

LAPORAN IMPLEMENTASI PROGRAM KRIPTOGRAFI KLASIK

KELOMPOK 11

Anggota:

- Rizky Rifaldi Wahab
- Thia Shopy Latifah

1. Pendahuluan

Kriptografi merupakan ilmu yang mempelajari teknik penyandian pesan agar isi pesan hanya dapat dipahami oleh pihak yang berhak. Dalam konteks klasik, metode yang digunakan biasanya berbasis pada operasi substitusi dan transposisi karakter.

Pada laporan ini, kelompok 11 mengimplementasikan lima algoritma cipher klasik, yaitu **Caesar Cipher**, **Vigenère Cipher**, **Affine Cipher**, **Playfair Cipher**, dan **Hill Cipher**, serta dua tugas mini tambahan yang berfokus pada **otomatisasi enkripsi-dekripsi Caesar Cipher** dan **analisis frekuensi Vigenère Cipher**.

2. Implementasi Algoritma Kriptografi Klasik

2.1 Caesar Cipher

Caesar Cipher merupakan metode substitusi sederhana di mana setiap huruf dalam plaintext digeser sejauh k posisi dalam alfabet.

Contoh: jika $k = 3$, maka $A \rightarrow D$, $B \rightarrow E$, $C \rightarrow F$, dan seterusnya.

```
PS D:\Kampus Doc\Season 5\Kriptografi\tugas1> python caesar_cipher.py
Original: Hello World!
Encrypted: Khoo Zruog!
```

Analisis Kelemahan

- Caesar Cipher sangat mudah dipecahkan karena hanya memiliki 25 kemungkinan kunci.
- Serangan brute force dapat menemukan plaintext dalam hitungan detik.

2.2 Vigenere Cipher

Vigenère Cipher menggunakan kunci berupa kata atau frasa untuk melakukan pergeseran huruf secara bergantian. Setiap huruf plaintext digeser berdasarkan nilai huruf kunci.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\Kampus Doc\Season 5\Kriptografi\tugas1> python vigenere.py
[!] Enter your message: Chelsea forever
[+] Plaintext: Chelsea forever
[+] Ciphertext: Mlcvwck dyvcfip

[?] Do you want to decrypt the message?
[?] Y or N: Y
[+] Decrypted text: Chelsea forever
```

Analisis Kelemahan

- Jika kunci terlalu pendek, pola huruf ciphertext dapat terdeteksi dengan analisis frekuensi.
- Serangan Kasiski dan Friedman Test dapat digunakan untuk menebak panjang kunci.

2.3 Affine Cipher

Affine Cipher merupakan cipher substitusi yang memanfaatkan operasi linear:

$$E(x) = (a \times x + b) \bmod 26$$

dengan a dan 26 harus saling relatif prima.

```
PS D:\Kampus Doc\Season 5\Kriptografi\tugas1> python affine.py
[?] Enter text to encrypt: Manchester is Red
[+] Plaintext: Manchester is Red
[+] Encrypted Text: Uanchester is Jed
```

Analisis Kelemahan

- Mudah diserang dengan metode *known plaintext attack* karena hanya memiliki jumlah kunci yang terbatas.
- Jika sepasang plaintext-ciphertext diketahui, nilai a dan b dapat dihitung dengan cepat.

2.4 Playfair Cipher

Playfair Cipher bekerja dengan mengganti pasangan huruf (digraph) menggunakan matriks 5x5 berdasarkan kata kunci tertentu. Cipher ini lebih kuat dibanding substitusi tunggal karena mengenkripsi dua huruf sekaligus.

```
PS D:\Kampus Doc\Season 5\Kriptografi\tugas1> python playfair.py

=== Playfair Cipher ===
Plaintext : INSTRUMENTS
Encrypted  : GATLMZCLRQXA
Decrypted  : INSTRUMENTSX
```

KEY: monarchy

Analisis Kelemahan

- Masih rentan terhadap analisis frekuensi digraph.
- Pola pengulangan huruf bisa terdeteksi pada ciphertext panjang.

2.5 Hill Cipher

Hill Cipher memanfaatkan aljabar linear dengan representasi huruf sebagai vektor, kemudian dikalikan dengan matriks kunci mod 26.

```
PS D:\Kampus Doc\Season 5\Kriptografi\tugas1> python hill.py

=== Hill Cipher ===
Plaintext : HELLO
Encrypted  : HIOZHN
Decrypted  : HELLOX

key_matrix = np.array([[3, 3], [2, 5]])
```

Analisis Kelemahan

- Kelemahannya muncul jika matriks kunci tidak memiliki invers modulo 26.
- Cipher ini tidak aman terhadap serangan linear algebra dengan cukup banyak pasangan plaintext-ciphertext.

3. Tugas Mini

3.1 Tugas mini 1 - Implementasi Caesar Cipher

Program Caesar Cipher ini mampu melakukan enkripsi dan dekripsi otomatis, serta menyimpan hasil ke dalam file .txt.

```
def caesar_encrypt(text, shift):
    result = ""
```

```

for char in text.upper():
    if char.isalpha():
        result += chr((ord(char) - 65 + shift) % 26 + 65)
    else:
        result += char
return result

def caesar_decrypt(text, shift):
    return caesar_encrypt(text, -shift)

def main():
    text = input("Masukan text: ")
    shift = int(input("masukan shift: "))
    enc = caesar_encrypt(text, shift)
    dec = caesar_decrypt(enc, shift)

    print("\n === HASIL ===")
    print(f"Teks Asli      : {text.upper()}")
    print(f"Hasil Enkripsi   : {enc}")
    print(f"Hasil Deskripsi  : {dec}")

    filename = "hasil_caesar_cipher.txt"
    with open(filename, "w") as f:
        f.write("=== HASIL PROGRAM CAESAR CIPHER ===\n")
        f.write(f"Teks Asli      : {text}\n")
        f.write(f"Hasil Enkripsi : {enc}\n")
        f.write(f"Hasil Deskripsi: {dec}\n")
        f.write(f"Shift          : {shift}\n")

    print(f"\n✅ Hasil telah disimpan ke file: {filename}")

if __name__ == "__main__":
    main()

```

Hasil Output:

```

PS D:\Kampus Doc\Season 5\Kriptografi\Tugas mini> python cipher.py
Masukan text: London is Blue, Manchester is Red
masukan shift: 5

=== HASIL ===
Teks Asli      : LONDON IS BLUE, MANCHESTER IS RED
Hasil Enkripsi : QTSITS NX GQZJ, RFSHMJXYJW NX WJI
Hasil Deskripsi: LONDON IS BLUE, MANCHESTER IS RED

✅ Hasil telah disimpan ke file: hasil_caesar_cipher.txt

```

3.2 Tugas Mini 2 - Vigenere Cipher + Analisis Frekuensi

Selain enkripsi dan dekripsi, program ini menampilkan analisis frekuensi ciphertext untuk memperlihatkan pola distribusi huruf.

```

import sys
from colorama import init, Fore
from collections import Counter

# Initialise colorama
init()

# Fungsi enkripsi
def vigenere_encrypt(plain_text, key):
    encrypted_text = ''
    key_repeated = (key * (len(plain_text) // len(key))) +
key[:len(plain_text) % len(key)]
    for i in range(len(plain_text)):
        if plain_text[i].isalpha():
            shift = ord(key_repeated[i].upper()) - ord('A')
            if plain_text[i].isupper():
                encrypted_text += chr((ord(plain_text[i]) + shift -
ord('A')) % 26 + ord('A'))
            else:
                encrypted_text += chr((ord(plain_text[i]) + shift -
ord('a')) % 26 + ord('a'))
        else:
            encrypted_text += plain_text[i]
    return encrypted_text

# Fungsi dekripsi
def vigenere_decrypt(cipher_text, key):

```

```

    decrypted_text = ''
    key_repeated = (key * (len(cipher_text) // len(key))) +
key[:len(cipher_text) % len(key)]
    for i in range(len(cipher_text)):
        if cipher_text[i].isalpha():
            shift = ord(key_repeated[i].upper()) - ord('A')
            if cipher_text[i].isupper():
                decrypted_text += chr((ord(cipher_text[i]) - shift -
ord('A')) % 26 + ord('A'))
            else:
                decrypted_text += chr((ord(cipher_text[i]) - shift -
ord('a')) % 26 + ord('a'))
            else:
                decrypted_text += cipher_text[i]
    return decrypted_text

# Analisis frekuensi huruf
def frequency_analysis(text):
    text = ''.join([c.upper() for c in text if c.isalpha()])
    counter = Counter(text)
    total = sum(counter.values())
    freq = {char: round((count / total) * 100, 2) for char, count in
counter.items()}
    sorted_freq = dict(sorted(freq.items(), key=lambda item: item[1],
reverse=True))
    return sorted_freq

# Input dan proses
key = input('[!] KEY: ')
plaintext = input('[!] Enter your message: ')

cipher_text = vigenere_encrypt(plaintext, key)
print(f"\n[+] Plaintext : {plaintext}")
print(f"{Fore.GREEN}[+] Ciphertext: {cipher_text}")

# Analisis frekuensi
freq = frequency_analysis(cipher_text)
print(f"\n{Fore.YELLOW}=== Frequency Analysis of Ciphertext ===")
for char, f in freq.items():
    print(f"{char} : {f}%")

ask_to_decrypt = input('\n[?] Do you want to decrypt the message?
(Y/N): ').lower()

```

```

if ask_to_decrypt == 'y':
    decrypted_text = vigenere_decrypt(cipher_text, key)
    print(f"{Fore.GREEN}[+] Decrypted text: {decrypted_text}")
elif ask_to_decrypt == 'n':
    sys.exit()
else:
    print(f"{Fore.RED}[-] Invalid input.")

```

Hasil Outputnya:

```

PS D:\Kampus Doc\Season 5\Kriptografi\vigenere2> python vigenere.py
[!] KEY: merah
[!] Enter your message: saya thia dan saya bangga menjadi fans emyu walau malam ini akan kalah lawan nottingham

[+] Plaintext : saya thia dan saya bangga menjadi fans emyu walau malam ini akan kalah lawan nottingham
[+] Ciphertext: wryh xyih hrn eepa neegnm deuveui rees qqpu iecab qrlhy znp ebau orlht cadmr nvfxznnted

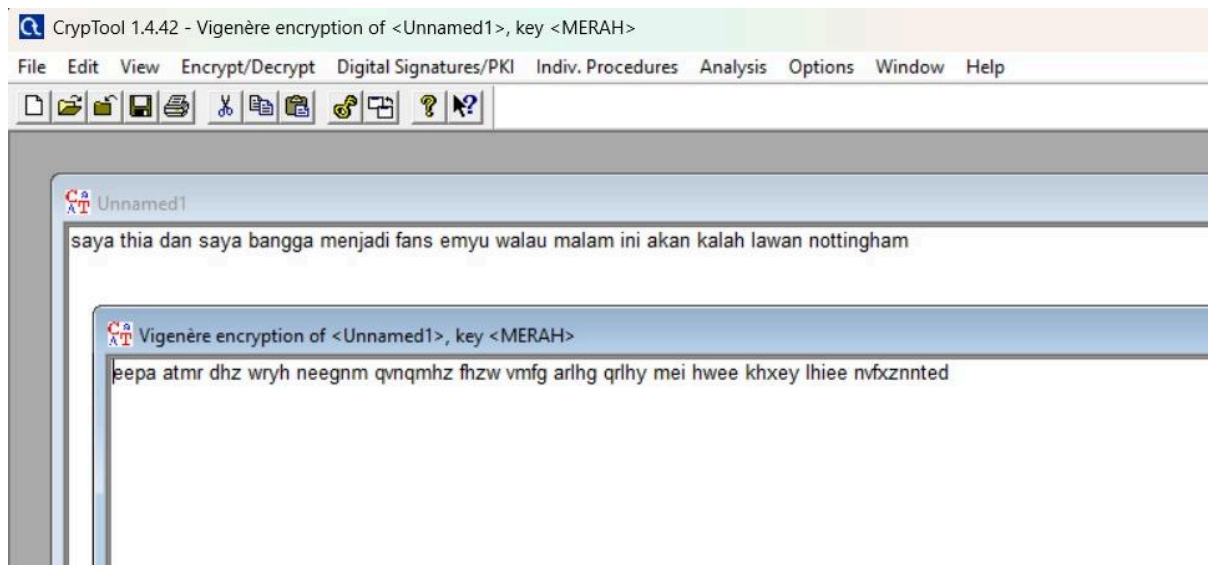
```

Hasil Analisis Frekuensinya:

=== Frequency Analysis of Ciphertext ===

E : 15.07%
 N : 9.59%
 R : 8.22%
 H : 6.85%
 A : 5.48%
 U : 5.48%
 Y : 4.11%
 I : 4.11%
 P : 4.11%
 D : 4.11%
 Q : 4.11%
 X : 2.74%
 M : 2.74%
 V : 2.74%
 C : 2.74%
 B : 2.74%
 L : 2.74%
 Z : 2.74%
 T : 2.74%
 W : 1.37%
 G : 1.37%
 S : 1.37%
 O : 1.37%
 F : 1.37%

Perbandingan Hasil dengan di Cryptool:



4. Kesimpulan

Dari implementasi lima algoritma cipher klasik dan dua tugas mini:

- Setiap cipher memiliki karakteristik unik namun kelemahan utama terletak pada mudahnnya dilakukan analisis frekuensi dan brute force.
- Penerapan dalam bentuk program membantu memahami cara kerja matematis dan logika substitusi/transposisi dalam kriptografi klasik.
- Cipher klasik tidak digunakan dalam sistem keamanan modern, namun penting sebagai dasar untuk memahami konsep kriptografi modern.

5. Lampiran & Link Repositori GitHub

Rizky Rifaldi Wahab

Link Github: https://github.com/rrw788/Kriptografi1_Implementasi-Cipher-Klasik

Thia Shopy Latifah

Tugas 1: <https://github.com/Thiashopy04/KRIPTOGRAFI>