# 比特币：一种点对点的电子现金系统

中本聪
satoshin@gmx.com
www.bitcoin.org

摘要。 一个纯粹的点对点电子现金版本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，但如果仍然需要一个可信任的第三方来防止双重支付，则主要优势将会丧失。

我们提出了使用点对点网络解决双重支付问题的方案。
网络通过将交易的时间戳哈希成一个持续的基于哈希的工作证明链，形成一个不可更改的记录。 最长的链不仅作为事件顺序的证明，还证明它来自最大的CPU算力池。 只要大多数CPU算力由不合作的节点控制，它们将生成最长的链并超过攻击者。 网络本身需要最少的结构。 消息以最佳努力的方式广播，节点可以随意离开和重新加入网络，接受最长的工作证明链作为他们离开期间发生的事情的证明。

## 1. 介绍

互联网上的商业几乎完全依赖于金融机构作为可信第三方来处理电子支付。 虽然该系统对大多数交易而言运作良好，但仍然存在基于信任模型的固有弱点。

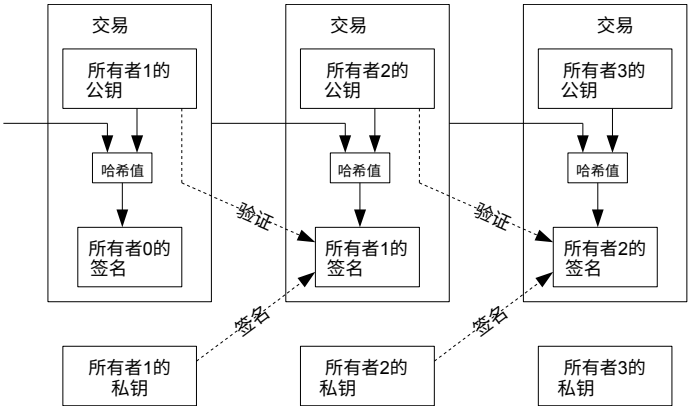
完全不可逆转的交易实际上是不可能的，因为金融机构无法避免调解争议。 调解的成本增加了交易成本，限制了最小实际交易规模，并且失去了进行非可逆支付的能力会对非可逆服务造成更广泛的成本。 由于可能发生逆转，信任的需求扩散开来。 商家必须对顾客保持警惕，要求他们提供比通常所需更多的信息。

某种程度上接受欺诈是不可避免的。 这些成本和支付的不确定性可以通过使用实体货币来避免，但在没有可信方的通信渠道上进行支付的机制不存在。

所需的是一种基于加密证明而不是信任的电子支付系统，允许任何两个愿意的方直接进行交易，而无需可信第三方。 计算上不可逆转的交易将保护卖家免受欺诈，而常规的托管机制可以轻松实施以保护买家。 在本文中，我们提出了一种解决双花问题的方案，该方案使用点对点分布式时间戳服务器来生成交易的时间顺序的计算证明。 只要诚实节点共同控制的CPU计算能力超过任何攻击节点的合作组，系统就是安全的。

## 2. 交易

我们将电子货币定义为一串数字签名。 每个所有者通过对前一笔交易的哈希和下一个所有者的公钥进行数字签名，并将其添加到货币的末尾来转移货币。 收款方可以验证签名以验证所有权链。

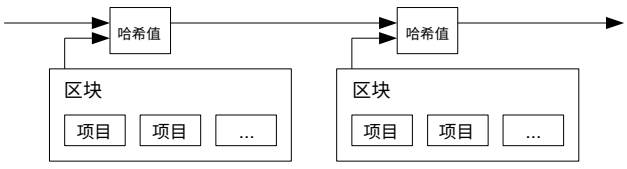



问题在于收款方无法验证所有者中是否有人重复花费了货币。 一个常见的解决方案是引入一个可信的中央机构或铸币厂，以检查每笔交易是否存在重复花费。 每次交易后，货币必须退回铸币厂以发行新的货币，只有直接由铸币厂发行的货币才能被信任不会被重复花费。
这种解决方案的问题在于整个货币系统的命运取决于运营铸币厂的公司，每笔交易都必须经过他们，就像银行一样。

我们需要一种方法让收款方知道前任所有者没有签署任何早期交易。 对于我们的目的来说，最早的交易才是有效的，所以我们不关心后来的重复花费尝试。 确认没有交易的唯一方法是了解所有交易。 在基于铸币厂的模型中，铸币厂知道所有交易并决定哪个先到达。 为了在没有可信方的情况下实现这一点，交易必须公开宣布[1]，并且我们需要一个参与者就接收到的交易顺序达成一致的系统。 收款方需要证明在每笔交易时，大多数节点都同意它是第一笔接收到的。

## 3. 时间戳服务器

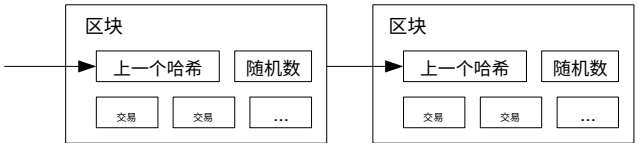
我们提出的解决方案始于一个时间戳服务器。 时间戳服务器通过对要进行时间戳的项目块进行哈希运算，并广泛发布哈希值，例如在报纸或Usenet帖子中[2-5]。 时间戳证明了数据必须在那个时间点之前存在，显然，为了进入哈希值中。 每个时间戳都包含前一个时间戳的哈希值，形成一个链条，每个额外的时间戳都加强了之前的时间戳。



## 4. 工作量证明

为了在点对点基础上实现分布式时间戳服务器，我们将需要使用类似于Adam Back的Hashcash [6]的工作量证明系统，而不是报纸或Usenet帖子。
工作证明涉及扫描一个数值，当进行哈希运算时，例如使用SHA-256，哈希值以一定数量的零位开始。 所需的平均工作量呈指数增长，可以通过执行单个哈希来验证所需的零位数。

对于我们的时间戳网络，我们通过递增块中的一个随机数来实现工作证明，直到找到一个值，使得块的哈希值具有所需的零位。 一旦CPU投入了满足工作证明的努力，块就无法更改，除非重新进行工作。 随后的块链接在其后，更改块的工作将包括重新进行所有后续块的工作。




工作证明还解决了在多数决策中确定代表的问题。 如果多数基于一个IP地址一票制，那么任何能够分配多个IP的人都可以破坏它。工作证明本质上是一个CPU一票制。 多数决策由最长的链表示，该链在其中投入了最大的工作证明努力。如果大多数CPU算力由诚实节点控制，诚实链将增长最快，并超过任何竞争链。 要修改过去的块，攻击者必须重新进行该块及其后续所有块的工作证明，然后赶上并超过诚实节点的工作。 我们将在后面展示，随着添加后续块，较慢的攻击者追赶上来的概率呈指数级下降。

为了弥补硬件速度的增加和随时间变化的节点运行兴趣的差异，工作量证明的难度由移动平均数决定，目标是每小时平均区块数。 如果生成得太快，难度会增加。

## 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区块中。
3）每个节点努力为其区块找到一个困难的工作量证明。
4）当一个节点找到一个工作量证明时，它将该区块广播到所有节点。
5）节点只有在其中的所有交易都有效且未被花费时才接受该区块。
6）节点通过使用接受的区块的哈希作为前一个哈希来工作，表达对该区块的接受。

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不同版本的下一个区块，一些节点可能会先收到其中一个。 在这种情况下，它们会处理首先收到的那个，但会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，将打破平局； 然后，正在处理另一个分支的节点将切换到更长的分支。

新的交易广播不一定需要到达所有节点。 只要它们到达了很多节点，它们很快就会进入一个区块。 区块广播也容忍丢失的消息。 如果一个节点没有收到一个区块，在它接收到下一个区块并意识到自己错过了一个区块时，它会请求该区块。

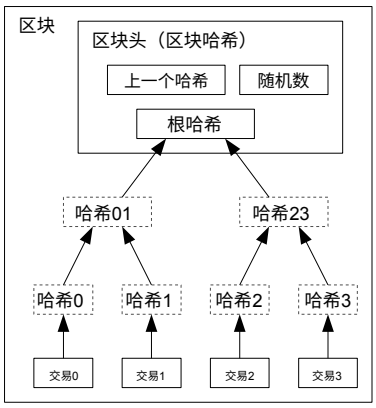
## 6. 激励

按照惯例，一个区块中的第一笔交易是一笔特殊的交易，它开始了一个由该区块的创建者拥有的新币。 这为节点支持网络增加了一个激励，并提供了一种最初将币分发到流通中的方式，因为没有中央机构来发行它们。
持续增加一定数量的新币类似于黄金矿工耗费资源将黄金投入流通。 在我们的情况下，耗费的是CPU时间和电力。
激励也可以通过交易费来资助。 如果一笔交易的输出价值小于其输入价值，差额将作为交易费添加到包含该交易的区块的激励价值中。 一旦预定数量的币进入流通，激励可以完全过渡到交易费，并完全没有通货膨胀。

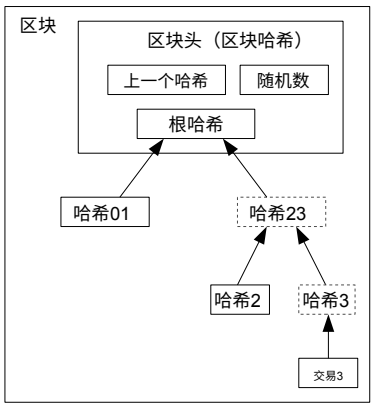
激励措施可能有助于鼓励节点保持诚实。 如果一个贪婪的攻击者能够比所有诚实节点拥有更多的CPU算力，他将不得不在使用它来欺诈性地窃取他的支付或使用它来生成新的比特币之间做出选择。 他应该发现按照规则进行游戏更有利可图，这些规则使他比其他人加起来获得更多的新比特币，而不是破坏系统和自己财富的有效性。

## 7. 回收磁盘空间

一旦一枚比特币的最新交易被足够多的区块深埋，之前的已花费交易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，交易被哈希在一个默克尔树中[7][2][5]，只有根哈希被包含在区块的哈希中。
旧的区块可以通过截断树的分支来压缩。 内部哈希不需要被存储。



在默克尔树中哈希的交易

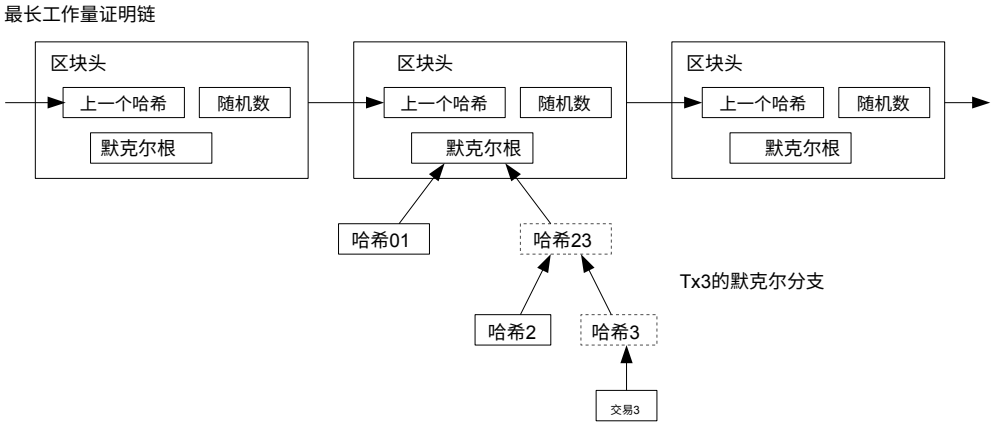


从区块中修剪掉交易0-2之后

一个没有交易的区块头大约为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 = 每年4.2MB。 随着计算机系统通常以2GB的RAM销售（截至2008年），并且摩尔定律预测每年增长1.2GB，即使区块头必须保留在内存中，存储也不应该成为问题。

## 8. 简化支付验证

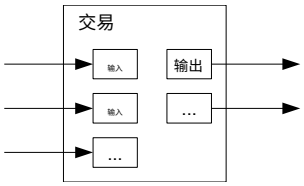
可以在不运行完整网络节点的情况下验证支付。 用户只需保留最长工作量证明链的区块头副本，可以通过查询网络节点获得，直到他确信拥有最长的链，并获得将交易与其所在区块时间戳链接的默克尔分支。他无法自行验证交易，但通过将其与链中的某个位置相关联，他可以看到网络节点已接受它，并且在其后添加的区块进一步确认网络已接受它。




因此，只要诚实节点控制网络，验证就是可靠的，但如果网络被攻击者控制，它就更容易受到攻击。 虽然网络节点可以为自己验证交易，但简化的方法可以被攻击者伪造的交易欺骗，只要攻击者能够继续压倒网络。 一种防范措施是接受网络节点检测到无效区块时发出的警报，促使用户软件下载完整的区块和被警报的交易以确认不一致性。 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. 合并和拆分价值

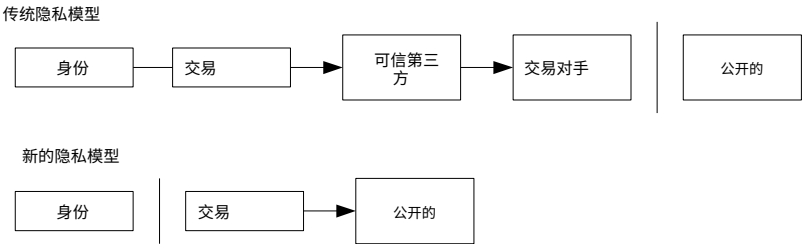
虽然可以单独处理硬币，但为每一分钱进行单独交易将会很不方便。 为了允许价值的拆分和合并，交易包含多个输入和输出。 通常情况下，要么只有一个来自较大先前交易的输入，要么是多个输入组合较小金额，并且最多有两个输出：一个用于支付，一个用于将找零（如果有的话）返回给发送者。




需要注意的是，在这里，一个交易依赖于多个交易，而这些交易又依赖于更多的交易，这种扇出并不是一个问题。 从来不需要提取交易历史的完整独立副本。

## 10. 隐私

传统银行模式通过限制信息访问仅限于相关方和可信第三方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，但通过在另一个地方中断信息流，仍然可以保持隐私：通过保持公钥匿名。 公众可以看到某人向他人发送一定金额，但没有信息将该交易与任何人联系起来。 这类似于股票交易所发布的信息水平，其中个别交易的时间和规模（即交易记录）是公开的，但不会透露交易方是谁。




作为额外的防火墙，每次交易都应使用新的密钥对，以防止它们与一个共同的所有者关联起来。 对于多输入交易，仍然无法避免一些关联，这必然会暴露它们的输入是由同一个所有者拥有的。 风险是，如果密钥的所有者被揭示，关联可能会揭示属于同一所有者的其他交易。

## 11. 计算

我们考虑一个攻击者试图比诚实链更快地生成一个替代链的情况。 即使这样做成功，它也不会使系统对任意更改敞开大门，比如无中生有地创造价值或者拿走从未属于攻击者的钱。 节点不会接受无效的交易作为支付，诚实的节点也永远不会接受包含这些交易的区块。 攻击者只能尝试更改自己的一笔交易以取回最近花费的钱。

诚实链和攻击者链之间的竞争可以被描述为一个二项随机漫步。 成功事件是诚实链被扩展一个区块，将其领先优势增加1，而失败事件是攻击者链被扩展一个区块，将差距减少1。

从给定的赤字追赶上来的攻击者的概率类似于赌徒的破产问题。 假设一个信用无限的赌徒从赤字开始，并进行无限次试验以达到盈亏平衡。 我们可以计算他达到盈亏平衡的概率，或者攻击者追赶上诚实链的概率，如下所示[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
$q_z$= 攻击者从落后z个区块追赶上的概率

$$q_z=\begin{cases} 1 & \text{如果 } p \le q \\ \text{则} q/p)z \ if \ p>q \end{cases}$$

根据我们的假设，$p > q$，随着攻击者需要追赶的区块数量增加，概率呈指数下降。 在不利的情况下，如果他在早期没有幸运的突进，随着他落后的距离越来越远，他的机会将变得微乎其微。

我们现在考虑一下，在新交易的接收者足够确定发送者无法更改交易之前，需要等待多长时间。 我们假设发送者是一个攻击者，他希望让接收者相信他支付了一段时间，然后在一段时间过去后将其切换为支付给自己。 当发生这种情况时，接收者将收到警报，但发送者希望为时已晚。

接收者在签名之前生成一个新的密钥对，并将公钥交给发送者。 这样可以防止发送者提前准备好一连串的区块，通过不断地工作直到他足够幸运地超前一段距离，然后在那一刻执行交易。 一旦交易被发送，不诚实的发送者会秘密地在一个并行的区块链上工作，其中包含他交易的另一个版本。

接收者等待直到交易被添加到一个区块中，并且在其后链接了$z$个区块。他不知道攻击者已经取得了多少进展，但假设诚实的区块按照每个区块的平均预期时间进行，攻击者的潜在进展将是一个泊松分布，其期望值为：

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

为了得到攻击者现在仍然能够追赶上的概率，我们将每个进展量的泊松密度与他能够从那一点追赶上的概率相乘：

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

重新排列以避免对分布的无限尾部求和...

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

转换为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;
}
```

通过运行一些结果，我们可以看到概率随着z的增加呈指数下降。

```
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
```

求解小于0.1%的P...

```
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。 我们从数字签名制作的硬币的常规框架开始，这提供了对所有权的强大控制，但是如果没有一种防止双重支付的方法，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证明的点对点网络，用于记录交易的公共历史，如果诚实节点控制了大多数CPU计算能力，那么这个历史将很快变得计算上不可行以防止攻击者更改。 这个网络在其非结构化的简单性中是强大的。 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位置，只需要尽力交付。 节点可以随意离开和重新加入网络，接受工作证明链作为他们离开期间发生的事情的证明。 他们通过他们的CPU计算能力进行投票，通过扩展有效块来表达他们的接受，并通过拒绝在无效块上工作来拒绝无效块。 任何需要的规则和激励都可以通过这种共识机制来执行。

## 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, 如何给数字文档打时间戳, In密码学杂志, vol 3, no 2, pages 99-111, 1991.

[4] D. Bayer, S. Haber, W.S. Stornetta, 提高数字时间戳的效率和可靠性, In序列*II*：通信、安全和计算机科学方法, pages 329-334, 1993.

[5] S. Haber, W.S. Stornetta, 为位串提供安全名称, In第四届*ACM*计算机与通信安全会议论文集, *p*ages 28-35, *Ap*ril 1997.

[6] A. Back, Hashcash - 一种防止拒绝服务的对策，http://www.hashcash.org/papers/hashcash.pdf，2002年。

[7] R.C. Merkle，公钥密码系统的协议，InProc。 *1980*年安全与隐私研讨会，IEEE计算机学会，*1*22-*1*33页，*1980*年4月。

[8] W. Feller，概率论及其应用导论，1957年。