CIPHER KLASIK

Ditujukan untuk memenuhi Tugas Matakuliah Kriptografi

Oleh:

Wildan Hanif(20123074)

Zulfitrah Akbar(20123084)

PROGRAM STUDI INFORMATIKA S1

UNIVERSITAS TEKNOLOGI DIGITAL

BANDUNG

2025

- Caesar Cipher

```
def caesar_encrypt(text, shift):
    return ''.join(chr((ord(c) - s + shift) % 26 + s) if
c.isalpha()
                   else c for c, s in [(ch, ord('A') if
ch.isupper() else ord('a'))
                   for ch in text])

def caesar_decrypt(ciphertext, shift):
    return caesar_encrypt(ciphertext, -shift)
```

```
--- 1. Caesar Cipher ---
Plaintext: Hello World! This is a secret message.
Kunci (Shift): 3
Ciphertext: Khoor Zruog! Wklv lv d vhfuhw phvvdjh.
Decrypted: Hello World! This is a secret message.
```

- Rumus:

$$C = (P + k) \mod 26$$

$$P = (C - k) \mod 26$$

dengan:

- P = plaintext (huruf sebagai angka 0–25)
- C = ciphertext
- k = kunci (nilai geser)

Teori Singkat:
Caesar Cipher adalah salah satu bentuk cipher substitusi paling sederhana, ditemukan oleh Julius Caesar. Algoritma ini bekerja dengan cara menggeser setiap huruf dalam alfabet sejauh *n* langkah. Misalnya jika pergeseran = 3, maka huruf **A → D, B → E, C → F**, dan seterusnya. Jika sampai di ujung alfabet, maka dilanjutkan lagi dari awal (wrap-around). Cipher ini hanya bekerja pada huruf, sedangkan karakter lain seperti angka dan tanda baca tidak berubah.
Kelemahan:
Caesar Cipher sangat lemah terhadap serangan brute force karena hanya ada 25 kemungkinan kunci. Selain itu, pola frekuensi huruf tetap terlihat jelas sehingga mudah ditebak dengan analisis frekuensi.

- Affine Cipher

```
def affine_encrypt(text, a, b):
    return ''.join(chr(((a*(ord(c)-s)+b) % 26)+s) if
c.isalpha()
                   else c for c, s in [(ch, ord('A') if
ch.isupper() else ord('a'))
                   for ch in text])

def affine_decrypt(ciphertext, a, b):
    inv_a = pow(a, -1, 26)
    return ''.join(chr((inv_a*(ord(c)-s-b) % 26)+s) if
c.isalpha()
                   else c for c, s in [(ch, ord('A') if
ch.isupper() else ord('a'))
                   for ch in ciphertext])
```

```
--- 2. Affine Cipher ---
Plaintext: Hello World! This is a secret message.
Kunci (a, b): (5, 8)
Ciphertext: Rclla Oaplx! Zrwu wu i ucspcz qcuuimc.
Decrypted: Hello World! This is a secret message.
```

Rumus enkripsi:

$$C = (aP + b) \mod 26$$

Rumus dekripsi:

$$P = a^{-1}(C - b) \mod 26$$

di mana a^{-1} adalah invers modulo dari a.

Teori Singkat:
Affine Cipher merupakan pengembangan dari Caesar Cipher dengan menambahkan fungsi perkalian dan penjumlahan. Cipher ini menggunakan dua kunci: **a** (harus koprima dengan 26) dan **b** (bilangan bulat).
Kelemahan:
- Masih rentan terhadap analisis frekuensi.
- Kunci terbatas karena nilai a harus koprima dengan 26.

- Vigenere Cipher

```
def vigenere_encrypt(text, key):
    res, j = "", 0
    for c in text:
        if c.isalpha():
            s = ord('A') if c.isupper() else ord('a')
            k = ord(key[j % len(key)].lower()) - ord('a')
            res += chr((ord(c)-s+k)%26+s); j += 1
        else: res += c
    return res

def vigenere_decrypt(cipher, key):
    res, j = "", 0
    for c in cipher:
        if
c.isalpha():
            s =
ord('A') if
c.isupper() else
ord('a')
            k =
ord(key[j %
len(key)].lower()) - ord('a')
            res += chr((ord(c)-s-k)%26+s); j += 1
        else: res += c
    return res
```

- **Rumus:**

$$C_i = (P_i + K_i) \mod 26$$

$$P_i = (C_i - K_i) \mod 26$$

dengan:

- P_i = huruf plaintext ke-i
- C_i = huruf ciphertext ke-i
- K_i = huruf kunci ke-i (diulang sepanjang plaintext)

```
--- 3. Vigenere Cipher ---
Plaintext: Hello World! This is a secret message.
Kunci: UNIVERSITAS
Ciphertext: Brtgs Ngzed! Lbva dw r kmvrwn zmnwrym.
Decrypted: Hello World! This is a secret message.
```

Teori Singkat:
Vigenere Cipher ditemukan oleh Blaise de Vigenère pada abad ke-16. Cipher ini adalah pengembangan dari Caesar Cipher, menggunakan kata kunci untuk menentukan besar pergeseran tiap huruf. Misalnya plaintext HELLO dengan kunci KEY: huruf pertama digeser sesuai ‘K’, kedua sesuai ‘E’, ketiga sesuai ‘Y’, lalu diulang lagi. Dengan metode ini, cipher menjadi lebih sulit ditebak karena pola pergeseran tidak seragam.
Kelemahan:
Vigenere Cipher lebih kuat dibanding Caesar, tetapi masih bisa dipecahkan dengan analisis Kasiski atau Index of Coincidence yang mencari pola pengulangan kunci.

- Playfair Cipher

```python
def generate_playfair_matrix(key):
    key = key.upper().replace("J","I")
    seen = ""
    for c in key:
        if c not in seen and c.isalpha(): seen += c
    for c in "ABCDEFGHIKLMNOPQRSTUVWXYZ":
        if c not in seen: seen += c
    return np.array(list(seen)).reshape(5,5)

def playfair_encrypt(text, matrix):
    text = text.upper().replace("J","I").replace(" ","")
    if len(text)%2: text += "X"
    res, i = "", 0
    while i < len(text):
        a,b = text[i], text[i+1] if i+1<len(text) else "X"
        if a==b: b="X"; i-=1
        r1,c1 = np.where(matrix==a); r2,c2 = np.where(matrix==b)
        r1,c1,r2,c2 = r1[0],c1[0],r2[0],c2[0]
        if r1==r2: res+=matrix[r1][(c1+1)%5]+matrix[r2][(c2+1)%5]
        elif c1==c2: res+=matrix[(r1+1)%5][c1]+matrix[(r2+1)%5][c2]
        else: res+=matrix[r1][c2]+matrix[r2][c1]
        i+=2
    return res
```

```
--- 4. Playfair Cipher ---
Plaintext: HELLO
Kunci: KEYWORD
Matriks Kunci Playfair:
[['K' 'E' 'Y' 'W' 'O']
 ['R' 'D' 'A' 'B' 'C']
 ['F' 'G' 'H' 'I' 'L']
 ['M' 'N' 'P' 'Q' 'S']
 ['T' 'U' 'V' 'X' 'Z']]
Ciphertext: GYIZSC
Decrypted: HELXLO
```

Teori Singkat:
Playfair Cipher ditemukan oleh Charles Wheatstone (1854) dan dipopulerkan oleh Lord Playfair. Cipher ini bekerja dengan pasangan huruf (digraph) menggunakan tabel 5×5 huruf (I dan J digabung).

Aturan Rumus:
- Satu baris: $(x,y)\to(x,(y+1)\mod 5)(x, y) \to (x, (y+1) \mod 5)(x,y)\to(x,(y+1)\mod 5)$
- Satu kolom: $(x,y)\to((x+1)\mod 5,y)(x, y) \to ((x+1) \mod 5, y)(x,y)\to((x+1)\mod 5,y)$
- Beda baris & kolom: tukar posisi menjadi koordinat silang dalam matriks.

Kelemahan:
Lebih kuat daripada Caesar/Vigenere karena berbasis digraph, tetapi masih dapat dianalisis dengan frekuensi pasangan huruf.

-Hill Cipher

```python
def hill_encrypt(text, key):
    text = text.upper().replace(" ","")
    if len(text)%2: text+="X"
    res=""
    for i in range(0,len(text),2):
        pair=[ord(text[i])-65, ord(text[i+1])-65]
        c=np.dot(key,pair)%26
        res+=chr(c[0]+65)+chr(c[1]+65)
    return res
```

```
--- 5. Hill Cipher ---
Plaintext: HI
Matriks Kunci Hill:
[[3 3]
 [2 5]]
Ciphertext: TC
Decrypted: HI
```

- Rumus:

$$C = K \times P \mod 26$$

$$P = K^{-1} \times C \mod 26$$

dengan:

- P = vektor plaintext
- C = vektor ciphertext
- K = matriks kunci (harus memiliki determinan relatif prima dengan 26)

Teori Singkat: Hill Cipher menggunakan aljabar linear. Plaintext diubah menjadi vektor angka, lalu dikalikan dengan matriks kunci (mod 26). Matriks kunci harus invertible agar dekripsi bisa dilakukan.

Kelemahan: Jika ada cukup banyak pasangan plaintext–ciphertext, kunci bisa dihitung.

Kode lebih lengkapnya:

```
# -*- coding: utf-8 -*-
"""
Tugas 1: Implementasi Cipher Klasik
Nama: Wildan Hanif, Zulfitrah Akbar
NIM: [20123074], [20123084]
Mata Kuliah: Kriptografi

Program ini mengimplementasikan tiga algoritma cipher klasik:
1. Caesar Cipher
2. Affine Cipher
3. Vigenere Cipher
"""

import numpy as np

# ============================================================================
# 1. CAESAR CIPHER
# ============================================================================
def caesar_encrypt(text, shift):
    """
    Mengenkripsi teks menggunakan Caesar Cipher.
    Hanya karakter alfabet yang dienkripsi, karakter lain diabaikan.
    """
    result = ""
    for char in text:
        if char.isalpha(): # Hanya proses huruf
            start = ord('A') if char.isupper() else ord('a')
            # Rumus Enkripsi Caesar: C = (P + K) mod 26
            encrypted_char = chr((ord(char) - start + shift) % 26 + start)
            result += encrypted_char
        else:
            result += char # Karakter non-alfabet tidak diubah
    return result

def caesar_decrypt(ciphertext, shift):
    """
    Mendekripsi teks dari Caesar Cipher.
    Ini sama dengan enkripsi dengan pergeseran negatif.
    """
    # Rumus Dekripsi Caesar: P = (C - K) mod 26
    return caesar_encrypt(ciphertext, -shift)

# ============================================================================
# 2. AFFINE CIPHER
# ============================================================================
def egcd(a, b):
    """
    Extended Euclidean Algorithm untuk mencari modular inverse.
    """
    if a == 0:
        return (b, 0, 1)
    else:
        g, y, x = egcd(b % a, a)
        return (g, x - (b // a) * y, y)

def mod_inverse(a, m):
    """
    Mencari modular inverse dari a mod m.
    Diperlukan untuk dekripsi Affine Cipher.
    """
    g, x, y = egcd(a, m)
    if g != 1:
        raise Exception('Modular inverse tidak ada')
    return x % m

def affine_encrypt(text, a, b):
    """
    Mengenkripsi teks menggunakan Affine Cipher.
    Parameter 'a' harus koprima dengan 26.
    """
    if egcd(a, 26)[0] != 1:
        raise ValueError("'a' harus koprima dengan 26.")

    result = ""
    for char in text:
        if char.isalpha():
            start = ord('A') if char.isupper() else ord('a')
            # Rumus Enkripsi Affine: C = (a*P + b) mod 26
            encrypted_char = chr(((a * (ord(char) - start) + b) % 26) + start)
            result += encrypted_char
        else:
            result += char
    return result

def affine_decrypt(ciphertext, a, b):
    """
    Mendekripsi teks dari Affine Cipher.
    """
    if egcd(a, 26)[0] != 1:
        raise ValueError("'a' harus koprima dengan 26.")

    result = ""
    mod_inv_a = mod_inverse(a, 26)

    for char in ciphertext:
        if char.isalpha():
            start = ord('A') if char.isupper() else ord('a')
            # Rumus Dekripsi Affine: P = a^-1 * (C - b) mod 26
            decrypted_char = chr((mod_inv_a * (ord(char) - start - b)) % 26 + start)
            result += decrypted_char
        else:
            result += char
```

```python
    return result

# ============================================================================
# 3. VIGENERE CIPHER
# ============================================================================
def vigenere_encrypt(text, key):
    """
    Mengenkripsi teks menggunakan Vigenere Cipher.
    """
    result = ""
    key_index = 0
    key = key.lower() # Standarisasi kunci ke huruf kecil

    for char in text:
        if char.isalpha():
            start = ord('A') if char.isupper() else ord('a')
            # Tentukan pergeseran dari huruf kunci saat ini
            shift = ord(key[key_index % len(key)]) - ord('a')

            # Enkripsi karakter
            encrypted_char = chr((ord(char) - start + shift) % 26 + start)
            result += encrypted_char

            # Pindah ke huruf kunci berikutnya
            key_index += 1
        else:
            result += char
    return result

def vigenere_decrypt(ciphertext, key):
    """
    Mendekripsi teks dari Vigenere Cipher.
    """
    result = ""
    key_index = 0
    key = key.lower()

    for char in ciphertext:
        if char.isalpha():
            start = ord('A') if char.isupper() else ord('a')
            # Tentukan pergeseran (negatif) dari huruf kunci
            shift = ord(key[key_index % len(key)]) - ord('a')

            # Dekripsi karakter
            decrypted_char = chr((ord(char) - start - shift) % 26 + start)
            result += decrypted_char

            key_index += 1
        else:
            result += char
    return result

# ============================================================================
# 4. PLAYFAIR CIPHER
# ============================================================================
def generate_playfair_matrix(key):
    """
    Membuat matriks 5x5 untuk Playfair Cipher dari kunci.
    """
    key = key.upper().replace("J", "I")
    matrix = ""
    for char in key:
        if char not in matrix and char.isalpha():
            matrix += char
    for char in "ABCDEFGHIKLMNOPQRSTUVWXYZ":
        if char not in matrix:
            matrix += char
    return np.array(list(matrix)).reshape(5, 5)

def find_position(matrix, char):
    """
    Mencari posisi (baris, kolom) dari huruf dalam matriks Playfair.
    """
    if char == "J":
        char = "I"
    pos = np.where(matrix == char)
    return pos[0][0], pos[1][0]

def playfair_encrypt(text, matrix):
    """
    Mengenkripsi teks menggunakan Playfair Cipher.
    """
    text = text.upper().replace("J", "I").replace(" ", "")
    pairs = []
    i = 0
    while i < len(text):
        a = text[i]
        b = text[i+1] if i+1 < len(text) else 'X'
        if a == b:
            pairs.append(a + 'X')
            i += 1
        else:
            pairs.append(a + b)
            i += 2
    result = ""
    for pair in pairs:
        row1, col1 = find_position(matrix, pair[0])
        row2, col2 = find_position(matrix, pair[1])
        if row1 == row2:
            result += matrix[row1][(col1+1) % 5]
            result += matrix[row2][(col2+1) % 5]
        elif col1 == col2:
            result += matrix[(row1+1) % 5][col1]
            result += matrix[(row2+1) % 5][col2]
        else:
            result += matrix[row1][col2]
```

```
            result += matrix[row2][col1]
    return result

def playfair_decrypt(ciphertext, matrix):
    """
    Mendekripsi teks dari Playfair Cipher.
    """
    result = ""
    i = 0
    while i < len(ciphertext):
        a = ciphertext[i]
        b = ciphertext[i+1]
        row1, col1 = find_position(matrix, a)
        row2, col2 = find_position(matrix, b)
        if row1 == row2:
            result += matrix[row1][(col1-1) % 5]
            result += matrix[row2][(col2-1) % 5]
        elif col1 == col2:
            result += matrix[(row1-1) % 5][col1]
            result += matrix[(row2-1) % 5][col2]
        else:
            result += matrix[row1][col2]
            result += matrix[row2][col1]
        i += 2
    return result

# ==============================================================================
# 5. HILL CIPHER (2x2 matrix)
# ==============================================================================
def hill_encrypt(text, key_matrix):
    """
    Mengenkripsi teks menggunakan Hill Cipher (matriks 2x2).
    """
    text = text.upper().replace(" ", "")
    if len(text) % 2 != 0:
        text += 'X'
    result = ""
    for i in range(0, len(text), 2):
        pair = [ord(text[i]) - 65, ord(text[i+1]) - 65]
        res = np.dot(key_matrix, pair) % 26
        result += chr(res[0] + 65) + chr(res[1] + 65)
    return result

def matrix_mod_inverse(matrix, modulus):
    """
    Menghitung invers matriks 2x2 dalam modulo tertentu.
    """
    det = int(np.round(np.linalg.det(matrix))) % modulus
    det_inv = pow(det, -1, modulus)
    matrix_adj = np.array([[matrix[1][1], -matrix[0][1]],
                           [-matrix[1][0], matrix[0][0]]])
    return (det_inv * matrix_adj) % modulus

def hill_decrypt(ciphertext, key_matrix):
    """
    Mendekripsi teks dari Hill Cipher (matriks 2x2).
    """
    inv_matrix = matrix_mod_inverse(key_matrix, 26)
    result = ""
    for i in range(0, len(ciphertext), 2):
        pair = [ord(ciphertext[i]) - 65, ord(ciphertext[i+1]) - 65]
        res = np.dot(inv_matrix, pair) % 26
        result += chr(int(res[0]) + 65) + chr(int(res[1]) + 65)
    return result

# ==============================================================================
# FUNGSI UTAMA UNTUK DEMONSTRASI
# ==============================================================================
def main():
    print("===== DEMO PROGRAM CIPHER KLASIK =====")

    # Contoh Plaintext dan Kunci
    plaintext = "Hello World! This is a secret message."

    # --- Caesar Cipher ---
    print("\n--- 1. Caesar Cipher ---")
    caesar_key = 3
    print(f"Plaintext: {plaintext}")
    print(f"Kunci (Shift): {caesar_key}")
    encrypted_caesar = caesar_encrypt(plaintext, caesar_key)
    print(f"Ciphertext: {encrypted_caesar}")
    decrypted_caesar = caesar_decrypt(encrypted_caesar, caesar_key)
    print(f"Decrypted: {decrypted_caesar}")

    # --- Affine Cipher ---
    print("\n--- 2. Affine Cipher ---")
    affine_key_a = 5
    affine_key_b = 8
    print(f"Plaintext: {plaintext}")
    print(f"Kunci (a, b): ({affine_key_a}, {affine_key_b})")
    try:
        encrypted_affine = affine_encrypt(plaintext, affine_key_a, affine_key_b)
        print(f"Ciphertext: {encrypted_affine}")
        decrypted_affine = affine_decrypt(encrypted_affine, affine_key_a, affine_key_b)
        print(f"Decrypted: {decrypted_affine}")
    except ValueError as e:
        print(f"Error: {e}")

    # --- Vigenere Cipher ---
    print("\n--- 3. Vigenere Cipher ---")
    vigenere_key = "UNIVERSITAS"
    print(f"Plaintext: {plaintext}")
    print(f"Kunci: {vigenere_key}")
    encrypted_vigenere = vigenere_encrypt(plaintext, vigenere_key)
    print(f"Ciphertext: {encrypted_vigenere}")
    decrypted_vigenere = vigenere_decrypt(encrypted_vigenere, vigenere_key)
```

```
    print(f"Decrypted: {decrypted_vigenere}")

     # --- Playfair Cipher ---
    print("\n--- 4. Playfair Cipher ---")
    playfair_key = "KEYWORD"
    playfair_matrix = generate_playfair_matrix(playfair_key)
    playfair_text = "HELLO"
    print(f"Plaintext: {playfair_text}")
    print(f"Kunci: {playfair_key}")
    print(f"Matriks Kunci Playfair:\n{playfair_matrix}")
    encrypted_playfair = playfair_encrypt(playfair_text, playfair_matrix)
    print(f"Ciphertext: {encrypted_playfair}")
    print(f"Decrypted: {playfair_decrypt(encrypted_playfair, playfair_matrix)}")

    # --- Hill Cipher ---
    print("\n--- 5. Hill Cipher ---")
    hill_key = np.array([[3, 3],
                         [2, 5]])
    hill_text = "HI"
    print(f"Plaintext: {hill_text}")
    print(f"Matriks Kunci Hill:\n{hill_key}")
    encrypted_hill = hill_encrypt(hill_text, hill_key)
    print(f"Ciphertext: {encrypted_hill}")
    print(f"Decrypted: {hill_decrypt(encrypted_hill, hill_key)}")

if __name__ == '__main__':
    main()
```

TEST DENGAN CYBERTOOL

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
===== PROGRAM CIPHER KLASIK =====
Pilih Cipher:
1. Caesar Cipher
2. Affine Cipher
3. Vigenere Cipher
4. Playfair Cipher
5. Hill Cipher
0. Keluar
Masukkan pilihan Anda: 1
Pilih mode (1: Enkripsi, 2: Dekripsi): 1
Masukkan teks: DIGITECH
Masukkan kunci pergeseran (angka): 7

Ciphertext: KPNPALJO

Apakah Anda ingin menyimpan hasil ini ke file? (y/n):
```