Trabalho de Segurança Computacional

Universidade de Brasília, 2025.1

Segurança Computacional turma 01, professora Priscila Solis

Aluno: Andrey Calaça Resende

Matrícula: 180062433

1. Ideia do trabalho

Estudar a cifra de Vigenère e implementar um cifrador/decifrador, bem como um ataque baseado em análise de frequências para recuperar a senha geradora.

2. Cifrador/decifrador

Primeiro, foi criada a matriz de Vigenère:

```
linhaA = ['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z']
linhaB = ['b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z','a']
linhaC = ['c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z','a','b']
linhaD = ['d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z','a','b','c']
linhaE = ['e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z','a','b','c','d']
linhaF = ['f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z','a','b','c','d','e']
linhaG = ['g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z','a','b','c','d','e','f']
linhaH = ['h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z','a','b','c','d','e','f','g']
linhaI = ['i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z','a','b','c','d','e','f','g','h']
linhaJ = ['j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z','a','b','c','d','e','f','g','h','i']
linhaK = ['k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z','a','b','c','d','e','f','g','h','i','j']
linhaL = ['l','m','n','o','p','q','r','s','t','u','v','w','x','y','z','a','b','c','d','e','f','g','h','i','j','k']
linhaM = ['m','n','o','p','q','r','s','t','u','v','w','x','y','z','a','b','c','d','e','f','g','h','i','j','k','l']
linhaN = ['n','o','p','q','r','s','t','u','v','w','x','y','z','a','b','c','d','e','f','g','h','i','j','k','l','m']
linhaO = ['o','p','q','r','s','t','u','v','w','x','y','z','a','b','c','d','e','f','g','h','i','j','k','l','m','n']
linhaP = ['p','q','r','s','t','u','v','w','x','y','z','a','b','c','d','e','f','g','h','i','j','k','l','m','n','o']
linhaQ = ['q','r','s','t','u','v','w','x','y','z','a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p']
linhaR = ['r','s','t','u','v','w','x','y','z','a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q']
linhaS = ['s','t','u','v','w','x','y','z','a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r']
linhaT = ['t','u','v','w','x','y','z','a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s']
linhaU = ['u','v','w','x','y','z','a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t']
linhaV = ['v','w','x','y','z','a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u']
linhaW = ['w','x','y','z','a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v']
linhaX = ['x','y','z','a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w']
linhaY = ['y','z','a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x']
linhaZ = ['z','a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y']

matriz = [linhaA, linhaB, linhaC, linhaD, linhaE, linhaF, linhaG, linhaH, linhaI, linhaJ, linhaK, linhaL, linhaM,
          linhaN, linhaO, linhaP, linhaQ, linhaR, linhaS, linhaT, linhaU, linhaV, linhaW, linhaX, linhaY, linhaZ]
```

Depois, foram feitas as funções para cifrar e para decifrar a mensagem, a partir de uma chave conhecida:

```python
def encrypt_vigenere(mensagem, chave):
    criptograma = ''
    i = 0
    for caractere in mensagem:
        ascii_msg   = ord(caractere)-ord('a')
        ascii_chave = ord(chave[i])-ord('a')
        criptograma = criptograma + matriz[ascii_chave][ascii_msg]
        i = (i+1)%len(chave)
    return criptograma

def decrypt_vigenere(criptograma, chave):
    mensagem_original = ''
    i = 0
    for caractere in criptograma:
        ascii_chave = ord(chave[i])-ord('a')
        for linha in matriz:
            if(linha[ascii_chave] == caractere):
                mensagem_original = mensagem_original + linha[0]
        i = (i+1)%len(chave)
    return mensagem_original
```

Para esse projeto as mensagens devem estar formatadas usando apenas letras. Foi feita uma função para remover caracteres especiais, converter todas as letras para minúsculo e chegar no formato ideal. O criptograma também precisa seguir esse formato.

3. Ideia do ataque

O ataque se baseia no fato de que línguas possuem distribuições de frequência bem estudadas para cada uma de suas letras em textos genéricos.

Primeiro, precisamos criar os subtextos considerando cada tamanho de chave possível, nesse caso o valor máximo considerado foi para uma chave de 20 caracteres, então para cada tamanho de chave N variando de 1 a 20 foram criados os N subtextos.

Depois, foi calculado o índice de coincidência para cada subtexto, usando a fórmula:

$$IC = \frac{\sum f_i(f_i - 1)}{N(N-1)}$$

E, a partir dos valores do índice para cada subtexto, foi calculada a média para cada tamanho de chave, valor que foi posteriormente analisado visualmente em tempo de execução buscando valores próximos a 0,072 para textos em português e 0,066 para textos em inglês.

Com o tamanho da chave em mãos foi feita uma outra análise dos subtextos, onde, para cada letra da chave, foram criados 26 vetores, cada um contendo 26 valores correspondendo à frequência das letras a-z no subtexto considerando o deslocamento variando de 1-26.

Com essa informação visual na tela, procura-se o vetor que mais se aproxima dos valores normais da distribuição de letras da língua do texto e descobre-se qual letra foi utilizada para o deslocamento.

```
1º Letra:
Deslocando pela letra: a
Ocorrencias = [0.0, 8.3, 4.7, 10.7, 4.333, 1.3, 0.7, 0.7, 0.7, 0.0, 7.3, 1.7, 2.3, 2.3, 16.0, 1.7, 4.7, 6.0, 7.3, 0.0, 0.0, 4.0, 1.0, 8.7, 4.7, 1.0]
Deslocando pela letra: b
Ocorrencias = [8.333, 4.7, 10.7, 4.3, 1.333, 0.7, 0.7, 0.7, 0.0, 7.3, 1.7, 2.3, 2.3, 16.0, 1.667, 4.7, 6.0, 7.333, 0.0, 0.0, 4.0, 1.0, 8.7, 4.7, 1.0, 0.0]
Deslocando pela letra: c
Ocorrencias = [4.667, 10.7, 4.3, 1.3, 0.667, 0.7, 0.7, 0.0, 7.3, 1.7, 2.3, 2.3, 16.0, 1.7, 4.667, 6.0, 7.3, 0.0, 0.0, 4.0, 1.0, 8.7, 4.7, 1.0, 0.0, 8.3]
Deslocando pela letra: d
Ocorrencias = [10.667, 4.3, 1.3, 0.7, 0.667, 0.7, 0.0, 7.3, 1.7, 2.3, 2.3, 16.0, 1.7, 4.7, 6.0, 7.3, 0.0, 0.0, 4.0, 1.0, 8.7, 4.7, 1.0, 0.0, 8.3, 4.7]
Deslocando pela letra: e
Ocorrencias = [4.333, 1.3, 0.7, 0.7, 0.667, 0.0, 7.3, 1.7, 2.3, 2.3, 16.0, 1.7, 4.7, 6.0, 7.333, 0.0, 0.0, 4.0, 1.0, 8.7, 4.7, 1.0, 0.0, 8.3, 4.7, 10.7]
Deslocando pela letra: f
Ocorrencias = [1.333, 0.7, 0.7, 0.7, 0.0, 7.3, 1.7, 2.3, 2.3, 16.0, 1.7, 4.7, 6.0, 7.3, 0.0, 0.0, 4.0, 1.0, 8.7, 4.7, 1.0, 0.0, 8.3, 4.7, 10.7, 4.3]
Deslocando pela letra: g
Ocorrencias = [0.667, 0.7, 0.7, 0.0, 7.333, 1.7, 2.3, 2.3, 16.0, 1.7, 4.7, 6.0, 7.3, 0.0, 0.0, 4.0, 1.0, 8.667, 4.7, 1.0, 0.0, 8.3, 4.7, 10.7, 4.3, 1.3]
Deslocando pela letra: h
Ocorrencias = [0.667, 0.7, 0.0, 7.3, 1.667, 2.3, 2.3, 16.0, 1.7, 4.7, 6.0, 7.3, 0.0, 0.0, 4.0, 1.0, 8.7, 4.667, 1.0, 0.0, 8.3, 4.7, 10.7, 4.3, 1.3, 0.7]
Deslocando pela letra: i
Ocorrencias = [0.667, 0.0, 7.3, 1.7, 2.333, 2.3, 16.0, 1.7, 4.7, 6.0, 7.3, 0.0, 0.0, 4.0, 1.0, 8.7, 4.7, 1.0, 0.0, 8.3, 4.7, 10.7, 4.3, 1.3, 0.7, 0.7]
Deslocando pela letra: j
Ocorrencias = [0.0, 7.3, 1.7, 2.3, 2.333, 16.0, 1.7, 4.7, 6.0, 7.3, 0.0, 0.0, 4.0, 1.0, 8.667, 4.7, 1.0, 0.0, 8.3, 4.7, 10.7, 4.3, 1.3, 0.7, 0.7, 0.7]
Deslocando pela letra: k
Ocorrencias = [7.333, 1.7, 2.3, 2.3, 16.0, 1.7, 4.7, 6.0, 7.3, 0.0, 0.0, 4.0, 1.0, 8.7, 4.667, 1.0, 0.0, 8.333, 4.7, 10.7, 4.3, 1.3, 0.7, 0.7, 0.7, 0.0]
Deslocando pela letra: l
Ocorrencias = [1.667, 2.3, 2.3, 16.0, 1.667, 4.7, 6.0, 7.3, 0.0, 0.0, 4.0, 1.0, 8.7, 4.7, 1.0, 0.0, 8.3, 4.667, 10.7, 4.3, 1.3, 0.7, 0.7, 0.7, 0.0, 7.3]
Deslocando pela letra: m
Ocorrencias = [2.333, 2.3, 16.0, 1.7, 4.667, 6.0, 7.3, 0.0, 0.0, 4.0, 1.0, 8.7, 4.7, 1.0, 0.0, 8.3, 4.7, 10.667, 4.3, 1.3, 0.7, 0.7, 0.7, 0.0, 7.3, 1.7]
Deslocando pela letra: n
Ocorrencias = [2.333, 16.0, 1.7, 4.7, 6.0, 7.3, 0.0, 0.0, 4.0, 1.0, 8.7, 4.7, 1.0, 0.0, 8.333, 4.7, 10.7, 4.333, 1.3, 0.7, 0.7, 0.7, 0.0, 7.3, 1.7, 2.3]
Deslocando pela letra: o
Ocorrencias = [16.0, 1.7, 4.7, 6.0, 7.333, 0.0, 0.0, 4.0, 1.0, 8.7, 4.7, 1.0, 0.0, 8.3, 4.667, 10.7, 4.3, 1.333, 0.7, 0.7, 0.7, 0.0, 7.3, 1.7, 2.3, 2.3]
Deslocando pela letra: p
Ocorrencias = [1.667, 4.7, 6.0, 7.3, 0.0, 0.0, 4.0, 1.0, 8.7, 4.7, 1.0, 0.0, 8.3, 4.7, 10.667, 4.3, 1.3, 0.667, 0.7, 0.7, 0.0, 7.3, 1.7, 2.3, 2.3, 16.0]
Deslocando pela letra: q
Ocorrencias = [4.667, 6.0, 7.3, 0.0, 0.0, 4.0, 1.0, 8.7, 4.7, 1.0, 0.0, 8.3, 4.7, 10.7, 4.333, 1.3, 0.7, 0.667, 0.7, 0.0, 7.3, 1.7, 2.3, 2.3, 16.0, 1.7]
Deslocando pela letra: r
Ocorrencias = [6.0, 7.3, 0.0, 0.0, 4.0, 1.0, 8.7, 4.7, 1.0, 0.0, 8.3, 4.7, 10.7, 4.3, 1.333, 0.7, 0.7, 0.667, 0.0, 7.3, 1.7, 2.3, 2.3, 16.0, 1.7, 4.7]
Deslocando pela letra: s
Ocorrencias = [7.333, 0.0, 0.0, 4.0, 1.0, 8.7, 4.7, 1.0, 0.0, 8.3, 4.7, 10.7, 4.3, 1.3, 0.667, 0.7, 0.7, 0.0, 7.3, 1.7, 2.3, 2.3, 16.0, 1.7, 4.7, 6.0]
Deslocando pela letra: t
Ocorrencias = [0.0, 0.0, 4.0, 1.0, 8.667, 4.7, 1.0, 0.0, 8.3, 4.7, 10.7, 4.3, 1.3, 0.7, 0.667, 0.7, 0.0, 7.333, 1.7, 2.3, 2.3, 16.0, 1.7, 4.7, 6.0, 7.3]
Deslocando pela letra: u
Ocorrencias = [0.0, 4.0, 1.0, 8.7, 4.667, 1.0, 0.0, 8.3, 4.7, 10.7, 4.3, 1.3, 0.7, 0.7, 0.667, 0.0, 7.3, 1.667, 2.3, 2.3, 16.0, 1.7, 4.7, 6.0, 7.3, 0.0]
Deslocando pela letra: v
Ocorrencias = [4.0, 1.0, 8.7, 4.7, 1.0, 0.0, 8.3, 4.7, 10.7, 4.3, 1.3, 0.7, 0.7, 0.7, 0.0, 7.3, 1.7, 2.333, 2.3, 16.0, 1.7, 4.7, 6.0, 7.3, 0.0, 0.0]
Deslocando pela letra: w
Ocorrencias = [1.0, 8.7, 4.7, 1.0, 0.0, 8.3, 4.7, 10.7, 4.3, 1.3, 0.7, 0.7, 0.7, 0.0, 7.333, 1.7, 2.3, 2.333, 16.0, 1.7, 4.7, 6.0, 7.3, 0.0, 0.0, 4.0]
Deslocando pela letra: x
Ocorrencias = [8.667, 4.7, 1.0, 0.0, 8.333, 4.7, 10.7, 4.3, 1.3, 0.7, 0.7, 0.7, 0.0, 7.3, 1.667, 2.3, 2.3, 16.0, 1.7, 4.7, 6.0, 7.3, 0.0, 0.0, 4.0, 1.0]
Deslocando pela letra: y
Ocorrencias = [4.667, 1.0, 0.0, 8.3, 4.667, 10.7, 4.3, 1.3, 0.7, 0.7, 0.7, 0.0, 7.3, 1.7, 2.333, 2.3, 16.0, 1.667, 4.7, 6.0, 7.3, 0.0, 0.0, 4.0, 1.0, 8.7]
Deslocando pela letra: z
Ocorrencias = [1.0, 0.0, 8.3, 4.7, 10.667, 4.3, 1.3, 0.7, 0.7, 0.7, 0.0, 7.3, 1.7, 2.3, 2.333, 16.0, 1.7, 4.667, 6.0, 7.3, 0.0, 0.0, 4.0, 1.0, 8.7, 4.7]
```

Na imagem percebe-se que o vetor da letra K é o que mais se aproxima da distribuição das letras no inglês, portanto essa é a primeira letra da chave que foi usada para criptografar a mensagem.

Os valores correspondentes às letras ‘a’, ‘e’, ‘o’ e ‘r’ foram formatados com um arredondamento diferente apenas para facilitar a visualização, uma vez que são letras com a tendência de aparecerem com valores altos e relevantes.

```
Escolha a 1ª letra da chave
k
Escolha a 2ª letra da chave
e
Escolha a 3ª letra da chave
y
Mensagem original =
theheartpumpsbloodwitharhythmdeterminedbyagroupofpacemakercellsinthesinoatrialnodethesegenerateanelectriccurrentthatcausestheheartto contracttravelingthroughtheatrioventricularnodeandalongtheconductionsystemoftheheartinhumansdeoxygenatedbloodenterstheheartthroughtherightatriumfromthesuperiorandinferiorvenaecavaeandpassestotherightventriclefromhereitispumpedintopulmonarycirculationtothelungswhereitreceivesoxygenandgivesoffcarbondioxideoxygenatedbloodthenreturnstotheleftatriumpassesthroughtheleftventricleandispumpedoutthroughtheaortaintosystemiccirculationtravelingthrougharteriesarteriolesandcapillarieswherenutrientsandothersubstancesareexchangedbetweenbloodvesselsandcellslosingoxygenandgainingcarbondioxidebeforebeingreturnedtotheheartthroughvenulesandveinstheadultheartbeatsatarestingratecloseto beatsperminuteexercisetemporarilyincreasestheratebutlowersitinthelongtermandisgoodforhearthealth
```

Após escolher a senha, o programa decodifica a mensagem e imprime na tela, oferecendo também a opção de tentar novamente com uma senha diferente, caso o texto decodificado não seja o desejado ou não faça sentido.