Bitcoin: Um Sistema de Dinheiro Eletrônico Peer-to-Peer

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

Traduzido para Português de bitcoin.org/bitcoin.pdf
por Rodrigo Silva Pinto - http://linkedin.com/in/rodrigosilvap

Resumo. Uma versão puramente peer-to-peer de dinheiro eletrônico permitiria que pagamentos on-line fossem enviados diretamente de uma parte para outra, sem passar por uma instituição financeira. As assinaturas digitais fornecem parte da solução, mas os principais benefícios são perdidos se um terceiro confiável ainda é necessário para evitar o gasto duplo. Nós propomos uma solução para o problema de gasto duplo usando uma rede peer-to-peer. A rede insere data e hora nas transações através de um hash em uma cadeia contínua de prova-de-trabalho à base de hash, formando um registro que não pode ser alterado sem refazer a prova-de-trabalho. A cadeia mais longa não só serve como prova da seqüência de eventos testemunhados, mas prova de que ela veio do maior pool de CPUs. Enquanto a maioria do poder das CPUs é controlado por nós que não estão cooperando para atacar a rede, eles irão gerar a cadeia mais longa e superar os atacantes. A própria rede requer estrutura mínima. As mensagens são espalhadas em regime de melhor esforço, e nós podem sair e regressar a rede à vontade, aceitando a cadeia mais longa de prova-de-trabalho, como prova do que aconteceu enquanto eles estavam fora.

1. Introdução

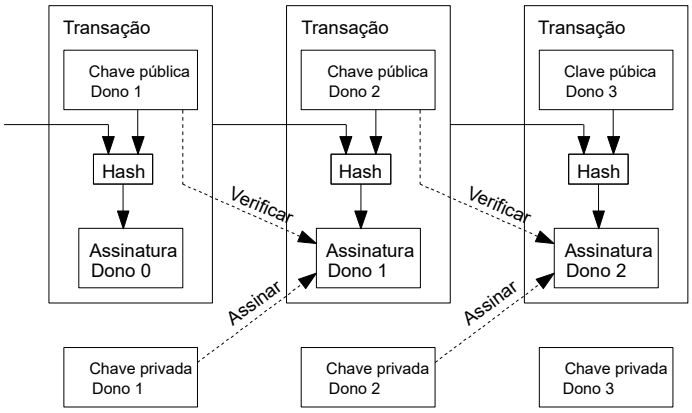
O comércio na Internet tem dependido quase exclusivamente de instituições financeiras que servem como terceiros confiáveis para processar pagamentos eletrônicos. Enquanto o sistema funciona bem para a maioria das operações, ainda sofre com as deficiências inerentes ao modelo baseado em confiança. Transações completamente não-reversíveis não são possíveis, uma vez que as instituições financeiras não podem evitar a mediação de conflitos. O custo da mediação aumenta os custos de transação, o que limita o tamanho mínimo prático da transação e elimina a possibilidade de pequenas transações ocasionais, e há um custo mais amplo na perda da capacidade de fazer pagamentos não reversível para serviços não reversíveis. Com a possibilidade de reversão, a necessidade de confiança se espalha. Comerciantes devem ser cautelosos com os seus clientes, incomodando-os para obter mais informações do que seria de outra forma necessária. Uma certa percentagem de fraude é aceita como inevitável. Estes custos e incertezas de pagamento podem ser evitados ao vivo usando moeda física, mas não existe nenhum mecanismo para fazer pagamentos ao longo de um canal de comunicação sem uma parte confiável.

O que é necessário é um sistema de pagamento eletrônico baseado em prova criptográfica em vez de confiança, permitindo a quaisquer duas partes dispostas a transacionar diretamente uma com a outra sem a necessidade de um terceiro confiável. Transações que são computacionalmente impraticáveis de reverter protegeriam os vendedores de fraudes e mecanismos rotineiros de disputa poderiam ser facilmente implementados para proteger os compradores. Neste artigo, nós propomos uma solução para o problema de gasto duplo usando um servidor de horas distribuído peer-to-peer para gerar prova computacional da ordem cronológica das operações. O sistema é seguro desde que nós honestos controlem coletivamente mais poder de CPU do que qualquer grupo cooperado de nós atacantes.

2. Transações

Nós definimos uma moeda eletrônica como uma cadeia de assinaturas digitais. Cada proprietário transfere a moeda para o seguinte por uma assinatura digital de hash da operação anterior e a

chave pública do dono da próxima e adicionando-os ao fim da moeda. Um sacador pode verificar as assinaturas para verificar a cadeia de propriedade.

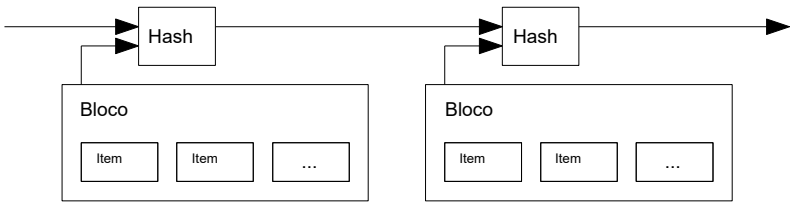


O problema, claro, é o sacador não poder confirmar se um dos pagadores não fez gasto duplo da moeda. Uma solução comum é a introdução de uma autoridade central confiável, ou casa da moeda, que verifique o gasto duplo para todas as transações. Depois de cada transação, a moeda deve ser devolvida à casa da moeda para a emissão de uma nova, e apenas moedas emitidas diretamente da casa da moeda são confiáveis de não ser gastas duplamente. O problema desta solução é que o destino de todo o sistema monetário depende da empresa que gerencia a casa da moeda, com todas as transações tendo de passar por ela, assim como um banco.

Nós precisamos de uma maneira que o sacador possa saber se os proprietários anteriores não assinaram quaisquer transações anteriores. Para nossos propósitos, a transação mais antiga é a que conta, por isso, nós não nos preocupamos com as tentativas posteriores de gasto duplo. A única maneira de confirmar a ausência de uma transação é estar ciente de todas as transações. No modelo baseado em casa da moeda, a mesma está ciente de todas as transações e decide qual chegou primeiro. Para alcançar este objetivo sem uma parte confiável, as transações devem ser anunciadas publicamente [1], e precisamos de um sistema para que os participantes concordem em um histórico único a ordem em que foram recebidas. O sacador precisa da prova que, no momento de cada transação, a maioria dos nós concorda que ela está sendo recebida pela primeira vez.

3. Servidor Timestamp

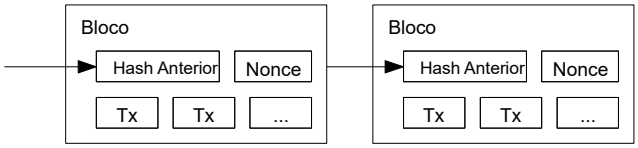
A solução que propomos começa com um servidor timestamp. Um servidor timestamp trabalha gerando um hash de um bloco de itens e publicando amplamente o hash, como em um jornal ou post Usenet [2-5]. O timestamp prova que os dados devem ter existido na época, obviamente, a fim de entrar no hash. Cada timestamp inclui o anterior em seu hash, formando uma cadeia, com cada timestamp adicional reforçando os que vieram antes dele.



4. Prova-de-Trabalho

Para implementar um servidor timestamp distribuído numa base peer-to-peer, teremos de usar um sistema de prova-de-trabalho semelhante ao Hashcash de Adam Back [6], em vez de jornal ou posts Usenet. A prova-de-trabalho envolve a procura por um valor que quando calculado o hash, tal como utilizando o SHA-256, o mesmo comece com um número de bits zero. A média do trabalho requerida é exponencial ao número de bits zero requeridos e pode ser verificada por meio da execução de um único hash.

Para nossa rede de timestamps, nós implementamos a prova-de-trabalho incrementando um nonce no bloco até que seja encontrado um valor que dê ao hash do bloco a quantidade necessária de bits zero. Uma vez que o esforço da CPU tem sido dispendido para satisfazer a prova-de-trabalho, o bloco não pode ser alterado sem refazer o trabalho. Como outros blocos são encadeados posteriormente, o trabalho para mudar um bloco incluiria refazer todos os blocos após ele.



A prova-de-trabalho também resolve o problema da determinação da representação na tomada de decisão da maioria. Se a maioria fosse baseada em um-endereço-IP-um-voto, isso poderia ser subvertido por qualquer pessoa capaz de alocar muitos IPs. Prova-de-trabalho é, essencialmente, uma-CPU-um-voto. A decisão da maioria é representada pela cadeia mais longa, que tem o maior esforço de prova-de-trabalho investido nela. Se a maioria do poder de CPU é controlado por nós honestos, a cadeia honesta vai crescer mais rápido e superar quaisquer cadeias concorrentes. Para modificar um bloco passado, o atacante terá de refazer a prova-de-trabalho do bloco e depois de todos os blocos posteriores e, em seguida, alcançar e superar o trabalho dos nós honestos. Vamos mostrar mais tarde que a probabilidade de um atacante mais lento ter sucesso diminui exponencialmente quando blocos subseqüentes são adicionados.

Para compensar o aumento da velocidade de hardware e o interesse variado em rodar nós ao longo do tempo, a dificuldade da prova-de-trabalho é determinado por uma média móvel visando um número médio de blocos por hora. Se eles são gerados muito rápido, a dificuldade aumenta.

5. Rede

Os passos para executar a rede são os seguintes:

1) Novas transações são transmitidas para todos os nós.
2) Cada nó coleta novas transações em um bloco.
3) Cada nó trabalha para encontrar uma difícil prova-de-trabalho para o seu bloco.
4) Quando um nó encontra uma prova-de-trabalho, ele transmite o bloco para todos os nós.
5) Os nós aceitam o bloco somente se todas as suas transações são válidas e já não foram gastas.
6) Os nós expressam sua aceitação do bloco, trabalhando na criação do próximo bloco na cadeia, usando o hash do bloco aceito como o hash anterior.

Os nós sempre consideram a cadeia mais longa para ser a correta e continuarão trabalhando para estendê-la. Se dois nós transmitirem diferentes versões do bloco seguinte simultaneamente, alguns nós podem receber um ou o outro primeiro. Nesse caso, eles trabalham com o primeiro recebido, mas salvam o outro ramo caso ele se torne o mais longo. O empate será quebrado quando a próxima prova de trabalho for encontrada e um ramo se torne mais longo; os nós que estavam trabalhando em outro ramo, então, mudarão para o mais longo.

Novas transmissões de transação não precisam necessariamente alcançar todos os nós. Contanto que elas atinjam muitos nós, elas vão entrar em um bloco em breve. Transmissões de bloco também são tolerantes com mensagens abandonadas. Se um nó não recebe um bloco, ele irá solicitá-lo ao receber o próximo e perceber um faltante.

6. Incentivo

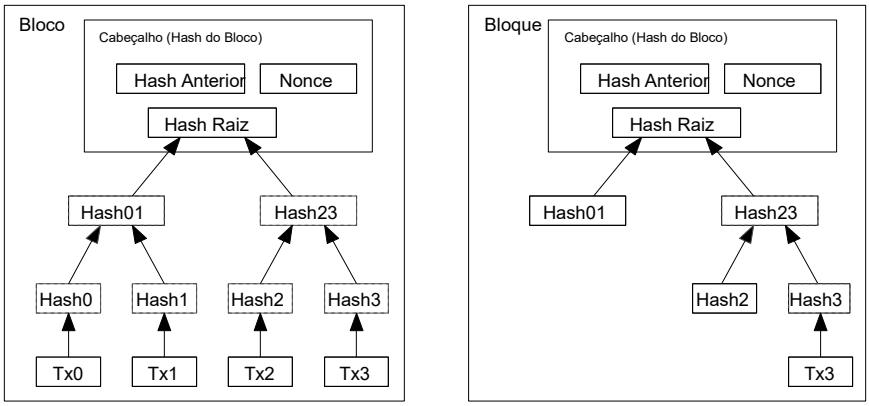
Por convenção, a primeira transação de um bloco é uma operação especial que inicia uma nova moeda de propriedade do criador do bloco. Isso é um incentivo aos nós para apoiar a rede, e fornece uma maneira inicial de colocar moedas em circulação, uma vez que não existe uma autoridade central para emiti-las. A adição estável de uma quantidade constante de novas moedas é análogo a garimpeiros dispender recursos para colocar mais ouro em circulação. No nosso caso, tempo de CPU e eletricidade que estão sendo consumidos.

O incentivo também pode ser financiado com taxas de transação. Se o valor de uma transação de saída é menor que o valor de entrada, a diferença é uma taxa de transação que é adicionada ao valor de incentivo do bloco que contém a transação. Uma vez que um número predeterminado de moedas já entraram em circulação, o incentivo pode migrar inteiramente para taxas de transação e ser completamente livre de inflação.

O incentivo deve encorajar os nós a se manterem honestos. Se um atacante ganancioso é capaz de reunir mais poder de CPU do que todos os nós honestos, ele teria que escolher entre usá-lo para defraudar as pessoas roubando seus pagamentos, ou usá-lo para gerar novas moedas. Ele deve achar que é mais rentável jogar pelas regras, pois tais normas lhe favorecem com mais novas moedas do que todos os outros combinados, além de prejudicar o sistema e a validade de sua própria riqueza.

7. Recuperação do espaço em disco

Uma vez que a transação mais recente de uma moeda está enterrada sob blocos suficientes, as transações gastas anteriormente podem ser descartadas para economizar espaço em disco. Para facilitar isso sem quebrar o hash do bloco, o hash das transações é feito em uma árvore Merkle [7] [2] [5], e apenas a raiz é incluída no hash do bloco. Blocos velhos podem então ser compactados removendo galhos da árvore. Os hashes internos não precisam ser armazenados.



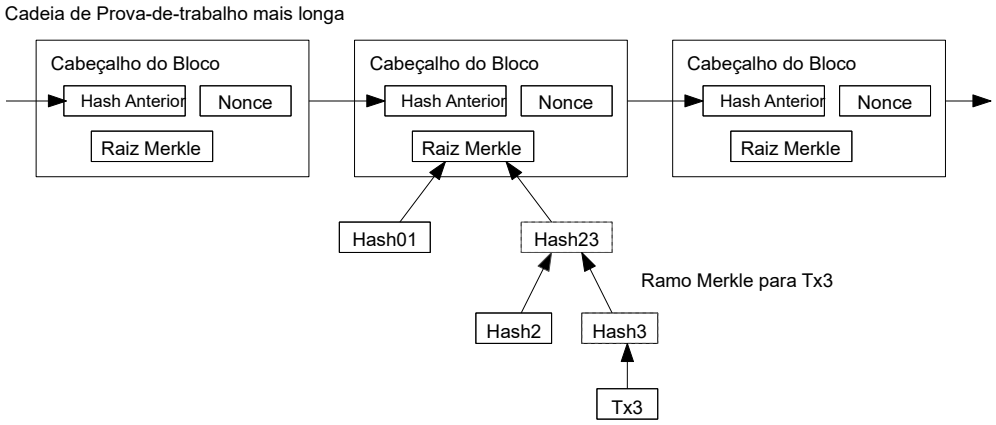
Transações *Hasheadas* numa árvore Merkle

Depois de podar Tx0-2 do Bloco

Um cabeçalho do bloco com nenhuma transação seria de cerca de 80 bytes. Se supormos que blocos são gerados a cada 10 minutos, 80 bytes * 6 * 24 * 365 = 4,2MB por ano. Com computadores normalmente sendo vendidos com 2 GB de RAM em 2008, e pela Lei de Moore prevendo um crescimento de 1,2 GB por ano, o armazenamento não deve ser um problema, mesmo que os cabeçalhos do bloco devam ser mantidos em memória.

8. Verificação de Pagamento Simplificada

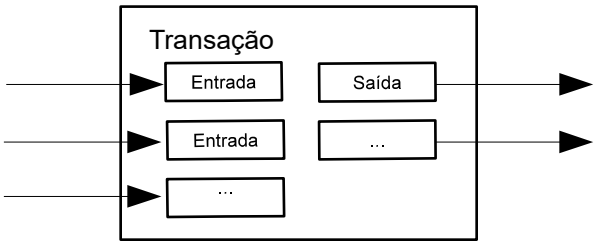
É possível verificar os pagamentos sem executar um nó de rede completo. Um usuário só precisa manter uma cópia dos cabeçalhos do bloco da cadeia com a prova de trabalho mais longa, que ele pode recuperar consultando os nós da rede até que ele esteja convencido de que tem a cadeia mais longa, e obteve o ramo Merkle ligando a transação ao bloco que ela está registrada. Não se pode verificar a transação por si, mas ligando-a a um lugar na cadeia, pode-se ver que um nó de rede a aceitou, e blocos adicionados depois desse confirmam que a rede a aceitou.



Como tal, a verificação é confiável enquanto os nós honestos controlarem a rede, mas é mais vulnerável se a rede for subjugada por um atacante. Enquanto nós da rede podem verificar as transações por si mesmos, o método simplificado pode ser enganado por transações fabricadas de um atacante durante o tempo que o atacante pode continuar a dominar a rede. Uma estratégia para se proteger disso seria aceitar alertas dos nós da rede quando detectam um bloco inválido, levando o software do usuário a baixar o bloco completo e as transações alarmadas para confirmar a inconsistência. As empresas que recebem pagamentos freqüentes provavelmente desejarão executar seus próprios nós para uma segurança mais independente e verificação mais rápida.

9. Combinando e Dividindo Valor

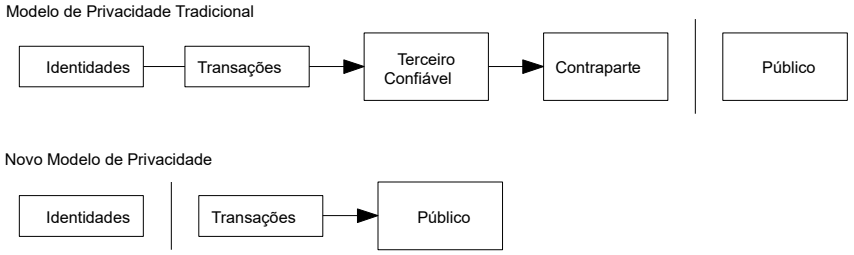
Apesar de ser possível manipular as moedas individualmente, seria pesado fazer uma transação separada para cada centavo numa transferência. Para permitir que o valor seja dividido e combinado, as transações contem múltiplas entradas e saídas. Normalmente haverá uma única entrada de uma transação anterior maior ou várias entradas combinando quantidades menores, e no máximo duas saídas: uma para o pagamento e, se houver, uma retornando o troco ao remetente.



Pode-se notar que a difusão, quando uma transação depende de várias transações, e estas dependem de muitas outras, não é um problema. Nunca há a necessidade de extrair uma cópia autônoma completa do histórico de uma transação.

10. Privacidade

O modelo bancário tradicional atinge um nível de privacidade, limitando o acesso à informação para as partes envolvidas e o terceiro confiável. A necessidade de anunciar todas as transações publicamente se opõe a este método, mas a privacidade ainda pode ser mantida quebrando o fluxo de informação em outro lugar: mantendo chaves públicas anônimas. O público pode ver que alguém está enviando uma quantidade para outra pessoa, mas sem informações que ligam a transação a qualquer um. Isto é semelhante ao nível de informação divulgada pelas bolsas de valores, onde o tempo e o tamanho dos negócios individuais, a "fita", é tornada pública, mas sem dizer quem são as partes.



Como um firewall adicional, um novo par de chaves deve ser utilizado para cada transação para evitar que eles sejam ligados a um proprietário comum. Alguns links ainda são inevitáveis com operações multi-entrada, que necessariamente revelam que as entradas foram para a posse do mesmo proprietário. O risco é que, se o proprietário de uma chave é revelado, associações poderiam revelar outras transações que pertenceram ao mesmo dono.

11. Cálculos

Nós consideramos o cenário de um atacante tentando gerar uma cadeia alternativa mais rápido que a cadeia honesta. Mesmo que seja feito, isso não deixa o sistema aberto a mudanças arbitrárias, como a criação de valor do nada ou tomar dinheiro que nunca pertenceu ao atacante. Os nós não vão aceitar uma transação inválida como forma de pagamento, e nós honestos nunca aceitarão um bloco contendo-as. Um invasor só pode tentar mudar uma de suas próprias transações para ter de volta o dinheiro que recentemente dispendeu.

A corrida entre a cadeia honesta e uma cadeia atacante pode ser caracterizada como uma Caminhada Aleatória Binomial. O evento de sucesso é a cadeia honesta sendo prorrogada por mais um bloco, aumentando a sua liderança por +1, e o evento de falha é cadeia do atacante sendo prorrogada por um bloco, reduzindo a lacuna por -1.

A probabilidade de um atacante recuperar-se de um dado déficit é análogo ao problema da Ruína do Jogador. Suponha que um jogador com crédito ilimitado comece em um deficit e jogue potencialmente um número infinito de tentativas para atingir o breakeven. Podemos calcular a probabilidade de ele atingir o breakeven, ou de um atacante alcançar a cadeia honesta, como segue [8]:

p = probabilidade de um nó honesto encontrar o próximo bloco
q = probabilidade do atacante encontrar o próximo bloco
q_z = probabilidade do atacante alcançar partindo de z blocos atrás

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

Dada a nossa suposição que p > q, a probabilidade cai exponencialmente à medida que o número de blocos que o atacante tem que apanhar aumenta. Com as probabilidades contra, se ele não faz uma arrancada de sorte logo no início, suas chances se tornam muito pequenas quanto mais ele fica para trás.

Vamos agora considerar quanto tempo o destinatário de uma nova transação precisa esperar antes de ficar suficientemente certo que o remetente não pode mudar a transação. Assumimos que o remetente é um atacante que quer fazer o destinatário acreditar por um momento que ele pagou, porém altera o pagamento de volta para si mesmo passado um tempo. O receptor será alertado quando isso acontecer, mas o remetente espera que seja tarde demais.

O receptor gera um novo par de chaves e dá a chave pública para o remetente, pouco antes de assinar. Isso previne que o remetente prepare uma cadeia de blocos antes do tempo, trabalhando de forma contínua até que tenha a sorte de chegar à frente e executar a operação naquele momento. Uma vez que a transação é enviada, o remetente desonesto começa a trabalhar em segredo em uma cadeia paralela contendo uma versão alternativa de sua transação.

O destinatário espera até que a transação tenha sido adicionada a um bloco e z blocos sejam ligados depois. Ele não sabe a quantidade exata de progresso do atacante, mas assumindo que os blocos honestos tomaram o tempo médio esperado por bloco, o progresso potencial do atacante será uma distribuição de Poisson com valor esperado:

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

Para obter a probabilidade do atacante ainda poder alcançar agora, multiplicamos a densidade de Poisson para cada quantidade de progresso que ele poderia ter feito pela probabilidade que ele pudesse alcançar a partir daquele ponto:

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

Rearranjando para evitar a soma da cauda infinita da distribuição...

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

Convertendo para código C...

```
#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;
}
```

Rodando alguns resultados, podemos ver a probabilidade cair exponencialmente com z.

```
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
```

Resolvendo para P menor que 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. Conclusão

Propusemos um sistema de transações eletrônicas, sem depender de confiança. Começamos com o framework habitual de moedas feitas de assinaturas digitais, que fornece um forte controle de propriedade, mas é incompleto sem uma maneira de evitar o gasto duplo. Para resolver isso, propusemos uma rede peer-to-peer usando prova de trabalho para gravar um histórico público de transações que rapidamente se torna computacionalmente impraticável para um atacante para mudar se nós honestos controlarem a maioria do poder de CPU. A rede é robusta em sua simplicidade não estruturada. Os nós trabalham todos de uma vez, com pouca coordenação. Eles não precisam ser identificados, uma vez que as mensagens não são roteadas para qualquer lugar particular e só precisam ser apresentadas em regime de melhor esforço. Os nós podem sair e voltar a rede à vontade, aceitando a cadeia de prova de trabalho, como prova do que aconteceu enquanto eles estavam fora. Eles votam com seu poder de CPU, expressando a aceitação de blocos válidos, trabalhando em estendê-los e rejeitando blocos inválidos, recusando-se a trabalhar com eles. Todas as regras e incentivos necessários podem ser aplicados com este mecanismo de consenso.

Referências

[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.