Demonstrando o conceito de prova de trabalho da blockchain

Carlos Campello
Dec 18 · 7 min read

Ou *Porque a prova de trabalho deixa a blockchain segura e, ao mesmo tempo, energeticamente ineficiente*.

A experiência nos diz que a blockchain é segura: há anos no ar e nunca foi *hackeada*. Nenhuma carteira teve seus fundos comprometidos por problemas de implementação no protocolo da blockchain. O que ouvimos falar na imprensa e nos *sites* especializados, na maioria das vezes, são ataques bem sucedidos às *exchanges* de criptoativos (expondo as chaves privadas das carteiras custodiadas, etc.) ou a *smart contracts* mal implementados (como no fatídico caso do DAO do Ethereum).

Mas a experiência não é suficiente. E isso foi um dos motivos pelos quais busquei a resposta à seguinte pergunta: *por que a blockchain é tão segura?*

A resposta dessa pergunta me trouxe também a evidência de que a prova de existência torna a blockchain energeticamente ineficiente. E sobre isso que vamos falar nesse artigo.

Porém é necessário conhecermos um pouco dos conceitos antes de demonstrarmos.

Hashes

Encontrei um bom conceito do que são *hashes* na Wikipédia:

> *Uma função hash é um algoritmo que mapeia dados de comprimento variável para dados de comprimento fixo.*

Na prática funciona da seguinte maneira: não importa o que você entregue para esse algoritmo processar, seja um byte, um arquivo PDF, uma imagem JPG ou um arquivo de filme em 4K com vários gigabytes: sempre vai retornar uma sequência de caracteres do mesmo tamanho.

Sempre.

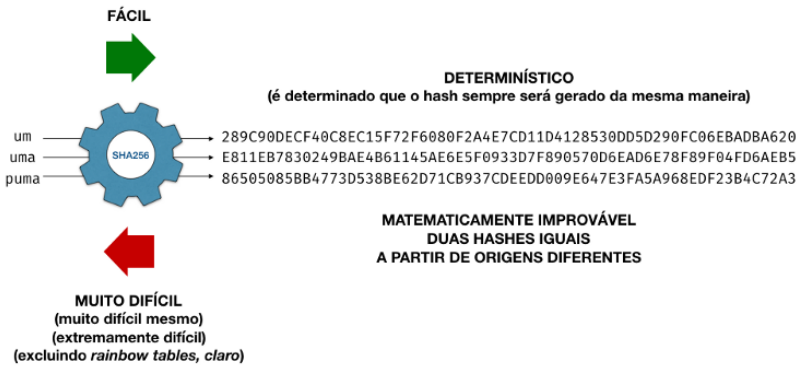
Além de ser uma operação fácil de realizar (rápida e que não exige esforço do computador), gera uma identificação única e previsível do conjunto de dados passado como único parâmetro dessa função. Para cada conjunto de dados, um hash. Para o mesmo conjunto de dados, o mesmo hash.

É uma função rápida e determinística.

Apesar de ser fácil ser gerado, é computacionalmente impossível ser revertido. Em outras palavras: não se tem como chegar à origem do hash se baseando nele mesmo. Não existe poder computacional no mundo que entregue isso.

Resumindo e revisando para fixar:

- Qualquer conjunto de dados submetido a uma função hash resulta em uma sequência de caracteres do mesmo tamanho.
- Esta sequência de caracteres é única para cada conjunto de dados submetido a este algoritmo e sempre que esse mesmo conjunto de dados for submetido ao mesmo algoritmo gerará o mesmo hash, o que implica dizer que é determinístico.
- É rápido fazer essa transformação do arquivo para o hash, mas é impossível saber o estado original de um hash (exceto se você já conhece a origem, claro).



Simplificação da função hash funcionando para três string distintas. Note a falta de semelhança entre os hashes gerados a partir de string parecidas.

Caso queira visualizar isso na prática, basta acessar o site abaixo.

https://passwordsgenerator.net/sha256-hash-generator

O site acima utiliza a mesma função hash a qual várias blockchains utilizam: SHA-256.

Para finalizar esse conceito, um exemplo.

> *Todos os seres humanos nascem livres e iguais em dignidade e em direitos. Dotados de razão e de consciência, devem agir uns para com os outros em espírito de fraternidade.*

O artigo primeiro da Declaração Universal dos Direitos Humanos, na sua forma textual, pode ser representado pelo seguinte hash:

> *BE730D98B5CA13C6AC1984BE71D05ABEE84507E2453B5B2054CD5E8A25D27055*

É fácil e rápido checar se esse hash é gerado a partir desse texto (se duvidar, pode testar no link lá de cima). Porém garanto que se eu mostrasse somente o hash, seria praticamente impossível descobrir sua origem (exceto por um artifício chamado *rainbow tables*, que seria um grande dicionário contendo uma relação *de-para* entre strings conhecidas e hashes gerados, muito utilizado por hackers que tentam descobrir senhas fracas quando conseguem uma base de dados contendo credenciais de acesso hasheadas e não criptogradas / salteadas).

Blocos e Transações

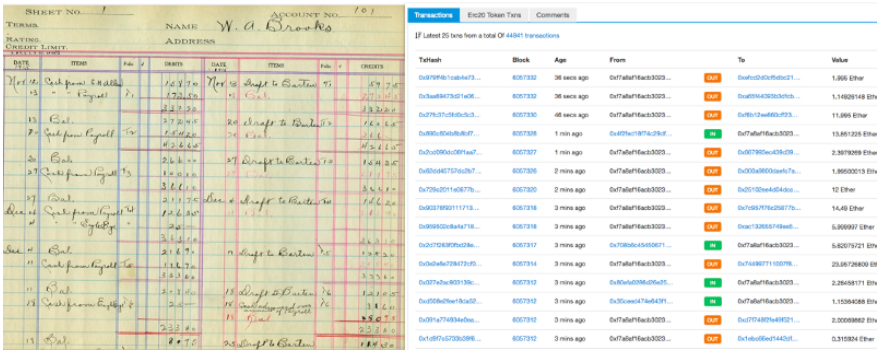
A blockchain é um livro contendo transações financeiras, isto é, um livro caixa (ou livro razão) distribuído entre os nós da rede.

Transação é a anotação de que uma quantidade de recursos saiu de uma origem para um destino, ou na linguagem da blockchain *saiu de uma carteira para outra carteira*.

Blocos nada mais são do que um conjunto de transações, além de um cabeçalho contendo algumas informações. Gosto de dizer que o bloco é a página com anotações no topo (cabeçalho) e as transações são as linhas dessa página, com cada linha contendo, no mínimo, as seguintes informações:

- um identificador único (id),
- carteira de origem,
- carteira de destino e

- valor transferido.



Semelhança entre a página de um livro caixa do início do Século XX e um bloco na blockchain do Ethereum.

As informações que estão no cabeçalho do bloco geralmente são:

- o número do bloco (feito o número da página, por exemplo),
- a data e a hora que o bloco foi criado (*minerado*),
- um hash gerado a partir das transações daquele bloco,
- um hash gerado a partir do cabeçalho do bloco anterior e
- o **nonce**.

O nonce é um número inteiro que prova que alguém trabalhou muito para poder encontrá-lo. É a chave, o resultado final, de um desafio computacional promovido pelo protocolo da blockchain.

E é sobre isso que vamos falar agora.

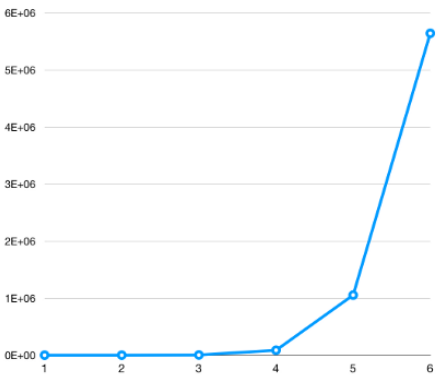
Em busca do nonce

O protocolo da blockchain entrega aos membros da rede, os nós, o seguinte desafio:

> *Concatene um inteiro ao final do hash gerado a partir das transações e gere um novo hash a partir dessa nova string. Caso o hash gerado a partir da junção do hash oriundo das transações mais esse número inteiro não tenha 20 zeros a sua frente, vá incrementando esse número inteiro até achar um novo hash com 20 zeros na frente.*

Se pode parecer um desafio fácil, tenha certeza que não é. Note que não temos a mínima ideia de que inteiro será esse então teremos testar um a um até encontrá-lo. Como num ataque de força-bruta.

Para demonstrar isso, fiz um simples script cujo desafio começava com achar um novo hash com apenas um zero na frente e fui dificultando até chegar até sete zeros. O resultado plotei no gráfico abaixo.



Zeros	Nonce	Tempo
1	14	0,0002
2	184	0,005
3	3.013	0,004
4	85.082	0,13
5	1.052.288	1,56
6	5.640.702	8,45
7	👎	160,21*

Para achar um hash com um zero na frente, foram necessárias apenas 14 tentativas, isto é, meu nonce foi 14. A dificuldade foi aumentando exponencialmente. Para achar um hash com apenas 5 zeros na frente, foram necessárias mais de 1 milhão de tentativas e 1,5 segundo. Para achar com 6 zeros na frente, mais de 5 milhões de tentativas e absurdos 8 segundos! Meu script estava programado para escapar com 100 milhões de tentativas, e foi isso que aconteceu quando tentou, em vão, achar um hash com 7 zeros na frente. E durou incríveis 160 segundos.

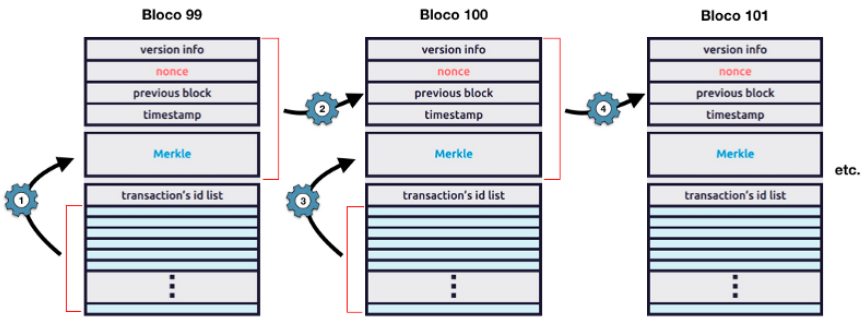
Dessa forma se atesta que existe um desafio computacional envolvido na obtenção do nonce, a chave para o *puzzle* do protocolo da blockchain.

E depois?

Após achar o fatídico nonce, o nó sortudo informa aos demais que conseguiu resolver o desafio primeiro que os demais e, para provar o seu trabalho, informa o número do nonce e o conjunto de transações as quais serão inclusas nesse bloco.

Os demais nós facilmente atestam o nonce e o bloco enfim é fechado (*minerado*), partindo para o próximo bloco. Antes disso são creditadas na carteira do nó sortudo algumas criptomoedas, justo visto que o vencedor minerou e através de seu esforço computacional (e sorte) conseguiu achar o nonce e fechar um bloco.

Lembrando que no cabeçalho do próximo bloco existe um hash gerado a partir do cabeçalho do anterior o que inclui tanto o nonce quanto o hash do cabeçalho anterior ao anterior.



Das transações é gerado um hash que é incluso no cabeçalho do próprio bloco junto com o nonce. O hash do cabeçalho, por sua vez, compõe o cabeçalho do bloco em seguida, formando uma cadeia de blocos: a blockchain.

Desta forma temos uma sequência de blocos encadeados (blockchain) garantido o estado de todos os blocos visto que caso se altere qualquer

mínima informação numa transação, será necessário gerar outro hash da transação e ir atrás de outro nonce. E se não for do bloco imediatamente anterior, o pretenso hacker teria que fazer isso em todos os blocos posteriores. Mesmo se ele tivesse poder computacional pra isso, o comportamento anômalo dele na rede já seria notado e teria sido desconectado pelos nós saudáveis, tudo isso graças a outra grande premissa da blockchain: o consenso.

Ineficiência e futuro

Esse enorme esforço para solucionar esse desafio computacional traz grandes consequências: o alto consumo energético. Fazendas mineradoras em busca desse nonce funcionam a todo vapor utilizando milhares de GPU. Clusters envolvendo um número absurdo de máquinas dividem o trabalho e os ganhos obtidos nessa mineração.

Estima-se que se consome entre 39 e 42 TWh (Terawatt-hora) de energia para manter a rede Bitcoin funcionando, isso é o equivalente ao consumo de países como Peru, Catar, Hungria e Nova Zelândia, por exemplo (fonte).

Talvez para que a segurança da blockchain fosse testada, realmente fosse necessária a implementação desse *feature* em 2008. Para uma tecnologia com aproximadamente dez anos, realmente é um feito notável nunca ter sido comprometida, principalmente por ser pública e por envolver bilhões de dólares em transações financeiras.

Outras blockchains estão migrando da prova de trabalho para outros tipos de mecanismos de consenso ou até mesmo sendo criadas utilizando outro paradigma de segurança, como prova de autoridade, *delegate proof of stake*, etc. Em 2019 a rede Ethereum vai implementar o *proof of stake* em busca de eficiência e agilidade.

Não há planos conhecidos para que o Bitcoin migre para outro tipo de mecanismo de consenso, abandonando a prova de trabalho, porém, em busca de dar velocidade às transações, a Lightning Network chega com o propósito de criar uma camada acima da blockchain para realizar transações sem depender diretamente dos custosos (aproximadamente) dez minutos de mineração de um bloco que o protocolo atual exige.

Disclaimer

Por fins *didáticos*, por assim dizer, algumas minúcias técnicas foram propositalmente suprimidos para tentar se dar alguma fluidez no texto

sem comprometer o entendimento geral.