IMPLEMENTASI 5 ALGORITMA CIPHER KLASIK

Nama : Tini Meilani

NIM : 20123055

Kelas : C2.23 S1 Informatika

1. Caesar Cipher
   - Implementasi Algoritma menggunakan Python

Caesar Cipher

```
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

print(caesar_encrypt('HELLO', 3))
```

```
KHOOR
```

   - Implementasi Cipher menggunakan CrypTool

   - Analisis Kekuatan dan Kelemahan

     Algoritma ini sangat sederhana, mudah dipahami dan di implementasikan, proses enkripsi dan dekripsi nya juga cepat karena hanya perlu menggeser beberapa langkah saja. Tapi algoritma ini sangat mudah dibobol karena jumlah kemungkinan kuncinya sedikit (hanya 25) dikarenakan hal itu pola huruf menjadi mudah dikenali dan bisa ditebak hanya lewat analisis frekuensi huruf. Algoritma ini tidak cocok untuk mengamankan data penting.

2. Vignere Cipher
   - Implementasi Algoritma menggunakan Python

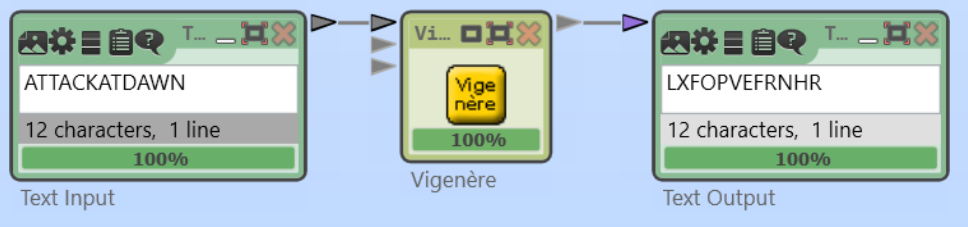
Vigenère Cipher

```
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('ATTACKATDAWN', 'LEMON'))
```

```
LXFOPVEFRNHR
```

- Implementasi Cipher menggunakan CrypTool



- Analisis Kekuatan dan Kelemahan

  Algoritma ini lebih aman dibandingkan Caesar karena menggunakan kunci berupa kata bukan angka tunggal jadi membuat pola huruf lebih sulit ditebak karena pergeserannya berubah-ubah sesuai huruf kunci. Tetapi jika kunci yang dipakai pendek atau berulang-ulang maka pola masih bisa ditemukan lewat analisi frekuensi, jika seseorang tahu sebagian plaintextnya maka kunci dapat ditebah dengan teknik tertentu.

3. Affine Cipher
   - Implementasi Algoritma menggunakan Python

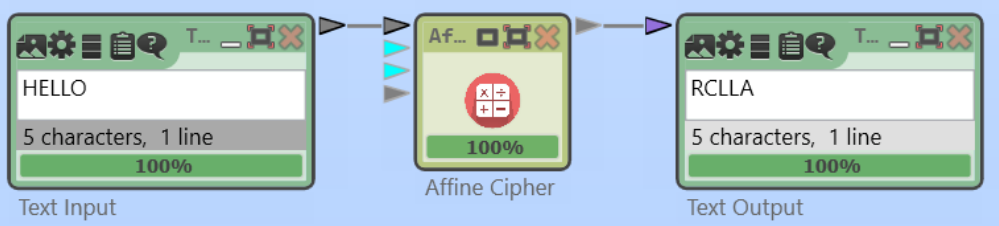
Affine Cipher

```
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

print(affine_encrypt('HELLO', 5, 8))
```

```
RCLLA
```

- Implementasi Cipher menggunakan CrypTool



- Analisis Kekuatan dan Kelemahan

  Algoritma ini juga lebih kuat dari Caesar karena menggunakan dua operasi yaitu perkalian dan penjumlahan, kombinasi dua kunci membuat hasil enkripsinya terlihat lebih acak, algoritma ini juga mudah dipahami dan di implementasikan. Tetapi jika peretas tahu dua huruf hasil enkripsi dan huruf aslinya, dia bisa menghitung kuncinya. Algoritma ini juga rentan terhadap analisis frekuensi dan tidak cocok untuk pesan yang perlu keamanan tinggi.

4. Playfair Cipher
   - Implementasi Algoritma menggunakan Python

Playfair Chiper

```
def generate_table(key):
    alphabet = 'ABCDEFGHIKLMNOPQRSTUVWXYZ'
    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)]

table = generate_table('KEYWORD')
for row in table:
    print(row)
```

```
KEYWO
RDABC
FGHIL
MNPQS
TUVXZ
```

- Implementasi Cipher menggunakan CrypTool

- Analisis Kekuatan dan Kelemahan

  Algoritma ini menggunakan pasangan huruf yang menjadikan algoritma ini sulit dianalisis dari pada cipher lain yang bekerja per huruf, algoritma ini juga menghasilkan ciphertect yang tidak memiliki pola huruf berulang yang jelas. Tetapi algoritma ini masih bisa dianalisis menggunakan Teknik statistic jika ada cukup ciphertext, algoritma ini juga tidak bisa mengenkripsi angka atau karakter non-huruf dengan baik.

5. Hill Cipher
   - Implementasi Algoritma menggunakan Python

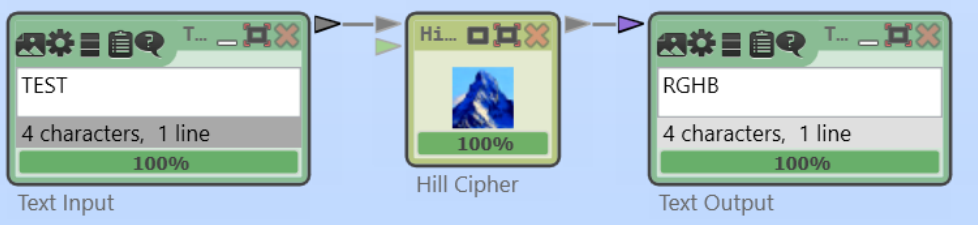
```python
import numpy as np

def hill_encrypt(text, key):
    text = text.upper().replace(' ', '')
    n = int(len(key) ** 0.5)
    key = np.array(key).reshape(n, n)
    result = ''
    for i in range(0, len(text), n):
        block = [ord(c) - 65 for c in text[i:i+n]]
        # jika blok terakhir kurang panjang, tambahkan 'X'
        while len(block) < n:
            block.append(ord('X') - 65)
        cipher = np.dot(key, block) % 26
        result += ''.join(chr(c + 65) for c in cipher)
    return result

print(hill_encrypt('TEST', [3, 3, 2, 5]))
```

```
RGHB
```

   - Implementasi Cipher menggunakan CrypTool



   - Analisis Kekuatan dan Kelemahan

     Algoritma ini menggunakan konsep matriks dan aljabar linear hal itu menjadikan hasil enkripsinya terlihat sangat acak dan lebih sulit dipecahkan dengan metode analisis frekuensi biasa. Tetapi algoritma ini harus berbentuk matriks yang bisa dibalik agar pesan bisa di dekripsikan, jika ada beberapa pasangan plaintext dan ciphertext yang diketahui kunci bisa dihitung lewat persamaan linear.