LAPORAN PRAKTIKUM
KEAMANAN SISTEM INFORMASI DAN JARINGAN

Disusun Oleh :

Nama	: Fahmi Adi Setiawan
NIM	: 22230010
Mata Kuliah	: Keamanan Sistem Informasi dan Jaringan

Program Studi Sistem Informasi

Fakultas Sains dan Teknologi

Universitas Respati Yogyakarta

2025/2026

Kodingan dan hasil Running

[1]:
```
import numpy as np

# Fungsi untuk melakukan invers matriks modulo 26
def matrix_inverse(matrix, mod=26):
    det = int(np.round(np.linalg.det(matrix)))  # Determinan matriks
    det_inv = mod_inverse(det, mod)  # Mencari invers determinan

    # Matriks kofaktor
    cofactor_matrix = np.round(det * np.linalg.inv(matrix)).astype(int) % mod

    # Matriks adjugate
    adjugate_matrix = cofactor_matrix.T % mod

    # Matriks inverse
    inverse_matrix = (det_inv * adjugate_matrix) % mod
    return inverse_matrix

# Fungsi untuk mencari inverse dari suatu bilangan modulo
def mod_inverse(a, m):
    for x in range(1, m):
        if (a * x) % m == 1:
            return x
    return -1

# Fungsi untuk mengubah teks menjadi angka
def text_to_numbers(text):
    text = text.upper()
```

```
def text_to_numbers(text):
    text = text.upper()
    numbers = [ord(char) - 65 for char in text if char.isalpha()]
    return numbers

# Fungsi untuk mengubah angka menjadi teks
def numbers_to_text(numbers):
    text = ''.join([chr(num + 65) for num in numbers])
    return text

# Fungsi untuk mengenkripsi plaintext menggunakan matriks kunci
def encrypt(plaintext, key_matrix):
    n = len(key_matrix)
    plaintext_numbers = text_to_numbers(plaintext)

    # Membagi plaintext menjadi blok ukuran n
    blocks = [plaintext_numbers[i:i + n] for i in range(0, len(plaintext_numbers), n)]

    # Enkripsi blok menggunakan matriks kunci
    ciphertext_numbers = []
    for block in blocks:
        if len(block) < n:
            # Menambahkan padding untuk blok yang kurang
            block += [0] * (n - len(block))

        block_matrix = np.array(block).reshape(-1, 1)
        encrypted_block = np.dot(key_matrix, block_matrix) % 26
        ciphertext_numbers.extend(encrypted_block.flatten().astype(int))
```

```
    ciphertext = numbers_to_text(ciphertext_numbers)
    return ciphertext

# Fungsi untuk mendekripsi ciphertext menggunakan matriks kunci
def decrypt(ciphertext, key_matrix):
    n = len(key_matrix)
    ciphertext_numbers = text_to_numbers(ciphertext)

    # Membagi ciphertext menjadi blok ukuran n
    blocks = [ciphertext_numbers[i:i + n] for i in range(0, len(ciphertext_numbers), n)]

    # Mencari inverse matriks kunci
    inverse_key_matrix = matrix_inverse(key_matrix)

    # Dekripsi blok menggunakan matriks kunci inverse
    plaintext_numbers = []
    for block in blocks:
        block_matrix = np.array(block).reshape(-1, 1)
        decrypted_block = np.dot(inverse_key_matrix, block_matrix) % 26
        plaintext_numbers.extend(decrypted_block.flatten().astype(int))

    plaintext = numbers_to_text(plaintext_numbers)
    return plaintext

# Contoh Matriks Kunci 2x2
key_matrix = np.array([[6, 24], [1, 16]])
```

```
# Contoh Matriks Kunci 2x2
key_matrix = np.array([[6, 24], [1, 16]])

# Teks yang akan dienkripsi
plaintext = "HELLO"

# Enkripsi
ciphertext = encrypt(plaintext, key_matrix)
print("Ciphertext:", ciphertext)

# Dekripsi
decrypted_text = decrypt(ciphertext, key_matrix)
print("Decrypted Text:", decrypted_text)
```

```
Ciphertext: ITSFGO
Decrypted Text: VADMWI
```