Nama : Fazar Rizwanul Ikhlas

NPM : 20123065

Kelas : C.2.23

TUGAS 1 – Kelompok 13

IMPLEMENTASI 5 ALGORITMA KRIPTOGRAFI KLASIK (CAESAR, VIGENERE, AFFINE, PLAYFAIR, HILL)

1. Tujuan

   Tujuan dari praktikum ini adalah untuk memahami konsep dasar dan implementasi lima algoritma kriptografi klasik, yaitu:

   a. Caesar Cipher
   b. Vigenère Cipher
   c. Affine Cipher
   d. Playfair Cipher
   e. Hill Cipher

   Melalui implementasi ini, diharapkan dapat memahami cara kerja proses enkripsi dan dekripsi, serta menganalisis kelemahan keamanan dari masing-masing algoritma klasik tersebut.

2. Dasar Teori

   2.1. Kriptografi Klasik

   Kriptografi klasik merupakan metode pengamanan pesan dengan cara mengganti atau menyusun ulang karakter dari plaintext berdasarkan aturan tertentu. Meskipun tergolong sederhana, algoritma klasik menjadi dasar pengembangan kriptografi modern.

   2.2. Jenis Algoritma

   a. Caesar Chiper

      Caesar Cipher merupakan bentuk substitusi sederhana yang menggeser setiap huruf sejauh *k* posisi dalam alfabet.

      Rumus:

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

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

   b. Vigenère Cipher

      Vigenère Cipher adalah pengembangan Caesar yang menggunakan kata kunci (key) untuk menentukan nilai pergeseran berbeda di tiap huruf.

      Rumus:

      $C_i = (P_i + K_i) \bmod 26$

   c. Affine Chiper

      Affine Cipher menggunakan transformasi linear dalam substitusi huruf.

      Rumus:

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

      $P = a^{-1}(C - b) \bmod 26$ Syarat: $\gcd(a, 26) = 1$

d. Playfair Chiper
Playfair Cipher menggunakan matriks 5×5 untuk mengenkripsi dua huruf sekaligus (*digraph*). Setiap pasangan huruf dipetakan berdasarkan posisi dalam matriks kunci.

e. Hill Chiper
Hill Cipher bekerja dengan aljabar matriks mod 26**,** di mana teks plaintext diubah menjadi vektor, kemudian dikalikan dengan matriks kunci.
Rumus:
$C = K \times P \bmod 26$

3. Alat dan Bahan

No	Alat / Bahan	Keterangan
1	Bahasa Pemrograman Python 3.12	Implementasi manual algoritma
2	Library numpy	Untuk operasi matriks pada Hill Cipher
3	Terminal / PowerShell/CMD	Menjalankan program interaktif
4	Text Editor (VS Code)	Penulisan kode
5	CrypTool 2	Pembanding hasil enkripsi

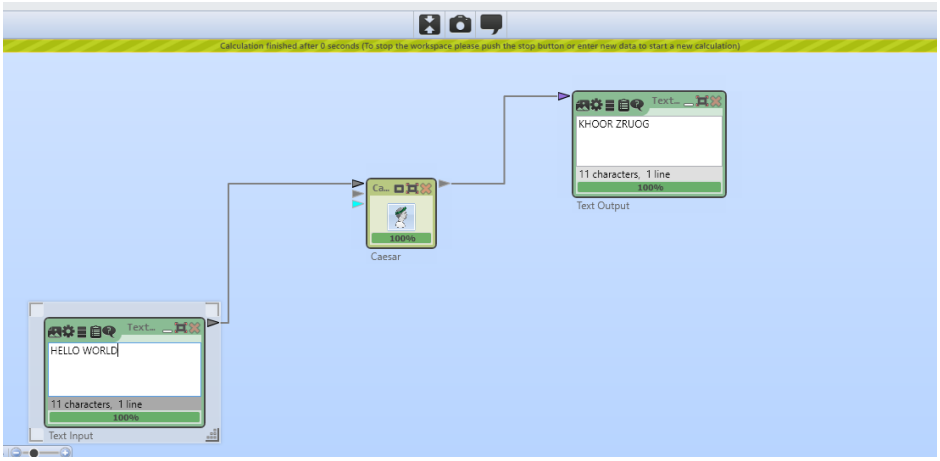
4. Hasil dan Pembahasan
   1. Caesar Chiper
      a. Program Python

```python
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(cipher, shift):
    return caesar_encrypt(cipher, -shift)

if __name__ == "__main__":
    print("=== Caesar Cipher ===")
    text = input("Masukkan teks: ")
    shift = int(input("Masukkan kunci (angka pergeseran): "))
    cipher = caesar_encrypt(text, shift)
    print("Ciphertext:", cipher)
    print("Dekripsi:", caesar_decrypt(cipher, shift))

```

      b. Output

```
PS C:\Users\LENOVO\Documents\Kriptografi\TUGAS 1> python caesar.py
=== Caesar Cipher ===
Masukkan teks: HELLO WORLD
Masukkan kunci (angka pergeseran): 3
Ciphertext: KHOOR ZRUOG
Dekripsi: HELLO WORLD
PS C:\Users\LENOVO\Documents\Kriptografi\TUGAS 1>
```

c. Cryptool



2. Vigenère Cipher

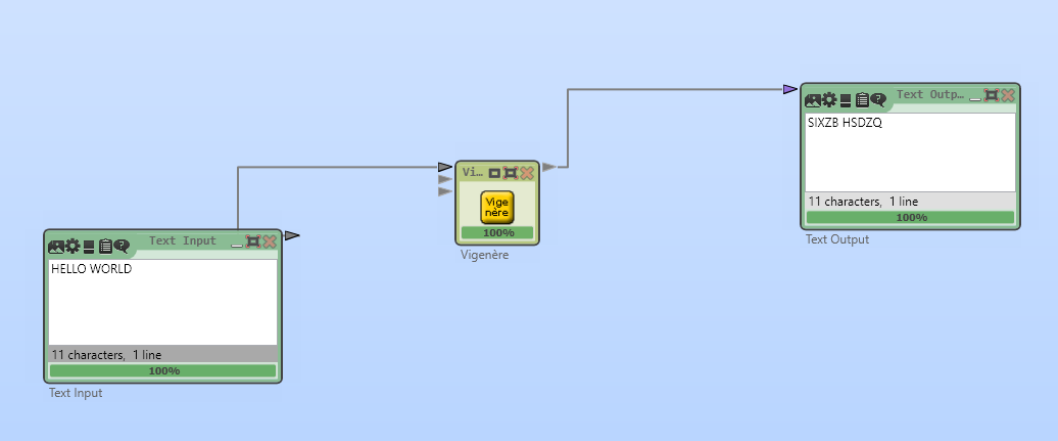
a. Program Python

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

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

if __name__ == "__main__":
    print("=== Vigenère Cipher ===")
    text = input("Masukkan teks: ")
    key = input("Masukkan kunci (kata): ")
    cipher = vigenere_encrypt(text, key)
    print("Ciphertext:", cipher)
    print("Dekripsi:", vigenere_decrypt(cipher, key))
```

b. Output

```
PS C:\Users\LENOVO\Documents\Kriptografi\TUGAS 1> python vignere.py
=== Vigenère Cipher ===
Masukkan teks: HELLO WORLD
Masukkan kunci (kata): LEMON
Ciphertext: SIXZB HSDZQ
Dekripsi: HELLO WORLD
PS C:\Users\LENOVO\Documents\Kriptografi\TUGAS 1>
```

c. Cryptool



3. Affine Chiper

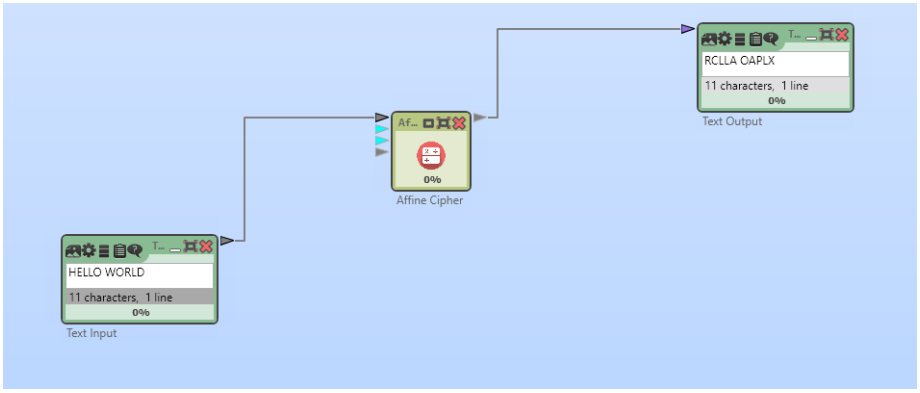
a. Program Python

```python
def mod_inverse(a, m):
    for i in range(m):
        if (a * i) % m == 1:
            return i
    return None

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

def affine_decrypt(cipher, a, b):
    inv = mod_inverse(a, 26)
    result = ""
    for char in cipher.upper():
        if char.isalpha():
            result += chr(((inv * ((ord(char) - 65) - b)) % 26) + 65)
        else:
            result += char
    return result

if __name__ == "__main__":
    print("=== Affine Cipher ===")
    text = input("Masukkan teks: ")
    a = int(input("Masukkan nilai a (coprime dengan 26): "))
    b = int(input("Masukkan nilai b: "))
    cipher = affine_encrypt(text, a, b)
    print("Ciphertext:", cipher)
    print("Dekripsi:", affine_decrypt(cipher, a, b))

```

b. Output

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\LENOVO\Documents\Kriptografi\TUGAS 1> C:/Users/LENOVO/miniforge3/Scripts/activate
PS C:\Users\LENOVO\Documents\Kriptografi\TUGAS 1> conda activate base
PS C:\Users\LENOVO\Documents\Kriptografi\TUGAS 1> python affine.py
=== Affine Cipher ===
Masukkan teks: HELLO WORLD
Masukkan nilai a (coprime dengan 26): 5
Masukkan nilai b: 8
Ciphertext: RCLLA OAPLX
Dekripsi: HELLO WORLD
PS C:\Users\LENOVO\Documents\Kriptografi\TUGAS 1>
```

c. Cryptool



4. Playfair Chipper

a. Program Python

```python
import string

def generate_key_matrix(key):
    key = "".join(dict.fromkeys(key.upper().replace("J", "I")))
    alphabet = "ABCDEFGHIKLMNOPQRSTUVWXYZ"
    matrix = [c for c in key + ''.join([x for x in alphabet if x not in key])]
    return [matrix[i:i+5] for i in range(0, 25, 5)]

def find_position(matrix, char):
    for i in range(5):
        for j in range(5):
            if matrix[i][j] == char:
                return i, j

def playfair_encrypt(text, key):
    matrix = generate_key_matrix(key)
    text = text.upper().replace("J", "I")
    text = "".join([c for c in text if c.isalpha()])
    pairs = []
    i = 0
    while i < len(text):
        a = text[i]
        b = text[i+1] if i+1 < len(text) and text[i+1] != a else 'X'
        pairs.append((a, b))
        i += 2 if i+1 < len(text) and text[i+1] != a else 1

    result = ""
    for a, b in pairs:
        r1, c1 = find_position(matrix, a)
        r2, c2 = find_position(matrix, b)
        if r1 == r2:
            result += matrix[r1][(c1+1)%5] + matrix[r2][(c2+1)%5]
        elif c1 == c2:
            result += matrix[(r1+1)%5][c1] + matrix[(r2+1)%5][c2]
        else:
            result += matrix[r1][c2] + matrix[r2][c1]
    return result

if __name__ == "__main__":
    print("=== Playfair Cipher ===")
    text = input("Masukkan teks: ")
    key = input("Masukkan kunci: ")
    cipher = playfair_encrypt(text, key)
    print("Ciphertext:", cipher)
```

b. Output

```
PS C:\Users\LENOVO\Documents\Kriptografi\TUGAS 1> python playfair.py
=== Playfair Cipher ===
Masukkan teks: HELLO WORLD
Masukkan kunci: MONARCY
Ciphertext: KBSUPMVNMTIA
PS C:\Users\LENOVO\Documents\Kriptografi\TUGAS 1>
```

5. Hill Chiper

a. Program Phyton

```python
import numpy as np

def hill_encrypt(text, key_matrix):
    text = text.upper().replace(" ", "")
    if len(text) % 2 != 0:
        text += 'X'
    result = ""
    for i in range(0, len(text), 2):
        pair = [ord(text[i]) - 65, ord(text[i+1]) - 65]
        enc = np.dot(key_matrix, pair) % 26
        result += chr(int(enc[0]) + 65) + chr(int(enc[1]) + 65)
    return result

if __name__ == "__main__":
    print("=== Hill Cipher ===")
    text = input("Masukkan teks: ")
    print("Gunakan matriks 2x2 (contoh: 3 3; 2 5)")
    a, b = map(int, input("Baris 1 (dua angka): ").split())
    c, d = map(int, input("Baris 2 (dua angka): ").split())
    key = np.array([[a, b], [c, d]])
    cipher = hill_encrypt(text, key)
    print("Ciphertext:", cipher)
```

b. Output

```
PS C:\Users\LENOVO\Documents\Kriptografi\TUGAS 1> python hill.py
=== Hill Cipher ===
Masukkan teks: Hello World
Gunakan matriks 2x2 (contoh: 3 3; 2 5)
Baris 1 (dua angka): 4 3
Baris 2 (dua angka): 5 1
Ciphertext: ONZOSODJBG
PS C:\Users\LENOVO\Documents\Kriptografi\TUGAS 1>
```

5. Analis Kelemahan

a. Caesar Cipher

Caesar Cipher merupakan salah satu cipher substitusi paling sederhana, di mana setiap huruf digeser sejauh k posisi dalam alfabet. Misalnya, jika $k = 3$, maka A menjadi D, B menjadi E, dan seterusnya.

Kelemahan:

1) Ruang kunci sangat kecil. Terdapat hanya 25 kemungkinan kunci (untuk alfabet A–Z), sehingga serangan brute force dapat dilakukan dengan sangat cepat.

2) Pola huruf tetap terlihat. Struktur bahasa plaintext masih tampak pada ciphertext, misalnya huruf yang sering muncul (E, A, N) tetap dapat diidentifikasi.

3) Rentan terhadap analisis frekuensi**.** Penyerang dapat membandingkan frekuensi huruf ciphertext dengan frekuensi bahasa alami untuk menebak pergeseran huruf.

4) Tidak ada difusi. Pergeseran hanya memindahkan huruf tanpa mengubah hubungan antarhuruf.

Kesimpulan:

Keamanan Caesar Cipher hanya bergantung pada kerahasiaan kunci kecil, sehingga cipher ini **tidak aman** untuk digunakan dalam komunikasi modern.

b. Vigenère Cipher
Vigenère Cipher merupakan pengembangan Caesar Cipher dengan menggunakan *kunci huruf berulang*. Setiap huruf pada plaintext digeser sesuai huruf kunci yang bersesuaian.
Kelemahan :

1) Pola pengulangan kunci. Jika kunci pendek dan plaintext panjang, pola pergeseran akan berulang, menyebabkan ciphertext memiliki pola periodik.

2) Rentan terhadap analisis Kasiski dan Friedman. Dua metode ini dapat menentukan panjang kunci dengan menghitung jarak antar pengulangan ciphertext.

3) Masih dapat diserang analisis frekuensi. Setelah panjang kunci diketahui, ciphertext dapat dipecah menjadi beberapa Caesar Cipher terpisah.

4) Tidak adaptif terhadap panjang pesan. Panjang kunci yang kecil meningkatkan kemungkinan terdeteksinya pola.

Kesimpulan

Meskipun lebih kuat dibanding Caesar, Vigenère Cipher tetap mudah dipecahkan dengan analisis statistik bila panjang kunci tidak cukup besar atau digunakan berulang.

c. Affine Cipher
Affine Cipher menggunakan fungsi matematis linear untuk mengenkripsi tiap huruf:
$C=(aP+b) \bmod 26$

dengan *a* dan *b* sebagai kunci, serta *a* harus relatif prima terhadap 26.

Kelemahan

1. Jumlah kunci terbatas. Karena *a* harus relatif prima terhadap 26, hanya ada 12 nilai yang valid untuk *a* dan 26 nilai untuk *b*, total hanya 312 kemungkinan kunci.
2. Masih mempertahankan pola substitusi monoalfabetik. Setiap huruf plaintext selalu dienkripsi menjadi huruf ciphertext yang sama, sehingga analisis frekuensi tetap dapat dilakukan.
3. Mudah dipecahkan dengan plaintext-ciphertext pairs. Dua pasangan huruf plaintext–ciphertext sudah cukup untuk menentukan nilai *a* dan *b*.

Kesimpulan

Affine Cipher masih termasuk cipher sederhana dan **tidak aman** karena sifat substitusi tunggal dan ruang kunci yang kecil.

d. Playfair Cipher

Playfair Cipher menggunakan matriks 5×5 berdasarkan kunci untuk mengenkripsi pasangan huruf (*digraph*). Setiap dua huruf diubah berdasarkan posisi mereka dalam tabel.

Kelemahan

1) Masih dapat dianalisis dengan frekuensi pasangan huruf (digraph frequency). Walaupun menyulitkan analisis huruf tunggal, frekuensi digraph dalam suatu bahasa tetap dapat dimanfaatkan.
2) Tidak menambah kompleksitas secara signifikan. Hanya mengganti unit analisis dari 1 huruf menjadi 2 huruf, tanpa perubahan struktural besar.
3) Rentan terhadap serangan known-plaintext attack. Dengan beberapa pasangan plaintext–ciphertext, tabel kunci dapat direkonstruksi.
4) Tidak ada difusi dan konfusi kuat. Perubahan satu huruf pada plaintext hanya memengaruhi satu pasangan huruf di ciphertext.

Kesimpulan

Playfair Cipher memperkuat substitusi dibanding Caesar, namun masih tidak cukup kuat terhadap analisis kriptografi modern.

e. Hill Cipher

Hill Cipher merupakan cipher berbasis matriks linear. Setiap blok huruf diubah menjadi vektor angka, dikalikan dengan matriks kunci, lalu dimodulo 26 untuk menghasilkan ciphertext.

Kelemahan

1. Kunci harus invertible (determinan ≠ 0 mod 26). Jika tidak, ciphertext tidak dapat didekripsi.
2. Rentan terhadap known-plaintext attack. Dengan *n* pasangan plaintext–ciphertext, penyerang dapat menghitung matriks kunci melalui sistem persamaan linear.
3. Tidak memiliki sifat non-linear. Operasi matriks bersifat linear, sehingga hubungan antarhuruf plaintext dan ciphertext tetap dapat dianalisis.
4. Tidak cocok untuk data pendek. Jika ukuran blok besar tapi teks pendek, cipher tidak efisien.

6. Kesimpulan

Secara keseluruhan, kelima algoritma kriptografi klasik — Caesar, Vigenère, Affine, Playfair, dan Hill Cipher — memiliki nilai historis penting dalam perkembangan ilmu kriptografi, namun seluruhnya memiliki kelemahan mendasar yang membuatnya tidak aman untuk digunakan dalam sistem keamanan modern. Kelemahan utama terletak pada ruang kunci yang sempit, pola substitusi yang mudah dianalisis, serta sifat linear yang memungkinkan dilakukannya serangan seperti brute force, analisis frekuensi, dan known-plaintext attack. Meskipun demikian, algoritma-algoritma tersebut tetap relevan sebagai sarana pembelajaran dasar dalam memahami konsep enkripsi, dekripsi, serta prinsip-prinsip kriptografi yang menjadi dasar bagi pengembangan metode enkripsi modern yang lebih kuat dan kompleks.

REFERENSI

1. Stallings, W. (2017). Cryptography and Network Security: Principles and Practice (7th Edition). Pearson Education.
2. Singh, S. (2000). The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography. Anchor Books
3. Schneier, B. (1996). Applied Cryptography: Protocols, Algorithms, and Source Code in C (2nd Edition). John Wiley & Sons
4. Kessler, G. C. (2019). An Overview of Cryptography. Retrieved from https://www.garykessler.net/library/crypto.html
5. Das, A. (2021). Classical Cryptography – Caesar, Vigenère, Playfair & Hill Cipher Explained. YouTube Video. Channel: Computer Science Hub. Retrieved from https://www.youtube.com/watch?v=DLjzI5dX8j
6. Computerphile. (2014). Classical Ciphers – Explained by Dr. Mike Pound. YouTube Video. Channel: Computerphile. Retrieved from https://www.youtube.com/watch?v=sMOZf4GN3oc