

**LAPORAN PRAKTIKUM  
IMPLEMENTASI ALGORITMA CIPHER KLASIK PADA  
PYTHON**



**Disusun oleh :**

**Syifa Sopian Nurdiansyah (20123049)**

**Mayasari Agustina (20123060) (rekan)**

**DIGITECH UNIVERSITY**

**S1 INFORMATIKA**

**2025/2026**

## A. Tujuan Praktikum

Tujuan dari praktikum ini adalah Untuk memahami dan mengimplementasikan berbagai metode cipher klasik seperti caesar cipher, affine cipher, vigenere cipher, playfair cipher dan hill cypher menggunakan bahasa Python, serta membandingkan hasilnya dengan sebuah tools cryptool.

## B. Dasar Teori

### 1. Enkripsi dan Deskripsi

Enkripsi atau *Encryption* merupakan sebuah proses untuk membuat suatu susunan acak dari teks yang dapat dibaca oleh manusia yang biasa disebut sebagai *plaintext* menjadi teks yang tidak dapat dibaca oleh manusia dan hanya dimengerti oleh sistem saja. Teks hasil dari enkripsi disebut dengan *ciphertext*.

Sementara itu, Dekripsi atau *Decryption* merupakan proses yang mengubah teks acak atau *ciphertext* menjadi teks biasa atau *plaintext*. Dekripsi ini hanya bisa terjadi jika orang memiliki atau diberi akses untuk melihat data yang dienkripsi.[1]

### 2. Caesar Cipher

Caesar Cipher merupakan sebuah metode *cipher* yang menggunakan pergeseran urutan huruf alfabet ke kiri atau ke kanan dengan jumlah geseran tertentu yang biasa disebut shift. Algoritma ini menggunakan nama Caesar karena terinspirasi oleh Julius Caesar seorang Kaisar Romawi yang menggunakan algoritma ini saat berkomunikasi dengan para petinggi negara agar informasi tidak bocor.[2]

Cara kerja dari metode caesar ini cukup mudah, yaitu dengan melakukan pergeseran huruf sebanyak yang dibutuhkan. Contohnya huruf A dengan jumlah shift = 3 maka ciphertextnya adalah D.

### 3. Affine Cipher

Affine cipher merupakan perluasan dari algoritma caesar cipher yang diperoleh dengan mengalikan *plaintexts* dengan suatu bilangan  $m$  yang relatif prima dengan nilai pergeseran  $b$ , kemudian hasilnya dijumlahkan dengan nilai pergeseran  $b$ . Affine cipher menggunakan rumus  $C = (mP + b) \bmod n$ , untuk melakukan enkripsi. Sementara rumus untuk melakukan dekripsi nya sebagai berikut yaitu  $P = m^{-1} (C - b) \bmod n$ . [3]

#### 4. Vigenere Cipher

Vigenere cipher adalah metode enkripsi dan deskripsi teks alfabet dengan menggunakan serangkaian caesar cipher yang berbeda berdasarkan huruf dari kata kunci dan merupakan bentuk substitusi *polyalphabetic* yang sederhana. Karakter yang digunakan dalam Vigenere Cipher yaitu A, B, C, ..., Z dan dikonversi kedalam angka 0, 1, 2, ..., 25.[4]

#### 5. Playfair Cipher

Playfair cipher, Playfair box, atau Wheatstone–Playfair cipher merupakan teknik enkripsi simetris yang manual dan merupakan cipher penggantian digram literal pertama. Skema ini diciptakan pada tahun 1854 oleh Charles Wheatstone, tetapi dinamakan menurut Lord Playfair karena perannya dalam mempromosikan penggunaannya.[5]

#### 6. Hill Cipher

Hill cipher adalah sandi substitusi poligrafik berbasis aljabar linear. Setiap huruf direpresentasikan oleh suatu angka modulo 26. Skema sederhana  $A = 0, B = 1, \dots, Z = 25$  seringkali digunakan, tetapi ini bukanlah fitur esensial untuk cipher. Untuk mengenkripsi pesan, setiap blok  $n$  huruf (dianggap sebagai vektor  $n$ -komponen) dikalikan dengan matriks  $n \times n$  yang dapat dibalik, dengan modulus 26. Untuk mendekripsi pesan, setiap blok dikalikan dengan invers matriks yang digunakan untuk enkripsi. Matriks yang digunakan untuk enkripsi adalah kunci sandi, dan harus dipilih secara acak dari himpunan matriks  $n \times n$  yang dapat dibalik (modulo 26). [6]

### C. Alat dan Bahan

- Algoritma : Caesar cipher, Affine cipher, Vigenere cipher, Playfair cipher, Hill cypher.
- Bahasa pemrograman : python, sebagai wadah untuk implementasi algoritma cipher klasik.
- Library: Tkinter, NumPy, Math.
- Perangkat : Laptop/PC ASUS expertbook B1402CB Intel Core I3 gen-12
- Editor teks : Visual Studio Code.

## D. Langkah-Langkah Praktikum

Pertama siapkan software yang dibutuhkan yaitu text editor seperti Visual Studio Code dan juga bahasa program python. Kemudian membuat file untuk implementasi algoritma.

a. Menyiapkan library yang dibutuhkan

```
import math
import numpy as np
import tkinter as tk
from tkinter import ttk, messagebox, filedialog
```

Library math digunakan untuk proses matematis dalam python. NumPy digunakan supaya pengolahan data numerik secara efisien di Python, khususnya pada metode Hill cipher. Tkinter dibuat untuk membuat GUI sederhana supaya program dapat berjalan secara lebih interaktif. Ketiga library ini biasanya sudah tersedia secara default dan bisa digunakan langsung tanpa harus mengunduhnya terlebih dahulu.

b. Implementasi algoritma Caesar cipher

```
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)
```

Dalam algoritma caesar cipher digunakan SHIFT untuk pergeseran huruf sehingga enkripsi dapat dilakukan. “result += chr((ord(char) - 65 + shift) % 26 + 65)” ini merupakan bagian pentingnya dimana:

- ord(char) akan mengubah huruf menjadi angka ASCII.
- ord(char) - 65 mengubah ‘A’ jadi 0, ‘B’ jadi 1, dst.
- + shift akan menggeser huruf sejumlah nilai shift.
- % 26 memastikan hasil tetap dalam rentang 0–25 (jumlah huruf alfabet).
- + 65 akan mengubah kembali ke kode ASCII huruf besar.
- chr(...) akan mengonversi angka ASCII kembali ke huruf.

- Dekripsi dilakukan dengan membalikkan nilai shift (-shift)
- c. Implementasi algoritma 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

def affine_decrypt(text, a, b):
    result = ""
    if math.gcd(a, 26) != 1:
        return "Error: Nilai 'a' harus coprime dengan 26!"
    for i in range(26):
        if (a * i) % 26 == 1:
            a_inv = i
            break
    for char in text.upper():
        if char.isalpha():
            result += chr(((a_inv * ((ord(char) - 65) - b)) % 26) + 65)
        else:
            result += char
    return result
```

- Fungsi `affine_encrypt` dan `affine_decrypt` memiliki 3 parameter yaitu `text` untuk plaintext, sementara `a` dan `b` untuk kunci enkrip.
  - “`result += chr(((a * (ord(char) - 65) + b) % 26) + 65)`” ini merupakan kode atau rumus untuk melakukan *ciphering* terutama pada bagian  $a * (...) + b$ .
  - (`if math.gcd(a, 26) != 1`) digunakan untuk mengecek apakah nilai `a` coprime dengan 26
  - “`chr(((a_inv * ((ord(char) - 65) - b)) % 26) + 65)`” ini merupakan rumus untuk melakukan dekrip dimana disini menggunakan “`a_inv`” sebagai inversi dari nilai `a`.
- d. Implementasi algoritma Vigenere cipher

```
def vigenere_encrypt(text, key):
    result = ""
    key = key.upper()
    key_index = 0
    for char in text.upper():
        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(text, key):
    result = ""
    key = key.upper()
    key_index = 0
    for char in text.upper():
        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
```

- Fungsi dari vigenere ini cipher ini memiliki 2 parameter yaitu text untuk plaintext, dan key untuk kunci, misalnya key = KEYLOCK.
- “ $\text{shift} = \text{ord}(\text{key}[\text{key\_index} \% \text{len}(\text{key})]) - 65$ ” disini dilakukan perhitungan nilai pergeseran dari huruf kunci.
- “ $\text{chr}(((\text{ord}(\text{char}) - 65 + \text{shift}) \% 26) + 65)$ ” ini merupakan model matematis yang digunakan dalam enkripsi vigenere.
- “ $\text{chr}(((\text{ord}(\text{char}) - 65 - \text{shift}) \% 26) + 65)$ ” dalam dekrip, huruf ciphertext dikurangi dengan shift atau kunci.

e. Implementasi algoritma Playfair cipher

```
def generate_playfair_matrix(key):
    key = "".join(dict.fromkeys(key.upper().replace("J", "I")))
    alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    matrix = []
    for char in key + alphabet:
        if char not in matrix:
            matrix.append(char)
    return [matrix[i:i+5] for i in range(0, 25, 5)]

def find_position(matrix, char):
    for i, row in enumerate(matrix):
        if char in row:
            return i, row.index(char)
    return None
```

- Fungsi `generate_playfair_matrix` digunakan untuk membuat matriks 5x5 yang dibutuhkan dalam metode playfair cipher ini.
- Sementara Fungsi `find_position` digunakan untuk mencari posisi (baris dan kolom) suatu huruf dalam tabel.

```
def playfair_prepare(text):
    text = text.upper().replace("J", "I")
    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
    return pairs
```

- Ini merupakan Fungsi untuk mempersiapkan teks sebelum di enkripsi.

- Teks akan dipecah menjadi 2 huruf , jika ada 2 huruf sama maka huruf akhirnya akan digantikan dengan huruf X. Begitu juga apabila hurufnya berjumlah ganjil.

```
def playfair_encrypt(text, key):
    matrix = generate_playfair_matrix(key)
    pairs = playfair_prepare("".join([c for c in text.upper() if c.isalpha()]))
    result = ""
    for a, b in pairs:
        row1, col1 = find_position(matrix, a)
        row2, col2 = find_position(matrix, b)
        if row1 == row2:
            result += matrix[row1][(col1 + 1) % 5] + matrix[row2][(col2 + 1) % 5]
        elif col1 == col2:
            result += matrix[(row1 + 1) % 5][col1] + matrix[(row2 + 1) % 5][col2]
        else:
            result += matrix[row1][col2] + matrix[row2][col1]
    return result

def playfair_decrypt(text, key):
    matrix = generate_playfair_matrix(key)
    pairs = playfair_prepare(text)
    result = ""
    for a, b in pairs:
        row1, col1 = find_position(matrix, a)
        row2, col2 = find_position(matrix, b)
        if row1 == row2:
            result += matrix[row1][(col1 - 1) % 5] + matrix[row2][(col2 - 1) % 5]
        elif col1 == col2:
            result += matrix[(row1 - 1) % 5][col1] + matrix[(row2 - 1) % 5][col2]
        else:
            result += matrix[row1][col2] + matrix[row2][col1]
    return result
```

- Ini merupakan Fungsi utama dalam enkripsi dan dekripsi playfair dengan dua parameter untuk plaintext dan key/kunci.
  - “matrix” dan “pairs” bertugas untuk memanggil fungsi yang membuat matriks, dan juga memanggil fungsi untuk pasangan huruf.
  - “if row1 == row2” kondisi ini apabila sama baris, maka geser ke kanan satu kolom.
  - “elif col1 == col2” kondisi ini apabila sama kolom, Maka geser kebawah satu kolom.
  - Sementara untuk “else” apabila beda baris dan beda kolom, maka tukar kolom antar baris.
  - Untuk fungsi dekripnya sama, namun hanya arah pergeserannya saja yang dibalik.
- f. Implementasi algoritma Hill cipher



```

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 = np.array([[ord(text[i]) - 65], [ord(text[i+1]) - 65]])
        encrypted = np.dot(key_matrix, pair) % 26
        result += chr(encrypted[0][0] + 65) + chr(encrypted[1][0] + 65)
    return result

def hill_decrypt(text, key_matrix):
    det = int(np.round(np.linalg.det(key_matrix))) % 26
    det_inv = None
    for i in range(26):
        if (det * i) % 26 == 1:
            det_inv = i
            break
    if det_inv is None:
        return "Error: Determinan tidak memiliki invers modulo 26!"

    adj = np.round(det * np.linalg.inv(key_matrix)).astype(int) % 26
    inv_matrix = (det_inv * adj) % 26

    result = ""
    for i in range(0, len(text), 2):
        pair = np.array([[ord(text[i]) - 65], [ord(text[i+1]) - 65]])
        decrypted = np.dot(inv_matrix, pair) % 26
        result += chr(int(decrypted[0][0]) + 65) + chr(int(decrypted[1][0]) + 65)
    return result

```

- Hill cipher menggunakan metode yang sama dengan playfair, yaitu membagi kata menjadi dua huruf, bila ada huruf yang sama atau pasangan bernilai ganjil/satu, maka akan diganti atau ditambahkan huruf X.
  - “encrypted = np.dot(key\_matrix, pair) % 26” namun dalam bagian ini, huruf huruf pasangan dikalikan dengan key matrix dan ambil hasilnya mod 26.
  - Metode dekripsinya sedikit rumit. untuk dekripsi, diperlukan matriks invers modulo 26 dari kunci.
- g. Membuat fitur save file ke txt

```

def export_to_file(metode, mode, teks_asli, hasil, shift=None, a=None, b=None):
    file_path = filedialog.asksaveasfilename(
        title="Simpan hasil sebagai...",
        defaultextension=".txt",
        filetypes=[("Text Files", "*.txt"), ("All Files", "*.*")]
    )
    if not file_path:
        return

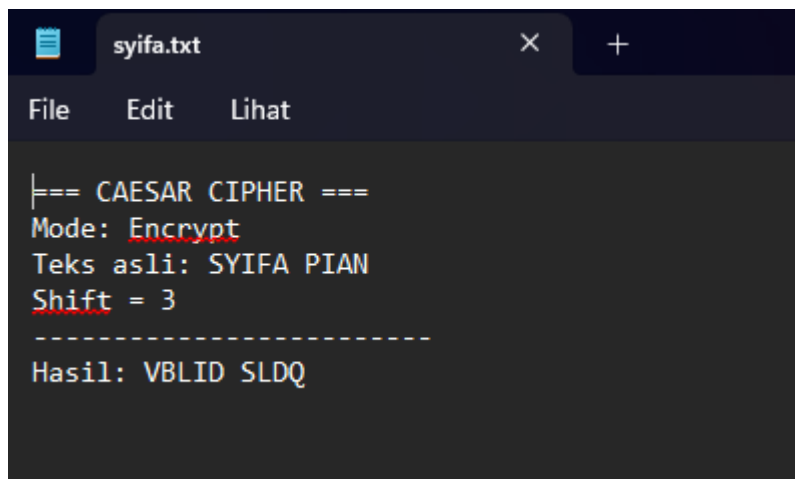
    with open(file_path, "w", encoding="utf-8") as f:
        f.write(f"=== {metode.upper()} ===\n")
        f.write(f"Mode: {mode}\n")
        f.write(f"Teks asli: {teks_asli}\n")
        if metode == "Caesar Cipher":
            f.write(f"Shift = {shift}\n")
        elif metode == "Affine Cipher":
            f.write(f"a = {a}\n")
            f.write(f"b = {b}\n")
        f.write(f"- " * 25 + "\n")
        f.write(f"Hasil: {hasil}\n")

    messagebox.showinfo("Berhasil", f"Hasil berhasil disimpan ke:\n{file_path}")

```



Contoh hasil output .txt



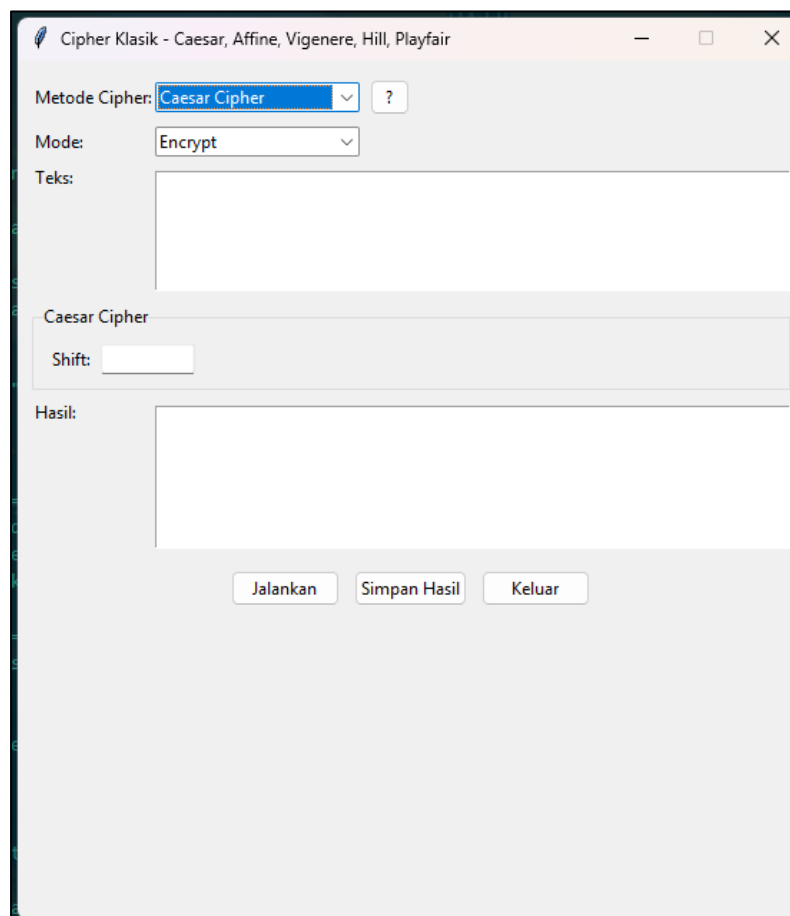
```
syifa.txt
File Edit Lihat

|=== CAESAR CIPHER ===
Mode: Encrypt
Teks asli: SYIFA PIAN
Shift = 3
-----
Hasil: VBLID SLDQ
```

h. Fungsi utama untuk cipher

Setelah semua algoritma dan fitur di implementasikan. Selanjutnya buat GUI sederhana menggunakan Tkinter GUI.

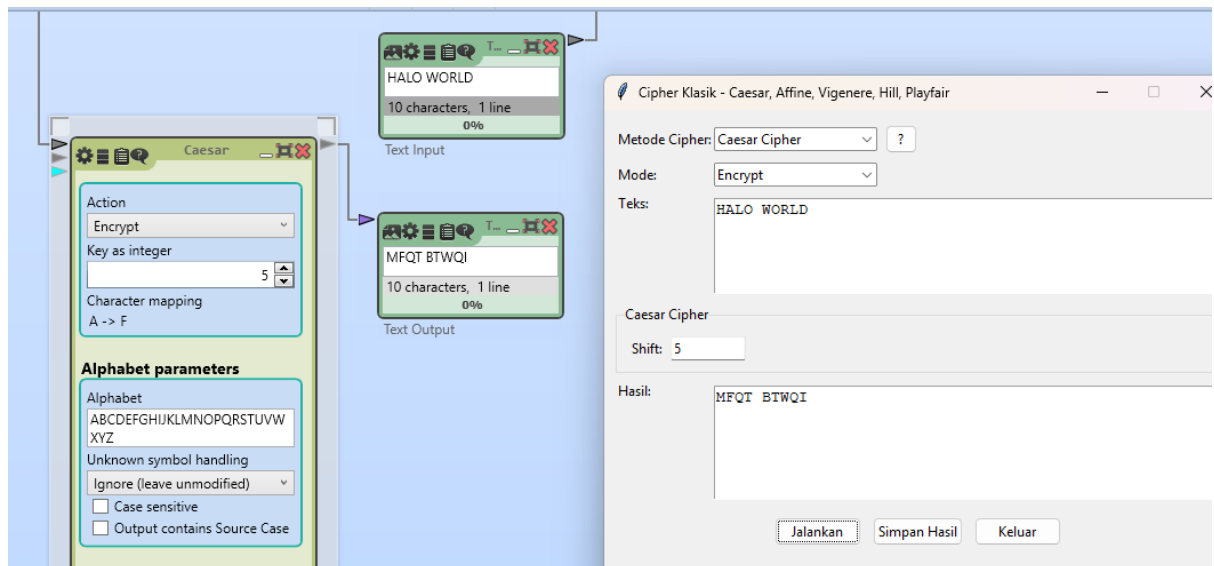
Contoh GUI sederhana dari program



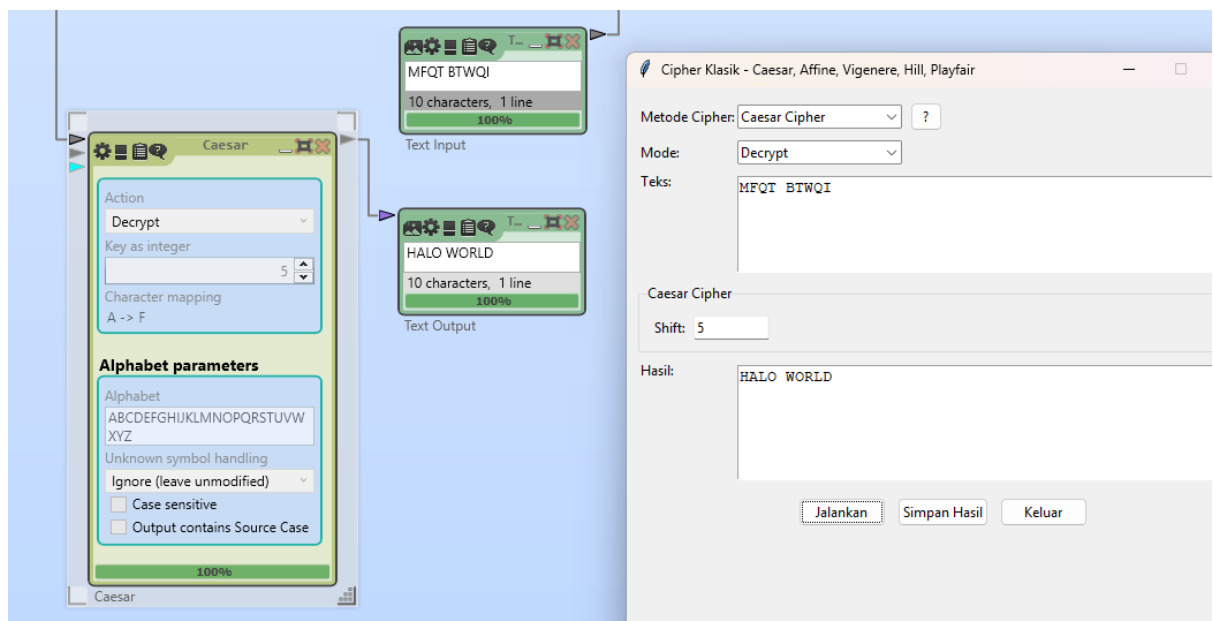
## E. Hasil dan Pembahasan

Setelah program berhasil dibuat, langkah selanjutnya adalah dengan melakukan uji coba terhadap aplikasi dengan melakukan enkripsi dan deskripsi, juga dilakukan perbandingan dengan salah satu *software cryptool*.

### a. Caesar cipher

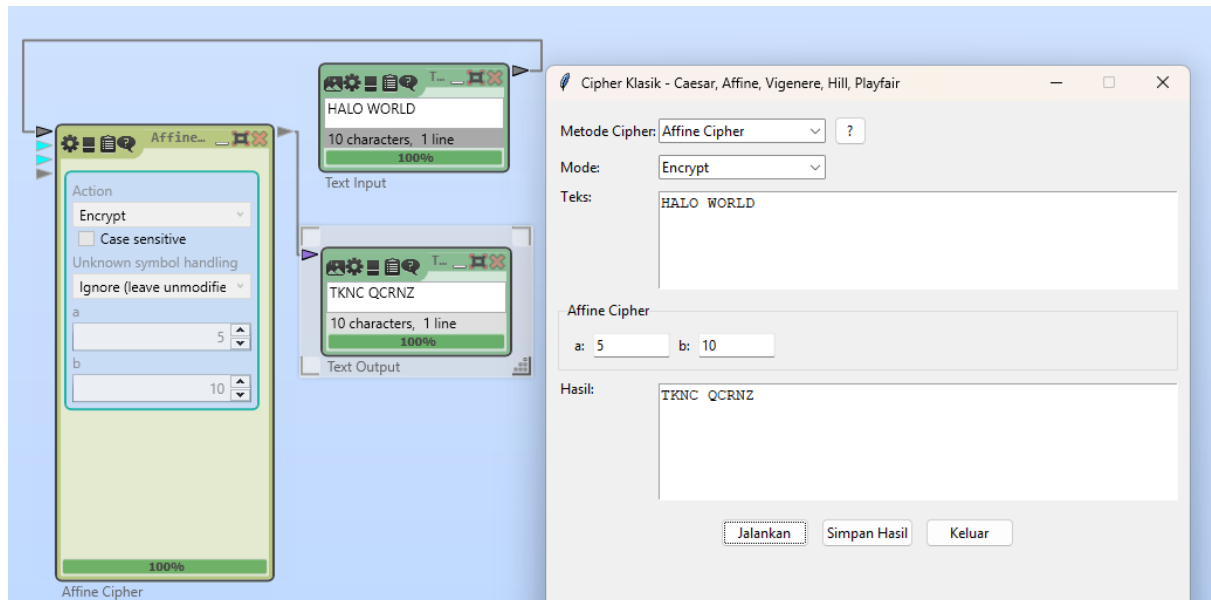


Dalam enkripsi, program berjalan dengan lancar. Dengan shift = 5, *plaintext* “HALO WORLD” di enkrip menjadi “MFQT BTWQI”. Kedua hasil baik dari program yang dibuat, dan cryptool menunjukkan hasil yang sama dan benar.

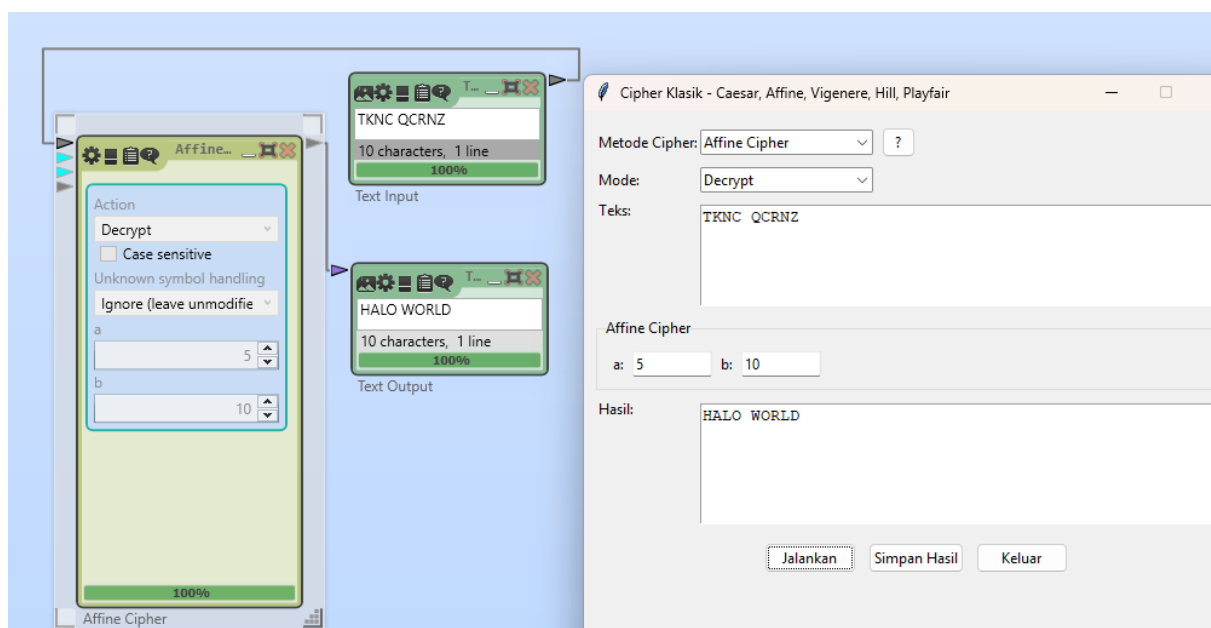


Dalam proses dekrip juga terlihat berjalan dengan lancar dan baik. *Ciphertext* sebelumnya kembali lagi sama seperti *plaintext* awal. Kelemahan dari metode ini bahwa metode ciphernya cukup simple dan mudah dipecahkan dengan *brute force* dan analisis frekuensi.

b. Affine cipher



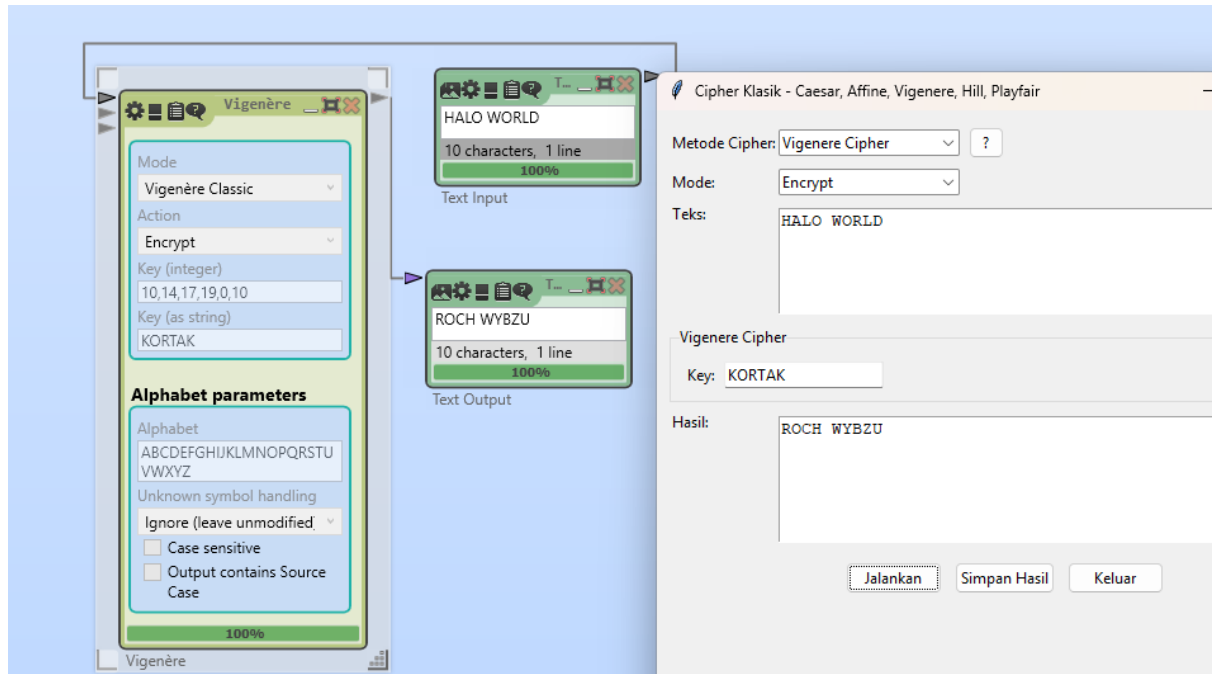
Dalam proses enkripsi, kedua program, baik buatan maupun *cryptool* sama-sama berhasil meng-enkrip *plaintext* “HALO WORLD” dengan nilai kunci  $a = 5$  dan  $b = 10$ , menjadi “TKNC QCRNZ”.



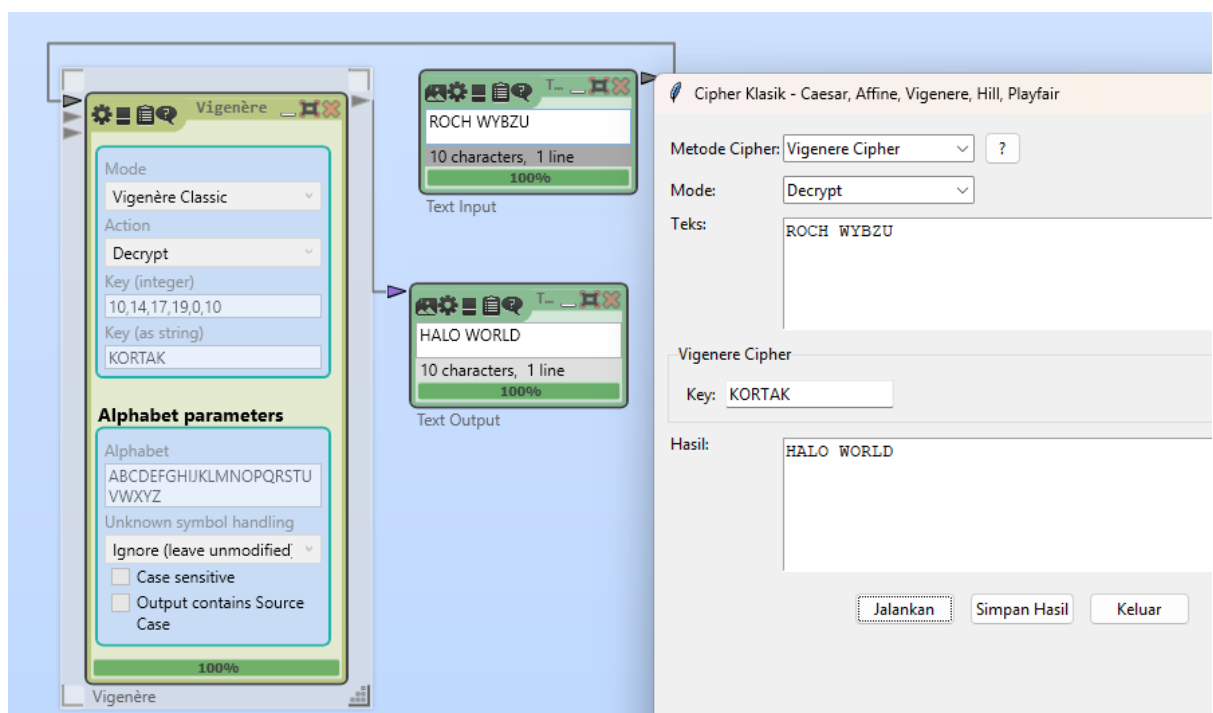
Pada proses dekrip juga keduanya berhasil membalikkan *ciphertext* sebelumnya menjadi *plaintext* seperti di awal. Kelemahannya sama seperti metode caesar, sangat mudah di

pecahkan dengan metode analisis frekuensi, serta tuang kuncinya yang bergantung pada nilai a dan b, Dan juga nilai a harus relatif prima terhadap nilai 26.

c. Vigenere cipher

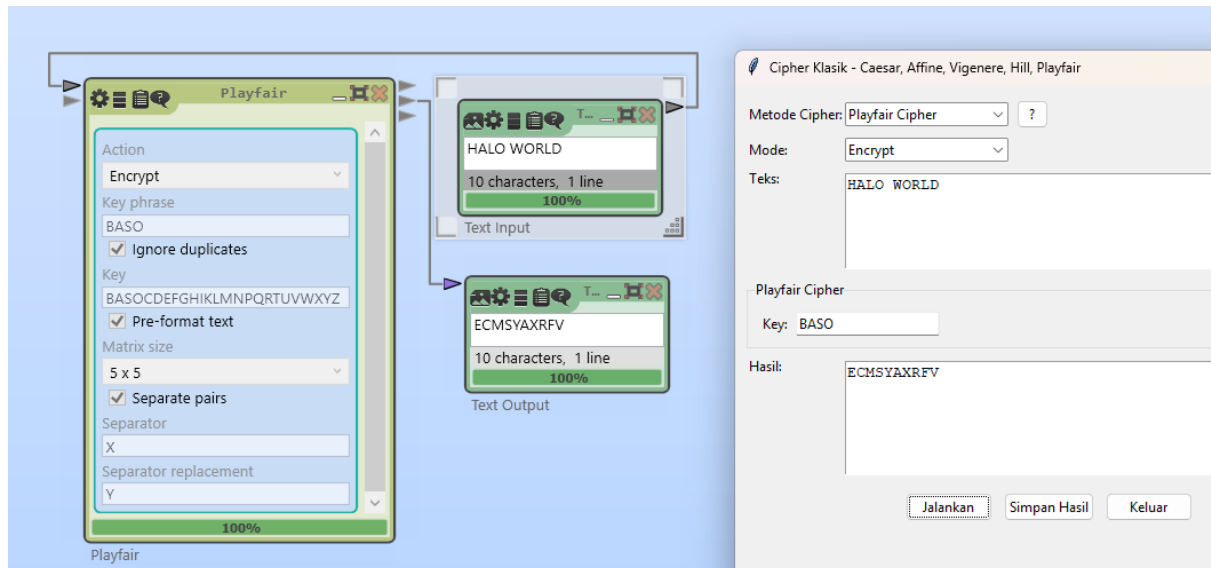


Pada proses enkripsi keduanya menunjukkan hasil yang sama, *plaintext* “HALO WORLD” di enkrip dengan kata kunci “KORTAK”, menghasilkan sebuah *ciphertext* “ROCH WYBZU”.

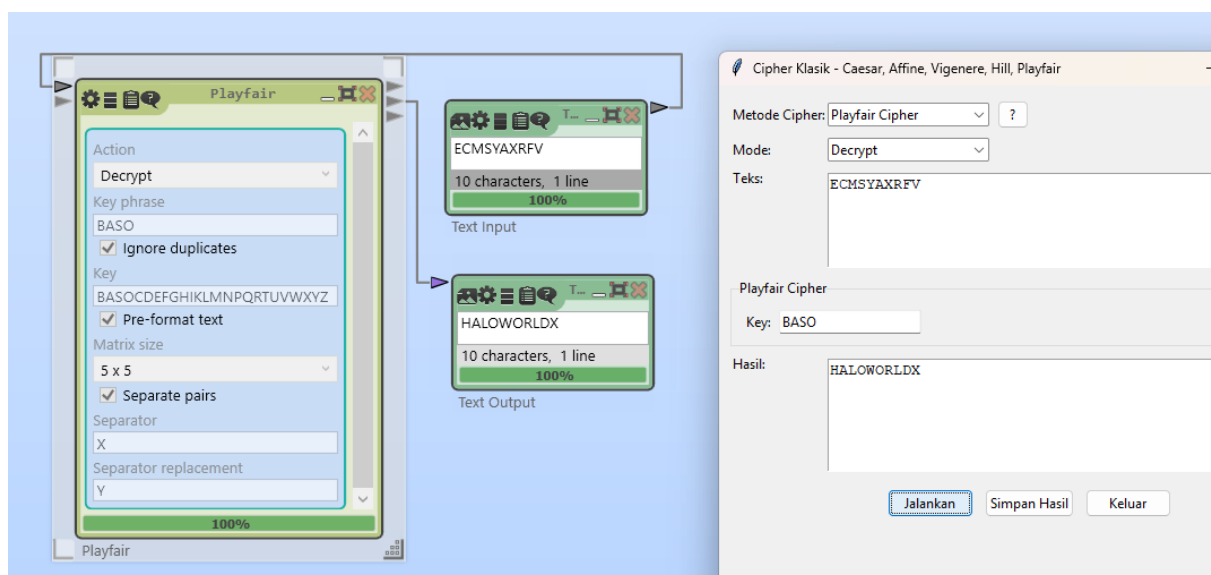


Dalam proses dekripsi juga keduanya berhasil membalikkan *ciphertext* menjadi kembali seperti semula. Kelemahan dari metode vigenere ini adalah Tidak tahan terhadap analisis frekuensi jika panjang kunci diketahui. Artinya rentan terhadap kuncinya sendiri.

#### d. Playfair cipher



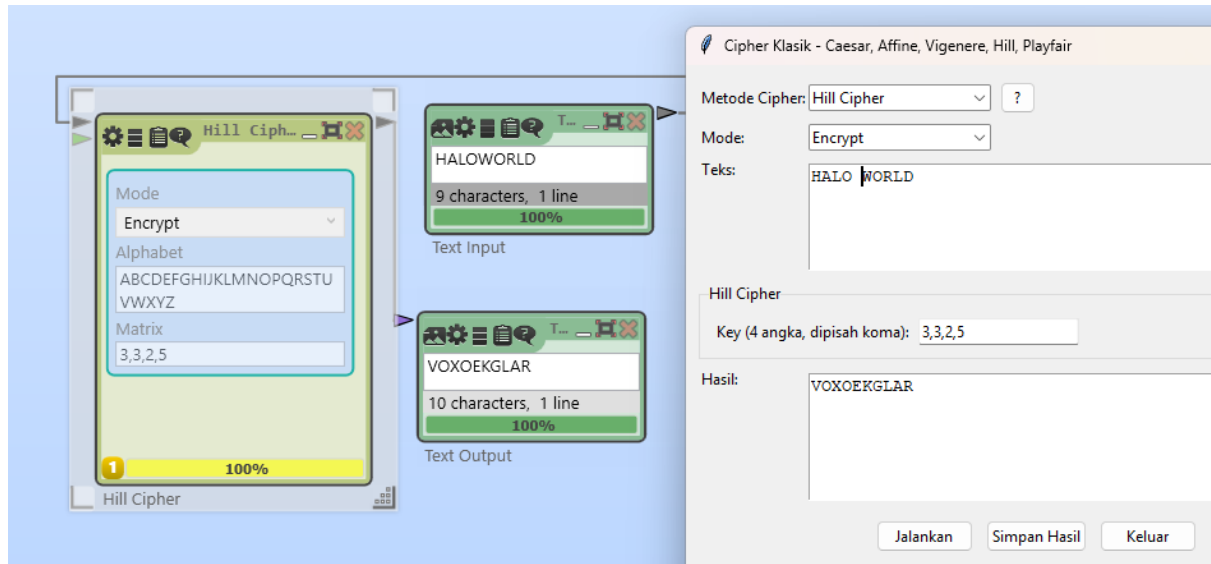
Pada proses enkripsi, keduanya berhasil meng-enkripsi teks “HALO WORLD” dengan kata kunci “BASO” menjadi “ECMSYAXRFV”.



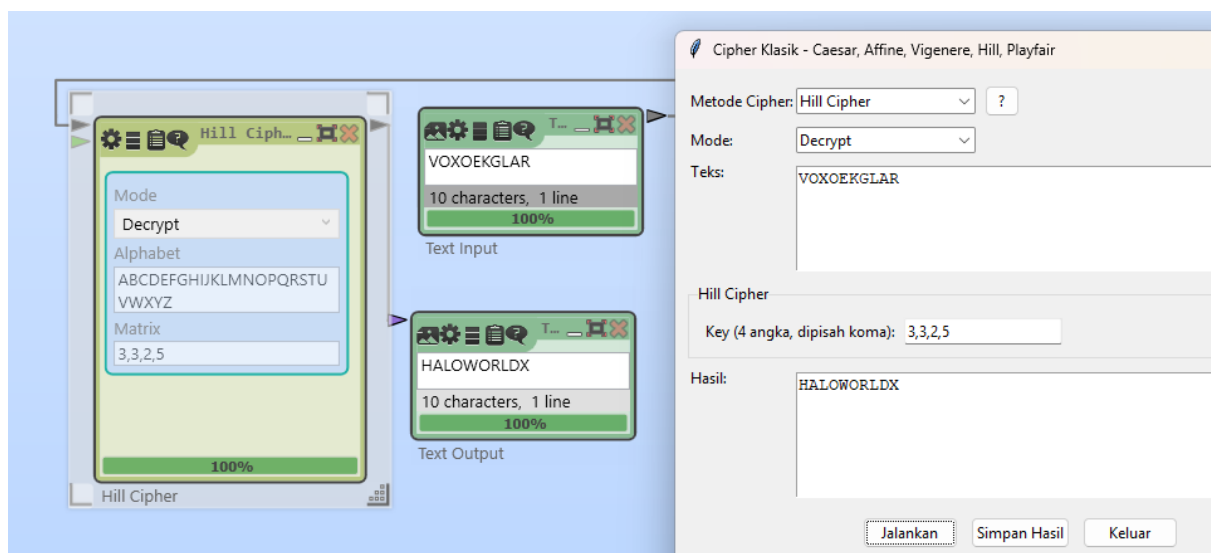
Pada proses dekripsi, keduanya juga berhasil mengembalikan teks menjadi seperti pada awalnya, namun karena huruf ganda dan spasi juga mempengaruhi hasil khususnya huruf ganda dan ganjil menjadi X, sehingga hasil dekripnya kurang sempurna, sehingga hasilnya menjadi “HALOWORLDX”. Kelemahan metode ini adalah pada huruf ganda atau huruf

yang tidak berpasangan. Serta masih bisa dipecahkan dengan frekuensi analisis meskipun sebenarnya cukup rumit juga.

e. Hill cipher



Dalam proses enkripsi Hill cipher, kedua program berhasil meng-enkripsi kata “HAL WORLD” dengan kunci matriks 2x2 “3,3,2,5” menjadi “VOXOEKGLAR”.



Sementara untuk proses dekripsinya juga, metode ini berhasil mengembalikan *ciphertext* menjadi seperti di awal. Dan permasalahannya sama dengan metode playfair, yaitu ketentuan huruf ganda. Kelemahannya adalah memerlukan invers matriks modulo 26. Sehingga pemilihan kunci-nya harus berhati-hati. Serta rentan terhadap *plaintext* yang diketahui atau *known plaintext attack*.



cipher	plaintext	key	enkrip	dekrip	kelemahan
Caesar	HALO WORLD	5	MFQT BTWQI	HALO WORLD	Rentan terhadap brute force
Affine	HALO WORLD	5, 10	TKNC QCRNZ	HALO WORLD	Rentan terhadap frekuensi analisis
Vigenere	HALO WORLD	KORTAK	ROCH WYBZU	HALO WORLD	Kunci yang diketahui
Playfair	HALO WORLD	BASO	ECMSYAXRFV	HALOWORLDX	Huruf ganda dan analisis frekuensi
Hill	HALO WORLD	3,3,2,5	VOXOEKGLAR	HALOWORLDX	Rentan terhadap known plaintext.

## F. Kesimpulan

Kesimpulannya bahwa metode *cipher* klasik dalam kriptografi sudah cukup baik apabila digunakan pada zamannya. Di zaman sekarang, metode-metode dari *cipher* klasik ini mungkin cocok untuk dijadikan sebagai pelajaran mendasar saja untuk kriptografi. Karena banyaknya kelemahan yang membuat metode-metode ini kurang cocok jika digunakan sebagai alat untuk keamanan enkripsi utama, khususnya kerentanannya terhadap metode analisis frekuensi. Namun terlepas dari itu, kita bisa saja menggunakan beberapa metode ini secara bersamaan atau kolaborasi metode dalam enkripsi, tujuannya supaya meningkatkan keamanan dari enkripsi supaya ciphertext tidak mudah di pecahkan dengan begitu saja.

## Referensi

- [1] R. Setiawan, “Konsep Enkripsi & Dekripsi untuk Keamanan Data,” Dicoding Blog. Diakses: 29 Oktober 2025. [Daring]. Tersedia pada: <https://www.dicoding.com/blog/enkripsi-untuk-keamanan-data/>
- [2] “Enkripsi Caesar’s Cipher: Penjelasan dan Cara Kerja Proteksi - Bobo.” Diakses: 29 Oktober 2025. [Daring]. Tersedia pada: <https://bobo.grid.id/read/083497828/enkripsi-caesars-cipher-penjelasan-dan-cara-kerja-proteksi>
- [3] I. N. Diana, “Algoritma Affine Cipher dan Modifikasi Affine Cipher, serta Kombinasinya dengan Cipher Transposisi Grup Simetri untuk Mengamankan Pesan Teks,” vol. 7, no. 1, 2022.
- [4] A. Amrulloh dan E. Ujianto, “Kriptografi Simetris Menggunakan Algoritma Vigenere Cipher,” 2019.
- [5] “Playfair cipher,” *Wikipedia*. 11 Oktober 2025. Diakses: 29 Oktober 2025. [Daring]. Tersedia pada: [https://en.wikipedia.org/w/index.php?title=Playfair\\_cipher&oldid=1316254736](https://en.wikipedia.org/w/index.php?title=Playfair_cipher&oldid=1316254736)
- [6] “Hill Cipher,” *GeeksforGeeks*. Diakses: 29 Oktober 2025. [Daring]. Tersedia pada: <https://www.geeksforgeeks.org/dsa/hill-cipher/>