LAPORAN ANALISIS CIPHER KLASIK

Laporan ini disusun untuk memenuhi nilai Tugas Besar 1 dengan dosen pengampu

Bapak Kodrat Mahatma S.T.,M.Kom.

Sindi Rahmadani (20123040)

C1.23 S1 INFORMATIKA

UNIVERSITAS TEKNOLOGI DIGITAL

2025

1. Caesar Cipher
    a. Teori Caesar cipher

        Caesar Cipher merupakan salah satu algoritma kriptografi klasik tertua yang ditemukan oleh Julius Caesar. Metode ini mengenkripsi teks dengan cara menggeser setiap huruf pada plaintext sejauh nilai tertentu (shift).

        Misalnya, jika pergeseran bernilai 3, maka huruf “A” akan menjadi “D”, “B” menjadi “E”, dan seterusnya.

        Secara matematis, proses enkripsi dapat ditulis sebagai:

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

        dan dekripsi sebagai:

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

        dengan:

        - P: huruf pada plaintext
        - C: huruf pada ciphertext
        - k: jumlah pergeseran (key)

        Pada implementasi program, huruf-huruf diubah menjadi nilai numerik (A=0, B=1, dst) untuk memudahkan perhitungan modulo 26.

    b. Implementasi Program

        Program Caesar Cipher diimplementasikan menggunakan bahasa Python.

        Fungsi utama yang dibuat yaitu caesar_encrypt() dan caesar_decrypt().

        Fungsi enkripsi bekerja dengan menggeser setiap huruf berdasarkan nilai shift, sedangkan dekripsi melakukan proses kebalikannya.

```python
def caesar_encrypt(plaintext, shift):
    result = ""
    for char in plaintext.upper():
        if char.isalpha():
            result += chr((ord(char) - 65 + shift) % 26 + 65)
        else:
            result += char
    return result

def caesar_decrypt(ciphertext, shift):
    result = ""
    for char in ciphertext.upper():
        if char.isalpha():
            result += chr((ord(char) - 65 - shift) % 26 + 65)
        else:
            result += char
    return result
```

Kode di atas mengubah setiap karakter huruf menjadi huruf baru yang sudah digeser sejumlah nilai shift.

c. Input – Output Program

- Input

```
PS D:\kkriptografi\TugasBesar1> & C:/Users/DELL/AppData/Loca
e d:/kkriptografi/TugasBesar1/Caesar.py
=== Program Caesar Cipher ===
Masukkan teks yang ingin dienkripsi: SINDI CANTIK
Masukkan jumlah pergeseran: 3
```

- Output

```
Hasil Enkripsi : VLQGL FDQWLN
Hasil Dekripsi : SINDI CANTIK
```

Hasil menunjukkan bahwa setiap huruf bergeser sejauh 3 posisi. Misalnya, S menjadi V, I menjadi L, dan seterusnya.Program dapat mengembalikan teks asli dengan menggunakan proses dekripsi menggunakan shift yang sama.

d. Analisis Kelemahn

Berdasarkan hasil implementasi program Caesar Cipher, ditemukan bahwa algoritma ini memiliki beberapa kelemahan yang signifikan dari sisi keamanan data. Berikut analisis kelemahannya:

a) Ruang kunci sangat kecil — hanya memiliki 25 kemungkinan pergeseran, sehingga mudah dipecahkan dengan brute-force.

b) Pola huruf tetap terlihat — frekuensi huruf pada ciphertext masih sama seperti plaintext, sehingga dapat dianalisis secara statistik.
c) Tidak menyamarkan pola berulang — kata atau huruf yang muncul berulang tetap terlihat berurutan setelah dienkripsi.
d) Tidak mendukung karakter non-huruf — angka, spasi, dan simbol tidak terenkripsi, sehingga pesan masih mudah dibaca sebagian.
e) Struktur enkripsi sederhana — hanya menggunakan operasi pergeseran linear tanpa variasi, membuatnya mudah ditebak.

Secara keseluruhan, Caesar Cipher tergolong algoritma yang sangat lemah karena struktur dan pola pergeserannya terlalu sederhana.Meskipun dapat berfungsi sebagai media pembelajaran untuk memahami dasar enkripsi, namun secara praktis algoritma ini tidak layak digunakan untuk melindungi informasi sensitif..

2. Vigenere Cipher

a. Teori Caesar cipher

Vigenère Cipher merupakan pengembangan dari Caesar Cipher yang menggunakan kata kunci (key) untuk menentukan nilai pergeseran setiap huruf pada plaintext. Setiap huruf pada kunci menentukan besar pergeseran berdasarkan posisinya dalam alfabet (A=0, B=1, C=2, dst).

Dengan begitu, huruf yang sama pada plaintext dapat dienkripsi menjadi huruf yang berbeda tergantung posisi huruf kunci.

Secara matematis:

$$Ci=(Pi+Ki)\text{mod}26 \quad Pi=(Ci-Ki)\text{mod } 26$$

$$P_i = (C_i - K_i) \mod 26 Pi=(Ci−Ki)\text{mod}26$$

Dengan:

- Pi: huruf plaintext ke-i
- Ci : huruf ciphertext ke-i
- Ki: huruf kunci ke-i (diulang sesuai panjang plaintext)

Vigenère Cipher dianggap lebih kuat daripada Caesar karena setiap huruf memiliki pergeseran berbeda, tergantung huruf kuncinya.

b. Implementasi Program

Program ini mengenkripsi teks dengan menggeser setiap huruf berdasarkan nilai huruf pada kunci yang diulang sepanjang pesan. Hanya huruf alfabet yang diproses, sementara spasi dan tanda baca dibiarkan. Fungsi dekripsi melakukan

operasi sebaliknya untuk mengembalikan ciphertext menjadi plaintext. Program ini menunjukkan prinsip dasar substitusi polialfabetik, di mana setiap huruf memiliki pergeseran berbeda sesuai kunci.

```python
def vigenere_encrypt(plaintext, key):
    plaintext = plaintext.upper()
    key = key.upper()
    result = ""
    j = 0
    for char in plaintext:
        if char.isalpha():
            shift = ord(key[j % len(key)]) - 65
            result += chr((ord(char) - 65 + shift) % 26 + 65)
            j += 1
        else:
            result += char
    return result

def vigenere_decrypt(ciphertext, key):
    ciphertext = ciphertext.upper()
    key = key.upper()
    result = ""
    j = 0
    for char in ciphertext:
        if char.isalpha():
            shift = ord(key[j % len(key)]) - 65
            result += chr((ord(char) - 65 - shift) % 26 + 65)
            j += 1
        else:
            result += char
    return result
```

c. Input – Output

- Input

```
e d:/kkriptografi/TugasBesar1/Vigenere.py
=== Program Vigenère Cipher ===
Masukkan teks yang ingin dienkripsi: SINDI CANTIK
Masukkan kunci (key): LUCY
```

- Output

```
Hasil Enkripsi : DCPBT WCLECM
Hasil Dekripsi : SINDI CANTIK
```

Dalam contoh di atas, kunci "LUCY" diulang untuk menyesuaikan panjang plaintext.Setiap huruf plaintext bergeser sesuai nilai huruf pada kunci. Misalnya, huruf “S” bergeser sejauh “L” (11 posisi).

d. Analisis Kelemahan

Meskipun lebih kuat dari Caesar Cipher, algoritma ini masih memiliki kelemahan utama. **Pertama**, kunci yang pendek akan berulang, sehingga pola huruf dalam ciphertext tetap bisa dianalisis. **Kedua**, cipher ini masih rentan terhadap analisis frekuensi jika panjang kunci diketahui.**Selain itu**, program hanya mengenkripsi huruf alfabet, sehingga karakter lain tetap terbaca.

3. Affine Cipher

a. Teori Caesar cipher

Affine Cipher adalah algoritma substitusi huruf tunggal yang menggabungkan pergeseran (shift) dan perkalian (multiplicative cipher). Setiap huruf plaintext diubah menjadi angka (A=0, B=1, ..., Z=25), lalu dihitung menggunakan rumus:

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

Untuk dekripsi, digunakan rumus kebalikannya:

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

di mana a dan 26 harus relatif prima agar kunci memiliki invers.

b. Implementasi Program

```python
def affine_encrypt(plaintext, a, b):
    plaintext = plaintext.upper()
    result = ""
    for char in plaintext:
        if char.isalpha():
            x = ord(char) - 65
            result += chr(((a * x + b) % 26) + 65)
        else:
            result += char
    return result

# Fungsi dekripsi
def affine_decrypt(ciphertext, a, b):
    ciphertext = ciphertext.upper()
    result = ""
    a_inv = pow(a, -1, 26)  # invers modular a (mod 26)
    for char in ciphertext:
        if char.isalpha():
            y = ord(char) - 65
            result += chr(((a_inv * (y - b)) % 26) + 65)
        else:
            result += char
    return result
```

c. Input – Output Program

- Input

```
Masukkan teks yang ingin dienkripsi: SINDI CANTIK
Masukkan nilai a (harus koprima dengan 26, contoh 5): 5
Masukkan nilai b (contoh 8): 8
```

- Output

```
Hasil Enkripsi : UWVXW SIVZWG
Hasil Dekripsi : SINDI CANTIK
```

d. Analisis Kelemahna

Affine Cipher memiliki beberapa kelemahan penting.**Pertama**, ruang kunci kecil, sehingga mudah dipecahkan dengan brute-force. **Kedua**, pola huruf masih dapat dianalisis secara frekuensi, karena setiap huruf plaintext selalu dipetakan ke satu huruf ciphertext yang sama.

Selain itu, jika nilai a tidak relatif prima terhadap 26, dekripsi tidak dapat dilakukan karena tidak memiliki invers.

4. Playfair Cipher

a. Teori Caesar cipher

Playfair Cipher adalah algoritma substitusi digraf, yaitu mengenkripsi dua huruf sekaligus. Cipher ini menggunakan tabel 5×5 berisi huruf-huruf alfabet berdasarkan kunci yang diberikan. Huruf "J" biasanya digabung dengan "I" agar cukup 25 huruf.

Aturan enkripsi:

- Jika sepasang huruf sama, sisipkan huruf "X" di antara keduanya.
- Jika kedua huruf berada pada baris yang sama, diganti dengan huruf di sebelah kanannya.
- Jika berada pada kolom yang sama, diganti dengan huruf di bawahnya.
- Jika membentuk persegi, maka masing-masing huruf diganti dengan huruf pada sudut yang sama dalam persegi tersebut.

Dekripsi dilakukan dengan aturan kebalikannya.

b. Implementasi Program

```
def generate_playfair_matrix(key):
    key = key.upper().replace("J", "I")
    alphabet = "ABCDEFGHIKLMNOPQRSTUVWXYZ"
    matrix = []
    used = set()
    for c in key + alphabet:
        if c not in used:
            matrix.append(c)
            used.add(c)
    return [matrix[i:i+5] for i in range(0, 25, 5)]

def find_position(matrix, letter):
    for r in range(5):
        for c in range(5):
            if matrix[r][c] == letter:
                return r, c
    return None

def playfair_encrypt(plaintext, key):
    plaintext = plaintext.upper().replace("J", "I").replace(" ", "")
    matrix = generate_playfair_matrix(key)
```

```
# buat pasangan huruf
pairs = []
i = 0
while i < len(plaintext):
    a = plaintext[i]
    b = plaintext[i + 1] if i + 1 < len(plaintext) else 'X'
    if a == b:
        pairs.append((a, 'X'))
        i += 1
    else:
        pairs.append((a, b))
        i += 2

# enkripsi
ciphertext = ""
for a, b in pairs:
    ra, ca = find_position(matrix, a)
    rb, cb = find_position(matrix, b)
    if ra == rb:
        ciphertext += matrix[ra][(ca + 1) % 5]
        ciphertext += matrix[rb][(cb + 1) % 5]
    elif ca == cb:
        ciphertext += matrix[(ra + 1) % 5][ca]
        ciphertext += matrix[(rb + 1) % 5][cb]
    else:
        ciphertext += matrix[ra][cb]
        ciphertext += matrix[rb][ca]
return ciphertext
```

```
def playfair_decrypt(ciphertext, key):
    ciphertext = ciphertext.upper()
    matrix = generate_playfair_matrix(key)
    plaintext = ""

    for i in range(0, len(ciphertext), 2):
        a, b = ciphertext[i], ciphertext[i + 1]
        ra, ca = find_position(matrix, a)
        rb, cb = find_position(matrix, b)
        if ra == rb:
            plaintext += matrix[ra][(ca - 1) % 5]
            plaintext += matrix[rb][(cb - 1) % 5]
        elif ca == cb:
            plaintext += matrix[(ra - 1) % 5][ca]
            plaintext += matrix[(rb - 1) % 5][cb]
        else:
            plaintext += matrix[ra][cb]
            plaintext += matrix[rb][ca]
    return plaintext
```

c. Input – Output Program

- Input

```
=== Program Playfair Cipher ===
Masukkan teks yang ingin dienkripsi: SINDI CANTIK
Masukkan kunci (key): PLAYFAIR
```

- Output

```
Hasil Enkripsi : NCTIRDPQNDSY
Hasil Dekripsi : SINDICANTIKX
```

d. Analisis Kelemahna

Meskipun lebih aman dari Caesar dan Affine, Playfair Cipher masih memiliki kelemahan.**Pertama**, frekuensi pasangan huruf (digraf) tetap dapat dianalisis jika teks cukup panjang.**Kedua**, cipher ini tidak menyamarkan pola posisi huruf sepenuhnya, karena struktur tabel 5×5 membuat hubungan antarhuruf tetap terbatas.

Selain itu, huruf “I” dan “J” dianggap sama, yang dapat menimbulkan ambiguitas pada pesan tertentu.

5. Hill Cipher

a. Teori Caesar cipher

Hill Cipher merupakan algoritma kriptografi klasik berbasis aljabar linear yang menggunakan matriks kunci untuk mengenkripsi teks.Plaintext diubah menjadi vektor angka (A=0 sampai Z=25), kemudian dikalikan dengan matriks kunci dan hasilnya diambil modulo 26:

$$C=(K\times P)\bmod 26$$

Untuk dekripsi digunakan invers dari matriks kunci:

$$P=(K-1\times C)\bmod 26$$

Cipher ini mengenkripsi blok huruf sekaligus (biasanya 2 atau 3 huruf), sehingga lebih kuat dibanding substitusi tunggal.

b. Implementasi program

```python
import numpy as np
# Fungsi untuk konversi huruf ke angka (A=0, B=1, ... Z=25)
def text_to_numbers(text):
    text = text.upper().replace(" ", "")
    return [ord(c) - 65 for c in text]
# Fungsi untuk konversi angka ke huruf
def numbers_to_text(numbers):
    return ''.join([chr((num % 26) + 65) for num in numbers])
# Fungsi enkripsi Hill Cipher
def hill_encrypt(plaintext, key_matrix):
    plaintext = plaintext.upper().replace(" ", "")
    if len(plaintext) % 2 != 0:
        plaintext += "X"  # tambahkan X jika panjang ganjil
    encrypted_numbers = []
    for i in range(0, len(plaintext), 2):
        pair = np.array(text_to_numbers(plaintext[i:i+2]))
        result = np.dot(key_matrix, pair) % 26
        encrypted_numbers.extend(result)
    return numbers_to_text(encrypted_numbers)
# Fungsi dekripsi Hill Cipher
def hill_decrypt(ciphertext, key_matrix):
    det = int(np.round(np.linalg.det(key_matrix)))
    det_inv = None
    # cari invers determinan mod 26
    for i in range(26):
        if (det * i) % 26 == 1:
            det_inv = i
            break
```

```python
    if det_inv is None:
        raise ValueError("Matriks kunci tidak memiliki invers mod 26, ga
    # matriks adjoin
    adjugate = np.array([[key_matrix[1][1], -key_matrix[0][1]],
                         [-key_matrix[1][0], key_matrix[0][0]]])
    inverse_matrix = (det_inv * adjugate) % 26

    decrypted_numbers = []
    for i in range(0, len(ciphertext), 2):
        pair = np.array(text_to_numbers(ciphertext[i:i+2]))
        result = np.dot(inverse_matrix, pair) % 26
        decrypted_numbers.extend(result)
    return numbers_to_text(decrypted_numbers)
```

c. Input – Output Program

- Input

```
=== Program Hill Cipher (2x2) ===
Masukkan teks yang ingin dienkripsi: SINDI

Masukkan elemen matriks kunci 2x2 (angka 0–25):
a11: 3
a12: 3
a21: 2
a22: 5
```

- Output

```
Hasil Enkripsi : AYWPPB
Hasil Dekripsi : SINDIX
```

e. Analisis Kelemahna

Hill Cipher memiliki keamanan lebih tinggi dibanding cipher klasik lain karena mengenkripsi blok huruf sekaligus. Namun, terdapat beberapa kelemahan:

- Kunci harus memiliki determinan yang relatif prima terhadap 26, jika tidak maka matriks tidak dapat diinvers dan dekripsi gagal.
- Rentan terhadap serangan known-plaintext, karena jika penyerang mengetahui beberapa pasangan plaintext–ciphertext, kunci dapat dihitung dengan mudah menggunakan aljabar linear.
- Sulit diterapkan tanpa komputer, karena membutuhkan operasi matriks dan invers modulo yang cukup kompleks.