FACULDADE ENGENHEIRO SALVADOR ARENA

CAYO VINÍCIUS NEVES MAGALHÃES RA:081200050

LEONARDO LEAL FERNANDES RA:081200040

LUCAS ALVES SILVA RA:081200031

LUCAS QUEIROZ VIEIRA RA:081200020

ESPECIFICAÇÃO FUNCIONAL

São Bernardo do Campo
2024

Sumário

1. DESCRIÇÃO DO PROJETO

Este projeto implementa um sistema de comunicação seguro entre um servidor e um cliente utilizando duas técnicas clássicas de criptografia: **Diffie-Hellman** para a troca de chaves seguras e a **Cifra de César** para criptografia e descriptografia das mensagens trocadas. O foco principal é garantir a confidencialidade das mensagens durante a comunicação, mesmo em canais potencialmente inseguros.

O projeto é ideal para aprender e experimentar com conceitos básicos de criptografia e segurança da informação, além de fornecer uma base sólida para entender como sistemas de criptografia simétricos e assimétricos podem ser combinados para proteger dados em trânsito.

Este README fornecerá uma visão detalhada do funcionamento interno do projeto, instruções sobre como utilizá-lo, e uma explicação extensiva da criptografia implementada.

2. ESTRUTURA DO PROJETO

2.1. Servidor

O servidor é responsável por:

1. Receber Conexões do Cliente:
    - O servidor aguarda na porta TCP especificada (1300 por padrão) por conexões de clientes.
2. Troca de Chaves com o Cliente:
    - Ao receber uma conexão, o servidor realiza uma troca de chaves Diffie-Hellman com o cliente para gerar uma chave secreta compartilhada.
3. Descriptografia e Processamento do Texto:
    - O servidor descriptografa a mensagem recebida usando a chave fornecida pelo cliente, converte o texto para maiúsculas e o recriptografa usando a chave secreta gerada.
4. Envio da Resposta ao Cliente:
    - A mensagem recriptografada é enviada de volta ao cliente.

O servidor é estruturado com as seguintes funções principais:

- setup_server(port): Configura o servidor para ouvir em uma porta específica.
- handle_client(connection_socket): Lida com a comunicação com o cliente.
- run_server(port): Função principal que mantém o servidor em execução.
- diffie_hellman_r1: Calcula o primeiro valor de Diffie-Hellman (R1).
- diffie_hellman_k1: Calcula a chave secreta K1 usando Diffie-Hellman.
- shift_char: Aplica a Cifra de César em um caractere específico.
- encrypt_cesar e decrypt_cesar: Criptografia.

2.2. Cliente

O cliente é responsável por:

1. Escolher Parâmetros Criptográficos (G e N):
    - O cliente escolhe valores primos para G (base) e N (módulo), essenciais para a troca de chaves Diffie-Hellman.
2. Criptografar o Texto com Cifra de César:
    - O texto inserido pelo usuário é criptografado usando a Cifra de César, com um deslocamento baseado em uma chave calculada (R2).
3. Enviar os Dados ao Servidor:
    - O cliente envia os valores de R2, G, N, e o texto criptografado ao servidor.
4. Receber e Descriptografar a Resposta:
    - O cliente recebe o texto recriptografado do servidor e o descriptografa usando a chave compartilhada.

As funções principais do cliente incluem:

- get_prime_input(prompt): Solicita ao usuário que insira um número primo.
- send_data_to_server(result_json, server_name, server_port): Configura e envia os dados ao servidor.
- receive_and_process_response(client_socket, N, y): Recebe a resposta do servidor e processa o texto criptografado.

- run_client(): Função principal que coordena o fluxo do cliente.

- diffie_hellman_r1: Calcula o primeiro valor de Diffie-Hellman (R1).
- diffie_hellman_k1: Calcula a chave secreta K1 usando Diffie-Hellman.
- shift_char: Aplica a Cifra de César em um caractere específico.
- encrypt_cesar e decrypt_cesar: Criptografia.

3. EXPLICAÇÃO DAS TÉCNICAS DE CRIPTOGRAFIA

3.1. Cifra de César

A Cifra de César é um dos métodos mais simples e conhecidos de criptografia por substituição. Ela funciona deslocando cada letra do texto original (plaintext) por um número fixo de posições no alfabeto.

3.1.1. Características Implementadas

- Suporte a Letras Acentuadas: Além das letras do alfabeto inglês (A-Z), a implementação suporta letras acentuadas comuns na língua portuguesa, como á, é, í, ó, ú, ç etc.
- Distinção Entre Maiúsculas e Minúsculas: O código diferencia entre letras maiúsculas e minúsculas, aplicando a cifra de maneira apropriada a cada caso.

3.1.2. Código Explicado

```
# Função para criptografar um texto usando a Cifra de César
def encrypt_cesar(text, shift):
    result = ""
    lower_alphabet = "abcdefghijklmnopqrstuvwxyz"
    upper_alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    lower_accented = "áéíóúàèìòùâêîôûãõäëïöüñç"
    upper_accented = "ÁÉÍÓÚÀÈÌÒÙÂÊÎÔÛÃÕÄËÏÖÜÑÇ"

    for char in text:
        if char.islower():
            result += shift_char(char, shift, lower_alphabet + lower_accented)
        elif char.isupper():
            result += shift_char(char, shift, upper_alphabet + upper_accented)
        else:
            result += char  # Não modifica caracteres que não são letras
    return result
```

- Alfabetos: As variáveis lower_alphabet e upper_alphabet contêm as letras do alfabeto inglês. As variáveis lower_accented e upper_accented contêm letras acentuadas.
- Iteração e Verificação: O código verifica se cada caractere é uma letra minúscula ou maiúscula e, em seguida, aplica a cifra correspondente usando a função shift_char.
- Não-Alfabéticos: Caracteres que não são letras, como números ou símbolos (@, #, !), permanecem inalterados.

3.2. Troca de Chaves Diffie-Hellman

O algoritmo de Diffie-Hellman permite que duas partes (neste caso, o cliente e o servidor) compartilhem uma chave secreta comum de forma segura, mesmo que estejam se comunicando através de um canal inseguro.

3.2.1. Passos Implementados

- Escolha de Parâmetros Públicos:
    - Os valores G (base) e N (módulo) são escolhidos pelo cliente e compartilhados com o servidor. Ambos devem ser números primos.
- Geração das Chaves Privadas:
    - O servidor utiliza uma chave privada fixa x = 5.
    - O cliente escolhe uma chave privada y (definida como y = 23 no código).
- Cálculo das Chaves Públicas:
    - O servidor calcula sua chave pública R1 = (G^x) % N.
    - O cliente calcula sua chave pública R2 = (G^y) % N.
- Geração da Chave Compartilhada:
    - O servidor calcula a chave secreta K1 = (R2^x) % N utilizando a chave pública do cliente.
    - O cliente calcula a chave secreta K2 = (R1^y) % N utilizando a chave pública do servidor.
    - As chaves K1 e K2 são iguais e são utilizadas para criptografar e descriptografar as mensagens trocadas.

4. EXEMPLO DE USO

A comunicação descrita envolve um processo de criptografia, transmissão e descriptografia de dados entre um cliente e um servidor utilizando técnicas clássicas como a **Cifra de César** e a **troca de chaves de Diffie-Hellman**.

4.1. Início do Servidor

- O servidor é iniciado e fica aguardando conexões na porta TCP 1300.

```
TCP Server iniciado e aguardando conexões...
```

4.2. Configuração e Conexão do Cliente

- O cliente, ao iniciar, solicita ao usuário que insira os valores para G e N. Estes devem ser números primos.

```
Digite o G: 32
32 não é primo!

Digite o G: 52
52 não é primo!

Digite o G: 2
2 é primo!

Digite o N: 3
3 é primo!
```

- Após escolher valores válidos (G = 2 e N = 3), o cliente solicita o texto a ser criptografado.

```
Digite o texto: "Às vezes, é necessário pôr à prova a fé que há em nós, para que possamos alcançar
 o ápice da realização! #$%&*()_+[]{};:.,<>?/~"
```

4.3. Criptografia e Envio pelo Cliente

- O cliente utiliza a Cifra de César para criptografar o texto, utilizando o valor R2 = 2, que é calculado como parte do processo de troca de chaves Diffie-Hellman.
- O texto criptografado, junto com os valores R2, G e N, é empacotado em um JSON e enviado ao servidor.

```
resultJson:{"R2": 2, "text": "\"\u00ccu xg\u00e9gu, \u00f3 pgeguu\u00edtkq r\u00e3t \u00ec rtqxc c
 h\u00f3 swg j\u00ed go p\u00e0u, rctc swg rquucoqu cnecpbct q \u00edrkeg fc tgcnk\u00e9cb\u00e4q!
 #$%&*()_+[]{};:.,<>?/~\"", "G": 2, "N": 3}
```

4.4. Recebimento e Processamento pelo Servidor

- Usando o valor R2, o servidor descriptografa o texto utilizando a Cifra de César.

```
Texto descriptografado:  "Às vezes, é necessário pôr à prova a fé que há em nós, para que possamos alcançar o ápice da realização! #$%&*()_+[]{};:.,<>?/~"
```

4.5. Resposta Enviada pelo Servidor

- O servidor então processa o texto, convertendo-o para maiúsculas. Após o processamento, o servidor recriptografa o texto utilizando a chave secreta compartilhada K1, que foi gerada durante a troca de chaves Diffie-Hellman.

```
Resposta enviada ao cliente:  {"R1": 2, "text": "\"\u00ccU XG\u00c9GU, \u00d3 PGEGUU\u00cdTKQ R\u00c3T \u00cc RTQXC C H\u00d3 SWG J\u00cd GO P\u00c0U, RCTC SWG RQUUCO
QU CNECPBCT Q \u00cdRKEG FC TGCNK\u00c9CB\u00c4Q! #$%&*()_+[]{};:.,<>?/~\""}
```

4.6. Descriptografia e Exibição pelo Cliente

- O cliente recebe a resposta do servidor e descriptografa o texto usando a chave K2, que foi gerada da mesma forma que K1 no servidor.

```
Descriptografado: "ÀS VEZES, É NECESSÁRIO PÔR À PROVA A FÉ QUE HÁ EM NÓS, PARA QUE POSSAMOS ALCANÇ
AR O ÁPICE DA REALIZAÇÃO! #$%&*()_+[]{};:.,<>?/~"
```

4.7. Controle de tráfego (Wireshirk)

O controle de tráfego de rede é uma parte crucial da análise de segurança e desempenho em sistemas de comunicação. Ferramentas como o **Wireshark** permitem capturar e analisar pacotes de dados que são transmitidos e recebidos em uma rede. Neste projeto, o Wireshark pode ser utilizado para monitorar a comunicação entre o cliente e o servidor, garantindo que os dados estão sendo transmitidos e criptografados corretamente.

4.7.1. Configuração do Wireshark

1. **Instalação do Wireshark:**
    - Baixe e instale o Wireshark a partir do site oficial.
2. **Selecionando a Interface de Rede:**
    - Ao abrir o Wireshark, selecione a interface de rede que está sendo usada para a comunicação (por exemplo, Ethernet ou Wi-Fi).
3. **Aplicação de Filtros:**
    - Para monitorar especificamente a comunicação entre o cliente e o servidor no projeto, você pode usar um filtro para capturar pacotes TCP na porta específica (porta 1300 neste caso).
    - O filtro pode ser configurado como: “ip.addr == 10.1.70.2 && tcp.port == 1300”
        - ip.addr == 10.1.70.2: Filtra os pacotes onde o endereço IP de origem ou destino é 10.1.70.2 (o IP do servidor neste caso).
        - tcp.port == 1300: Filtra os pacotes que estão usando a porta 1300 (a porta onde o servidor está ouvindo).

4. **Captura de Pacotes:**

    - Clique no botão de captura (ícone de tubarão verde) para iniciar a captura de pacotes.
    - Execute o cliente para iniciar a comunicação com o servidor.
    - O Wireshark começará a capturar todos os pacotes que atendem aos critérios do filtro.

4.7.2. Análise de Pacotes

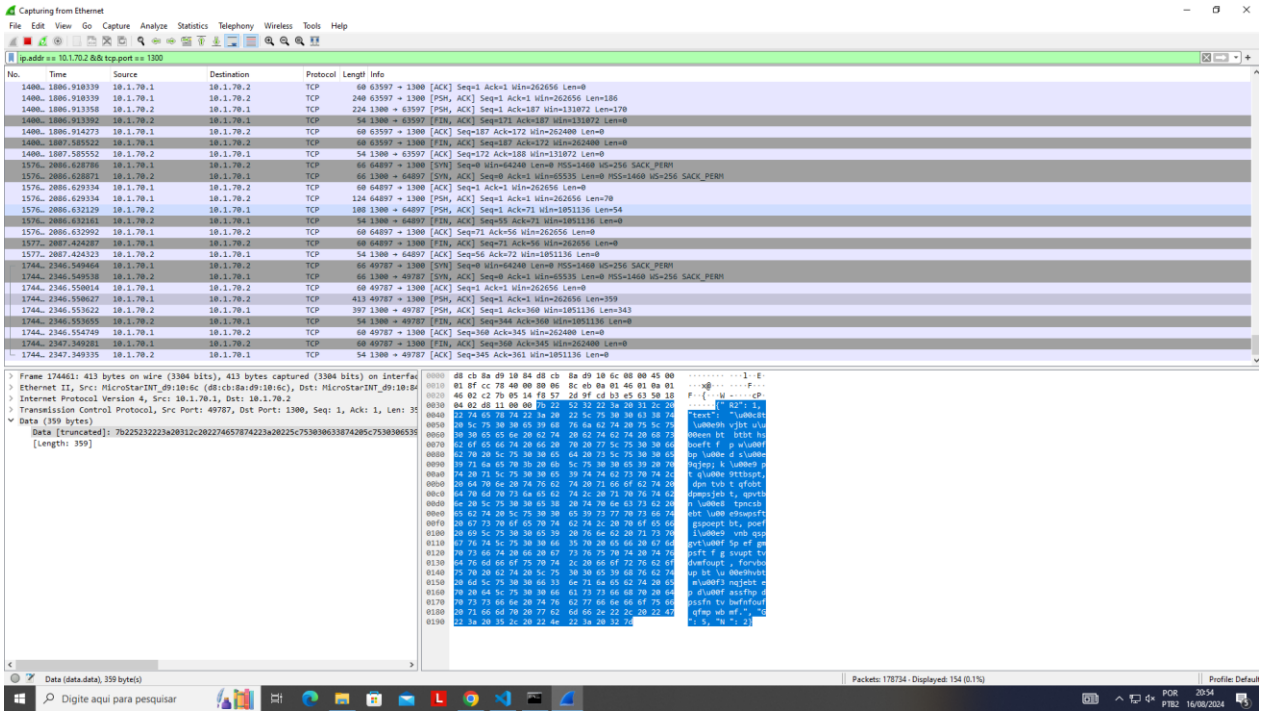
Após a captura, os pacotes podem ser analisados para entender o fluxo de dados entre o cliente e o servidor:

1. **Identificação de Pacotes Importantes:**

    - Você verá pacotes TCP relacionados à conexão (SYN, ACK), à transmissão dos dados criptografados, e às respostas do servidor.
    - O Wireshark mostrará os pacotes em detalhes, incluindo informações sobre o protocolo, o tamanho do pacote, e os dados brutos transmitidos.

2. **Verificação de Dados Criptografados:**

    - No painel inferior do Wireshark, você pode visualizar o conteúdo dos pacotes. Como os dados estão criptografados, você verá um conteúdo que não é imediatamente legível.
    - Por exemplo, um pacote pode mostrar uma string criptografada como:



- Isso indica que a criptografia está funcionando corretamente, já que o conteúdo não é imediatamente legível.

3. **Análise de Segurança:**

   - Verifique se a comunicação está ocorrendo conforme esperado, sem vazamentos de informações sensíveis em texto plano.

   - Confirme que o tráfego segue o padrão esperado, sem pacotes inesperados que poderiam indicar uma tentativa de ataque ou um problema de configuração.

4.7.3. Salvando e Documentando a Análise

- Salvar Captura: Você pode salvar a captura de pacotes clicando em File > Save As no Wireshark. Isso permite que a captura seja revisitada ou compartilhada posteriormente.

- Documentação: Inclua capturas de tela e descrições das análises em relatórios ou documentações para fornecer evidências de que a comunicação entre o cliente e o servidor foi realizada de forma segura e conforme esperado.

5. Conclusão

Este projeto demonstra como utilizar criptografia clássica para proteger a comunicação entre um cliente e um servidor em um ambiente de rede. A troca de chaves de Diffie-Hellman e a Cifra de César são implementadas de maneira a garantir a segurança dos dados, mesmo em canais não seguros.

O suporte a caracteres acentuados e a distinção entre maiúsculas e minúsculas tornam este exemplo particularmente relevante para aplicações em contextos multilíngues, onde a integridade do texto é crucial. Esta implementação serve como uma base educacional sólida para compreender conceitos fundamentais de criptografia e comunicação segura em redes.