Implementação Simplificada do DES e do Protocolo Diffie-Hellman em Python

Maira Larissa Da Silva Fernandes[1]

[1]Instituto Federal da Paraiba
E-mail: maira.fernandes@academico.ifpb.edu.br

Abstract. *Este artigo apresenta a implementação, em Python, de uma abordagem simplificada do algoritmo DES para criptografia simétrica, associada ao mecanismo Diffie-Hellman para a troca de chaves. O objetivo é demonstrar os fundamentos teóricos e práticos por trás da criação de uma chave secreta compartilhada e da proteção de dados.*

1. Introdução

A segurança na transmissão de dados é uma preocupação central em sistemas de informação. Algoritmos de criptografia simétrica, como o DES (Data Encryption Standard), historicamente desempenharam um papel fundamental na proteção de informações. O protocolo Diffie-Hellman possibilita a troca segura de chaves, permitindo que duas partes estabeleçam uma chave secreta compartilhada mesmo através de um canal inseguro, sem a necessidade de um segredo prévio.

O presente trabalho tem como objetivo implementar, em Python e sem o uso de bibliotecas auxiliares, dois mecanismos: uma versão simplificada de um algoritmo inspirado no DES para criptografia/descriptografia e o protocolo Diffie-Hellman para a criação de uma chave compartilhada.

2. Metodologia

A solução foi dividida em duas partes inter-relacionadas:

2.1. Troca de Chaves Diffie-Hellman

A implementação do protocolo Diffie-Hellman foi realizada utilizando dois parâmetros públicos:

- P — um número primo, escolhido para a demonstração;
- G — a base, definida conforme os requisitos do protocolo.

Cada parte (emissor e receptor) gera uma chave privada aleatória e calcula sua respectiva chave pública pela fórmula:

$$\text{chave pública} = G^{\text{chave privada}} \mod P.$$

Após a troca das chaves públicas, ambas as partes computam a chave secreta compartilhada usando:

$$\text{chave compartilhada} = (\text{chave pública recebida})^{\text{chave privada}} \mod P.$$

Este processo garante que, mesmo em um canal inseguro, a chave compartilhada seja conhecida somente pelas partes envolvidas.

2.2. Criptografia Simétrica Inspirada no DES

Embora o algoritmo DES real incorra em diversas operações complexas (como permutações, substituições e múltiplas rodadas), a implementação proposta adota uma abordagem simplificada para demonstrar o conceito de criptografia simétrica. Nesta versão, a mensagem é cifrada aplicando uma operação XOR entre cada caractere e o valor da chave compartilhada.

O operador XOR (do inglês, "exclusive OR") é um operador lógico que compara dois bits e retorna 1 se eles forem diferentes e 0 se forem iguais. Em Python, o XOR é representado pelo símbolo `^`. No arquivo `des.py`, as funções `encrypt_message` e `decrypt_message` realizam a operação XOR. Em cada uma dessas funções, o caractere da mensagem é primeiro convertido para seu valor numérico com a função `ord()`. Em seguida, é aplicado o operador XOR entre esse valor e a chave compartilhada, e o resultado é convertido de volta para um caractere por meio da função `chr()`. Essa propriedade do XOR, que permite que a mesma operação aplicada duas vezes com a mesma chave recupere o valor original, torna possível tanto criptografar quanto descriptografar a mensagem utilizando o mesmo procedimento.

A integração destes mecanismos foi realizada de modo que:

- O emissor gera sua chave pública e envia para o receptor.
- O receptor responde com sua chave pública, permitindo que ambos calculem a chave compartilhada.
- Utilizando esta chave, o emissor cifra a mensagem e a envia ao receptor, que então decifra o conteúdo.

Embora a abordagem seja didática e simplificada, ela ilustra os princípios básicos da criptografia simétrica e da troca de chaves.

3. Resultados

A aplicação prática dos algoritmos demonstrou os seguintes resultados:

- **Troca de Chaves:** As partes envolvidas no processo conseguiram trocar suas chaves públicas e, a partir disso, calcularam uma chave secreta compartilhada idêntica, validando o funcionamento do protocolo Diffie-Hellman.
- **Criptografia/Descriptografia:** Após o estabelecimento da chave compartilhada, a mensagem enviada foi criptografada através de uma operação XOR e, ao ser recebida, foi corretamente decifrada. O exemplo prático mostrou que o conteúdo original foi integralmente recuperado, comprovando a eficácia do método.
- **Integração e Comunicação:** A utilização de sockets para simular a comunicação entre emissor e receptor possibilitou um ambiente de teste que reforçou a compreensão da sincronização entre as etapas de troca de chaves e de criptografia.

4. Discussão

Durante a implementação, diversos desafios foram identificados:

- **Simplificação do DES:** A abordagem baseada em XOR não reproduz a complexidade do DES, que envolve operações de permutação e múltiplas rodadas de transformação. Todavia, ela foi suficiente para demonstrar o conceito de uso de uma chave compartilhada na criptografia simétrica.

- **Parâmetros do Diffie-Hellman:** Os valores escolhidos para P e G foram adequados para fins didáticos, mas não atenderiam aos requisitos de segurança para aplicações reais, que demandam números primos muito maiores.
- **Integração e Comunicação via Sockets:** Embora a simulação da comunicação entre emissor e receptor utilizando sockets tenha sido eficaz, aspectos como tratamento de erros e sincronização poderiam ser aprimorados para melhorar a robustez do sistema.

A análise destes pontos evidencia que, apesar da implementação atender aos objetivos propostos, há espaço para aprimoramentos, sobretudo na complexidade e segurança dos algoritmos empregados.

5. Conclusão

Este trabalho demonstrou a viabilidade de implementar, em Python, uma abordagem simplificada inspirada no DES associada ao protocolo Diffie-Hellman para a troca de chaves. Os principais aprendizados foram:

- A troca de chaves Diffie-Hellman possibilita o estabelecimento de uma chave secreta compartilhada, mesmo através de um canal inseguro.
- Uma criptografia simplificada baseada em operações XOR ilustra os fundamentos da criptografia simétrica, apesar de não reproduzir a complexidade do DES.
- A integração dos mecanismos de troca de chaves e de criptografia, utilizando sockets, reforça a importância de uma comunicação segura e sincronizada entre os componentes.

Como sugestões para trabalhos futuros, recomenda-se a implementação completa do algoritmo DES, a utilização de parâmetros robustos para o Diffie-Hellman e a exploração de algoritmos modernos, como o AES, para melhorar a segurança e a eficiência do sistema.

Referências

[1] RIVEST, R. L. et al. *The DES Algorithm*. Disponível em: `https://www.itl.nist.gov/fipspubs/fip46-3.htm`. Acesso em: [data de acesso].

[2] DIFFIE, W.; HELLMAN, M. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 1976.