Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

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.

1. Introduction

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 non-reversible 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.

比特币：点对点电子现金系统

中本聪
satoshin@gmx.com
www.bitcoin.org

抽象的。纯粹的点对点版本的电子现金将允许在线支付直接从一方发送到另一方，而无需通过金融机构。数字签名提供了部分解决方案，但如果仍然需要可信第三方来防止双重支出，那么主要好处就会丧失。我们提出了一种使用点对点网络来解决双重支出问题的解决方案。网络通过将交易散列到持续的基于散列的工作量证明链中来为交易添加时间戳，形成一条记录，如果不重做工作量证明就无法更改。最长的链不仅可以证明所见证的事件序列，而且可以证明它来自最大的 CPU 算力池。只要大部分 CPU 能力由不合作攻击网络的节点控制，它们就会生成最长的链并超过攻击者。网络本身需要最少的结构。消息以尽力而为的方式广播，节点可以随意离开和重新加入网络，接受最长的工作量证明链作为它们离开时发生的事情的证明。

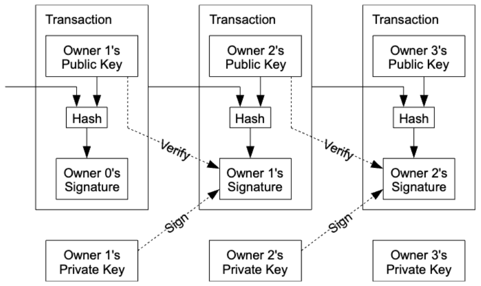
1. 介绍

互联网上的商务几乎完全依赖金融机构作为受信任的第三方来处理电子支付。虽然该系统对于大多数交易来说运行良好，但它仍然存在基于信任的模型的固有弱点。完全不可逆的交易实际上是不可能的，因为金融机构无法避免调解纠纷。调解成本增加了交易成本，限制了最小实际交易规模并切断了小额临时交易的可能性，并且丧失对不可逆服务进行不可逆支付的能力会带来更广泛的成本。随着逆转的可能性，对信任的需求也随之扩大。商家必须对他们的顾客保持警惕，骚扰他们，要求他们提供比他们原本需要的更多信息。一定比例的欺诈被认为是不可避免的。这些成本和支付的不确定性可以通过使用实物货币来避免，但不存在在没有可信方的情况下通过通信渠道进行支付的机制。

我们需要的是一种基于密码证明而不是信任的电子支付系统，允许任何两个愿意的一方直接相互交易，而不需要可信的第三方。在计算上无法逆转的交易将保护卖家免受欺诈，并且可以轻松实施常规托管机制来保护买家。在本文中，我们提出了一种解决双花问题的方案，使用点对点分布式时间戳服务器来生成交易时间顺序的计算证明。只要诚实节点共同控制比任何合作的攻击者节点组更多的 CPU 功率，系统就是安全的。

2. Transactions

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.

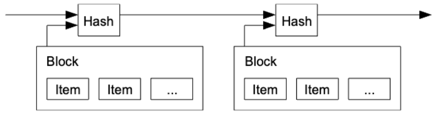


The problem of course is the payee can't verify that one of the owners did not double-spend the coin. A common solution is to introduce a trusted central authority, or mint, that checks every transaction for double spending. After each transaction, the coin must be returned to the mint to issue a new 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.

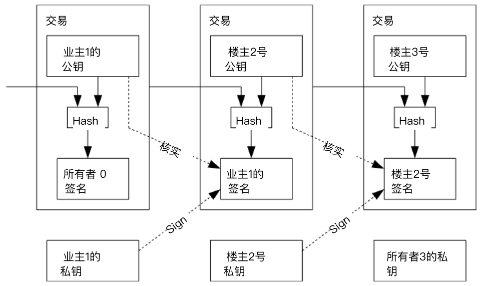
3. Timestamp Server

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 [2-5]. 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.



2. 交易

我们将电子硬币定义为数字签名链。每个所有者通过对前一个交易的哈希值和下一个所有者的公钥进行数字签名并将这些添加到硬币的末尾，将硬币转移到下一个所有者。收款人可以验证签名以验证所有权链。

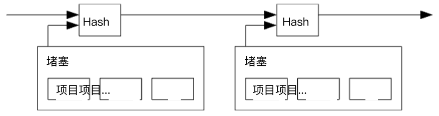


当然，问题是收款人无法验证其中一位所有者没有重复使用该货币。一个常见的解决方案是引入一个值得信赖的中央机构或铸币厂，检查每笔交易是否存在双重支出。每次交易后，硬币必须返回造币厂以发行新的硬币，并且只有直接从造币厂发行的硬币才被信任不会被双重花费。这个解决方案的问题在于，整个货币系统的命运取决于运营铸币厂的公司，每笔交易都必须经过他们，就像银行一样。

我们需要一种方法让收款人知道以前的所有者没有签署任何早期的交易。就我们的目的而言，最早的交易才是最重要的，因此我们不关心后来的双花尝试。确认不存在交易的唯一方法是了解所有交易。在基于铸币厂的模型中，铸币厂了解所有交易并决定哪些交易先到达。为了在没有可信方的情况下实现这一目标，交易必须公开宣布 [1]，并且我们需要一个系统，让参与者就接收顺序的单一历史达成一致。收款人需要证明在每次交易时，大多数节点都同意这是第一个收到的交易。

3. 时间戳服务器

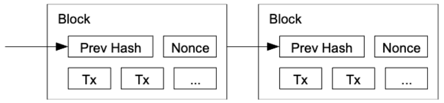
我们提出的解决方案从时间戳服务器开始。时间戳服务器的工作原理是获取要加时间戳的项目块的哈希值并广泛发布该哈希值，例如在报纸或新闻组帖子中[2-5]。显然，时间戳证明数据当时必须存在，才能进入哈希值。每个时间戳在其散列中都包含前一个时间戳，形成一条链，每个附加时间戳都会增强其之前的时间戳。



4. Proof-of-Work

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.

5. Network

The steps to run the network are as follows:

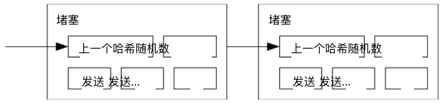
1) New transactions are broadcast to all nodes.
2) Each node collects new transactions into a block.
3) Each node works on finding a difficult proof-of-work for its block.
4) When a node finds a proof-of-work, it broadcasts the block to all nodes.
5) Nodes accept the block only if all transactions in 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.

4. 工作量证明

为了在点对点的基础上实现分布式时间戳服务器，我们需要使用类似于 Adam Back 的 Hashcash [6] 的工作量证明系统，而不是报纸或 Usenet 帖子。工作量证明涉及扫描一个值，该值在进行哈希处理（例如使用 SHA-256）时，哈希值以多个零位开始。所需的平均工作量与所需的零位数量成指数关系，并且可以通过执行单个哈希来验证。

对于我们的时间戳网络，我们通过增加块中的随机数来实现工作量证明，直到找到一个值，该值为块的哈希提供所需的零位。一旦花费了 CPU 的努力来满足工作量证明，则在不重做工作的情况下无法更改块。由于后面的块链接在它后面，因此更改块的工作将包括重做它后面的所有块。



工作量证明还解决了在多数决策中确定代表权的问题。如果大多数是基于一IP地址一票，那么任何能够分配多个IP的人都可以颠覆它。工作量证明本质上是一 CPU 一投票。多数决策由最长的链代表，该链投入了最大的工作量证明努力。如果大部分CPU算力由诚实节点控制，那么诚实链将增长最快并超过任何竞争链。要修改过去的区块，攻击者必须重做该区块及其后所有区块的工作量证明，然后赶上并超越诚实节点的工作量。稍后我们将证明，随着后续块的添加，速度较慢的攻击者追上的可能性呈指数级下降。

为了补偿不断增加的硬件速度和随着时间的推移对运行节点的兴趣的变化，工作量证明难度由针对每小时平均块数的移动平均值确定。如果它们生成得太快，难度就会增加。

5. 网络

运行网络的步骤如下：

1）新交易被广播到所有节点。
2) 每个节点将新交易收集到一个块中。
3) 每个节点都致力于为其区块寻找困难的工作量证明。
4) 当节点找到工作量证明时，它会将块广播到所有节点。
5) 仅当块中的所有交易均有效且尚未花费时，节点才会接受该块。
6) 节点通过使用已接受块的哈希作为前一个哈希来创建链中的下一个块来表达对块的接受。

节点始终认为最长的链是正确的，并将继续努力扩展它。如果两个节点同时广播下一个区块的不同版本，某些节点可能会先接收其中一个。在这种情况下，他们会处理收到的第一个分支，但保存另一个分支，以防它变得更长。当找到下一个工作量证明并且一个分支变得更长时，平局就会被打破；然后，在另一个分支上工作的节点将切换到较长的分支。

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.

6. Incentive

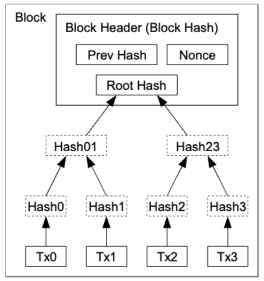
By convention, the first transaction in a block is a special transaction that starts a new coin owned by the creator of the block. This adds an incentive for nodes to support the network, and provides a way to initially distribute coins into circulation, since there is no central authority to issue them. The steady addition of a constant of 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.

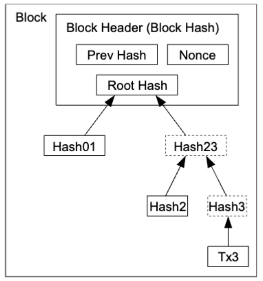
The incentive may help encourage nodes to stay honest. If a greedy attacker is able to assemble more CPU power than all the honest nodes, he would have to choose between using it to defraud people by stealing back his payments, or using it to generate new coins. He ought to find it more profitable to 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.

7. Reclaiming Disk Space

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.



Transactions Hashed in a Merkle Tree



After Pruning Tx0-2 from the Block

A block header with no transactions would be about 80 bytes. If we suppose blocks are generated every 10 minutes, 80 bytes * 6 * 24 * 365 = 4.2MB 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.

新的交易广播不一定需要到达所有节点。只要到达很多节点，不久就会进入区块。块广播还可以容忍丢失的消息。如果一个节点没有收到一个块，它会在收到下一个块并意识到它错过了一个块时请求它。

6. 激励

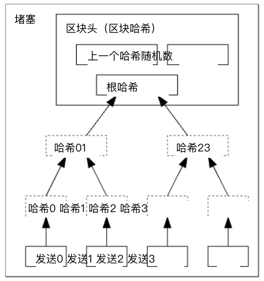
按照惯例，区块中的第一笔交易是一项特殊交易，它启动了该区块创建者拥有的新硬币。这增加了节点支持网络的激励，并提供了一种最初将硬币分配到流通中的方法，因为没有中央机构来发行它们。稳定增加一定数量的新硬币类似于金矿开采者消耗资源以将黄金添加到流通中。在我们的例子中，消耗的是 CPU 时间和电力。

该激励措施也可以通过交易费来资助。如果交易的输出值小于其输入值，则差额就是交易费用，该费用将添加到包含该交易的区块的激励值中。一旦预定数量的代币进入流通，激励措施就可以完全转变为交易费用，并且完全不受通货膨胀影响。

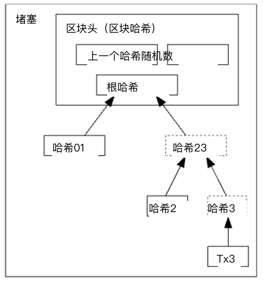
这种激励措施可能有助于鼓励节点保持诚实。如果贪婪的攻击者能够聚集比所有诚实节点更多的 CPU 算力，他将不得不选择是用它来偷回他的付款来欺骗人们，还是用它来生成新的硬币。他应该发现遵守规则更有利可图，这些规则有利于他获得比其他人加起来更多的新硬币，而不是破坏系统和他自己财富的有效性。

7. 回收磁盘空间

一旦硬币中的最新交易被埋在足够多的区块下，之前花费的交易就可以被丢弃以节省磁盘空间。为了在不破坏区块哈希的情况下实现这一点，交易在默克尔树 [7][2][5] 中进行哈希处理，仅根包含在区块哈希中。然后可以通过砍掉树的树枝来压缩旧块。不需要存储内部哈希值。



交易在 Merkle 树中进行哈希处理

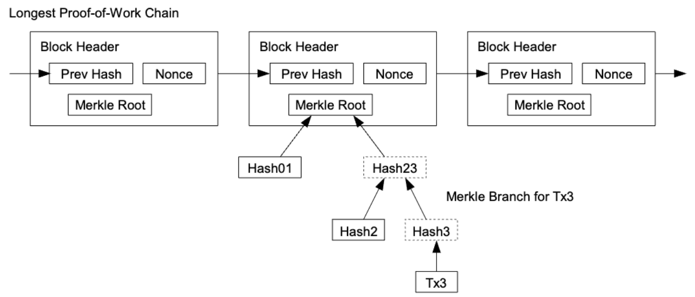


从块中修剪 Tx0–2 之后

没有交易的块头大约为 80 字节。如果我们假设每 10 分钟生成一个块，则每年 80 字节 * 6 * 24 * 365 = 4.2MB。截至 2008 年，计算机系统通常配备 2GB RAM，并且摩尔定律预测当前每年增长 1.2GB，因此即使块头必须保存在内存中，存储也不应该成为问题。

8. Simplified Payment Verification

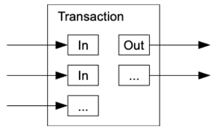
It is possible to verify payments without running a full network node. A user only needs to keep a copy of the block headers of the longest proof-of-work chain, which he can get by querying network nodes until he's convinced he has the longest chain, and obtain the Merkle branch linking the transaction 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.

9. Combining and Splitting Value

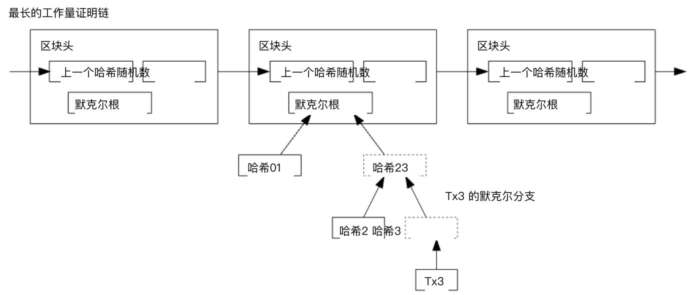
Although it would be possible to handle coins individually, it would be unwieldy to make a separate transaction for every cent in a transfer. To allow value to be split and combined, transactions contain multiple inputs and outputs. Normally there will be either a single input from a larger previous 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.

8. 简化付款验证

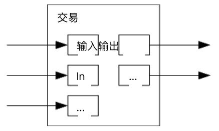
无需运行完整的网络节点即可验证付款。用户只需要保留最长工作量证明链的区块头副本，他可以通过查询网络节点来获取该副本，直到他确信自己拥有最长的链，并获得将交易链接到区块的 Merkle 分支他无法自己检查交易，但通过将其链接到链中的某个位置，他可以看到网络节点已接受该交易，并在进一步确认网络已接受该交易后添加块。



因此，只要诚实的节点控制网络，验证就是可靠的，但如果网络被攻击者制服，验证就更容易受到攻击。虽然网络节点可以自行验证交易，但只要攻击者能够继续压制网络，简化的方法就可能会被攻击者捏造的交易所欺骗。防止这种情况的一种策略是在网络节点检测到无效块时接受来自网络节点的警报，提示用户软件下载完整块并发出警报交易以确认不一致。经常收到付款的企业可能仍然希望运行自己的节点，以获得更独立的安全性和更快的验证。

9. 合并和分割值

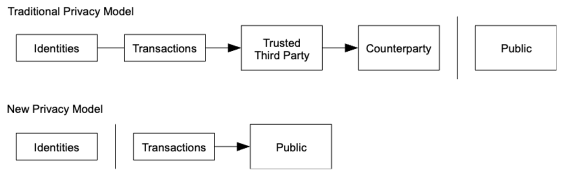
尽管可以单独处理硬币，但为转账中的每一分钱进行单独的交易会很麻烦。为了允许价值的分割和组合，交易包含多个输入和输出。通常情况下，要么是来自较大的先前交易的单个输入，要么是组合较小金额的多个输入，并且最多有两个输出：一个用于付款，一个将找零（如果有）返回给发送者。



应该注意的是，扇出（其中一个事务依赖于多个事务，而这些事务又依赖于更多事务）在这里不是问题。永远不需要提取交易历史的完整独立副本。

10. Privacy

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.

11. Calculations

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.

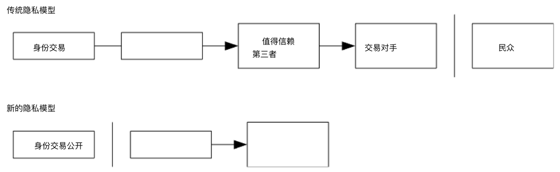
The probability of an attacker catching up from a given deficit is analogous to a Gambler's Ruin problem. Suppose a gambler with unlimited credit starts at a deficit and plays potentially an infinite number of trials to try to reach breakeven. We can calculate the probability he ever reaches breakeven,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

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

10. 隐私

传统的银行模式通过限制相关方和受信任的第三方对信息的访问来实现一定程度的隐私。公开宣布所有交易的必要性排除了这种方法，但仍然可以通过在另一个地方中断信息流来维护隐私：通过保持公钥匿名。公众可以看到某人正在向其他人发送一笔金额，但没有将交易与任何人联系起来的信息。这类似于证券交易所发布的信息水平，其中单笔交易的时间和规模（即“磁带”）是公开的，但不告诉交易方是谁。



作为附加防火墙，每笔交易都应使用新的密钥对，以防止它们链接到共同的所有者。对于多输入交易来说，一些链接仍然是不可避免的，这必然表明它们的输入由同一所有者拥有。风险在于，如果密钥的所有者被泄露，链接可能会泄露属于同一所有者的其他交易。

11. 计算

我们考虑攻击者试图比诚实链更快地生成替代链的情况。即使实现了这一点，它也不会让系统接受任意更改，例如凭空创造价值或拿走不属于攻击者的金钱。节点不会接受无效的交易作为付款，诚实的节点也永远不会接受包含它们的区块。攻击者只能尝试更改自己的一笔交易来取回他最近花费的钱。
诚实链和攻击者链之间的竞争可以被描述为二项式随机游走。成功事件是诚实链延长一个区块，领先优势增加+1，失败事件是攻击者链延长一个区块，差距缩小-1。
攻击者追赶给定赤字的概率类似于赌徒破产问题。假设一个拥有无限信用的赌徒从赤字开始，并可能进行无限次尝试以试图达到收支平衡。我们可以计算他达到盈亏平衡的概率，或者攻击者追上诚实链的概率，如下[8]：

p = 诚实节点找到下一个区块的概率 q = 攻击者找到下一个区块的概率
q = 攻击者从后面的 z 个区块追上的概率

q = {如果 p⩽q □q / }
p□如果 p□q

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:

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

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

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

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;
}
```

鉴于我们假设 p > q，随着攻击者必须追上的区块数量增加，概率呈指数下降。在形势对他不利的情况下，如果他没有尽早幸运地向前冲，随着他越来越落后，他的机会就会变得微乎其微。
我们现在考虑新交易的接收者需要等待多长时间才能充分确定发送者无法更改交易。我们假设发送者是一个攻击者，他想让接收者相信他已经支付了他一段时间，然后在一段时间过后将其转回给自己。当这种情况发生时，接收者会收到警报，但发送者希望为时已晚。
接收方生成一个新的密钥对，并在签名前不久将公钥提供给发送方。这可以防止发送者通过连续处理来提前准备一系列区块，直到他足够幸运地提前足够远，然后在那一刻执行交易。一旦交易被发送，不诚实的发送者就开始在包含其交易的替代版本的平行链上秘密工作。
接收者等待，直到交易被添加到一个块中并且 z 个块已经链接到它之后。他不知道攻击者取得的确切进展量，但假设诚实区块花费了每个区块的平均预期时间，则攻击者的潜在进展将是具有预期值的泊松分布：

□=z q
p

为了得到攻击者现在仍然可以追上的概率，我们将他可以取得的每个进步量的泊松密度乘以他从该点可以追上的概率：

∞ □e _. {□q / p□如果 k ⩽ z}
∑=0 k! 1 如果 k □ z

重新排列以避免对分布的无限尾部求和......

1− z □e — 1− q / p
∑=0 k!

转换为 C 代码...

```
#include <math.h> double AttackerSuccessProbability(double
q, int z) {

    双 p = 1.0 - q; 双 lambda = z * (q
    / p);双倍总和 = 1.0; 整数 i, k;  for
    (k = 0; k <= z; k++) {



        双泊松 = exp(-lambda);对于 (i = 1; i
        <= k; i++)
泊松 *= lambda / i; sum -= 泊松 * (1 - pow(q / p, z -
k));返回总和;  }
```

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
```

12. Conclusion

We have proposed a system for electronic transactions without relying on trust. We started with the usual framework of coins made from digital signatures, which provides strong control of ownership, but is incomplete without a way to prevent double-spending. To solve this, we proposed a peer-to-peer network using proof-of-work to record a public history of transactions that quickly becomes computationally impractical for an attacker to change if honest nodes control a majority of CPU power. The network is robust in its unstructured simplicity. Nodes work all at once with little coordination. They do not need to be identified, since messages are not routed to any particular place and only need to be delivered on a best effort basis. Nodes can leave and rejoin the network at will, accepting the proof-of-work chain as proof of what happened while they were gone. They vote with their CPU power, expressing their acceptance of valid blocks by working on extending them and rejecting invalid blocks by refusing to work on them. Any needed rules and incentives can be enforced with this consensus mechanism.

运行一些结果，我们可以看到概率随 z 呈指数下降。

```
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
```

求解 P 小于 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
```

12. 结论

我们提出了一种不依赖信任的电子交易系统。我们从由数字签名制成的通常的硬币框架开始，它提供了对所有权的强大控制，但如果没有防止双重支出的方法，它是不完整的。为了解决这个问题，我们提出了一个使用工作量证明的点对点网络来记录交易的公共历史记录，如果诚实的节点控制了大部分 CPU 能力，那么攻击者在计算上很快就无法进行更改。该网络因其非结构化的简单性而非常强大。节点同时工作，几乎不需要协调。它们不需要被识别，因为消息不会路由到任何特定的地方，只需要尽力传递。节点可以随意离开和重新加入网络，接受工作量证明链作为它们离开时发生的事情的证明。他们用自己的 CPU 能力进行投票，通过扩展有效块来表达对有效块的接受，并通过拒绝处理无效块来表达对无效块的接受。任何需要的规则和激励措施都可以通过这种共识机制来执行。

References

[1] W. Dai, "b-money," http://www.weidai.com/bmoney.txt, 1998.

[2] H. Massias, X.S. Avila, and J.-J. Quisquater, "Design of a secure timestamping service with minimal trust requirements," In *20th Symposium on Information Theory in the Benelux*, May 1999.

[3] S. Haber, W.S. Stornetta, "How to time-stamp a digital document," In *Journal of Cryptology*, vol 3, no 2, pages 99-111, 1991.

[4] D. Bayer, S. Haber, W.S. Stornetta, "Improving the efficiency and reliability of digital time-stamping," In *Sequences II: Methods in Communication, Security and Computer Science*, pages 329-334, 1993.

[5] S. Haber, W.S. Stornetta, "Secure names for bit-strings," In *Proceedings of the 4th ACM Conference on Computer and Communications Security*, pages 28-35, April 1997.

[6] A. Back, "Hashcash - a denial of service counter-measure," http://www.hashcash.org/papers/hashcash.pdf, 2002.

[7] R.C. Merkle, "Protocols for public key cryptosystems," In *Proc. 1980 Symposium on Security and Privacy*, IEEE Computer Society, pages 122-133, April 1980.

[8] W. Feller, "An introduction to probability theory and its applications," 1957.

参考

[1] W. Dai，“b-money”，http://www.weidai.com/bmoney.txt，1998。

[2] H. Massias，X.S.阿维拉，J.-J. Quisquater，“设计具有最低信任要求的安全时间戳服务”，比荷卢经济联盟第 20 届信息论研讨会，1999 年 5 月。

[3] S.哈伯，W.S. Stornetta，“如何为数字文档添加时间戳”，《密码学杂志》，第 3 卷，第 2 期，第 99-111 页，1991 年。

[4] D.拜耳、S.哈伯、W.S. Stornetta，“提高数字时间戳的效率和可靠性”，《Sequences II：通信、安全和计算机科学方法》，第 329-334 页，1993 年。

[5] S.哈伯，W.S. Stornetta，“位串的安全名称”，第 4 届 ACM 计算机和通信安全会议记录，第 28-35 页，1997 年 4 月。

[6] A. Back，“Hashcash - 拒绝服务对策”，http://www.hashcash.org/papers/hashcash.pdf，2002 年。

[7] R.C. Merkle，“公钥密码系统协议”，In Proc。 1980 年安全和隐私研讨会，IEEE 计算机协会，第 122-133 页，1980 年 4 月。

[8] W. Feller，“概率论及其应用简介”，1957 年。