

BITCOIN 原文解读

本文是对中本聪的论文《Bitcoin: A Peer-to-Peer Electronic Cash System》进行研究，通过**原文—翻译—解读**的方式对该论文进行研究学习。

作者：中本聪

整理：xfli5

推特：@xfli5

0. Abstract (摘要)

原文：

Abstract: 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.

翻译：

[摘要]：本文提出了一种完全通过点对点技术实现的电子现金系统，它使得在线支付能够直接由一方发起并支付给另外一方，中间不需要通过任何的金融机构。虽然数字签名（Digital signatures）部分解决了这个问题，但是如果仍然需要第三方的支持才能防止双重支付（double-spending）的话，那么这种系统也就没有多大的意义。**我们在此提出一种解决方案，通过使现金系统在点对点网络环境下运行来防止双重支付问题。**该网络对全部交易加上时间戳（timestamps），并将所有交易（transactions）执行哈希（hash）后合并入一个不断延伸的基于随机散列（hash-based）的工作量证明（proof-of-work）的链条作为交易记录，除非重新完成全部的工作量（pow）证明，否则形成的交易记录将不可更改。最长的链条不仅将作为被观察到的事件序列(sequence)的证明,而且被看做是来自 CPU 计算能力最大的池（pool）。只要大多数的 CPU 计算能力都没有打算合作起来对全网进行攻击，那么诚实的节点将会生成最长的、超过攻击者的链条。**这个系统**

本身需要的基础设施非常少。信息尽最大努力在全网传播即可，节点(nodes)可以随时离开和重新加入网络，并将最长的工作量证明（pow）链条作为在该节点离线期间发生的交易的证明。

解读：

任何一篇白皮书或论文摘要部分都非常重要，其实摘要最重要的是告诉读者，我发现了什么问题，并且通过什么办法解决这些问题，且说明了解决这个问题的主要方式和优势。我看过一些白皮书，基本上摘要部分连问题都没有抛出，就开始介绍自己的东西，这是不恰当的。其实你只要仔细看了比特币白皮书摘要部分就知道中本聪抛出的问题了，“**一个需要第三方支持的点对点电子现金支付系统是没有多大价值的**”这容易和传统上的 p2p 形成误解，很多 p2p 系统其实是需要一个第三方支持的，比如有些 p2p 系统需要一个或多个索引服务器，迅雷下载就是比较典型的，他需要一个中心化的服务器帮助点对点建立连接，所以我们要搞清楚 p2p 与去中心化的 p2p，知道了这些，**比特币以及其他竞争币就可以一句话进行概括了，“一个去中心化的 p2p 支付系统”**，既然是支付系统就当然要解决双重支付问题，这和是否是去中心化没有关系，只是中心化的系统更容易解决双重支付问题而已。

通过摘要我们了解到，比特币系统要解决的两个主要问题，**一个是去中心化的 p2p 系统，一个是支付要解决的双重支付问题**，所以整个白皮书提到的技术方案都是围绕这两个问题展开的，其实中本聪在写这篇论文时，有关去中心化的 p2p 系统已经有比较完善的解决方案了，所以白皮书的重点都放在如何通过 p2p 系统解决双花问题。**要解决双重支付问题就必须记账核对**，那么如何在一个没有中心的 p2p 系统中记账又受到认同呢，比特币引入了**基于时间戳的随机散列**

(hash)，并且让其形成前后文相关的序列，这就是为什么称之为区块链的原因。

这里必须插一句，我认为**记账的不可更改性**是一个比特币的副产品，首先区块链记账不可更改是有前提的，这经常被称为 51%攻击，根据原文摘要的这句话“除非重新完成全部的工作量证明，否则形成的交易记录将不可更改”，由于采用了复杂的 POW 工作量证明计算，更改交易记录将非常的耗时和需要大量运算，从客观上实现了“不可更改”的目的，但是从业务设计的角度，这是一个缺陷，如果更改是交易双方认同的而且合法的，或者说被绝大多数节点认同的呢？由于技术的限制比特币系统进行了取舍，所以不得不保留了这个缺陷(参见 CAP 原则)，同时在白皮书中把这个缺陷讲成了一个特点，这是中本聪先生非常聪明的地方。其实比特币后来的一些事实证明，比特币的滥用也是因为这个缺陷，基于暗网的非法交易，网络勒索，包括黑客盗用比特币，“门头沟”交易所跑路等，都是利用了这个缺陷，其实根据“多数人正义”这点来讲，这些交易都应该撤销，但是这个要回滚交易的代价实在太太大，大到不可实现（需要分叉），看到这里你就知道比特币系统不是你想象中那么完美了吧

摘要中还透露一个比特币系统的重要特点“这个系统本身需要的基础设施非常少”，因为有了中心化的服务器存在，相关的维护、备份、容灾、管理等成本会大大降低。

1.Introduction（简介）

1.1 原文

Commerce on the Internet has come to rely almost exclusively on financial institutions serving as trusted third parties to process electronic payments. While the system works well enough for most transactions, it still suffers from the inherent weaknesses of the trust based

model. Completely non-reversible transactions are not really possible, since financial institutions cannot avoid mediating disputes. The cost of mediation increases transaction costs, limiting the minimum practical transaction size and cutting off the possibility for small casual transactions, and there is a broader cost in the loss of ability to make non-reversible payments for nonreversible services. With the possibility of reversal, the need for trust spreads. Merchants must be wary of their customers, hassling them for more information than they would otherwise need. A certain percentage of fraud is accepted as unavoidable. These costs and payment uncertainties can be avoided in person by using physical currency, but no mechanism exists to make payments over a communications channel without a trusted party.

What is needed is an electronic payment system based on cryptographic proof instead of trust, allowing any two willing parties to transact directly with each other without the need for a trusted third party. Transactions that are computationally impractical to reverse would protect sellers from fraud, and routine escrow mechanisms could easily be implemented to protect buyers. In this paper, we propose a solution to the double-spending problem using a peer-to-peer distributed timestamp server to generate computational proof of the chronological order of transactions. The system is secure as long as honest nodes collectively control more CPU power than any cooperating group of attacker nodes.

1.2 翻译

互联网上的贸易，几乎都需要借助金融机构作为可资信赖的第三方来处理电子支付信息。虽然这类系统在绝大多数情况下都运作良好，但是这类系统仍然内生性地受制于“基于信用的模式”(trust based model)的弱点。我们无法实现完全不可逆的交易，因为金融机构总是不可避免地会出面协调争端。而金融中介的存在，也会增加交易的成本，并且限制了实际可行的最小交易规模，也限制了日常的小额支付交易。并且潜在的损失还在于，很多商品和服务本身是无法退货的，如果缺乏不可逆的支付手段，互联网的贸易就大大受限。因为有潜在的退款的可能，就需要交易双方拥有信任。而商家也必须提防自己的客户，因此会向客户索取完全不必要的个人信息。而实际的商业行为中，一定比例的欺诈性客户也被认为是不可避免的，相关损失视作销售费用处理。而在使用物理现金的情况下，这些销售费用和支付问题上的不确定性却是可以避免的，因为此时没有第三方信用中介的存在。

因此，我们非常需要这样一种电子支付系统，它基于密码学原理而不基于信用，使得任何达成一致的双方，能够直接进行支付，从而不需要第三方中介的参与。杜绝回滚(reverse)支付交易的可能，这就可以保护特定的卖家免于欺诈；而对于想要保护买家的人来说，在此环境下设立通常的第三方担保机制也可谓轻松愉快。在这篇论文中，我们将提出一种通过点对点分布式的时间戳服务器来生成依照时间前后排列并加以记录的电子交易证明，从而解决双重支付问题。只要诚实的节点所控制的计算能力的总和，大于有合作关系的(cooperating)攻击者的计算能力的总和，该系统就是安全的。

1.3 解读

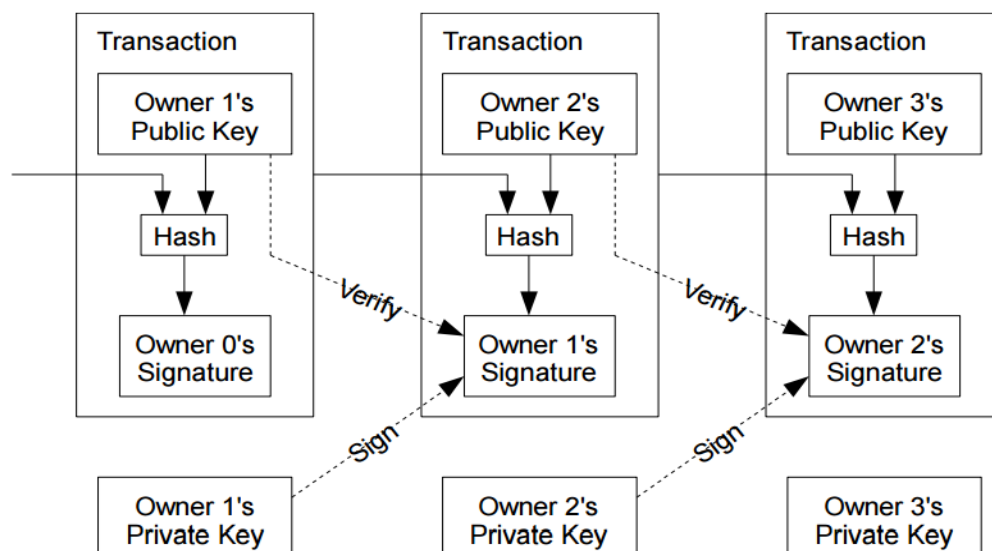
我是这样看待简介的，中本聪很清楚比特币系统因为妥协所造成的缺陷，就是刚才说的**记账不可更改**，所以在简介中就阐述了可更改的弱点，这也是我认为最有争议的地方，“如果缺乏不可逆的支付手段，互联网的贸易就大大受限”这点有些牵强。但是，简介中说明的**通过数学，通过密码原理代替中心化中介信用（金融机构）使交易达成却是革命性的**，这也是比特币最有研究价值的地方，这里面需要注意的是比特币系统代替的是**中心化中介的信用**，并不能保证交易双方的信用，相反一旦交易的某一方出现信用问题这笔交易是无法撤销的，对于初次接触比特币的人一定要尤为注意，也就是说和传统银行交易不同，一旦你确定支付，哪怕对方是骗了你，你的交易损失是无法追回的。这里面又一个悖论，**你是信任中心化中介的信用还是信任你的交易对方的信用**，这就要仁者见仁了，如果你选择后者那么比特币系统是你的好选择，如果你选择前者，那么传统的中心化支付是你的选择。

总之，白皮书第一节传达给我们的就是，你需要一个去中心化中介，并且主要保护卖家的系统，如果你需要保护买家，你仍然需要一个第三方担保机制，"而对于想要保护买家的人来说，在此环境下设立通常的第三方担保机制也可谓轻松愉快"。而对于系统的安全性完全通过攻击算力成本保障，这是一种算力的比拼，你可以简单的理解为，**正常的算力是大于想作恶的算力这个前提的**。了解了这一点，你就会知道如何选择合适的数字货币，避免山寨币陷阱了。

2.Transactions (交易)

2.1 原文

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.



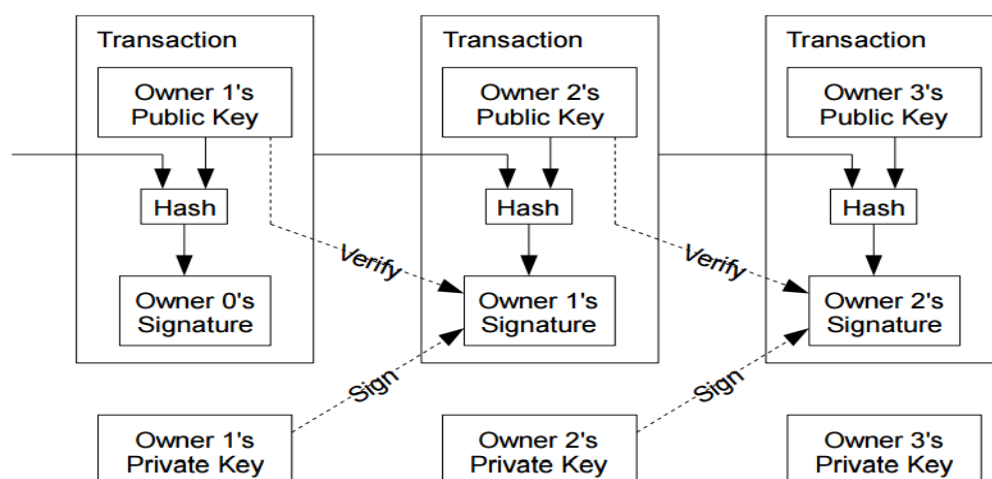
The problem of course is the payee can't verify that one of the owners did not double-spend the coin. A common solution is to introduce a trusted central authority, or mint, that checks every transaction for double spending. After each transaction, the coin must be returned to the mint to issue a new coin, and only coins issued directly from the mint are trusted not to be double-spent. The problem with this solution is that the fate of the entire money system depends on the company running the mint, with every transaction having to

go through them, just like a bank.

We need a way for the payee to know that the previous owners did not sign any earlier transactions. For our purposes, the earliest transaction is the one that counts, so we don't care about later attempts to double-spend. The only way to confirm the absence of a transaction is to be aware of all transactions. In the mint based model, the mint was aware of all transactions and decided which arrived first. To accomplish this without a trusted party, transactions must be publicly announced [1], and we need a system for participants to agree on a single history of the order in which they were received. The payee needs proof that at the time of each transaction, the majority of nodes agreed it was the first received.

2.2 翻译

我们将一枚电子货币（an electronic coin）定义为一串数字签名。每一位所有者通过对前一次交易和下一位拥有者的公钥(Public key) 签署一个随机散列(hash) 的数字签名，并将这个签名附加在这枚电子货币的末尾的方式就算将电子货币发送给了下一位所有者。而收款人通过对签名进行检验，就能够验证该链条的所有者。



该过程的问题在于，收款人将难以检验，之前的某位所有者，是否对这枚电子货币进行了双重支付。通常的解决方案，就是引入信得过的第三方权威，或者类似于造币厂(mint)的机构，来对每一笔交易进行检验，以防止双重支付。在每一笔交易结束后，这枚电子货币就要被造币厂回收，而造币厂将发行一枚新的电

子货币；而只有造币厂直接发行的电子货币，才算作有效，这样就能够防止双重支付。可是该解决方案的问题在于，整个货币系统的命运完全依赖于运作造币厂的公司，因为每一笔交易都要经过该造币厂的确认，而该造币厂就好比是一家银行。

我们需要收款人有某种方法，能够确保之前的所有者没有对更早发生的交易实施签名。为了达到目的，实际上我们需要关注的**只是于本交易之前发生的交易，而不需要关注这笔交易发生之后是否会有双重支付的尝试**。为了确保某一次交易是不存在的，那么唯一的方法就是获悉之前发生过的所有交易。在造币厂模型里面，造币厂获悉所有的交易，并且决定了交易完成的先后顺序。**如果想要在电子系统中排除第三方中介机构，那么交易信息就应当被公开宣布（publicly announced），我们需要整个系统内的所有参与者，都有唯一公认的历史交易序列**。收款人需要确保在交易期间绝大多数的节点都认同该交易是首次出现。

2.3 解读

2.3.1 比特币交易构成

（1）比特币交易简介及分类

交易(Transaction)是比特币系统的信息载体，最小单元。而块(Block)就是将这些基础单元打包装箱，贴上封条，并串联起来。巨大算力保障了块的安全，也就保障了单个交易的安全。交易（transaction）有三种常见类型：创世交易(Generation)，合成地址交易(Script Hash)，通用地址交易(Pubkey Hash)。该分类并非严格意义的，只是根据交易的输入输出做的简单区分。

1) Generation TX

每个 Block 都对应一个创世交易(Generation TX)，该类交易是没有输入交易的，挖出的新币是所有币的源头。

2) Script Hash TX

该类交易目前不是很常见，大部分人可能没有听说过，但是非常有意义。未来应该会在某些场合频繁使用。该类交易的接受地址不是通常意义的地址，而是一个合成地址，以 3 开头（对，以 3 开头的也是比特币地址！）。三对公私钥，可以生成一个合成地址。在生成过程时指定 n of 3 中的 n，n 范围是[1, 3]，若 n=1，则仅需一个私钥签名即可花费该地址的币，若 n=3，则需要三把私钥依次签名才可以。

3) Pubkey Hash TX

该类是最常见的交易类型，由 N 个输入、M 个输出构成。

(2) 数据结构

交易中存放的是**货币所有权**的流转信息，所有权登记在比特币地址上(Public Key)。这些信息是全网公开的，以**明文形式**存储（**比特币系统里的所有数据都是明文的**），只有当需要转移货币所有权时，才需要用私钥签名来验证。

字段大小	描述	数据类型	解释
4	version, 版本	uint32_t	交易数据结构的版本号
1+	tx_in count, 输入数量	var_int	输入交易的数量
41+	tx_in	tx_in[]	输入交易的数组，每个输入>=41字节
1+	tx_out count, 输出数量	var_int	输出地址的数量
9+	tx_out	tx_out[]	输入地址的数组，每个输入>=9字节
4	lock_time, 锁定时间	uint32_t	见下方解释

lock_time 是一个多意字段，表示在某个高度的 Block 之前或某个时间点之前该交易处于锁定态，无法收录进 Block。关于 lock_time 取值及含义如下：

值	含义
0	立即生效
< 500000000	含义为 Block 高度, 处于该 Block 之前为锁定 (不生效)
>= 500000000	含义为 Unix 时间戳, 处于该时刻之前为锁定 (不生效)

若该笔交易的所有输入交易的 sequence 字段, 均为 INT32 最大值(0xffffffff), 则忽略 lock_time 字段。否则, 该交易在未达到 Block 高度或达到某个时刻之前, 是不会被收录进 Block 中的。

uint32:表示无符号整数, 范围 $0 \sim 4294967295(2^{32}-1)$ 。

int32:表示带符号整数, 范围 $-4294967295(2^{32}-1) \sim 4294967295(2^{32}-1)$ 。

2.3.2 其他解读

上图说明了**比特币的交易记录**方式, 是一个**基于时间序列的链式结构**, 如果说链式结构可能有人理解起来有困难, 你可以把每笔交易想象成一条记录, 这条记录记录了**付款人地址和收款人地址以及交易数据**等重要信息, 我们还是用简单语言来举例, 我们假设一条交易记录 (Tx) 是“用户 xfli 给用户 jcshu 十个比特币”, 这条交易信息将写成如下形式:

Tx:{

发送者 (xfli) 地址: xfli 的公钥

接受者 (jcshu) 地址: jcshu 的公钥

Bitcoin 数量: 10

}

注：bitcoin 的地址是由公钥经过 hash 及编码生成的，是与公钥不同的，但本质上是一致的，此处我们假设公钥就是 bitcoin 地址（也就是我们常说的钱包地址）。

在原文中“**每一位所有者通过对前一次交易和下一位拥有者的公钥(Public key) 签署一个随机散列 (hash) 的数字签名**”，那么为啥要对前一次的交易和下一位拥有者的公钥进行 hash 后进行数字签名？这是因为交易信息“用户 xfli 给用户 jcshu 十个比特币”创建完成后，需要通过网络传播到其他的节点进行检验。其他的节点检验的内容：

(1) 用户 xfli 是真正的发送者。由于用户 xfli 会公布自己的公钥，网络中的其他节点通过公钥就可以来验证数字签名，因为数字签名是用户 xfli 用自己的私钥加密。

(2) 用户 xfli 具有足够的 bitcoin 金额。用户 xfli 告诉全网络的节点说自己要支付 10 个比特币给 jcshu，那么网络的节点就要检验你到底有没有这么多的比特币，这也是原文中“**上一次交易**”的作用，通过核查用户 xfli 的上一次的交易信息，我们就知道该用户到底有没有这么多比特币。

原文中“我们将一枚电子货币 (an electronic coin) 定义为一串数字签名”，很容易让人误解的是这里的电子货币是个具体数字，其实**它是一条记录（数据）**，也就是说你拥有的交易记录才是你的数字资产，这里面你需要注意的是，这条记录里面没有你的姓名、身份证号等这些中心化系统中常见的识别所有者的字段，**唯一表示你身份的是你的公钥（钱包），证明你所有权的是你的私钥（密码）。**

原文中的第二段里面最重要的是提出了**关于双重支付（双花）**的问题，我们前面说过，根据这个交易链条，我们可以知道之前的每位所有者，但是有一个东

西是不知道的，那就是某位记录的拥有者是否同时创建了两条或多条新的记录，将他的比特币花出去，如果没有一个机制来保证一切都会乱套。“通常的解决方案，就是引入信得过的第三方权威，或者类似于造币厂(mint)的机构，来对每一笔交易进行检验。中本聪在白皮书指出了传统的两种方法，并且说明了他们的坏处。第三方权威验证就是典型的中心化系统，这个和设计初衷违背 pass 掉了。造币厂模型其实也是中心化的系统，中本聪也将其 pass 掉了，但是中本聪创造了交易公布模型，即人人都可以参与验证是否双花的过程，在比特币系统中存在功能不同的节点，如果你**参与到交易的验证过程中那么你也就不是一名“矿工”了**，通过矿工模型你不但可以挖到新的比特币，也可以参与验证计算获得奖励，这就是为什么比特币交易过程中你需要花费手续费的原因了，下面我们接着看中本聪提出的如何解决造币厂中心化问题：

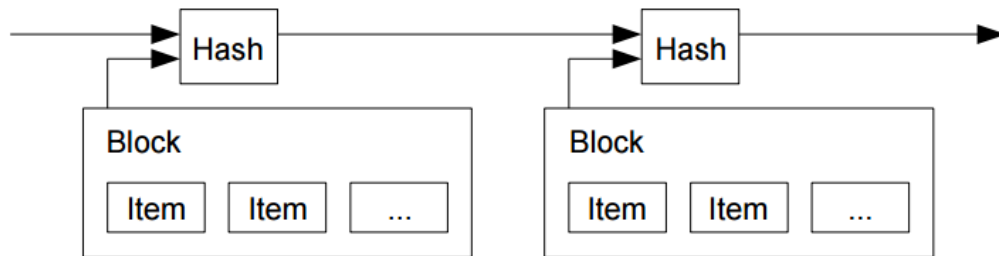
原文第三段里面最需要理解是这句“**为了达到目的，实际上我们需要关注的只是于本交易之前发生的交易**”，前文我们知道比特币的交易是一个时间序列的链条，在产生一条新的交易记录时永远有先后顺序，即便是双重支付你总有先后顺序，你不可能同时创造两笔交易，那么我们只需要证明其中一条是有效的即可，并且将其记录到交易链条上即可，那么其他的交易就是无效的了（这也就是比特币的交易确认原理，交易确认是需要时间的）。

要证明其中一条是有效的又不允许中心化从存在，那么只有一个办法了，就是发动所有的人参与这项活动，进行“多数人的正义”，所以原文中“**那么交易信息就应当被公开宣布（publicly announced）**”这句话就容易理解了，如果你不公布交易信息，其他人怎么进行交易验证呢。比特币就是在交易创建时将信息公布出去，如果有多数人优先确认并将其记录下来，这个过程就完成了。

3.Timestamp Server (时间戳服务)

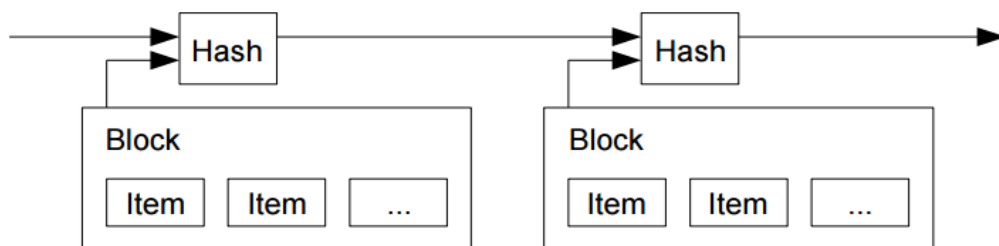
3.1 原文

The solution we propose begins with a timestamp server. A timestamp server works by taking a hash of a block of items to be timestamped and widely publishing the hash, such as in a newspaper or Usenet post. The timestamp proves that the data must have existed at the time, obviously, in order to get into the hash. Each timestamp includes the previous timestamp in its hash, forming a chain, with each additional timestamp reinforcing the ones before it.



3.2 翻译

本解决方案首先提出一个“时间戳服务”。时间戳服务通过对以区块(block)形式存在的一组 item 实施随机散列(hash)并加上时间戳，并将该随机散列(hash)进行广播，就像在新闻或世界性新闻组网络(Usenet)的发帖一样。显然，该时间戳能够**证实特定数据必然于某特定时刻是的确存在**的，因为只有在该时刻存在了才能获取相应的随机散列值(hash)。每个时间戳应当将前一个时间戳(也就是前一个区块的hash值)纳入其随机散列值(hash)中，每一个随后的时间戳都对之前的一个时间戳进行增强(reinforcing)，这样就形成了一个链条(Chain)。



3.3 解读

3.3.1 解读 1

时间戳很容易理解，原文中说“时间戳服务器通过对以区块(block)形式存在的一组数据实施随机散列后加上时间戳”，加上时间戳容易，**但是如何取得时间呢**，如果是加上本机时间，那么每个节点的时间都不统一，况且有人恶意更改本机时间呢。如果不是本机时间，去网络上取国际标准组织的时间似乎是可行的，但是，这是典型的中心化操作啊，和中本聪倡导的去中心化理念不符。

那么如何获得一个去中心化的时间呢，其实方法很简单，还是利用“多数人的正义”，时间来自于连接的其他节点（node）时间的中位数（median，是个数学概念，比平均值更不受极端数字影响），要求连接的节点（node）数量至少为5，中位数和本地系统时间差别不超过70分钟，否则会提醒你更新本机的时间。同时，在接收到新的block时会拒绝时间与自己差距+2小时和-(前11个block时间中位数)的block，**由此我们知道了这个时间的正确来源了，还是利用大多数人的机器时间，其实比特币的核心理念一直是认为“多数人是正义”的**，我一直在想如果从一开始比特币的系统时间就比实际时间快1年的话，其实对整个系统是没有影响的，原因很简单后加入的节点都会按照前面多数人的时间中位数更改自

己的节点时间，最后成为比特币系统自有的时间模式（如果大多数人都光着屁股跑，穿衣服的可能就不正常了，呵呵），好在比特币系统运行刚开始，时间还是比较准确的，不过我一直认为这个时间机制有漏洞可以钻。

了解比特币时间戳的机制后，后面的就容易理解了，将一段时间内，准确的说是，**两个区块的产生间隔中的所有交易进行 Hash 运算，再盖上时间戳**，这样我们就有了一个按时间序列产生的所有交易的记录了，区块链就这样形成了。因为时间是线性增长的，所以区块链是按照时间先后顺序组织起来的，所以核心代码中关于时间的机制异常重要，我认为这是安全上比较脆弱的环节。

3.3.2 解读 2

(1) 存在证明

存在证明就是向第三方证明某个物品/事件，在过去的某个时刻存在过。

这是一件很简单的事情，提供票据、通信记录之类的就可以办到。但这些并不严格，因为这些证据都是非常易伪造或销毁。要完成证明，必须依赖强有力的证据链，这个必须是任何人都无法伪造与销毁的，或者说伪造成本极其高昂近乎不可能。

回忆一下，电影里经常出现的绑匪镜头，他们为了证明在某个时间确实拥有人质，而不是事前拍摄的视频，通常会用当天的发行量很大的报纸来辅助证明。当香港媒体误报“成龙高楼坠亡”时，成龙也不得不拿报纸来证明自己的存在：



报纸之所以能够成为有效的时间证明系统是因为：

1) 不可伪造性。新闻等信息是无法预测的，尤其是证券大盘数据，报纸上大量充满这样的信息，所以无人能够提前伪造。

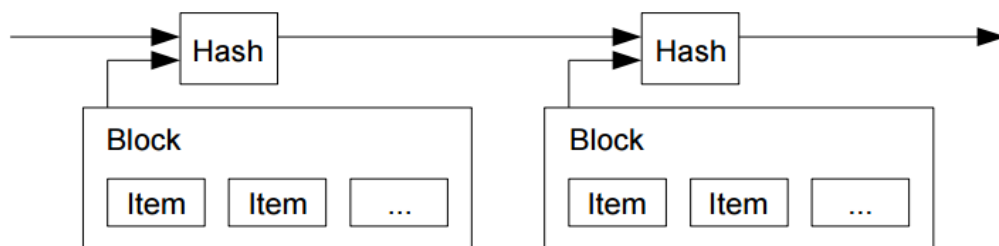
2) 公开且不可销毁。报纸通常拥有很大的发行数量，受众广泛，一旦发布出去就分散到各个角落，很难再次收集齐全并全部销毁。通常图书馆也会存档数十年期限的报纸。

3) 具有时间特征。报纸具有很强时间特征，版面到处可见的是时间标记。

借助报纸可以完成某个时间之后的存在证明，但无法完成某个时间之前的。例如，你拿 9 月 1 号的报纸拍摄进照片，那么仅能证明其在 9 月 1 号之后拍摄，可能是 9 月 1 号，也可能是 9 月 15 号。

(2) 时间戳服务

比特币本质是构造了一个永不停息、无坚不摧的时间戳系统。



然后该系统上添加若干特性后使得具有货币的功能。报纸从另一个角度讲也是一种时间戳服务。比特币具有下列优良的特性可以更完美的用于存在证明：

1) 不可预测/伪造。因 block 的计算是随机事件，其 hash 值是一个 32 字节的随机大数(256bit)。想蒙对该数的概率实在是太低了。

2) 不可销毁/修改。Block Chain 拥有巨大的算力在维护与延续，对于 N 个确认的 block，想篡改是不可能的。

3) block 具有天然时间特性。timestamp 是 block header（区块头）字段之一。

(3) 比特币做存在证明

1) 时间点向后证明

因为 block hash 的不可伪造性，能提供 Block Hash 即可证明存在于该 Block 时刻之后。例如，你在拍照的时候，拿着打印有 block hash 的纸即可证明：你在该 block 时刻之后进行的拍摄。

2) 时间点前向证明

前向证明需要精心构造一个包含消息摘要（hash 值）的交易，待该交易进入 block 中。便可以证明你在该 block 时刻之前拥有该数字摘要。前向证明的关键是能把信息写入时间戳服务载体。

3) 时间区间证明

有时候, 仅仅证明时间点之前或之后是不够的, 需要能够确认到某一个时刻。

将上述方式综合即可完成：

将 block A 的 hash 值添入数据文件, 并制作文件消息摘要。(时间点前向证明)

将摘要信息构造至交易中, 广播之。(时间点前向证明)

当交易被 block B 收录进去, 那么即可证明, 该文件于 block A 与 B 的时间间隔中存在。

如果交易给了足够的矿工费(Transaction Fee), 具有较高优先级的话, 便很有可能被紧随其后的 block 收录。连续的 block 约 10 分钟, 那么就在一个相对小的时间内作了证明, 可以近似认为是时间点。

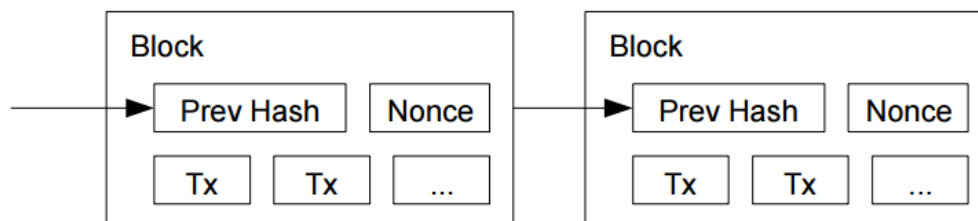
4. Proof-of-Work (工作量证明)

4.1 原文

To implement a distributed timestamp server on a peer-to-peer basis, we will need to use a proof-of-work system similar to Adam Back's Hashcash [6], rather than newspaper or Usenet posts. The proof-of-work involves scanning for a value that when hashed, such as with SHA-256, the hash begins with a number of zero bits. The average work required is exponential in the number of zero bits required and can be verified by executing a single hash.

For our timestamp network, we implement the proof-of-work by incrementing a nonce in the block until a value is found that gives the block's hash the required zero bits. Once the CPU effort has been expended to make it satisfy the proof-of-work, the block cannot be changed without redoing the work. As later blocks are chained after it, the work to change the block would include redoing all the blocks after it.

The proof-of-work also solves the problem of determining representation in majority decision making. If the majority were based on one-IP-address-one-vote, it could be subverted by anyone able to allocate many IPs. Proof-of-work is essentially one-CPU-one-vote. The majority decision is represented by the longest chain, which has the greatest proof-of-work effort invested in it. If a majority of CPU power is controlled by honest nodes, the honest chain will grow the fastest and outpace any competing chains. To modify a past block, an attacker would have to redo the proof-of-work of the block and all blocks after it and then



catch up with and surpass the work of the honest nodes. We will show later that the probability of a slower attacker catching up diminishes exponentially as subsequent blocks are added.

To compensate for increasing hardware speed and varying interest in running nodes over time, the proof-of-work difficulty is determined by a moving average targeting an average number of blocks per hour. If they're generated too fast, the difficulty increases.

4.2 翻译

为了在点对点的基础上构建一组散列化的时间戳服务, 仅仅像报纸或世界性新闻网络组一样工作是不够的, 我们还需要一个类似于亚当·柏克 (Adam Back) 提出的哈希现金 (Hashcash)。在进行随机散列 (hash) 运算时, 工作量证明机制引入了对某一个特定 hash 值 (比方说 hash 算法为 SHA-256) 的寻找工作, 该 hash 值以一个或多个 0 开始。那么随着 0 的数目的上升, 找到符合条件的 hash 值所需要的工作量将呈指数增长, 但是检验结果仅需要一次随机散列 (hash) 运算 (也就是将找的值通过执行 hash 后与给定的特定值核对)。

我们在区块中补增一个随机数 (Nonce), 这个随机数要使得该给定区块的随机散列值 (hash) 出现了所需的那么多个 0。我们通过反复尝试来找到这个随机数 (nonce), 找到为止。这样我们就构建了一个工作量证明机制。只要该 CPU 耗

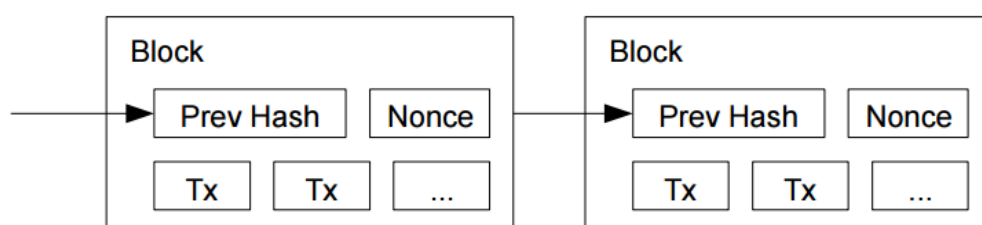
费的工作量能够满足该工作量证明机制，那么除非重新完成相当的工作量，否则该区块的信息就不可更改。由于之后的区块是链接在该区块之后的，所以想要更改该区块中的信息，就还需要重新完成之后所有区块的全部工作量。

同时，该工作量证明机制还解决了在集体投票表决时，谁是大多数的的问题。如果决定大多数的方式是基于 IP 地址的一个 IP 地址一票，那么如果有人拥有分配大量 IP 地址的权力，则该机制就被破坏了。**而工作量证明机制的本质则是一 CPU 一票。**“大多数”的决定表达为最长的链，因为最长的链包含了最大的工作量。如果大多数的 CPU 为诚实的节点控制，那么诚实的链条将以最快的速度延长，并超越其他的竞争链条。**如果想要对已出现的区块进行修改，攻击者必须重新完成该区块的工作量外加该区块之后所有区块的工作量**，并最终赶上和超越诚实节点的工作量。我们将在后文证明，设想一个较慢的攻击者试图赶上随后的区块，那么其成功概率将呈指数化递减。

另一个问题是，硬件的运算速度在高速增长，且节点参与网络的程度会有所起伏。为了解决这个问题，工作量证明的难度(the proof-of-work difficulty)将采用移动平均目标的方法来确定，即令难度指向令每小时生成区块的速度为某一预设的平均数。如果区块生成的速度过快，那么难度就会提高。

4.3 解读

4.3.1 解读 1



这个图实际上和上一个图一样，只是这里着重体现出每个区块的内容，指明一个区块是包含上一个 hash 的(Prev Hash)、Nonce、交易信息(Tx) 等等。一旦更改 Nonce，Prev Hash, Tx 其中任意一个，那么这个区块的 Hash 也会改变，之后的区块也全部都要改。(注：这里针对区块包含内容进行简化，详细参考区块链研究中一.区块链的整体介绍中的 4.区块链主要概念中的 4.9 比特币区块)

在第三讲我们知道了比特币的时间戳机制，可能还是有人有疑虑，虽然比特币取多个节点和前几个区块的中位数时间作为标准，但是还有一个问题就是，如果有人恶意更改时间，同时修改比特币源码造跳过时间检测机制怎么办或者恶意构造交易（图中的 Tx）实现双重支付，这就需要一个机制来保证。这个机制就是 POW 工作量证明，简单讲就是提升作恶的成本，当然这样也让正常的操作要付出更多的计算成本，初学者大可不必去了解哈希现金原理，这里我来举个更简单的例子，我们经常会在论坛发现各种机器灌水帖子，只需要启动程序就可以简单实现对于正常内容的干扰（可以视为攻击），这些东西干扰了我们的正常阅读，一般论坛会引入人机识别机制，就是常见的图片验证码，包括 12306 使用的更复杂图片验证码，这些验证码你可以想象成是一种提升作恶成本的机制，所不同的

是人识别这些东西会很快，而机器是识别需要更复杂的算法或者在某些领域这些机器识别技术还无法完成，这就完成了“真人证明”，这个机制的后果就是，恶意发帖的人，说能发帖的频率和真人一样了，那么在一个大多数人正常的论坛，干扰信息就是少数，保证了正常信息的发布。

比特币的 POW 证明思路一样，需要一种机制来保证正常节点和恶意节点付出一样的成本来生成区块，而不同于“真人证明”的是，我们面对的都是运行的程序，这就牵涉到如何保证正常的程序和恶意程序付出一样的多的成本，程序的复制和传输不需要太多的成本，要让程序运行最大成本就是硬件（CPU、内存）和消耗（电力），所以**比特币引入 SHA256 算法**（你可以不用了解原理），你只需要知道进行 SHA256 运算需要消耗大量的 CPU 时间和电力，这不像进行加法运算那样在一瞬间就可以完成。POW 利用了信息安全的另一个特点，就是安全和成本相关性，这是我以前文章一再强调的，这个世界没有绝对安全一说。安全有三个延伸层面可管理、可察觉、攻击成本，比特币的系统利用了其中两个可管理与攻击成本，通过原文**“随机散列值（hash 值）以一个或多个 0 开始。那么随着 0 的数目的上升,找到符合条件的这个解所需要的工作量将呈指数增长”**实现可以管理，通过 SHA256 碰运气实现攻击成本提升，所以比特币并不关注可察觉性，你可以作恶但这不重要，因为没有人会花 100 块钱去获取 1 块钱，这属于经济学原理。

说个题外话我以前看过一篇生物学文章，讲的是在进化过程中其实是存在一种可以无限分裂的长寿细胞，也就是说存在过一种长生不老的生物，为什么这种生物不存在了呢，是因为老的生物永远不死，挤压了新生代的生存空间，同时没有变异最后被自然界淘汰，而现存的癌细胞是典型的可以无限分裂的细胞遗留，

结果是癌细胞无限分裂最后导致个体死亡，癌细胞也就 over 了。

这和比特币系统多么类似啊，如果一个 51%攻击达成，攻击者会达成目的，但是整个系统的价值会崩溃掉，那么攻击中获取的比特币也没有任何意义了，攻击者付出的大量硬件投资和电费就没有意义了，所以发动 51%攻击的人要考虑的是，是否要把自己变成癌细胞最后导致系统死亡。

本节的后两段阐述了为什么不使用**基于 IP 地址**的投票权问题（IP 地址共识机制），原因很简单因为 IP 地址分配是一个中心化问题（IP 地址的分配受 ICP 控制，政府组织可以很轻易回收和控制这些 ip 地址资源），可以看出中本聪设计的一切都在为去中心化考虑，包括很多细节。**其实基于 CPU 也是有缺陷的**，但在当时来看中本聪的设计还是非常完美的，毕竟中本聪不是神，比特币诞生后的大量竞争币也是着重改良了 **POW 机制包括引入 POS 机制**力求完善数字货币，到目前为止，我认为前期 POW 机制+后期 POS 机制是一个比较好的选择（个人观点），可以有效避免算力集中实现藏富于民；这也是为什么现在比特币 BU 的争议需要争取获得大量矿场的支持，而普通比特币用户基本上属于吃瓜群众的原因了，等什么时候一个数字货币需要听取每一个用户的声音时，那么 DAC 就真的实现了。

本节白皮书传达的另一个重要信息时，根据比特币“大多数人正义”的设计，比特币区块链只认可最长的那条链（交易记录），因为这是一开始正义的人构造的，后来者需要付出指数级的成本才能更改，验证了 POW 的有效性。另外白皮书中“另一个问题是，硬件的运算速度在高速增长，且节点参与网络的程度会有所起伏”需要了解，也就是说中本聪在设计的时候考虑今后几年硬件摩尔定律的增长，可能到某一天具有海量运算能力的硬件会非常便宜，以前需要计算一天的

工作量可能 1 秒就完成；中本聪的解决方案非常巧妙，**算力的增长会让区块生成的数据加快，那么我们就要让区块的增长和计算难度相关**，通过前一节的“**碰撞 SHA 算法的 0**”来实现，越到后来你需要找到的连续 0 的数量就越多，根据 SHA 算法原理，你需要消耗的计算能力也是成指数增长的，这样就抵消了硬件计算成本的下滑，也就是说，你以前花 1000 块钱买个 486 机器可以干的事情，你现在还是需要花 1000 块钱买个小型机才能干。

本节白皮书最需要理解的就是经济学的成本概念，理解了计算、作恶和成本相关，同时这个成本也是考虑到单位计算能力成本的不断下降与抵消，你就理解了比特币 POW 的核心思想。

4.3.2 解读 2

为了保证平均 10 分钟左右只有一个人可以记账，就必须要提高记账的难度，使得寻找的 Hash 值必须以若干个 0 开头。同是为了满足这个条件，在进行 Hash 时引入一个随机数变量，使得

Hash(区块头) = 满足要求个数的 0 开头的 hash 值

区块头 = 区块版本 + 前一个区块的 hash 值 + merkel root + bits + timestamp + nonce



注：在比特币中区块的 hash 值就是指该区块头的 hash 值

我们知道改变 Hash 的原始信息的任何一部分，Hash 值也会随之不断的变化，因此在运算 Hash 时，**不断的改变随机数的值，总可以找的一个随机数使的 Hash 的结果以要求的若干个 0 开头**（下文把这个过程称为猜谜），率先找到随机数的节点就获得此次记账的唯一记账权。

我们简单分析下记账难度有多大，Hash 值（以 SHA256 为例）的十六进制是由数字（0-9）和字母（a~f）构成的字符串，每一位有 16 种可能性，假设任何一个字符出现的概率是均等的，那么第一位为 0 的概率是 1/16（其他位出现什么字符先不管），理论上需要尝试 16 次 Hash 运算才会出现一次第一位为 0 的情况，如果前两 2 位为 0，就得尝试 16 的平方次 Hash 运算，以 n 个 0 开头就需要尝试 16 的 n 次方次运算。我们结合当前实际区块#512676 信息来看看：

下图来源于网站：<https://blockchain.info/>

Block #512676

Summary		Hashes	
Number Of Transactions	937	Hash	000000000000000000000040e8b4d29b752798599ee8100bdc496648562b3a1722db
Output Total	3,488.69437145 BTC	Previous Block	000000000000000000000053031e0641d088d389e6a33c7e1332e98e35250720764d
Estimated Transaction Volume	726.38726649 BTC	Next Block(s)	000000000000000000000015c03efb15b04bc08b493f0d693471248f14328143d030
Transaction Fees	0.28745011 BTC	Merkle Root	e73f8fa8774c1b6cc31c7e9a0245b5a8a4ddb5d9494c0e0c2720c16e2cdcb26
Height	512676 (Main Chain)	 <p>Be Your Own Bank. Use your Blockchain wallet to buy bitcoin now. GET STARTED →</p> <p> BLOCKCHAIN</p>	
Timestamp	2018-03-09 01:52:06		
Received Time	2018-03-09 01:52:06		
Relayed By	58COIN		
Difficulty	3,290,605,988,755		
Bits	391481763		
Size	342.18 kB		
Weight	1236.915 kWU		
Version	0x20000000		
Nonce	793078603		
Block Reward	12.5 BTC		

当前区块的 hash 值为：

000000000000000000000040e8b4d29b752798599ee8100bdc496648562b3a1722db

共包含 18 个 0 开头。

随机数（nonce）：793078603

区块的回报（block reward）：12.5 个比特币

Height：表示当前的区块

我们可以看到 Hash 值以 18 个 0 开头，理论上需要尝试 16^{18} 次方次，这个数是非常非常巨大的，我已经算不清楚了，应该是亿亿级别以上了。如此大的计算量需要投入大量的计算设备、电力等，目前应该没有单矿工独立参与挖矿了，基本都是由矿工联合起来组成矿池进行挖矿（矿池里的矿工按算力百分比来分收益）。

从经济的角度讲，只有挖矿还有收益（比特币价格不断上涨也让收益变大），就会有新的矿工加入，从而加剧竞争，提高算力难度，挖矿就需要耗费更多的运算和电力，相互作用引起最终成本会接近收益。

在节点成功找到**满足的 Hash 值**之后，会马上对全网进行广播打包区块，网络的节点收到广播打包区块，会立刻对其进行验证。

如果验证通过，则表明已经有节点成功解谜，自己就不再竞争当前区块打包，而是选择接受这个区块，记录到自己的账本中，然后进行下一个区块的竞争猜谜。

网络中只有最快解谜的区块，才会添加到账本中，其他的节点进行复制，这样就保证了整个账本的唯一性。

假如节点有任何的作弊行为，都会导致网络的节点验证不通过，直接丢弃其打包的区块，这个区块就无法记录到总账本中，作弊的节点耗费的成本就白费了，因此在巨大的挖矿成本下，也使得矿工自觉自愿的遵守比特币系统的共识协议，也就确保了整个系统的安全。

矿工的收益其实不仅仅包含新发行的 12.5 比特币奖励，同时还有交易费收益（本文忽略一些细节是为了让主干更清晰）。

5. Network (网络)

5.1 原文

The steps to run the network are as follows:

- 1) New transactions are broadcast to all nodes.
- 2) Each node collects new transactions into a block.
- 3) Each node works on finding a difficult proof-of-work for its block.
- 4) When a node finds a proof-of-work, it broadcasts the block to all nodes.
- 5) Nodes accept the block only if all transactions in it are valid and not already spent.
- 6) Nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash.

Nodes always consider the longest chain to be the correct one and will keep working on extending it. If two nodes broadcast different versions of the next block simultaneously, some nodes may receive one or the other first. In that case, they work on the first one they received, but save the other branch in case it becomes longer. The tie will be broken when the next proof-of-work is found and one branch becomes longer; the nodes that were working on the other branch will then switch to the longer one.

New transaction broadcasts do not necessarily need to reach all nodes. As long as they reach many nodes, they will get into a block before long. Block broadcasts are also tolerant of dropped messages. If a node does not receive a block, it will request it when it receives the next block and realizes it missed one.

5.2 翻译

运行该网络的步骤如下：

- 1)新的交易（transaction）向全网节点进行广播；
- 2)每一个节点都将收到的交易信息纳入一个区块中；
- 3)每个节点都尝试为自己的区块中找到一个具有足够难度的工作量证明；
- 4)当一个节点找到了一个工作量证明（pow），它就向全网进行广播；
- 5)当且仅当包含在该区块中的所有交易都是有效的且之前未存在过的，其他节点才认同该区块的有效性；
- 6)其他节点表示他们接受该区块，而表示接受的方法，则是在跟随该区块的末尾，

制造新的区块以延长该链条，而新区块的随机散列值（hash）则基于上一个区块的随机散列值。

节点始终都将最长的链条视为正确的链条，并持续工作和延长它。如果有两个节点同时广播不同版本的新区块，那么其他节点在接收到该区块的时间上将存在先后差别。当此情形，他们将在率先收到的区块基础上进行工作，**但也会保留另外一个链条，以防后者变成最长的链条**。该僵局（tie）的打破要等到下一个工作量证明被发现，而其中的一条链条被证实为是较长的一条，那么在另一条分支链条上工作的节点将转换阵营，开始在较长的链条上工作。

所谓“新的交易要广播”，实际上不需要抵达全部的节点。只要交易信息能够抵达足够多的节点，那么他们将很快被整合进一个区块中。而区块的广播对被丢弃的信息是具有容错能力的。如果一个节点没有收到某特定区块，那么该节点将会发现自己缺失了某个区块，也就可以提出自己下载该区块的请求。

5.3 解读

本节需要注意的是，白皮书里面没有具体说明交易是如何向全网进行广播的，根据找到的资料**比特币使用了类似 DHT 网络的 P2P 网络协议进行无中心化的网络连接**，这种协议在比特币出现前已经被广泛使用，Kad，eMule、BT 等都使用了类似的协议，这可能是中本聪觉得这不是比特币主要特点，没有在白皮书里阐述的原因吧。比特节点通常采用 **TCP 协议、使用 8333 端口**（该端口号通常是比特币所使用的，除 8333 端口外也可以指定使用其他端口）与已知的对等节点建立连接。你可以理解为有一种机制可以让所有运行比特币代码的节点收到交易广播，而我们需要关心的是，收到交易后要做的事情，这里面有很多问题要解决。

首先，“每一个节点都将收到的交易信息纳入一个区块中”，这也就是区块中包含的交易记录。

第二个问题是，那么谁有权利记录这个数据呢，准确的说是一组时序相关的交易数据，这就用到了上一篇说的 POW 工作量证明，这就是文中的“每个节点都尝试为自己的区块中找到一个具有足够难度的工作量证明”，只有“运气”足够好的节点找到**那个包含多个 0 的 HASH 值**得节点可以生成这个区块（矿工就是干这事），然后就可以把这个区块广播出去了。

问题又来了，万一有两个节点“同时”运气不错，都找到了相同的区块 hash，从时间原理上来说，是不可能存在“同时”的，总有先后顺序，哪怕是毫秒级的差距或者是网络广播造成的延时，这是个哲学问题“你不可能同时踏进两条相同的河流”，一旦有节点收到了这个区块的广播，那么节点就会进行验证“**当且仅当包含在该区块中的所有交易都是有效的且之前未存在过的，其他节点才认同该区块的有效性**”；验证通过后，这个节点就不会再接受**别的节点的同样区块**了，需要注意的是，同时这个节点会终止自己正在进行的包含同样交易的区块计算，也就说不会在进行无用功了，这些节点就会在这个区块基础上启动新的交易区块计算，如此往复，形成链条。

那么问题又又来了，因为网络的复杂性，如果有两个节点同时广播**不同版本的新区块**，那么其他节点在接收到该区块的时间上将存在先后差别。在这种情形下，他们将在率先收到的区块基础上进行工作，但也会保留另外一个链条（分叉），**“该僵局的打破要等到下一个工作量证明被发现”**，通过一段时间的运行，总有一条区块链条是时序上最长的（还记得前面说过的时序相关吗），那么到最后长的那条就是最终被认可的链条，比特币的区块链条就是在不停的分叉、抛弃、又分

又、又合并的过程中，最终最长的那条才是系统认可的区块链。多么象贪吃蛇游戏啊，看谁最后最长。

其实问题又又来了，如果有一大批节点（比如矿池），就是要和别人不一样，修改了代码，就是要创造一个他们自己的最长的链条呢，这就是比特币的里面的"51%攻击"。51%攻击是理论上的值，而本人认为，不需要控制 51%的算力，除非剩下的 49%保持绝对的正义，只要作恶的节点大于正义节点就可以，也就是说，在一个假定的环境，大家组团作恶形成作恶集团，其中一个集团控制的算力最多即可，比如 A 集团控制 30%，B 集团控制 25%，C 集团控制 25%，正义控制 20%，那么最终 A 集团会获胜，这完全是从技术理论出发，一旦形成这种局面，其实就是硬分叉，整个体系会分裂，形成 ABC 三种新币，那么根据经济学理论，最后整体价值下降对谁都没有好处。之所以说这些是让初学者明白 51%攻击是理论值。

6. Incentive（激励机制）

6.1 原文

By convention, the first transaction in a block is a special transaction that starts a new coin owned by the creator of the block. This adds an incentive for nodes to support the network, and provides a way to initially distribute coins into circulation, since there is no central authority to issue them. The steady addition of a constant amount of new coins is analogous to gold miners expending resources to add gold to circulation. In our case, it is CPU time and electricity that is expended.

The incentive can also be funded with transaction fees. If the output value of a transaction is less than its input value, the difference is a transaction fee that is added to the incentive value of the block containing the transaction. Once a predetermined number of coins have entered circulation, the incentive can transition entirely to transaction fees and be completely inflation free.

The incentive may help encourage nodes to stay honest. If a greedy attacker is able to assemble more CPU power than all the honest nodes, he would have to choose between

using it to defraud people by stealing back his payments, or using it to generate new coins. He ought to find it more profitable to play by the rules, such rules that favour him with more new coins than everyone else combined, than to undermine the system and the validity of his own wealth.

6.2 翻译

我们约定如此：每个区块的第一笔交易进行特殊化处理，该交易产生由该区块创造者拥有的新的电子货币。这样就增加了节点支持该网络的激励，并在没有中央集权机构发行货币的情况下，提供了一种将电子货币分配到流通领域的一种方法。这种将一定数量新货币持续增添到货币系统中的方法，非常类似于耗费资源去挖掘金矿并将黄金注入到流通领域。此时，CPU 的时间和电力消耗就是消耗的资源。

另外一个激励的来源则是交易费 (transaction fees)。如果某笔交易的输出值小于输入值，那么差额就是交易费，该交易费将被增加到该区块的激励中。只要既定数量的电子货币已经进入流通，那么激励机制就可以逐渐转换为完全依靠交易费，那么本货币系统就能够免于通货膨胀。

激励系统也有助于鼓励节点保持诚实。如果有一个贪婪的攻击者能够调集比所有诚实节点加起来还要多的 CPU 计算力，那么他就面临一个选择：要么将其用于诚实工作产生新的电子货币，或者将其用于进行二次支付攻击。那么他就会发现，按照规则行事、诚实工作是更有利可图的。因为该等规则使得他能够拥有更多的电子货币，而不是破坏这个系统使得其自身财富的有效性受损。

6.3 解读

第一段终于说明前面一直回避的一个问题：货币从哪来？在 bitcoin 的系统

中，中本聪已经规定了总量就是 2100W 个 bitcoin。而这些 bitcoin 的产生是每产生 210000 个区块就减半(以现在约定大约每 10 分钟产生一个区块的速度大约到 2140 年产生为 0)。而 bitcoin 产生的方法就是给抢到记账权的人凭空给予一定量的货币，这样就同时解决的货币发行和矿工记账的奖励(就像现实中挖黄金的黄金矿工一样)。这种**凭空奖励**的方法，就是每个区块的第一交易，是一个特殊的交易，这个交易就是只有 output，且对于输入这个 output 的 input(前一个交易的 output)是个空(后文会提到)。就像 A 给 B 100 块钱，在记录这个信息的时候可以记录 A 的“账户-100，B 的账户+100”这个事物，也可以记录“A 给了 B 100 块钱”这一条交易信息。区块链选择了后者。而这个凭空出现的钱，就是一条特殊的交易信息。

说句实话，在没有看到白皮书时，我当时单纯的认为比特币的激励机制是为了解决算力问题，因为在此之前的 p2p 系统都没有完善的激励机制，NASA 的寻找外星人计划，基本上依靠的是纯粹的兴趣，利用对一件事物的热情分享你的 CPU 算力，但是热情很难持续的保持，最终知道这个分布式计算项目的人都是激情爱好者，包括我自己后来也不再运行这段程序了。最类似的激励探索我认为是 PT 分享机制，混过 PT 站的人都知道，不同于 BT 分享，PT 站有着严格的制度保障有足够多的人分享他们的带宽和资源，PT 是一种以你需要的资源为代价的带宽激励机制，你必须有足够的上传量才能获取下载量，这种激励制度保障了 PT 站点的长期存在，因为激励规则的存在 PT 站点比单纯的依靠道德建立起来的 BT 站点活的更久，资源更丰富，系统良性循环更有效。

说以上这些的目的，就是为了告诉您，经过以前积累的经验 and 探索，**不要试图去建立一个没有激励机制的去中心化系统，这已经被证明是行不通的**。要维持

一个 p2p 系统的运作，必须要有足够多的节点参与，且这种参与是持久性的而非激情，激励机制可以形成一种纽带关系，而最好的激励机制莫过于直接的经济利益了，以利益纽带维持的系统是最牢固的；纵观大多 p2p 系统的发展，大部分都是以激情开始以疲劳结束，中本聪肯定看到了这一点，他希望的是“以激情开始，以利益维持”。事实上他的设计做到了，所谓以激情开始，就是系统刚开始运行时，需要在小范围内传播，依靠爱好者或激情兴趣者运行，逐步的通过利益激励传播到更多参与者中，最后形成一种规则共识，保障所有参与人的权利和利益。

原文中“**每个区块的第一笔交易进行特殊化处理，该交易产生一枚由该区块创造者拥有的新的电子货币**”，这个设计非常的重要，这个设计其实是阐述了如何开始这个激励，从前面几篇文章我们已经知道比特币系统实际上是一个非常依赖 CPU 算力的系统，需要大量的运算进行 **POW 和交易确认**，激励机制的设计最重要的就是要维持系统核心需求，BT 下载需要的是带宽和存储，所以 PT 的激励机制和带宽相关，而比特币需要的是计算，所以自然的就必须要把计算和激励挂钩，这就是为什么比特系统算力越大获得的收益越多的原因。

白皮书另一个说明的就是“**交易费**”制度，有了前面的分析你也就知道了为什么有这个设计了，原因就是交易的确认也需要足够多的节点参与和运算，如果你设计的系统交易时不依靠大量运算那么你大可不必设计“交易费”激励制度了。白皮书中“**那么激励机制就可以逐渐转换为完全依靠交易费，那么本货币系统就能够免于通货膨胀**”，我认为是存在风险的。比特币的发行是总量控制的，这种设计的目的是为了防止货币的通胀不假，但是有一个风险需要注意，这需要有矿机的同学进行一下计算，一旦比特币进入后期，需要单位挖矿的电力成本超过比特币价值时矿场就无法通过挖矿盈利，结果就是矿场只能靠交易盈利，但是如果比

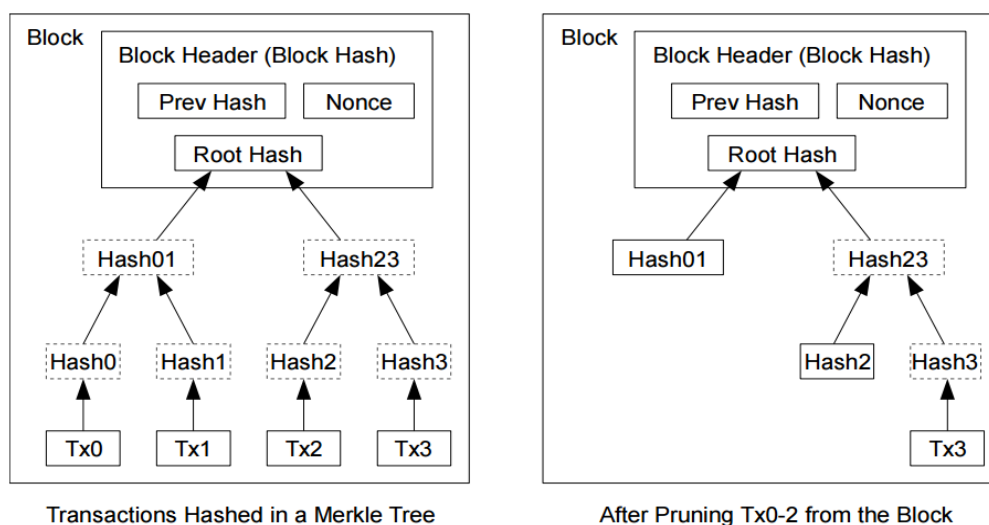
特币不能有有效的大规模商业应用，会出现一个问题，就是没有足够的交易量维持计算所需的“交易费”，计算节点就会萎缩，导致交易时间过长，只能不断提升“交易费”，最终导致整个系统崩盘，这绝对不是危言耸听，所以目前比特币进行的闪电网络包括 BU 分叉都是为了解决这个问题，越到后期越要注意交易成本和确认效率。

最后介绍原文中的“**激励系统也有助于鼓励节点保持诚实**”，我觉得比较牵强，我读到这里是也是反复思考和假设的，如果作恶者认同比特币的价值这个设计是合理的，但是通过如果攻击者或做恶者拥有绝对多数算力，他可以创造新的规则（新的货币种类），而不是维持原有的比特币，如果新的规则获利更大，那么这个激励机制是无法保证算力绝对优势节点的诚实的，否则目前国际上对于比特币算力过于集中于中国矿池产生的担心也不是空穴来风的了。

7. Reclaiming Disk Space（回收磁盘空间）

7.1 原文

Once the latest transaction in a coin is buried under enough blocks, the spent transactions before it can be discarded to save disk space. To facilitate this without breaking the block's hash, transactions are hashed in a Merkle Tree [7][2][5], with only the root included in the block's hash. Old blocks can then be compacted by stubbing off branches of the tree. The interior hashes do not need to be stored.



A block header with no transactions would be about 80 bytes. If we suppose blocks are generated every 10 minutes, $80 \text{ bytes} \times 6 \times 24 \times 365 = 4.2\text{MB}$ per year. With computer systems typically selling with 2GB of RAM as of 2008, and Moore's Law predicting current growth of 1.2GB per year, storage should not be a problem even if the block headers must be kept in memory.

7.2 翻译

如果最近的交易已经被纳入了足够多的区块之中，那么就可以丢弃该交易之前的数据，以回收硬盘空间。为了同时确保不损害区块的随机散列值（hash），交易信息通过执行 hash 构建成为一种 Merkle 树（Merkle tree）的形态，使得只有根(root)被纳入了区块的随机散列值（hash）。通过将该树（tree）的分支拔除（stubbing）的方法，老区块就能被压缩。而内部的随机散列值（hash）是不必保存的。

不包含交易的区块头大约 80 字节。我们假设每 10 分钟生成一次区块，每年大小为 $80 \text{ bytes} \times 6 \times 24 \times 365 = 4.2\text{MB}$ 。2008 年 PC 系统通常的内存容量为 2GB，按照摩尔定理预言的每年增长 1.2GB 的大小，即使将全部的区块头存储在内存之中都不是问题。

7.3 解读

本节是针对区块链系统会不断产生区块导致占用空间变大问题的一个解决方案。如截止 2017/1/28，区块链总数已经超过了 90G，虽然存储是越来越不值钱了，但是要普通公众使用是不可能的，因为信息在一直膨胀。这里就体现出区块链系统的精妙之处，区块头不存储交易详情，而是使用 Merkel Hash Tree 的方式存储 Root Hash，达到“0 知识证明”。个人并不一定需要这个区块，而是具有这个区块的“hash”就足够了，有 IPFS，公共节点，信任度高节点帮助存储这些区块。”0 知识证明“保证了区块是绝对正确的而不是伪造的。

从上图中，我们可以看出，**区块分为区块头（block header）和区块体。**

区块头包括了版本、上一个区块的 hash、nonce、bits（目标值）、timestamp（时间戳）及 merkel root hash 等信息

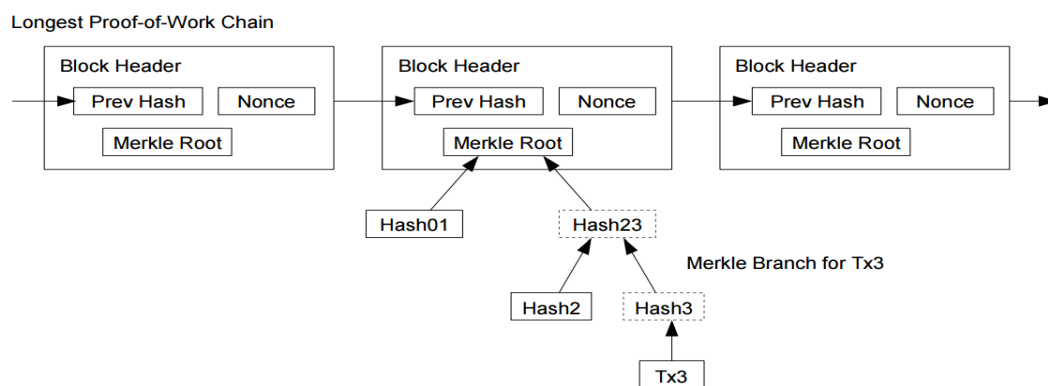
区块体包括当前区块经过验证的、区块创建过程中生成的所有交易记录。这些记录通过 Merkle 树的哈希过程生成唯一的 Merkle 根并记入区块头。

区块哈希值（hash）实际上就是**区块头的哈希值**，只有区块头被用于计算 hash，常说的区块哈希值实际是区块头哈希值，它可以用来唯一、明确地标识一个区块。

8. Simplified Payment Verification (简化的支付验证)

8.1 原文

It is possible to verify payments without running a full network node. A user only needs to keep a copy of the block headers of the longest proof-of-work chain, which he can get by querying network nodes until he's convinced he has the longest chain, and obtain the Merkle branch linking the transaction to the block it's timestamped in. He can't check the transaction for himself, but by linking it to a place in the chain, he can see that a network node has accepted it, and blocks added after it further confirm the network has accepted it.



As such, the verification is reliable as long as honest nodes control the network, but is more vulnerable if the network is overpowered by an attacker. While network nodes can verify transactions for themselves, the simplified method can be fooled by an attacker's fabricated transactions for as long as the attacker can continue to overpower the network. One strategy to protect against this would be to accept alerts from network nodes when they detect an invalid block, prompting the user's software to download the full block and alerted transactions to confirm the inconsistency. Businesses that receive frequent payments will probably still want to run their own nodes for more independent security and quicker verification.

8.2 翻译

在不运行完整网络节点的情况下，也能够对支付进行检验。一个用户需要保留最长的工作量证明链条的区块头的拷贝，它可以不断向网络发起询问，直到它确信自己拥有最长的链条，并能够通过 merkle 的分支通向它被加上时间戳并纳入区块的那次交易。节点想要自行检验该交易的有效性是不可能的，但可以通过

追溯到链条的某个位置，它就能看到某个节点曾经接受过它，并且于其后追加的区块也进一步证明全网曾经接受了它。

因此，只要诚实的节点控制了网络，检验机制就是可靠的。但是，当全网被一个计算力占优的攻击者攻击时，将变得较为脆弱。因为网络节点能够自行确认交易的有效性，只要攻击者能够持续地保持计算力优势，简化的机制会被攻击者焊接的 (fabricated) 交易欺骗。那么一个可行的策略就是，只要他们发现了一个无效的区块，就立刻发出警报，收到警报的用户将立刻开始下载被警告有问题的区块或交易的完整信息，以便对信息的不一致进行判定。对于日常会发生大量收付的商业机构，可能仍会希望运行他们自己的完整节点，以保持较大的独立完全性和检验的快速性。

8.3 解读

回收硬盘空间所带来的问题就是简化支付认证的问题，因为有些节点已经不会持有全部区块信息，这里相当于是一个博弈了，使用空间换认证的便捷。但总之是不会有信息的安全性上出问题的。因为只要持有了 hash 作为标识，无论什么节点总是能从其他节点上请求到原始信息。

这种机制感觉有点像是打补丁的方式。但是好像确实是要削减硬盘存储的一个解决方案，是存储与安全便捷的一个博弈结果。很可能有机会在这里做文章，或许是区块链运用推广的一个障碍。因为要是保留区块的节点太少了就有可能造成问题。虽然不能窃取篡改网络，但是却可导致崩溃。(当然不是说 bitcoin 网络，是说一些小型区块链网络)。

SPV (简单支付认证) 节点只下载区块头，不下载包含在每个区块中的交易

信息。这样的不含交易信息的区块链，大小只有完整区块链的几千分之 1，那简单支付认证节点是如何验证交易的呢？

简单文件验证：我们通常用哈希来检验下载的文件是否完整，我经常看到这样的下载页面：

Files					
Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		e9180c69ed9a878a4a8a3ab221e32fa9	22673115	SIG
XZ compressed source tarball	Source release		b9c2c36c33fb89bda1fed37ad5af9be	16974296	SIG
Mac OS X 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	ce31f17c952c657244a5cd0cccae34ad	27696231	SIG

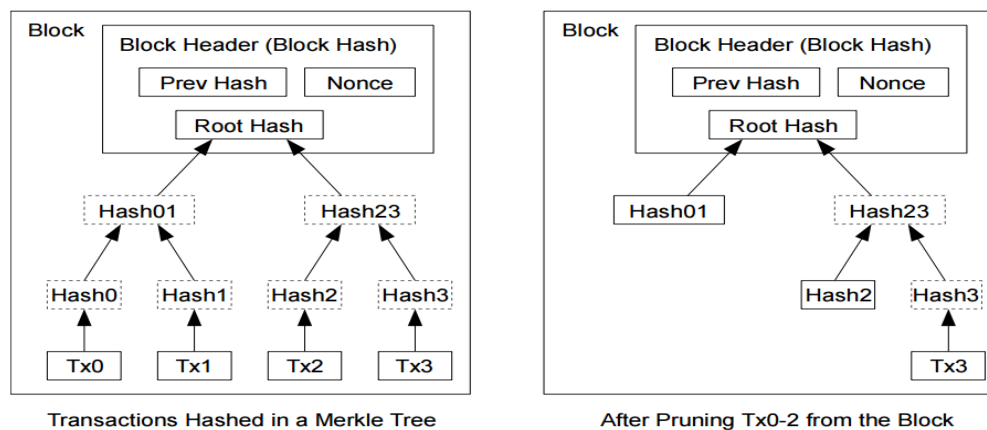
可以看到下载链接后面提供了一个 MD5（MD5 也是一种 Hash 算法），这样我们可以在下载之后对文件计算 MD5，如果 MD5 与提供的 MD5 相等，说明文件有没有被损坏，这个验证过程相信大家都能理解。

多点文件验证(哈希列表)：现在复杂度提高一点，在 P2P 网络中下载时，会把大文件切成小文件，同时从多个机器上下载数据，这个时候怎么验证数据呢？以 BT 下载为例，在下载真正的数据之前，我们会先下载一个哈希列表的（每个下小块计算出一个哈希），如果有一个小块数据在传输过程中损坏了，那我只要重新下载这一个数据块就行了，这时有一个问题就出现了，那么多的哈希，怎么保证它们本身(哈希列表中的哈希值)都是正确地呢？答案是把每个小块数据的哈希值拼到一起，然后对这个长字符串在作一次哈希运算，得到哈希列表的根哈希。只要根哈希校对比一样就说明哈希列表是正确的，再通过哈希列表校验小数据块，如果所有的小数据块验证通过则说明大文件没有被损坏。

Merkle 树：验证交易的过程和文件验证很相似，可以将每个交易看做一个

小数据块，但比特币使用 Merkle 树的方式进行验证，相对于哈希列表，Merkle 树是一种哈希二叉树，它的明显的一个好处是可以单独拿出一个分支来（作为一个小树）对部分数据进行校验，更加高效。我们看下面的区块结构图，区块体就包含这样一个 Merkle 树,Merkle 树被用来归纳一个区块中的所有交易。

每个叶子节点是每个交易信息的哈希，往上对相邻的两个哈希合并成字符串再哈希，继续类似的操作直到只剩下顶部的一个节点，即 Merkle 根，存入区块头。



因为 Merkle 树是二叉树，所以它需要偶数个叶子节点。如果仅有奇数个交易需要归纳，那最后的交易就会被复制一份以构成偶数个叶子节点，这种偶数个叶子节点的树也被称为平衡树。

简化支付验证：简单支付认证节点不保存所有交易也不会下载整个区块，**仅仅保存区块头**，我们来看看它是如何对交易数据进行验证的。

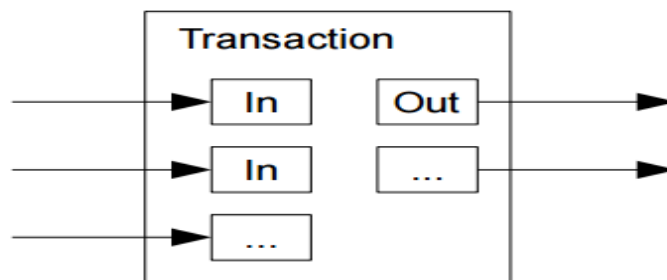
假如要验证区块结构图中交易 2，SPV 节点会通过向相邻节点索要（通过 Merkleblock 消息）包括从交易 2 哈希值沿 Merkle 树上溯至区块头根哈希处的哈希序列（即哈希节点 2, 3, 23, 01, root hash- 称为认证路径）来确认交易的存在

性和正确性。

9. Combining and Splitting Value (组合和分割价值)

9.1 原文

Although it would be possible to handle coins individually, it would be unwieldy to make a separate transaction for every cent in a transfer. To allow value to be split and combined, transactions contain multiple inputs and outputs. Normally there will be either a single input from a larger previous transaction or multiple inputs combining smaller amounts, and at most two outputs: one for the payment, and one returning the change, if any, back to the sender.



It should be noted that fan-out, where a transaction depends on several transactions, and those transactions depend on many more, is not a problem here. There is never the need to extract a complete standalone copy of a transaction's history.

9.2 翻译

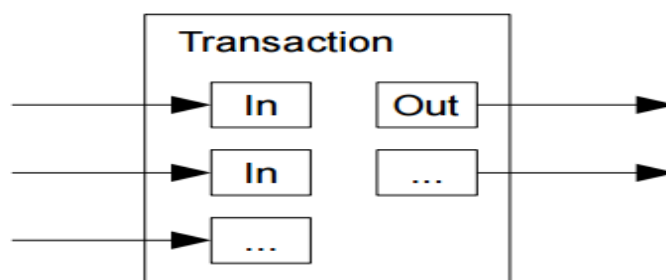
虽然可以独立地对电子货币进行处理, 但是对于每一枚电子货币单独发起一次交易将是一种笨拙的办法。为了使得价值易于组合与分割, **交易被设计为可以包含多个输入和输出**。一般而言是某次价值较大的前次交易构成的单一输入, 或者由某几个价值较小的前次交易共同构成的并行输入, 但是输出最多只有两种类型: 一个类型用于支付, 另一个类型用于找零 (如有)。

需要指出的是, 当一笔交易依赖于之前的多笔交易时, 这些交易又各自依赖

于多笔交易，但这并不存在任何问题。因为这个工作机制并不需要展开检验之前发生的所有交易历史。

9.3 解读

这段终于说到了一个**交易(Tx)**的构成，前面的论述已经说了很多，只要抓住一个关键点：**一个交易是由前面的交易进行验证+转移的数目+目的地构成**。inputs 就是之前的交易的 outputs。对所有的 inputs 进行验证，就可得到该交易的付款者的余额是否大于要转移的数目。inputs 关联的所有 Tx 就是之前的“历史信息”，节点可检索自己的区块获得结果。每个区块的第一个交易(激励交易)就相当于只有 output 而没有 input。



这里要注意的额外一点就是：每次的交易必须被花完，只不过是所有 input 的综合，其中一部分给了收款方，而找零则是全部返回到发送方。这种机制在数目的分割上有天然优势。现有的货币总是有一个最小单位作为“元”单位(比如人民币：1 分)，但是 bitcoin 却没有这样的限制，它只关心差额，而不关心最小元单位。所以这就是价值的组合和分割。

下面我们通过

<https://blockchain.info/block/00000000000000000040e8b4d29b752798599ee8100bdc496648562b3a1722db> 来查看这个区块#512676 包含的交易记录：

Transactions

805128dc60bbd5364b997a4ba0d7fb58170a20da4d1d4c9656f3312e170ea84		2018-03-09 01:52:06
No Inputs (Newly Generated Coins)	➡ 14DjTuAUh87cwRsbU1z6W8hZY6FnEkpLS Unable to decode output address	12.78745011 BTC 0 BTC
		12.78745011 BTC
4bd86b7787ec93eafa267824ffdc6c58b47a467b822fa470144ee896899d898e		2018-03-09 01:49:00
bc1qwqdg6squsna38e46795at95yu9atm8azzmyvc kulcc7kyltccxswvvzej	➡ 3BTnLDTkFqb4ZUIIN6mRjSTb8bX6LdwKByq bc1qwqdg6squsna38e46795at95yu9atm8azzmyvc kulcc7kyltccxswvvzej	0.11 BTC 0.1382 BTC
		0.2482 BTC
693ccafe851945eec4607fe717c111387e928a62875ca709703b64dd12fc66dd		2018-03-09 01:48:16
bc1qwqdg6squsna38e46795at95yu9atm8azzmyvc kulcc7kyltccxswvvzej	➡ 38Lh2RAUKcKJhNjXBrDaCE4PuSBPLoYj3b bc1qwqdg6squsna38e46795at95yu9atm8azzmyvc kulcc7kyltccxswvvzej	0.1 BTC 0.0929327 BTC
		0.1929327 BTC
a8dc9eb57a058a3c129af25637d6040266a828fa11084563b0a40fcd6933df05		2018-03-09 01:48:16
bc1qzjeg3h996kw24zrg69nge97fw8jc4v7v7yznftzk06j3429t52vse9tkp9	➡ 1QCvowRtCfImN4yetG3eGxMVmU2dSJ3L1 bc1qwqdg6squsna38e46795at95yu9atm8azzmyvc kulcc7kyltccxswvvzej	4.3228 BTC 0.08919 BTC
		4.41199 BTC

我们看第一条, 这表示区块的第一条记录, 也就是比特币产生的来源, 将 12.5 个的比特币发给第一个找到区块合适 hash 的人。我们接下来看第二条:

4bd86b7787ec93eafa267824ffdc6c58b47a467b822fa470144ee896899d898e		2018-03-09 01:49:00
bc1qwqdg6squsna38e46795at95yu9atm8azzmyvc kulcc7kyltccxswvvzej	➡ 3BTnLDTkFqb4ZUIIN6mRjSTb8bX6LdwKByq bc1qwqdg6squsna38e46795at95yu9atm8azzmyvc kulcc7kyltccxswvvzej	0.11 BTC 0.1382 BTC
		0.2482 BTC

左侧是发送方, 右侧是接收方。发送方发送了 0.2482 个比特币, 但是发送方接受 0.11 个比特币, 找零 0.1362 个比特币。

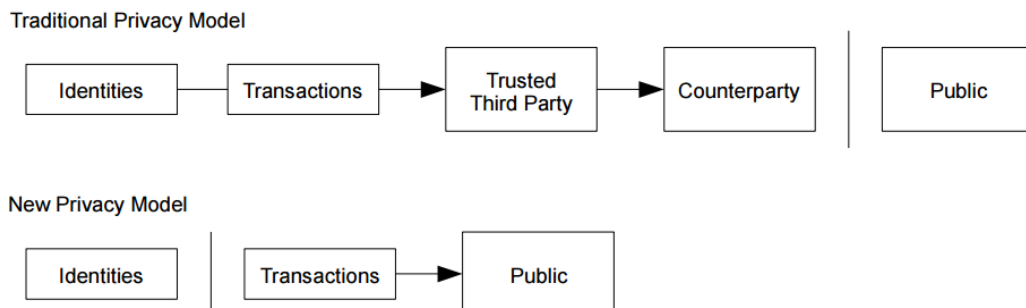
下图是一个小额多输入情形

2151a6881c8fb5ede5a3815fd541f8e1484b4cec5311e3ae77d79c061aee618		2018-03-09 01:51:47
1BEHvzGV8nR1cMX5V2GrYfP89vkgS61GHU 1PKVSJFACRHSJEbnP4bZwWphEJxZJuEUN	➡ 1LBjnUaT5pp36eNE192pxumSh7owYkit2f	0.10894423 BTC
		0.10894423 BTC

10. Privacy (隐私)

10.1 原文

The traditional banking model achieves a level of privacy by limiting access to information to the parties involved and the trusted third party. The necessity to announce all transactions publicly precludes this method, but privacy can still be maintained by breaking the flow of information in another place: by keeping public keys anonymous. The public can see that someone is sending an amount to someone else, but without information linking the transaction to anyone. This is similar to the level of information released by stock exchanges, where the time and size of individual trades, the "tape", is made public, but without telling who the parties were.



As an additional firewall, a new key pair should be used for each transaction to keep them from being linked to a common owner. Some linking is still unavoidable with multi-input transactions, which necessarily reveal that their inputs were owned by the same owner. The risk is that if the owner of a key is revealed, linking could reveal other transactions that belonged to the same owner.

10.2 翻译

传统的银行模型为交易的参与者提供了一定程度的隐私保护，因为试图向可信任的第三方索取交易信息是严格受限的。但是如果将交易信息向全网进行广播，就意味着这样的方法失效了。但是隐私依然可以得到保护：将公钥保持为匿名的方式。公众得知的信息仅仅是有某个人将一定数量的货币发给了另外一个人，但是难以将该交易同特定的人联系在一起，也就是说，公众难以确信，这些人究竟

是谁。这同股票交易所发布的信息是类似的，股票交易发生的时间、交易量是记录在案且可供查询的，但是交易双方的身份信息却不予透露。

作为额外的预防措施，使用者可以让每次交易都生成一个新的密钥对，以确保这些交易不被追溯到一个共同的所有者。但是由于并行输入的存在，一定程度上的追溯还是不可避免的，因为并行输入表明这些货币都属于同一个所有者。此时的风险在于，如果某个人的某一个公钥被确认属于他，那么就可以追溯出此人的其它很多交易。

解读：

这里原文中推荐这样做是相当有道理的。因为椭圆加密算法是能被量子暴力破解的，在整个系统中要是产生的转账公钥是被公开的。虽然从密码学上不应该从公钥推导到私钥，但是还是有例外嘛。所以这里中本聪强烈建议，一次转账就使用一对新的密钥对(因为交易只要被作为 inputs 后那么作为 inputs 的交易也就没用了)，那么就可以保证每次都是新的公私钥进行交易。

后半段是说对于公私钥和现实中的人的对应关系(匿名性失效)的问题，换钥匙同样可以防止这点发生，但是要注意因为所有的记录的全网可查的，所以要是其中一个公私密钥和人对应起来了，那么所有关联该公私密的交易就能和这个人对应起来。

11. Calculations (计算)

11.1 原文

We consider the scenario of an attacker trying to generate an alternate chain faster than the honest chain. Even if this is accomplished, it does not throw the system open to arbitrary changes, such as creating value out of thin air or taking money that never belonged to the

attacker. Nodes are not going to accept an invalid transaction as payment, and honest nodes will never accept a block containing them. An attacker can only try to change one of his own transactions to take back money he recently spent.

The race between the honest chain and an attacker chain can be characterized as a Binomial Random Walk. The success event is the honest chain being extended by one block, increasing its lead by +1, and the failure event is the attacker's chain being extended by one block, reducing the gap by -1.

The probability of an attacker catching up from a given deficit is analogous to a Gambler's Ruin problem. Suppose a gambler with unlimited credit starts at a deficit and plays potentially an infinite number of trials to try to reach breakeven. We can calculate the probability he ever reaches breakeven, or that an attacker ever catches up with the honest chain, as follows [8]:

p = probability an honest node finds the next block

q = probability the attacker finds the next block

q_z = probability the attacker will ever catch up from z blocks behind

$$q_z = \begin{cases} 1 & \text{if } p \leq q \\ (q/p)^z & \text{if } p > q \end{cases}$$

Given our assumption that $p > q$, the probability drops exponentially as the number of blocks the attacker has to catch up with increases. With the odds against him, if he doesn't make a lucky lunge forward early on, his chances become vanishingly small as he falls further behind.

We now consider how long the recipient of a new transaction needs to wait before being sufficiently certain the sender can't change the transaction. We assume the sender is an attacker who wants to make the recipient believe he paid him for a while, then switch it to pay back to himself after some time has passed. The receiver will be alerted when that happens, but the sender hopes it will be too late.

The receiver generates a new key pair and gives the public key to the sender shortly before signing. This prevents the sender from preparing a chain of blocks ahead of time by working on it continuously until he is lucky enough to get far enough ahead, then executing the transaction at that moment. Once the transaction is sent, the dishonest sender starts working in secret on a parallel chain containing an alternate version of his transaction.

The recipient waits until the transaction has been added to a block and z blocks have been linked after it. He doesn't know the exact amount of progress the attacker has made, but assuming the honest blocks took the average expected time per block, the attacker's potential progress will be a Poisson distribution with expected value:

$$\lambda = z \frac{q}{p}$$

To get the probability the attacker could still catch up now, we multiply the Poisson density for each amount of progress he could have made by the probability he could catch up from that point:

$$\sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} \begin{cases} (q/p)^{(z-k)} & \text{if } k \leq z \\ 1 & \text{if } k > z \end{cases}$$

Rearranging to avoid summing the infinite tail of the distribution...

$$1 - \sum_{k=0}^z \frac{\lambda^k e^{-\lambda}}{k!} (1 - (q/p)^{(z-k)})$$

Converting to C code...

```
#include <math.h>
double AttackerSuccessProbability(double q, int z)
{
    double p = 1.0 - q;
    double lambda = z * (q / p);
    double sum = 1.0;
    int i, k;
    for (k = 0; k <= z; k++)
    {
        double poisson = exp(-lambda);
        for (i = 1; i <= k; i++)
            poisson *= lambda / i;
        sum -= poisson * (1 - pow(q / p, z - k));
    }
    return sum;
}
```

Running some results, we can see the probability drop off exponentially with z.

q=0.1

z=0 P=1.0000000

z=1 P=0.2045873

z=2 P=0.0509779

z=3 P=0.0131722

z=4 P=0.0034552

z=5 P=0.0009137

z=6 P=0.0002428

z=7 P=0.0000647

z=8 P=0.0000173

z=9 P=0.0000046

z=10 P=0.0000012


```

q=0.3
z=0 P=1.0000000
z=5 P=0.1773523
z=10 P=0.0416605
z=15 P=0.0101008
z=20 P=0.0024804
z=25 P=0.0006132
z=30 P=0.0001522
z=35 P=0.0000379
z=40 P=0.0000095
z=45 P=0.0000024
z=50 P=0.0000006
Solving for P less than 0.1%...
P < 0.001
q=0.10 z=5
q=0.15 z=8
q=0.20 z=11
q=0.25 z=15
q=0.30 z=24
q=0.35 z=41
q=0.40 z=89
q=0.45 z=340

```

11.2 翻译

设想如下场景：一个攻击者试图比诚实节点产生链条更快地制造替代性区块链。即便它达到了这一目的，但是整个系统也并非就此完全受制于攻击者的独断意志了，比方说凭空创造价值，或者掠夺本不属于攻击者的货币。这是因为节点将不会接受无效的交易，而诚实的节点永远不会接受一个包含了无效信息的区块。一个攻击者能做的，最多是更改他自己的交易信息，并试图拿回他刚刚付给别人的钱。

诚实链条和攻击者链条之间的竞赛，可以用二叉树随机漫步（Binomial Random Walk）来描述。成功事件定义为诚实链条延长了一个区块，使其领先性

+1, 而失败事件则是攻击者的链条被延长了一个区块, 使得差距-1。

攻击者成功填补某一既定差距的可能性, 可以近似地看做赌徒破产问题 (Gambler's Ruin problem)。假定一个赌徒拥有无限的透支信用, 然后开始进行潜在次数为无穷的赌博, 试图填补上自己的亏空。那么我们可以计算他填补上亏空的概率, 也就是该攻击者赶上诚实链条, 如下所示[8] :

p = probability an honest node finds the next block

q = probability the attacker finds the next block

q_z = probability the attacker will ever catch up from z blocks behind

$$q_z = \begin{cases} 1 & \text{if } p \leq q \\ (q/p)^z & \text{if } p > q \end{cases}$$

假定 $p > q$, 那么攻击成功的概率就因为区块数的增长而呈现指数化下降。由于概率是攻击者的敌人, 如果他不能幸运且快速地获得成功, 那么他获得成功的机会随着时间的流逝就变得愈发渺茫。那么我们考虑一个收款人需要等待多长时间, 才能足够确信付款人已经难以更改交易了。我们假设付款人是一个支付攻击者, 希望让收款人在一段时间内相信他已经付过款了, 然后立即将支付的款项重新支付给自己。虽然收款人届时会发现这一点, 但为时已晚。

收款人生成了新的一对密钥组合, 然后只预留一个较短的时间将公钥发送给付款人。这将可以防止以下情况: 付款人预先准备好一个区块链然后持续地对此区块进行运算, 直到运气让他的区块链超越了诚实链条, 方才立即执行支付。当此情形, 只要交易一旦发出, 攻击者就开始秘密地准备一条包含了该交易替代版本的平行链条。

然后收款人将等待交易出现在首个区块中, 然后在等到 z 个区块链接其后。此时, 他仍然不能确切知道攻击者已经进展了多少个区块, 但是假设诚实区块将

耗费平均预期时间以产生一个区块，那么攻击者的潜在进展就是一个泊松分布，分布的期望值为：

$$\lambda = z \frac{q}{p}$$

当此情形，为了计算攻击者追赶上的概率，我们将攻击者取得进展区块数量的泊松分布的概率密度，乘以在该数量下攻击者依然能够追赶上的概率。

$$\sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} \cdot \begin{cases} \left(\frac{q}{p}\right)^{(z-k)} & \text{if } k \leq z \\ 1 & \text{if } k > z \end{cases}$$

化为如下形式，避免对无限数列求和：

$$1 - \sum_{k=0}^z \frac{\lambda^k e^{-\lambda}}{k!} \cdot \left(1 - \left(\frac{q}{p}\right)^{(z-k)}\right)$$

写为如下 C 语言代码：

```
#include double AttackerSuccessProbability(double q, int z)

{

double p = 1.0 - q;

double lambda = z * (q / p);

double sum = 1.0;

int i, k;

for (k = 0; k <= z; k++)

{

double poisson = exp(-lambda);
```

```

for (i = 1; i <= k; i++)

    poisson *= lambda / i;

sum -= poisson * (1 - pow(q / p, z - k));

}

return sum;

}

```

对其进行运算，我们可以得到如下的概率结果，发现概率对 z 值呈指数下降。

当 $q=0.1$ 时

$z=0$ $P=1.0000000$

$z=1$ $P=0.2045873$

$z=2$ $P=0.0509779$

$z=3$ $P=0.0131722$

$z=4$ $P=0.0034552$

$z=5$ $P=0.0009137$

$z=6$ $P=0.0002428$

$z=7$ $P=0.0000647$

$z=8$ $P=0.0000173$

$z=9$ $P=0.0000046$

$z=10$ $P=0.0000012$

当 $q=0.3$ 时

$z=0$ $P=1.0000000$

$z=5$ $P=0.1773523$

$z=10$ $P=0.0416605$

$z=15$ $P=0.0101008$

$z=20$ $P=0.0024804$

$z=25$ $P=0.0006132$

$z=30$ $P=0.0001522$

$z=35$ $P=0.0000379$

$z=40$ $P=0.0000095$

$z=45$ $P=0.0000024$

$z=50$ $P=0.0000006$

求解令 $P<0.1\%$ 的 z 值：

为使 $P<0.001$ ，则

$q=0.10$ $z=5$

$q=0.15$ $z=8$

$q=0.20$ $z=11$

$q=0.25$ $z=15$

$q=0.30$ $z=24$

$q=0.35$ $z=41$

$q=0.40$ $z=89$

$q=0.45$ $z=340$

解读：

这里就是指明，等 6 个区块或者更多是稳妥的方法。

