# 比特币系统的实现

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B5个比特币，B花掉了。A也给了C3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。

要UTXO集合有什么作用? 为了检测double spending。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。

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。

21万个区块大概要挖多长时间呢?大约是4年。比特币系统设计的平均出块时间是10分钟，就是整个系统平均10分钟会产生一个新的区块。

除了比特币这种基于交易的模式，与之对应的还有基于账户的模式(account-based ledger)，比如以太坊系统。在这种模式中，系统是要显示的记录每个账户上有多少币。

比特币基于交易的模式，隐私保护性较好。缺点是比特币当中的转账交易要说明币的来源，而基于账户的模式就不用。

如图⑥(第五节视频 16分钟处) 一个区块的例子
第一行表明:该区块包含了686个交易 第二行:总输出XXX个比特币 第四行:总交易费(686个交易的交易费之和) 最下面一行:区块奖励(矿工挖矿的主要动力) 第五行:区块的序号 第六行:区块的时间戳 第九行:挖矿的难度(每隔2016个区块要调整挖矿的难度，保持出块时间在10分钟左右) 倒数第二行:挖矿时尝试的随机数

右边:第一行:该区块块头的哈希值 第二行:前一个区块块头的哈希值 (注意:计算哈希值只算块头) 两个哈希值的共同点:前面都有一串0。是因为，设置的目标预值，表示成16进制，就是前面一长串的0。所以凡是符合难度

要求的区块，块头的哈希值算出来都是要有一长串的0。 第四行:merkle root 是该区块中包含的那些交易构成的merkle tree的根哈希值。

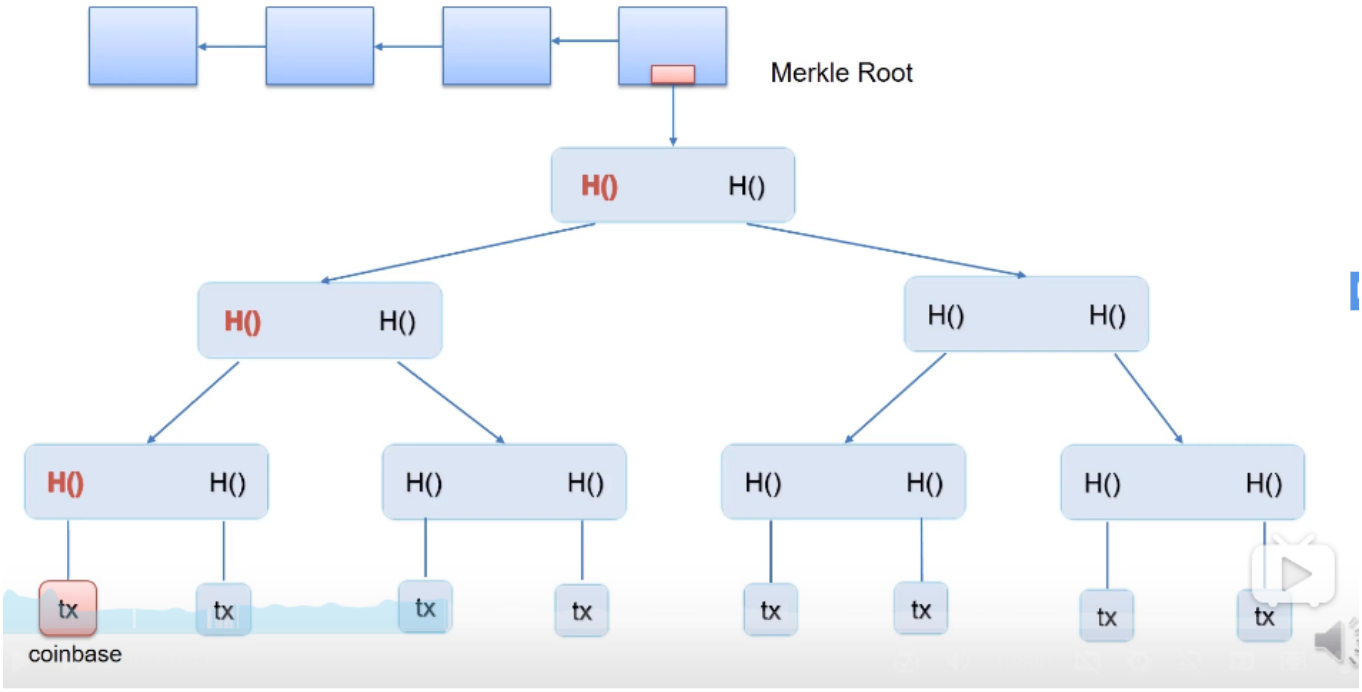
如图⑥(见第五节视频 第20分钟)块头的数据结构 最后一行:是32位的无符号整数。nonce只有2的32次方个可能的取值。按照比特币现在的挖矿情况来说，很可能把2的32次方个取值都验了一遍也找不到合适的。那怎么办呢?block header 的数据结构里还有哪些域是可以调整的呢?

如图⑦ 块头里各个域的描述(见第五个视频 第21分钟) 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

挖矿时只改随机数不够，还可以更改根哈希值。

如图⑧(见第五节视频 第23分钟) 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。

那这个域对我们有什么用呢?

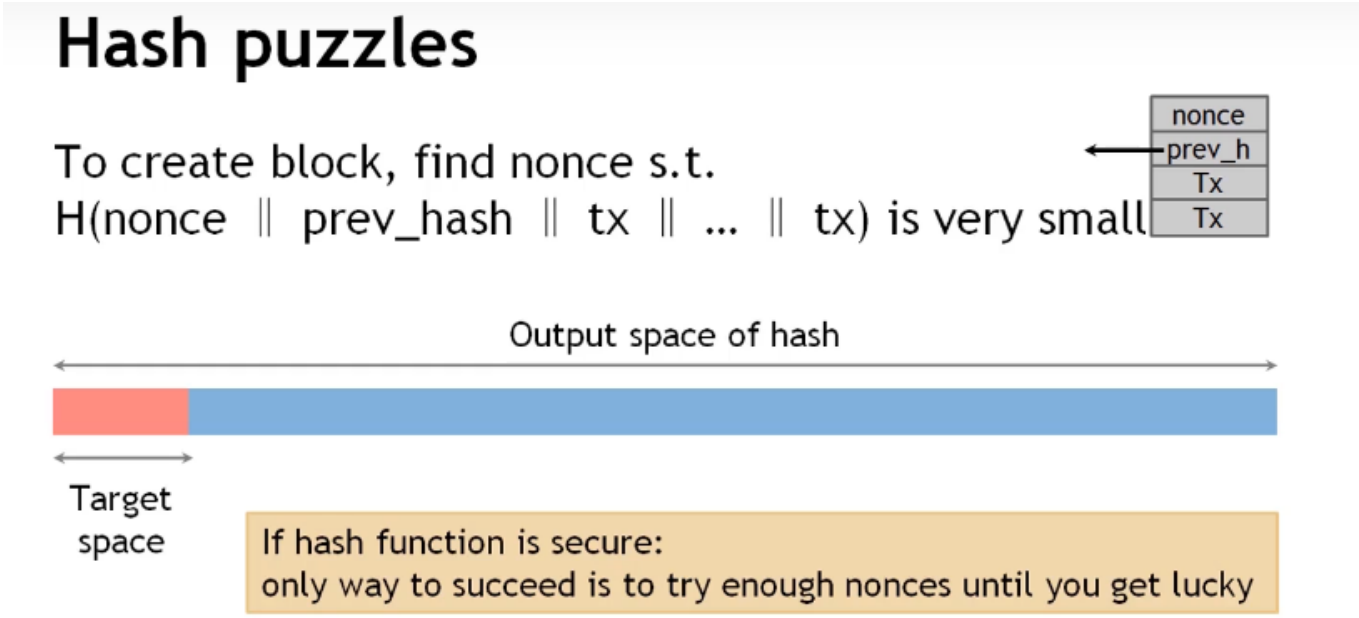



如图⑨(见第五节视频 第24分钟) 对应的是最后一个block header里的根哈希值对应的merkle tree，左下角的交易是coinbase，把它的域改了之后，其上的哈希值就发生了变化，然后沿着merkle tree的结构往上传递。最后导致block header里的根哈希值发生变化(merkle root是block header的一部分)。**块头里4个字节的nonce不够用，还有其他字节可以用，比如coinbase域的前八个字节当做extra nonce来用，这样子搜索空间就增大到了2的96次方。**

**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。**

如图⑩ 普通的转账交易的例子(见第五节视频 第26分钟) 该交易有两个输入和两个输出。 左上角:这里的output其实是输入，指的是之前交易的output。 右上角:这里的output都是unspent，都没有被花掉，会保存在UTXO里面。 右边表格第一行:输入的总金额。 依次往下:输出总金额、两者之间的差值。 两表格下面:可以看出输入和输出都是用脚本的形式来指定的。

比特币系统中验证交易的合法性，就是把input scripts和output script配对后执行来完成的。注意:不是把图中的input scripts 和output scripts配对，因为这两个脚本是一个交易中的脚本。不是把同一个交易里的输入脚本和输出脚本配对，而是把这里的输入脚本和前面提供币来源的交易的输出脚本配对。如果输入输出脚本拼接在一起，能顺利执行不出现错误，那么该交易就是合法的。




如图十一，是在求解puzzle的过程。 注意:求哈希时只用到了block header的内容，而交易的具体信息在block header里面是没有的。block header里面只有merkle tree 的根哈希值，这个就已经能保证交易是没有被篡改的。

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。

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。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)。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。

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:progress free。

如果没有progress 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progress 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。

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?

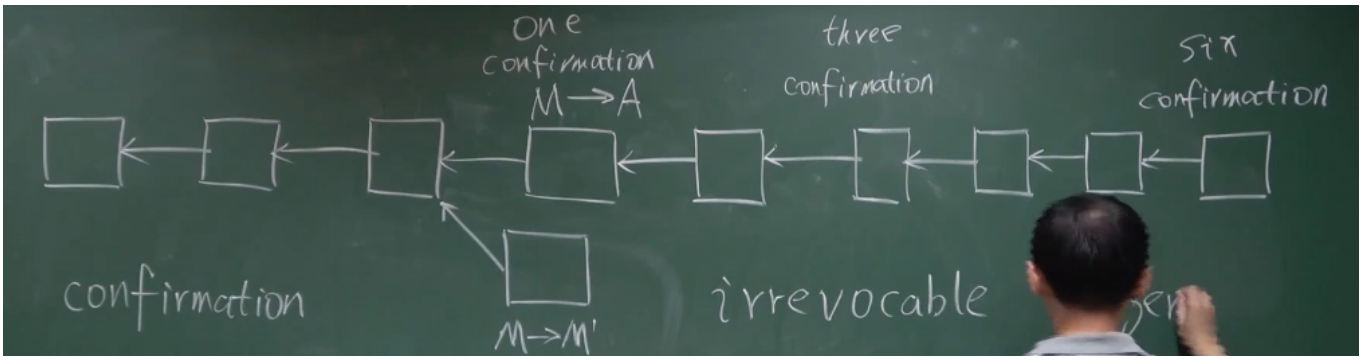
先考虑第一个问题:他能不能偷币?能不能把别人账上的钱转给自己?不能，因为他没有办法伪造别人的签名。

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'。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。




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。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

**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)。

其实还有一种，叫**零确认**(其具体位置可见第五节视频 第62分第26秒)。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。

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。

# 零确认

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，这是分岔工具的一种手段。

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。