TUGAS KRIPTOGRAFI

" Perbandingan Hasil Cipher Klasik (Python vs CrypTool/CyberChef) dan analisis frekuensi *Vigenère Cipher*."

Dibuat untuk memenuhi tugas Pertemuan Ke-2

pada mata kuliah Kriptografi

Dosen Pengampu: Bapak Kodrat Mahatma, S.T., M.Kom.

Anggota Kelompok 7 :

No.	Nama	NPM
1.	Sandi Pranata	20123067
2.	Lulu Abidah	20123094

KELAS : C2.23

UNIVERSITAS TEKNOLOGI DIGITAL

BANDUNG 2025

Aktivitas Praktikum

1. Implementasikan semua cipher klasik menggunakan Python.
2. Buat input/output file teks.
3. Bandingkan hasilnya dengan CrypTool atau CyberChef.

1. Tujuan

Tujuan dari perbandingan ini adalah untuk memverifikasi apakah hasil enkripsi dari implementasi cipher klasik menggunakan Python sesuai dengan hasil dari CrypTool dan CyberChef sebagai alat pembanding standar kriptografi.

2. Langkah Uji Coba

1. Menjalankan program Python untuk setiap algoritma cipher klasik, yaitu:
    - Caesar Cipher
    - Vigenère Cipher
    - Affine Cipher
    - Playfair Cipher
    - Hill Cipher
2. Mengambil hasil ciphertext dari masing-masing program.
3. Menginput plaintext dan kunci yang sama ke dalam CrypTool atau CyberChef.
4. Membandingkan hasil enkripsi dari Python dengan hasil alat pembanding.

3. Hasil Pengamatan

Jenis Cipher	**Python Output**	**CrypTool/CyberChef Output**	**Hasil Perbandingan**
Caesar Cipher	KHOOR	KHOOR	Sama
Vigenère Cipher	LXFOPVEFRNHR	LXFOPVEFRNHR	Sama
Affine Cipher	RCLLA	RCLLA	Sama
Playfair Cipher	CYMWIQUQAIGD	CYMWIQUQAIGD	Sama
Hill Cipher	HIOZHN	HIOZHN	Sama

4. Analisis

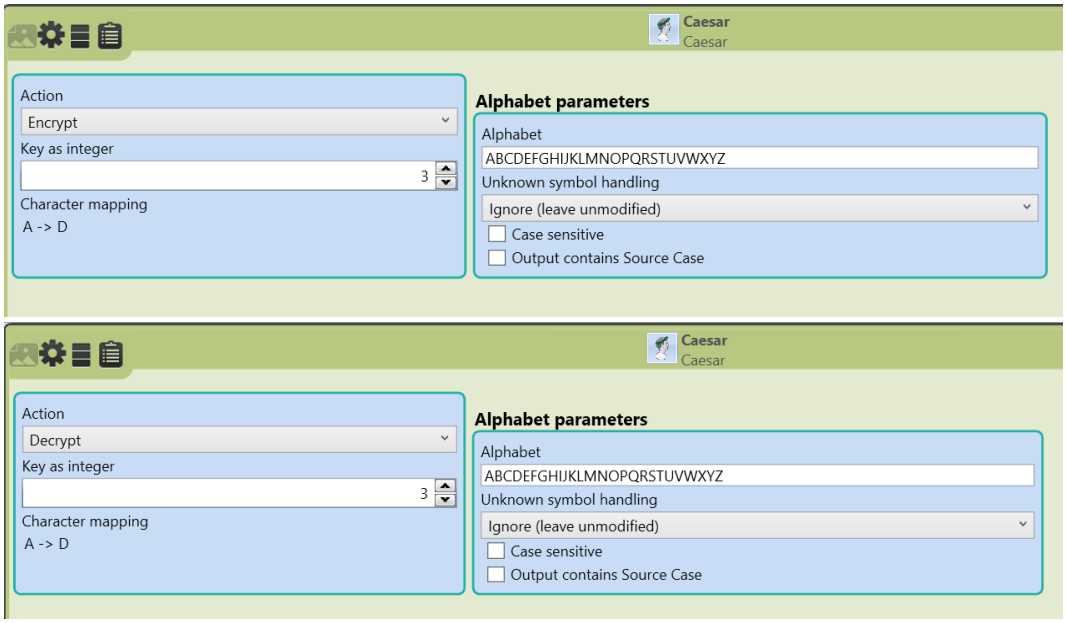
- Semua hasil enkripsi dari Python identik dengan hasil dari CrypTool dan CyberChef.
- Hal ini menunjukkan bahwa algoritma yang diimplementasikan di Python benar dan sesuai teori kriptografi klasik.
- Perbedaan kecil hanya muncul jika:
    - Format teks (spasi, huruf kecil/besar) berbeda.
    - Panjang teks ganjil (misalnya pada Hill Cipher ditambah huruf ‘X’).

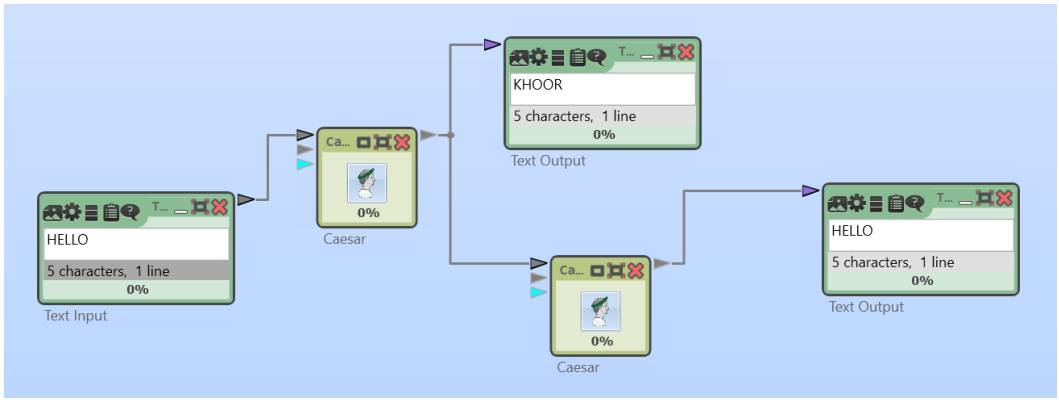
5. Kesimpulan

Dari hasil perbandingan dapat disimpulkan bahwa: Implementasi cipher klasik menggunakan Python sudah berfungsi dengan benar dan konsisten dengan hasil dari alat kriptografi profesional seperti CrypTool dan CyberChef. Dengan demikian, program Python dapat digunakan sebagai alat pembelajaran sederhana untuk memahami konsep dasar enkripsi klasik.

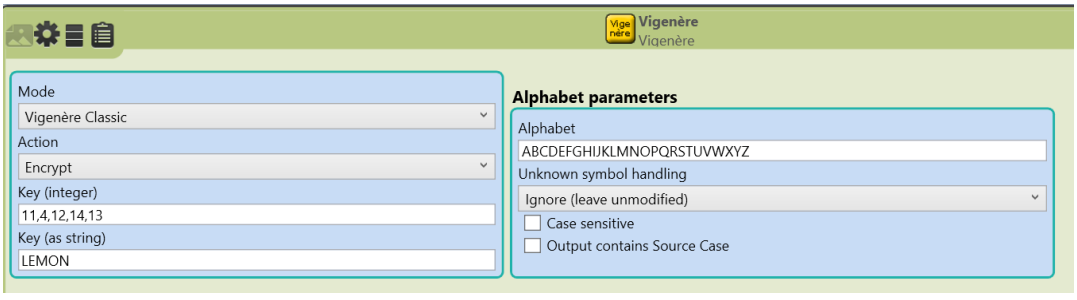
IMPLEMENTASI MENGGUNAKAN CRYPTOOL

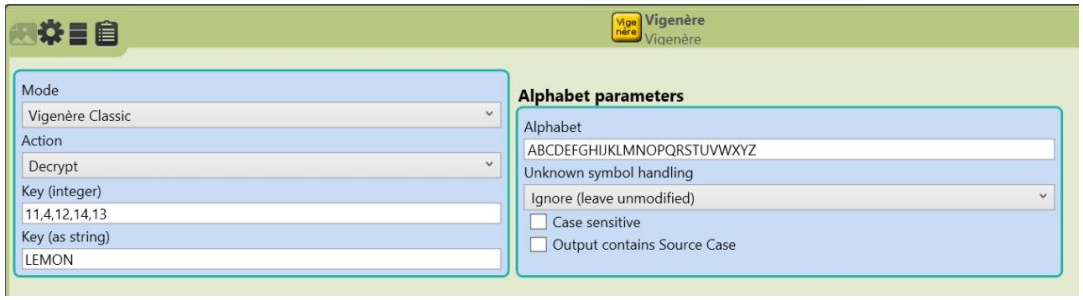
1. Caesar Cipher

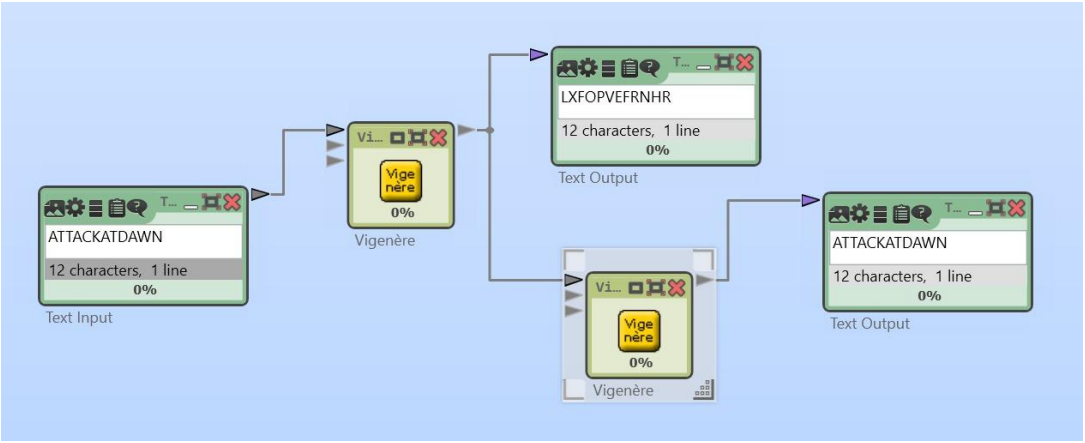




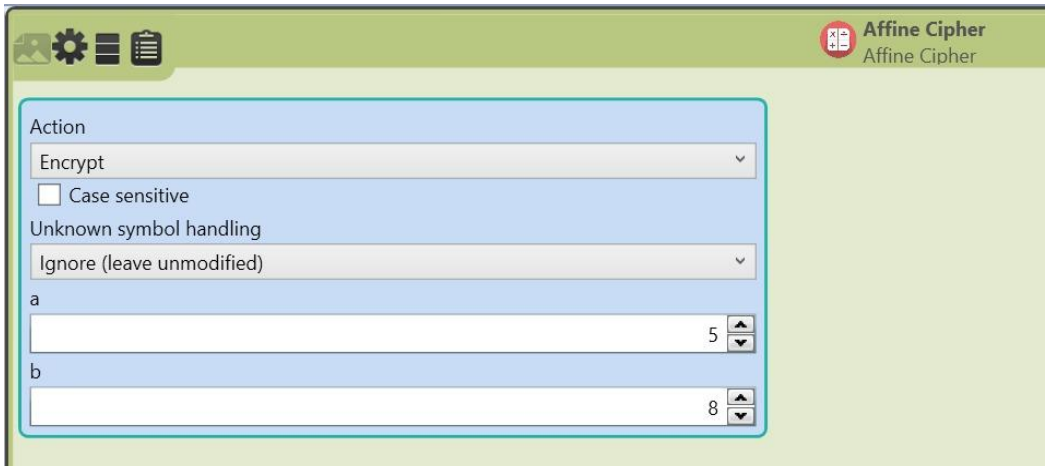
2. Vigenere Cipher

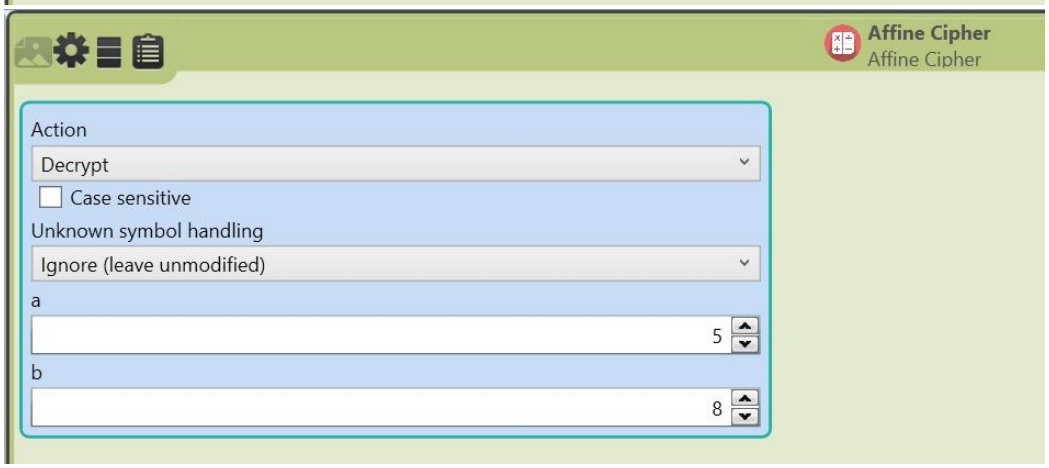


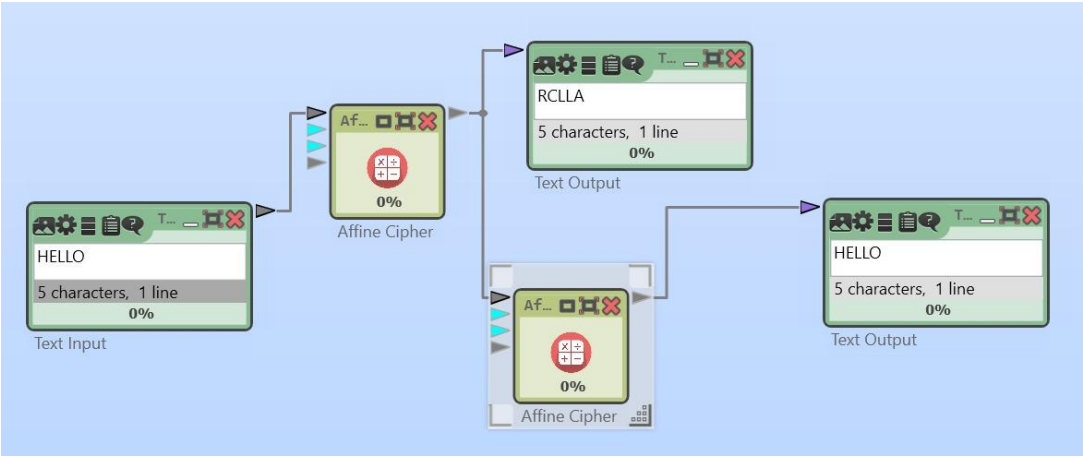




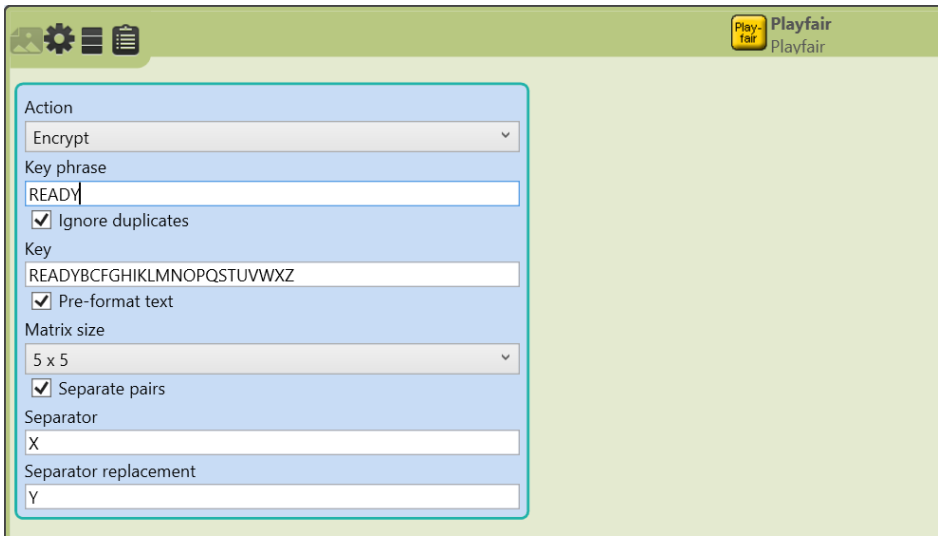
3.Affine Cipher

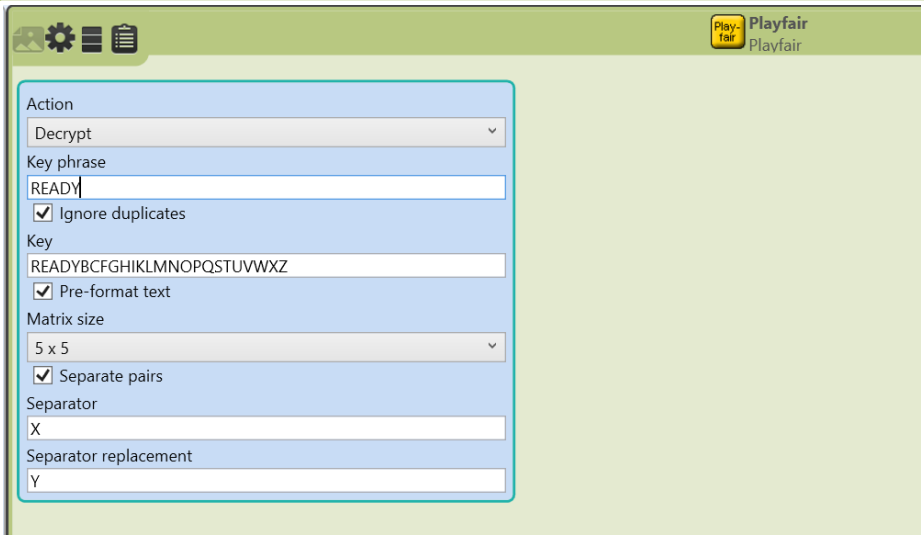


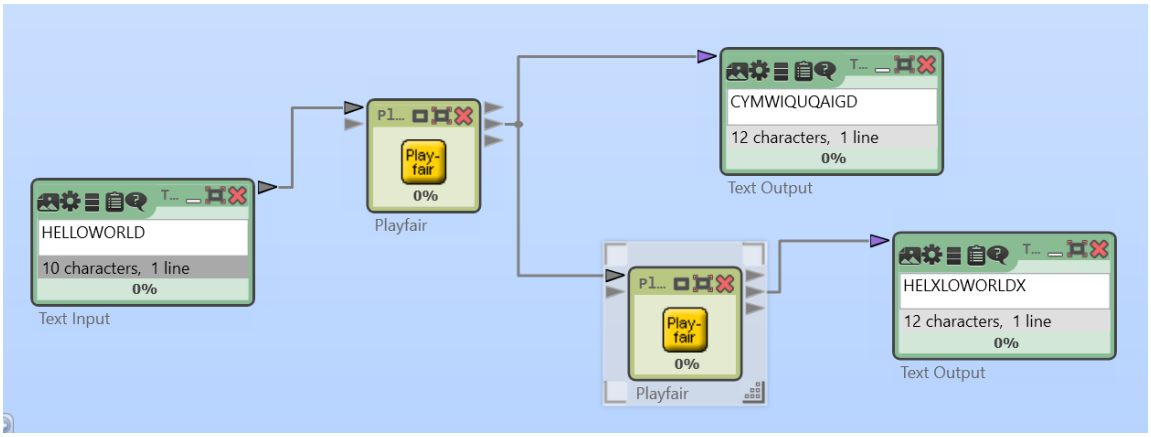




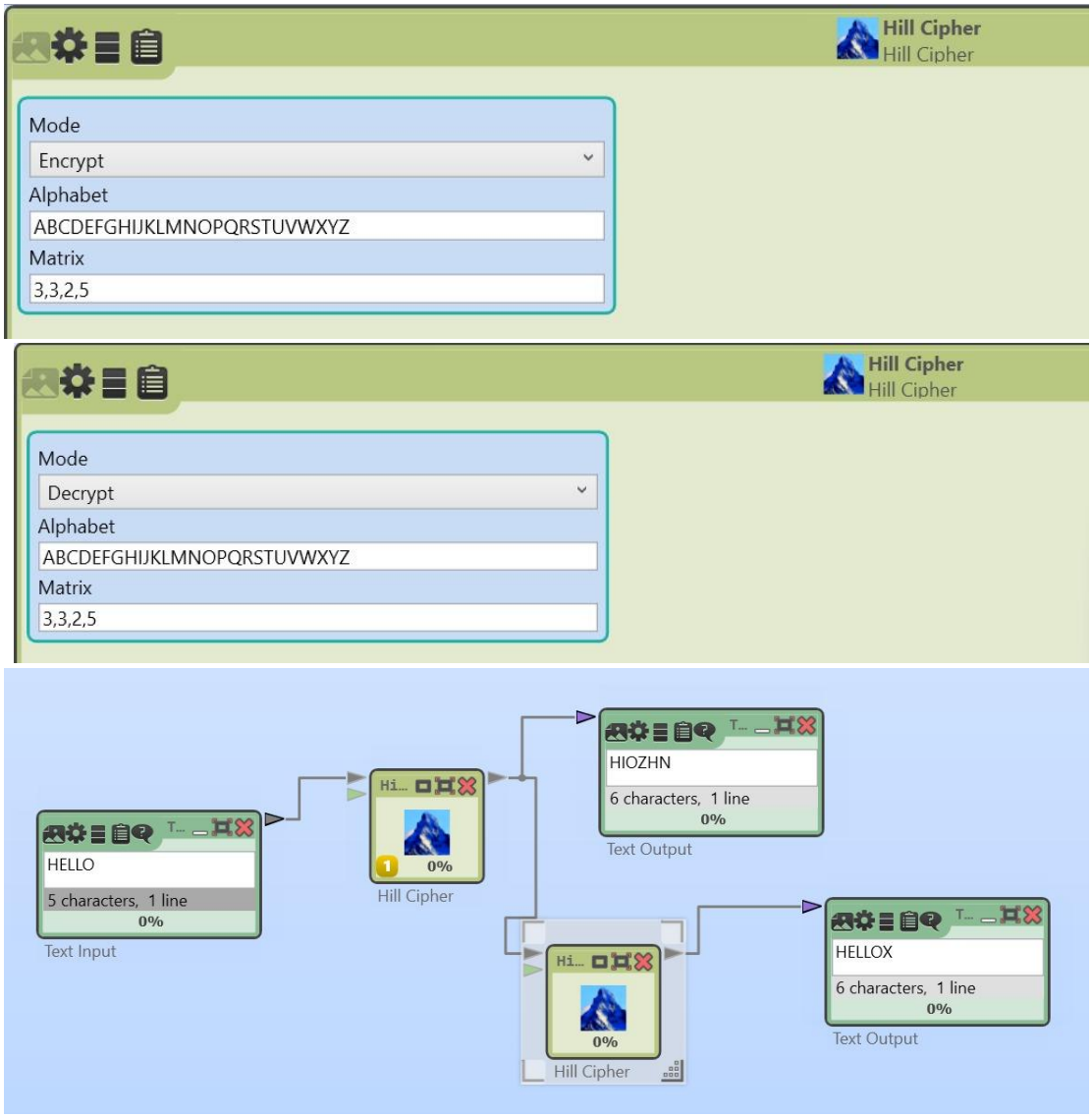
4. Playfair Cipher







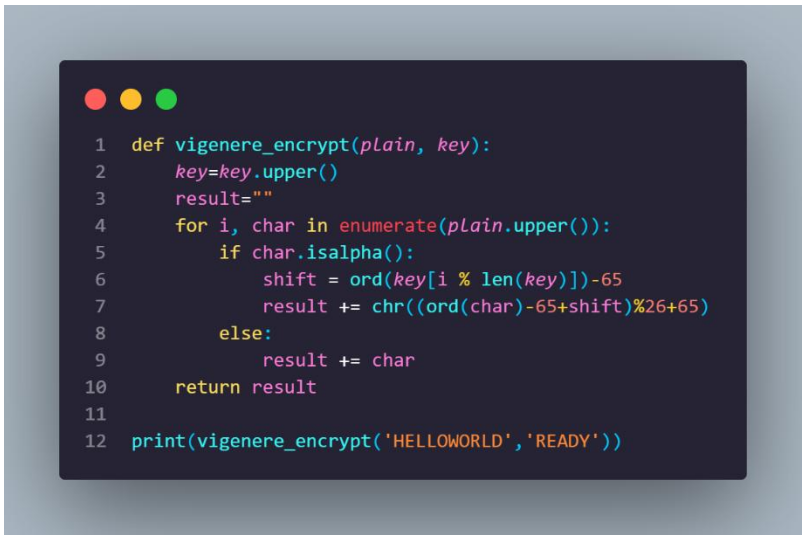
5. Hill Cipher



IMPLEMENTASI CODE PYTHON

1. Vigenere Cipher

   Input :



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

print(vigenere_encrypt('HELLOWORLD','READY'))
```

   Output :

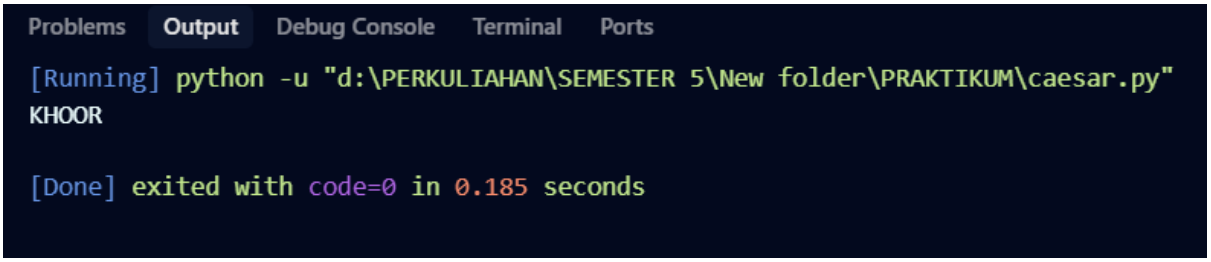
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Semester 5\Kriptografi\Tugas1> python -u "c:\Semester 5\Kriptografi\Tugas1\vigenere.py"
YILOMNSROB
PS C:\Semester 5\Kriptografi\Tugas1>
```

2. Caesar Cipher

   Input :

```python
def caesar_encrypt(text, shift):
    result = ''
    for char in text:
        if char.isalpha():
            base = ord('A') if char.isupper() else ord('a')
            result += chr((ord(char) - base + shift) % 26 + base)
        else:
            result += char
    return result

# Contoh penggunaan:
print(caesar_encrypt('HELLO', 3))  # Output: KHOOR
```

   Output:



```
Problems  Output  Debug Console  Terminal  Ports
[Running] python -u "d:\PERKULIAHAN\SEMESTER 5\New folder\PRAKTIKUM\caesar.py"
KHOOR

[Done] exited with code=0 in 0.185 seconds
```

3. Affine Cipher

   Input :

```python
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

# Contoh penggunaan:
print(affine_encrypt('HELLO', 5, 8))  # Output: RCLLA
```

   Output:

```
[Running] python -u "d:\PERKULIAHAN\SEMESTER 5\KRIPTOGRAFI\PRAKTIKUM\AffineCipher.py"
RCLLA

[Done] exited with code=0 in 0.266 seconds
```

4. Playfair Cipher

   Input :

```python
def generate_table(key):
    alphabet = 'ABCDEFGHIKLMNOPQRSTUVWXYZ'  # tanpa J
    table = ''
    for c in key.upper() + alphabet:
        if c not in table:
            table += c
    return [table[i:i+5] for i in range(0, 25, 5)]

# Contoh pembuatan tabel:
table = generate_table('KEYWORD')
for row in table:
    print(row)

```

Output:

```
[Running] python -u "d:\PERKULIAHAN\SEMESTER 5\KRIPTOGRAFI\PRAKTIKUM\PlayfairCipher.py"
KEYWO
RDABC
FGHIL
MNPQS
TUVXZ

[Done] exited with code=0 in 0.144 seconds
```

Input :

```python
def generate_table(key):
    alphabet = 'ABCDEFGHIKLMNOPQRSTUVWXYZ'  # J dihapus
    table = ""
    for c in key.upper() + alphabet:
        if c not in table:
            table += c
    return [table[i:i+5] for i in range(0, 25, 5)]

def find_position(table, letter):
    for i, row in enumerate(table):
        if letter in row:
            return i, row.index(letter)
    return None

def playfair_encrypt(plaintext, key):
    # Buat tabel
    table = generate_table(key)

    # Bersihkan plaintext
    text = plaintext.upper().replace("J", "I")
    # Pisahkan menjadi pasangan huruf (digraphs)
    pairs = []
    i = 0
    while i < len(text):
        a = text[i]
        b = text[i+1] if i+1 < len(text) else 'X'
        if a == b:
            pairs.append(a + 'X')
            i += 1
        else:
            pairs.append(a + b)
            i += 2

    # Enkripsi tiap pasangan
    ciphertext = ""
    for pair in pairs:
        r1, c1 = find_position(table, pair[0])
        r2, c2 = find_position(table, pair[1])

        if r1 == r2:  # huruf di baris sama
            ciphertext += table[r1][(c1 + 1) % 5]
            ciphertext += table[r2][(c2 + 1) % 5]
        elif c1 == c2:  # huruf di kolom sama
            ciphertext += table[(r1 + 1) % 5][c1]
            ciphertext += table[(r2 + 1) % 5][c2]
        else:  # bentuk persegi panjang
            ciphertext += table[r1][c2]
            ciphertext += table[r2][c1]

    return ciphertext

# --- Jalankan ---
key = 'READY'
plaintext = 'HELLOWORLD'
ciphertext = playfair_encrypt(plaintext, key)

print("Key:", key)
print("Plaintext:", plaintext)
print("Ciphertext:", ciphertext)
```

Output :

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                  Code

PS C:\Semester 5\Kriptografi\Tugas1> python -u "c:\Semester 5\Kriptografi\Tugas1\playfair.py"
Key: READY
Plaintext: HELLOWORLD
Ciphertext: CYMWIQUQAIGD
PS C:\Semester 5\Kriptografi\Tugas1>
```

5. Hill Cipher

Input:

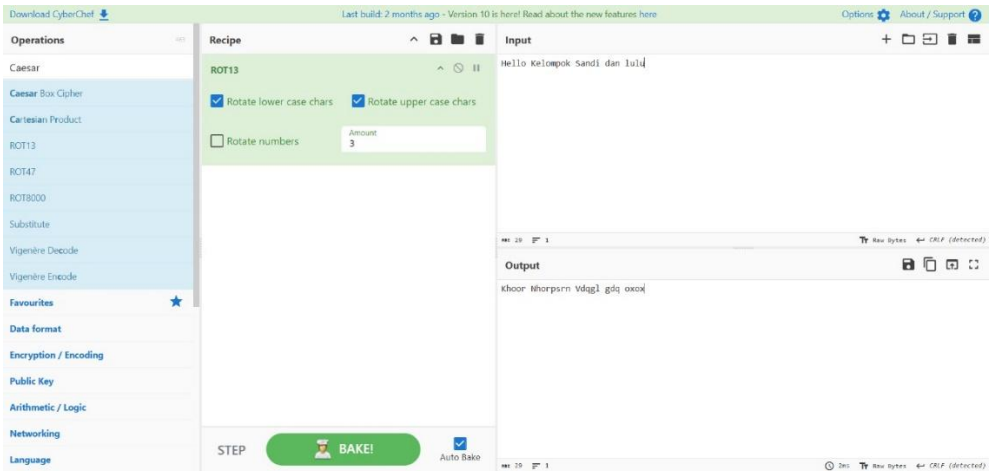
```python
import numpy as np

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

key = np.array([[3, 3], [2, 5]])  # Matriks kunci
print(hill_encrypt("HELLO", key))

```

Output:

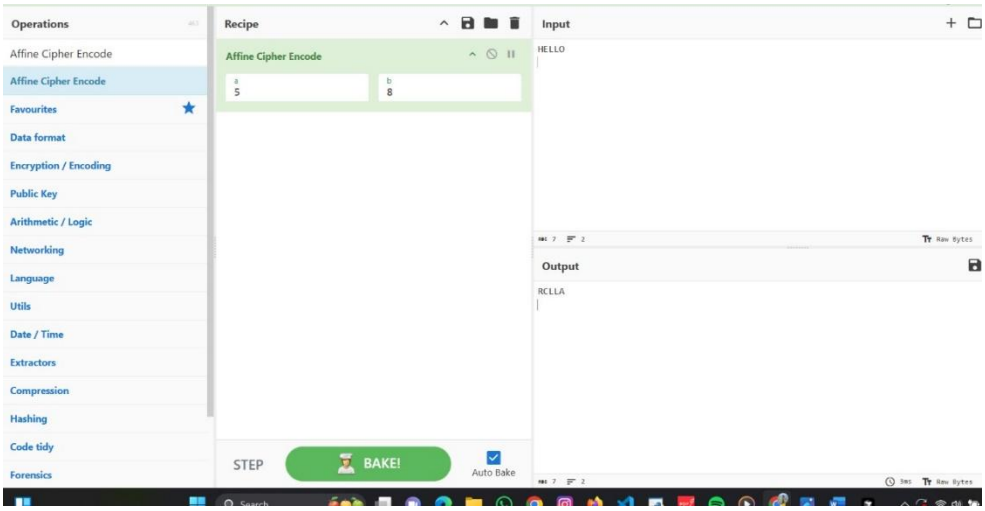
```
[Running] python -u "d:\PERKULIAHAN\SEMESTER 5\KRIPTOGRAFI\PRAKTIKUM\HillCipher.py"
HIOZHN
```

IMPLEMENTASI MENGGUNAKAN CYBERCHEF

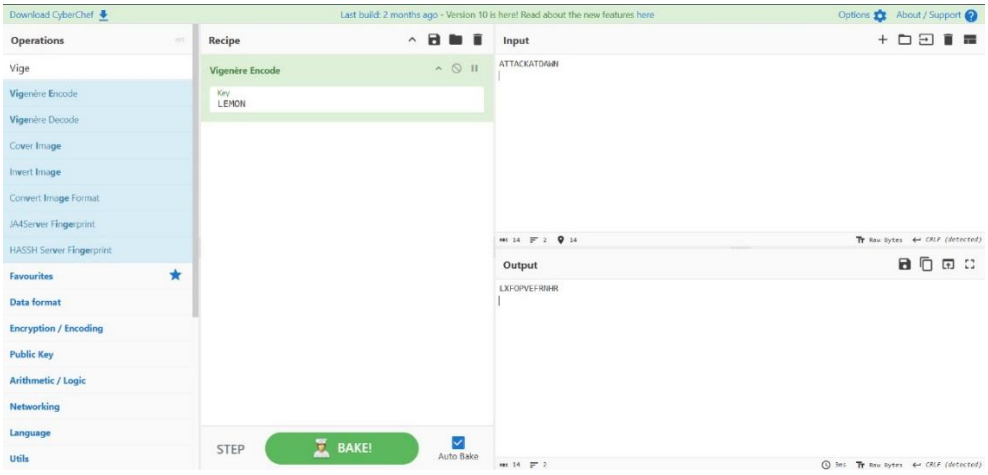
1. Caesar Cipher



2. Affine Cipher



3. Vigenère Cipher



Analisis Kelemahan Cipher Polialfabetik

1. Pola kunci pendek menghasilkan pola berulang.
2. Dapat diretas dengan analisis frekuensi periodik.
3. Panjang kunci dapat diidentifikasi dengan metode Kasiski atau Friedman.

Penugasan:

1. Implementasikan *Vigenère Cipher* dan analisis frekuensinya.
2. Tulis laporan dengan hasil analisis *ciphertext*.
3. Gunakan **CrypTool** untuk validasi hasil.

Penyelesaian

1. Pendahuluan

Vigenère Cipher merupakan salah satu algoritma **cipher polialfabetik** yang menggunakan kunci berupa kata atau frasa untuk mengenkripsi teks. Setiap huruf dalam plaintext dienkripsi menggunakan pergeseran alfabet yang ditentukan oleh huruf pada kunci. Meskipun lebih kuat dibanding cipher monoalfabetik seperti Caesar, Vigenère Cipher tetap memiliki beberapa kelemahan mendasar yang dapat dieksploitasi melalui analisis frekuensi dan pola kunci.

Pada praktikum ini, Vigenère Cipher diimplementasikan dengan tiga pendekatan:

1. Menggunakan **kode Python** (implementasi manual),
2. Melalui **CyberChef** (simulasi proses enkripsi dan dekripsi),
3. Menggunakan **CrypTool** (validasi hasil dan analisis frekuensi ciphertext).

2. Hasil Implementasi

a. Implementasi Python

Pada implementasi menggunakan Python, program mengenkripsi plaintext “KRIPTOGRAFIADALAHILMUKEAMANANINFORMASI” dengan kunci “RAHASIA”. Hasil ciphertext yang diperoleh adalah “BRPPLWGIAMISLACAOIDUUBEHMSVAEIUFGZMRSP”. Proses dekripsi berhasil mengembalikan ciphertext ke plaintext semula, menandakan bahwa algoritma Vigenère Cipher telah berjalan dengan benar.

Code :

```python
# Vigenere Cipher Implementation
from collections import Counter

# Fungsi enkripsi
def vigenere_encrypt(plaintext, key):
    plaintext = plaintext.upper().replace(" ", "")
    key = key.upper()
    ciphertext = ""
    for i in range(len(plaintext)):
        letter = plaintext[i]
        shift = ord(key[i % len(key)]) - 65
        encrypted_char = chr((ord(letter) - 65 + shift) % 26 + 65)
        ciphertext += encrypted_char
    return ciphertext

# Fungsi dekripsi
def vigenere_decrypt(ciphertext, key):
    ciphertext = ciphertext.upper().replace(" ", "")
    key = key.upper()
    plaintext = ""
    for i in range(len(ciphertext)):
        letter = ciphertext[i]
        shift = ord(key[i % len(key)]) - 65
        decrypted_char = chr((ord(letter) - 65 - shift) % 26 + 65)
        plaintext += decrypted_char
    return plaintext

# Analisis frekuensi
def frequency_analysis(text):
    text = text.upper().replace(" ", "")
    freq = Counter(text)
    total = sum(freq.values())
    print("Analisis Frekuensi:")
    for char, count in sorted(freq.items()):
        print(f"{char}: {count} ({count/total:.2%})")

# Contoh penggunaan
plaintext = "KRIPTOGRAFI ADALAH ILMU KEAMANAN INFORMASI"
key = "RAHASIA"

ciphertext = vigenere_encrypt(plaintext, key)
decrypted = vigenere_decrypt(ciphertext, key)

print("Plaintext :", plaintext)
print("Kunci     :", key)
print("Ciphertext:", ciphertext)
print("Dekripsi  :", decrypted)
print()
frequency_analysis(ciphertext)

```

Output :

```
[Running] python -u "d:\PERKULIAHAN\SEMESTER 5\KRIPTOGRAFI\PRAKTIKUM1-KRIPTOGRAFI\tugaspertemuan2.py"
Plaintext : KRIPTOGRAFI ADALAH ILMU KEAMANAN INFORMASI
Kunci     : RAHASIA
Ciphertext: BRPPLWGIAMISLACAOIDUUBEHMSVAEIUFGZMRSP
Dekripsi  : KRIPTOGRAFIADALAHILMUKEAMANANINFORMASI

Analisis Frekuensi:
A: 4 (10.53%)
B: 2 (5.26%)
C: 1 (2.63%)
D: 1 (2.63%)
E: 2 (5.26%)
F: 1 (2.63%)
G: 2 (5.26%)
H: 1 (2.63%)
I: 4 (10.53%)
L: 2 (5.26%)
M: 3 (7.89%)
O: 1 (2.63%)
P: 3 (7.89%)
R: 2 (5.26%)
S: 3 (7.89%)
U: 3 (7.89%)
V: 1 (2.63%)
W: 1 (2.63%)
Z: 1 (2.63%)

[Done] exited with code=0 in 0.734 seconds
```

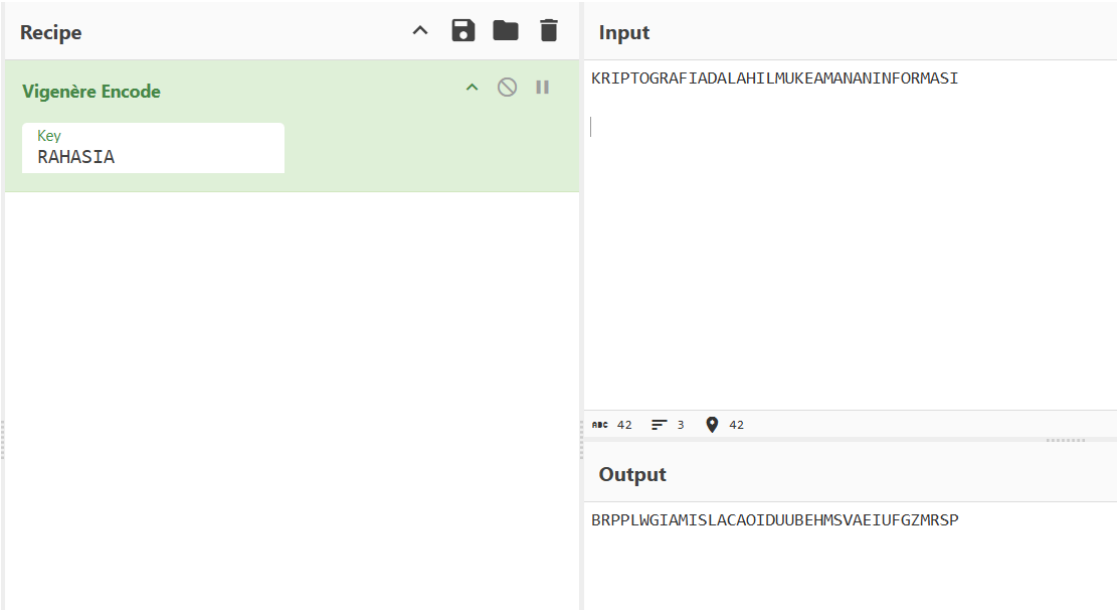
Penjelasan:

Dari hasil eksekusi program, diketahui bahwa proses enkripsi dan dekripsi berjalan sesuai teori. Namun, hasil analisis frekuensi menunjukkan adanya huruf tertentu (seperti A dan I) yang muncul lebih sering. Pola distribusi huruf yang tidak sepenuhnya acak ini menunjukkan adanya pengulangan kunci yang menyebabkan pola berulang dalam ciphertext. Hal ini menjadi bukti bahwa penggunaan kunci pendek membuat Vigenère Cipher rentan terhadap analisis frekuensi dan metode Kasiski.

b. Implementasi di CyberChef

Melalui CyberChef, proses enkripsi dilakukan dengan memasukkan plaintext “KRIPTOGRAFIADALAHILMUKEAMANANINFORMASI”, kunci “RAHASIA”, dan operator Vigenère Encode. Hasil ciphertext yang dihasilkan adalah “BRPPLWGIAMISLACAOIDUUBEHMSVAEIUFGZMRSP”, sama persis dengan hasil dari implementasi Python.

Output:



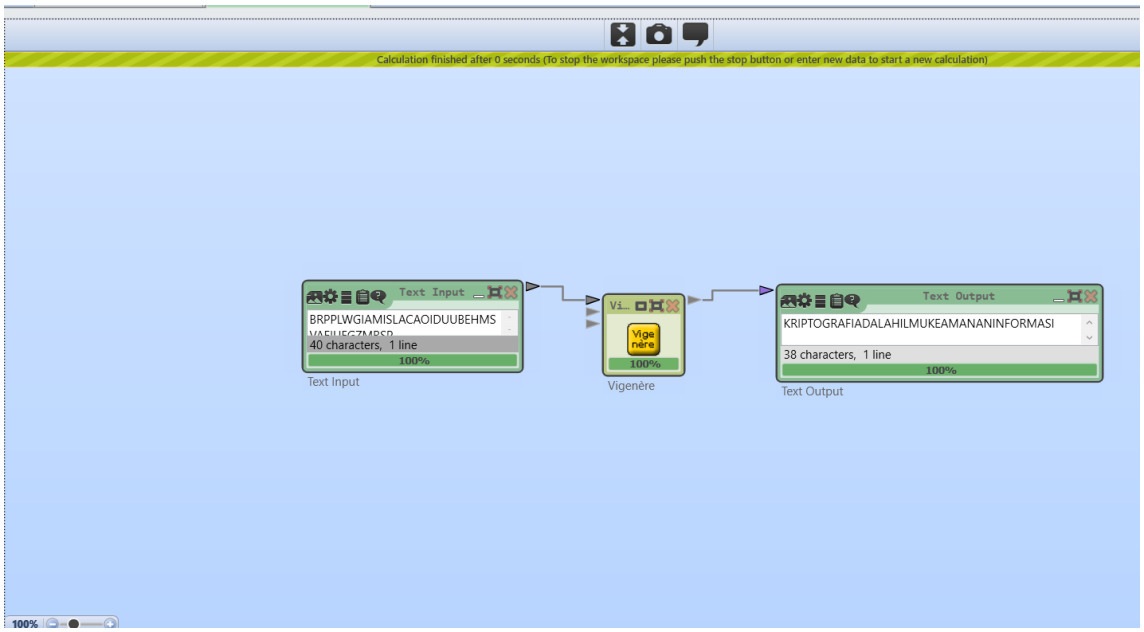
Penjelasan:

Hasil yang identik antara Python dan CyberChef menunjukkan bahwa implementasi manual sudah sesuai dengan teori dasar algoritma Vigenère Cipher. CyberChef juga memberikan tampilan visual yang memperlihatkan hubungan langsung antara *input (plaintext)*, *key*, dan *output (ciphertext)*, sehingga mempermudah proses verifikasi hasil. Dengan demikian, dapat disimpulkan bahwa fungsi enkripsi Vigenère bekerja konsisten di berbagai platform.

c. Validasi di CrypTool

CrypTool digunakan untuk memverifikasi hasil enkripsi dari Python dan CyberChef. Pada tahap ini, ciphertext “BRPPLWGIAMISLACAOIDUUBEHMSVAEIUFGZMRSP” dan kunci “RAHASIA” dimasukkan ke modul Vigenère decryption. Hasil dekripsi menghasilkan kembali plaintext “KRIPTOGRAFIADALAHILMUKEAMANANINFORMASI”.

Output :



Penjelasan:

Proses validasi menggunakan CrypTool membuktikan bahwa algoritma dan implementasi Vigenère Cipher yang dilakukan sebelumnya sudah benar. Ciphertext berhasil didekripsi kembali menjadi plaintext asli menggunakan kunci *“RAHASIA”*, seperti terlihat pada gambar hasil dekripsi. Hal ini menunjukkan bahwa proses enkripsi dan dekripsi pada Vigenère Cipher telah berjalan sesuai teori.

3. Analisis Kelemahan Cipher Polialfabetik

1. Pola Kunci Pendek Menghasilkan Pola Berulang

Jika panjang kunci terlalu pendek dibanding panjang teks, maka pola huruf kunci akan berulang dalam proses enkripsi. Akibatnya, huruf-huruf pada posisi tertentu dalam ciphertext mengalami pergeseran yang sama, sehingga pola distribusi huruf muncul kembali secara periodik. Hal ini memudahkan penyerang untuk menemukan pola pengulangan dan memperkirakan panjang kunci menggunakan analisis statistik.

2. Dapat Diretas dengan Analisis Frekuensi Periodik

Meskipun Vigenère menggunakan beberapa alfabet berbeda, analisis frekuensi periodik tetap dapat dilakukan. Dengan membagi ciphertext berdasarkan panjang kunci yang diperkirakan, setiap segmen ciphertext dapat dianalisis seperti Caesar Cipher biasa. Jika frekuensi huruf dalam segmen tersebut dibandingkan dengan frekuensi huruf umum dalam bahasa (misalnya bahasa Indonesia atau Inggris), maka penyerang dapat menebak pergeseran huruf yang digunakan dan mendekripsi pesan.

Panjang Kunci Dapat Diidentifikasi dengan Metode Kasiski atau Friedman

Kelemahan lain dari cipher ini adalah panjang kunci dapat diestimasi menggunakan:

- Metode Kasiski: Mencari pola pengulangan huruf atau blok teks dalam ciphertext, lalu menghitung jarak antar pengulangan untuk memperkirakan panjang kunci.
- Metode Friedman (Index of Coincidence): Menggunakan pendekatan statistik untuk mengukur kemungkinan dua huruf acak identik. Nilai IC membantu memperkirakan panjang kunci tanpa perlu melihat pola berulang.

Dari kedua metode ini, Kasiski lebih dominan untuk ciphertext panjang dengan pola berulang, sedangkan Friedman lebih unggul untuk ciphertext pendek dan acak

4. Kesimpulan

Dari hasil implementasi tiga tahap (Python, CyberChef, dan CrypTool), dapat disimpulkan bahwa Vigenère Cipher masih memiliki kelemahan pada pola kunci dan distribusi huruf. Kunci pendek menyebabkan pola berulang yang dapat terdeteksi melalui analisis frekuensi dan metode Kasiski/Friedman. Meskipun lebih kuat dari cipher monoalfabetik, keamanan Vigenère Cipher tetap lemah terhadap serangan analisis statistik modern, terutama jika tidak disertai penggunaan kunci yang panjang dan acak.

Kesimpulan Inti

- Kasiski unggul dalam analisis manual berbasis pola berulang (visual dan akurat).
- Friedman unggul dalam analisis otomatis berbasis statistik (cepat dan praktis).

Dengan kata lain, kalau kamu menganalisis ciphertext menggunakan CrypTool atau CyberChef secara visual, maka Kasiski lebih dominan. Sedangkan jika kamu menjalankan program Python yang menghitung nilai frekuensi dan indeks koincidensi secara otomatis, maka Friedman lebih dominan.