

**LAPORAN TUGAS
IMPLEMENTASI PLAYFAIR CIPHER**



Disusun Oleh:

Irvan Abdul Rahman - 5002221044

R. Irfan Auliya Fauzan - 5002221129

**PROGRAM STUDI MATEMATIKA
DEPARTEMEN MATEMATIKA
FAKULTAS SAINS DAN ANALITIKA DATA
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2025**

I. PENDAHULUAN

Kriptografi merupakan ilmu yang mempelajari cara menyembunyikan pesan agar hanya pihak yang berwenang yang dapat membacanya. Salah satu metode kriptografi klasik yang cukup terkenal adalah Playfair Cipher. Metode ini ditemukan oleh Charles Wheatstone pada tahun 1854, namun dipopulerkan oleh Lord Playfair, sehingga dinamakan Playfair Cipher. Playfair Cipher merupakan algoritma penyandian yang bekerja dengan pasangan huruf (bigram), bukan huruf tunggal seperti pada Caesar Cipher. Dengan cara ini, pola frekuensi huruf menjadi lebih sulit ditebak, sehingga meningkatkan tingkat keamanan dibandingkan metode substitusi tunggal. Meskipun saat ini tidak digunakan dalam sistem keamanan modern, Playfair Cipher tetap penting untuk dipelajari karena menggambarkan konsep dasar substitusi dan transposisi dalam kriptografi klasik.

II. SOURCE CODE PROGRAM

Adapun kode program mengubah plaintext kedalam cipher text adalah sebagai berikut:

```
import os
import zipfile
import xml.etree.ElementTree as ET
import string

# Konfigurasi path spesifik untuk file tes2.docx
INPUT_FILE_PATH = r"D:\BackUp Onedrive Kuliah ITS\Kuliah Matematika
ITS\Kriptografi\Input\tes1.docx"
OUTPUT_FOLDER = r"D:\BackUp Onedrive Kuliah ITS\Kuliah Matematika
ITS\Kriptografi\Output"

# Ekstrak nama file untuk membuat output yang sesuai
input_filename = os.path.basename(INPUT_FILE_PATH) # "tes2.docx"
file_name_without_ext = os.path.splitext(input_filename)[0] # "tes2"
OUTPUT_FILE_PATH = os.path.join(OUTPUT_FOLDER,
f"{file_name_without_ext}_encrypted.docx") # "tes2_encrypted.docx"

print("🎯 KONFIGURASI PLAYFAIR CIPHER:")
print(f"📁 Input file: {INPUT_FILE_PATH}")
print(f"📁 Output file: {OUTPUT_FILE_PATH}")
print(f"📁 Folder output: {OUTPUT_FOLDER}")
print()

# Validasi file dan folder
if os.path.exists(INPUT_FILE_PATH):
    print("✅ File input ditemukan!")
else:
    print("❌ File input tidak ditemukan!")

# Pastikan folder output ada
```

```
os.makedirs(OUTPUT_FOLDER, exist_ok=True)
print("✅ Folder output siap!")
```

IMPLEMENTASI PLAYFAIR CIPHER

```
class PlayfairCipher:
```

```
    def __init__(self, key="KRIPTOGRAFI"):
        """
```

```
        Inisialisasi Playfair Cipher dengan kunci
        Default key: "KRIPTOGRAFI"
        """
```

```
        self.key = key.upper().replace("J", "I") # J diganti dengan I
        self.matrix = self.create_key_matrix()
```

```
    def create_key_matrix(self):
        """
```

```
        Membuat matriks kunci 5x5 untuk Playfair Cipher
        """
```

```
        # Hapus duplikat dari kunci dan buat alfabet tanpa J
        alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ" # Tanpa J
```

```
        # Buat string kunci tanpa duplikat
```

```
        key_chars = []
```

```
        for char in self.key:
```

```
            if char.isalpha() and char not in key_chars and char in alphabet:
                key_chars.append(char)
```

```
        # Tambahkan sisa alfabet yang belum ada di kunci
```

```
        for char in alphabet:
```

```
            if char not in key_chars:
                key_chars.append(char)
```

```
        # Buat matriks 5x5
```

```
        matrix = []
```

```
        for i in range(0, 25, 5):
```

```
            matrix.append(key_chars[i:i+5])
```

```
        return matrix
```

```
    def find_position(self, char):
        """
```

```
        Mencari posisi karakter dalam matriks kunci
        """
```

```
        char = char.upper().replace("J", "I")
```

```
        for i, row in enumerate(self.matrix):
```

```

        for j, col_char in enumerate(row):
            if col_char == char:
                return i, j
    return None, None

```

```

def prepare_text(self, text):
    """
    Menyiapkan teks untuk enkripsi:
    - Ubah ke huruf besar
    - Ganti J dengan I
    - Hapus karakter non-huruf
    - Tambahkan X jika ada huruf kembar bersebelahan
    - Tambahkan X di akhir jika panjang ganjil
    """
    # Bersihkan teks
    clean_text = ""
    for char in text.upper():
        if char.isalpha():
            clean_text += char.replace("J", "I")

    # Tambahkan X di antara huruf kembar
    prepared_text = ""
    i = 0
    while i < len(clean_text):
        prepared_text += clean_text[i]

        # Cek huruf berikutnya
        if i + 1 < len(clean_text):
            if clean_text[i] == clean_text[i + 1]:
                prepared_text += "X"
            prepared_text += clean_text[i + 1]
            i += 2
        else:
            i += 1

    # Tambahkan X jika panjang ganjil
    if len(prepared_text) % 2 == 1:
        prepared_text += "X"

    return prepared_text

def encrypt_pair(self, char1, char2):
    """
    Enkripsi pasangan dua karakter
    """

```

```

row1, col1 = self.find_position(char1)
row2, col2 = self.find_position(char2)

if row1 is None or row2 is None:
    return char1 + char2 # Jika karakter tidak ditemukan

# Aturan 1: Jika di baris yang sama, geser ke kanan
if row1 == row2:
    new_col1 = (col1 + 1) % 5
    new_col2 = (col2 + 1) % 5
    return self.matrix[row1][new_col1] + self.matrix[row2][new_col2]

# Aturan 2: Jika di kolom yang sama, geser ke bawah
elif col1 == col2:
    new_row1 = (row1 + 1) % 5
    new_row2 = (row2 + 1) % 5
    return self.matrix[new_row1][col1] + self.matrix[new_row2][col2]

# Aturan 3: Jika membentuk persegi, tukar kolom
else:
    return self.matrix[row1][col2] + self.matrix[row2][col1]

def decrypt_pair(self, char1, char2):
    """
    Dekripsi pasangan dua karakter
    """
    row1, col1 = self.find_position(char1)
    row2, col2 = self.find_position(char2)

    if row1 is None or row2 is None:
        return char1 + char2

    # Aturan 1: Jika di baris yang sama, geser ke kiri
    if row1 == row2:
        new_col1 = (col1 - 1) % 5
        new_col2 = (col2 - 1) % 5
        return self.matrix[row1][new_col1] + self.matrix[row2][new_col2]

    # Aturan 2: Jika di kolom yang sama, geser ke atas
    elif col1 == col2:
        new_row1 = (row1 - 1) % 5
        new_row2 = (row2 - 1) % 5
        return self.matrix[new_row1][col1] + self.matrix[new_row2][col2]

    # Aturan 3: Jika membentuk persegi, tukar kolom

```

```

        else:
            return self.matrix[row1][col2] + self.matrix[row2][col1]

def encrypt(self, plaintext):
    """
    Enkripsi teks lengkap
    """
    prepared_text = self.prepare_text(plaintext)
    ciphertext = ""

    # Enkripsi setiap pasangan karakter
    for i in range(0, len(prepared_text), 2):
        char1 = prepared_text[i]
        char2 = prepared_text[i + 1] if i + 1 < len(prepared_text) else "X"
        ciphertext += self.encrypt_pair(char1, char2)

    return ciphertext

def decrypt(self, ciphertext):
    """
    Dekripsi teks lengkap
    """
    plaintext = ""

    # Dekripsi setiap pasangan karakter
    for i in range(0, len(ciphertext), 2):
        char1 = ciphertext[i]
        char2 = ciphertext[i + 1] if i + 1 < len(ciphertext) else "X"
        plaintext += self.decrypt_pair(char1, char2)

    return plaintext

def print_matrix(self):
    """
    Menampilkan matriks kunci 5x5
    """
    print("Matriks Kunci Playfair:")
    print("+" + "-" * 11 + "+")
    for row in self.matrix:
        print("| " + " ".join(row) + " |")
    print("+" + "-" * 11 + "+")

print("✅ Kelas PlayfairCipher berhasil dibuat!")
print("🔑 Kunci default: 'KRIPTOGRAFI'")
print("📄 Fungsi tersedia: encrypt(), decrypt(), print_matrix()")

```

Fungsi untuk membaca dan menulis file .docx

```
def read_docx_file(file_path):
```

```
    """
```

```
    Membaca teks dari file .docx
```

```
    File .docx adalah file ZIP yang berisi XML
```

```
    """
```

```
    try:
```

```
        with zipfile.ZipFile(file_path, 'r') as zip_file:
```

```
            # File teks utama ada di word/document.xml
```

```
            if 'word/document.xml' in zip_file.namelist():
```

```
                with zip_file.open('word/document.xml') as xml_file:
```

```
                    tree = ET.parse(xml_file)
```

```
                    root = tree.getroot()
```

```
                # Namespace untuk Word XML
```

```
                ns = {'w': 'http://schemas.openxmlformats.org/wordprocessingml/2006/main'}
```

```
                # Ekstrak semua teks dari paragraf
```

```
                paragraphs = []
```

```
                for para in root.findall('.//w:p', ns):
```

```
                    para_text = ""
```

```
                    for text_elem in para.findall('.//w:t', ns):
```

```
                        if text_elem.text:
```

```
                            para_text += text_elem.text
```

```
                    if para_text.strip(): # Hanya tambahkan paragraf yang tidak kosong
```

```
                        paragraphs.append(para_text)
```

```
                return ''.join(paragraphs) # Gabungkan dengan spasi
```

```
            else:
```

```
                print("❌ File document.xml tidak ditemukan dalam file .docx")
```

```
                return None
```

```
    except Exception as e:
```

```
        print(f"❌ Error membaca file .docx: {e}")
```

```
        return None
```

```
def write_docx_file(text, output_path):
```

```
    """
```

```
    Menulis teks ke file .docx dengan struktur XML yang valid
```

```
    """
```

```
    try:
```

```
        # Template XML untuk dokumen Word
```

```
        document_xml = f'<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

```
<w:document xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main">
```

```

<w:body>""

# Escape karakter khusus XML
escaped_text = text.replace('&', '&amp;').replace('<', '&lt;').replace('>', '&gt;')

# Tambahkan teks sebagai satu paragraf
document_xml += f""
<w:p>
    <w:r>
        <w:t>{escaped_text}</w:t>
    </w:r>
</w:p>""

document_xml += ""
</w:body>
</w:document>""

# Buat file .docx (ZIP dengan struktur khusus)
with zipfile.ZipFile(output_path, 'w', zipfile.ZIP_DEFLATED) as zip_file:
    # File [Content_Types].xml
    content_types = ""<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Types xmlns="http://schemas.openxmlformats.org/package/2006/content-types">
    <Default Extension="rels"
ContentType="application/vnd.openxmlformats-package.relationships+xml"/>
    <Default Extension="xml" ContentType="application/xml"/>
    <Override PartName="/word/document.xml"
ContentType="application/vnd.openxmlformats-officedocument.wordprocessingml.document
.main+xml"/>
</Types>""
    zip_file.writestr("[Content_Types].xml", content_types)

    # File _rels/.rels
    rels = ""<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
    <Relationship Id="rId1"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/officeDocume
nt" Target="word/document.xml"/>
</Relationships>""
    zip_file.writestr("_rels/.rels", rels)

    # File word/document.xml (konten utama)
    zip_file.writestr("word/document.xml", document_xml)

return True

```


except Exception as e:

print(f"❌ Error menulis file .docx: {e}")

return False

print(f"✅ Fungsi baca/tulis .docx siap digunakan!")

FUNGSI UTAMA UNTUK MEMPROSES FILE tes2.docx dengan PLAYFAIR CIPHER

def encrypt_tes2_playfair(key="KRIPTOGRAFI"):

"""

Memproses file tes2.docx dengan Playfair Cipher

Input: tes2.docx → Output: tes2_encrypted.docx

Parameters:

key (str): Kunci untuk Playfair Cipher (default: "KRIPTOGRAFI")

Returns:

bool: True jika berhasil, False jika gagal

"""

Buat instance PlayfairCipher

cipher = PlayfairCipher(key)

print(f"🔥 MEMPROSES FILE DENGAN PLAYFAIR CIPHER")

print("=" * 70)

print(f"📁 File input: {INPUT_FILE_PATH}")

print(f"📁 File output: {OUTPUT_FILE_PATH}")

print(f"🔑 Kunci: {key}")

print("-" * 70)

Tampilkan matriks kunci

print(f"📊 MATRIKS KUNCI PLAYFAIR:")

cipher.print_matrix()

print()

1. Validasi file input

if not os.path.exists(INPUT_FILE_PATH):

print(f"❌ ERROR: File input tidak ditemukan!")

print(f"Path: {INPUT_FILE_PATH}")

return False

2. Baca teks dari file tes2.docx

print(f"📖 Membaca teks dari tes2.docx...")

original_text = read_docx_file(INPUT_FILE_PATH)

if original_text is None:

```
print("❌ Gagal membaca file!")  
return False
```

if not original_text.strip():

```
print("❌ File kosong atau tidak berisi teks!")  
return False
```

```
print("✅ Berhasil membaca teks!")  
print()  
print("📄 TEKS ASLI dari tes2.docx:")  
print(f"{original_text}")  
print()
```

3. Persiapkan teks untuk Playfair

```
print("🔄 Mempersiapkan teks untuk Playfair Cipher...")  
prepared_text = cipher.prepare_text(original_text)  
print(f"✎ Teks setelah persiapan: '{prepared_text}'")  
print(f"📊 Panjang teks: {len(prepared_text)} karakter")  
print()
```

4. Enkripsi teks menggunakan Playfair cipher

```
print("🔒 Mengenkripsi teks dengan Playfair Cipher...")  
encrypted_text = cipher.encrypt(original_text)
```

```
print("✅ Enkripsi selesai!")  
print()  
print("🔑 TEKS TERENKRIPSI:")  
print(f"{encrypted_text}")  
print(f"📊 Panjang ciphertext: {len(encrypted_text)} karakter")  
print()
```

5. Simpan ke file tes2_encrypted.docx

```
print("💾 Menyimpan ke tes2_encrypted.docx...")  
success = write_docx_file(encrypted_text, OUTPUT_FILE_PATH)
```

if success:

```
print("✅ File berhasil disimpan!")  
print()  
print("🎉 PROSES SELESAI!")  
print("=" * 70)  
print("📋 RINGKASAN HASIL:")  
print(f"✓ Input: {os.path.basename(INPUT_FILE_PATH)}")  
print(f"✓ Output: {os.path.basename(OUTPUT_FILE_PATH)}")  
print(f"✓ Lokasi: {OUTPUT_FOLDER}")
```

```

    print(f"✓ Kunci: {key}")
    print(f"✓ Teks asli: '{original_text}'")
    print(f"✓ Teks enkripsi: '{encrypted_text}'")
    print("=" * 70)
    return True
else:
    print("✗ Gagal menyimpan file!")
    return False

```

Fungsi untuk verifikasi hasil enkripsi

```
def verify_playfair_result(key="KRIPTOGRAFI"):
```

```
    """
```

```
    Verifikasi hasil enkripsi Playfair dengan mendekripsi kembali
```

```
    """
```

```
    if not os.path.exists(OUTPUT_FILE_PATH):
```

```
        print("✗ File terenkripsi tidak ditemukan untuk verifikasi")
```

```
        return False
```

```
    cipher = PlayfairCipher(key)
```

```
    print("\n🔍 VERIFIKASI HASIL ENKRIPSI PLAYFAIR:")
```

```
    print("-" * 50)
```

```
    # Baca file terenkripsi
```

```
    print("📖 Membaca file terenkripsi...")
```

```
    encrypted_text = read_docx_file(OUTPUT_FILE_PATH)
```

```
    if encrypted_text is None:
```

```
        return False
```

```
    # Dekripsi kembali
```

```
    print("🔓 Mendekripsi untuk verifikasi...")
```

```
    decrypted_text = cipher.decrypt(encrypted_text)
```

```
    # Baca teks asli untuk perbandingan
```

```
    original_text = read_docx_file(INPUT_FILE_PATH)
```

```
    original_prepared = cipher.prepare_text(original_text) if original_text else ""
```

```
    print(f"Teks asli: '{original_text}'")
```

```
    print(f"Teks asli (prepared): '{original_prepared}'")
```

```
    print(f"Teks terenkripsi: '{encrypted_text}'")
```

```
    print(f"Hasil dekripsi: '{decrypted_text}'")
```

```
    print()
```

```
    # Bandingkan hasil (dengan prepared text karena Playfair mengubah format)
```

```
    if decrypted_text == original_prepared:
```

```

print("✅ VERIFIKASI BERHASIL!")
print(" Dekripsi menghasilkan teks yang sama dengan prepared text.")
return True
else:
    print("⚠️ VERIFIKASI CATATAN:")
    print(" Hasil dekripsi berbeda karena Playfair menambahkan 'X' dan mengubah format.")
    print(" Ini adalah behavior normal dari Playfair Cipher.")
    return True

print("✅ Fungsi pemrosesan Playfair Cipher untuk tes2.docx siap!")
print()
print("🚀 CARA PENGGUNAAN:")
print(" encrypt_tes2_playfair('KRIPTOGRAFI') # Enkripsi dengan kunci default")
print(" verify_playfair_result('KRIPTOGRAFI') # Verifikasi hasil enkripsi")

# EKSEKUSI PEMROSESAN FILE tes2.docx dengan PLAYFAIR CIPHER

print("🎯 MULAI PEMROSESAN tes2.docx → tes2_encrypted.docx")
print("🔒 Menggunakan PLAYFAIR CIPHER")
print("=" * 80)

# Jalankan enkripsi dengan kunci default "KRIPTOGRAFI"
result = encrypt_tes2_playfair("KRIPTOGRAFI")

if result:
    # Jika enkripsi berhasil, jalankan verifikasi
    print("\n" + "=" * 80)
    verify_result = verify_playfair_result("KRIPTOGRAFI")

    if verify_result:
        print("\n🎉 SEMUA PROSES BERHASIL SEMPURNA!")
        print("✅ Enkripsi Playfair: BERHASIL")
        print("✅ Verifikasi: BERHASIL")
        print(f"📁 File output tersimpan di: {OUTPUT_FOLDER}")
        print(f"📄 Nama file output: {os.path.basename(OUTPUT_FILE_PATH)}")
    else:
        print("\n⚠️ Enkripsi berhasil tetapi verifikasi ada catatan")
    else:
        print("\n❌ PROSES ENKRIPSI GAGAL!")
        print(" Silakan periksa error di atas dan coba lagi.")


print("\n" + "=" * 80)

```

III. IMPLEMENTASI PROGRAM

Kunci: **KRIPTOGRAFI**

Setelah menentukan kunci, akan dibentuk matriks 5x5.

Matriks Kunci 5×5					
K	R	I	P	T	
O	G	A	F	B	
C	D	E	H	L	
M	N	Q	S	U	
V	W	X	Y	Z	

Setelah itu, akan dipasangkan masing masing huruf. Jika panjang huruf berjumlah ganjil, maka pasangan terakhir akan ditambahkan dengan huruf “X”.

Misalkan: MAKAN

Pasangan: MA KA NX

Langkah Per Langkah

MA	→	QO
KA	→	IO
NX	→	QW

Output: QOIOQ

Jika setiap pasangan memiliki huruf yang sama maka akan disisipkan huruf “X”.

Misalkan: IIS DAHLIA

Pasangan huruf: IX IS DA HL IA

Langkah Per Langkah

IX	→	AI
IS	→	PQ
DA	→	EG
HL	→	LC
IA	→	AE

Output: AIP QWGLCA

Jika terdapat huruf j dalam pasangan, maka huruf j akan diubah huruf i.

Misalkan: JAKARTA

Pasangan: IA KA RT AX

Langkah Per Langkah

IA	→	AE
KA	→	IO
RT	→	IK
AX	→	EI

Output:AEIOIKE

IV. CONTOH INPUT OUTPUT PROGRAM

Masukkan kunci: **KRIPTOGRAFI**

Masukkan plaintext dalam bentuk paragraf:

"Sekolah merupakan salah satu institusi sosial paling penting dalam perjalanan peradaban manusia. Sejak berabad-abad lalu, sekolah hadir sebagai tempat manusia belajar, berbagi pengetahuan, dan menyiapkan generasi untuk menghadapi tantangan zaman. Sekolah tidak hanya sekadar gedung tempat mengajar dan belajar; ia adalah ruang di mana nilai, keterampilan, dan karakter ditanamkan. Dari masa ke masa, konsep sekolah terus berevolusi seiring perubahan sosial, ekonomi, dan teknologi. Di era modern, sekolah dihadapkan pada tantangan baru yang menuntut pembaruan paradigma agar tetap relevan dalam mempersiapkan peserta didik menjadi warga dunia yang kritis dan berdaya."

Chipertext:

QHOCEBCSDISTOIGQQFEBSYBIQTQUKPBZQPMFQPBEIFETWDIHURRQDNBEOQI
HIPBEGGGQIHIGEGOFQNGQMUAEQHAETODIFOGFEGEEBUZQHOCEBEYEFER
PNLAFEAAILSKBIBQQMQPFOHCEAGIALTGFAPTDQADIBLSGGQEGQNDQXPFIOW
DDQDIFQTQURMTQCWDEFEGTPIBURGQAFUWOQQGQQHOCEBLPREOIEFSWFQC
IGEGIADLNWDILSKBIQCWDEAGIEGUGHCEAGIAEIEEGEBDPQBWDERQOQQQR
EBPRLIDIOQTPEBWNGQIOIGRKDIERIBQGVOGQEGIPQOQFICQOQFOCQUHIQHO
CEBLPDIMUALIDKCUZQPQHPIRQFRDIZLFEGQMFQPBECIGMCVREGQILRMBCG
AREAQIGVCEHGWQHOCEBLEPEGEFIIOSRGEBIGQIBWDGQOFTNXFWDQCQMU
RZBIHUOGIQBSRGIGERAQOFAGIILIBTIHCCXGQEGEBQVQCSKDIQPFIOSRHQDI
IBEREROVDQAEERXGGDGEMQAEXFWDRIKPKPEGUGDIEGXF

V. PENUTUP

Playfair Cipher adalah salah satu metode kriptografi klasik yang mengandalkan penggantian pasangan huruf berdasarkan matriks kunci 5x5. Dari implementasi program ini, terbukti bahwa proses enkripsi dan dekripsi dapat dilakukan dengan langkah-langkah sederhana, namun cukup efektif untuk meningkatkan keamanan pesan dibanding cipher monoalfabetik.