Segurança Computacional - Trabalho 1

Gabriel Cruz Vaz Santos - 200049038

Descrição do código - Cifrador/Decifrador

A função principal recebe a mensagem e senha digitadas pelo usuário e chama as demais funções responsáveis pelos processos.

main.py > ...

```python
from cypher import cypher
from decypher import decypher
from calculateFrequency import calculateFrequency

if __name__ == "__main__":
  key = input("digite uma chave cifradora: ")
  message = input("digite uma mensagem para ser encriptada: ")

  cyphertext, keystream = cypher(message, key)
  print(f"cyphertext gerado: {cyphertext}")

  originalMessage = decypher(cyphertext, keystream)
  print(f"mensagem original: {originalMessage}")
```

A função de cifra recebe a mensagem e a senha inseridas pelo usuário. Com isso, chama a função `generateKeystream()`, para que a chave utilizada na cifra possua o mesmo tamanho que a mensagem. Checamos se ele é um caractere dentro do escopo [a-z]. Caso seja, ele é cifrado utilizando o keyStream, se não, ele é ignorado pelo cifrador.

generateKey.py > generateKeystream

```python
def generateKeystream (message, key):
    messageSize = len(message)
    keySize = len(key)
    if messageSize == keySize:
        return key
    elif messageSize > keySize:
        keyStream =  [letter for letter in key]
        for i in range(len(message) - len(key)):
            keyStream.append(keyStream[i % len(key)])
        return "".join(keyStream)
    else:
        return key[0: messageSize]
```

cypher.py > cypher

```python
from generateKey import generateKeystream
from checkEspecialCharacter import checkEspecialCharacter

def cypher(message, key):
    messageSize = len(message)
    cyphertext = ""
    try:
        keystream = generateKeystream(message, key)
    except:
        print("Erro ao gerar KeyStream, tente novamente")
    else:
        print(f"Keystream gerado: {keystream}")
        for i in range(messageSize):
            if checkEspecialCharacter(message[i]):
                cyphertext += message[i]
            else:
                cypherCharacter = (ord(message[i]) + ord(keystream[i]) - 2 * ord('a')) % 26
                cypherCharacter += ord('a')  # shift
                cyphertext += chr(cypherCharacter)
    return cyphertext, keystream
```

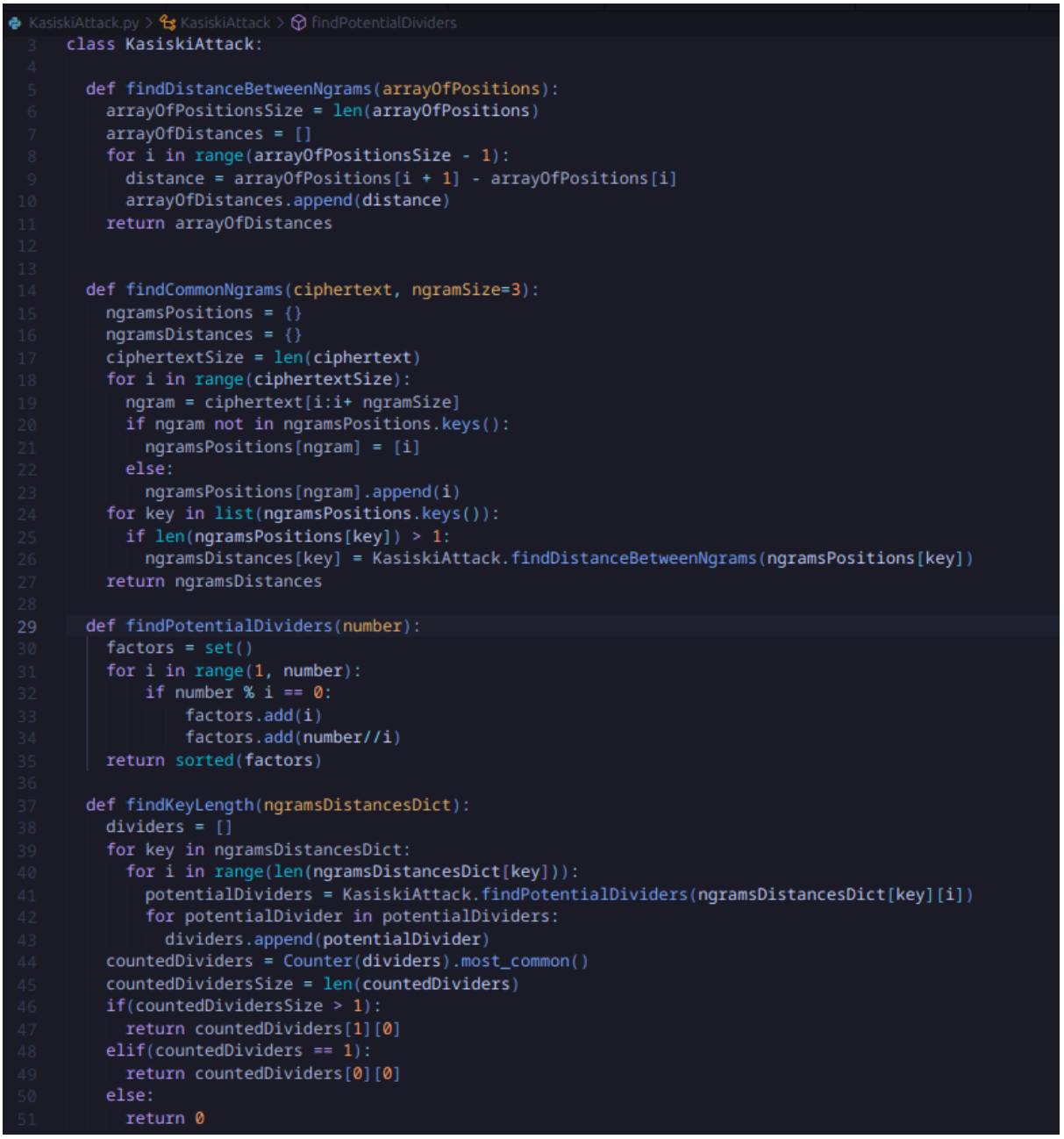
checkEspecialCharacter.py > checkEspecialCharacter

```python
def checkEspecialCharacter(character):
    asciiCharacter = ord(character)
    if asciiCharacter < 97 or asciiCharacter > 122:
        return True
    else:
        return False
```

A função para decifrar recebe o criptograma junto com nossa chave já formatada para possuir o mesmo tamanho que a mensagem e segue o mesmo procedimento que a função de cifra: Caso o caractere esteja dentro do escopo [a-z] ele é decifrado, caso o contrário ele é ignorada pelo decifrador.
Descrição do código - Quebra cifra Vigenere análise de frequência

A classe KasiskiAttack possui todas as funções necessárias para encontrar o comprimento de chave da cifra de Vigenere. Para isso, iremos dividir a

mensagem cifrada em ngrams de tamanho N e então encontrarmos padrões de repetições para esses Ngrams. Após encontrados, é feito o cálculo entre a distância dessas repetições. Essas operações são realizadas nas funções `findDistanceBetweenNgrams` e `findCommonNgrams`.



Agora possuímos um dicionário no qual cada chave corresponde a um padrão de repetição encontrado e seu respectivo valor é um array com as posições em que ele é encontrado na mensagem cifrada. Note que esse dicionário possui apenas os padrões que ocorrem mais de uma vez.
Após isso, é calculado os múltiplos comum entre cada uma das distâncias através da função `findPotencialDividers()`.

Com todos esses dados, a função `findKeyLength()` retorna o múltiplo comum que mais se repete nessa análise.
Iremos dividir o criptograma em grupos do mesmo tamanho, sendo este o tamanho da chave encontrada. Note que cada grupo é cifrado utilizando uma letra da chave, a letra mais frequente em cada grupo nos irá indicar a letra correspondente na chave.
Após isso, temos uma função responsável por reconstruir a chave dados os dados encontrados no processo acima.

- Repositórios no github
    - https://github.com/Leir-Cruz/VigenereCipher