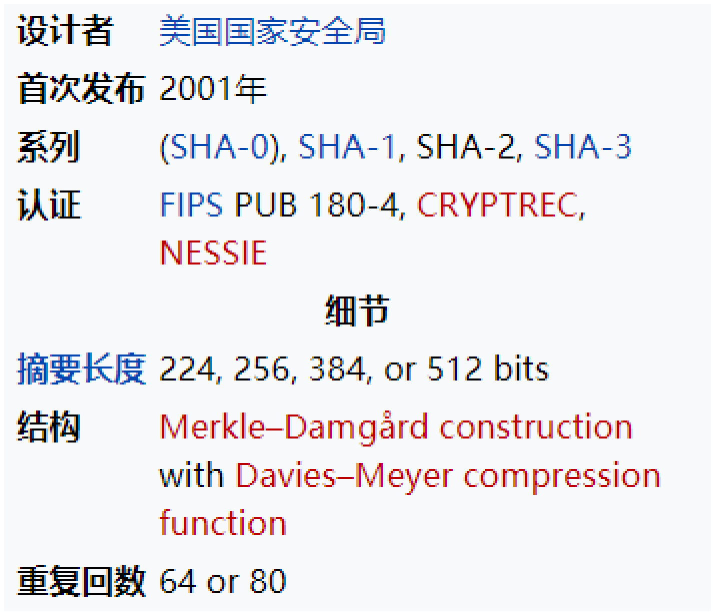
概述	
设计者	Hans Dobbertin, Antoon Bosselaers and Bart Preneel
首次发布	1996
认证	RIPEMD-160: CRYPTREC (监控)
细节	
摘要长度	128, 160, 256, 320 bits



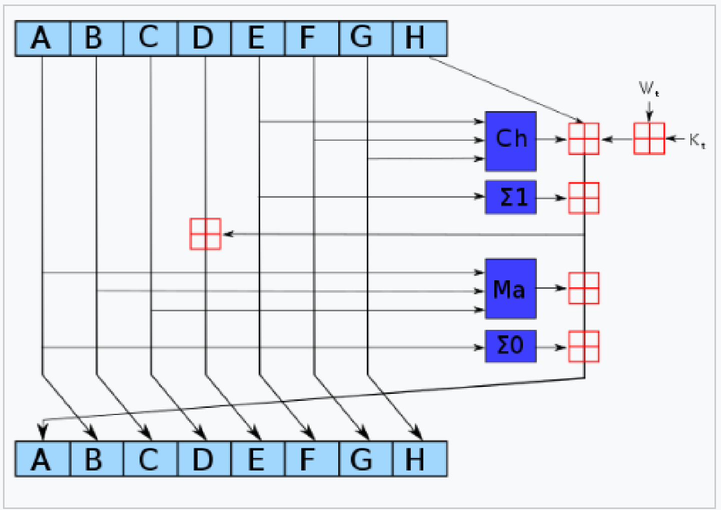
一个RIPEMD160哈希算法压缩函数的子块

Hash functions-Bitcoin

- **SHA256:** 一种密码散列函数算法标准，由美国国家标准与技术研究院（NIST）在2001年发布。属于SHA算法之一



设计者	美国国家安全局
首次发布	2001年
系列	(SHA-0), SHA-1, SHA-2, SHA-3
认证	FIPS PUB 180-4, CRYPTREC, NESSIE
细节	
摘要长度	224, 256, 384, or 512 bits
结构	Merkle–Damgård construction with Davies–Meyer compression function
重复回数	64 or 80



SHA-2的第t个加密循环。图中的深蓝色方块是事先定义好的非线性函数。ABCDEFGH一开始分别是八个初始值，K_t是第t个密钥，W_t是本区块产生第t个word。原消息被切成固定长度的区块，对每一个区块，产生n个word（n视算法而定），透过重复运作循环n次对ABCDEFGH这八个工作区段循环加密。最后一次循环所产生的八段字符串合起来即是此区块对应到的散列字符串。若原消息包含数个区块，则最后还要将这些区块产生的散列字符串加以混合才能产生最后的散列字符串。

Merkle Tree (1/4)

广播多播中密钥分配和认证问题

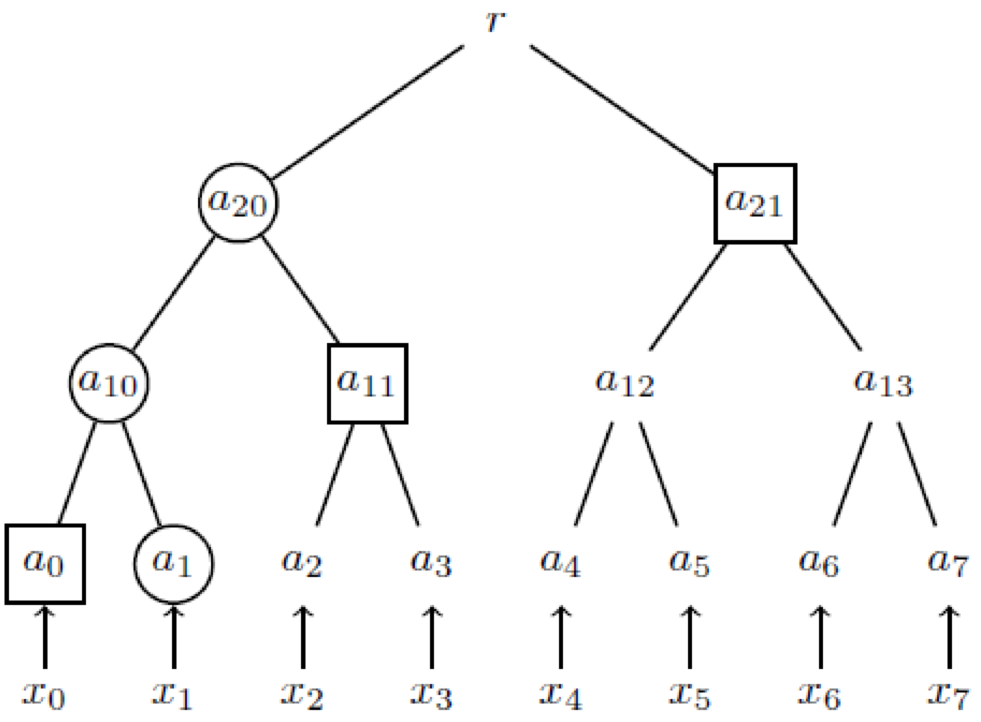
一个解决办法：采用数字签名

更有效率的解决方法：有‘组织’的Hash

区块链中，可以用做交易的‘成员证明’

Merkle Tree (2/4)

- A complete binary tree where each parent vertex is attached by a hash value with inputs from two children.



Merkle Tree (3/4)

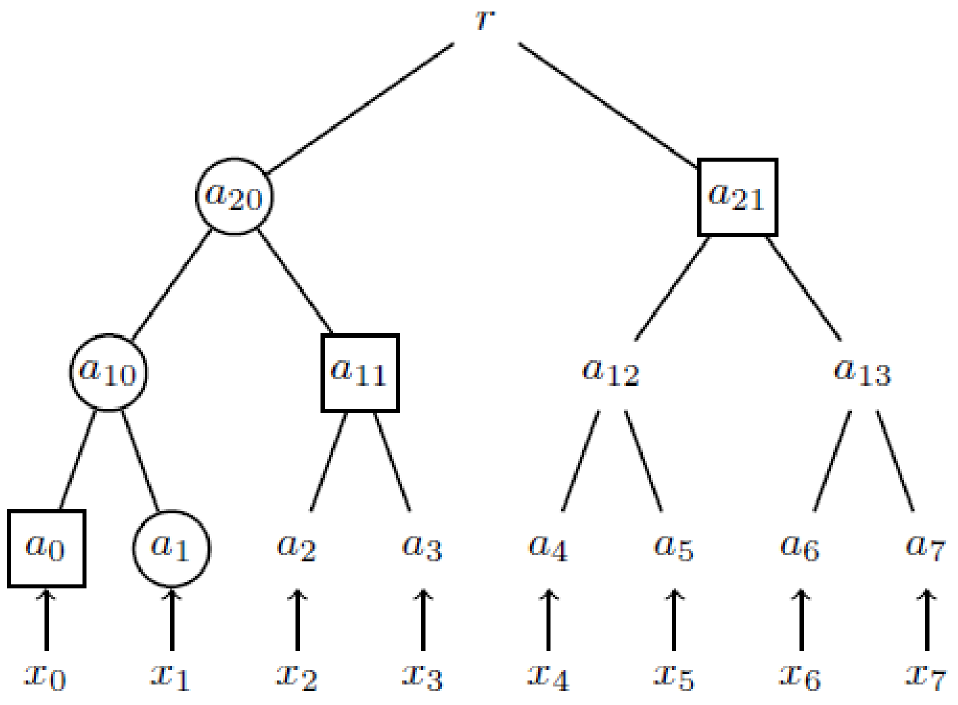
$$a_i = h(x_i), i = 0, \cdots, 7$$

$$a_{1,0} = h(a_0||a_1) = h(h(x_0)||h(x_1))$$
$$a_{1,1} = h(a_2||a_3) = h(h(x_2)||h(x_3))$$
$$a_{1,2} = h(a_4||a_5) = h(h(x_4)||h(x_5))$$
$$a_{1,3} = h(a_6||a_7) = h(h(x_6)||h(x_7))$$

$$a_{2,0} = h(a_{1,0}||a_{1,1}) = h(h(h(x_0)||h(x_1))||h(h(x_2)||h(x_3)))$$
$$a_{2,1} = h(a_{1,2}||a_{1,3}) = h(h(h(x_4)||h(x_5))||h(h(x_6)||h(x_7)))$$

$$\begin{aligned} r &= h(a_{2,0}||a_{2,1}) \\ &= h(h(h(h(x_0)||h(x_1))||h(h(x_2)||h(x_3)))||h(h(h(x_4)||h(x_5))||h(h(x_6)||h(x_7)))) \end{aligned}$$

Merkle Tree (4/4)



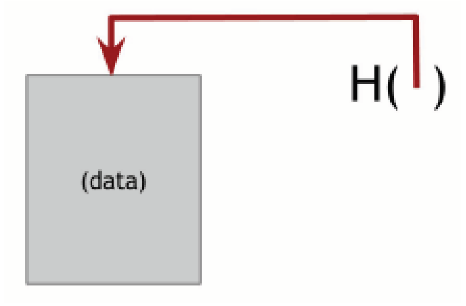
$$Auth(x_1) = \{a_1, a_{1,0}, a_{2,0}, r\}.$$

$$Sib(x_1) = \{a_0, a_{1,1}, a_{2,1}\}.$$

1.2 哈希指针及数据结构

- 指针：指向数据存储位置

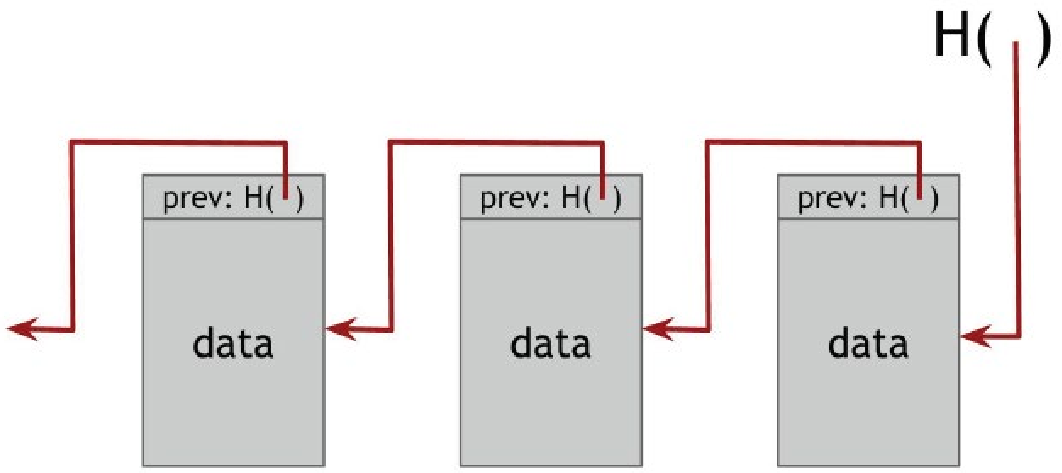
- **哈希指针(Hash Pointer):** 不仅告诉数据存储的位置，而且还可以让你验证数据是否篡改过



1.2 哈希指针及数据结构

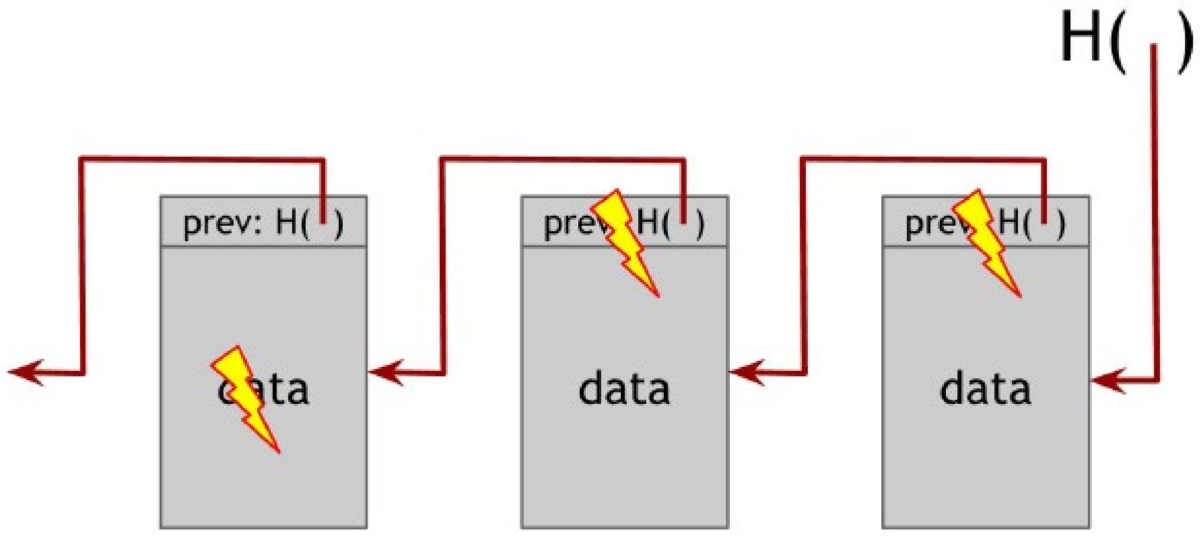
区块链(Blockchain)

- 通过哈希指针构建一个链表



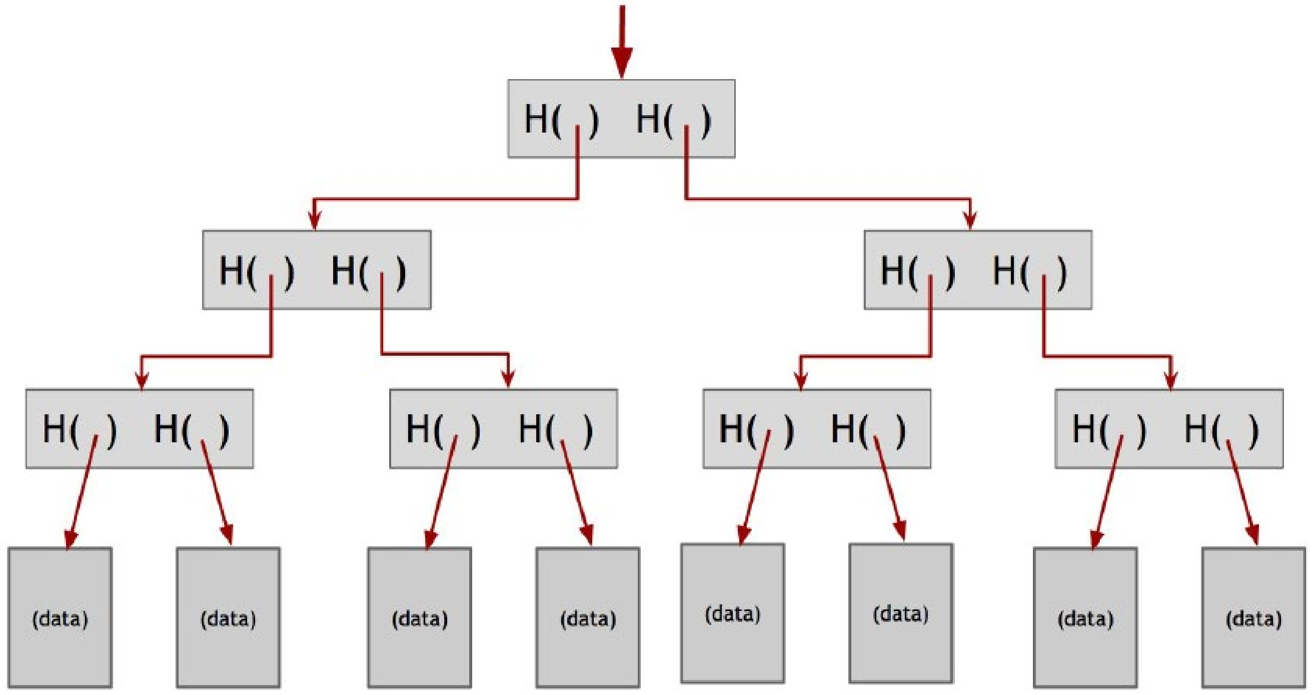
1.2 哈希指针及数据结构

防篡改日志： 牵一发而动全身



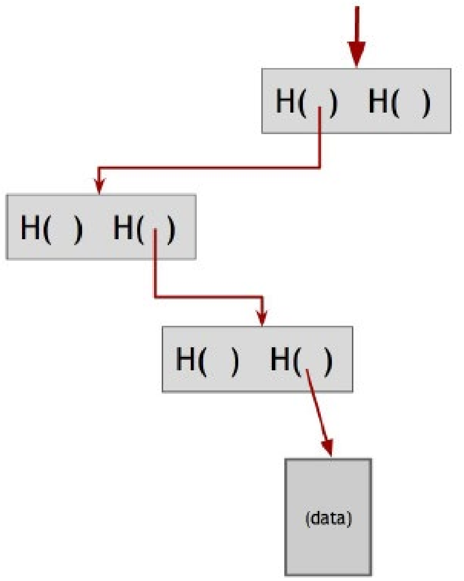
1.2 哈希指针及数据结构

- Merkle Tree



1.2 哈希指针及数据结构

- Proof of Membership



O(logn)

- Proof of Nonmembership?

公钥

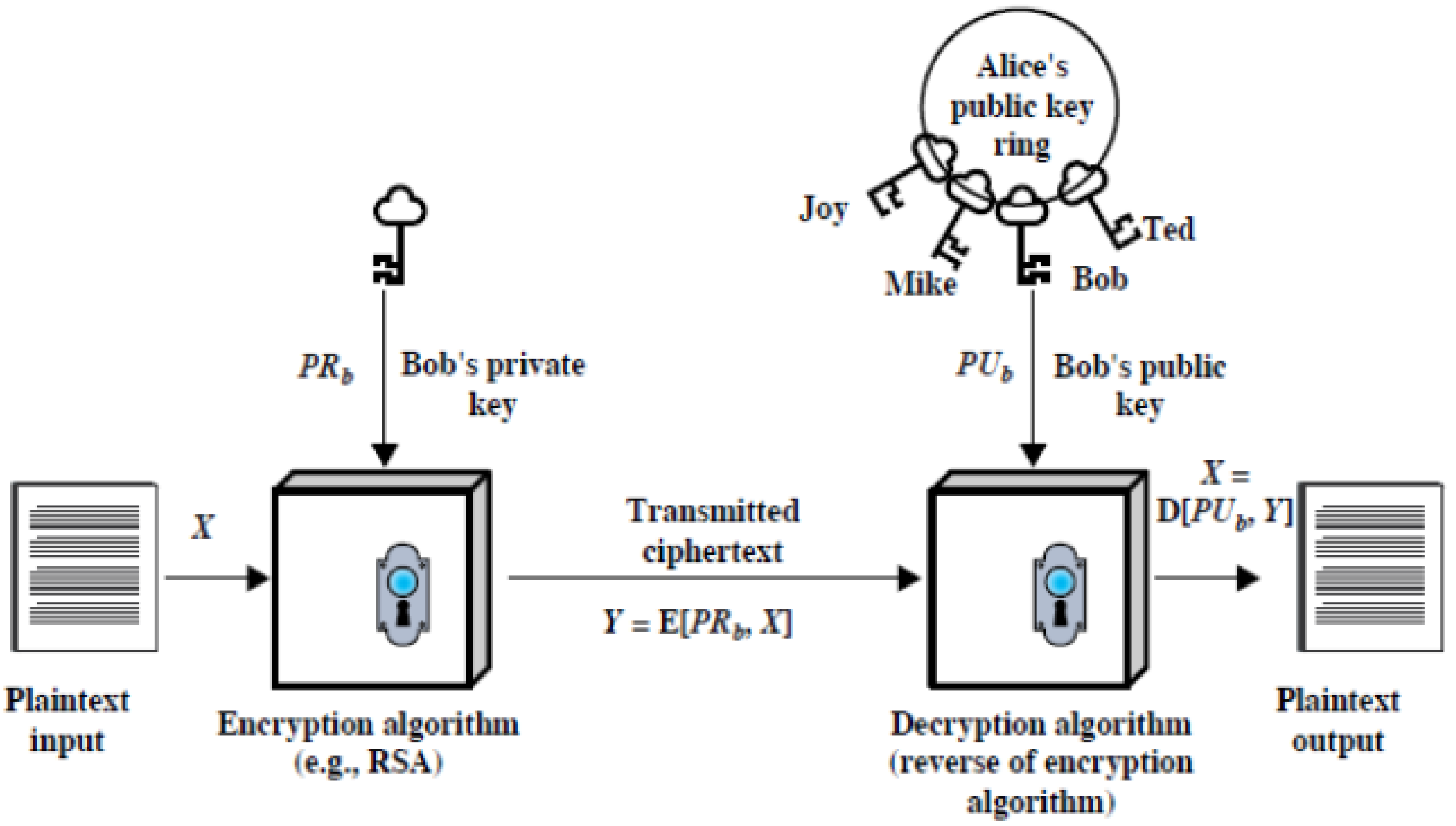
- 1976 Diffie, Hellman

- 公钥算法基于数学函数，而不像对称加密算法是基于比特模式的操作

公钥加密算法

- RSA公钥加密算法
- 1977 Ron Rivest, Adi Shamir, Len Adleman: MIT
- 1978: “A Method for Obtaining Digital Signatures and Public Key Cryptosystems”, Communications of the ACM, Feb. 1978

公钥体制



Alice's public key ring
Joy
Mike
Bob
Ted
PR_b
Bob's private key
PU_b
Bob's public key
X
Transmitted ciphertext
$Y = E[PR_b, X]$
$X = D[PU_b, Y]$
Plaintext input
Encryption algorithm (e.g., RSA)
Decryption algorithm (reverse of encryption algorithm)
Plaintext output

RSA公钥加密算法

- RSA是分组密码，对于某个n,它的明文和密文定义在Z_n上。
- 对于明文块M和密文块C,加密和解密有如下的形式：

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

$$M = C^d \bmod n = M^{ed} \bmod n$$

RSA公钥加密算法

发送者和接收者必须都知道n,e

- 公钥KU={e,n}
- 私钥KR={d,n}
- 安全性要求：

✓ 可以找到e,d,n使得对于M<n,M^{ed}=M mod n;

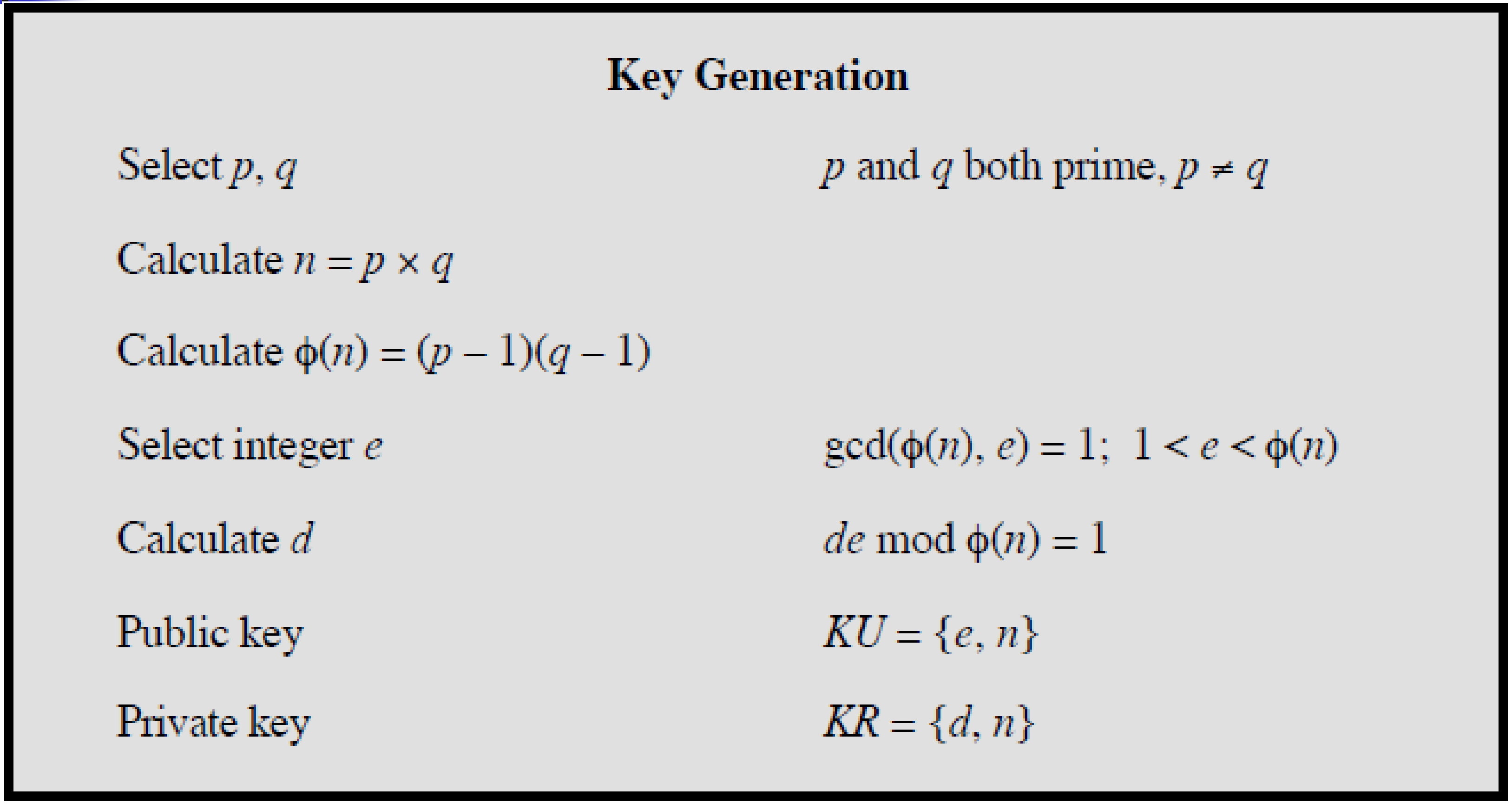
✓ 对所有的M<n,计算M^{e}和C^{d}比较容易;

✓ 给定e,n, 不可能推出d.

RSA公钥加密算法

- 发现封闭的数学结构能够符合上面的性质：乘法群（选择两个素数)
- 相关参数
  - ✓ 开始选择素数p,q
  - ✓ 计算n, Ø(n)
  - ✓ 对于选定的e,计算e关于模Ø(n)的乘法逆元d.

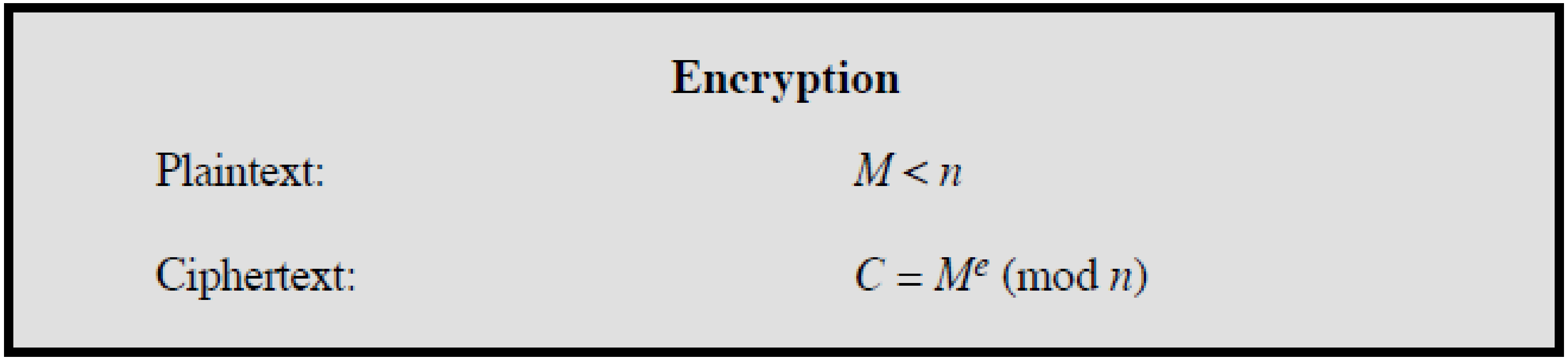
RSA公钥加密算法



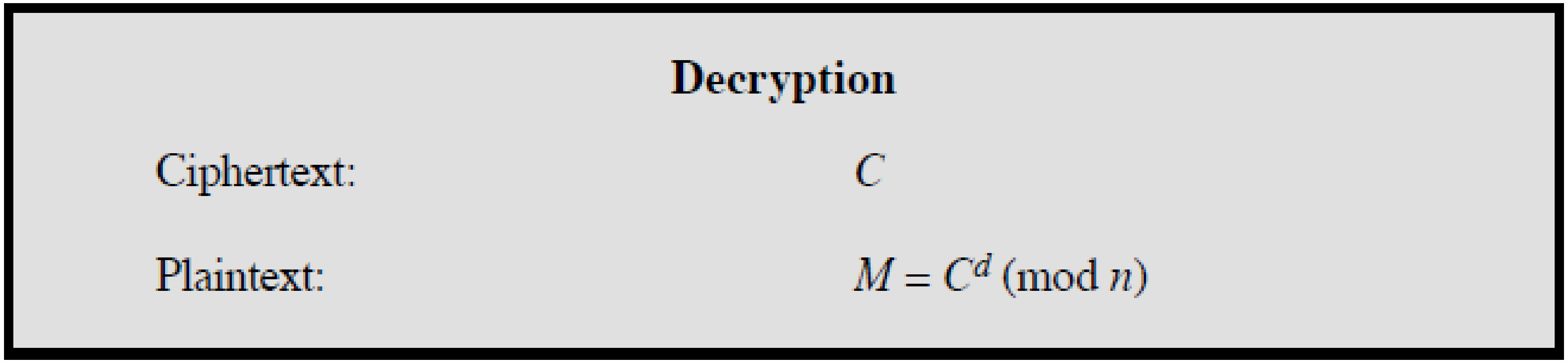
Key Generation

Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calculate $\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\phi(n), e) = 1;\ 1 < e < \phi(n)$
Calculate d	$de \bmod \phi(n) = 1$
Public key	$KU = \{e, n\}$
Private key	$KR = \{d, n\}$

RSA公钥加密算法



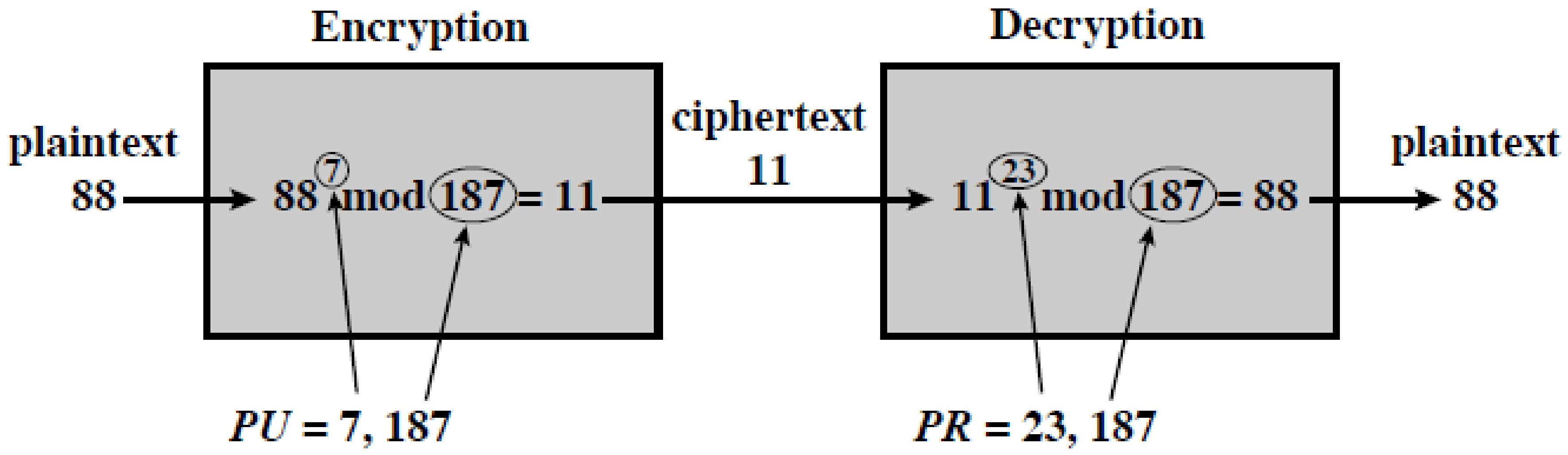
Encryption	
Plaintext:	$M < n$
Ciphertext:	$C = M^e \pmod{n}$



Decryption	
Ciphertext:	C
Plaintext:	$M = C^d \pmod{n}$

RSA公钥加密算法

- RSA算法实例

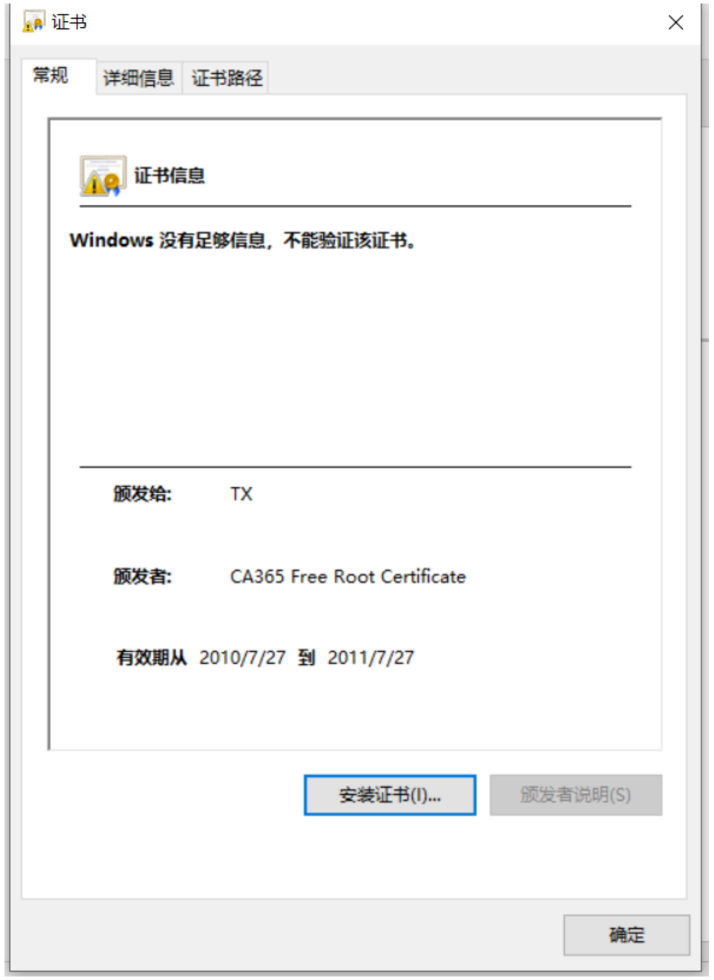


数字证书

X.509是密码学里公钥证书的格式标准。X.509证书已应用在包括TLS/SSL在内的众多网络协议里

- 证书
  - 版本号
  - 序列号
  - 签名算法
  - 颁发者
  - 证书有效期主题
  - 主题公钥信息
  - 颁发者唯一身份信息（可选项）
  - 主题唯一身份信息（可选项）
  - 扩展信息（可选项）…
- 证书签名算法
- 数字签名

数字证书


证书

常规 | 详细信息 | 证书路径

证书信息

Windows 没有足够信息，不能验证该证书。

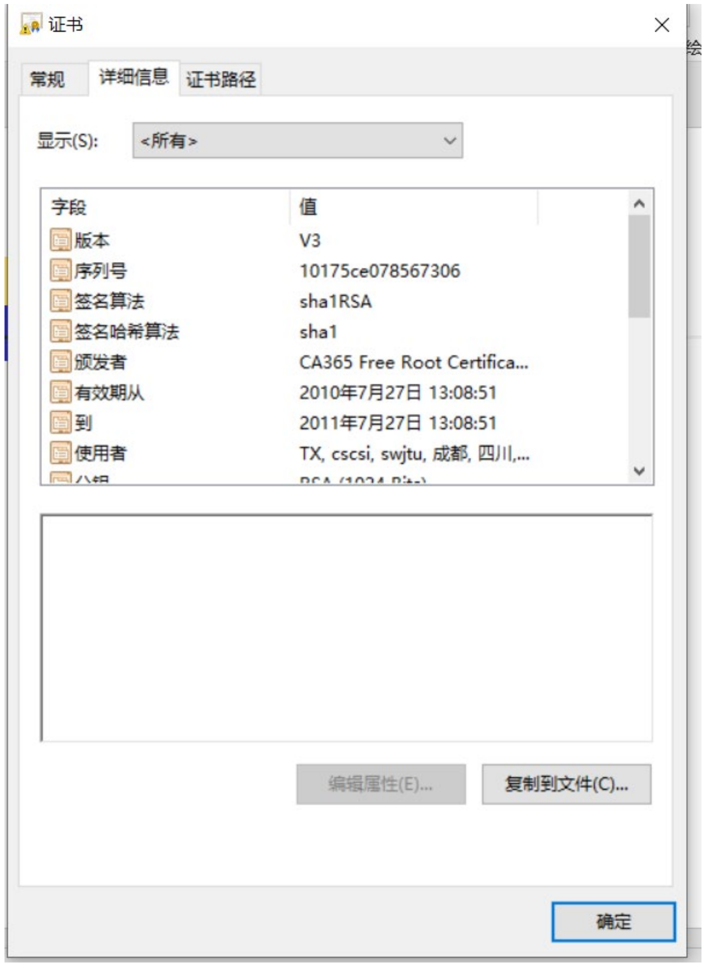
颁发给: TX

颁发者: CA365 Free Root Certificate

有效期从 2010/7/27 到 2011/7/27

安装证书(I)... 颁发者说明(S)

确定


证书

常规 | 详细信息 | 证书路径

显示(S): <所有>

字段	值
版本	V3
序列号	10175ce078567306
签名算法	sha1RSA
签名哈希算法	sha1
颁发者	CA365 Free Root Certifica...
有效期从	2010年7月27日 13:08:51
到	2011年7月27日 13:08:51
使用者	TX, cscsi, swjtu, 成都, 四川,...

编辑属性(E)... 复制到文件(C)...

确定

椭圆曲线加密技术

- ECC(Elliptive Curve Cryptosystem)

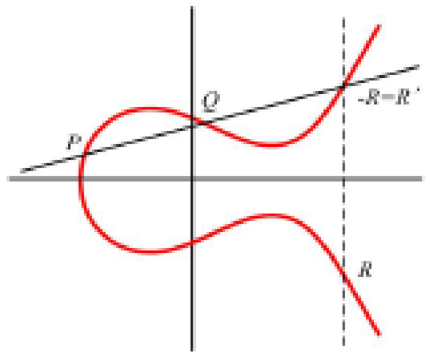
椭圆曲线E是一个光滑的**Weierstrass**方程在 P²(K)中的全部解集合。

$$Y^2Z+a_1XYZ+a_3YZ^2=X^3+a_2X^2Z+a_4XZ^2+a_6Z^3$$

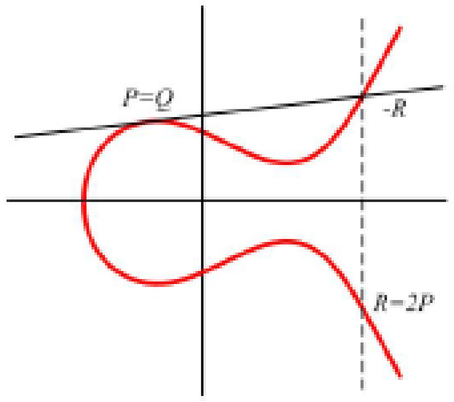
Elliptic Curve Cryptosystem

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

$(E, +, \mathcal{O})$ is an abelian group where the infinity point is the identity in the group.



(a)



(b)

Point addition and point doubling

Elliptic Curve Cryptosystem

- Addition Formulas: Nonbinary case, i.e., F = GF(p); p > 3.
- For P = (x1; y1) and Q =(x2; y2), then P = (x1; y1) and P + Q = R = (x3; y3) for Q !=-P

$$\boxed{\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2 \\ y_3 &= \lambda(x_1 - x_3) - y_1 \end{aligned} \quad \text{where } \lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{if } P \neq Q \\ \frac{3x_1^2 + a}{2y_1} & \text{if } P = Q \end{cases}}$$

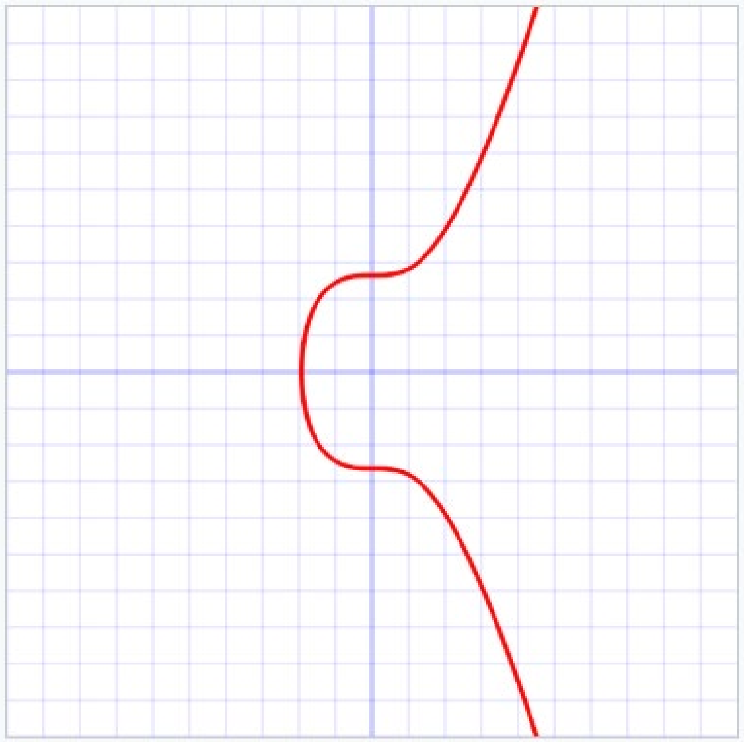
Bitcoin ECDSA- Elliptic curve Digital Signature

- a cryptographic algorithm used by Bitcoin to ensure that funds can only be spent by their rightful owners.
- Secp256k1： used in Bitcoin's public-key cryptography

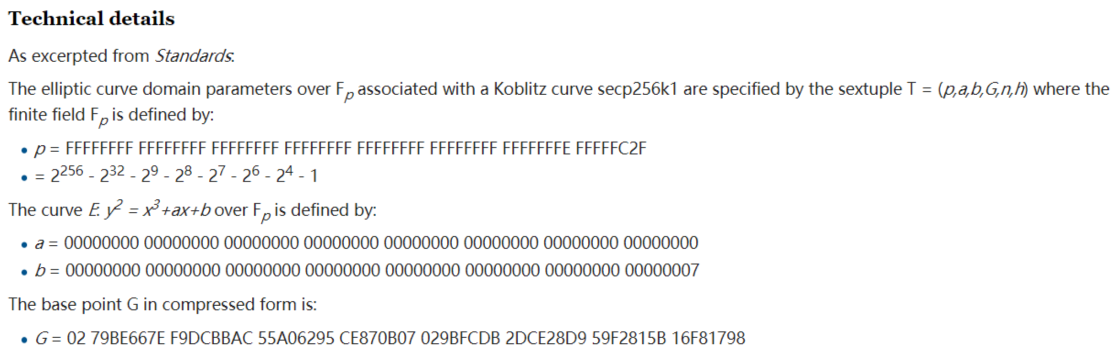
✓ 相比其他椭圆曲线，运行效率高30%

✓ 选取参数公开透明，值得信任

Bitcoin ECDSA



This is a graph of secp256k1's elliptic curve $y^2 = x^3 + 7$ over the real numbers.



Technical details

As excerpted from *Standards*:

The elliptic curve domain parameters over F_p associated with a Koblitz curve secp256k1 are specified by the sextuple T = (p,a,b,G,n,h) where the finite field F_p is defined by:

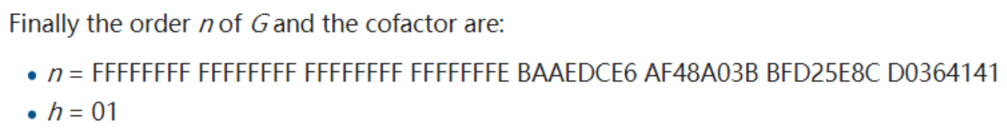
- p = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F
- $= 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$

The curve E: $y^2 = x^3+ax+b$ over F_p is defined by:

- a = 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
- b = 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000007

The base point G in compressed form is:

- G = 02 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798



Finally the order n of G and the cofactor are:

- n = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141
- h = 01

椭圆曲线加密技术

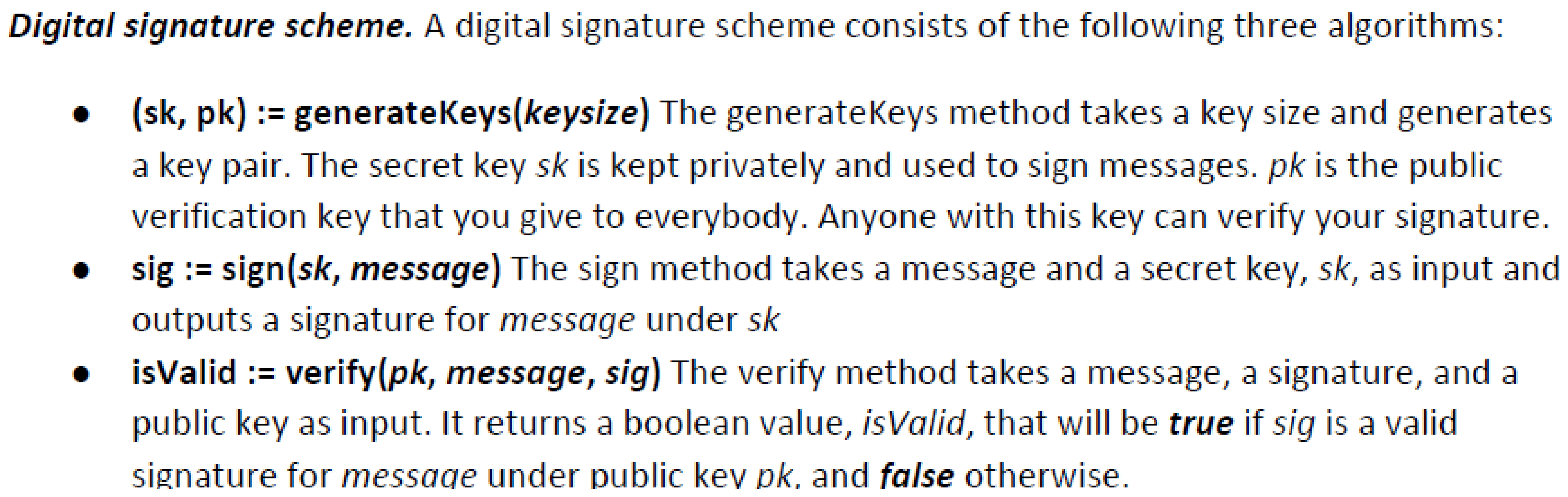
- RSA使用的比特长度不断增加,加大了RSA应用处理的负担。
- ECC的运算虽然本质上与RSA一样，但是群的基本运算更加“复杂”，因此增加了基本单位运算的复杂度。
- RSA 1024 bits~ ECC 160 bits

1.3 数字签名

手写签名的数字模拟

1. 只有你可**制作**你自己的签名
2. 任何其他人都可以**验证**其有效性
3. 签名只与某一特定文件发生**联系**

1.3 数字签名

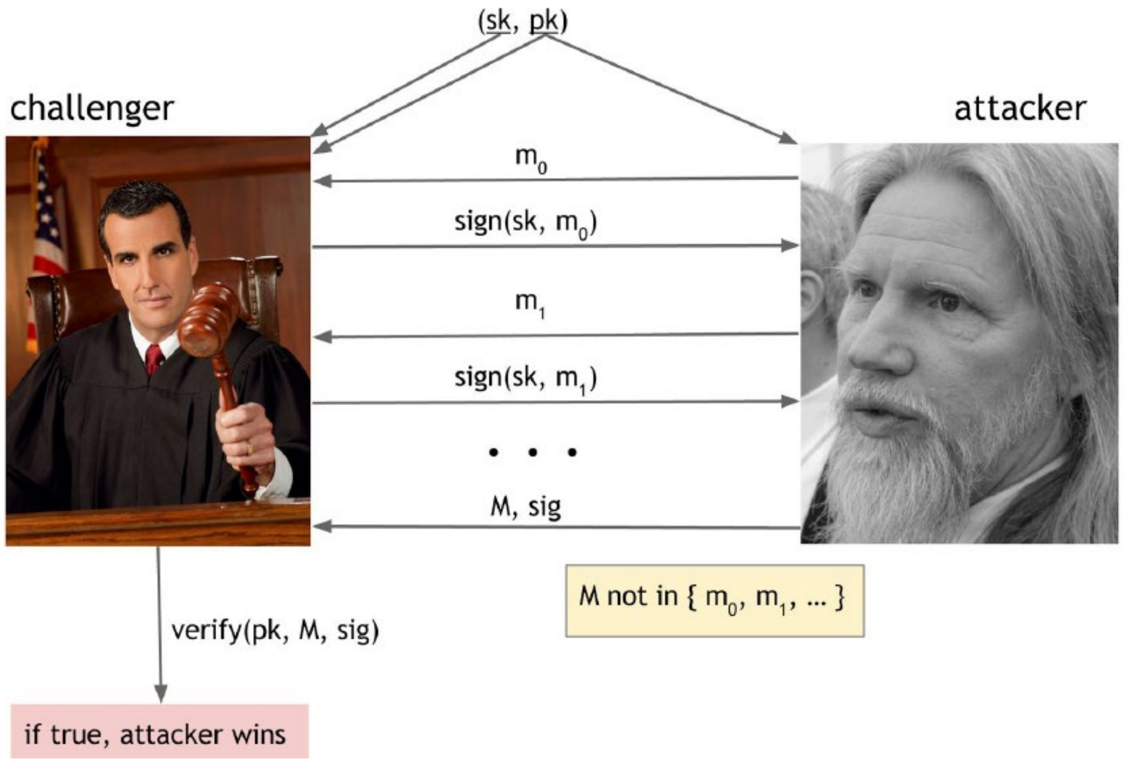


Digital signature scheme. A digital signature scheme consists of the following three algorithms:

- **(sk, pk) := generateKeys(*keysize*)** The generateKeys method takes a key size and generates a key pair. The secret key *sk* is kept privately and used to sign messages. *pk* is the public verification key that you give to everybody. Anyone with this key can verify your signature.
- **sig := sign(*sk*, *message*)** The sign method takes a message and a secret key, *sk*, as input and outputs a signature for *message* under *sk*
- **isValid := verify(*pk*, *message*, *sig*)** The verify method takes a message, a signature, and a public key as input. It returns a boolean value, *isValid*, that will be ***true*** if *sig* is a valid signature for *message* under public key *pk*, and ***false*** otherwise.

1. 有效签名可以通过验证
2. 签名不可以伪造

1.3 数字签名



不可伪造性游戏

1.3 数字签名

注记

- 随机性来源(High entropy)
- Sign the Hash of a Message instead of the Message

数字签名

- 传统签名

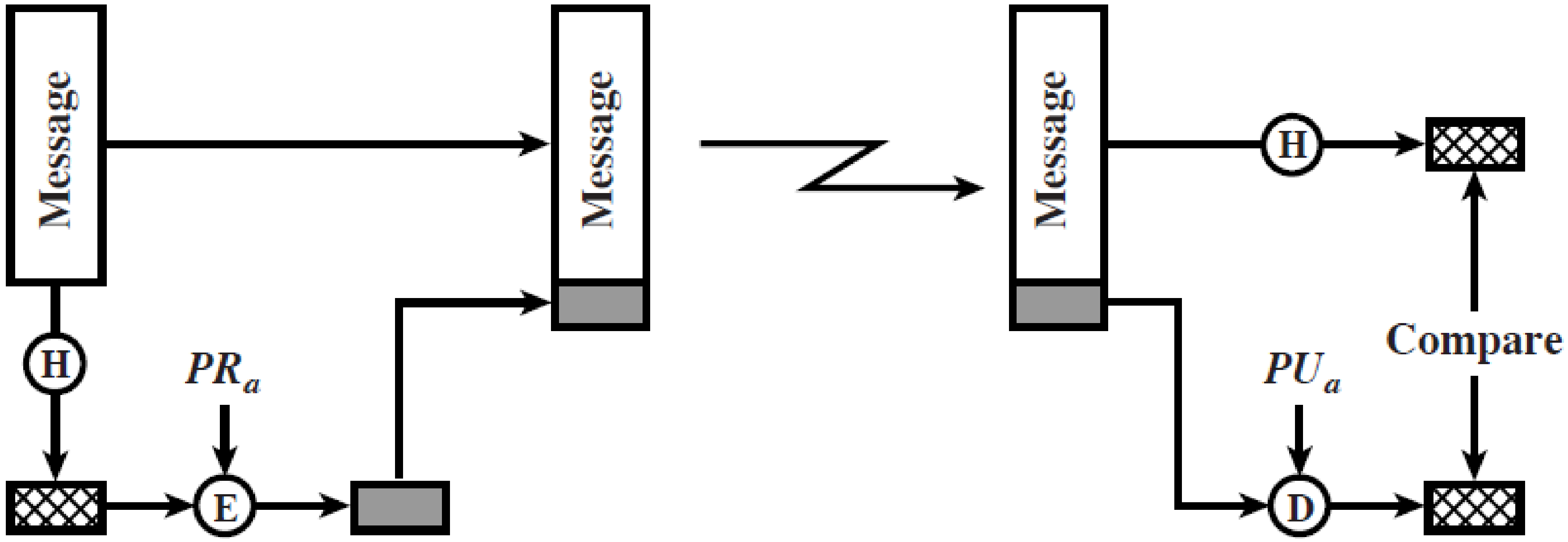
  保证信息真实性，验证签名者**身份**

- 数字签名（又称公钥数字签名）是只有信息的发送者才能产生的别人无法伪造的***一段数字串***，这段数字串同时也是对信息的发送者发送信息真实性的一个有效证明。

数字签名

- 假设Bob->Alice 发送消息
- Bob用*私钥*对消息加密
- Alice用*公钥*对消息验证
- 因为Bob私钥的保密性，其他任何人都不能创建由Bob公钥解密的密文
- 这时整个加密的消息成为一个数字签名（Digital Signature）

数字签名



RSA Digital Signature Algorithm

- Initialization

1. Select two large prime numbers p and q, compute $n = pq$.
2. Select $1 < e < \phi(n)$ with $\gcd(e, \phi(n)) = 1$.
3. Compute $d = e^{-1} \bmod \phi(n)$.
4. Bob's public key: $pk_{Bob} = (n, e)$ with certificate $certi(pk_{Bob})$, and private key $sk_{Bob} = (d, p, q)$.

Signing Message m
Compute: $h(m)$ and $r = h(m)^d \pmod{n}$
r is a digital signature of the message m

Verifying the Digital Signature r of Message m	
Compute:	$r^e \pmod{n}$
Check	$r^e = h(m)$?
If it is true, accept it as a valid signature; otherwise, reject it.	

Elgamal Digital Signature

ElGamal DSA Signing Process: Signer Bob	
a) Randomly pick, i.e., generate by PNG:	$k, 0 < k < p-1$
	with $\gcd(k, p-1) = 1$ (per message)
b) Compute:	$r = g^k$
c) Solve for s in the equation:	$h(m) \equiv xr + ks \pmod{p-1}$, (A)
	(A) is called the signing equation.
If $s = 0$, start the process again.	
The pair (r, s) is a digital signature of the message m: $\underline{Sig_A(m) = (r, s)}$	

ElGamal DSA Verifying Process	
Compute:	$h(m), g^{h(m)}, y^r$ and r^s
Check whether	$g^{h(m)} = y^r r^s$
If it is true, accept it as a valid signature; otherwise, reject it.	

Security: Elgamal Digital Signature

1. Finite field Discrete Logarithm
2. Hash function
3. Randomness of PRNG
4. Solving a linear equation with two unknowns

DSS

Signature Generation: To sign a message m the signer performs the following process

a) Randomly pick or generate by PRNG $k, 0 < k < q$ (per message).

b) Compute $r = g^k$.

c) Solve for s in the equation: $\underline{h(m) \equiv xr + ks \pmod{q}}$.

Then the pair (r, s) is a digital signature of the message m (both r and s are 160-bit numbers).

Verification Process: A signature (r, s) of a message m is verified as follows.

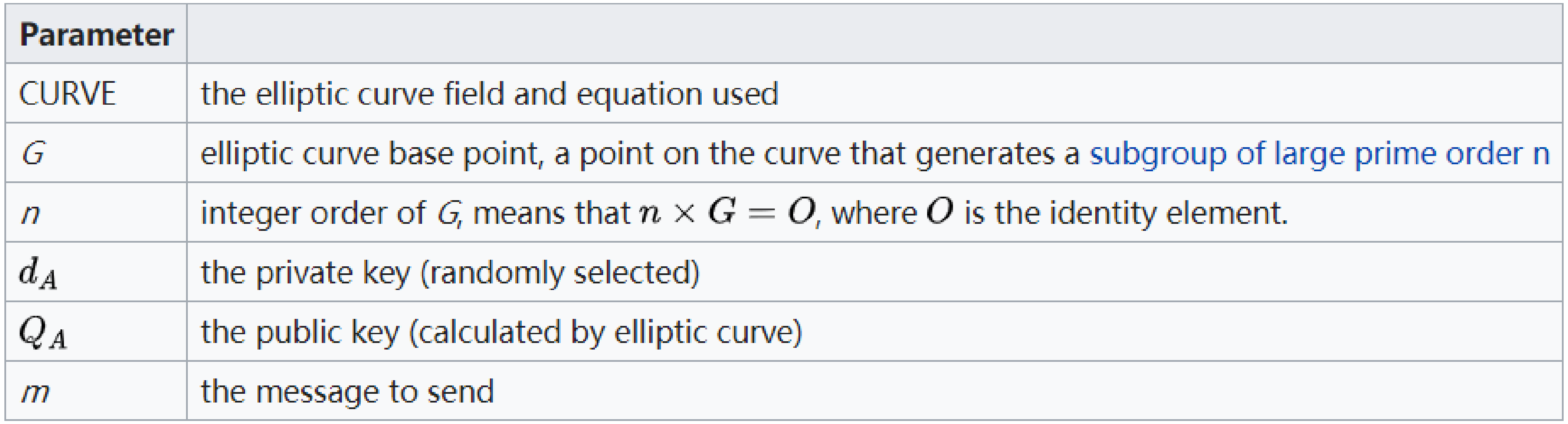
a) Reject the signature if either $0 < r < q$ or $0 < s < q$ is not satisfied.

b) Set $u = h(m)s^{-1} \bmod q$, and $v = -rs^{-1} \bmod q$.

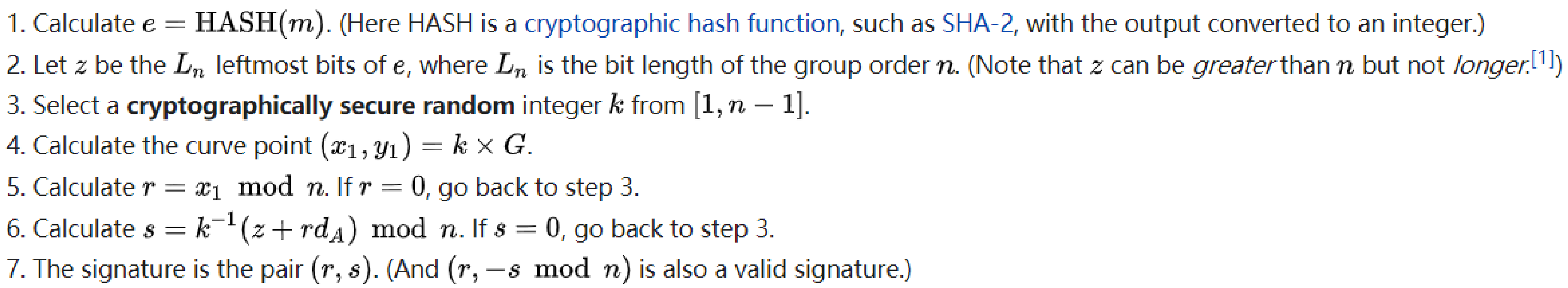
c) Compute $w = (g^u y^v \bmod p) \bmod q$.

d) Check whether $w = r$. If it is true, accept it as a valid signature; otherwise, reject it.

ECDSA

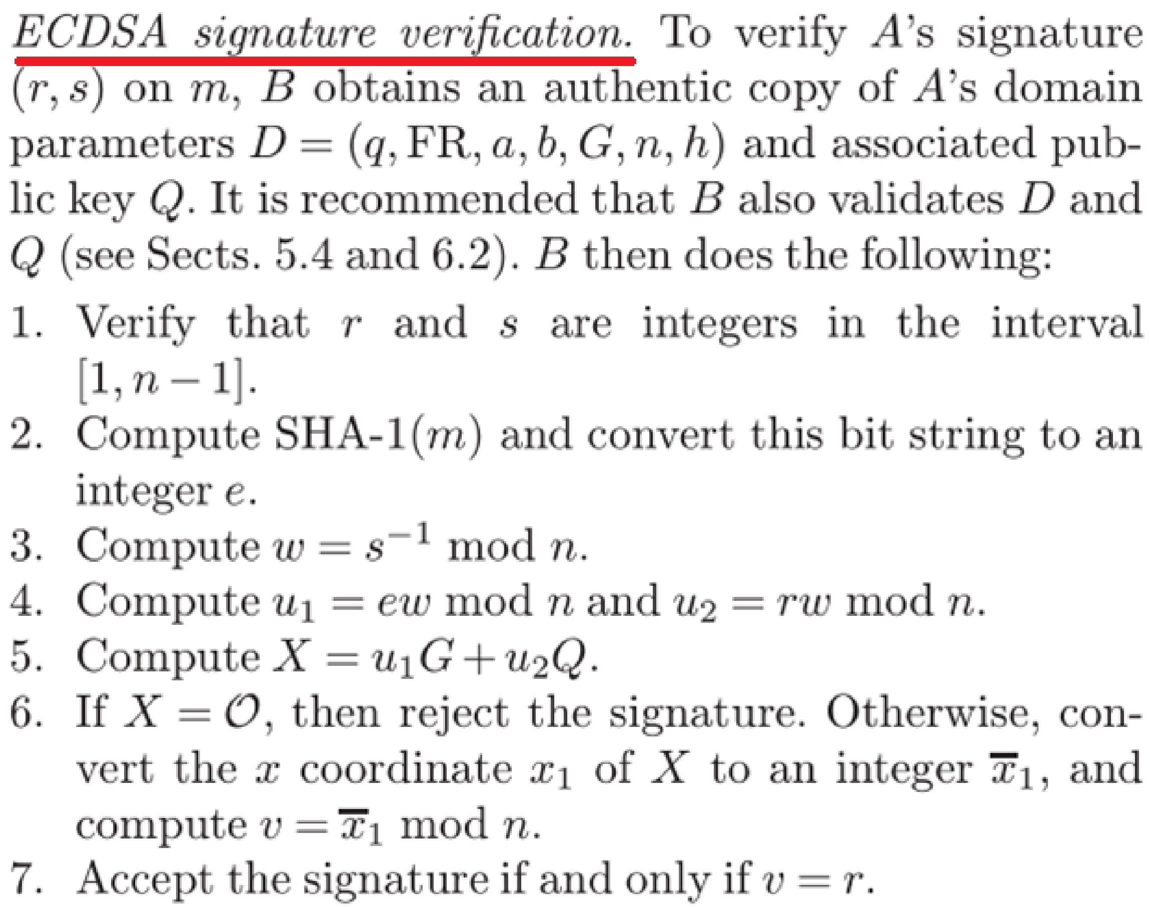


Parameter	
CURVE	the elliptic curve field and equation used
G	elliptic curve base point, a point on the curve that generates a subgroup of large prime order n
n	integer order of G, means that $n \times G = O$, where O is the identity element.
d_A	the private key (randomly selected)
Q_A	the public key (calculated by elliptic curve)
m	the message to send



1. Calculate $e = \mathrm{HASH}(m)$. (Here HASH is a cryptographic hash function, such as SHA-2, with the output converted to an integer.)
2. Let z be the L_n leftmost bits of e, where L_n is the bit length of the group order n. (Note that z can be *greater* than n but not *longer*.[1])
3. Select a **cryptographically secure random** integer k from $[1, n-1]$.
4. Calculate the curve point $(x_1, y_1) = k \times G$.
5. Calculate $r = x_1 \mod n$. If $r = 0$, go back to step 3.
6. Calculate $s = k^{-1}(z + rd_A) \mod n$. If $s = 0$, go back to step 3.
7. The signature is the pair (r, s). (And $(r, -s \mod n)$ is also a valid signature.)

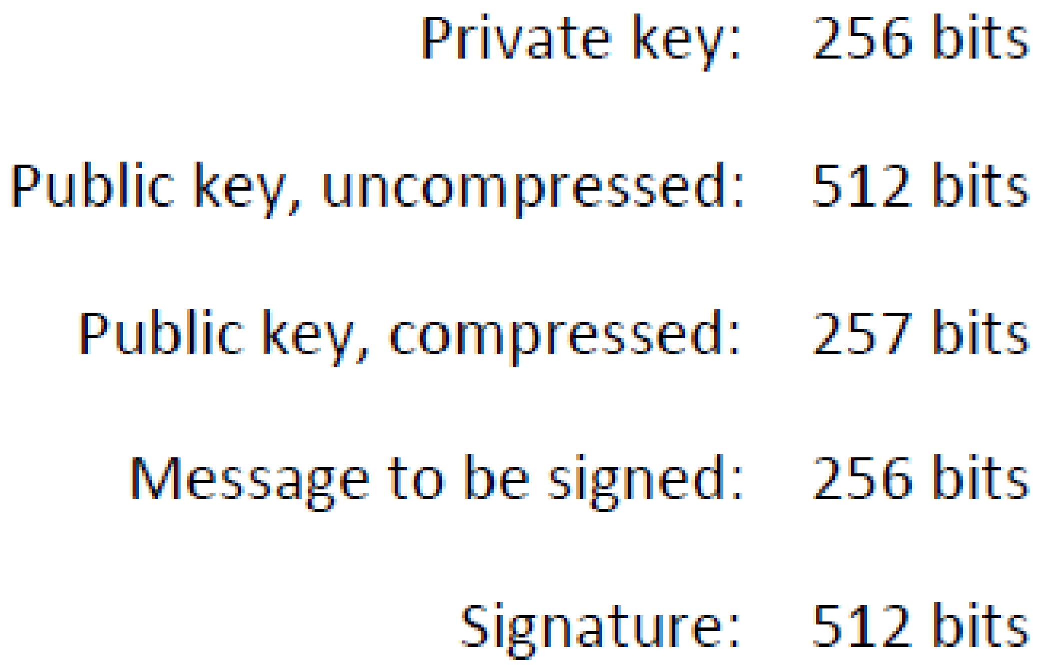
ECDSA



ECDSA signature verification. To verify A's signature (r, s) on m, B obtains an authentic copy of A's domain parameters $D = (q, \mathrm{FR}, a, b, G, n, h)$ and associated public key Q. It is recommended that B also validates D and Q (see Sects. 5.4 and 6.2). B then does the following:

1. Verify that r and s are integers in the interval $[1, n-1]$.
2. Compute SHA-1(m) and convert this bit string to an integer e.
3. Compute $w = s^{-1} \bmod n$.
4. Compute $u_1 = ew \bmod n$ and $u_2 = rw \bmod n$.
5. Compute $X = u_1 G + u_2 Q$.
6. If $X = \mathcal{O}$, then reject the signature. Otherwise, convert the x coordinate x_1 of X to an integer $\overline{x}_1$, and compute $v = \overline{x}_1 \bmod n$.
7. Accept the signature if and only if $v = r$.

ECDSA



Private key:	256 bits
Public key, uncompressed:	512 bits
Public key, compressed:	257 bits
Message to be signed:	256 bits
Signature:	512 bits

Parameters of ECDSA

1.4 公钥即身份

- 身份=地址=公钥的哈希

- 利用GenerateKeys-→（SK, PK）

- 验证身份：

1. 你的身份的确是Hash(PK)
2. （签名）信息能经过公钥PK验证

1.4 公钥即身份

- 去中心化身份管理：

  自己作为用户注册，随时生成一个

- 安全性与随机性

- 虽然真实身份不会泄漏，但是行为模式本身可以识别

1.5两种简单的加密货币

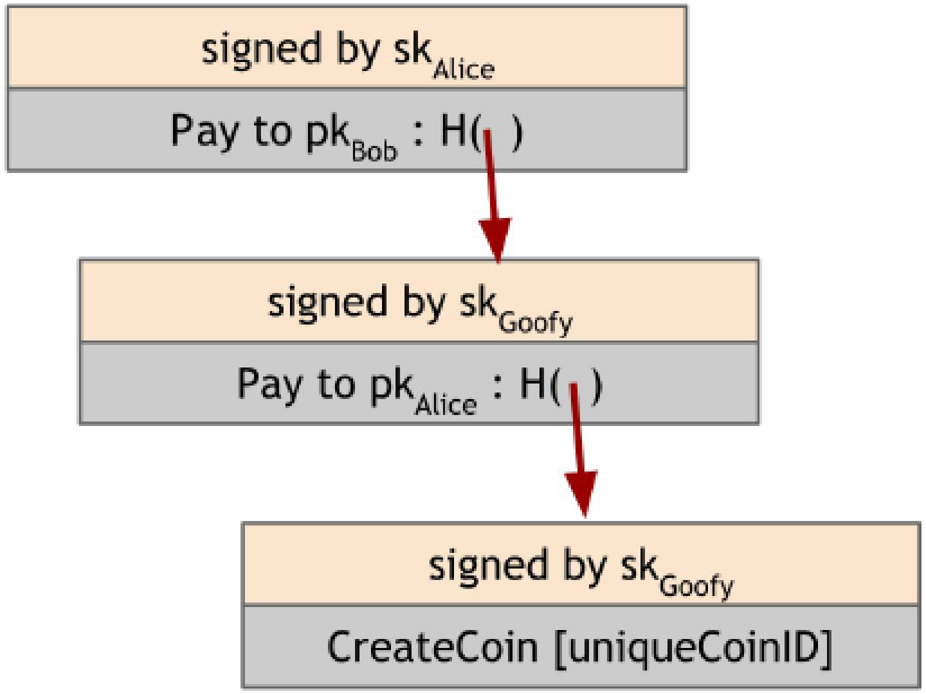
高飞币(GoofyCoin)

- 1.高飞（Goofy）可以随时创建新币，属于它

- 2. 拥有货币的人可以将其转移给其他人

1.5两种简单的加密货币

- 高飞币(GoofyCoin)



1.5 两种简单的加密货币

- 1. 高飞(Goofy)可以签署声明表示他使用唯一的货币编号来创建一个新币
- 2. 货币的所有人可以通过签署声明表示货币的转让，即 “将这个货币转移给X (X为公钥)”
- 3. 任何人可以**追根溯源**，验证一枚货币的有效性

1.5 两种简单的加密货币

高飞币(Goofy)优缺点

缺点：

1. 货币发行权控制在Goofy手中
2. 双重支付(Double Spending)

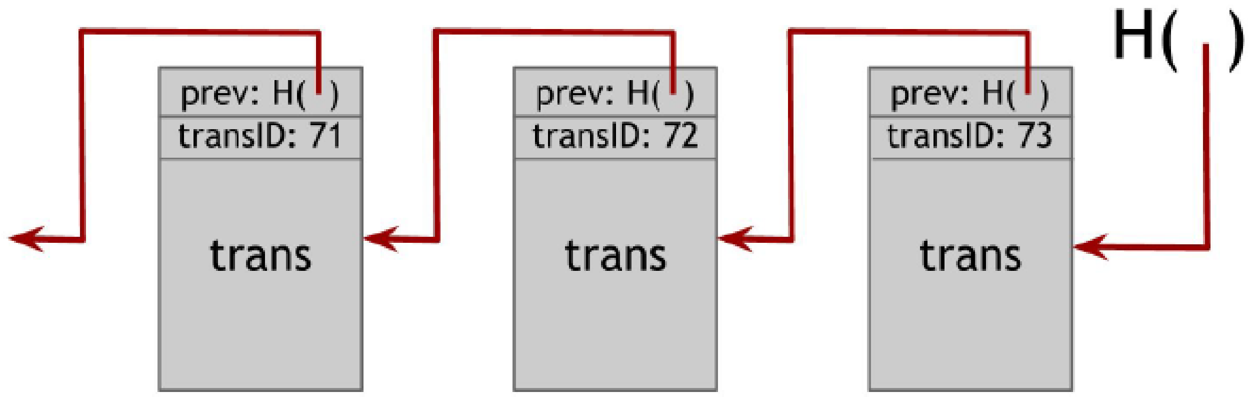
优点：

货币的转让技术

1.5 两种简单的加密货币

财奴币(ScroogeCoin)

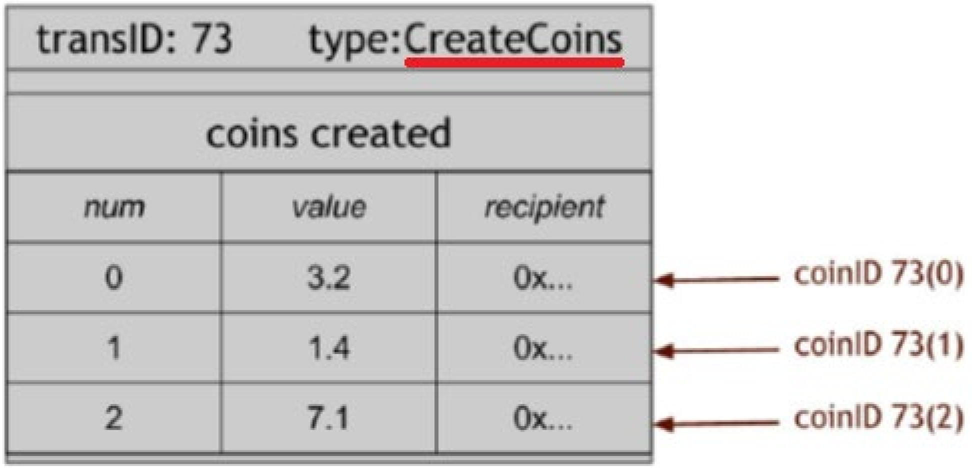
- 解决双重支付问题：Append-only ledger
- 建立一个“区块链”，每个数据块包含一次交易
- Scrooge digitally signs the final hash pointer: 确保不会双重支付



ScroogeCoin block chain.

1.5 两种简单的加密货币

- 财奴币(ScroogeCoin)

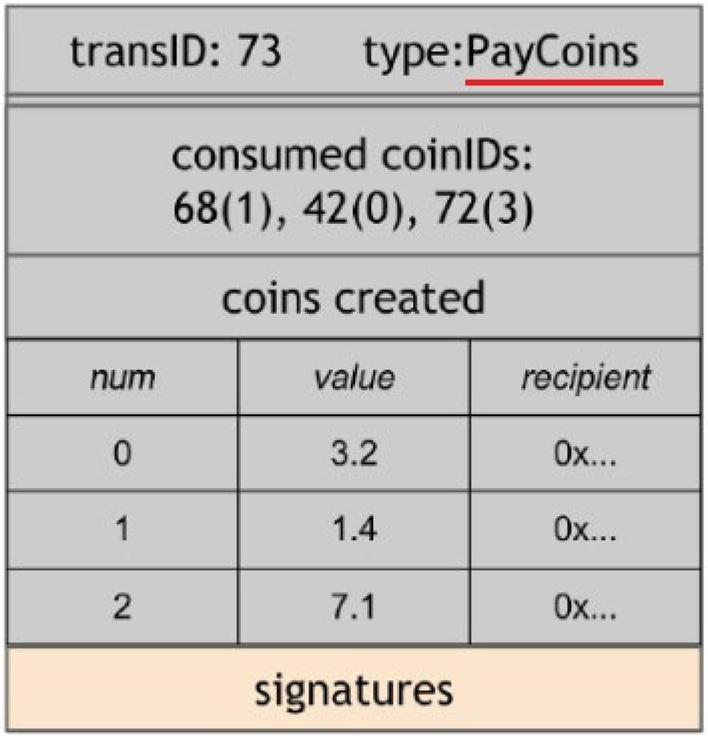


transID: 73 type:CreateCoins

coins created

num	value	recipient
0	3.2	0x...
1	1.4	0x...
2	7.1	0x...

coinID 73(0)
coinID 73(1)
coinID 73(2)



transID: 73 type:PayCoins

consumed coinIDs:
68(1), 42(0), 72(3)

coins created

num	value	recipient
0	3.2	0x...
1	1.4	0x...
2	7.1	0x...

signatures

1.5 两种简单的加密货币

财奴币(ScroogeCoin)

- 交易有效当且仅当

1. 被消耗的货币为有效货币，之前的交易中已经创建
2. 本次交易不是双重支付
3. 本次交易产生的币值等于消耗币值
4. 本次交易消耗的所有货币均有其所有者的有效签名

1.5 两种简单的加密货币

财奴币(ScroogeCoin)优缺点

- 优点

引入了Ledger来防止双重支付

- 缺点：

中心化

财奴可以拒绝公开交易，不提供服务

财奴可以随时造币