《Bitcoin: A Peer-to-Peer Electronic Cash System》读书报告

0. 前言

阅读区块链经典论文《Bitcoin: A Peer-to-Peer Electronic Cash System》，报告内容主要以提炼原文关键内容，外加部分知识的扩展与自己的理解组成。

1. 论文解读

1.0. 摘要

> A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

摘要开头指出，比特币是一种**完全的点对点(P2P)电子货币，可直接从一方交易到另一方，不需要任何中间金融机构**，此处说明了比特币的第一大特性：**去中心化。**

去中心化的电子货币交易，从字面意思上很好理解，就是把中间商去掉。但实际上，却带来了非常多的问题：

1. 谁来发币？
2. 如何证明你的币是真的？或者，如何证明币的来源正确？
3. 如何证明交易有效？这个问题又包含两个子问题： 3.1. A要向B支付10个币，如何保证10个币是由A支付的？又如何保证B收到了这10个币？ 3.2. 如何保证A支付给 B 10个币的同时，没有把这10个币支付给另一个人C？

去中心化电子货币的部分问题3.1可用**数字签名**解决，但3.2 **双重支付(double-spending)** 才是主要问题。而论文提出了一种方法，用于解决双重支付的问题，具体解决方法将在后文讲解。

同时，在比特币网络中，只要多数CPU算力掌握在好节点的手上，那系统就是安全的。

摘要最后指出，比特币网络只需要简单的结构，节点可以随时离开或加入网络。

1.1. 简介

简介中指出了当下互联网交易的一些缺点：

- 完全依赖于可信任的第三方机构
- 交易可以撤销

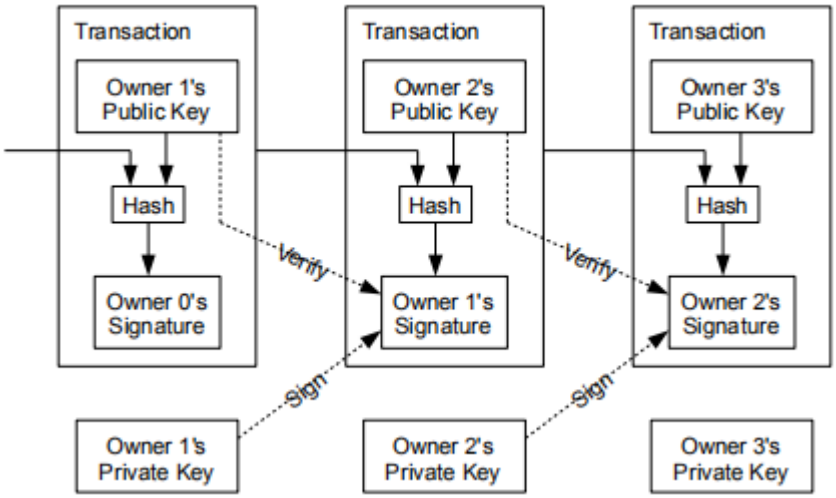
- 杜绝了小额交易
- 由于交易可撤销，商家会要求客户提供本不必要的信息

而比特币系统将具备以下优点：

- 基于密码学原理，不需要可信任的第三方
- 交易不可撤销，将保护商家不被欺诈
- 保护客户的合约机制也比较容易实现

1.2. 交易

> We define an electronic coin as a chain of digital signatures. Each owner transfers the coin to the next by digitally signing a hash of the previous transaction and the public key of the next owner and adding these to the end of the coin. A payee can verify the signatures to verify the chain of ownership.



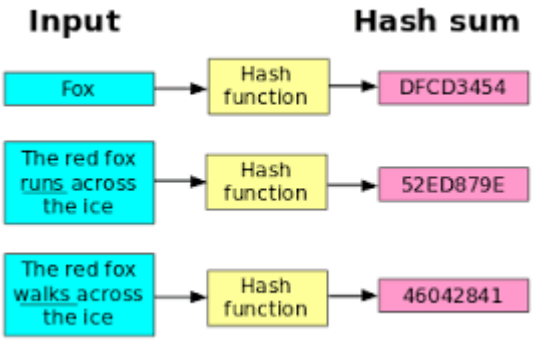
在这一节中，中本聪定义比特币是一串数字签名：每一个拥有者将货币支付给下一个拥有者时，需要将**下一个拥有者的公钥**和**上一笔交易**取**哈希**，并对哈希值进行**数字签名**，最后把数字签名添加到交易末尾。

为了理解这段话，我们需要先了解**哈希**和**非对称加密**。

密码学基础

哈希

哈希，又称散列，英文名`Hash`。是将**任意长度**的输入x通过哈希函数（又称哈希算法）H，得到**固定长度**的哈希值$H(x)$的过程。



优秀的哈希函数具有**不可逆性**，即哈希函数不存在反函数。那么，指定$H(x)$无法高效地推出输入x，唯一的方法是遍历所有输入。

由于哈希函数定义域（输入x的范围）是无限的，而值域（输出y的范围）是有限的，理论上会出现**哈希碰撞**：对于$x \neq y$，存在$H(x) = H(y)$。但是，**优秀的哈希函数，出现哈希碰撞的概率几乎为0**，即：若$x \neq y$，则$H(x) \neq H(y)$。比如，**比特币中用到的哈希函数SHA-256**，输出的哈希值有256位(bit)，相当于有 $2^{256} \approx 10^{77}$ 种不同的结果。对于这样的哈希函数，给定一个输入x，不存在高效的方法找到另一个y使得$H(x) = H(y)$，唯一的方法也是遍历所有输入，但这在现实中基本不可能。

根据上述特性，哈希就有两大功能：

1. 保证数据的完整性，即**防止数据被篡改**。比如，你有一份数据文件想通过互联网发给另一个人，怎么保证在传输过程中数据没有被篡改呢？那就可以事先告诉对方哈希值，对方收到文件后计算哈希值并进行对比，如果不相等则说明被篡改。
2. **数据摘要**。对于任意长度的输入都能得到固定长度的输出，且对于任一个输入，输出是都是唯一的。当你需要一个固定的输入，又不要求跟原文一定相同，只要求存在唯一关联时，就可以使用哈希，比如下文提到的数字签名。

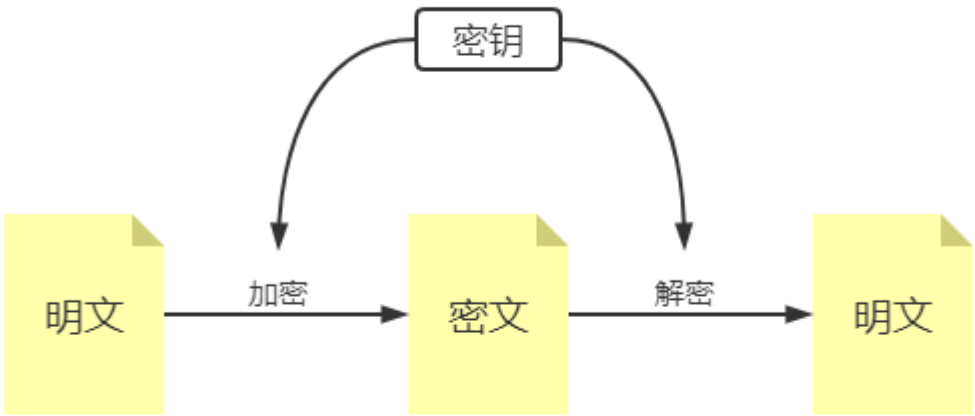
至于哈希函数究竟是如何实现的，此处不做介绍（咱也不会）。

对称加密

了解非对称加密之前，需要先了解对称加密。

众所周知，互联网上的数据传输是不安全的。任意两方在传输数据时，都有可能被恶意的第三方监听。因此，对于机密数据，在传输过程中必须进行加密，这样第三方即便获取也无法知道其中的内容。但是，接收方也要知道如何进行解密，不然他也看不懂。于是，出现了对称加密。

对称加密是一类加密算法，在加密与解密时必须使用同一个**密钥**（密钥可自定义）。



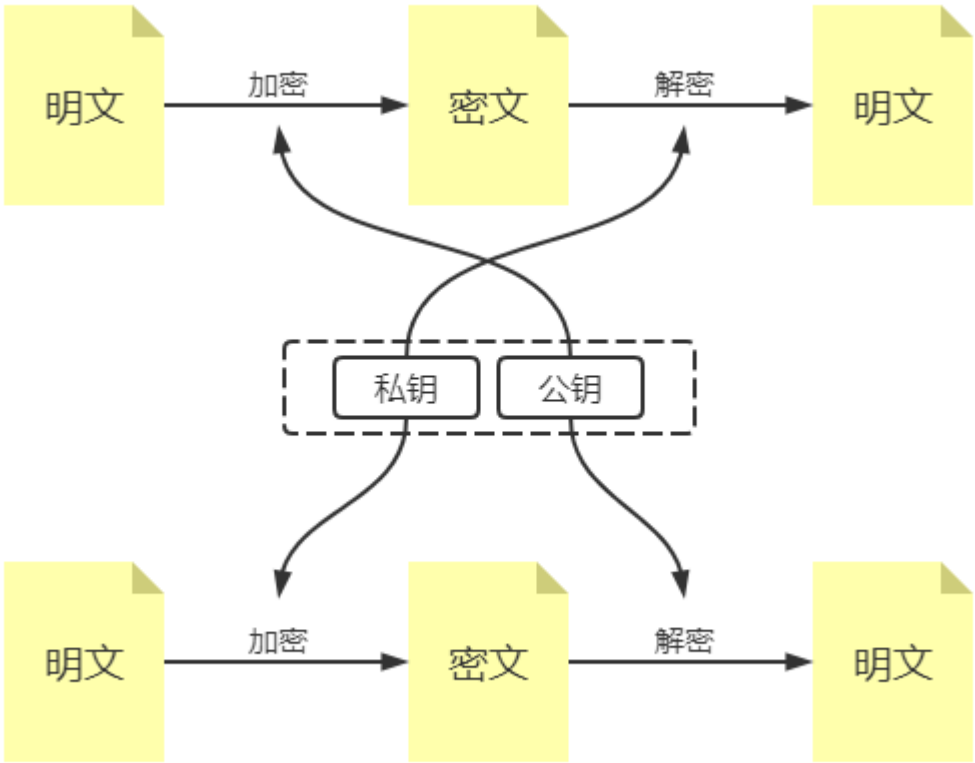
在古代，寄送机密信件时，为了防止信件在途中被截获导致泄密，存在一种方法：将关键文字分散在一篇诗或文章中，事先告诉对方第几行第几个文字拼凑在一起，才是真正的内容，这样即使被截获也能防止机密泄露。这其实就是一种对称加密，将关键文字分散的过程和提取关键文字的过程，就是加密算法中加密解密的过程，而第几行第几个字就是密钥。

在互联网通信中，发送方可事先告知接收方一个密钥K，发送方用该密钥对数据m进行加密$c = K(m)$，接收方再用该密钥进行解密$m = K(c)$。只要密钥不泄露，第三方将无法得知传输中的数据，除非能够破解。当下常见的对称加密算法有**DES**、**AES**等，它们高效且安全，在不知道密钥的情况下几乎无法破解密文。

但是，对称加密的一个大问题是：**如何将密钥告知对方**？若通过互联网发送密钥，则很可能导致密钥泄露，密钥如果泄露，加密的密文将被第三方轻松解密。若能通过线下当面告知，则是最安全的，但这在现实中很不方便。因此，出现了非对称加密。

非对称加密

非对称加密也是一类加密算法，不同于对称加密，非对称加密有两个密钥：**公钥(对外公开)**和**私钥(对外保密)**，**公钥加密的内容需要私钥才能解密，私钥加密的内容需要公钥才能解密，且公钥无法推出私钥。**



互联网上通信时，每个人都可以生成一对唯一的、互不相同的公私钥。假设发送方A的公私钥分别为K_A^+和K_A^-，接收方B的公私钥分别为K_B^+和K_B^-。**A给B发送数据前，需要先询问B的公钥**，B将公钥K_B^+告知A。**发送数据时，A用B的公钥****K_B^+对数据加密$c = K_B^+(m)$。**B收到数据后，用自己的私钥K_B^-对数据解密**$m = K_B^-(c)$。反之，若B要给A发信息，同样需要询问A的公钥，并用A的公钥对数据加密。**

> 你把你家的邮箱放在楼下, 所有的人都可以往里扔邮件, 而只有你能打开邮箱查看邮件。

或许你会问，**公钥在传输过程不一样会被第三方截取吗？是，但没用。**比如，A给B传输数据前，会询问B的公钥，B将他的公钥发送给A的过程中被恶意第三方C截获了。但是，A发给B的数据C依旧不能解开，因为用B公钥加密的数据必须用B私钥解密，而B的私钥并没有泄露给C（假设只有传输过程才会泄露，不考虑电脑被入侵等情况）。而这就是非对称加密的独到之处。

仔细想想，这个过程真的没有漏洞吗？答案是否定的。同样是上述例子，B将他的公钥K_B^+发送给A的过程中被C截获了。此时，C可以将他自己的公钥K_C^+发送给A。由于互联网中验证身份困难，A会以为收到的是B的公钥，其实是C的公钥。当A发送数据时，会用K_C^+加密，这样加密数据就能被C解密。C篡改数据后，再用K_B^+加密发给B。A以为接收方是B，B以为发送方是A，谁都不知道数据已经被篡改了。对于这个问题，可以使用CA证书等方法解决，具体内容不在此阐述。

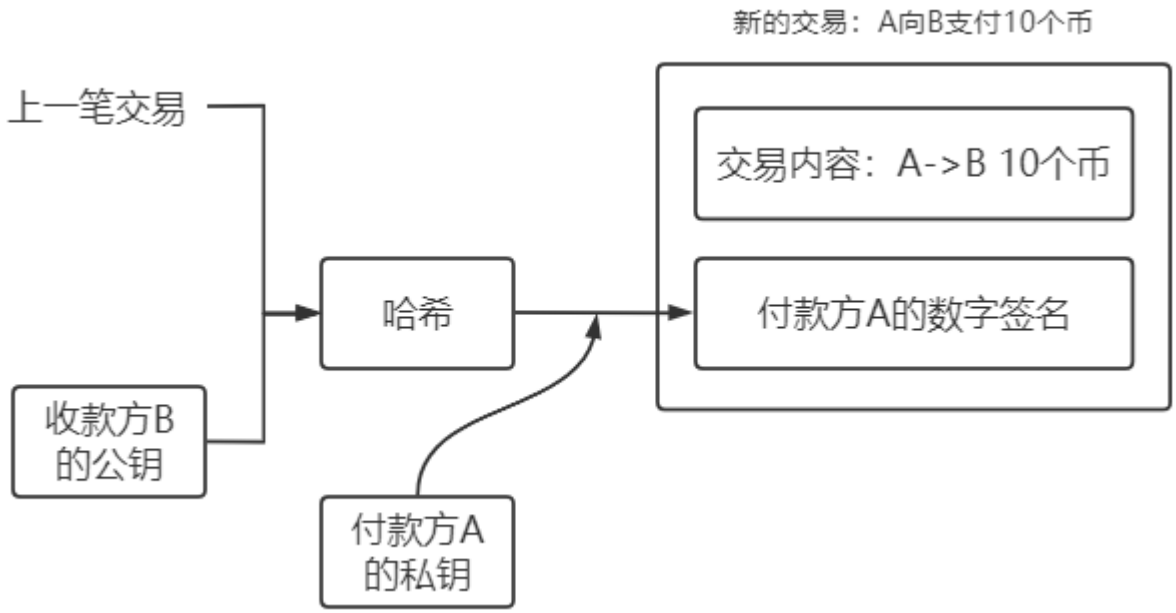
除了用来加密，非对称加密算法还可以用来**数字签名**，对应非对称加密的另一大功能：私钥加密的内容需要公钥才能解密。某个用户A用自己的私钥K_A^-对发布的内容加密，任何人都可以用A的公钥K_A^+解密。由于能用K_A^+解密，则可以证明数据是由K_A^-加密的，而只有A拥有K_A^-，进而可以**证明这个文件或内容是由A发布/签名的。**

如今最著名、使用最广泛的非对称加密算法是**RSA算法**（由发明者Rivest、Shmir和Adleman姓氏首字母缩写而来）。

理解区块链中的交易

在比特币系统中，由于没有中心，每个参与者（节点）都需要记录系统中所有成功的交易。如此，每个参与者在交易时，才能证明对方的钱是有真实来源的。

对于新发生的一笔交易，比特币系统规定：在交易末尾，需要对**收款方公钥和上一笔交易的哈希**进行**数字签名。**



在上图中，有三个重要的点：付款方A的数字签名、收款方B的公钥、上一笔交易。它们的作用分别是：

- **付款方A的数字签名很好理解，是为了证明这笔交易是由A发起的**，即付款方是A。当然，为了所有人都能验证这个数字签名，交易内容中还会给出付款方A的公钥。
- 而**收款方B的公钥是为了证明这笔钱是给B的**，即收款方是B。
- **那为什么需要上一笔交易呢？原因很简单，用来证明A的币的来源有效**，比如：上一笔交易是E支付给A10个币。如果上一笔交易的收款方与当前交易的付款方不相同（对比公钥），或上一笔收款金额不够当前交易的支付金额，则交易无效。这样也能**防止不经别人同意，伪造交易从别人钱包中“偷钱”。**

新交易创建后，需要告诉系统中所有参与者，只有大家都验证交易有效，交易才算成功。

最后一点，为什么要使用哈希呢？这是因为数字签名大多都需要先哈希，这样能保证数字签名的输入是固定长度的，进而保证输出是固定长度。同时，哈希还能防止数据被篡改。

存在的问题

交易规则能保证交易有效，但依旧存在一个问题：**双重支付**——A只有10个币，他同时给B和C各支付10个币，单独验证这两笔交易都没有问题，但这显然是不正确的。

当存在可信的第三方机构（比如银行）时，银行会记录每笔交易的时间，并保存所有的交易记录。A支付的两笔交易总有先后，那么先产生的交易成功，后产生的交易肯定会失败，因为银行会判定A的10个币已经花完。但在去中心化的系统中，没有银行，那应该怎么办呢？更多精彩，请见下文。

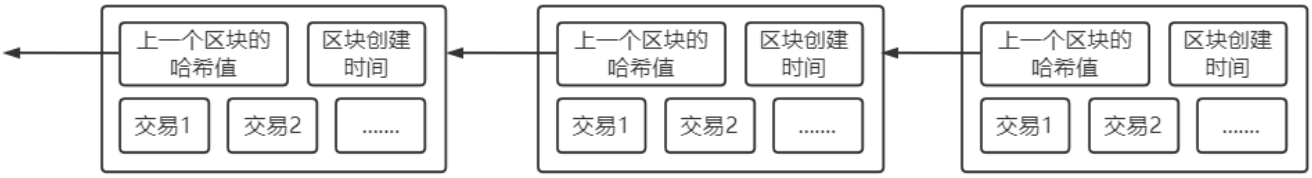
1.3. 时间戳服务器

为了解决双重支付的问题，有两个关键：首先，每个参与者都需要知晓系统中所有的交易，即所有交易必须公开；其次，每个参与者需要对一个有先后顺序、完整的交易序列达成共识。如此，我们就能验证：在新交易之前，这笔钱有没有被花过。

公开交易简单，但如何确实先后顺序呢？解决办法是给每个交易打上时间戳，即记录交易时间。但是在比特币这种P2P网络中，没有中心服务器统一时间，时间由每个节点自定义，不仅可能存在误差，恶意节点还可能乱改时间。

比特币中的解决方案是：

1. 系统中**最新的多笔交易**会被打包成一个区块，区块中还会包含**上一个区块的哈希值**和**区块创建时间戳**（这里的时间戳是unix时间戳只是时间的一种表现形式）。
2. 新区块创建后，需要对新区块计算哈希值，**新区块及其哈希值会被公布在系统中。**
3. 若新区块不满足要求，比如：哈希值不正确、或时间小于上一个区块的时间等，那其它参与者将不会认同这个区块，区块被抛弃，**区块中的交易将无效。**
4. 若新区块被大家认可，**区块中的交易才算成功，新区块的哈希值将会被包含在下一个更新的区块中，依次形成一个链条**，这就是区块链的雏形。



在论文中，把负责创建新区块的节点称为“时间戳服务器”，这个“时间戳服务器”并不是一个中心服务器，比特币中任何节点都可以成为“时间戳服务器”。不过，并不是每一个节点都愿意成为“时间戳服务器”，毕竟谁愿意干苦力活呢？这个问题后文再进行解答。

此处还有一个关键点，**为什么区块要包含上一个区块的哈希值呢**？假设有个恶意节点打算篡改区块链上某个区块的交易数据，区块数据就会发生变化，区块的哈希值也会改变。因此，恶意节点需要重新计算区块哈希值，并修改下一个区块中的prev_block_hash(上一个区块的哈希值)。而这又导致下一个区块的数据及其哈希值发生改变，又要修改下下个区块中的prev_block_hash，一直要修改到区块链的最后一个区块。而后文我们会提到，每一次修改区块数据、重新计算区块哈希值，都将付出昂贵的代价。因此，恶意节点篡改数据的成本将非常高，甚至可能高于他篡改数据所获得的收益。

有了这个区块链雏形，就能保证交易的先后顺序，防止双重支付。但是，新的问题又出现了：由于网络存在延迟，新交易被公布之后，每个节点接收到的时间都不同。同一笔交易，在不同节点中，可能在不同的区块里。而新区块被发布之后，也可能在不同时间到达不同节点。如此，每个节点的区块链都可能不相同，交易序列则会出现混乱，那**大家怎么对一条正确的区块链达成共识呢?**

1.4. 工作量证明

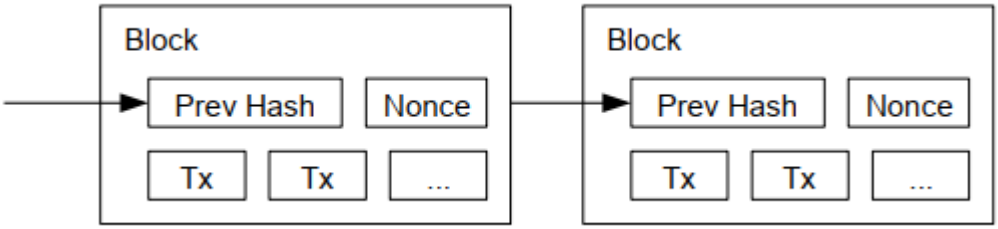
中心化系统相当于权力集中制，中心者负责系统中的决策，中心者所保存的交易序列肯定是被大家都认可的，所有交易也必须通过中心者的检验与确定。至于比特币这样的P2P（点对点）模式，则是民主制，需要大家共同决策并达成一致，而规定如何决策并达成一致的被称为**共识算法。**

最简单的共识算法就是一人一票的投票机制，票数最少的服从票数多的。但这个机制会出现恶意者收买大多数，从而获胜的情况。尤其在网络世界中，还可能出现“女巫攻击”——同时伪造多个节点参与投票从而获胜。因此这种简单的共识算法在网络中肯定是不可行的。

在比特币中，中本聪提出了一种出色的共识算法——**工作量证明(PoW, proof-of-work)。**工作量证明通过计算机算力进行投票，拥有最大计算工作量的一方获胜，换句话说：**谁干的活多就听谁的**（“谁”指一群人）。那怎么证明谁干的活多呢？谁的区块链最长谁干的活就最多，而在比特币系统中，最长的链就代表大多数所认同的。但根据“时间戳服务器”中所提到的，区块创建后只要计算哈希值，整个过程并不需要高时间复杂度的计算，那只要用高算力的机器

一会就能超过系统中的最长链。

因此，为了增加区块创建的难度，中本聪向每个区块中添加了一个**随机数字段(nonce)**，并规定：**每次创建区块，计算区块哈希值时，都要找到一个随机数，使得整个区块的哈希值小于某个数**（哈希值前有n个0），如此区块才能满足要求、被大家认可。由于哈希函数不可逆，无法从输出高效地推导出输入，只能通过尝试每一个输入，直到输出满足要求，也就是遍历随机数的值，而这个随机数有32字节（256位，10的77次方）。整个过程不仅靠算力，也靠一定的运气，因此这个过程又被称为**挖矿**，至于挖到什么，将在后文介绍。



当然，计算机算力每年都在提高，怎么保证难度一直合理呢？比特币规定**区块哈希数值前面0的个数为难度值，它会随着计算机算力的提高而增大**，0个数越多挖矿成功的概率也就越小、工作量也就越大。这个难度值会由比特币社区规定，当新区块未达到难度值（即区块哈希值前面0的个数不够），这个区块将不会被大家所接受。

由于区块会包含上一个区块的哈希值，恶意节点想要修改过去区块，则必须重做之后所有的区块从而赶上诚实节点们的工作（恶意节点在重做区块的同时，诚实节点也在创建新区块），而这个概率呈指数级递减。

1.5. 网络

上文介绍了工作量证明这种共识算法的规则，接下来将介绍其在比特币网络中的具体运行步骤：

1. 新交易向所有节点广播。
2. 每个节点将新交易收集到一个区块。
3. 每个节点为它的区块寻找工作量证明。即挖矿，找随机数使得区块哈希值满足难度要求。
4. 当一个节点找到了工作量证明，就向所有节点广播这个区块。
5. 节点只有在**区块内所有交易都是有效的且之前没有被支付**的情况下接收这个区块。同时，也会检查这个区块是否满足要求。
6. 节点通过在链中创建下一个区块时，将这个区块的哈希值作为上一个区块哈希值的方式，表示对这个区块的接受。

节点总是认为最长的链为正确的并持续致力于延长它。

如果两个节点同时广播了不同的下一个区块，有些节点可能先收到其中一个而其他节点先收到另一个。这种情况，节点基于他们收到的第一个区块工作，但是也保存另一个分支以防它变为更长的链。当下一个工作量证明被找到后僵局就会被打破从而其中一个分支变得更长；在另一个分支上工作的节点将切换到更长的链上来。

新交易的广播不必到达所有的节点。只要到达一些节点，不久就会进入到一个区块。区块广播也是能容忍消息丢失的。**如果一个节点没有收到某个区块**，它将在收到下一个区块时发现它丢失了一个区块（上一个区块哈希值对不上），然后**会跟其它节点请求这个区块。**

1.6. 激励

前文提到，恶意节点想要修改过去的数据，需付出昂贵的代价，还很可能追不上诚实节点们的最长链，导致努力白费。但是，这毕竟有利可图，恶意节点们还是会想尽办法去尝试。而接下来所介绍的激励机制，将让这些恶意者陷入纠结，甚至转而为比特币系统做贡献。

比特币系统约定：**区块创建者将获得一笔奖励币**，这份奖励将以交易的方式作为区块的第一笔交易。这笔交易由区块创建者自己创建，支付方可写成coinbase。当然，金额不能随便写，需要满足约定，不然其它人不会接受。但在比特币系统中，比特币是有限的，有可能到最后没有币发了。因此又约定：**区块创建者可从交易中获得手续费。**自然，谁给的钱多，谁的交易就会先被纳入到区块中。

激励机制不仅**解决了没有中心者如何发币的问题**，还有助于**鼓励节点保持诚实。**

> 如果一个贪心的攻击者有能力聚集比所有诚实节点更多的 CPU 算力，他将面临是以骗回已付款的方式欺诈别人还是使用这些算力生成新货币的抉择。他将发现遵守规则比破坏系统和他自己财产的有效性更有利，因为这些规则准许他获得比所有其他人都多的新货币。

这里解释一下，**为什么恶意者只能骗回已付款，而不能偷走别人的钱？**因为，若要创建一笔别人支付给恶意者的交易，或者将某笔交易的付款方修改成自己，则需要别人的私钥进行签名，而恶意者并没有别人的私钥，若恶意者用自己的私钥或不相干者的私钥，其它参与者将不会接受。

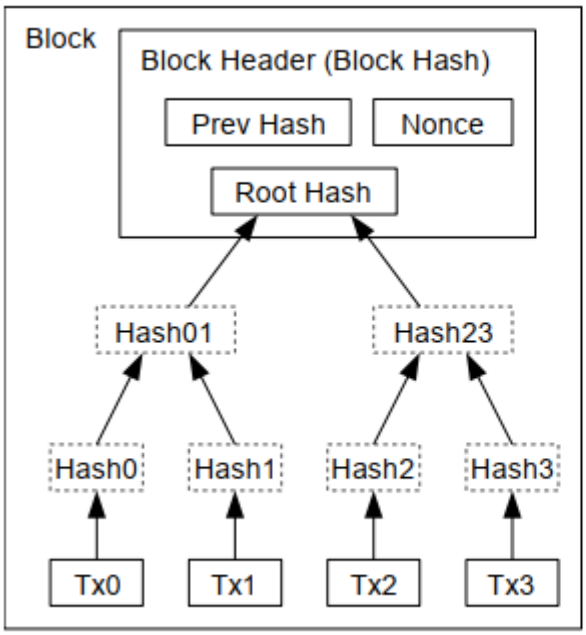
至此，比特币系统已基本成型！

1.7. 回收磁盘空间

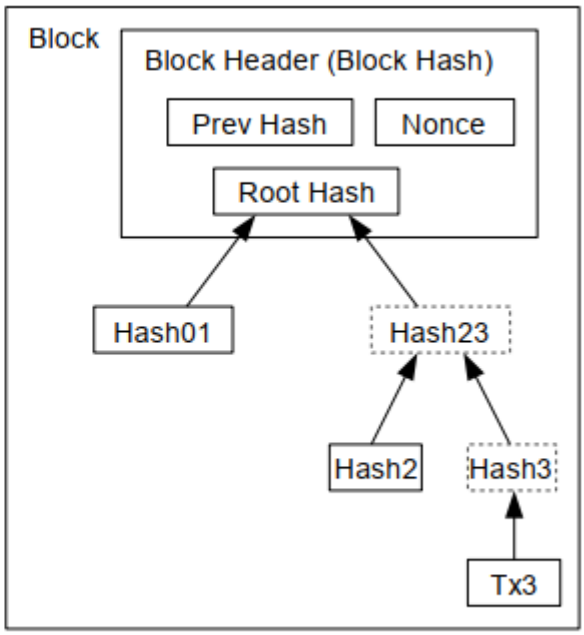
一旦某个币的最新交易被写入区块链中，且其所在区块之后有足够多的区块，那之前的支付交易就可以被丢弃以节省磁盘空间。为什么要保证交易所在区块之后有足够多的区块呢？因为系统中区块链可能会出现分叉的情况，有时一笔交易只在其中一个分支上，由于无法保证哪一条分支会成长为更长的链。所以，只有这个区块之后有足够多的区块，才能保证它被大多数人接受且在最长的链上，才能保证交易真正被写入区块链中，交易成功。而这个足够多通常是指**6个区块。**

举例解释：E支付给A 10个币，A再将这10个币支付给B。当A支付B 10个币的交易被写入区块链中，且这笔交易所在区块之后已有超过6个区块，那之前E支付A 10个币的交易就可以丢弃以节省磁盘空间，因为它已经不可能再作为交易来源了。当然这是自愿的，也可以不丢弃。

而为了方便于此又不影响区块的哈希值，交易将用**默克尔树(Merkle Tree)**保存，只有默克尔树的根节点才会被纳入区块哈希值的输入。于是，**区块被分为区块头和区块体，区块头包括上一个区块的哈希值、随机数、区块创建时间戳、默克尔树根节点等，区块体包含交易及其它信息，区块哈希值只需计算区块头即可。**



Transactions Hashed in a Merkle Tree



After Pruning Tx0-2 from the Block

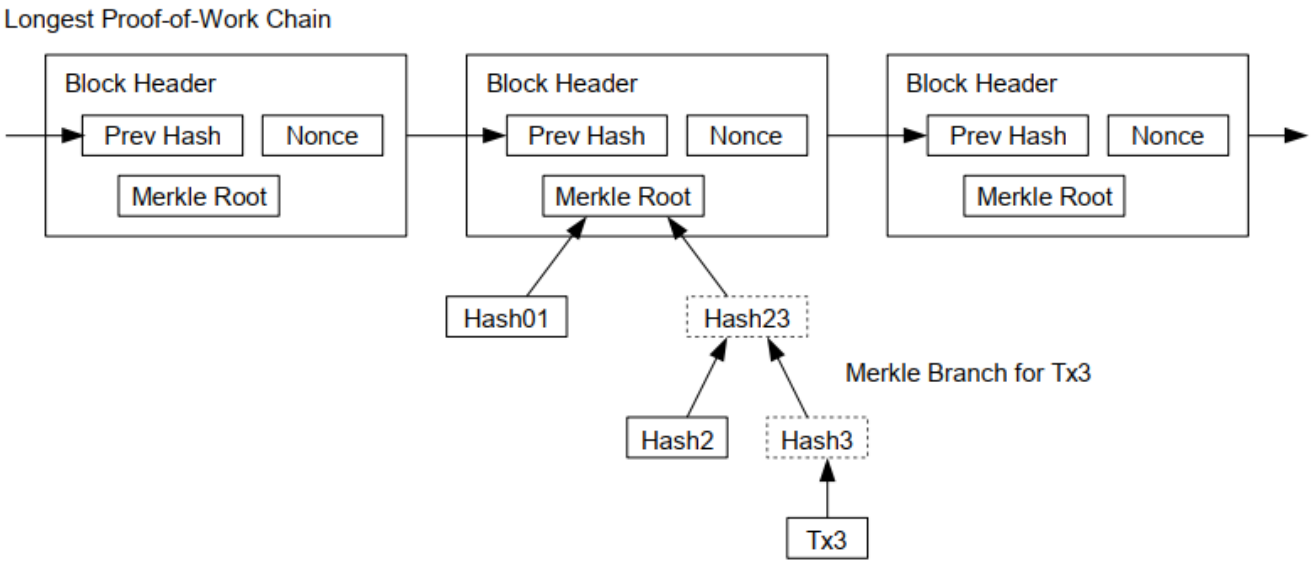
但不把交易纳入区块哈希值的计算，就不怕交易数据被篡改吗？这就不得不提到默克尔树了。默克尔树又称哈希树，是一颗完全二叉树，叶子节点的值是数据的哈希值，非叶子节点的值则是两个孩子节点的哈希值。当任意数据被篡改，其对应叶子节点的值会发生变化，进而影响父节点，层层向上最终影响到根节点。所以，**只要由底向上重新计算默克尔树的根哈希值，就能检验数据是否被篡改。**同时，可以通过剪除树枝的方式压缩数据，比如在上图中，Tx0-Tx2可以被丢弃了，那只需要保存Tx3、Hash01、Hash2就可以了。

每个不包含交易的区块头大约是 80 bytes。如果每 10 分钟生成一个区块，每年仅生成80 B * 6 * 24 * 365 = 4.2 MB。

1.8. 简化的支付验证

结合默克尔树，用户甚至**只需拥有最长链的最后一区块的区块头**，就可以完成**简化的支付验证(SPV, Simplified Payment Verification)。**

当他要验证某笔交易是否成功时，他可以通过向其他网络节点询问以确认他拥有了最长的链，并获取交易所在的区块头及其之后的区块头，同时获取交易所在的默克尔树分支。虽然他不能核实这个交易，但他可以通过**计算并验证哈希值的方式，证明交易已经存在于某个区块的默克尔分支上。**如果交易已存在与某个区块中，且其后有足够多的区块，就可以确认网络已经接受了这笔交易。

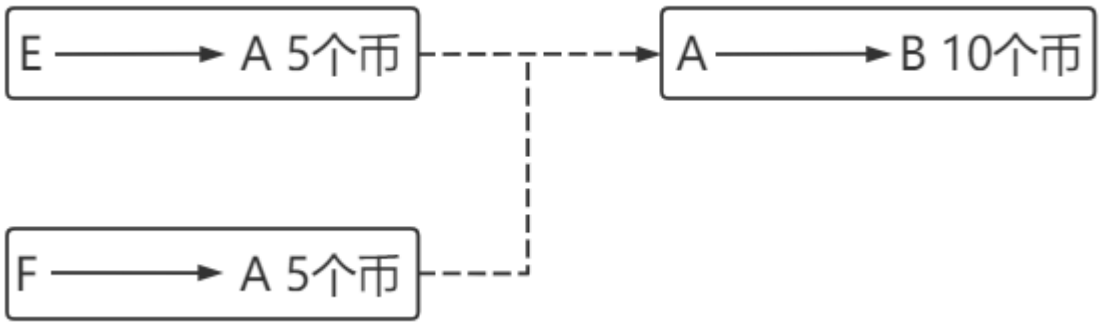


但是，如果部分网络被恶意者控制，简化验证将会变得比较脆弱。恶意者可以伪造一条更长的链和错误的交易，来欺骗他所控制网络中的节点。一种对策是发现一个无效区块时，发出警告。而为了更加独立的安全性以及更快的支付确认，比特币交易频繁的公司仍需保存最长的完整的区块链。

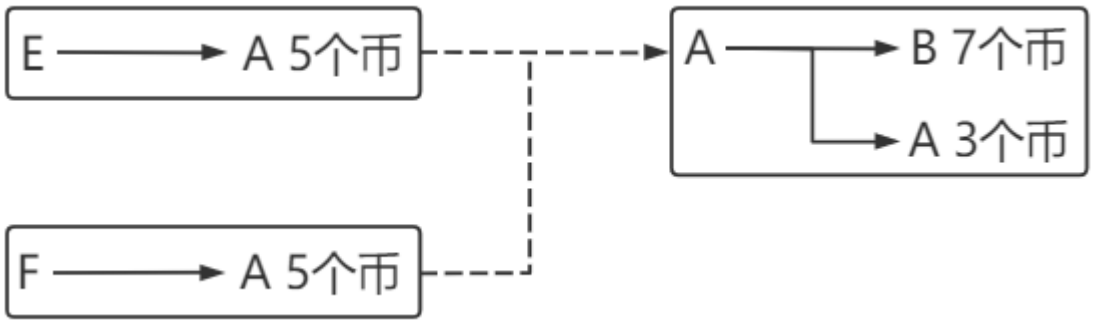
通常，保存所有区块数据且具备完整功能的节点被称为**全节点**，而只保存少量区块数据的节点被称为**轻节点。**轻节点常用来交易比特币，可通过SPV进行支付验证，常见的轻节点钱包有Electrum等。插一嘴：火币、OKEx等交易平台所提供的客户端软件并不是一个节点，所有支付都是由平台完成与验证的，用户需要信任且完全依赖平台，本质还是一个中心化系统。

1.9. 合并和分割交易额

每笔需指定交易来源，但往往交易来源不止一处，比如：F支付A 5个币，E支付A 5个币，最终A将这10个币支付给B，那A支付B的这笔交易就有两个来源。所以为了方便，比特币规定**一笔交易可有多个输入值来源，同时可有一至两个输出值**（现在可以有多个输出）。



那为什么要有两个输出呢？当输入值大于输出值时，可用于找零。同时，也可以表示支付方未花费的余额，这也被称为**UTXO(Unspent Transaction Outputs，未花费交易输出)**。



另外，若总输出值加起来小于总输入值，差值将作为交易的手续费支付给矿工。

1.10. 隐私

很好理解，比特币上的交易内容中，付款方与收款方都以他们的公钥显示，我们并不知道背后是谁。所以，比特币比普通货币更具隐匿性，当然这也为不法分子提供了一个天堂。

1.11. 计算

通过计算证明恶意者链条赶上诚实者链条的概率，将随着区块数的增加而呈指数下降。感兴趣可自行阅读论文，此处不做讲解。