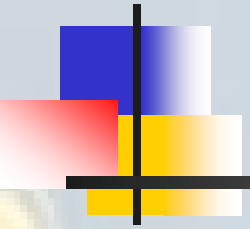
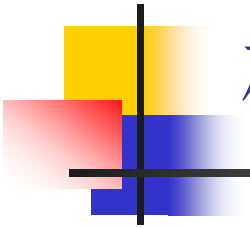
区块链基础及应用



Chapter 3 比特币的运行机制

苏 明



概览

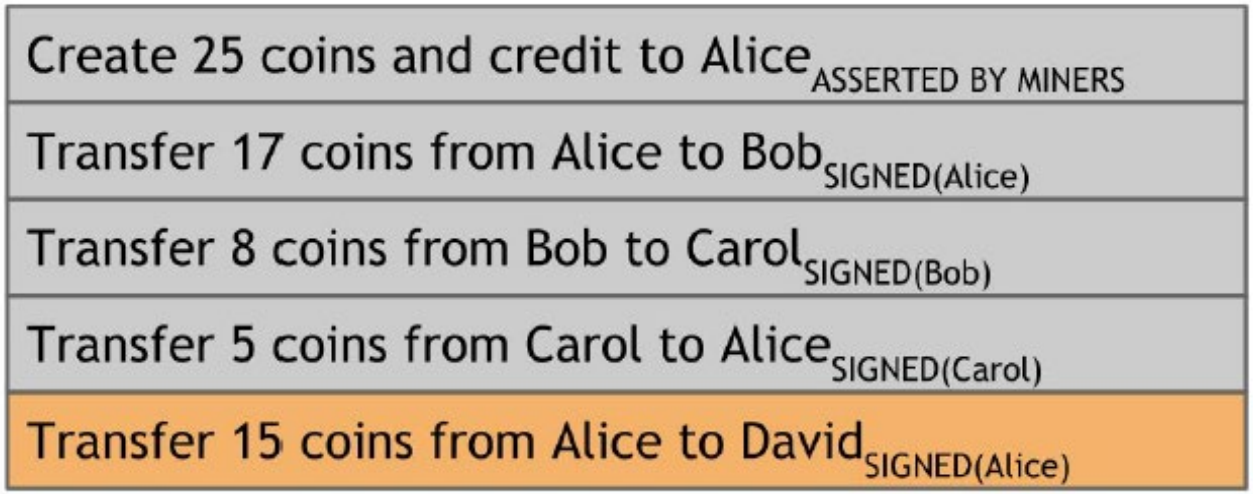
- 3.1 比特币的交易
- 3.2 比特币的脚本
- 3.3 比特币脚本的应用
- 3.4 比特币的区块
- 3.5 比特币网络
- 3.6 限制与优化

3.1 比特币的交易

- Blockchain----Public Ledger 公开账簿

- First Impression: 记账方式
- 基于帐户的系统：与银行卡的帐户类似

3.1 比特币的交易



an account-based ledger

3.1 比特币的交易

需要维护账户余额

Account	Balance
Alice	13
Bob	9
Carl	3

时间

系统创造25个币给Alice，由矿工确认

Alice转17个币给Bob，由Alice签名

Bob转8个币给Carol，由Bob签名

Carol转5个币给Alice，由Carol签名

Alice转15个币给David，由Alice签名

这笔交易是否合法？

可能需要回溯到最初始的地方

3.1 比特币的交易

问题？

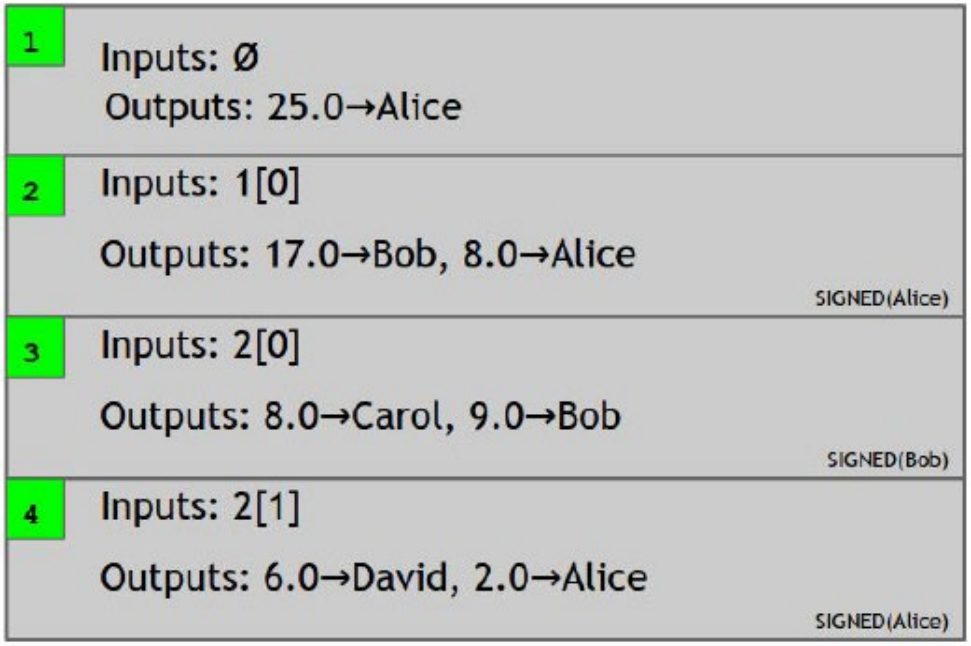
- 如果要去确认一笔交易是否真实，就必须追溯每一个帐户的余额

增加一个数据字段，更新每次交易后的账户余额

- 但实际不可行：*分布式网络*，**延时确认**

3.1 比特币的交易

借鉴了**Scroogecoin**里面的记账方式



a transaction-based ledger close to Bitcoin

3.1 比特币的交易

公钥、身份、PKI, CA, 证书

比特币“创新性”地“不用证书绑定身份”，从而“不需要CA”：公钥即身份！

- 使去中心化成为可能：绑定身份的话，就需要CA，就需要有一个中心机构
- 匿名性

比特币的交易

- 交易的输出使用“接收者的公钥的Hash值”作为输出的币的接收“地址”
- 交易时，接收者不需要把自己的公钥告诉发送方，而只需要告诉发送方一个Hash值
- 当接收者需要把自己的“地址”上的币花出去时，提供一个公钥和一个数字签名出来，“公钥的Hash值等于地址”且“数字签名能被公钥验证通过”，则可以合法的把币花出去

1	**Inputs: ∅** **Outputs: 25.0 → hval1** H(PK1)=hval1	$Verify(PK1, Tx2data, \sigma) = 1$
2	Inputs: 1[0] Outputs: 17.0 → hval2, 8.0 → hval3	PK1,Sign(SK1, Tx2-Data)
3	Inputs: 2[0] Outputs: 8.0 → hval4, 9.0 → hval5	PK2,Sign(SK2, Tx3-Data)
…	……	……
11	Inputs: 3[0] Outputs: 6.0 → hval6, 2.0 → hval7	PK3,Sign(SK3, Tx11-Data)
12	Inputs: 3[1] Outputs: 6.0 → hval8, 3.0 → hval9	PK7,Sign(SK7, Tx12-Data)
13	**Inputs: 11[0], 12[0]** **Outputs: 12.0 → hval10**	**PK4,Sign(SK4, Tx13-Data)** **PK6,Sign(SK6, Tx13-Data)**

3.1 比特币的交易

- 地址转换 (Change Addresses)--处理余额
- 有效验证 (Efficient Verification)--无需从头核查
- 资金合并--支持多输入输出
- 共同支付--多人数字签名

3.1 比特币的交易

- 交易语法

3.2 比特币的脚本

- 最常见的比特币交易：通过某人的签名获得他在前一笔交易中分配的资金

- 交易输出描述为：凭借哈希值为X的公钥，以及这个公钥所有者的签名，才能获得这笔资金

3.2 比特币的脚本

- 把交易的**输入脚本**(*scriptSig*)，和上一笔交易的**输出脚本**(*scriptPubKey*)串联起来

- 串联脚本必须被成功执行以后，才能获得资金

3.2 比特币的脚本

```
OP_DUP
OP_HASH160
69e02e18...
OP_EQUALVERIFY
OP_CHECKSIG
```

an example Pay-to-PubkeyHash script, the most common type of output script in Bitcoin

3.2 比特币的脚本

比特币脚本语言

- Forth: 简单的堆栈式编程语言(stack-based)
- 每个指令只被执行一次，线性的，无法循环执行
- 非图灵完备

3.2 比特币的脚本

```
<sig>
<pubKey>
--------------
OP_DUP
OP_HASH160
<pubKeyHash?>
OP_EQUALVERIFY
OP_CHECKSIG
```

To check if a transaction correctly redeems an output, we create a combined script by appending the scriptPubKey of the referenced output transaction (bottom) to the scriptSig of the redeeming transaction (top). Notice that <pubKeyHash?> contains a '?'. We use this notation to indicate that we will later check to confirm that this is equal to the hash of the public key provided in the redeeming script.

3.2 比特币的脚本

OP_DUP	Duplicates the top item on the stack
OP_HASH160	Hashes twice: first using SHA-256 and then RIPEMD-160
OP_EQUALVERIFY	Returns true if the inputs are equal. Returns false and marks the transaction as invalid if they are unequal
OP_CHECKSIG	Checks that the input signature is a valid signature using the input public key for the hash of the current transaction
OP_CHECKMULTISIG	Checks that the *k* signatures on the transaction are valid signatures from *k* of the specified public keys.

a list of common Script instructions and their functionality

3.2 比特币的脚本

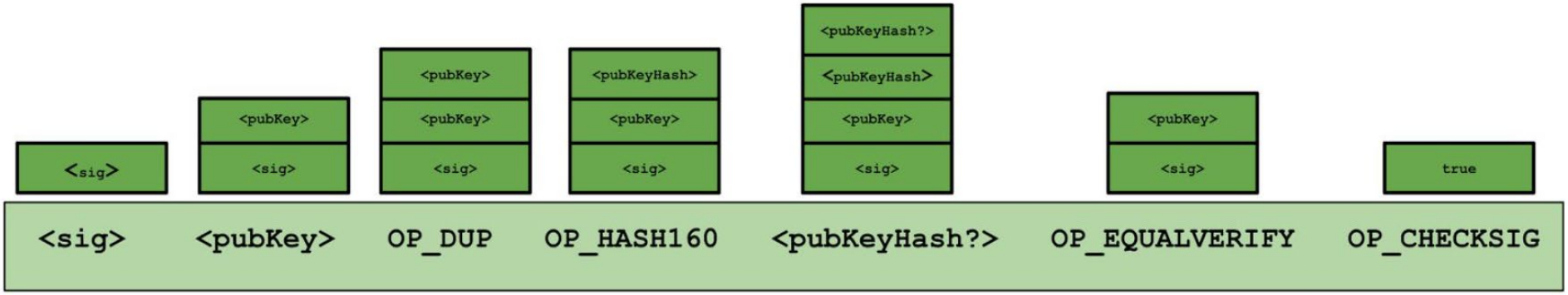
- 数据指令

  <pubKey>

- 工作码指令

  <OP_***>

3.2 比特币的脚本



比特币脚本的执行堆栈状态图

3.2 比特币的脚本

Stack	Script	Description
Empty.	<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	scriptSig and scriptPubKey are combined.
<sig> <pubKey>	OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	Constants are added to the stack.
<sig> <pubKey> <pubKey>	OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	Top stack item is duplicated.
<sig> <pubKey> <pubHashA>	<pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	Top stack item is hashed.
<sig> <pubKey> <pubHashA> <pubKeyHash>	OP_EQUALVERIFY OP_CHECKSIG	Constant added.
<sig> <pubKey>	OP_CHECKSIG	Equality is checked between the top two stack items.
true	Empty.	Signature is checked for top two stack items.

OP_RETURN

- A script opcode used to mark a transaction output as invalid
- burn bitcoins
- convey additional information needed:

  ?NOT a record for arbitrary data?

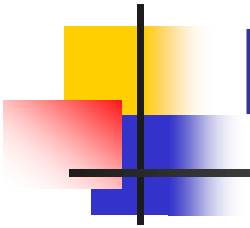
3.2 比特币的脚本

实际情况

- 可以随意地为比特币支付设定条件:

MULTISIG (Due to a bug, one extra unused value is removed from the stack)

Pay-to-Script-Hash P2SH



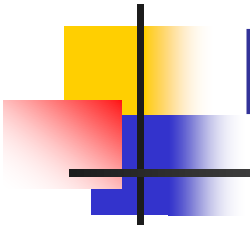
P2SH

支付给脚本的哈希值

- 把比特币**支付**给某个脚本地址，脚本的哈希值是×××
- 取款的时候，*提供上述哈希值对应的脚本*，**同时***提供数据能通过脚本的验证*

P2SH

- 支付工作简单化：收款方只需告诉付款方一个哈希值即可
- P2SH的输出脚本(ScriptPubKey)会变得很小，复杂的事情放在输入脚本(ScriptSig)里面了



P2SH

Before P2SH

Locking Script: 2 <Public Key 1> <Public Key 2> <Public Key 3> <Public Key 4> <Public Key 5> 5 CHECKMULTISIG

Unlocking Script: <Sig 1> <Sig 2>

After P2SH

Redeem Script: 2 <Public Key 1> <Public Key 2> <Public Key 3> <Public Key 4> <Public Key 5> 5 CHECKMULTISIG

Locking Script: HASH160 <Hash of Redeem Script> EQUAL

Unlocking Script: <Sig 1> <Sig 2> <Redeem Script>

3.3 比特币脚本的应用

第三方支付交易

MultiSig

三个人中有两个人签名以后，资金才能被支取

Alice, Bob; Judy（第三方仲裁）

发生纠纷的时候

3.3 比特币脚本的应用

绿色地址

- 一个交易确认：延迟
- 第三方银行：

Alice→Bob（传统）

Alice→Bank：

（Alice: 支付Bob这些币，能代办吗？）

(Bank: 从你的帐户扣钱，然后从(银行)绿色地址转帐给Bob)

不是Bitcoin技术系统的保障，而是现实世界中银行的**声誉**保证：不会发生双重支付

3.3 比特币脚本的应用

高效小额支付(efficient micro-payments)

- 设想手机流量使用场景：按照分钟计费，但每分钟支付一次不现实
- Solutions?
- MULTISIG

3.3 比特币脚本的应用

锁定时间（lock_time）

- 等待t时间之后才能把**退款**交易计入区块链
- 如果过了t时间Bob还没有在最后一个交易上签名确认；Alice可以通过退款交易收回所有的Bitcoin

3.3 比特币脚本的应用

智能合约 (Smart Contract)

可以用技术手段来强制执行的合约；

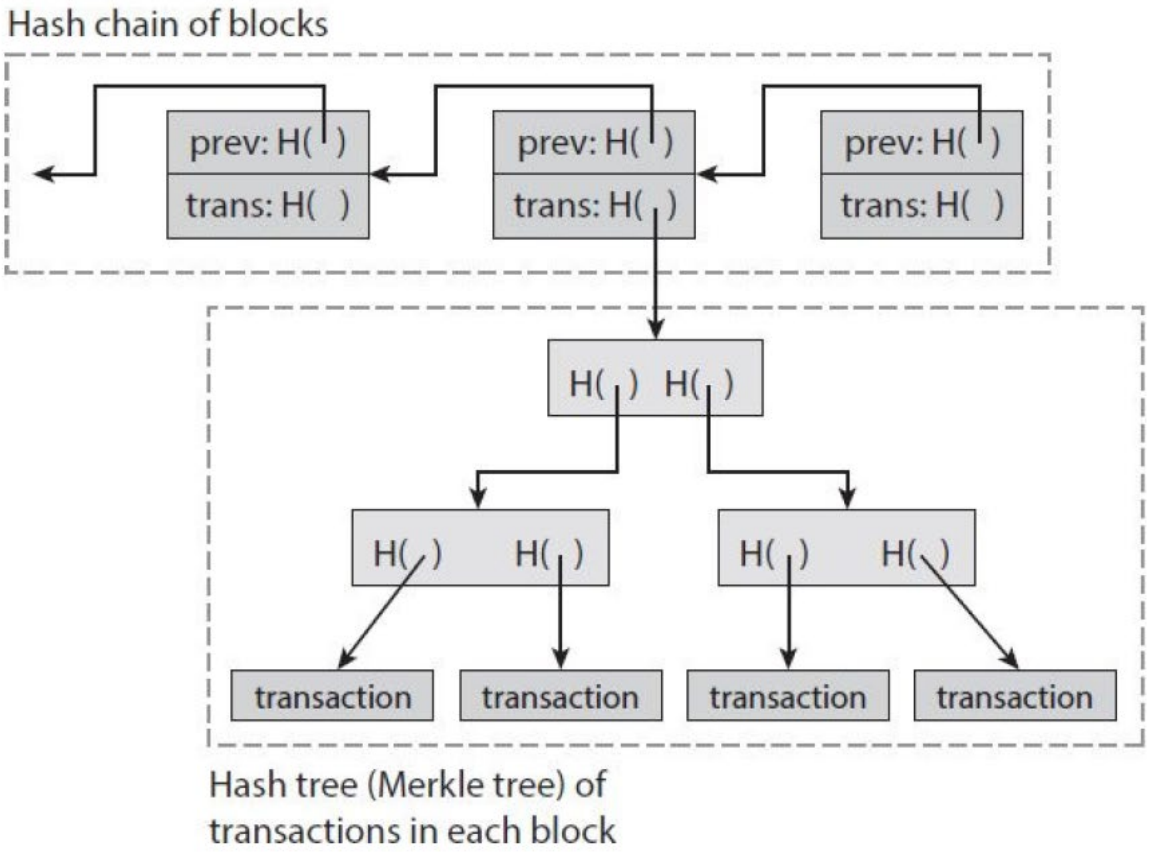
可以用脚本、矿工、和交易验证来实现 第三方托管协议或者是小额支付；而不是中心化权威机构

3.4 比特币的区块

- 把大量交易组织起来放入一个区块，得到的哈希链会更短；可以提高验证区块链数据结构的效率

- 区块链把两个基于哈希值的数据结构结合起来：1. **区块的哈希链** 2. 树状结构把所有交易组织起来（**Merkle Tree**）

3.4 比特币的区块



The Bitcoin block chain contains **two different hash structures**

3.4 比特币的区块

- 币基交易 (Coinbase Transaction)

1. It always has a single input and a single output.
2. The input doesn't redeem a previous output and thus contains a null hash pointer
3. The value of the output: Mining Income
4. a special "coinbase" parameter, which is completely arbitrary

```
"in":[
        {
          "prev_out":{
            "hash":"000000.....0000000",
            "n":4294967295
          },
          "coinbase":"..."
    },
    [
"out":[
        {
          "value":"25.03371419",
          "scriptPubKey":"OPDUP OPHASH160 ... "
        }
       ]
```

3.4 比特币的区块

https://www.blockchain.com/explorer

创世块是比特币区块链的原始块。也称为块0，它是所有其他块构建的基础。没有创世块，就不能创建新块，也就没有区块链。

创世块来自比特币的创始人中本聪（Satoshi Nakamoto）。他是用微软的visual studio和c++编写的，

他从2009年1月3日开始开采，**持续了六天**；中本聪正在彻底测试创世块，以确保它是完美的，因为它将永远被硬连接到系统中。

3.4 比特币的区块

```
GetHash()      = 0x000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f
hashMerkleRoot = 0x4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b
txNew.vin[0].scriptSig     = 486604799 4 0x736B6E616220726F662074756F6C69616220646E6F63657320666F206B6E697262206E6F20726F6C6C65636E616843203930303
txNew.vout[0].nValue       = 5000000000
txNew.vout[0].scriptPubKey = 0x5F1DF16B2B704C8A578D0BBAF74D385CDE12C11EE50455F3C438EF4C3FBCF649B6DE611FEAE06279A60939E028A8D65C10B73071A6F16719274
block.nVersion = 1
block.nTime    = 1231006505
block.nBits    = 0x1d00ffff
block.nNonce   = 2083236893

CBlock(hash=000000000019d6, ver=1, hashPrevBlock=00000000000000, hashMerkleRoot=4a5e1e, nTime=1231006505, nBits=1d00ffff, nNonce=2083236893, vtx=1
  CTransaction(hash=4a5e1e, ver=1, vin.size=1, vout.size=1, nLockTime=0)
    CTxIn(COutPoint(000000, -1), coinbase 04ffff001d0104455468652054696d65732030332f4a616e2f32303039204368616e63656c6c6f72206f6e206272696e6b206f66
    CTxOut(nValue=50.00000000, scriptPubKey=0x5F1DF16B2B704C8A578D0B)
  vMerkleTree: 4a5e1e
```

The coinbase parameter (seen above in hex) contains, along with the normal data, the following text:

The Times 03/Jan/2009 Chancellor on brink of second bailout for banks

3.4 比特币的区块

Hash	000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f
Confirmations	655,111
Timestamp	2009-01-04 02:15
Height	0
Miner	Unknown
Number of Transactions	1
Difficulty	1.00
Merkle root	4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b
Version	0x1
Bits	486,604,799
Weight	1,140 WU
Size	285 bytes
Nonce	2,083,236,893
Transaction Volume	0.00000000 BTC
Block Reward	50.00000000 BTC

3.4 比特币的区块-Latest

Block Hash

000000000000000000362fa5436b32fb84db5333e5bcd67a6b1c2c3e31b08e2

Summary

Height	◀ 812,460 ▶	Relayed By	ViaBTC
Confirmations	4	Difficulty	83.11 T / 61.03 T
Block Size	1,408,196 Bytes	Block Reward	6.25000000 BTC
Stripped Size	861,748 Bytes	Fee Reward	0.20569288 BTC
Weight	3,993,440	Tx Count	2,137
Time	2023-10-16 21:28:28	Tx Volume	6,554.71554628 BTC

Merkle Root	f167bd39796a6201a59b32e7ecc66d2948d4a21306584a39d6e6e9ef711844c7
Version	0x3e000000
Nonce	0x8a86f1d1

3.5 比特币网络

- 所有的节点都是平等的
- 发起一个交易： Flooding算法 (gossip协议)

3.5 比特币网络

核验新交易信息

1. 对每个前序交易的输出运行核验脚本
2. 校验是否有双重支付
3. 检查这笔交易信息是不是已经被本节点接收过
4. 节点只会接收和传递在白名单上的标准脚本

3.5 比特币网络

- 哪一个交易被纳入区块链产生分歧(Race condition)
- 每个节点**默认**保留最早接收到的交易（or Replace-by-fee）
- 如果两个矛盾的交易或者区块在不同地方发起，向整个网络广播；那么节点接收那个交易取决于它在网络的位置

3.5 比特币网络

区块的传播与交易传播类似

- 竞态条件的限制：哪个区块被纳入长期共识链取决于其他节点选择在哪个区块上扩展区块链

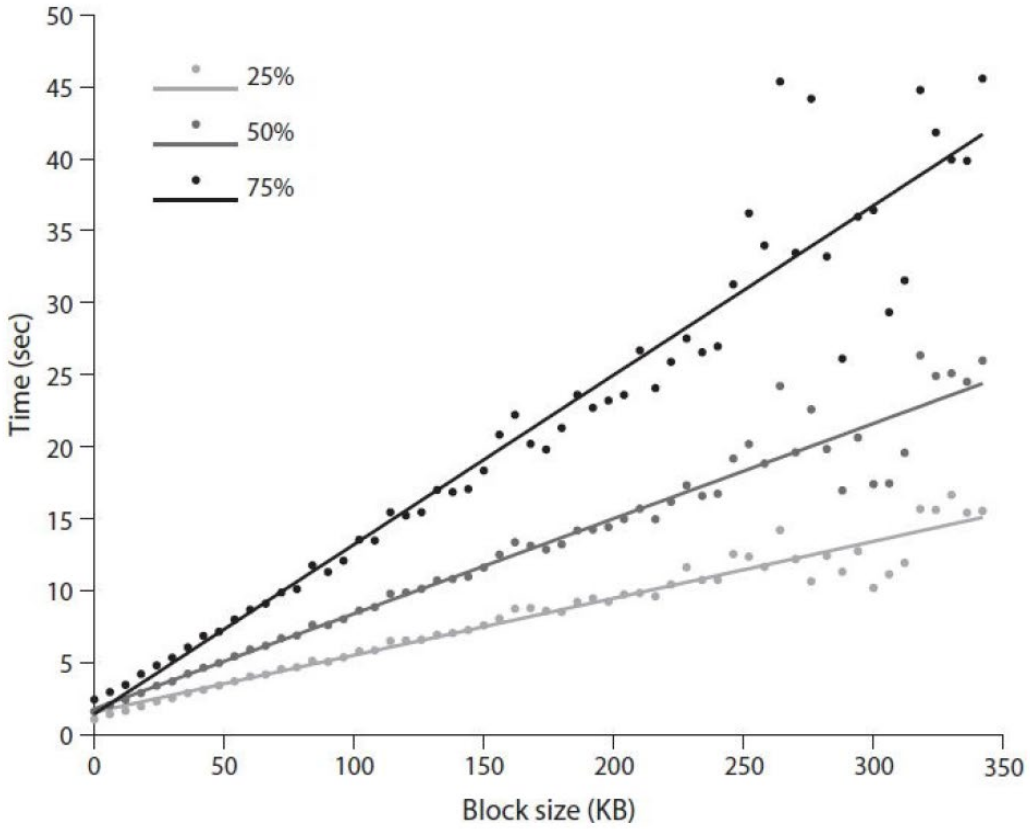
- 核验一个区块：

确认区块头部；　确定哈希值在**给定范围**；

确认区块里面的**每个交易**；　**最长**的一条区块链进行扩展

3.5 比特币网络

- Flooding algorithm(泛洪算法)

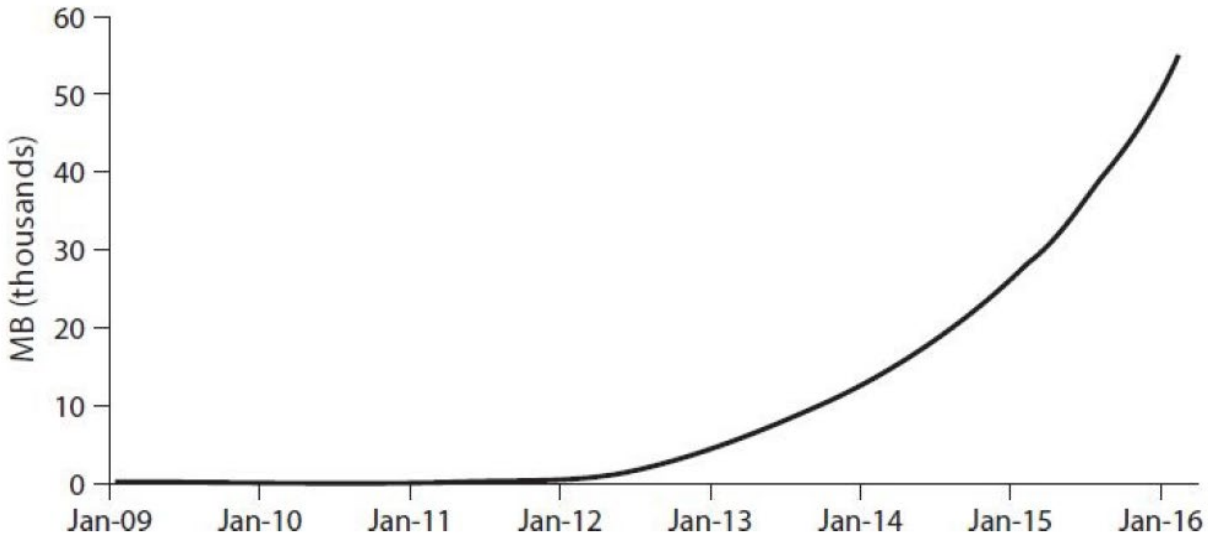


Block propagation time

3.5 比特币网络

- 存储空间需求

- **全节点：**需要把完整的共识区块链存储下来



Size of the block chain

3.5 比特币网络

- **轻量节点**：简单付款验证(Simple Payment Verification) SPV客户端
- 只存储他们所关心的，需要进行核验的部分交易
- SPV节点只验证相关交易；依赖全节点去验证网络上的其他所有交易
- 只需要几十MB数据(VS 几十GB)

3.6 限制与优化

- 比特币的总体数量与记账奖励 社区内基本达成共识：不应该改变
- 交易性能： 7笔/秒 (Calculation)
- Limitation: Visa 2000 笔/秒

3.6 限制与优化

Weakness of Crypto Suite

- ECDSA
- SHA-1

担心在一生中，这些算法可能会被攻破

Reference: ABCMint

抗量子攻击

3.6 限制与优化

修订协议：无法假定所有节点都会更新版本

- 硬分叉

  引入新的特性，使得前一版本的协议失效

- 软分叉

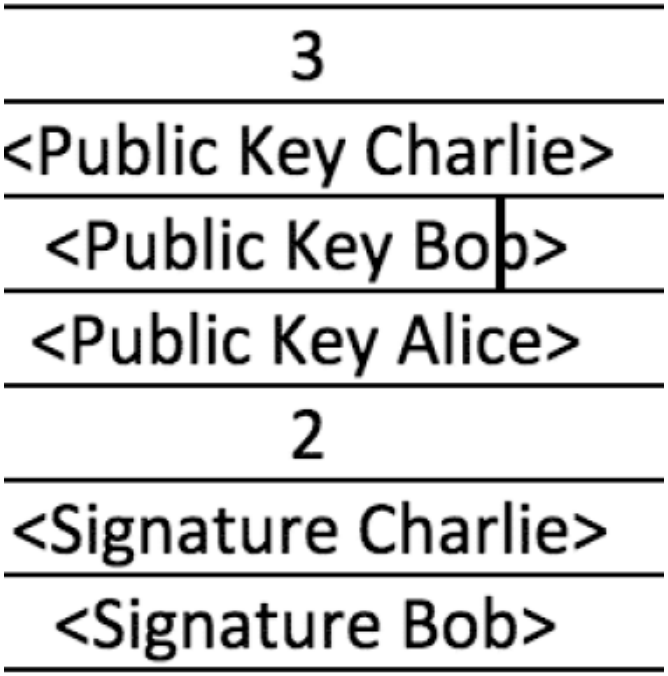
加入新特性，让现有的核验规则更加严格。老的节点依然会接受所有的区块，而新的节点会拒绝

向下兼容--P2SH

3.6 限制与优化

- MULTISIG的Bug: 推送给堆栈一个无用的值

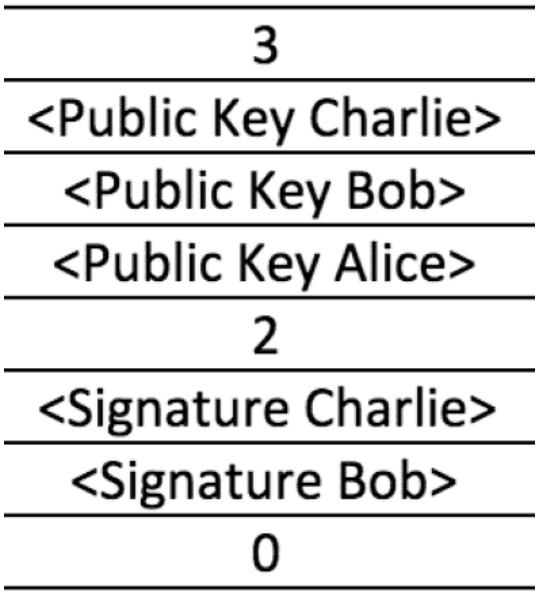
Ideally, a CHECKMULTISIG opcode should pop out M+N+2 elements



3.6 限制与优化

However, there is a glitch in the CHECKMULTISIG opcode which makes it pop one more item than are available in the stack.

0 <Signature Bob> <Signature Charlie> 2 <Public Key Alice> <Public Key Bob> <Public Key Charlie> 3 CHECKMULTISIG



但修复这个缺陷，会产生硬分叉

Research Problem

- 如何改变区块大小：难以达成共识

- **提高**（比特币）**交易处理能力**

  BTC VS BCH

Reference

- blockchain.info (区块链浏览器)

- Antonopoulos, Andreas M. Mastering Bitcoin: unlocking digital cryptocurrencies. O'Reilly Media, 2014.
  （比特币开发手册：技术细节）