

**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 | 2 |
| 1.1 | Introdução | 2 |
| 1.2 | Objetivo do Trabalho | 2 |
| 2 | Capítulo 2 | 3 |
| 2.1 | Desenvolvimento | 3 |
| 2.2 | Estrutura de Dados | 3 |
| 2.3 | Calculo do Hash | 3 |
| 2.4 | Inserir um bloco | 3 |
| 2.5 | Mineração de um bloco | 3 |
| 2.6 | MerkleTree | 3 |
| 2.7 | Simulação de um Ataque | 4 |
| 2.8 | Testes | 4 |
| 3 | Capitulo 3 | 6 |
| 3.1 | Conclusão | 6 |
| 3.2 | Ambiente de Desenvolvimento | 6 |

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.

```
// 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, tornando uma alteração nos blocos mais antigos inviável.

A nossa simulação de ataque possui um contador de tempo de espera para o recálculo 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.

```
Run
Criando blocos e minerando...
Blockchain Criada:
Dificuldade: 2

Bloco #1
Data e Hora: Sat Feb 8 01:02:28 2025
Nonce: 50
Hash Anterior: 00cffffb96e0606ffcd6ec1df7fc92ce923333ab626d978e63266de1773ac2
Hash Atual: 00c9b638ee03e9ff614c9806cfff6c9a9461c5b29667ff563ab06172090c6d863

Bloco #0
Data e Hora: Sat Feb 8 01:02:28 2025
Nonce: 341
Hash Anterior: 0000000000000000000000000000000000000000000000000000000000000000
Hash Atual: 00cffffb96e0606ffcd6ec1df7fc92ce923333ab626d978e63266de1773ac2

Ataque ao bloco inicial simulado com sucesso!
Tempo gasto para reajustar os nonces: 0.00 segundos

Blockchain Após o Ataque:
Dificuldade: 2

Bloco #1
Data e Hora: Sat Feb 8 01:02:28 2025
Nonce: 212
Hash Anterior: 004ee7c3cfb2124f49c418593b0e71eacefdb9306c31cb15727094aacf6b18b6
Hash Atual: 00eefc71df3df0bf15d007426e15039c05cc8e625cba98eb4c114b25b3cd43fe

Bloco #0
Data e Hora: Sat Feb 8 01:02:28 2025
Nonce: 401
Hash Anterior: 0000000000000000000000000000000000000000000000000000000000000000
Hash Atual: 004ee7c3cfb2124f49c418593b0e71eacefdb9306c31cb15727094aacf6b18b6
```

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

```

Run
Criando blocos e minerando...

Blockchain Criada:
Dificuldade: 4

Bloco #1
Data e Hora: Sat Feb  8 01:02:43 2025

Nonce: 77405
Hash Anterior: 0000e41b213a58796da2970f823936d1300da315bcd480c873e97569b7323c55
Hash Atual: 0000660ce07e0391c45fa3a6c061c3c0cabb2bdb6c7d94c24d37a82f3c7b9495
-----
Bloco #0
Data e Hora: Sat Feb  8 01:02:43 2025

Nonce: 4811
Hash Anterior: 0000000000000000000000000000000000000000000000000000000000000000
Hash Atual: 0000e41b213a58796da2970f823936d1300da315bcd480c873e97569b7323c55
-----
Ataque ao bloco inicial simulado com sucesso!
Tempo gasto para reajustar os nonces: 0.29 segundos

Blockchain Após o Ataque:
Dificuldade: 4

Bloco #1
Data e Hora: Sat Feb  8 01:02:43 2025

Nonce: 20467
Hash Anterior: 0000685526601f4910f645fee8db30bfdcd0c1c5fb15268ddb87311052823069
Hash Atual: 0000a8c452d8ce6f1512f2b9e561814006317c13be1f5e06e1a5ffeb805192e
-----
Bloco #0
Data e Hora: Sat Feb  8 01:02:43 2025

Nonce: 22247
Hash Anterior: 0000000000000000000000000000000000000000000000000000000000000000
Hash Atual: 0000685526601f4910f645fee8db30bfdcd0c1c5fb15268ddb87311052823069
-----

```

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

```

Run
Criando blocos e minerando...

Blockchain Criada:
Dificuldade: 6

Bloco #1
Data e Hora: Sat Feb  8 01:04:04 2025

Nonce: 204174
Hash Anterior: 000000830d03e3102707b9df7339ab427693d04120a23a6c07ae6f6a1bfa7f78
Hash Atual: 0000008985106d77a62eed9958ddd7557acce40b47c63318aa18841e4f61da10
-----
Bloco #0
Data e Hora: Sat Feb  8 01:04:02 2025

Nonce: 42338
Hash Anterior: 0000000000000000000000000000000000000000000000000000000000000000
Hash Atual: 000000830d03e3102707b9df7339ab427693d04120a23a6c07ae6f6a1bfa7f78
-----
Ataque ao bloco inicial simulado com sucesso!
Tempo gasto para reajustar os nonces: 204.89 segundos

Blockchain Após o Ataque:
Dificuldade: 6

Bloco #1
Data e Hora: Sat Feb  8 01:04:04 2025

Nonce: 20526546
Hash Anterior: 0000009dadf9e003b37f6de67c004248720bb399f941c64f01ebc43f75673d67
Hash Atual: 0000009a631d1589af041864d0aae16c0850a06c38f476e0aede0cec191fda5
-----
Bloco #0
Data e Hora: Sat Feb  8 01:04:02 2025

Nonce: 7019866
Hash Anterior: 0000000000000000000000000000000000000000000000000000000000000000
Hash Atual: 0000009dadf9e003b37f6de67c004248720bb399f941c64f01ebc43f75673d67
-----

```

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

```

Stop
main.c  Console x
Run
Criando blocos e minerando...

```

3 Capítulo 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>