# 前言

这周把比特币的内容进行了完善，就还差一点知识没有补充完全，下周我会把整个知识总结下来，并且开始转向以太坊方向去探索。同时下周开始，我也要学习英语方面的知识，买一些六级真题，争取到学校了把英语六级过了，并且每天都指定一些学习计划来提升自己的学习状态，这几周的学习状态都不是很好，下周要继续加油了。

# 1.比特币有哪些组成

## 1.1比特币有哪些组成?

如果说把比特币当成真正的货币，那比特币的组成有哪些成分呢？这些成分有涉及到什么知识呢？《Master Bitcoin》提到了比特币的组成：

比特币包括：

* 一个去中心化的P2P网络 （比特币协议）
* 一个公共交易账本 （区块链）
* 一套规则，用于独立的交易确认和货币发行 （共识规则）
* 一个机制，用于对有效区块链实现全球去中性化共识 （工作量证明算法）

简化来说，1.协议。2.区块链。3.共识规则。4.工作量证明。

其中所涉及到的知识大体包括：**1.密码学基础。2.比特币的数据结构。3.共识协议4.系统实现。5。比特币网络。6.挖矿算法和难度调整。7.比特币脚本。8.软分叉和硬分叉。9.匿名和隐私保护作用**。其中每一部分知识我会在下面慢慢展开。

# 2.比特币的知识

## 2.1密码学基础

由于比特币属于加密货币(crypto-currency)，当然会涉及到密码学的知识，比特币属于一种开源信息的货币，其中的交易记录，交易金额，交易信息都面向全网开放。

比特币中加密涉及到两个内容：1.hash(哈希)。2.签名

1.密码学中用到的哈希函数被称为cryptographic hash function: 它有两个重要的性质:

1. collision(这里指哈希碰撞) resistance
2. hiding 哈希函数的计算过程是单向的，不可逆的。

除了密码学中要求的这两个性质，比特币中用到的哈希函数还有第三个性质：

1. puzzle friendly 指哈希值的预算事先是不可预测的。

比特币中用的哈希函数叫做**SHA-256(secure hash algorithm)**以上三个性质它都是满足的。这里这三个性质其实有详细解释，但是考虑到字数，我会放到附录的链接中。

2.签名又涉及到哪些密码学呢？在比特币系统中创建账户:在本地创立一个公私钥匙对(public key ,private key)，这就是一个账户。其中公私钥匙对是来自于非对称的加密技术(asymmetric encryption algorithm)。非对称密钥是用一对密钥而不是一个，加密用公钥，解密用私钥，加密和解密用的都是接收方的公钥和私钥。公钥是不用保密的，私钥要保密但是私钥只要保存在本地就行，不用传给对方。

通俗来讲，公钥相当于银行账号，别人转账只要知道公钥就行，私钥相当于账户密码，知道私钥可以把账户上钱转走。总体上来讲就是公钥和私钥都是用来签名的。然而这种签名机制就是采用密码学中的非对称加密。

## 2.2比特币的数据结构

比特币所涉及到的数据结构包括：

1. 哈希指针(Hash pointer)
2. 默克尔树(Markle Tree)

1.其实比特币最基本的结构就是区块链，区块链就是一个一个组成的链表，但区块链与普通链表不同的是：区块链采用了哈希指针代替了普通指针。这样的特点是，每个区块都根据自己的区块内容生成自己的哈希值，并且每个区块都要保存前一个区块的哈希值。其实就是有了哈希指针就可以保证该区块的唯一性，并且保证整个区块一旦写入就无法进行篡改。

2.而对于默克尔树Merkle Tree，该数据结构类似于我们之前学到的二叉树一样。

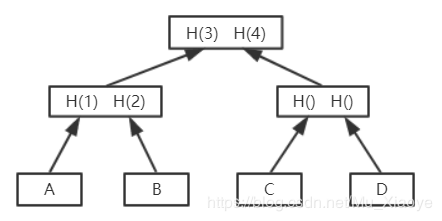


图2-1Markle Tree的示例

这种结构的好处：只要记住哈希值，就能检测出对书中的任何部位的修改。比特币当中各区块之间用哈希指针连接在一起，每个区块所包含的交易组织成一个merkle tree的形式，最下面一行data blocks每个区块实际上是一个交易，每个区块分为两部分，分别是块头和块身(block header ,block body)。块头里面有根哈希值，每个区块所包含的所有交易组成的merkle tree的根哈希值存在于区块的块头里面，但是，块头里没有交易的具体内容，只有一个根哈希值，块身里面是有交易的列表的。同时merkle tree也提供了merkle proof 来证明merkle tree 里包含某个交易。如下图所示轻节点向全节点发送请求获取H()来验证tx交易是否在区块中，如果最后H()所求得的值与Merkle Root值相同的话，就达到了merkle proof的作用，证明了交易中心确实存在这个交易。

利用下面的图可以理解merkle tree的应用场景：

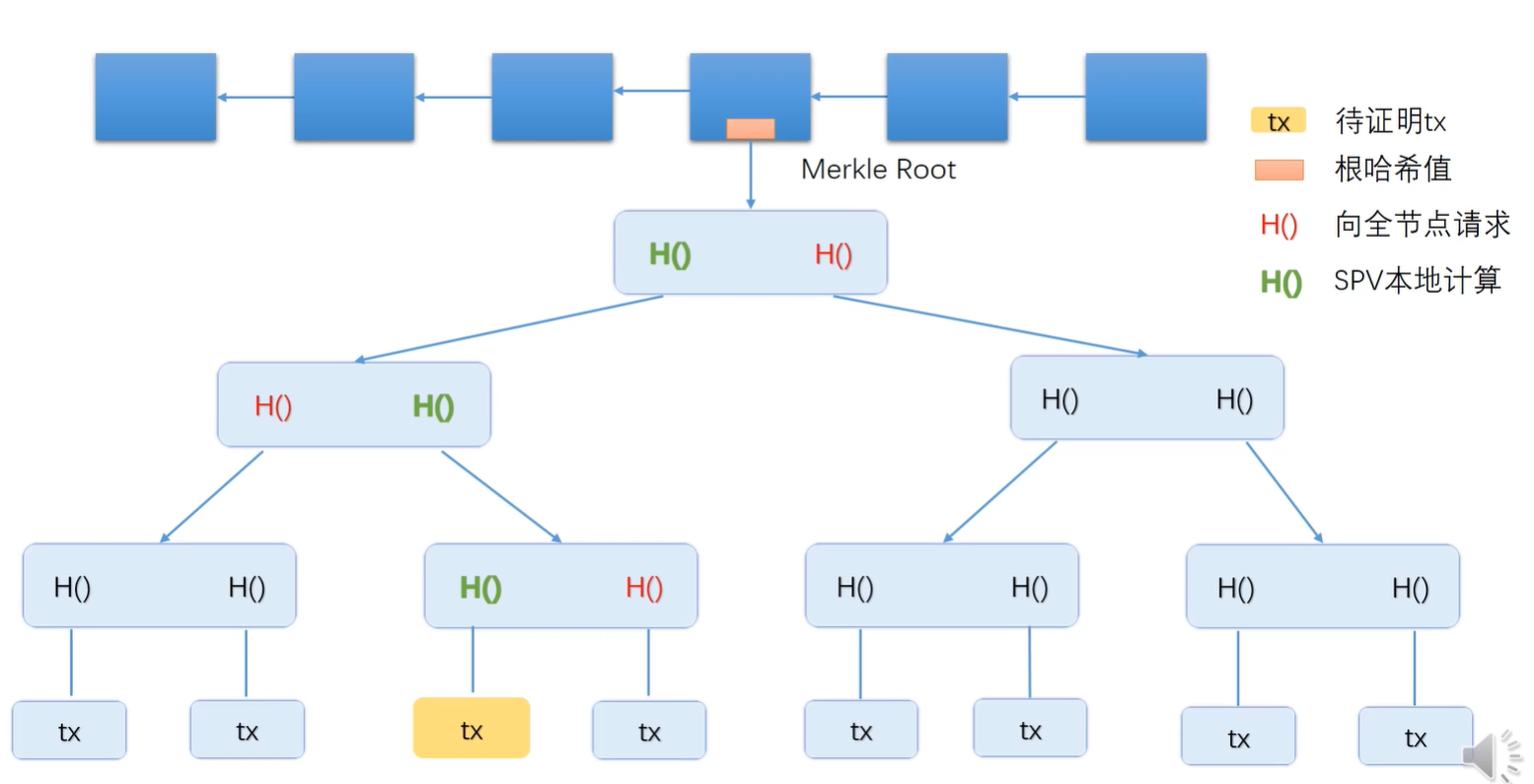


图2-2Merkle Tree的应用场景

## 2.3比特币的协议

### 2.3.1比特币的发行和交易

去中心化货币要解决两个问题:①数字货币的发行②怎么验证交易的有效性，防止double spending attack。在传统中心化货币体系中，这些问题我们可以交给第三方机构（如：央行）。当引入去中心化思想后，系统中节点平等，交易不通过第三方，那么货币发行权的分配必然是一个需要解决的问题。而双花攻击，顾名思义就是一个货币花了两次。对用户来说，可以将同一货币花费两次。因为数字货币本身为带有签名的数据文件，是可以进行复制的。

当然作为去中心化的货币，比特币是要克服这些问题的，比特币采取：1.比特币的发行是由挖矿决定的来解决发行问题，比特币协议中规定获得记账权的节点在发布的区块里可以有一个特殊的交易:铸币交易。在这个交易里可以发布一定数量的比特币。2.依靠区块链的数据结构来解决双花问题。

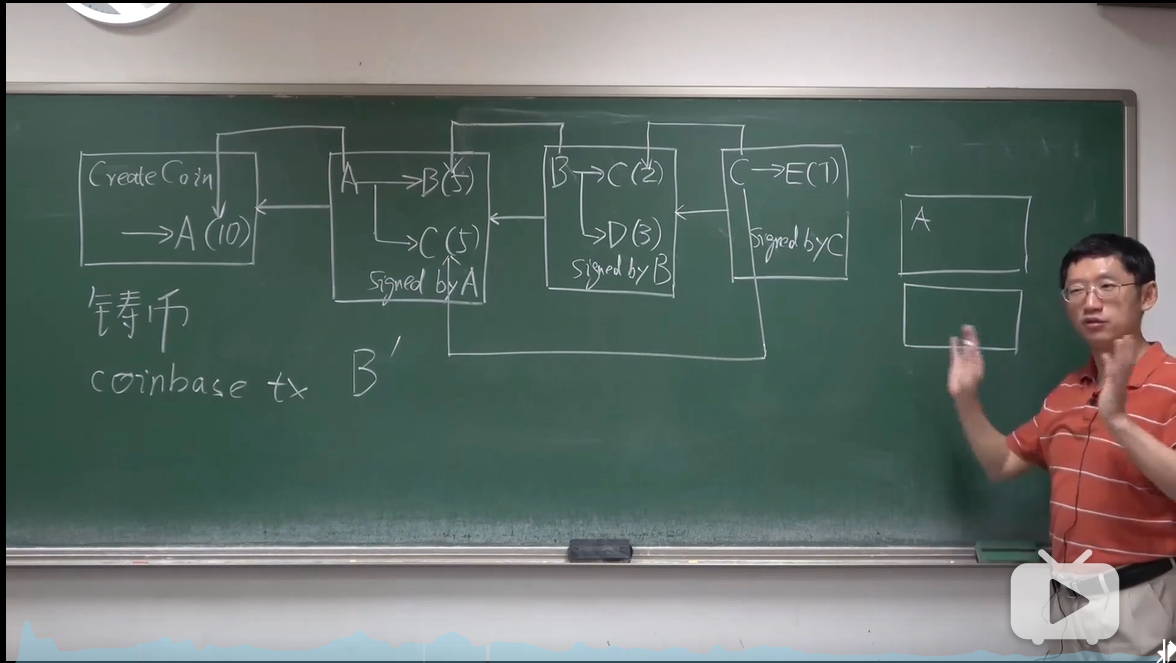


图2-3 Bitcoin的交易情况

从图中我们能够看到初始的币来源于矿工挖块得到的区块奖励，也就是铸币，是由系统分配的，也就是上面我们所提到的：比特币的发行是由挖矿决定的来解决发行问题。然后其中当A和B进行交易时，系统回去寻找A这个币的来源，通过哈希指针去寻找。同时当A与B,C交易后，B，C在进行交易转账时，系统也会根据哈希指针去寻找是谁给了B和C钱，依次类推。总之，当我们在比特币系统中进行交易时，系统会查看我们当前交易之前的记录，才能完成我们现在的交易，来保证我们这些钱是真的存在，而且还没有花出去，系统才能执行我们所发出的交易。这样就保证了防止双花攻击，不能同一份钱花两次。

### 2.3.2比特币共识协议(consensus in BitCoin)

区块链里的内容是如何写到区块链里面的呢:每个节点，每个账户都可以发布交易，交易是广播给所有节点的。有些交易是合法的，有些是非法的。谁来决定哪些交易应该被写入下一个区块中呢?按照什么顺序写呢?如果每个节点自己决定可以吗?如果每个人在本地维护一个区块链，那区块链的统一性得不到保证，而账本的内容是要取得分布式的共识(distributed consensus)。于是就引来了比特币共识协议，在这里我也了解到，一个去中心化的系统必然是分布式的，但分布式系统不一定是一个去中心化的。

概括来讲，比特币中的共识协议需要解决的最大的问题就是，有些节点可能是有恶意的。我们假设系统中大多节点是好的，如何取得共识就很重要了。

我们通常在生活中一般采取的共识都是投票，但是这种方式往往带来很多问题，如果比特币种按照账户的数目进行投票的话，如果每个人可以创建多个用户，也就可以投多张票，这中方法就会显得很危险了,很有可能导致51%攻击，俗称女巫攻击(sybil attack)。

比特币系统巧妙的解决了这个问题，虽然仍然是投票，但并非简单的根据账户数目，而是采用**计算力进行投票，**也就是采用POW(Proof Of Work)机制，每个节点都可以在本地组装出一个候选区块，把它认为合法的交易放在里面，然后开始尝试各种nonce值(占4 byte)，看哪一个能满足不等式H(block header)≤target的要求。如果某个节点找到了符合要求的nonce，它就获得了**记账权**。从而可以将区块发布到系统中。其他节点受到区块后，验证区块合法性，如果系统中绝大多数节点验证通过，则接收该区块为最新的区块并加入到区块链中。

所谓的记账权，就是往比特币账本里写入下一个区块的权利。只有找到这个nonce，获得记账权的节点才有权利发布下一个区块。其他节点收到这个区块之后，要验证这个区块的合法性，同时也要验证区块中的交易是否也合法。如果有一项不符合要求，这个区块就是不能被接受的。如果所有条件都符合，也不一定接受。

那所有条件都符合，为什么还不一定接受呢？这就引出了区块分叉的问题，如图

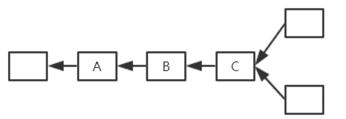


图2-4 区块的分叉

图中的两个区块都符合上述的判断，但由于是同一时间段发布，导致区块链产生了分叉，这种分叉情况一般分两种：一种是确实两个恰好同时挖出区块来，一种就是属于分叉攻击了。前者发生分叉的情况下，会暂时保存分叉情况，在缺省情况下，节点接收最先听到的区块，该节点会沿着该区块继续延续。但随着时间延续，必然有一个链胜出，由此保证了区块链的一致性。（被扔掉的区块称为“孤儿区块orphan block”），但区块链只承认最长合法链，随着时间推移，必然存在某一条链变成最长合法链。这样，也就会导致合法区块也会被拒绝。而后者就相对比较严重是属于分叉攻击，是一些非法矿工想要从中获取利益，分叉攻击如图：

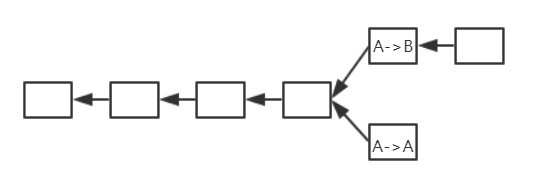


图2-4 分叉攻击

A用户对上面的A转账给B的记录回滚，从而非法获取利益。在两条链上，发现交易都合法。这是一个典型的双花攻击。A给B转账后，用分叉攻击将钱又转回来，覆盖掉原来的记录。

在比特币系统中，这种情况实际上很难发生。因为大多数矿工认可的是最长的合法链，会沿着上面的链继续挖下去。而A这个攻击者要想回退记录，就必须使得下面的链变得比上面的链还长。理论上来说，攻击者需要达到整个系统中51%的计算力，才能使得这种攻击成功。但要达到51%的计算力这种能力去办坏事，显然所涉及到的成本是在太大了。

## 2.4比特币的系统的实现

### 2.4.1UTXO

区块链是去中心化的账本，比特币使用的是基于交易的这种账本模式(transaction【交易】-based ledger【账本】)。系统当中并不会显示每个账户有多少钱。

比特币系统的全节点要维护一个叫UTXO (unspent transaction output)(还没有被花出去的交易的输出)的数据结构。区块链上有很多交易，有些交易的输出可能已经被花掉，有些还没有被花掉。所有没有被花掉的输出的集合就叫做UTXO。

一个交易可能有多个输出。假如A给B 5个比特币，B花掉了。A也给了C 3个比特币，C没有花掉。这时5个比特币就不算UTXO，而3个比特币算。UTXO集合当中的每个元素要给出产生输出的交易的哈希值，以及它在这个交易里是第几个输出。这两个信息就可以定位到UTXO中的输出。

那要UTXO集合有什么作用？为了检测上文的double spending(双花)。即检测新发布的交易是否合法。因此全节点要在内存中维护UTXO这样一个数据结构，以便快速检测double spending。

每个交易要消耗掉一部分输出，也会产生新的输出。还看上面的例子，B花掉的5个比特币虽然不在UTXO里面，但如果他转账给D，而D没有花掉，那么这5个比特币又要保存在UTXO里面。如果D始终不花，那么这个信息要永久保存在UTXO里面。有可能是不想花，也有可能是把密钥丢了。每个交易可以有多个输入，也可以有多个输出，所有输入金额之和要等于输出金额之和。即total inputs=total outputs。因此一个交易可能来自多个地址，可能有多个签名。有些交易total inputs略微大于total outputs。假如输入1比特币，输出0.99比特币，另外0.01比特币作为交易费给获得记账权发布区块的节点。

### 2.4.2比特币的激励机制

可能我们看到上面，每个节点需要提供能算力和电力成本，要投入大量的资金去挖矿，就是为了竞争记账权，为什么呢？

比特币系统设计之初便考虑到了这个问题，那就是引入激励机制。比特币通过设置**出块奖励**来解决该问题，一个获得合法区块的节点，可以在区块中加入一个特殊交易（铸币交易）。事实上，这种方式也是唯一一个产生新比特币的途径。比特币系统设计规定，起初每个区块可以获得50个比特币，但之后每隔21万个区块，奖励减半。

区块奖励也不能完全作为挖矿的奖励，发布区块的节点为什么一定要把你的交易打包在区块呢?他们还要验证你的交易的合法性，如果交易较多占用的带宽会比较大，网络传播速度也会更慢。所以这时只有区块奖励是不够的。

因此比特币系统设计了第二个激励机制:交易费(transaction fee)。也就是你把我的交易打包在区块里，我给你一些小费。交易费一般很小，也有一些简单的交易没有交易费。

### 2.4.3比特币的挖矿

这里简要提及挖矿的概念，后面还会详细讲解，其实挖矿就是不断的通过随机值nonce来算出对应的哈希值与目标值进行比对，但是nonce毕竟只有32位，如今随着挖矿难度越来越大，该范围已经不足以去挖矿了，这是就要考虑block header里面的其他域的值是否能够改变。

block区块头里面的各个域包括：

1. 版本号(32位)
2. 前一个区块头的哈希值(32\*8位)
3. merkle tree的根哈希值(32\*8位)
4. 区块产生的时间time(32位)
5. 目标域值(32位)
6. nonce(32位)

第一行:比特币协议的版本号(无法更改的)

第二行:前一个区块的块头的哈希值(无法更改)

第三行:merkle tree的根哈希值(可以更改)

第四行:区块产生的时间(可以调整)比特币系统不要求特别精确的时间，可以在一定范围内调整。

第五行:目标预值(编码后的版本)(只能按协议中的要求定期调整)

第六行:随机数

综上所述，挖矿时只改随机数nonce还不够，我们可以更改根哈希值(merkle tree )，这里可能会有一些疑问？根哈希值不是为了提供merkle of proof，来证明交易的正确性吗(这点知识图2-2有所提到)？其实这里利用了区块的铸币交易的coinbase域，如图2-5，铸币交易没有输入，它有一个coinbase，可以写入任何的内容。也可以把digital commitment里的commit的哈希值写入里面。也可以把第一节讲到的预测股市的内容写入里面，coinbase的内容是没有人会检查的，甚至可以写你的心情。

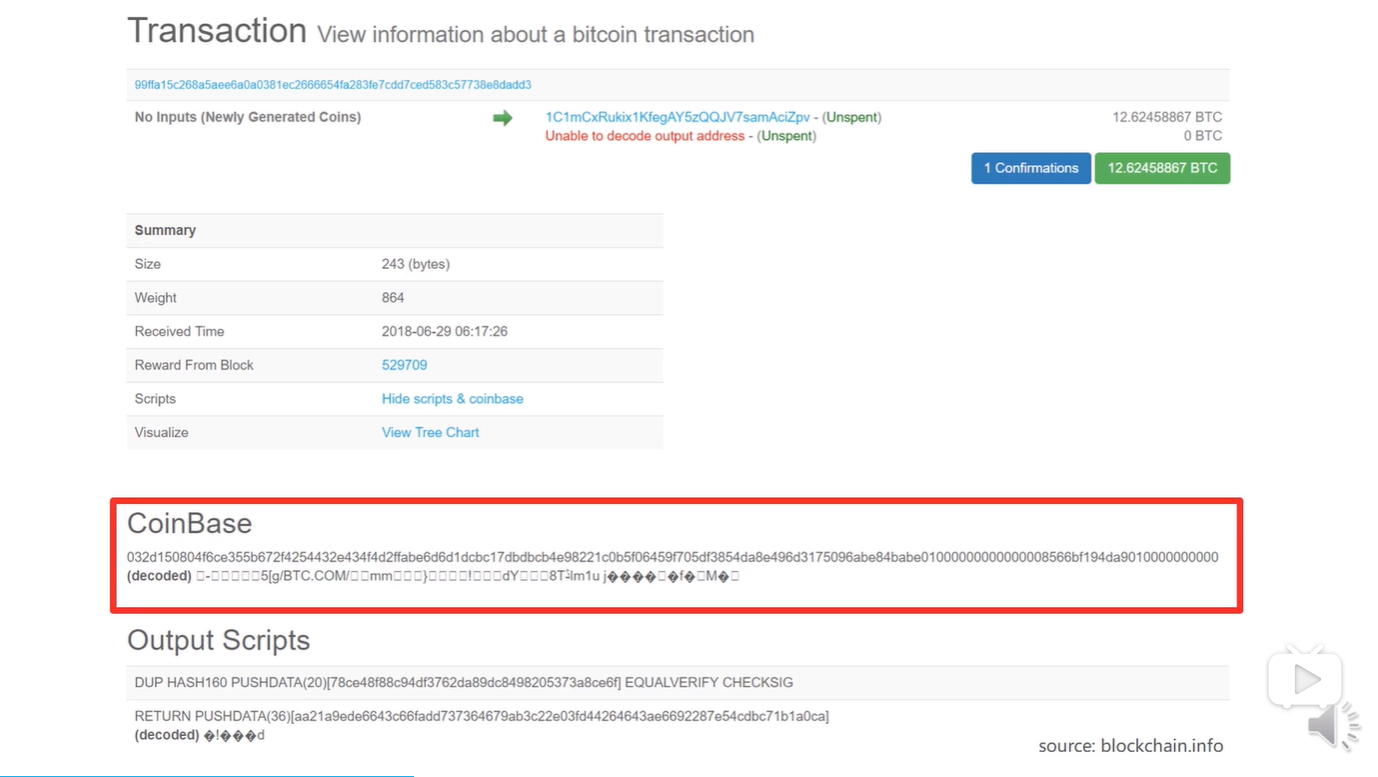


图2-5铸币交易的coinbase域

铸币交易没有输入，它有一个coinbase域，可以写入任何的内容。也可以把digital commitment里的commit的哈希值写入里面。也可以把预测股市的内容写入里面，coinbase的内容是没有人会检查的，甚至可以写你的心情。

如上面所提到的图2-2对应的是最后一个block header里的根哈希值对应的merkle tree，左下角的交易是coinbase，把它的域改了之后，其上的哈希值就发生了变化，然后沿着merkle tree的结构往上传递。最后导致block header里的根哈希值发生变化(merkle root是block header的一部分)。块头里4个字节的nonce不够用，还有其他字节可以用，比如coinbase域的前八个字节当做extra nonce来用，这样子搜索空间就增大到了2的96次方。

所以真正挖矿的时候只有两层循环，外层循环调整coinbase域的extra nonce。算出block header里的根哈希值之后，内层循环再调整header里的nonce。

总体来说，挖矿就是求解puzzle的过程。挖矿过程每次尝试一个nonce可以看作是一个Bernoulli trial(伯努利实验)。每一个随机的伯努利实验就构成了一个伯努利过程。它的一个性质是:**无记忆性（memoryless）**。每尝试一个nonce成功的概率是很小的，要进行大量的实验。这时可以用泊松过程(Poisson process)来代替伯努利过程。我们真正关心的是系统出块时间，出块时间是服从**指数分布(exponential distribution)**。可以画出一个坐标轴，纵轴表示概率密度，横轴表示出块时间(整个系统的出块时间，并不是每个矿工的出块时间)，如图2-6。具体到每一个矿工，他能挖到下一个区块的时间取决于矿工的算力占系统算力的百分比。

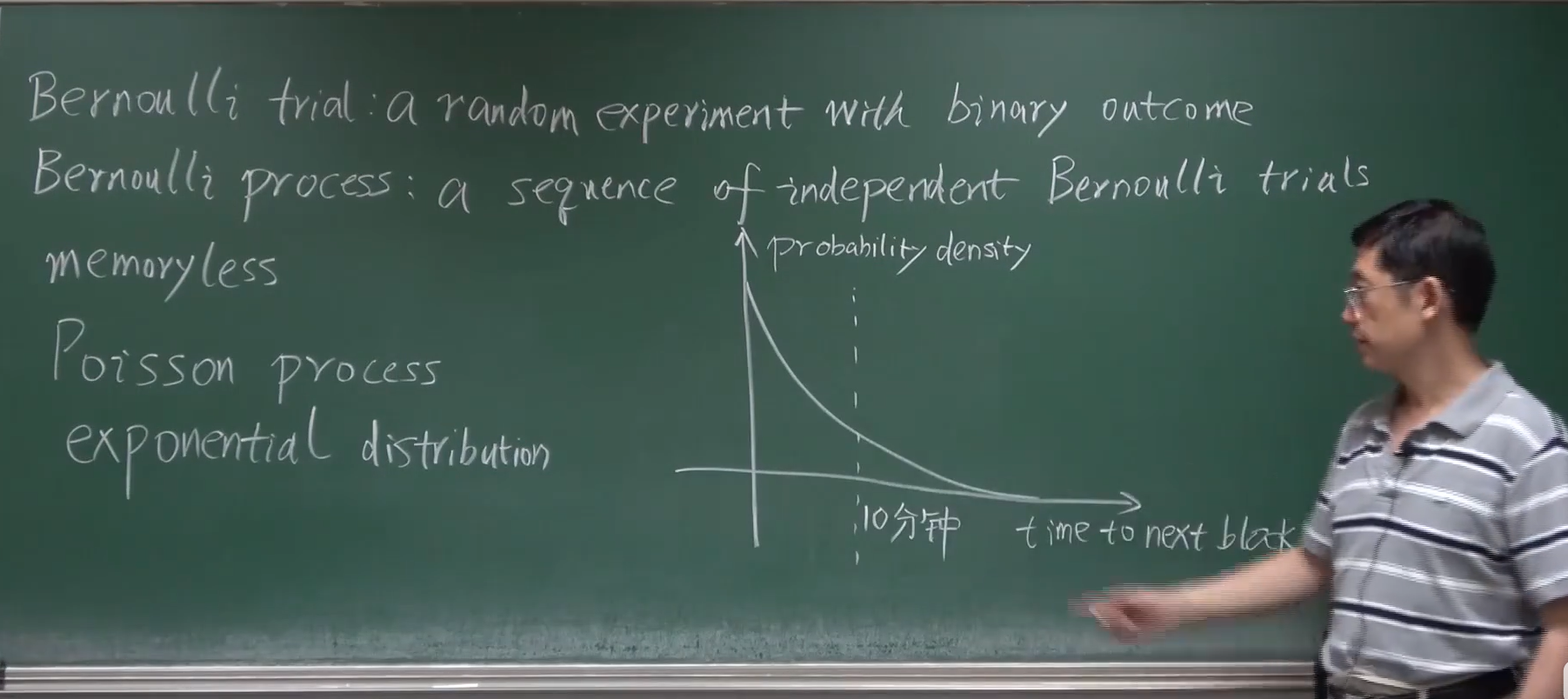


图2-6概率密度和出块时间坐标轴

假如一个人的算力占系统总算力的1%，那么系统出100个区块，就有一个区块是这个人挖的。指数分布也是无记忆性的。因为概率分布曲线的特点是:随便从一个地方截断，剩下一部分曲线跟原来是一样的。比如:已经等十分钟了，还没有人找到合法的区块，那么还需要等多久呢?仍然参考概率密度函数分布 ，平均仍然要等十分钟。将来还要挖多长时间，跟过去已经挖了多长时间是没有关系的。这个过程也叫:**process free**。如果没有process free ，会出现什么现象:算力强的矿工会有不成比例的优势。因为算力强的矿工过去做的工作是更多的，过去尝试了那么多不成功的nonce之后，后面nonce成功的概率就会增大。以此process free 是挖矿公平性的保证。

出块奖励是系统中产生新的比特币的唯一途径。产生的比特币构成的一个几何序列。21万＊50+21万＊25+21万＊12.5+......=21万＊50＊(1+1/2+1/4+......)=2100万。比特币求解的puzzle，除了比拼算力之外，没有其他实际意义。比特币的稀缺性是人为造成的。

虽然挖矿求解puzzle本身没有实际意义，但是挖矿的过程对于维护比特币系统的安全性是至关重要的。挖矿提供一种凭借算力投票的有效手段，只要大部分算力是掌握在诚实的节点手里，系统的安全性就能够得到保证。

虽然挖矿奖励越来越小，难度越来越大，但这几年挖矿的竞争是越来越激烈的，因为比特币的价格是飙升的。最终区块奖励为0了，是不是就没有动力挖矿了呢?不是的，因为还有交易费激励机制。

### 2.4.5比特币交易的问题

我们可能会考虑这样的问题：假设大部分算力是掌握在诚实的矿工手里，我们能得到什么样的安全保证?能不能保证写入区块链的交易都是合法的。挖矿给出的只是概率上的保证，只能说有比较大的概率下一个区块是由诚实的矿工发布的，但是不能保证记账权不会落到有恶意的节点手里。

比如好的矿工占90%的算力，坏的矿工占10%的算力。那么10%的概率下记账权会落在有恶意的矿工手里，这时候会出现什么情况?

首先考虑第一个问题:他能不能偷币？能不能把别人账上的钱转给自己?不能，因为他没有办法伪造别人的签名。

假设M是有恶意的，他想把A账上的钱转走，所以他发布一个A转给M的交易，但这个交易需要有A的签名，M虽然获得记账权，但他不知道A的私钥，所以伪造不了签名。如果M把交易硬写在区块链上，诚实的节点不会接受这个区块，因为它包含有非法的交易。所以诚实的节点会继续沿前一个区块挖，生成新的区块代替非法的区块，其他诚实的区块会沿着这个合法的区块继续挖。比特币要求是扩展正常合法链，M生成的不是合法区块，所以该区块作废。这对他造成的代价是很大的，因为没有了区块奖励，又没有偷到钱。

第二个问题:他能不能把已经花了的币再花一遍(即double spending)?假如他把M→A的交易写在了一个区块里面，现在他获得了记账权，他又发布另一个交易，把这个钱转回给自己，即M→M'，如图2-7。同样，这很明显是double spending，只要是诚实的节点都不会接受这个区块。他如果想发布这个区块，只能连在写了M→A交易区块的前一个区块。注意:区块插在哪个位置，在刚挖矿时就是要决定的，因为设置的block header里要填上前一个block header的哈希。所以他想插到那个区块的话，一开始就要认定，而不是等获得记账权以后再认定。

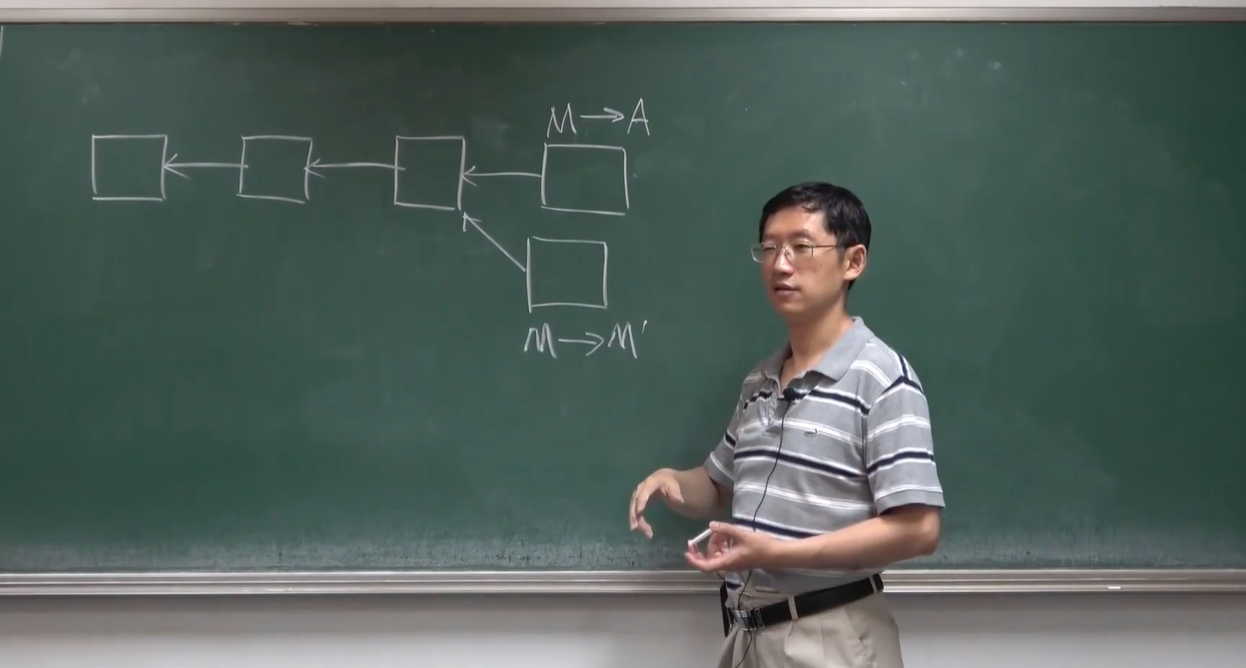


图2-7 Double spending

如图2-7所示，这样生成的两条区块链，都是合法的。要看其他节点沿着哪一个链往下扩展，最后一个胜出一个作废。这种攻击的目的是什么？如果M→A的交易，产生了某种不可逆的外部效果，然后M→M'再把M→A的交易回滚了，那么M就可以从中不当获利。比如：网上购物时，M购买一些商品，然后该网站接受比特币支付，M发起一个交易把账转给网站。网站监听到交易写入了区块链里，以为支付成功了，所以就把商品给了M。M拿到商品之后，又发起一个交易，把支出的钱转给自己，然后把下面的链拓展成最长合法链。这样的结果是:既得到了商品，又收回了花掉的钱，就达到了double spending的目的。

如何防范这种攻击呢？如果M→A的交易所在的区块不是最后一个区块，那么这种攻击的难度就会大大增加。要是想回滚M→A的交易，还是要插在它之前的一个区块，然后想办法成为最长合法链。这个难度是很大的。因为诚实的节点，不会沿着它生成的区块往下扩展，因为它不是最长合法链。因此防范这种攻击的方法就是多等几个区块，或者叫多等几个确认confirmation。

M→A交易刚刚写入区块里时，我们把它叫作one confirmation。这时后面加的区块，依次叫two confirmation、three confirmation...比特币协议当中，缺省(系统默认)的是要等六个confirmation。有了六个confirmation，才认定M→A的交易是不可篡改的，如图2-8所示。这需要等多长时间呢?平均出块时间是10分钟，因此要等一个小时。

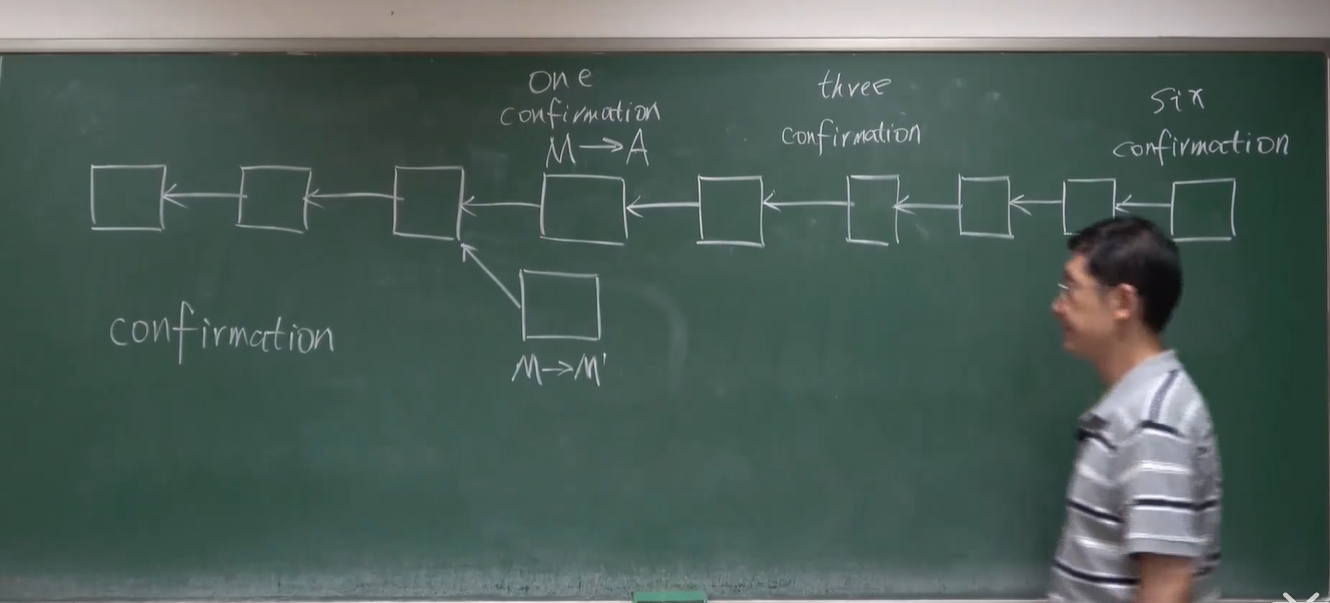


图2-8 six confirmation

区块链是不可篡改的账本，那是不是意味着凡是写入区块链中的内容就永远改不了呢?经上述分析可以看出，这种分析只是一种概率上的保证。刚刚写入区块链的内容，还是比较容易被改动的。经过一段等待时间之后，或者后面几个区块被确认之后，被篡改的概率就大幅度下降(指数级别的下降)。

其实还有第二种解决方法，叫零确认(如图2-9)。意思是说，这个转账交易发布出去了，但还没又被写入区块链里。即M→A的交易已经发布，但下面包含M→M'的区块还没有被挖出来。

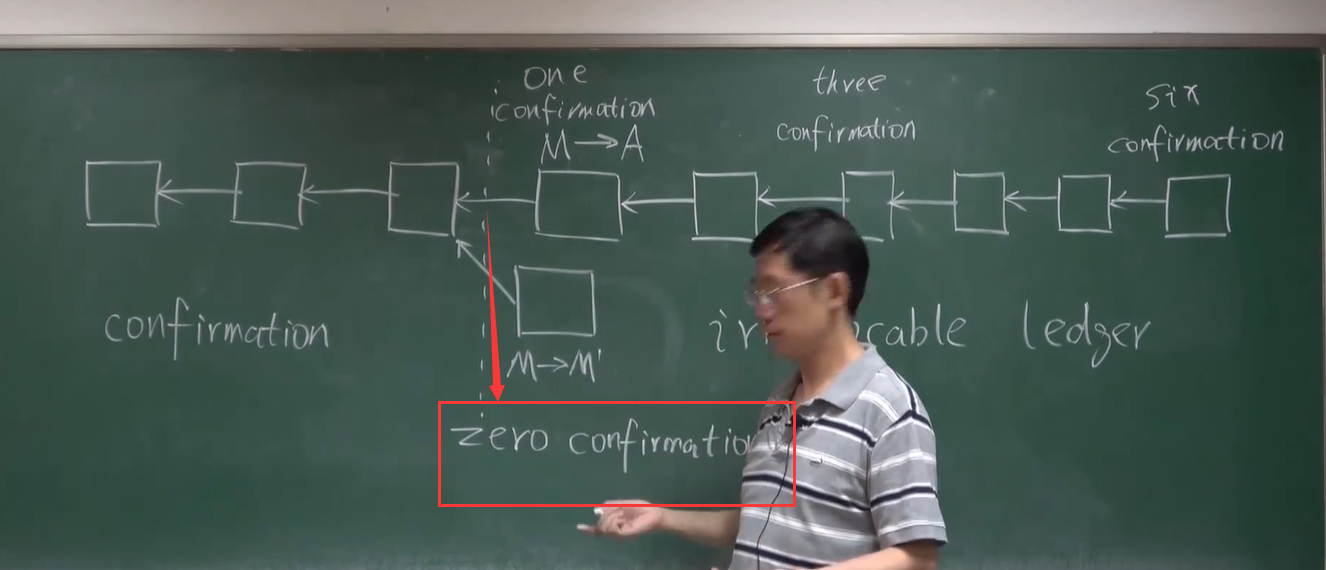


图2-9 zero confirmation

这个概念相当于电商购物的例子中，在支付时你发布一个转账交易，告诉电商自己已经转过钱了。电商运行一个全节点或委托一个全节点监听区块链上的交易，他收到转账交易之后要验证该交易的合法性(有合法的签名，以前没有被花过)，甚至不用等到该交易写入区块链里。这种操作听起来风险很大，交易刚发布出去，都没往区块链里写呢。其实，零确认在实际当中，用的还是比较普遍的。为什么呢?

这其中有两个原因:①比特币协议缺省的设置是节点接收最先听到的那个交易。所以在零确认的位置，M→A的节点收到后，再发M→M'的交易，有比较大的概率诚实的节点是不会接受的。②很多购物网站，从支付成功，到发货，是有一定的时间间隔的，即有一定的处理时间。

回到前面的问题:假设某个有恶意的节点获得记账权，它还能做什么坏事?能不能故意不把某些合法的交易写入区块链里?即发布的区块故意不包含某些交易。这是可以的。比特币协议并没有规定获得记账权的节点一定要把那些交易发布到区块里。但出现这种情况问题也不大，因为这些合法的交易一定会被写入下一个区块里，总有诚实的节点愿意发布这些交易。其实，区块链在正常工作下，也会出现合法的交易没有被包含进去的情况，可能就是这段时间交易的数目太多了。比特币协议中规定，每个区块的大小是有限制的，最多不能超过一兆字节。所以如果交易的数目太多了，那么有些交易可能就只能等到下一个区块再发布。

会不会出现这种情况？M→M'的交易所在的区块所在的链条虽然短，但是先偷偷的生成比上面更多的区块，然后等上面的链条公布后再公布，就能够胜过上面的几个区块了?这种方法叫作selfish mining。正常情况下挖到一个区块马上就发布，原因是你不发布别人可能就发布了，那样就拿不到区块奖励了。而selfish mining是先藏着不急着发布，这是分叉攻击的一种手段。但这样成功的概率并不大，因为有恶意的节点本来算力占比就不高，还要生成更多的区块，就非常困难。以上是selfish mining的其中一个目的，它还有另一个目的。假如A挖了两个区块都没有发布，而在B挖到一个区块公布后立马公布，这样B挖的区块就作废了。这样的好处就是减少竞争，因为A在挖第二个区块时，别人还在挖第一个区块(前提是A算力足够强)。但这样也有不好的地方，假如A挖出一个区块，A以为他能赶在别人面前再挖一个区块，结果这时有人挖出了第一个区块，那这样的话A就要在别人发布之后立马发布，去争取区块奖励。

## 2.5比特币的网络

比特币工作在应用层(application layer:Bitcoin block chain)，它的底层是一个网络层(network layer:P2P overlay network)。比特币的P2P网络是非常简单的，所有节点都是对等的。不像有的P2P网络有所谓的超级节点、主节点。要加入P2P网络首先得知道至少有一个种子节点，然后你要跟种子节点联系，它会告诉你它所知道的网络中的其他节点，节点之间是通过TCP通信的，这样有利于穿透防火墙。当你要离开时不需要做任何操作，不用通知其他节点，退出应用程序就行了。别的节点没有听到你的信息，过一段时间之后就会把你删掉。

比特币网络的设计原则是:simple、robust，but not efficient。每个节点维护一个零度节点的集合，消息传播在网络中采取flooding的方式。节点第一次听到某个消息的时候，把它传播给去他所有的邻居节点，同时记录一下这个消息我已经收到过了。下次再收到这个消息的时候，就不用转发给邻居节点了。

零度节点的选取是随机的，没有考虑底层的拓扑结构。比如一个在加利福尼亚的节点，它选的零度节点可能是在阿根廷的。这样设计的好处是增强鲁棒性，它没有考虑底层的拓扑结构，但是牺牲的是效率，你向身边的人转账和向美国的人转账速度是差不多的。比特币系统中，每个节点要维护一个等待上链的交易的集合。假如一个集合的交易都是等待写入区块链里的，那么第一次听到某个交易的时候，把这个交易加入这个集合，并且转发这个交易给节点，以后再收到这个交易就不用转发了，这样避免交易会在网络上无线的传播下去。转发的前提是该交易是合法的。

这里有冲突的情况，有可能你会有两个有冲突的交易，差不多同时被广播到网络上。比如说A→B和A→C，这两个如果同时广播在网络上，那么每个节点根据在网络中的位置的不同，收到两个交易的先后顺序不同。比如一个人先收到第一个交易，就写入到集合里，再收到第二个交易的时候就不会写入集合，因为跟上一个交易有冲突，就认定是非法的。假设这两个交易花的是同一个币，那么写入集合的交易就会被删掉。

比如说节点听到一个新发布的区块，里面包含了A→B的交易，那么这个交易就可以删掉了，因为已经写入到了区块链里。如果节点又听到了A→C的交易，该怎么办?这时候也要把A→B删掉。因为A→C如果已经被写入到了区块里，那么A→B就变成了非法交易，就变成了double spending，这就是冲突的情况。可能某个先收到A→C的节点，抢先挖到了矿，发布了区块。新发布的区块在网络上的传播有很多方式，跟新发布的交易是类似的。每个节点除了要检查区块的内容合法性之外，还要查它是不是在最长合法链里。越是大的区块，在网络上传播速度越慢。

比特币协议对区块的大小有1M字节的限制。比特币系统采用的传播方式是非常耗费带宽的，带宽是瓶颈。按1M的区块大小限制来算的话，一个新发布的区块有可能需要几十秒，才能传输到网络大部分节点，这已经是挺长时间了，所以这个限制值不算小。

还需要注意的一点:我们讲的比特币网络的传播属于**best effort** 。一个交易发布到比特币网络上，不一定所以的节点都能收到，而且不同的节点收到这个交易的顺序也不一定是一样的。网络传播存在延迟，而且这个延迟有的时候可能会很长，有的节点也不一定按照比特币协议的要求进行转发。

可能有的该转发的不转发，导致某些合法的交易收不到，也有的节点可能转发一些不该转发的消息，比如说有些不合法的交易也被转发了。这就是我们面临的一个实际问题。

## 2.6比特币的挖矿难度

谈到挖矿难度我们就复习一下挖矿，就是尝试nonce值，使Block header组成的hash值<=给定的目标域值，即**H(block\_header)<=target** 。所以target值越少，挖矿难度越大，调整挖矿的难度就是调整目标空间在整个输出空间中所占的比例。

比特币用的哈希算法是SHA-256，这个产生的哈希值是256位。所以整个输出空间是2的256次方。调整这个比例，即目标空间占输出空间的比例，通俗的说，就是哈希值前面要有多少个0。比如说256位的哈希值，要是合法的区块，要求算出来的哈希，前面至少有70个0。当然这只是通俗的说法，因为这个目标预值，并不是说前面都是0，从某一个位置开始，后面都变成了1。

挖矿的难度跟目标预值（target）是成反比的，公式是:difficulty=difficulty 1 target / target。上面是指挖矿难度等于1的时候所对应的目标预值，挖矿难度最小就是1，这个时候对应的目标预值是个非常大的数。即target越大，挖矿是越容易的。所以公式里很大的一个数，除以当前的目标预值，得到的就是当前的挖矿难度。所以difficulty和target大小是成反比的。

为什么要调整挖矿难度呢?如果不调会有什么问题呢?系统里的总算力越来越强，挖矿难度保持不变的话，出块时间是越来越短的。

首先我们想想出块时间越来越短，会有什么问题吗?

比如说不到一秒就出一个区块，区块在网络上传播的时间可能需要几十秒，底层的比特币网络可能需要几十秒才能让其他节点都收到。别的节点没有收到这个区块之前还是继续沿着已有的区块链往下扩展。如果有两个节点同时都发布一个区块，这个时候就会出现分叉。出块时间如果越来越短的话，这种分叉会成为常态，而且不仅会出现二分叉，可能会出现很多的分叉。比如10个区块同时被挖出来，系统可能会出现10分叉。分叉如果过多，对于系统达成共识是没有好处的，而且危害了系统的安全性。

比特币协议是假设大部分算力掌握在诚实的矿工手里。系统当中的总算力越强，安全性就越好，因为有恶意的节点想掌控51%的算力就越难。如果掌握了51%的算力，它就可以干很多坏事，比如分叉攻击。如果后面分叉多的话，前面某个区块里的某个交易，很可能就遭受分叉攻击，恶意节点会试图回滚。因为后面分叉多，算力就会分散，恶意节点得逞的概率更大。这个时候恶意节点就不需要51%的算力了，可能10%的算力就够了，因此出块时间不是越短越好，如图2-10。

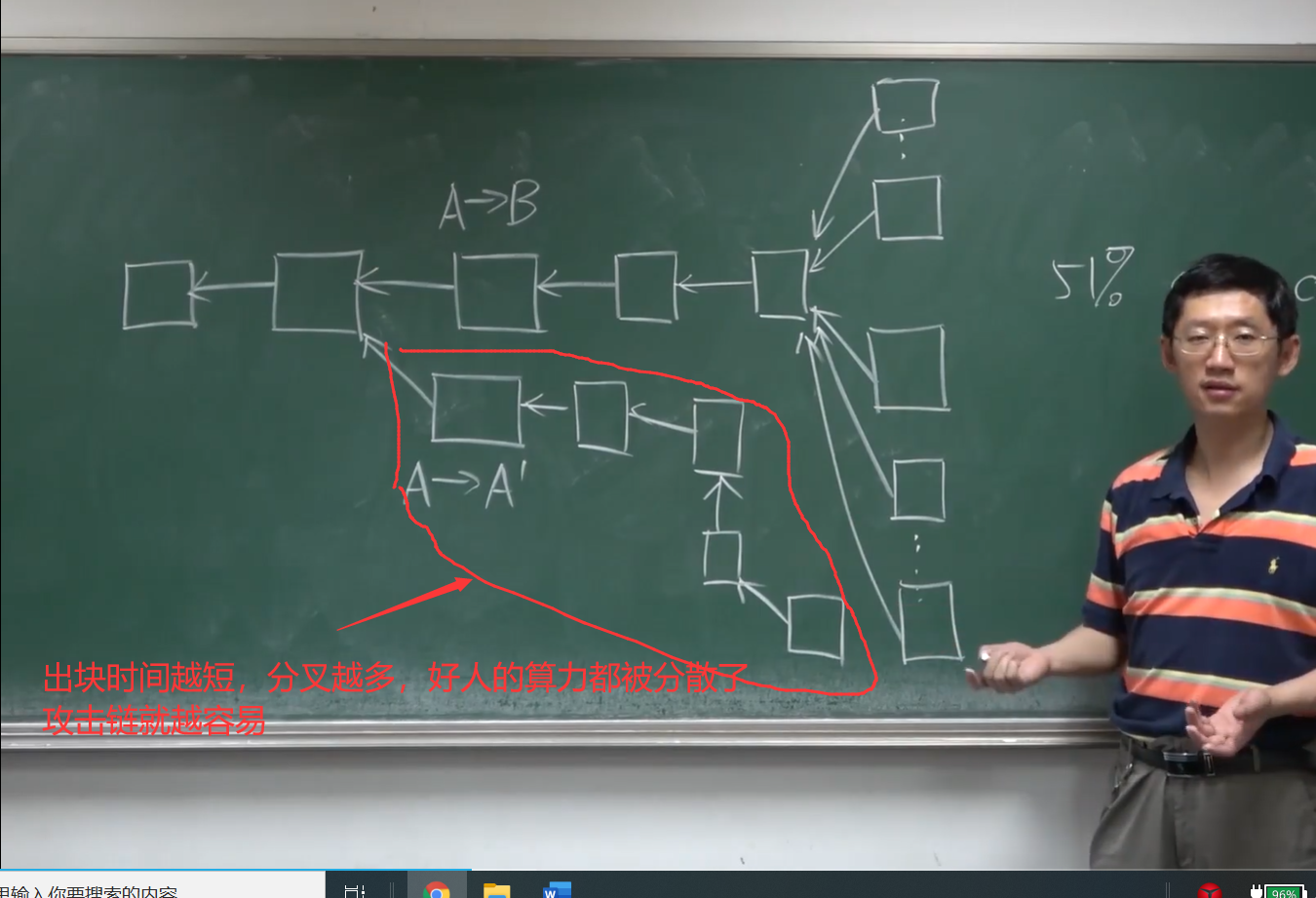


图2-10 出块时间的太短的情况

那10分钟的出块时间是不是最优的呢?不一定。改成其他值也可以，有间隔只是说应该有个常数范围。以太坊系统出块时间就降低到了15s，所以以太坊的出块速度是比特币的40倍。出块时间大幅度下降之后，以太坊就要设计新的协议，叫ghost。在该协议中，这些分叉，产生的orphan block(即产生最长合法链后另一个要被丢弃的区块)就不能丢弃掉了，而是也要给它们一些奖励，这叫uncle reward。以太坊也要调整挖矿难度，使出块时间保持在15s。

讨论完了为什么要调整挖矿难度，那怎样调整挖矿难度呢？比特币协议中规定，每2016个区块后就要调整目标预值，这大概是每两个星期调整一次，如图2-11。具体的调整公式:target =target×(actual time/expected time)。actual time指产生2016个区块实际花费的时间，expected time指产生2016个区块应用的时间，即2016×10min。如果实际花费时间超过了两周，即平均出块时间超过了10min。那么这时候挖矿难度要调的低一点，应该让出块更容易。因此该公式算出来的target会变大，则难度会下降。

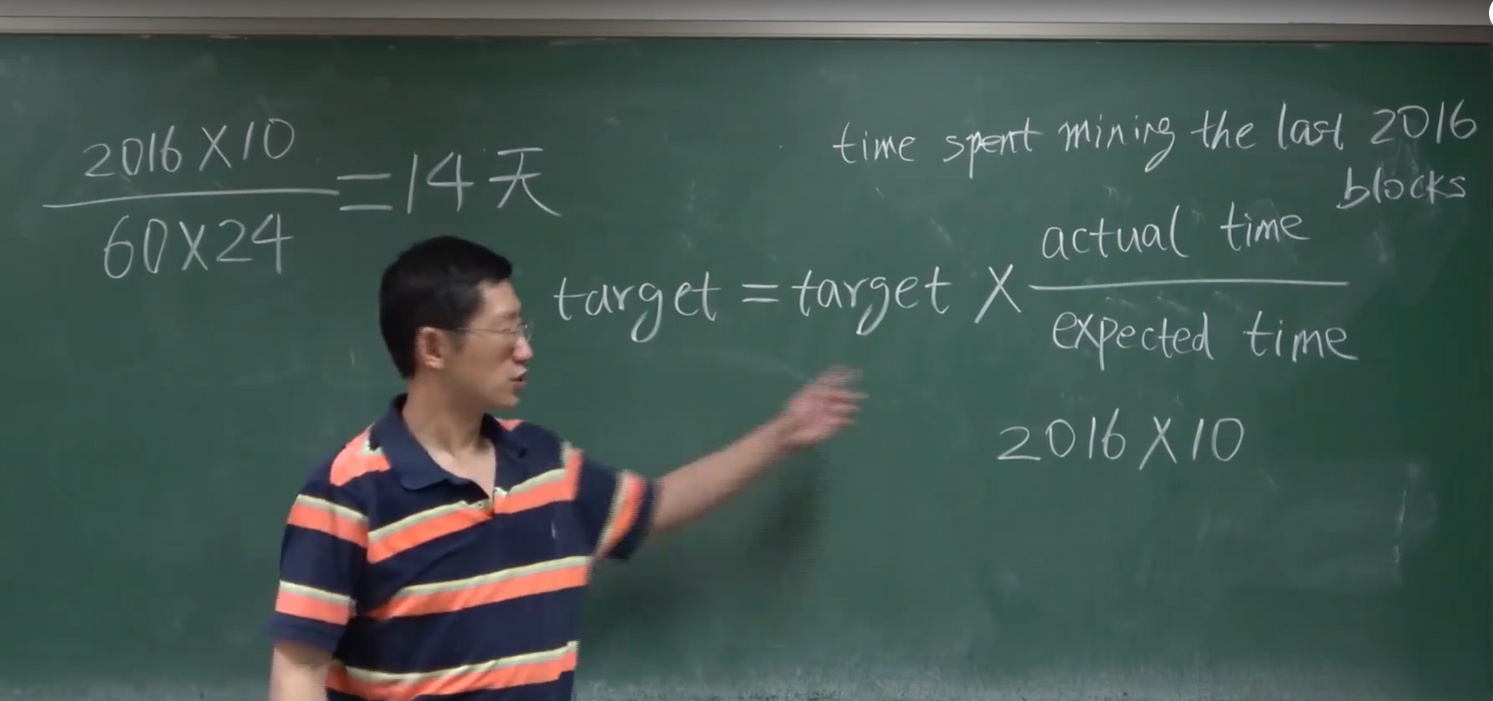


图2-11具体的调整公式

实际上，上调和下调都有四倍的限制。假如实际时间超过了8个星期，那么我们计算公式时也只能按4倍算，目标预值增大最多只能增大4倍。

那怎么才能让所有的矿工同时调整目标预值呢?计算target的方法写在比特币系统的代码里，每挖到2016个区块会自动进行调整。如果有有恶意的节点故意不调，会怎么样?

如果一个节点不调，将区块发布出去，诚实的节点是不会认的。**nBits**是target一个编码的版本，在block header里没有直接存储target的域，因为target的域是256位，直接存target的话要32个字节。nBits在header里只有四个字节，所以可以认为是它的一个压缩编码。如果遇到有恶意的矿工，该调的时候不调，这时检查区块的合法性就通不过。因为每个节点要独立的验证发布的区块的合法性。检查的内容就包括:nBits，目标预值设的对不对。如果投机取巧设计一个过大的目标预值，使得你自己挖矿容易了，但这个区块是不会被接受的。

## 2.7比特币的挖矿

### 2.7.1 比特币的节点

由于硬件限制，比特币的系统中分为全节点和轻节点，

| **全节点** | **轻节点** |
| --- | --- |
| 一直在线 | 不是一直在线 |
| 在本地硬盘上维护完整区块链信息 | 不保存整个区块链，只需要保存每隔区块块头 |
| 在内存中维护UTXO集合，以便于快速检验交易合法性 | 不保存全部交易，只保存和自己有关的交易 |
| 监听比特币网络中交易内容，验证每个交易合法性 | 无法验证大多数交易合法性，只能检验和自己相关的交易合法性 |
| 决定哪些交易会打包到区块中 | 无法检测网上发布的区块正确性 |
| 监听其他矿工挖出的区块，验证其合法性 | 可以验证挖矿难度 |
| 挖矿： 1. 决定沿着哪条链挖下去。 2. 当出现等长分叉，选择哪一个分叉 | 只能检测哪个是最长链，不知道哪个是最长合法链 |

在比特币网络中，大多数节点都是轻节点。如果只是想进行转账操作，不需要挖矿，就无需运行一个全节点。在挖矿过程中，如果监听到别人已经挖出区块延伸了最长合法链，此时应该立刻放弃当前区块，在本地重新组装一个指向最后这个新合法区块的候选区块，重新开始挖矿。这时可能有人就提出，这样的方式是不是有些可惜，之前花费好多资源，全部白挖了。实际上并不可惜。之前文章中提及，挖矿本身具有无记忆性，前面无论挖多久，对后续继续挖矿没有影响。

### 2.7.2 比特币的挖矿设备和挖矿趋势

目前，挖矿设备逐渐趋于专业化，其经历了三个过程，总体趋势从通用到越来越专用。普通CPU -> GPU ->ASIC芯片（挖矿专用矿机）

挖矿机的变化趋势，是从通用变得越来越专用，CPU是通用计算，GPU是通用并行计算，ASIC是专用计算。很多人觉得这是不好的，是跟去中心化的理念是不相符的，也违背了比特币设计的初衷。最民主的情况是，大家都用家里的CPU计算机挖矿。而有些新的加密货币设计的是Alternative mining puzzle。而设计它的出发点是asic resistance(抗asic芯片化)，目的是让通用的计算机也能参与挖矿的过程。

挖矿的另一个趋势是大型矿池的出现，单个矿工即使用了ASIC芯片，挖矿从平均收益上看是有利可图的，但是收入是非常不稳定的。比特币系统中平均每10分钟出一个区块，这是说比特币系统中所有的矿工做一个整体来看平均10min会产生一个区块。但如果具体到某一个矿工来说，他可能要挖很长时间，如果他用一个矿机可能要挖一两年。而且单矿工还有其他问题，他除了挖矿之外还要承担全节点的其他责任。

因此，为了解决这些问题，便引入了**矿池**的概念。矿池的架构如下图2-12，通常是一个全节点驱动多台矿机。矿工只需要不停计算哈希值，而全节点其他职责由矿主来承担。ASIC芯片只能计算哈希值，不能实现全节点其他功能。此外，矿池出现解决了单个矿工收益不稳定的问题。当获得收益后，所有矿工对收益进行分配，从而保证了收益的稳定性。

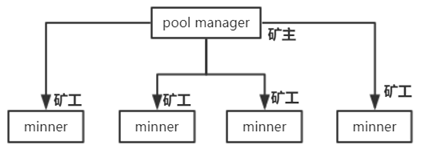


图2-12 矿池的架构

矿池一般具有两种组织形式。

1.类似大型数据中心（同一机构），有的互联网公司，有成千上万个服务器，大的矿池里面也有成千上万的矿机，这些矿机如果是属于同一个机构的话，那么收入怎么分配就不重要了。

2.分布式。矿工与矿主不认识(不同机构)，矿工与矿主联系，自愿加入其矿池，矿主分配任务，矿工进行计算，获得收益后整个矿池中所有矿工进行利益分配。

如果矿工是来自五湖四海的，不是属于同一个机构的，那么利益该怎么分配? 平均分配行不行?比如每个矿工挖到一个区块，得到了出块奖励，然后平分给其他矿工，这样行吗?不行，因为会有矿工偷懒。因此要按矿工的贡献大小进行分配，也就是这里同样需要工作量证明。那该怎么证明每个矿工做了多少工作呢？

我们先考虑一下为什么之前矿工的收入不稳定，因为挖矿太难了，如果把挖矿的难度降低之后，挖矿就会变得稳定了。怎么降低难度呢？以前的要求是，矿工要找到一个nonce，用nonce计算block header 的哈希值，前面至少有70个0才是合法的区块。降低挖矿难度之后，比如说前面只要有60个0就行了，这样挖到的叫作一个share，这个share叫做almost valid block。矿工挖到share或almost valid block之后，把它提交给矿主。矿主拿到这个区块有什么用呢？用来证明矿工所做的工作量，而没有其他用途。矿主无法得到区块奖励以及任何好处。所以矿主就统计每个矿工提交了多少这样的share，将来等到某个矿工真正挖到了合法的区块之后， 再将出块奖励按照每个矿工所做的工作量，提交的share数目进行分配。

我们想一想：这样做为什么是可行的？因为每个矿工挖到矿的概率取决于他尝试的nonce数目，尝试的nonce越多，能找到的share就越多，这些就可以作为他的工作量证明，尝试的越多，说明你干的越多，你的收入自然就多了。

有没有可能一个矿工挖到一个合法的区块之后，不把它提交给矿主，而是自己偷偷摸摸发布出去，得到出块奖励?即平时挖到的share提交，但挖到了合法区块就不提交？不可能，因为每个矿工的任务是由矿主分配的，矿主负责组装好一个区块，然后交给矿工去尝试各种nonce，而且挖矿仅仅调nonce是不够的，还需要调整coinbase parameter。所以矿主会把不同的coinbase parameter所对应的nonce值的范围交给不同的矿工去尝试。那么这个区块里包含什么?coinbase transaction里面有收款人的地址，这个地址填的是矿主的地址，即pool manager的地址，所以矿工挖到区块之后，如果他不提交给矿主自己发不出去是没有用的。里面的收款地址是矿主的，他取不出钱来。所以只要是当初按矿主给分配的任务进行挖矿的，就不可能偷区块奖励。

如果他一开始就不管矿主的任务，自己组装一个区块，偷偷把收款地址改成自己地址，会怎样?那样他提交share给矿主的话，矿主是不认的，因为里面交易列表被改过了，coinbase transaction里面的内容发生了变化，算出的merkle tree 的根哈希值也是不一样的。这种情况下矿主是不会给他工作量证明的。那就相当于矿工一开始就单干，跟矿池是没关系的。

虽然不可能偷区块奖励，但会不会有人捣乱，比如平时挖到一个share，提交给矿主，作为工作量证明。等他挖到一个真正合法的区块之后，把它扔掉。这是有可能的，虽然没有经济好处，但有可能是别的矿池派来的卧底，不想让这个矿池得到区块奖励。这些矿工还是会分红，分的是别的矿工挖出来的区块奖励。

### 2.7.3矿池的弊端

一般来说，矿池的矿主要收取一定比例的出块奖励作为管理费。矿主也要按照比例收取管理费，有的是按照出块奖励的比例，也有的是抽取交易费。有的一些有恶意的矿池在发动攻击之前，可能故意把管理费降得特别低，甚至是赔本赚吆喝，吸引足够多的矿工加入之后就可以发动攻击了。这是大型矿池的一个弊端，使得51%的攻击更加容易了。

假如某个矿池占到了半数以上的算力，他具体能够发动哪些攻击呢?一个最常见的就是分岔攻击。假如一个区块链，其中一个区块包含了一个大笔的交易，又等了几个确认区块之后，自认为已经安全了。然后这时就可能有人在该交易前面的区块发动分岔攻击。

如图2-13，看上去好像追赶的道路是很漫长的，但如果拥有51%的算力，最终还是可以成功攻击。另外，不要把51%当成绝对的门槛，有可能不到51%就可以。算力都是估计的，而且算力还在不断变化。

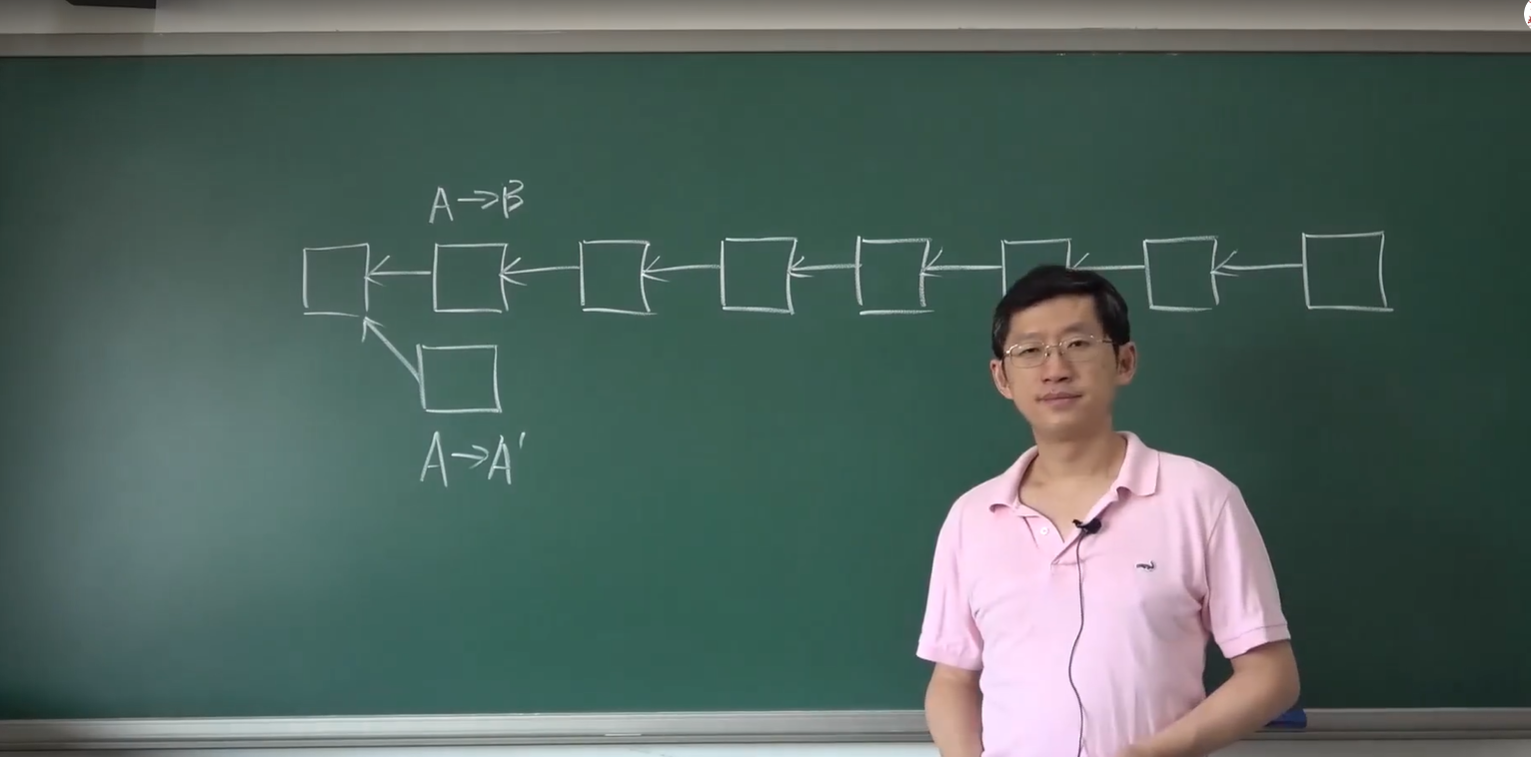


图2-13 分叉攻击

攻击者还能做什么坏事?还可以做**boycott(封锁境域)。**比如说攻击者不喜欢某个账户，怀疑某个账户参与非法交易，想把这个账户封锁掉，所有跟这个账户相关的交易都不让上链。假如图A把某个交易A→B发布到区块链上，攻击者就会马上进行分岔，产生一个不包含这个交易的区块，所有跟A有关的交易也都不包含进去。

这种攻击跟分岔攻击区别是什么?他没必要等后面几个确认区块。这时候如果攻击者等待确认区块，是为了让B放心，B以为后面有六个确认区块，已经没事了，然后攻击者再发动分岔攻击。而如果目的是为了boycott的话，就没有必要等后面区块生成。A→B交易一上链马上进行分岔，越早越好，因为攻击者是希望别人沿着他的链往下挖的。前面讲过，有些有恶意的节点故意不把某些交易写入区块里，是可以的。但没有关系，后面的区块还是会包含的。但是如果这个坏人拥有51%的算力的话，他可能仗着自己算力强，公开抵制他想抵制的交易。这样别的矿工也不敢随便把交易打包进去了。

总的来说：矿池的出现减轻了矿工的负担，矿工只需要挖矿，计算哈希值就行了，别的事情都由矿主来完成。矿工的收入分配也更加稳定。但矿池的出现也有危害，发动51%的攻击变得容易了。他不一定自己有这么强的算力，只要动员召集这些算力就可以了。

## 2.8比特币脚本（The BitCoin Scripting Language）

### 2.8.1.交易实例



图2-14 比特币交易实例的脚本

比特币系统中使用的脚本语言非常简单，唯一可以访问的内存空间只有栈，所以也被称为“基于栈的语言”

1.交易的宏观信息：

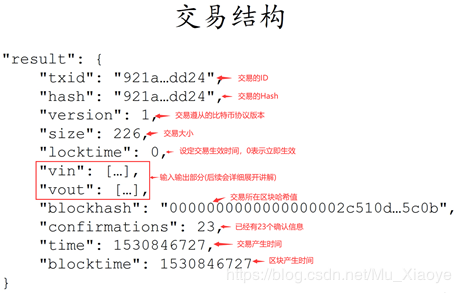


图2-15 比特币交易的结构

2.Vin的内容：

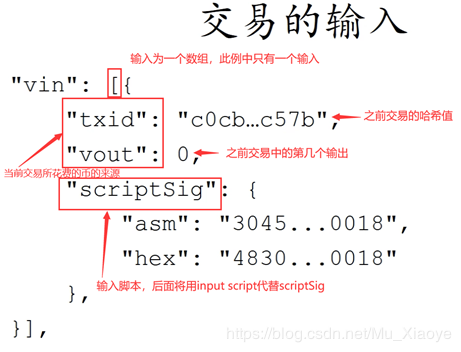


图2-16 比特币交易的输入

如果存在 一个交易有多个输入，那么每个输入都要说明币的来源并给出签名（BTC中一个交易可能需要多个签名）

3. Vout的内容：



图2-17 比特币交易的输出

### 2.8.2. 输入输出脚本的执行

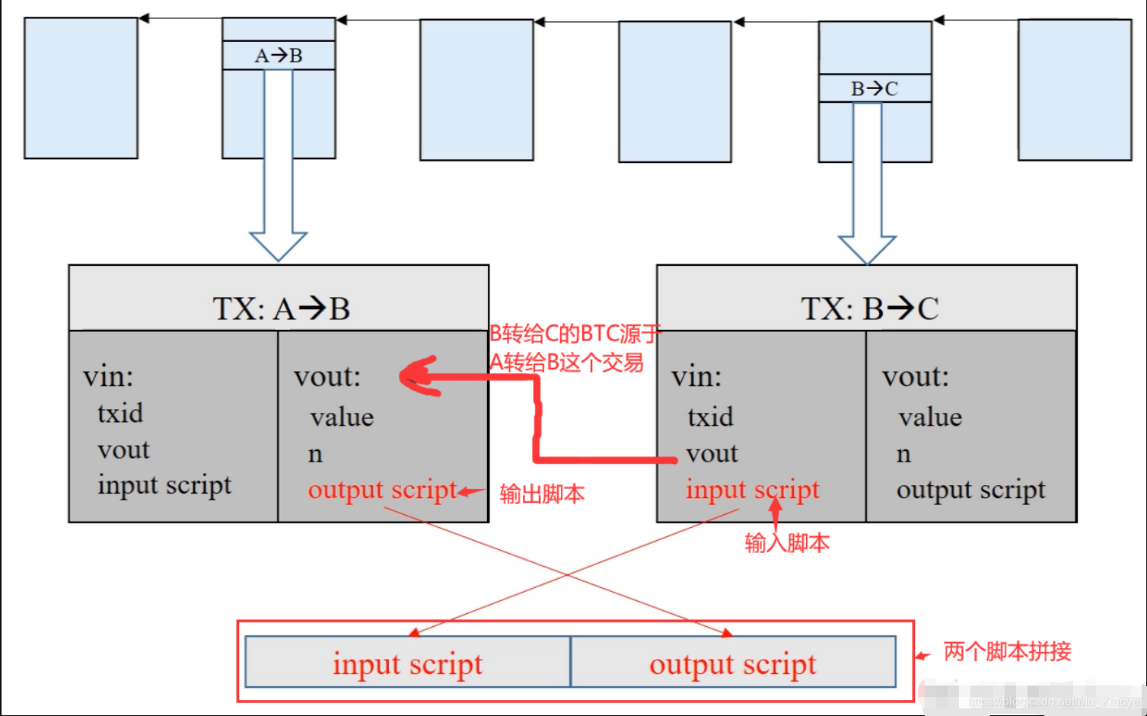


图2-18 输入输出脚本的执行

如图2-18所示，为脚本执行流程。在早期，直接将两个脚本按照如图顺序(input script在前，output script在后) 拼接后执行，后来考虑到安全性问题，两个脚本改为分别执行：先执行input script，若无出错，再执行output script。如果脚本可以顺利执行，最终栈顶结果为true，则验证通过，交易合法；如果执行过程中出现任何错误，则交易非法。如果一个交易有多个输入脚本，则每个输入脚本都要和对应的输出脚本匹配执行，全部验证通过才能说明该交易合法。

### 2.8.3 输入输出脚本的几种形式

1.P2PK形式(Pay to public key)，是一种最简单的形式。特点是：输出脚本直接给出收款人公钥，CHECKSIG为检查签名操作。

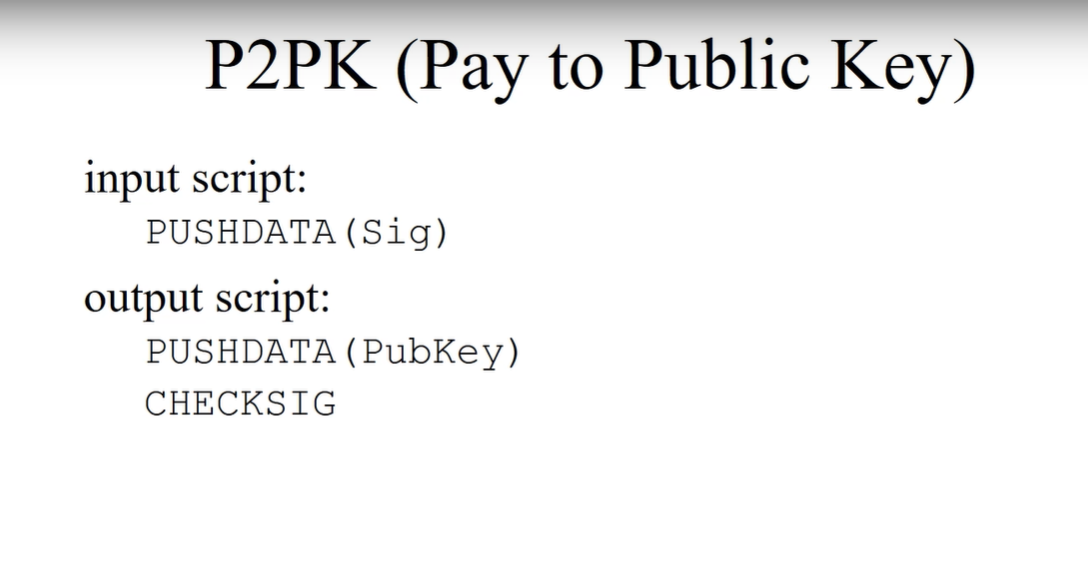


图2-19 P2PK形式

执行过程(将两个脚本拼接起来)：

(实际执行已经不再拼接两个脚本,而是分别执行)

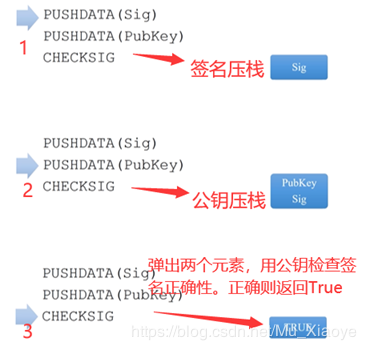


图2-19 P2PK执行过程

P2PK实例：



图2-19 P2PK执行实例

2. P2PKH形式(Pay to public key hash)——最常用的执行脚本，其特点是：输出脚本不直接给出收款人的公钥，而是公钥的哈希。

# 3.本周总结

本周主要还是进行完善比特币方面的知识，下周可以把比特币方面统计完成，并开始以太坊方面的知识，本周的内容有很多都是来自公开课讲的内容，我把其中的知识点和一些公开课笔记联系整合起来，形成我要记得内容，以便于我以后复习进行回顾，同时也可以加深我听课的知识。争取争取下周完成，下周主要以看以太坊为主，如果有时间也可以看看rust语言，当然以后英语学习也不能落下。

# 附录

**1.北京大学肖臻老师《区块链技术与应用》公开课(来自于B站):** <https://www.bilibili.com/video/BV1Vt411X7JF?p=1>

**2. 北京大学肖臻老师《区块链技术与应用》公开课系列笔记**

<https://blog.csdn.net/Mu_Xiaoye/article/details/104299664>

**3. 个人的github的开源库：** <https://github.com/Hello-Afa/Graduate_career.git>