Segurança Computacional - Relatório Trabalho 1

Felipe Fontenele dos Santos - 190027622

Setembro 2023

1 Introdução

Este trabalho tem como objetivo implementar o cifrador e o decifrador de Vigenère e o ataque de recuperação de senha por análise de frequência utilizando a linguagem de programção python.

2 Desenvolvimento

2.1 Cifrar e decifrar

Foi criada uma classe VigenereCipher que contém os métodos necessários para encriptar e decriptar as mensagens, além do método que realiza toda a lógica e um outro auxiliar.

```python
class VigenereCipher:
    alphabet = 'abcdefghijklmnopqrstuvwxyz'
    ALPHABET = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

    def decrypt_message(self, cipher_message, key):
        return self._vigenere_cipher(cipher_message, key, mode='decrypt')

    def encrypt_message(self, message, key):
        return self._vigenere_cipher(message, key)

    def _vigenere_cipher(self, message, key, mode='encrypt'):
        message_encrypted = ''
        key_index = 0
        key_shift = 0

        message, key = self._handle_message_and_key(message, key)

        for message_letter in message:
            new_char = message_letter
            if new_char.isalnum():
                key_character = key[key_index % len(key)]

                if new_char in self.alphabet:
                    key_shift = self.alphabet.index(key_character.lower())
                    message_shift = self.alphabet.index(new_char)
                else:
                    key_shift = self.ALPHABET.index(key_character)
                    message_shift = self.ALPHABET.index(new_char)

                if mode == 'decrypt':
                    key_shift = -key_shift

                new_char = self.alphabet[(message_shift + key_shift) % 26] if new_char in self.alphabet else self.ALPHABET[(message_shift + key_shift)
                key_index += 1

            message_encrypted += new_char

        return message_encrypted
```

1. O método principal, *_vigenere_cipher*, recebe a mensagem, a chave e se ele vai encriptar ou decriptar a mensagem.

2. É feito um tratamento tanto na mensagem quanto na chave chamando o método *_handle_message_and_key* para remover qualquer caractere especial ou mesmo substituir letras com acentuação. Para facilitar a implementação da limpeza da string, foi utilizado a biblioteca unicode.

```python
def _handle_message_and_key(self, message, key):
    """
    Método para tornar todas as letras maiúsculas, remover acentuações e qualquer caractere especial da mensagem e da chave informada.

    Args:
        message: a mensagem a ser cifrada.
        key: chave da cifra.

    Returns:
        Mensagem e chave sem pontuações, caracteres especiais e maiúsculas.
    """
    message = unidecode(re.sub(r'[^\d\w\s,-]', '', message))
    key = unidecode(re.sub(r'[^\w\s]', '', key))

    return message, key.upper()
```

3. Após realizar a limpeza, é feita uma interação nas letras da mensagem, onde, primeiramente, é pego o caractere da chave em relação à posição de leitura da mensagem através do resto da posição do caractere da mensagem dividido pelo tamanho da chave, garantindo que a letra obtida sempre estará dentro de um intervalo válido [1].

4. É feito uma busca na variável alphabet pelo carácter da chave e da mensagem para descobrir a posição atual da letra da mensagem que irá ser somada com a posição da letra da chave, gerando a posição da nova letra da cifra.

5. Caso ele esteja descriptografando ele irá apenas transformar o deslocamento da chave em um valor negativo.

6. Descobre o caractere da cifra pegando o resto da divisão da soma (ou subtração) da posição do caractere da letra da mensagem pela posição do caractere da chave por 26, que garante que o índice sempre estará dentro do vetor do alfabeto.

7. Soma o novo caractere à cifra resultante e soma um no iterador da mensagem.

2.2 Ataque

O ataque é realizado seguindo os seguintes passos:

1. É realizado uma limpeza na mensagem recebida, removendo tudo que não seja número, letra, vírgula ou ponto e transformando ela em maiúscula para ser manipulada.

2. É chamado o método *get_key_size* que, através da checagem de repetição de trigramas na mensagem, busca os fatores da distância entre os trigramas. São apresentados os 3 tamanhos de chaves com a maior quantidade de fatores, possibilitando o usuário escolher qual delas ele prefere ou mesmo se prefere outro valor.

3. Após determinar o tamanho da chave, a etapa subsequente envolve a invocação do método *discover_key*. Em essência, esse método executa uma análise de frequência com base no idioma especificado, isto é, ele contrasta a frequência das letras na mensagem com a frequência típica do idioma em que a mensagem está redigida. As letras são agrupadas em intervalos

de acordo com o tamanho da chave. Suponhamos uma chave de tamanho N; para cada posição i dessa chave, é necessário analisar a frequência do conjunto de letras cujas posições são múltiplos de i. Após a conclusão dessa comparação, torna-se possível determinar o deslocamento aplicado a esse conjunto de letras e, consequentemente, deduzir os caracteres que compõem a chave. A análise de frequência é executada por meio da multiplicação das frequências da língua-alvo pelas frequências encontradas na mensagem. O deslocamento que resultar no maior produto entre essas frequências é aquele que exibe a maior semelhança, revelando-se, assim, como a provável chave de descriptografia.

3 Executando o código

Para executar o código, basta utilizar o compilador do python chamando o arquivo *main.py* na pasta raiz da seguinte maneira:

```
python3 main.py
```

Caso não tenha a dependência do unicode, é possível instá-lo através do seguinte comando:

```
python –m pip install —upgrade pip && pip install unidecode
```

4 Conclusão

A implementação da cifra e decifragem foi simples e até mesmo divertida. O desafio começou no momento de desenvolver o ataque à cifra. Apesar de tudo, o trabalho contribuiu para o entendimento do funcionamento de uma cifra e da implementação de algorítimos que pudessem solucioná-la.

References

[1] Proof of Concept. *Cryptanalysis of Vigenere cipher: not just how, but why it works*. Aug. 2019. URL: `https://www.youtube.com/watch?v=QgHnr8-h0xI`.