Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Translated into Simple Chinese from bitcoin.org/bitcoin.pdf
by weibo.com/tiehexue

摘要. 一个完全点对点的数字货币系统无须通过金融机构，允许一方直接与另一方进行在线付款。数字签名技术是个方案，但须依赖第三方信用中心防止双重支付，因而不够好。我们提出一种利用点对点网络解决双重支付的方案。该网络把交易打上时间戳，通过哈希（hash）把这些交易编织进工作量（proof-of-work）链，创建的记录在不重新编织整个工作量链的情况下无法篡改。最长工作量链不但见证所有交易事件，也源自最强大 CPU 池。只要大部分节点的 CPU 没有被用来攻击，该网络能编织最长的链并踢走攻击者。该网络结构简单。它通过广播发布消息，节点可以自由退出或入列，接受最长的工作量链即可获知离线时的情况。

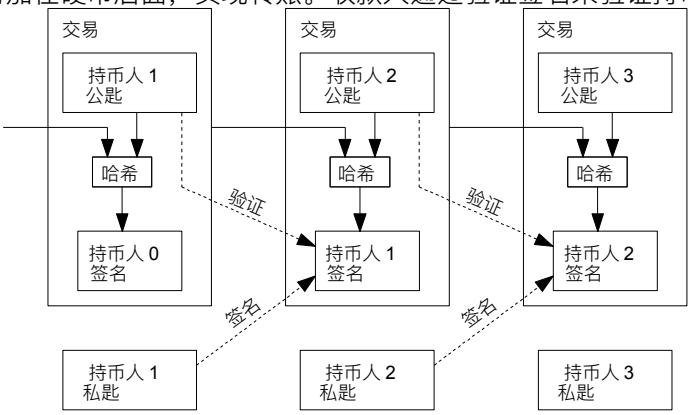
1. 介绍

电子商务如今已完全依赖金融机构作为第三方信用中心处理电子付款。大部分情况下这没什么问题，但以信用为基础的模式存在天生的弱点，令人烦恼。金融机构承担了协调争端的责任，因而很难实现完全不可撤回的交易。而争端协调增加了交易成本，限制了最小交易金额，降低了小规模常规交易的意愿，以及压制了不可撤回类支付和服务的发展。为了应对可撤回的情况，信用依赖不得不普遍存在。商家需要警惕客户，纠缠他们提供超出所需的信息。相当比例的欺诈行为，人们也习以为常。现金交易不存在类似的交易成本和支付不确定性，但暂无机制使得在线交易可以摆脱第三方信用中心。

我们需要一个立足于密码学的信用模型，允许双方无须第三方信用中心，直接进行交易。无法通过计算破解来撤回支付保证卖家远离欺诈，而常规托管机制可以保护买家。本论文提供了一种方解决双重支付的方案，通过点对点的分布式时间戳服务器按时间顺序制造交易证据。只要该系统正常节点集体拥有的 CPU 计算能力超过攻击者节点，系统就是安全的。

2. 交易

我们把一链数字签名称为数字硬币。某个持币人对上一次交易和本次交易收款方的公匙进行数字签名，再加在硬币后面，实现转账。收款人通过验证签名来验证持币人。

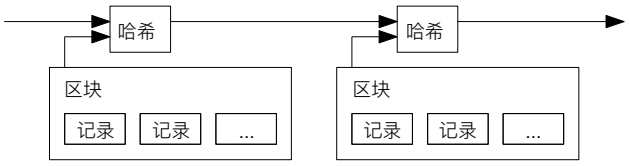


这里有个问题，收款人无法验证持币人有没有双重支付。通常方案是引入权威信用中心，或铸币厂，检查每笔交易是否重双重支付。每笔交易，硬币先被铸币厂回收，然后重新签发新的给收款人，只有铸币厂发行的硬币才会被交易各方接受。这个方案的问题是整个系统都依赖于铸币厂，每笔交易都必须经过它，它就像一个银行。

我们需要给收款人一个方法确定该硬币之前的持有人没有用它签署其他交易。这样，我们只关心此交易是不是该硬币最早的交易，而不关心随后的双重支付。确认一个硬币没被交易的唯一办法需要对所有交易知情。在铸币厂模式，铸币厂了解所有的交易，并决定哪个交易发生早。没有信用中心又需要对所有交易知情，就需要把所有交易公开【1】，我们需要一个多方参与系统，能就共同单一的交易历史记录达成一致。收款方需要有证据能确认，在交易后大部分节点都认同他是第一个收款方。

3. 时间戳服务器

方案建立在时间戳服务器上。该服务器哈希一块块记录集，并像报纸和新闻组那样发布出去【2-5】。时间戳证明交易的时间，其本身也纳入哈希。每个时间戳哈希包含之前的时间戳，形成一个哈希链，每个新加的时间戳证明前一个。

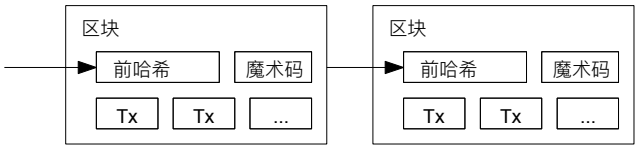


4. 工作量

和报纸或新闻组不同，以点对点方式实现分布式时间戳服务器，我们需要一个类似 Adam Back 的哈希黄金的工作量系统【6】。它指的是寻找能哈希（如 SHA-256 算法)成有指定

个数零开始的哈希值的字符串。平均工作量随着零的位数增长而指数增长，并且一条哈希命令即可检验。

本文所述的方案给区块增加不同的魔术码，直到发现一个符合要求。一旦通过 CPU 的计算满足了工作量的要求，该区块即不能改变，除非重新计算。因为后续的区块都链接在当前区块之后，改变当前区块意味着也要重新计算之后的所有区块。



工作量的机制也顺便解决了多数表决的问题。如果多数与否是基于一个 IP 一票，那会被拥有更多 IP 的人破坏。工作量机制最重要的是基于一个 CPU 一票。多数决策形成最长的链，也即投资了最多的工作量。如果多数的 CPU 都被正常的节点掌握，那正常的链会增长的非常快，并踢走任何攻击链。要修改一个区块，攻击者需要重新计算该区块和所有后续区块，并赶上和超过正常节点。下文会阐述一个较慢的攻击者赶上正常链的概率随着区块的增加而指数级降低。

为了协调节点不断增加的硬件和投入，工作量获取的难度由一个可调整的每小时生成的平均区块数决定。如果产生的太快，就增加相应的难度。

5. 网络

网络的运行步骤：

1) 新的交易被广播到所有节点
2) 每个节点把所有新的交易放到一个区块
3) 每个节点寻找该区块的工作量魔术码
4) 当一个节点发现魔术码后，广播给其他节点
5) 其他节点检查是否所有交易都被包含，且交易都合法和硬币没有被支付过。
6) 节点表决接受该区块，并把该区块作为“前哈希”，开始创建下一个区块。

节点总是把最长的链作为正确的链，并不断扩展它。如果两个节点同时广播新建区块，其他节点可能先后收到两个新建区块。这种情况下，该节点在先到达的区块上继续工作，同时为后收到的节点建立分支。下一个新区块纳入后，其中某个分支会更长，短的分支被丢弃；在较短分支上工作的节点会切换回最长分支。

新的交易并不需要广播到所有的节点。只要大部分节点收到，它就会被纳入到一个区块。区块广播能容忍信息丢失。如果一个节点漏掉某个区块，它在接收到之后的区块时会发现并重新下载该区块。

6. 激励

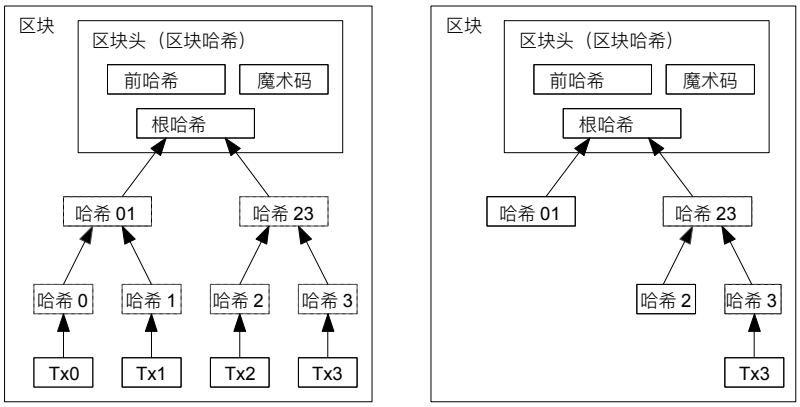
约定区块的第一个交易是给该区块的创建者创建一个新的硬币。这即是对节点运营的支持，也是硬币分配初始化并启动硬币流通的办法，因为没有发行硬币的权威中心。这种机制下硬币会被持续地定量地产生，恰似消耗各种资源去淘金并把黄金引入到流通环节。这里消耗的资源是 CPU 计算能力和电力。

激励也可以通过收取交易费来实现。区块创建者从每笔交易中直接扣除一定的费用作为他的激励。一旦流通的硬币数量达到限定值， 可完全依靠交易费激励网络继续运行，这时硬币也不会通胀贬值。

激励机制可以鼓励网络节点遵守规则。如果一个贪婪的攻击者拥有足够多的 CPU 运算能力，比正常节点还多，他可以选择把已支付的款项撤回进行欺诈，也可以去争取激励。他会发现遵守规则时赚得钱比其他所有玩家都多，而不应该去破坏这个系统，动摇他自己的财富根基。

7. 磁盘空间

一旦最新的交易被足够多区块记录后，之前的交易可以删除以节省磁盘空间。要达到这个目的且不破坏区块哈希，需把每笔交易的哈希组织为默克尔树【7】【2】【5】，根节点是区块哈希。删除哈希树的分支可以压缩更早的区块。中间生成的哈希也不需要保留。

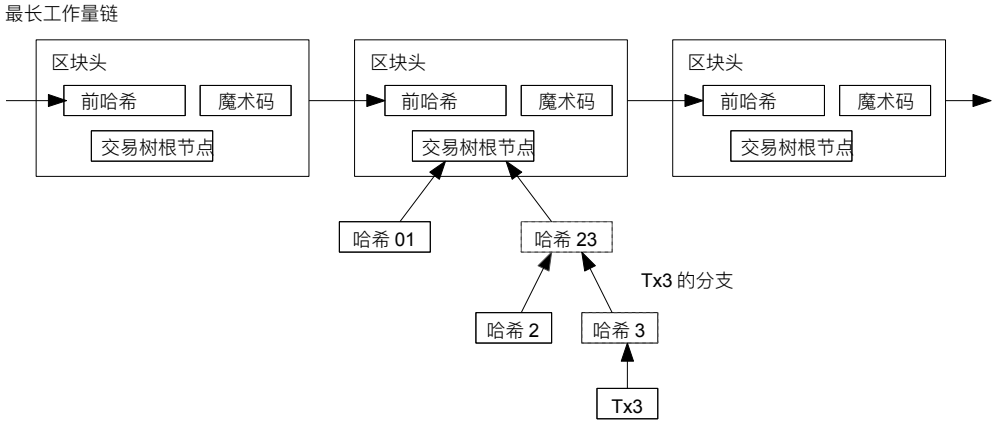


交易的哈希以默克尔树组织　　剪去哈希树里的交易 0-2

一个不包含交易的区块头有 80 字节。假设每十分钟生成一个区块，80 字节 * 6 * 24 * 365 = 4.2M 每年。2008 年普通计算机即有 2GB 内存，并且摩尔定律预测每年增加 1.2G，区块头的存储不是问题，甚至可以装载进内存。

8. 简易支付验证

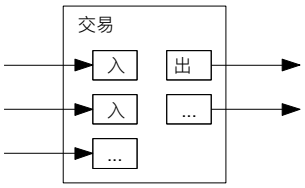
用户为完成支付并不需要运行一个完整的节点，他只需要通过网络获取并确认最长工作量链，保存链上所有区块头，并获得交易发生时段的区块默克尔树。用户这时不能自行提交交易，而是把交易链接到链条里并观察网络是否接受，如果有其他区块被加到该区块后面则进一步表明网络接受了该交易。



因此，只要网络中的正常节点占据主导地位支付机制就是有效的，但攻击者的节点如果压倒正常节点，系统会变得脆弱。网络节点可以验证交易，因此只要攻击者能持续压倒正常网络，就可以编造交易，骗过简易支付验证。对应的策略是用户要注意接收正常节点发现非法区块的警告，用户客户端下载完整的区块，检查交易是否完整。需要频繁进行支付的商业机构为了安全和效率，可以独立运营自有节点。

9. 硬币的合并和分割

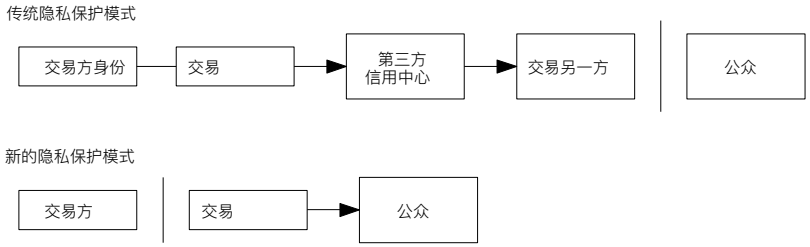
虽然每个硬币都可以单独使用，但一次转账使用多个交易是非常不方便的。因此系统的交易设计必需支持多个输入和输出，以支持款项可以被合并或分割。一般情况下，如果之前的交易比较大可以只有一个输入，也可以合并多个小额，但输出最多只有两项：一个属于收款人；如果还有找零，还有一个返还给付款人。



需要声明的是这种扇出结构并不是大问题，因为永远不需要摘取一个交易的完整的历史。扇出结构指的是一个交易依赖多个之前的交易，这些之前的交易又依赖更早更多的交易。

10. 隐私

传统的银行模式保护隐私的办法是限制对相关方和第三方信用中心的信息获取。因为必须公开发布所有交易，这个方法行不通了，但我们仍然可以在另外一个流程节点上截断信息流来保护隐私，即使用匿名公匙。公众能看到某个人转账给另外一个人，但没有任何信息把交易和具体的人关联。这很像股票交易中，“股票交易带”显示了每个交易的时间和数量但不会泄露是谁在交易。



要想更好的隐私，每次交易都启用新的密匙对。在一个交易里使用多个输入，容易被认为多个输入对应的密匙是同一个人。最关键的是如果一个具体的人和密匙被曝光，那他其他使用相同密匙的交易都会曝光。

11. 计算

考虑攻击者尝试更快地生成区块链的情景。即使他能做到这一点，也不意味着系统可以被任意修改，比如凭空创造出货币或偷窃别人的钱。节点不会接受非法交易作为支付，正常节点也不会接受包含非法交易的区块。一个攻击者能做的只有把之前他支付的钱重新拿回来。

正常链和攻击链之间的竞赛可以看作是二叉树随机游走过程。正常链扩展一个区块就

领先一步，攻击链扩展一块就缩小了一步差距。

攻击者从某个落后位置开始追赶的过程类似赌徒破产问题。假设赌徒具有无限的金钱，从输了一定的钱开始，一直赌下去直到不赢不输。我们可以计算赌徒达到不赢不输的概率，也即计算攻击者赶上正常链的概率，如下【8】：

p = 正常链发现下一区块的概率
q = 攻击链发现下一区块的概率
q_z = 攻击链落后 z 个区块赶上正常链的概率

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

如果 p>q，攻击者赶上正常链的概率随着区块增加指数下降。因此，如果攻击者不是一开始就行动，他赶上的机会渺茫，差距会越来越大。

现在考虑收款人需要等待多久才能确定付款人不可能撤回付款。我们假设付款人是个攻击者，他想让对方误以为已经付款了，一段时间后把款项再次转给他自己。收款人到时会收到警示，而付款人希望收款人发现时已经太晚。

收款人必须在付款人确实准备支付时生成一个新的密匙对，并把公匙发送给付款人。这可以防止付款人提前准备好足够长的区块链，然后执行交易。交易发出后，他立刻秘密发起另一交易，并生成另外一条区块链。

收款人等到该交易被纳入一个区块，且该区块后面还被追加了 z 个区块。这是他仍然不知道攻击者的进度，但可以假定正常网络以平均速度生成新的区块，且攻击者的进度符合泊松分布，其期望值是：

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

把攻击者可能取得不同进展的泊松密度乘以他能赶上正常链的概率，得到攻击者实现欺诈的概率：

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

把公式转换一下，避免无限求和...

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

转换为 C 语言代码...

```
#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;
}
```

计算一下，可以看到随着 z 的增加，概率指数下降。

```
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
```

攻击者获胜概率小于 0.1%时，q 和 z 的值...

```
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

我们规划了一个不以信任为基础电子交易系统。我们从数字签名作为硬币的框架开始，框架保证了硬币的所有权，但阻止不了双重支付。为了解决这个问题，我们采用了点对点的

网络，通过工作量公开记录所有的交易。这样，只要正常的网络节点拥有的 CPU 运算能力具占据主导地位，对系统的攻击很快就会超出攻击者的计算能力。系统的简单结构使得它非常健壮。节点之间的通讯简单，无须过多协调。节点也不需要具有可识别性，因为节点之间的消息发送不依赖于具体某个节点，只需要正常广播出去就行。节点可以自由退出或入列，接受最长的工作量链即可获知离线时的情况。网络依据节点的运算能力投票，接受合法的区块并扩展，拒绝不合法的区块。其他需要的规则和激励都可以通过相同投票机制决定。

参考

[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, "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, 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.