Universidade de Brasília - UnB

Disciplina: Fundamento de Redes para Computadores

Engenharia de Software

Professor: Dr. Fernando William Cruz

Gustave A. Persijn – 190046091

Daniel Vinícius R. A. – 190026375

Lorrayne A. Cardozo - 190032863

Laboratório sobre criptografia simétrica e assimétrica.

1. Objetivo

Neste projeto de laboratório, os alunos têm como objetivo criar um diálogo entre Alice e Bob para compartilhar um fractal por meio de técnicas de criptografia simétrica e assimétrica no corpo do arquivo BMP. Serão implementados os algoritmos DES (Data Encryption Standard) e RSA para garantir a segurança dos dados durante a transmissão. Alice será responsável pela geração e envio da chave de criptografia para Bob, que conseguirá visualizar a imagem BMP tanto como criptograma quanto como texto claro. O laboratório busca aprimorar os conhecimentos dos alunos em criptografia e aplicá-los em um cenário prático de comunicação segura.

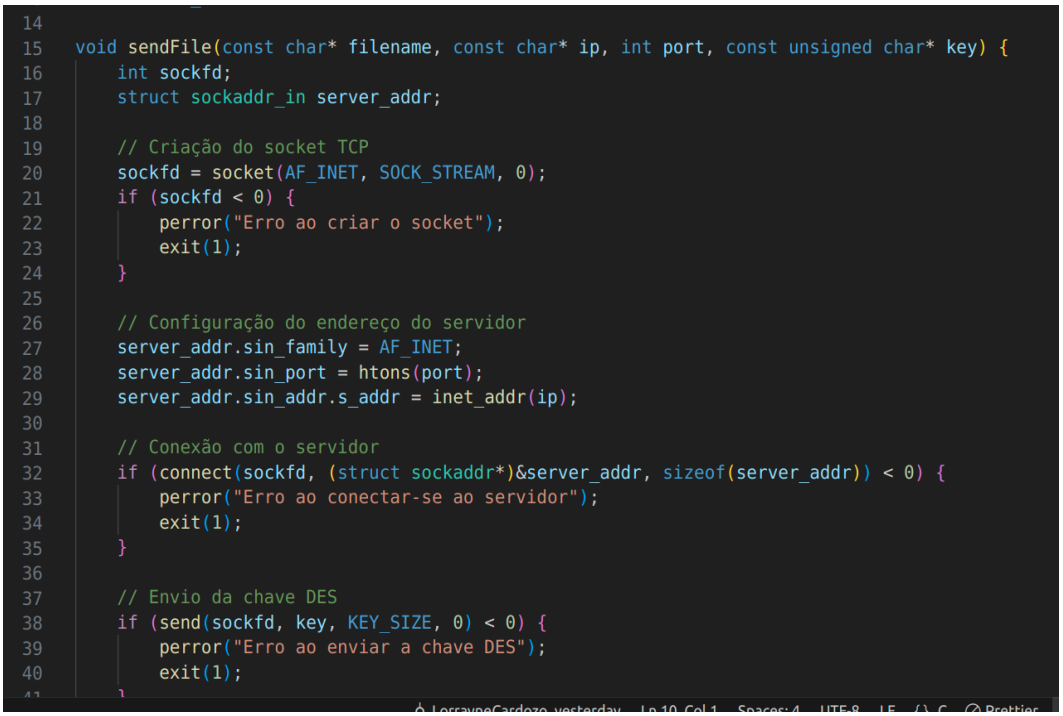
2. Criptografia Simétrica

Para implementar a criptografia simétrica utilizando o algoritmo DES e o diálogo TCP entre Alice e Bob, foram criados dois programas em C: **alice_client.c** e **bob_server.c**. O primeiro é responsável por enviar o arquivo fractaljulia.bmp para o servidor, enquanto o segundo recebe o arquivo criptografado, descriptografa-o e salva a versão original em um arquivo.

- Alice_client.c:

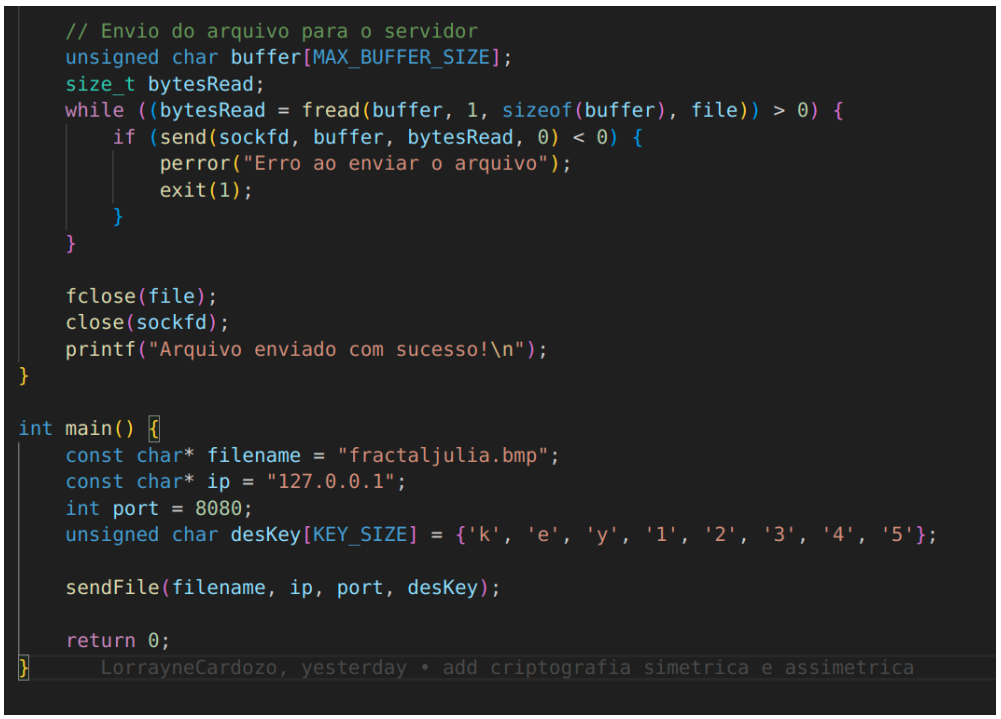
A função sendFile() é responsável por enviar o arquivo BMP para Bob através do socket TCP estabelecido. O arquivo é lido em chunks de até 1024 bytes e enviado para o servidor. Nele também faz a conexão via socket. Então,

primeiramente há a criação do socket TCP, posteriormente a configuração do endereço do servidor, conexão do servidor e o envio da chave DES.



```
14
15  void sendFile(const char* filename, const char* ip, int port, const unsigned char* key) {
16      int sockfd;
17      struct sockaddr_in server_addr;
18
19      // Criação do socket TCP
20      sockfd = socket(AF_INET, SOCK_STREAM, 0);
21      if (sockfd < 0) {
22          perror("Erro ao criar o socket");
23          exit(1);
24      }
25
26      // Configuração do endereço do servidor
27      server_addr.sin_family = AF_INET;
28      server_addr.sin_port = htons(port);
29      server_addr.sin_addr.s_addr = inet_addr(ip);
30
31      // Conexão com o servidor
32      if (connect(sockfd, (struct sockaddr*)&server_addr, sizeof(server_addr)) < 0) {
33          perror("Erro ao conectar-se ao servidor");
34          exit(1);
35      }
36
37      // Envio da chave DES
38      if (send(sockfd, key, KEY_SIZE, 0) < 0) {
39          perror("Erro ao enviar a chave DES");
40          exit(1);
41      }
```

A chave DES utilizada para criptografar o arquivo é definida como um array de 8 bytes (unsigned char desKey[KEY_SIZE]). Essa chave é enviada para o servidor antes do arquivo, garantindo que ambos os lados possuam a mesma chave para a criptografia e descriptografia.



```
    // Envio do arquivo para o servidor
    unsigned char buffer[MAX_BUFFER_SIZE];
    size_t bytesRead;
    while ((bytesRead = fread(buffer, 1, sizeof(buffer), file)) > 0) {
        if (send(sockfd, buffer, bytesRead, 0) < 0) {
            perror("Erro ao enviar o arquivo");
            exit(1);
        }
    }

    fclose(file);
    close(sockfd);
    printf("Arquivo enviado com sucesso!\n");
}

int main() {
    const char* filename = "fractaljulia.bmp";
    const char* ip = "127.0.0.1";
    int port = 8080;
    unsigned char desKey[KEY_SIZE] = {'k', 'e', 'y', '1', '2', '3', '4', '5'};

    sendFile(filename, ip, port, desKey);

    return 0;
}      LorrayneCardozo, yesterday • add criptografia simetrica e assimetrica
```

- **Bob_server.c:**

O server recebe um arquivo BMP criptografado do cliente, descriptografa-o e salva tanto o criptograma quanto o texto claro (sem criptografia). Ele utiliza o algoritmo DES (Data Encryption Standard) para realizar a criptografia simétrica do arquivo.

A função receiveFile() é responsável por receber o arquivo enviado por Alice. O arquivo é salvo como 'fractaljulia_received.bmp'.

```
// Função para receber o arquivo do cliente
void receiveFile(int sockfd, const char* filename) {
    // Definir o nome do arquivo para salvar o arquivo recebido com o nome "fractaljulia_received.bmp
    const char* receivedFilename = "fractaljulia_received.bmp";

    FILE* file = fopen(receivedFilename, "wb");
    if (file == NULL) {
        perror("Erro ao abrir o arquivo para escrita");
        exit(1);
    }

    unsigned char buffer[MAX_BUFFER_SIZE];
    int bytesRead;
    while ((bytesRead = read(sockfd, buffer, sizeof(buffer))) > 0) {
        fwrite(buffer, 1, bytesRead, file);
    }
    if (bytesRead < 0) {
        perror("Erro ao receber o arquivo");
        exit(1);
    }

    fclose(file);
    printf("Arquivo recebido com sucesso: %s\n", receivedFilename);
}
```

Para criptografar e descriptografar o arquivo, são utilizadas as funções encryptFile() e decryptFile(), respectivamente. Ambas as funções utilizam o modo CBC (Cipher Block Chaining) para garantir a segurança da criptografia e descriptografia dos dados.

```
// Copiar o cabeçalho BMP para o arquivo de saída
fseek(inputFile, 0, SEEK_SET);
fread(inputBuffer, 1, BMP_HEADER_SIZE, inputFile);
fwrite(inputBuffer, 1, BMP_HEADER_SIZE, outputFile);

// Inicializar o vetor de inicialização para CBC
DES_cblock ivec;
memset(&ivec, 0, sizeof(ivec));

size_t bytesRead;
while ((bytesRead = fread(inputBuffer, 1, sizeof(inputBuffer), inputFile)) > 0) {
    // Criptografar usando CBC (Cipher Block Chaining)
    DES_ncbc_encrypt(inputBuffer, outputBuffer, bytesRead, &keySchedule, &ivec, DES_ENCRYPT);
    fwrite(outputBuffer, 1, bytesRead, outputFile);
}

fclose(inputFile);
fclose(outputFile);
```

```c
// Ler e copiar o cabeçalho BMP para o arquivo de saída
fseek(inputFile, 0, SEEK_SET);
fread(outputBuffer, 1, BMP_HEADER_SIZE, inputFile);
fwrite(outputBuffer, 1, BMP_HEADER_SIZE, outputFile);

// Inicializar o vetor de inicialização para CBC
DES_cblock ivec;
memset(&ivec, 0, sizeof(ivec));

size_t bytesRead;
while ((bytesRead = fread(inputBuffer, 1, sizeof(inputBuffer), inputFile)) > 0) {
    // Descriptografar usando CBC (Cipher Block Chaining)
    DES_ncbc_encrypt(inputBuffer, outputBuffer, bytesRead, &keySchedule, &ivec, DES_DECRYPT);
    fwrite(outputBuffer, 1, bytesRead, outputFile);
}

fclose(inputFile);
fclose(outputFile);
```

```c
// Criptografia do arquivo recebido
const char* encryptedFilename = "fractaljulia_encriptado.bmp";
encryptFile(receivedFilename, encryptedFilename, desKey);

// Desencriptação do arquivo
const char* decryptedFilename = "fractaljulia_desencriptado.bmp";
decryptFile(encryptedFilename, decryptedFilename, desKey);
```

3. Criptografia Assimétrica

- server.c:

1. Geração dos números primos:

A função generatePrimesToFile() é chamada, ela gera dois números primos aleatórios entre um intervalo específico e os salva no arquivo primos.txt.

```c
int generateRandomPrime(int minDigits, int maxDigits) {
    int prime = 0;

    while (1) {
        prime = (rand() % (maxDigits - minDigits + 1)) + minDigits;

        int isPrime = 1;
        for (int i = 2; i <= prime / 2; ++i) {
            if (prime % i == 0) {
                isPrime = 0;
                break;
            }
        }

        if (isPrime) {
            break;
        }
    }

    return prime;      LorrayneCardozo, yesterday • add criptografia s
}

void generatePrimesToFile(const char* filename) {
    FILE* file = fopen(filename, "w");
```

2. Leitura dos números primos:

A função readPrimesFromFile() é chamada para ler os números primos (p e q) do arquivo primos.txt e armazená-los nas variáveis p e q.

```
void readPrimesFromFile(const char* filename, int* p, int* q) {
    FILE* file = fopen(filename, "r");
    if (file == NULL) {
        printf("Erro ao abrir o arquivo %s.\n", filename);
        return;
    }

    fscanf(file, "%d#%d", p, q);
    fclose(file);
}
```

3. Criação das chaves pública e privada:

A função createKeys() é chamada, ela recebe os números primos p e q como entrada e gera as chaves pública e privada usando o algoritmo RSA. A chave pública (e) é salva no arquivo "chave.pub" e a chave privada (d) é salva no arquivo "chave.priv".

```
void createKeys(int p, int q) {
    int n = p * q;
    int phi = (p - 1) * (q - 1);

    int e;
    for (e = 2; e < phi; e++) {
        if (gcd(e, phi) == 1) {
            break;
        }
    }

    int d = modInverse(e, phi);

    saveToFile("chave.pub", e);
    saveToFile("chave.priv", d);
}
```

4. Espera por conexões do cliente:

O servidor cria um socket e fica esperando por conexões de clientes através da função accept(). Quando uma conexão é estabelecida com um cliente, o servidor aceita a conexão através de accept() e recebe o arquivo encriptado enviado pelo cliente.

```
fd = socket(AF_INET, SOCK_STREAM, 0);
printf("Socket created\n");

memset(&serv_addr, '0', sizeof(serv_addr));
memset(buff, '0', sizeof(buff));

serv_addr.sin_family = AF_INET;
serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
serv_addr.sin_port = htons(5000);      DanielViniciusAlves, 7

bind(fd, (struct sockaddr*)&serv_addr, sizeof(serv_addr));
listen(fd, 10);

while(1){
    confd = accept(fd, (struct sockaddr*)NULL, NULL);
    if (confd==-1) {
        perror("Accept");
        continue;
    }
    FILE* fp = fopen( "fractal_received.bmp", "wb");
    tot=0;
    if(fp != NULL){
        while( (b = recv(confd, buff, 1024,0))> 0 ) {
            tot+=b;
            fwrite(buff, 1, b, fp);
```

5. Desencriptação do arquivo recebido:

O arquivo encriptado recebido é desencriptado usando a chave privada p * q através da função decryptFile(). A desencriptação é feita aplicando novamente o mesmo algoritmo de encriptação com o XOR.

```
void decryptFile(const char* inputFile, const char* outputFile, int key) {
    encryptFile(inputFile, outputFile, key); // A desencriptação é o mesmo processo da encriptação
}
```

6. Salvando o arquivo desencriptado:

O arquivo desencriptado é salvo como "decrypted.bmp" usando a função save_bmp_file().

```c
void save_bmp_file(char *header, char *buffer, int size) {
    FILE *file = fopen("received_image.bmp", "wb");
    if (file != NULL) {
        fwrite(header, sizeof(char), HEADER_SIZE, file);
        fwrite(buffer, sizeof(char), size, file);
        fclose(file);
        printf("Arquivo .bmp recebido e salvo com sucesso!\n");
    } else {
        perror("Erro ao salvar o arquivo");
    }
}
```

- **client.c**

1. Conexão com o servidor:

O cliente cria um socket e tenta estabelecer uma conexão com o servidor usando a função connect(). O servidor deve estar em execução e aguardando conexões.

```c
b=connect(sfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr));
if (b==-1) {
    perror("Connect");
    return 1;
}
```

2. Abertura do arquivo BMP:

O cliente abre o arquivo BMP "fractaljulia.bmp" em modo de leitura binária ("rb"). Esse arquivo conterá o fractal que será enviado ao servidor.

```c
    DanielViniciusAlves, 7 hours ago • Adicionando
FILE *fp = fopen("fractaljulia.bmp", "rb");
if(fp == NULL){
    perror("File");
    return 2;
}
```

3. Envio do arquivo encriptado:

O cliente lê o conteúdo do arquivo BMP e o envia ao servidor através do socket, utilizando a função send() para enviar os dados.

```
while( (b = fread(sendbuffer, 1, sizeof(sendbuffer), fp))>0 ){
    send(sfd, sendbuffer, b, 0);
}
```

4. Conclusão

Neste laboratório, exploramos a aplicação prática da criptografia simétrica e assimétrica em um diálogo TCP entre Alice e Bob para compartilhar um fractal. Ao implementar os algoritmos DES e RSA, conseguimos garantir a segurança dos dados durante a transmissão, protegendo a imagem BMP de possíveis interceptações não autorizadas. A utilização cuidadosa da criptografia apenas no corpo do arquivo BMP permitiu que a imagem mantivesse sua aparência visual inalterada, protegendo-a de detecção. A troca segura da chave de encriptação entre Alice e Bob assegurou que somente o destinatário pudesse visualizar a imagem descriptografada. Ao concluir este laboratório, os alunos aprimoraram seus conhecimentos em criptografia e puderam aplicá-los em um contexto real de comunicação segura, fortalecendo suas habilidades na área de segurança da informação.

5. Autoavaliação

- **Lorrayne Alves Cardozo:** Aprendi sobre criptografia simétrica e assimétrica, troca segura de chaves e importância da proteção de informações sensíveis. 10.

- **Gustave Augusto Persijn:** Compreendi a implementação dos algoritmos DES e RSA, garantindo confidencialidade dos dados em comunicações. 10.

- **Daniel Vinicius Alves:** Adquiri conhecimentos sobre DES, RSA e troca segura de chaves, reforçando a importância da criptografia na segurança de dados. 10.

6. Referências Bibliográficas

TANENBAUM, Andrew S.; WETHERALL, David. Redes de Computadores. 5. ed. São Paulo: Pearson, 2012.