**比特币：点对点电子现金系统**

中本聪satoshin@gmx.com

www.bitcoin.org

**摘要：**纯点对点版本的电子现金将允许在线支付从一方直接发送到另一方，而无需通过金融机构。数字签名提供了解决方案的一部分，但是如果仍然需要可信的第三方来防止重复支出，那么主要的好处就丧失了。提出了一种利用对等网络解决双重开销问题的方法。网络通过将事务散列到一个基于散列的工作证明的持续链中来给事务添加时间戳，形成一个在不重做工作证明的情况下无法更改的记录。最长的链不仅可以证明所见证的事件序列，还可以证明它来自于最大的CPU资源池。只要大部分CPU能量由不合作攻击网络的节点控制，它们就会产生最长的链并超过攻击者。网络本身需要最小的结构。消息在尽力而为的基础上进行广播，节点可以随意离开和重新加入网络，接受最长的工作链证明作为它们离开时发生的事情的证明。

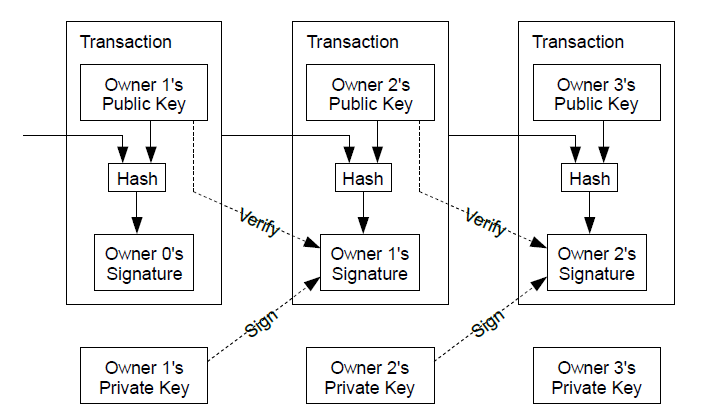
# 1.       介绍

互联网上的商业几乎完全依赖金融机构作为可信的第三方来处理电子支付。虽然该系统对于大多数交易都能很好地工作，但它仍然存在基于信任模型的固有弱点。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，限制了最低实际交易规模，切断了小规模临时交易的可能性，而且失去对不可逆服务进行不可逆支付的能力也会产生更大的成本。由于有逆转的可能，对信任的需求蔓延开来。商家必须提防他们的顾客，因为他们需要更多的信息。一定比例的欺诈被认为是不可避免的。这些成本和支付的不确定性可以通过使用实物货币亲自避免，但没有一种机制可以在没有可信方的情况下通过通信渠道进行支付。

现在需要的是一个基于密码证明而不是信任的电子支付系统，允许任何两个愿意的人直接与对方进行交易，而不需要可信的第三方。在计算上无法撤销的交易将保护卖方免受欺诈，并且可以很容易地实施常规的托管机制来保护买方。在本文中，我们提出了一个解决双重支出问题的方法，使用对等分布式时间戳服务器来生成事务时间顺序的计算证明。只要诚实的节点共同控制比任何合作的攻击者节点组更多的CPU能力，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。每一个拥有者通过对上一次交易的哈希值和下一个拥有者的公钥进行数字签名并将其添加到硬币的末尾，将硬币转移到下一个拥有者。收款人可以验证签名以验证所有权链。

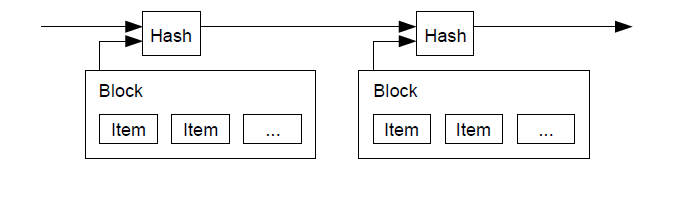


当然，问题是受款人无法证实其中一位主人没有双倍消费硬币。一个常见的解决方案是引入一个可信任的中央机构（mint），检查每笔交易的双倍支出。每次交易后，硬币必须归还铸币厂才能发行新硬币，只有直接从铸币厂发行的硬币才可以放心不被重复使用。这种解决方案的问题在于，整个货币体系的命运取决于运营铸币厂的公司，每一笔交易都必须通过铸币厂，就像银行一样。

我们需要一个方法让收款人知道，以前的业主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交易，所以我们不关心以后的加倍消费尝试。确认没有事务的唯一方法是了解所有事务。在以铸币厂为基础的模型中，铸币厂知道所有的交易，并决定哪个交易最先到达。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]，我们需要一个系统，让参与者就接收顺序的单一历史达成一致。收款人需要证明，在每次交易时，大多数节点都同意它是第一次收到的。

# 3.       时间戳服务器

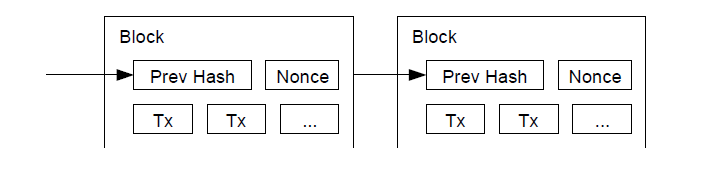
我们建议的解决方案从时间戳服务器开始。时间戳服务器的工作原理是对要加时间戳的项目块进行散列，并广泛发布散列，例如在报纸或Usenet帖子中[2-5]。时间戳证明数据必须在那个时候存在，显然，为了进入散列。每个时间戳在其散列中包括前一个时间戳，形成一个链，每个附加的时间戳加强它之前的时间戳。



# 4.       工作证明

为了在点对点的基础上实现分布式时间戳服务器，我们需要使用类似于Adam Back的Hashcash[6]的工作证明系统，而不是报纸或Usenet posts。工作证明包括扫描一个值，该值在散列时（如使用SHA-256）以零位开始。所需的平均功与所需的零位数成指数关系，可以通过执行单个散列来验证。

对于我们的时间戳网络，我们通过增加块中的nonce来实现工作证明，直到找到一个值，该值为块的散列提供所需的零位。一旦花费CPU的精力使其满足工作证明，就不能在不重做工作的情况下更改块。由于后面的块是链接在它后面的，因此更改块的工作将包括重做它后面的所有块。



工作证明还解决了在多数决策中确定代表性的问题。如果多数是基于一个IP地址一票表决，它可能被任何人能够分配许多IP颠覆。工作证明基本上是一票。多数决策由最长链表示，最长链具有投入的最大工作量证明。如果大部分CPU能力由诚实节点控制，那么诚实链将增长最快，超过任何竞争链。要修改过去的块，攻击者必须重新对该块及其之后的所有块进行工作证明，然后赶上并超过诚实节点的工作。稍后我们将展示，随着后续块的添加，较慢的攻击者追赶的概率呈指数级减小。

为了补偿硬件速度的提高和对运行节点的兴趣随时间的变化，工作难度的证明是通过以每小时平均块数为目标的移动平均数来确定的。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） 新事务将广播到所有节点。

2） 每个节点将新事务收集到一个块中。

3） 每个节点都致力于为其块找到一个困难的工作证明。

4） 当节点找到工作证明时，它将块广播给所有节点。

5） 节点仅在块中的所有事务都有效且尚未花费时才接受该块。

6） 节点通过创建链中的下一个块来表示对块的接受，将接受块的哈希用作上一个哈希。

节点总是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下一个块的不同版本，则一些节点可能首先接收一个或另一个。在这种情况下，它们处理接收到的第一个分支，但保存另一个分支以防它变长。当发现下一个工作证明并且一个分支变长时，领带就会断开；在另一个分支上工作的节点将切换到较长的分支。

新的事务广播不一定需要到达所有节点。只要它们到达多个节点，不久就会进入一个块。块广播也可以容忍丢弃的消息。如果一个节点没有收到一个块，它将在收到下一个块时请求它，并意识到它错过了一个块。

# 6.       激励

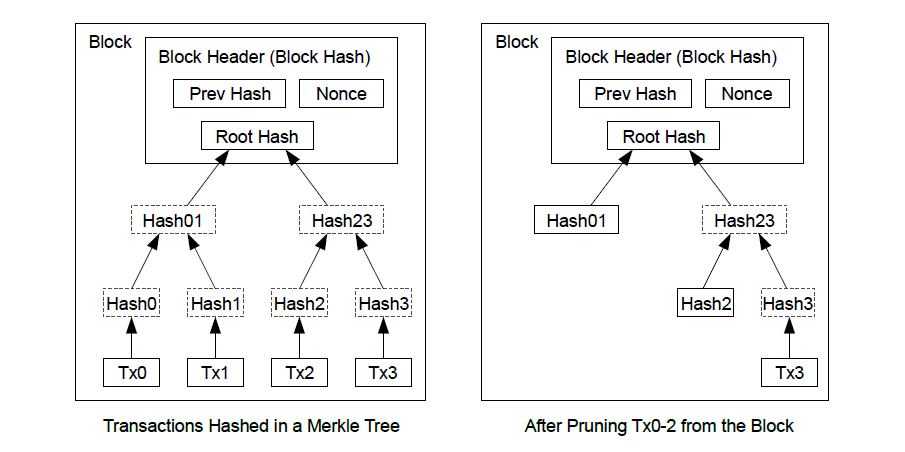
按照惯例，区块中的第一个交易是一个特殊的交易，它启动区块创建者拥有的新硬币。这增加了节点支持网络的动力，并提供了一种最初将硬币分发到流通中的方法，因为没有中央机构来发行硬币。稳定地增加一定数量的新硬币，类似于金矿开采者消耗资源为流通增加黄金。在我们的例子中，消耗的是CPU时间和电力。

奖励还可以由交易费用提供资金。如果一项交易的输出值小于其输入值，则差额是一笔交易费用，该费用加在包含该交易的区块的激励值上。一旦预定数量的硬币进入流通，奖励就可以完全转变为交易费用，完全不受通货膨胀的影响。

这种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装配比所有诚实的节点更多的CPU能力，那么他将不得不在使用它通过偷回他的付款来欺骗人们，或者使用它来生成新的硬币之间做出选择。他应该发现，遵守规则比其他人加起来更有利，而不是破坏制度和他自己财富的有效性。

# 7.       回收磁盘空间

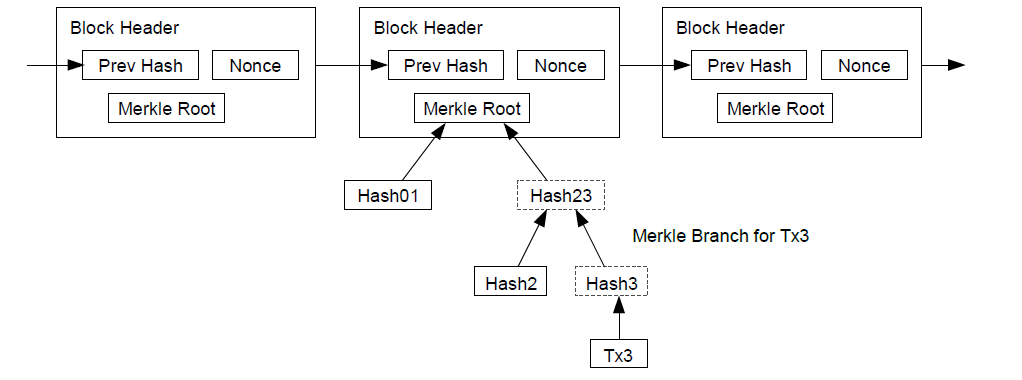
一旦硬币中的最新事务被掩埋在足够的块下，在它被丢弃之前花费的事务就可以被丢弃以节省磁盘空间。为了在不破坏块散列的情况下实现这一点，事务在Merkle树[7][2][5]中进行散列，只有根包含在块的散列中。旧的石块可以通过砍掉树枝来压实。内部散列不需要存储。



没有事务的块头大约是80字节。如果我们假设每10分钟生成一个块，那么80字节\*6\*24\*365=4.2MB/年。从2008年起，计算机系统的内存通常为2GB，而摩尔定律预测当前的年增长率为1.2GB，因此即使块头必须保存在内存中，存储也不应该成为问题。

# 8.       简化支付验证

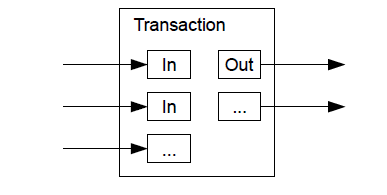
可以在不运行完整网络节点的情况下验证支付。用户只需保留最长工作链证明的块头的副本，在确信自己拥有最长的链之前，他可以通过查询网络节点获得该副本，并获得将事务链接到其时间戳所在的块的Merkle分支。他不能自己检查事务，但是通过将它链接到链中的某个位置，他可以看到网络节点已经接受了它，并且在它之后添加的块进一步确认网络已经接受了它。



因此，只要诚实的节点控制网络，验证是可靠的，但是如果网络被攻击者制服，验证就更容易受到攻击。虽然网络节点可以自己验证事务，但只要攻击者能够继续控制网络，简化方法就可以被攻击者捏造的事务所欺骗。防范这种情况的一种策略是，当网络节点检测到无效的块时，接受它们发出的警报，提示用户的软件下载完整的块，并提醒事务确认不一致性。接收频繁支付的企业可能仍希望运行自己的节点，以实现更独立的安全性和更快的验证。

# 9.       价值的组合与分割

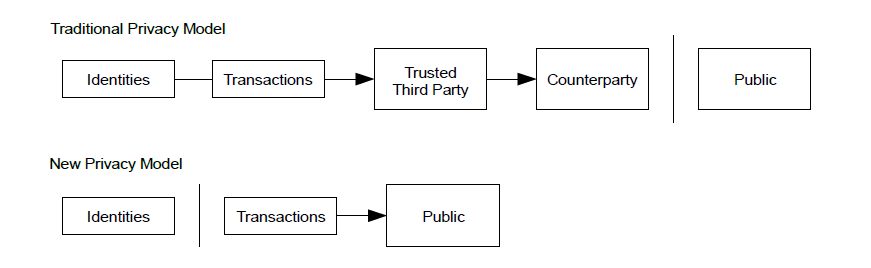
虽然可以单独处理硬币，但在一次转账中，每一分钱都要进行单独的交易是很不方便的。为了允许值被拆分和组合，事务包含多个输入和输出。通常会有一个较大的前一笔交易的单一输入或合并较小金额的多个输入，最多两个输出：一个用于支付，一个将更改（如果有的话）返回给发送者。



应该注意，扇出（fan out）在这里不是问题，其中一个事务依赖于多个事务，而那些事务依赖于更多事务。不需要提取事务历史的完整独立副本。

# 10.    隐私

传统的银行模式通过限制相关方和受信任的第三方对信息的访问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，但是仍然可以通过在另一个地方中断信息流来维护隐私：保持公钥匿名。公众可以看到有人正在向其他人发送金额，但没有将交易与任何人联系起来的信息。这类似于证券交易所发布的信息水平，在证券交易所，个人交易的时间和规模，即“磁带”，是公开的，但不告诉当事人是谁。



作为一个额外的防火墙，应该为每个事务使用一个新的密钥对，以防止它们链接到公共所有者。对于多输入事务，某些链接仍然是不可避免的，这必然表明它们的输入属于同一所有者。风险在于，如果密钥的所有者被泄露，链接可能会泄露属于同一所有者的其他交易。

# 11.    计算

我们考虑攻击者试图以比诚实链更快的速度生成备用链的情况。即使实现了这一点，它也不会使系统面临任意更改，例如凭空创造价值或拿走从不属于攻击者的钱。节点不会接受无效的事务作为支付，诚实的节点也不会接受包含它们的块。攻击者只能尝试更改自己的一项交易以收回最近花费的钱。

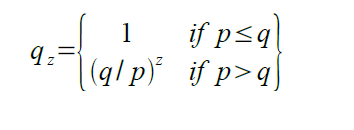
诚实链和攻击者链之间的竞争可以描述为二项随机游走。成功事件是诚实链被扩展一个区块，增加其领先+1，失败事件是攻击者的链被扩展一个区块，减少差距-1。

攻击者从给定赤字中追上来的概率类似于赌徒的破产问题。假设一个拥有无限信用的赌徒从一个赤字开始，并可能进行无限次的尝试以达到盈亏平衡。我们可以计算他达到盈亏平衡的概率，或者攻击者赶上诚实链的概率，如下[8]：

*p*=诚实节点找到下一个块的概率

q=攻击者找到下一个块的概率

*qz*=攻击者从后面z个街区追上来的概率

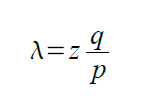


假设p>q，随着攻击者必须赶上的块数的增加，概率呈指数下降。如果他没有很早就幸运地向前冲，那么他落后的几率就会越来越小。

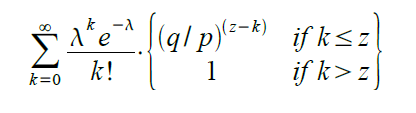
我们现在考虑一个新事务的接收者需要等待多长时间才能充分确定发送者不能更改该事务。我们假设发送者是一个攻击者，他想让接收者相信他支付了他一段时间，然后在一段时间后将其切换为自己支付。当这种情况发生时，会提醒接收者，但发送者希望一切都太迟了。

接收方生成一个新的密钥对，并在签名前不久将公钥提供给发送方。这可以防止发送者提前准备一个区块链，方法是不断地处理它，直到他足够幸运地走在前面足够远，然后在那一刻执行事务。一旦事务被发送，不诚实的发送者就开始秘密地在包含其事务的备用版本的并行链上工作。

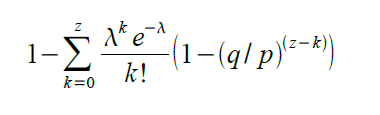
接收者等待直到事务被添加到一个块中并且z块在它之后被链接。他不知道攻击者取得的确切进度，但假设诚实的块占用了每个块的平均预期时间，攻击者的潜在进度将是一个具有预期值的泊松分布：



为了得到攻击者现在仍然可以追上的概率，我们将他可以取得的每一个进步量的泊松密度乘以他从该点可以追上的概率：



重新排列以避免求和分布的无穷尾……



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z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=0.000046 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。我们从通常的数字签名硬币框架开始，它提供了强有力的所有权控制，但如果没有防止双重消费的方法，它是不完整的。为了解决这个问题，我们提出了一个点对点网络，它使用工作证明来记录事务的公共历史，如果诚实的节点控制了大部分的CPU功率，那么攻击者很快就无法更改这些事务。网络的非结构化简单性很强。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的地方，只需要在尽力而为的基础上进行传递。节点可以随意离开和重新加入网络，接受工作链的证明作为他们离开时发生的事情的证明。他们用自己的CPU能力投票，通过扩展有效的块来表示他们的接受，通过拒绝处理无效的块来表示他们的拒绝。任何需要的规则和激励都可以通过这种共识机制来执行。

# 工具书类

[1] W。戴，“b-钱”http://www.weidai.com/bmoney.txt, 1998.

[2] H。马西亚斯、X.S.阿维拉和J.-J。Quisquater，“具有最小信任要求的安全时间戳服务的设计”，在1999年5月于比荷卢的第20届信息理论研讨会上发表。

[3] S。Haber，W.S.Stornetta，“如何对数字文档进行时间戳”，载于《密码学杂志》，第3卷，第2期，第99-111页，1991年。

[4] D。拜耳公司。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329-3341993页。

[5] S。Haber，W.S.Stornetta，“位串的安全名称”，第4届ACM计算机与通信安全会议论文集，第28-35页，1997年4月。

[6] 答。返回“Hashcash-拒绝服务反措施”http://www.hashcash.org/papers/hashcash.pdf, 2002.

[7] R.C.Merkle，“公开密钥密码系统协议”，过程。1980年安全与隐私研讨会，IEEE计算机学会，第122-133页，1980年4月。

[8] W。费勒，“概率论及其应用导论”，1957年。