



INSTITUTO POLITÉCNICO DE BEJA
Escola Superior de Tecnologia e Gestão
Mestrado em Engenharia de Segurança Informática
Criptografia e Criptanalise Aplicadas

(De)cipher™

Martinho José Novo Caeiro - 23917
Paulo António Tavares Abade - 23919



INSTITUTO POLITÉCNICO DE BEJA
Escola Superior de Tecnologia e Gestão
Mestrado em Engenharia de Segurança Informática
Criptografia e Criptanalise Aplicadas

(De)cipher™

Martinho José Novo Caeiro - 23917
Paulo António Tavares Abade - 23919

Orientador: Rui Miguel Silva Daniel José Da Graça Peceguinha Franco

Beja, novembro de 2025

Resumo

Este relatório descreve o desenvolvimento de uma aplicação de cifra e decifra de ficheiros, implementada em Python, uma linguagem de alto nível. O objetivo é explorar cifras de criptografia simétrica e demonstrar a aplicação prática das mesmas. Esta aplicação é desenvolvida no âmbito da unidade curricular de Criptografia e Criptanalise Aplicadas IPBeja, 2025.

Keywords: python, criptanalise, criptografia simétrica

Abstract

This report describes the development of a file encryption and decryption application, implemented in Python, a high-level language. The objective is to explore symmetric encryption ciphers and demonstrate their practical application. This application is developed within the scope of the course Cryptography and Applied Cryptanalysis IPBeja, 2025.

Keywords: python, cryptanalysis, symmetric cryptography

Índice

Índice de Figuras

1 Introdução

Esta aplicação consistirá na junção de 4 cifras de criptografia simétrica, AES, DES, Vigenère e PlayFair. A aplicação será desenvolvida em Python (Python Software Foundation, 2025) e tem como base uma interface gráfica simples (GUI), permitindo ao utilizador interagir com o sistema de forma simples e eficiente. Todo o progresso do projeto será documentado no repositório GitHub (Martinho Caeiro & Paulo Abade, 2025).

2 Teoria

2.1 AES

O AES (Advanced Encryption Standard) (Wikipedia, 2025a) é um padrão de criptografia simétrica amplamente utilizado para proteger dados. Ele utiliza blocos de 128 bits e chaves de 128, 192 ou 256 bits, oferecendo alta segurança e eficiência.

2.2 DES

O DES (Data Encryption Standard) (Wikipedia, 2025d) é um algoritmo de criptografia simétrica que foi amplamente utilizado no passado. Ele opera em blocos de 64 bits e utiliza uma chave de 56 bits. Embora tenha sido substituído por algoritmos mais seguros, o DES ainda é relevante para fins educacionais.

2.3 Vigenère

A cifra de Vigenère (Wikipedia, 2025c) é um método de criptografia que utiliza uma palavra-chave para cifrar o texto. Ela é baseada na cifra de César, mas em vez de usar um único deslocamento, ela aplica deslocamentos diferentes com base nas letras da palavra-chave.

2.4 PlayFair

A cifra de PlayFair (Wikipedia, 2025b) é um método de criptografia que utiliza uma matriz de 5x5 para cifrar pares de letras. Ela é mais segura do que a cifra de César, pois não utiliza um deslocamento fixo.

3 Desenvolvimento da Aplicação

Todos os seguintes scripts foram desenvolvidos em Python com o uso do VS Code (Microsoft Corporation, 2025) e todos verificam se a maquina possui os pacotes necessários para o funcionamento da ferramenta, irá perguntar se quer instalar para poder continuar a utilizar. Caso o utilizador deseje prosseguir, o script irá instalar os pacotes necessários e depois irá ativá-los.

3.1 GUI

A GUI é o metodo principal de interação com a ferramenta. Para aceder a um dos metodos de cifra/decifra, o utilizador deve selecionar a aba correspondente. Para executar a GUI apenas é necessário escrever 'python gui.py'.

A GUI apresenta-se da seguinte forma:

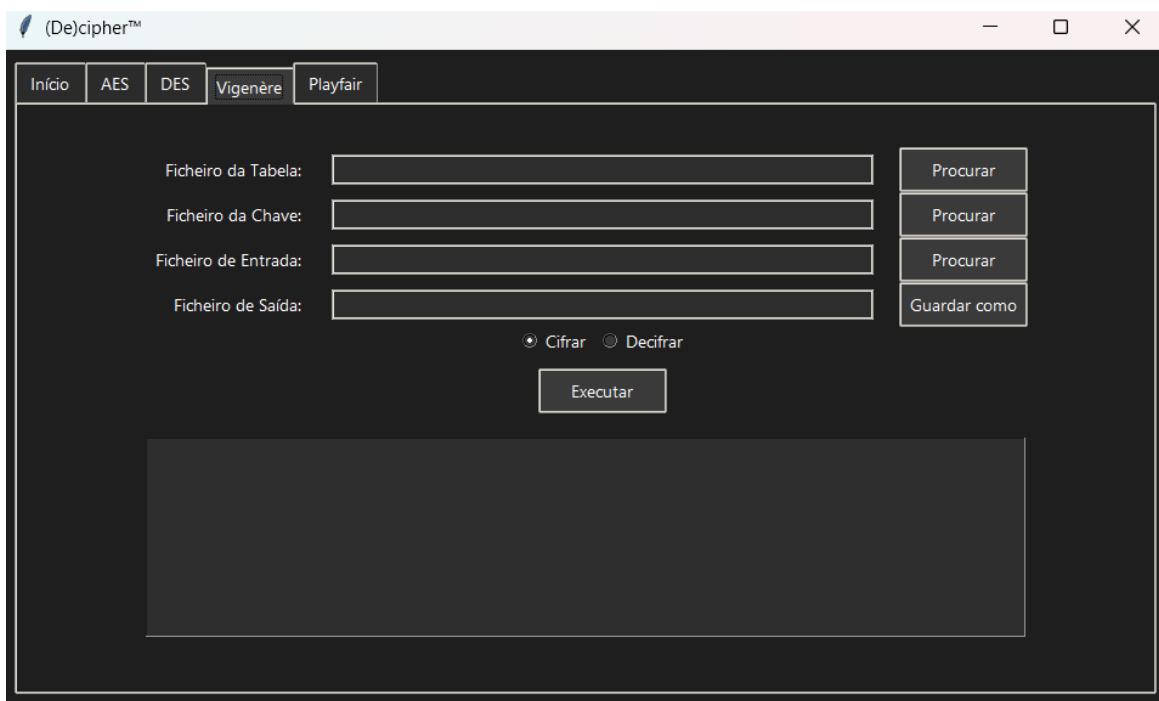


Figura 1: Exemplo da interface gráfica da aplicação.

3.2 AES

Implementou-se uma ferramenta para cifragem e decifragem de ficheiros utilizando AES em modo ECB (biblioteca PyCryptodome). A chave é carregada a partir de ficheiro através da rotina interna `_read_key_file`. Essa rotina aceita uma string hexadecimal (preferencial) ou texto UTF-8, converte para `bytes` e valida o comprimento (16, 24 ou 32 bytes). O formato do ficheiro cifrado utilizado pelo módulo é o ciphertext puro (sem IV). A encriptação aplica preenchimento compatível com PKCS#7 para completar blocos de 16 bytes; a descifragem valida e remove esse preenchimento. Em caso de ficheiro de chave inexistente, tamanho de chave inválido ou padding inválido, são lançadas exceções apropriadas.

```
def encrypt_file(input_path: str, output_path: str, key_path: str):
    """Encrypt a file with AES-ECB and write ciphertext to output.

Args:
    input_path: Path to the plaintext input file (opened in binary).
    output_path: Path to write the encrypted file (binary).
    key_path: Path to the key file (hex or text).

"""
    key = _read_key_file(key_path)
    cipher = AES.new(key, AES.MODE_ECB)

    with open(input_path, "rb") as f:
        data = f.read()

    pad = AES.block_size - len(data) % AES.block_size
    ciphertext = cipher.encrypt(data + bytes([pad])*pad)

    with open(output_path, "wb") as f:
        f.write(ciphertext)
```

(a) Cifragem

```
def decrypt_file(input_path: str, output_path: str, key_path: str):
    """Decrypt a file produced by :func:`encrypt_file` (ECB mode).

Args:
    input_path: Path to the encrypted input file.
    output_path: Path to write the decrypted plaintext.
    key_path: Path to the key file (hex or text).

Raises:
    ValueError: if padding is invalid (likely wrong key).
"""
    key = _read_key_file(key_path)
    cipher = AES.new(key, AES.MODE_ECB)

    with open(input_path, "rb") as f:
        data = cipher.decrypt(f.read())

    pad = data[-1]
    if not 1 <= pad <= AES.block_size:
        raise ValueError("Padding inválido ou chave incorreta.")

    with open(output_path, "wb") as f:
        f.write(data[:-pad])
```

(b) Decifragem

Figura 2: Cifragem e Decifragem com AES

3.3 DES

Implementou-se uma ferramenta para cifragem e decifragem de ficheiros utilizando DES em modo ECB (biblioteca PyCryptodome). A chave é carregada a partir de ficheiro pela rotina interna `_read_key_file`, que aceita uma string hexadecimal (preferencial) ou texto UTF-8, converte para `bytes` e valida o comprimento (exatamente 8 bytes). O formato do ficheiro cifrado é o ciphertext puro (sem IV). A encriptação aplica preenchimento compatível com PKCS#5 (blocos de 8 bytes); a descifragem valida e remove esse preenchimento. Em caso de ficheiro de chave inexistente, tamanho de chave inválido ou padding inválido, são lançadas exceções apropriadas.

```
def encrypt_file(input_path: str, output_path: str, key_path: str):
    """Encrypt a file using DES-CBC and write IV||ciphertext to output.
    Encrypt a file using DES-ECB and write the raw ciphertext to output.

    Args:
        input_path: Path to the plaintext file (binary).
        output_path: Path where the encrypted file will be written.
        key_path: Path to the key file (hex or text).
    """
    key = _read_key_file(key_path)
    cipher = DES.new(key, DES.MODE_ECB)

    with open(input_path, "rb") as f:
        data = f.read()

    pad = DES.block_size - (len(data) % DES.block_size)
    ciphertext = cipher.encrypt(data + bytes([pad]) * pad)

    with open(output_path, "wb") as f:
        f.write(ciphertext)
```

(a) Cifragem

```
def decrypt_file(input_path: str, output_path: str, key_path: str):
    """Decrypt a file produced by :func:`encrypt_file` and write plaintext.

    Args:
        input_path: Path to the encrypted input file.
        output_path: Path to write the recovered plaintext.
        key_path: Path to the key file (hex or text).

    Raises:
        ValueError: if padding is invalid (likely wrong key).
    """
    key = _read_key_file(key_path)
    cipher = DES.new(key, DES.MODE_ECB)

    with open(input_path, "rb") as f:
        data = cipher.decrypt(f.read())

    pad = data[-1]
    if not 1 <= pad <= DES.block_size:
        raise ValueError("Padding inválido ou chave incorreta.")

    with open(output_path, "wb") as f:
        f.write(data[:-pad])
```

(b) Decifragem

Figura 3: Cifragem e Decifragem com DES

3.4 Vigenère

Implementou-se uma ferramenta para cifragem e decifragem de ficheiros usando a cifra de Vigenère com base numa tabela 95x95 carregada através de um ficheiro. A rotina interna `read_table_from_file` lê e valida uma tabela com 95 linhas de 95 caracteres ASCII visíveis (32–126); `read_key_from_file` lê a chave, filtra apenas caracteres ASCII visíveis e devolve uma string (valida que não esteja vazia). O texto é normalizado para caracteres ASCII visíveis antes do processamento; a função `process_text` avança o índice da chave apenas sobre caracteres da tabela e usa `encrypt_char/decrypt_char` para cifrar/decifrar carácter a carácter segundo a linha da tabela definida pelo carácter da chave. As funções `encrypt_file/decrypt_file` recebem `key` como o par `[table_path, key_path]`, lêem os ficheiros de entrada e escrevem ficheiros de saída contendo apenas letras maiúsculas A–Z. Em caso de ficheiro de tabela/chave inexistente, tabela com formato inválido ou chave vazia, são lançadas exceções apropriadas.

```
def process_text(text, key, table, mode="encrypt"):
    """Apply the Vigenère cipher to `text` using `key` and `table`.

    The function:
    - builds charset/mappings from `table` (first row defines charset),
    - filters `text` to only characters present in the table charset,
    - repeats `key` as necessary and applies per-character encrypt or
    | decrypt using `encrypt_char`/`decrypt_char`.

    `mode` may be "encrypt" or "decrypt".
    """
    charset, mapping, reverse = build_mapping(table)
    allowed = set(charset)

    text = ''.join(ch for ch in text if ch in allowed)

    res = []
    ki = 0

    for ch in text:
        kc = key[ki % len(key)]
        if mode == "encrypt":
            res.append(encrypt_char(ch, kc, table, mapping))
        else:
            res.append(decrypt_char(ch, kc, table, mapping, reverse))
        ki += 1

    return ''.join(res)
```

Figura 4: Cifragem e Decifragem com Vigenère

3.5 PlayFair

Implementou-se uma ferramenta para cifragem e decifragem de ficheiros usando a cifra de Playfair com uma tabela 5×5 carregada através de um ficheiro. A rotina interna `read_board_from_file` lê e valida a tabela (linhas de letras; caracteres não-alfabéticos são ignorados), mapeia 'J' para 'I' e, se necessário, preenche os restantes caracteres A..Z (sem J). A normalização do texto é feita por `fix_message`: conserva apenas letras ASCII, converte para maiúsculas, insere 'X' entre letras duplicadas num digrafo e adiciona 'X' se necessário para obter comprimento par. O par (digrafo) é transformado por `process_pair` segundo as regras de Playfair (mesma linha \rightarrow deslocamento horizontal; mesma coluna \rightarrow deslocamento vertical; caso rectangular \rightarrow troca de colunas). As funções `encrypt_text/decrypt_text` aplicam o processamento ao texto; `encrypt_file/decrypt_file` leem/escrevem ficheiros de texto usando o caminho da board como `key`. Em caso de ficheiro de board inexistente são lançadas exceções apropriadas.

```
def process_pair(board, a, b, decrypt=False):
    """Process a digraph (a, b) using Playfair rules and return two chars.

    The boolean parameter `decrypt` selects decryption behavior when
    True; otherwise encryption rules are applied. Both input letters
    are mapped so that 'J' is treated as 'I' during lookup. The function
    expects both letters to exist in `board`; if a letter is missing the
    call will raise an error when attempting to unpack the search result.
    """
    pos = search_letter(board, a)
    if pos is None:
        raise ValueError(f"Letter {a}!r not found in Playfair board")
    r1, c1 = pos

    pos = search_letter(board, b)
    if pos is None:
        raise ValueError(f"Letter {b}!r not found in Playfair board")
    r2, c2 = pos

    if r1 == r2:
        if decrypt:
            return board[r1][(c1 - 1) % 5] + board[r2][(c2 - 1) % 5]
        else:
            return board[r1][(c1 + 1) % 5] + board[r2][(c2 + 1) % 5]

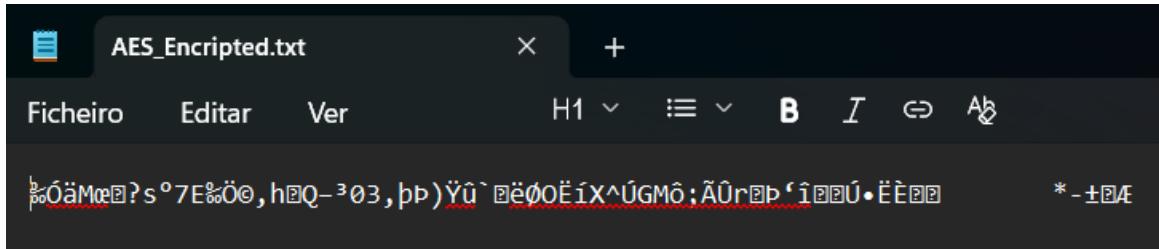
    if c1 == c2:
        if decrypt:
            return board[(r1 - 1) % 5][c1] + board[(r2 - 1) % 5][c2]
        else:
            return board[(r1 + 1) % 5][c1] + board[(r2 + 1) % 5][c2]

    return board[r1][c2] + board[r2][c1]
```

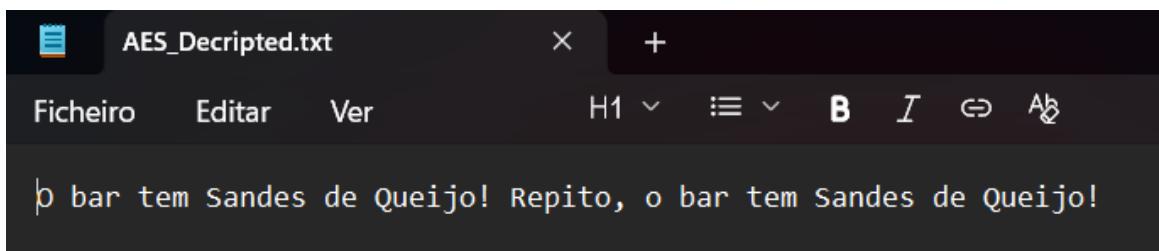
Figura 5: Cifragem e Decifragem com PlayFair

3.6 Testes e Resultados

Os seguintes testes foram realizados para verificar o correto funcionamento das cifras implementadas. Os dados de entrada utilizados foram ficheiros de texto simples contendo frases comuns em português. Todos os ficheiros de chave e tabela foram criados previamente e armazenados em ficheiros separados.

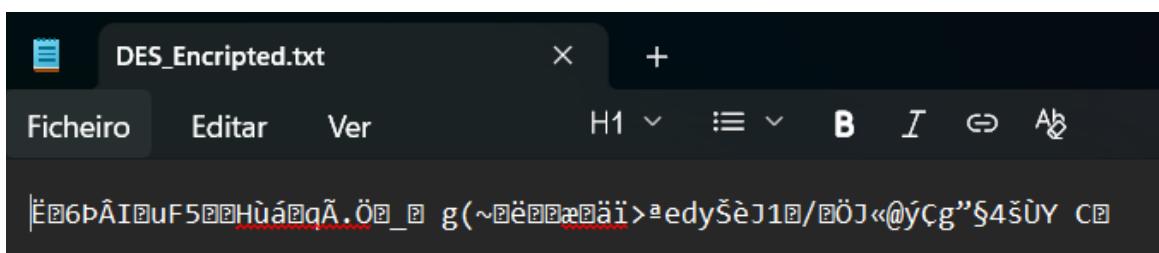


(a) Depois da Cifragem

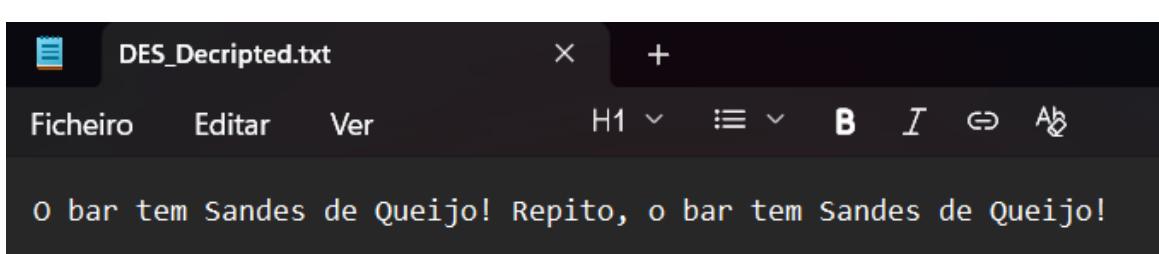


(b) Depois da Decifragem

Figura 6: Testes com AES

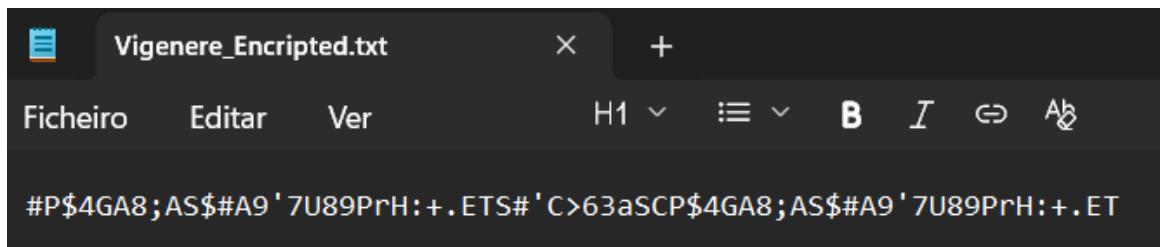


(a) Depois da Cifragem



(b) Densão da Desifragem

Figura 7: Testes com DES

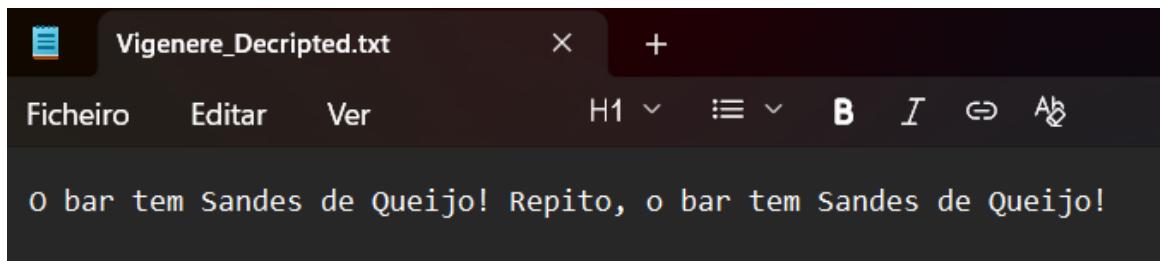


Vigenere_Encrypted.txt

Ficheiro Editar Ver H1 \equiv B I \oplus Ab

```
#P$4GA8;AS##A9'7U89PrH:+.ETS#'C>63aSCP$4GA8;AS##A9'7U89PrH:+.ET
```

(a) Depois da Cifragem



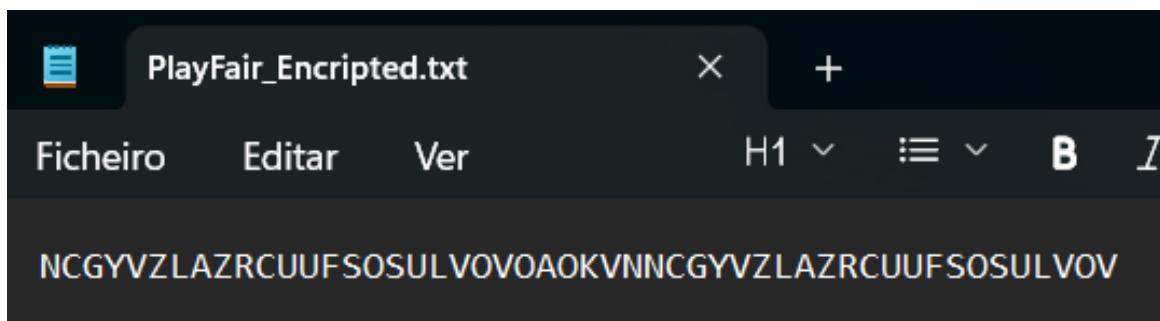
Vigenere_Decripted.txt

Ficheiro Editar Ver H1 \equiv B I \oplus Ab

```
O bar tem Sandes de Queijo! Repito, o bar tem Sandes de Queijo!
```

(b) Depois da Decifragem

Figura 8: Testes com Vigenère

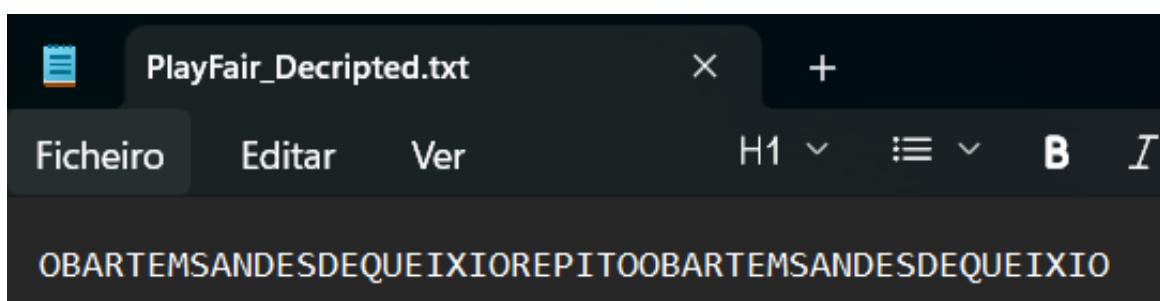


PlayFair_Encrypted.txt

Ficheiro Editar Ver H1 \equiv B I

```
NCGYZLAZRCUUFSOSULVOVOAOKVNNCGYZLAZRCUUFSOSULVOV
```

(a) Depois da Cifragem



PlayFair_Decripted.txt

Ficheiro Editar Ver H1 \equiv B I

```
OBARTEMSANDESDEQUEIXIOREPITOOBARTEMSANDESDEQUEIXIO
```

(b) Depois da Decifragem

Figura 9: Testes com PlayFair

4 Conclusão

O desenvolvimento desta aplicação permitiu explorar conceitos fundamentais sobre cifras simétricas. Apesar de alguns desafios encontrados, foi possível implementar as cifras solicitadas e configurar uma interface centralizada para facilitar a utilização das mesmas. Durante este processo, foram revistos conceitos previamente estudados na licenciatura de Engenharia Informática.

5 Notas finais

Irá ser entregue juntamente com o relatório e o código fonte da aplicação, os ficheiros de chave e tabela utilizados nos testes de modo a confirmar a sua veracidade e permitir a sua replicação em fase de avaliação, de uma forma igual à apresentada no relatório.

Bibliografia

- IPBeja. (2025). *Disciplina: Criptografia e Criptanalise Aplicadas* / IPBeja [Página LPD]. Obtido outubro 30, 2025, de <https://cms.ipbeja.pt/course/view.php?id=544>
- Martinho Caeiro & Paulo Abade. (2025). *(De)cipher™ - Repositório de Código* [Repositório da Aplicação (De)cipher™]. Obtido outubro 30, 2025, de <https://github.com/MartinhoCaeiro/De-cipher>
- Microsoft Corporation. (2025). *Visual Studio Code* [Editor de Texto Visual Studio Code]. Obtido outubro 30, 2025, de <https://code.visualstudio.com/>
- Python Software Foundation. (2025). *Welcome to Python.org* [Linguagem de Programação Python]. Obtido outubro 30, 2025, de <https://www.python.org/>
- Wikipedia. (2025a). *Advanced Encryption Standard*. Obtido outubro 30, 2025, de https://en.wikipedia.org/wiki/Advanced_Encryption_Standard
- Wikipedia. (2025b). *Cifra de Playfair*. Obtido outubro 30, 2025, de https://en.wikipedia.org/wiki/Playfair_cipher
- Wikipedia. (2025c). *Cifra de Vigenère*. Obtido outubro 30, 2025, de https://en.wikipedia.org/wiki/Vigen%C3%A8re_cipher
- Wikipedia. (2025d). *Data Encryption Standard*. Obtido outubro 30, 2025, de https://en.wikipedia.org/wiki/Data_Encryption_Standard