比特币：P2P 电子货币系统

摘要：一个纯粹的 P2P 电子货币版本将让线上支付不通过金融机构，实现直接传递。数字签名解决了一部分问题，但如果一个值得信任的第三方仍然被要求双重支付，那么它将失去优势。网络通过将交易散列到一个持续的基于哈希函数的工作量证明链中来给交易添加时间戳，形成一个在不重新执行工作量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，还可以证明它来自最大的 CPU 功率池。只要 CPU 的大部分功率是由不合作攻击网络的节点控制的，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。消息以最佳方式广播，节点可以随意离开或重新加入网络，接受最长的工作量证明链作为它们离开时发生的事情的证据。

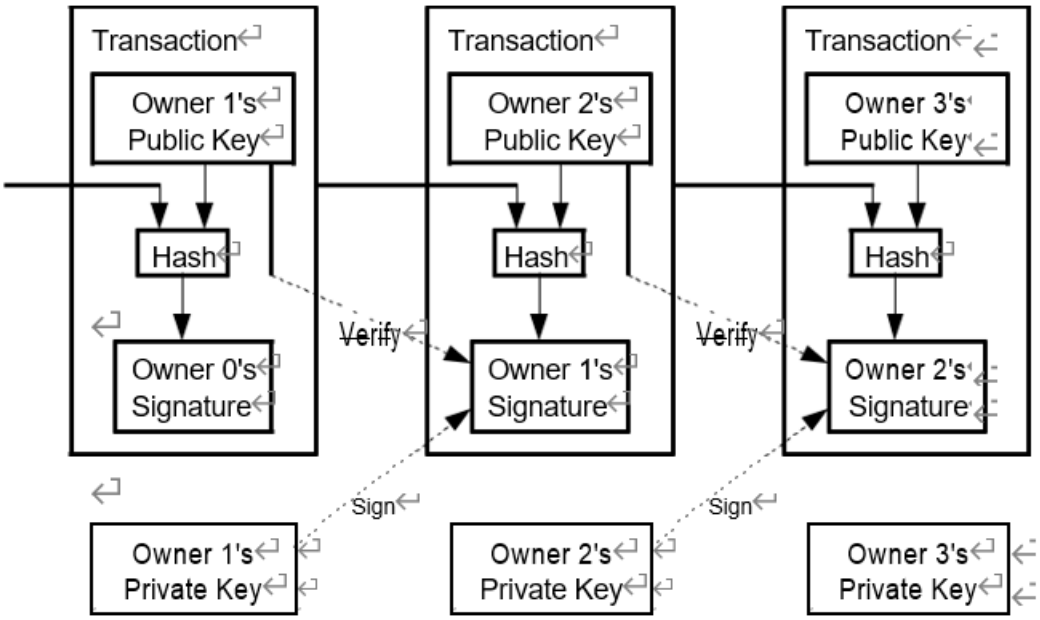
1. 介绍

互联网上的商业几乎完全依赖金融机构作为可信赖的第三方来处理电子支付。虽然该系统对于大多数事务都能很好地工作，但它仍然受到基于信任模型的固有弱点的影响。完全不可逆转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。中介的成本增加了交易成本，限制了最小的实际交易规模，切断了小型非正式交易的可能性，对不可逆服务失去进行不可逆支付的能力，这是更广泛的成本。随着逆转的可能性，对信任的需求扩大了。商家必须对他们的顾客保持警惕，为了获取他们原本不需要的更多信息而与他们纠缠不休。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任方的通信通道上进行支付。

所需要的是基于密码证明而不是信任的电子支付系统，该系统允许任意两个愿意的一方直接彼此进行交易，而无需受信任的第三方。 在计算上不可行进行逆转的交易将保护卖方免受欺诈，并且可以容易地实施常规托管机制来保护买方。 在本文中，我们提出了一种使用对等分布式时间戳服务器生成事务的时间顺序的计算证明的双花问题的解决方案。只要诚实节点集体控制比任何协作的攻击者节点组都更多的 CPU 能力，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。每个所有者通过对上次交易的哈希值和下一个所有者的公钥进行数字签名并将它们添加到硬币的末尾，将硬币转移到下一个硬币。收款人可以验证签名以验证所有权链。

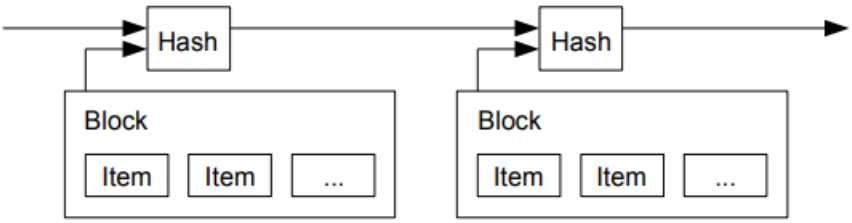


当然，问题在于收款人无法验证其中一位所有者是否将比特币重复使用。常见的解决方案是引入受信任的中央机构（mint），该机构检查每笔交易是否存在双重支出。每次交易后，必须将硬币退还给铸币厂发行新的硬币，并且仅信任直接从铸币厂发行的硬币不会被重复使用。这种解决方案的问题在于，整个货币体系的命运取决于经营铸币厂的公司，每笔交易都必须像银行一样通过铸币厂。

我们需要一种让收款人知道以前的所有者未签署任何较早交易的方法。就我们的目的而言，最早的交易才是最重要的交易，因此我们不关心以后进行双倍支出的尝试。确认没有事务的唯一方法是知道所有事务。在基于造币厂的模型中，造币厂知道所有交易，并确定哪个先到。为了在没有受信任方的情况下完成此任务，必须公开宣布交易[1]，并且我们需要参与者可以就单一订单接收历史达成一致的系统。收款人需要证明，在每次交易时，大多数节点都同意这是第一个收到的。

3. 时间戳服务器

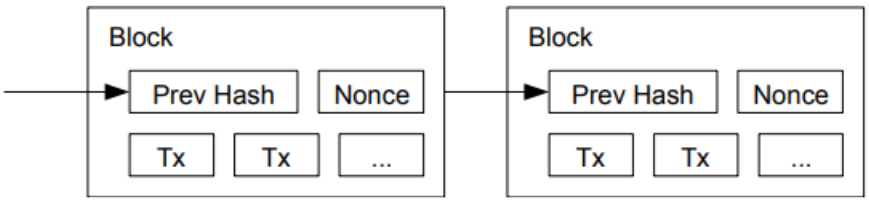
我们建议的解决方案从时间戳服务器开始。 时间戳服务器通过对要打时间戳的项目块进行散列并广泛发布散列来工作，例如在报纸或 Usenet 帖子中[2-5]。 时间戳证明，为了进入哈希表，数据必须在当时已经存在。每个时间戳在其哈希中都包含前一个时间戳，形成一个链，每个附加时间戳都会增强之前的时间戳。



4. 工作量证明

为了在对等基础上实现分布式时间戳服务器，我们将需要使用类似于 Adam Back 的 Hashcash [6]的工作量证明系统，而不是报纸或 Usenet 帖子。 工作量证明包括扫描一个值，该值在进行散列处理时（例如，使用 SHA-256），散列以零位开始。 所需的平均功在所需的零位数上是指数级的，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随机数来实现工作量证明，直到找到一个值，该值为块的哈希提供所需的零位。 一旦花费了 CPU 工作量来满足工作量证明，就不能在不重做工作的情况下更改该块。 当后面的块链接到其后时，更改该块的工作将包括重做其后的所有块。



工作量证明还解决了在多数决策中确定代表制的问题。如果大多数基于一个 IP 地址一票，那么任何能够分配许多 IP 的人都可以颠覆它。工作量证明实质上是一个 CPU 一票。多数决策以最长的链为代表，链上投入的工作量最大。如果大多数 CPU 功能由诚实节点控制，那么诚实链将以最快的速度增长，并超越任何竞争链。要修改过去的块，攻击者必须重做该块及其后所有块的工作量证明，然后赶上并超越诚实节点的工作。稍后我们将说明，随着后续块的增加，较慢的攻击者追赶的概率将呈指数下降。

为了补偿硬件速度的提高和随着时间的推移对运行节点的兴趣的不断变化，工作证明的难度由以每小时平均块数为目标的移动平均值来确定。如果生成速度太快，难度会增加。

5. 网络

运行网络的步骤如下：

1）新交易将广播到所有节点。

2）每个节点将新交易收集到一个块中。

3）每个节点都在为其块寻找困难的工作量证明。

4）当节点找到工作量证明时，它将块传播到所有节点。

5）仅当块中的所有事务有效且尚未花费时，节点才会接受该块。

6）节点通过使用接受的块的散列作为前一个散列来创建链中的下一个块，从而表达对块的接受。

节点始终将最长的链视为正确的链，并将继续努力进行扩展。 如果两个节点同时广播下一个块的不同版本，则某些节点可能首先接收一个或另一个。 在这种情况下，他们会在收到的第一个分支上工作，但要保留另一个分支，以防分支变得更长。 找到下一个工作量证明并且一个分支变长时，关系将断开。 然后，在另一个分支上工作的节点将切换到更长的节点。

新的事务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块。 块广播还可以容忍丢失的消息。 如果一个节点没有收到一个块，它将在收到下一个块并意识到丢失了一个块时提出请求。

6. 激励

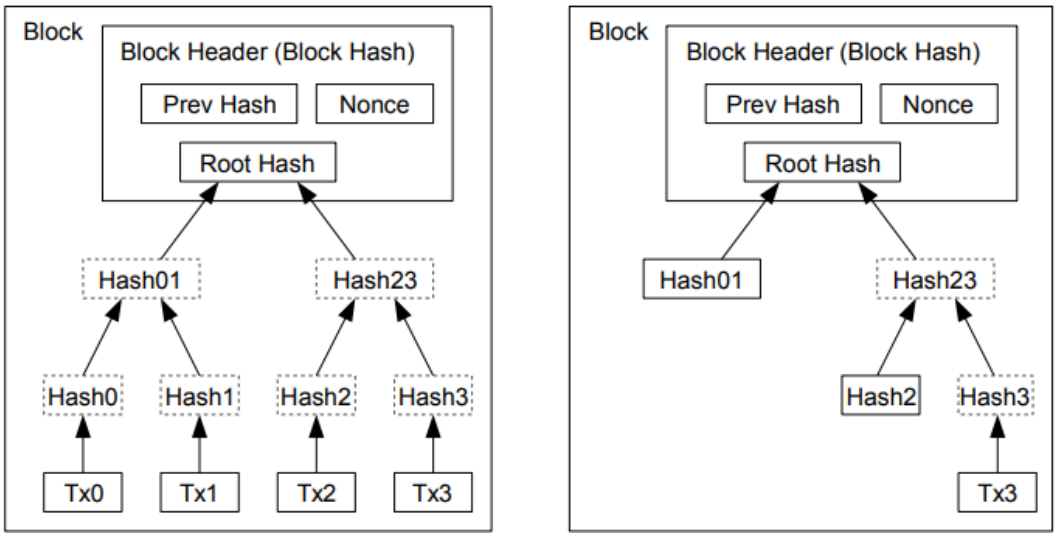
按照惯例，区块中的第一笔交易是一种特殊交易，它启动了区块创建者拥有的新硬币。这增加了节点支持网络的动力，并提供了一种最初将硬币分发到流通中的方式，因为没有中央授权来发行它们。稳定增加一定数量的新硬币类似于黄金开采商消耗资源以增加黄金的流通量。在我们的例子中，所消耗的是 CPU 时间和电力。

激励措施也可以由交易费用提供资金。如果交易的输出值小于其输入值，则差额是一笔交易费用，该费用被加到包含该交易的区块的激励值中。一旦预定数量的硬币进入流通，激励措施就可以完全过渡为交易费用，并且完全免于通货膨胀。

该激励可以帮助鼓励节点保持诚实。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 能力，他将不得不选择使用它来窃取他的付款来欺骗人们，或者使用它来生成新的代币。他应该发现，遵守这些规则比从其他任何人那里获得的钱要多得多，这些规则比其他任何人加起来都更有利于他，而不是破坏制度和自己财富的有效性。

7. 回收磁盘空间

一旦硬币中的最新交易被埋在足够的块下，已用完的交易就可以丢弃，以节省磁盘空间。 为了在不破坏块的哈希的情况下简化此过程，在 Merkle Tree [7] [2] [5]中对事务进行哈希处理，仅将根包含在块的哈希中。 然后，可以通过砍掉树的分支来压缩旧块。 内部哈希不需要存储。

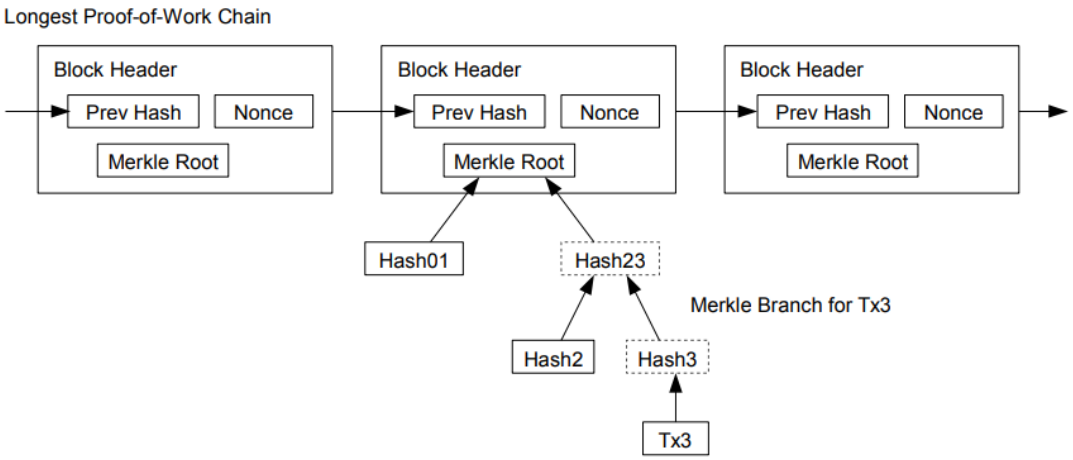


Transactions Hashed in a Merkle Tree　　After Pruning Tx0-2 from the Block

没有事务的块头大约为 80 个字节。 如果我们假设每 10 分钟生成一次块，则每年 80 字节* 6 * 24 * 365 = 4.2MB。 截止到 2008 年，通常销售 2GB RAM 的计算机系统以及摩尔定律预测当前每年将增长 1.2GB，即使必须将块头保存在内存中，存储也不成问题。

8. 简化的付款验证

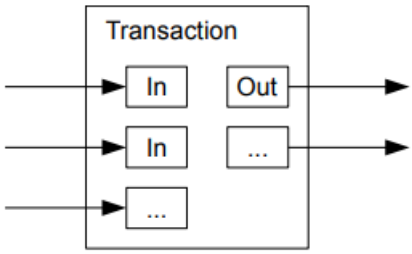
无需运行完整的网络节点就可以验证付款。 用户只需要保留最长的工作量证明链的区块头的副本，就可以通过查询网络节点来获取该副本，直到确信自己拥有最长的链，然后获得将交易与区块链接起来的 Merkle 分支。 他无法亲自检查交易，但是通过将交易链接到链中的某个位置，他可以看到网络节点已接受交易，并在进一步确认网络已接受交易之后添加了块。



这样，只要诚实节点控制网络，验证就可靠了，但是如果网络受到攻击者的压制，验证就更容易受到攻击。 尽管网络节点可以自己验证事务，但是只要攻击者可以继续使网络胜过攻击，简化的方法就可能被攻击者的虚假交易所欺骗。 防止这种情况发生的一种策略是在网络节点检测到无效块时接受来自网络节点的警报，提示用户软件下载完整的块并警告交易以确认不一致。 收到频繁付款的企业可能仍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. 合并和分割价值

尽管可以单独处理硬币，但要对转账中的每一分钱进行单独交易是不方便的。 为了允许价值被分割和合并，交易包含多个输入和输出。 通常情况下，要么是一笔来自较大笔交易的单笔输入，要么是一笔金额较小的多笔输入，至多有两笔输出：一笔用于支付，另一笔将零钱（如果有的话）退还给发件人。

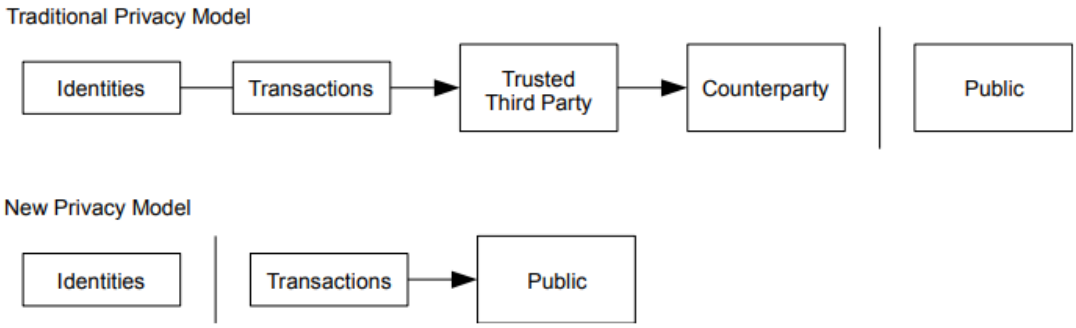


应该注意的是，扇出（fan-out）不是一个问题，在扇出中，一个事务取决于多个事务，而这些事务又取决于多个事务。 永远不需要提取交易历史记录的完整独立副本。

10. 隐私权

传统的银行业务模式通过限制相关方和受信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此方法，但仍可以通过在另一位置中断信息流来保持隐私：通过使公用密钥保持匿名。 公众可以看到有人正在向其他人汇款，但是没有信息将交易链接到任何人。 这类似于证券交易所发布的信息水平，在该水平上，公开了单个交易的时间和规模，即“磁带”，但没有告知双方是谁。



作为额外的防火墙，每笔交易都应使用新的密钥对，以防止将其链接到公共所有者。多输入交易仍然无法避免某些链接，这必然表明它们的输入是同一所有者拥有的。风险是，如果密钥的所有者被泄露，则链接可能会揭示属于同一所有者的其他交易。

11. 计算

我们考虑攻击者试图比诚实链更快地生成备用链的情况。即使做到这一点，也不会使系统面临任意更改的威胁，例如凭空创造价值或获取从未属于攻击者的资金。节点将不会接受无效交易作为付款，诚实节点将永远不会接受包含它们的区块。攻击者只能尝试更改其自己的交易之一，以取回最近花费的资金。

诚实链和攻击者链之间的种族可以描述为二项式随机游走。成功事件是诚实链延长了一个区块，将其领先优势增加了+1，失败事件是攻击者链扩展了一个区块，从而将差距减小了-1。

攻击者追赶给定赤字的可能性类似于“赌徒的破产”问题。假设拥有无限信用的赌徒从赤字开始，并可能进行无数次尝试以达到收支平衡。我们可以计算出他达到收支平衡或攻击者赶上诚实链的可能性，如下所示[8]：

p =诚实节点找到下一个块的概率

q =攻击者找到下一个区块的可能性

q_z=攻击者可能追上后面 z 个区块的概率

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

在我们假设 p> q 的情况下，随着攻击者必须赶上的块数增加，概率呈指数下降。面对他的机会很大，如果他不及早地向前走，那么随着他的落后，他的机会就会越来越小。

现在，我们考虑新交易的接收者需要等待多长时间，才能确定发送者无法更改交易。我们假设发件人是攻击者，他想让收件人相信他花了一段时间，然后在一段时间后将其切换为还钱。发生这种情况时，将通知接收方，但发送方希望为时已晚。

接收者生成一个新的密钥对，并在签名前不久将公共密钥提供给发送者。这可以防止发件人通过不断地进行处理，直到他有幸能够取得足够的领先优势，然后在此时执行交易，从而提前准备区块链。一旦发送了交易，不诚实的发送者就开始在包含其交易的替代版本的并行链上秘密工作。

接收者等待，直到将事务添加到一个块中，并在其后链接了 z 个块。他不知道攻击者已取得的确切进展量，但是假设诚实的区块花费了每个区块的平均预期时间，则攻击者的潜在进度将是具有期望值的 Poisson 分布：

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

为了获得攻击者现在仍然可以追上的可能性，我们将 Poisson 密度乘以

从那时起他可以追上的可能性就可以取得的每一个进步：

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

重新安排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 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=0.0000046
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

我们已经提出了一种不依赖信任的电子交易系统。我们从数字签名制成的硬币的通常框架开始，该框架提供了对所有权的强大控制，但是如果没有防止双重支出的方法，它是不完整的。为了解决这个问题，我们提出了一种使用工作量证明的对等网络来记录交易的公共历史记录，如果诚实的节点控制了大部分 CPU 能力，则对于攻击者而言，更改记录很快就在计算上不切实际。该网络的非结构化简单性十分强大。节点几乎不需要协调就可以全部工作。由于消息不会路由到任何特定的地方，只需要尽最大的努力进行传递，因此不需要标识它们。节点可以随意离开并重新加入网络，接受工作量证明链作为它们离开时发生的事情的证明。他们以 CPU 能力投票，通过扩展有效块来表示接受有效块，并通过拒绝对其进行操作来拒绝无效块。任何需要的规则和激励措施都可以通过这种共识机制来实施。

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, "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.