Trab1Seg

Davi Mansur Costa - 211042701

October 2, 2023

1 Introdução

Esse relatório é sobre o trabalho 1 da matéria de Segurança Computacional, o qual tem como tema a cifra de Vigenère. A cifra de Vigenère é um método de criptografia que aplica várias cifras de César e deslocamentos baseados nas diferentes posicões das letras da chave escolhida no alfabeto. O trabalho é dividido em 2 partes:

Cifrador e Decifrador
Quebrar

As implementações foram feitas com base nos materiais auxiliares fornecidos pelo professor J.Gondim e pelo monitor Paulo e conteúdos online.

2 Metodologia

2.1 Parte I

Para fazer a primeira parte que é a realização de um codificador e um decodificador foi tratada a mensagem ignorando as pontuações, passando as letras para minúsculo e retirando os espaços.

2.1.1 Cifrador

Para a implementação do cifrador foi utilizada uma função que recebe uma chave e uma mensagem em texto simples e com laços de repetição e operações matemáticas realiza deslocamentos para direita de cada caracter da mensagem pelo número correspondente ao da chave no alfabeto, que é feito conferindo o indice da letra na lista de letras que está ordenada, tendo o caracter cifrado como resultado.

```
from unidecode import unidecode
import string
def cifrador(mensagem, chave):
    mensagem = mensagem.lower()
```

```
#removendo espacos, se houver, na mensagem
while(" " in mensagem):
    mensagem = mensagem.replace(" ","")
#lista das letras posicionadas
letras = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', '
chavecheia=""
#adicionando na chave inteira o correspondente da chave, se for uma letra, o
n=0
for caracter in mensagem:
    testa_acento = unidecode(caracter)
    if (caracter != testa_acento) or (caracter in string.punctuation):
        chavecheia+="*"
    else:
        if n>=len(chave):
            n=n-len(chave)
        chavecheia+=chave[n]
        n+=1
criptograma=""
#deslocando cada letra da mensagem pela chave para formar o criptograma
n=0
for letra in mensagem:
    testa_acento = unidecode(letra)
    if (letra != testa_acento) or (letra in string.punctuation):
        criptograma+=letra
    else:
        numero_da_proxima = letras.index(letra) + letras.index(chavecheia[n]
        while(numero_da_proxima>=26):
            numero_da_proxima-=26
        proxima_letra = letras[numero_da_proxima]
        criptograma+=proxima_letra
    n+=1
print(criptograma)
```

2.1.2 Decifrador

Para a implementação do decifrador foi utilizada uma função que recebe uma chave e uma mensagem cifrada e com laços de repetição e operações matemáticas realiza deslocamentos para esquerda de cada caracter da mensagem pelo número correspondente ao da chave no alfabeto, que é feito conferindo o indice da letra na lista de letras que está ordenada, tendo o caracter decifrado como resultado.

O código do decifrador é basicamente o mesmo do Cifrador alterando o + pelo - na função de deslocamento, por isso não está aqui, mas está no github.

2.2 Parte II

A segunda parte consiste em fazer um ataque para recuperação da senha(chave) por análise de frequência das letras.

2.2.1 Descobrir o tamanho da chave

A checagem do tamanho é feita pela checagem de todas repetições de trigramas, anotando os fatores das distancias entre eles. Os 3 valores de chave com mais fatores devem ser escolhidos para a análise de frequência.

```
def tamanhoDaChave(textocifrado):
    factors = [0] * 22
    for i in range(len(textocifrado)-2):
        trigrama=textocifrado[i:i+3]
        for j in range(i+3,len(textocifrado)-2):
            if textocifrado[j:j+3] == trigrama:
                distancia = j-(i+2)
                print(trigrama,distancia)
                for k in range(2, 22):
                    if distancia%k==0:
                        factors[k]+=1
                break

    print("Tamanhos de chaves e sua quantidade de fatores encontrada:")
    for i in range(2,len(factors)):
        print(f'\t{i} - {factors[i]}')
    selected = int(input('Selecione o tamanho da chave desejada:'))
    return selected
```

2.2.2 Análise de frequência

Foi calculada a frequência de cada letra no texto cifrado e comparado com a frequência padrão da língua inglesa e a parte que não foi possível implementar foi fazer essa análise para cada letra da chave para encontrar a chave, pois estava sempre dando a mesma letra "a"

```
def calcular_freq(texto):
    total_caracteres = len(texto)
    contagem = [0] * 26
    for letra in texto:
        if letra.isalpha():
            n=0
            for let in letras:
                if letra.lower() == let:
                    indice = n
                n+=1
```

```
            contagem[indice] += 1
    n=0
    for x in contagem:
        contagem[n]=x/total_caracteres
        n+=1
    return contagem
def analise_frequencia(textocifrado):
    freq = calcular_freq(textocifrado)
    dif = []
    for j in range(26):
        score = sum([lang_freq[i] * freq[i] for i in range(26)])
        dif.append(score)
        freq.append(freq.pop(0))
    return dif.index(max(dif))
```

3 Conclusão

A primeira parte do experimento foi tranquila, já a segunda só foi possível avançar após os conteúdos adicionais do monitor Paulo e mesmo assim a parte de análise de frequência não foi bem executada, no geral aprendi bastante com esse trabalho, principalmente sobre as vulnerabilidades da cifra de Vigenère. Os códigos em funcionamento encontram-se no github

4 Referências

https://pages.mtu.edu/ shene/NSF-4/Tutorial/VIG/Vig-Examples.html

https://medium.com/asecuritysite-when-bob-met-alice/for-the-love-of-ciphers-vigen

https://pt.wikipedia.org/wiki/Frequência$_d e_l etras$