CEFET/RJ - CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA CELSO SUCKOW DA FONSECA

Projeto de Blockchain em C

Rafael Fulgoni
Renan Machado
Tiago Carvalho

Petrópolis
Fevereiro de 2025

Sumário

1 Capítulo 1

1.1 Introdução

A blockchain é uma tecnologia inovadora que permite a criação de registros seguros e imutáveis, sendo amplamente utilizada em sistemas financeiros, contratos inteligentes e diversas outras aplicações. Este trabalho apresenta a implementação de uma blockchain utilizando a linguagem C.

1.2 Objetivo do Trabalho

O principal objetivo deste projeto é desenvolver uma implementação funcional de uma blockchain em C, explorando conceitos fundamentais, tais como:

- **Listas encadeadas**: Estrutura de dados utilizada para armazenar os blocos da blockchain.
- **Criptografia**: Garantia de segurança e integridade dos dados armazenados.
- **Validação de blocos**: Processo de verificação para garantir a autenticidade das transações.

2 Capítulo 2

2.1 Desenvolvimento

Nesta seção, será apresentada a implementação da blockchain em C, abordando os principais aspectos técnicos envolvidos no projeto.

2.2 Estrutura de Dados

A blockchain será implementada utilizando uma lista duplamente encadeada, onde cada bloco conterá as seguintes informações:

- **Número do bloco**: Identificação única do bloco na cadeia.
- **Nonce**: Valor utilizado para modificar o hash e garantir a validade do bloco.
- **Hash do bloco anterior**: Ligação com o bloco anterior, garantindo a integridade da cadeia.
- **Hash das transações**: Hash das transações armazenadas no bloco.
- **Hash do bloco atual**: Identificação única do bloco gerada a partir dos seus dados.
- **Timestamp**: Registro da data e hora da criação do bloco.
- **Árvore de Merkle**: Estrutura para armazenar e validar transações.
- **Ponteiros**: Ligações para o bloco anterior e o próximo bloco na cadeia.

```c
// Estrutura do bloco
typedef struct block{
    int num;                // Número do bloco atual
    int nonce;              // Número para modificar o hash
    unsigned char previous_hash[64];  // Hash do bloco anterior
    unsigned char transacoes[64];
    unsigned char hash[64];          // Hash do bloco atual
    time_t timestamp;       // Tempo em que o bloco foi criado
    //Merkletree* arvore;
    struct block *proximo;  // Ponteiro para o próximo nó
    struct block *anterior; // Ponteiro para o nó anterior
} Block;
```

2.3 Calculo do Hash

O hash do bloco é gerado a partir da concatenação dos dados principais, incluindo o número do bloco, nonce, hash do bloco anterior e hash das transações.

2.4 Inserir um bloco

A adição de um novo bloco ocorre ao final da cadeia, garantindo que cada novo bloco aponte corretamente para o anterior.

2.5 Mineração de um bloco

A mineração envolve encontrar um nonce que gere um hash com uma determinada quantidade de zeros iniciais. Esse processo é chamado de Proof of Work (PoW).

2.6 MerkleTree

A MerkleTree é utilizada para armazenar e validar transações dentro de um bloco. Ela permite verificar a integridade de um conjunto de transações sem precisar armazenar todas elas diretamente.

2.7 Simulação de um Ataque

Para demonstrar a segurança da blockchain, podemos simular um ataque onde um invasor ataca o bloco inicial, altera sua transação e recalcula os hashes dos blocos subsequentes para tentar ocultar a modificação.

O ataque segue os seguintes passos:

1. O invasor localiza o bloco inicial na blockchain e altera os dados da transação.
2. O hash do bloco é recalculado para refletir a alteração.
3. Como cada bloco depende do hash do bloco anterior, todos os hashes subsequentes devem ser recalculados.
4. Esse processo exige um alto poder computacional, tornando o ataque inviável em blockchains com grande quantidades de blocos, tornado uma alteração nos blocos mais antigos inviável .

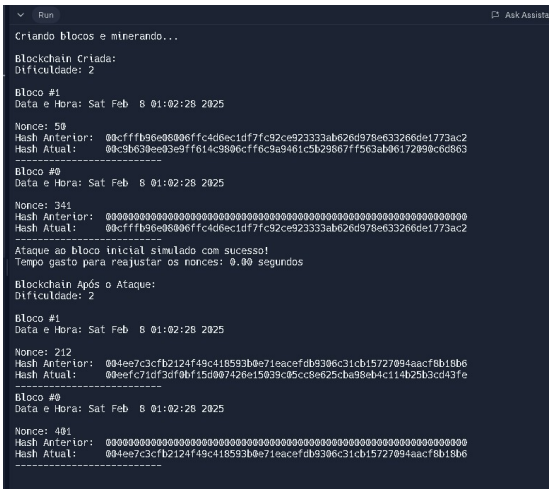
A nossa simulação de ataque possui um contador de tempo de espera para o recalculo de todos os hashes dos blocos posteriores ao inicial. Por conta da baixa dificuldade em relação à uma blockchain real, o ataque é realizado com sucesso.

2.8 Testes

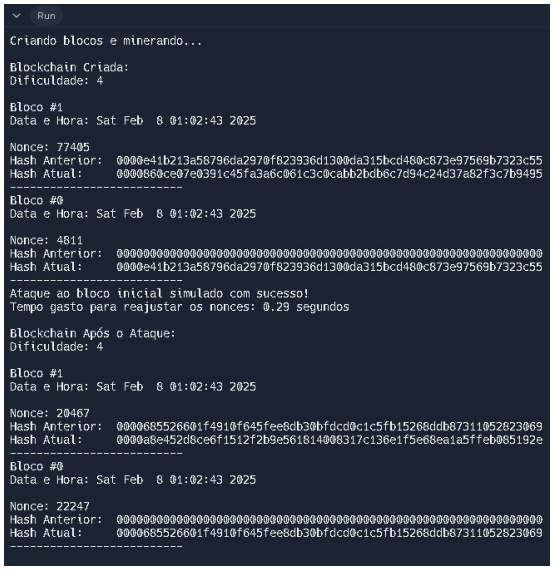
Alguns testes realizados com diferentes valores de dificuldade(2,4,6) evidenciando que quanto maior ela for, maior será o tempo o tempo necessário para o cálculo da hash, o que demanda bastante poder computacional.

Neste caso, como não utilizamos valores altos, o custo não foi tão grande

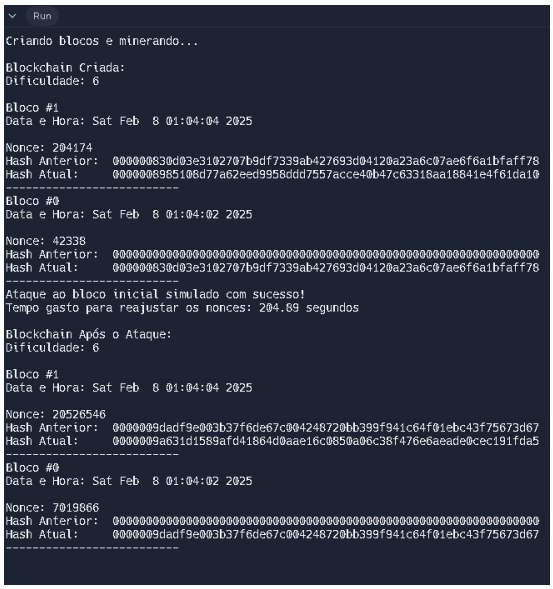
- **Dificuldade 2**: O hash do bloco deve começar com pelo menos 2 zeros.



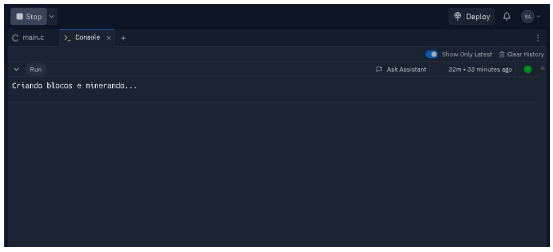
- **Dificuldade 4**: O hash do bloco deve começar com pelo menos 4 zeros.



- **Dificuldade 6**: O hash do bloco deve começar com pelo menos 6 zeros.



- **Dificuldade 8**: O hash do bloco deve começar com pelo menos 8 zeros.



3 Capitulo 3

3.1 Conclusão

A implementação de uma blockchain em C permitiu explorar conceitos fundamentais dessa tecnologia, como listas encadeadas, criptografia e Proof of Work. Durante o desenvolvimento, foi possível compreender os desafios envolvidos na criação de uma estrutura segura e eficiente para o armazenamento de transações, bem como os mecanismos que garantem a integridade da cadeia de blocos.

Além disso, a simulação de um ataque demonstrou a robustez do sistema, evidenciando a dificuldade de modificar blocos anteriores sem um alto poder computacional.

3.2 Ambiente de Desenvolvimento

Os códigos foram testados utilizando o ambiente de desenvolvimento disponibilizado no site Replit, com o objetivo de garantir que todos os integrantes do grupo pudessem executar o código de maneira uniforme, independentemente do ambiente local. Dessa forma, o código foi validado para funcionar corretamente na plataforma indicada, facilitando a execução e o acompanhamento do progresso do projeto.

O código pode ser acessado através do seguinte link: `https://replit.com/join/sighvrwsbx-tiagocarvalho25`