

Relatório – Implementação da Cifra de Vigenère utilizando a tabela ASCII

1. Introdução

Este trabalho implementa a **cifra de Vigenère** usando toda a tabela ASCII (0–255) para encriptação e decriptação. A saída encriptada é sempre uma **string legível** (Base64), permitindo visualização direta no terminal.

Para mais detalhes, consulte o relatório completo.

2. Estrutura do projeto

project/

```
cipher/  
    __init__.py  
    encrypt.py  
    decrypt.py
```

```
main.py  
testes.txt
```

- cipher/encrypt.py → função `encrypt(plaintext, key)`
- cipher/decrypt.py → função `decrypt(ciphertext, key)`
- main.py → interface para o usuário
- testes.txt → plano de testes com 5 casos em português

3. Desenvolvimento

3.1. Função de encriptação

- Converte cada caractere do texto em código ASCII.
- Soma o valor do caractere com o correspondente da chave (ciclo repetitivo) **módulo 256**.
- Converte os valores em bytes e depois em **Base64**.
- Retorna a string Base64 legível.

3.2. Função de decriptação

- Recebe a string Base64 encriptada e a chave.
- Decodifica Base64 em bytes.
- Subtrai o valor do byte com a chave correspondente **módulo 256**.
- Converte os valores de volta em caracteres.
- Retorna o texto original.

3.3. Função main

- Solicita ao usuário que digite o texto a ser encriptado.
- Solicita ao usuário que digite a chave utilizada na cifra.
- Chama a função `encrypt(text, key)` para gerar a string encriptada em Base64.
- Exibe no terminal o texto encriptado, garantindo que seja legível.
- Chama a função `decrypt(encrypted, key)` para recuperar o texto original.
- Exibe no terminal o texto decryptado, confirmando que a encriptação e decryptação funcionaram corretamente.
- A função `main()` é protegida pelo bloco `if __name__ == "__main__":`, garantindo que o código seja executado apenas quando o script for rodado diretamente, e não quando importado como módulo.

3.4. Plano de Testes Resumido

- O arquivo `testes.txt` contém cinco casos para validar a cifra de Vigenère
 - a. Caso simples;
 - a. Caso com texto longo;
 - b. Chave maior que o texto;
 - c. Com caracteres especiais;
 - d. Caso com caracteres ASCII altos;
 - e. Caso texto vazio.

4. Como executar

- Abra o terminal na pasta do projeto.
- Execute o programa:

```
python main.py
```

- Insira o **texto** e a **chave** quando solicitado

```
Digite o seu texto: BATATA
```

```
Digite a sua chave: CHAVE
```

- O terminal exibirá (Exemplo de saída):

```
Encriptado: hYmVl5mE  # string Base64 legível
```

```
Decryptado: BATATA    # texto original
```

5. Considerações finais

- O uso de Base64 garante **saída legível** mesmo com caracteres não imprimíveis do ASCII.
- O algoritmo preserva acentos, símbolos e caracteres especiais.

- O plano de testes garante confiabilidade, contemplando casos simples e extremos.