**概述.**

真正的点对点电子现金应允许发端人向另一方直接在线支付，而无需通过第三方金融服务机构。虽然现有的数字签名技术提供了部分解决方案，但如果需要可信的第三方机构来阻止电子现金的“双重支付”，将失去电子现金给人类带来的最大利益。利用对等网络技术解决了电子现金中的“双倍支付”问题。网络对交易记录加时间戳，通过哈希加密，然后将其合并成由不断增长的哈希记录组成的链文件，形成新的交易记录，该哈希记录链文件（以下简称链文件）是系统网络提供的需要证明工作负载的存储和计算服务。没有基于工作证明的系统网络来重新完成所有工作证明，并且不能修改已经形成的记录。基于工作负载证明的系统概念，最长链文件不仅提供这些工作负载序列事件记录的证据，而且还由最大的CPU处理能力池生成。只要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，它们的处理能力就会超过攻击者，生成最大的链文件。

这个网络本身需要最简单的结构，以便尽最大努力将信息包广播到网络上的计算机节点。同时，计算机节点在离开时只需要接受工作负载网络生成的数据链路，也可以随时加入和离开互联网。

**1.介绍**

互联网商务的电子支付已经发展到几乎所有的电子支付都需要专有金融机构提供第三方委托进行处理的阶段。尽管该系统对于大多数交易都能很好地工作，但它仍然需要面对基于信任的基本模型的固有缺点。由于金融机构不可避免地开始调解纠纷，完全不可撤销的交易就不可能真正实现。调解成本增加了交易成本，限制了实际交易的最小规模，同时完全切断了为日常小额交易提供服务的可能性。从广义上讲，系统失去了为不可撤销服务提供不可撤销支付的能力。能力。因为用户有取款的可能，在一定时间内需要持续的信任，这就导致商家为了获取更多不再需要的信息而对客户进行防范和骚扰。不可避免地，一定比例的欺诈交易是可以接受的。虽然使用实物货币可以避免这些成本和支付的不确定性，但没有一个商户在没有经过可信的三方通信渠道的情况下进行支付。

这就是为什么需要一个基于加密认证的电子支付系统来取代原来基于信任的基本模式，允许希望交易的任何双方直接支付，而不使用基于信任的第三方。计算出的无效交易将自动撤销，以保护卖方免受欺诈。有条件的传统合同将是机械化的，保护买方将非常简单。本文提出了一种点对点分布式时间戳服务器来生成基于交易订单时间序列的计算证明方案，从而解决了重复支付问题。只要所有诚实节点控制的CPU的计算能力之和大于联合攻击节点的计算能力之和，系统就是安全的。

**2.交易**

我们首先将电子货币定义为包含一串数字签名的链。每个硬币交易者使用哈希技术对前一个交易信息和下一个所有者的公钥进行加密，然后执行数字签名，最后将此信息添加到电子货币的末尾。下一个收款人使用链中的私钥和公钥进行签名验证，以确认他是链的所有者，即电子货币。



当然，这个过程的问题是受款人仍然无法验证货币的某个所有者是否双倍使用了货币。通常的解决方案是引入一个可信的中央机构，或一个薄荷来检查每一笔交易。交易是否双倍花费。每次交易后，铸币厂必须回收货币以发行新货币。只有直接从铸币厂发行的货币才可以被认为不是双重消费。这种解决方案的灾难在于，整个货币系统依赖于某个公司来运营铸币厂，就像银行一样，每一笔交易都要经过铸币厂。

我们需要使用一种方法让收款人知道，货币的前所有人没有在任何早期交易中签署授权书，从而导致双重支出。我们的目的是计算上一笔交易，我们不需要关心下一笔交易是否会加倍花费。唯一的方法是了解所有以前的交易以确认该交易不存在。基于造币厂的模型，造币厂知道所有的交易，同时确定哪个交易请求第一时间到达。为了在没有三个受信任方的情况下实现这一目的，交易必须公开发布，我们需要系统中的每个参与者都同意他们收到的单个订单历史记录。收款人需要通过主节点证明每一笔交易，同意他们第一次收到交易。

**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。显然，为了进入散列，时间戳证明数据此时必须存在。每个时间戳在其哈希表中包含前一个交易的时间戳，并且每个后续交易的时间戳都是前一个交易的时间戳被加强而形成的链。我们的解决方案从一个时间戳服务器开始。时间戳服务器对一组已被时间戳标记的数据块进行散列和加密，然后广泛发布此散列，就像新闻或以前的论坛帖子一样。显然，为了进入散列，时间戳证明数据此时必须存在。每个时间戳在其哈希表中包含前一个交易的时间戳，并且每个后续交易的时间戳都是前一个交易的时间戳被加强而形成的链。

**4.工作量证明**

我们需要使用一个工作证明系统来构建一个基于点对点的分布式时间戳服务器，这与Adam Back提出的hash cash非常相似，而不是以前的新闻组和论坛机制。在对数据进行哈希和加密之后，工作证明使用安全哈希算法sha-256来检查数据的哈希值。散列从一定数量的0字节开始，平均检查工作量随着0字节的增长呈指数增长，而验证只需要执行散列操作。

为了使我们的时间戳网络是可行的，我们添加了一个随机数，将不会重复到数据块和执行一定量的工作找到它。此数据块的哈希已包含所需的0字节数。一旦CPU处理能力被证明能够满足所需的工作负载，就不能在不重做所有工作的情况下修改这个数据块。随后的数据块在末尾链接。修改数据块的信息需要重做所有后续数据块的工作量重做。



这种工作量制度也解决了谁代表多数的集体决策问题。如果大多数都是基于一个IP地址，一票表决的机制，它会被那些可以分配大量IP地址的人所破坏。工作证明是基于一个CPU，一票。大多数决策都用最长的链来表示，这也代表了最大努力效果的输入。如果大多数CPU都由诚实节点控制，那么诚实链的增长速度将最快，超过任何竞争链。修改过去的块时，攻击者必须重做该块和所有后续块中的所有工作，然后赶上超出诚实节点的工作。然后，我们将证明，随着数据块的增加，缓慢的攻击者追赶后续数据块的概率呈指数增长。

为了补偿硬件速度的提高和节点操作变化带来的好处，工作证明将由移动平均值确定，即每小时生成的平均数据块数。如果生成得太快，难度就会增加。

**5.网络**

运行此网络的步骤如下：

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。

节点总是认为最长的数据链路是正确的，并且总是在其上扩展。如果两个节点一起广播不同版本的数据块，则一些节点首先接收一个或另一个。在本例中，它们将开始处理第一个接收到的数据块，但将另一个存储为下一个分支以防止其变长。当工作证明网络发现其中一个分支变长时，工作在短链上的节点将切换到长链上，其从属关系也将中断。

新的交易广播不需要到达所有节点，它们只需要到达尽可能多的节点，并且它们将被集成到数据块中。数据块广播也允许丢弃信息。如果一个节点没有收到一个数据块，它将继续请求它，直到它收到下一个数据块并认为它是丢失的数据块。

**6.激励**

根据规则，数据块中的第一个交易是一个特定的交易，它创建了一个新的货币，该货币归此数据块的创建者所有。这为支持网络的节点增加了动力，同时提供了一种在整个周期内分发货币的方法，而无需中央当局的影响。一个稳定和不断增加的新货币和黄金矿商花费资源将黄金纳入黄金周期体系。在本例中，处理器时间和功耗是需要花费的资源。

奖励也可以通过交易费用获得。如果交易的输出值小于输入地址，那么差额就是交易费，作为对包含该交易的数据块的激励而添加。整个周期内的货币量已经提前设定，同时，交易费用可以作为交易的激励，完全避免通货膨胀。

激励也可以帮助节点保持诚实。如果一个贪婪的攻击者能够收集到比诚实节点更强大的处理器，他要么选择从自己的交易中欺骗他人，要么利用它来生成新的货币。他应该发现遵守规则能获得更多的利润。这样的规则有利于他和其他人赚取新的货币，比他削弱体制、损害自己财富和健康的效力更大。

**7.回收磁盘空间**

一种货币的最后一笔交易已经被足够的数据块覆盖，那么支付交易之前的数据就不能再用来节省磁盘空间了。为了方便它而不中断数据块的散列。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，这种支付验证依赖于尽可能诚实的节点来控制网络，但是如果网络被具有大量计算能力的攻击者控制，那么它将很容易受到攻击。当网络节点可以确认自己的交易时，只要攻击者的计算能力比网络强，这种简单的方法就可以被攻击者捏造的交易欺骗。当一个网络节点检测到一个无效的数据块时，一个策略将保护对手。它们将收到来自网络节点的警告，提示软件用户下载整个数据块，以确认警告交易的一致性。频繁支付的商户还可以运行自己的节点，获得更独立的安全性和更快的确认。

**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**

传统的银行模式是给合作伙伴有限的访问权限，同时通过可信的第三方来调用查看一定程度的隐私。除此方法外，维护隐私还需要中断部分信息流，通过匿名公钥，公开所有需要的交易。公众可以看到某人发送给其他人的号码，但没有关于交易者的信息。这与证券交易所的信息披露水平非常相似。公众记录了单笔交易的时间和规模，但他们不知道谁在交易。



作为附加防火墙，同一所有者的每个新交易都可以连接到一对新的配对密钥。某些连接仍然不可避免地包含多个交易的输入，这些交易必须公开同一所有者的其他过去的输入。风险在于，如果所有者的密钥被暴露，连接将暴露属于同一所有者的其他交易。

**11.计算**

我们假设攻击者试图生成更快的链而不是诚实的链。即使完全完成，也会让系统随意改变，比如凭空创造价值或者拿走永远不属于他的钱，节点不会接受无效的支付交易，诚实的节点也不会接受包含在其链中的交易。攻击者所能做的就是试图更改自己的交易，从最近的付款中取回资金。

诚实链和攻击链之间的博弈特征是二项分布的随机游走。一个成功的事件是诚实节点被一个数据块扩展，其领先优势增加一个点；一个失败的事件是攻击者的链被扩展一个数据块，差距减少一个点。

攻击者从给定赤字中追上来的概率类似于赌徒破产的问题。假设一个赌徒一开始有赤字，拥有无限的信用，同时他试图无限地下注以达到收支平衡。我们可以计算出他达到收支平衡的概率，也就是说，攻击者赶上了诚实的链条。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



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。

我们现在考虑需要等待多长时间才能确认发送方不能更改交易。我们假设发送者是攻击者，他想说服接收者他已经付了钱，然后过了一段时间，他把钱付给了自己。当它发生时，收件人将收到一个警告，但发件人希望它以后发生。

接收方在签名之前生成一个新的配对密钥，然后快速将公钥提供给发送方。这可以防止发送方在此时间点之前准备数据块链并开始连续工作，直到他有幸跑到前面，然后在此时执行交易。一旦交易被发送出去，不诚实的发送者已经开始秘密地在一个链上工作，这个链包含他的交易的并行的备用版本。

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。



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：



重新整理避免分布无限循环的尾部求和



转换到C代码...

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。



求解 P 小于 0.1%...



**12.结论**

我们提出了一种不需要基于第三方信任的电子交易系统。我们从包含数字签名的常用货币框架开始。虽然它提供了强有力的控制，但它在防止双重支付方面是不完整的。为了解决这个问题，我们提出了一种基于工作证明的点对点网络来记录公共交易历史。如果一个诚实的节点控制了主处理能力，那么经过计算，攻击者希望修改记录的努力将会改变。这是不现实的。这个网络简单而健壮。网络上的所有节点只需要一点协调。它们不需要经过身份验证，信息也不需要路由到任何特殊的地方，只需要尽可能地传播。它只需要在节点离开时接受工作负载网络产生的数据链路，计算节点可以随时加入和离开网络。它们利用处理器的能力投票，在数据块上扩展新数据以表示对数据块有效性的认可，在数据块上拒绝扩展以拒绝无效数据块。任何必要的规则和奖励都已纳入这一一致的机制并得到执行。

备注：觉得它好多点赞，把它顶上去

**13.参考链接**

[1] W. Dai, "b-money,"
[http://www.weidai.com/bmoney.txt](https://link.zhihu.com/?target=http%3A//www.weidai.com/bmoney.txt" \t "_blank), 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](https://link.zhihu.com/?target=http%3A//www.hashcash.org/papers/hashcash.pdf" \t "_blank), 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.

编辑于 2017-08-20

[比特币 (Bitcoin)](https://www.zhihu.com/topic/19600228%22%20%5Ct%20%22_blank)