O protocolo Bitcoin

bruno cuconato

https://github.com/odanoburu/

Dezembro 2016

Preâmbulo

Este texto foi escrito como trabalho final para um curso superior acadêmico ao mesmo tempo em que foi pensado como uma introdução didática (porém razoavelmente detalhada) ao protocolo Bitcoin, em português (já que em inglês há muitos outros recursos). Desculpem os formalismos, as notas de rodapé, e as (até que poucas) referências e citações.

licença

Conteúdo

1 Introdução

> *"When the division of labour has been once thoroughly established, it is but a very small part of a man's wants which the produce of his own labour can supply. [...] Every man thus lives by exchanging, or becomes in some measure a merchant, and the society itself grows to be what is properly a commercial society."*
>
> — Adam Smith, *Wealth of Nations* [1]

A Bitcoin é mais que uma moeda digital: é um protocolo, um conjunto de regras escritas em *software* que governa desde a emissão de Bitcoin até as suas formas de transação. Outra característica dessa criptomoeda é a sua inserção em uma nova tecnologia de balanços descentralizados — a *block chain* —, que será detalhada posteriormente.

A ideia que deu origem à Bitcoin foi exposta em um *white paper* [2] publicado sob o pseudônimo Satoshi Nakamoto em 1º de novembro de 2008 em uma *mailing list* sobre criptografia.[i] Até agosto de 2016 já se haviam levantado diversas hipóteses sobre quem seria(m) o(s) criador(es) da criptomoeda, indo de colaboradores de Nakamoto, passando por prósperos empresários e por acadêmicos, chegando até imigrantes japoneses em solo americano.[ii]

O criador (ou criadora, ou criadores) da Bitcoin deixou o projeto em abril de 2011, pouco tempo depois de lançar a primeira versão funcional do protocolo Bitcoin, escrita majoritariamente na linguagem de programação C++. [3] Nenhum de seus vários colaboradores sabia sua identidade e as aproximadamente 1 milhão de bitcoins em seu nome estão imóveis desde então — seu valor de mercado supera BRL 1 bilhão.

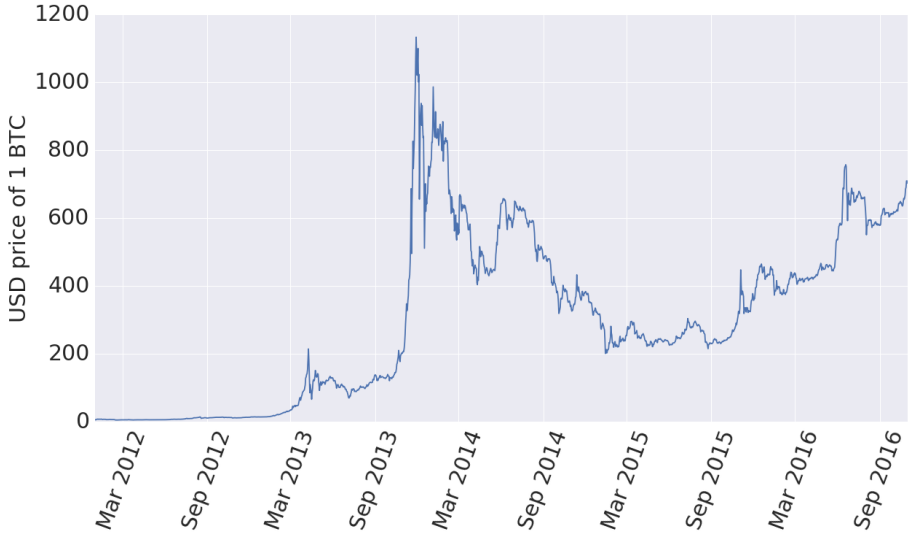
Mesmo sem Nakamoto, a comunidade Bitcoin cresceu junto com o valor da criptomoeda. Além de uma média de 200.000 transações diárias por volta de setembro de 2016, há um ecossistema de aplicativos, de empresas e de organizações ligadas à

[i] Mensagem disponível em https://web.archive.org/web/20101207101311/http://www.mail-archive.com/cryptography@metzdowd.com/msg09959.html.

[ii] A Wikipédia brasileira em setembro de 2016 ainda continha uma afirmação conclusiva sobre a identidade de Satoshi Nakamoto, baseada em um de diversos furos de reportagem que posteriormente se revelaram furos de raciocínio.

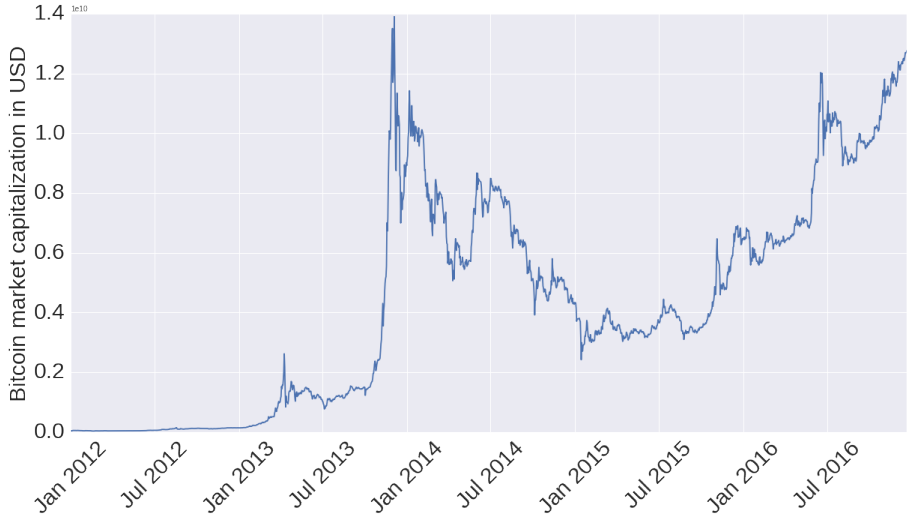
Bitcoin. O sucesso dessa inovação também deu origem a mais de cem criptomoedas diversas, tais como Ethereum, Ripple, Litecoin, etc. A Bitcoin continua preponderante em relação às demais — seu valor de mercado (USD 9 bilhões em setembro de 2016) com folga o de todas as outras criptomoedas somadas (ver Figura 2).

Figura 1: Taxa de câmbio $\frac{USD}{BTC}$, 2012-2016



Fonte: Elaboração própria, a partir de API BitcoinAverage.

Figura 2: Capitalização de mercado da Bitcoin em dezenas de bilhões de USD, 2012-2016



Fonte: Elaboração própria, a partir de API Blockchain.info.

2 O protocolo Bitcoin

Ao longo deste trabalho o protocolo Bitcoin será explicado com o uso de dois recursos didáticos: analogias com a economia tradicional e o uso de dois exemplos motivadores, uma transação entre dois amigos — Ada e Babbage — e uma competição entre os mineradores Fili e Kili.

Na economia tradicional, se Ada quer comprar um livro de Babbage pela quantia de 50 reais, ela segue um procedimento simples: entrega uma cédula de papel desse valor para ele. Com isso, Ada deixa de ter os 50 reais e recebe o livro. Babbage, por sua vez, passa a ter 50 reais a mais do que tinha. Simples, mas para que isso aconteça é preciso que ambos estejam fisicamente no mesmo lugar, ou então que eles confiem em um intermediário que faça a cédula chegar de Ada a Babbage. Como isso poderia funcionar de modo digital, sem a necessidade de proximidade física entre os dois?

Pode-se substituir, por exemplo, a cédula de papel pelo seu número de série. Como cada número de série é único, em tese é possível saber com quem está cada nota. Transferir dinheiro, nesse caso, é simplesmente informar o número de série de uma cédula ao destinatário. Assim, Ada envia para Babbage a seguinte mensagem: `AH051984677`, correspondente a uma nota de 50 reais.

Um problema surge, no entanto. Babbage recusa a mensagem de Ada porque ele não tem certeza de que ela não vai enviar o mesmo número de série para outra pessoa.

Esse é um dos obstáculos para uma moeda digital, a possibilidade de *gasto duplo*. Se as partes estão fisicamente próximas, esse problema não existe — por exemplo, no momento em que Ada entrega sua cédula a Babbage, ela não tem a possibilidade de gastá-la novamente. Se os dois estão em cidades diferentes, no entanto, Ada precisa confiar que Babbage vai colocar nos correios o livro comprado ao receber o dinheiro que ela enviou, ou então ele precisa confiar que ela vai enviar o dinheiro por correio quando o livro chegar — a depender do acordo que eles fizerem. Na economia digital imaginada, por sua vez, o mesmo problema ocorre: ao pagar Babbage, Ada continua tendo todas as informações para recriar a cédula que gastou.

A economia tradicional solucionou esse problema de maneira inteligente: se Ada e Babbage confiam em um intermediário — geralmente, um banco — esse fica responsável por debitar e creditar os saldos dos dois, impedindo gastos duplos.

O protocolo Bitcoin resolve esse problema de forma diferente. Ao invés de criar um intermediário (como o banco), o protocolo torna todas as transações públicas. Nesse caso, Babbage aceitaria a mensagem de Ada porque ele pode verificar no histórico de transações que Ada recebeu de alguém `AH051984677` no passado e que ela nunca enviou esse número de série para ninguém antes.

Ainda que as transações em bitcoin sejam públicas, um certo grau de privacidade é resguardado pelo uso de pseudônimos, de modo que é possível ver todas as transações, mas sem saber quem foi envolvido em cada operação — a menos que se conheça o dono do pseudônimo. Como uma pessoa pode ter vários pseudônimos, a privacidade no uso da criptomoeda deixa de ser uma questão importante.[iii]

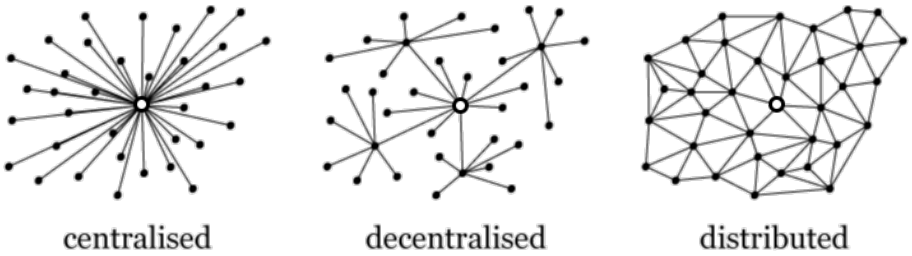
As transações feitas em bitcoin são gravadas na *block chain*, que é um balanço público distribuído. É um balanço porque nela estão registradas todas as operações e, portanto, é possível obter o saldo de cada pseudônimo; e é distribuída porque é feita em rede — qualquer pessoa pode obter uma cópia do balanço de outros membros e passar a editá-la. Não há autoridade central — o que é registrado no balanço depende do consenso de seus membros.

[iii] Ainda assim, há outras criptomoedas que buscam a anonimidade total, como a Dash e a ZCash.

3 Transações

A Bitcoin é uma rede *peer-to-peer*, i.e., uma malha que não apresenta hierarquização entre seus componentes, não existindo um nó central que coordene todas as transações. Analogamente, quando Ada decide enviar 1 BTC para Babbage em troca de um livro, na prática, ela envia a transação para alguns membros da rede, que, por sua vez, a transmitem para outros membros e, assim, rapidamente a transação chega a todos os integrantes do protocolo Bitcoin.

Figura 3: Propagação de uma transação em vários tipos de rede



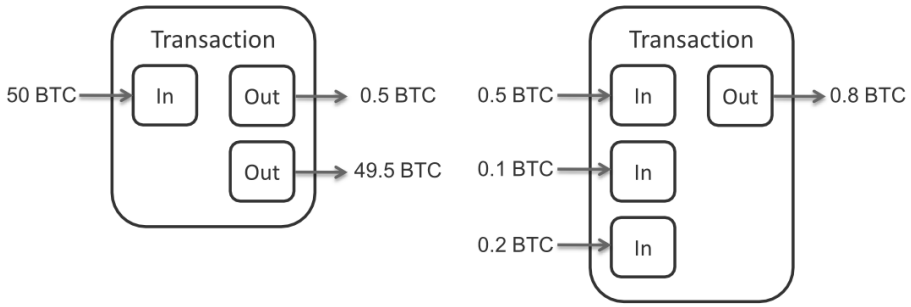
Fonte: Elaboração própria a partir de Wikimedia Commons.

Uma transação em bitcoin é composta por dois principais elementos: as transações-entrada ("entradas") e as transações-saída ("saídas"). A criptomoeda em questão é divisível em até 8 casas decimais, com cada 0.00000001 BTC sendo chamado de satoshi em homenagem ao seu criador. Ao contrário da moeda tradicional, bitcoin não existe em formatos pré-definidos, como em moedas de 25 e 50 centavos ou em cédulas de 10 e 20 reais. Enviar 1 BTC usando 100.000.000 de satoshis seria tão prático quanto pagar uma conta num restaurante usando moedas de um centavo! É por isso que ao invés de registrar cada satoshi independentemente, o protocolo pensa as bitcoins como cédulas que podem ser de qualquer tamanho. Essas são, na verdade, transações-saída de operações anteriores, como se vê na Figura 4. Assim, em Bitcoin não há saldos em conta corrente — sequer há contas. O que há são *outputs* não gastos de transações anteriores.

Caso Ada pretenda enviar 1 BTC para Babbage, ela precisa escolher saídas de alguma transação anterior como entradas da presente transação.[iv] Suponha-se que

[iv] Na prática, essa tarefa é feita automaticamente por um algoritmo do *software* usado para transmitir as operações.

Figura 4: Transações podem juntar mais de uma entrada em uma única saída, ou podem desmembrar uma entrada em várias transações-saída



Fonte: Matthäus Wander, Wikimedia Commons.

há dois dias, Ada recebeu 2.5 BTC de Morgan. A saída dessa transação é escolhida por ela como entrada da operação com Babbage.

O protocolo Bitcoin não aceita gasto parcial de uma transação-entrada: é preciso gastar tudo o que entrou em transações-saída ou em taxas de transação.[v] Assim, se Ada não quiser gastar toda bitcoin das transações-entrada com Babbage e com as taxas de transação, ela precisa colocar uma transação-saída para si mesma, numa espécie de transação-troco — embora na prática ela esteja recebendo o troco de si mesma e não de Babbage (ver Figura 5).

Além disso, Ada precisa escolher saídas e taxas de transação de modo a obedecer:

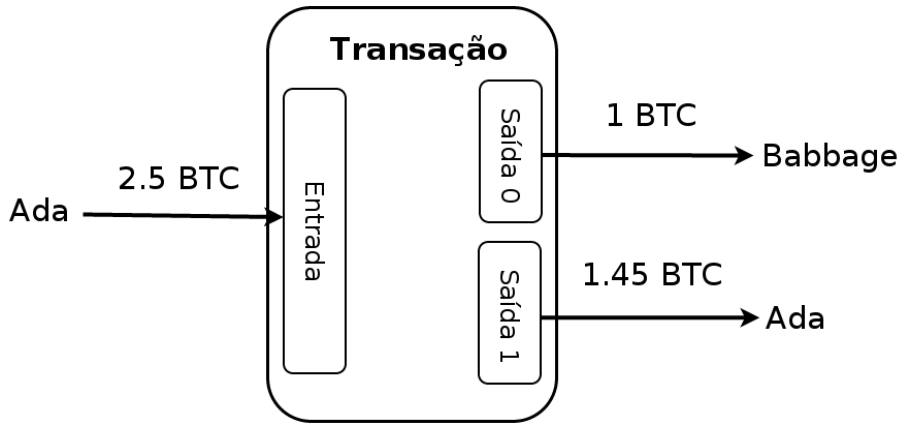
$$input\_txs \equiv output\_txs + tx\_fees \tag{1}$$

Ou seja, o valor das transações-entrada tem de ser igual ao valor das transações-saída somado ao valor das taxas de transação. É importante mencionar que as taxas de transação são calculadas implicitamente, i.e., elas não são uma forma de transação-saída.[vi]

[v] O motivo para isso será melhor compreendido quando a *block chain* for explicada, mas, resumidamente, essa regra foi criada por razões de simplicidade e de segurança.

[vi] O motivo para isso será explicado na Seção 5.

Figura 5: Ada usa 2.5 BTC recebidos de Morgan para enviar 1 BTC a Babbage, pagando 0.05 BTC como taxa de transação



Fonte: Elaboração própria.

3.1 Criptografia

Transações como a feita por Ada são enviadas aos membros da rede Bitcoin. Antes que um dos membros dessa rede possa aceitar a transação feita como válida, é preciso verificar que a transação recebida foi, de fato, feita por Ada. No mundo digital, seria muito simples enviar uma transação para a rede fingindo ser Ada ou qualquer outra pessoa.

Para que se entenda como funciona a verificação de transações é preciso aprender um pouco de criptografia.

3.1.1 Criptografia Simétrica e Assimétrica

Há dois tipos de criptografia: a simétrica e a assimétrica. Na criptografia simétrica, há uma chave secreta que é usada tanto para encriptar quanto para decriptar mensagens. Para descrever o funcionamento da criptografia, é útil fazer uma analogia como a proposta por Vryonis.[vii]

Para Vryonis, a criptografia simétrica equivale a um baú com uma chave. Ada pode usar sua chave para abrir o baú e depositar, por exemplo, uma mensagem. Se ela deseja entregar o conteúdo do baú para Babbage, ela pode enviá-lo por correio

[vii] Disponível em https://web.archive.org/web/20161025182457/https://blog.vrypan.net/2013/08/28/public-key-cryptography-for-non-geeks/.

(ou usar qualquer outro intermediário), desde que Babbage já tenha uma cópia da chave que abre o baú. Essa condição imposta ao receptor garante a inviolabilidade do baú. Por esse motivo, Babbage tem certeza de que o conteúdo está íntegro e que a mensagem foi colocada por Ada. Esse é o objetivo da criptografia no protocolo Bitcoin: garantir a autoria do conteúdo enviado, além de sua integridade.[viii]

No mundo digital, a criptografia simétrica é muito utilizada para encriptar arquivos *offline*. Ela é relativamente mais rápida do que a criptografia assimétrica e está presente em celulares e computadores. Para fins de comunicação na internet, entretanto, a criptografia simétrica falha. Pessoas que estejam fisicamente próximas podem trocar pessoalmente uma chave secreta a fim de se comunicarem, mas isso não funciona pela internet — afinal, se a mensagem não-criptografada estiver vulnerável a *hackers*, uma senha enviada por mensagem não-criptografada também estaria vulnerável, o que ameaça toda a comunicação posterior criptografada por essa senha. No Apêndice A, mostra-se um esquema básico de criptografia simétrica, que pressupõe o compartilhamento prévio de uma chave secreta.

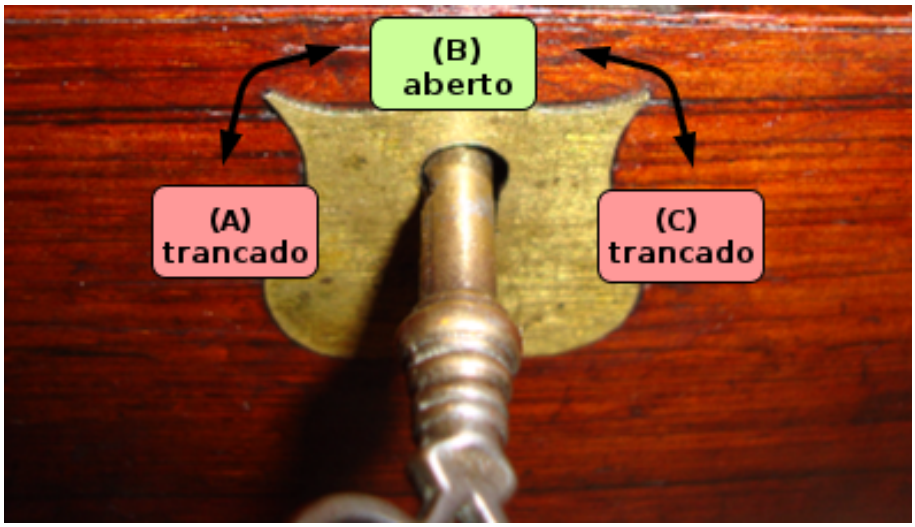
Na criptografia assimétrica, por outro lado, não há uma só chave, mas duas. Uma delas é pública e serve tanto para encriptar mensagens dirigidas ao seu dono como para decriptar mensagens enviadas por ele. Essa chave pública é semelhante a um endereço postal ou a um endereço de e-mail — as pessoas precisam conhecê-la para enviar mensagens, porque a chave identifica a pessoa que se comunica. Ela pode ser enviada por um canal não-seguro, uma vez que não é possível usá-la para se passar por seu dono — da mesma forma como se pode publicar um endereço de e-mail sem temer pela segurança de futuras mensagens.

A outra chave empregada na criptografia assimétrica é privada, funcionando como uma senha pessoal. Somente essa chave consegue desvendar uma mensagem criptografada usando a chave pública, por conta da relação matemática entre elas. Supondo a existência de um *hacker* sem conhecimento da chave privada, ele teria grande dificuldade em decifrar a criptografia porque a chave privada não é facilmente deduzível da chave pública. O Apêndice A implementa em Python uma versão simplificada de um dos esquemas mais utilizados de criptografia assimétrica, o RSA.

[viii] Uma pequena introdução à criptografia, em português, pode ser encontrada em https://web.archive.org/web/20160408113912/http://www.di.ufpe.br/~flash/ais98/cripto/criptografia.htm

Na analogia de Vryonis, a criptografia assimétrica também equivale a um baú inviolável. Esse baú, contudo, tem uma fechadura diferente, com três estágios, como se vê na Figura 6.

Figura 6: Criptografia assimétrica no baú de Vryonis



Fonte: Elaboração própria, a partir de Koppas e Panayotis Vryonis. Wikimedia Commons.

Nessa analogia, o baú possui dois tipos de chaves: as públicas e a privada. Ada, por exemplo, mantém sua chave privada segura, mas distribui para os conhecidos as chaves públicas. A chave privada de Ada, quando inserida no baú, só se movimenta no sentido horário, i.e., de A (trancado) para B (aberto) e de B para C (trancado). As chaves públicas, ao contrário, só se movem no sentido anti-horário: elas vão de C (trancado) para B (aberto) e de B para A (trancado). Não é possível realizar voltas completas na fechadura, i.e., não é possível ir de A para C sem passar por B e vice-versa.

Como Babbage pode usar esse baú para se comunicar de maneira segura com Ada? Ora, basta que ele deposite uma carta no baú de Ada, trancando-o em seguida ($A \leftarrow B$). A mensagem foi encriptada. Com o baú trancado na posição A, só a chave privada de Ada — que se movimenta no sentido horário — pode abrir o baú, i.e., decriptar a mensagem de Babbage.

Se Ada deseja enviar uma mensagem segura para Babbage, são precisos dois baús de Vryonis. No primeiro baú, pertencente a Ada, ela deposita sua carta e o tranca em C ($B \rightarrow C$). Ada então coloca seu baú no de Babbage. Ela só dispõe da

chave pública de Babbage e por isso tranca o segundo baú na posição A ($A \leftarrow B$). Assim, temos uma mensagem em dois baús: o baú exterior só pode ser destrancado (decriptado) com a chave privada de Babbage; o baú interior pode ser aberto com qualquer chave pública de Ada. A combinação dos dois garante que só Babbage pode abrir o baú e que ele tenha certeza de que o conteúdo dos baús foi preparado por Ada — o que garante tanto a integridade da mensagem quanto sua segurança. Caso Ada empregasse seu baú para enviar a mensagem, haveria apenas a garantia de sua autoria. Nesse caso, a segurança estaria comprometida porque qualquer pessoa com a chave pública de Ada poderia abrir o baú (decriptar a mensagem).

3.1.2 Função *hash*

O segundo componente criptográfico fundamental para a compreensão do protocolo Bitcoin são as funções *hash*. Neste trabalho não será explicado como funcionam essas funções, somente o que elas fazem.

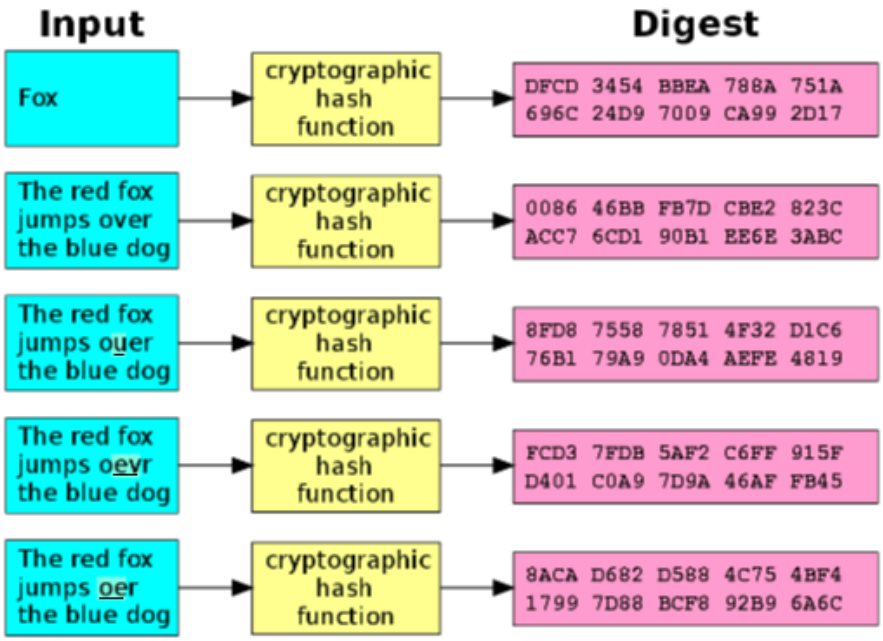
As funções *hash* mapeiam informação de tamanho arbitrário para um valor fixo chamado de valor *hash*, ou *digest* [4]. Uma característica das funções *hash* é que, apesar de seus *outputs* aparentemente aleatórios, um mesmo valor inserido numa mesma função *hash* produzirá sempre o mesmo resultado; se, no entanto, um valor ligeiramente diferente for inserido, o resultado será completamente diferente. Funções *hash* normalmente são usadas para fornecer um resumo criptográfico (*digest*) de informações, de modo a facilitar a comparação entre elas. Na Figura 7, a função *hash* SHA-1[ix] é aplicada a vários *inputs*.

Os *digests* produzidos por uma função *hash* facilitam a comparação entre informações por reduzirem seus tamanhos. Ao invés de se comparar dois textos letra por letra ou palavra por palavra para ver se houve alteração de um para o outro, pode-se olhar para o *digest* dos dois textos. Caso tenha ocorrido uma pequena alteração no texto, por menor que seja, o *digest* deles será diferente.[x]

[ix] Na internet há diversas páginas que aplicam essa função *hash* a textos e números, e.g., http://sha1-hash-online.waraxe.us/.

[x] Na verdade, como o *output* da função *hash* tem tamanho fixo, o domínio da função é potencialmente maior do que a imagem, i.e., haverá mais de um *input* para cada *output* possível. Isso não é um problema caso seja computacionalmente difícil de encontrar quais *inputs* possuem o mesmo *output*.

Figura 7: Pequenas variações no *input* provocam grandes variações no *output* de uma função *hash* comum



Fonte: Jorge Stolfi. Wikimedia Commons.

Para o uso criptográfico feito no protocolo Bitcoin, uma função *hash* precisa atender a alguns requisitos adicionais. Além de produzir resultados aparentemente não-correlacionados para *inputs* semelhantes e seu cálculo computacional ter de ser rápido (a fim de garantir a eficiência dos algoritmos), deve ser difícil encontrar dois *inputs* com o mesmo *digest*. No entanto, a propriedade fundamental de uma função *hash* para uso criptográfico é que não deve haver meio conhecido de obter o *input* a partir de um *output*, i.e., o cálculo da função inversa da função *hash* deve ser desconhecido ou inexistente.

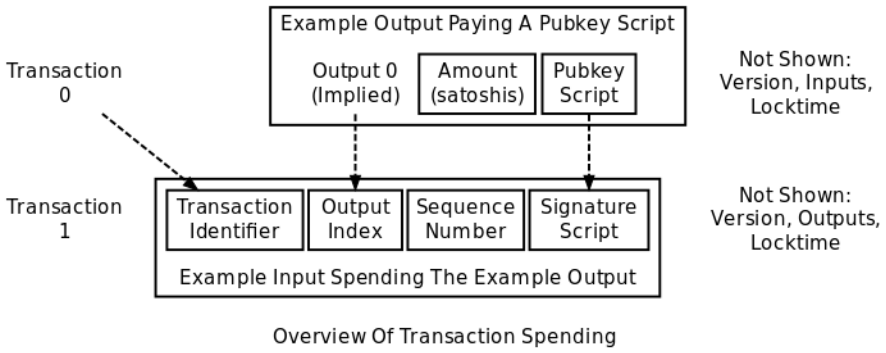
Um exemplo esclarece essa propriedade desejável das funções *hash*. Caso um *hacker* conheça o *digest* `03a1 ba61 0409 9a85 fb20 202a e912 b171 472c 758a` ele só tem um caminho para tentar descobrir o *input* que o originou: a força bruta. O *hacker* terá de inserir vários valores possíveis na função *hash* até encontrar o *digest* procurado. Isso ocorre porque não se conhece a operação inversa das funções *hash*, ao contrário, por exemplo, das operações matemáticas elementares (se multiplicamos um número por outro, sabemos que para inverter essa operação basta dividir o resultado pelo número multiplicador e, assim, obter o número original).

Matematicamente, dadas uma mensagem x_1 e uma função *hash* $h(x)$ com as propriedades acima, temos: $\forall\, x_1,\, x_2 \neq x_1$, é difícil encontrar $\{\, h(x_2) \mid h(x_2) = h(x_1)\}$.

3.2 Verificação de propriedade das bitcoins

Retomando o exemplo motivador do início da Seção 3, para que Ada envie 1 BTC para Babbage usando os 2.5 BTC que recebeu de Morgan, ela precisa primeiro provar para os seus pares da rede Bitcoin que de fato é proprietária dos 2.5 BTC. Em toda transação, as saídas são trancadas por uma condição que autoriza seu gasto. Essa condição é expressa numa linguagem de programação simples [5]. Essa linguagem, chamada *Script*, permite uma série de possibilidades de trancamento.[xi] No tipo mais comum, *Pay-to-Public-Key-Hash*, tranca-se o valor da transação-saída com o *hash* da chave pública e com uma assinatura digital do destinatário pretendido. Outros tipos de trancamento permitem a liberação dos recursos somente com a combinação de várias assinaturas digitais, o que permite a gestão coletiva de bitcoins em uma empresa, por exemplo.

Figura 8: O gasto de uma transação-saída tipo *Pay-to-Public-Key*



Fonte: Harding, Bitcoin Developer Documentation [6]

Na Figura 8 vê-se uma transação *Pay-to-Public-Key*,[xii] um dos tipos mais simples de transação. Pode-se ver como a Transação 1 referencia a Transação 0: o

[xi] Essas possibilidades não estão plenamente disponíveis em Bitcoin no momento, pois há um acordo coletivo dos participantes em aceitar somente alguns tipos padrão de transação. Ver a criptomoeda ethereum para possibilidades menos limitadas.

[xii] Esse tipo de transação quase não é mais utilizado, tendo sido substituído pelo *Pay-to-Public-Key-Hash* mencionados previamente. Os dois tipos são muito semelhantes, o primeiro não exigindo o *hash* da chave pública, mas sim a própria chave pública.

transaction identifier é o *hash* da transação e o *output index* é o índice da saída a que se faz referência.

Na transação 0, índice 0 está presente a condição de liberação dos fundos — nesse caso, o *PubKey Script*. Esse requisito demanda a apresentação de uma assinatura digital que comprove a posse da chave privada que corresponde à chave pública presente na condição. Empregando novamente a analogia do baú de Vryonis, se Ada quiser gastar o dinheiro que recebeu de Morgan, basta que forme uma transação válida e a coloque no baú, trancando-o com sua chave privada. Qualquer membro da rede Bitcoin poderá confirmar que Ada possui a chave privada que corresponde à chave pública à qual foi enviada o dinheiro de Morgan.

Matematicamente, a assinatura digital corresponde à encriptação da transação com a chave privada, a qual pode ser desfeita (e verificada) com a chave pública, valendo:

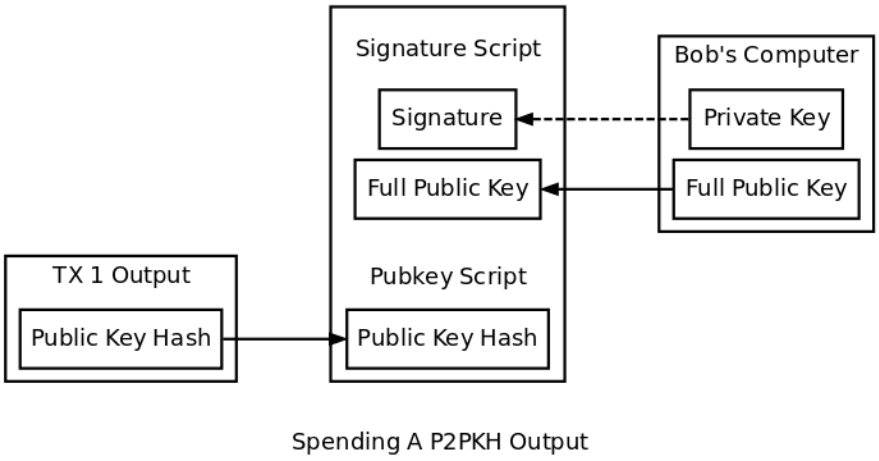
$$E(m, \mathit{ChavePriv}) = c \tag{2}$$

$$D(c, \mathit{ChavePub}) = m \tag{3}$$

nas quais m é a mensagem (conteúdo da transação) e c é a mensagem cifrada (mensagem encriptada pela chave privada, a assinatura digital). Note que a assinatura, c, depende tanto da chave privada quanto da mensagem a ser transmitida, o que impede a alteração do conteúdo da mensagem sem alteração da assinatura, i.e., $E(m', \mathit{ChavePriv}) = c' \neq c = E(m, \mathit{ChavePriv})$. Uma assinatura analógica, por sua vez, independe do conteúdo assinado, o que permite um *transplante* de assinaturas impossível no mundo digital.

Na Figura 9, vê-se as partes da transação que formam m, a serem assinadas. Como os pares da rede Bitcoin conhecem essas partes previamente, eles podem formar m de maneira independente de Ada. Ao receberem c de Ada, eles usam a Equação 3, com a chave pública presente na condição de liberação da transação-saída. Ao decriptarem a assinatura dela com sua chave pública, espera-se obter a mesma mensagem m. Se o resultado obtido for de fato igual a m, a assinatura fornecida por Ada corresponde à chave pública que libera os fundos da transação enviada por

Figura 9: O gasto de uma transação-saída tipo *Pay-to-Public-Key-Hash*



Fonte: Harding, Bitcoin Developer Documentation [6]

Morgan. Veja a Tabela 1 para mais detalhes.

O *script* de liberação de fundos é composto pela chave pública do destinatário (Ada) `<pubKey>` e pelo comando `OP_CHECKSIG`. `<sig>` é a assinatura digital fornecida na tentativa de liberação dos fundos, i.e., c. `OP_CHECKSIG` é um comando que compara a assinatura digital recebida à chave pública que consta no *script*: decripta-se c; se o resultado corresponde a m (formado independentemente), aceita-se que Ada é a dona daquelas bitcoins.

3.3 Nós e a verificação de transações

Existem dois tipos de nós na rede Bitcoin — os completos e os simples[xiii] — e é preciso distingui-los. Os nós completos são computadores ligados à rede Bitcoin que mantêm uma cópia completa da *block chain*, atualizando-a sempre e verificando independentemente todas as transações e blocos recebidos. Os nós completos costumam ser servidores rodando uma versão do cliente-referência Bitcoin [3]. A maior parte dos nós da rede, no entanto, é composta por nós simples. Os nós simples somente armazenam as transações que dizem respeito ao dono desse nó,

[xiii] Há um *trade-off* na escolha entre um nó simples e um nó completo. Os nós completos, apesar de mais seguros, possuem custos de manutenção relativamente altos. Mais informações estão disponíveis na página do Projeto Bitcoin (https://web.archive.org/web/20161109222317/https://bitcoin.org/en/full-node, acesso em 2016-11-21). Esses requerimentos impedem que a maioria das pessoas usem nós completos; geralmente, só usuários avançados, desenvolvedores e empresas mantém nós completos.

Tabela 1: O processo de liberação dos fundos de uma transação-saída, tipo *Pay-to-Public-Key*

Stack	Script	Description
empty	<sig> <pubKey> OP_CHECKSIG	stack is empty, ScriptSig and ScriptPubKey are lined up to be processed
<sig>	<pubKey> OP_CHECKSIG	<sig> is copied to the stack
<sig> <pubKey>	OP_CHECKSIG	<pubKey> is copied to the stack
True/False	empty	OP_CHECKSIG is executed, signature is checked

Fonte: Adaptado de Bischoff [7].

dependendo dos nós completos para verificar a correção de todas as transações da rede. Os nós simples costumam ser aplicativos de carteira Bitcoin em celulares.

A verificação da propriedade de uma transação não é a única feita por um nó da rede Bitcoin. Uma lista completa e atualizada das verificações pode ser obtida de uma consulta ao código-fonte do cliente-referência Bitcoin [3].[xiv]

Quanda Ada emite sua transação para a rede, cada nó verifica independentemente a transação e a transmite aos seus pares. As principais etapas feitas por um nó ao receber uma transação são as seguintes:

- Checar a correção sintática.

  Como visto na Tabela 2, uma transação Bitcoin é composta por vários componentes de alto nível. Além de uma ordem específica, eles devem seguir um padrão. A variável `lock_time`, que estabelece um tempo mínimo para adição da transação à *block chain*, por exemplo, é expressa em tempo Unix, i.e., em segundos após o horário arbitrado de 1970-01-01 00:00 UTC. Não se pode

[xiv] Uma lista mais facilmente acessível pode ser encontrada na BitcoinWiki https://web.archive.org/web/20161113233843/https://en.bitcoin.it/wiki/Protocol_rules#.22tx.22_messages.

Tabela 2: Componentes de alto nível de uma transação Bitcoin

Bytes	**Name**	**Data Type**	**Description**
4	version	uint32_t	Transaction version number, currently version 1.
Varies	tx_in count	compactSize uint	Number of inputs in this transaction.
Varies	tx_in	txIn	Transaction inputs
Varies	tx_out count	compactSize uint	Number of outputs in this transaction
Varies	tx_out	txOut	Transaction outputs
4	lock_time	uint32_t	A time (Unix epoch time) or block number

Fonte: Adaptado de Bitcoin Developer Documentation [6]

expressar essa variável de nenhuma outra forma, sob pena de ver a transação rejeitada pelos nós da rede.

- Verificar se há pelo menos uma transação-entrada e uma transação-saída.
- Rejeitar transações que não sejam padrão.

  Como mencionado na Subseção 3.2 e na nota de rodapé xi, há vários tipos de transação possíveis, mas os nós que fazem parte da rede Bitcoin optam por só aceitar alguns tipos de transação cuja segurança é reconhecida.

- Rejeitar se essa transação já existir na *pool* ou na *block chain*.
- Para cada transação-entrada, se a transação-saída referenciada por ela já existir em alguma outra transação na *pool*, rejeitar a transação.
- Para cada transação-entrada, procurar na *block chain* e na *pool* de transações pela transação-saída a que ela se refere. Se a saída inexistir, essa transação é orfã. Por conta da descentralidade da rede Bitcoin, uma transação α que é criada e transmitida por um nó ao meio-dia pode chegar a um outro nó depois de uma transação β criada e transmitida pelo primeiro nó um minuto depois. Se a transação β tem como entrada uma saída de α, o segundo nó não saberá identificar essa saída até que a transação α chegue a ele. A transação-orfã β fica na *pool* esperando a chegada da transação-mãe α cuja saída é referenciada por uma das entradas da transação-orfã β.

- Para cada transação-entrada, se a transação-saída a que ela se refere já houver sido gasta, rejeitar essa transação.
- Rejeitar se não valer a Equação 1.
- Verificar o script de liberação de fundos de cada transação-entrada; rejeitar a transação se algum deles não for satisfeito.
- Adicionar a transação à *pool*.
- Transmitir a transação para os pares.

  Como se pode ver, cada nó filtra as transações ruins das transações que recebe antes de repassá-las. Por isso, é difícil congestionar a rede produzindo uma série de transações inválidas.
- Para cada transação-orfã que usar essa transação como entrada, aplicar todos esses passos (inclusive este) recursivamente nessa transação.

  Essa etapa resgata a transação-orfã β da *pool* e a valida seguindo esses passos.

Além de verificar cada transação recebida, os nós da rede Bitcoin também armazenam todas as transações validadas pela rede de maneira independente, em uma estrutura chamada de *block chain*.

4 *Block Chain*

A *block chain* é exatamente o que o nome diz: uma série de blocos em cadeia. Em cada bloco há um agrupamento de transações que fazem referência a saídas presentes em blocos passados. Esse agrupamento, uma vez completo, é transmitido para a rede, assim como são transmitidas as transações. Cada nó completo recebe o bloco, o valida, verifica as transações nele presentes e o repassa a seus pares.

Na Seção 3, evidenciou-se que as bitcoins não são análogas digitais de moedas. Cada transação Bitcoin faz referência a pelo menos uma transação anterior, que é usada como entrada. Sem os registros contidos na *block chain* — que caracterizam um único histórico ordenado das transações — há mais de uma possibilidade de histórico ordenado. Isso ocorre porque não há forma confiável de contar o tempo universalmente em uma rede distribuída.

Como a comunicação entre os nós não é instantânea (a passagem de um nó para outro demora algum tempo e pode haver problemas de conexão), qualquer nó da cadeia pode arbitrar um horário para a mensagem que ele cria ou repassa. Se houver incentivos econômicos para a honestidade, como no caso do uso do tempo universal coordenado (UTC) no planejamento de voos em um aeroporto, não há problema. Em Bitcoin, entretanto, há um ganho a ser auferido ao mentir sobre o momento em que se fez as transações. É por conta disso que o protocolo Bitcoin ignora o *timestamp* das transações ao decidir sobre o seu ordenamento.

Um exemplo esclarece esse problema. Voltando ao exemplo da Seção 2, caso Ada tivesse intenções maliciosas contra Babbage, ela poderia criar duas transações: uma em que pagasse 1 BTC a Babbage e uma em que pagasse esses 1 BTC a si mesma. A primeira transação ela transmitiria para Babbage e a segunda, ao restante da rede. Babbage receberia a transação e a verificaria com seu nó completo. Satisfeito, ele postaria o livro nos correios para ser enviado a Ada. O nó completo de Babbage, no entanto, repassa a transação recebida de Ada para os outros nós da rede. Esses nós haviam recebido anteriormente a transação em que Ada pagava 1 BTC a si mesma. Ambas as transações usam como entrada a mesma transação-saída, aquela em que Morgan paga 2.5 BTC a Ada. Além disso, essas operações possuem a assinatura

digital de Ada e portanto são legítimas — embora sejam uma tentativa de gasto duplo. Assim, fica claro que depender do *timestamp* de uma transação para sua aceitação é insuficiente, uma vez que o tempo em uma rede distribuída é relativo.

Já que o tempo é uma medida relativa em uma rede distribuída, existem vários possíveis ordenamentos para as transações feitas em Bitcoin. É preciso que todos os nós[xv] concordem com uma única versão dos fatos — deve haver consenso. Caso contrário, começariam a aparecer divergências nas versões de cada nó e a rede deixaria de funcionar. No exemplo dado, parte da rede consideraria Ada como a detentora de 2.5 BTC e a outra parte consideraria Ada a dona de 1.5 BTC e Babbage o dono de 1 BTC. Se Babbage tentasse enviar esse 1 BTC para outra pessoa, parte da rede consideraria a transação ilegítima, i.e., sem valor. Dessa forma, é como se a moeda Bitcoin houvesse se dividido em duas. Na economia tradicional, essa situação seria análoga a uma emissão de novas notas de real (em que as antigas deixassem de valer), mas que parte da população brasileira não considerasse as novas notas legítimas — continuando o uso das antigas.

Na economia tradicional, uma solução possível é a manutenção de um balanço. A gestão de balanços costuma ser centralizada, o que não seria possível no protocolo bitcoin. Para que se veja o por quê disso, faz-se uma nova analogia. Suponha uma planilha digital aberta em uma rede compartilhada de escritório. Quando um outro usuário tenta editar a planilha, ele é avisado de que a planilha já está aberta em um computador e que, portanto, ele não pode editá-la para evitar divergências. O segundo usuário tem, no entanto, a possibilidade de fazer uma cópia e fazer edições dessa cópia. Ao final, ele terá de unir (*merge*) as suas edições com aquelas feitas pelo primeiro usuário. Caso as edições não sejam conflitantes, a junção pode ser feita.

Agora imagine milhares de usuários editando a mesma planilha em uma mesma rede, tendo de concordar ao final em uma planilha única. O consenso sobre essa versão final não pode ser decidido de forma manual como no exemplo acima. Há *software* de bases de dados mais sofisticados que gerenciam melhor os conflitos de edição. Apesar disso, se permanecerem discordâncias é necessária a intervenção de uma autoridade central ou até mesmo o estabelecimento de um sistema de votação.

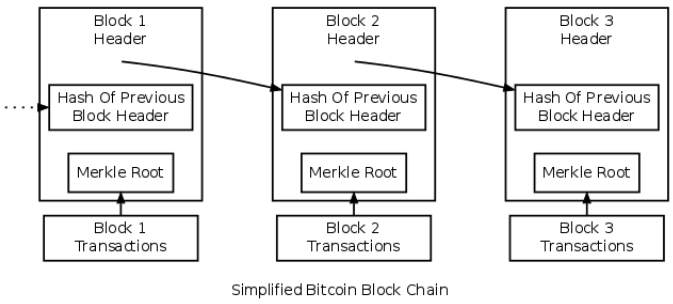
[xv] Ou ao menos uma maioria significativa deles.

No caso do protocolo Bitcoin, não há autoridade central, de modo que cada nó da rede teria sua cópia do histórico de operações (balanço), mas essas cópias seriam diferentes e, por vezes, incompatíveis com as outras. A solução para esse problema está na *block chain*, que é um balanço distribuído. O encadeamento dos blocos fornece um ordenamento às transações que independe do horário em que se alegou que elas foram feitas. Dessa forma, é possível escolher entre duas transações efetuadas que utilizem as mesmas entradas, de modo que aquela avaliada como tentativa de gasto duplo seja rejeitada por todos os membros da rede.

4.1 O funcionamento da *block chain*

A inserção dos blocos na *block chain* é simples. O cabeçalho de um bloco contém dois componentes fundamentais: um *digest* do cabeçalho do bloco anterior e um *merkle root*, que é um *hash* de todas as suas transações organizadas em uma estrutura chamada de *merkle tree* (ver Figura 13). É justamente o *hash* do bloco anterior que estabelece o encadeamento: cada um dos blocos traz uma referência do bloco anterior até o bloco original, o bloco gênesis.[xvi] Ao invés de referenciá-los pelos seus números na ordenação, essa referência é feita pelo *hash* porque, como visto na Subsubseção 3.1.2, o *hash* do bloco funciona como um resumo de seu conteúdo. Assim, caso o conteúdo de um bloco seja alterado por um nó desonesto, o *digest* desse bloco muda e o ordenamento se quebra — o que efetivamente o protege da alteração maliciosa de transações anteriores.

Figura 10: *Block chain*: encadeamento de blocos de transações



Fonte: Harding, Bitcoin Developer Documentation [6].

[xvi] Disponível, entre outros lugares, em https://blockchain.info/block/000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f. Acesso em 2016-11-22.

5 O processo de mineração

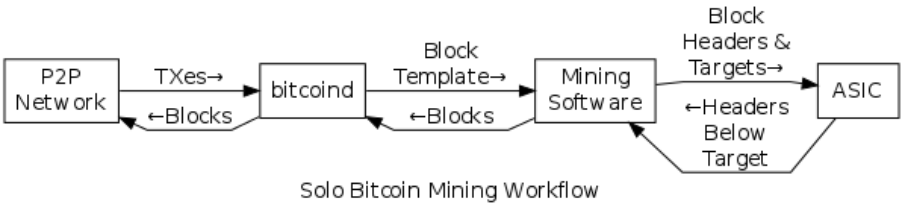
"If you view mining only as the process by which coins are created, you are mistaking the means (incentives) as a goal of the process. Mining is the main process of the decentralized clearinghouse, by which transactions are validated and cleared. Mining secures the bitcoin system and enables the emergence of network-wide consensus without a central authority."

— Andreas Antonopoulos [5]

A *block chain* promove um ordenamento discreto das transações — em oposição a um ordenamento contínuo no tempo.[xvii] Entretanto, esse mecanismo ainda não é suficiente para explicar qual ordenamento das transações é escolhido entre os vários encadeamentos possíveis. Como a rede faz essa escolha? Para responder a essa pergunta, é preciso entender qual o papel do minerador na criação de um bloco.

Um minerador é, antes de tudo, um nó completo da rede Bitcoin. Ao receber a transação de um nó, ele verifica se ela é válida. Em caso afirmativo, ele adiciona essa transação ao bloco que está tentando formar.

Figura 11: O *workflow* de um minerador comum



Fonte: Harding, Bitcoin Developer Documentation [6].

O *workflow* de um minerador comum está explícito na Figura 11. O minerador recebe transações continuamente da rede *peer-to-peer* (P2P), que é o protocolo Bitcoin. Essas transações são verificadas como mostrado na Subseção 3.3; as operações que são validadas são adicionadas a uma *pool* com a qual se pode formar o bloco. Note que cada minerador possui uma *pool* de transações semelhante, porém não necessariamente igual — uma vez que cada minerador é um nó distinto da rede, que recebe transações de diferentes outros nós e em tempos diferentes.

[xvii] Como demonstrado acima, esse ordenamento no tempo é impossível em uma rede *peer-to-peer*.

As transações que formarão um determinado bloco-candidato são escolhidas arbitrariamente pelo minerador, desde que sejam válidas.[xviii] Mineradores costumam priorizar transações de acordo com a relação $\frac{tx\_fees}{tx\_size}$, em que tx_size é expressa em bytes.

Escolhidas as transações da *pool* de cada minerador, um bloco candidato é formado. Assim como para as transações, esse bloco precisa seguir algumas regras de formação. Todo bloco (candidato ou não) é composto por três partes principais: um cabeçalho (mencionado na Seção 4), um contador de transações e todas as transações contidas no bloco.

5.1 A transação *coinbase*

A primeira transação de um bloco é especial. Essa transação é chamada de *coinbase* e sua função é dupla: remunerar o minerador do bloco e emitir bitcoins. Cada bloco só pode apresentar uma transação *coinbase* e essa transação é constituída por duas partes: o subsídio e as taxas de transação.

O subsídio é a forma como as bitcoins são emitidas. Conforme estabelecido por Nakamoto [2, p. 4] e presente até hoje no protocolo Bitcoin [3], o subsídio por bloco começou em 50 BTC quando o primeiro bloco foi minerado pelo próprio Nakamoto e é reduzido pela metade a cada 210.000 blocos.

As taxas de transação são, como discutido na Seção 3, recompensas aos mineradores incluídas implicitamente em transações. À medida que o subsídio por bloco cai, a importância das taxas de transação tende a crescer.

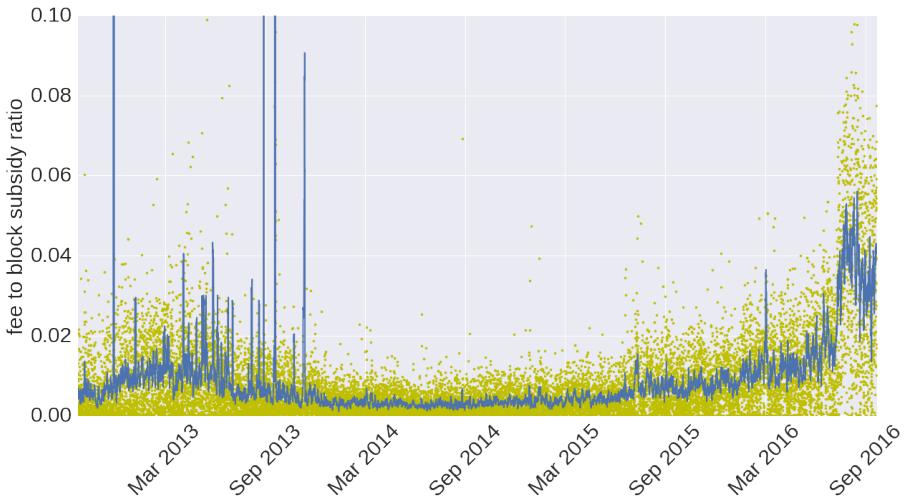
Unidas na transação *coinbase*, as taxas de transação e o subsídio por bloco possuem uma outra característica especial. A *coinbase* é o único tipo de transação que não possui entradas, por ser uma forma de emissão monetária.[xix]

As transações *coinbase* foram pensadas por Nakamoto como um incentivo

[xviii] De fato, há até mesmo blocos 'vazios', formados somente pela única transação obrigatória de cada bloco.

[xix] O subsídio de bloco é uma emissão monetária pura. Já as taxas de transação, como mencionado em Seção 3, são calculadas implicitamente, não estando em nenhuma transação-saída. Por conta disso, não estão disponíveis para serem transações-entrada de uma transação *coinbase*.

Figura 12: Razão entre as taxas de transação e o subsídio por bloco (com efeito suavização), 2013-2016



Fonte: Elaboração própria, a partir de API Blockchain.info.

econômico aos mineradores para a boa manutenção da *block chain* e, por conseguinte, do valor da Bitcoin enquanto moeda.

5.2 O cabeçalho de um bloco

Na Seção 4, viu-se que o encadeamento dos blocos é feito via o *hash* do cabeçalho do bloco anterior. A Tabela 3 mostra o cabeçalho do bloco 358230 da *block chain* como exemplo:

5.3 Dificuldade

Cada bloco representa um ordenamento possível de transações. Entre diversos blocos-candidatos, é a dificuldade de se obter um bloco aceitável que determina qual será escolhido ou não. Essa dificuldade é ajustada por todos os nós da rede de duas em duas semanas, a fim de manter o padrão estabelecido por Nakamoto [2], o de que um bloco deve ser minerado a cada 10 minutos, em média. Se isso de fato ocorrer, a cada duas semanas serão minerados 2016 blocos.

Blocos são comumente referidos pela sua altura (*height*) na *block chain*. O bloco 0, também chamado de bloco-gênesis, foi minerado por Nakamoto em 2009-01-03.

Tabela 3: Cabeçalho do bloco 358230

Campo	Propósito	Atualizado quando...	Exemplo
Version	Versão do protocolo Bitcoin usada para gerar bloco	o minerador atualiza seu software	2
hashPrevBlock	Hash do bloco anterior	um novo bloco é recebido	000000000000000 00a3ed9a4e254075 18aa854f09fa1981 adaae9455a91d1966
hashMerkleRoot	Hash do conjunto de transações do bloco	uma nova transação é aceita	9b7d5896398581a7 ff26be4b3684ddd9 5a7c1dc1aab1df37 cbb2127379ae8584
Time	hora atual (em segundos desde 1970-01-01 T00:00 UTC)	se passam alguns segundos	1432723472
nBits	hash-alvo em formato compacto	a dificuldade é alterada	404129525
Nonce	número (começa em 0)	hash é tentado (incrementa o nonce)	226994584

Fonte: Adaptado de BitcoinWiki. Acesso em 2016-09-10.

Cada agrupamento de transações subsequente tem sua altura definida pelo número de blocos entre ele e o bloco gênesis (inclusive).

Dessa forma, ao término de um período de duas semanas, cada nó observa sua cópia da *block chain* e calcula o tempo que se passou entre a mineração do bloco atual de altura n e o bloco 2016 posições atrás do atual (altura $n - 2016$).[xx]. É calculada, em seguida, a razão entre o tempo que seria esperado para a mineração de 2016 blocos — 2 semanas, ou 1.209.600 segundos — e o intervalo de tempo observado entre a mineração dos últimos 2016 blocos.

$$\frac{new\_difficulty}{old\_difficulty} = \frac{1209600}{time\_block_n - time\_block_{n-2016}}$$

A dificuldade para que um bloco seja aceito é então reajustada proporcionalmente ao resultado, com um limite superior de 300% e um inferior de 75%. Isto é, se os últimos 2016 blocos demoraram mais do que duas semanas para serem minerados, o intervalo de tempo médio entre os blocos foi maior do que 10 minutos; logo, é preciso diminuir a dificuldade para aceitação de um deles para manter o patamar arbitrário de um bloco a cada 10 minutos. Caso a mineração dos últimos 2016 blocos tenha demorado menos tempo do que duas semanas, a dificuldade para a aceitação de um bloco precisa aumentar para que se possa manter o patamar de um bloco a cada dez minutos.

5.4 *Proof-of-work*

O protocolo Bitcoin, lembre-se, é um *software* que implementa as regras da moeda Bitcoin, além de seu funcionamento. No protocolo, é definido que um bloco válido precisa, além de possuir somente transações válidas e outras especificações, atender a um último requisito: o *hash* de seu cabeçalho tem de ser menor do que um certo valor. Esse valor é regulado segundo a dificuldade — quanto maior a dificuldade, menor tem de ser o *hash* do cabeçalho. Esse regra está inserida no cabeçalho de cada bloco minerado (ver Tabela 3) por meio da variável `nBits`, que

[xx] Curiosamente, por um erro de programação do código do cliente-referência Bitcoin [3], esse intervalo é calculado com base no bloco minerado 2015 posições atrás do bloco corrente e não no de altura $n - 2016$. Não se leva esse erro em consideração neste trabalho [6]

registra o *hash*-alvo da rede naquele momento de forma truncada.

Na Subsubseção 3.1.2, mostrou-se que as funções *hash* tomam um *input* qualquer e o transformam um *digest* de tamanho fixo. Esse *digest* possui, no caso do protocolo Bitcoin, 64 caracteres, cada caractere podendo ser uma letra de `a-f` ou um número de `0-9`, formando uma base hexadecimal. Como não é possível saber de antemão o *digest* de um *input* qualquer, os mineradores têm de alterar o cabeçalho de seus blocos diversas vezes até obter um *digest* adequado à regra.

Quando um bloco-candidato apresenta *hash* menor do que o *hash*-alvo, diz-se que aquele bloco demonstrou seu trabalho, i.e., que sua *proof-of-work* foi apresentada.

Como exemplo, vê-se no texto abaixo algumas tentativas de obtenção de um *digest* de um cabeçalho de bloco fictício que comece com um `0` (i.e., seu *hash* tem de ser menor do que o alvo `1000[...]0`), exemplo semelhante ao feito por Antonopoulos [5].

```
trying hash input: "cabeçalho de bloco-candidato número 0"
hash digest: 18b34598d215b1103f4cd2313b89a2258e2ee0c5803f5f59f15[...]
trying again...

trying hash input: "cabeçalho de bloco-candidato número 1"
hash digest: 57cc3a304f9e4eb246a10c82d08840738c126f87fd248630bbd[...]
trying again...

trying hash input: "cabeçalho de bloco-candidato número 2"
hash digest: 091519f78f862828d112bc9460ee53dfc324c1e6b7ef49f03e6[...]
found!
```

Para o primeiro caractere do *digest* há 16 valores possíveis (`0-9|a-f`). Por isso, espera-se obter um resultado adequado a essa regra-exemplo a cada 16 tentativas — assumindo que os *digests* têm resultado aleatório. Caso exigíssemos um número maior de `0`s à frente do *digest*, o número de tentativas esperadas para a obtenção de um *digest* dentro da regra cresceria exponencialmente.

Na prática do protocolo Bitcoin, o *hash*-alvo determina um valor muito baixo para o *digest* — o que faz com que o processo de mineração (a apresentação da

proof-of-work) exija muito mais de um trilhão de tentativas. No exemplo dado acima, iterou-se um número ao final do cabeçalho fictício a fim de mudar seu *hash*. No protocolo, itera-se também um número presente no cabeçalho, o `nonce` (ver Tabela 3).

5.5 Verificação dos blocos

Quando um minerador consegue um cabeçalho de bloco abaixo do *hash*-alvo da rede Bitcoin, diz-se que ele minerou um bloco. Em seguida, esse minerador anexa o cabeçalho encontrado ao bloco que ele montou e o transmite aos seus pares. Cada nó o recebe e segue algumas etapas para sua validação.

- Checar correção sintática.
- Rejeitar se duplicata de um bloco já recebido.
- A lista de transações não pode estar vazia, primeira transação do bloco deve ser a transação *coinbase* e só pode haver uma *coinbase*.
- *hash* do cabeçalho do bloco deve atender ao *hash*-alvo da rede.
- Verificar todas as transações como na Subseção 3.3. À exceção da *coinbase*, verificar se as saídas a que se referem estão no galho principal da *block chain*.
- Para cada entrada de uma transação que fizer referência a uma saída que seja de uma transação *coinbase*, rejeitar caso sua maturidade for menor ou igual a 100.

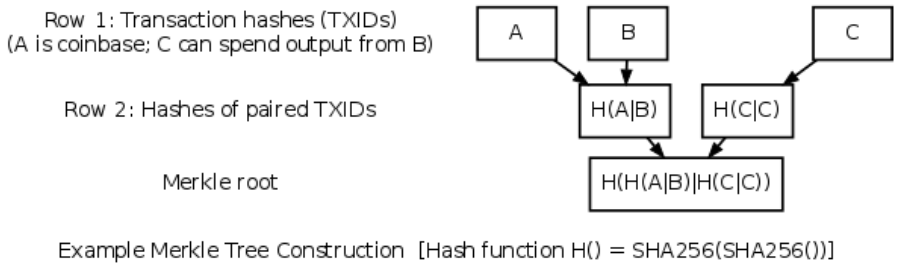
  A maturidade é definida como o número de blocos entre o atual e aquele em que a *coinbase* em questão foi gerada. Esse mecanismo impede a reversão de transações caso haja um *fork* temporário na rede (*forking* será explicado na Subseção 5.6).[xxi]
- Verificar o *merkle root*.

  Presente no cabeçalho de um bloco, o *merkle root* é um *hash* das transações do bloco. Esse *digest* é calculado a partir de uma estrutura de árvore (*merkle*

[xxi] Ver https://web.archive.org/web/20160329142324/http://bitcoin.stackexchange.com/questions/1991/what-is-the-block-maturation-time para mais detalhes.

tree) em que são organizadas as transações, como se vê na Figura 13.

Figura 13: Um exemplo de *merkle tree*



Fonte: Harding, Bitcoin Developer Documentation [6].

- Rejeitar se valor da transação *coinbase* for maior do que a *block reward*:

$$block\_reward \leq block\_subsidy + \sum_{i=1}^{N} tx\_fees_i \tag{4}$$

  em que N é o total de transações do bloco.

- Para cada transação do bloco, deletar a transação correspondente da *pool*.

- Repassar o bloco aos pares.

5.6 Forking

Uma vez visto que há um padrão dinâmico para a aceitação de blocos de transações, sabemos uma forma de distinguir quais blocos-candidatos válidos podem ser incluídos na *block chain* e quais não podem. Ainda assim, não se resolve o problema de como escolher entre dois blocos válidos e que apresentem a *proof-of-work*.

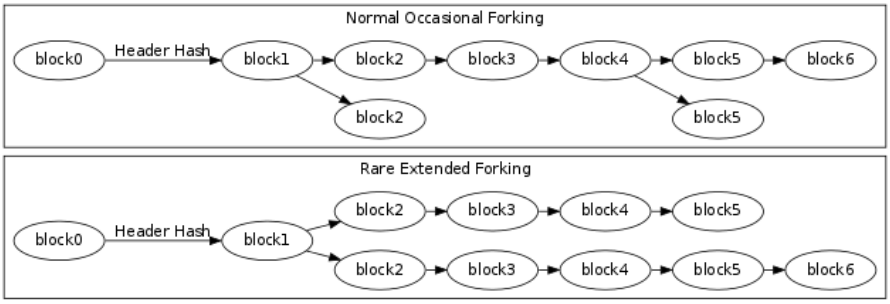
Um exemplo ajuda a esclarecer a solução para esse problema. Dois mineradores em competição, Fili e Kili, cujas *pools* de transações são semelhantes, montam seus blocos-candidatos. Eles tentam vários cabeçalhos válidos diferentes até obterem um *hash* que se adeque ao *hash*-alvo da rede Bitcoin. Caso Kili minere seu bloco pouco depois de Fili, não houve tempo para que o bloco de Fili chegasse até ele (caso contrário, ele abandonaria esse bloco-candidato). De modo análogo, nós mais

próximos de Fili na rede Bitcoin tendem a receber o bloco minerado por ele antes do bloco minerado por Kili.

Da mesma forma que a rede Bitcoin não confia nos horários em que nós afirmam terem feito suas transações (como mencionado na Seção 4), ela também não confia nos *timestamps* dos blocos minerados. Isso ocorre porque numa rede distribuída seria impossível garantir que alguém minerou um bloco no momento em que afirmou tê-lo feito; seria muito simples mudar o horário de mineração que consta no cabeçalho do bloco e não seria possível verificar a veracidade desse horário.

A decisão entre qual bloco é aceito depende, na verdade, da extensão da *block chain*. Quando dois blocos válidos são minerados em momentos quase iguais, ocorre uma *fork* na *block chain*.

Figura 14: Possibilidades de *fork* na *block chain*



Fonte: Harding, Bitcoin Developer Documentation [6].

Geralmente, cada nó da rede Bitcoin aceita o primeiro bloco válido que recebe. Caso um nó receba primeiro o bloco de Fili e em seguida o bloco de Kili, ele armazena o bloco de Kili num galho lateral (*side branch*) da *block chain*, como se vê na Figura 14. Caso o nó em questão seja também um minerador, ele montará um bloco-candidato que se encadeia no bloco de Fili, e não no de Kili. De maneira análoga, um minerador que tenha recebido o bloco de Kili antes do de Fili irá tentar minerar um bloco que se encadeia no bloco de Kili.

O próximo bloco a ser minerado tem papel fundamental. Caso o bloco seguinte a ser minerado se encadeie no bloco de Fili, o galho da *block chain* que contém o bloco de Fili se torna mais longo.[xxii] Ao receberem esse novo bloco, todos os nós que

[xxii] Tecnicamente, o que importa aqui não é o número de blocos do galho em si, mas sim a

haviam aceitado anteriormente o bloco de Kili mudam de posição: passam a aceitar o galho que contém o bloco de Fili como o galho *legítimo*. A probabilidade de um *fork* prolongado é baixa, mas caso fossem minerados dois blocos com curto intervalo de tempo entre eles, a situação se repetiria até que um galho ultrapassasse o outro e se tornasse o galho principal para todos os nós.

O encadeamento dos blocos não é trivial: graças às propriedades das funções hash (Subsubseção 3.1.2), os blocos ficam protegidos de edições subsequentes a sua mineração.

Imaginemos novamente que Ada tenha intenções maliciosas contra Babbage e queira reverter o pagamento que fez a ele. Depois que um bloco foi minerado com a transação em que ela paga Babbage, ele coloca o livro no correio — já que a transação foi confirmada por toda a rede Bitcoin. Ada pode tentar alterar o registro da *block chain* para reverter esse pagamento, mudando a saída da transação para que ela pague todos os 2.5 BTC a si mesma. Fazendo isso, Ada está tentando implementar um *fork* na *block chain*. Ao alterar a transação, ela altera o *hash* do bloco como um todo, uma vez que pequenas mudanças no *input* de uma função *hash* alteram substancialmente seu resultado. Ao alterar o *hash* do bloco — que está contido no cabeçalho do bloco —, o *hash* do cabeçalho deixa de atender ao *hash*-alvo estabelecido pela rede e, portanto, Ada precisa minerá-lo novamente, a fim de apresentar a *proof-of-work* desse bloco modificado. Enquanto Ada trabalha nesse bloco modificado, todos os mineradores da rede continuam tentando extender a *block chain* original. Caso o *hashing power* de Ada seja inferior ao da rede, é provável que enquanto Ada esteja procurando a *proof-of-work* do bloco modificado, algum outro minerador já tenha achado um novo bloco, que faz referência ao bloco original que contém a transação não-alterada.

O achado do novo minerador não impede que Ada continue tentando o *fork* da *block chain*. Apesar disso, o encadeamento de um novo bloco significa que Ada precisa agora alterar dois blocos e apresentar a *proof-of-work* deles, ao invés de fazer isso para um bloco só. Isso acontece porque o segundo bloco que foi minerado

dificuldade acumulada pelo galho como um todo. Assim, o galho que demonstrar mais trabalho (que apresenta *proofs-of-work* de maior dificuldade) é sempre o preferido, mesmo que ele possua menos blocos do que o galho concorrente.

incluía o *hash* do bloco original anterior, que Ada terá de substituir pelo *hash* do bloco modificado que fez. Ao fazer essa substituição, ela novamente altera o *hash* do cabeçalho (dessa vez, do bloco novo) e precisa procurar a *proof-of-work* desse novo bloco modificado.

Dessa forma, vê-se que o encadeamento dos blocos também encadeia o trabalho feito pelos mineradores na verificação e registro das transações. Assim, qualquer tentativa de modificação de um bloco significa ter de alterar todos os blocos subsequentes, apresentando suas respectivas *proofs-of-work*, em uma velocidade maior do que o resto da rede Bitcoin, que continua adicionando blocos ao galho original da *block chain*. Como mostrado por Nakamoto [2, pp. 6-8], se os nós honestos da rede controlam mais *hashing power* do que uma coalização de nós desonestos, a probabilidade de sucesso dessa tentativa diminui exponencialmente a cada bloco honesto que é adicionado ao galho original.

Esse cenário sequer deve acontecer. Por conta da remuneração aos mineradores feita pela transação *coinbase* mencionada na Subseção 5.1, Satoshi afirma:

> If a greedy attacker is able to assemble more CPU power than all the honest nodes, he would have to choose between using it to defraud people by stealing back his payments, or using it to generate new coins. He ought to find it more profitable to play by the rules, such rules that favour him with more new coins than everyone else combined, than to undermine the system and the validity of his own wealth [2, p. 4].

6 Modelos de mineração

Como visto na Seção 5, a mineração que mantém a Bitcoin em funcionamento é uma competição pela remuneração oferecida a cada bloco criado. Essa remuneração é grande o suficiente para atrair um número considerável de interessados, que investem milhões de dólares em *mining rigs*, placas de computador especializadas no cálculo de *hashes*, produzidas especialmente para a mineração de criptomoedas como a Bitcoin. Esses empreendimentos de mineração também consomem vastas quantidades de eletricidade, tanto para alimentar os computadores, quanto para dissipar o calor produzido pelas máquinas.[xxiii]

Nesta seção, serão discutidos dois modelos econômicos de mineração. O primeiro deles é um modelo abstrato, ao passo que o segundo emprega variáveis reconhecidas pelo protocolo Bitcoin. Nos dois modelos obtém-se a condição marginal de mineração. Adicionalmente, no segundo modelo, chega-se a um custo de produção da criptomoeda.

Em seguida, verifica-se as conclusões dos modelos analisando o comportamento da rede Bitcoin nos últimos anos. Em especial, analisa-se as tendências de receita por bloco e de taxa de *hash* da rede para ver se os mineradores reagem a variações na receita com mudanças em sua atividade.

6.1 Modelo de Kroll et al.

No modelo apresentado por Kroll et al. [8], analisa-se a situação de uma pessoa que avalia se deve investir recursos na mineração de bitcoin. Seu custo para a mineração de bitcoins é de C_i USD por segundo, ao passo que seu investimento lhe possibilita fazer $h_i = f(C_i)$ *hashes* por segundo. Espera-se fazer G tentativas (*guesses*) para atingir o *hash*-alvo e a mineração bem-sucedida de um bloco garante um retorno de V bitcoins, em dólares. Kroll et al. fazem uma hipótese adicional: a de que nenhum minerador dispõe de uma tecnologia ou de um desconto que não esteja disponível para os outros mineradores. Unida essa hipótese às características do protocolo Bitcoin, tem-se que a mineração é um mercado competitivo sem barreiras

[xxiii] Um exemplo desses empreendimentos pode ser visto em https://www.youtube.com/watch?v=K8kua5B5K3I, acesso em 2016-11-29.

à entrada.[xxiv]

Dessa forma, a mineradora escolherá minerar se seu ganho marginal esperado for igual ou superior ao seu custo marginal:

$$\frac{h_i V}{G} \geq C_i \tag{5}$$

No mundo, existem um total de N mineradores. G depende da dificuldade de mineração discutida na Subseção 5.3 de modo a manter a taxa de mineração mais ou menos constante em R novos blocos por segundo.

$$R = \sum_{i=1}^{N} \frac{P_i}{G} \tag{6}$$

Se $\overline{P} = \sum_{i=1}^{N} P_i$,

$$G = \frac{\overline{P}}{R} \tag{7}$$

Unindo as equações 5 e 7, obtém-se a decisão individual entre minerar ou não minerar levando em conta as escolhas dos outros mineradores:

$$\frac{P_i V}{\frac{\overline{P}}{R}} \geq C_i \tag{8}$$

Como todos os mineradores tomam a mesma decisão, com $\overline{C} = \sum_{i=1}^{N} C_i$, tem-se:

$$\sum_{i=1}^{N} \frac{P_i V}{\frac{\overline{P}}{R}} \geq \sum_{i=1}^{N} C_i$$

$$RV \geq \overline{C} \tag{9}$$

i.e., o minerador só minerará se a sua receita marginal esperada for maior do que seu custo marginal. Uma desigualdade estrita ($>$) incentivaria a entrada de novos mineradores, o que aumentaria o custo global de mineração ($\overline{C}' > \overline{C}$), levando a Equação 9 ao equilíbrio $RV = \overline{C}'$.

[xxiv] Nesse modelo, não há economias de escala, embora elas existam no mercado de mineração real.

6.2 Modelo de custo de produção

A dificuldade de mineração de um bloco é uma medida arbitrária, definida como:

$$D = \frac{T_{max}}{T} \tag{10}$$

em que D é a dificuldade corrente, T_{max} é o *hash*-alvo máximo — definido arbitrariamente no protocolo Bitcoin como $65535 \cdot 2^{208}$ —, e T é o *hash*-alvo corrente.

Como foi discutido na Subseção 5.4, a mineração de um bloco se dá quando o minerador encontra um valor para o `nonce` que torna o *hash* do cabeçalho do bloco-candidato um valor menor do que o *hash*-alvo corrente. Assim, a probabilidade de se minerar um bloco tentando um único *hash* é:

$$P(1) = \frac{T}{2^{256}} \tag{11}$$

em que o denominador é simplesmente o número de *hashes* possíveis,[xxv] e T é, novamente, o *hash*-alvo corrente. Combinando as equações 10 e 11, obtém-se :

$$P(1) = \frac{T_{max}}{2^{256}D} \tag{12}$$

Em 2016-11-22, $D = 281800917193.2$, de modo que com $T_{max} = 65535 \cdot 2^{208}$, a probabilidade $P(1)$ de minerar um bloco com uma única tentativa era de aproximadamente $8 \cdot 10^{-22}$. Alternativamente, o número esperado de *hashes* para se obter um *hash* abaixo do alvo é igual a $P(1)^{-1} \equiv G \approx 1210344191714312585216$.

O tempo esperado para a obtenção de um bloco válido ($E[T_{hash}]$) por um minerador que dispõe de um *hashing power* de h *hashes*/segundo (`H/s`) é, portanto:

$$E[T_{hash}] \equiv \frac{G}{h} = \frac{2^{256}D}{hT_{max}} \tag{13}$$

A receita esperada de um minerador em bitcoins por segundo é, por sua vez, igual ao *hashing power* (em `H/s`) multiplicado pelo *block reward* β (em bitcoins),

[xxv] Como visto na Subsubseção 3.1.2, o *output* de uma função *hash* tem tamanho fixo. No caso do protocolo bitcoin, esse *digest* é um número hexadecimal de 64 casas. Como cada dígito hexadecimal equivale a 4 bits (dígitos binários), temos 256 dígitos binários, ou 2^{256} possibilidades no sistema decimal costumeiro.

divididos pelo número esperado de *hashes* para a mineração de um bloco:

$$E\left[\frac{BTC}{s}\right] = \frac{h\beta}{G} = \frac{h\beta T_{max}}{2^{256}D} \tag{14}$$

Como mencionado no início desta seção, o processo de mineração envolve o gasto de tempo, eletricidade e *hardware* especializado, além de internet. Esse gasto se divide numa componente fixa afundada, ignorada pelo modelo e numa componente marginal, que é representada principalmente pela eletricidade.

Dessa forma, temos que o custo marginal da mineração é:

$$CMg_s = \frac{hp_e}{\epsilon} = \left(\frac{H}{s}\right)\left(\frac{J}{H}\right)\left(\frac{USD}{J}\right) \tag{15}$$

em que CMg_s é o custo marginal (por segundo), h é o *hashing power*, ϵ é a eficiência energética do *hashing power* (i.e., quantos *hashes* são feitos com cada Joule), e p_e é o custo da eletricidade local.

Se os mineradores forem agentes racionais, eles só continuarão a minerar enquanto a receita marginal deles for maior do que o custo marginal, ou seja, $E[\frac{BTC}{s}] \geq CMg_s$. Dessa forma, é possível obter um custo de produção P^* fazendo a razão entre o custo e a receita marginais, como colocado por Hayes [9].

$$P^* \equiv \frac{E\left[\frac{BTC}{s}\right]}{CMg_s}$$

$$P^* = \frac{\frac{hp_e}{\epsilon}}{\frac{h\beta T_{max}}{2^{256}D}}$$

$$P^* = \frac{2^{256}Dp_e}{\epsilon\beta T_{max}} \tag{16}$$

Assim, vê-se que esse custo de produção depende positivamente da dificuldade da mineração de blocos e do custo da eletricidade local, ao passo que depende negativamente da *block reward* e da eficiência do *hashing power*.

Hayes imagina esse custo de produção como um valor mínimo para o preço da

Bitcoin:

> The actual observed market price is determined by the supply and demand for bitcoin at any given moment, while the cost of production might set a lower bound in value around which miners will decide to produce or not. While this lower bound could represent an intrinsic value, the actual observed price may deviate from that expected value for long periods of time, or may never converge to it [9, p. 13].

A Equação 16 permite algumas conclusões sobre os mineradores. Como β e D são os mesmos para todos os mineradores, essas variáveis não nos dizem nada sobre o perfil deles. Apesar do mercado de *hardware* para a mineração ser livre, de modo que os mineradores têm acesso às mesmas máquinas para a execução de *hashes*, a eficiência ϵ também depende da dissipação de calor no ambiente. Assim, mineradores em locais mais frios ou em locais com melhor ventilação têm uma vantagem sobre os outros. p_e, por sua vez, também é diferente para cada minerador. É por isso que o maior determinante para a competitividade de um minerador é a sua localização, a qual tende a determinar tanto sua eficiência energética ϵ quanto seu preço local de eletricidade p_e.

6.3 Testando os modelos: o *second halvening*

Como mencionado na Seção 4, o subsídio por bloco minerado é divido pela metade a cada 210.000 blocos, em um evento chamado de *halvening*. Essa redução tem forte impacto na *block reward*, segundo a Equação 4 (ver Figura 12). Em 2016-07-09, ocorreu o último desses eventos, quando o subsídio por bloco caiu de 25 para 12.5 BTC. Como havia ocorrido no *halvening* anterior, houve muita discussão sobre as possíveis consequências desse evento no preço e na segurança da rede Bitcoin.[xxvi]

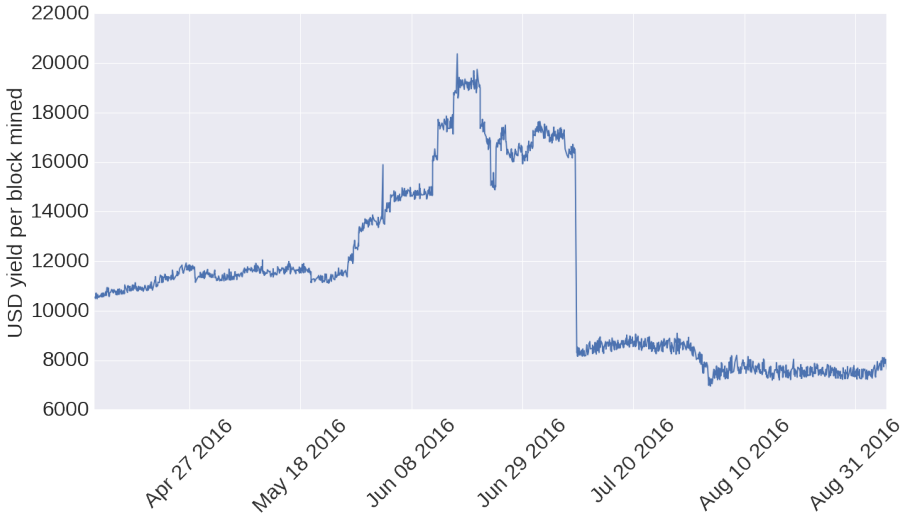
Quanto ao preço, esperava-se um aumento substancial devido à queda na taxa de emissão da moeda. Alguns comentaristas que compreendiam mais de expectativas afirmavam que o aumento de preço seria antecipado pelos agentes econômicos. Assim,

[xxvi] Um resumo dessas discussões está disponível em https://web.archive.org/web/20161211204604/https://bitcoinmagazine.com/articles/how-bitcoin-s-second-halving-came-and-went-and-not-much-happened-1468856719

não seria observada nenhuma valorização da Bitcoin após o *halvening*.

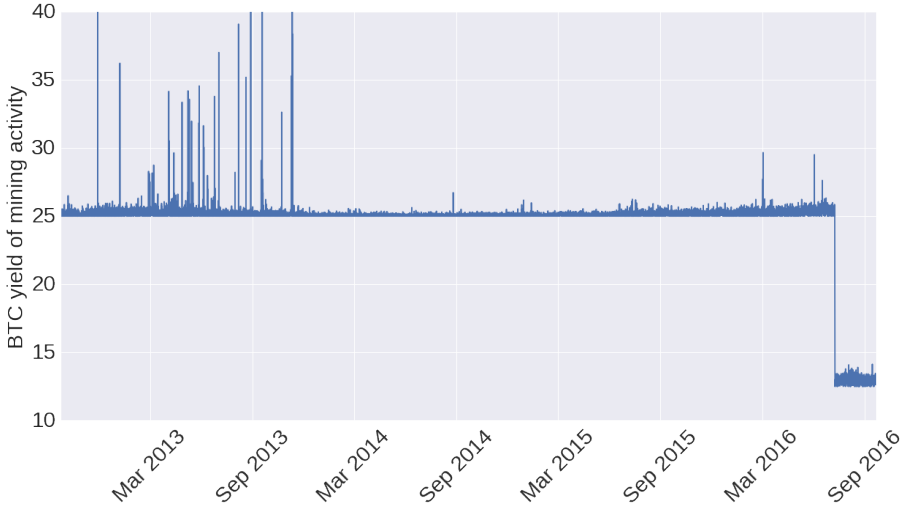
A outra preocupação envolvia a taxa de *hash* da rede. O raciocínio era simples: a redução no número de bitcoins que remunera um bloco minerado seria superior ao correspondente aumento de preço, reduzindo a receita dos mineradores.

Figura 15: Receita por bloco minerado em USD, 2016



Fonte: Elaboração própria, a partir de API Blockchain.info.

Figura 16: Receita por bloco minerado em BTC, 2013-2016

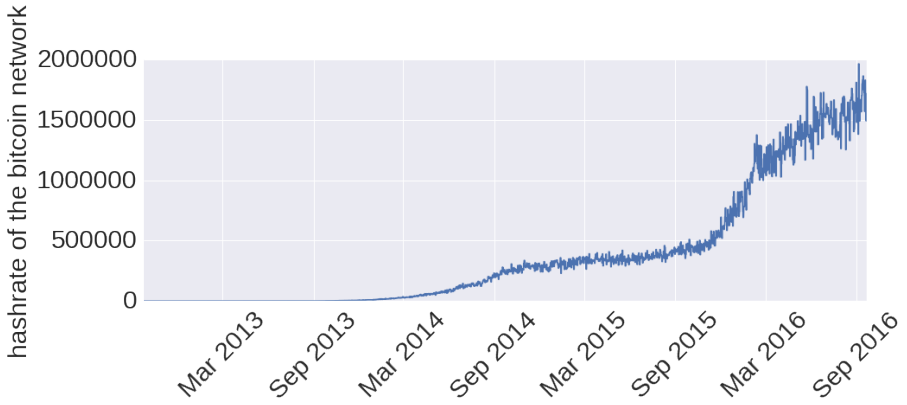


Fonte: Elaboração própria, a partir de API Blockchain.info.

Como se vê nas figuras 15 e 16, tanto a receita em USD quanto em BTC sofreu queda considerável após o *halvening*. Kroll et al. imaginam um cenário semelhante, em que há uma queda na receita por conta de uma depreciação da taxa de câmbio.

Nesse cenário, eles preveem a possibilidade do que chamam de "espiral da morte" [8, p. 8], em que uma queda na variável V do modelo (a receita marginal do minerador em USD) reduz o incentivo à mineração. Com uma queda na taxa de *hash*, a *block chain* se torna menos segura — uma vez que é preciso menos *hashing power* para alterar maliciosamente a *block chain*. Por conta da maior insegurança das transações Bitcoin, haveria nova depreciação da taxa de câmbio, que reduziria o incentivo à mineração, a qual, por sua vez, diminuiria novamente a taxa de *hash*, concluindo a espiral.

Figura 17: Taxa de *hash* da rede Bitcoin, em trilhões de *hashes* por segundo, 2013-2016



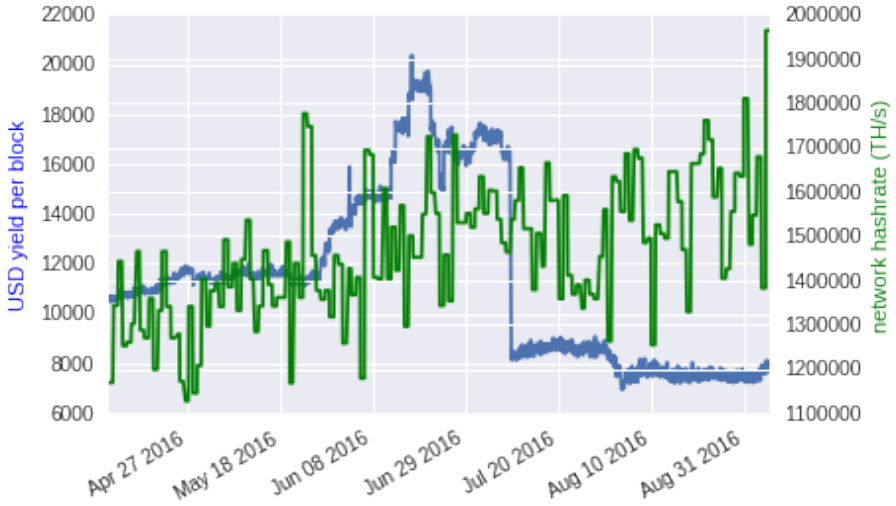
Fonte: Elaboração própria, a partir de API Blockchain.info.

Como se pode ver na Figura 17, apesar da queda na receita por bloco, não houve queda substancial na taxa de *hash* da rede bitcoin. Isso fica ainda mais claro na Figura 18, em que se mostra o período de março a setembro de 2016, i.e., o antes e o depois do *second halvening*.

Isso quer dizer que, apesar de uma redução significativa na receita marginal dos mineradores, sua atividade não se reduziu. Pelos dois modelos estudados nessa seção, a decisão de minerar dependia de um equilíbrio entre a receita e o custo marginal. Se há uma queda na receita enquanto o custo se mantém constante,[xxvii] os dois modelos preveem uma redução no número de mineradores e, portanto, uma queda na taxa de *hash* da rede.

[xxvii] Supõe-se que o custo marginal se manteve constante porque a taxa de *hash* da rede se manteve constante (o número de *hashes* feitos pelos mineradores não mudou) e porque não há razões para supor que houve uma mudança no custo da eletricidade deles ou na sua eficiência.

Figura 18: Taxa de *hash* da rede Bitcoin (em TH/s) *versus* a receita em USD por bloco, 2016



Fonte: Elaboração própria, a partir de API Blockchain.info.

Os próprios autores reconhecem as lacunas de seus modelos. Kroll et al. [8, p. 8] admitem que os equilíbrios (entre receita e custo) obtidos valem para decisões marginais, a cada instante. Na prática, entretanto, dois fatores tornam as decisões dos mineradores reais mais complexas. O primeiro fator é a existência de um custo fixo para a mineração de bitcoins. Como os mineradores precisam amortizar esses investimentos, eles têm um incentivo a continuar minerando enquanto sua receita marginal for igual ou maior do que o custo marginal, mesmo que eles estejam sofrendo prejuízo se levados em conta os custos fixos. O outro fator é a taxa de câmbio entre a Bitcoin e a moeda local. Já que os mineradores têm de pagar suas contas de eletricidade em moeda local e a taxa de câmbio Bitcoin é altamente volátil, essa volatilidade impede um equilíbrio constante entre receita e custo marginal, de modo que os mineradores fazem suas decisões levando em conta intervalos de tempo maiores do que um instante.

Há ainda um terceito fator: Bitcoin possui uma tendência de valorização. Isso acontece porque à medida que a criptomoeda se populariza, a sua demanda aumenta. Sua oferta, por outro lado, só varia a cada *halvening*, i.e., de maneira indiferente à procura. Consequentemente, a crescente demanda pressiona o valor da Bitcoin. Essa característica da criptomoeda assemelha a posse de bitcoins a um investimento. Assim, os mineradores podem escolher incorrer em perdas momentâneas, na espera

de uma valorização futura que contrabalanceie o prejuízo corrente. Como coloca Hayes, "individual decision makers may operate regardless of cost if they believe that there is enough speculative potential to the upside" [9, p. 13].

7 Conclusão

Neste trabalho, buscou-se explicar o protocolo Bitcoin em um nível intermediário. Foram abordados os principais mecanismos que garantem sua segurança e sua eficiência.

Na Seção 3, apresentou-se uma visão detalhada de como são feitas transações em bitcoin, inclusive a criptografia envolvida. Explicou-se, assim, como é possível autorizar operações financeiras de forma pública sem altos riscos de roubos e sem o uso de intermediários.

Nas seções 4 e 5 mostrou-se como se faz o registro público e distribuído de todas as transações. Em especial, mostrou-se como se previne tentativas de gasto duplo em uma moeda digital e como se atinge e se protege um consenso sobre o histórico de transações entre todos os membros da rede. Os incentivos econômicos que garantem o funcionamento do sistema também foram estudados; mais especificamente, o papel dos mineradores na manutenção dos registros das transações.

Na Seção 6, apresentou-se dois modelos de mineração em que se obtém uma condição marginal para a decisão de minerar, além de um 'custo de produção' para a Bitcoin. Esses resultados são averiguados contra as séries de tempo de variáveis como a taxa de câmbio entre USD e BTC e a taxa de atividade dos mineradores. Encontradas previsões imprecisas, procurou-se explicar o comportamento inesperado da criptomoeda à luz de fatores não abordados pelos modelos.

Este trabalho foi pensado como uma introdução ao estudo das criptomoedas e dos incentivos econômicos que as tornam possíveis. Ainda há muito o que ser explorado, pois o protocolo Bitcoin se desenvolve a cada dia. Além disso, há outras criptomoedas de grande interesse[xxviii]: a *Ethereum*, por exemplo, implementa contratos inteligentes que são executados por código, permitindo a substituição de empresas como AirBnB e Uber por código de programação; já a *Freicoin* implementa um mecanismo de *demurrage* (custo de deter moeda) proposto pelo economista Silvio Gesell. Assim, ficam demonstradas as possibilidades das criptomoedas e espera-se maior uso dessas tecnologias na economia do futuro.

[xxviii] Ver https://www.ethereum.org/ e http://freico.in/.

Referências

[1] SMITH, A. *An inquiry into the nature and causes of the wealth of nations.* 5th. ed. London: Methuen & Co., Ltd., 1904.

[2] SATOSHI, N. Bitcoin: A peer-to-peer electronic cash system. *https://bitcoin.org/bitcoin.pdf*, Utopia, 2008.

[3] Bitcoin core: reference implementation. *https://github.com/bitcoin/bitcoin*, Utopia, Acesso em 2016-11-20.

[4] MOLLIN, R. A. *An introduction to cryptography.* 2th. ed. Boca Raton: Chapman & Hall/CRC, 2007.

[5] ANTONOPOULOS, A. M. *Mastering bitcoin: unlocking digital cryptocurrencies.* Sebastopol, CA: O'Reilly Media, Inc., 2014.

[6] Bitcoin: Developer documentation. *https://bitcoin.org/en/developer-documentation*, Utopia, Acesso em 2016-11-20.

[7] BISCHOFF, L. Bitcoin: How it works and what's behind it. *http://www.mathcces.rwth-aachen.de/_media/3teaching/0classes/archiv/036_ces_semainr_bitcoins.pdf*, Utopia, 2015.

[8] KROLL, J. A.; DAVEY, I. C.; FELTEN, E. W. The economics of bitcoin mining, or bitcoin in the presence of adversaries. *Proceedings of WEIS*, Washington, D.C., v. 2013, 2013.

[9] HAYES, A. Cryptocurrency value formation: An empirical analysis leading to a cost of production model for valuing bitcoin. *Telematics and Informatics (forthcoming)*, New York, 2015.

A Apêndices

Esta obra possui dois apêndices: um exemplifica uma cifra simétrica e o outro exemplifica uma cifra assimétrica. Esses apêndices podem ser encontrados em https://github.com/odanoburu/misc/blob/master/tcc/caesar-cipher.ipynb e em https://github.com/odanoburu/misc/blob/master/tcc/RSA.ipynb.