比特币白皮书

比特币：点对点电子现金系统
版本号0.02

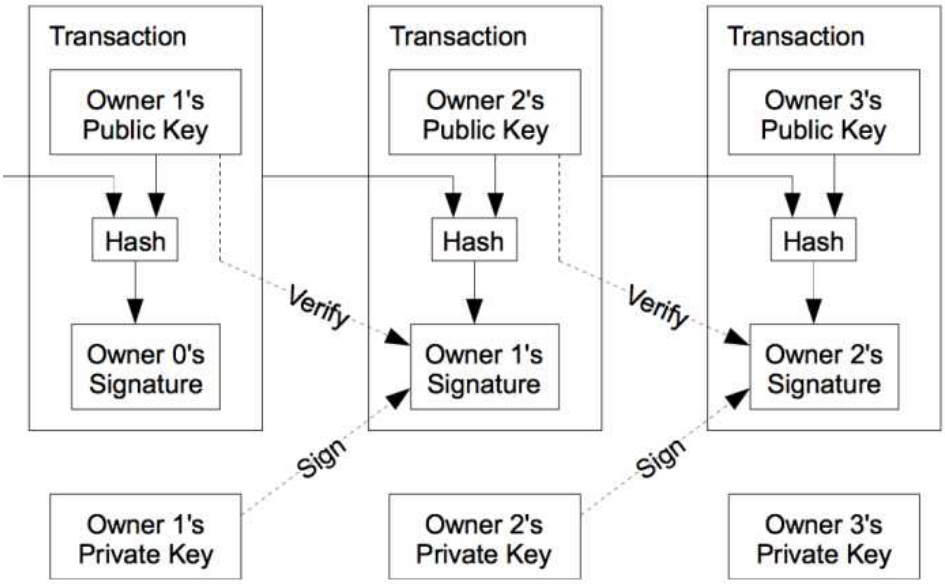
原著 中本聪 satoshin@gmx.com
出处 https://bitcoin.org/bitcoin.pdf
翻译 胡震生 huzhensheng@gmail.com
作者私人微信 130760807 标明来自知乎

概述
一个真正的的点对点的电子现金应该允许从发起方直接在线支付给对方，而不需要通过第三方的金融服务机构。现有的数字签名技术虽然提供了部分解决方案，但如果还需要经过一个可以信任的第三方机构来防止电子现金的“双重支付”，那就丧失了电子现金给人类带来的最大好处。我们针对电子现金会出现的“双重支付”问题，用点对点的网络技术提供了一个解决方案。该网络通过给交易记录打上时间戳，并通过哈希对其加密，然后将其并入一个不断增长的哈希记录所组成的链条文件中，以此形成一个新的交易记录，这个哈希记录链条文件（以下简称链条文件）是由一个需要证明工作量的系统网路所提供存储和计算服务的。没有基于工作量证明的系统网络来重新完成所有的工作量证明，一个已经形成的记录是不能被修改的。基于工作量证明的系统理念，最长的链条文件不仅仅是提供这些工作量序列事件记录的证据，而且它也是由最大的CPU处理能力池产生的。只要由计算机节点控制的大部分CPU处理能力不联合攻击网络本身，它们的处理能力将超过攻击者，并生成最大的链条文件。
这个网络它自身需要最简单的结构，以方便信息包尽最大努力广播给网络上的计算机节点，同时计算机节点只需要接受它们离开时工作量网络的产生的数据链,它也可以随时加入和离开网络。

1 介绍
互联网商业的电子支付已经发展到了几乎都需要专有的金融机构来提供第三方信任来处理的阶段虽然大部分交易，系统都能工作的足够好，但它还是需要面对基于信任的基础模型带来的天生的缺点。 自从金融机构不可避免的开始调解纠纷，完全而不可撤销的交易就不能真正的实现。调解成本增加了交易成本，限制了实际可行交易的最小规模，同时彻底切断了为日常小额交易提供服务的可能性，广义上的成本让系统失去了为不可撤销类型的服务提供不可撤销支付的能力。因为用户有撤销支付的可能性，所以需要某个时间段内连续性的信任，这导致商家必须防备他们的客户，骚扰他们以为了得到更多的他们不再需要的信息。不可避免的，一定比例下的欺诈性交易是可以接受的。虽然使用物理货币可以避免这些成本和支付的不确定性，但是没有商家在不通过可信任的三方的沟通渠道前提下去支付。
这就是为什么需要一个基于加密证明的电子支付系统取代原来的基于信任的基础模型，允许任意两个希望交易的双方不通过基于信任的第三方来直接支付。经过计算的无效交易将自动被撤销，以保护卖家远离欺诈，常规附带条件的契约，将被机械化的执行，将保护买家变得很简单。这篇论文中，我们提供了一个基于点对点的分布式时间戳服务器去生成基于时间序列的交易订单的计算证明方案，从而解决双重支付问题。只要诚实的节点全体所控制的CPU计算能力的总和，大于联合攻击节点计算能力组的总和，该系统就是安全的。

2 交易

我们首先定义一个电子货币就是一个包含了一串数字签名的链。每一个币交易者通过用哈希技术对上一个交易信息和下一个拥有者的公共密钥两者进行加密，然后进行数字签名，最后将这些信息添加到这枚电子货币的末尾。下一个收款人通过私有密钥与链里的公有密钥进行签名验证，以确认自己是链也就是这枚电子货币的拥有者。

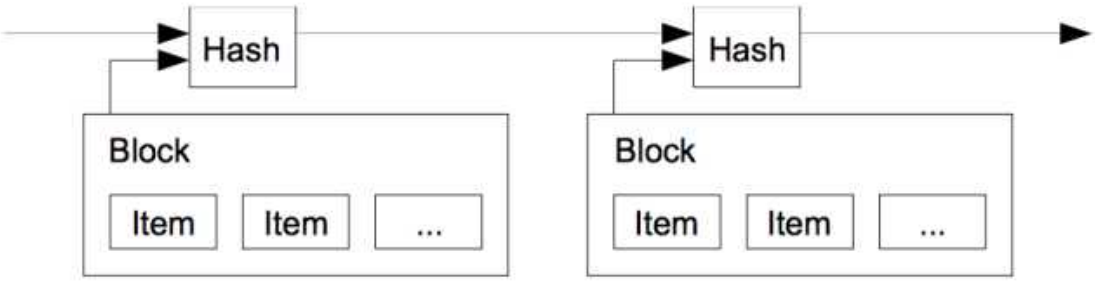


当然这个流程的问题在于，收款人还是不能验证币的某位拥有者是否对这个货币进行了双重花费，通常的解决办法是引入一个值得信任的中心权威，或者造币厂来检查每一笔交易的是否被双重花费。每一笔交易结束后，货币必须由造币厂回收以发行一个新的货币，只有从造币厂直接发行的货币才会被信任为没有被双重花费。这个解决方案的灾难之处是，整个金钱系统依靠某家公司来运营造币厂，就像银行一样，每一笔交易不得不通过它们。

我们需要用一个方法来让收款人知道货币的上一个拥有者没有在更早的任何一个交易里签名授权以导致双重花费。我们的目的是，去计算以前的交易，我们不需要关心之后的交易是否对其进行双重花费。唯一的方法是知道之前所有的交易，才会确认交易不存在。基于造币厂的模型，造币厂知道所有的交易，同时决定哪一个交易请求第一时间到达。在没有可信任的三方下完成这个目的，交易必须公开发布，我们需要一个系统的每个参与者，都同意一个它们已经接受到的单一的订单历史。收款人需要通过主要节点认同它们已经第一时间收到了这笔交易来证明每一笔交易。

3 时间戳服务器

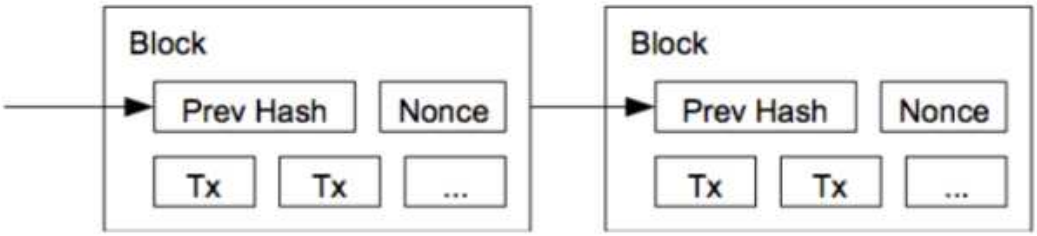
我们的解决方案从一个时间戳服务器开始，一个时间戳服务器对一组已经被时间戳标示过的数据块进行哈希加密，然后广泛公开发布这个哈希，就像新闻或者以前的论坛发帖一样。明显的，为了进入哈希里，时间戳证明了数据在这个时间是必须必定存在的，每个时间戳包含了上一个交易的时间戳在它的哈希里，以后每次交易的时间戳都对上一个进行了加强，以此形成了一个链。



4 工作量证明

我们将需要使用工作量证明系统，在点对点的基础之上构建一个分布式的时间戳服务器，和adam back提出的的哈希现金很像，而不是以前的新闻组及论坛的机制。当数据被哈希加密后，工作量证明用安全散列算法sha-256对一个数据的哈希值进行检查。哈希从一定数量的0字节开始，检查的平均工作量随着0的字节的数量增长而呈指数增长，而校验只需要执行一次哈希操作。

为了我们的时间戳网络可行，我们增加一个不会被重复的随机数进到数据块内并执行一定的工作量来找到它，这个数据块的的哈希已经包含已经所需数量的0字节。CPU处理能力一经被证明它满足了所需的工作量，不重做所有的工作，这个数据块将不能被修改。随后的数据块被链接在其最后，修改数据块的信息需要把其后所有的数据块的工作量重做。



这个工作量系统也解决了集体决策谁代表大多数的问题。如果大多数是基于一个IP地址一票的机制，它将被能分配大量IP地址的人所破坏，工作量证明是基于一CPU一票的。大多数决策被最长的链代表，也代表了最大工作量效果的投入。如果大多数CPU被诚实节点控制，诚实链将最快速的增长，超过任何竞争的链。 修改一个过去的块，一个攻击者将不得不把块和之后所有的块里所有的工作量重做，然后追上超过诚实节点的工作。我们随后将展示一个慢速攻击者追上随后的数据块的可能性随数据块的增加呈指数增加。

为了抵偿硬件增加的速度和节点运行时的变化的收益，工作量证明将被一个移动平均值来确定，即每小时平均生成的数据块数。如果它们生成的太快，难度也在增加。

5 网络

运行这个网络的步骤如下：

1 新的交易被广播给所有节点

2每一个节点收集新的交易写进一个数据块

3每个节点发现这个数据块的工作量的难度

4 当一个节点证明了它的工作量，它将广播这个数据块给所有的节点。

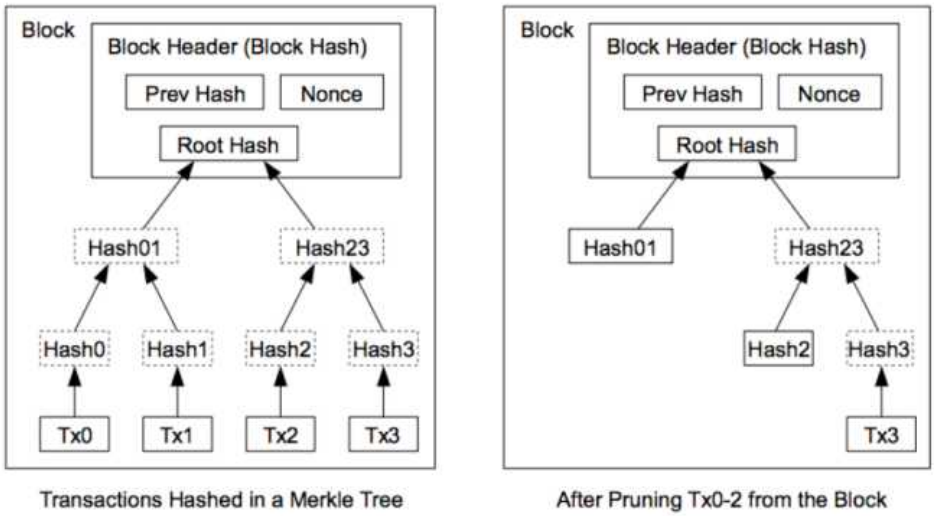
5节点接受这个数据块，只有数据块中所有的交易都是有生效和没有被支付过的，节点才会接受这个数据块。
6 节点通过创建下一个数据块在数据链上，同时把发送节点数据块的哈希作为创建数据块的上一个哈希，表示它们接受了这个数据块。
节点始终认为最长的数据链是正确的，会一直在上面延展。如果两个节点一起广播不同版本的数据块，一些节点先接收到一个或者另一个。在这个例子里，它们会先在第一个接收的数据块上开始工作，但是储存另一个作为下一个分支，防止它变的更长。当工作量证明网络发现其中一个分支变的更长，在短链上工作的节点会切换到更长的链上，其所属关系也会被打断，
新的交易广播不需要到达所有节点，它们只需要尽可能到达多的节点，它们将被整合进数据块中。数据块广播也容忍丢弃信息。如果一个节点没有收到一个数据块，它将持续请求它，直到它接受到下一个数据块，并相信它是丢失的那个。

6 激励

根据规则，在数据块里的第一个交易是一个特定的交易,它创建了一个新的货币，由这个数据块的创建者拥有。这给支持网络的节点添加了一个激励，同时提供一个在整个循环里分布式发行货币的方法，没有中心权威去影响它们。稳定的、数量不断增加的新货币和挖金人花费资源添加黄金进如黄金循环系统一样。在这个例子里，处理器时间和电力是所需要花费的资源。
激励也可以通过交易费用获得。如果交易的一个输出值小于输入址，差值就是交易费，它作为激励被被添加进包含这个交易的数据块。整个循环的货币数量已经被预先设定，同时交易费作为交易的激励可以完全避免通货膨胀。
激励也可以有助于节点保持诚实。如果一个贪婪的攻击者有能力集合很多处理器能力超过诚实的节点，他要么选择从自己的交易里欺诈他人，要么使用它生成新的货币。他应该发现遵守规则获利更多，这样的规则有利于他联合其他人赚取新货币，超过了他削弱这个系统和损害自身的财富健康的有效性。

7 回收磁盘空间

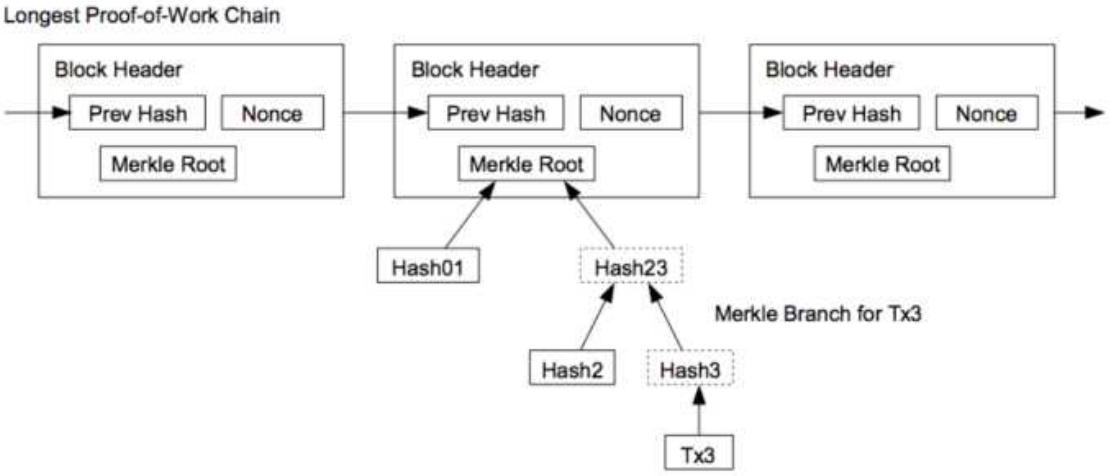
一个货币里最后的交易已经被足够多的数据块覆盖，那么这个支付交易之前的数据可以不再被使用以节省磁盘空间。为了在不打断数据块的哈希前提下促进它。交易被哈希进默克尔树（Merkle Tree），这样只这个数据块哈希的根需要被包含进来，老的数据块可以被压缩进树的接下的分支而拔除。内部的哈希不需要被存储



一个不含交易信息的数据块头部大概80字节。如果我们支持每十分钟生成一个数据块。80字节*6*24*365=4.2M/每年。2008年，每个通用的计算机都有2G内存。根据摩尔定律预测，每年增长1.2GB,数据头都被存储进内存也不是问题。

8. 简化的支付验证

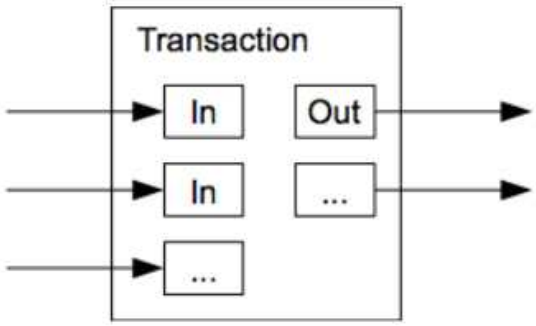
不需要运行一个完整的网路节点也可以认证支付，一个用户仅仅需要保存工作量网络的最长数据链的数据块头部的复本，他可通过在网络节点上排队等待直到他相信他自己已经得到了最长的链，并且包含交易的数据块已经被默克尔分支连接上。他不能检查他自己的交易，但是通过连接到链的某个位置，他能看到网络节点已经接受了这个数据，并且其后增加的数据块也证明网络节点已经接受了它。



例如，这个支付验证依靠尽可能诚实的节点控制网络，但是如果网络被一个拥有大量算力的攻击者控制，它是会容易受到攻击的。当网络节点可以确认它们自身的交易，这个简单的方法可以被一个攻击者的编造的交易来欺骗，只要攻击者拥有超过网络的算力。当网络节点侦测到一个无效的数据块，一个策略会保护反对者，他们将会从网络节点接收到警告，促使软件的用户去下载整个数据块，以确认被警告交易的一致性。支付频繁的商家还可以去运行它们自己的节点，以获得更独立的安全和更快的确认。

9 合并和拆分数据

尽管它可以控制单个货币的交易，但针对交易的每一分钱分开处理是很笨的方法。交易包含的多个输入和输出，应该允许数值的拆分和合并。通常要么就是从上一个更大的交易里单一输入，要么就是把多个输入合并成更小的数字，同时最多只有两个输出，一个负责支付，一个负责找零，如果有，则返回给发送者。

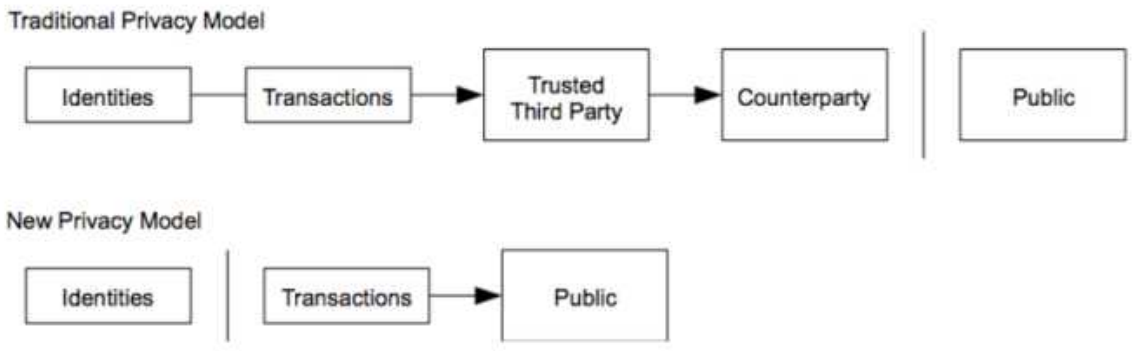


.

这里需要注意的是输出端。一个交易从几个交易而来，同时这些交易从更多交易而来，这不是问题。这里永远不需要去展开一个交易的历史完整复本。

10 隐私

传统银行的模式是给合作伙伴有限的访问权限，同时通过一个被信任的第三方来调用，以来查看一定级别的隐私。除了这个方法，维护隐私还需要通过打断信息流的一些地方，通过匿名的公共密钥，以公开需要的所有的交易。公众可以看到某人发送给其他人的数字，但是没有交易人的信息，这个很像股票交易所的信息发布级别，公众记录了单比的交易的时间和规模，但是不知道谁交易的。



作为一个额外的防火墙，同一个拥有者的每一比新交易可以连接一对新的配对密钥。一些连接还是不可避免的包含多个交易的输入，这必须暴露同一个拥有者过去的其他输入，这个风险是如果拥有者的密钥被暴露了，连接将暴露属于同一拥有者的其他交易。

11 计算

我们假设一个场景，一个攻击者试图去生成一个更快的链以替代诚实的链。甚至如果它完成的比较彻底，它抛开这个系统去随意改变，例如凭空创造一个值或者拿走从不属于他的钱，节点不会

接受一个无效的支付交易，诚实节点将永远不会接受包含他们的链。一个攻击者能做的仅仅是可以努力去改变他自己的交易，以从他最近的支付里把钱拿回来。
诚实链和攻击链的之间比赛的特征是一个二项分布的随机漫步。成功的事件是诚实的节点被一个数据块扩展，它的领先增加一个点，失败的事件是攻击者的链扩展一个数据块，差距减少一个点。

攻击者从一个给定的赤字追上成功的可能性和赌徒破产问题相似。假设一个赌徒从一个赤字开始拥有无限的信用，同时无限尝试的次数去赌以达到盈亏平衡。我们可以计算他达到盈亏平衡的可能性，那也就是一个攻击者追上诚实的链。如下所示。
p = 诚实节点发现下一个数据块的可能性
q = 攻击者发现下一个数据块的可能性
qz = 攻击者尝试从z数据块以后追上的可能性

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

我们假设p>q,可能性随着攻击者追上数据块的增加呈指数下降，随着概率和他做对，如果他不能在早期幸运的赶上，越往后，他的机会会变的很渺茫。
我们现在思考需要等待多久才能确认这个发送者不能改变交易。我们假设这个发送者是个攻击者，他想使接收者相信他已经把钱付给了他，然后过一会，他把钱再付给他自己。当发生时，接受者将接收到警告，但是发送者希望它迟些发生。
接收者在签名前生成一个新的配对密钥然后把很快把公钥给了发送者。这防止发送者在这个时间点前准备一个数据块链并开始持续的工作，直到他幸运的跑到前面，然后在这时执行这个交易。交易一经发出，不诚实的发送者就已经开始在一个并行包含了他交易的替代版本的链上秘密工作。

接收者等待直到这个交易已经被添加进一个数据块同时Z数据块已经被链接在它的后面。他不知道攻击者已经制造的数据块进展的准确数字，但是假设诚实的数据块按每个数据块的期望的平均值生成，攻击者可能的进展的期望值将呈柏松分布。

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

为了得到攻击者在这时追上的概率，我们用柏松分布密度乘以他可能在这个点上追上可能的概率的进展数：

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

重新整理避免分布无限循环的尾部求和

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

转换到C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;
}
```

运行一些结果，我们可以看到随着z值的增加，概率呈指数下降。

```
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

我们已经提出了一个不需要基于第三方信任的电子交易系统。我们从常用的包含数字签名的货币框架开始，虽然它提供了强大的控制力，但是在防止双重支付方面做的不完整。为了解决这个问题，我们提出了一个依靠工作量证明的点对点网络，用其记录一个公共的交易历史，如果诚实的节点控制了主要的处理能力，那么经过计算，攻击者希望修改记录的努力将变的不切实际。这个网络简单且具备鲁棒性。所有网络上的节点仅需要一点点的协调。它们不需要被认证，信息不需要路由到任何特别的地方，仅仅需要尽最佳效果传播。只需要接受它们离开时工作量网络的产生的数据链,计算节点可以随时加入和离开网络。它们用处理器能力投票，通过在数据块上扩展新的数据，以表示对数据块有效性的赞同，拒绝在数据块上扩展以拒绝无效数据块。任何需要的规则和奖励已经被整合进这个一致的机制且强制执行。

参考链接

[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.