比特币：一个点到点的电子现金系统

中本聪satoshin@gmx.comwww.bitcoin.org

摘要。 一个纯粹点到点版本的电子现金将会允许在线支付从一方直接发送给另一方，而无需经过金融机构。数字签名提供了部分解决方案，但是，如果仍然需要一个受信任第三方来防止双重花费（double-spending）的话，那么其主要优点就丧失殆尽了。我们提出一个使用点到点网络来解决双重花费问题的解决方案。网络给交易打上时间戳，通过计算交易的哈希（hash）并将其插入到一条持续延长的、基于哈希的工作量证明链之中去，从而形成一条记录，除非重做工作量证明（proof-of-work），否则便不能被更改。最长链不仅提供了一系列被见证事件的证明，而且是其来自于最大CPU算力池的证明。只要大多数CPU算力被不会合谋攻击网络的节点所控制，它们就将以超过攻击者的速度生成最长链。网络本身只需要最为精炼的结构。消息按照最大努力原则进行广播，节点可以自由地离开和重新加入网络，并接受最长的工作量证明链作为它们离开时发生了什么事情的证明。

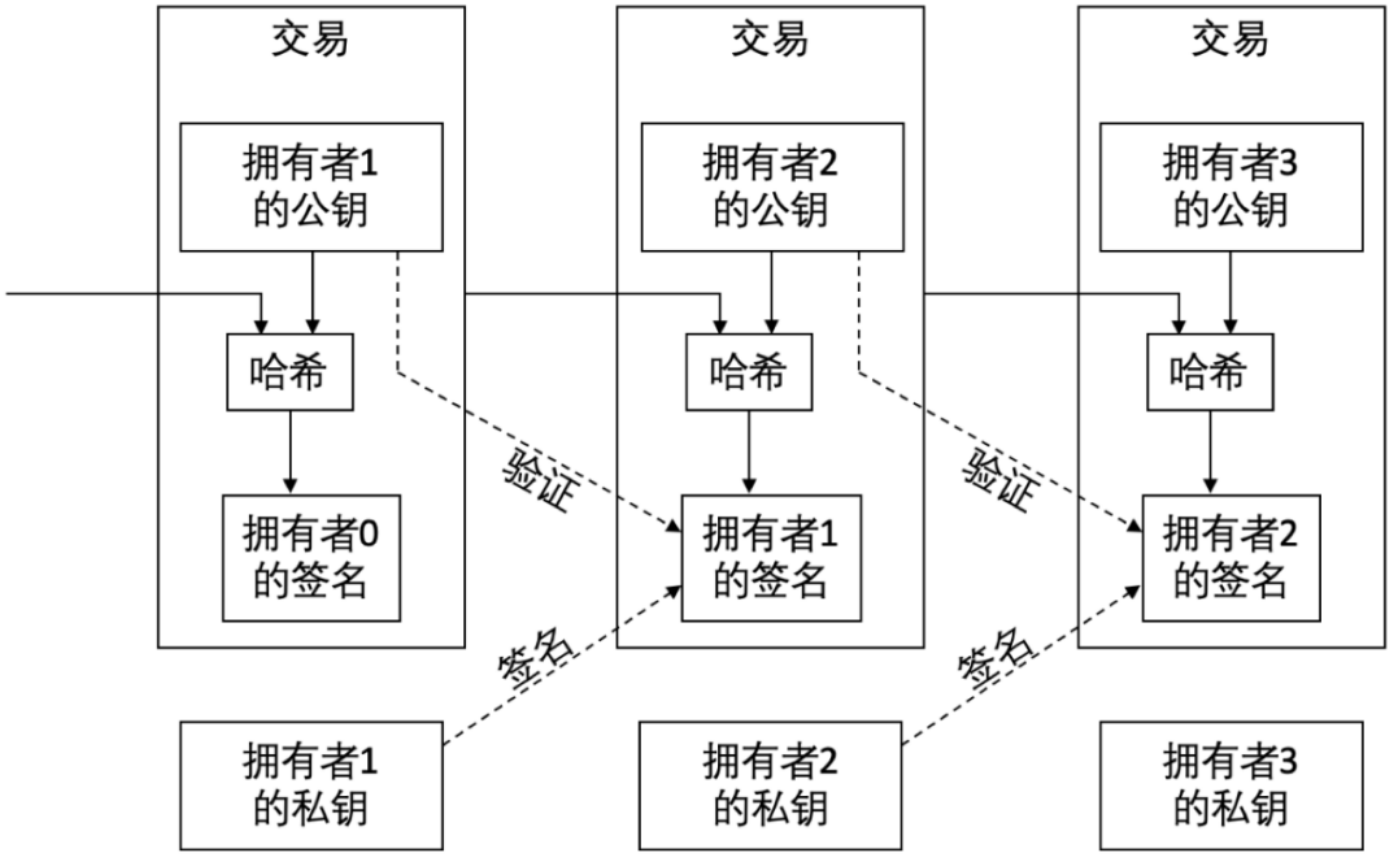
1. 引言

互联网上的贸易几乎已经是完全依赖金融机构作为受信任第三方来处理电子支付。尽管该系统对于大多数交易而言运转的足够良好，但它仍然受制于基于信任的模型所固有的弱点。完全不可逆的交易是不太可能的，因为金融机构不可避免地要调解纠纷。调解成本增加了交易成本，限制了最小实际交易尺寸并降低了小额零星交易的可能性，而且，失去了对不可逆服务进行不可逆支付的能力，就会有更多的成本。当存在逆转的可能性时，对信任的需求就扩散了。商家必须警惕他们的顾客，麻烦他们以获取原本并不需要的更多信息。一定百分比的欺诈被认为是不可避免的。这些成本和支付不确定性可以通过当面使用实物货币而得以避免，但是没有一种现存的机制可以通过通信信道进行支付而无需一个受信任的第三方。

我们需要的是一个基于密码学证明而非信任的电子支付系统，允许有意愿的任意双方互相直接交易而无需一个受信任的第三方。在计算上实际不可逆的交易将会保护卖家免受欺诈，并且可以很容易实现常规的支付托管机制（escrow）以保护买家。在这篇论文里，我们提出一个双重花费问题的解决方案，该方案使用一个点到点的分布式时间戳服务器来生成交易时间顺序的计算证明。只要诚实节点集体控制着比任何一组合谋攻击节点更多的CPU算力，系统就是安全的。

2. 交易

我们定义一枚电子硬币（electronic coin）就是一个数字签名链。每个拥有者通过对上一笔交易以及下一个拥有者的公钥的哈希进行数字签名，并将其添加到硬币的尾端来把硬币转移给下一个拥有者。收款人可以通过验证这些签名来验证所有权的链条。

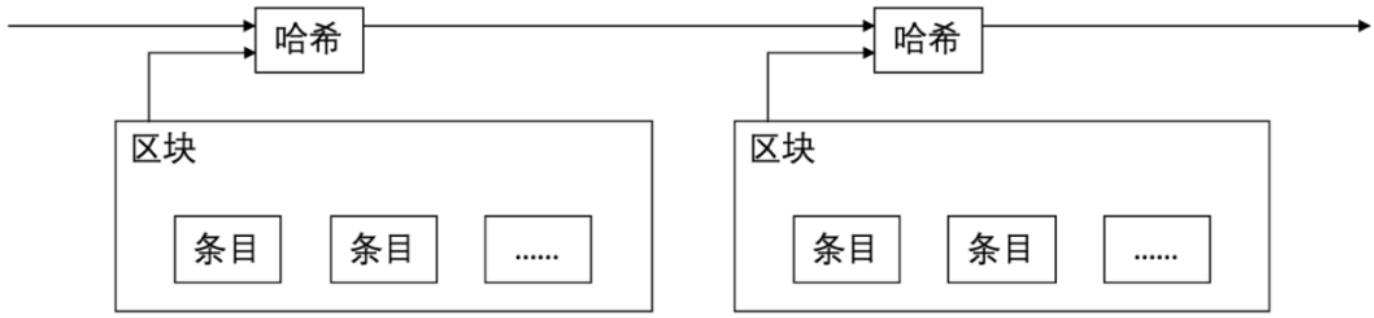


当然，问题在于收款人无法验证其中某个拥有者未曾双重花费这枚硬币。常见的解决方案是引入受信任的中央权威，或者铸币厂，来检查每一笔交易的双重花费。在每一笔交易完成后，硬币都必须返回到铸币厂并发行一枚新的硬币，而且只有直接从铸币厂发行出来的硬币才可以被信任为未被双重花费。这个解决方案的问题在于，整个货币系统的命运取决于运行铸币厂的公司，每一笔交易都不得不经过它们，就像一家银行一样。

我们需要一个方法，让收款人可以知道上一个拥有者没有对任何更早的交易进行签名。对我们的目的而言，最早的交易是那个算数的，这样我们就不必关心后来的双重花费尝试。在基于铸币厂的模型里，铸币厂知道所有的交易并决定哪一个先到达。为了在没有一个受信任方的情况下达成这一点，交易必须公开发布[1]，并且我们需要一个系统让参与者就交易接收顺序的单一历史达成一致。收款人需要证据来证明在每一次交易时，大多数节点都同意这笔交易是第一个被接收到的。

3. 时间戳服务器

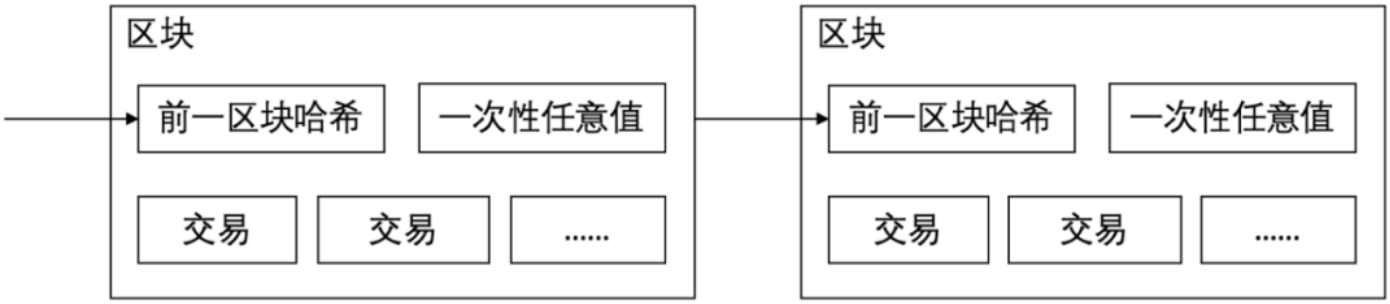
我们提出的解决方案从一个时间戳服务器开始。一个时间戳服务器是这样工作的：取一个区块的哈希——该区块中含有要加盖时间戳的条目——然后广泛发布该哈希，就像在一份报纸或者新闻组帖子上所做的那样[2-5]。时间戳证明了，为了被包含在哈希中，显然的，数据在那个时刻必定已经存在。每一个时间戳都在其哈希中包含了上一个时间戳，形成一条链，这样每一个额外的时间戳都对其之前的所有时间戳进行了加固。



4. 工作量证明

为了实现一个基于点到点的分布式时间戳服务器，我们需要使用类似于亚当·贝克的哈希现金[6]那样的工作量证明系统，而不是报纸或者新闻组帖子。工作量证明需要扫描查找一个数值，当该数值被哈希，比如使用SHA-256算法，这个哈希会以若干个零位（zero bits）打头。平均所需的工作量会随着所需的零位个数的增长而呈指数级增加，并且通过执行一次哈希就可以验证它。

对于我们的时间戳网络，我们通过递增区块中包含的一个一次性任意值（Nonce），直到找到一个值，让区块哈希得到所需的零位数，以此实现工作量证明。一旦CPU算力被消耗来使其满足工作量证明，区块就不能被改变了，除非重做工作量证明。由于后续区块会链接在它后面，改变该区块所需的工作量就会包括重做所有后续区块。



工作量证明还解决了在多数决策中确定代表制的问题。如果基于一个IP地址一票制确定多数，那么该制度就有可能被任何有能力分配很多IP的人所颠覆。工作量证明本质上是一个CPU一票。多数决策由最长链来代表，它有着最大的投入其中的工作量证明。如果多数CPU算力被诚实节点所控制，诚实链就会以最快的速度增长并超过任何竞争链。要修改一个过去的区块，攻击者将不得不重做该区块以及所有后续区块的工作量证明，然后追上并超越诚实节点的工作量。我们稍后会展示，随着后继区块被添加，一个较慢的攻击者追上的概率会以指数级衰减。

为了抵消硬件的不断提速以及运行节点的收益随时间的不断变化，工作量证明的难度将取决于一个以每小时平均区块数量为目标的移动平均值。如果区块产生得太快，难度就会增加。

5. 网络

运行网络的步骤如下：

1) 新交易被广播给全体节点。2) 各个节点把新交易收集到区块内。3) 各个节点进行工作，为它的区块寻找一个困难的工作量证明。4) 当一个节点找到了一个工作量证明，它就把该区块广播给全体节点。5) 节点接受该区块，仅当其中的所有交易都有效并且未被花费的情况下。6) 节点通过在链上工作，使用已接受区块的哈希作为上一个哈希，创建下一个区块，以表示它们对该区块的接受。

节点总是认为最长链是正确的那一个，并将会保持工作以延长它。如果两个节点同时广播了不同版本的下一个区块，一些节点可能会首先接收到其中一个，而其他节点则首先接收到另一个。在那种情况下，它们基于最先接收到的一个进行工作，但是保留另外一个分支以备其变得更长。当下一个工作量证明被发现，一个分支变得更长时，平局就会被打破；然后，正在另一个分支上工作的节点就会切换到这个更长的分支上来。

新交易广播不是必须到达全体节点。只要它们到达很多的节点，它们很快就会被放到一个区块里面去。区块广播对于丢失消息也具有容忍度。如果一个节点未接收到某一个区块，当它接收到下一个区块，从而意识到自己错过了一个区块时，它就会请求该区块。

6. 激励

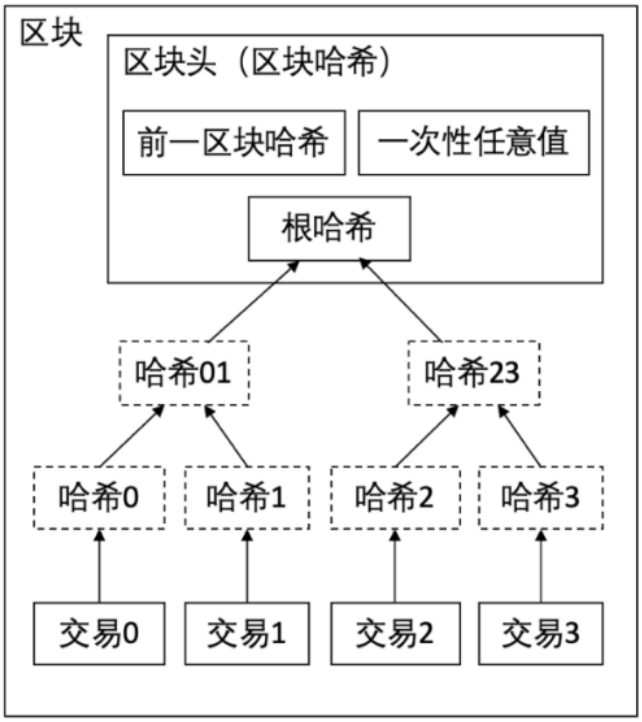
按照惯例，一个区块中的第一笔交易是一笔特殊的交易，它生成了一个新硬币，并为区块的创建者所拥有。这为支撑网络的节点们增加了激励，并提供了一个把币初始分发到流通中去的方法，因为没有中央权威来发行这些币。固定数量新币的稳步增加与黄金开采者消耗资源把黄金投入流通是类似的。在我们的例子里，被消耗掉的是CPU时间和电力。

激励也可以由交易手续费提供。如果一笔交易的输出值小于其输入值，则差额作为交易手续费，就会被添加到包含该笔交易的区块的激励值之中。一旦既定数量的币全部进入流通，激励就可以完全过渡到交易手续费，也就完全没有通胀了。

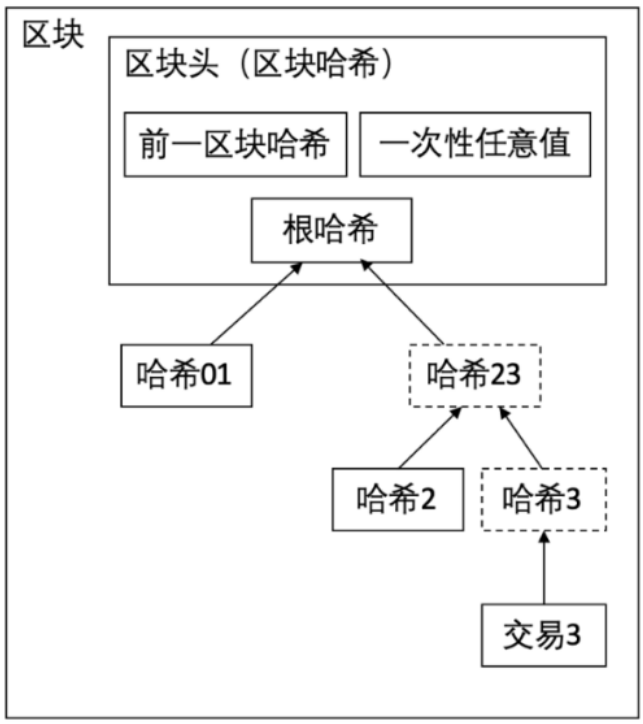
激励可以有助于鼓励节点保持诚实。如果一个贪婪的攻击者能够组织起比所有诚实节点更多的CPU算力，他将不得不做出一个选择，或者用算力欺诈他人，偷回他的付款，或者用算力生成新硬币。他应该会发现按规则出牌会更加有利可图——因为这个规则会给予他更多硬币，比其他所有人加起来还要多——而不是破坏系统及其自身财富的有效性。

7. 回收磁盘空间

一旦一枚硬币里的最后一笔交易被深埋在足够多的区块下面，在此之前已经花费掉的交易就可以被丢弃掉，以节省磁盘空间。为了便于如此处理而不破坏区块哈希，交易被哈希成一颗默克尔树（Merkle Tree）[7][2][5]，只有根才被包含在区块哈希中。然后旧区块就可以通过剪掉树枝而压缩。内部哈希不需要存储。



交易被哈希成一颗默克尔树

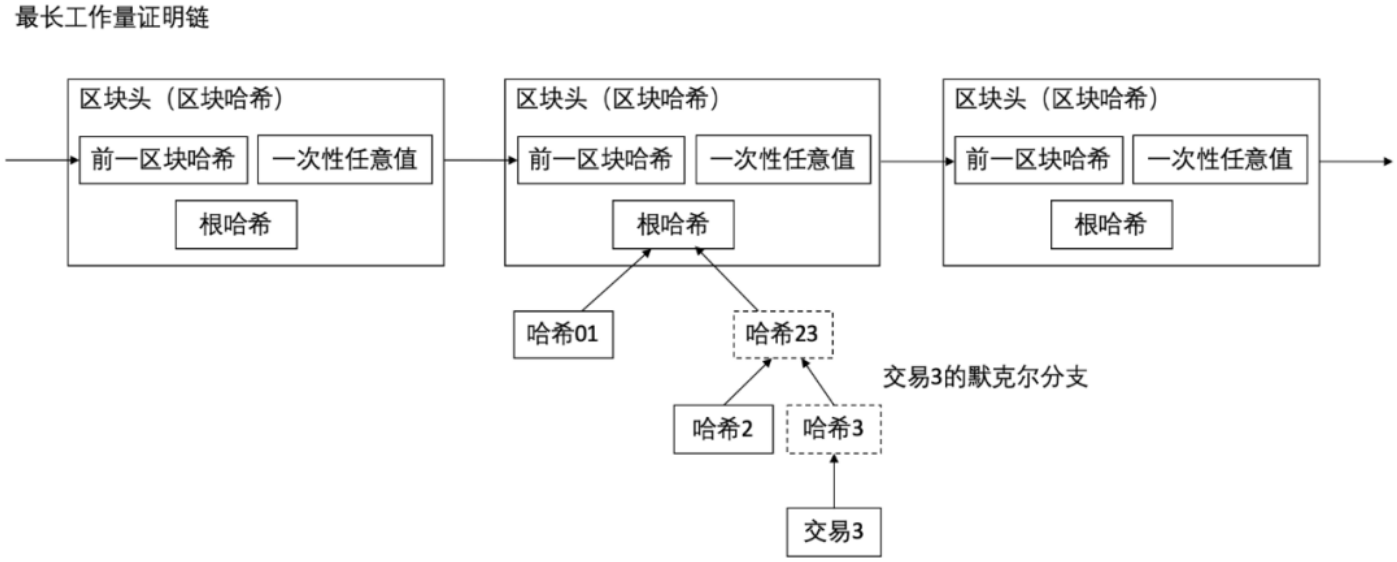


从区块中剪枝掉交易0-2之后

一个不含交易的区块头大概有80字节。如果我们假设每10分钟生成一个区块，80字节 * 6 * 24 * 365 = 4.2MB每年。考虑到2008年售卖的计算机系统大多有着2GB内存，且摩尔定律预测当前增长率为每年1.2GB，即便区块头都要存到内存里，存储应该也不会是一个问题。

8. 简化支付验证

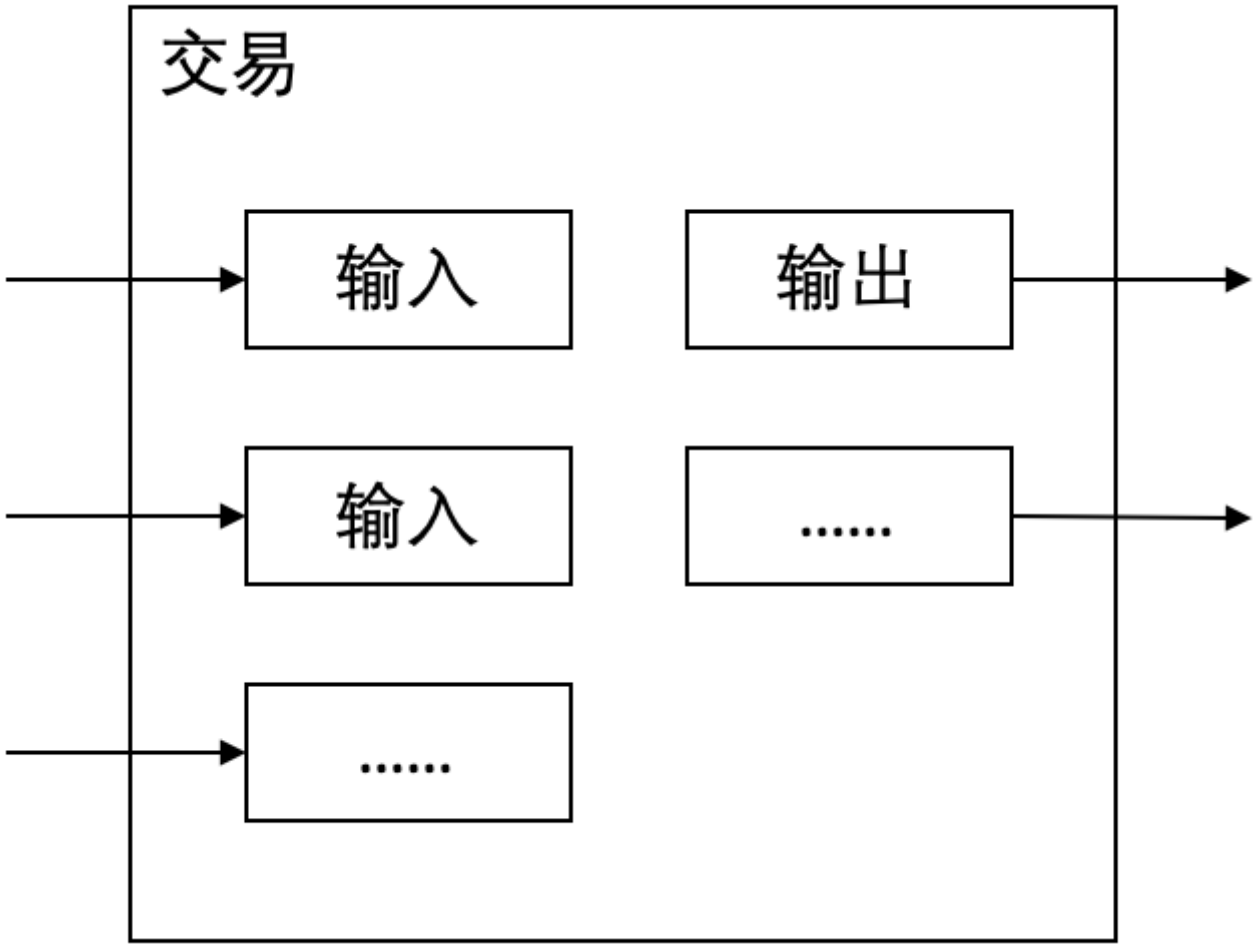
要验证支付而不运行一个网络全节点是可能的。一个用户只需要保留最长工作量证明链的区块头的一份拷贝，该拷贝的获得可以通过查询网络节点进行，直到他相信自己已经拿到了最长链，并获取把该交易链接到将其包含并打上时间戳的区块的那一支默克尔分支。他无法独自检查该交易，但是通过将其链接到一个链上的位置，他可以知道一个网络节点已经接受了这笔交易，并且后续增加的区块进一步确认网络已经接受了这笔交易。



由此，只要诚实节点控制网络，这样验证就是可靠的；但是，如果网络被攻击者压制的话，就会更脆弱。尽管网络节点可以自行验证交易，但只要攻击者可以一直压制住网络的话，简化方法就可能被攻击者伪造的交易所欺骗。一种防范的对策就是接受来自于网络节点的警报，当它们探测到非法区块时，提示用户软件下载完整的区块和被警报的交易，以证实不一致性。频繁接收付款的企业可能仍然会想要运行它们自己的节点，以获得更独立的安全性和更快的验证。

9. 合并和分割价值

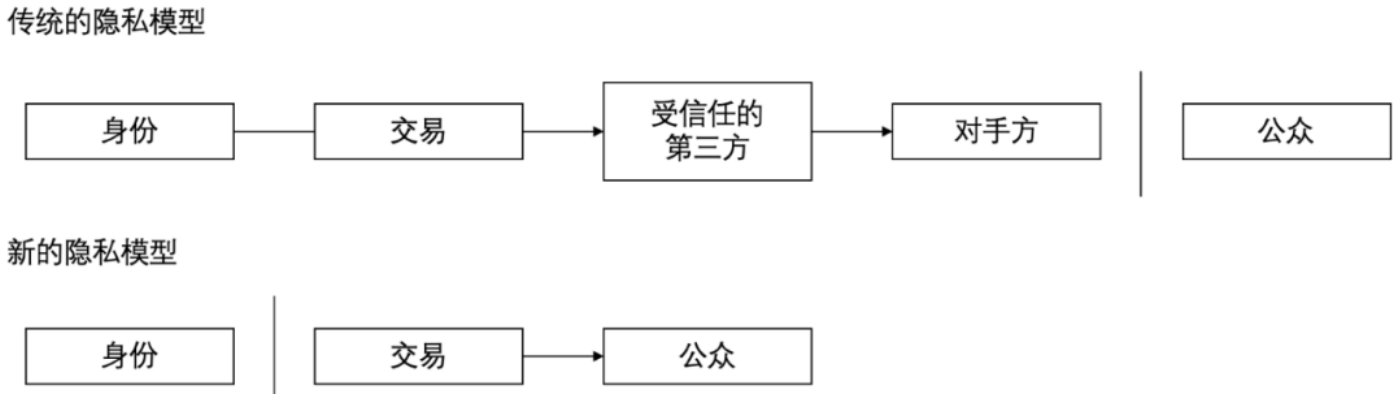
尽管可以分别处理每个硬币，但是在转账中为每一分钱生成一笔单独的交易是很笨拙的。为了让价值可以被分割和合并，交易包含多项输入和输出。正常情况下，要么是来自先前更大交易的单笔输入，要么是合并了多个较小金额的多笔输入，并且最多（刘教链注：“最多(at most)”疑为中本聪笔误，似乎用“多数(mostly)”更为通顺）有两项输出：一项用于付款，一项把找零返还给发送者——如果有的话。



应当指出的是，扇出（fan-out），也就是一笔交易依赖于若干笔交易，那些交易又依赖于更多笔交易，在这里并不成为一个问题。永远都不需要提取一笔交易历史的完整独立拷贝。

10. 隐私

传统的银行模型通过对参与方以及受信任第三方的信息限制访问来实现一定水平的隐私性。公布所有交易的必要性就排除了这种方法，但是仍然可以通过在另一处切断信息流而保持隐私性：将公钥保持匿名。公众可以看到有人发送了一定金额给另一个人，但是没有信息可以把交易关联到任何人。这有点儿类似于股票交易所的信息披露水平：各个交易的时间和规模，即"行情（tape）"，是公开的，但是并不会透露各方都是谁。



作为额外的一层防火墙，应当为每一笔交易使用一个新的密钥对，以防止其被关联到一个共同拥有者。由于多输入交易必然会暴露出其输入为同一人所有，一些关联就仍然无法避免。风险在于如果一个密钥的拥有者被暴露，这一关联就可以暴露出属于同一拥有者的其他交易。

11. 计算

我们考虑攻击者尝试以比诚实链更快的速度生成替代链的情景。即使攻击可以得逞，这也不会把系统置于可任意更改的境地，比如凭空创造价值，或者攫取从未属于攻击者的钱。节点不会接受非法交易支付，诚实节点也从来不会接受包含该交易的区块。攻击者只能尝试更改他自己的交易以取回他刚刚花掉的钱。

在诚实链和攻击链之间展开的竞赛可以使用二项随机游走（Binomial Random Walk）进行刻画。成功情形是诚实链延长了一个区块，领先扩大 +1；失败情形是攻击者的链延长了一个区块，缺口减小 -1。

攻击者从给定赤字追上的可能性和赌徒破产问题（Gambler's Ruin）类似。假定一个有着无限透支额度的赌徒从一个赤字开始，并可能进行无数次尝试以试图达到盈亏平衡。我们就能计算出他最终达到盈亏平衡，或者说攻击者最终追上诚实链的概率，如下[8]：

p = 诚实节点发现下一区块的概率q = 攻击者发现下一区块的概率qz = 攻击者从落后z个区块处最终追上诚实链的概率

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

基于我们假设p > q，随着攻击者必须追赶的区块数增加，追上的概率将呈指数级下降。由于机会不在他那一边，如果他不能在早期幸运地突飞猛进的话，当落后更远时他的机会就越来越小渐渐消失了。

我们现在考虑一笔新交易的接收者需要等待多久，才能充分确信发送者无法更改交易。我们假设发送者是攻击者，他想让接收者一时相信他已经支付，然后过些时间之后再切换为付回给他自己。当这些发生时，接收者会得到警报，但是发送者希望一切为时已晚。

接收者生成一个新的密钥对，并在签名前不久才把公钥给到发送者。这可以防止发送者提前准备区块链，并通过在上面持续工作直至幸运地领先足够多，然后到那时才执行交易。一旦交易被发出，不诚实的发送者就偷偷地在一条包含其交易的替代版本的平行链上开始工作。

接收者一直等到交易被添加进一个区块，并且之后又链接了z个区块。他不知道攻击者确切的进度，但是假设诚实区块花费了每区块的平均期望时间，则攻击者潜在的进度就将会是一个泊松分布（Poisson distribution），其期望值为：

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

为了获得攻击者现在仍能追上的概率，我们把他可能已经完成的每种进度情况的柏松密度乘上他能从那一点追上的概率：

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

重整以避免对分布的无穷尾部进行求和……

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

转化为C语言代码……

```
#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;
}
```

运行出一些结果，我们可以看到概率随z值而呈指数级下降。

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

对P小于0.1%进行求解.....

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

我们已经提出了一种不依赖于信任的电子交易系统。我们从把数字签名做成硬币的常规框架开始，这提供了对所有权的强控制，但因缺乏一个防止双重花费的方法而不够完整。为了解决这个问题，我们提出了一个点到点的网络，使用工作量证明来记录交易的公共历史，如果诚实节点控制大部分CPU算力的话，攻击者要改变这一历史很快就在计算上变的不切实际了。网络的鲁棒性根植于其非结构化的简单性之中。节点几乎不需要协调就可以同时工作。它们不需要被辨识，因为消息不会被路由到任何特定的地方，而仅仅需要按照最大努力原则去传递就可以了。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了什么事情的证明。它们用CPU算力来投票，通过在有效区块之上工作以使其延长来表示接纳，通过拒绝在无效区块上工作来抵制。任何所需的规则和激励都可以通过这一共识机制来得到执行。

参考文献

[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.