Trabalho sobre Criptoanálise

CCF - 330

Eduardo T. Tristão (4219),
Luís Henrique S. de Carvalho (4254),
Marcus Vinícius G. Ribeiro (4240)

Instituto de Ciências Exatas e Tecnológicas
Universidade Federal de Viçosa - Campus Florestal (UFV)
Florestal - MG - Brasil

Sumário

1. Introdução

O trabalho tem como objetivo desenvolver, em linguagem C, um algoritmo para manipulação de um texto criptografado utilizando-se de métodos descodificadores para revelar o texto decifrado bem como a chave de criptografia e outros dados relativos ao texto. Para cada grupo foi designado um texto diferente com chaves de criptografia distintas.

2. Algoritmo

2.1. Funções

2.1.1 Função SetColor

A função "SetColor" tem como objetivo modificar a cor das letras trocadas para verde após a execução da função "mostrarEstado" que mostra a versão parcialmente decifrada do código.

```c
void SetColor(int ForgC){ //modifica a cor do texto
    WORD wColor;
    //This handle is needed to get the current background attribute
    HANDLE hStdOut = GetStdHandle(STD_OUTPUT_HANDLE);
    CONSOLE_SCREEN_BUFFER_INFO csbi;
    //csbi is used for wAttributes word

    if(GetConsoleScreenBufferInfo(hStdOut, &csbi)){
        //To mask out all but the background attribute, and to add the color
        wColor = (csbi.wAttributes & 0xF0) + (ForgC & 0x0F);
        SetConsoleTextAttribute(hStdOut, wColor);
    }
}
```

Figura 1. Função SetColor

2.1.2. Função descodificaChar

A finalidade da função "descodificaChar" é alterar o texto original de acordo com a chave de decodificação durante a execução da função "mostrarEstado".

```c
char descodificaChar(char * chave, char c){
    if (c >= 65 && c <= 90){
        char retorno = 0;
        for (int i = 0; i < 26; i++){
            if (chave[i] == c)
                retorno = i+65;
        }
        return retorno;
    }
    else return c;
}
```

Figura 2. Função descodificaChar

2.1.3. Função leArq

Esta função lê caractere por caractere do texto contido no arquivo de entrada e o adiciona em um vetor para manipulação futura.

```c
void leArq(char* frase,char * entrada){
    FILE* ptr;
    char ch;
    int q=0;
    ptr = fopen(entrada,  _Mode: "r");
    if (NULL == ptr) {
        printf( _Format: "arquivo nao pode ser aberto \n");
    }
    do {
        ch = fgetc(ptr);
        frase[q] = ch;
        q++;
    } while (ch != EOF);
    frase[q]= '\0';
    fclose(ptr);
}
```

Figura 3. Função leArq

2.1.4. Função menu

A função “menu” fica responsável somente por imprimir as opções do programa e coletar a escolha do usuário.

```c
int menu(){ //printa o menu e retorna a opção escolhida
    printf( _Format: "Digite o numero de acordo com a opcao desejada:\n");
    printf( _Format: "1. Apresentar o estado atual da criptoanalise;\n");
    printf( _Format: "2. Fazer analise de frequencia no texto criptografado;\n");
    printf( _Format: "3. Realizar casamento exato de caracteres no texto criptografado;\n");
    printf( _Format: "4. Realizar casamento aproximado de caracteres no texto parcialmente decifrado;\n");
    printf( _Format: "5. Alterar chave de criptografia;\n");
    printf( _Format: "6. Exportar resultado e encerrar o programa.\n");
    int r;
    scanf( _Format: "  %d",&r);
    return r;
}
```

Figura 4. Função menu

2.1.5. Função chaveToString

Esta função fica responsável por transformar a chave decodificadora para o formato de string para sua exibição durante a função “mostrarEstado”. Para isso, é alocado dinamicamente um vetor de caracteres com tamanho para armazenar dois alfabetos de 26 letras. Inicialmente ele armazenará o alfabeto da língua portuguesa em ordem e então analisará cada caractere até então presente na chave de decodificação. Caso exista, esse caractere será concatenado ao final do vetor. Caso contrário será adicionado um espaço vazio.

```c
char * chaveToString(char * chave){ //transforma a chave numa string
    char * chaveS = (char *) malloc( _Size: sizeof(char)*52);
    strcpy (chaveS, _Source: "ABCDEFGHIJKLMNOPQRSTUVWXYZ\n");
    char str[2] = "\0";
    for (int i = 0; i < 26; i++){
        if (chave[i] != 0)
            str[0] = chave[i];
        else str[0] = ' ';
        strcat(chaveS, str);
    }
    return chaveS;
}
```

Figura 5. Função chaveToString

2.1.6. Função mostrarEstado

A função “mostrarEstado” apresenta o atual estado de codificação do texto de entrada. Em um primeiro momento ela exibe o texto original sem nenhuma alteração. Após isso, apresenta o alfabeto e o mapeamento de cada letra para a chave de criptografia. Em última instância, é exibido o texto parcialmente decifrado seguindo o padrão de decodificação ditado pela chave alterando os caracteres decifrados para a cor verde.

```c
void mostrarEstado(char * frase, char * chave){ //mostra o estado atual
    printf( _Format: "=== Texto criptografado ===\n");
    printf( _Format: "%s\n",frase); //printa o texto
    printf( _Format: "=== Chave ===\n");
    printf( _Format: "%s\n\n", chaveToString(chave)); //printa a chave
    printf( _Format: "=== Texto parcialmente decifrado ===\n");
    for (int i = 0; i < strlen(frase); i++){...}
    SetColor( ForgC: 7);
    printf( _Format: "\n");
}
```

Figura 6. Função mostrarEstado

2.1.7. Função mostrarFrequencia

Esta função retorna a contagem de cada letra codificada no texto bem como sua frequência relativa ao próprio texto. Para isso, todo o texto é percorrido caractere por caractere (Figura 8)e é feita então a contagem das letras. Além disso, a função também apresenta a frequência com que as letras do alfabeto português aparecem em relação às palavras do próprio idioma (Figura 7).

Letra	Freqüência		Letra	Freqüência
A	14.63%		N	5.05%
B	1.04%		O	10.73%
C	3.88%		P	2.52%
D	4.99%		Q	1.20%
E	12.57%		R	6.53%
F	1.02%		S	7.81%
G	1.30%		T	4.34%
H	1.28%		U	4.63%
I	6.18%		V	1.67%
J	0.40%		W	0.01%
K	0.02%		X	0.21%
L	2.78%		Y	0.01%
M	4.74%		Z	0.47%

Figura 7 . Frequência relativa entre letras do alfabeto portugês e suas aparições em palavras

```c
for (int i = 0; i < strlen(frase); i++){ //soma a quantidade de cada letra
    if (frase[i]-65 >= 0 && frase[i]-65 < 26)
        qnts[frase[i]-65] += 1;
    total += 1;
}

printf( _Format: "Letra, Cont., Freq., No portugues: Letra, Freq.\n");
for (int i = 0; i < 26; i++){ //mostra a tabela com cada letra e suas frequencias
    int mi = 0;
    int mi2 = 0;
    for (int j = 1; j < 26; j++){
        if(qnts[mi] < qnts[j]){
            mi = j;
        }
        if(freq[mi2] < freq[j]){
            mi2 = j;
        }
    }
```

Figura 8. Fragmento da função mostrarFrequencia que faz a contagem das letras e definição das frequências

2.1.8. Função shiftAndPre

Esta função fica responsável pelo pré processamento das duas funções que utilizam o algoritmo Shift-And apenas gerando a máscara para os padrões

```
void shifAndPre(int mask[26], char * padrao){ //realiza o pre processamento das funções relativas a shift and
    for (int i = 0; i < 26; i++)
        mask[i] = 0;
    for (int i = 0; i < strlen(padrao); i++) //inicializa a mascara com seus valores adequados
        mask[padrao[i]-65] += pow( _X: 2, _Y: strlen(padrao) - i - 1);
}
```

Figura 2. Função shiftAndPre

2.1.9. Função shiftAnd

Para realizar um casamento exato no texto criptografado foi utilizado o algoritmo de casamento de caracteres Shift-And presente na função “shiftAnd”. Em um primeiro momento são criadas máscaras para cada letra do alfabeto iniciadas com valor 0 e então esse valor é atualizado de acordo com o padrão definido pelo usuário para a busca. Após isso todo o texto é percorrido e a cada letra detectada é realizado o procedimento do algoritmo Shift-And.

A variável “state” é responsável por armazenar os estados de comparação. Caso esta em algum momento seja ímpar, significa que o último bit do numeral que a representa é 1 o que representa um casamento de caracteres.

A função trabalha em alguns momentos sobre a codificação ASCII o que facilita a criação das máscaras bem como a identificação de caracteres especiais no texto como pontos finais e vírgulas. Ao final da função, é retornado o valor com a quantidade de vezes que o padrão foi encontrado.

```c
int shiftAnd(char * frase, char * padrao){ //detectar padrao com shift and

    int mask[26];

    shifAndPre(mask, padrao);

    int qnt = 0;
    int state = 0;
    for (int i = 0; i < strlen(frase); i++){ //cada passo do algoritmo shift and
        if(frase[i] >= 65 && frase[i] < 90){ //detecta se e letra
            state = state>>1; //shift
            state += pow( _X: 2, _Y: strlen(padrao)-1); //ligar o primeiro bit

            state = state & mask[frase[i]-65]; //and com a mascara

            if (state % 2 == 1) //verifica se o ultimo bit esta ligado
                qnt += 1;
        }
        else
            state = 0;
    }
    return qnt; //retorna a quantidade de vezes que encontrou o padrao
}
```

Figura 3. Função shiftAnd

2.1.10. Função shiftAndApx

A função “shiftAndApx” faz o casamento aproximado de caracteres sobre o texto parcialmente decifrado baseando-se no algoritmo Shift-And aproximado com suporte apenas para erros de substituição. Para isso, o usuário deve inserir apenas o padrão a ser analisado e o nível de tolerância. Ao ser executada, a função exerce trabalho bastante semelhante a função “shiftAnd”.

Diferente da função citada, caso a operação “and” entre duas palavras resulte em 0 e caso haja algum nível de tolerância, esta será desconsiderada assim continuando a operação até que não haja tolerância ou comece uma nova palavra. Caso a operação realize um casamento é informada ao usuário a ocorrência deste casamento no texto bem como a sua posição e também o número total das mesmas.

```c
if (state == 0){
    if (tol != 0){
        state += pow( _X: 2, _Y: strlen(padrao) - d);
        tol --;}}

d++;

if (state % 2 == 1){
    qnt += 1;
    printf( _Format: "@[%d, %d):", i-strlen(padrao), i);
    for (int j = strlen(padrao)-1; j > -1; --j) {
        printf( _Format: "%c", texto[i-j]);}
    printf( _Format: "\n");}
```

Figura 4. Partes da função responsáveis pela tolerância e exibição dos resultados

2.2. Chave de Criptografia

A chave de criptografia para a decodificação do código foi declarada como um vetor de caracteres de tamanho 26. Inicialmente esse vetor é preenchido com elementos nulos e a medida que o usuário altera o mapeamento esses espaços são preenchidos (figura 3). A alteração da chave é feita ao escolher a opção 5 do menu principal onde deve ser inserida a letra original e então a letra para a qual foi mapeada nesta ordem (figura 4).

```
char chave[26];
for (int i = 0; i < 26; i++)
    chave[i] = 0;
```

Figura 5. Declaração do vetor para a chave de criptografia

```
case 5:
    printf( _Format: "Informe a letra original, seguida da letra para a qual foi mapeada:\n> ");
    char a, b;
    scanf( _Format: " %c %c", &a, &b);
    chave[a-65] = b;
    printf( _Format: "Registrado: %c -> %c\n", a, b);
    break;
```

Figura 6. Operação para alteração da chave de criptografia

3. Execução

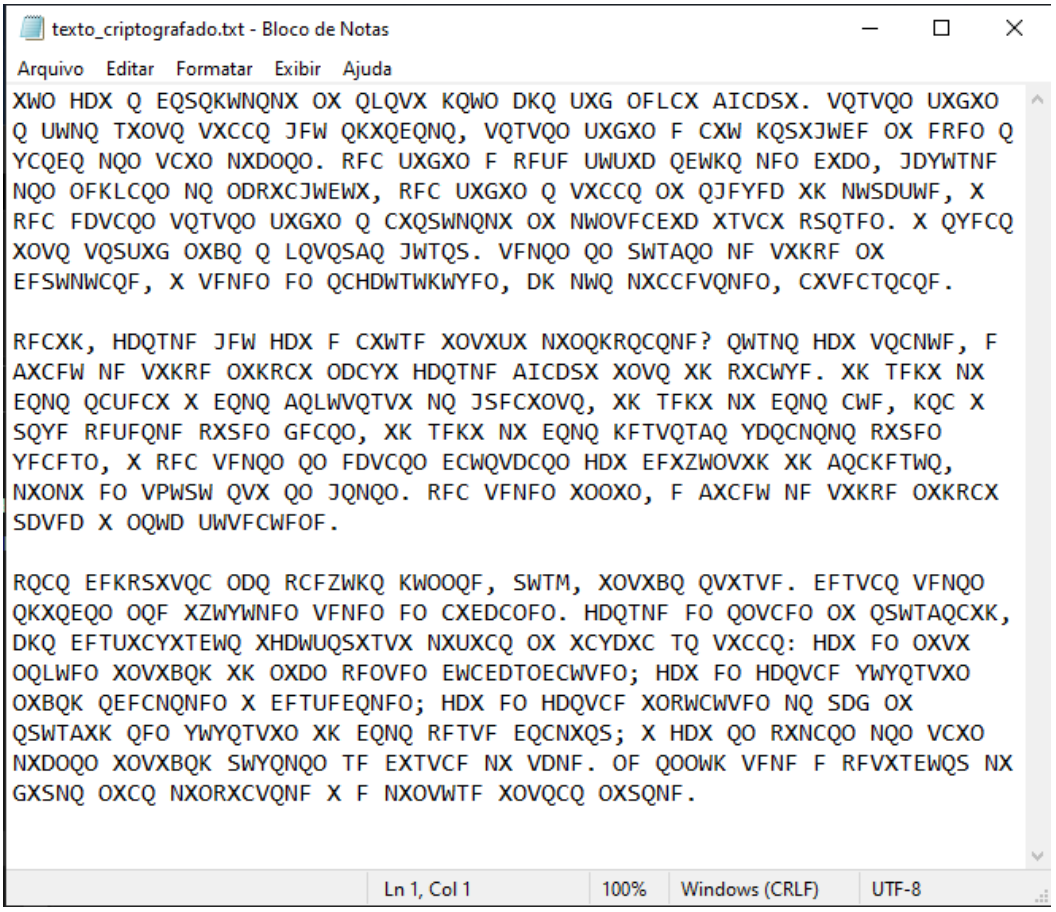

Figura 7. Arquivo de entrada com texto criptografado

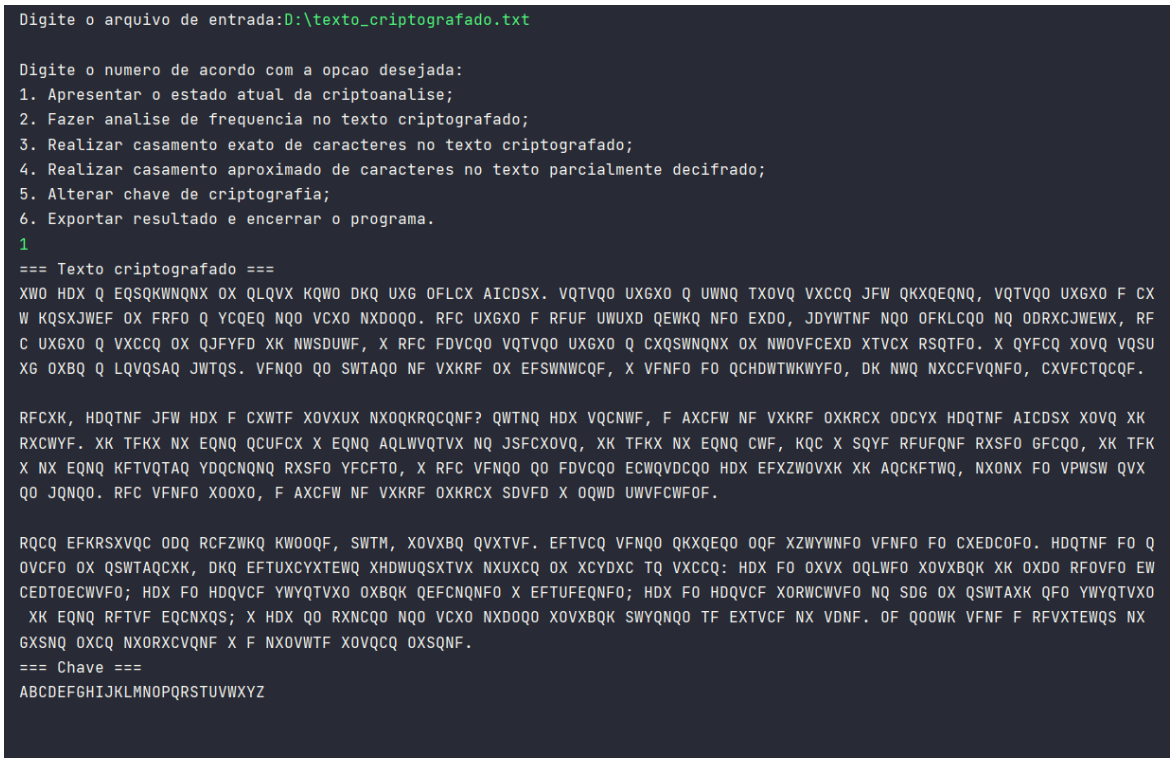

Figura 8. Inserção do arquivo de entrada e apresentação do estado da criptoanálise sem nenhuma alteração na chave

```
=== Texto parcialmente decifrado ===
XWO HDX Q EQSQKWNQNX OX QLQVX KQWO DKQ UXG OFLCX AICDSX. VQTVQO UXGXO Q UWNQ TXOVQ VXCCQ JFW QKXQEQNQ, VQTVQO UXGXO F CX
W KQSXJWEF OX FRFO Q YCQEQ NQO VCXO NXDOQO. RFC UXGXO F RFUF UWUXD QEWKQ NFO EXDO, JDYWTNF NQO OFKLCQO NQ ODRXCJWEWX, RF
C UXGXO Q VXCCQ OX QJFYFD XK NWSDUWF, X RFC FDVCQO VQTVQO UXGXO Q CXQSWNQNX OX NWOVFCEXD XTVCX RSQTFO. X QYFCQ XOVQ VQSU
XG OXBQ Q LQVQSAQ JWTQS. VFNQO QO SWTAQO NF VXKRF OX EFSWNWCQF, X VFNFO FO QCHDWTWKWYFO, DK NWQ NXCCFVQNFO, CXVFCTQCQF.

RFCXK, HDQTNF JFW HDX F CXWTF XOVXUX NXOQKRQCQNF? QWTNQ HDX VQCNWF, F AXCFW NF VXKRF OXKRCX ODCYX HDQTNF AICDSX XOVQ XK
RXCWYF. XK TFKX NX EQNQ QCUFCX X EQNQ AQLWVQTVX NQ JSFCXOVQ, XK TFKX NX EQNQ CWF, KQC X SQYF RFUFQNF RXSFO GFCQO, XK TFK
X NX EQNQ KFTVQTAQ YDQCNQNQ RXSFO YFCFTO, X RFC VFNQO QO FDVCQO ECWQVDCQO HDX EFXZWOVXK XK AQCKFTWQ, NXONX FO VPWSW QVX
QO JQNQO. RFC VFNFO XOOXO, F AXCFW NF VXKRF OXKRCX SDVFD X OQWD UWVFCWFOF.

RQCQ EFKRSXVQC ODQ RCFZWKQ KWOOQF, SWTM, XOVXBQ QVXTVF. EFTVCQ VFNQO QKXQEQO OQF XZWYWNFO VFNFO FO CXEDCOFO. HDQTNF FO Q
OVCFO OX QSWTAQCXK, DKQ EFTUXCYXTEWQ XHDWUQSXTVX NXUXCQ OX XCYDXC TQ VXCCQ: HDX FO OXVX OQLWFO XOVXBQK XK OXDO RFOVFO EW
CEDTOECWVFO; HDX FO HDQVCF YWYQTVXO OXBQK QEFCNQNFO X EFTUFEQNFO; HDX FO HDQVCF XORWCWVFO NQ SDG OX QSWTAXK QFO YWYQTVXO
 XK EQNQ RFTVF EQCNXQS; X HDX QO RXNCQO NQO VCXO NXDOQO XOVXBQK SWYQNQO TF EXTVCF NX VDNF. OF QOOWK VFNF F RFVXTEWQS NX
GXSNQ OXCQ NXORXCVQNF X F NXOVWTF XOVQCQ OXSQNF.
```

Figura 9. Apresentação do estado da criptoanálise sem nenhuma alteração na chave

```
Digite o numero de acordo com a opcao desejada:
1. Apresentar o estado atual da criptoanalise;
2. Fazer analise de frequencia no texto criptografado;
3. Realizar casamento exato de caracteres no texto criptografado;
4. Realizar casamento aproximado de caracteres no texto parcialmente decifrado;
5. Alterar chave de criptografia;
6. Exportar resultado e encerrar o programa.
2
Letra, Cont., Freq., No portugues: Letra, Freq.
Q       159   11.00%                A   14.63%
X       154   10.65%                E   12.57%
F       127    8.78%                O   10.73%
O       121    8.37%                S    7.81%
C        79    5.46%                R    6.53%
N        69    4.77%                I    6.18%
V        69    4.77%                N    5.05%
W        61    4.22%                D    4.99%
D        44    3.04%                M    4.74%
K        42    2.90%                U    4.63%
T        42    2.90%                T    4.34%
E        31    2.14%                C    3.88%
R        29    2.01%                L    2.87%
S        29    2.01%                P    2.52%
U        20    1.38%                V    1.67%
Y        18    1.24%                G    1.30%
H        15    1.04%                H    1.28%
A        11    0.76%                Q    1.20%
G        10    0.69%                B    1.04%
J         9    0.62%                F    1.02%
L         6    0.41%                Z    0.47%
B         5    0.35%                J    0.40%
Z         3    0.21%                X    0.21%
I         2    0.14%                K    0.02%
M         1    0.07%                W    0.01%
P         1    0.07%                Y    0.01%
```

Figura 10. Resultado da análise de frequência do texto criptografado

```
Digite o numero de acordo com a opcao desejada:
1. Apresentar o estado atual da criptoanalise;
2. Fazer analise de frequencia no texto criptografado;
3. Realizar casamento exato de caracteres no texto criptografado;
4. Realizar casamento aproximado de caracteres no texto parcialmente decifrado;
5. Alterar chave de criptografia;
6. Exportar resultado e encerrar o programa.
3
Qual o padrao utilizado?
>Q
 Ocorrencias: 159
```

Figura 11. Casamento exato de um caractere no texto

```
Digite o numero de acordo com a opcao desejada:
1. Apresentar o estado atual da criptoanalise;
2. Fazer analise de frequencia no texto criptografado;
3. Realizar casamento exato de caracteres no texto criptografado;
4. Realizar casamento aproximado de caracteres no texto parcialmente decifrado;
5. Alterar chave de criptografia;
6. Exportar resultado e encerrar o programa.
3
Qual o padrao utilizado?
>HDX
 Ocorrencias: 8
```

Figura 12. Casamento exato de um conjunto de caracteres no texto

```
Digite o numero de acordo com a opcao desejada:
1. Apresentar o estado atual da criptoanalise;
2. Fazer analise de frequencia no texto criptografado;
3. Realizar casamento exato de caracteres no texto criptografado;
4. Realizar casamento aproximado de caracteres no texto parcialmente decifrado;
5. Alterar chave de criptografia;
6. Exportar resultado e encerrar o programa.
5
Informe a letra original, seguida da letra para a qual foi mapeada:
>A Q
 Registrado: A -> Q
```

Figura 13. Alteração na chave de criptografia

```
=== Chave ===
ABCDEFGHIJKLMNOPQRSTUVWXYZ
Q   X           C

=== Texto parcialmente decifrado ===
EWO HDE A EASAKWNANE OE ALAVE KAWO DKA UEG OFLRE AIRDSE. VATVAO UEGEO A UWNA TEOVA VERRA JFW AKEAEANA, VATVAO UEGEO F RE
W KASEJWEF OE FRFO A YRAEA NAO VREO NEDOAO. RFR UEGEO F RFUF UWUED AEWKA NFO EEDO, JDYWTNF NAO OFKLRAO NA ODRERJWEWE, RF
R UEGEO A VERRA OE AJFYFD EK NWSDUWF, E RFR FDVRAO VATVAO UEGEO A REASWNANE OE NWOVFREED ETVRE RSATFO. E AYFRA EOVA VASU
EG OEBA A LAVASAA JWTAS. VFNAO AO SWTAAO NF VEKRF OE EFSWNWRAF, E VFNFO FO ARHDWTWKWYFO, DK NWA NERRFVANFO, REVFRTARAF.

RFREK, HDATNF JFW HDE F REWTF EOVEUE NEOAKRARANF? AWTNA HDE VARNWF, F AERFW NF VEKRF OEKRRE ODRYE HDATNF AIRDSE EOVA EK
RERWYF. EK TFKE NE EANA ARUFRE E EANA AALWVATVE NA JSFREOVA, EK TFKE NE EANA RWF, KAR E SAYF RFUFANF RESFO GFRAO, EK TFK
E NE EANA KFTVATAA YDARNANA RESFO YFRFTO, E RFR VFNAO AO FDVRAO ERWAVDRAO HDE EFEZWOVEK EK AARKFTWA, NEONE FO VPWSW AVE
AO JANAO. RFR VFNFO EOOEO, F AERFW NF VEKRF OEKRRE SDVFD E OAWD UWVFRWFOF.

RARA EFKRSEVAR ODA RRFZWKA KWOOAF, SWTM, EOVEBA AVETVF. EFTVRA VFNAO AKEAEAO OAF EZWYWNFO VFNFO FO REEDROFO. HDATNF FO A
OVRFO OE ASWTAAREK, DKA EFTUERYETEWA EHDWUASETVE NEUERA OE ERYDER TA VERRA: HDE FO OEVE OALWFO EOVEBAK EK OEDO RFOVFO EW
REDTOERWVFO; HDE FO HDAVRF YWYATVEO OEBAK AEFRNANFO E EFTUFEANFO; HDE FO HDAVRF EORWRWVFO NA SDG OE ASWTAEK AFO YWYATVEO
 EK EANA RFTVF EARNEAS; E HDE AO RENRAO NAO VREO NEDOAO EOVEBAK SWYANAO TF EETVRF NE VDNF. OF AOOWK VFNF F RFVETEWAS NE
GESNA OERA NEORERVANF E F NEOVWTF EOVARA OESANF.
```

Figura 14. Apresentação do estado da criptoanálise após algumas alterações na chave

```
Digite o numero de acordo com a opcao desejada:
1. Apresentar o estado atual da criptoanalise;
2. Fazer analise de frequencia no texto criptografado;
3. Realizar casamento exato de caracteres no texto criptografado;
4. Realizar casamento aproximado de caracteres no texto parcialmente decifrado;
5. Alterar chave de criptografia;
6. Exportar resultado e encerrar o programa.
4
Qual o padrao e a tolerancia utilizados?
>ZELDA 3
 @[76, 81):TEOVA
@[82, 87):VERRA
@[155, 160):NEDOA
@[186, 191):AEWKA
@[249, 254):VERRA
@[1015, 1020):REEDR
@[1105, 1110):VERRA
@[1309, 1314):RENRA
@[1325, 1330):NEDOA
@[1396, 1401):GESNA

Ocorrencias: 10
```

Figura 15. Casamento aproximado de caracteres no texto parcialmente decifrado de acordo com a chave apresentada (Fig. 12)

```
=== Chave ===
ABCDEFGHIJKLMNOPQRSTUVWXYZ
QLENXJYAWBMSKTFRHCOVDUPZIG

=== Texto parcialmente decifrado ===
EIS QUE A CALAMIDADE SE ABATE MAIS UMA VEZ SOBRE HYRULE. TANTAS VEZES A VIDA NESTA TERRA FOI AMEACADA, TANTAS VEZES O RE
I MALEFICO SE OPOS A GRACA DAS TRES DEUSAS. POR VEZES O POVO VIVEU ACIMA DOS CEUS, FUGINDO DAS SOMBRAS DA SUPERFICIE, PO
R VEZES A TERRA SE AFOGOU EM DILUVIO, E POR OUTRAS TANTAS VEZES A REALIDADE SE DISTORCEU ENTRE PLANOS. E AGORA ESTA TALV
EZ SEJA A BATALHA FINAL. TODAS AS LINHAS DO TEMPO SE COLIDIRAO, E TODOS OS ARQUINIMIGOS, UM DIA DERROTADOS, RETORNARAO.

POREM, QUANDO FOI QUE O REINO ESTEVE DESAMPARADO? AINDA QUE TARDIO, O HEROI DO TEMPO SEMPRE SURGE QUANDO HYRULE ESTA EM
PERIGO. EM NOME DE CADA ARVORE E CADA HABITANTE DA FLORESTA, EM NOME DE CADA RIO, MAR E LAGO POVOADO PELOS ZORAS, EM NOM
E DE CADA MONTANHA GUARDADA PELOS GORONS, E POR TODAS AS OUTRAS CRIATURAS QUE COEXISTEM EM HARMONIA, DESDE OS TWILI ATE
AS FADAS. POR TODOS ESSES, O HEROI DO TEMPO SEMPRE LUTOU E SAIU VITORIOSO.

PARA COMPLETAR SUA PROXIMA MISSAO, LINK, ESTEJA ATENTO. CONTRA TODAS AMEACAS SAO EXIGIDOS TODOS OS RECURSOS. QUANDO OS A
STROS SE ALINHAREM, UMA CONVERGENCIA EQUIVALENTE DEVERA SE ERGUER NA TERRA: QUE OS SETE SABIOS ESTEJAM EM SEUS POSTOS CI
RCUNSCRITOS; QUE OS QUATRO GIGANTES SEJAM ACORDADOS E CONVOCADOS; QUE OS QUATRO ESPIRITOS DA LUZ SE ALINHEM AOS GIGANTES
 EM CADA PONTO CARDEAL; E QUE AS PEDRAS DAS TRES DEUSAS ESTEJAM LIGADAS NO CENTRO DE TUDO. SO ASSIM TODO O POTENCIAL DE
ZELDA SERA DESPERTADO E O DESTINO ESTARA SELADO.
```

Figura 16. Apresentação do texto descriptografado após alterações em toda a chave

4. Conclusão

O progresso do trabalho se deu de forma satisfatória uma vez que conhecimentos aplicados em sala de aula puderam ser experimentados durante o desenvolvimento do algoritmo e execução dos testes.

Como forma de melhor organização do trabalho, foi utilizado o sistema de versão de controle Git na plataforma GitHub para compartilhamento do código, além do uso do aplicativo Whatsapp para melhor comunicação do grupo quanto a divisão de tarefas e resolução de dúvidas sobre o projeto.

O trabalho foi uma maneira prática e eficiente de evidenciar e verificar conhecimentos sobre criptoanálise utilizando-se de um tema que gerou bastante interesse no grupo. De maneira geral o trabalho se deu de forma concisa visto que o ambiente era colaborativo entre os membros.

5. Bibliografia

[1] **CPROGRAM.** GitHub. Disponível em: https://github.com/seeditsolution/cprogram/blob/master/Calendar. Acesso em 17 de março de 2022. 2.1

[2] **DECIFRANDO TEXTOS EM PORTUGUÊS.** Disponível em https://www.gta.ufrj.br/grad/06_2/alexandre/criptoanalise.html. Acesso em 18 de março de 2022. 2.7