Bitcoin: Um Sistema de Dinheiro Eletrónico Ponto-a-Ponto

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

Translated in Portuguese from bitcoin.org/bitcoin.pdf
by @rhlinden

Sinopse. Uma versão puramente ponto-a-ponto de dinheiro eletrónico permitiria o envio de pagamentos interativos diretamente de um interveniente para outro sem passar por uma instituição financeira. Assinaturas digitais proporcionam parte da solução, mas os principais benefícios perdem-se se continuar a ser necessária uma terceira entidade de confiança para evitar gastos duplos. Propomos uma solução para o problema do gasto duplo usando uma rede ponto-a-ponto. A rede marca a hora nas transações codificando-as numa cadeia continua de provas-de-trabalho baseada em *hash*, formando um registo que não pode ser alterado sem refazer a prova-de-trabalho. A cadeia mais longa, não só serve de prova da sequência de acontecimentos testemunhados, mas prova que tem origem no grupo de maior capacidade de processamento. Desde que a maioria da capacidade de processamento seja controlada por nós que não estejam conjugados para atacar a rede, eles produzirão a cadeia mais longa e prevalecerão sobre atacantes. A própria rede necessita uma estrutura mínima. As mensagens são difundidas numa base do melhor esforço, e os nós podem abandonar e reintegrar a rede à vontade, aceitando a cadeia mais longa de provas-de-trabalho como prova do que aconteceu enquanto estiveram fora.

1. Introdução

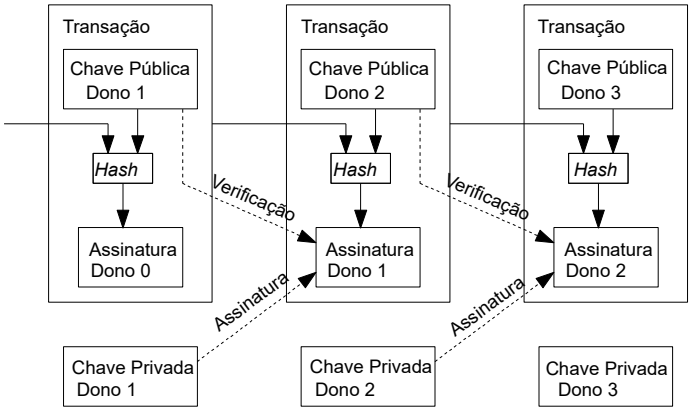
O comércio na Internet vem dependendo quase exclusivamente de instituições financeiras atuando como terceira parte de confiança para o processamento de pagamentos eletrónicos. Embora o sistema funcione suficientemente bem para a maioria das transações, continua a sofrer das fraquezas inerentes ao modelo baseado na confiança. Transações completamente irreversíveis não são possíveis, uma vez que as instituições financeiras não podem evitar a mediação de disputas. O custo da mediação aumenta os custos da transação, limitando o tamanho mínimo praticável e restringindo a possibilidade de pequenas transações casuais, e há um custo mais alargado na perda da capacidade de efetuar pagamentos irreversíveis de serviços irreversíveis. Com a possibilidade de reembolso, a necessidade de confiança aumenta. Os comerciantes devem ser cuidadosos com os seus clientes, exigindo mais informação que a de outra forma seria necessária. Uma certa percentagem de fraude é aceite como inevitável. Estes custos e incertezas do pagamento podem ser evitadas usando moeda física em pessoa, mas não existe mecanismo para fazer pagamentos sobre um canal de comunicações sem uma terceira parte de confiança.

O que é necessário é um sistema eletrónico de pagamento baseado em prova criptográfica e não em confiança, permitindo a duas partes interessadas transacionar diretamente sem a necessidade de uma terceira parte de confiança. Transações de reversão computacionalmente impraticável protegeriam os vendedores da fraude, e mecanismos de garantia podem ser facilmente implementados para proteger compradores. Neste trabalho, propomos uma solução para o problema do gasto duplo usando um servidor distribuído ponto-a-ponto de marcas

temporais para gerar prova computacional da ordem cronológica das transações. O sistema é seguro enquanto os nós honestos controlarem coletivamente mais capacidade de processamento que qualquer grupo coodenado de nós atacantes.

2. Transações

Definimos uma moeda eletrónica como uma cadeia de assinaturas digitais. Cada proprietário transfere a moeda para o próximo assinando digitalmente um *hash* da transação anterior e a chave pública do próximo proprietário e adicionando estas ao fim da moeda. Um recetor pode verificar as assinaturas para verificar a cadeia de propriedade.

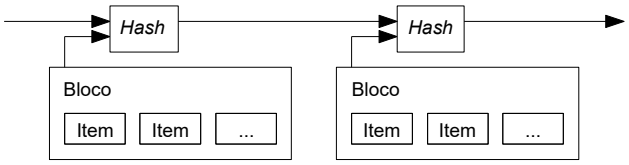


O problema obviamente é que o recetor não pode verificar se um dos proprietários anteriores não terá gasto duas vezes a mesma moeda. Uma solução comum é introduzir uma autoridade central de confiança, ou emissor, que verifique cada transação para gasto duplo. A cada transação, a moeda deveria retornar ao emissor que emitiria então uma nova moeda, e apenas as moedas emitidas diretamente pelo emissor teriam a garantia de não terem sido gastas duas vezes. O problema desta solução é que todo o sistema monetário dependeria da empresa encarregada do emissor, e cada transação teria que passar por ele, tal como um banco.

Precisamos de uma forma do recetor saber que os proprietários anteriores não assinaram nenhuma transação anteriormente. Para este propósito, a transação mais antiga é a transação que conta, pelo que não nos interessam tentativas posteriores de gasto duplo. A única forma de confirmar a falta de uma transação é ter conhecimento de todas as transações. No modelo baseado num emissor, o emissor conhecia todas as transações e saberia qual chegou primeiro. Para conseguir isso sem uma parte de confiança, as transações precisam ser anunciadas publicamente [1], e precisamos de um sistema em que os participantes acordem num único histórico da ordem em que elas foram recebidas. O recetor precisa da prova que na altura de cada transação, a maioria dos nós concordou que ela foi a primeira a ser recebida.

3. Servidor de Marca Temporal

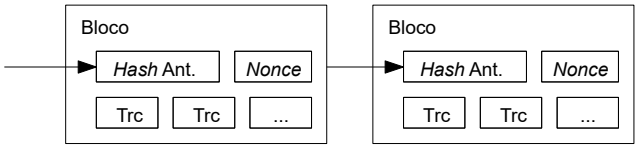
A solução que propomos começa com um servidor de marca temporal. Um servidor de marca temporal funciona usando um *hash* de um bloco de items a ser marcados temporalmente e publicando o *hash* amplamente, como num jornal ou um publicação na Usenet [2-5]. A marca temporal prova que os dados tiveram que existir nessa altura, evidentemente, para que tivessem sido incluídos no *hash*. Cada marca temporal inclui a marca temporal anterior no seu *hash*, formando uma corrente, com cada marca temporal reforçando a anterior.



4. Prova-de-Trabalho

Para implementar um servidor de marca temporal distribuído numa base ponto-a-ponto, precisamos usar um sistema de prova-de-trabalho similar ao de Adam Back Hashcash[6], em vez de um jornal ou publicação na Usenet. A prova-de-trabalho implica procurar um valor que quando codificado, por um algoritmo tal como SHA-256, o *hash* comece por um número de bits zero. O trabalho médio necessário é exponencial com o número de bits zero necessários e pode ser verificado executando um único *hash*.

Para a nossa rede de marcas temporais, implementamos a prova-de-trabalho incrementando um *nonce* no bloco até encontrar um valor que produza o *hash* do bloco com os bits zero necessários. Uma vez despendido o esforço de processamento para satisfazer a prova-de-trabalho, o bloco não pode ser modificado sem refazer o trabalho. Como os blocos seguintes são encadeados após, o trabalho para modificar o bloco inclui também refazer todos os blocos seguintes.



A prova-de-trabalho também resolve o problema de determinar a representatividade na tomada de decisão da maioria. Se a maioria fosse baseada em um-endereço-IP-um-voto, poderia ser subvertida por alguém capaz de reservar muitos IPs. Prova-de-trabalho é essencialmente um-processador-um-voto. A decisão da maioria é representada pela cadeia mais longa, que tem investido em si o maior esforço de prova-de-trabalho. Se a maioria da capacidade de processamento for controlada por nós honestos, a cadeia honesta crescerá mais rapidamente e ultrapassará qualquer cadeia concorrente. Para modificar um bloco anterior, um atacante teria que refazer a prova-de-trabalho desse bloco e de todos os blocos seguintes e ainda alcançar e ultrapassar o trabalho dos nós honestos. Vamos mostrar adiante que a probabilidade de um atacante lento alcançar os honestos diminui exponencialmente à medida blocos subsequentes são adicionados.

Para compensar a velocidade crescente do hardware e a variação de interesse em manter nós em execução ao longo do tempo, a dificuldade da prova-de-trabalho é determinada por uma média variável visando um número médio de blocos por hora. Se forem gerados muito rápido, a dificuldade aumenta.

5. Rede

Os passos para manter a rede são os seguintes:

1) Novas transações são difundidas para todos os nós.
2) Cada nó recolhe novas transações para um bloco.
3) Cada nó tenta encontrar uma prova-de-trabalho difícil para o seu bloco.
4) Quando um nó encontra uma prova-de-trabalho, difunde o bloco para todos os nós.

5) Os nós aceitam o bloco apenas se todas as transações neste são válidas e não foram ainda gastas.
6) Os nós expressam a aceitação do bloco criando o próximo bloco na cadeia, usando o hash do bloco aceite como o hash anterior.

Os nós consideram sempre a cadeia mais comprida como a correta e continuaram a tentar aumentá-la. Se dois nós difundirem versões diferentes do bloco seguinte simultaneamente, alguns nós poderão receber um ou o outro em primeiro lugar. Nesse caso, processam o primeiro recebido, mas guardam o outro ramo caso venha a ser maior. O desempate faz-se quando a próxima prova-de-trabalho for determinada e um ramo ficar maior; os nós que estavam a processar o outro ramo mudarão para o maior..

A difusão de novas transações não necessita chegar necessariamente a todos os nós. Desde que chegue a bastantes nós, eles acabarão por determinar um bloco. As difusões de blocos também toleram a perda de mensagens. Se um nó não receber um bloco, irá solicitá-lo quando receber o próximo bloco e constatar que faltou um.

6. Incentivo

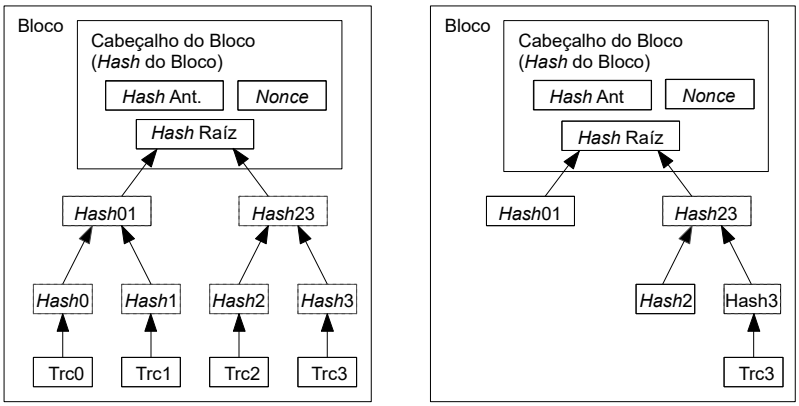
Por convenção, a primeira transação de um bloco é uma transação especial que inicía uma nova moeda de propriedade do criador desse bloco. Isto dá um incentivo para os nós suportarem a rede, e constitui uma forma de introduzir moedas em circulação uma vez que não há uma autoridade central que as emita. A constante adição de uma quantidade constante de novas moedas é semelhante a mineiros gastando recursos para adicionar ouro à circulação. No nosso caso, tempo de processamento e eletricidade investidos.

O incentivo também pode ser financiado por taxas sobre as transações. Se o valor de saída de uma transação for menor que o valor de entrada, a diferença é a taxa que é adicionada ao valor do incentivo do bloco que contem a transação. Depois de um determinado número de moedas entrar em circulação, o incentivo pode passar a ser exclusivamente constituído por taxas sobre a transação e completamente livre de inflação.

O incentivo pode encorajar os nós a permanecer honestos. Se um atacante ganancioso conseguir reunir maior capacidade de processamento que todos os nós honestos, terá ainda que escolher entre usá-la para enganar as pessoas roubando os seus pagamentos, ou usá-la para gerar novas moedas. Deverá achar mais rentável cumprir as regras, as mesmas que o favorecem com mais novas moedas que todos os restantes em conjunto, que comprometer o sistema e a validade da sua própria riqueza.

7. Recuperação do Espaço em Disco

Depois da última transação de uma moeda ter sido sobreposta por um número suficiente de blocos, as transações de despesa anterior podem ser descartadas para poupar espaço em disco. Para facilitar esse objetivo sem comprometer o *hash* do bloco, as transações são codificadas numa árvore Merkle [7][2][5], incluindo apenas a raiz no *hash* do bloco. Blocos antigos podem ser compactados removendo ramos da árvore. Os *hashes* interiores não necessitam ser mantidos.



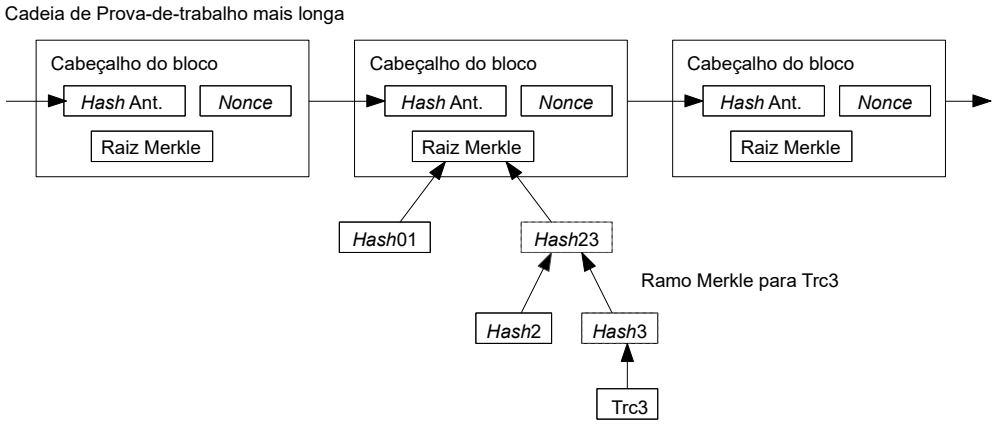
Transações *Hashed* numa árvore Merkle

Depois de remover Trc0-2 do Bloco

Um cabeçalho de bloco sem transações deverá ter cerca de 80 bytes. Se considerarmos blocos gerados a cada 10 minutos, 80 bytes * 6 * 24 * 365 = 4.2MB por ano. Com computadores a ser vendidos tipicamente com 2GB de RAM em 2008, e a Lei de Moore prevendo o crescimento de 1.2GB por ano, o armazenamento não deverá ser um problema mesmo que os blocos tenham que ser mantidos em memória.

8. Verificação de Pagamento Simplificada

É possível verificar pagamentos sem operar um nó de rede completo. O utilizador só necessita manter uma cópia dos cabeçalhos de bloco da cadeia de maior prova-de-trabalho, que pode obter questionando nós de rede até estar convencido que obteve a mais longa, e obter o ramo Merkle ligando a transação ao bloco com a marca temporal correspondente. Não pode confirmar a transação por si próprio, mas ao estabelecer uma relação com um ponto da cadeia, pode verificar que um nó da rede a aceitou, e blocos posteriores confirmam ainda que a rede a aceitou.

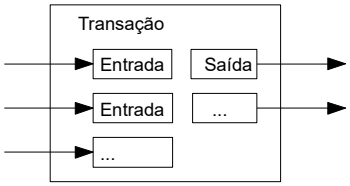


Assim a verificação é confiável desde que os nós honestos controlem a rede, mas é mais vulnerável se a rede for dominada por um atacante. Apesar dos nós da rede poderem verificar eles próprios transações, o método simplificado pode ser enganado por transações fabricadas de um atacante enquanto este puder continuar a dominar a rede. Uma estratégia para proteger contra isso seria aceitar alertas dos nós da rede quando estes detetam um bloco inválido, solicitando ao software do utilizador para descarregar o bloco e as transações sujeitas ao alerta para confirmar a inconsistência. Empresas que recebam pagamentos frequentes ainda vão querer provavelmente

operar os seus próprios nós para mais segurança independente e rápida verificação.

9. Combinando e dividindo valores

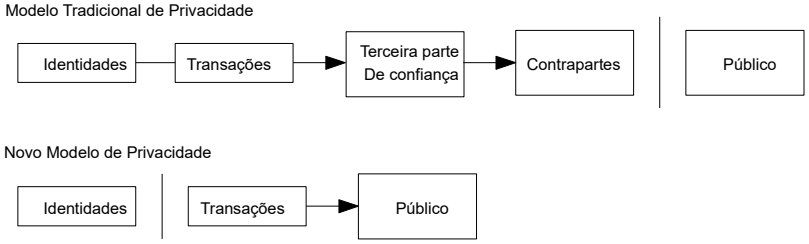
Embora fosse possível gerir moedas individualmente, seria trabalhoso fazer uma transação separada para cada cêntimo de uma transferência. Para permitir a divisão e combinação de valor, as transações contêm múltiplas entradas e saídas. Normalmente haverá uma entrada única de uma transação maior anterior ou múltiplas entradas combinando montantes inferiores, e no máximo duas saídas: uma para o pagamento, e uma devolvendo o troco, se houver, de volta para o remetente.



De notar que a dispersão, onde uma transação depende de múltiplas transações, e essas dependem de ainda mais, não representa um problema. Nunca há a necessidade de extrair uma cópia completamente independente da história de uma transação.

10. Privacidade

O modelo bancário tradicional obtém algum nível de privacidade limitando o acesso à informação às partes envolvidas e à terceira parte de confiança. A necessidade de anunciar todas as transações publicamente inviabiliza este método, mas a privacidade pode ainda ser mantida quebrando o fluxo de informação em outro ponto: mantendo as chaves públicas anónimas. O público pode ver que alguém está a enviar um montante a outra pessoa, mas sem informação que possa relacionar a transação a alguém. É similar ao nível de informação publicado pela bolsa de valores, onde data e montante dos negócios individuais, a "fita", é tornada pública, mas sem divulgar quais as partes.



Como proteção adicional, um novo par de chaves deverá ser usado para cada transação para evitar que sejam relacionados a um proprietário comum. Algum relacionamento é inevitável com transações de múltipla entrada, que revelam necessariamente que as entradas pertencem ao mesmo proprietário. O risco é que, se o proprietário de uma chave for revelado, o relacionamento pode revelar outras transações pertencentes ao mesmo proprietário.

11. Cálculos

Consideremos o cenário de um atacante tentar gerar uma cadeia alternativa mais rapidamente que a cadeia honesta. Mesmo que o consiga, não torna o sistema aberto a alterações arbitrárias, como

criar valor a partir do nada ou apropriar-se de dinheiro que nunca pertenceu ao atacante. Os nós não iram aceitar uma transação inválida como pagamento, e os nós honestos nunca aceitarão um bloco que a contenha. Um atacante só poderá tentar alterar uma das suas próprias transacções para recuperar dinheiro que tenha gasto recentemente.

A competição entre a cadeia honesta e a cadeia atacante pode ser caracterizada como Passeio Aleatório Binomial. O evento de sucesso será a cadeia honesta ser aumentada de um bloco, aumentando a sua liderança de +1, e o insucesso é a cadeia atacante ser aumentada de um bloco, reduzindo a diferença de -1.

A probabilidade de um atacante recuperar de um determinado deficit é semelhante ao problema da Ruína do Jogador. Imagine-se um jogador com crédito ilimitado iniciando com deficit jogando um número potencialmente infinito de jogadas para tentar atingir o equilíbrio. Podemos calcular a probabilidade de alguma vez atingir o equilíbrio, ou que um atacante alguma vez atingir a cadeia honesta, como se segue [8]:

p = probabilidade de um nó honesto encontrar o próximo bloco
q = probabilidade de um atacante encontrar o próximo bloco
q_z = probabilidade do atacante alguma vez recuperar de z blocos de atraso

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

Se assumirmos que p > q, a probabilidade cai exponencialmente com o aumento do número de blocos que o atacante tem que recuperar. Com a probabilidade contra ele, se não fizer um avanço de sorte muito cedo, as probabilidades tornam-se minúsculas enquanto vai ficando para trás.

Consideremos agora quanto tempo o recetor de uma nova transação deverá esperar até ter suficiente certeza que o emissor não pode modificar a transação. Assumimos que o emissor é um atacante que pretende que o recetor acredite momentaneamente que lhe pagou, e após algum tempo reverterá o pagamento para ser reembolsado. O recetor será alertado quando isso acontecer, mas o emissor espera que seja demasiado tarde.

O recetor gera uma novo par de chaves e dá a chave pública ao emissor pouco tempo antes da assinatura. Isto impede o emissor de preparar uma cadeia de blocos antecipadamente, trabalhando nela continuamente até ter a sorte de se distanciar e efetuar a transação nesse momento. Assim que a transação for enviada, o emissor desonesto começará a trabalhar secretamente numa cadeia paralela contendo uma versão diferente desta transação.

O recetor aguarda até que a transação seja adicionada a um bloco e z blocos tenham sido ligados após este. Ele não sabe exatamente o progresso atingido pelo atacante, mas assumindo que o bloco honesto demorou a média esperada para cada bloco, o potencial progresso do atacante será uma distribuição Poisson com o valor esperado:

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

Para obter a probabilidade que o atacante poderá recuperar, multiplicamos a densidade do Poisson por cada progresso que poderíamos efetuar pela probabilidade de alinharmos desde esse ponto:

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

Reorganizando para evitar a 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 ProbabilidadeSucessoAtacante(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;
}
```

Obtendo alguns resultados, podemos ver a redução exponencial da probabilidade 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 para transações eletrónicas sem depender da confiança. Começamos com a estrutura usual de moedas baseadas em assinaturas digitais, que proporcionam um grande controlo de posse, mas que é incompleta sem um meio de prevenir o gasto duplo. Para resolver este problema, propomos uma rede ponto-a-ponto usando uma prova-de-trabalho para fazer o registo histórico das transações que rapidamente se tornam de mudança impraticável para um atacante se os nós honestos controlarem a maioria da capacidade de processamento. A rede é robusta dada a sua simplicidade desestruturada. Os nós trabalham em simultâneo com pouca coordenação. Eles não necessitam ser identificados, uma vez que as mensagens não são encaminhadas para para uma localização particular e só precisam ser entregues numa base do melhor esforço. Os nós podem abandonar e retornar à rede à vontade, aceitando a cadeia de prova-de-trabalho como prova do que aconteceu enquanto estiveram ausentes. Votam com a sua capacidade de processamento, exprimindo a sua aceitação de blocos válidos trabalhando na sua extensão e rejeitando blocos inválidos rejeitando a sua evolução. Quaisquer regras ou incentivos podem ser obrigados com base neste mecanismo de consenso.

References

[1] W. Dai, "b-money," http://www.weidai.com/bmoney.txt, 1998.

[2] H. Massias, X.S. Avila, e J.-J. Quisquater, "Design of a secure timestamping service with minimal trust requirements," In *20th Symposium on Information Theory in the Benelux*, Maio 1999.

[3] S. Haber, W.S. Stornetta, "How to time-stamp a digital document," In *Journal of Cryptology*, vol 3, num 2, páginas 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*, páginas 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*, páginas 28-35, Abril 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, páginas 122-133, Abril 1980.

[8] W. Feller, "An introduction to probability theory and its applications," 1957.