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。 一个纯粹的点对点版本的电子现金系统将允许在线支付直接从一方发送到另一方，而无需经过金融机构。 数字签名提供了部分解决方案，但如果仍然需要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。
网络通过将交易时间戳哈希成基于工作证明的持续链，形成一个记录，如果不重新进行工作证明，则无法更改。 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算力由不合作攻击网络的节点控制，它们将生成最长的链并超越攻击者。 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他们离开时发生的事情的证据。

1. 引言

互联网上的商业几乎完全依赖于作为信任第三方的金融机构来处理电子支付。 虽然该系统对大多数交易运作良好，但仍然受到基于信任模型固有弱点的影响。

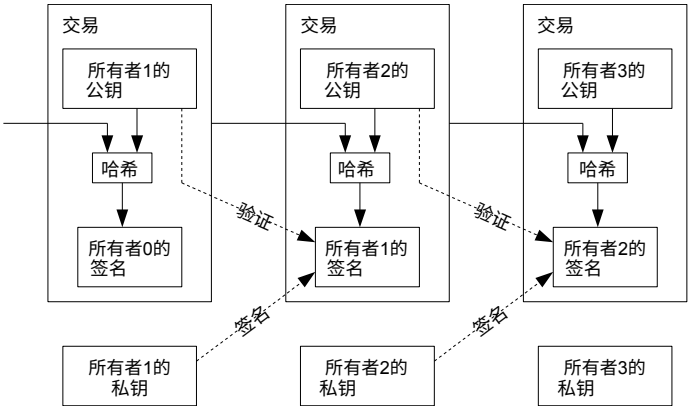
完全不可逆转的交易实际上并不可能，因为金融机构无法避免调解纠纷。 调解成本增加了交易成本，限制了最低实际交易规模，并切断了小额偶发交易的可能性，而且在无法进行不可逆转支付的情况下，还存在更广泛的成本。 随着可逆转的可能性，信任需求扩散。 商家必须谨慎对待他们的客户，为了比他们本来需要的更多信息而烦扰他们。

接受一定比例的欺诈是不可避免的。 这些成本和支付不确定性可以通过使用实体货币来避免，但是没有机制可以在没有信任方的情况下通过通信渠道进行支付。

所需的是基于密码学证明而不是信任的电子支付系统，允许任何两个愿意的当事人直接进行交易，而无需信任第三方。 计算上不可逆转的交易将保护卖家免受欺诈，而常规的担保机制可以轻松实施以保护买家。 在本文中，我们提出了使用点对点分布式时间戳服务器解决双重支付问题的方案，以生成交易的时间顺序的计算证明。 只要诚实的节点集体控制的CPU功率超过任何合作攻击节点组，系统就是安全的。

2. 交易

我们将电子硬币定义为一系列数字签名。 每个所有者通过数字签名前一笔交易的哈希和下一个所有者的公钥将硬币转移给下一个所有者，并将这些添加到硬币的末尾。 收款人可以验证签名以验证所有权链。



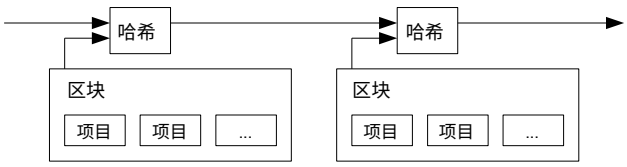
当然问题在于收款人无法验证所有者中是否有人双重花费了硬币。 一个常见的解决方案是引入一个受信任的中央机构，或者说是铸币厂，来检查每笔交易是否存在双重花费。 每次交易后，硬币必须退回铸币厂以发行新硬币，只有直接由铸币厂发行的硬币才能被信任不会被双重花费。

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，每笔交易都必须通过他们进行，就像银行一样。

我们需要一种方法让收款人知道之前的所有者没有签署任何早期交易。 对于我们的目的来说，最早的交易才是最重要的，所以我们不关心后来的双重花费尝试。 确认交易不存在的唯一方法是了解所有交易。 在基于铸币的模型中，铸币机构知晓所有交易并决定哪笔交易先到达。 为了在没有信任方的情况下实现这一点，交易必须公开宣布[1]，我们需要一个让参与者就交易接收顺序达成一致意见的系统。 收款方需要证明，在每笔交易时，大多数节点都同意这是第一笔收到的交易。

3. 时间戳服务器

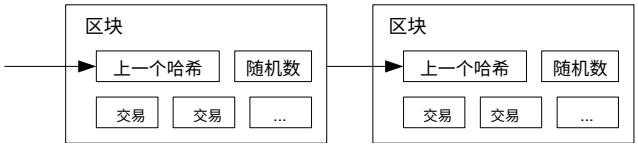
我们提出的解决方案始于一个时间戳服务器。 时间戳服务器通过对待时间戳的一组项目进行哈希运算，并广泛发布哈希值，比如在报纸或Usenet帖子中[2-5]。 时间戳证明了数据必须在那个时间点存在，显然，为了进入哈希值。 每个时间戳在其哈希中包含前一个时间戳，形成一个链条，每个额外的时间戳都加强之前的时间戳。



4. 工作量证明

为了在点对点基础上实现一个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。 所需的平均工作量与所需的零位数量呈指数关系，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，我们通过递增块中的一个随机数（nonce）来实现工作量证明，直到找到一个数值，使得块的哈希值具有所需的零位。 一旦CPU的工作量已经花费在使其满足工作证明上，区块就不能在不重新做工作的情况下被更改。 随后的区块链接在其后，更改区块的工作将包括重新做所有在其后的区块。



工作证明也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，那么任何能够分配多个IP的人都可以颠覆它。工作证明本质上是一个CPU一票。 多数决策由最长的链表示，其中投入了最大的工作证明。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。 要修改过去的区块，攻击者必须重新做区块和其后所有区块的工作证明，然后赶上并超过诚实节点的工作。 我们将在稍后展示，随着后续区块的添加，较慢攻击者追赶的概率呈指数级下降。

为了补偿硬件速度的增加和随时间变化的节点运行兴趣，工作量证明的难度由移动平均值确定，以达到每小时平均区块数的目标。 如果生成速度过快，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尚未花费时才接受该区块。
6) 节点通过努力创建链中的下一个区块来表达对该区块的接受，使用接受的区块的哈希作为前一个哈希。

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，它们会处理收到的第一个分支，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；那些正在处理另一个分支的节点将转而处理更长的那个分支。

新交易广播不一定需要到达所有节点。 只要它们到达许多节点，它们很快就会进入一个区块。 区块广播也能容忍丢失的消息。 如果一个节点没有收到一个区块，它将在收到下一个区块时请求它，并意识到自己错过了一个。

6. 激励

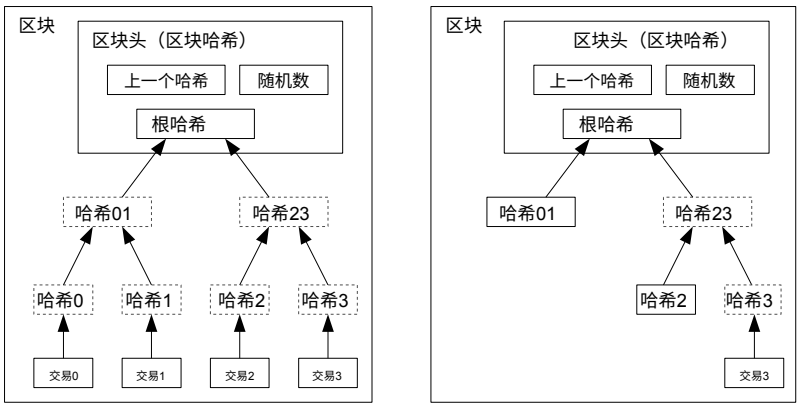
按照惯例，区块中的第一笔交易是一笔特殊的交易，开始一个由区块创建者拥有的新币。 这为节点支持网络增加了激励，并提供了一种最初将硬币分发到流通中的方式，因为没有中央机构来发行它们。
稳定地增加一定数量的新硬币类似于黄金矿工耗费资源将黄金添加到流通中。 在我们的情况下，耗费的是CPU时间和电力。

激励也可以通过交易费用来资助。 如果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。 一旦预定数量的硬币进入流通，激励就可以完全过渡到交易费用，并完全无通货膨胀。

激励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集合比所有诚实节点更多的CPU算力，他将不得不在利用它欺诈性地夺回他的支付，或者利用它生成新的硬币之间做出选择。 他应该发现遵守规则更有利可图，这些规则使他比其他人加起来获得更多的新硬币，而不是破坏系统和自己财富的有效性。

7. 回收磁盘空间

一旦一个硬币中的最新交易被足够多的区块埋藏，它之前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，交易被哈希在默克尔树中[7][2][5]，只有根被包含在区块的哈希中。
旧区块可以通过截断树的分支来压缩。 内部哈希值不需要被存储。

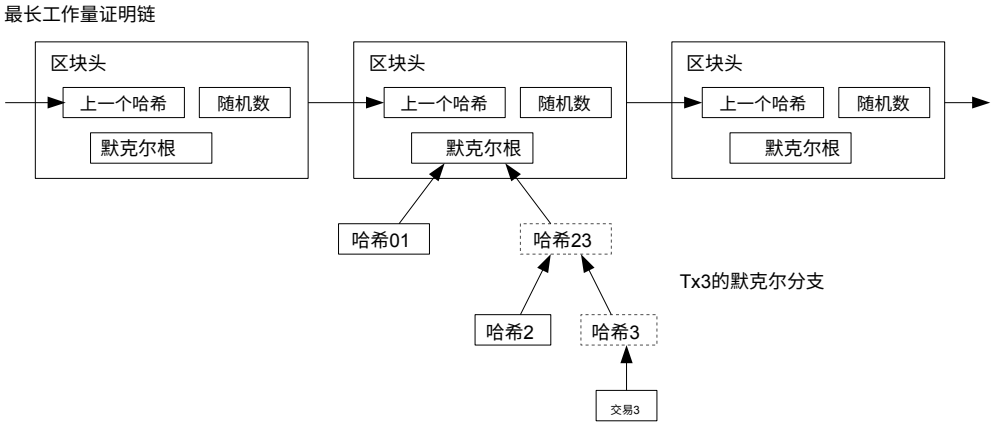


在默克尔树中哈希的交易　　从区块中修剪Tx0-2之后

一个没有交易的区块头大约是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。 2008年计算机系统通常配备2GB的RAM，而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，存储也不应该是一个问题。

8. 简化支付验证

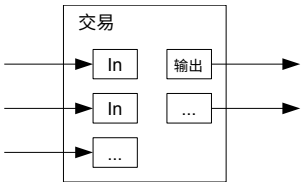
可以在不运行完整网络节点的情况下验证支付。 用户只需保留最长工作量证明链的区块头副本，他可以通过查询网络节点获得，直到他确信自己拥有最长的链，并获取将交易链接到其时间戳所在区块的默克尔分支。他无法自行检查交易，但通过将其链接到链中的某个位置，他可以看到网络节点已接受它，并在其后添加的区块进一步确认网络已接受它。



因此，只要诚实的节点控制网络，验证就是可靠的，但如果网络被攻击者控制，就会更容易受到攻击。 虽然网络节点可以为自己验证交易，但简化的方法可能会被攻击者伪造的交易欺骗，只要攻击者能继续控制网络。 一种防范措施是接受网络节点发出的警报，当它们检测到无效区块时，提示用户的软件下载完整的区块和警报的交易以确认不一致性。 接收频繁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. 价值的合并和拆分

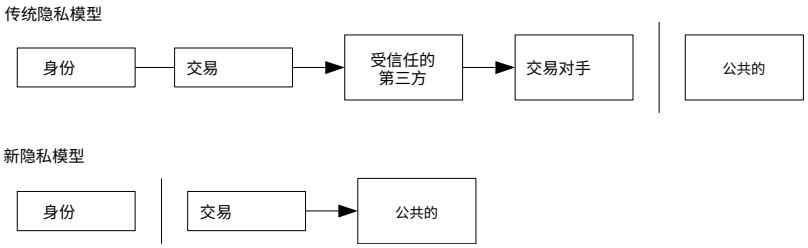
虽然可以单独处理硬币，但为每一分钱进行单独交易将会很不方便。 为了允许价值的拆分和合并，交易包含多个输入和输出。 通常情况下，要么是来自较大之前交易的单个输入，要么是结合较小金额的多个输入，最多有两个输出：一个用于付款，另一个将找零（如果有的话）返回给发送者。



应该注意到，扇出，即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，在这里并不是一个问题。 从未有必要提取一个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涉及的各方和受信任的第三方，实现了一定程度的隐私。 必须公开宣布所有交易，这种方法排除了这种隐私保护方式，但可以通过在另一个地方中断信息流来保持隐私：通过保持公钥匿名。 公众可以看到某人向另一个人发送金额，但没有信息将交易与任何人联系起来。 这类似于股票交易所发布的信息级别，其中个别交易的时间和规模，“交易记录”，是公开的，但不会透露交易双方是谁。



作为额外的防火墙，每笔交易都应使用新的密钥对，以防止它们被关联到一个共同的所有者。 一些关联仍然是不可避免的，特别是在多输入交易中，这些交易必然会透露它们的输入是由同一所有者拥有的。 风险在于，如果一个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替代链的情景。 即使这样做成功，也不会使系统对任意更改敞开大门，比如创造价值或窃取从未属于攻击者的资金。 节点不会接受无效交易作为支付，而诚实节点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，以取回最近花费的资金。

诚实链和攻击者链之间的竞争可以被描述为二项随机漫步。成功事件是诚实链被延长一个区块，其领先优势增加+1，失败事件是攻击者链被延长一个区块，缩小差距-1。

攻击者从给定的赤字追赶上来的概率类似于赌徒破产问题。 假设一个信用无限的赌徒从赤字开始，并可能进行无限次试验以达到盈亏平衡。 我们可以计算他是否会达到盈亏平衡的概率，或者攻击者是否会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个区块追赶上来的概率

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

鉴于我们的假设$p > q$，随着攻击者需要追赶的区块数量增加，概率呈指数下降。 面对不利的情况，如果他在早期没有幸运的突进，随着他落后的步伐越来越大，他的机会会变得微乎其微。

我们现在考虑接收新交易的接收方需要等待多长时间才能足够确定发送方无法更改交易。 我们假设发送方是一个想要让接收方相信他支付了一段时间，然后在一段时间后将其切换为支付给自己的攻击者。 当这种情况发生时，接收方将收到警报，但发送方希望为时已晚。

接收方生成一个新的密钥对，并在签名之前不久将公钥提供给发送方。 这可以防止发送方通过不断地工作来提前准备一系列区块，直到他幸运地领先一段距离，然后在那一刻执行交易。 一旦交易发送出去，不诚实的发送方就会秘密地开始在一个包含他交易的替代版本的并行链上工作。

接收方等待直到交易被添加到一个区块中，并且 z个区块在其后被链接。他不知道攻击者已经取得了多少进展，但假设诚实的区块花费了每个区块的平均预期时间，攻击者的潜在进展将是一个带有期望值的泊松分布：

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

为了计算攻击者现在仍然能够赶上的概率，我们将每个可能的进度量的泊松密度与他能够从那一点赶上的概率相乘：

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

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;
}
```

运行一些结果，我们可以看到概率随着z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
```

解决P小于0.1%的问题...

```
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

我们提出了一种无需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严格控制，但如果没有防止双重支付的方法，这一框架就是不完整的。 为了解决这个问题，我们提出了使用工作证明的点对点网络，记录交易的公共历史，如果诚实节点控制大多数CPU算力，那么对于攻击者来说很快变得计算上不可行。 网络在其非结构化的简单性中是强大的。 节点同时工作，几乎没有协调。 它们不需要被识别，因为消息不会被路由到任何特定位置，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它们离开时发生的事情的证据。 它们通过CPU算力投票，通过扩展有效区块来表达对其接受的接受，并通过拒绝对无效区块进行工作来拒绝无效区块。 任何必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, "如何给数字文档加时间戳," 在密码学杂志, 第3卷, 第2期, 1991年, 页码99-111.

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

[5] S. Haber, W.S. Stornetta, "比特串的安全名称," 在第*4*届*ACM*计算机与通信安全会议论文集, 1997年*4*月, 页码28-35.

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

[7] R.C. Merkle, "公钥密码系统的协议," 在*1980*年安全与隐私研讨会论文集, IEEE计算机学会, 1980年4月, 页码122-133.

[8] W. 费勒, "概率论及其应用导论," 1957.