

TUGAS 1 II4031 KRIPTOGRAFI DAN KODING
SEMESTER II TAHUN 2022/2023

Kriptografi Klasik



Dosen Pengampu

Dr. Ir. Rinaldi Munir, M.T.

Oleh

Natanael Dias 18220051

Brandon Jonathan Latif 18220103

PROGRAM STUDI SISTEM DAN TEKNOLOGI INFORMASI
SEKOLAH TEKNIK ELEKTRO & INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2023

Daftar Isi

A. Source Code	3
B. Interface	27
C. Testing	34
D. Github	43

A. Source Code

Pada tugas pertama, diberikan tugas kriptografi klasik menggunakan bahasa pemrograman Python. Fungsi-fungsi kriptografi yang ditugaskan melingkupi :

1. *Vigenere Cipher standard* (26 huruf alfabet)
2. *Extended Vigenere Cipher* (256 karakter ASCII)
3. *Playfair Cipher* (26 huruf alfabet)
4. *One-time pad* (26 huruf alfabet)
5. *Enigma Cipher 3 rotor* (26 huruf alfabet) sebagai bonus

Berikut merupakan potongan program yang terdiri dari beberapa *files*, sebagai berikut :

- main.py

Pada *file* main.py berisi *code* untuk menampilkan antarmuka (GUI) menggunakan PyQt. Pada main.py akan dipanggil fungsi-fungsi yang ada pada tiap *file cipher*. Berikut merupakan *code* pada main.py.

```
import sys
from tkinter import Widget
from PyQt5.QtWidgets import QDialog,
QApplication, QWidget, QMainWindow
from PyQt5.uic import loadUi
from PyQt5 import QtWidgets, QtGui, QtCore
from PyQt5.QtGui import QPixmap

from vigenere import *
from extended_vigenere import *
from playfair import *
from one_time_pad import *
from check import *

import sqlite3
import os
```

```

import csv
import datetime
import random

class Menu(QMainWindow):
    def __init__(self):
        super(Menu, self).__init__()
        loadUi("main.ui", self)
        self.label.setText("Welcome To Cipher
Program!!!")

self.pushButton_2.clicked.connect(self.Vigenere)

self.pushButton_3.clicked.connect(self.ExtendVigenere
)

self.pushButton_4.clicked.connect(self.OneTimePad)

self.pushButton_5.clicked.connect(self.Playfair)

self.pushButton_6.clicked.connect(self.Enigma)

    def Vigenere(self):
        vigenere = Vigenere()
        widget.addWidget(vigenere)
        widget.setCurrentIndex(widget.currentIndex()
+ 1)

    def ExtendVigenere(self):
        extendVigenere = ExtendVigenere()
        widget.addWidget(extendVigenere)
        widget.setCurrentIndex(widget.currentIndex()
+ 1)

```

```

        def OneTimePad(self):
            oneTimePad = OneTimePad()
            widget.addWidget(oneTimePad)
            widget.setCurrentIndex(widget.currentIndex()
+ 1)

        def Playfair(self):
            playfair = Playfair()
            widget.addWidget(playfair)
            widget.setCurrentIndex(widget.currentIndex()
+ 1)

        def Enigma(self):
            enigma = Enigma()
            widget.addWidget(enigma)
            widget.setCurrentIndex(widget.currentIndex()
+ 1)

class Vigenere(QMainWindow):
    def __init__(self):
        super(Vigenere, self).__init__()
        loadUi("cipher.ui", self)

        self.label_2.setText("Vigenere Cipher")
        self.pushButton_10.clicked.connect(self.Menu)

self.pushButton_6.clicked.connect(self.Encrypt)

self.pushButton_7.clicked.connect(self.Decrypt)

self.pushButton_11.clicked.connect(self.AddFile)

self.pushButton_8.clicked.connect(self.Default)

```

```

self.pushButton_9.clicked.connect(self.Groupped)
    self.pushButton_12.clicked.connect(self.Save)

    def Menu(self):
        menu = Menu()
        widget.addWidget(menu)
        widget.setCurrentIndex(widget.currentIndex()
+ 1)

    def Encrypt(self):
        plaintext_awal = self.textEdit.toPlainText()
        plaintext_tmp = [x for x in plaintext_awal]
        plaintext = check_alphabet(plaintext_tmp)
        key_awal = self.textEdit_2.toPlainText()
        key_tmp = [x for x in key_awal]
        key = check_alphabet(key_tmp)
        keyFinal = create_kunci(plaintext, key)
        result = vigenere_encrypt(plaintext,
keyFinal)
        self.textBrowser.setText(result)

    def Decrypt(self):
        ciphertext_awal = self.textEdit.toPlainText()
        ciphertext_tmp = [x for x in ciphertext_awal]
        ciphertext = check_alphabet(ciphertext_tmp)
        key_awal = self.textEdit_2.toPlainText()
        key_tmp = [x for x in key_awal]
        key = check_alphabet(key_tmp)
        keyFinal = create_kunci(ciphertext, key)
        result = vigenere_decrypt(ciphertext,
keyFinal)
        self.textBrowser.setText(result)

    def AddFile(self):

```

```

        fname = QFileDialog.getOpenFileName(self,
"Choose File", "E:\Kripto\kriptomanjaa", "All Files
(*)")

        with open(fname[0], 'r') as file :
            lines = file.read().rstrip()
            self.textEdit.setPlainText(str(lines))

def Default(self):
    output = []
    text_awal = self.textBrowser.toPlainText()
    text_tmp = [x for x in text_awal]
    for i in range(len(text_tmp)):
        if (text_tmp[i] != ' '):
            output.append(text_tmp[i])
    self.textBrowser.setText(''.join(output))

def Grouped(self):
    output_tmp = []
    output = []
    text_awal = self.textBrowser.toPlainText()
    text_tmp = [x for x in text_awal]
    for i in range(len(text_tmp)):
        if (text_tmp[i] != ' '):
            output_tmp.append(text_tmp[i])
    for i in range(len(output_tmp)):
        if i % 5 == 0 and i > 0:
            output.append(' ')
            output.append(output_tmp[i])
    self.textBrowser.setText(''.join(output))

def Save(self):
    name = QFileDialog.getSaveFileName(self,
"Save File")

```

```

        file = open(name[0], 'w')
        text = self.textBrowser.toPlainText()
        file.write(text)
        file.close()

class ExtendVigenere(QMainWindow):
    def __init__(self):
        super(ExtendVigenere, self).__init__()
        loadUi("cipher.ui", self)

        self.label_2.setText("Extend Vigenere
Cipher")
        self.pushButton_10.clicked.connect(self.Menu)

self.pushButton_6.clicked.connect(self.Encrypt)

self.pushButton_7.clicked.connect(self.Decrypt)

self.pushButton_11.clicked.connect(self.AddFile)

self.pushButton_8.clicked.connect(self.Default)

self.pushButton_9.clicked.connect(self.Grouped)
        self.pushButton_12.clicked.connect(self.Save)

    def Menu(self):
        menu = Menu()
        widget.addWidget(menu)
        widget.setCurrentIndex(widget.currentIndex()
+ 1)

    def Encrypt(self):
        plaintext_awal = self.textEdit.toPlainText()

```



```

        plaintext_tmp = [x for x in plaintext_awal]
        key_awal = self.textEdit_2.toPlainText()
        key_tmp = [x for x in key_awal]
        keyFinal = create_kunci(plaintext_tmp,
key_tmp)
        result =
extended_vigenere_encrypt(plaintext_tmp, keyFinal)
        self.textBrowser.setText(result)

    def Decrypt(self):
        ciphertext_awal = self.textEdit.toPlainText()
        ciphertext_tmp = [x for x in ciphertext_awal]
        key_awal = self.textEdit_2.toPlainText()
        key_tmp = [x for x in key_awal]
        keyFinal = create_kunci(ciphertext_tmp,
key_tmp)
        result =
extended_vigenere_decrypt(ciphertext_tmp, keyFinal)
        self.textBrowser.setText(result)

    def AddFile(self):
        fname = QFileDialog.getOpenFileName(self,
"Choose File", "E:\Kripto\kriptomanjaa", "All Files
(*)")

        with open(fname[0], 'r') as file :
            lines = file.read().rstrip()
            self.textEdit.setPlainText(str(lines))

    def Default(self):
        output = []
        text_awal = self.textBrowser.toPlainText()
        text_tmp = [x for x in text_awal]
        for i in range(len(text_tmp)):
            if (text_tmp[i] != ' '):

```

```

        output.append(text_tmp[i])
    self.textBrowser.setText(''.join(output))

def Grouped(self):
    output_tmp = []
    output = []
    text_aval = self.textBrowser.toPlainText()
    text_tmp = [x for x in text_aval]
    for i in range(len(text_tmp)):
        if (text_tmp[i] != ' '):
            output_tmp.append(text_tmp[i])
    for i in range(len(output_tmp)):
        if i % 5 == 0 and i > 0:
            output.append(' ')
            output.append(output_tmp[i])
    self.textBrowser.setText(''.join(output))

def Save(self):
    name = QFileDialog.getSaveFileName(self,
    "Save File")
    file = open(name[0], 'w')
    text = self.textBrowser.toPlainText()
    file.write(text)
    file.close()

class Playfair(QMainWindow):
    def __init__(self):
        super(Playfair, self).__init__()
        loadUi("cipher.ui", self)

        self.label_2.setText("Playfair Cipher")
        self.pushButton_10.clicked.connect(self.Menu)

```

```

self.pushButton_6.clicked.connect(self.Encrypt)

self.pushButton_7.clicked.connect(self.Decrypt)

self.pushButton_11.clicked.connect(self.AddFile)

self.pushButton_8.clicked.connect(self.Default)

self.pushButton_9.clicked.connect(self.Grouped)
    self.pushButton_12.clicked.connect(self.Save)

    def Menu(self):
        menu = Menu()
        widget.addWidget(menu)
        widget.setCurrentIndex(widget.currentIndex()
+ 1)

    def Encrypt(self):
        plaintext_awal = self.textEdit.toPlainText()
        plaintext_tmp = [x for x in plaintext_awal]
        plaintext = check_alphabet(plaintext_tmp)
        key_awal = self.textEdit_2.toPlainText()
        key_tmp = [x for x in key_awal]
        key = check_alphabet(key_tmp)
        keyFinal = create_kunci_playfair(key)
        plaintextFinal =
create_plaintext_playfair(plaintext)
        result = playfair_encrypt(plaintextFinal,
keyFinal)
        self.textBrowser.setText(result)

    def Decrypt(self):
        ciphertext_awal = self.textEdit.toPlainText()
        ciphertext_tmp = [x for x in ciphertext_awal]

```

```

        ciphertext = check_alphabet(ciphertext_tmp)
        key_awal = self.textEdit_2.toPlainText()
        key_tmp = [x for x in key_awal]
        key = check_alphabet(key_tmp)
        keyFinal = create_kunci_playfair(key)
        ciphertextFinal =
create_plaintext_playfair(ciphertext)
        result = playfair_decrypt(ciphertext,
keyFinal)
        self.textBrowser.setText(result)

    def AddFile(self):
        fname = QFileDialog.getOpenFileName(self,
"Choose File", "E:\Kripto\kriptomanjaa", "All Files
(*)")

        with open(fname[0], 'r') as file :
            lines = file.read().rstrip()
            self.textEdit.setPlainText(str(lines))

    def Default(self):
        output = []
        text_awal = self.textBrowser.toPlainText()
        text_tmp = [x for x in text_awal]
        for i in range(len(text_tmp)):
            if (text_tmp[i] != ' '):
                output.append(text_tmp[i])
        self.textBrowser.setText(''.join(output))

    def Grouped(self):
        output_tmp = []
        output = []
        text_awal = self.textBrowser.toPlainText()
        text_tmp = [x for x in text_awal]
        for i in range(len(text_tmp)):

```

```

        if (text_tmp[i] != ' '):
            output_tmp.append(text_tmp[i])
    for i in range(len(output_tmp)):
        if i % 5 == 0 and i > 0:
            output.append(' ')
        output.append(output_tmp[i])
    self.textBrowser.setText(''.join(output))

    def Save(self):
        name = QFileDialog.getSaveFileName(self,
"Save File")
        file = open(name[0], 'w')
        text = self.textBrowser.toPlainText()
        file.write(text)
        file.close()

class OneTimePad(QMainWindow):
    def __init__(self):
        super(OneTimePad, self).__init__()
        loadUi("cipher.ui", self)

        self.label_2.setText("One Time Pad Cipher")
        self.pushButton_10.clicked.connect(self.Menu)

self.pushButton_6.clicked.connect(self.Encrypt)

self.pushButton_7.clicked.connect(self.Decrypt)

self.pushButton_11.clicked.connect(self.AddFile)

self.pushButton_8.clicked.connect(self.Default)

self.pushButton_9.clicked.connect(self.Grouped)
        self.pushButton_12.clicked.connect(self.Save)

```

```

self.pushButton_13.clicked.connect(self.RandKey)

    def Menu(self):
        menu = Menu()
        widget.addWidget(menu)
        widget.setCurrentIndex(widget.currentIndex()
+ 1)

    def Encrypt(self):
        plaintext_awal = self.textEdit.toPlainText()
        plaintext_tmp = [x for x in plaintext_awal]
        plaintext = check_alphabet(plaintext_tmp)
        key_awal = self.textEdit_2.toPlainText()
        key_tmp = [x for x in key_awal]
        key = check_alphabet(key_tmp)
        keyFinal = create_kunci(plaintext, key)
        result = one_time_pad_encrypt(plaintext,
keyFinal)
        self.textBrowser.setText(result)

    def Decrypt(self):
        ciphertext_awal = self.textEdit.toPlainText()
        ciphertext_tmp = [x for x in ciphertext_awal]
        ciphertext = check_alphabet(ciphertext_tmp)
        key_awal = self.textEdit_2.toPlainText()
        key_tmp = [x for x in key_awal]
        key = check_alphabet(key_tmp)
        keyFinal = create_kunci(ciphertext, key)
        result = one_time_pad_decrypt(ciphertext,
keyFinal)
        self.textBrowser.setText(result)

    def AddFile(self):

```

```

        fname = QFileDialog.getOpenFileName(self,
"Choose File", "E:\Kripto\kriptomanjaa", "All Files
(*)")

        with open(fname[0], 'r') as file :
            lines = file.read().rstrip()
            self.textEdit.setPlainText(str(lines))

def Default(self):
    output = []
    text_awal = self.textBrowser.toPlainText()
    text_tmp = [x for x in text_awal]
    for i in range(len(text_tmp)):
        if (text_tmp[i] != ' '):
            output.append(text_tmp[i])
    self.textBrowser.setText(''.join(output))

def Grouped(self):
    output_tmp = []
    output = []
    text_awal = self.textBrowser.toPlainText()
    text_tmp = [x for x in text_awal]
    for i in range(len(text_tmp)):
        if (text_tmp[i] != ' '):
            output_tmp.append(text_tmp[i])
    for i in range(len(output_tmp)):
        if i % 5 == 0 and i > 0:
            output.append(' ')
            output.append(output_tmp[i])
    self.textBrowser.setText(''.join(output))

def Save(self):
    name = QFileDialog.getSaveFileName(self,
"Save File")
    file = open(name[0], 'w')

```

```

        text = self.textBrowser.toPlainText()
        file.write(text)
        file.close()

    def RandKey(self):
        fname = QFileDialog.getOpenFileName(self,
        "Choose File", "E:\Kripto\kriptomanjaa", "All Files
        (*)")

        with open(fname[0], 'r') as file :
            lines = file.read().rstrip()
            tmp = [x for x in lines]
            tmp2 = []
            randomnumber = random.randint(0, 19999)
            plaintextlen = len([x for x in
self.textEdit.toPlainText()])
            for i in range(plaintextlen):
                tmp2.append(tmp[(randomnumber + i) %
20000])

            str1 = ""
            tmp3 = str1.join(tmp2)
            self.textEdit_2.setPlainText(str(tmp3))

class Enigma(QMainWindow):
    def __init__(self):
        super(Enigma, self).__init__()
        loadUi("cipher.ui", self)

        self.label_2.setText("Enigma Cipher")
        self.pushButton_10.clicked.connect(self.Menu)

        self.pushButton_6.clicked.connect(self.Encrypt)

        self.pushButton_7.clicked.connect(self.Decrypt)

```



```
self.pushButton_11.clicked.connect(self.AddFile)

self.pushButton_8.clicked.connect(self.Default)

self.pushButton_9.clicked.connect(self.Grouped)


    def Menu(self):
        menu = Menu()
        widget.addWidget(menu)
        widget.setCurrentIndex(widget.currentIndex()
+ 1)


    def Encrypt(self):
        print("")


    def Decrypt(self):
        print("")


    def AddFile(self):
        print("")


    def Default(self):
        print("")


    def Grouped(self):
        print("")


# main
app = QApplication(sys.argv)
welcome = Menu()
widget = QtWidgets.QStackedWidget()
widget.addWidget(welcome)
widget.setFixedHeight(600)
widget.setFixedWidth(800)
```

```

widget.show()
try:
    sys.exit(app.exec_())
except:
    print("Exit Program")

```

- vigenere.py

Pada file vigenere.py berisi *code* untuk menjalankan fungsi *encrypt* dan *decrypt* menggunakan metode *vigenere*. Berikut merupakan *code* pada vigenere.py.

```

def vigenere_encrypt(plaintext, kunci_asli):
    result = []
    for i in range(len(plaintext)):
        x = ord(plaintext[i]) - 65
        y = ord(kunci_asli[i]) - 65
        result.append(chr((x + y) % 26 + 65))
    return(''.join(result))

def vigenere_decrypt(ciphertext, kunci_asli):
    result = []
    for i in range(len(ciphertext)):
        x = ord(ciphertext[i]) - 65
        y = ord(kunci_asli[i]) - 65
        result.append(chr((x - y) % 26 + 65))
    return(''.join(result))

```

- extended_vigenere.py

Pada file extended_vigenere.py berisi *code* untuk menjalankan fungsi *encrypt* dan *decrypt* menggunakan metode *extended vigenere*. Berikut merupakan *code* pada extended_vigenere.py.

```

def extended_vigenere_encrypt(plaintext,
kunci_extended):
    result = []
    for i in range(len(plaintext)):
        x = ord(plaintext[i])
        y = ord(kunci_extended[i])
        result.append(chr((x + y) % 256))
    return(''.join(result))

def extended_vigenere_decrypt(ciphertext,
kunci_extended):
    result = []
    for i in range(len(ciphertext)):
        x = ord(ciphertext[i])
        y = ord(kunci_extended[i])
        result.append(chr((x - y) % 256))
    return(''.join(result))

```

- playfair.py

Pada file playfair.py berisi *code* untuk menjalankan fungsi *encrypt* dan *decrypt* menggunakan metode *playfair*. Berikut merupakan *code* pada playfair.py.

```

def create_kunci_playfair(kunci):
    tmp = []

    # Menghilangkan spasi, 'J', huruf redundan
    for i in range(len(kunci)):
        if (kunci[i] != ' ' and kunci[i] != 'J'):
            found = False
            for j in range(len(tmp)):
                if (kunci[i] == tmp[j]):
                    found = True

```

```

        break
    if not(found):
        tmp.append(kunci[i])

# Menambahkan huruf yang belum ada
for i in range(26):
    found = False
    for j in range(len(tmp)):
        if (tmp[j] == chr(i + 65)):
            found = True
    if not(found):
        if (chr(i + 65) != 'J'):
            tmp.append(chr(i + 65))

# Menjadikan matrix
result = []
for i in range(5):
    tmp2 = []
    for j in range(5):
        tmp2.append(tmp[5 * i + j])
    result.append(tmp2)

return(result)

def create_plaintext_playfair(plaintext):
    tmp = []
    result = []

    # Menghilangkan spasi, mengganti 'J' menjadi 'I'
    for i in range(len(plaintext)):
        if (plaintext[i] == 'J'):
            tmp.append('I')
        elif (plaintext[i] != ' '):
            tmp.append(plaintext[i])

```

```

# Merancang plaintext playfair
i = 0
while(i < len(tmp) - 1):
    first = tmp[i]
    second = tmp[i + 1]
    if (first == second):
        result.append(first)
        result.append('X')
        i += 1
    else:
        result.append(first)
        result.append(second)
        i += 2
if (i != len(tmp)):
    result.append(tmp[len(tmp) - 1])
    result.append('X')

return(result)

def playfair_encrypt(plaintext, kunci):
    result = []
    for i in range(int(len(plaintext) / 2)):
        x1 = 0
        y1 = 0
        x2 = 0
        y2 = 0
        for j in range(5):
            for k in range(5):
                if (plaintext[i * 2] == kunci[j][k]):
                    x1 = j
                    y1 = k
                if (plaintext[i * 2 + 1] ==
kunci[j][k]):

```

```

        x2 = j
        y2 = k
    if (x1 == x2 and y1 != y2):
        result.append(kunci[x1][(y1 + 1) % 5])
        result.append(kunci[x2][(y2 + 1) % 5])
    elif (x1 != x2 and y1 == y2):
        result.append(kunci[(x1 + 1) % 5][y1])
        result.append(kunci[(x2 + 1) % 5][y2])
    else:
        result.append(kunci[x1][y2])
        result.append(kunci[x2][y1])
    return(''.join(result))

def playfair_decrypt(ciphertext, kunci):
    result = []
    for i in range(int(len(ciphertext) / 2)):
        x1 = 0
        y1 = 0
        x2 = 0
        y2 = 0
        for j in range(5):
            for k in range(5):
                if (ciphertext[i * 2] ==
kunci[j][k]):
                    x1 = j
                    y1 = k
                    if (ciphertext[i * 2 + 1] ==
kunci[j][k]):
                        x2 = j
                        y2 = k
                    if (x1 == x2 and y1 != y2):
                        result.append(kunci[x1][(y1 - 1 + 5) %
5])
                        result.append(kunci[x2][(y2 - 1 + 5) %

```

```

5])
        elif (x1 != x2 and y1 == y2):
            result.append(kunci[(x1 - 1 + 5) %
5][y1])
            result.append(kunci[(x2 - 1 + 5) %
5][y2])
        else:
            result.append(kunci[x1][y2])
            result.append(kunci[x2][y1])
    return(''.join(result))

```

- one_time_pad.py

Pada file one_time_pad.py berisi *code* untuk menjalankan fungsi *encrypt* dan *decrypt* menggunakan metode *one time pad*. Berikut merupakan *code* pada one_time_pad.py.

```

def one_time_pad_encrypt(plaintext, kunci):
    result = []
    for i in range(len(plaintext)):
        result.append(chr(((int(ord(plaintext[i])) -
65) + (int(ord(kunci[i])) - 65)) % 26 + 65))
    return(''.join(result))

def one_time_pad_decrypt(ciphertext, kunci):
    result = []
    for i in range(len(ciphertext)):
        result.append(chr(((int(ord(ciphertext[i])) -
65) - (int(ord(kunci[i])) - 65)) % 26 + 65))
    return(''.join(result))

```

- enigma.py

Pada file `enigma.py` berisi *code* untuk menjalankan fungsi *encrypt* dan *decrypt* menggunakan metode *enigma*. Berikut merupakan *code* pada `enigma.py`.

```
# Karena terdapat terlalu banyak jenis mesin enigma,
# digunakan sistem yang diajarkan di kelas
# dengan setting wiring masing-masing rotor yang
# didapat dari
# https://en.wikipedia.org/wiki/Enigma\_rotor\_details

default = []
for i in range(26):
    default.append(chr(i + 65))

#           = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
wiring1 = "DMTWSILRUYQNKFEJCAZBPGXOHV"
wiring2 = "HQZGPJTMOBLNCIFDYAWVEUSRKX"
wiring3 = "UQNTLSZFMREHDPXKIBVYGJCWOA"

rotor1 = [x for x in wiring1]
rotor2 = [x for x in wiring2]
rotor3 = [x for x in wiring3]

position1 = 1
position2 = 1
position3 = 1

ring1 = 1
ring2 = 1
ring3 = 1

notch1 = 7
notch2 = 3
notch3 = 2
```



```

plaintext =
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAA"
plaintext = [x for x in plaintext]

def find_coordinate(target, plaintext, default):
    for j in range(26):
        if (target == default[j]):
            return j

for i in range(len(plaintext)):
    plaintext[i] =
rotor1[find_coordinate(plaintext[i], plaintext,
default)]
    plaintext[i] =
rotor2[find_coordinate(plaintext[i], plaintext,
default)]
    plaintext[i] =
rotor3[find_coordinate(plaintext[i], plaintext,
default)]

    tmp = []
    for j in range(25):
        tmp.append(rotor1[j + 1])
    tmp.append(rotor1[0])

    rotor1 = tmp

    if (position2 == notch2 and position1 == notch1):
        position3 = (position3 + 1) % 26

    if (position1 == notch1):

```

```

        position2 = (position2 + 1) % 26

    position1 = (position1 + 1) % 26

    print(position1, position2, position3)

print(plaintext)

```

- check.py

Pada file check.py berisi *code* untuk menjalankan fungsi tambahan agar pada main.py, *code* tidak terlalu ribet dan panjang. Berikut merupakan *code* pada check.py.

```

def check_alphabet(text):
    checked = []
    for i in range(len(text)):
        for j in range(26):
            if (text[i] == chr(j + 65)):
                checked.append(text[i])
            elif (text[i] == chr(j + 97)):
                checked.append(chr(j + 65))
    return(checked)

def create_kunci(plaintext, kunci):
    kunci_asli = []
    for i in range(len(plaintext)):
        kunci_asli.append(kunci[i % len(kunci)])
    return(kunci_asli)

```

B. Interface

Berikut merupakan tampilan antarmuka yang dibuat menggunakan PyQt. Terdapat *homepage* yang berfungsi sebagai tampilan utama yang memperbolehkan *user* untuk memilih metode *cipher* yang diinginkan. Tiap metode *cipher* memiliki tampilannya masing-masing. Berikut hasil tampilan dari program tugas 1 kriptografi klasik.



Gambar 1. *Homepage*

python

Home

Vigenere Cipher

Encrypt Decrypt

Text Add File

Key Random Key

Without Space Group 5 Letters

Result Save

Gambar 2. *Vigenere Page*

python

Home

Extend Vigenere Cipher

Encrypt Decrypt

Text Add File

Key Random Key

Without Space Group 5 Letters

Result Save

Gambar 3. *Extended Vigenere Page*

python

Home

Playfair Cipher

Encrypt Decrypt

Text Add File

Key Random Key

Without Space Group 5 Letters

Result Save

Gambar 4. *Playfair Page*

python

Home

One Time Pad Cipher

Encrypt Decrypt

Text Add File

Key Random Key

Without Space Group 5 Letters

Result Save

Gambar 5. *One Time Pad Page*

python

Home

Enigma Cipher

Encrypt Decrypt

Text Add File

Key Random Key

Without Space Group 5 Letters

Result Save

Gambar 6. *Enigma Page*

C. Testing

Setelah itu, dilakukan pengujian pada setiap metode dengan masukan yang berbeda-beda. *Testing* dilakukan dengan metode *black box*. Berikut hasil pengujian dan hasil yang didapatkan.

Tabel 1. Hasil Uji Enkripsi

No	Spek	Plain Text	Key	Cipher Text	Hasil Uji	Keterangan
1.	Vigenere	THIS IS PLAIN TEXT	SONY	LVVQAG CJSWAR WLK	Berhasil	Program dapat melakukan proses enkripsi vigenere dengan benar dengan mengabaikan spasi (baik dari ketikan maupun inputan <i>file</i> dan bisa menyimpan hasilnya pada sebuah <i>file</i>).
		THIS IS PLAIN TEXT	SONY	LVVQA GCJSW ARWLK	Berhasil	Program dapat melakukan proses enkripsi vigenere dengan benar dan mengeluarkan output <i>cipher text</i> dengan <i>group of 5 letters</i> (baik dari ketikan maupun inputan <i>file</i> dan bisa menyimpan hasilnya pada sebuah <i>file</i>).
2.	Extended Vigenere	THIS is PLAIN text	SoNy	§.—İsØÁ ™£»Â¡ÂÞ Ëä	Berhasil	Program dapat melakukan proses enkripsi extended vigenere dengan benar dengan mengabaikan spasi (baik dari ketikan

						maupun inputan <i>file</i> dan bisa menyimpan hasilnya pada sebuah <i>file</i>).
		THIS is PLAIN text	SoNy	§.—İs ØÁ™£» ÂĴÂ ÞËă	Berhasil	Program dapat melakukan proses enkripsi extended vigenere dengan benar dan mengeluarkan output <i>cipher text</i> dengan <i>group of 5 letters</i> (baik dari ketikan maupun inputan <i>file</i> dan bisa menyimpan hasilnya pada sebuah <i>file</i>).
3.	Playfair	TEMUI IBU NANTI MALAM	JALAN GANESH A SEPULU H	ZBRSFY KUPGLG RKVSNL QV	Berhasil	Program dapat melakukan proses enkripsi playfair dengan benar dengan mengabaikan spasi (baik dari ketikan maupun inputan <i>file</i> dan bisa menyimpan hasilnya pada sebuah <i>file</i>).
		TEMUI IBU NANTI MALAM	JALAN GANESH A SEPULU H	ZBRSF YKUPG LGRKV SNLQV	Berhasil	Program dapat melakukan proses enkripsi playfair dengan benar dan mengeluarkan output <i>cipher text</i> dengan <i>group of 5 letters</i> (baik dari ketikan maupun inputan <i>file</i> dan bisa menyimpan hasilnya pada

						sebuah <i>file</i>).
4.	One Time Pad	BRAND ON	QPXFVR A	RGXSYF N	Berhasil	Program dapat melakukan proses enkripsi one time pad dengan benar dengan mengabaikan spasi (baik dari ketikan maupun inputan <i>file</i> dan bisa menyimpan hasilnya pada sebuah <i>file</i>). Key pada one time pad dimasukkan dari <i>file</i> otp_key yang berisikan 20000 karakter dan diambil key secara random sepanjang plain text.
		BRAND ON	QPXFVR A	RGXSY FN	Berhasil	Program dapat melakukan proses enkripsi one time pad dengan benar dan mengeluarkan output <i>cipher text</i> dengan <i>group of 5 letters</i> (baik dari ketikan maupun inputan <i>file</i> dan bisa menyimpan hasilnya pada sebuah <i>file</i>). Key pada one time pad dimasukkan dari <i>file</i> otp_key yang berisikan 20000 karakter dan diambil key secara random sepanjang plain text.
5.	Enigma	AAAAA	wiring1 =	['Z', 'N', 'J',	Kurang	Program belum

				'W', 'I', 'T', 'Y']		
--	--	--	--	------------------------	--	--

Tabel 2. Hasil Uji Dekripsi

No	Spek	Cipher Text	Key	Plain Text	Hasil Uji	Keterangan
1.	Vigenere	LVVQA GCJSW ARWLG	SONY	THISISPL AINTEXT	Berhasil	Program dapat melakukan proses dekripsi vigenere dengan benar dengan mengabaikan spasi (baik dari ketikan maupun inputan <i>file</i> dan bisa menyimpan hasilnya pada sebuah <i>file</i>).
		LVVQA GCJSW ARWLG	SONY	THISI SPLAI NTEXT	Berhasil	Program dapat melakukan proses dekripsi vigenere dengan benar dan mengeluarkan output <i>plain text</i> dengan <i>group of 5 letters</i> (baik dari ketikan maupun inputan <i>file</i> dan bisa menyimpan hasilnya pada sebuah <i>file</i>).
2.	Extended Vigenere	§·—ÌsØ Â TM £»Â _i ÂþËã	SoNy	THISisPL AINtext	Berhasil	Program dapat melakukan proses dekripsi extended vigenere dengan benar dengan mengabaikan spasi (baik dari ketikan maupun inputan <i>file</i> dan

						bisa menyimpan hasilnya pada sebuah <i>file</i>).
		§·—ÏsØ Â™£»Â; ÂþËä	SoNy	THIS is PLAIN text	Berhasil	Program dapat melakukan proses dekripsi extended vigenere dengan benar dan mengeluarkan output <i>plain text</i> dengan <i>group of 5 letters</i> (baik dari ketikan maupun inputan <i>file</i> dan bisa menyimpan hasilnya pada sebuah <i>file</i>).
3.	Playfair	ZBRSF YKUPG LGRKV SNLQV	JALAN GANESH A SEPULU H	TEMUIXI BUNANT IMALAM X	Berhasil	Program dapat melakukan proses dekripsi playfair dengan benar dengan mengabaikan spasi (baik dari ketikan maupun inputan <i>file</i> dan bisa menyimpan hasilnya pada sebuah <i>file</i>).
		ZBRSF YKUPG LGRKV SNLQV	JALAN GANESH A SEPULU H	TEMUI XIBUN ANTIM ALAMX	Berhasil	Program dapat melakukan proses dekripsi playfair dengan benar dan mengeluarkan output <i>plain text</i> dengan <i>group of 5 letters</i> (baik dari ketikan maupun inputan <i>file</i> dan bisa menyimpan hasilnya pada sebuah <i>file</i>).

4.	One Time Pad	RGXSY FN	QPXFVR A	BRAND O N	Berhasil	Program dapat melakukan proses dekripsi one time pad dengan benar dengan mengabaikan spasi (baik dari ketikan maupun inputan <i>file</i> dan bisa menyimpan hasilnya pada sebuah <i>file</i>). Key pada one time pad dimasukkan dari <i>file</i> otp_key yang berisikan 20000 karakter dan diambil key secara random sepanjang cipher text.
		RGXSY FN	QPXFVR A	BRAND O N	Berhasil	Program dapat melakukan proses dekripsi playfair dengan benar dan mengeluarkan output <i>plain text</i> dengan <i>group of 5 letters</i> (baik dari ketikan maupun inputan <i>file</i> dan bisa menyimpan hasilnya pada sebuah <i>file</i>). Key pada one time pad dimasukkan dari <i>file</i> otp_key yang berisikan 20000 karakter dan diambil key secara random sepanjang cipher text.
5.	Enigma	-	-	-	Tidak Berhasil	Program sama sekali belum dapat melakukan proses

						dekripsi enigma cipher. Hal ini disebabkan oleh belum selesainya proses enkripsi.
--	--	--	--	--	--	---

Tabel 3. Hasil Uji Keseluruhan

No	Spek	Berhasil (✓)	Kurang Berhasil (✓)	Keterangan
1.	Vigenere	✓		Fungsi dapat melakukan enkripsi dan dekripsi text dari ketikan maupun masukan <i>file</i> yang berisi teks. Fungsi juga dapat melakukan save hasil enkripsi maupun dekripsi.
2.	Extended Vigenere		✓	Fungsi dapat melakukan enkripsi dan dekripsi text dari ketikan maupun masukan <i>file</i> yang berisi teks. Namun, fungsi belum dapat melakukan <i>encode</i> dan <i>decode</i> dari file lainnya. Fungsi juga dapat melakukan save hasil enkripsi maupun dekripsi.
3.	Playfair	✓		Fungsi dapat melakukan enkripsi dan dekripsi text dari ketikan maupun

				masukan <i>file</i> yang berisi teks. Fungsi juga dapat melakukan save hasil enkripsi maupun dekripsi.
4.	One Time Pad	✓		Fungsi dapat melakukan enkripsi dan dekripsi text dari ketikan maupun masukan <i>file</i> yang berisi teks. Fungsi dapat membaca masukan <i>file</i> otp_key.txt dan secara otomatis <i>generate key</i> sepanjang text yang dimasukan. Fungsi juga dapat melakukan save hasil enkripsi maupun dekripsi.
5.	Enigma		✓	Fungsi belum dapat melakukan proses enkripsi secara sempurna dan belum sama sekali dapat melakukan proses dekripsi. Fungsi juga belum dapat ditampilkan pada GUI. Hal ini disebabkan oleh kurangnya memahami konsep dari enigma cipher.

D. Github

Berikut merupakan *link* Github yang dipakai dalam pengerjaan tugas kali ini

<https://github.com/brandonjonathann/kriptomanjaa>