Shift Cipher e Transposição

Gustavo de Sousa Santos - 222011534

Segurança Computacional - Turma 02

Link do repositório com os códigos: Repositóriop

Descrição

Este projeto implementa algoritmos para cifrar e decifrar mensagens usando o algoritmo de **shift cipher** e transposição colunar. A descriptografia do algoritmo shift cipher e da transposição colunar ocorre por **força bruta** e análise por **distribuição de frequências**.

Introdução

Esse projeto utiliza-se de duas abordagens principais para cifrar e decifrar:

1. **Shift Cipher**: Cada letra é substituída por outra com base em uma chave de deslocamento fixa.
2. **Transposição Colunar**: A posição das letras é alterada de acordo a chave informada pelo usuário (tamanho das colunas).

Shift Cipher

O algoritmo mapeia as letras do alfabeto em inteiros módulo 26, sendo `A = 0` e `Z = 25`. Diante de uma chave especificada, a criptografia se dá pelo deslocamento de todas as letras no valor escolhido.

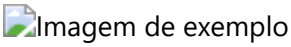
```
def shift_cipher(texto, chave):
    resultado = ""
    for char in texto:
        if char.isalpha():
            base_char = unidecode(char)
            base = ord('A') if char.isupper() else ord('a')
            deslocado = (ord(base_char.lower()) - ord('a') + chave) % 26
            novo_char = chr(base + deslocado)
```

```
            resultado += novo_char.upper() if char.isupper() else novo_char
        else:
            resultado += char
    return resultado
```

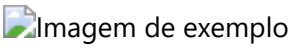
1. Recebe um texto e a chave que indica o tamanho do deslocamento
2. Itera sobre todo o texto, verificando se o caracter trata-se de uma letra e se é um caracter acentuado.
3. Desloca o caracter e o adiciona à string resultado.
4. Retorna resultado que é o texto cifrado.

- **Complexidade**: Aproximadamente O(n), onde n é a quantidade de letras no texto.

Exemplo:

 Imagem de exemplo

Descriptografia com a chave:

 Imagem de exemplo

Descriptografia do Shift Cipher

Força Bruta

A descriptografia por força bruta tenta todos os possíveis deslocamentos e associa-os com um texto correspondente, dando como saída uma lista com esses valores. Este método pode ser demorado para problemas que contenham mensagens muito grandes, mas se torna viável neste caso por se limitar aos 26 caracteres do alfabeto romano.

```
def forca_bruta(mensagem_criptografada):
    print("\nTentativas de descriptografia por força bruta:\n")
    for chave in range(1, 27):
        tentativa = shift_cipher(mensagem_criptografada, -chave)
        print(f"Deslocamento {chave:2}: {tentativa}")
```

1. Repete 26 vezes, buscando cada possibilidade de deslocamento.
2. Itera sobre cada letra do texto, retirando a diferença entre as letras e as chaves para achar o novo caractere.
3. Retorna todas as possíveis respostas.

- **Complexidade**: Aproximadamente `O(nm)`, onde `n` é o número de caracteres únicos da chave e `m` o espaço amostral de caracteres, 26 nesse caso.

- **Tempo de Execução**: Pode ser muito alto para grandes mensagens, tornando a força bruta pouco viável em textos extensos e com grande variedades de caracteres.

Exemplo:  Imagem de exemplo

Distribuição de Frequências

Esta técnica compara a letra mais recorrente do ciphertext com as mais frequentes da língua portuguesa, obtendo-se o deslocamento da mensagem a partir da diferença na posição no alfabeto entre as letras. A análise é feita com base nas letras `A`, `E`, `O`, `R` e `S` que são as letras mais frequentes na língua portuguesa.

```
def substituicao_por_frequencia_simples(mensagem_criptografada):
    print("\nTentativas de descriptografia por substituição simples (letras mais
frequentes do português):\n")
    texto_sem_espacos = ''.join(c for c in mensagem_criptografada if c.isalpha())
    if not texto_sem_espacos:
        print("Mensagem inválida ou vazia.")
        return

    letra_mais_frequente = max(set(unidecode(texto_sem_espacos.upper())),
key=unidecode(texto_sem_espacos.upper()).count)
    letras_comuns_pt = ['E', 'A', 'O', 'S', 'R']

    for letra_pt in letras_comuns_pt:
        desloc = (letras_para_numeros[letra_mais_frequente] -
letras_para_numeros[letra_pt]) % 26
        tentativa = shift_cipher(mensagem_criptografada, -desloc)
        print(f"Assumindo que '{letra_mais_frequente}' = '{letra_pt}' →
Deslocamento {desloc:2}: {tentativa}")
```

1. Trabalha com 5 possibilidades de letras mais frequêntes, `A`, `E`, `O`, `S` e `R`.
2. Reduz a diferença entre o inteiro associado à letra recorrente do ciphertext e as letras recorrentes do português, para obter o deslocamento da mensagem.
3. Itera sobre cada letra do texto e concatena sua versão pós deslocamento com cada um dos 5 plaintext.
4. Retorna as 5 possibilidades de mensagens, e seus deslocamentos associados.

- **Complexidade**: Aproximadamente `O(n)`, onde `n` é a quantidade de letras no texto.
- **Tempo de Execução**: Geralmente mais rápido que a força bruta, mas depende da correspondência com as frequências do idioma.

Exemplo:

 Imagem de exemplo

Transposição Colunar

Criptografia

No método de **transposição colunar**, a mensagem é organizada em uma matriz a partir de uma chave específica (número inteiro fornecido pelo usuário), associando-se cada coluna com uma letra da chave. A mensagem cifrada dá-se associando as colunas uma a uma, começando pela coluna 0 e indo até o tamanho da chave informada pelo usuário.

```
def cifra_transposicao(mensagem, chave):
    mensagem = unidecode(mensagem)
    colunas = [''] * chave
    for i in range(len(mensagem)):
        col = i % chave
        colunas[col] += mensagem[i]
    return ''.join(colunas)
```

1. Cria uma matriz com a quantidade de colunas definida pela chave do usuário.
2. Distribui a mensagem linha a linha da matriz criada.
3. Retorna todas as colunas juntas que forma o texto cifrado.

- **Complexidade**: Aproximadamente `O(nm)`, onde `n` é a quantidade de letras no texto e `m` o tamanho da chave.
- **Técnica de Permutação**: A mensagem é organizada em uma matriz e ordenada conforme a chave.

Exemplo:  Imagem de exemplo

Descriptografia com a chave:  Imagem de exemplo

Descriptografia por força bruta

O programa busca todas as alternativas possíveis baseadas no tamanho do texto cifrado, testando um número de colunas de 2 à len(mensagem) + 1.

```
def forca_bruta(mensagem):
    print("\nTentativas de descriptografia por força bruta:\n")
    for chave in range(2, len(mensagem) + 1):
        tentativa = decifra_transposicao(mensagem, chave)
        print(f"Chave {chave:2}: {tentativa}")
```

1. Testa todos as possíveis chaves.
2. Chama a função decifra_transposicao para retornar a mensagem descriptografada de acordo com a possível chave.

3. Retorna todas as possíveis possibilidades indo de 2 até len(mensagem) + 1.

- **Complexidade**: Aproximadamente `O(n)`, onde `n` é o tamanho da mensagem.
- **Tempo de Execução**: Pode ser muito grande para mensagens extensas, porém apresenta um bom desempenho.

 Imagem de exemplo ...  Imagem de exemplo

Descriptografia com a chave

```
def decifra_transposicao(mensagem, chave):
    mensagem = unidecode(mensagem)
    num_colunas = chave
    num_linhas = len(mensagem) // chave
    if len(mensagem) % chave != 0:
        num_linhas += 1

    colunas = []
    tamanho_coluna_base = len(mensagem) // chave
    num_colunas_maiores = len(mensagem) % chave

    idx = 0
    for i in range(chave):
        tamanho_coluna = tamanho_coluna_base + (1 if i < num_colunas_maiores else
0)
        colunas.append(mensagem[idx:idx + tamanho_coluna])
        idx += tamanho_coluna

    resultado = ''
    for i in range(num_linhas):
        for col in colunas:
            if i < len(col):
                resultado += col[i]
    return resultado
```

1. Cria uma matriz de acordo com o tamanho total da mensagem.
2. Remonta a matriz por colunas.
3. Lê os caracteres linha por linha remontando a mensagem original.

- **Complexidade**: Aproximadamente `O(n)`, onde `n` é o tamanho da mensagem.

Descriptografia Distribuição de Frequência

Para a criptografia por transposição colunar, não foi implementado um algoritmo de descriptografia por distribuição de frequência, principalmente por problemas que somente a partir da frequência das letras era muito difícil entender qual era a chave que foi utilizada. Sendo assim, esse provou-se ser um método mais seguro contra a descriptografia do que o shift cipher.

Conclusão

O desenvolvimento deste projeto forneceu uma visão abrangente sobre a implementação e análise do algoritmo de shift cipher e transposição colunar. A utilização dos métodos de substituição e transposição permitiu observar as forças e limitações de cada técnica em termos de segurança, complexidade computacional e viabilidade de uso.

Os resultados demonstraram que:

- **Força Bruta** é um método eficaz para mensagens curtas, demonstrando utilidade quando limitado ao alfabeto romano.
- **Distribuição de Frequências** se mostrou mais eficiente que o método de força bruta, pois reduz drásticamente a quantidade de análises, limitando-se a um grupo pequeno de caracteres escolhidos pelo programador.
- **Transposição Colunar** provou-se como o método mais difícil de ser quebrado, com dificuldades para se implementar um algoritmo simples de quebra de cifra por distribuição de frequência, ao passo que o método de quebra por força bruta apresenta um maior tempo de execução quando comparado ao shift cipher, visto que ele baseia-se no tamanho total da mensagem, podendo levar grande tempo para ser completado se o texto cifrado for muito extenso..

O uso de qual o melhor algoritmo para encriptação é uma escolha do programador, porém o algoritmo de transposição colunar apresenta ser mais difícil para o atacante que tentar decifrá-lo quando se tratar de uma mensagem mais longa.

Em conclusão, a escolha do algoritmo deve considerar tanto o nível de segurança desejado quanto a eficiência necessária para a aplicação. Esta análise serve como um guia para entender como essas técnicas podem ser usadas e otimizadas, além de fornecer preparo para algoritmos ainda mais complexos e inteligentes que estão por vir.

Referências

- https://www.dcc.fc.up.pt/~rvr/naulas/tabelasPT/
- https://wiki.imesec.ime.usp.br/books/criptografia/page/cifras-de-transposi%C3%A7%C3%A3o
- https://crypto.stackexchange.com/questions/1550/obtaining-the-key-length-of-a-columnar-transposition-given-a-known-plaintext-wo/1578#1578