Universidade de Brasília
Departamento de Ciência da Computação

Disciplina: CIC0201 – Segurança Computacional – Turma 02 - 2024/2.
Prof. Lorena de Souza Bezerra Borges.

Integrantes: Emanuel de Oliveira Barbosa. **Matrícula:** 211010403.

Lista de Exercícios 01 - LE01

Exercício 1

Implementação de uma cifra de deslocamento (Cifra de César) com k =3, além de duas técnicas para decodificar o texto cifrado (CipherText-only): uma baseada na distribuição de frequência das letras em português e outra por força bruta, testando todos os deslocamentos possíveis.

Repositório: https://github.com/emanuelob/CIC0201_LE01

É de conhecimento que a Cifra de César faz com que cada letra de um texto seja substituída por outra letra através de um deslocamento sequencial, preservando o tamanho original do texto. Assim, o processo inverso é de fácil decodificação se:

1. Conhecemos o número de deslocamento: bastar voltar o número de "casas" deslocadas da *String*.
2. Utilizamos uma tabela de frequência da língua: também é conhecido a frequência de letras nas palavras em português, por exemplo. Basta observar a distribuição de frequência de letras, comparando a distribuição de frequência do texto cifrado com a distribuição de frequência da língua-alvo (https://www.dcc.fc.up.pt/~rvr/naulas/tabelasPT/). Ao final, podemos tentar deduzir o deslocamento (chave) utilizado na cifra por meio da menor diferença de deslocamento encontrado (o quanto está desviado do padrão).
3. Ataque de força bruta: testar todas as combinações possíveis. São 26 letras do alfabeto que geram 26 deslocamentos possíveis.

Sobre a complexidade de algoritmos, podemos considerar que:

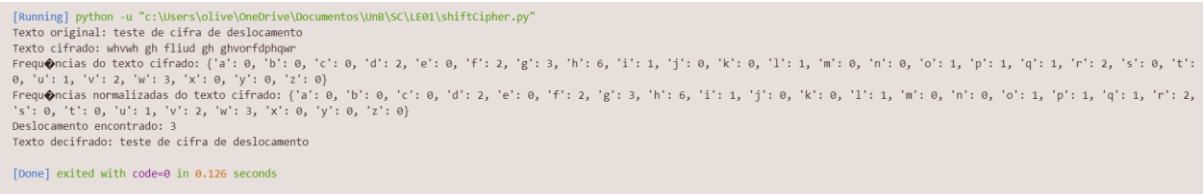
- Cifrar: O(n), onde n é o tamanho do texto a ser cifrado, uma vez que ela precisa verificar e deslocar cada caractere individualmente.
- Decifrar: O(n), onde n é o tamanho do texto a ser cifrado.
- Ataque por distribuição de frequência: O(k * n), onde k é o tamanho do alfabeto e n é o tamanho do texto cifrado.
- Ataque de força bruta: O(k * n), onde k é o tamanho do alfabeto e n é o tamanho do texto cifrado.

Para codificar o texto, temos que a operação ocorreu em 0.18 segundos:

```
[Running] python -u "c:\Users\olive\OneDrive\Documentos\UnB\SC\LE01\shiftCipher.py"
Texto original: teste de cifra de deslocamento
Texto cifrado: whvwh gh fliud gh ghvorfdphqwr

[Done] exited with code=0 in 0.18 seconds
```

Para decodificar o texto por distribuição de frequência, temos que a operação ocorreu em 0.126 segundos:



Para decodificar o texto por ataque de força bruta, temos que a operação ocorreu em 0.119 segundos:

```
[Running] python -u "c:\Users\olive\OneDrive\Documentos\UnB\SC\LE01\shiftCipher.py"
Texto original: teste de cifra de deslocamento
Texto cifrado: whvwh gh fliud gh ghvorfdphqwr
Tentativas de decifra&#65533;&#65533;o usando todos os deslocamentos poss&#65533;veis:

Deslocamento 0: whvwh gh fliud gh ghvorfdphqwr
Deslocamento 1: vguvg fg ekhtc fg fgunqecogpvq
Deslocamento 2: uftuf ef djgsb ef eftmpdbnfoup
Deslocamento 3: teste de cifra de deslocamento
Deslocamento 4: sdrsd cd bheqz cd cdrknbzldmsn
Deslocamento 5: rcqrc bc agdpy bc bcqjmaykclrm
Deslocamento 6: qbpqb ab zfcox ab abpilzxjbkql
Deslocamento 7: paopa za yebnw za zaohkywiajpk
Deslocamento 8: oznoz yz xdamv yz yzngjxvhzioj
Deslocamento 9: nymny xy wczlu xy xymfiwugyhni
Deslocamento 10: mxlmx wx vbykt wx wxlehvtfxgmh
Deslocamento 11: lwklw vw uaxjs vw vwkdgusewflg
Deslocamento 12: kvjkv uv tzwir uv uvjcftrdvekf
Deslocamento 13: juiju tu syvhq tu tuibesqcudje
Deslocamento 14: ithit st rxugp st sthadrpbtcid
Deslocamento 15: hsghs rs qwtfo rs rsgzcqoasbhc
Deslocamento 16: grfgr qr pvsen qr qrfybpnzragb
Deslocamento 17: fqefq pq ourdm pq pqexaomyqzfa
Deslocamento 18: epdep op ntqcl op opdwznlxpyez
Deslocamento 19: docdo no mspbk no nocvymkwoxdy
Deslocamento 20: cnbcn mn lroaj mn mnbuxljvnwcx
Deslocamento 21: bmabm lm kqnzi lm lmatwkiumvbw
Deslocamento 22: alzal kl jpmyh kl klzsvjhtluav
Deslocamento 23: zkyzk jk iolxg jk jkyruigsktzu
Deslocamento 24: yjxyj ij hnkwf ij ijxqthfrjsyt
Deslocamento 25: xiwxi hi gmjve hi hiwpsgeqirxs

[Done] exited with code=0 in 0.119 seconds
```

Finalmente, explicando trechos de código:

1. **Codificação**

```python
def cifrar(texto):
    texto_cifrado = ''

    for char in texto.lower():
        if char in alfabeto:
            indice = alfabeto.index(char)
            novo_indice = (indice + 3) % len(alfabeto) #aritmética modular (resto) com k=3
            texto_cifrado += alfabeto[novo_indice]
        else:
            texto_cifrado += char
    return texto_cifrado
```

A função percorre cada caractere do texto, verifica se o caractere é uma letra e, em caso afirmativo, calcula o novo índice usando uma aritmética modular de deslocamento k=3. Se o caractere não for uma letra, como um espaço (" "), ele é adicionado ao texto cifrado sem alterações.

2. **Decodificação por distribuição de frequência (Linhas 17 a 76)**

Calcula a frequência de cada letra no texto cifrado, gerando um dicionário com o número de ocorrências de cada letra. Em seguida, *ataque_frequencia(texto_cifrado)* utiliza essa frequência e compara com a distribuição de frequência típica das letras em português, fornecida pela variável *frequencias_caracteres*. Cumpre destacar que *frequencias_caracteres* é simplesmente um dicionário contendo as percentagens de frequência dos caracteres em português, disponibilizadas no endereço https://www.dcc.fc.up.pt/~rvr/naulas/tabelasPT/.

Aqui, a ideia é testar todos os possíveis deslocamentos (de 0 a 25), deslocando as letras do texto cifrado e somando as diferenças absolutas entre as frequências reais e as esperadas. As frequências das letras cifradas devem se aproximar das frequências esperadas das letras originais e o deslocamento que minimiza essa diferença é considerado o mais provável.

A função *ataque_frequencia(texto_cifrado)* retorna o valor do deslocamento.

3. **Decodificação por ataque de força bruta (Linhas 90 a 94)**

Por fim, a função *ataque_forca_bruta(texto_cifrado)* tenta decifrar o texto com todos os deslocamentos possíveis (de 0 a 25) e imprime cada tentativa, permitindo a análise manual do texto decifrado.

Conclusão

O tempo de execução dependerá do tamanho do texto e do tamanho do alfabeto. Em termos de viabilidade, o ataque de força bruta é menos sofisticado e adequado para textos curtos, pois imprime todas as possibilidades sem tentar identificar padrões. Por outro lado, o ataque por distribuição de frequência é mais eficiente em textos longos, onde a distribuição de letras é mais próxima da esperada, aumentando a precisão sem exigir uma verificação manual de todas as tentativas.