Criptografia RSA

Leonardo Lima Barbosa Pereira
Júlia Gonçalves dos Santos
Ricardo Pinto Cardoso Júnior
Matheus Almeida Souza

Criptografia RSA

Universidade Federal de Alagoas
Matemática discreta
AC Simões

Introdução

RSA(Rivest-Shamir-Adleman) é um dos percursores dos sistemas de criptografia de chave pública. Ela se baseia na dificuldade de fatorização do produto de dois números primos grandes(p e q), sendo ela necessária para a obtenção da chave privada usada na desencriptação da mensagem passada.

O RSA é um algoritmo relativamente lento e, por isso, é menos usado para criptografar diretamente os dados do usuário. Mais frequentemente, o RSA passa chaves criptografadas compartilhadas para criptografia de chave simétrica que, por sua vez, pode executar operações de criptografia-descriptografia em massa a uma velocidade muito maior.

Criptografia RSA

Universidade Federal de Alagoas
Matemática discreta
AC Simões

Requisitos do projeto

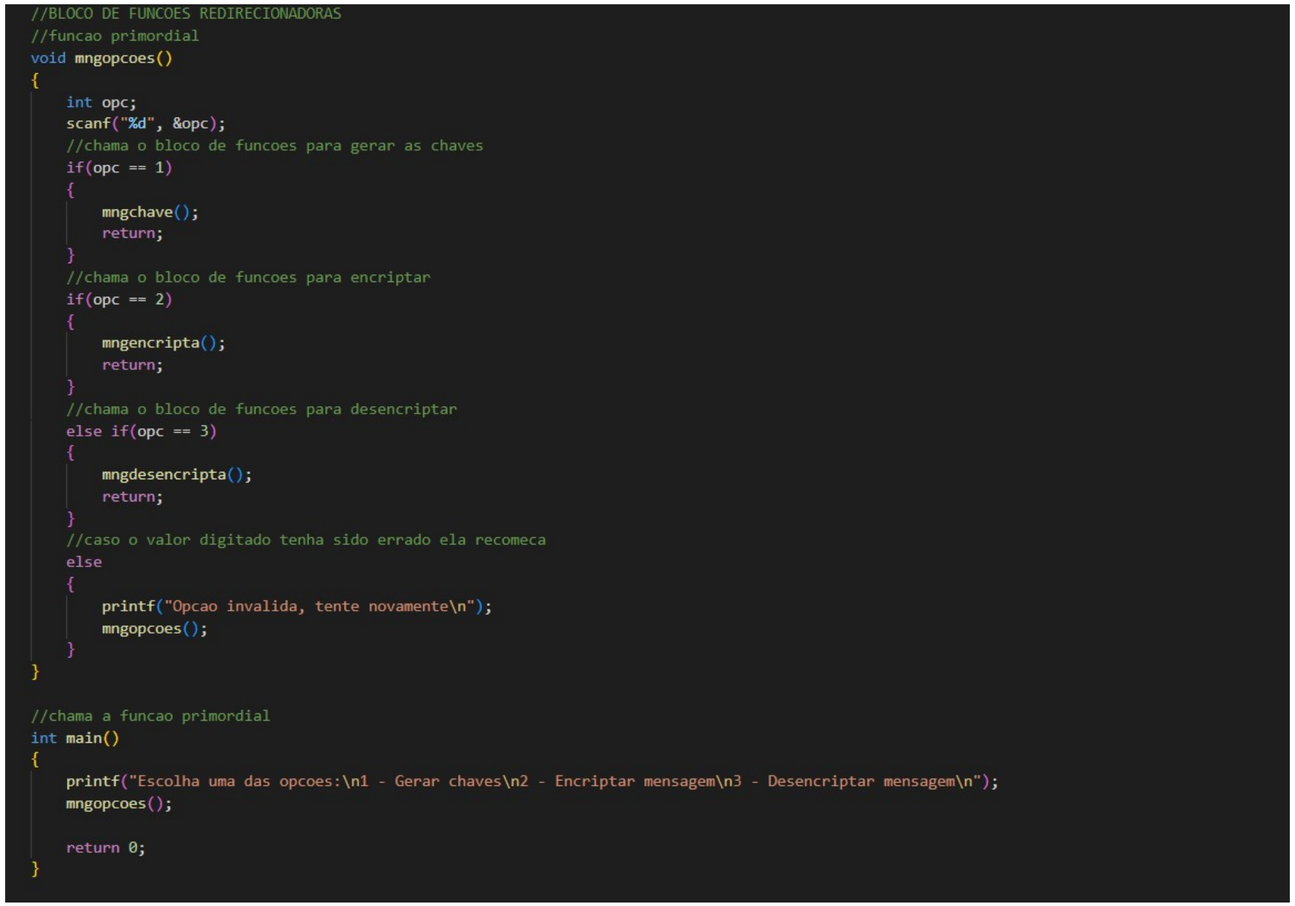
- Desenvolver um programa capaz de aplicar o RSA
- Gerar chave pública
- Encriptar
- Desencriptar
- Utilização do alfabeto de A - Z com seus valores de 2 a 27 e espaço = 28

Dinâmica do RSA:

- O usuário gera uma chave pública composta por ('P', 'Q' e 'E'). Sendo "p" e "q" numeros primos e "e" coprimo de (p -1) *(q - 1))
- É passada a mensagem desejada seguida da chave pública gerada anteriormente, a encriptação é feita e os dados serão gravados em um arquivo txt
- O usuário digita a chave pública, a partir dela é gerada a chave privada, logo após é executada a desencriptação

Criptografia RSA

Universidade Federal de Alagoas
Matemática discreta
AC Simões



```c
//BLOCO DE FUNCOES REDIRECIONADORAS
//funcao primordial
void mngopcoes()
{
    int opc;
    scanf("%d", &opc);
    //chama o bloco de funcoes para gerar as chaves
    if(opc == 1)
    {
        mngchave();
        return;
    }
    //chama o bloco de funcoes para encriptar
    if(opc == 2)
    {
        mngencripta();
        return;
    }
    //chama o bloco de funcoes para desencriptar
    else if(opc == 3)
    {
        mngdesencripta();
        return;
    }
    //caso o valor digitado tenha sido errado ela recomeca
    else
    {
        printf("Opcao invalida, tente novamente\n");
        mngopcoes();
    }
}

//chama a funcao primordial
int main()
{
    printf("Escolha uma das opcoes:\n1 - Gerar chaves\n2 - Encriptar mensagem\n3 - Desencriptar mensagem\n");
    mngopcoes();

    return 0;
}
```

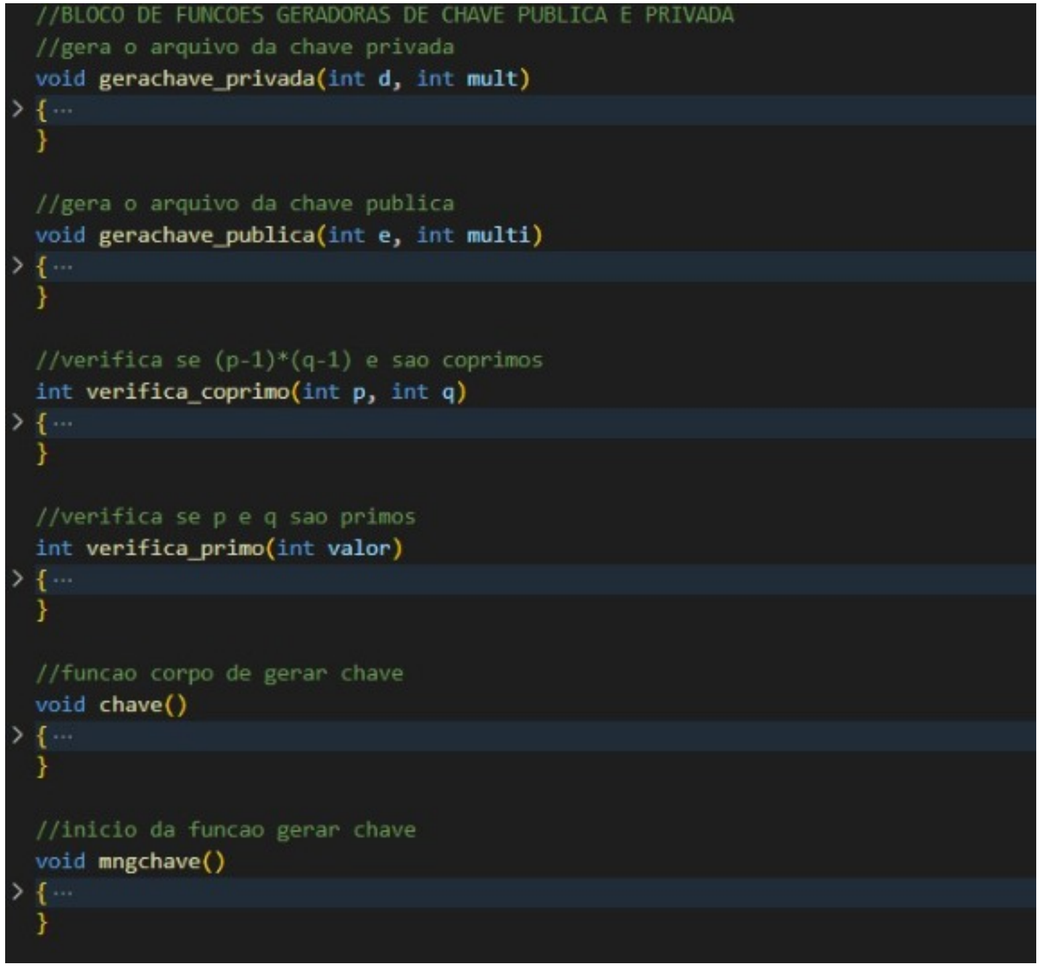
Funções redirecionadoras

Servem para encaminhar o usuário para as demais funcionalidades do programa, sendo elas:

1 - Gerar chave pública
2 - Encriptar uma mensagem
3 - Desencriptar uma mensagem

Criptografia RSA

Universidade Federal de Alagoas
Matemática discreta
AC Simões



```
//BLOCO DE FUNCOES GERADORAS DE CHAVE PUBLICA E PRIVADA
//gera o arquivo da chave privada
void gerachave_privada(int d, int mult)
{...
}

//gera o arquivo da chave publica
void gerachave_publica(int e, int multi)
{...
}

//verifica se (p-1)*(q-1) e sao coprimos
int verifica_coprimo(int p, int q)
{...
}

//verifica se p e q sao primos
int verifica_primo(int valor)
{...
}

//funcao corpo de gerar chave
void chave()
{...
}

//inicio da funcao gerar chave
void mngchave()
{...
}
```

Funções que geram a chave pública e privada

Certifica-se que todos os critérios necessários para os valores da chave sejam atendidos:

1 - Verifica os numeros primos
2 - Verifica o coprimo
(utiliza euclides)
3- salva os resultados em um txt

Criptografia RSA

Universidade Federal de Alagoas
Matemática discreta
AC Simões

```c
//gera o arquivo da chave publica
void gerachave_publica(int e, int multi)
{
    //Cria um ponteiro para o arquivo
    FILE *pont_num;

    //Abrindo ou criando o arquivo no modo 'w' de escrita
    pont_num = fopen("chave_publica.txt", "w");

    //Testando se realmente o arquivo foi criado
    if(pont_num == NULL)
    {
        printf("Erro na abertura/criacao do arquivo!\nO programa sera fechado\n");
        return;
    }

    //Usando o fprintf para gravar no txt
    fprintf(pont_num, "%d %d", e, multi);

    //Usando o fclose para fechar o arquivo
    fclose(pont_num);

    printf("Chave publica: %d %d\nDados gravados com sucesso em: chave_publica.txt!\n", e, multi);
    return;

}
```

Gera o arquivo txt através de ponteiro
Verifica se houve a criação do arquivo

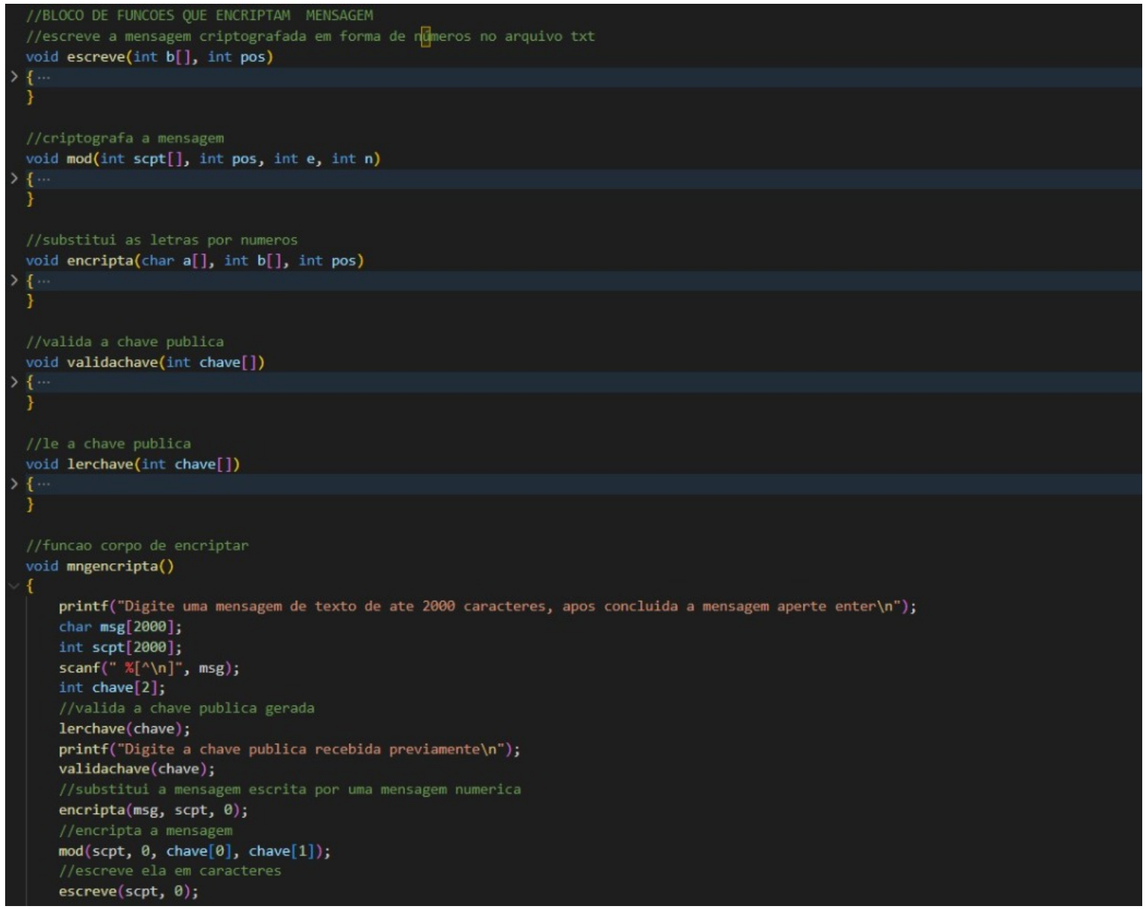
Se houver problema, será apresentado:

"Erro na abertura/criação do arquivo!
O programa sera fechado"

E é redirecionado ao inicio.

Criptografia RSA

Universidade Federal de Alagoas
Matemática discreta
AC Simões



```
//BLOCO DE FUNCOES QUE ENCRIPTAM  MENSAGEM
//escreve a mensagem criptografada em forma de números no arquivo txt
void escreve(int b[], int pos)
{...
}

//criptografa a mensagem
void mod(int scpt[], int pos, int e, int n)
{...
}

//substitui as letras por numeros
void encripta(char a[], int b[], int pos)
{...
}

//valida a chave publica
void validachave(int chave[])
{...
}

//le a chave publica
void lerchave(int chave[])
{...
}

//funcao corpo de encriptar
void mngencripta()
{
    printf("Digite uma mensagem de texto de ate 2000 caracteres, apos concluida a mensagem aperte enter\n");
    char msg[2000];
    int scpt[2000];
    scanf(" %[^\n]", msg);
    int chave[2];
    //valida a chave publica gerada
    lerchave(chave);
    printf("Digite a chave publica recebida previamente\n");
    validachave(chave);
    //substitui a mensagem escrita por uma mensagem numerica
    encripta(msg, scpt, 0);
    //encripta a mensagem
    mod(scpt, 0, chave[0], chave[1]);
    //escreve ela em caracteres
    escreve(scpt, 0);
```

Funções que encriptam a mensagem

Recebe a mensagem do usuário de até 2 mil caracteres e aplica a chave pública para a encriptação:

Encripta utilizando exponciação modular rápida

verifica os valores recebidos

Criptografia RSA

Universidade Federal de Alagoas
Matemática discreta
AC Simões

Obtenção do script através do mod:

$$c = m \wedge e \mod (p*q)$$

c - cifra

m - valor numérico escolhido para cada letra

p , q , e - componentes da chave pública

Exemplo:

Chave = 17, 11 , 7

Letra = A (2)

$c = 2 \wedge 7 \mod (17*11)$

c = 128

Criptografia RSA

Universidade Federal de Alagoas
Matemática discreta
AC Simões

Exponenciação modular rápida:

- Implementa para numeros muito altos do expoente
- trasnforma em binário
- Divide o expoente em potências de 2

Como nós podemos calcular A^B mod C rapidamente para qualquer B ?

$$5^{117} \bmod 19$$

Passo 1: Divida B em potências de 2 escrevendo-o em binário

$$117 = 1110101 \text{ in binary}$$

Comece com o dígito mais à direita, sendo k = 0 **e** para **cada dígito**:

- **Se o dígito for 1, precisaremos de uma parte para 2 ^ k, caso contrário não precisaremos**
- **Some 1 em k e mova para a esquerda para o próximo dígito**

$$117 = (2^0 + 2^2 + 2^4 + 2^5 + 2^6)$$
$$117 = 1 + 4 + 16 + 32 + 64$$

$$5^{117} \bmod 19 = 5^{(1 + 4 + 16 + 32 + 64)} \bmod 19$$

$$5^{117} \bmod 19 = (5^1 * 5^4 * 5^{16} * 5^{32} * 5^{64}) \bmod 19$$

Criptografia RSA

Universidade Federal de Alagoas
Matemática discreta
AC Simões

Exponenciação modular rápida:

Passo 2: Calcule mod C das potências de dois ≤ B

5^1 mod 19 = 5

5^2 mod 19 = (5^1 * 5^1) mod 19 = (5^1 mod 19 * 5^1 mod 19) mod 19
5^2 mod 19 = (5 * 5) mod 19 = 25 mod 19
5^2 mod 19 = 6

5^4 mod 19 = (5^2 * 5^2) mod 19 = (5^2 mod 19 * 5^2 mod 19) mod 19
5^4 mod 19 = (6 * 6) mod 19 = 36 mod 19
5^4 mod 19 = 17

5^8 mod 19 = (5^4 * 5^4) mod 19 = (5^4 mod 19 * 5^4 mod 19) mod 19
5^8 mod 19 = (17 * 17) mod 19 = 289 mod 19
5^8 mod 19 = 4

5^16 mod 19 = (5^8 * 5^8) mod 19 = (5^8 mod 19 * 5^8 mod 19) mod 19
5^16 mod 19 = (4 * 4) mod 19 = 16 mod 19
5^16 mod 19 = 16

5^32 mod 19 = (5^16 * 5^16) mod 19 = (5^16 mod 19 * 5^16 mod 19) mod 19
5^32 mod 19 = (16 * 16) mod 19 = 256 mod 19
5^32 mod 19 = 9

5^64 mod 19 = (5^32 * 5^32) mod 19 = (5^32 mod 19 * 5^32 mod 19) mod 19
5^64 mod 19 = (9 * 9) mod 19 = 81 mod 19
5^64 mod 19 = 5

Etapa 3: Use as propriedades de multiplicação modular para combinar os valores calculados de mod C

5^117 mod 19 = (5^1 * 5^4 * 5^16 * 5^32 * 5^64) mod 19
5^117 mod 19 = (5^1 mod 19 * 5^4 mod 19 * 5^16 mod 19 * 5^32 mod 19 * 5^64 mod 19) mod 19
5^117 mod 19 = (5 * 17 * 16 * 9 * 5) mod 19
5^117 mod 19 = 61200 mod 19 = 1
5^117 mod 19 = 1

Notas:

Existem outras técnicas de otimização, mas estão fora do escopo deste artigo. Deve-se notar que ao realizamos a exponenciação modular em criptografia, não é incomum usar expoentes de B > 1000 bits.

Criptografia RSA

Universidade Federal de Alagoas
Matemática discreta
AC Simões

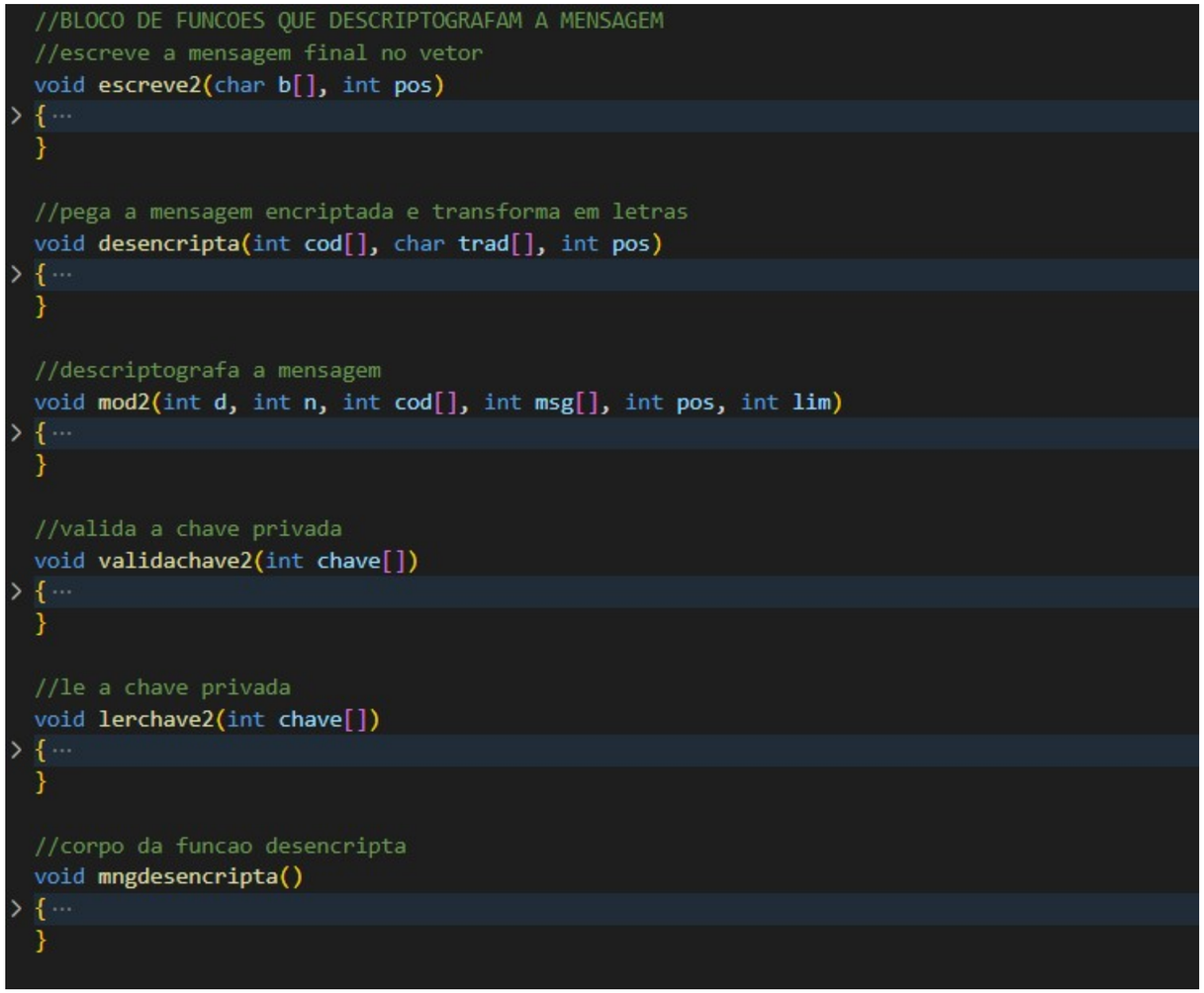

```c
//BLOCO DE FUNCOES QUE ENCRIPTAM  MENSAGEM
//escreve a mensagem criptografada em forma de numeros no arquivo txt
void escreve(int b[], int pos)
{
    //Cria um ponteiro para o arquivo
    FILE *pont_msg;

    //Abrindo ou criando o arquivo no modo 'w' de escrita
    pont_msg = fopen("script.txt", "a");

    //Testando se realmente o arquivo foi criado
    if(pont_msg == NULL)
    {
        printf("Erro na abertura/criacao do arquivo!\nO programa sera fechado\n");
        return;
    }

    if(b[pos] != -1)
    {
        //Usando o fprintf para gravar no txt
        if(b[pos+1] != -1)
        {
            fprintf(pont_msg, "%d ", b[pos]);
        }
        else
        {
            fprintf(pont_msg, "%d ", b[pos]);
        }

        escreve(b, pos+1);
    }

    else
    {
        //Usando o fclose para fechar o arquivo
        fclose(pont_msg);

        printf("Dados gravados com sucesso em: script.txt!\n");
        return;
    }
}
```

Escrevendo o script no txt:

1 - Cria um ponteiro para o arquivo
2- Abre/cria o arquivo
3 - Testa se realmente foi criado
4 - Usa fprintf para gravar a mensagem numerica no txt

Criptografia RSA

Universidade Federal de Alagoas
Matemática discreta
AC Simões



```
//BLOCO DE FUNCOES QUE DESCRIPTOGRAFAM A MENSAGEM
//escreve a mensagem final no vetor
void escreve2(char b[], int pos)
{...
}

//pega a mensagem encriptada e transforma em letras
void desencripta(int cod[], char trad[], int pos)
{...
}

//descriptografa a mensagem
void mod2(int d, int n, int cod[], int msg[], int pos, int lim)
{...
}

//valida a chave privada
void validachave2(int chave[])
{...
}

//le a chave privada
void lerchave2(int chave[])
{...
}

//corpo da funcao desencripta
void mngdesencripta()
{...
}
```

Funções que desencriptam a mensagem:

Faz a leitura da chave privada e traduz o script:

1 - lê a chave privada e checa sua validação
2 - pega a mensagem encriptografada e trasnforma em letras
3 - descriptografa a mensagem

Criptografia RSA

Universidade Federal de Alagoas
Matemática discreta
AC Simões

Desencriptando os valores do txt:

$$decrypt = c \wedge d \mod (p*q)$$

c - número do txt

d - chave privada

p , q - elementos da chave pública

Exemplo:

Chave pública: 17, 11, 7

Chave privada: 23

Número a ser traduzido: 128

decrypt = 128 ^ 23 mod 187

decrypt = 2

Cifra de César -> 2 = A

Resultado = A

Criptografia RSA

Universidade Federal de Alagoas
Matemática discreta
AC Simões

Escrevendo no txt e encerrando o programa:

```c
//escreve a mensagem final no vetor
void escreve2(char b[], int pos)
{
    //Cria um ponteiro para o arquivo
    FILE *pont_trad;

    //Abrindo ou criando o arquivo no modo 'a' de escrita
    pont_trad = fopen("mensagem.txt", "a");

    //Testando se realmente o arquivo foi criado
    if(pont_trad == NULL)
    {
        printf("Erro na abertura/criacao do arquivo!\nO programa sera fechado\n");
        return;
    }

    if(b[pos] == 'a' || b[pos] == 'b' || b[pos] == 'c' || b[pos] == 'd' || b[pos] == 'e' || b[pos] == 'f' || b[pos] == 'g' ||
    b[pos] == 'h' || b[pos] == 'i' || b[pos] == 'j' || b[pos] == 'k' || b[pos] == 'l' || b[pos] == 'm' || b[pos] == 'n' ||
    b[pos] == 'o' || b[pos] == 'p' || b[pos] == 'q' || b[pos] == 'r' || b[pos] == 's' || b[pos] == 't' || b[pos] == 'u' ||
    b[pos] == 'v' || b[pos] == 'w' || b[pos] == 'x' || b[pos] == 'y' || b[pos] == 'z')
    {
        //Usando o fprintf para gravar no txt
        fprintf(pont_trad, "%c", b[pos]);

        escreve2(b, pos+1);
    }
    else if(b[pos] == ' ')
    {
        fprintf(pont_trad, "%c", b[pos]);
        escreve2(b, pos+1);
    }
    else
    {
        //Usando o fclose para fechar o arquivo
        fclose(pont_trad);

        printf("Dados gravados com sucesso em: mensagem.txt!\n");
        return;
    }
}
```

1 - Cria um ponteiro para o arquivo
2 - abre/cria o arquivo
3 - testa de realmente foi executado
4 - grava a mensagem no arquivo
5 - fecha o arquivo
6 - imprime: "Dados grvados com sucesso em: *nome do arquivo*!"

Universidade Federal de Alagoas
Matemática discreta
AC Simões

Considerações finais

Referências

Sousa, R. Entendendo algoritmo RSA (de verdade).hackingnaweb,2020.Disponível em: https://hackingnaweb.com/criptografia/entendendo-algoritmo-rsa-de-verdade/

Exponenciação modular rápida.khanacademy,2019. Disponível em:hhttps://pt.khanacademy.org/computing/computer-science/cryptography/modarithmetic/a/fast-modular-exponentiation