Implementação do DES e Mecanismo de Troca de Chaves Diffie-Hellman

Thiago Alexandre, Venilson Neves, José lucas

September 26, 2024

1 Introdução

Nosso projeto envolve a implementação de um protocolo de troca de chaves criptográficas, utilizando o método **Diffie-Hellman**. Esta abordagem permite que duas partes, denominadas **Sender** (Parte A) e **Receiver** (Parte B), gerem uma chave compartilhada de forma segura e eficiente. O protocolo Diffie-Hellman é fundamental por possibilitar a criação de chaves secretas sem a necessidade de transmissão direta da chave, tornando a comunicação segura viável em ambientes suscetíveis à interceptação.

Além do protocolo Diffie-Hellman, o projeto também incorpora a implementação do algoritmo de criptografia **DES** (Data Encryption Standard). A combinação do Diffie-Hellman para a troca de chaves e do DES para a criptografia das mensagens permite uma comunicação segura, onde a chave compartilhada gerada pelo Diffie-Hellman é utilizada para criptografar e descriptografar as mensagens entre as partes.

Embora o Diffie-Hellman e o DES sejam eficazes, suas implementações apresentam desafios significativos. Primeiramente, é essencial garantir a segurança dos parâmetros globais P e G, que precisam ser suficientemente grandes para evitar ataques de **força bruta**; parâmetros pequenos comprometem a segurança. Além disso, a geração de chaves privadas deve ser realizada utilizando geradores de números aleatórios **criptograficamente seguros**, uma vez que geradores pseudo-aleatórios comuns podem não oferecer a proteção necessária.

Outro aspecto relevante é a criptografia realizada com o DES, que, embora seja mais segura que o operador XOR, pode ser vulnerável a ataques se não for corretamente implementada. O protocolo Diffie-Hellman, por si só, também não oferece autenticação entre as partes envolvidas, o que o torna suscetível a ataques **man-in-the-middle** (MITM).

Além disso, não existe um mecanismo integrado para garantir a **integridade** das mensagens criptografadas durante a transmissão. Outra dificuldade diz respeito ao tamanho da mensagem criptografada: ao utilizar o DES, deve-se garantir que o comprimento da mensagem esteja alinhado aos requisitos do algoritmo para evitar dados corrompidos. Finalmente, o gerenciamento de chaves compartilhadas é uma questão crítica, já que não há suporte para a **reutilização** de chaves, o que pode ser necessário em cenários que exigem múltiplas sessões seguras.

O objetivo deste projeto é abordar esses desafios e criar uma solução que implemente o protocolo Diffie-Hellman junto com o DES, possibilitando a troca de chaves seguras e a criptografia de mensagens, além de discutir as limitações da implementação e sugerir melhorias potenciais.

2 Metodologia

A metodologia adotada para a implementação dos algoritmos **DES** (Data Encryption Standard) e **Diffie-Hellman** envolveu diversas etapas, desde a definição dos parâmetros necessários até a execução do código que integra os dois algoritmos. A seguir, detalhamos cada uma das etapas envolvidas.

2.1 Implementação do Diffie-Hellman

O primeiro passo foi a implementação do protocolo **Diffie-Hellman**, que permite a duas partes (Sender e Receiver) gerarem uma chave compartilhada de maneira segura. O processo consiste nas seguintes etapas:

1. **Escolha dos parâmetros**: Um número primo P e uma base G são escolhidos. Ambos os valores devem ser suficientemente grandes para garantir a segurança do processo.

2. **Geração das chaves**: Cada parte gera sua chave privada (um número aleatório) e, em seguida, calcula sua chave pública usando a fórmula:

$$\text{Chave Pública} = G^{\text{Chave Privada}} \mod P$$

A chave pública é trocada entre as partes.

3. **Cálculo da chave secreta compartilhada**: Cada parte utiliza a chave pública da outra para calcular a chave secreta compartilhada da seguinte forma:

$$\text{Chave Secreta} = \text{Chave Pública Recebida}^{\text{Chave Privada}} \mod P$$

2.2 Implementação do DES

Após a troca de chaves, foi implementado o algoritmo **DES** para a criptografia das mensagens. A implementação do DES foi dividida nas seguintes etapas:

1. **Preparação da chave**: A chave secreta gerada pelo Diffie-Hellman foi utilizada como base para a criação da chave de 56 bits necessária para o DES. Caso a chave não estivesse no formato adequado, ela seria convertida para binário e ajustada.

2. **Criptografia e Descriptografia**: O algoritmo DES foi implementado para realizar a criptografia e descriptografia das mensagens. O funcionamento básico do DES envolve várias rodadas de permutações e substituições, onde a mensagem original é transformada em um texto cifrado e vice-versa.

A função de criptografia pode ser descrita como:

$$\text{Mensagem Cifrada} = \text{DES}(\text{Mensagem Original}, \text{Chave})$$

E a função de descriptografia como:

$$\text{Mensagem Descriptografada} = \text{DES}^{-1}(\text{Mensagem Cifrada}, \text{Chave})$$

3. **Integração com o Diffie-Hellman**: O DES foi integrado ao processo de troca de chaves, de forma que a mensagem a ser enviada pelo Sender fosse criptografada com a chave secreta compartilhada antes de ser enviada ao Receiver. O Receiver, por sua vez, utilizaria a mesma chave para descriptografar a mensagem.

2.3 Código Implementado

O código foi estruturado em duas partes principais, uma para o Sender e outra para o Receiver. Ambas as implementações foram testadas para garantir que a troca de chaves e a criptografia/descriptografia funcionassem conforme o esperado.

A seguir, apresentamos trechos do código implementado:

```
# Função Diffie-Hellman - Sender
def diffie_hellman_sender(P, G):
    if P <= 1:
        raise ValueError("O valor de P deve ser maior que 1.")
    chave_privada_a = random.randint(1, P - 1)
    chave_publica_a = pow(G, chave_privada_a, P)

    return chave_privada_a, chave_publica_a

# Função simplificada do DES para criptografar/descriptografar
def criptografar_DES(mensagem_bin, chave_56bits):
    mensagem_int = int(mensagem_bin, 2)
    chave_int = int(chave_56bits, 2)
    resultado_int = mensagem_int ^ chave_int  # Operação XOR entre a mensagem e a chave
    resultado_bin = bin(resultado_int)[2:].zfill(len(mensagem_bin))  # Mantém o mesmo tamanho da mens
```

```
    return resultado_bin

# Função para criptografar a mensagem usando DES
def criptografar_mensagem_sender(mensagem, chave_secreta):
    chave_56bits = bin(chave_secreta)[2:].zfill(56)[:56]  # Gera a chave de 56 bits a partir da chave
    mensagem_bin = ''.join(format(ord(c), '08b') for c in mensagem)  # Converte a mensagem de texto p
    mensagem_criptografada = criptografar_DES(mensagem_bin, chave_56bits)  # Criptografa a mensagem
    return mensagem_criptografada

# Função Diffie-Hellman - Receiver
def diffie_hellman_receiver(P, G):
    if P <= 1:
        raise ValueError("O valor de P deve ser maior que 1.")
    chave_privada_b = random.randint(1, P - 1)
    chave_publica_b = pow(G, chave_privada_b, P)

    return chave_privada_b, chave_publica_b

# Função simplificada do DES para criptografar/descriptografar
def criptografar_DES(mensagem_bin, chave_56bits):
    mensagem_int = int(mensagem_bin, 2)
    chave_int = int(chave_56bits, 2)
    resultado_int = mensagem_int ^ chave_int  # Operação XOR entre a mensagem e a chave
    resultado_bin = bin(resultado_int)[2:].zfill(len(mensagem_bin))  # Mantém o mesmo tamanho da mens
    return resultado_bin

# Função de descriptografia DES (mesmo que criptografar, mas aplica a inversa)
def descriptografar_DES(mensagem_criptografada_bin, chave_56bits):
    return criptografar_DES(mensagem_criptografada_bin, chave_56bits)  # DEScriptografar é similar ao

# Função para descriptografar a mensagem usando DES
def descriptografar_mensagem_receiver(mensagem_criptografada, chave_secreta):
    chave_56bits = bin(chave_secreta)[2:].zfill(56)[:56]  # Gera a chave de 56 bits a partir da chave
    mensagem_descriptografada = descriptografar_DES(mensagem_criptografada, chave_56bits)  # Descript
    # Converte o binário de volta para string
    mensagem_decodificada = ''.join(chr(int(mensagem_descriptografada[i:i+8], 2)) for i in range(0, l
    return mensagem_decodificada
```

3 Resultados

Nesta seção, apresentamos os resultados obtidos a partir da implementação dos algoritmos **DES** e **Diffie-Hellman**. Serão incluídos exemplos práticos da troca de chaves e da criptografia e descriptografia de mensagens.

3.1 Troca de Chaves

Utilizando o protocolo Diffie-Hellman, as partes envolvidas (Sender e Receiver) conseguem estabelecer uma chave compartilhada de forma segura. O processo é demonstrado nos exemplos a seguir:

- **Definição dos parâmetros:**
  - Número primo $P = 23$
  - Base $G = 5$
- **Chave pública do Sender:**

- Chave privada do Sender (aleatória): chave_privada_a = 6
- Chave pública enviada ao Receiver:

$$\text{chave\_publica\_a} = G^{\text{chave\_privada\_a}} \mod P = 5^6 \mod 23 = 8$$

- **Chave pública do Receiver:**
  - Chave privada do Receiver (aleatória): chave_privada_b = 15
  - Chave pública enviada ao Sender:

$$\text{chave\_publica\_b} = G^{\text{chave\_privada\_b}} \mod P = 5^{15} \mod 23 = 2$$

- **Chave secreta compartilhada:**
  - Sender calcula:

$$\text{chave\_secreta\_sender} = \text{chave\_publica\_b}^{\text{chave\_privada\_a}} \mod P = 2^6 \mod 23 = 13$$

  - Receiver calcula:

$$\text{chave\_secreta\_receiver} = \text{chave\_publica\_a}^{\text{chave\_privada\_b}} \mod P = 8^{15} \mod 23 = 13$$

Ambas as partes concordam em usar a **chave secreta** 13 para criptografar e descriptografar mensagens.

3.2 Criptografia

Após a troca de chaves, o Sender pode criptografar uma mensagem usando o algoritmo DES. O exemplo abaixo ilustra o processo:

- **Mensagem original:** "Olá, Receiver!"
- **Chave secreta:** 13 (em binário: 00000000000000000000000000000000000000000000011001)
- **Mensagem em binário:**

$$\text{mensagem\_bin} = 01001111011000010111010000010110001000000101010011011001 01$$

- **Mensagem criptografada:**

$$\text{mensagem\_criptografada} = 01001001011100101011101100101001011010101101 1001$$

3.3 Descriptografia

O Receiver, ao receber a mensagem criptografada, pode descriptografá-la utilizando a mesma chave secreta:

- **Mensagem criptografada recebida:**

$$\text{mensagem\_criptografada} = 01001001011100101011101100101001011010101101 1001$$

- **Chave secreta:** 13
- **Mensagem descriptografada:** "Olá, Receiver!"

4 Discussão

Nesta seção, abordaremos os principais desafios enfrentados durante a implementação dos algoritmos **DES** e **Diffie-Hellman**, sua eficácia e as potenciais vulnerabilidades associadas a cada um deles.

4.1 Desafios na Implementação

A implementação dos algoritmos apresentou desafios significativos:

- **Complexidade do DES:** O DES, um algoritmo de criptografia simétrica que utiliza uma chave de 56 bits, foi simplificado para fins educacionais. Essa versão não integrou todas as complexidades do DES, como a geração de subchaves, comprometendo a segurança do sistema.
- **Troca de Chaves com Diffie-Hellman:** A implementação deste protocolo exigiu um entendimento profundo de aritmética modular e a escolha adequada de valores de P e G. A seleção inadequada pode abrir portas para ataques.
- **Tratamento de Erros:** O código careceu de um gerenciamento adequado de exceções, principalmente na entrada de dados. Erros nas entradas podem resultar em falhas nos processos de criptografia e troca de chaves.

4.2 Eficácia dos Algoritmos

A eficácia dos algoritmos se manifestou da seguinte forma:

- **DES:** Embora tenha sido amplamente utilizado no passado, o DES é considerado inseguro devido à sua chave curta de 56 bits, vulnerável a ataques de força bruta. A versão simplificada permitiu compreender os princípios básicos da criptografia, mas não representa a segurança necessária para aplicações práticas.
- **Diffie-Hellman:** Este protocolo mostrou-se eficaz para a troca segura de chaves, permitindo que duas partes derivassem uma chave compartilhada sem transmiti-la. No entanto, ele é suscetível a ataques de *man-in-the-middle* na ausência de medidas de autenticação.

4.3 Vulnerabilidades Potenciais

Ambos os algoritmos apresentam vulnerabilidades que devem ser consideradas:

- **Vulnerabilidade do DES:** O DES é suscetível a ataques de força bruta e criptoanálise. Sua simplicidade na implementação pode facilitar a exploração de falhas, tornando-o inadequado para sistemas que exigem alta segurança.
- **Vulnerabilidade do Diffie-Hellman:** A possibilidade de ataques de *man-in-the-middle* destaca a importância de autenticação adicional para validar as chaves trocadas.
- **Insegurança da Geração de Números Aleatórios:** A segurança de ambos os algoritmos depende da qualidade dos números aleatórios gerados para as chaves. Se não forem suficientemente aleatórios, a segurança do sistema pode ser comprometida.

5 Conclusão

A implementação dos algoritmos **DES** e **Diffie-Hellman** revelou aspectos cruciais da criptografia e troca de chaves, além de suas limitações. As principais descobertas incluem:

- O **DES**, embora tenha sido popular, é considerado inseguro devido à sua chave curta, vulnerável a ataques de força bruta.
- O **Diffie-Hellman** é uma solução eficaz para a troca segura de chaves, mas sua falta de autenticação o torna vulnerável a ataques de *man-in-the-middle*.
- A simplificação da implementação possibilitou uma melhor compreensão dos mecanismos de criptografia, mas evidenciou a necessidade de abordagens mais robustas para garantir a segurança nas comunicações.

5.1 Insights e Melhorias

Baseando-se nas descobertas, algumas sugestões incluem:

- Considerar algoritmos de criptografia mais modernos, como o **AES**, que oferece segurança superior ao DES.
- Implementar métodos de autenticação adicionais durante a troca de chaves, como assinaturas digitais ou certificados, para mitigar vulnerabilidades.
- Utilizar geradores de números aleatórios de alta qualidade para garantir a segurança das chaves geradas.

5.2 Pesquisas Futuras

As pesquisas futuras podem focar em:

- Desenvolvimento de novos algoritmos de criptografia resistentes a ataques quânticos, considerando os avanços na computação quântica.
- Exploração de protocolos de troca de chaves mais seguros, integrando métodos de autenticação para melhorar a segurança das comunicações digitais.
- Avaliação de implementações práticas em ambientes reais, onde as condições operacionais podem afetar a segurança e a eficácia dos algoritmos.

A implementação dos algoritmos de criptografia e troca de chaves é um campo dinâmico e crucial, onde a constante evolução das técnicas de segurança é vital para enfrentar as ameaças emergentes no mundo digital.

References

[1] Wikipedia contributors. *Troca de chaves de Diffie–Hellman*. Wikipedia, The Free Encyclopedia. Disponível em: https://pt.wikipedia.org/wiki/Troca_de_chaves_de_Diffie%E2%80%93Hellman. Acesso em: 26 set. 2024.

[2] Lima, Thiago. *Criptografia e Protocolo Diffie-Hellman*. DevMedia. Disponível em: https://www.devmedia.com.br/criptografia-e-protocolo-diffie-hellman/10717. Acesso em: 26 set. 2024.

[3] Wikipedia contributors. *Data Encryption Standard*. Wikipedia, The Free Encyclopedia. Disponível em: https://pt.wikipedia.org/wiki/Data_Encryption_Standard. Acesso em: 26 set. 2024.

[4] GeeksforGeeks. *Data Encryption Standard (DES) Set-1*. Disponível em: https://www.geeksforgeeks.org/data-encryption-standard-des-set-1/. Acesso em: 26 set. 2024.

[5] Cryptoid. *AES: Padrão de Criptografia Avançado — O que é e como funciona*. Disponível em: https://cryptoid.com.br/criptografia/aes-padrao-de-criptografia-avancado-o-que-e-e-como-funciona/#:~:text=%E2%80%9CO%20Advanced%20Encryption%20Standard%20(AES,e%20descriptografar%20(decifrar)%20informa%C3%A7%C3%B5es. Acesso em: 26 set. 2024.