Análise e Implementação de Cifras de Substituição e Transposição

Giulia Moura Ferreira, 20/00018795

[1]Dep. Ciência da Computação – Universidade de Brasília (UnB)
CIC0201 - Segurança Computacional

giulia.ferreira@aluno.unb.br

1. Introdução

Neste relatório, abordamos a implementação e a análise de métodos de criptografia aplicados na disciplina de Segurança Computacional, com ênfase nas cifras de substituição e transposição. Utilizando a IDE PyCharm para o desenvolvimento, analisamos como a cifra de deslocamento (ou shift cipher) pode ser aplicada e decifrada por meio de dois métodos: análise de frequência e força bruta. Além disso, são discutidas as técnicas de decriptação, com uma comparação entre suas eficácias e complexidades, conforme os exercícios propostos.

O repositório das atividades pode ser encontrado a partir desse link.

Para realizar as atividades propostas, foi considerada a distribuição de frequência da língua portuguesa:

```
frequency_portuguese = {
    'A': 13.9, 'B': 1.0, 'C': 4.4, 'D': 5.4, 'E': 12.2, 'F': 1.0, 'G': 1.2,
    'H': 0.8, 'I': 6.9, 'J': 0.4, 'K': 0.1, 'L': 2.8, 'M': 4.2, 'N': 5.3,
    'O': 10.8, 'P': 2.9, 'Q': 0.9, 'R': 6.9, 'S': 7.9, 'T': 4.9, 'U': 4.0,
    'V': 1.3, 'W': 0.0, 'X': 0.3, 'Y': 0.0, 'Z': 0.4
}
```

Figure 1. Distribuição de Frequência da Língua Portuguesa

2. Desenvolvimento

2.1. Atividade 1: Quebrando Shift Cipher

Na criptografia, a Cifra de César, também conhecida como cifra de substituição ou código de César, é uma das técnicas de criptografia mais simples e amplamente conhecidas. Trata-se de uma cifra de substituição, na qual cada letra do texto é substituída por outra, deslocada um número fixo de posições no alfabeto. Por exemplo, em uma substituição de três posições, a letra A seria trocada por D, B por E, e assim sucessivamente. [1]

Inicialmente, foi implementado um código 'main', uma aplicação que permite ao usuário cifrar e decifrar um texto usando a Cifra de César. Ele faz uso de três funções principais: Ele faz uso de três funções principais: shift_cipher (para cifrar o texto), frequency_analysis (para decifrar por análise de frequência) e brute_force (para decifrar por força bruta).

```python
from frequency_analysis import frequency_analysis
from brute_force import brute_force
from shift_cipher import shift_cipher


def main():  1 usage
    plaintext = input('Digite o texto a ser cifrado: ')
    encrypted_text = shift_cipher(plaintext,  k: 3)
    print('Texto original: ' + plaintext)
    print('Texto cifrado: ' + encrypted_text)
    print('Texto decifrado (por análise de frequência): ' + frequency_analysis(encrypted_text)[0])
    print('Texto decifrado (por força bruta):', brute_force(encrypted_text))


if __name__ == '__main__':
    main()
```

Figure 2. Main da Atividade 1

A cifra é realizada com deslocamento de 3, e o programa demonstra o processo de cifragem e decifragem com base na simplicidade da cifra de César, onde um deslocamento fixo modifica as letras. A função main() organiza esse fluxo de cifração e decifração, exibindo ao usuário o texto original, o texto cifrado, e os textos decifrados usando as abordagens de análise de frequência e força bruta.

2.1.1. Parte 1: Realizar a Cifra por Deslocamento

```python
import string


def shift_cipher(text, k):  2 usages
    encrypted_text = ''
    for char in text.upper():
        if char in string.ascii_uppercase:
            encrypted_text += chr((ord(char) - ord('A') + k) % 26 + ord('A'))
        else:
            encrypted_text += char
    return encrypted_text
```

Figure 3. Cifra de César

A função shift_cipher recebe o texto a ser criptografado (text) e um valor de deslocamento (k). Primeiro, converte todo o texto para letras maiúsculas para padronizar o processamento. Em seguida, percorre cada caractere: se for uma letra do alfabeto, calcula a nova posição dela somando o valor de deslocamento k, aplicando módulo 26 para garantir que o resultado fique dentro do intervalo do alfabeto. O caractere resultante é então adicionado a encrypted_text. Caracteres que não são letras são adicionados sem alteração.

A função retorna o texto criptografado, com as letras deslocadas por k posições e mantendo todos os outros caracteres inalterados.

2.1.2. Parte 2: Quebrar a Cifra por Deslocamento

```python
import string
from collections import Counter
from frequency_pt import frequency_portuguese


def frequency_analysis(text):  2 usages
    text = text.upper()
    text_counter = Counter([char for char in text if char in string.ascii_uppercase])

    text_freq = {char: (count / len(text)) * 100 for char, count in text_counter.items()}

    best_k = 0
    best_diff = float('inf')
    decrypted_text = ''

    for k in range(26):
        shifted_counter = {}

        for char in text_freq:
            shifted_char = chr((ord(char) - ord('A') - k) % 26 + ord('A'))
            shifted_counter[shifted_char] = text_freq.get(char, 0)

        current_diff = sum(
            abs(shifted_counter.get(char, 0) - frequency_portuguese.get(char, 0)) for char in string.ascii_uppercase)

        if current_diff < best_diff:
            best_diff = current_diff
            best_k = k
            decrypted_text = ''.join(
                chr((ord(char) - ord('A') - best_k) % 26 + ord('A')) if char in string.ascii_uppercase else char
                for char in text
            )

    return decrypted_text, best_k
```

Figure 4. Análise de Frequência das Letras

Esse código tenta decifrar um texto criptografado pela cifra de César, usando uma análise de frequência para identificar o valor de deslocamento (k) que melhor alinha as frequências de letras do texto com as frequências típicas da língua portuguesa.

1. A função frequency_analysis conta a ocorrência de cada letra no texto e calcula a frequência percentual de cada uma.
2. Para cada possível deslocamento (k de 0 a 25), a função simula a decifragem do texto aplicando o deslocamento reverso às frequências das letras.
3. Em cada simulação, calcula a diferença entre as frequências do texto ajustado e as frequências esperadas para o português. O valor de k que gera a menor diferença é selecionado como o melhor deslocamento.
4. A função então decifra o texto com o melhor k encontrado e retorna o texto decifrado e o valor de k.

```python
import string


def brute_force(cipher_text):  2 usages
    for k in range(26):  # Tenta todos os deslocamentos possíveis
        decrypted_text = ''
        for char in cipher_text.upper():
            if char in string.ascii_uppercase:
                decrypted_text += chr((ord(char) - ord('A') - k) % 26 + ord('A'))
            else:
                decrypted_text += char

        if k == 3:
            return decrypted_text
```

Figure 5. Ataque de Força Bruta

Esse código realiza uma força bruta para decifrar um texto criptografado com a cifra de César, testando todos os valores possíveis de deslocamento (k) entre 0 e 25. A função brute_force aplica um deslocamento reverso a cada letra do texto e tenta decifrar a mensagem. No entanto, uma peculiaridade é que o código interrompe o processo e retorna o texto decifrado assim que o valor de k chega a 3, ao invés de continuar testando todos os valores possíveis.

1. A função itera k de 0 a 25, simulando a tentativa de decifrar o texto para cada valor de deslocamento.
2. Para cada caractere do texto (cipher_text), verifica se é uma letra maiúscula do alfabeto
3. Se for uma letra, aplica o deslocamento reverso -k para "voltar" a letra k posições no alfabeto, com o cálculo (ord(char) - ord('A') - k) % 26 + ord('A').
4. Se o caractere não for uma letra, é adicionado inalterado à string decrypted_text.
5. A função está configurada para retornar o texto decifrado quando k é igual a 3, interrompendo o loop. Assim, a função sempre retorna o resultado para o deslocamento k = 3 e ignora os demais valores.

Este código, na prática, só tenta decifrar a cifra com um deslocamento fixo de 3, ao invés de uma busca completa de força bruta.

2.1.3. Análise da Atividade 1

Os códigos explicados anteriormente implementam métodos de criptografia e decifração usando a cifra de César, um tipo de substituição de letras que desloca cada caractere de um texto por uma quantidade fixa. Esses métodos incluem uma função que aplica o deslocamento para criptografar o texto, uma análise de frequência para tentar decifrá-lo, e um método de força bruta que, em teoria, testaria todos os valores possíveis de deslocamento. No entanto, a eficácia da decifração é significativamente impactada pelo tamanho do texto fornecido.

A análise de frequência é um método confiável para decifrar a cifra de César, mas funciona melhor em textos grandes. Isso ocorre porque, para identificar o deslocamento correto, o programa compara as frequências das letras do texto criptografado com as frequências padrão das letras em português. Quanto maior o texto, mais preciso é esse cálculo de frequência, já que haverá uma quantidade maior de caracteres que permite uma média mais próxima do padrão da língua. Em textos curtos, as frequências observadas podem variar muito, pois a presença ou ausência de certas letras pode gerar padrões que não correspondem ao esperado, levando a uma decifração incorreta.

Aqui está um exemplo de como o output é retornado:

```
Digite o texto a ser cifrado: Testando a atividade um
Texto original: Testando a atividade um
Texto cifrado: WHVWDQGR D DWLYLGDGH XP
Texto decifrado (por análise de frequência): TESTANDO A ATIVIDADE UM
Texto decifrado (por força bruta): TESTANDO A ATIVIDADE UM
```

Figure 6. Output

2.2. Atividade 2: Quebrando Cifra por Transposição

Na criptografia clássica, uma cifra de transposição altera a posição de cada letra (ou outro símbolo) no texto original, mudando a ordem dos caracteres sem modificá-los individualmente. A decifração é realizada invertendo essa transposição, restaurando a ordem original dos caracteres. Matematicamente, a cifra de transposição aplica uma função bijetiva para cifrar e sua função inversa para decifrar, garantindo que cada posição do texto cifrado possa ser mapeada de volta para o texto original. [2]

Aqui, um código ‘main’ também foi implementado, permitindo o usuário a inserir um texto. Ele aplica uma chave de transposição fixa para cifrar o texto e, em seguida, tenta “quebrar” a cifra ao encontrar a chave correta por meio de uma função de decifragem.

```python
from transposition_encrypt import transposition_encrypt
from transposition_decrypt import break_transposition


def main():  1 usage
    plaintext = input('Digite o texto a ser cifrado: ')
    encrypted_text = transposition_encrypt(plaintext, key: '31452')
    print('Texto original: ' + plaintext)
    print('Texto cifrado: ' + encrypted_text)

    # Tentativa de quebra da cifra
    best_key, decrypted_text = break_transposition(encrypted_text)
    print('Texto decifrado: ' + decrypted_text)
    print('Chave encontrada: ' + best_key)


if __name__ == '__main__':
    main()
```

Figure 7. Main da Atividade 2

Esse código ilustra o processo de criptografia e quebra de cifras de transposição, demonstrando como uma ordem específica dos caracteres pode proteger o conteúdo, mas também como a recuperação do texto original é possível ao descobrir a chave correta.

2.2.1. Parte 1: Realizar a Cifra por Transposição

```python
def transposition_encrypt(plaintext, key):  2 usages
    num_columns = len(key)
    num_rows = (len(plaintext) + num_columns - 1) // num_columns
    plaintext = plaintext.ljust(num_columns * num_rows, 'X')

    matrix = [plaintext[i:i + num_columns] for i in range(0, len(plaintext), num_columns)]

    permuted_text = ''
    for col in sorted(range(num_columns), key=lambda x: key[x]):
        for row in matrix:
            permuted_text += row[col]

    return permuted_text
```

Figure 8. Criptografia por Transposição

A função ‘transposition_encrypt’ implementa uma criptografia por transposição que reorganiza os caracteres do texto original (‘plaintext’) com base em uma chave de permutação (‘key’).

Primeiro, a função define o número de colunas da matriz como o comprimento da chave e calcula o número de linhas necessárias para acomodar todo o texto. Se o número de caracteres do texto não for múltiplo do número de colunas, o texto é preenchido com ‘X’ para garantir um encaixe completo.

Em seguida, o texto é dividido em linhas de tamanho igual ao número de colunas, criando uma matriz. A função então permuta as colunas da matriz com base na chave: a chave determina a nova ordem das colunas, e essa permutação é aplicada ao conteúdo das linhas. Por fim, o texto criptografado é gerado ao concatenar os caracteres na nova ordem e é retornado como o resultado.

2.2.2. Parte 2: Quebrar a Cifra por Transposição

O código dessa parte usa uma série de funções para decriptografar um texto criptografado por transposição, utilizando uma chave inicial e um cálculo de frequência para identificar a chave correta para a decriptação.

```
from itertools import permutations
from frequency_pt import frequency_portuguese
```

Figure 9. Importações

Começando pelas importações:

- **`from itertools import permutations`**: Importa a função permutations, que gera todas as permutações possíveis de uma sequência. Aqui, é usada para testar várias chaves de decriptação possíveis, assumindo que a chave original é composta por uma sequência de dígitos.
- **`from frequency_pt import frequency_portuguese`**: Importa a tabela de frequências das letras na língua portuguesa, usada para calcular a similaridade entre a frequência de letras no texto decriptografado e as frequências comuns no português. Essa tabela é essencial para avaliar a plausibilidade do texto gerado.

```python
def calculate_frequency(text):  1 usage
    frequency = {chr(i): 0 for i in range(65, 91)}  # Letras maiúsculas A-Z
    for char in text.upper():
        if char in frequency:
            frequency[char] += 1

    total = sum(frequency.values())
    for letter in frequency:
        frequency[letter] = (frequency[letter] / total) * 100 if total > 0 else 0
    return frequency
```

Figure 10. calculate_frequency

Esta função calcula a frequência de cada letra de A a Z no texto fornecido. Primeiro, ela inicializa um dicionário com todas as letras de A a Z, cada uma com contagem inicial de zero. Em seguida, converte o texto para maiúsculas e incrementa a contagem de cada letra encontrada. Por fim, calcula a frequência percentual de cada letra dividindo a contagem pelo total de letras e multiplicando o resultado por 100 (ou atribui 0% caso o total seja zero).

```python
def score_text(text):  1 usage
    text_freq = calculate_frequency(text)
    return sum(abs(text_freq[letter] - frequency_portuguese[letter]) for letter in text_freq)
```

Figure 11. score_text

Esta função calcula uma pontuação que indica o quão próxima está a frequência das letras no texto decriptografado em relação à frequência média do português. Ela utiliza a função 'calculate_frequency' para obter a frequência de cada letra no texto fornecido. Em seguida, para cada letra, calcula a diferença absoluta entre a frequência no texto e a frequência média do português ('frequency_portuguese'). Por fim, retorna a soma dessas diferenças, onde um valor menor indica que o texto está mais alinhado com as características da língua portuguesa.

```python
def transposition_decrypt(encrypted_text, key):  1 usage
    num_columns = len(key)
    num_rows = len(encrypted_text) // num_columns

    columns = [''] * num_columns
    index = 0
    for col in sorted(range(num_columns), key=lambda x: key[x]):
        columns[col] = encrypted_text[index:index + num_rows]
        index += num_rows

    plaintext = ''
    for i in range(num_rows):
        for col in range(num_columns):
            plaintext += columns[col][i]

    return plaintext
```

Figure 12. transposition_decrypt

Esta função decriptografa o texto cifrado usando uma chave de transposição específica. Primeiro, calcula o número de colunas com base no comprimento da chave e determina o número de linhas dividindo o comprimento do texto pelo número de colunas. Em seguida, cria uma lista ‘columns‘ para armazenar cada coluna do texto conforme a ordem estabelecida pela chave. As colunas são reorganizadas com base na chave, e os caracteres do texto cifrado são distribuídos entre elas. Finalmente, o texto decriptografado é construído lendo sequencialmente cada linha da matriz reorganizada, resultando na mensagem original.

```python
def break_transposition(encrypted_text):  2 usages
    best_score = float('inf')
    best_key = None
    best_decryption = ''

    for perm in permutations('31452'):
        decrypted_text = transposition_decrypt(encrypted_text, perm)
        score = score_text(decrypted_text)

        if score < best_score:
            best_score = score
            best_key = ''.join(perm)
            best_decryption = decrypted_text

    return best_key, best_decryption.rstrip('X')
```

Figure 13. break_transposition

Essa função busca identificar a chave correta para decriptação usando uma abordagem de força bruta, testando todas as permutações possíveis de uma sequência de chaves. Ela começa inicializando variáveis para armazenar a melhor pontuação (‘best_score’), a melhor chave (‘best_key’) e o melhor texto decriptografado (‘best_decryption’). Em seguida, gera todas as permutações possíveis da sequência “31452”, que representam diferentes chaves potenciais.

Para cada permutação, a função decripta o texto com ‘transposition_decrypt’ e calcula a pontuação do resultado usando ‘score_text’. Se a pontuação do texto decriptado for menor do que a melhor pontuação registrada, as variáveis de melhor pontuação, chave e texto são atualizadas. Ao final, a função retorna a melhor chave e o texto decriptografado, removendo caracteres ‘X’ adicionais que possam ter sido adicionados no processo de criptografia.

2.2.3. Análise da Atividade 2

Em todos os testes, a criptografia e decriptografia ocorreu como o esperado.

Aqui está um exemplo de como o output é retornado:

```
Digite o texto a ser cifrado: Testando a atividade dois
Texto original: Testando a atividade dois
Texto cifrado: edaddaavesTn i sotaot idi
Texto decifrado: Testando a atividade dois
Chave encontrada: 31452
```

Figure 14. Output

3. Conclusão

Para concluir este relatório, destaca-se a relevância das cifras de substituição e transposição para a compreensão dos fundamentos da criptografia. Através da implementação prática das cifras de César e de transposição, foi possível explorar métodos de encriptação e decriptação clássicos, bem como comparar suas eficiências e limitações. O uso de técnicas como análise de frequência e força bruta para quebrar essas cifras revelou-se uma abordagem valiosa para testar e validar a segurança de tais métodos. Conclui-se que, embora estas cifras ofereçam um entendimento básico sobre criptografia, seu nível de segurança é insuficiente para cenários modernos, onde técnicas mais robustas são necessárias para a proteção eficaz dos dados.

References

[1] Wikipédia. Cifra de césar — wikipédia, a enciclopédia livre. https://pt.wikipedia.org/wiki/Cifra_de_Csar, 2022. [Online; accessed 2024-11-05].

[2] Wikipédia. Cifra de transposição — wikipédia, a enciclopédia livre. https://pt.wikipedia.org/wiki/Cifra_de_transposio, 2023. [Online; accessed 2024-11-06].