Praktikum Kriptografi - Tugas 3

Nama : Muhammad Wildan Kamil
NPM : 140810220009 Selasa, 17 September 2024

---

1. Kumpulkan Exercise (dalam format pdf) tadi di Classroom

2. Buatlah program untuk enkripsi, dekripsi, dan mencari kunci Hill Cipher (bahasa pemrograman bebas)

3. Push program tersebut ke repository **NPM-Kripto24** dan sertakan juga screenshot di dalamnya.

4. Jelaskan program yang sudah dibuat di dalam 1 file pdf lalu kumpulkan di classroom

```
import numpy as np

# Fungsi untuk mengubah karakter menjadi indeks (0-25)
def char_to_index(char):
    return ord(char) - ord('A')

# Fungsi untuk mengubah indeks kembali menjadi karakter (A-Z)
def index_to_char(index):
    return chr(index + ord('A'))

# Fungsi Extended Euclidean Algorithm untuk mencari GCD dan invers modular
def extended_gcd(a, b):
    if a == 0:
        return b, 0, 1
    gcd, x1, y1 = extended_gcd(b % a, a)
    x = y1 - (b // a) * x1
    y = x1
    return gcd, x, y

# Fungsi untuk mencari invers modular
def find_mod_inverse(a, m):
    gcd, x, _ = extended_gcd(a, m)
    if gcd != 1:
        return None  # Tidak ada invers jika gcd(a, m) bukan 1
    return x % m

# Fungsi untuk memvalidasi determinan kunci
def validate_key(matrix):
    determinant = int(round(np.linalg.det(matrix))) % 26
    if determinant % 2 == 0 or determinant % 13 == 0:
        return False  # Validasi bahwa determinan tidak habis dibagi 2 atau 13
```

```
    return True

# Fungsi untuk menghitung invers modular dari matriks
def modular_inverse(matrix, mod):
    det = int(np.round(np.linalg.det(matrix))) % mod
    det_inv = find_mod_inverse(det, mod)
    if det_inv is None:
        return None
    adjugate = np.round(np.linalg.inv(matrix) * np.linalg.det(matrix)).astype(int)
% mod
    return (det_inv * adjugate) % mod

# Fungsi Hill Cipher (enkripsi dan dekripsi)
def hill_cipher(message, key_matrix, block_size, operation='encrypt'):
    determinant = int(round(np.linalg.det(key_matrix)))

    if operation == 'decrypt':
        mod_inv_det = find_mod_inverse(determinant % 26, 26)
        if mod_inv_det is None:
            print("Determinant tidak memiliki invers. Dekripsi gagal.")
            return
        inv_key_matrix = mod_inv_det * np.round(np.linalg.inv(key_matrix) *
determinant).astype(int) % 26
        key_matrix = inv_key_matrix  # Menggunakan invers matriks kunci untuk
dekripsi

    if len(message) % block_size != 0:
        message += message[-1] * (block_size - len(message) % block_size)  #
Padding jika perlu

    message_indices = [char_to_index(char) for char in message]
    message_matrix = np.array(message_indices).reshape(-1, block_size)

    result_matrix = np.dot(message_matrix, key_matrix) % 26
    result_message = ''.join([index_to_char(int(num)) for num in
result_matrix.flatten()])

    return result_message

# Fungsi untuk menemukan matriks kunci berdasarkan plaintext dan ciphertext
def find_key_matrix(plaintext, ciphertext):
    pt_matrix = np.array([char_to_index(c) for c in plaintext]).reshape(2, 2)
    ct_matrix = np.array([char_to_index(c) for c in ciphertext]).reshape(2, 2)

    print("Plaintext Matrix:\n", pt_matrix)
    print("Ciphertext Matrix:\n", ct_matrix)

    # Menghitung invers modular dari matriks plaintext
    pt_inv = modular_inverse(pt_matrix, 26)
    if pt_inv is None:
```

```
        print("Plaintext tidak dapat diinversi karena determinannya tidak memiliki
invers modular.")
        return None

    # Menghitung matriks kunci
    key_matrix = np.dot(ct_matrix, pt_inv) % 26
    return key_matrix

# Fungsi untuk menyiapkan teks (tanpa spasi, uppercase)
def prepare_text():
    text = input("Masukkan teks: ").replace(" ", "").upper()
    return text

# Fungsi menu utama
def main_menu():
    while True:
        print("\n═══ MENU UTAMA ═══")
        print("1. Enkripsi\n2. Dekripsi\n3. Temukan Kunci\n4. Keluar")
        print("═══════════════════")
        choice = input("Pilih opsi: ")

        if choice == '1' or choice == '2':
            block_size = 2  # Untuk Hill Cipher sederhana, kita gunakan blok 2×2
            key_data = list(map(int, input("Masukkan elemen matriks kunci (pisahkan
dengan spasi): ").split()))
            key_matrix = np.array(key_data).reshape(block_size, block_size) % 26

            message = prepare_text()
            result = hill_cipher(message, key_matrix, block_size,
operation='encrypt' if choice == '1' else 'decrypt')
            print("Hasil:", result)

        elif choice == '3':
            plaintext = prepare_text()
            ciphertext = prepare_text()
            key_matrix = find_key_matrix(plaintext[:4], ciphertext[:4])
            if key_matrix is not None:
                print("Matriks Kunci:\n", key_matrix)

        elif choice == '4':
            break

        else:
            print("Pilihan tidak valid.\n")

if __name__ == "__main__":
    main_menu()
```

Screenshot :

```
====== MENU UTAMA ======
1. Enkripsi
2. Dekripsi
3. Temukan Kunci
4. Keluar
========================
Pilih opsi: 1
Masukkan elemen matriks kunci (pisahkan dengan spasi): 7 2 6 5
Masukkan teks: PYTHON
Hasil: PUTVUP
```

```
====== MENU UTAMA ======
1. Enkripsi
2. Dekripsi
3. Temukan Kunci
4. Keluar
========================
Pilih opsi: 2
Masukkan elemen matriks kunci (pisahkan dengan spasi): 7 2 6 5
Masukkan teks: PUTVUP
Hasil: PYTHON
```

```
====== MENU UTAMA ======
1. Enkripsi
2. Dekripsi
3. Temukan Kunci
4. Keluar
========================
Pilih opsi: 3
Masukkan teks: TEST
Masukkan teks: IGHJ
Plaintext Matrix:
 [[19  4]
 [18 19]]
Ciphertext Matrix:
 [[8 6]
 [7 9]]
Matriks Kunci:
 [[ 6 10]
 [25 13]]
```