LAPORAN HASIL TUGAS 1

Implementasi Algoritma Kriptografi Klasik

DIAJUKAN UNTUK MEMENUHI TUGAS MATA KULIAH

IF4020 – Kriptografi

SEMESTER I TAHUN 2021/2022

Disusun oleh :

Daru Bagus Dananjaya	**13519080**
Karel Renaldi	**13519180**

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG 2021

1. Source Code Program

affineCipher.py

```
import re

def textCleaning(text):
    text = text.upper()
    text = re.sub(r'\s*\d+\s*', '',text)
    text = re.sub(r'[^\w\s]', '', text)
    text = text.replace(' ', '')
    return text

def postProcess(text):
    text = [text[i:i+5] for i in range(0, len(text), 5)]
    text = ' '.join(text)

    return text

# Extended Euclidean Algorithm
def egcd(a, b):
    # example egcd(7,26) = 15
    #Basis
    if a == 0 :
        return b,0,1

    #Recursive
    gcd, x1, y1 = egcd(b%a, a)

    x = y1 - (b//a)*x1
    y = x1

    return gcd,x,y

def modinv(a, m):
    gcd, x, y = egcd(a, m)
    if gcd != 1:
        return None  # modular inverse doesn't exist
    else:
        return x % m


def affineEncrypt(text, key):
    # C = (a*P + b) % 26
    cipher = ''.join([ chr((( key[0]*(ord(t) - ord('A')) + key[1] ) % 26)
                  + ord('A')) for t in text.upper().replace(' ', '') ])

    cipher = postProcess(cipher)
    return cipher


def affineDecrypt(cipher, key):
    # P = (a^-1 * (C - b)) % 26
    plainText = ''.join([ chr((( modinv(key[0], 26)*(ord(c) - ord('A') -
key[1]))
                    % 26) + ord('A')) for c in cipher ])

    plainText = postProcess(plainText)

    return plainText
```

vignere_cipher.py

```
import re

alphabet = [chr(97 + i) for i in range(26)]

def clean_text(text: str) -> str:
    res = text

    # Convert to lowercase
    res = res.lower()

    # Remove whitespace
    res.strip()

    res = res.replace(" ", "")

    # Remove number
    res = ''.join([i for i in res if not i.isdigit()])

    # Remove punctuation
    res = re.sub(r'[^\w\s]', '', res)

    return res


def generate_key_standard(plain_text: str, key: str) -> str:
    if(len(key) >= len(plain_text)):
        return key[:len(key)]

    full_key: str = key
    for i in range(len(plain_text) - len(key)):
        full_key += key[i % len(key)]
    return full_key


def generate_key_auto(plain_text: str, key: str) -> str:
    if(len(key) >= len(plain_text)):
        return key[:len(key)]

    full_key: str = key
    for i in range(len(plain_text) - len(key)):
        full_key += plain_text[i]

    return full_key


def vignere_cipher_encrypt(plain_text: str, key: str) -> str:
    cipher_text = ""

    for i in range(len(plain_text)):
        curr_plain_text_num = ord(plain_text[i]) - ord('a')
        curr_key_text_num = ord(key[i]) - ord('a')
        curr_cipher_text_num = (curr_plain_text_num + curr_key_text_num) %
26

        cipher_text += alphabet[curr_cipher_text_num]

    return cipher_text

def vignere_cipher_decrypt(cipher_text: str, key: str) -> str:
    plain_text = ""

    for i in range(len(cipher_text)):
        curr_cipher_text_num = ord(cipher_text[i]) - ord('a')
        curr_key_text_num = ord(key[i]) - ord('a')
        curr_plain_text_num = (curr_cipher_text_num - curr_key_text_num) %
26

        plain_text += alphabet[curr_plain_text_num]

    return plain_text

def vignere_cipher_standard_encrypt(plain_text: str, key: str):
    plain_text = clean_text(plain_text)
    key = clean_text(key)
    full_key = generate_key_standard(plain_text, key)

    return vignere_cipher_encrypt(plain_text, full_key)

def vignere_cipher_standard_decrypt(cipher_text: str, key: str):
    cipher_text = clean_text(cipher_text)
    key = clean_text(key)
```

```python
    full_key = generate_key_standard(cipher_text, key)

    return vignere_cipher_decrypt(cipher_text, full_key)


def vignere_cipher_auto_key_encrypt(plain_text: str, key: str):
    plain_text = clean_text(plain_text)
    key = clean_text(key)
    full_key = generate_key_auto(plain_text, key)

    return vignere_cipher_encrypt(plain_text, full_key), full_key

def vignere_cipher_auto_key_decrypt(cipher_text: str, key: str):
    cipher_text = clean_text(cipher_text)
    key = clean_text(key)
    full_key = generate_key_standard(cipher_text, key)

    return vignere_cipher_decrypt(cipher_text, full_key)
```

fullVigenere.py

```python
import re
import random
import string

alphabetUppercase = list(string.ascii_uppercase)


def textCleaning(text):
    text = text.upper()
    text = re.sub(r'\s*\d+\s*', '', text)
    text = re.sub(r'[^\w\s]', '', text)
    text = text.replace(' ', '')
    return text


def postProcess(text):
    text = [text[i:i+5] for i in range(0, len(text), 5)]
    text = ' '.join(text)

    return text


def generateKey(text, key):
    key = list(key)
    if len(text) == len(key):
        return(key)
    else:
        for i in range(len(text)-len(key)):
            key.append(key[i % len(key)])
    retVal = "".join(key)
    retVal.upper()

    return(retVal)


def generateFullVigenereMatrix():
    matrix = []
    for i in range(26):
        isDuplicate = True
        while isDuplicate:
            tempAlpha = alphabetUppercase
            random.shuffle(tempAlpha)
            tempStr = ''.join(tempAlpha)
            if tempStr not in matrix:
                isDuplicate = False
        matrix.append(tempStr)

    return matrix


def encrypt(text, key, matrix):
    text = textCleaning(text)
    key = generateKey(text, key).upper()
    # text is cleaned

    cipher = ''

    for i in range(len(text)):
        idxKey = i % len(key)
        col = string.ascii_uppercase.index(text[i])
        row = string.ascii_uppercase.index(key[idxKey])

        cipher += matrix[row][col]

    cipher = postProcess(cipher)

    return cipher


def decrypt(cipher, key, matrix):
    cipher = textCleaning(cipher)
    key = generateKey(cipher, key).upper()
    # ciphertext is cleaned

    plaintext = ''
    for i in range(len(cipher)):
        idxKey = i % len(key)
        row = string.ascii_uppercase.index(key[idxKey])
        vRow = matrix[row]
        idxLetter = vRow.index(cipher[i])

        plaintext += string.ascii_uppercase[idxLetter]

    plaintext = postProcess(plaintext)

    return plaintext
```

extendedVigenere.py

```
import vignere_cipher as vc

BYTE_MAX = 256

def extended_vignere_cipher_encrypt(src_path: str, key: str, dest_path: str)
-> bool :
    try:
        f = open(src_path, 'rb')

        fileData = bytearray(f.read())
        newKey = vc.generate_key_standard(fileData, vc.clean_text(key))

        for idx, plainText in enumerate(fileData):
            fileData[idx] = (plainText + ord(newKey[idx])) % BYTE_MAX

        f.close()

        f = open(dest_path, 'wb')
        f.write(fileData)
        f.close()

        return True
    except Exception as e:
        return False

def extended_vignere_cipher_decrypt(src_path: str, key: str, dest_path: str)
-> str :
    try:
        f = open(src_path, 'rb')

        fileData = bytearray(f.read())
        newKey = vc.generate_key_standard(fileData, vc.clean_text(key))
        for idx, cipherText in enumerate(fileData):
            fileData[idx] = (cipherText - ord(newKey[idx])) % BYTE_MAX

        f.close()

        f = open(dest_path, 'wb')
        f.write(fileData)
        f.close()

        return True
    except:
        return False
```

hill_cipher.py

```
import numpy as np
import re

alphabet = "abcdefghijklmnopqrstuvwxyz"

global char_to_num, num_to_char
char_to_num = dict(zip(alphabet, range(len(alphabet))))
num_to_char = dict(zip(range(len(alphabet)), alphabet))


def clean_text(text: str) -> str:
    res = text

    # Convert to lowercase
    res = res.lower()

    # Remove whitespace
    res.strip()

    res = res.replace(" ", "")

    # Remove number
    res = ''.join([i for i in res if not i.isdigit()])

    # Remove punctuation
    res = re.sub(r'[^\w\s]', '', res)

    return res


def egcd(m, n):
    if m == 0:
        return n, 0, 1

    gcd, x_hat, y_hat = egcd(n % m, m)

    x = y_hat - (n // m) * x_hat
    y = x_hat

    return gcd, x, y


def modinv(a, m):
    """
        modinv is a function for calculate a^-1 mod m, this function will
return result and
        if error this function will return -inf.
    """

    gcd, x, _ = egcd(a, m)
    if gcd != 1:
        return None
    else:
        return x % m


def matrix_modulo_invers(matrix: np.ndarray, modulus: int = 26) ->
np.ndarray:
    matrix_determinant = int(np.round(np.linalg.det(matrix)))
    matrix_adjoint = np.round(
        matrix_determinant * np.linalg.inv(matrix)
    ).astype(int)
    modulo_invers_determinant = modinv(matrix_determinant % modulus,
modulus)

    if(not(modulo_invers_determinant)):
        return None

    matrix_result = modulo_invers_determinant * matrix_adjoint

    return (matrix_result % modulus)


def hill_cipher_encrypt(plain_text: str, key: np.ndarray, modulus=26) ->
str:
    cipher_text = ""
    plain_text = clean_text(plain_text)

    n, _ = key.shape
    plain_text_num = [char_to_num[el] for el in plain_text]
    plain_text_matrix = []
    for i in range(0, len(plain_text), n):
        plain_text_arr = []
        for j in range(i, i + n):
            plain_text_arr.append(plain_text_num[j])

        plain_text_matrix.append(plain_text_arr)

    plain_text_matrix = np.array(plain_text_matrix)
    for el in plain_text_matrix:
        el = el.reshape(-1, 1)

        curr_res = np.dot(key, el) % modulus
        curr_res = curr_res.flatten()

        for num in curr_res:
            cipher_text += num_to_char[num]

    return cipher_text


def hill_cipher_decrypt(cipher_text: str, key: np.ndarray, modulus=26) ->
str:
    plain_text = ""

    key_invers = matrix_modulo_invers(key)

    n, _ = key.shape
    cipher_text_num = [char_to_num[el] for el in cipher_text]
    cipher_text_matrix = []

    for i in range(0, len(cipher_text), n):
        cipher_text_arr = []
        for j in range(i, i + n):
            cipher_text_arr.append(cipher_text_num[j])

        cipher_text_matrix.append(cipher_text_arr)

    cipher_text_matrix = np.array(cipher_text_matrix)
    for el in cipher_text_matrix:
        el = el.reshape(-1, 1)

        curr_res = np.dot(key_invers, el) % modulus
        curr_res = curr_res.flatten()

        for num in curr_res:
            plain_text += num_to_char[num]

    return plain_text
```

playfairCipher.py

```python
import re


def textCleaning(text):
    text = text.upper()
    text = re.sub(r'\s*\d+\s*', '', text)
    text = re.sub(r'[^\w\s]', '', text)
    text = text.replace(' ', '')
    return text


def postProcess(text):
    text = [text[i:i+5] for i in range(0, len(text), 5)]
    text = ' '.join(text)

    return text


def matrix(x, y, initial):
    return [[initial for i in range(x)] for j in range(y)]


def locateIndex(c, playFairMatrix):  # get location of each character
    loc = list()
    if c == 'J':
        c = 'I'
    for i, j in enumerate(playFairMatrix):
        for k, l in enumerate(j):
            if c == l:
                loc.append(i)
                loc.append(k)
                return loc


def encrypt(text, playFairMatrix):
    text = textCleaning(text)
    cipher = ''
    i = 0
    for s in range(0, len(text)+1, 2):
        if s < len(text)-1:
            if text[s] == text[s+1]:
                text = text[:s+1]+'X'+text[s+1:]

    if len(text) % 2 != 0:
        text = text[:]+'X'
    # print("CIPHER TEXT:", end='')

    while i < len(text):
        loc = list()
        loc = locateIndex(text[i], playFairMatrix)
        loc1 = list()
        loc1 = locateIndex(text[i+1], playFairMatrix)
        if loc[1] == loc1[1]:
            cipher += playFairMatrix[(loc[0]+1)%5][loc[1]] +
playFairMatrix[(loc1[0]+1)%5][loc1[1]]
            #
print("{}{}".format(playFairMatrix[(loc[0]+1)%5][loc[1]],playFairMatrix[(loc
1[0]+1)%5][loc1[1]]),end=' ')
        elif loc[0] == loc1[0]:
            cipher += playFairMatrix[loc[0]][(loc[1]+1) % 5] +
playFairMatrix[loc1[0]][(loc1[1]+1) % 5]
            #
print("{}{}".format(playFairMatrix[loc[0]][(loc[1]+1)%5],playFairMatrix[loc1
[0]][(loc1[1]+1)%5]),end=' ')
        else:
            cipher += playFairMatrix[loc[0]][loc1[1]] +
playFairMatrix[loc1[0]][loc[1]]
            #
print("{}{}".format(playFairMatrix[loc[0]][loc1[1]],playFairMatrix[loc1[0]][
loc[1]]),end=' ')
        i = i+2

    cipher = postProcess(cipher)

    return cipher


def decrypt(cipher, playFairMatrix):  # decryption
    cipher = textCleaning(cipher)
    plainText = ''
    # print("PLAIN TEXT:", end=' ')
    i = 0
    while i < len(cipher):
        loc = list()
        loc = locateIndex(cipher[i], playFairMatrix)
        loc1 = list()
        loc1 = locateIndex(cipher[i+1], playFairMatrix)
        if loc[1] == loc1[1]:
            plainText += playFairMatrix[(loc[0]-1) % 5][loc[1]] + \
                playFairMatrix[(loc1[0]-1) % 5][loc1[1]]
        elif loc[0] == loc1[0]:
            plainText += playFairMatrix[loc[0]][(loc[1]-1) % 5] +
playFairMatrix[loc1[0]][(loc1[1]-1) % 5]
        else:
            plainText += playFairMatrix[loc[0]][loc1[1]] +
playFairMatrix[loc1[0]][loc[1]]
        i = i+2

    plainText = postProcess(plainText)

    return plainText


def generatePlayfairSquare(key):
    key = key.upper()
    result = list()

    for c in key:  # storing key
        if c not in result:
            if c == 'J':  # replacing j with i
                result.append('I')
            else:
                result.append(c)

    flag = 0

    for i in range(65, 91):  # storing other character
        if chr(i) not in result:
            if i == 73 and chr(74) not in result:
                result.append("I")
                flag = 1
            elif flag == 0 and i == 73 or i == 74:
                pass
            else:
                result.append(chr(i))
    k = 0
    my_matrix = matrix(5, 5, 0)  # initialize matrix
    for i in range(0, 5):  # making matrix
        for j in range(0, 5):
            my_matrix[i][j] = result[k]
            k += 1

    return my_matrix
```

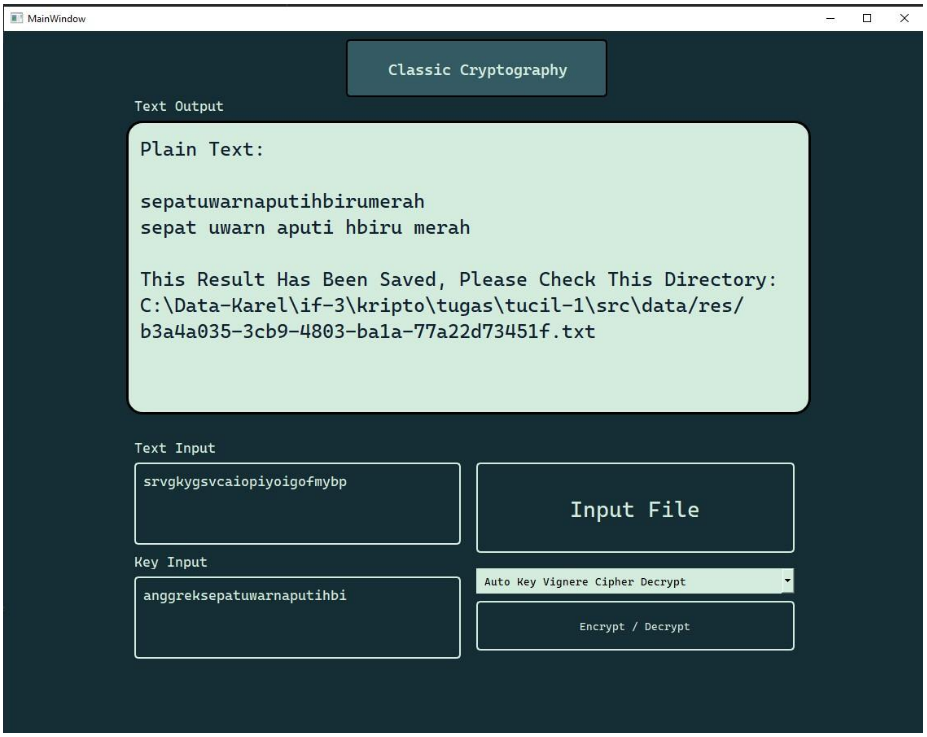
classic-crypto.py

```
import json
from PyQt5 import QtCore, QtWidgets
import sys
import numpy as np
import affineCipher
import extendedVigenere
import fullVigenere
import hill_cipher
import playfairCipher
import vignere_cipher

import uuid
import os


# sys.path.append('/src')


class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(1129, 868)
        MainWindow.setStyleSheet("background-color: rgb(21, 45, 53);")
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.label = QtWidgets.QLabel(self.centralwidget)
        self.label.setGeometry(QtCore.QRect(420, 10, 321, 71))
        self.label.setStyleSheet("background-color: rgb(52, 91, 99);\n"
                                 "font-family: \"Cascadia Code
SemiBold\";\n"
                                 "border: 2px solid black;\n"
                                 "border-radius: 5px;\n"
                                 "font-size: 10px;\n"
                                 "color: #D4ECDD;")
        self.label.setObjectName("label")
        self.outputTextArea = QtWidgets.QPlainTextEdit(self.centralwidget)
        self.outputTextArea.setGeometry(QtCore.QRect(150, 110, 841, 361))
        self.outputTextArea.setStyleSheet("background-color: #D4ECDD;\n"
                                          "font-size: 15px;\n"
                                          "font-weight: bold;\n"
                                          "border-radius: 20px;\n"
                                          "border: 3px solid black;\n"
                                          "font: 75 18pt \"Cascadia
Code\";\n"
                                          "padding: 10px;\n"
                                          "color: #112031")
        self.outputTextArea.setPlainText("")
        self.outputTextArea.setObjectName("outputTextArea")
        self.cipherAlgorithmComboBox =
QtWidgets.QComboBox(self.centralwidget)
        self.cipherAlgorithmComboBox.setGeometry(
            QtCore.QRect(580, 660, 391, 31))
        self.cipherAlgorithmComboBox.setStyleSheet("background-color:
rgb(212, 236, 221);\n"
                                                   "font: 75 10pt \"Cascadia
Code\";\n"
```

```
                                                   "padding-left: 10px;\n"
                                                   "border: none;")

self.cipherAlgorithmComboBox.setObjectName("cipherAlgorithmComboBox")
        self.cipherAlgorithmComboBox.addItem("")
        self.cipherAlgorithmComboBox.addItem("")
        self.cipherAlgorithmComboBox.addItem("")
        self.cipherAlgorithmComboBox.addItem("")
        self.cipherAlgorithmComboBox.addItem("")
        self.cipherAlgorithmComboBox.addItem("")
        self.cipherAlgorithmComboBox.addItem("")
        self.cipherAlgorithmComboBox.addItem("")
        self.cipherAlgorithmComboBox.addItem("")
        self.cipherAlgorithmComboBox.addItem("")
        self.cipherAlgorithmComboBox.addItem("")
        self.cipherAlgorithmComboBox.addItem("")
        self.cipherAlgorithmComboBox.addItem("")
        self.cipherAlgorithmComboBox.addItem("")
        self.encryptDecryptButton =
QtWidgets.QPushButton(self.centralwidget)
        self.encryptDecryptButton.setGeometry(QtCore.QRect(580, 700, 391,
61))
        self.encryptDecryptButton.setStyleSheet("color: #D4ECDD;\n"
                                                "font: 75 10pt \"Cascadia
Code\";\n"
                                                "border: 2px solid
#D4ECDD;\n"
                                                "border-radius: 5px;")
        self.encryptDecryptButton.setObjectName("encryptDecryptButton")
        self.inputText = QtWidgets.QPlainTextEdit(self.centralwidget)
        self.inputText.setGeometry(QtCore.QRect(160, 530, 401, 101))
        self.inputText.setStyleSheet("border: 2px solid #D4ECDD;\n"
                                     "border-radius: 5px;\n"
                                     "color: rgb(212, 236, 221);\n"
                                     "padding: 5px;\n"
                                     "font: 75 13pt \"Cascadia Code\";")
        self.inputText.setPlainText("")
        self.inputText.setObjectName("inputText")
        self.inputFileButton = QtWidgets.QPushButton(self.centralwidget)
        self.inputFileButton.setGeometry(QtCore.QRect(580, 530, 391, 111))
        self.inputFileButton.setStyleSheet("color: #D4ECDD;\n"
                                           "font: 75 20pt \"Cascadia
Code\";\n"
                                           "border: 2px solid #D4ECDD;\n"
                                           "border-radius: 5px;")
        self.inputFileButton.setObjectName("inputFileButton")
        self.label_2 = QtWidgets.QLabel(self.centralwidget)
        self.label_2.setGeometry(QtCore.QRect(160, 500, 111, 21))
        self.label_2.setStyleSheet("color: #D4ECDD;\n"
                                   "font: 75 13pt \"Cascadia Code\";")
```

```python
        self.label_2.setObjectName("label_2")
        self.label_3 = QtWidgets.QLabel(self.centralwidget)
        self.label_3.setGeometry(QtCore.QRect(160, 80, 111, 21))
        self.label_3.setStyleSheet("color: #D4ECDD;\n"
                                   "font: 75 13pt \"Cascadia Code\";")
        self.label_3.setObjectName("label_3")
        self.label_4 = QtWidgets.QLabel(self.centralwidget)
        self.label_4.setGeometry(QtCore.QRect(160, 640, 111, 21))
        self.label_4.setStyleSheet("color: #D4ECDD;\n"
                                   "font: 75 13pt \"Cascadia Code\";")
        self.label_4.setObjectName("label_4")
        self.inputText_2 = QtWidgets.QPlainTextEdit(self.centralwidget)
        self.inputText_2.setGeometry(QtCore.QRect(160, 670, 401, 101))
        self.inputText_2.setStyleSheet("border: 2px solid #D4ECDD;\n"
                                       "border-radius: 5px;\n"
                                       "color: rgb(212, 236, 221);\n"
                                       "padding: 5px;\n"
                                       "font: 75 13pt \"Cascadia Code\";")
        self.inputText_2.setPlainText("")
        self.inputText_2.setObjectName("inputText_2")
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 1129, 21))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        # Coding here....
        self.pathFile = ""
        self.matrix = fullVigenere.generateFullVigenereMatrix()
        self.inputFileButton.clicked.connect(self.inputFileHandler)

        # Submit event

self.encryptDecryptButton.clicked.connect(self.encryptDecryptHandler)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    # Add method here...
    def inputFileHandler(self):
        file = QtWidgets.QFileDialog.getOpenFileName()
        self.pathFile = file[0]

        self.inputFileButton.setText(self.pathFile.split('/')[-1])

    def encryptDecryptHandler(self):
        cipherAlgorithm = self.cipherAlgorithmComboBox.currentText()
        text = self.inputText.toPlainText()
        key = self.inputText_2.toPlainText()

        if(self.pathFile != ""):
            ext = os.path.splitext(self.pathFile)[1]
            if(ext == ".txt"):
                f = open(self.pathFile)
                text = f.read()

        if(len(key) == 0):
            return

        res = ""

        # Encrypt
        if("encrypt" in cipherAlgorithm.lower()):
            if cipherAlgorithm == "Vignere Cipher Standard Encrypt":
                cipherText =
vignere_cipher.vignere_cipher_standard_encrypt(text, key)

                res += "Cipher Text:\n\n"
                res += cipherText
                res += "\n"
                res += ' '.join([cipherText[i: i+5] for i in range(0,
len(cipherText), 5)])

            elif cipherAlgorithm == "Full Vignere Cipher Encrypt":
                res += fullVigenere.encrypt(text, key, self.matrix)
                res += "\n\n"

                for i in range(len(self.matrix)):
                    for j in range(len(self.matrix[0])):
                        res += ('{} '.format(self.matrix[i][j]))
                    res += '\n'

            elif cipherAlgorithm == "Auto Key Vignere Cipher Encrypt":
                cipherText, newKey =
vignere_cipher.vignere_cipher_auto_key_encrypt(text, key)

                res += "Cipher Text:\n"
                res += cipherText
                res += "\n"
                res += ' '.join([cipherText[i: i+5] for i in range(0,
len(cipherText), 5)])
                res += "\n\n"
                res += "New Key:\n"
                res += newKey

            elif cipherAlgorithm == "Extended Vignere Cipher Encrypt":
                if(self.pathFile != ""):
                    dir_path = os.path.dirname(os.path.realpath(__file__))
                    filename = "data/res/" + str(uuid.uuid4()) +
os.path.splitext(self.pathFile)[1]

                    full_path = os.path.join(dir_path, filename)

                    success =
extendedVigenere.extended_vignere_cipher_encrypt(
                        self.pathFile,
                        key,
                        full_path
                    )

                    if(success):
                        res += "Success Encrypt File, Please Check This
Directory:\n"
                        res += full_path
                    else:
                        res += "Fail encrypt file"
                else:
                    res = "Please input file!"

            elif cipherAlgorithm == "Playfair Cipher Encrypt":
                # encryption
                playfairSquare = playfairCipher.generatePlayfairSquare(key)
                res += playfairCipher.encrypt(text, playfairSquare) + '\n\n'
                for i in range(len(playfairSquare)):
                    for j in range(len(playfairSquare[0])):
                        res += ('{} '.format(playfairSquare[i][j]))
                    res += '\n'

            elif cipherAlgorithm == "Affine Cipher Encrypt":
                # parsing key
                newKey = key.split(',')
                newKey = [int(item) for item in newKey]

                # decryption
                res += affineCipher.affineEncrypt(text, newKey)

            elif cipherAlgorithm == "Hill Cipher Encrypt":
                key = json.loads(key)
                if(isinstance(key, list)):
                    try:
                        key = np.array(key)
                        res += "Result:\n"
                        res += hill_cipher.hill_cipher_encrypt(text, key)
                    except:
                        res = "Dimensi key harus bisa membagi panjang text
nya!"
                else:
                    res += "Please input valid key!"
        else:
            if cipherAlgorithm == "Vignere Cipher Standard Decrypt":
                plainText = 
vignere_cipher.vignere_cipher_standard_decrypt(text, key)

                res += "Plain Text:\n\n"
                res += plainText
                res += "\n"
                res += ' '.join([plainText[i: i+5] for i in range(0, 
len(plainText), 5)])

            elif cipherAlgorithm == "Full Vignere Cipher Decrypt":
                res += fullVigenere.decrypt(text, key, self.matrix)
                res += '\n\n'
                for i in range(len(self.matrix)):
                    for j in range(len(self.matrix[0])):
                        res += ('{} '.format(self.matrix[i][j]))
                    res += '\n'

            elif cipherAlgorithm == "Auto Key Vignere Cipher Decrypt":
                plainText = 
vignere_cipher.vignere_cipher_auto_key_decrypt(text, key)

                res += "Plain Text:\n\n"
                res += plainText
                res += "\n"
                res += ' '.join([plainText[i: i+5] for i in range(0, 
len(plainText), 5)])

            elif cipherAlgorithm == "Extended Vignere Cipher Decrypt":
                if(self.pathFile != ""):
                    dir_path = os.path.dirname(os.path.realpath(__file__))
                    filename = "data/res/" + str(uuid.uuid4()) + 
os.path.splitext(self.pathFile)[1]

                    full_path = os.path.join(dir_path, filename)

                    success = 
extendedVigenere.extended_vignere_cipher_decrypt(
                        self.pathFile,
                        key,
                        full_path
                    )

                    if(success):
                        res += "Success Decrypt File, Please Check This 
Directory:\n"
                        res += full_path
                    else:
                        res += "Fail decrypt file"
                else:
                    res = "Please input file!"

            elif cipherAlgorithm == "Playfair Cipher Decrypt":
                playfairSquare = playfairCipher.generatePlayfairSquare(key)
                res += playfairCipher.decrypt(text, playfairSquare) + '\n\n'
                for i in range(len(playfairSquare)):
                    for j in range(len(playfairSquare[0])):
                        res += ('{} '.format(playfairSquare[i][j]))
                    res += '\n'

            elif cipherAlgorithm == "Affine Cipher Decrypt":
                # parsing key
                newKey = key.split(',')
                newKey = [int(item) for item in newKey]

                # decryption
                text = affineCipher.textCleaning(text)
                res += affineCipher.affineDecrypt(text, newKey)

            elif cipherAlgorithm == "Hill Cipher Decrypt":
                key = json.loads(key)
                if(isinstance(key, list)):
                    try:
                        key = np.array(key)
                        plainText = hill_cipher.hill_cipher_decrypt(text,
key)

                        res += "Plain Text:\n\n"
                        res += plainText
                        res += "\n"
                        res += ' '.join([plainText[i: i+5] for i in range(0,
len(plainText), 5)])
                    except:
                        res += "Dimensi key harus bisa membagi panjang text
nya!"
                        res += "Panjang text sekarang =
{}".format(len(hill_cipher.clean_text(text)))
                        res += "Dimensi key sekarang =
{}".format(key.shape[0])
                else:
                    res += "Please input valid key!"

        if("Extended Vignere Cipher" not in cipherAlgorithm):
            dir_path = os.path.dirname(os.path.realpath(__file__))
            filename = "data/res/" + str(uuid.uuid4()) + ".txt"

            full_path = os.path.join(dir_path, filename)
            f = open(full_path, 'w')
            f.write(res)
            res += "\n\n"
            res += "This Result Has Been Saved, Please Check This
Directory:\n"
            res += full_path

        # Clear input
        self.outputTextArea.setPlainText(res)
        self.pathFile = ""
        self.inputFileButton.setText("Input File")

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
        self.label.setText(_translate(
            "MainWindow", "<html><head/><body><p align=\"center\"><span
style=\" font-size:14pt; font-weight:600;\">Classic
Cryptography</span></p></body></html>"))
        self.cipherAlgorithmComboBox.setItemText(0, _translate(
            "MainWindow", "Vignere Cipher Standard Encrypt"))
        self.cipherAlgorithmComboBox.setItemText(1, _translate(
            "MainWindow", "Vignere Cipher Standard Decrypt"))
        self.cipherAlgorithmComboBox.setItemText(
            2, _translate("MainWindow", "Full Vignere Cipher Encrypt"))
        self.cipherAlgorithmComboBox.setItemText(
            3, _translate("MainWindow", "Full Vignere Cipher Decrypt"))
        self.cipherAlgorithmComboBox.setItemText(4, _translate(
            "MainWindow", "Auto Key Vignere Cipher Encrypt"))
        self.cipherAlgorithmComboBox.setItemText(5, _translate(
            "MainWindow", "Auto Key Vignere Cipher Decrypt"))
        self.cipherAlgorithmComboBox.setItemText(6, _translate(
            "MainWindow", "Extended Vignere Cipher Encrypt"))
        self.cipherAlgorithmComboBox.setItemText(7, _translate(
            "MainWindow", "Extended Vignere Cipher Decrypt"))
        self.cipherAlgorithmComboBox.setItemText(
            8, _translate("MainWindow", "Playfair Cipher Encrypt"))
        self.cipherAlgorithmComboBox.setItemText(
            9, _translate("MainWindow", "Playfair Cipher Decrypt"))
        self.cipherAlgorithmComboBox.setItemText(
            10, _translate("MainWindow", "Affine Cipher Encrypt"))
        self.cipherAlgorithmComboBox.setItemText(
            11, _translate("MainWindow", "Affine Cipher Decrypt"))
        self.cipherAlgorithmComboBox.setItemText(
            12, _translate("MainWindow", "Hill Cipher Encrypt"))
        self.cipherAlgorithmComboBox.setItemText(
            13, _translate("MainWindow", "Hill Cipher Decrypt"))
        self.encryptDecryptButton.setText(
            _translate("MainWindow", "Encrypt / Decrypt"))
        self.inputFileButton.setText(_translate("MainWindow", "Input File"))
        self.label_2.setText(_translate("MainWindow", "Text Input"))
        self.label_3.setText(_translate("MainWindow", "Text Output"))
        self.label_4.setText(_translate("MainWindow", "Key Input"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

2. Screenshot Program



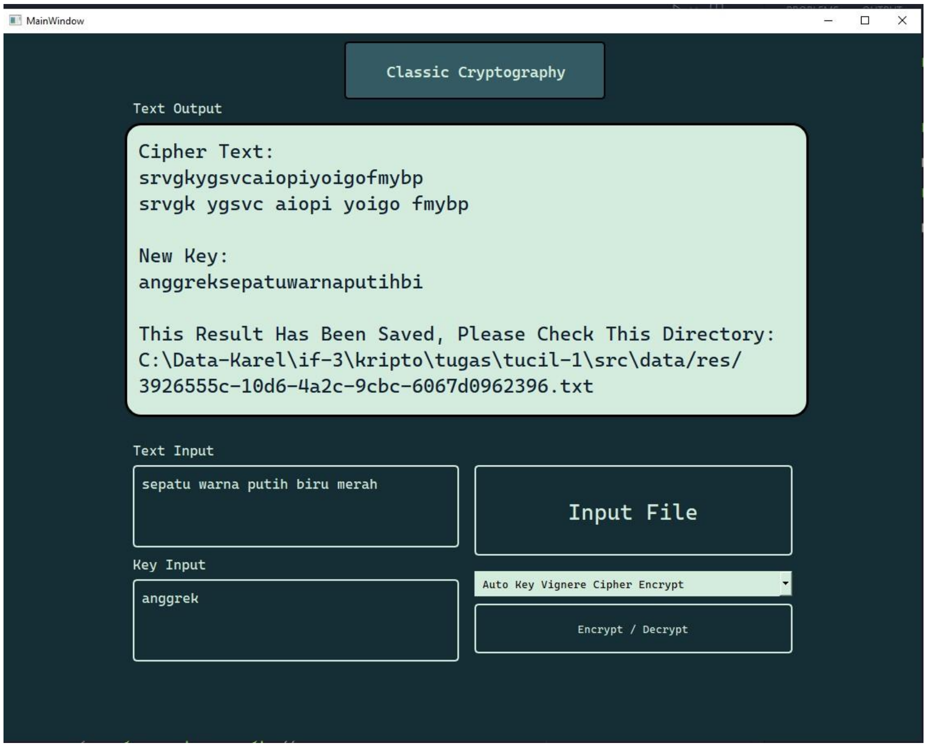
Text Output

```
Plain Text:

sepatuwarnaputihbirumerah
sepat uwarn aputi hbiru merah

This Result Has Been Saved, Please Check This Directory:
C:\Data-Karel\if-3\kripto\tugas\tucil-1\src\data/res/
b3a4a035-3cb9-4803-ba1a-77a22d73451f.txt
```

Text Input: srvgkygsvcaiopiyoigofmybp

Key Input: anggreksepatuwarnaputihbi

Auto Key Vignere Cipher Decrypt



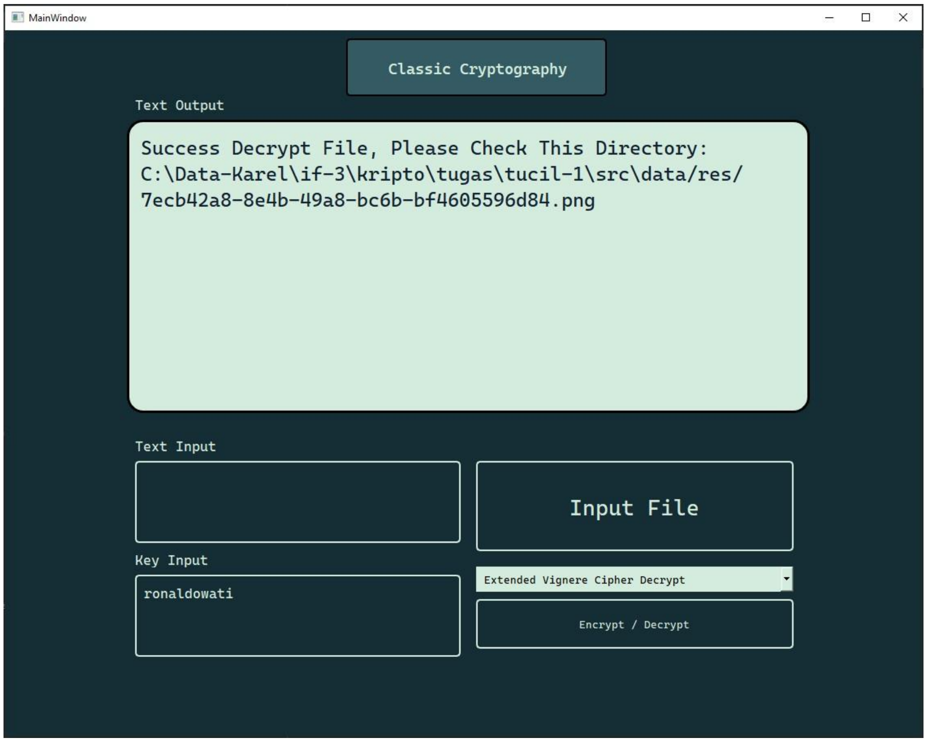
Text Output

```
Cipher Text:
srvgkygsvcaiopiyoigofmybp
srvgk ygsvc aiopi yoigo fmybp

New Key:
anggreksepatuwarnaputihbi

This Result Has Been Saved, Please Check This Directory:
C:\Data-Karel\if-3\kripto\tugas\tucil-1\src\data/res/
3926555c-10d6-4a2c-9cbc-6067d0962396.txt
```

Text Input: sepatu warna putih biru merah

Key Input: anggrek

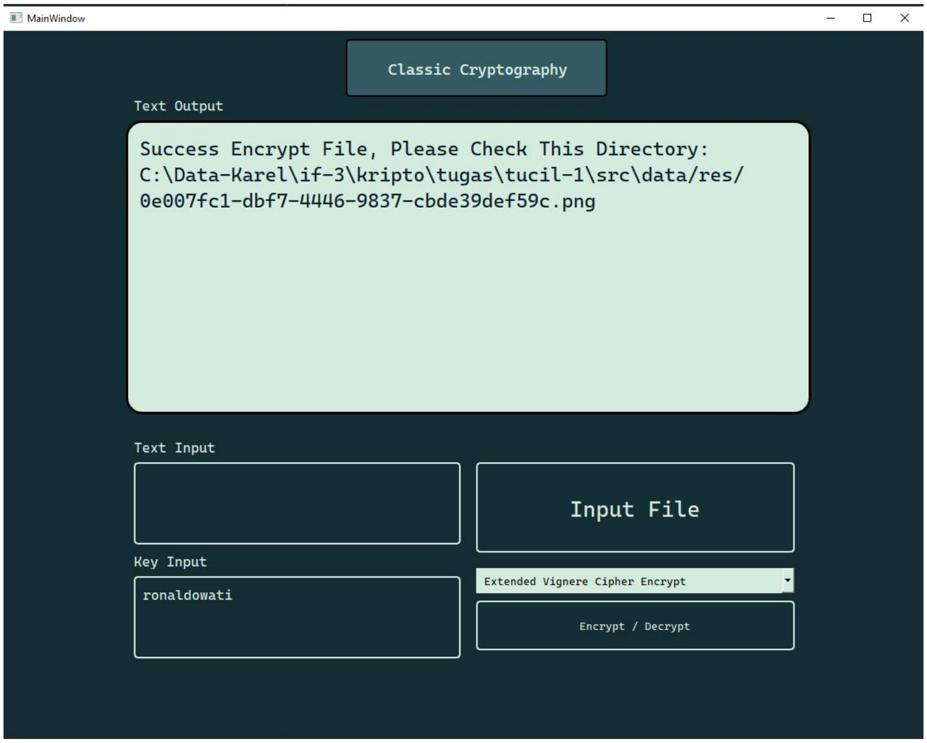
Auto Key Vignere Cipher Encrypt



```
Success Decrypt File, Please Check This Directory:
C:\Data-Karel\if-3\kripto\tugas\tucil-1\src\data/res/
7ecb42a8-8e4b-49a8-bc6b-bf4605596d84.png
```

Key Input: ronaldowati

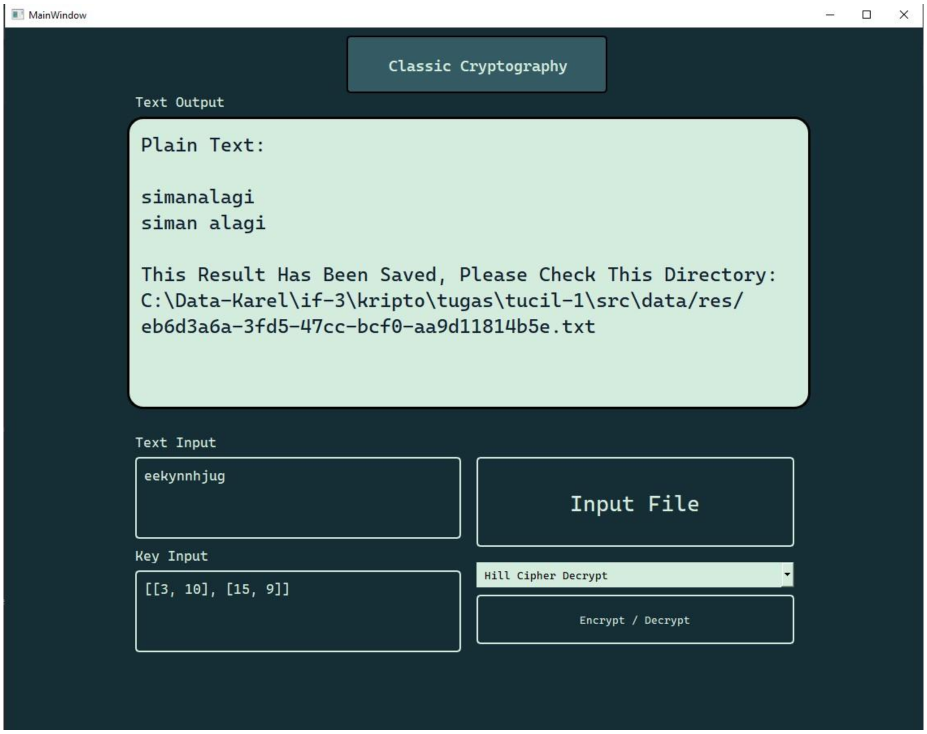
Extended Vignere Cipher Decrypt



```
Success Encrypt File, Please Check This Directory:
C:\Data-Karel\if-3\kripto\tugas\tucil-1\src\data/res/
0e007fc1-dbf7-4446-9837-cbde39def59c.png
```

Key Input: ronaldowati

Extended Vignere Cipher Encrypt



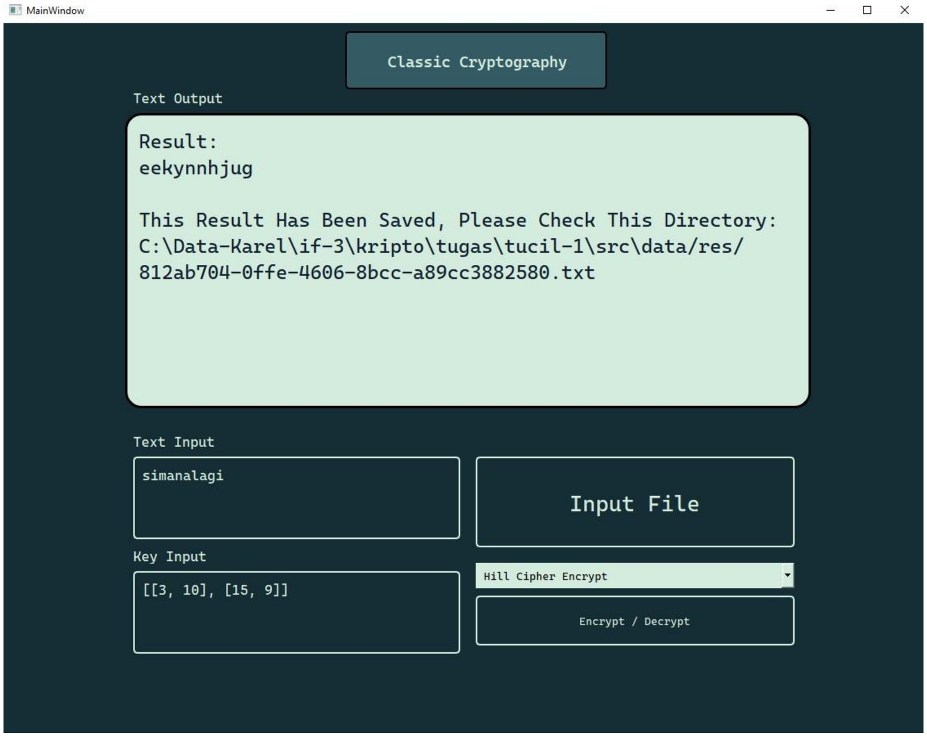
Text Output

```
Plain Text:

simanalagi
siman alagi

This Result Has Been Saved, Please Check This Directory:
C:\Data-Karel\if-3\kripto\tugas\tucil-1\src\data/res/
eb6d3a6a-3fd5-47cc-bcf0-aa9d11814b5e.txt
```

Text Input: eekynnhjug

Key Input: [[3, 10], [15, 9]]

Hill Cipher Decrypt



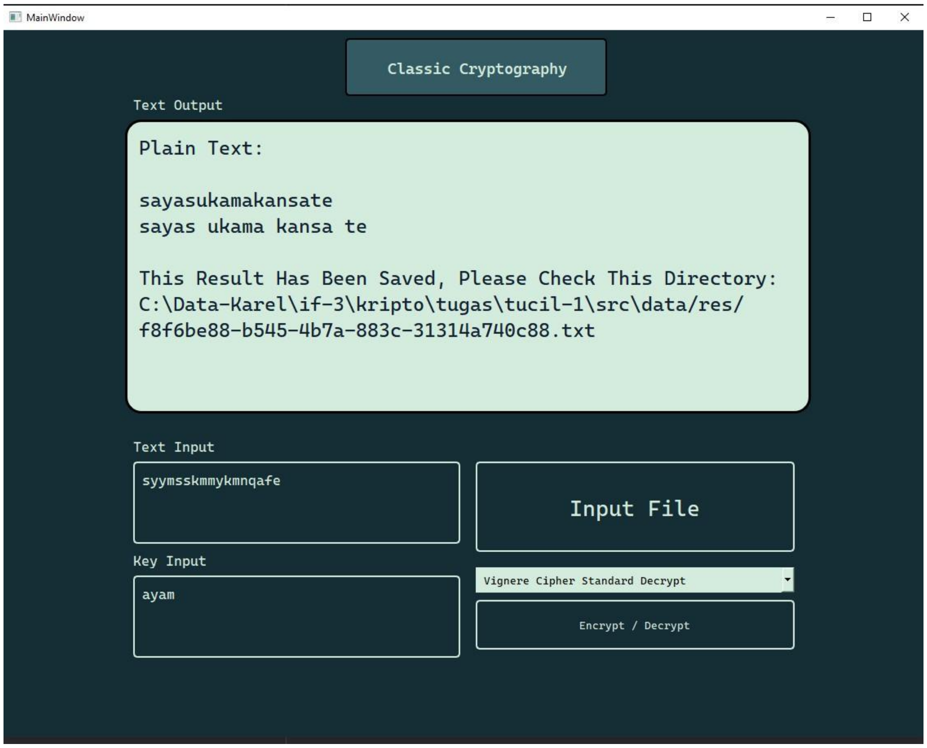
Text Output

```
Result:
eekynnhjug

This Result Has Been Saved, Please Check This Directory:
C:\Data-Karel\if-3\kripto\tugas\tucil-1\src\data/res/
812ab704-0ffe-4606-8bcc-a89cc3882580.txt
```

Text Input: simanalagi

Key Input: [[3, 10], [15, 9]]

Hill Cipher Encrypt

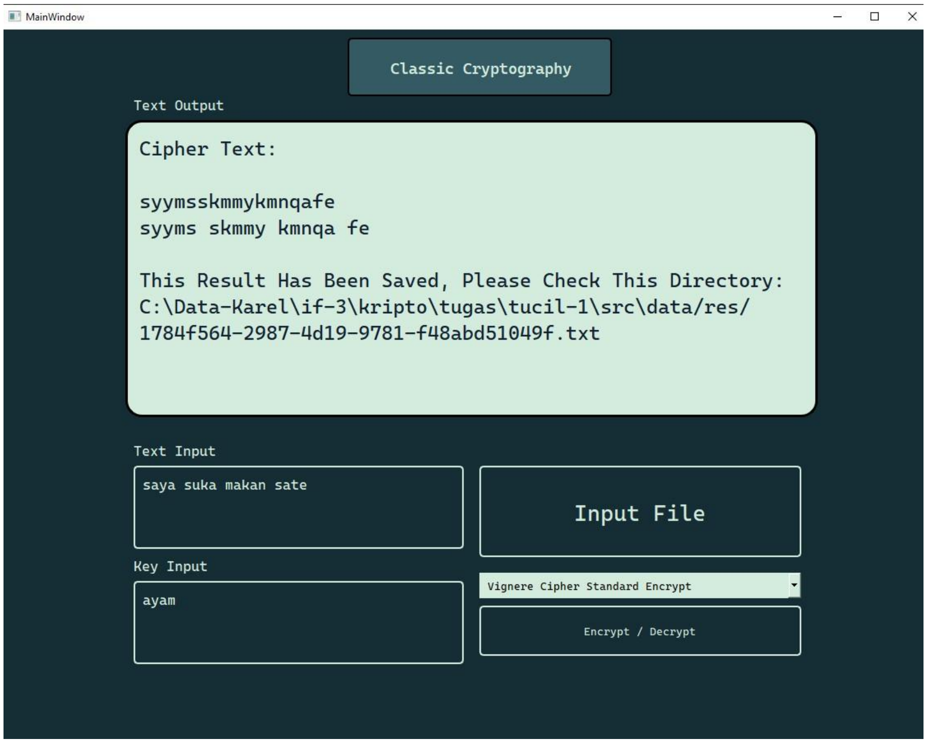


```
Plain Text:

sayasukamakansate
sayas ukama kansa te

This Result Has Been Saved, Please Check This Directory:
C:\Data-Karel\if-3\kripto\tugas\tucil-1\src\data/res/
f8f6be88-b545-4b7a-883c-31314a740c88.txt
```

Text Input: syymsskmmykmnqafe

Key Input: ayam

Vignere Cipher Standard Decrypt

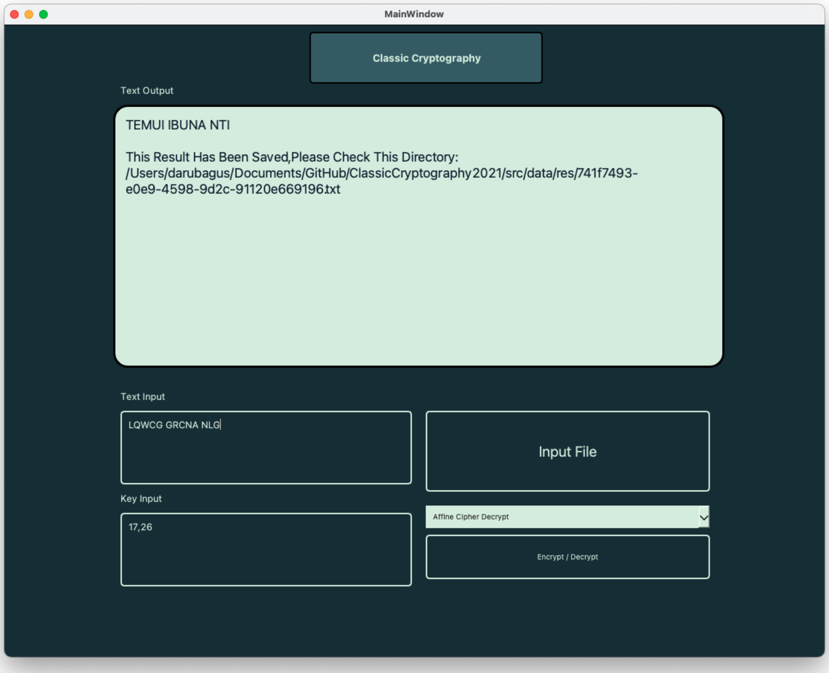


```
Cipher Text:

syymsskmmykmnqafe
syyms skmmy kmnqa fe

This Result Has Been Saved, Please Check This Directory:
C:\Data-Karel\if-3\kripto\tugas\tucil-1\src\data/res/
1784f564-2987-4d19-9781-f48abd51049f.txt
```

Text Input: saya suka makan sate

Key Input: ayam

Vignere Cipher Standard Encrypt



Text Output

TEMUI IBUNA NTI

This Result Has Been Saved,Please Check This Directory:
/Users/darubagus/Documents/GitHub/ClassicCryptography2021/src/data/res/741f7493-e0e9-4598-9d2c-91120e669196.txt

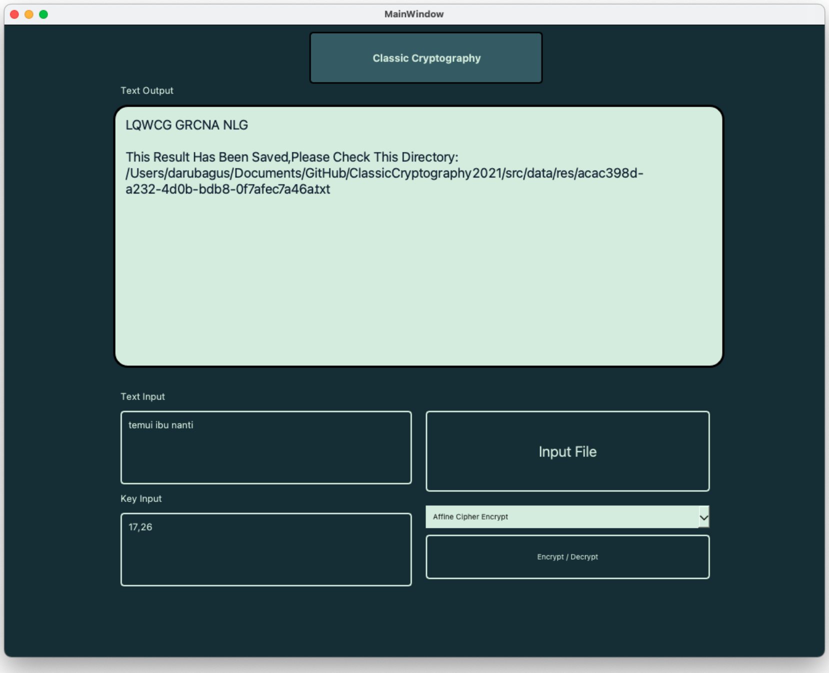
Text Input

LQWCG GRCNA NLG

Key Input

17,26

Affine Cipher Decrypt



Text Output

LQWCG GRCNA NLG

This Result Has Been Saved,Please Check This Directory:
/Users/darubagus/Documents/GitHub/ClassicCryptography2021/src/data/res/acac398d-a232-4d0b-bdb8-0f7afec7a46a.txt

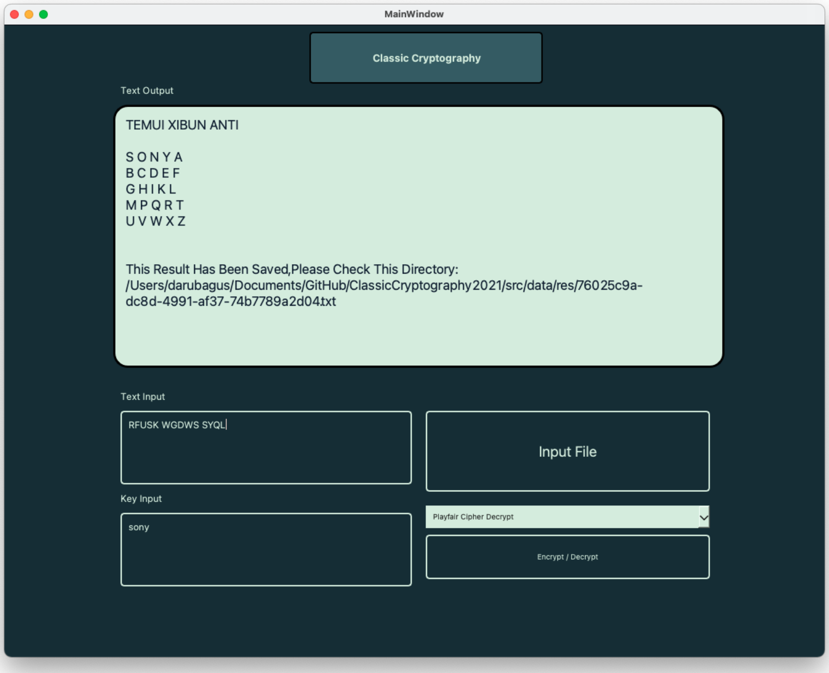
Text Input

temui ibu nanti

Key Input

17,26

Affine Cipher Encrypt



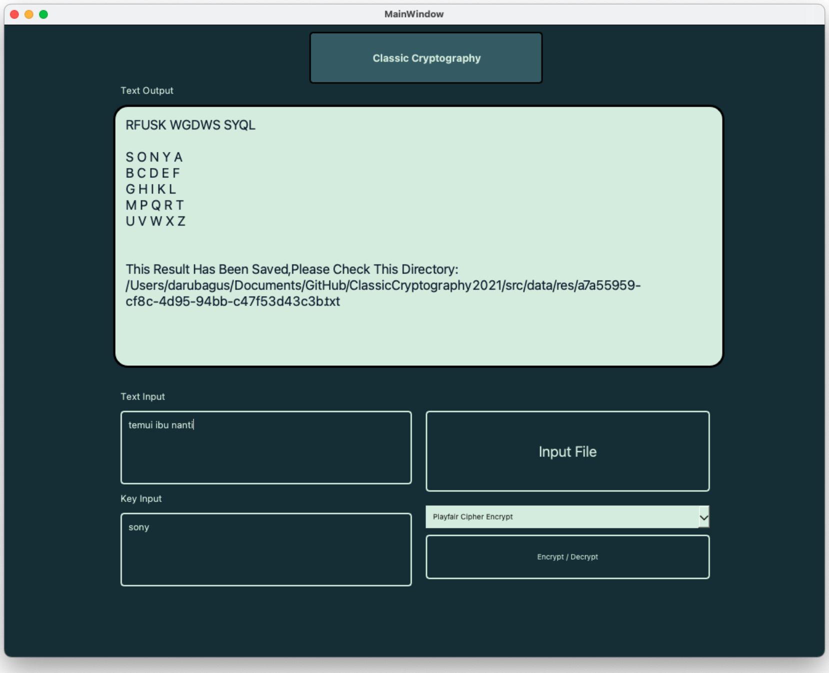
Text Output

```
TEMUI XIBUN ANTI

S O N Y A
B C D E F
G H I K L
M P Q R T
U V W X Z


This Result Has Been Saved,Please Check This Directory:
/Users/darubagus/Documents/GitHub/ClassicCryptography2021/src/data/res/76025c9a-
dc8d-4991-af37-74b7789a2d04.txt
```

Text Input: RFUSK WGDWS SYQL

Key Input: sony

Playfair Cipher Decrypt



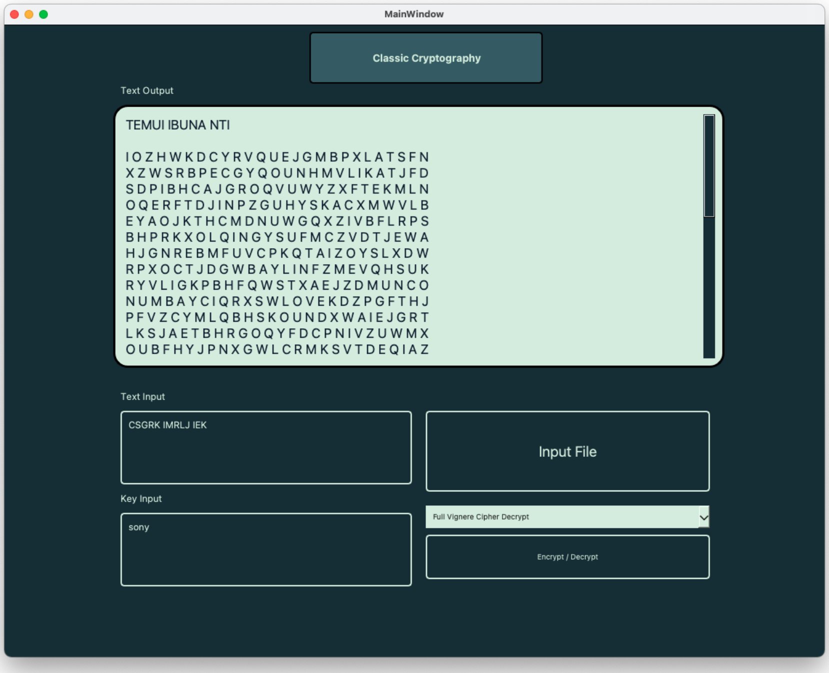
Text Output

```
RFUSK WGDWS SYQL

S O N Y A
B C D E F
G H I K L
M P Q R T
U V W X Z


This Result Has Been Saved,Please Check This Directory:
/Users/darubagus/Documents/GitHub/ClassicCryptography2021/src/data/res/a7a55959-
cf8c-4d95-94bb-c47f53d43c3b.txt
```

Text Input: temui ibu nanti

Key Input: sony

Playfair Cipher Encrypt



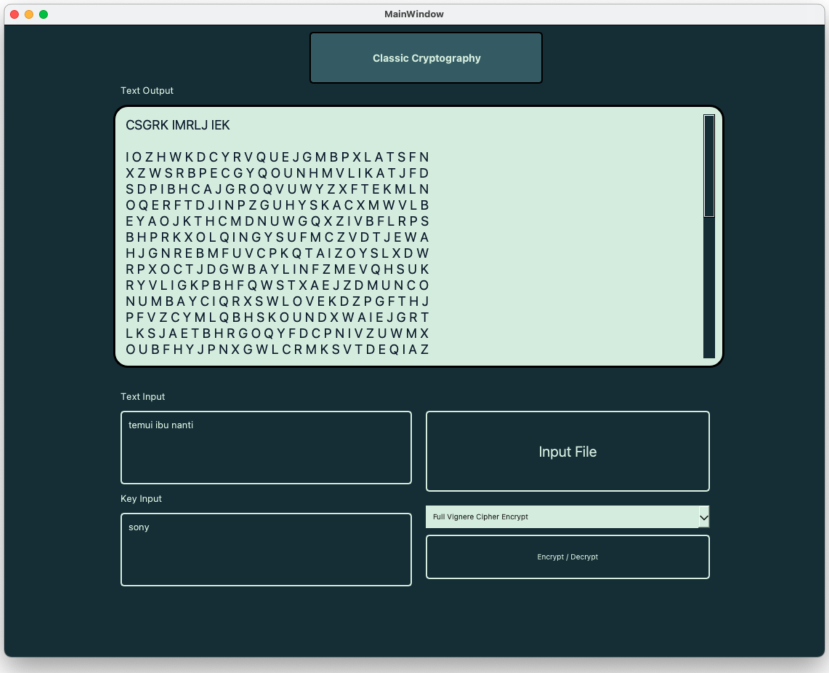
Text Output

```
TEMUI IBUNA NTI

I O Z H W K D C Y R V Q U E J G M B P X L A T S F N
X Z W S R B P E C G Y Q O U N H M V L I K A T J F D
S D P I B H C A J G R O Q V U W Y Z X F T E K M L N
O Q E R F T D J I N P Z G U H Y S K A C X M W V L B
E Y A O J K T H C M D N U W G Q X Z I V B F L R P S
B H P R K X O L Q I N G Y S U F M C Z V D T J E W A
H J G N R E B M F U V C P K Q T A I Z O Y S L X D W
R P X O C T J D G W B A Y L I N F Z M E V Q H S U K
R Y V L I G K P B H F Q W S T X A E J Z D M U N C O
N U M B A Y C I Q R X S W L O V E K D Z P G F T H J
P F V Z C Y M L Q B H S K O U N D X W A I E J G R T
L K S J A E T B H R G O Q Y F D C P N I V Z U W M X
O U B F H Y J P N X G W L C R M K S V T D E Q I A Z
```

Text Input: CSGRK IMRLJ IEK

Key Input: sony

Full Vignere Cipher Decrypt



Text Output

```
CSGRK IMRLJ IEK

I O Z H W K D C Y R V Q U E J G M B P X L A T S F N
X Z W S R B P E C G Y Q O U N H M V L I K A T J F D
S D P I B H C A J G R O Q V U W Y Z X F T E K M L N
O Q E R F T D J I N P Z G U H Y S K A C X M W V L B
E Y A O J K T H C M D N U W G Q X Z I V B F L R P S
B H P R K X O L Q I N G Y S U F M C Z V D T J E W A
H J G N R E B M F U V C P K Q T A I Z O Y S L X D W
R P X O C T J D G W B A Y L I N F Z M E V Q H S U K
R Y V L I G K P B H F Q W S T X A E J Z D M U N C O
N U M B A Y C I Q R X S W L O V E K D Z P G F T H J
P F V Z C Y M L Q B H S K O U N D X W A I E J G R T
L K S J A E T B H R G O Q Y F D C P N I V Z U W M X
O U B F H Y J P N X G W L C R M K S V T D E Q I A Z
```

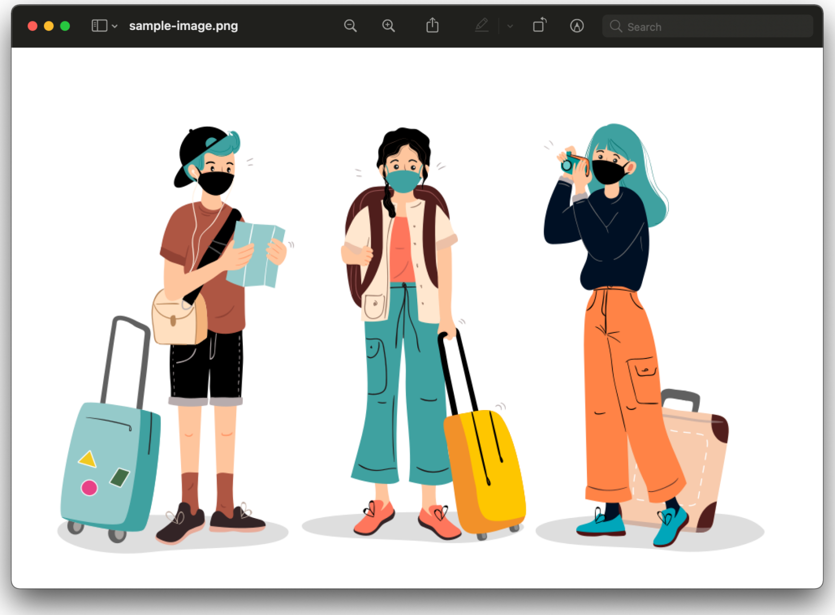
Text Input: temui ibu nanti

Key Input: sony

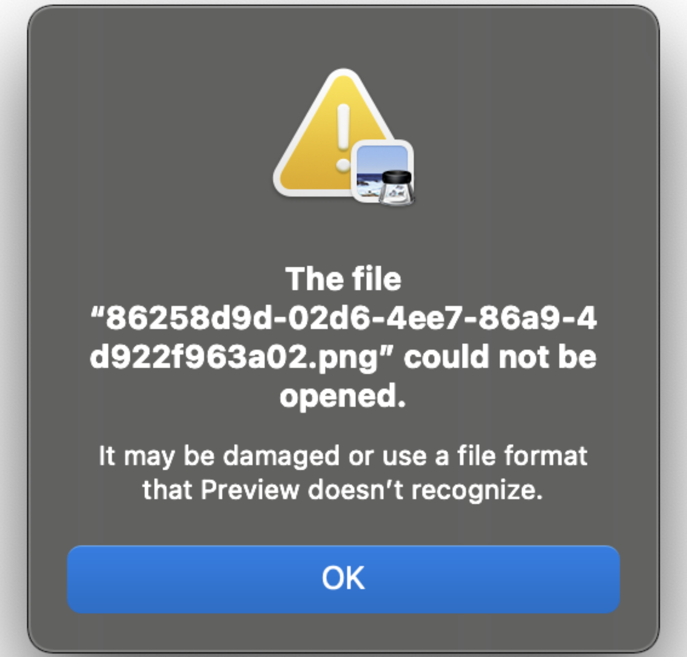
Full Vignere Cipher Encrypt

3. Contoh File

A. Gambar sebelum diencrypt



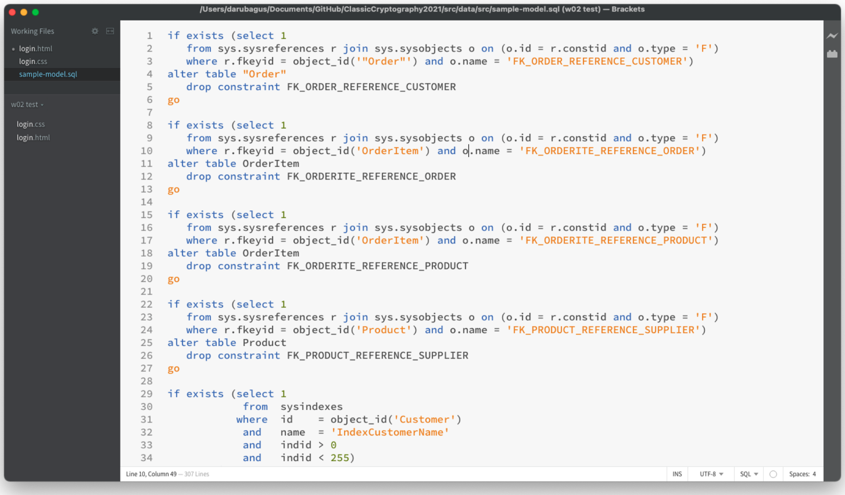
B. Gambar setelah diencrypt



The file “86258d9d-02d6-4ee7-86a9-4d922f963a02.png” could not be opened.

It may be damaged or use a file format that Preview doesn’t recognize.

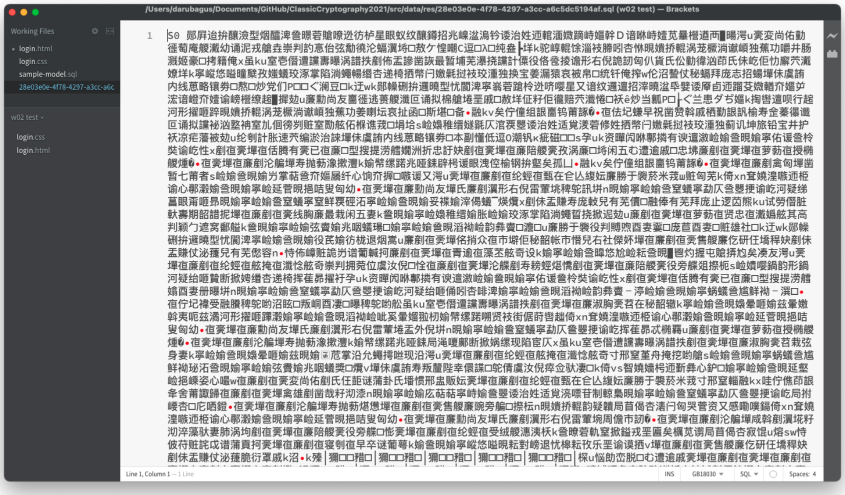
C. Database sebelum diencrypt



/Users/darubagus/Documents/GitHub/ClassicCryptography2021/src/data/src/sample-model.sql (w02 test) — Brackets
Working Files
login.html
login.css
sample-model.sql
w02 test
login.css
login.html
Line 10, Column 49 — 307 Lines
INS
UTF-8
SQL
Spaces: 4

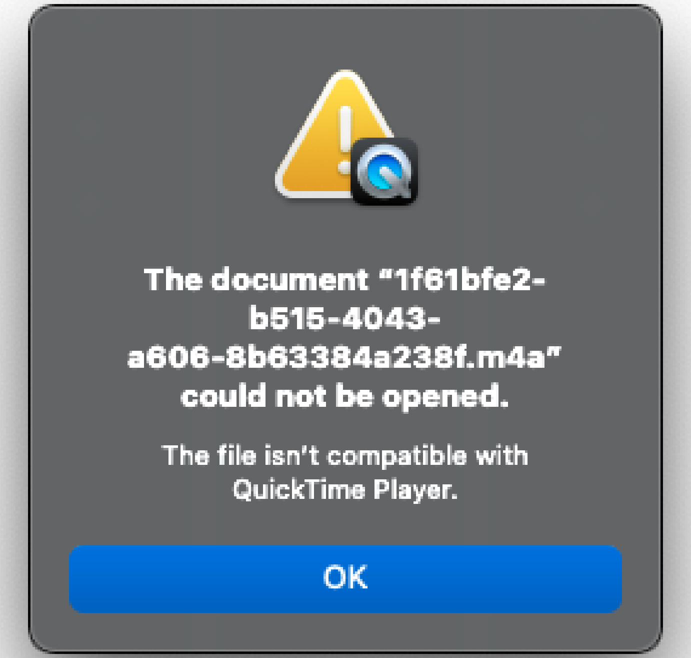
```sql
if exists (select 1
   from sys.sysreferences r join sys.sysobjects o on (o.id = r.constid and o.type = 'F')
   where r.fkeyid = object_id('"Order"') and o.name = 'FK_ORDER_REFERENCE_CUSTOMER')
alter table "Order"
   drop constraint FK_ORDER_REFERENCE_CUSTOMER
go

if exists (select 1
   from sys.sysreferences r join sys.sysobjects o on (o.id = r.constid and o.type = 'F')
   where r.fkeyid = object_id('OrderItem') and o.name = 'FK_ORDERITE_REFERENCE_ORDER')
alter table OrderItem
   drop constraint FK_ORDERITE_REFERENCE_ORDER
go

if exists (select 1
   from sys.sysreferences r join sys.sysobjects o on (o.id = r.constid and o.type = 'F')
   where r.fkeyid = object_id('OrderItem') and o.name = 'FK_ORDERITE_REFERENCE_PRODUCT')
alter table OrderItem
   drop constraint FK_ORDERITE_REFERENCE_PRODUCT
go

if exists (select 1
   from sys.sysreferences r join sys.sysobjects o on (o.id = r.constid and o.type = 'F')
   where r.fkeyid = object_id('Product') and o.name = 'FK_PRODUCT_REFERENCE_SUPPLIER')
alter table Product
   drop constraint FK_PRODUCT_REFERENCE_SUPPLIER
go

if exists (select 1
            from  sysindexes
           where  id    = object_id('Customer')
            and   name  = 'IndexCustomerName'
            and   indid > 0
            and   indid < 255)
```

D. Database setelah diencrypt



/Users/darubagus/Documents/GitHub/ClassicCryptography2021/src/data/res/28e03e0e-4f78-4297-a3cc-a6c5dc5194af.sql (w02 test) — Brackets
Working Files
login.html
login.css
sample-model.sql
28e03e0e-4f78-4297-a3cc-a6c
w02 test
login.css
login.html
Line 1, Column 1 — 1 Line
INS
GB18030
SQL
Spaces: 4

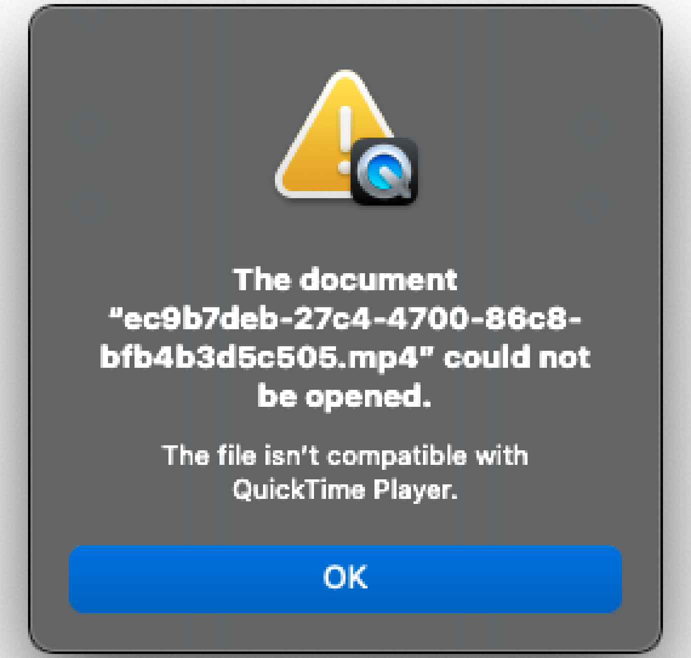
E. Musik setelah diencrypt



The document “1f61bfe2-b515-4043-a606-8b63384a238f.m4a” could not be opened.

The file isn't compatible with QuickTime Player.

F. Video setelah diencrypt



The document "ec9b7deb-27c4-4700-86c8-bfb4b3d5c505.mp4" could not be opened.

The file isn't compatible with QuickTime Player.

4. **Link repository**

Untuk melakukan run program, dapat dilakukan dengan cara mengeksekusi perintah

```
python3 src/classic-crypto.py
```

Github : https://github.com/darubagus/ClassicCryptography2021

No	Spek	Berhasil (✔)	Kurang Berhasil (✔)	Keterangan
1	Vigenere Standard	✔		
2	Full Vigenere Cipher	✔		
3	Auto-key Vigenere Cipher	✔		
4	Extended Vigenere Cipher	✔		
5	Playfair Cipher	✔		
6	Bonus : Hill Cipher	✔		