Implementação da cifra de Vigenère

Lucas Ramson Siefert, 222011543

May 2023

1 Introdução

A Cifra de Vigenère é um método de criptografia que utiliza uma série de diferentes cifras de César baseadas em diferentes letras de uma chave, tornando mais difícil a sua decifração. Foi inventada no século XVI pelo criptógrafo francês Blaise de Vigenère, e se tornou conhecida como uma das mais seguras cifras de substituição do seu tempo. O método é baseado em uma tabela de Vigenère, que é uma matriz retangular de caracteres, e usa um algoritmo de transposição para embaralhar a mensagem original. A cifra de Vigenère foi considerada inviolável por muito tempo, mas foi finalmente quebrada no século XIX pelo matemático britânico Charles Babbage.

2 Funcionamento da cifra

A cifra de Vigenère é uma técnica de substituição, onde cada letra da mensagem original é substituída por outra letra que se encontra à uma distância fixa na sequência alfabética. A distância varia de acordo com cada letra da chave. Para criptografar a mensagem, a cifra de Vigenère usa uma tabela, chamada de tabela de Vigenère. A tabela de Vigenère é uma matriz onde cada linha e coluna é uma letra do alfabeto, com exceção da primeira linha e coluna, que são usadas para rotular as linhas e colunas.

2.1 Implementação

A implementação da cifra de Vigenère foi realizada em Python e consiste em duas funções principais: encrypt() e decrypt(). A função encrypt() recebe a mensagem original e a chave, e retorna a mensagem criptografada, enquanto a função decrypt() recebe a mensagem criptografada e a chave, e retorna a mensagem original.

O funcionamento das duas funções é praticamente idêntico. A função encrypt() inicialmente faz uma "limpeza" da mensagem, retirando espaços, acentos e quaisquer caracteres que não sejam parte do alfabeto e, em seguida, gera uma keystream, isto é, repete os caracteres da chave em sequência até que esta tenha exatamente o mesmo tamanho da mensagem "limpa".

Em seguida, a função itera sobre cada caractere da mensagem e o desloca no alfabeto, de acordo com a chave atual, ou seja, o caractere de mesmo índice da keystream, e então os adiciona à string resultante. Caracteres especiais, isto é, caracteres não presentes no alfabeto, são simplesmente adicionados à string resultante e não incrementam o índice do keystream. Abaixo pode-se ver a implementação do algoritmo em Python:

```
def encrypt(message, key):
    i = 0  # keystream index
    decrypted_message = ''

    # generates a keystream based on the "clean" message,
    #which is the message without punctuation or accents
    clean_message = ''.join(list(filter(lambda x: x.isalpha() and x.lower()
    in alphabet, message))).strip()
    keystream = ''.join([key[i % len(key)] for i in range(len(clean_message))])

    for character in message:
        # keep non letter characters intact and
        # don't increment the keystream index
        if character not in alphabet:
            decrypted_message += character
            continue
        # shifts the ASCII value of the character based on
        # the current letter's alphabet index
        decrypted_message += chr((ord(character) - 97
            + string.ascii_lowercase.index(keystream[i])) % 26 + 97)
        i += 1

    return decrypted_message
```

Conforme explicitado anteriormente, a função decrypt() realiza um processo praticamente idêntico, sendo a única diferença presente no deslocamento dos caracteres, que são deslocados para a esquerda no lugar da direita, isto é, o valor do índice no alfabeto da chave atual é decrementado no lugar de incrementado.

3 Funcionamento da quebra da cifra

O ataque à cifra de Vigenère implementado é baseado em uma análise de frequência de letras. A ideia é que a frequência de letras em uma língua seja conhecida e que isso possa ser usado para inferir a chave que foi usada para criptografar uma mensagem em particular. Para isso, são realizadas as seguintes etapas:

1. Encontrar o comprimento da chave usando o método de Kasiski.
2. Dividir a mensagem em segmentos de comprimento igual ao da chave.

3. Analisar a frequência de letras em cada segmento.

4. Comparar as frequências de letras com as frequências esperadas para a língua em questão.

5. A partir das diferenças entre as frequências observadas e esperadas, inferir a chave.

3.1 Implementação

Para realizar o procedimento explicitado, foram criadas três funções principais: kasiski_examination(), frequency_analysis() e attack().

A primeira destas, kasiski_examination(), recebe como argumentos o texto a ser decifrado, um tamanho máximo de chave e um valor de "tolerância". Em seguida, a função percorre o texto buscando substrings de 3 caracteres que se repitam ao longo do texto e armazena em uma lista as distâncias entre tais substrings repetidas. A partir dessa lista é calculado o multiplicador comum mais frequente entre os itens da lista, o qual é retornado pela função como tamanho mais provável da chave.

A frequency_analysis() recebe como argumentos o texto a ser decifrado, o tamanho da chave e a linguagem utilizada. Inicialmente o texto é dividido em uma lista com base no tamanho da chave, separando os caracteres com base no módulo de seus índices pelo tamanho da chave. Em seguida, essa lista é iterada e são testadas todas as letras do alfabeto como chave. A chave que resultar em uma lista cuja frequência de letras é mais próxima da frequência da linguagem escolhida é então inserida na chave final. Ao final de todas as iterações, a chave mais provável é retornada.

Por fim, a função attack() simplesmente recebe os argumentos da mensagem e linguagem escolhida e chama as duas últimas funções, passando para a frequency_analysis() o tamanho de chave previsto pela kasiski_examination(), também retornando a chave prevista.