TUGAS 1

KEAMANAN KOMPUTER/KRIPTOGRAFI

DISUSUN OLEH :

NAMA : SITTI ROHANI

NIM : 222061

KELAS : 5TKKO-G

PRODI : TEKNIK INFORMATIKA

UNIVERSITAS DIPA MAKASSAR

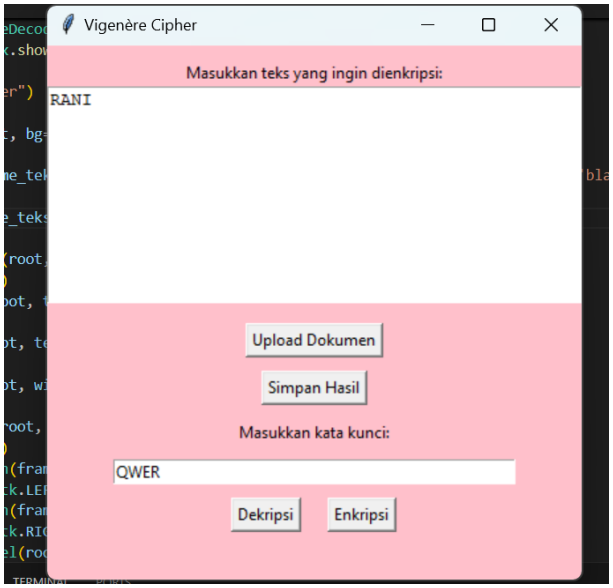
2024/2025

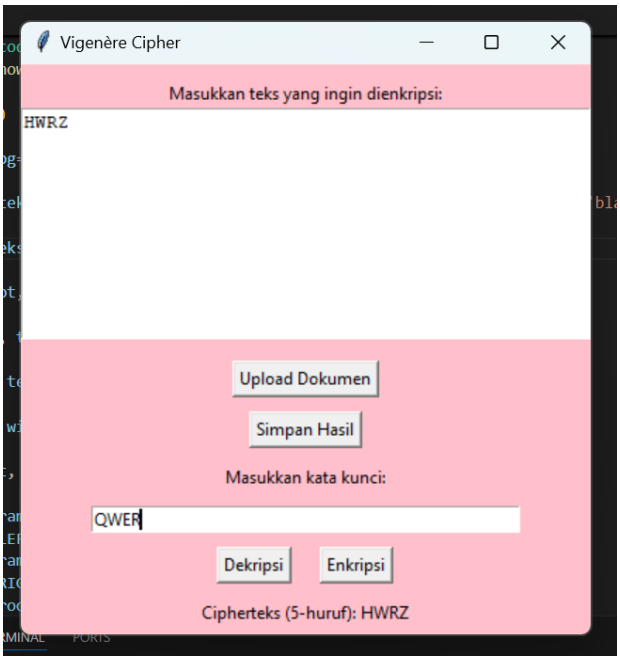
LAPORAN

| No. | Spesifikasi | Berhasil (√) | Kurang Berhasil (√) | Keterangan |
|---|---|---|---|---|
| 1 | Vigenere Cipher | ✅ | | |
| 2 | Extended Vigenere Cipher | ✅ | | |
| 3 | Playfair Cipher | ✅ | | |
| 4 | Enigma Cipher | ✅ | | |
| 5 | One-Time pad | ✅ | | |

1. VIGENERE CIPHER

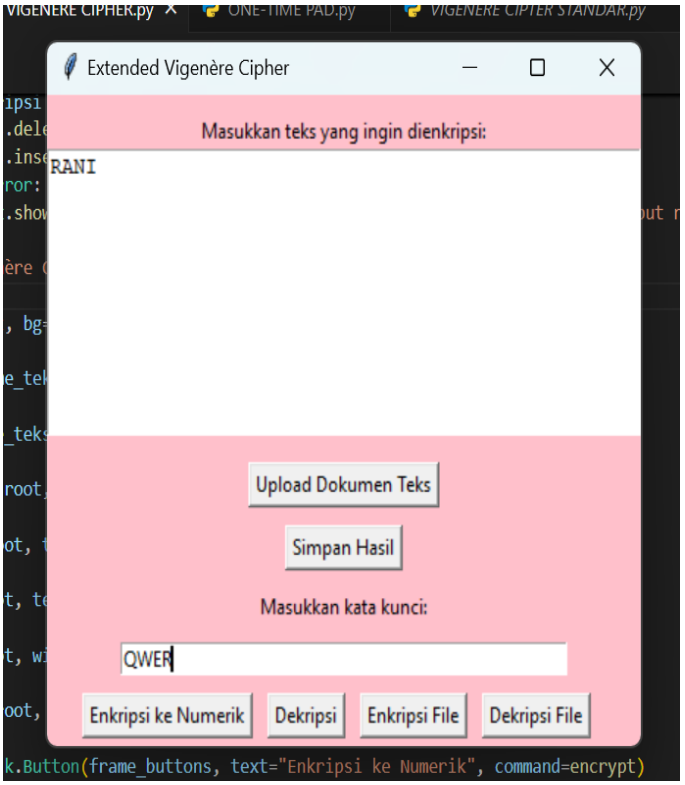
```python
import tkinter as tk
from tkinter import messagebox, filedialog
import os
def vigenere_encrypt(teks_asli, kata_kunci):
    teks_enkripsi = ""
    kata_kunci_diulang = (kata_kunci * (len(teks_asli) // len(kata_kunci) + 1))[:len(teks_asli)]
    for t_char, k_char in zip(teks_asli, kata_kunci_diulang):
        if t_char.isalpha():
            shift = (ord(t_char.upper()) - ord('A') + ord(k_char.upper()) - ord('A')) % 26
            huruf_enkripsi = chr(shift + ord('A'))
            teks_enkripsi += huruf_enkripsi
        else:
            teks_enkripsi += t_char
    return teks_enkripsi
def vigenere_decrypt(teks_enkripsi, kata_kunci):
    teks_dekripsi = ""
    kata_kunci_diulang = (kata_kunci * (len(teks_enkripsi) // len(kata_kunci) + 1))[:len(teks_enkripsi)]
    for e_char, k_char in zip(teks_enkripsi, kata_kunci_diulang):
        if e_char.isalpha():
            shift = (ord(e_char.upper()) - ord('A') - (ord(k_char.upper()) - ord('A'))) % 26
            huruf_dekripsi = chr(shift + ord('A'))
            teks_dekripsi += huruf_dekripsi
        else:
            teks_dekripsi += e_char
    return teks_dekripsi
def encrypt():
    teks_asli = entry_teks.get("1.0", tk.END).strip()
    kata_kunci = entry_kunci.get()
    if not teks_asli or not kata_kunci:
        messagebox.showwarning("Input Error", "Mohon masukkan teks dan kata kunci.")
        return
    hasil_enkripsi = vigenere_encrypt(teks_asli, kata_kunci)
    entry_teks.delete("1.0", tk.END)
    entry_teks.insert(tk.END, hasil_enkripsi)
    cipherteks_kelompok = ' '.join([hasil_enkripsi[i:i+5] for i in range(0, len(hasil_enkripsi), 5)])
    label_cipherteks.config(text=f"Cipherteks (5-huruf): {cipherteks_kelompok}")
def decrypt():
    teks_enkripsi = entry_teks.get("1.0", tk.END).strip()
    kata_kunci = entry_kunci.get()
    if not teks_enkripsi or not kata_kunci:
        messagebox.showwarning("Input Error", "Mohon masukkan teks enkripsi dan kata kunci.")
        return
    hasil_dekripsi = vigenere_decrypt(teks_enkripsi, kata_kunci)
    entry_teks.delete("1.0", tk.END)
    entry_teks.insert(tk.END, hasil_dekripsi)
def save_file():
    file_path = filedialog.asksaveasfilename(defaultextension=".txt", filetypes=[("Text files", "*.txt"), ("All files", "*.*")])
    if file_path:
        with open(file_path, 'w', encoding='utf-8') as file:
            teks = entry_teks.get("1.0", tk.END)
            file.write(teks)
def upload_file():
    file_path = filedialog.askopenfilename(filetypes=[("All files", "*.*")])
    if file_path:
        with open(file_path, 'rb') as file:
            content = file.read()
            try:
                teks = content.decode('utf-8')
                entry_teks.delete("1.0", tk.END)
                entry_teks.insert(tk.END, teks)
            except UnicodeDecodeError:
                messagebox.showwarning("File Error", "File tidak dapat dibaca sebagai teks.")
root = tk.Tk()
root.title("Vigenère Cipher")
root.configure(bg='pink')
frame_teks = tk.Frame(root, bg='pink')
frame_teks.pack(pady=10)
label_teks = tk.Label(frame_teks, text="Masukkan teks yang ingin dienkripsi:", bg='pink', fg='black')
label_teks.pack()
entry_teks = tk.Text(frame_teks, width=50, height=10)
entry_teks.pack()
button_upload = tk.Button(root, text="Upload Dokumen", command=upload_file)
button_upload.pack(pady=5)
button_save = tk.Button(root, text="Simpan Hasil", command=save_file)
button_save.pack(pady=5)
label_kunci = tk.Label(root, text="Masukkan kata kunci:", bg='pink', fg='black')
label_kunci.pack(pady=5)
entry_kunci = tk.Entry(root, width=50)
entry_kunci.pack(pady=5)
frame_buttons = tk.Frame(root, bg='pink')
frame_buttons.pack(pady=5)
button_decrypt = tk.Button(frame_buttons, text="Dekripsi", command=decrypt)
button_decrypt.pack(side=tk.LEFT, padx=(0, 10))
button_encrypt = tk.Button(frame_buttons, text="Enkripsi", command=encrypt)
button_encrypt.pack(side=tk.RIGHT, padx=(10, 0))
label_cipherteks = tk.Label(root, text="", bg='pink', fg='black')
label_cipherteks.pack(pady=5)
root.mainloop()

```

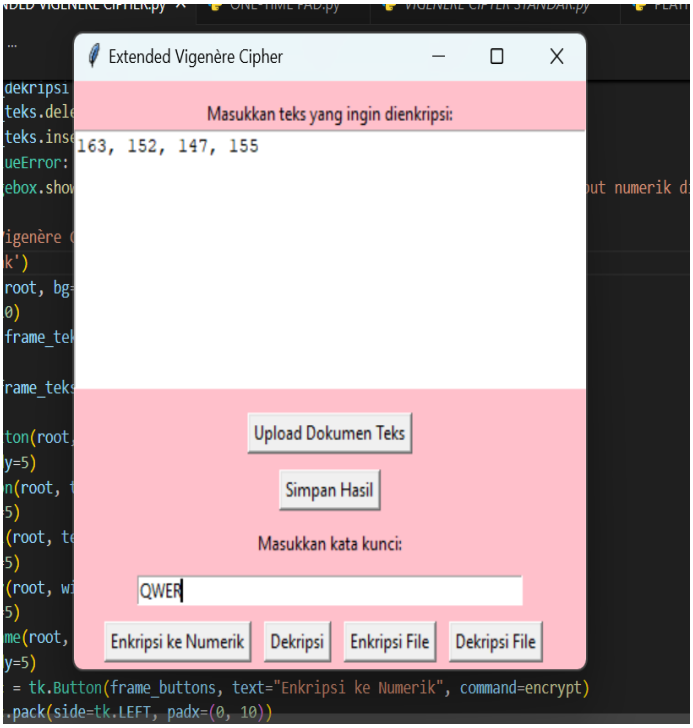


2. EXTENDED VIGENERE CIPHER

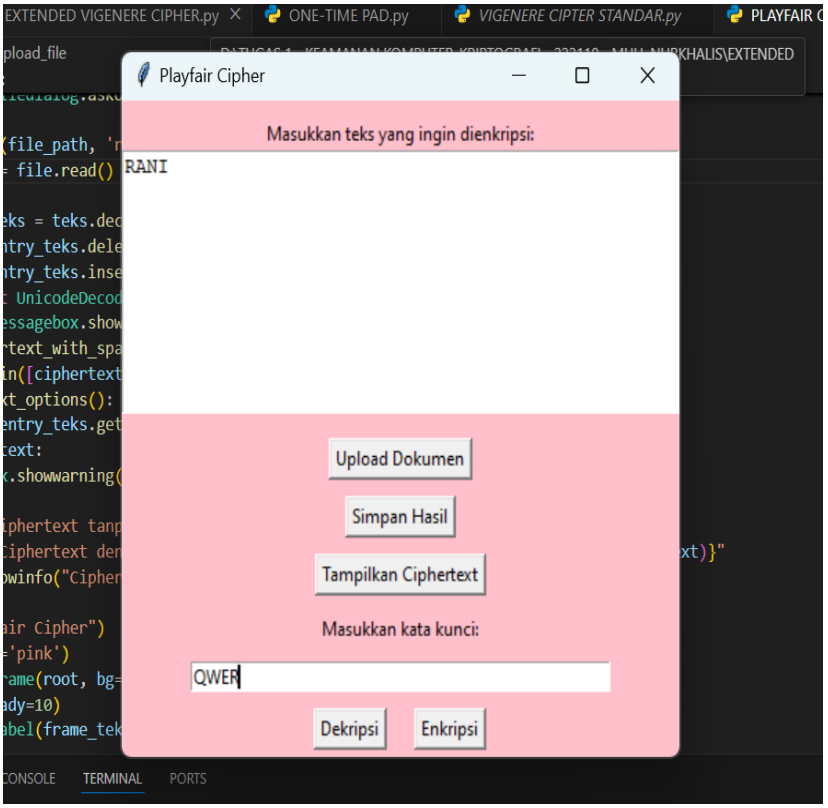
```python
import tkinter as tk
from tkinter import messagebox, filedialog
def extended_vigenere_encrypt(teks_asli, kata_kunci):
    teks_enkripsi = []
    kata_kunci_diulang = (kata_kunci * (len(teks_asli) // len(kata_kunci) + 1))[:len(teks_asli)]
    for t_char, k_char in zip(teks_asli, kata_kunci_diulang):
        shift = (ord(t_char) + ord(k_char)) % 256
        teks_enkripsi.append(shift)
    return teks_enkripsi
def extended_vigenere_decrypt(teks_enkripsi, kata_kunci):
    teks_dekripsi = ""
    kata_kunci_diulang = (kata_kunci * (len(teks_enkripsi) // len(kata_kunci) + 1))[:len(teks_enkripsi)]
    for e_char, k_char in zip(teks_enkripsi, kata_kunci_diulang):
        shift = (e_char - ord(k_char)) % 256
        teks_dekripsi += chr(shift)
    return teks_dekripsi
def encrypt():
    teks_asli = entry_teks.get("1.0", tk.END).strip()
    kata_kunci = entry_kunci.get()
    if not teks_asli or not kata_kunci:
        messagebox.showwarning("Input Error", "Mohon masukkan teks dan kata kunci.")
        return
    hasil_enkripsi = extended_vigenere_encrypt(teks_asli, kata_kunci)
    entry_teks.delete("1.0", tk.END)
    entry_teks.insert(tk.END, ', '.join(map(str, hasil_enkripsi)))
def decrypt():
    teks_enkripsi = entry_teks.get("1.0", tk.END).strip()
    kata_kunci = entry_kunci.get()
    if not teks_enkripsi or not kata_kunci:
        messagebox.showwarning("Input Error", "Mohon masukkan teks enkripsi dan kata kunci.")
        return
    try:
        teks_enkripsi_list = list(map(int, teks_enkripsi.split(', ')))
        hasil_dekripsi = extended_vigenere_decrypt(teks_enkripsi_list, kata_kunci)
        entry_teks.delete("1.0", tk.END)
        entry_teks.insert(tk.END, hasil_dekripsi)
    except ValueError:
        messagebox.showerror("Input Error", "Format enkripsi tidak valid. Pastikan input numerik dipisahkan dengan koma.")
def save_file():
    file_path = filedialog.asksaveasfilename(defaultextension=".txt", filetypes=[("Text files", "*.txt")])
    if file_path:
        with open(file_path, 'w', encoding='utf-8') as file:
            teks = entry_teks.get("1.0", tk.END)
            file.write(teks)
def upload_file():
    file_path = filedialog.askopenfilename(filetypes=[("All files", "*.*")])
    if file_path:
        with open(file_path, 'rb') as file:
            teks = file.read()
            entry_teks.delete("1.0", tk.END)
            entry_teks.insert(tk.END, teks.decode(errors='ignore'))
def encrypt_file():
    file_path = filedialog.askopenfilename(filetypes=[("All files", "*.*")])
    if file_path:
        kata_kunci = entry_kunci.get()
        if not kata_kunci:
            messagebox.showwarning("Input Error", "Mohon masukkan kata kunci.")
            return
        with open(file_path, 'rb') as file:
            data = file.read()
            hasil_enkripsi = extended_vigenere_encrypt(data.decode(errors='ignore'), kata_kunci)
            entry_teks.delete("1.0", tk.END)
            entry_teks.insert(tk.END, ', '.join(map(str, hasil_enkripsi)))
def decrypt_file():
    file_path = filedialog.askopenfilename(filetypes=[("Text files", "*.txt")])
    if file_path:
        kata_kunci = entry_kunci.get()
        if not kata_kunci:
            messagebox.showwarning("Input Error", "Mohon masukkan kata kunci.")
            return
        with open(file_path, 'r', encoding='utf-8') as file:
            teks_enkripsi = file.read().strip()
            try:
                teks_enkripsi_list = list(map(int, teks_enkripsi.split(', ')))
                hasil_dekripsi = extended_vigenere_decrypt(teks_enkripsi_list, kata_kunci)
                entry_teks.delete("1.0", tk.END)
                entry_teks.insert(tk.END, hasil_dekripsi)
            except ValueError:
                messagebox.showerror("Input Error", "Format enkripsi tidak valid. Pastikan input numerik dipisahkan dengan koma.")
root = tk.Tk()
root.title("Extended Vigenère Cipher")
root.configure(bg='pink')
frame_teks = tk.Frame(root, bg='pink')
frame_teks.pack(pady=10)
label_teks = tk.Label(frame_teks, text="Masukkan teks yang ingin dienkripsi:", bg='pink')
label_teks.pack()
entry_teks = tk.Text(frame_teks, width=50, height=10)
entry_teks.pack()
button_upload = tk.Button(root, text="Upload Dokumen Teks", command=upload_file)
button_upload.pack(pady=5)
button_save = tk.Button(root, text="Simpan Hasil", command=save_file)
button_save.pack(pady=5)
label_kunci = tk.Label(root, text="Masukkan kata kunci:", bg='pink')
label_kunci.pack(pady=5)
entry_kunci = tk.Entry(root, width=50)
entry_kunci.pack(pady=5)
frame_buttons = tk.Frame(root, bg='pink')
frame_buttons.pack(pady=5)
button_encrypt_numeric = tk.Button(frame_buttons, text="Enkripsi ke Numerik", command=encrypt)
button_encrypt_numeric.pack(side=tk.LEFT, padx=(0, 10))
button_decrypt = tk.Button(frame_buttons, text="Dekripsi", command=decrypt)
button_decrypt.pack(side=tk.LEFT, padx=(0, 10))
button_encrypt_file = tk.Button(frame_buttons, text="Enkripsi File", command=encrypt_file)
button_encrypt_file.pack(side=tk.LEFT, padx=(0, 10))
button_decrypt_file = tk.Button(frame_buttons, text="Dekripsi File", command=decrypt_file)
button_decrypt_file.pack(side=tk.LEFT, padx=(0, 10))
root.mainloop()

```

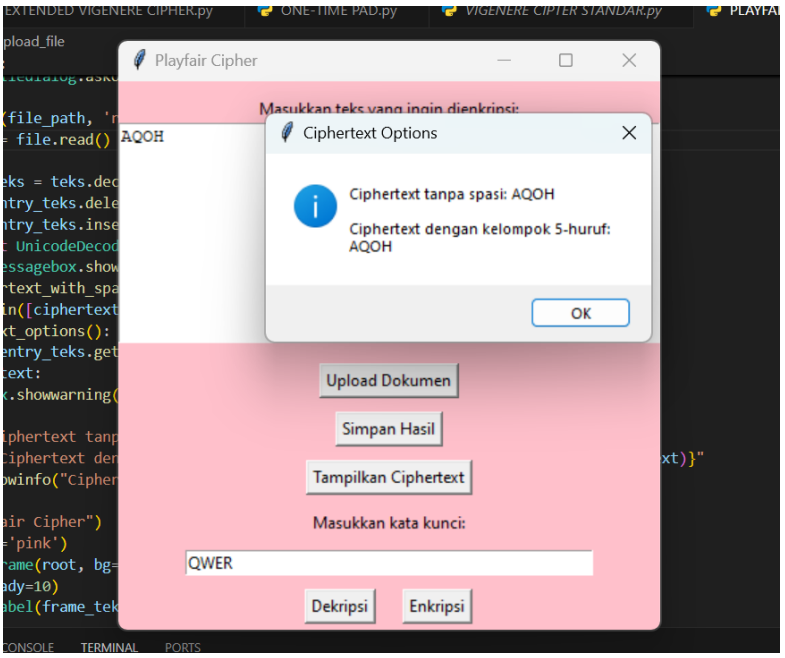


3. Playfair Cipher

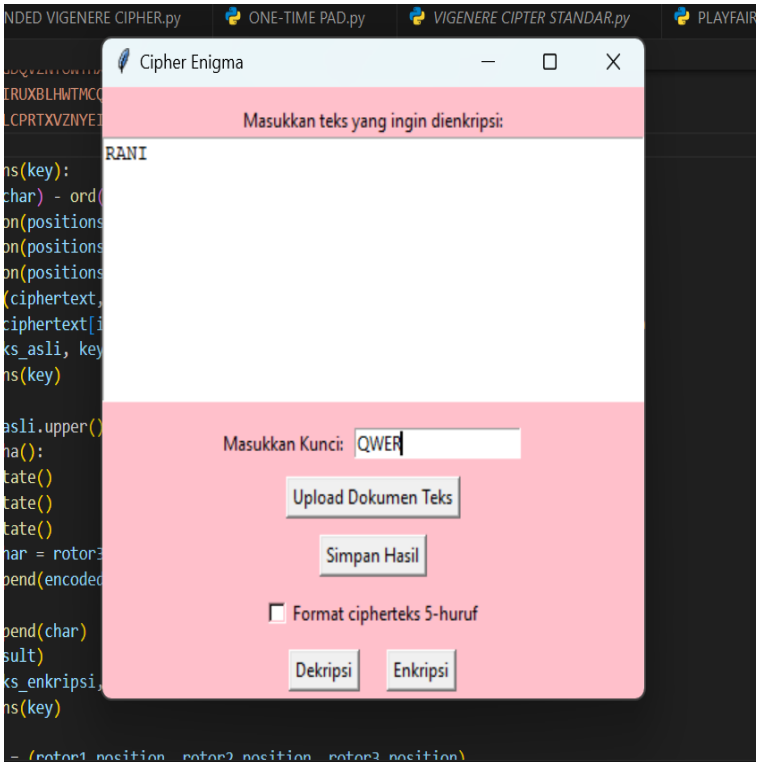
```python
import tkinter as tk
from tkinter import messagebox, filedialog
def create_playfair_table(kunci):
    kunci = kunci.upper().replace('J', 'I')
    seen = set()
    table = []
    for char in kunci:
        if char not in seen and char.isalpha():
            seen.add(char)
            table.append(char)
    for char in 'ABCDEFGHIKLMNOPQRSTUVWXYZ':
        if char not in seen:
            seen.add(char)
            table.append(char)
    return [table[i:i + 5] for i in range(0, 25, 5)]
def find_position(char, table):
    for i, row in enumerate(table):
        if char in row:
            return i, row.index(char)
    return None
def playfair_encrypt(teks_asli, kunci):
    table = create_playfair_table(kunci)
    teks_asli = teks_asli.upper().replace('J', 'I')
    pairs = []
    i = 0
    while i < len(teks_asli):
        char1 = teks_asli[i]
        if i + 1 < len(teks_asli):
            char2 = teks_asli[i + 1]
            if char1 == char2:
                pairs.append((char1, 'X'))
                i += 1
            else:
                pairs.append((char1, char2))
                i += 2
        else:
            pairs.append((char1, 'X'))
            i += 1
    enkripsi = ''
    for char1, char2 in pairs:
        row1, col1 = find_position(char1, table)
        row2, col2 = find_position(char2, table)
        if row1 == row2:
            enkripsi += table[row1][(col1 + 1) % 5]
            enkripsi += table[row2][(col2 + 1) % 5]
        elif col1 == col2:
            enkripsi += table[(row1 + 1) % 5][col1]
            enkripsi += table[(row2 + 1) % 5][col2]
        else:
            enkripsi += table[row1][col2]
            enkripsi += table[row2][col1]
    return enkripsi
def playfair_decrypt(teks_enkripsi, kunci):
    table = create_playfair_table(kunci)
    dekripsi = ''
    pairs = []
    i = 0
    while i < len(teks_enkripsi):
        pairs.append((teks_enkripsi[i], teks_enkripsi[i + 1]))
        i += 2
    for char1, char2 in pairs:
        row1, col1 = find_position(char1, table)
        row2, col2 = find_position(char2, table)
        if row1 == row2:
            dekripsi += table[row1][(col1 - 1) % 5]
            dekripsi += table[row2][(col2 - 1) % 5]
        elif col1 == col2:
            dekripsi += table[(row1 - 1) % 5][col1]
            dekripsi += table[(row2 - 1) % 5][col2]
        else:
            dekripsi += table[row1][col2]
            dekripsi += table[row2][col1]
    return dekripsi
def encrypt():
    teks_asli = entry_teks.get("1.0", tk.END).strip()
    kata_kunci = entry_kunci.get()
    if not teks_asli or not kata_kunci:
        messagebox.showwarning("Input Error", "Mohon masukkan teks dan kata kunci.")
        return
    hasil_enkripsi = playfair_encrypt(teks_asli, kata_kunci)
    entry_teks.delete("1.0", tk.END)
    entry_teks.insert(tk.END, hasil_enkripsi)
def decrypt():
    teks_enkripsi = entry_teks.get("1.0", tk.END).strip()
    kata_kunci = entry_kunci.get()
    if not teks_enkripsi or not kata_kunci:
        messagebox.showwarning("Input Error", "Mohon masukkan teks enkripsi dan kata kunci.")
        return
    hasil_dekripsi = playfair_decrypt(teks_enkripsi, kata_kunci)
    entry_teks.delete("1.0", tk.END)
    entry_teks.insert(tk.END, hasil_dekripsi)
def save_file():
    file_path = filedialog.asksaveasfilename(defaultextension=".txt", filetypes=[("Text files", "*.txt")])
    if file_path:
        with open(file_path, 'w', encoding='utf-8') as file:
            teks = entry_teks.get("1.0", tk.END)
            file.write(teks)
def upload_file():
    file_path = filedialog.askopenfilename(filetypes=[("All files", "*.*")])
    if file_path:
        with open(file_path, 'rb') as file:
            teks = file.read()
            try:
                teks = teks.decode('utf-8')
                entry_teks.delete("1.0", tk.END)
                entry_teks.insert(tk.END, teks)
            except UnicodeDecodeError:
                messagebox.showwarning("File Error", "File tidak dapat dibaca sebagai teks.")
def display_ciphertext_with_spaces(ciphertext):
    return ' '.join([ciphertext[i:i + 5] for i in range(0, len(ciphertext), 5)])
def show_ciphertext_options():
    ciphertext = entry_teks.get("1.0", tk.END).strip()
    if not ciphertext:
        messagebox.showwarning("Input Error", "Tidak ada cipherteks untuk ditampilkan.")
        return
    options = f"Ciphertext tanpa spasi: {ciphertext}\n\n"
    options += f"Ciphertext dengan kelompok 5-huruf:\n{display_ciphertext_with_spaces(ciphertext)}"
    messagebox.showinfo("Ciphertext Options", options)
root = tk.Tk()
root.title("Playfair Cipher")
root.configure(bg='pink')
frame_teks = tk.Frame(root, bg='pink')
frame_teks.pack(pady=10)
label_teks = tk.Label(frame_teks, text="Masukkan teks yang ingin dienkripsi:", bg='pink')
label_teks.pack()
entry_teks = tk.Text(frame_teks, width=50, height=10)
entry_teks.pack()
button_upload = tk.Button(root, text="Upload Dokumen", command=upload_file)
button_upload.pack(pady=5)
button_save = tk.Button(root, text="Simpan Hasil", command=save_file)
button_save.pack(pady=5)
button_show_ciphertext = tk.Button(root, text="Tampilkan Ciphertext", command=show_ciphertext_options)
button_show_ciphertext.pack(pady=5)
label_kunci = tk.Label(root, text="Masukkan kata kunci:", bg='pink')
label_kunci.pack(pady=5)
entry_kunci = tk.Entry(root, width=30)
entry_kunci.pack(pady=5)
frame_buttons = tk.Frame(root, bg='pink')
frame_buttons.pack(pady=5)
button_decrypt = tk.Button(frame_buttons, text="Dekripsi", command=decrypt)
button_decrypt.pack(side=tk.LEFT, padx=(0, 10))
button_encrypt = tk.Button(frame_buttons, text="Enkripsi", command=encrypt)
button_encrypt.pack(side=tk.RIGHT, padx=(10, 0))
root.mainloop()
```

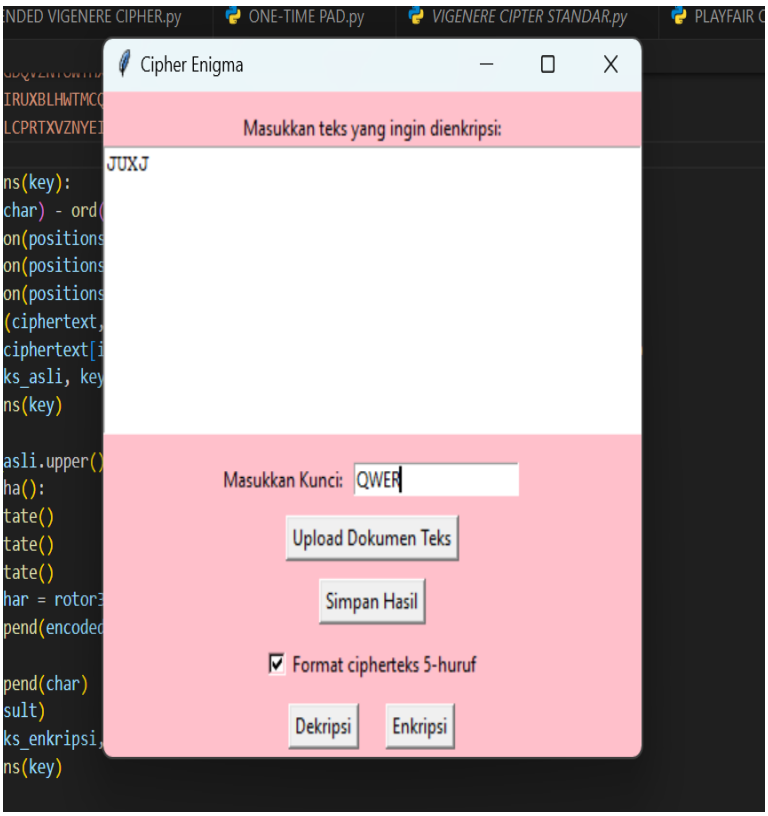


4. Enigma Cipher

```python
import tkinter as tk
from tkinter import messagebox, filedialog
class Rotor:
    def __init__(self, mapping):
        self.mapping = mapping
        self.position = 0
    def set_position(self, position):
        self.position = position % len(self.mapping)
    def rotate(self):
        self.position = (self.position + 1) % len(self.mapping)
    def encode(self, char):
        index = (ord(char) - ord('A') + self.position) % 26
        return self.mapping[index]
    def decode(self, char):
        index = self.mapping.index(char)
        return chr((index - self.position) % 26 + ord('A'))
rotor1 = Rotor('EKMFLGDQVZNTOWYHXUSPAIBRCJ')
rotor2 = Rotor('AJDKSIRUXBLHWTMCQGZNPYFVOE')
rotor3 = Rotor('BDFHJLCPRTXVZNYEIWGAKMUSQO')
original_text = ""
def set_rotor_positions(key):
    positions = [ord(char) - ord('A') for char in key.upper()]
    rotor1.set_position(positions[0])
    rotor2.set_position(positions[1] if len(positions) > 1 else positions[0])
    rotor3.set_position(positions[2] if len(positions) > 2 else positions[0])
def format_ciphertext(ciphertext, group_size=5):
    return ' '.join([ciphertext[i:i+group_size] for i in range(0, len(ciphertext), group_size)])
def enigma_encrypt(teks_asli, key):
    set_rotor_positions(key)
    result = []
    for char in teks_asli.upper():
        if char.isalpha():
            rotor1.rotate()
            rotor2.rotate()
            rotor3.rotate()
            encoded_char = rotor3.encode(rotor2.encode(rotor1.encode(char)))
            result.append(encoded_char)
        else:
            result.append(char)
    return ''.join(result)
def enigma_decrypt(teks_enkripsi, key):
    set_rotor_positions(key)
    result = []
    initial_positions = (rotor1.position, rotor2.position, rotor3.position)
    for char in teks_enkripsi.upper():
        if char.isalpha():
            decoded_char = rotor1.decode(rotor2.decode(rotor3.decode(char)))
            result.append(decoded_char)
            rotor1.rotate()
            rotor2.rotate()
            rotor3.rotate()
        else:
            result.append(char)
    rotor1.position, rotor2.position, rotor3.position = initial_positions
    return ''.join(result)
def encrypt():
    global original_text
    teks_asli = entry_teks.get("1.0", tk.END).strip()
    key = entry_key.get().strip()
    if not teks_asli:
        messagebox.showwarning("Input Error", "Mohon masukkan teks.")
        return
    if not key:
        messagebox.showwarning("Input Error", "Mohon masukkan kunci.")
        return
    original_text = teks_asli
    hasil_enkripsi = enigma_encrypt(teks_asli, key)
    if var_format.get() == 1:
        hasil_enkripsi = format_ciphertext(hasil_enkripsi)
    entry_teks.delete("1.0", tk.END)
    entry_teks.insert(tk.END, hasil_enkripsi)
def decrypt():
    teks_enkripsi = entry_teks.get("1.0", tk.END).strip()
    key = entry_key.get().strip()
    if not teks_enkripsi:
        messagebox.showwarning("Input Error", "Mohon masukkan teks enkripsi.")
        return
    if not key:
        messagebox.showwarning("Input Error", "Mohon masukkan kunci.")
        return
    hasil_dekripsi = enigma_decrypt(teks_enkripsi, key)
    entry_teks.delete("1.0", tk.END)
    entry_teks.insert(tk.END, original_text)
def save_file():
    file_path = filedialog.asksaveasfilename(defaultextension=".txt", filetypes=[("Text files", "*.txt")])
    if file_path:
        with open(file_path, 'w', encoding='utf-8') as file:
            teks = entry_teks.get("1.0", tk.END)
            file.write(teks)
def upload_file():
    file_path = filedialog.askopenfilename(filetypes=[("Text files", "*.txt")])
    if file_path:
        with open(file_path, 'r', encoding='utf-8') as file:
            teks = file.read()
            entry_teks.delete("1.0", tk.END)
            entry_teks.insert(tk.END, teks)
root = tk.Tk()
root.title("Cipher Enigma")
root.configure(bg='pink')
frame_teks = tk.Frame(root, bg='pink')
frame_teks.pack(pady=10)
label_teks = tk.Label(frame_teks, text="Masukkan teks yang ingin dienkripsi:", bg='pink')
label_teks.pack()
entry_teks = tk.Text(frame_teks, width=50, height=10)
entry_teks.pack()
frame_key = tk.Frame(root, bg='pink')
frame_key.pack(pady=5)
label_key = tk.Label(frame_key, text="Masukkan Kunci:", bg='pink')
label_key.pack(side=tk.LEFT)
entry_key = tk.Entry(frame_key)
entry_key.pack(side=tk.LEFT, padx=5)
button_upload = tk.Button(root, text="Upload Dokumen Teks", command=upload_file)
button_upload.pack(pady=5)
button_save = tk.Button(root, text="Simpan Hasil", command=save_file)
button_save.pack(pady=5)
var_format = tk.IntVar()
check_format = tk.Checkbutton(root, text="Format cipherteks 5-huruf", variable=var_format, bg='pink')
check_format.pack(pady=5)
frame_buttons = tk.Frame(root, bg='pink')
frame_buttons.pack(pady=5)
button_decrypt = tk.Button(frame_buttons, text="Dekripsi", command=decrypt)
button_decrypt.pack(side=tk.LEFT, padx=(0, 10))
button_encrypt = tk.Button(frame_buttons, text="Enkripsi", command=encrypt)
button_encrypt.pack(side=tk.RIGHT, padx=(10, 0))
root.mainloop()
```



5. One-Time pad

```python
import tkinter as tk
from tkinter import messagebox, filedialog
import random
import string
def generate_random_key(length):
    return ''.join(random.choice(string.ascii_uppercase) for _ in range(length))
def one_time_pad_encrypt(plain_text, key):
    encrypted_text = []
    for pt_char, key_char in zip(plain_text.upper(), key):
        if pt_char.isalpha():
            encrypted_char = chr((ord(pt_char) - ord('A') + ord(key_char) - ord('A')) % 26 + ord('A'))
            encrypted_text.append(encrypted_char)
        else:
            encrypted_text.append(pt_char)
    return ''.join(encrypted_text)
def one_time_pad_decrypt(encrypted_text, key):
    decrypted_text = []
    for enc_char, key_char in zip(encrypted_text.upper(), key):
        if enc_char.isalpha():
            decrypted_char = chr((ord(enc_char) - ord('A') - (ord(key_char) - ord('A'))) % 26 + ord('A'))
            decrypted_text.append(decrypted_char)
        else:
            decrypted_text.append(enc_char)
    return ''.join(decrypted_text)
def encrypt():
    teks_asli = entry_teks.get("1.0", tk.END).strip()
    if not teks_asli:
        messagebox.showwarning("Input Error", "Mohon masukkan teks.")
        return
    key = entry_key.get().strip()
    if len(key) < len(teks_asli):
        messagebox.showwarning("Key Error", "Kunci harus sepanjang atau lebih panjang dari teks.")
        return
    hasil_enkripsi = one_time_pad_encrypt(teks_asli, key)
    entry_teks.delete("1.0", tk.END)
    entry_teks.insert(tk.END, f"{hasil_enkripsi}")
def decrypt():
    teks_enkripsi = entry_teks.get("1.0", tk.END).strip()
    if not teks_enkripsi:
        messagebox.showwarning("Input Error", "Mohon masukkan teks enkripsi.")
        return
    key = entry_key.get().strip()
    if len(key) < len(teks_enkripsi):
        messagebox.showwarning("Key Error", "Kunci harus sepanjang atau lebih panjang dari ciphertext.")
        return
    hasil_dekripsi = one_time_pad_decrypt(teks_enkripsi, key)
    entry_teks.delete("1.0", tk.END)
    entry_teks.insert(tk.END, f"{hasil_dekripsi}")
def save_file():
    file_path = filedialog.asksaveasfilename(defaultextension=".txt", filetypes=[("Text files", "*.txt")])
    if file_path:
        with open(file_path, 'w', encoding='utf-8') as file:
            teks = entry_teks.get("1.0", tk.END)
            file.write(teks)
def upload_file():
    file_path = filedialog.askopenfilename(filetypes=[("All files", "*.*")])
    if file_path:
        with open(file_path, 'rb') as file:
            teks = file.read()
            entry_teks.delete("1.0", tk.END)
            entry_teks.insert(tk.END, teks.decode('utf-8', errors='ignore'))
def upload_key_file():
    key_file_path = filedialog.askopenfilename(filetypes=[("Text files", "*.txt")])
    if key_file_path:
        with open(key_file_path, 'r', encoding='utf-8') as file:
            key = file.read().strip()
            entry_key.delete(0, tk.END)
            entry_key.insert(0, key)
root = tk.Tk()
root.title("One-Time Pad Cipher")
root.configure(bg='pink')
frame_teks = tk.Frame(root, bg='pink')
frame_teks.pack(pady=10)
label_teks = tk.Label(frame_teks, text="Masukkan teks yang ingin dienkripsi:", bg='pink')
label_teks.pack()
entry_teks = tk.Text(frame_teks, width=50, height=10)
entry_teks.pack()
frame_key = tk.Frame(root, bg='pink')
frame_key.pack(pady=10)
label_key = tk.Label(frame_key, text="Masukkan kunci:", bg='pink')
label_key.pack()
entry_key = tk.Entry(frame_key, width=50)
entry_key.pack()
button_upload = tk.Button(root, text="Upload Dokumen", command=upload_file)
button_upload.pack(pady=5)
button_save = tk.Button(root, text="Simpan Hasil", command=save_file)
button_save.pack(pady=5)
button_upload_key = tk.Button(root, text="Upload Kunci dari File", command=upload_key_file)
button_upload_key.pack(pady=5)
frame_buttons = tk.Frame(root, bg='pink')
frame_buttons.pack(pady=5)
button_decrypt = tk.Button(frame_buttons, text="Dekripsi", command=decrypt)
button_decrypt.pack(side=tk.LEFT, padx=(0, 10))
button_encrypt = tk.Button(frame_buttons, text="Enkripsi", command=encrypt)
button_encrypt.pack(side=tk.RIGHT, padx=(10, 0))
root.mainloop()
```