TUGAS 3

PRAKTIKUM KRIPTOGRAFI

DI SUSUN OLEH :

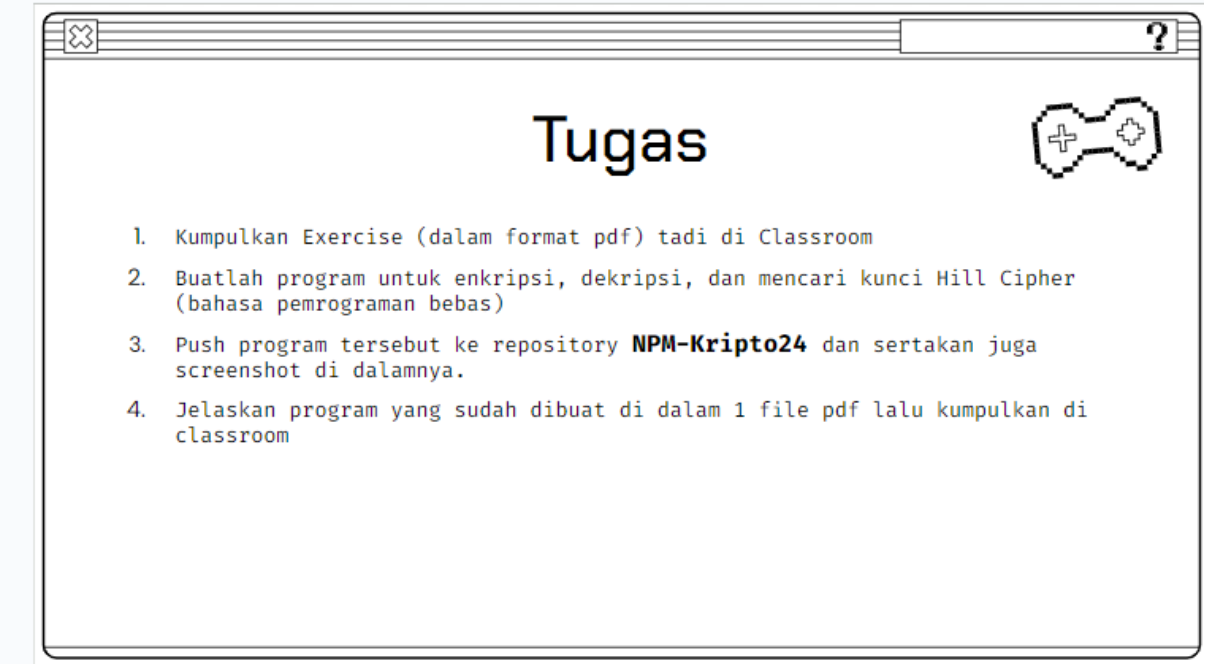
TABIN ADELIA RAFA – 140810220076

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PADJADJARAN

2024



Program nya

```
#   Nama     : Tabina adelia rafa
#   NPM      : 140810220076
#   Kelas    : B
#   Program  : Buatlah program untuk enkripsi, dekripsi, dan mencari kunci
Hill Cipher (bahasa pemrograman bebas)


import numpy as np

def char_to_number(x):
    return ord(x) - 65

def number_to_char(x):
    return chr(x + 65)

def mod_inverse(A, M):
    for X in range(1, M):
        if ((A % M) * (X % M)) % M == 1:
            return X
    return -1

def input_key(n):
    key = []
    print("Input angka/nilai key matrix (dipisahkan spasi):")
    for i in range(n):
        while True:
            row = list(map(int, input().split()))
            if len(row) == n:
                key.append([x % 26 for x in row])
```

```
                break
            else:
                print(f"Jumlah input pada baris ke-{i+1} tidak sesuai. Harus
{n} angka.")

    print("Key Matrix:")
    for row in key:
        print(f"[{' '.join(map(str, row))}]")

    return np.array(key)


def input_text(type_text):
    text = input(f"Input {type_text}: ").upper()
    return text

def hill(method, text, key, n):
    key_det = round(np.linalg.det(key))
    if key_det % 2 == 0 or key_det == 13:
        print("Determinan bukan ganjil selain 13. Key ga ada karena invers ga
ada.")
        return ""

    if len(text) % n != 0:
        last_char = text[-1]
        text += last_char * (n - len(text) % n)

    text_in_number = [char_to_number(c) for c in text]
    result = []

    if method == "dekripsi":
        det_inverse = mod_inverse(key_det % 26, 26)
        key = np.linalg.inv(key) * key_det
        key = (key * det_inverse).round().astype(int) % 26

    for i in range(0, len(text), n):
        block = np.dot(key, text_in_number[i:i + n]) % 26
        result.extend(block)

    output = ''.join(number_to_char(int(num)) for num in result)
    return output

def find_key(pt, ct, m):
    pt_in_number = [char_to_number(c) for c in pt]
    ct_in_number = [char_to_number(c) for c in ct]

    p_matrix = np.array(pt_in_number).reshape(m, m)
    c_matrix = np.array(ct_in_number).reshape(m, m)
```

```
    p_det = round(np.linalg.det(p_matrix))
    if p_det % 2 == 0 or p_det == 13:
        print("Determinan bukan ganjil selain 13. Key ga ada karena invers ga
ada.")
        return np.zeros((m, m), dtype=int)

    p_det_inverse = mod_inverse(p_det % 26, 26)
    p_matrix = np.linalg.inv(p_matrix) * p_det
    p_matrix = (p_matrix * p_det_inverse).round().astype(int) % 26

    key = np.dot(c_matrix, p_matrix) % 26
    return key

def main():
    while True:
        print("\n========Menu========")
        print("A. Enkripsi")
        print("B. Dekripsi")
        print("C. Cari Key")
        print("D. Exit")

        pilihan = input("Pilihan: ").upper()

        if pilihan in ['A', 'B']:
            n = int(input("\nInput ukuran key matrix (n x n): "))
            key = input_key(n)
            text = input_text("text")

            if pilihan == 'A':
                print("\nPlaintext:", text)
                output = hill("enkripsi", text, key, n)
                print("Ciphertext:", output)
            else:
                print("\nCiphertext:", text)
                output = hill("dekripsi", text, key, n)
                print("Plaintext:", output)

        elif pilihan == 'C':
            pt = input_text("plaintext")
            ct = input_text("ciphertext")
            m = int(input("\nInput nilai m: "))
            key = find_key(pt, ct, m)
            print(f"\nPlaintext: {pt}")
            print(f"Ciphertext: {ct}")
            print("Key Matrix:")
            for row in key:
                print(f"[{' '.join(map(str, row))}]")
```

```
        elif pilihan == 'D':
            break
        else:
            print("\nInput ga sesuai.")

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

Outputnya

```
========Menu========
A. Enkripsi
B. Dekripsi
C. Cari Key
D. Exit
Pilihan: B

Input ukuran key matrix (n x n): 2
Input angka/nilai key matrix (dipisahkan spasi):
2 2 3 4
Jumlah input pada baris ke-1 tidak sesuai. Harus 2 angka.
2 3
3 4
Key Matrix:
[2 3]
[3 4]
Input text: HELLO

Ciphertext: HELLO
Plaintext: KNPLMO

========Menu========
A. Enkripsi
B. Dekripsi
C. Cari Key
D. Exit
Pilihan: A

Input ukuran key matrix (n x n): 2
Input angka/nilai key matrix (dipisahkan spasi):
2 3
5 6
Key Matrix:
[2 3]
[5 6]
Input text: HELLO

Plaintext: HELLO
Ciphertext: AHDRSY
```

```
========Menu========
A. Enkripsi
B. Dekripsi
C. Cari Key
D. Exit
Pilihan: C
Input plaintext: HELLO
Input ciphertext: AHDRSY

Input nilai m: 2
Traceback (most recent call last):
```

Penjelsan programnya

1. Import Library

```
python
Salin kode
import numpy as np
```

Program ini menggunakan **NumPy**, sebuah library Python yang digunakan untuk operasi matriks dan vektor.

2. Fungsi `char_to_number(x)` dan `number_to_char(x)`

```
python
Salin kode
def char_to_number(x):
    return ord(x) - 65

def number_to_char(x):
    return chr(x + 65)
```

Fungsi ini mengonversi huruf menjadi angka dan sebaliknya:

- **`char_to_number(x)`**: Mengonversi karakter menjadi angka dengan basis ASCII. Misalnya, `'A'` akan dikonversi menjadi `0`, `'B'` menjadi `1`, dan seterusnya.
- **`number_to_char(x)`**: Mengonversi angka menjadi karakter. Misalnya, `0` akan dikonversi menjadi `'A'`, `1` menjadi `'B'`, dan seterusnya.

3. Fungsi `mod_inverse(A, M)`

```
python
Salin kode
def mod_inverse(A, M):
    for X in range(1, M):
        if ((A % M) * (X % M)) % M == 1:
            return X
    return -1
```

Fungsi ini digunakan untuk mencari **invers modulo** dari determinan matriks terhadap 26. Invers modulo diperlukan ketika Anda melakukan dekripsi atau mencari kunci. Algoritma ini mencoba mencari `X` sehingga $(A \times X) \mod M = 1$.

4. Fungsi `input_key(n)`

```
python
Salin kode
def input_key(n):
    key = []
    print("Input angka/nilai key matrix (dipisahkan spasi):")
    for i in range(n):
        while True:
```

```
            row = list(map(int, input().split()))
            if len(row) == n:
                key.append([x % 26 for x in row])
                break
            else:
                print(f"Jumlah input pada baris ke-{i+1} tidak
sesuai. Harus {n} angka.")

    # Tampilkan matriks kunci dengan rapi
    print("Key Matrix:")
    for row in key:
        print(f"[{' '.join(map(str, row))}]")

    return np.array(key)
```

Fungsi ini meminta pengguna untuk memasukkan **key matrix**:

- Pengguna harus memasukkan elemen-elemen matriks sebesar n×nn \times nn×n, dengan setiap elemen dipisahkan oleh spasi.
- Jika jumlah elemen yang dimasukkan kurang dari nnn, program akan meminta input ulang.
- Setiap elemen key dikonversi ke modulo 26 untuk memastikan elemen tetap dalam rentang yang sesuai.

Matriks kunci ditampilkan dengan format yang rapi dan dikembalikan dalam bentuk **NumPy array**.

5. Fungsi `input_text(type_text)`

```
python
Salin kode
def input_text(type_text):
    text = input(f"Input {type_text}: ").upper()
    return text
```

Fungsi ini meminta input berupa teks (plaintext atau ciphertext) dan mengonversinya menjadi huruf kapital. Ini memastikan bahwa teks yang digunakan adalah huruf-huruf alfabet dalam format yang konsisten.

6. Fungsi `hill(method, text, key, n)`

```
python
Salin kode
def hill(method, text, key, n):
    key_det = round(np.linalg.det(key))
    if key_det % 2 == 0 atau key_det == 13:
        print("Determinan bukan ganjil selain 13. Key ga ada karena
invers ga ada.")
        return ""

    if len(text) % n != 0:
        last_char = text[-1]
```

```
        text += last_char * (n - len(text) % n)

    text_in_number = [char_to_number(c) for c in text]
    result = []

    if method == "dekripsi":
        det_inverse = mod_inverse(key_det % 26, 26)
        key = np.linalg.inv(key) * key_det
        key = (key * det_inverse).round().astype(int) % 26

    for i in range(0, len(text), n):
        block = np.dot(key, text_in_number[i:i + n]) % 26
        result.extend(block)

    output = ''.join(number_to_char(int(num)) for num in result)
    return output
```

Fungsi ini merupakan implementasi Hill Cipher untuk **enkripsi** dan **dekripsi**:

- **Key Determinant**: Program memeriksa apakah determinan dari matriks kunci **bukan** bilangan genap atau 13, karena untuk mendekripsi diperlukan invers matriks, yang tidak ada jika determinan bernilai genap atau 13.
- **Padding**: Jika panjang teks tidak dapat dibagi rata ke dalam blok ukuran n×nn \times nn×n, karakter terakhir dari teks akan diulang untuk memastikan ukuran teks sesuai.
- **Enkripsi**: Program mengubah teks menjadi angka (dengan `char_to_number`), lalu mengalikan setiap blok dengan matriks kunci untuk menghasilkan ciphertext.
- **Dekripsi**: Program menghitung invers matriks dari key (dengan bantuan `mod_inverse`), lalu menggunakannya untuk mengubah ciphertext menjadi plaintext.

7. Fungsi `find_key(pt, ct, m)`

```
python
Salin kode
def find_key(pt, ct, m):
    pt_in_number = [char_to_number(c) for c in pt]
    ct_in_number = [char_to_number(c) for c in ct]

    p_matrix = np.array(pt_in_number).reshape(m, m)
    c_matrix = np.array(ct_in_number).reshape(m, m)

    p_det = round(np.linalg.det(p_matrix))
    if p_det % 2 == 0 or p_det == 13:
        print("Determinan bukan ganjil selain 13. Key ga ada karena
invers ga ada.")
        return np.zeros((m, m), dtype=int)

    p_det_inverse = mod_inverse(p_det % 26, 26)
    p_matrix = np.linalg.inv(p_matrix) * p_det
    p_matrix = (p_matrix * p_det_inverse).round().astype(int) % 26

    key = np.dot(c_matrix, p_matrix) % 26
    return key
```

Fungsi ini digunakan untuk **mencari kunci Hill Cipher** berdasarkan plaintext dan ciphertext:

- **Plaintext dan Ciphertext diubah menjadi Matriks**: Plaintext dan ciphertext diubah menjadi matriks m×mm \times mm×m.
- **Mencari Invers Matriks Plaintext**: Program menghitung determinan dari plaintext matrix, lalu mencari invers modulo untuk digunakan dalam mencari kunci.
- **Menghitung Matriks Kunci**: Setelah mendapatkan invers dari plaintext, kunci ditemukan dengan mengalikan ciphertext matrix dengan invers plaintext.

8. Fungsi `main()`

```
python
Salin kode
def main():
    while True:
        print("\n========Menu========")
        print("A. Enkripsi")
        print("B. Dekripsi")
        print("C. Cari Key")
        print("D. Exit")

        pilihan = input("Pilihan: ").upper()

        if pilihan in ['A', 'B']:
            n = int(input("\nInput ukuran key matrix (n x n): "))
            key = input_key(n)
            text = input_text("text")

            if pilihan == 'A':
                print("\nPlaintext:", text)
                output = hill("enkripsi", text, key, n)
                print("Ciphertext:", output)
            else:
                print("\nCiphertext:", text)
                output = hill("dekripsi", text, key, n)
                print("Plaintext:", output)

        elif pilihan == 'C':
            pt = input_text("plaintext")
            ct = input_text("ciphertext")
            m = int(input("\nInput nilai m: "))
            key = find_key(pt, ct, m)
            print(f"\nPlaintext: {pt}")
            print(f"Ciphertext: {ct}")
            print("Key Matrix:")
            for row in key:
                print(f"[{' '.join(map(str, row))}]")

        elif pilihan == 'D':
            break
        else:
            print("\nInput ga sesuai.")
```

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

Fungsi **main** ini adalah inti dari program:

- Menyediakan menu untuk **enkripsi**, **dekripsi**, atau **mencari kunci**.
- Mengambil input yang sesuai dari pengguna untuk ukuran matriks kunci, plaintext, ciphertext, dan metode yang dipilih.
- Menggunakan fungsi-fungsi yang dijelaskan sebelumnya untuk melakukan operasi yang diinginkan (enkripsi, dekripsi, atau mencari kunci).