Blockchain

Caio Regal

Eduardo Almeida

Luciano Tanaka

História

1991 - “How to timestamp a digital document” - Scott Stornetta e Stuart Haber

2008 - “Bitcoin Whitepaper” - Satoshi Nakamoto

2009 - Minerado o “Bloco Gênesis" e a Importância de um Sistema monetário descentralizado e confiável

2014 - “Ethereum Whitepaper” - Vitalik Buterin

2017 - NFTs

Conceitos

Sistema Monetário Descentralizado

Confiabilidade

Imutabilidade

Auditável

Proof of Work

exemplo prático: andersbrownworth.com/blockchain/

Bitcoin vs Ethereum

O Bitcoin foi feito com intuito de ser uma alternativa monetária, enquanto o Ethereum foi criado com a intenção de ser uma plataforma para facilitar o uso de contratos programados e imutáveis e uso de aplicações com a sua moeda como Contratos inteligentes, aplicações descentralizadas, NFTs, DeFi etc.

Velocidade dos blocos : 10 minutos para Bitcoin, 15 segundos para Ethereum.

Transações por segundo: 7 Bitcoin vs 30 Ethereum.

Nota sobre a criptografia

Os blockchains descentralizados como bitcoin e ethereum não possuem protocolos que envolvem o uso de criptografia em sua comunicação, plataforma, rede, database ou nós. Isso é para que todos possam verificar a veracidade de tudo que está ocorrendo para que haja o consenso.

Existe um avanço nas tecnologias de zero knowledge proofs e homomorphic encryption para que elas sejam usadas no futuro em tecnologias blockchain.

Como todas as mensagens são acessíveis e lidas por todos, há a necessidade de usar uma assinatura digital para autenticar as transações .

Hash

Hash ou função hash é um algoritmo que serve para converter um dado de qualquer tamanho em um tamanho fixo.

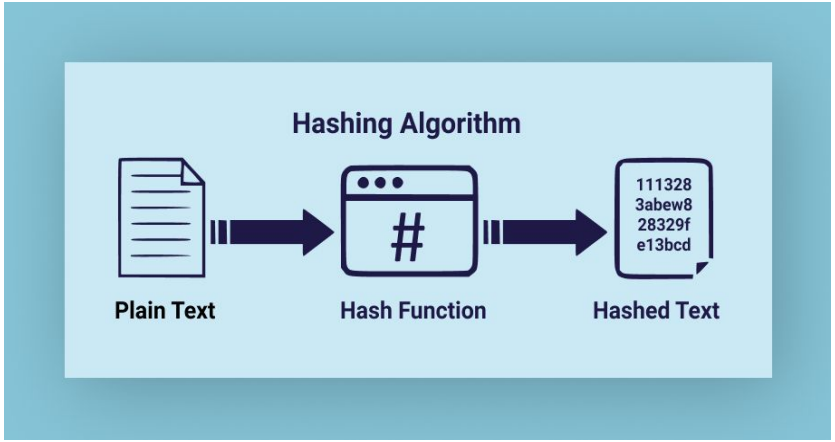
Um hash eficiente deve ser:

Determinístico : Dado uma certa entrada, seu resultado é sempre o mesmo.

Rápido Processamento : Não deve demorar muito tempo para ser calculado.

Resistir a colisões: Deve ser difícil de encontrar duas entradas que produzam a mesma hash.

Efeito Avalanche: Cada caractere deve produzir uma mudança enorme em cada hash.

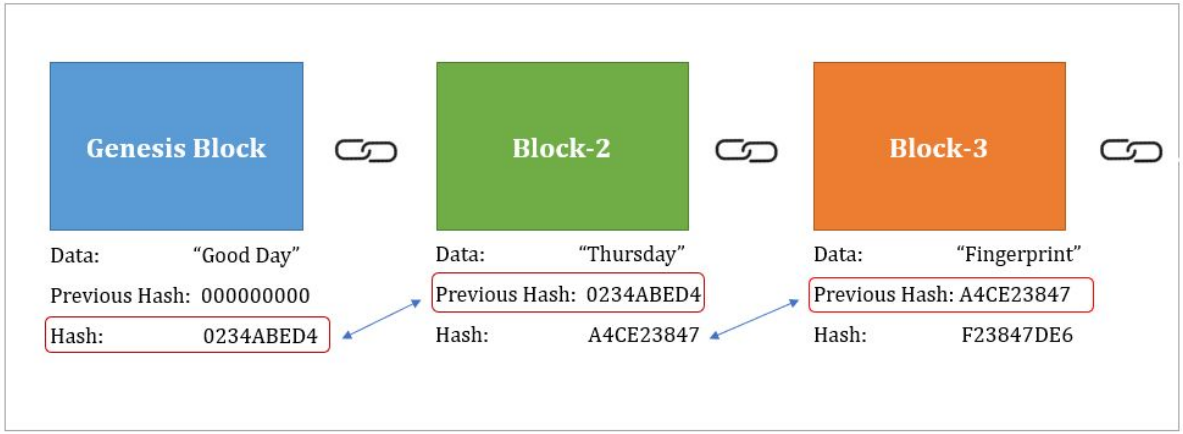


Hash

Hash é usado para implementar o mecanismo de Proof-of-Work e garantir a imutabilidade.

Dentro de cada bloco existe um hash de todas as transações desse bloco e a hash do bloco anterior, dessa forma uma vez adicionado a blockchain um bloco não pode ser alterado.

PoW tem sido popularizada pelo Bitcoin como uma base para o consenso na rede descentralizada sem permissão, onde os mineradores competem para anexar blocos e obter novas moedas, com a probabilidade de sucesso proporcional ao esforço computacional gasto.

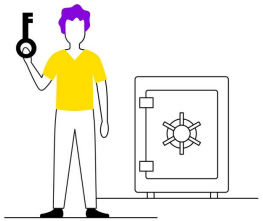


Mecanismos de consenso

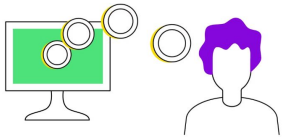
Consenso é a ideia de que pelo menos 51% dos nós da rede concordam com o próximo estado global da rede.

Em prova de trabalho, os moradores provam que possuem capital em risco gastando energia. Na prova de participação os validadores apostam capital na forma de Ether em contratos inteligentes no Ethereum. Após a aposta, ele atua como garantia da integridade do validador, sendo responsável por verificar e validar novos blocos propagados.

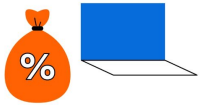
Proof of Stake



In a PoS network, the creator of a block is chosen according to certain criteria, such as wealth



Stakers receive transaction fees but there is no block reward



Cryptocurrencies mined based on PoS require less computational power and are thus more cost-effective

SHA-256 (Bitcoin)

Conjuntos de funções hash criptográficas, onde SHA-256 (Secure Hash Algorithm) se refere aos resumos (valores de hash) de 256 bits.

Parte da família SHA-2 e é baseada no conceito “Merkle–Damgård construction“ para criação de funções hash resistentes a colisões.

Projetada pela NSA (Agência de segurança nacional dos EUA) SHA-2 é publicada como o padrão oficial em crypto nos Estados Unidos.

Várias criptomoedas a usam, sendo a mais famosa o Bitcoin para verificar transações e calcular a prova de trabalho e participação.

Keccak-256(Ethereum)

Parte da família SHA-3 e é baseada no conceito “sponge construction” que baseia-se em uma permutação de comprimento fixo e uma regra de preenchimento.

Não é vulnerável ao “length extension attack” (ataque de extensão de comprimento).

Considerada mais segura que a família SHA-2

Length extension attack

Quando uma hash baseada em Merkle-Damgard é utilizada de forma errônea como MAC (message authentication code), autenticador de mensagem, H(senha||mensagem) e a mensagem e o tamanho da senha é sabido, o ataque de extensão de comprimento permite incluir informações extras ao final da mensagem que produzirá um valor de hash válido, mesmo sem saber o conteúdo da senha.

A Length Extension Attack Example

- Assuming the secret is “password”, the original data is “data”, then the SHA-1 signature is 6f5a7284246a7693c5f37f19f26609af84f56431
- Attackers attempt to append “attacking” to the original data.
- The new data is (you see %60 as the length of *(s||m)* = 12bytes = 96 (0x60) bits)

data%80%00%00%00%00%00%00%00%00%00%00%00%00%00%00%0
0%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%
00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%60
attacking

The new signature is
a2feef179114b40605307e0ca260a3e72a56017c

Length extension attack

As funções de hash vulneráveis funcionam pegando a mensagem de entrada e usando-a para transformar um estado interno. Depois que toda a entrada foi processada, o resumo (valor) de hash é gerado pela saída do estado interno da função.

É possível reconstruir o estado interno do resumo (valor) de hash, que pode ser usado para processar os novos dados.

Desta forma, pode-se estender a mensagem e computar o hash que é uma assinatura válida para a nova mensagem.

É usada para atacar algoritmos MD5, SHA-1 e principalmente em SHA-2.

Apesar da SHA-256 ser da família SHA-2, por ser uma versão truncada ela não é vulnerável a esse ataque.

Contas e endereços

Uma conta que mantém ativos em uma blockchain contém os atributos :

Private Key

Public key

Assinatura Digital

Private key

É o coração de todas as interações em uma blockchain, é usada para gerar os outros atributos únicos de cada conta.

Ela nunca será usada diretamente na rede.

Public Key

É a chave que vai ser usada para identificar a sua conta para outros usuários.

É o endereço da sua conta, porém nem todo endereço é uma public key(smart contracts).

É derivada da private key.

Assinatura digital

Acessa e controla os fundos da conta.

Serve para provar que a identidade da conta nas transações.

Gerada pela private key.

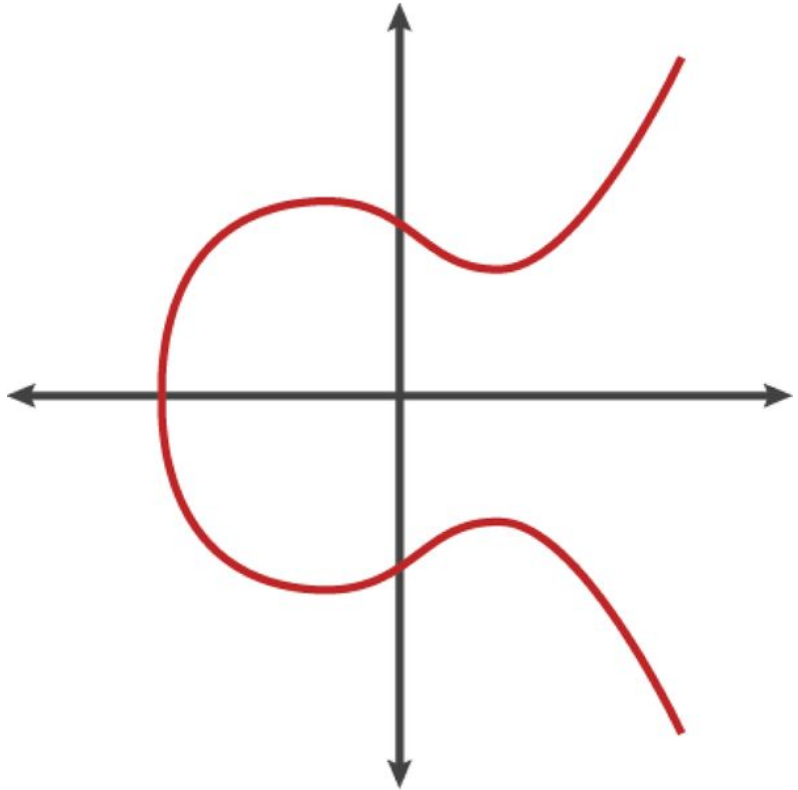
Transações

Os detalhes da transação são usados como mensagem, é usada uma criptografia de curva elíptica para combinar a mensagem com a private key gerando um código que só pode ser gerado utilizando a própria private key.

Esse código é chamado de assinatura digital e pela criptografia de curva elíptica é possível que qualquer um verifique a validade dessa transação.

Criptografia de curva elíptica

É um tipo de criptografia assimétrica baseada em um problema logaritmo discreto expresso por somar e multiplicar pontos de uma curva elíptica.



Contratos inteligentes

Solidity - Remix IDE.

O deploy é feito na EVM.

Garante uma transação sem intermediário.

Visto como um novo backend em aplicações de crypto.

Não possui dono ou private key.

Referências

https://www.oreilly.com/library/view/mastering-ethereum/9781491971932/ch04.html#:~:text=Ethereum's%20Cryptographic%20Hash%20Function%3A%20Keccak,Institute%20of%20Science%20and%20Technology.

https://en.bitcoin.it/wiki/Block_hashing_algorithm

https://www.investopedia.com/terms/t/target-hash.asp#:~:text=Bitcoin%20uses%20the%20SHA%2D256,amount%20of%20computer%20processing%20power.

https://river.com/learn/how-bitcoin-uses-cryptography/#:~:text=The%20Bitcoin%20network%20and%20database,interact%20over%20the%20Bitcoin%20network.

https://cryptobook.nakov.com/cryptographic-hash-functions/hash-functions-applications

http://www.umsl.edu/~siegelj/information_theory/projects/EllipticCurveEncyiption.pdf

https://tangany.com/blog/the-power-of-ethereum-virtual-machine-evm-and-the-tangany-custody-suite/#:~:text=The%20Ethereum%20Virtual%20Machine%20(EVM,then%20converted%20into%20EVM%20bytecode.

https://ethereum.org/en/developers/docs/consensus-mechanisms/

https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/

https://medium.com/0xcode/hashing-functions-in-solidity-using-keccak256-70779ea55bb0

https://pt.wikipedia.org/wiki/SHA-3