比特币关键技术
Bitcoin Key Technology

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)

比特币关键技术

- HASH算法 
- 数字签名 
- 比特币P2P网络
- 共识机制

哈希函数

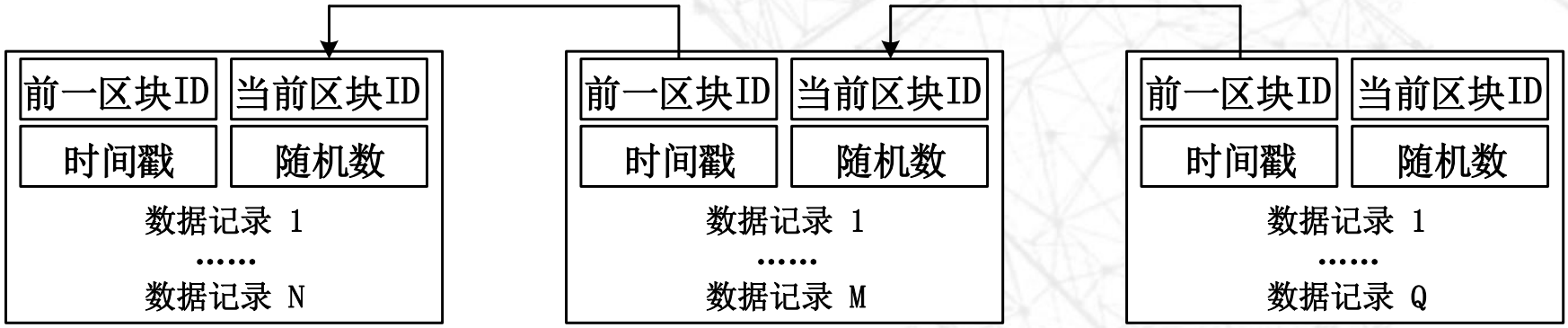
- **定义：**输入任意大小的字符串，产生固定大小的输出
  - **SHA256**（Everything goes well！）= 7465ac6a6f6e86ba6a0564a4158ee02787d5ead5d016b8a2d1abb2818d094adf
- **特征**
  - 正向迅速
  - 逆向困难
  - 输入敏感
  - 强碰撞性

哈希函数

- SHA-256函数在比特币中使用的位置：
  - •工作量证明
  - •比特币公钥转换为公钥哈希
  - •交易的输入输出部分（Transaction id，交易脚本等）
  - •比特币区块头Previous Block Hash
  - •Merkle树

哈希函数

- 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



数字签名

- 签名证明私钥的所有者，即资金所有者，已经授权支出这些资金；
- 授权证明是不可否认的（不可否认性）；
- 签名证明交易（或交易的具体部分）在签字之后没有也不能被任何人修改。

数字签名

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;

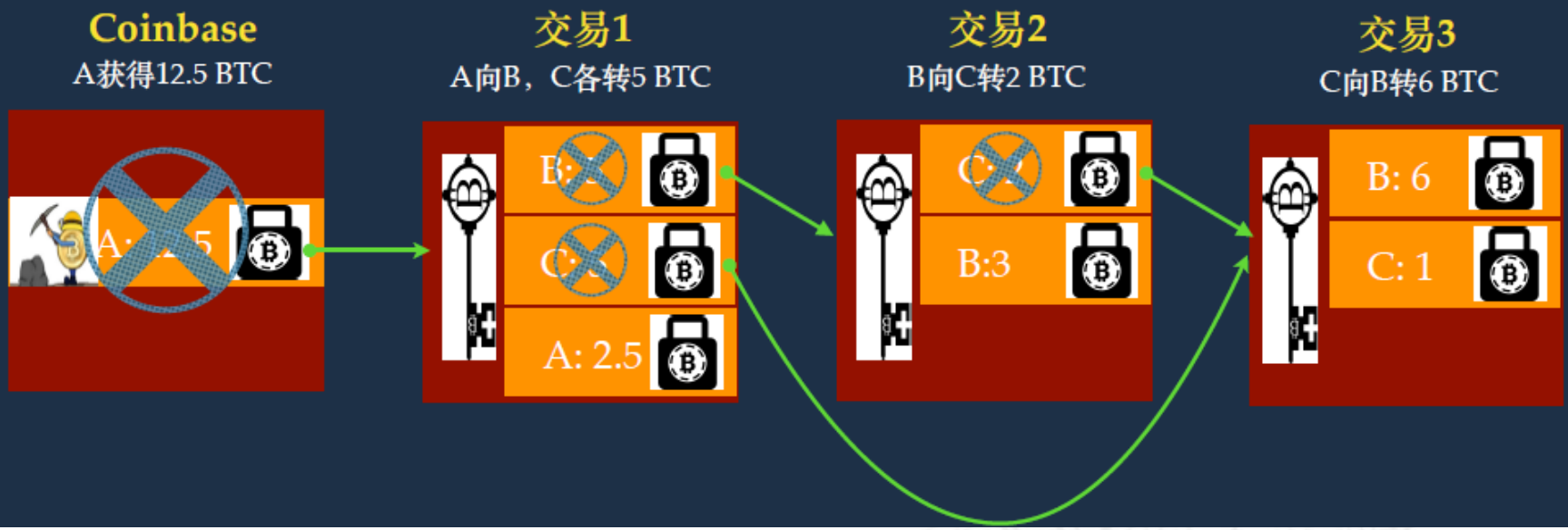
②计算点$R= k*G= (x_R, y_R)$，计算$r = x_R \bmod n$;

③利用私钥d计算$s= k^{-1} * ((H(m) - d * r)) \bmod n$;

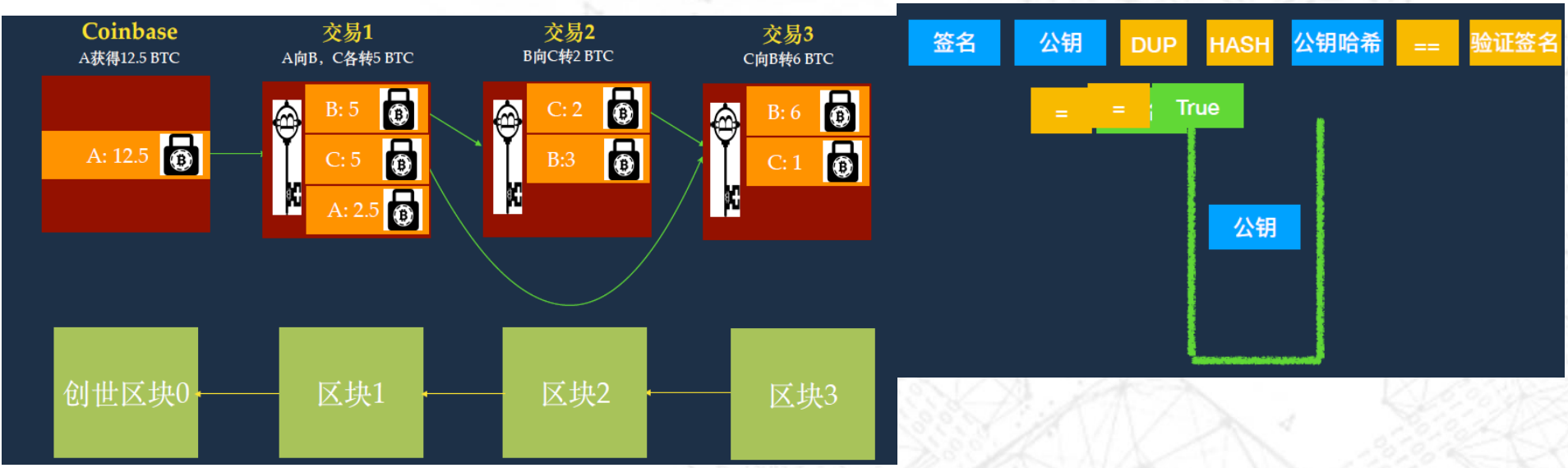
④输入签名(r, s).

交易

- 交易的输入（UTXO+ 解锁脚本）
  - 解锁脚本（签名，发送者公钥）
- 交易的输出 （UTXO ）
  - 锁定的比特币数量
  - 锁定脚本（用接收者的公钥哈希）



交易的确认



- 锁定脚本
  - OP_DUP OP_HASH160 <发送者公钥哈希> OP_EQUALVERIFY OP_CHECKSIG
- 解锁脚本
  - <发送者签名> <发送者公钥>
- 验证脚本： 运行解锁脚本+锁定脚本 => True

数字签名算法

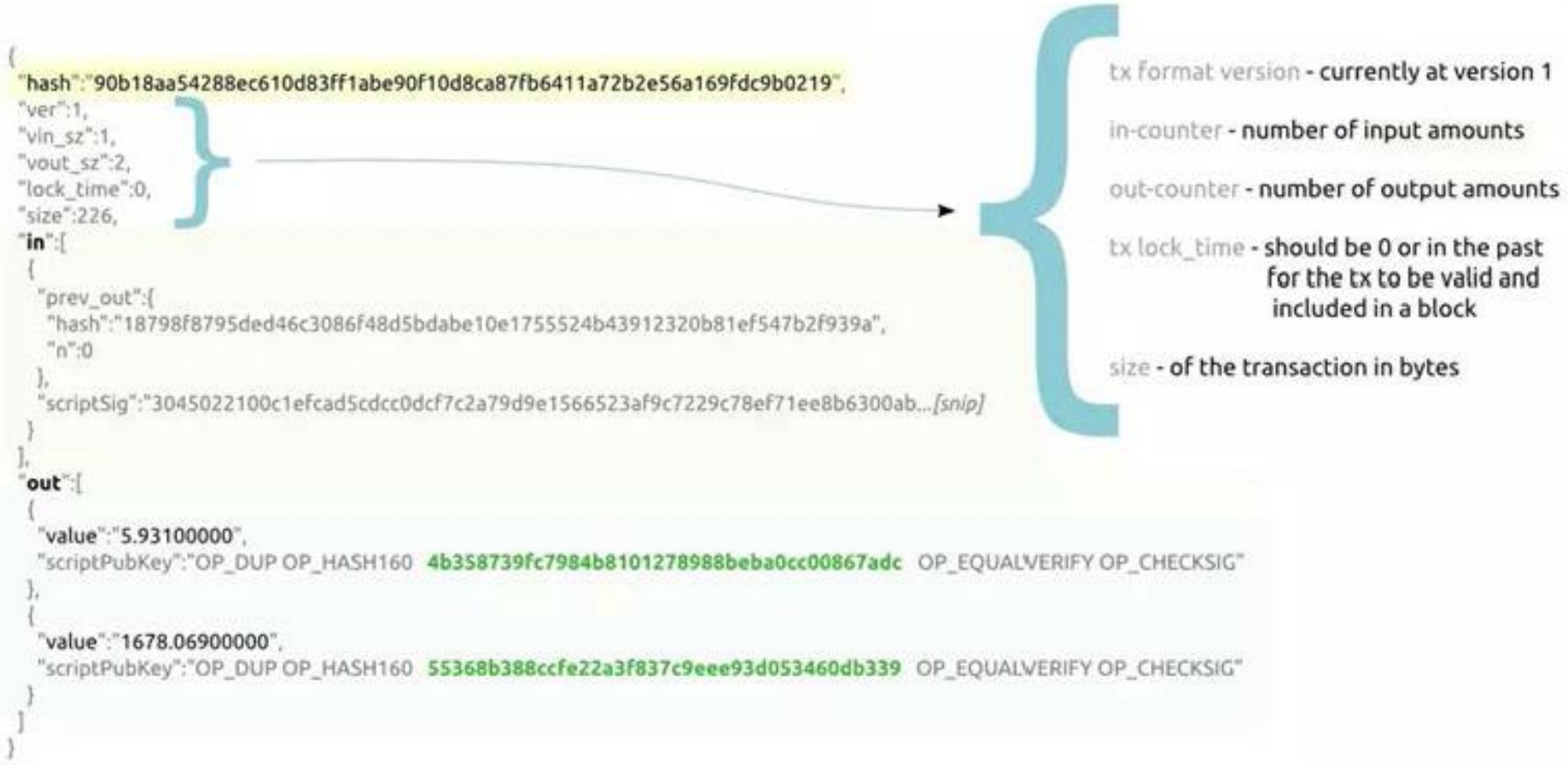
比特币、以太币等数字货币均采用ECDSA算法保证交易的安全性.简单的说法是：用户利用私钥对交易信息进行签名，并把签名发给旷工，旷工通过验证签名确认交易的有效性.

- 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.**

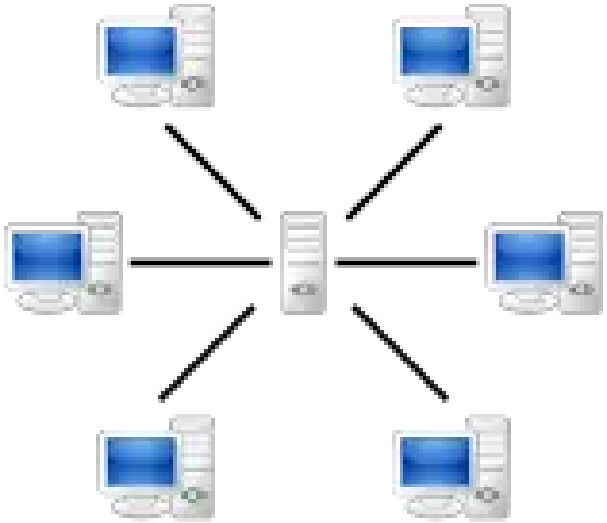

{
"hash":"90b18aa54288ec610d83ff1abe90f10d8ca87fb6411a72b2e56a169fdc9b0219",
"ver":1,
"vin_sz":1,
"vout_sz":2,
"lock_time":0,
"size":226,
"in":[
{
"prev_out":{
"hash":"18798f8795ded46c3086f48d5bdabe10e1755524b43912320b81ef547b2f939a",
"n":0
},
"scriptSig":"3045022100c1efcad5cdcc0dcf7c2a79d9e1566523af9c7229c78ef71ee8b6300ab...[snip]
}
],
"out":[
{
"value":"5.93100000",
"scriptPubKey":"OP_DUP OP_HASH160 4b358739fc7984b8101278988beba0cc00867adc OP_EQUALVERIFY OP_CHECKSIG"
},
{
"value":"1678.06900000",
"scriptPubKey":"OP_DUP OP_HASH160 55368b388ccfe22a3f837c9eee93d053460db339 OP_EQUALVERIFY OP_CHECKSIG"
}
]
}
tx format version - currently at version 1
in-counter - number of input amounts
out-counter - number of output amounts
tx lock_time - should be 0 or in the past for the tx to be valid and included in a block
size - of the transaction in bytes

比特币

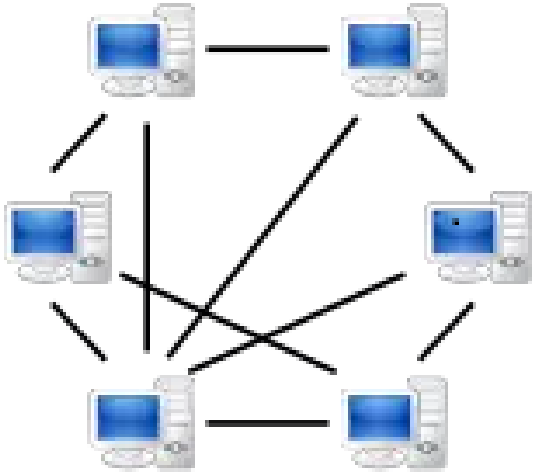
- 去中心化的**对等网络**（比特币协议）
- 公共交易总账（**区块链**）
- 独立交易确认和货币发行的一套规则（**共识机制**）
- 实现有效的区块链全球去中心化共识的机制（**工作量证明算法**）

比特币P2P网络

- 网络架构



有中心服务器的中央网络系统

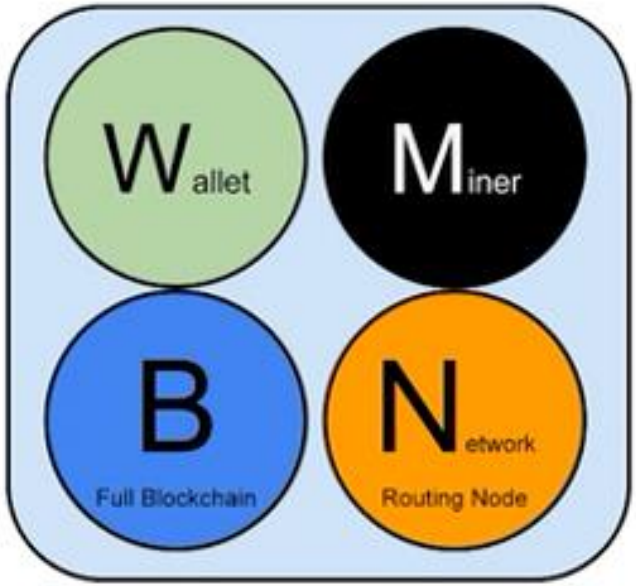


P2P网络

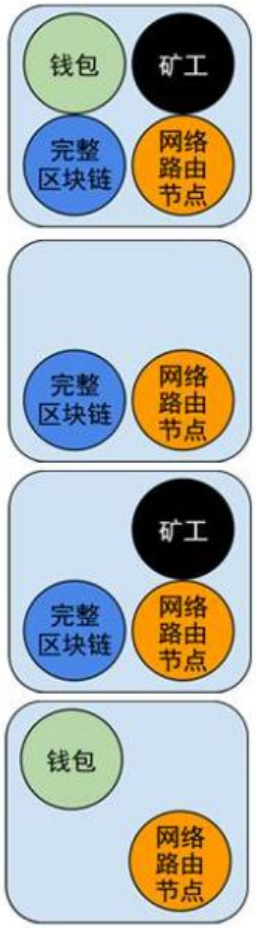
- P2P协议：建立在传输层的TCP协议之上，并采用8333端口进行通信

比特币节点类型/分工

- 钱包管理Wallet
- 挖矿管理Miner
- 区块链数据Full Blockchain
- 网络路由节点Network Routing Node



扩展比特币节点类型/分工



核心客户端 (Bitcoin Core)

在比特币P2P网络中，包含钱包、矿工、完整区块链数据库、网络路由节点。

完整区块链节点

在比特币P2P网络中，包含完整区块链以及网络路由节点。

独立矿工

包含具有完整区块链副本的挖矿功能、以及比特币P2P网络路由节点。

轻量(SPV)钱包

包含不具有区块链的钱包以及比特币P2P网络节点。

矿池协议服务器

将运行其他协议的节点(例如矿池挖矿节点、Stratum节点），连接至P2P网络的网关路由器。

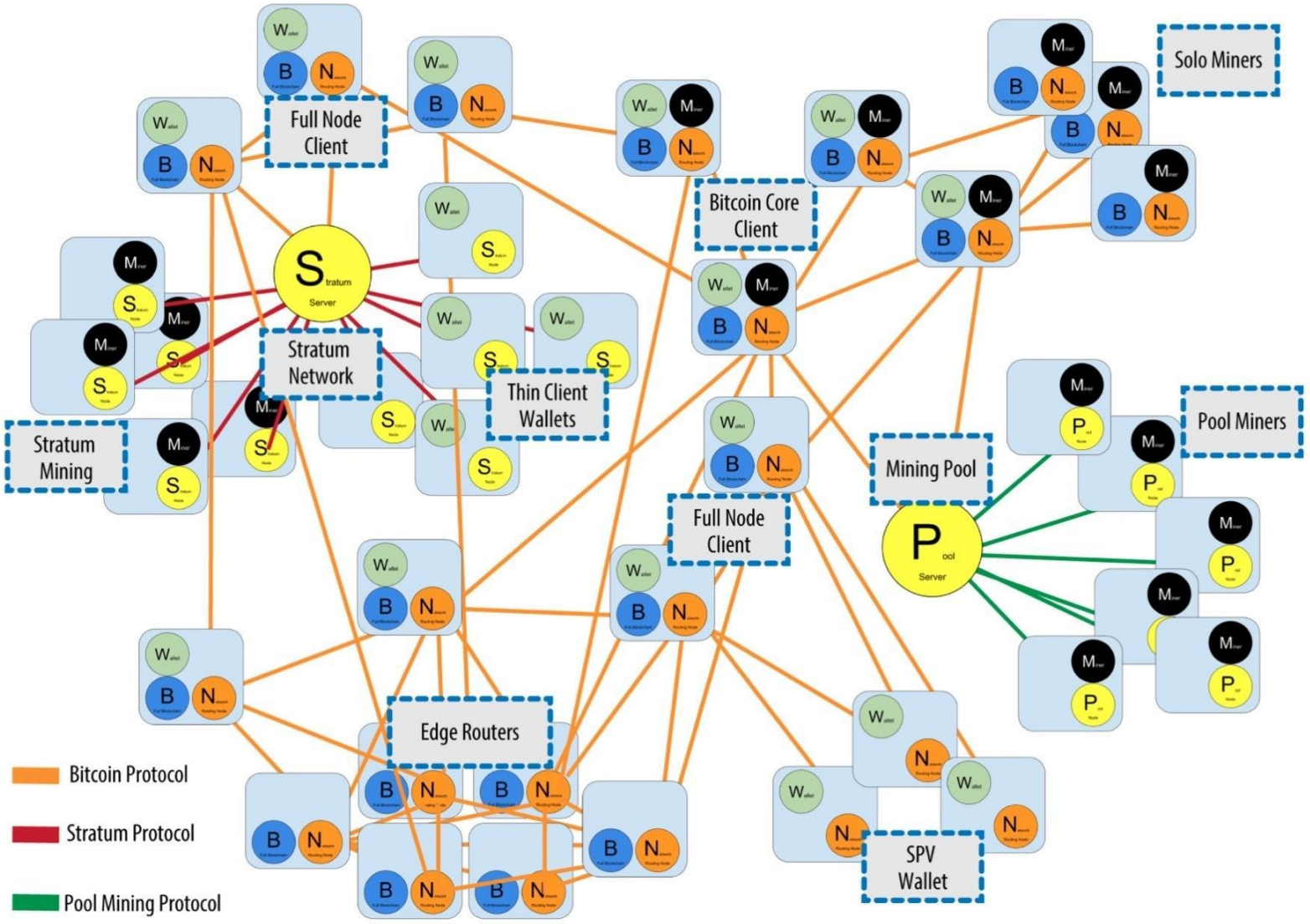
挖矿节点

包含不具有区块链、但具备Stratum协议节点（S）或其他矿池挖矿协议节点（P）的挖矿功能。

轻量(SPV) Stratum 钱包

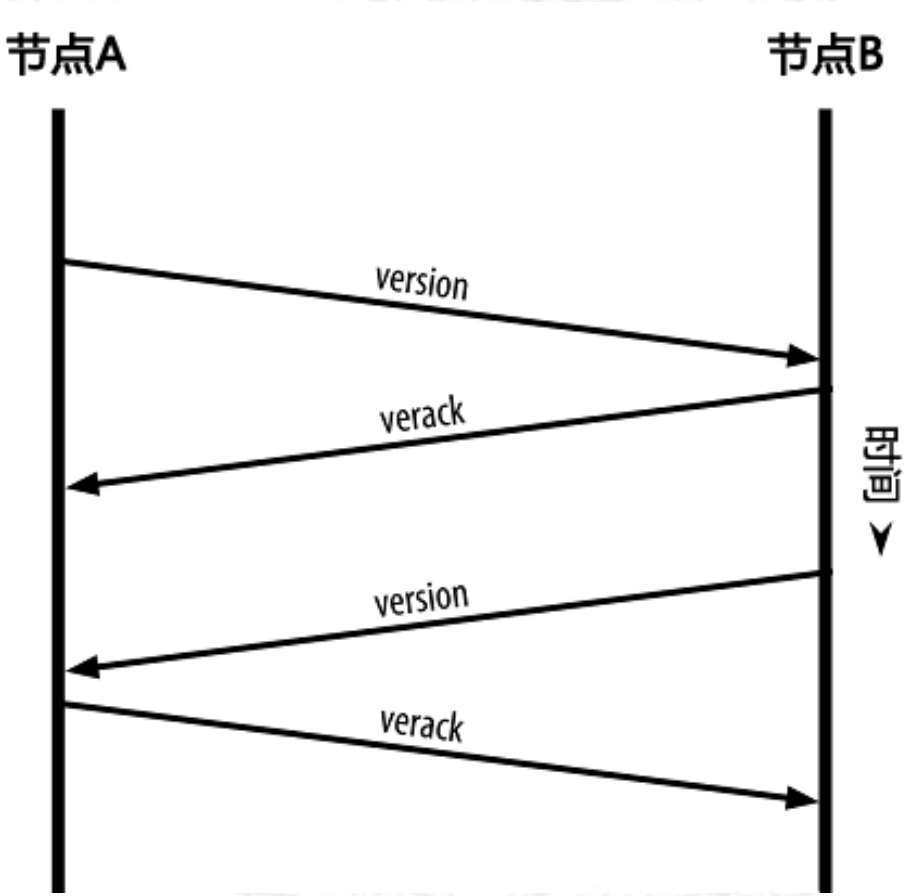
包含不具有区块链的钱包、运行 Stratum 协议的网络节点。

扩展比特币网络


Full Node Client
Solo Miners
Bitcoin Core Client
Stratum Server
Stratum Network
Thin Client Wallets
Stratum Mining
Mining Pool
Pool Miners
Full Node Client
Pool Server
Edge Routers
SPV Wallet
Bitcoin Protocol
Stratum Protocol
Pool Mining Protocol

比特币节点通信

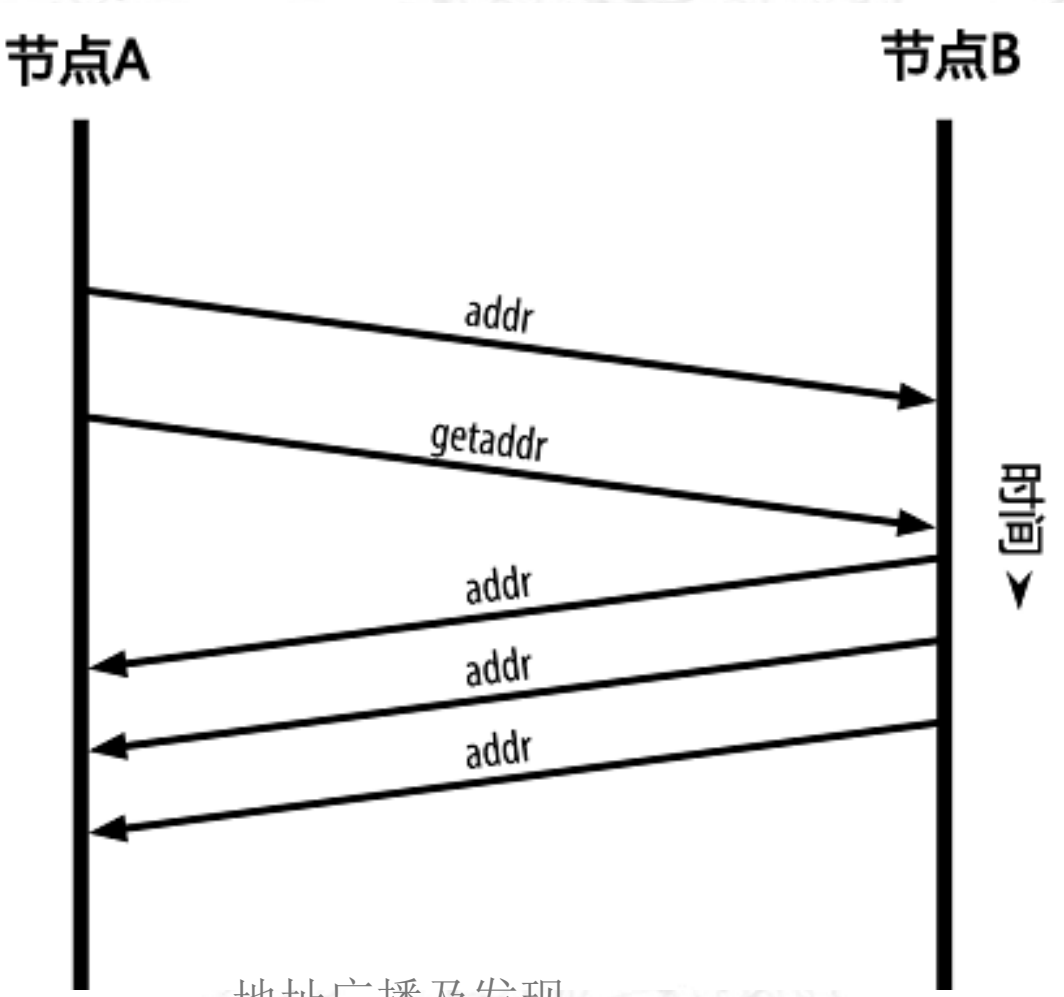
- version：有关程序版本号和区块数量等信息。用于节点之间初次建立连接时交换信息
- verack：当有节点接收到version消息后，如果该节点愿意进行连接，则可以通过回复verack消息进行确认。



对等节点之间的初始“握手”通信

比特币节点通信

● addr
列出一个或多个IP地址和对应的端口号。
● getaddr
包含一些已知的活跃节点，用于请求一个addr消息从而获取活跃节点的IP和端口号，通常在新节点刚加入进行网络初始化启动时（bootstrapping)调用。



地址广播及发现

比特币节点通信

- inv

库存清单，取”inventory”的前3个字母。“我有这些区块/交易：……”，通常在一个新的区块或交易被转发的时候发送inv消息。它只是一个清单，并不包括实际的交易数据。

- getdata

通过hash值请求一个单独的区块或交易。节点收到邻居节点的getdata消息后，向邻居节点返回tx消息或block消息。

- getblocks

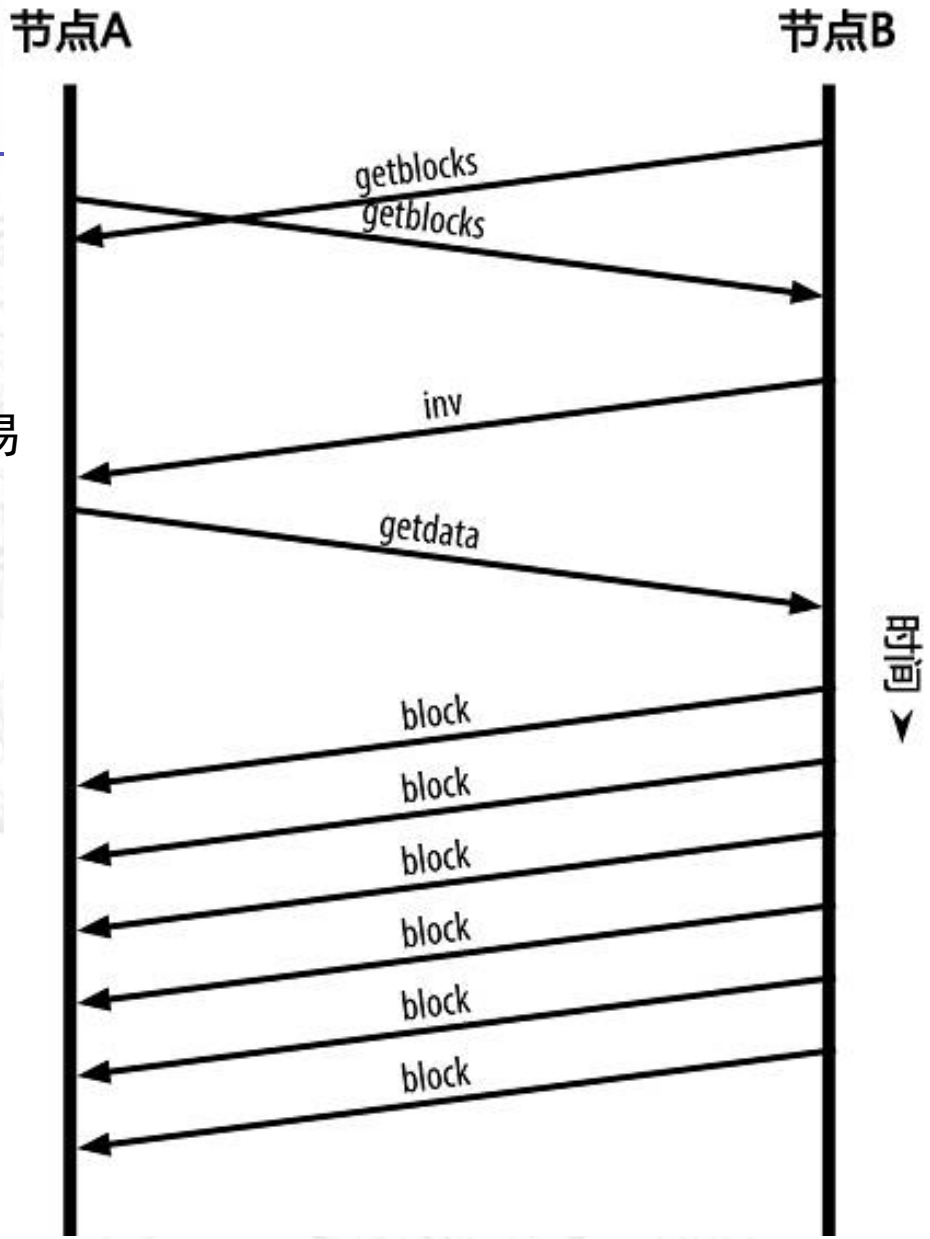
请求inv库存清单中的所有区块。

- tx

发送一笔交易。它只用于响应getdata请求消息。

- block

发送一个区块。同上，它只用于响应getdata请示消息。



比特币节点通信

- getheaders

请求获取指定范围内的区块的区块头信息。

- headers

发送区块头消息，最多一次能发送2000个区块头。非“全节点”（例如SPV节点）可以用它下载区块头信息，而不用下载完整的区块信息。

- submitorder, checkorder, and reply

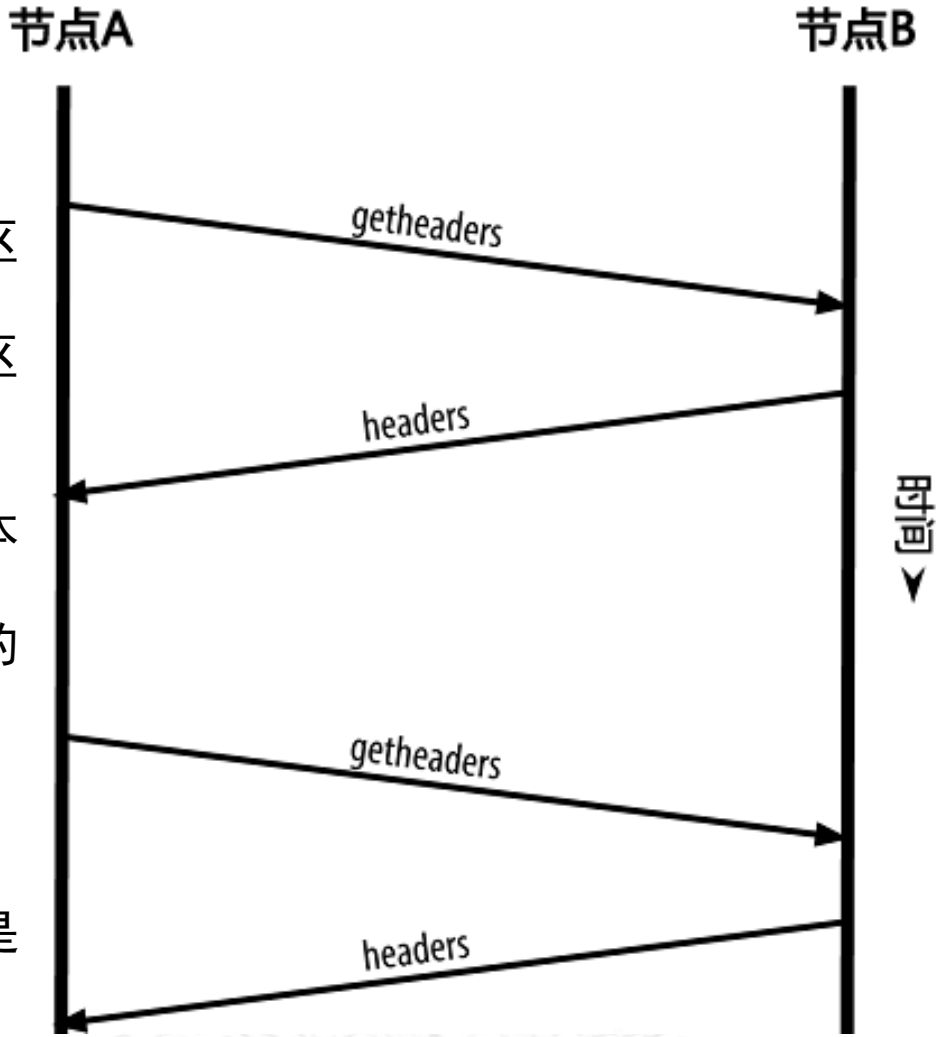
这些消息只在“基于IP交易”的比特币版本中使用，因为它容易受到“中间人攻击”，因此“基于IP交易”的功能在比特币core的v0.8.0版本中被删除了。

- alert

发送网络警告。

- ping

不做任何事情。它只是用来检测网络连接是否仍然有效。如果连接失效会报一个TCP异常。



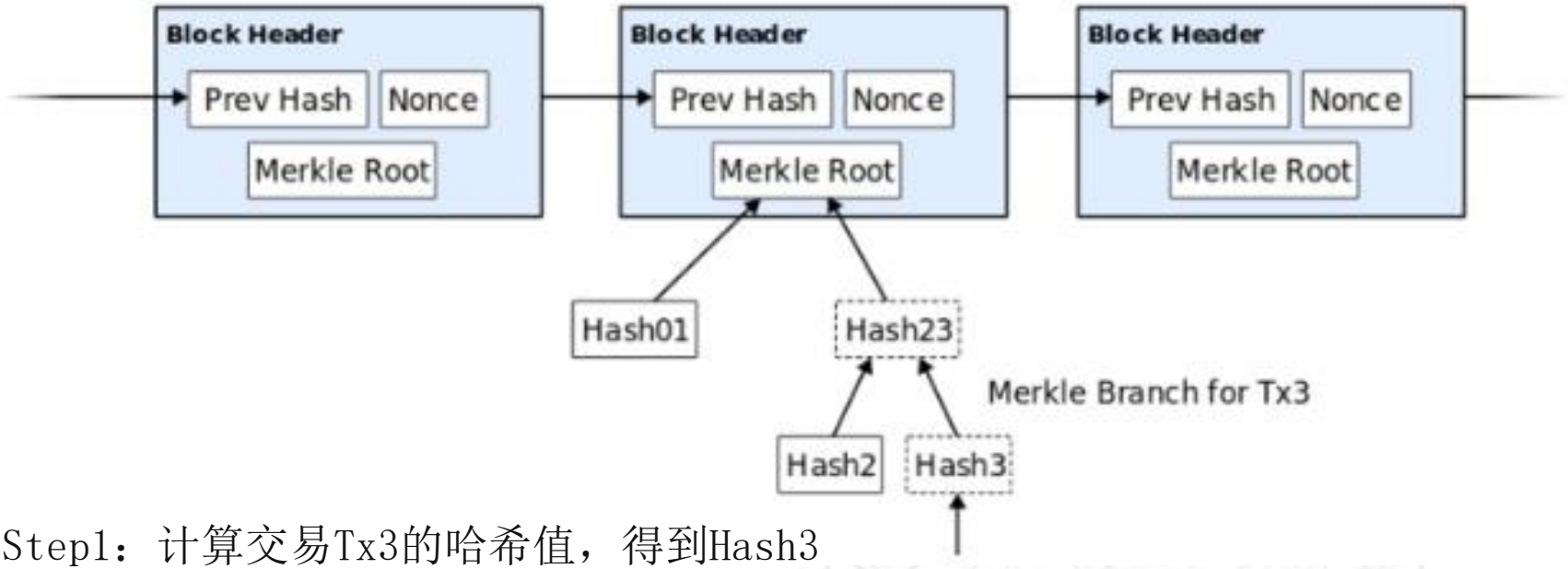
SPV的支付验证

- **SPV节点进行的是支付验证，而不是交易验证。**
- 交易验证：（复杂）
  包括账户余额验证、双花判断等，由全节点完成。
- 支付验证：（简单）
  验证交易是否等到其他节点的共识验证，即是否上链
  验证得到多少确认数（一般是在这笔交易所在区块之后，有6笔交易上链即可确认这笔交易是有效交易）

SPV节点验证过程

- 根据交易体计算待验证支付的交易hash
- SPV节点获取最长链所有区块头、储存在本地
- SPV请求得到merkle树hash认证路径
- 根据hash认证路径，计算得出merkle根hash值，与本地区块头中的merkle根hash做对比，找到待验证交易所在的区块
- 验证该区块是否在最长链上、得到6次以上的确认

区块链



•Step1：计算交易Tx3的哈希值，得到Hash3

•Step2：通过Hash2和Hash3的哈希值，得到父节点的哈希值Hash23

•Step3：同上，通过计算Hash23和Hash01哈希值，得到根节点的哈希值。

•Step4：将上一步得到的根哈希值对比区块头中MerkleTree的根哈希值，如果相同，则证明该区块中存在交易Tx3，否则说明不存在。

SPV节点的安全性？

SPV节点

- 加密与认证连接

- **Tor网络传输**

  --Tor代表洋葱路由网络，通过提供匿名，不可追踪和隐私的随机网络路径提供数据的加密和封装；

- **对等认证和加密**

交易池

- **交易池(transcation pool)**
- **孤立交易池(orphan pool)**
- **UXTO池(UTXO pool)**
- 交易池与孤立交易池包含的是**未确认的交易**，而UTXO池包含的是**已确认的交易**。

区块链

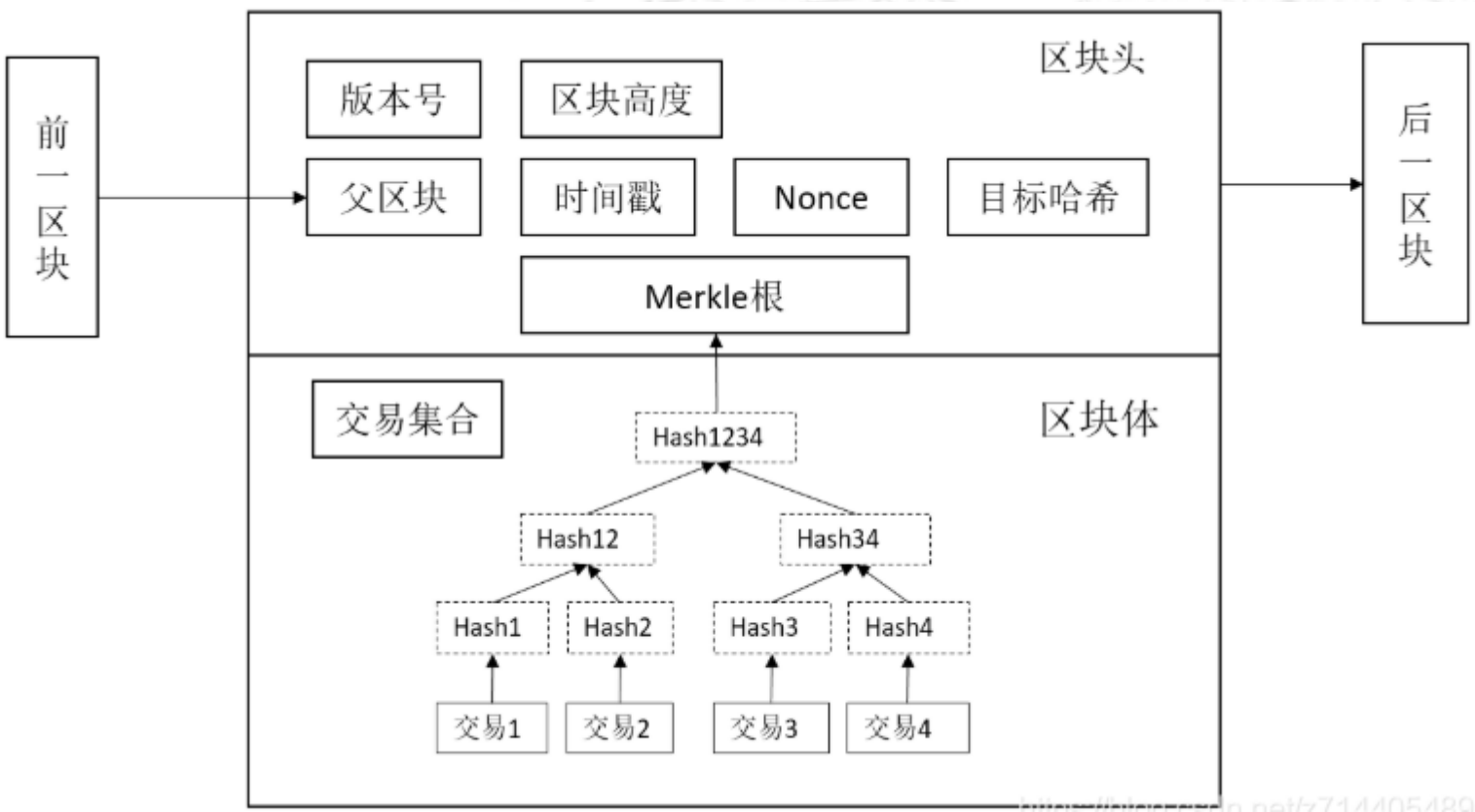
区块信息

Block 654625

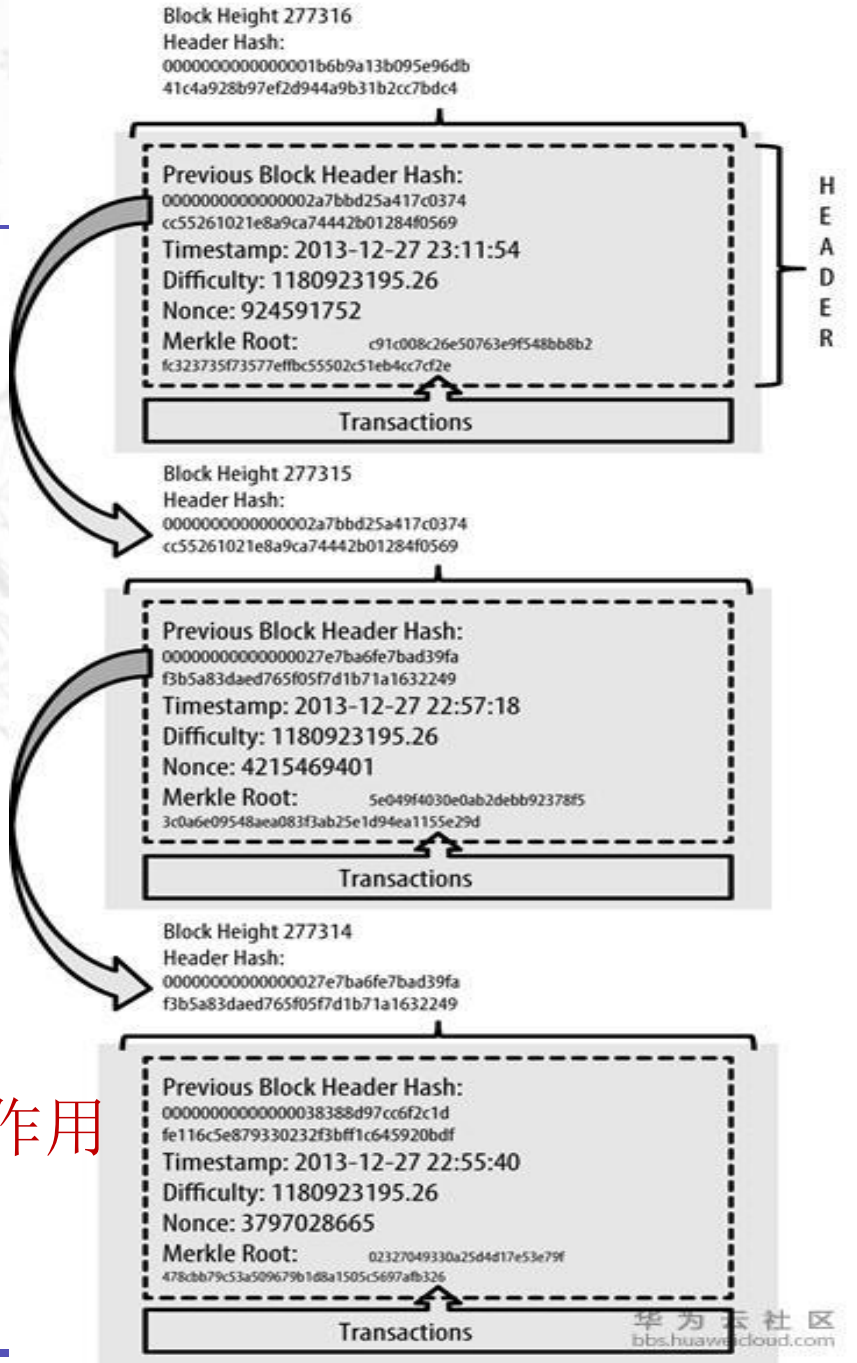
Hash	00000000000000000001a0f03d43f3d7ea2ce13a599880df4d9980c2ce1e8c67
Confirmations	1
Timestamp	2020-10-29 12:14
Height	654625
Miner	F2Pool
Number of Transactions	2,033
Difficulty	19,997,335,994,446.11
Merkle root	8ffb170cfca64226b23bbd98774b87b641ea751a54b2d40e7e55e561f4e1a72e
Version	0x3fffe000
Bits	386,798,414

区块结构

Size	Field	Description
4 bytes	Block Size	The size of the block, in bytes, following this field
80 bytes	Block Header	Several fields form the block header
1–9 bytes (VarInt)	Transaction Counter	How many transactions follow
Variable	Transactions	The transactions recorded in this block

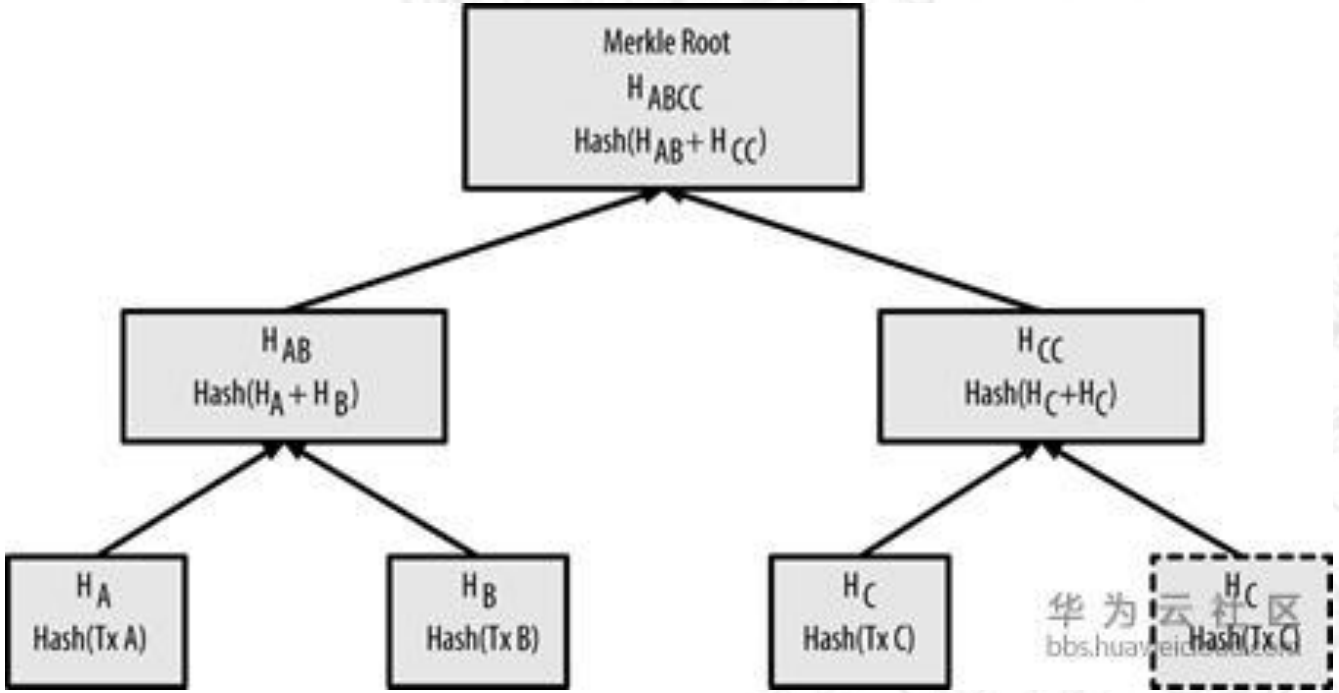

前一区块
区块头
版本号
区块高度
父区块
时间戳
Nonce
目标哈希
Merkle根
后一区块
交易集合
区块体
Hash1234
Hash12
Hash34
Hash1
Hash2
Hash3
Hash4
交易1
交易2
交易3
交易4

区块链

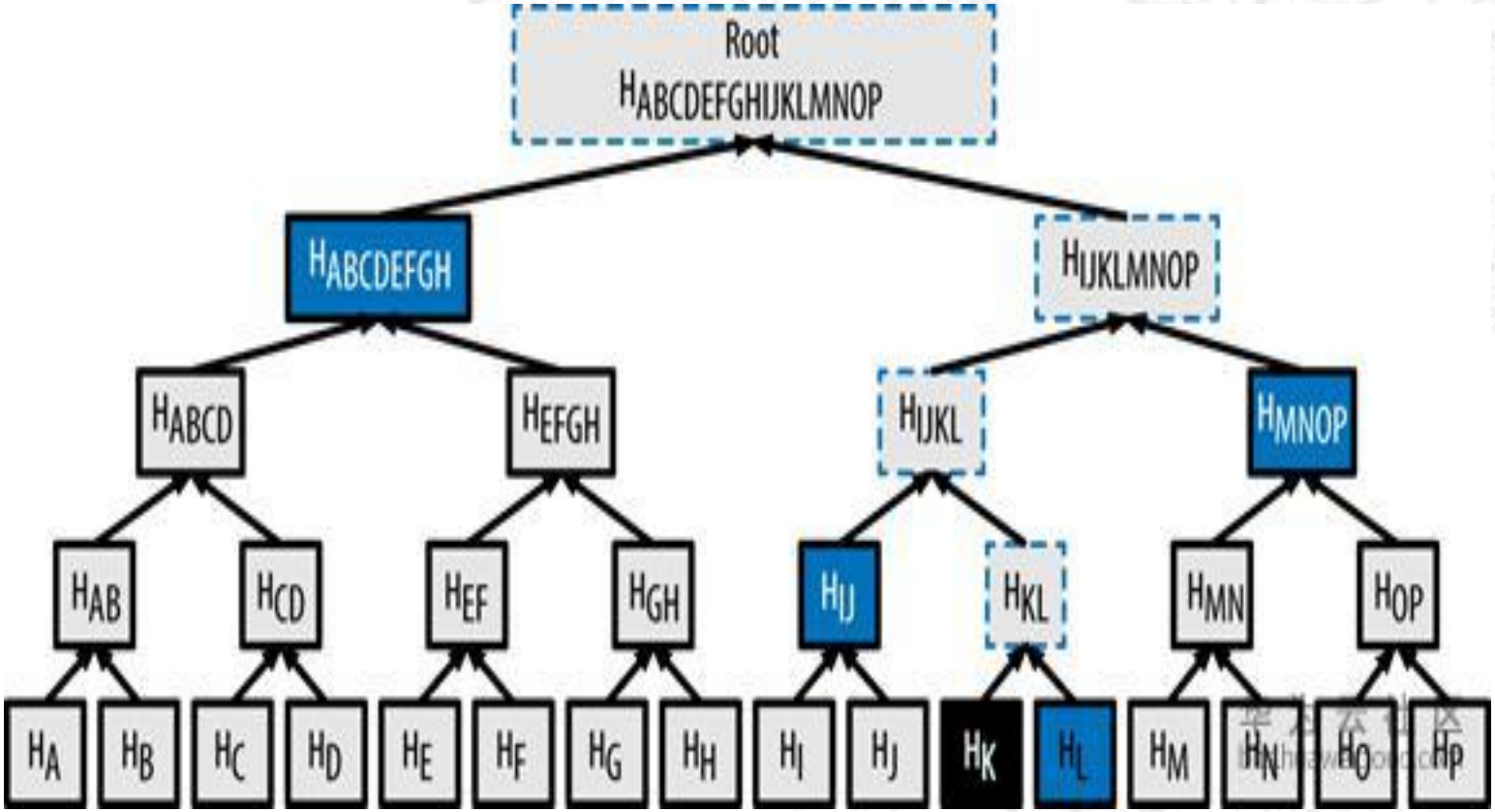


在比特币中，Merkle树的作用是什么呢？

如果交易数量为**奇数**，最后的叶子节点会被复制，组成**偶数**的叶子节点，如下：



如何通过Merkle树验证一笔交易？



- 随着区块中的交易增多，区块大小将线性增长，而Merkle路径的大小增长却极其缓慢

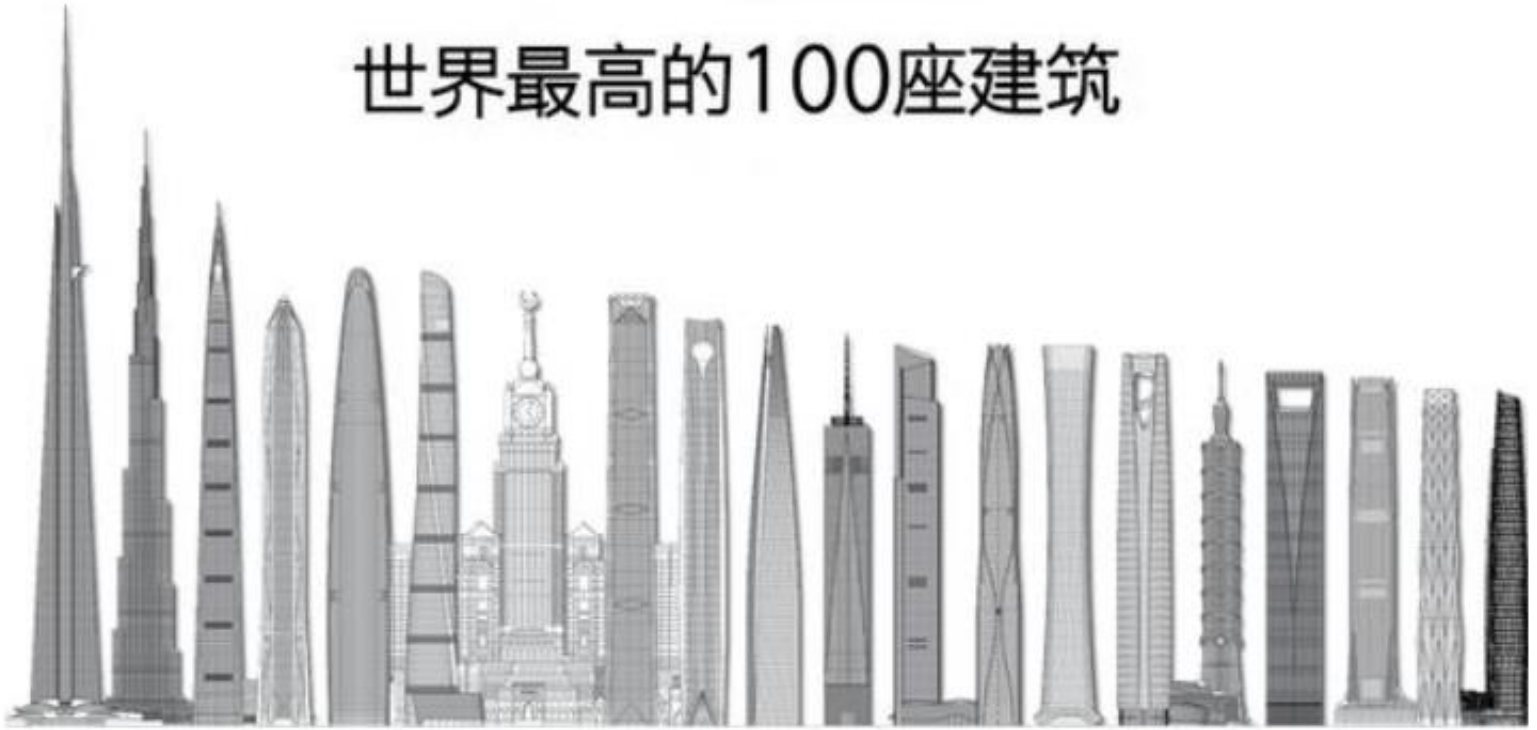
Number of transactions	Approx. size of block	Path size (hashes)	Path size (bytes)
16 transactions	4 kilobytes	4 hashes	128 bytes
512 transactions	128 kilobytes	9 hashes	288 bytes
2048 transactions	512 kilobytes	11 hashes	352 bytes
65,535 transactions	16 megabytes	16 hashes	512 bytes

共识机制

- 比特币网络是完全公开的，任何人都可以匿名接入
  - 共识协议的稳定性和防攻击性很关键
  - 比特币区块链采用工作量证明（PoW）机制来实现共识

- PoW=Power of Work

什么是工作量证明？

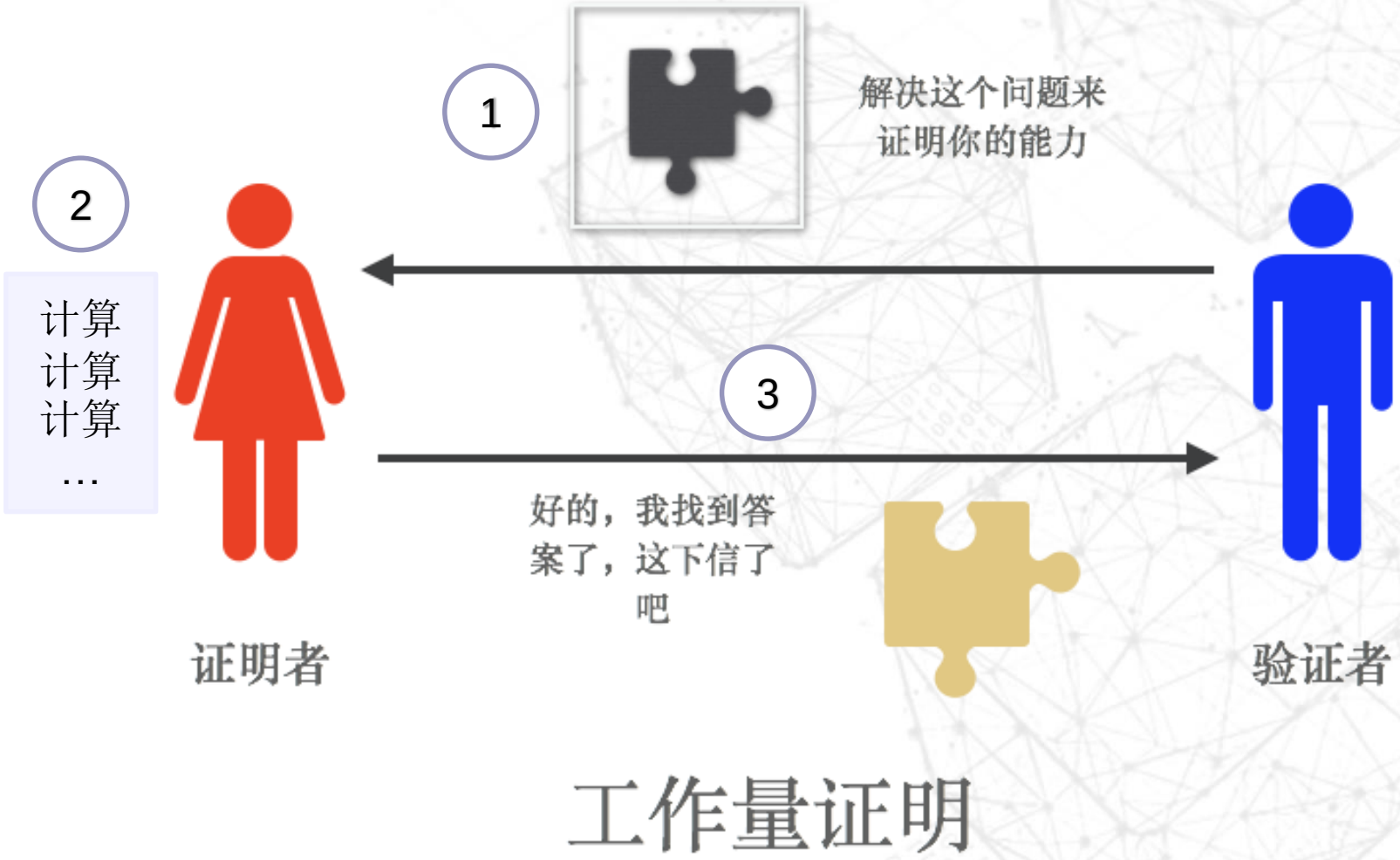


Proof of Work

- 工作量证明系统最早被用来解决解决拒绝服务攻击及其他形式的服务滥用攻击（ Cynthia Dwork and Moni Naor ,1993）
  - 用户在获得某种服务之前，解决一个小问题，这个问题对一般用户来说不是问题，但是如果需要计算非常多次，总计算量较大
- 工作量证明：由 Markus Jakobsson 和 Ari Juels1999年正式提出
- 最主要的特性就是非对称性，验证容易，但解决问题困难

验证容易

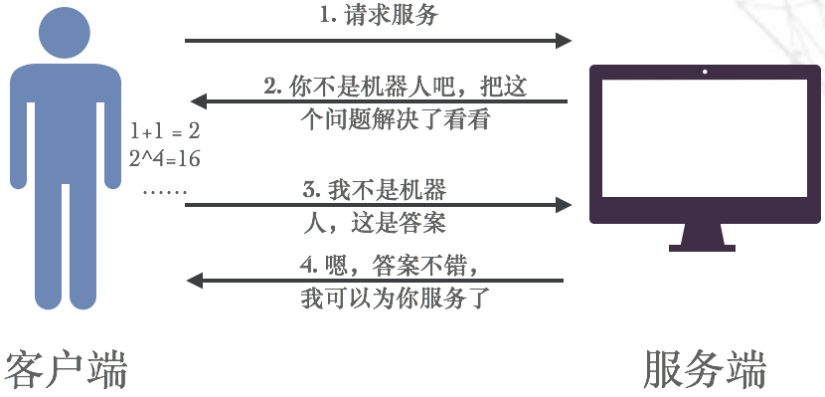
PoW(Proof of Work)



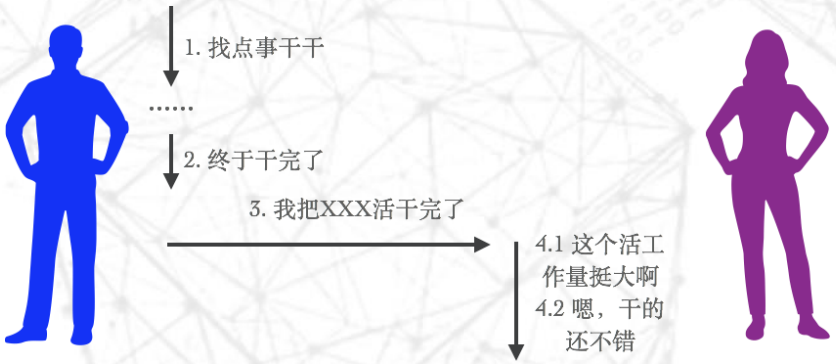
引自于清华大学徐恪教授的《区块链技术》

PoW协议类型

- 交互式
  - 采用一问一答的形式
  - 问题由服务端提出
  - 客户端根据问题求解答案并返回给服务端，以获取特定回报



- 验证式
  - 问题由用户自己产生
  - 验证端需要验证
    - 问题本身是否正确
    - 问题的答案是否正确
  - 如：Hashcash

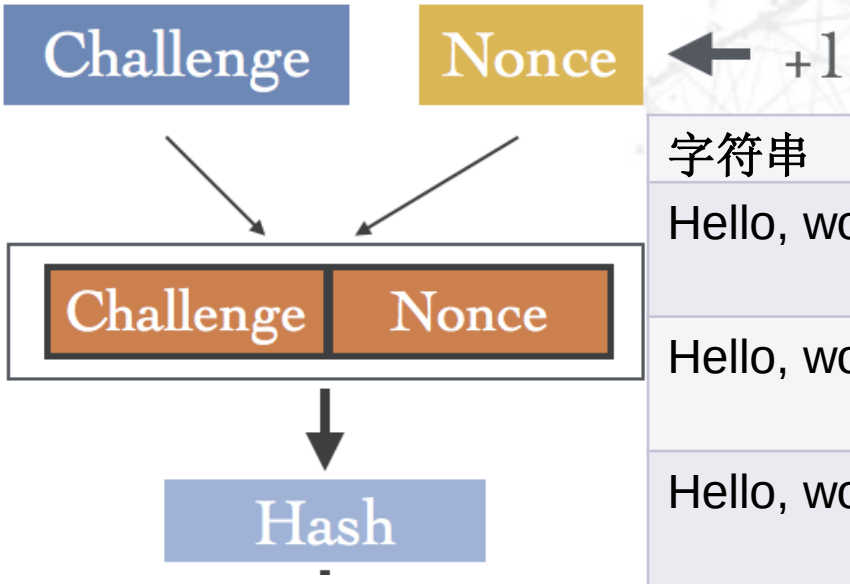


比特币的工作量证明

Size	Field	Description
4 bytes	Version	区块版本号，目前为2
32 bytes	Previous Block Hash	前置区块（父区块）的区块头Hash，Hash算法为double-SHA256
32 bytes	Merkle Root	区块中交易Merkle树根
4 bytes	Timestamp	区块创建UNIX时间戳
4 bytes	Difficulty Target	工作量证明算法难度
4 bytes	Nonce	通过变动该计数器来达成工作量证明要求的结果

比特币中的工作量证明

- 将字符串“ Hello, world!”通过末尾加上数字，得到一个哈希值的前4位为0000.



字符串	Sha256
Hello, world!0	1312af178c253f84028d480a6adc1e25e81caa44c749ec81976192e2ec934c64
Hello, world!1	e9afc424b79e4f6ab42d99c81156d3a17228d6e1eef4139be78e948a9332a7d8
Hello, world!2	ae37343a357a8297591625e7134cbea22f5928be8ca2a32aa475cf05fd4266b7
……	fcfc1253e8d2e4545e9f14f2ada4e7bfd5e2120183ad1e2a33ef8a6a40b3e2eb
Hello, world!4249	c004190b822f1669cac8dc37e761cb73652e7832fb814565702245cf26ebb9e6
Hello, world!4250	0000c3af42fc31103f1fdc0151fa747ff87349a4714df7cc52ea464e12dcd4e9

PoW(Proof of Work)

难度如何确定的？

挖矿难度

比特币难度是对挖矿困难程度的度量，即指：计算符合给定目标的一个哈希值的困难程度.

difficulty = difficulty_1_target / current_target

difficulty_1_target的长度为256比特， 前32位为0， 后面全部为1 ，一般显示为哈希值，

0x00000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

difficulty_1_target表示btc网络最初的目标哈希. current_target是当前块的目标哈希，先经过压缩然后存储在区块中，**区块的哈希值必须小于给定的目标哈希值**，表示挖矿成功.

PoW协议

- 向所有节点广播新的交易
- 每个节点把收到的交易放到块中
- 选中的节点广播它所保有的块
- 其它节点验证，接受
- 将该区块的哈希值放入下一个它们创建的区块中

PoW底层函数

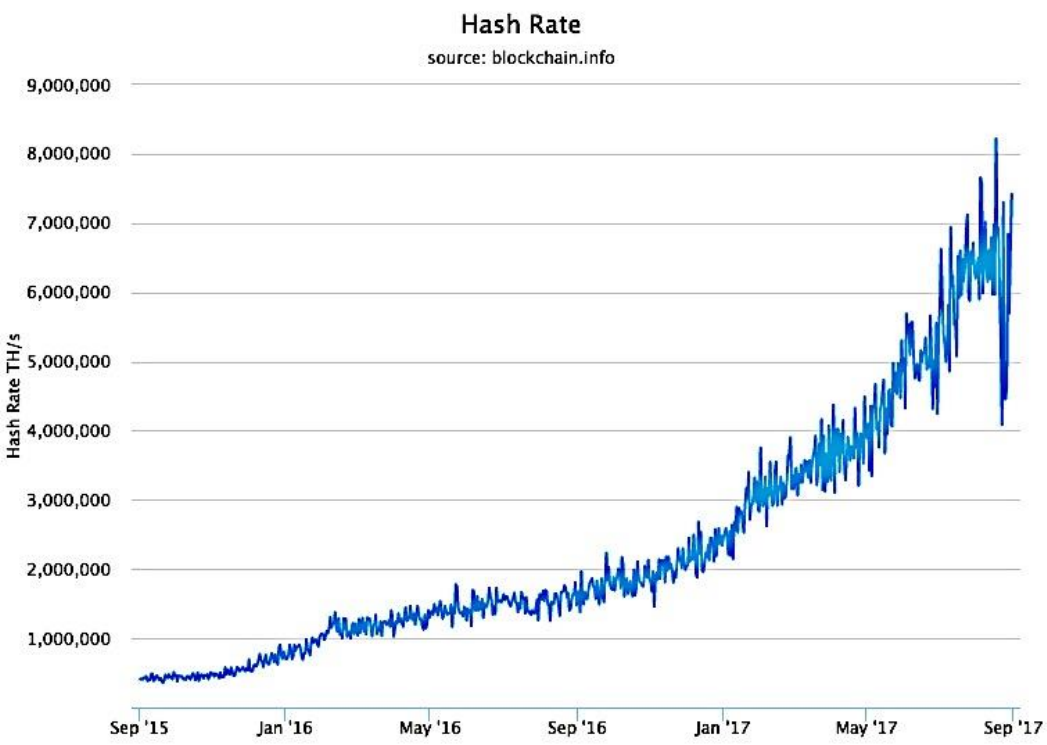
- PoW常用的函数有一下几种类型
  - 依赖**CPU**：比特币中使用的哈希函数SHA256
  - 依赖**内存**：计算速度的快慢主要依赖内存存取速度等
  - 依赖**网络**：客户端不需要进行大量的计算，但是必须按照服务器的要求访问一些结点，这个过程就引入了RTT延迟，可用来防止DDoS攻击

PoW底层函数

- CPU型
  - SHA256
  - Blake-256(哈希函数)
  - HEFTY1(哈希函数，用于阻止专用挖矿软件)
  - Quark（函数函数）
  - SHA-3（哈希函数）
  - ……
- 内存型
  - Scrypt (内存密集型哈希函数)
  - CryptoNight（依赖于内存的随机存取）
  - MOMENTUM （验证效率较Scrypt高）
- 网络型
  - Guided Tour Puzzle

比特币中PoW优缺点

- 优点
  - 可靠性高
- 缺点
  - 浪费能源
  - 51%攻击
  - 矿池导致中心化严重
    - 挖矿的硬件一旦不能挖矿，没有作用



两年内比特币哈希算力增长

平均不到半年，算力翻倍

ASIC-Resistance

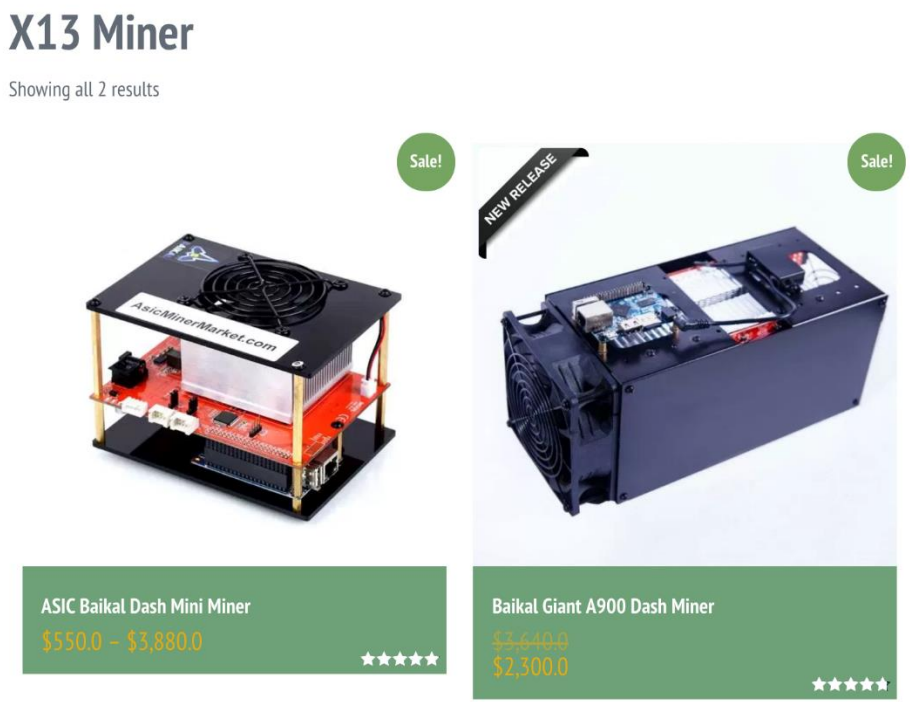
- Memory-hard
  - 需要大量的内存但是不需要大量的计算
- Memory-bound
  - 内存会成为计算的瓶颈
- Memory-hard型的方案可以有效抵制ASIC
- ASIC是指特定用途集成电路，它们被专门设计出来实现特定算法（如SHA256）

- 一种尝试：Scrypt
  - 首先生成许多为随机序列
  - 针对这些为随机队列进行随机存取、计算

- 缺点：
  - 需要同样大小的内存来验证
  - 目前已有针对Scrypt的*ASIC*;

其他抵制ASIC的方法

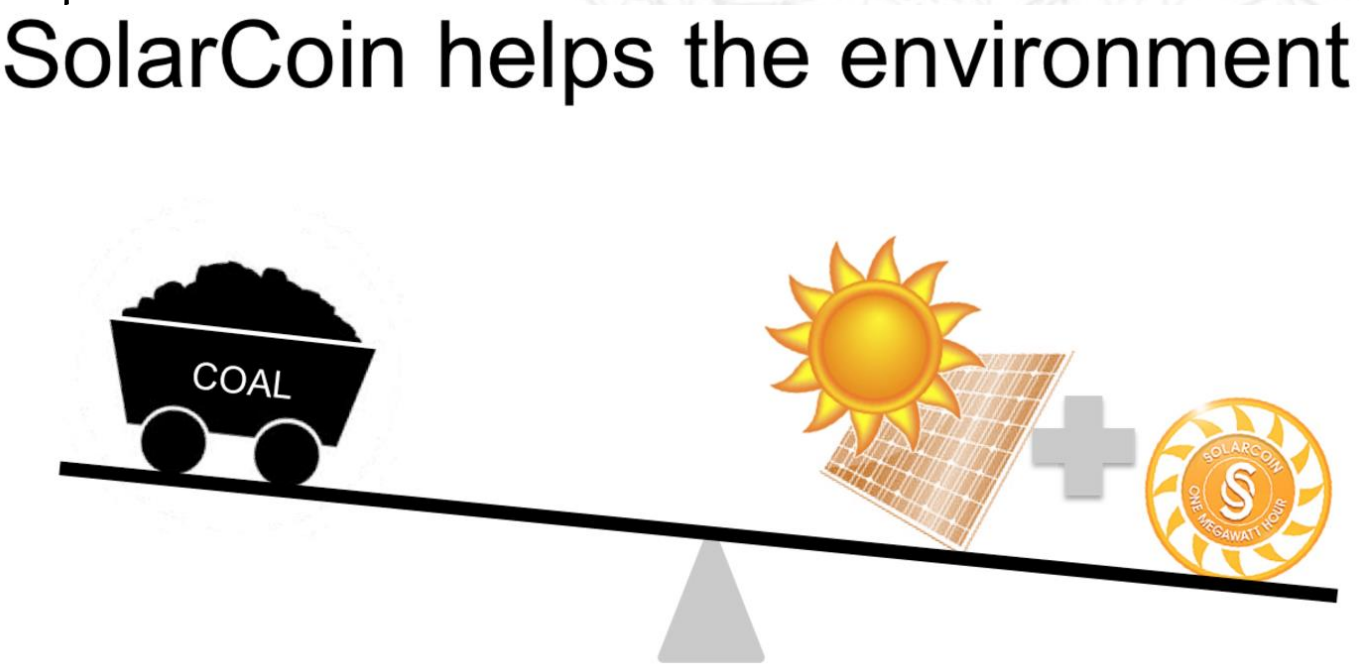
- X11或者x13，将11/13种哈希算法共同使用
  - DASH
  - 很难设计ASIC
  - Ps: 现在已经有专门针对此类PoW的矿机
- 频繁更换各种哈希算法
  - 如：SHA-1 -> SHA-3 -> Scrypt -> …
  - 目前还没有人做



针对dash的x13矿机

Proof of Usefull Work

- 不再进行单纯的计算无意义的计算
  - 搜索大素数
  - 蛋白质折叠问题
  - 生成气候预测模型
  - SolarCoin - 证明产生了太阳能就可以得到“太阳币”



Proof of Storage: Permacoin

- 寻找一个大文件
  - 重要的、公共的、需要备份
  - 任何个人无法单独存储全部文件
- 将大块分成许多小块，构建默克尔树
- 每一个矿工值存储其中一小部分
  - 根据同步字符串和一个随机nonce值，选取K个特定下标的本地块
  - 对nonce和被选择的块的头部进行哈希，若小于 target 就找到了一个答案
    - 如果客户端已经下载了区块，那么这个过程很简单，否则很困难
- 缺点
  - 很难找到大的文件、难度改变困难、文件修改困难

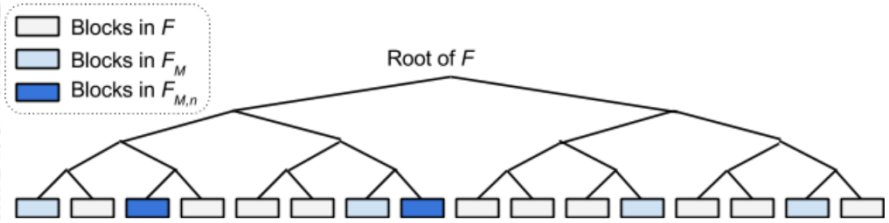
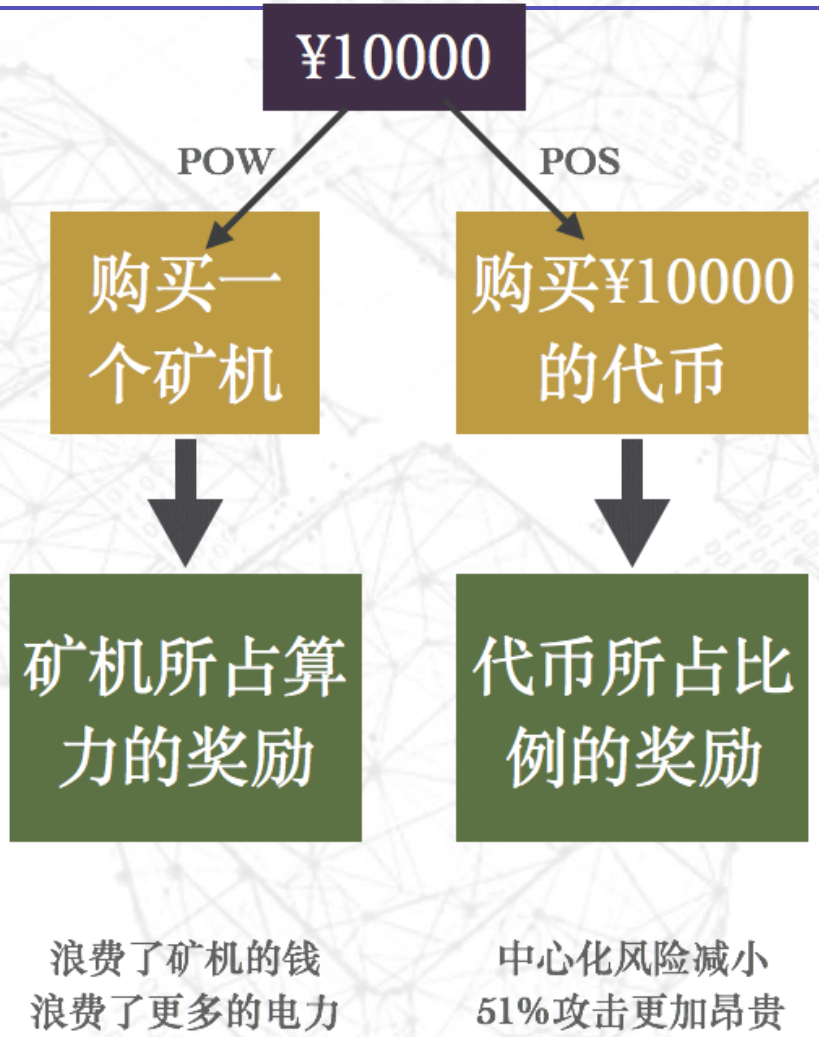


Figure 8.4: Choosing random blocks in a file in Permacoin.
In this example k_1=6 and k_2=2. In a real implementation these parameters would be much larger.

Proof of Stake （PoS）

- 新的共识算法
- 2011年首先在Peercoin中实现，现在已经有多个版本(NXT, Tendermint, Flying Fox, …)
- 占有多少代币，就能得到相应比例的奖励



Proof of Stake （PoS）

- 类似于财产存储在银行，根据持有的数字货币的量和时间，分配相应比例的收益
- 币龄：持币量×时间 （单位：币天）

  如持有1000个币30天，那么币龄就是30000

- 币龄越大，被选为区块生产者的概率越大
- 每次产生完区块，币龄就会清零

Proof of Stake （PoS）

- 优点：

  --减少能源消耗

  --减少对硬件设备的依赖

  --缩短了区块产生时间和确认时间

- 缺点：

  --实现规则复杂，中间步骤多

  --币龄依赖，攻击者在积累较大币龄后，易发动双花攻击