Tugas 3 Kriptografi

Pertemuan 7

Disusun Oleh:

JAYA GOVAL UNEDO HUTASOIT

PROGRAM STUDI S-1 TEKNIK INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PADJADJARAN

JATINANGOR

2024

Penjelasan Program

import numpy as np

from sympy import Matrix

Program ini menggunakan numpy untuk menangani operasi matriks dasar dan sympy untuk menghitung invers matriks modulo 26, yang diperlukan dalam dekripsi Hill Cipher.

```
def encrypt_hill(plain_text, key_matrix):
    plain_text = plain_text.replace(" ", "").upper()
    n = key_matrix.shape[0]
    padded_text = plain_text + "X" * ((n - len(plain_text) % n) % n)

    encrypted_text = ""
    for i in range(0, len(padded_text), n):
        vector = np.array([ord(char) - ord('A') for char in padded_text[i:i+n]])
        encrypted_vector = np.dot(key_matrix, vector) % 26
        encrypted_text += ''.join(chr(num + ord('A')) for num in encrypted_vector)

    return encrypted_text
```

Penjelasan:

- Menghilangkan spasi dan mengonversi teks ke huruf kapital.
- Jika panjang teks tidak sesuai dengan ukuran matriks, teks dipasangkan dengan karakter "X".
- Setiap kelompok huruf diubah menjadi vektor, lalu dikalikan dengan matriks kunci.
- Hasilnya dikonversi kembali menjadi karakter.

```
def decrypt_hill(cipher_text, key_matrix):
    n = key_matrix.shape[0]
    inverse_key_matrix = Matrix(key_matrix).inv_mod(26)
    inverse_key_matrix = np.array(inverse_key_matrix).astype(int)

    decrypted_text = ""
    for i in range(0, len(cipher_text), n):
```

```
        vector = np.array([ord(char) - ord('A') for char in cipher_text[i:i+n]])
        decrypted_vector = np.dot(inverse_key_matrix, vector) % 26
        decrypted_text += ''.join(chr(num + ord('A')) for num in decrypted_vector)

    return decrypted_text
```

Penjelasan:

- Menghitung invers dari matriks kunci dalam modulo 26.
- Mengubah setiap bagian ciphertext menjadi vektor, lalu mengalikannya dengan invers matriks kunci untuk mendapatkan teks asli.

```
def find_key_hill(plain_text, cipher_text, n):
    plain_text = plain_text.replace(" ", "").upper()
    cipher_text = cipher_text.replace(" ", "").upper()

    if len(plain_text) < n * n or len(cipher_text) < n * n:
        print("Jumlah karakter tidak cukup untuk menentukan kunci.")
        return None

    plain_matrix = []
    cipher_matrix = []

    for i in range(n):
        plain_matrix.append([ord(char) - ord('A') for char in plain_text[i*n:(i+1)*n]])
        cipher_matrix.append([ord(char) - ord('A') for char in cipher_text[i*n:(i+1)*n]])

    plain_matrix = Matrix(plain_matrix)
    cipher_matrix = Matrix(cipher_matrix)

    try:
        key_matrix = cipher_matrix * plain_matrix.inv_mod(26)
```

```
            key_matrix = np.array(key_matrix).astype(int) % 26
            return key_matrix
        except ValueError:
            print("Tidak dapat menemukan kunci karena matriks plaintext tidak invertible.")
            return None
```

Penjelasan:

- Membentuk matriks dari teks asli dan teks terenkripsi.
- Jika teks tidak dapat diinversi, fungsi akan mengembalikan pesan kesalahan.

```
def main():
    print("===== Program Hill Cipher =====")

    plain_text = input("Masukkan plaintext: ").strip()
    while True:
        try:
            n = int(input("Masukkan ukuran matriks kunci (misalnya 2 untuk matriks 2x2): "))
            if n <= 0 or n > 10:
                print("Ukuran matriks terlalu besar atau kecil. Masukkan nilai antara 1 hingga 10.")
                continue
            break
        except ValueError:
            print("Input tidak valid. Masukkan angka integer.")

    print(f"Masukkan {n*n} elemen matriks kunci secara berurutan (contoh: 3 3 2 5 untuk matriks 2x2):")
    while True:
        try:
            key_elements = list(map(int, input().split()))
            if len(key_elements) != n * n:
                print(f"Jumlah elemen tidak sesuai. Masukkan tepat {n*n} angka.")
```

```
            continue
        key_matrix = np.array(key_elements).reshape(n, n)
        break
    except ValueError:
        print("Input tidak valid. Masukkan angka integer.")

encrypted_text = encrypt_hill(plain_text, key_matrix)
print(f"Teks terenkripsi: {encrypted_text}")

decrypted_text = decrypt_hill(encrypted_text, key_matrix)
print(f"Teks didekripsi: {decrypted_text}")

find_key = input("Apakah Anda ingin mencari kunci? (y/n): ").strip().lower()
if find_key == 'y':
    known_plain_text = input("Masukkan plaintext yang diketahui: ").strip()
    known_cipher_text = input("Masukkan ciphertext yang diketahui: ").strip()
    found_key = find_key_hill(known_plain_text, known_cipher_text, n)
    if found_key is not None:
        print(f"Kunci yang ditemukan:\n{found_key}")
    else:
        print("Kunci tidak dapat ditemukan.")
```

- Mengambil input teks asli, ukuran matriks, dan elemen-elemen matriks kunci.
- Menampilkan teks terenkripsi dan hasil dekripsinya.
- Menawarkan opsi untuk menemukan kunci berdasarkan plaintext dan ciphertext yang diketahui.

Hasil Program :