Bitcoin: Um sistema de dinheiro eletrônico ponto-a-ponto

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org
Tradução: Daniel Ribeiro
daniel@outros.net

Prefácio. Uma versão puramente ponto-a-ponto de dinheiro eletrônico pode permitir o envio de pagamentos online diretamente de uma parte a outra sem ser através de uma instituição financeira. Assinaturas digitais providenciam parte da solução, mas os maiores benefícios são perdidos se um intermediário confiável ainda for necessário para prevenir o gasto duplo.

Nós propomos uma solução para o problema de gasto duplo usando uma rede ponto-a-ponto.

A rede "carimba uma data" nas transações codificando-as em uma corrente contínua de prova de trabalho baseada em codificação, formando um registro que não pode ser modificado sem que a prova de trabalho seja refeita. A maior corrente não apenas serve como prova de sequência de eventos testemunhados, mas prova que eles vieram do maior conjunto de poder de processamento. Enquanto a maior parte do poder de processamento é controlado por nós que não estão cooperando para atacar a rede, eles irão gerar a maior corrente e ultrapassar os atacantes. A rede em si requer uma estrutura mínima. Mensagens são distribuídas na base da "melhor forma possível", e os nós podem sair e voltar à rede á vontade, aceitando a corrente com a maior prova de trabalho como prova do que aconteceu enquanto ele esteve fora.

Introdução

O comércio pela Internet tem se tornado quase exclusivamente construído sobre instituições financeiras servindo como intermediários confiáveis para processar os pagamentos eletrônicos. Enquanto o sistema funciona bem o suficiente para a maioria das transações, ele sofre da fraqueza inerente do modelo baseado em confiança.

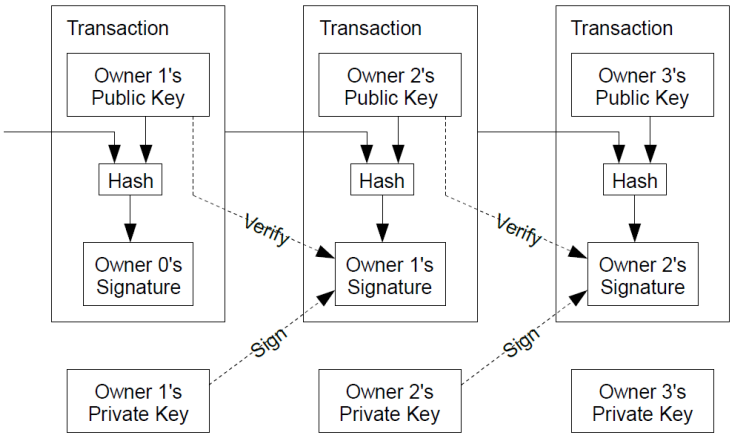
Transações completamente irreversíveis não são realmente possíveis, já que as instituições financeiras não podem evitar ter que mediar disputas. O custo desta mediação aumenta os custos de transação, criando então um valor mínimo para uma transação ser viável e cortando então a possibilidade de existirem transações pequenas e casuais, e há um grande custo em perder a habilidade de fazer pagamentos não-estornáveis para serviços não-estornáveis. Com a possibilidade do estorno, a necessidade de confiança se distribui. Vendedores devem tentar se proteger de seus clientes, obrigando-os a fornecer mais informações do que seriam necessárias de outra forma, pois os mesmos podem comprar produtos e serviços, pagar eletronicamente, e logo após estornar o pagamento.

Um certo percentual de fraude é aceito como inevitável. Estes custos e incerteza de pagamentos podem ser evitados com uma pessoa usando uma moeda física, mas não existe mecanismo que faça pagamentos sobre canais de comunicação sem um intermediário confiável.

O que é preciso é um sistema de pagamentos eletrônicos baseado em provas criptográficas em vez de confiança, que permita que duas partes interessadas em fazer transações diretamente façam-nas sem a necessidade de um intermediário confiável. Transações que são computacionalmente impraticáveis de reverter podem proteger vendedores de fraude, e serviços de proteção poderiam ser facilmente implementados para proteger os compradores (algo como PagSeguro, Paypal ou MercadoPago fazem atualmente). Neste artigo, nós propomos uma solução para o problema do gasto-duplo usando um servidor de "carimbos de tempo" distribuído ponto-a-ponto para gerar uma prova computacional de ordem cronológica de transações. O sistema é seguro enquanto os nós honestos coletivamente controlam mais poder de processamento do que qualquer grupo cooperado de nós atacantes.

Transações

Nós definimos uma moeda eletrônica como uma cadeia de assinaturas digitais. Cada proprietário transfere a moeda para o próximo assinando digitalmente uma codificação com as transações anteriores e a chave pública do próximo proprietário e adicionando estas ao fim da moeda. Um recebedor pode verificar as assinaturas para verificar a cadeia de propriedade.



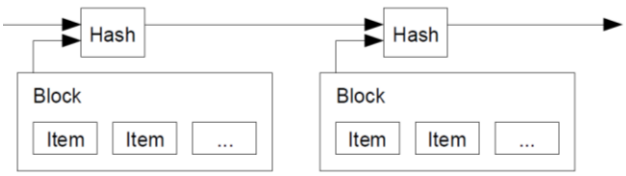
O problema obviamente é que o recebedor não pode verificar se um dos proprietários anteriores não gastou duplamente a moeda. Uma solução é introduzir uma autoridade central confiável, ou emissor, que verificaria cada transação contra o gasto duplo. A cada transação, a moeda seria retornada ao emissor que emitiria então uma nova moeda, e apenas as moedas emitidas diretamente do emissor seriam confiáveis de não terem sido gastas duas vezes.

O problema com esta solução é que todo o sistema monetário dependeria da empresa rodando o emissor, e cada transação teria que passar por ele, exatamente como um banco.

Nós precisamos de uma forma de o recebedor saber que os donos anteriores não assinaram nenhuma nova transação. Para este propósito, a transação mais antiga é a transação que conta, então nós não nos importamos sobre novas tentativas de gasto duplicado. A única forma de confirmar a validade de uma transação e estar ciente de todas as transações. Em um modelo baseado e emissor, o emissor está ciente de todas as transações e sabe qual delas chegou primeiro. Para atingir o objetivo sem um intermediário confiável, as transações precisam ser publicamente anunciadas [1], e nós precisamos de um sistema em que os participantes concordem em um único histórico da ordem em que elas foram recebidas. O recebedor precisa provar que no tempo de cada transação, a maioria dos nós concordou que ele foi o primeiro a receber.

Servidor de Carimbos de tempo

A solução que nós propomos começa com um servidor de carimbos de tempo. Um servidor de carimbos de tempo funciona pegando a codificação de um bloco de itens a serem “carimbados” e publicando largamente esta codificação, como em um jornal ou um post na Usenet [2-5]. O carimbo de tempo prova que os dados precisam ter existido naquele tempo, obviamente, para que sejam incluídos na codificação. Cada carimbo de tempo inclui o carimbo de tempo anterior em sua codificação, formando uma corrente, onde cada carimbo de tempo adicional reforçará os blocos anteriores.

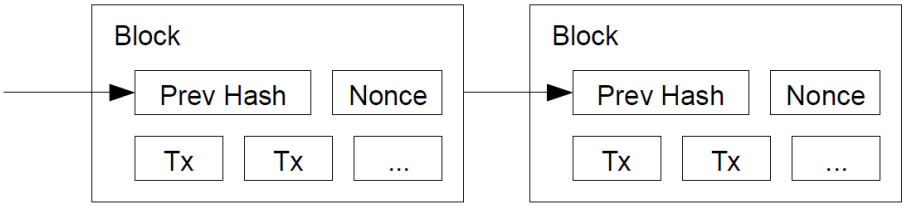


Prova de trabalho

Para implementar um servidor de carimbo de tempo distribuído em uma base ponto-a-ponto, nós vamos precisa usar um sistema de prova de trabalho similar ao Dinheiro Codificado de Adam Back [6], em vez de um jornal ou um post da Usenet.

A prova de trabalho envolve procurar por um valor que foi codificado por um algoritmo como SHA-256, a codificação começa com um número de zero bits. O trabalho médio necessário é exponencial em número de "zero bits" necessários e pode ser verificado executando uma única codificação.

Para a nossa rede de carimbos de tempo, nós vamos implementar uma prova de trabalho incrementando um número aleatório no bloco até que o valor encontrado sirva para decodificar o bloco com o número de zero bits. Uma vez que o esforço de CPU foi gasto para satisfazer a prova de trabalho, o bloco não pode ser modificado sem que o trabalho seja refeito. Como os blocos seguintes são encadeados logo após este, o trabalho para modificar o bloco inclui também refazer todos os blocos posteriores.



A prova de trabalho também envolve o problema em determinar a representação na tomada de decisão da maioria. Se a maioria é determinada como "um IP, um voto", ela pode ser subvertida por qualquer um capaz de alocar muitos IPs. Prova de trabalho é essencialmente "um CPU, um voto". A decisão da maioria é representada pela corrente mais longa, que tem o maior esforço de prova de trabalho investido nela. Se a maior parte do poder de processamento é controlado por nós honestos, a corrente honesta irá crescer mais rapidamente e ultrapassar qualquer corrente competidora. Para modificar um bloco antigo, um atacante teria que refazer a prova de trabalho do bloco e de todos os blocos posteriores e então ultrapassar o trabalho dos nós honestos. Nós vamos mostrar adiante que a probabilidade de um atacante lento alcançar os honestos diminui exponencialmente, a medida que os blocos subsequentes são adicionados a corrente honesta.

Para compensar o aumento da velocidade do hardware e a variedade de interesse em rodar os nós a medida que o tempo passa, a dificuldade da prova de trabalho é determinada por uma meta variável em número de blocos por hora. Se os blocos estão sendo gerados muito rápido, a dificuldade aumenta.

Rede

Os passos para rodar a rede são os seguintes:

1. Novas transações são transmitidas para todos os nós.
2. Cada nó coleta as novas transações em um bloco.
3. Cada nó trabalha para encontrar uma prova de trabalho difícil para o seu bloco.
4. Quando um nó encontra uma prova de trabalho, ele transmite o bloco para todos os nós.
5. Nós aceitam o bloco apenas se todas as transações nele são válidas e não houve gasto-duplo.
6. Nós expressam sua aceitação ao bloco ao começar a trabalhar na criação do próximo bloco da corrente, usando a codificação do bloco aceito como a codificação anterior.

Nós sempre consideram a maior corrente como a corrente correta e vão manter-se trabalhando em extende-la. Se dois nós transmitem versões diferentes do mesmo bloco simultaneamente, alguns nós vão receber um ou outro primeiro. Neste caso, eles vão trabalhar no primeiro que receberam, mas vão manter a outra versão no caso de ela se tornar a maior. O impasse será solucionado quando a próxima prova de trabalho for encontrada e uma das versões se tornar maior. Os nós que estavam trabalhando na outra versão então vão mudar e passar a trabalhar na versão maior.

Transmissões de novas transações não necessariamente precisam alcançar todos os nós. Enquanto elas alcançarem muitos nós, elas vão entrar em algum bloco uma hora ou outra. As transmissões de blocos também são tolerantes a mensagens perdidas. Se um nó não receber um bloco, ele vai solicitar ele quando ele receber o próximo bloco e perceber que perdeu um.

Incentivo

Por convenção, a primeira transação de um bloco é uma transação especial que inicia uma nova moeda de propriedade do criador do bloco. Isso adiciona um incentivo para os nós suportarem a rede, e provê uma forma de inicialmente distribuir moedas em circulação, já que não existe uma autoridade central para emitir essas moedas.

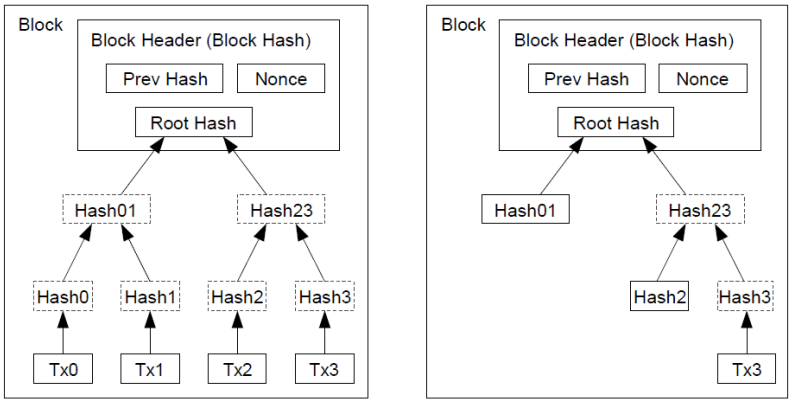
A adição de uma quantidade constante de novas moedas é análoga a dos mineradores de ouro gastando recursos para colocar mais ouro em circulação. No nosso caso, o tempo de processamento e a eletricidade são gastos.

O incentivo também pode ser financiado por taxas de transação. Se o valor de saída de uma transação é menor que o valor de entrada, a diferença é a taxa de transação que foi adicionada ao valor de incentivo do bloco que contem a transação. Uma vez que um número pré-determinado de moedas já tiver entrado em circulação, o incentivo pode mudar completamente para taxas de transação e poderá ser completamente livre de inflação. O incentivo também pode ajudar a encorajar os nós para permanecer honestos. Se um atacante poderoso é capaz de ter mais poder de processamento do que todos os nós honestos juntos, ele provavelmente vai ter que escolher entre fraudar as pessoas pegando de volta os pagamentos que fizer, ou usar este poder de processamento para ganhar novas moedas. Ele provavelmente vai achar mais lucrativo trabalhar conforme as regras, já que elas vão lhe mais moedas do que a todos os outros juntos do que prejudicar o sistema e, juntamente, suas próprias moedas.

Preocupação com espaço em disco

Uma vez que a última transação de uma moeda é embutida em blocos suficientes, as transações gastas anteriormente podem ser descartadas para salvar espaço em disco. Para facilitar isso sem quebrar a codificação do bloco, transações são codificadas em uma árvore Merkle [7][2][5], em que apenas a raiz é incluída na codificação dos blocos.

Blocos antigos podem então ser compactados ao remover as versões mais antigas da árvore. As codificações interiores não precisam ser armazenadas.

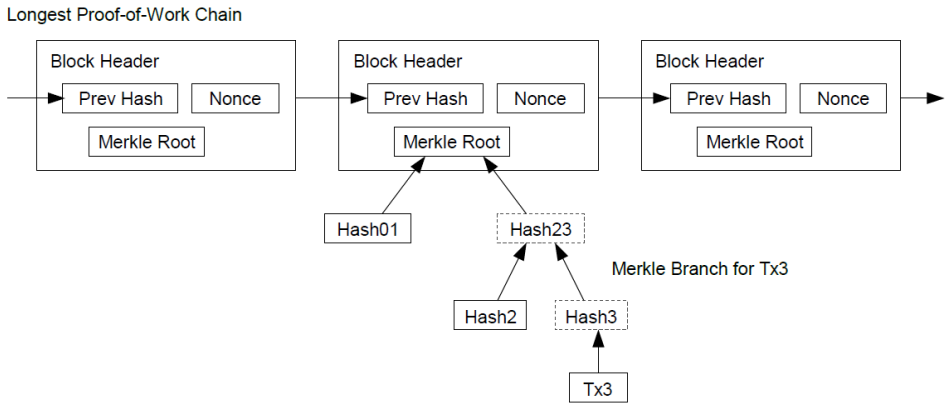


Transactions Hashed in a Merkle Tree — After Pruning Tx0-2 from the Block

Um cabeçalho de bloco sem transações deve ter em torno de 80 bytes. Se nós supusermos que os blocos são gerados a cada 10 minutos, 80 bytes * 6 * 24 * 365 = 4.2MB por ano. Com os sistemas de computador típicos a venda em 2008 com 2GB de RAM, e a lei de Moore prevendo um aumento de 1.2GB por ano, armazenamento não deve ser um problema mesmo se os cabeçalhos dos blocos forem mantidos na memória.

Verificação simplificada de pagamento

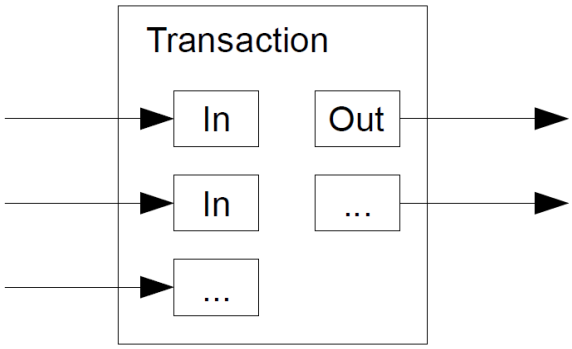
É possível verificar pagamentos sem ter que rodar um nó de rede completo. Um usuário precisa apenas manter uma cópia dos cabeçalhos dos blocos da corrente com a maior de prova de trabalho, que ele pode obter consultando os nós da rede até se convencer que ele tem a maior corrente, e obter a ramificação Merkle que une a transação ao bloco onde ela está carimbada. Ele não pode verificar a transação por si só, mas ligando ela ao local na corrente, ele pode ver que este nó aceitou a transação, e blocos adicionados após este irão confirmar que a rede a aceitou.



Como sempre, a verificação é confiável contanto que os nós honestos controlem a rede, mas é mais vulnerável se grande parte do poder de processamento da rede estiver trabalhando para um atacante. Enquanto os nós podem verificar transações por si só, o método mais simples pode ser enganado por uma transação fabricada por um atacante enquanto o atacante continuar a dominar o processamento da rede. Uma estratégia para proteger-se contra este problema seria aceitar alertas dos nós da rede quando eles detectam um bloco inválido, pedindo no software do usuário que ele baixe o bloco completo e as transações alertadas para confirmar a inconsistência. Negócios que recebem pagamentos frequentes vão provavelmente querer rodar seus próprios nós para uma segurança mais independente e verificações mais rápidas.

Combinando e dividindo valor

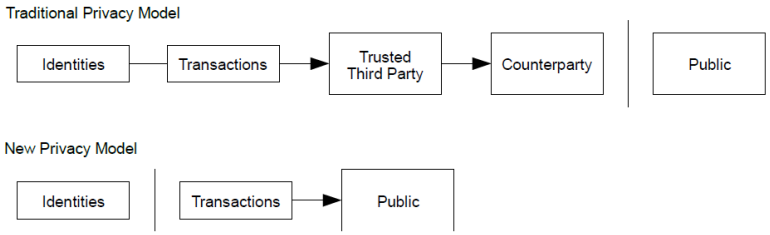
Mesmo que seja possível manipular as moedas individualmente, seria pouco prático separar as transações para cada centavo a ser transferido. Para permitir que o valor seja dividido e combinado, transações devem permitir múltiplas entradas e múltiplas saídas. Normalmente haverá tanto uma única entrada de uma grande transação anterior ou muitas entradas combinando pequenos valores e no máximo duas saídas: Uma para o pagamento e uma para devolução de troco, se houver, de volta para o pagador.



Deve-se notar que o fato de uma transação depender de diversas transações, e estas transações dependerem de muitas outras não é um problema. Nunca haverá a necessidade de extrair uma cópia completa do histórico de uma transação.

Privacidade

O modelo bancário tradicional atinge um nível de privacidade por limitar o acesso à informação apenas às partes envolvidas e ao intermediário confiável. A necessidade de anunciar todas as transações publicamente prejudica este método, mas a privacidade pode ainda ser mantida quebrando o fluxo da informação em outro local: mantendo as chaves públicas anônimas. O público pode ver que alguém está enviando um montante de dinheiro para outra pessoa, mas sem informações ligando a transação a qualquer pessoa. Isso é parecido com o nível de informação liberado no mercado de ações, onde o tempo e o tamanho das vendas individuais, a "fita", são tornados públicos, porém sem dizer quem são as partes envolvidas.



Como uma prevenção adicional, um novo par de chaves poderia ser usado para cada transação para evitar que sejam ligadas a um proprietário comum. Algum relacionamento ainda é inevitável com transações de muitas entradas, que necessariamente revelarão que suas entradas pertencem ao mesmo proprietário. O risco é que se este proprietário de uma chave for revelado, ligações poderiam revelar outras transações que pertenciam ao mesmo proprietário.

Cálculos

Nós consideramos o cenário em que um atacante tentando gerar uma corrente alternativa mais rápido do que a corrente honesta. Mesmo se isso for conseguido, ela não vai tornar o sistema aberto para mudanças arbitrárias, como criar valor do nada ou tomar o dinheiro que nunca pertenceu ao atacante. Nós não irão aceitar uma transação inválida como pagamento, e nós honestos nunca aceitarão um bloco contendo-as. Um atacante pode apenas tentar modificar uma de suas próprias transações para pegar de volta o dinheiro que ele gastou recentemente.

A corrida entre a corrente honesta e a corrente do atacante pode ser caracterizada como uma Caminhada Aleatória Binomial. O evento de sucesso é a corrente honesta ser estendida por um bloco, aumentando sua liderança em +1, e o evento de falha é a corrente do atacante ser estendida por um bloco, reduzindo o atraso em -1.

A probabilidade de um atacante alcançar de um determinado déficit é análoga ao problema da Ruina do Apostador. Suponha que um apostador com créditos ilimitados comece em déficit e jogue potencialmente um infinito número de vezes para tentar quebrar a banca. Nós vamos calcular a probabilidade de ele nunca conseguir isso, ou que um atacante simplesmente alcance a corrente honesta, como segue [8]:

p = probabilidade de um nó honesto encontrar o próximo bloco

q = probabilidade de um atacante encontrar o próximo bloco

qz = probabilidade algum dia o atacante alcançar estando z blocos atrás

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

Assumindo que p > q, a probabilidade cai exponencialmente a medida que o número de blocos que o atacante tem que alcançar aumenta. Com as possibilidades contra ele, se ele não levar sorte logo no inicio, suas chances vão sumindo e ele acaba ficando muito atrasado.

Nós agora consideramos o quão longo o recibo de uma nova transação precisa aguardar antes que seja suficientemente correto que o pagador não poderá mudar a transação. Nós assumimos que um pagador é um atacante que quer fazer seu recebedor acreditar que ele o pagou por algum tempo, então mudar a transação para receber o dinheiro de volta depois que algum tempo se passou. O recebedor vai ser alertado quando isso acontecer, mas o pagador espera que este aviso venha tarde demais.

O recebedor gera um novo par de chaves e dá a chave pública para o pagador logo antes de assinar. Isso previne que o pagador ganhe tempo e prepare uma corrente de blocos a frente do bloco atual, e então executar a transação neste momento. Uma vez que a transação seja enviada, o pagador desonesto começa a trabalhar em segredo em uma corrente paralela contendo uma versão alternada da transação.

O recebedor aguarda até que a transação seja adicionada ao bloco e z blocos já tenham sido ligados após ele. Ele não sabe o exato montante do progresso que o atacante conseguiu, mas assumindo que os blocos honestos tomaram a média esperada de tempo por bloco, o atacante potencial vai ter a posse de distribuição com o valor esperado:

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

Para obter a probabilidade de um atacante ainda alcançar agora, nós multiplicamos a porção de cada medida de progresso que ele poderia fazer pela probabilidade de ele alcançar a partir deste ponto:

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

Reorganizando para evitar a soma de uma causa infinita na distribuição...

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

Convertendo em 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;
}
```

Executando 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
```

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
```

Conclusão

Nós propusemos um sistema de transações eletrônicas sem basear-se em confiança. Nós começamos com a estrutura comum de moedas feitas de assinaturas digitais, que providenciam forte controle de propriedade, mas é incompleta sem um sistema de prevenção de gasto duplo. Para resolver isso, nós propusemos uma rede ponto-a-ponto usando provas de trabalho para armazenar em um histórico público de transações que rapidamente se torna inviável para um atacante modificar se os nós honestos controlarem a maior parte do poder de processamento. A rede é robusta em sua simplicidade não-estruturada. Nós trabalham todos ao mesmo tempo com pouca coordenação entre si. Eles não precisam ser identificados, já que as mensagem não são encaminhadas para nenhum local em particular e apenas precisam ser entregues na base da "melhor forma possível". Nós pode deixar e voltar a rede a vontade, aceitando a corrente de prova de trabalho como prova do que aconteceu enquanto ele esteve fora. Eles votam com seu poder de processamento, expressando sua aceitação dos blocos válidos ao trabalhar em estende-los e rejeitando blocos inválidos ao recusar a trabalhar neles. Quaisquer regras necessárias e incentivos podem ser forçados 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.

Notas de tradução

Alguns termos técnicos são usados no documento original em inglês, e que não fazem muito sentido ao traduzir para o português. Abaixo uma relação deles, para facilitar a compreensão por um leitor não muito familiarizado com os termos originais.

Carimbo de tempo

O "Carimbo de tempo" é uma tradução livre para "Timestamp" – "Time" é tempo (horário), e "Stamp" é carimbo. Em informática, um "Timestamp" é uma forma de registrar uma data e hora de maneira única e confiável, de modo inviolável. Quando um registro qualquer precisa ser fixado em uma data e hora específica, o recurso de "Carimbo de tempo" é usado de modo confiável, pois ao realizar qualquer alteração posterior neste registro, o carimbo original é perdido e um novo deve ser criado, já que o carimbo de tempo também possui em sua codificação parte do conteúdo do registro.

Deste modo, podemos garantir que qualquer alteração neste registro vai, obrigatoriamente, modificar o seu carimbo. Assim, se quisermos saber se o registro foi modificado, basta comparar o carimbo que tínhamos em memória com o carimbo atual do registro.

Codificação

Em muitos momentos durante o texto o termo "Codificação" é usado como uma tradução para "Hashing". Não existe palavra em português que traduza "Hashing", então a tradução foi baseada em seu significado contextual. "Hashing" vem de "Hash", que é uma sequência de caracteres que representa uma "assinatura" de uma determinada informação. A "assinatura" de uma informação é quando uma pequena cadeia de caracteres é gerada a partir de um volume maior de dados, de modo que qualquer modificação nestes dados também vai alterar esta assinatura.

A utilidade disso é que podemos garantir que os dados não foram modificados, basta apenas verificar a assinatura destes dados – já que qualquer modificação alteraria esta assinatura.

"Hashing" ficaria melhor como "Assinando", mas devido ao contexto, o processo de assinatura nada mais é do que o de gerar um código único que representa todos os dados do bloco, então a preferência pelo termo "Codificando".

Redes ponto-a-ponto

Uma rede ponto-a-ponto (peer-to-peer) consiste em uma topologia de rede onde não há um nó centralizado controlado o fluxo de informações entre os diversos nós da rede. Cada nó pode estar ligado a diversos outros nós diretamente, e caso um precise se comunicar com algum nó que não está ligado diretamente a este, ele simplesmente "divulga" a informação, e esta se propaga pela rede até atingir o nó necessário. Cada nó da rede é responsável não só por enviar e receber informações diversas, mas também por propagar os pacotes enviados com esta finalidade.

Este modelo, justamente por ser decentralizado, provê estabilidade contra controle externo, já que é praticamente impossível impedir o funcionamento de uma rede deste tipo. É possível isolar os nós, mas a rede como um todo, não.