比特币区块链技术简介

赵增哲（z3ustb@gmail.com）

目录

- 几个术语
- 几个算法
- 区块链
- 交易
- 挖矿

术语

术语

- 区块：一个区块就是若干交易数据的集合，它会被标记上时间戳和之前一个区块的独特标记。区块头经过哈希运算后会生成一份工作量证明，从而验证区块中的交易。有效的区块经过全网络的共识后会被追加到主区块链中。

- 交易：把比特币从一个地址转到另一个地址。更准确地说，一笔“交易”指一个经过签名运算的，表达价值转移的数据结构。每一笔“交易”都经过比特币网络传输，由矿工节点收集并封包至区块中，永久保存在区块链某处。

- 区块链：区块链是一串通过验证的区块，当中的每一个区块都与上一个相连，一直连到创世区块。

- 创世区块：区块链上的第一个区块，用来初始化相应的加密货币。

术语

- 工作量证明(POW)：工作量证明指通过有效计算得到的一小块数据。具体到比特币，矿工必须要在满足全网目标难度的情况下求解SHA256算法。

- 确认：当一项交易被区块收录时，我们可以说它有一次确认。矿工们在此区块之后每再产生一个区块，此项交易的确认数就再加一。当确认数达到六及以上时，通常认为这笔交易比较安全并难以逆转。

- 比特币钱包：保存比特币地址和私钥的软件，可以用它来接受、发送、储存你的比特币。

算法

拜占庭问题

- 拜占庭是古代东罗马帝国的首都，由于地域宽广，守卫边境的多个将军（系统中的多个节点）需要通过信使来传递消息，达成某些一致的决定。但由于将军中可能存在叛徒（系统中节点出错），这些叛徒将努力向不同的将军发送不同的消息，试图会干扰一致性的达成。

- Practical Byzantine Fault Tolerant 算法

  - 三个阶段来达成共识：Pre-Prepare、Prepare 和 Commit

- PoW（Proof of Work）：两个思路，一是限制一段时间内整个网络中出现提案的个数，另一个是放宽对最终一致性确认的需求，约定好大家都确认并沿着已知最长的链进行拓宽。

Hash算法

- 哈希算法将任意长度的二进制值映射为较短的固定长度的二进制值，这个小的二进制值称为哈希值
- 优秀的 hash 算法，将能实现：
  - 正向快速：给定明文和 hash 算法，在有限时间和有限资源内能计算出 hash 值。
  - 逆向困难：给定（若干） hash 值，在有限时间内很难（基本不可能）逆推出明文。
  - 输入敏感：原始输入信息修改一点信息，产生的 hash 值看起来应该都有很大不同。
  - 冲突避免：很难找到两段内容不同的明文，使得它们的 hash 值一致（发生冲突）。

Hash算法

- 流行的算法
  - MD4, MD5: 并不足够安全
  - SHA-1: 基于和 MD4 相同原理
  - SHA-2: SHA-224、SHA-256、SHA-384，和 SHA-512
  - SHA-3
- 性能
  - 一般的，hash 算法都是算力敏感型，意味着计算资源是瓶颈，主频越高的 CPU 进行 hash 的速度也越快。
  - 也有一些 hash 算法不是算力敏感的，例如 scrypt，需要大量的内存资源，节点不能通过简单的增加更多 CPU 来获得 hash 性能的提升。

加密算法

- 典型组件包括：加解密算法、公钥、私钥。
- 加密过程中，通过加密算法和公钥，对明文进行加密，获得密文。
- 解密过程中，通过解密算法和私钥，对密文进行解密，获得明文。

加密算法

- 对称加密：公钥和私钥相同
- 优点是加解密速度快，空间占用小，保密强度高。
- 缺点是参与多方都需要持有密钥，一旦有人泄露则安全性被破坏；另外如何其它分发密钥也是个问题。
- 代表算法包括 DES、3DES、AES、IDEA 等。
- 适用于大量数据的加解密，不能用于签名场景。

加密算法

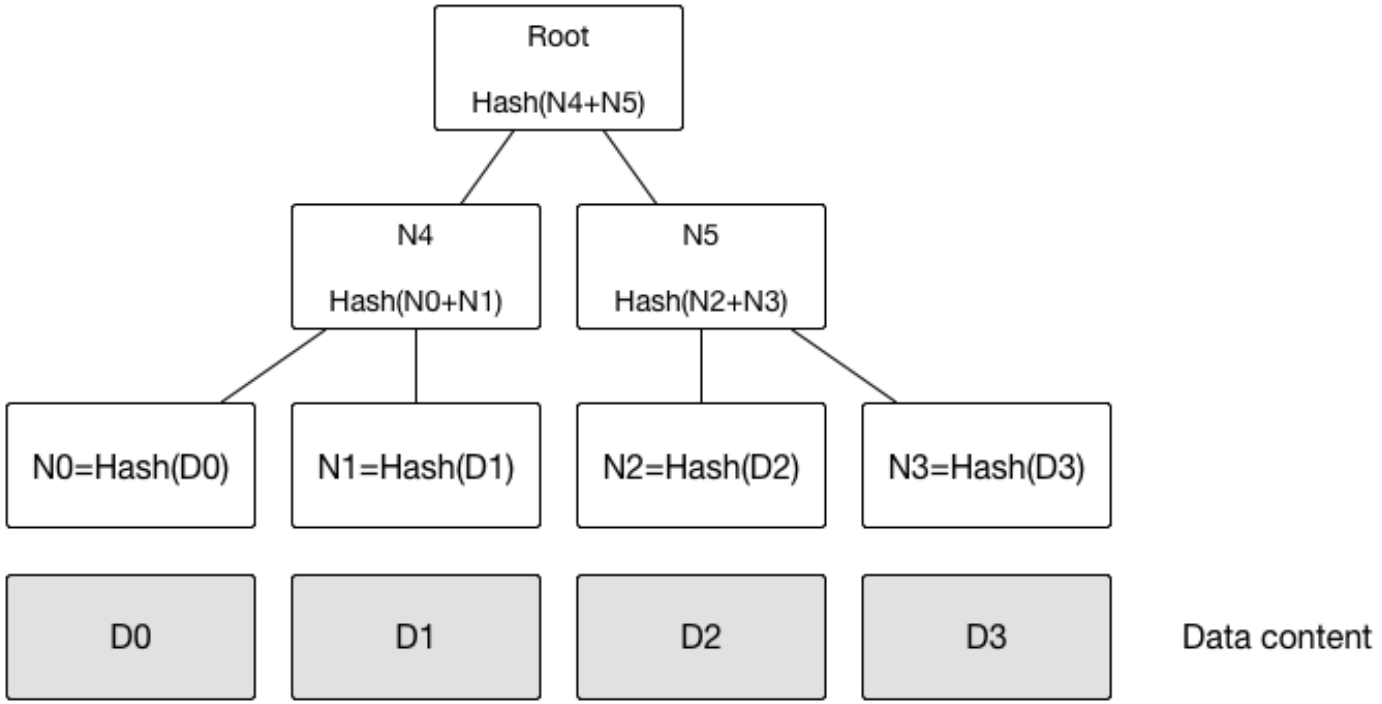
- 公钥和私钥是不同的。
- 优点是公私钥分开，容易管理，并且容易完成密钥分发。

- 缺点是加解密速度慢。
- 代表算法包括：RSA、ElGamal、椭圆曲线系列算法。
- 一般适用于签名场景或密钥协商，不适于大量数据的加解密。

加密算法

- 组合机制：先用计算复杂度高的非对称加密协商一个临时的对称加密密钥（会话密钥），然后双方再通过对称加密对传递的大量数据进行加解密处理。

Merkle树



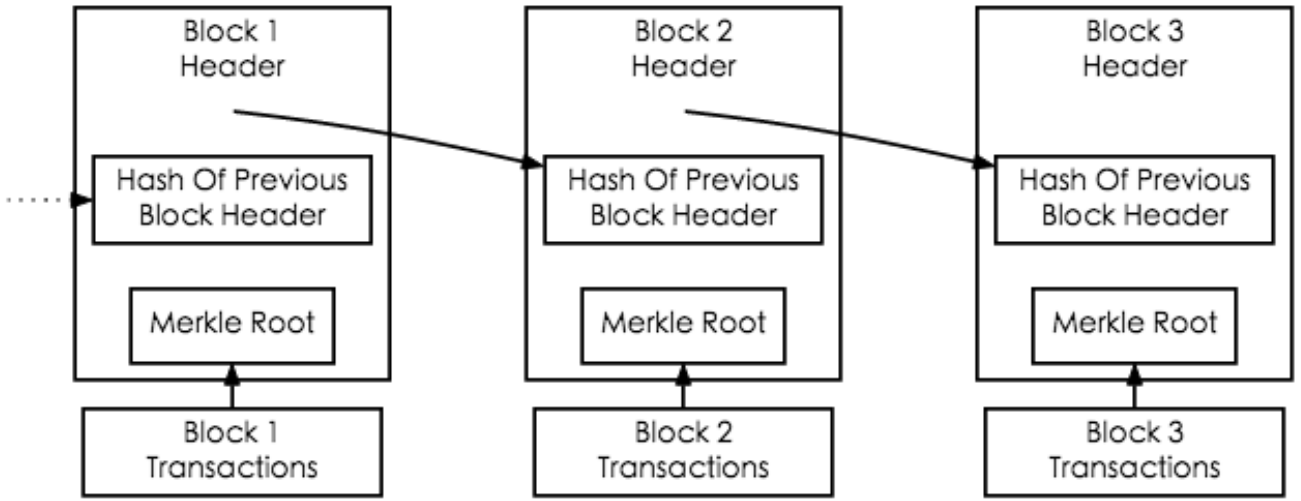
- 默克尔树（又叫哈希树）是一种二叉树，由一个根节点、一组中间节点和一组叶节点组成。最下面的叶节点包含存储数据或其哈希值，每个中间节点是它的两个孩子节点内容的哈希值，根节点也是由它的两个子节点内容的哈希值组成。

Merkle树

- 典型应用场景包括：
  - 快速比较大量数据：当两个默克尔树根相同时，则意味着所代表的数据必然相同。
  - 快速定位修改：例如上例中，如果 D1 中数据被修改，会影响到 N1，N4 和 Root。因此，沿着 Root --> N4 --> N1，可以快速定位到发生改变的 D1；
  - 零知识证明：例如如何证明某个数据（D0……D3）中包括给定内容 D0，很简单，构造一个默克尔树，公布 N0，N1，N4，Root，D0 拥有者可以很容易检测 D0 存在，但不知道其它内容。

区块链

区块链概览



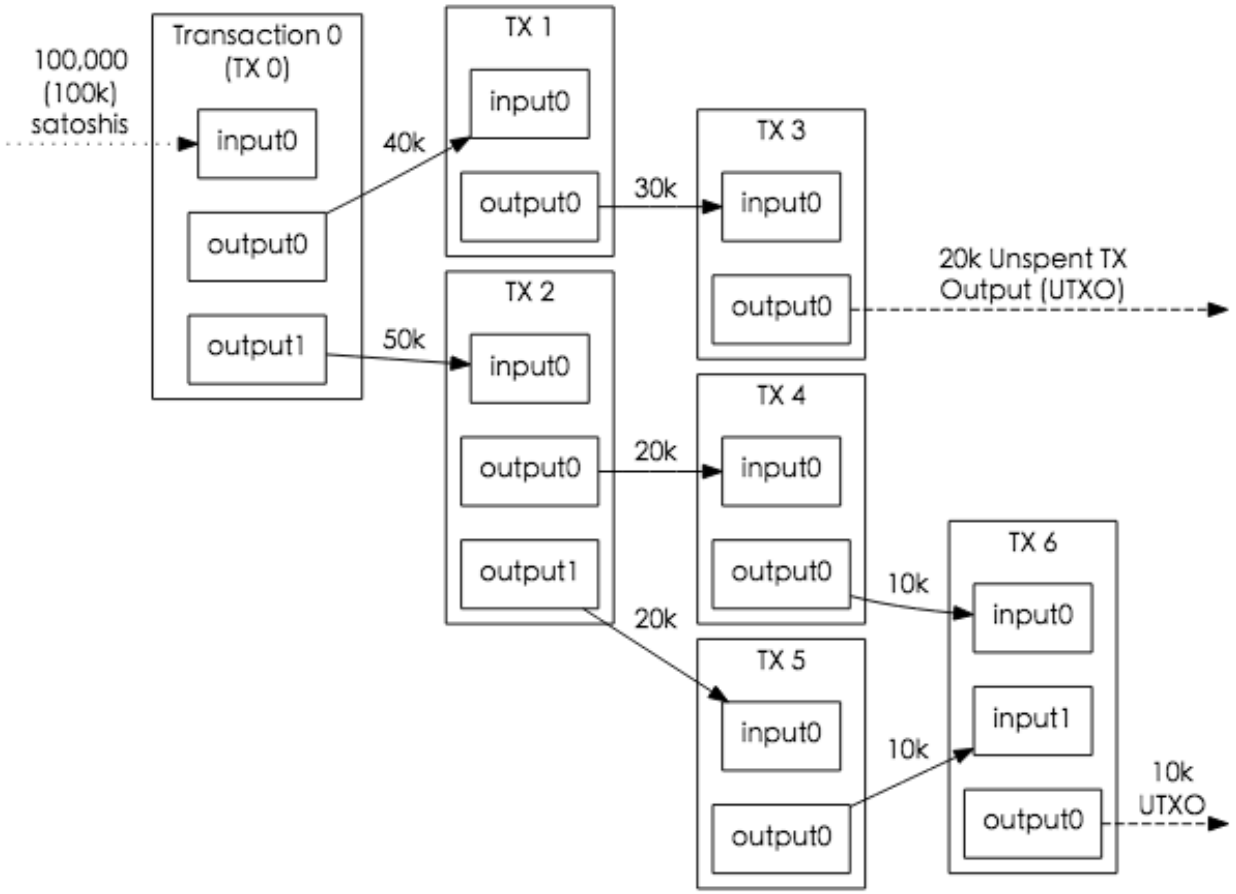
Simplified Bitcoin Block Chain

区块链概览

- 一个包含一个或者多个交易的区块 会被收集到区块的交易数据部分. 每个交易的副本都会被哈希，然后将哈希值进行配对，然后再进行哈希，再配对，再哈系，直到只剩下一个哈希值，这个剩下的哈希值就是Merkle tree(Merkle 树)的Merkle root(根节点)

- Merkle根节点被存储在区块头。每个区块也会存储上一个区块头的哈希值, 以把所有的区块链接在一起. 这保证了在不修改当前和后面的所有区块的情况下，交易记录是不会被修改的。

三方记账



Triple-Entry Bookkeeping (Transaction-To-Transaction Payments) As Used By Bitcoin

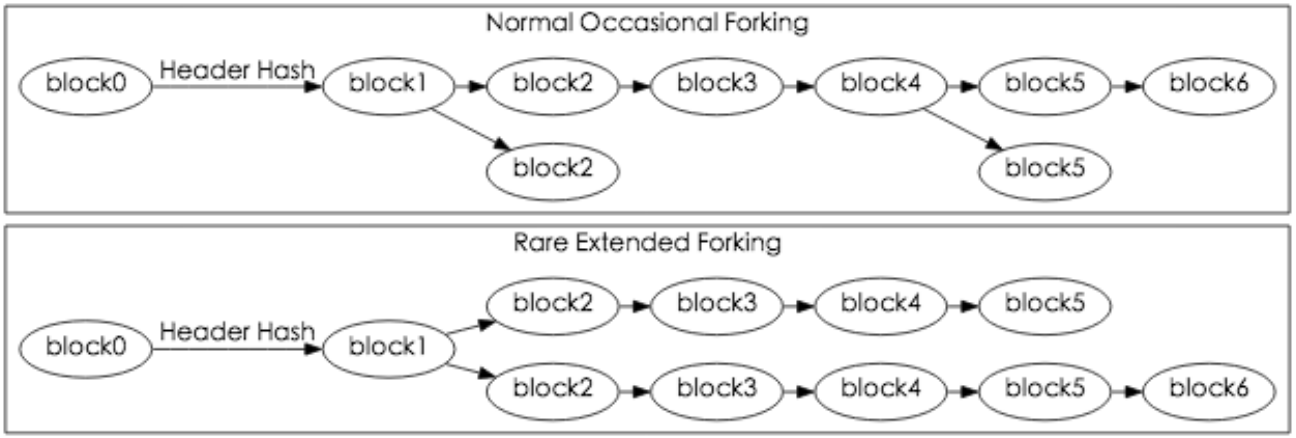
UTXO

- 一次输出在区块链中只能使用一次 所有的后续引用都禁止双花
  - 输出被分类为未使用交易输出Unspent Transaction Outputs (UTXOs)或者已使用输出
- 比特币在产生交易之后，不能留在UTXO中， 否则这个余额会永久的丢失掉, 所以输入和输出之间的差值就是交易费，都是对把这个交易打包到区块的矿工的奖励。

Proof Of Work 工作量证明

- 计算出一个不大于某个特定值的块头的哈希值
- 难度调整：比特币网络使用储存在2016个区块头的时间戳的差值来计算下一轮的难度。这个差值的理想的间隔是1,209,600秒(两周)
- 对交易历史进行51%的攻击

区块高度及分叉



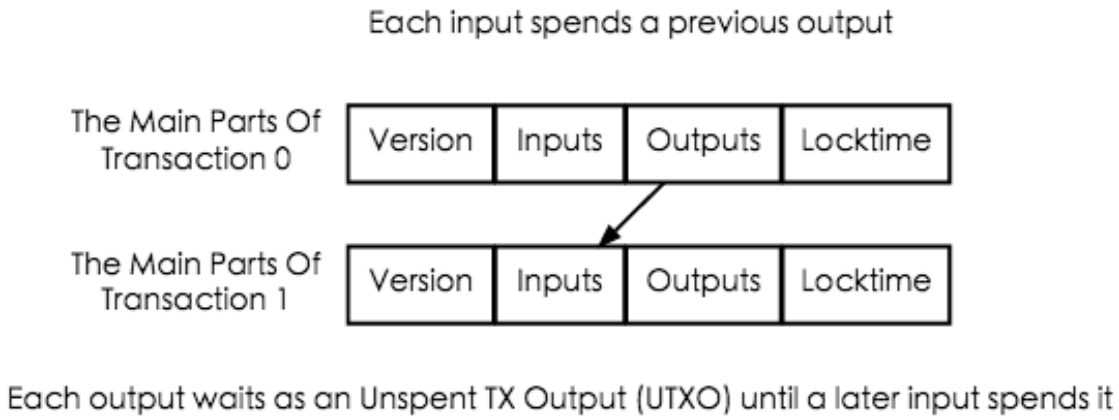
- 当前区块的高度：当前区块到第一个块(genesis block)的差值
- 正常的节点将选择最长的那个分支（最难创建的一个链），然后抛弃比它短的分支

交易数据

- 第一个交易记录必须是coinbase（币基）记录，这个记录包含了区块奖励和本区块的所有的交易费
- 挖矿获得的币在此后的100块不能使用

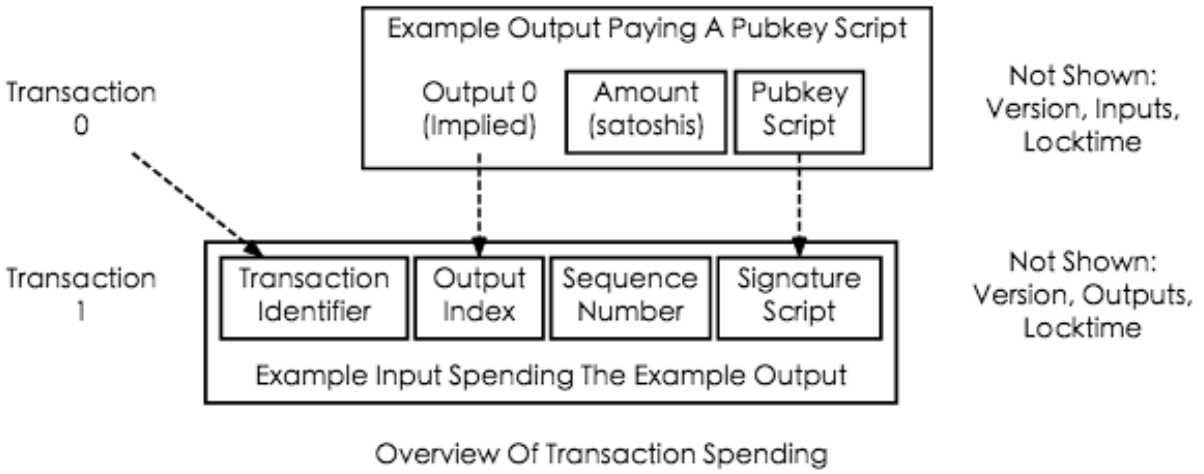
交易

交易的组成



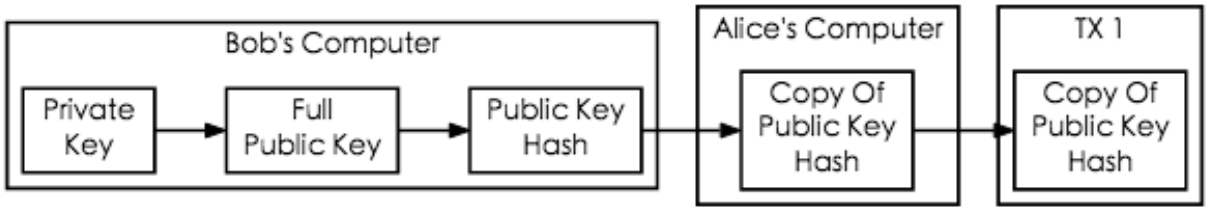
- 当你的比特币钱包告诉你余额有10BTC时，它真正的意思是说你有10BTC在一个或者多个UTXOs等待被花费出去。

交易概述



Overview Of Transaction Spending

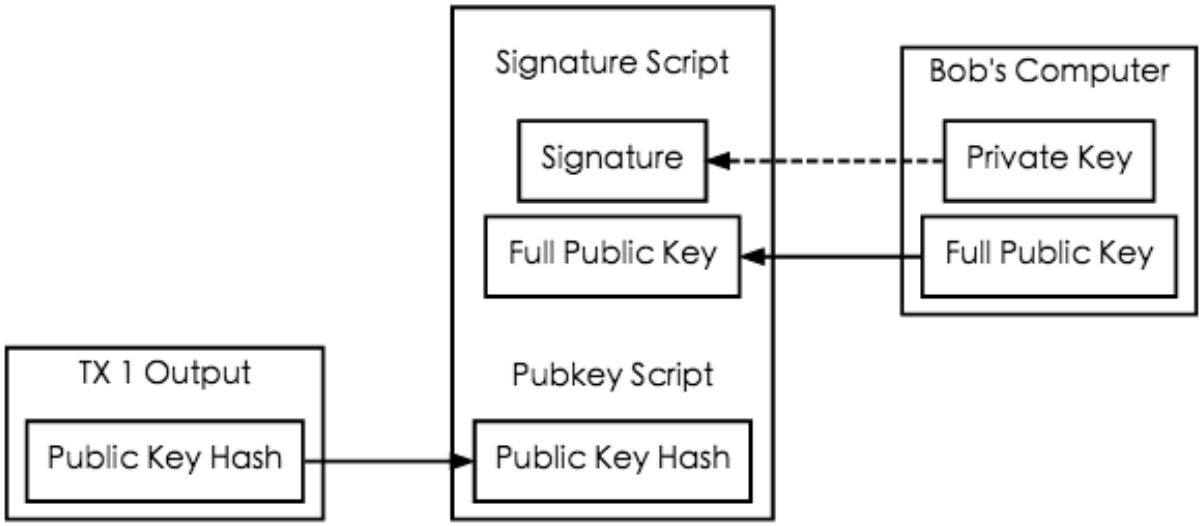
支付(P2PKH)



Creating A P2PKH Public Key Hash To Receive Payment

- 公钥Hash采用base58check编码：包含版本序号、哈希值以及用来校验错误的值

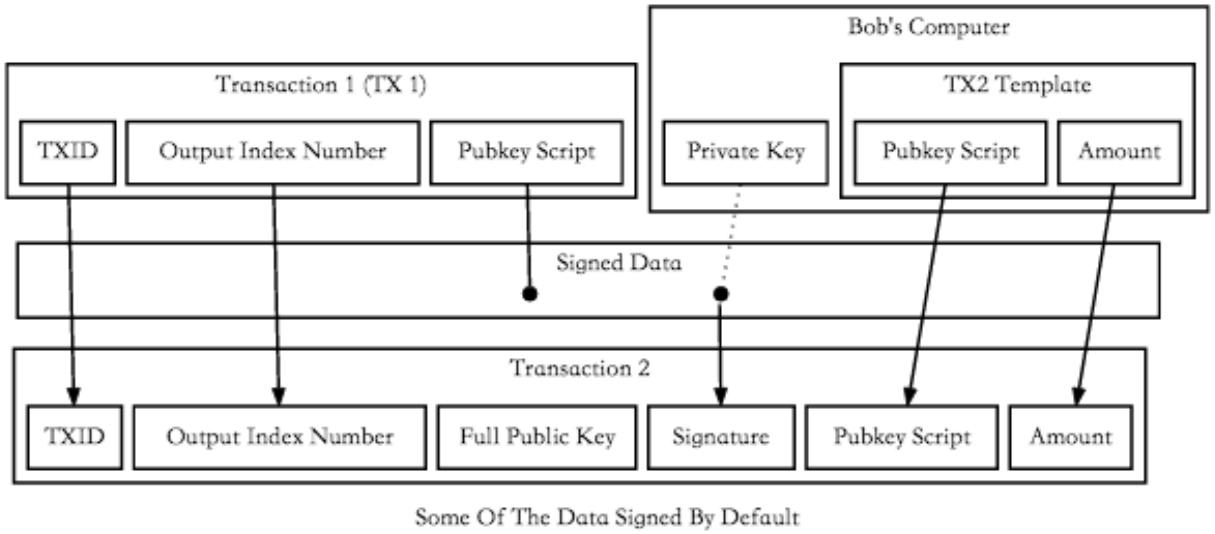
花费(P2PKH)



Spending A P2PKH Output

- Bob的scriptSig只需要包含下面两个条件即可：
  - 他提供的完整公钥（未经过哈希运算），以便于script 可以通过重新求得哈希值来验证ALICE提供的哈希值是否一致。
  - 一个用Bob提供的私钥采用ECDSA （椭圆曲线签名算法）加密算法计算出的包含确定的交易数据的签名。这个签名用来验证Bob是否拥有那个生成公钥的私钥。

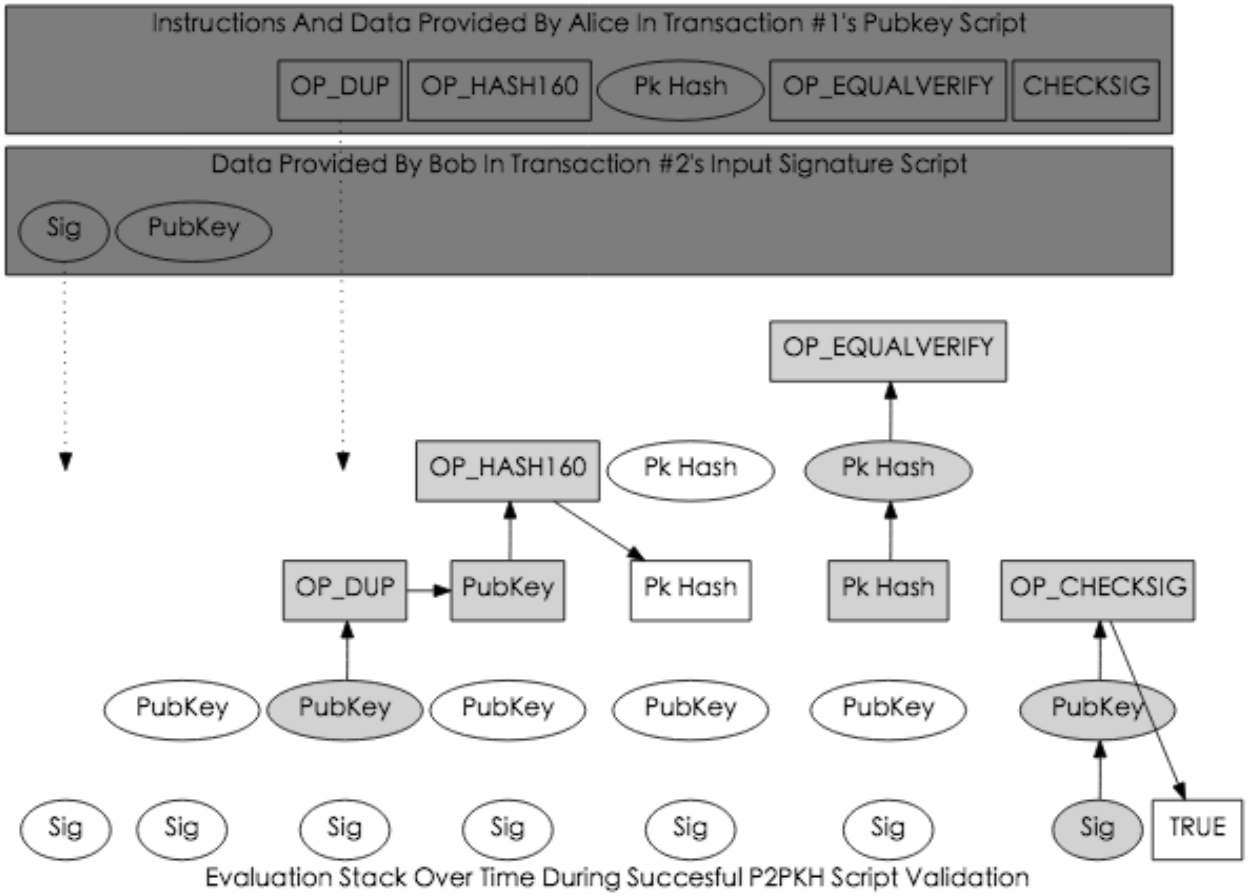
如何签名



Some Of The Data Signed By Default

- 签名包括了txid和前一个交易的输出索引、前一个交易的script、Bob创建的可以让下一个接收者花费它这个output的script、总共要转给下一具接者的比特币数量。本质上来说，整个交易都被签名除了scriptSigs。

P2PKH的交易验证

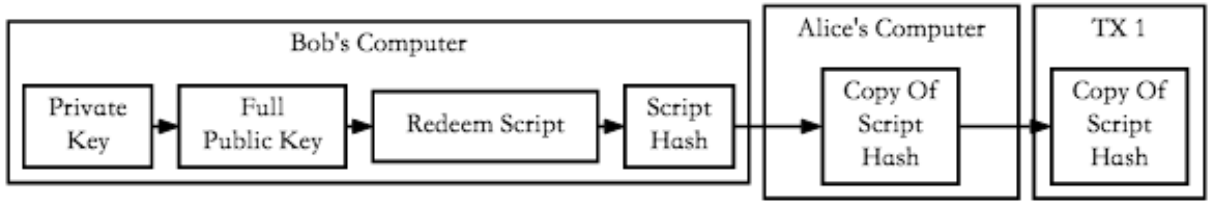


- script: OP_DUP OP_HASH160 <PubKeyHash> OP_EQUALVERIFY OP_CHECKSIG
- scriptSig: <sig> <pubkey>

标准交易-多重签名

- script: OP_HASH160 <redeemscripthash> OP_EQUAL
- redeemScript: <OP_2> <pubkey> <pubkey> <pubkey> <OP_3> OP_CHECKMULTISIG
- scriptSig: OP_0 <sig> <sig> <redeemscript>

标准交易-P2SH流程图



Creating A P2SH Redeem Script Hash To Receive Payment

- script: OP_HASH160 <redeemscripthash> OP_EQUAL
- scriptSig: <sig> [sig] [sig...] <redeemscript>

标准交易-Pubkey公钥脚本

- script: <pubkey> OP_CHECKSIG
- scriptSig: <sig>
- P2PKH脚本的简化形式，没有P2PKH脚本那么安全，新版交易里通常不用

标准交易-空数据

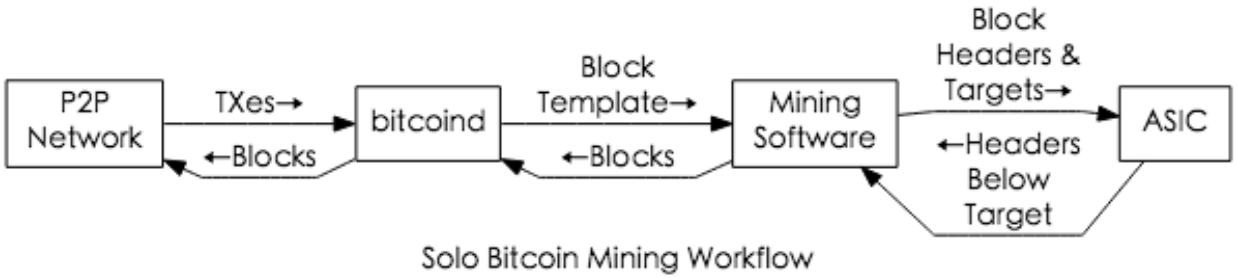
- 以支付一定交易费作为代价，添加一小块任意的数据到区块链里
- script: OP_RETURN <data>
- 空数据脚本不能被花费，所以这里不存在scriptSig

交易费用

- 交易所花的交易费通常基于签名交易的总字节大小所决定
- 每笔交易需要交最低手续费（0.9版本是1000聪）

挖矿

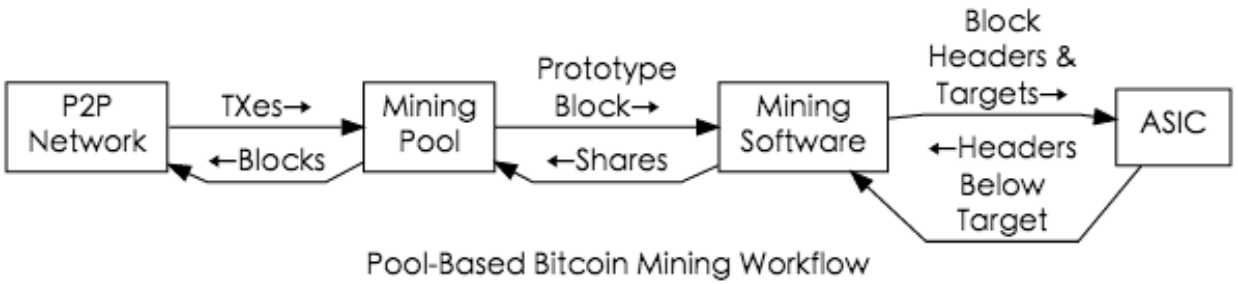
独立挖矿



Solo Bitcoin Mining Workflow

- 挖矿软件利用模板来构建一个块和创建一个块的头部，接着它把长度为80字节的块头发送到挖矿的硬件部分（如ASIC），同时发送的还有目标门限（难度系数）。挖矿硬件通过暴力方式尝试每一个可能的块头的值并产生相应的哈希值

矿池挖矿



Pool-Based Bitcoin Mining Workflow

- 矿池设置目标门槛比网络难度大了几个数量级（难度较低）。
- 如果矿池的目标门槛比全网目标门槛低100倍， 即平均每一个成功的区块需要产生100个份额，矿池将支付每个份额所得区块奖励的百分之一。

getwork RPC

- 最早且最简单的方式是现在已经弃用的核心挖矿协议是getwork RPC，它向矿工直接构建块头。
- 由于一个块头只包含一个4个字节的随机字符串（nonce），只能支持4.0 GH(gigahashes),

getblocktemplate RPC

- 这种方式给挖矿软件提供了更多信息：
  - 需要支付给矿池或者单独挖矿矿工bitcoind 钱包的coinbase交易的费用信息。
  - bitcoind钱包提供的交易信息或矿池建议需要包含的交易信息，允许挖矿软件检查交易，选择性地添加额外交易和去除非必须的交易。
  - 其他为了下一个区块构建区块标头所需的信息：区块的版本，前一个块Hash值和位（目标）。
  - 矿池当前接受份额(share)的目标阈值。（对于单独挖矿的矿工，就是全网的算力目标。）
- 采矿软件在coinbase上额外增加了一个nonce
- HTTP longpoll来保持更新

Stratum

- 最广泛使用的挖矿协议
- 侧重于给矿工自己构建标头所需的最少信息：
  - 构建一个coinbase交易需要支付矿池所需的信息。
  - 当coinbase交易更新，有了一个新的额外nonce，为了创造一个新的 Merkle root，Merkle树上的一部分需要被重新散列。而Merkle树的其它部分，如果存在的话，将不被发送。这样能够有效地限制在数据量，需要被发送的数据总量（最多）在大约一千字节。
  - 为构建下一个区块块头的非Merkle root的所有其他所需信息。
  - 矿池接受份额的当前目标阈值。
- 矿工无法在他们目前挖掘的区块上检查或添加交易
- 直接使用双向TCP套接字

挖矿软件

- https://github.com/luke-jr/bfgminer
- https://gitorious.org/bitcoin/eloipool

参考资料

- https://bitcoin.org/en/developer-documentation
- https://zh-cn.bitcoin.it
- http://book.8btc.com/master_bitcoin
- http://book.8btc.com/bitcoin-developer-guide
- http://book.8btc.com/blockchain_guide