比特币：点对点电子现金系统

中本聪
satoshin@gmx.com
www.bitcoin.org

摘要纯粹的点对点版电子现金可使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的第三方来防止双重消费，则会失去主要的好处。我们提出了一种利用点对点网络解决双重消费问题的方案。网络通过将交易散列到一个持续的基于散列的工作证明链中来为交易打上时间戳，这样就形成了一个记录，如果不重做工作证明，就无法更改该记录。最长的链不仅可以证明所见证的事件顺序，还可以证明它来自最大的 CPU 能力池。只要大部分 CPU 能力由不合作攻击网络的节点控制，它们就能生成最长的链，并超越攻击者。网络本身的结构要求极低。信息在尽最大努力的基础上进行广播，节点可以随意离开和重新加入网络，并接受最长的工作证明链作为它们离开时发生的事情的证明。

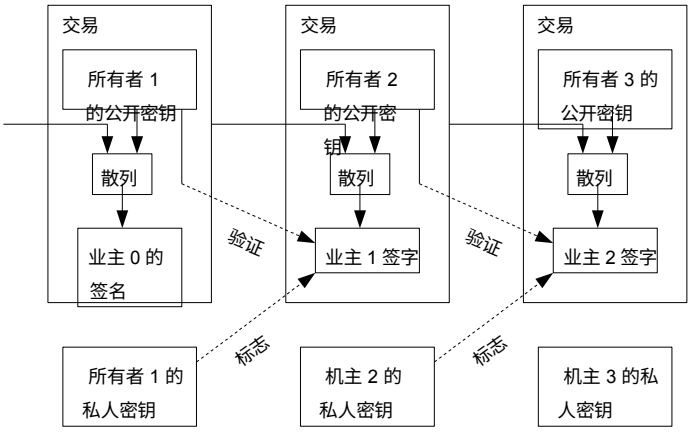
1. 导言

互联网商务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。虽然该系统在大多数交易中运作良好，但仍存在基于信任模式的固有缺陷。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。调解的成本增加了交易成本，限制了最小的实际交易规模，并 切断了小额临时交易的可能性。有了逆转的可能性，对信任的需求就会增加。商户必须对客户保持警惕，向他们索取更多的信息。一定比例的欺诈是不可避免的。使用实物货币可以避免这些成本和支付的不确定性，但没有任何机制可以在没有受信任方的情况下通过通信渠道进行支付。

我们需要的是一个基于加密证明而非信任的电子支付系统，允许任何愿意交易的双方直接进行交易，而无需信任的第三方。在计算上不可能逆转的交易可以保护卖方免受欺诈，而常规的托管机制可以很容易地保护买方。在本 文 中，我们提出了一种解决重复消费问题的方法，利用点对点分布式时间戳服务器生成交易时间顺序的计算证明。只要诚实节点集体控制的 CPU 能力超过任何合作的攻击者节点群，该系统就是安全的。

2. 交易

我们将电子硬币定义为数字签名链。每位所有者通过对上一笔交易的哈希值和下一位所有者的公开密钥进行数字签名，并将其添加到硬币的末尾，从而将硬币转移给下一位所有者。收款人可以通过验证签名来验证所有权链。

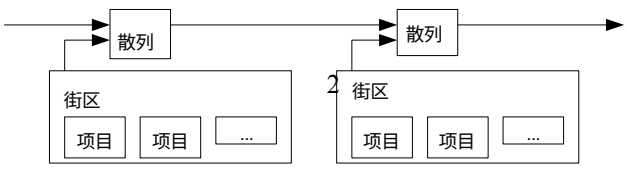


当然，问题在于收款人无法确认其中一位所有者是否重复消费了硬币。一种常见的解决方案是引入一个可信的中央机构或造币厂，负责检查每笔交易是否存在重复消费。每次交易后，硬币必须返回造币厂发行新硬币，只有造币厂直接发行的硬币才不会被重复消费。这种解决方案的问题在于，整个货币系统的命运取决于经营铸币厂的公司，每笔交易都必须经过他们，就像银行一样。

我们需要一种方法，让收款人知道之前的所有者没有签署任何早期交易。就我们的目的而言，最早的交易才是最重要的，因此我们并不关心后来是否有人试图重复消费。确认没有交易的唯一方法就是了解所有交易。在基于铸币厂的模式中，铸币厂知道所有交易，并决定哪些交易先到。要在没有可信方的情况下做到这一点，必须公开宣布交易[1]，而且我们需要一个系统，让参与者就收到交易的顺序达成一致。收款人需要证明，在每笔交易发生时，大多数节点都同意它是最先收到的。

3. 时间戳服务器

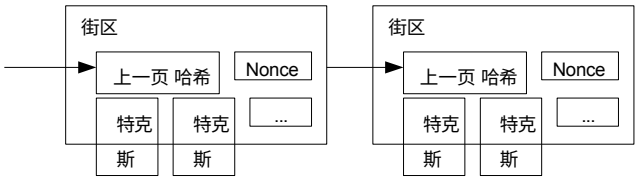
我们提出的解决方案从时间戳服务器开始。时间戳服务器的工作原理是获取待时间戳项目块的哈希值，并广泛发布该哈希值，例如在报纸或 Usenet 上发布 [2-5]。显然，时间戳证明数据必须在当 时 存在，才能进入哈希值。每个时间戳都会在哈希值中包含前一个时间戳，从而形成一个链条，每一个额外的时间戳都会加强之前的时间戳。



4. 工作证明

要在点对点基础上实现分布式时间戳服务器，我们需要使用类似 Adam Back 的 Hashcash [6] 的工作证明系统，而不是报纸或 Usenet 上的帖子。工作证明包括扫描一个值，该值在散列（如 SHA-256）时，散列以若干零比特开头。所需的平均工作量与所需的零位数量成指数关系，只需执行一次散列即可验证。

对于我们的时间戳网络，我们通过递增区块中的 nonce 来实现工作证明，直到找到一个值，使区块的哈希值达到所需的零位。一旦 CPU 为满足工作证明的要求付出了努力，如果不重做工作，就无法更改区块。由于后面的区块都是链式的，因此更改该区块的工作将包括重做后面的所有区块。



工作证明还解决了在多数决策中确定代表性的问题。如果多数决定基于一个 IP 地址一票制，那么任何能够分配许多 IP 地址的人都可能颠覆多数决定。工作证明本质上就是一 CPU 一票。最长的链代表了大多数人的决定，该链投入的工作量最大。如果大部分 CPU 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。要修改过去的一个区块，攻击者就必须重做该区块及其后所有区块的工作量证明，然后赶上并超过诚实节点的工作量。我们稍后会证明，随着后续区块的增加，速度较慢的攻击者赶上的概率会以指数形式递减。

为了补偿硬件速度的增加和运行节点兴趣的变化，工作证明的难度是由一个移动平均值决定的，目标是每小时的平均区块数。如果生成速度过快，难度就会增加。

5. 网络

运行网络的步骤如下：

1) 新交易会广播到所有节点。
2) 每个节点将新交易收集到一个区块中。
3) 每个节点都在为自己的区块寻找一个高难度的工作量证明。
4) 当一个节点发现一个工作证明时，它会向所有节点广播这个区块。
5) 只有当区块中的所有交易都是有效的，且尚未花费时，节点才会接受该区块。

6) 节点会使用已接受区块的哈希值作为前一个哈希值，创建链中的下一个区块，以表示接受该区块。

节点总是认为最长的链才是正确的链，并会继续扩展它。如果两个节点同时广播下一个区块的不同版本，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版本，但会保存另一个分支，以防其变长。当找到下一个工作证明时，其中一个分支变长，这种平局就会被打破；这时，正在处理另一个分支的节点就会转而处理较长的分支。

新交易广播不一定要到达所有节点。只要它们能到达许多节点，很快就能进入一个区块。区块广播还能容忍丢弃的信息。如果节点没有收到区块，当它收到下一个区块并意识到自己错过了一个区块时，就会请求接收。

6. 奖励

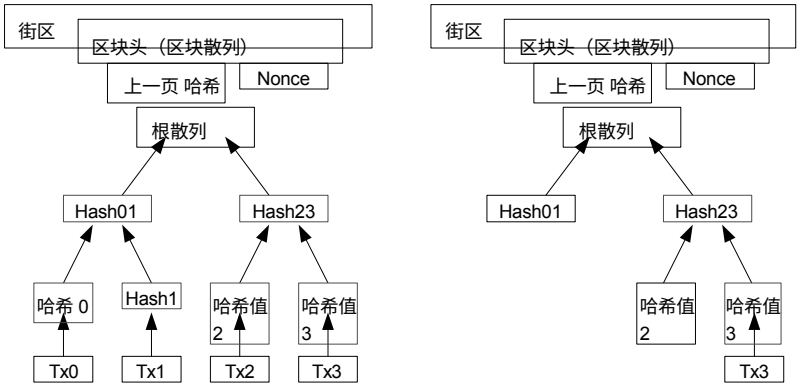
按照惯例，一个区块中的第一笔交易是一笔特殊交易，用于启动由区块创建者拥有的新币。这为节点提供了支持网络的动力，并提供了一种初始分配流通硬币的方式，因为没有中央机构来发行硬币。新币数量的稳定增加类似于黄金矿工耗费资源将黄金加入流通。在我们的案例中，消耗的是 CPU 时间和电力。

奖励也可以用交易费来资助。如果一笔交易的输出值小于输入值，差额就是交易费，这笔交易费会加到包含该笔交易的区块的奖励值上。一旦有预定数量的硬币进入流通，奖励就可以完全过渡到交易费，并且完全不受通货膨胀的影响。

这种激励措施有助于鼓励节点保持诚实。如果一个贪婪的攻击者比所有诚实的节点都拥有更多的 CPU 能力，那么他就必须做出选择，是用这些能力通过偷回付款来欺骗人们，还是用这些能力来产生新的硬币。他应该会发现，遵守规则（这种规则有利于他获得比其他人加起来都多的新币）比破坏系统和自己财富的有效性更有利可图。

7. 回收磁盘空间

一旦一个币的最新交易被埋在足够多的区块下，在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值的情况下实现这一目的，交易在梅克尔树（Merkle Tree）[7][2][5] 中进行哈希，只有根部包含在区块的哈希值中。然后，可以通过截断树的分支来压缩旧区块。内部哈希值无需存储。

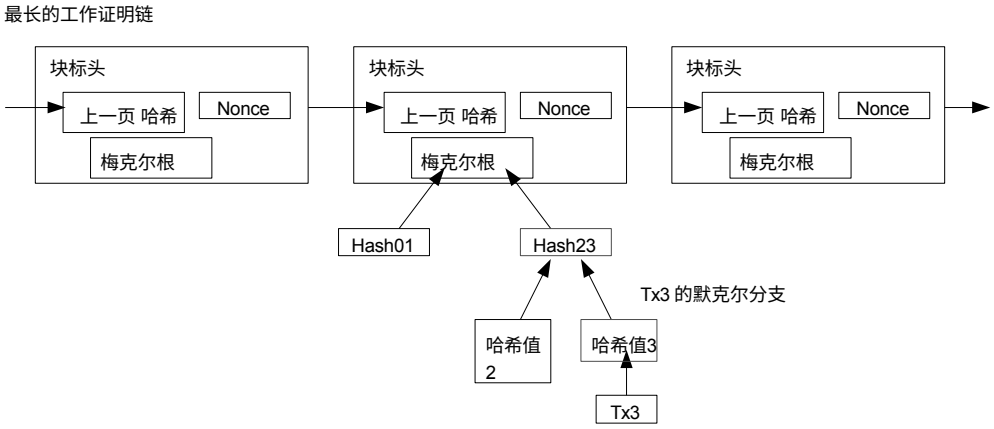


从区块中剪除 Tx0-2 后，在梅克尔　　树中对交易进行散列

一个没有交易的区块头大约为 80 字节。如果假设每 10 分钟产生一个区块，那么每年的数据量为 80 字节 * 6 * 24 * 365 = 4.2MB。截至 2008 年，计算机系统在销售时通常配备 2GB 内存，而摩尔定律预测目前的增长速度为
每年 1.2GB，即使块标头必须保存在内存中，存储也不成问题。

8. 简化付款验证

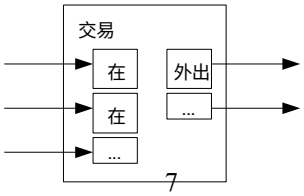
无需运行完整的网络节点就可以验证支付。用户只需保存一份最长工作证明链的区块头副本（他可以通过查询网络节点获得该副本，直到他确信自己拥有最长的链为止），并获得将交易与时间戳所 在 区 块 链接起来的梅克尔分支。他无法亲自检查该交易，但通过将其链接到链中的某个位置，他可以看到某个网络节点已经接受了该交易，而在该交易之后添加的区块则进一步确认了网络已经接受了该交易。



因此，只要诚实的节点控制着网络，验证就是可靠的，但如果网络被攻击者控制，验证就会变得更加脆弱。虽然网络节点可以自行验证交易，但只要攻击者能继续控制网络，简化方法就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，当网络节点检测到无效区块时，接受网络节点发出的警报，提示用户的软件下载完整的区块和警报交易，以确认不一致之处。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速的验证。

9. 合并和拆分价值

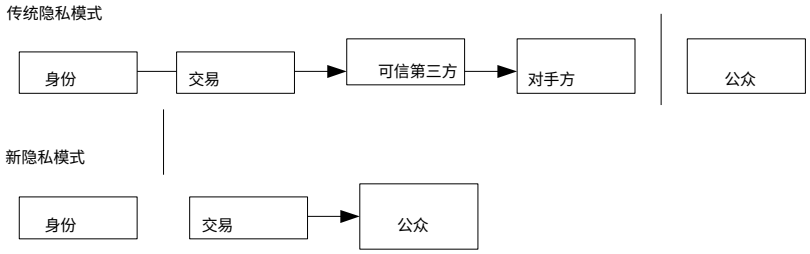
虽然可以单独处理硬币，但为转账中的每一美分进行单独交易会很不方便。为了能够分割和合并价值，交易包含多个输入和输出。通常情况下，会有一个来自先前较大交易的单一输入或多个较小金额组合的输入，最多会有两个输出：一个用于支付，另一个将零钱（如有）返还给发送方。



需要注意的是，这里不存在扇出（fan-out）问题，即一个事务依赖于多个事务，而这些事务又依赖于更多的事务。我们永远不需要提取事务历史的完整独立副本。

10. 隐私权

传统的银行模式通过限制只有相关方和可信第三方才能获取信息来实现一定程度的隐私保护。由于必须公开宣布所有交易，因此无法采用这种方法，但仍可以通过在另一个地方切断信息流来维护隐私：保持公开密钥的匿名性。公众可以看到某人向其他人发送了一笔金额，但没有将交易与任何人联系起来的信息。这与证券交易所发布信息的程度类似，在证券交易所，单笔交易的时间和规模，即 "带子"，都是公开的，但不告诉交易双方是谁。



作为额外的防火墙，每笔交易都应使用新的密钥对，以防 止 它们被链接到同一个所有者。在多输入交易中，某些链接仍然是不可避免的，因为这些交易必然会暴露其输入是由同一个所有者拥有的。这样做的风险是，如果密钥所有者身份暴露，链接可能会暴露属于同一所有者的 其他交易。

11. 计算

我们考虑的情况是，攻击者试图以比诚实链更快的速度生成另一条链。即使做到了这一点，系统也不会被任意改变，比如凭空创造价值或拿走从未属于攻击者的钱。节点不 会 接受无效交易作为支付，诚实节点也不会接受包含无效交易的区块。攻击者只能试图更改自己的交易，取回最近花掉的钱。

诚实链和攻击链之间的竞赛可以用二项式随机漫步来描述。成功事件是诚实链扩展了一个区块，使其领先优势增加 +1 ；失败事件是攻击者链扩展了一个区块，使差距缩小 -1 。

攻击者从给定赤字追上的概率类似于赌徒的毁灭问题。假设一个拥有无限信用的赌徒一开始就处于亏损状态，为了达到收支平衡，他可能会进行无限次赌博。我们可以计算出他达到盈亏平衡的概率，或者攻击者追上诚实链的概率[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 z 个街区后追上的概率

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

假设 $p > q$，随着攻击者需要追赶的棋块数量的增加，概率会以指数形式下降。在这种不利的情况下，如果攻击者不能在早期幸运地向前猛冲，那么随着他的落后，他的机会就会变得微乎其微。

现在我们要考虑的是，新交易的收款人需要等待多长时间才能充分确定发款人无法更改交易。我们假设发件人是一个攻击者，他想让收件人相信他给了他一段时间的钱，然后在一段时间过去后再把钱转还给自己。当这种情况发生时，接收方会收到警报，但发送方希望为时已晚。

接收方生成一个新的密钥对，并在签名前不久将公钥交给发送方。这样，发送方就无法提前准备好区块链，他只能不停地处理区块链，直到幸运地提前足够长的时间，然后再执行交易。一旦交易被发送，不诚实的发送者就会开始秘密处理一条平行链，其中包含其交易的另一个版本。

接收者要等到交易被添加到一个区块，并且在它之后有 z 个区块被链接。他不知道攻击者所取得的确切进展，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具有预期值的泊松分布：

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

为了得到攻击者现在仍能赶上的概率，我们将他可能取得的每一点进展的泊松密度乘以他能从这一点赶上的概率：

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

k 区块

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 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 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

我们提出了一个不依赖信任的电子交易系统。我们从由数字签名制成的硬币的常规框架入手，该框架提供了对所有权的有力控制，但如果没有防止重复消费的方法，则是不完整的。为了解决这个问题，我们提出了一个点对点网络，使用工作量证明来记录公开的交易历史， 如果诚实节点控制了大部分 CPU 处理能力，攻击者要改变这种历史很快就会变得不切实际。该网络因其非结构化的简单性而强大。节点同时工作，几乎不需要协调。它们不需要被识别，因为信息不会被路由到任何特定的地方，只需尽力传递即可。节点可以随 意离开和重新加入网络，并接受工作证明链作为它们离开时所发生事情的证明。它们用自己的 CPU 能力投票，通过扩展有效区块来表示接受有效区块，通过拒绝工作来拒绝无效区块。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。

参考资料

[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：《提高数字时间戳的效率和可靠性》，《*序列 II：通信、安全和计算机科学方法*》，第 329-334 页，1993 年。

[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--一种拒绝服务反措施"，http://www.hashcash.org/papers/hashcash.pdf，2002 年。

[7] R.C. Merkle，"公钥密码系统协议"，《*1980 年安全与隐私研讨会论文集*》，电气和电子工程师学会计算机协会，第 122-133 页，1980 年 4 月。

[8] W.费勒：《概率论及其应用导论》，1957 年。