Naufal Yahya Kurnianto 13519141
IF4020 Kriptografi

Laporan Tugas Kecil 1 Kriptografi

1. Source Code

Src.py

```
import numpy as np
import re

def outputmaker(a):
    return ' '.join([a[i:i + 5] for i in range(0, len(a), 5)])

def keymaker(message,key):
    keychar = []
    if (len(message)>(len(key))):
        for i in range(len(message)):
            keychar.insert(len(keychar),ord(key[i%len(key)])%97)
    else:
        for characters in key:
            keychar.insert(len(keychar),ord(characters)%97)

    return keychar

def autokeymaker(message,key):
    keychar = []
    if (len(message)>(len(key))):
        for i in range(len(key)):
            keychar.insert(len(keychar),ord(key[i])%97)
        for i in range(len(key),len(message)):
            keychar.insert(len(keychar),ord(message[i-len(key)])%97)
    else:
        for characters in key:
            keychar.insert(len(keychar),ord(characters)%97)

    return keychar
```

```python
def playfairkeymaker(key):
    key.replace('j','')
    keychar = []
    for i in range(len(key)):
        if (ord(key[i])%97) in keychar:
            continue
        else:
            keychar.insert(len(keychar),ord(key[i])%97)
    toInsert = 0
    while(len(keychar)<25):
        if (toInsert==9):
            toInsert += 1
            continue
        if (toInsert not in keychar):
            keychar.insert(len(keychar),toInsert)
        toInsert += 1

    pfkey = np.reshape(keychar,(5,5))
    return pfkey

def playfairtextmaker(message):
    message.replace('j','i')
    messagechar = []
    for characters in message:
        messagechar.insert(len(messagechar),ord(characters)%97)

    for i in range(1,len(messagechar)):
        if (messagechar[i]==messagechar[i-1]):
            messagechar.insert(i,23)
    if (len(messagechar)%2==1):
        messagechar.insert(len(messagechar),23)

    pftext = np.reshape(messagechar,(-1,2))
    return pftext
```

```python
def vigenere_encrypt(message,key):
    keychar = keymaker(message,key)
    messagechar = []
    for characters in message:
        messagechar.insert(len(messagechar),ord(characters)%97)

    encrypted = messagechar
    for i in range(len(encrypted)):
        encrypted[i] = vigenere_matrice[keychar[i]][messagechar[i]]

    return encrypted

def full_vigenere_encrypt(message,key):
    np.random.shuffle(full_vigenere_matrice)
    keychar = keymaker(message,key)
    messagechar = []
    for characters in message:
        messagechar.insert(len(messagechar),ord(characters)%97)

    encrypted = messagechar
    for i in range(len(encrypted)):
        encrypted[i] = full_vigenere_matrice[keychar[i]][messagechar[i]]

    return encrypted

def autokey_vigenere_encrypt(message,key):
    keychar = autokeymaker(message,key)
    messagechar = []
    for characters in message:
        messagechar.insert(len(messagechar),ord(characters)%97)

    encrypted = messagechar
    for i in range(len(encrypted)):
        encrypted[i] = vigenere_matrice[keychar[i]][messagechar[i]]

    return encrypted
```

```python
def playfair_calculate(row,pfkey,encrypt):
    position_matrice = np.zeros((2,2), dtype=int)
    x, y, i, j, loop = 0, 0, 0, 0, 0
    while(loop!=2):
        if (j==5):
            i += 1
            j = -1
        elif (pfkey[i][j]==row[loop]):
            position_matrice[x][y] = i
            y += 1
            position_matrice[x][y] = j
            x += 1
            y, i, j = 0, 0, -1
            loop += 1
        j += 1
    if (encrypt):
        if (position_matrice[0][0]==position_matrice[1][0]):
            position_matrice[0][1] = (position_matrice[0][1] + 1) % 5
            position_matrice[1][1] = (position_matrice[1][1] + 1) % 5
        elif (position_matrice[0][1]==position_matrice[1][1]):
            position_matrice[0][0] = (position_matrice[0][0] + 1) % 5
            position_matrice[1][0] = (position_matrice[1][0] + 1) % 5
        else:
            swap = position_matrice[0][1]
            position_matrice[0][1] = position_matrice[1][1]
            position_matrice[1][1] = swap
    else:
        if (position_matrice[0][0]==position_matrice[1][0]):
            position_matrice[0][1] = (position_matrice[0][1] - 1) % 5
            position_matrice[1][1] = (position_matrice[1][1] - 1) % 5
        elif (position_matrice[0][1]==position_matrice[1][1]):
            position_matrice[0][0] = (position_matrice[0][0] - 1) % 5
            position_matrice[1][0] = (position_matrice[1][0] - 1) % 5
        else:
            swap = position_matrice[0][1]
            position_matrice[0][1] = position_matrice[1][1]
            position_matrice[1][1] = swap

    newrow = row
    newrow[0] = pfkey[position_matrice[0][0]][position_matrice[0][1]]
    newrow[1] = pfkey[position_matrice[1][0]][position_matrice[1][1]]
    return newrow
```

```
def playfair(pftext,pfkey,encrypt):
    procctext = pftext
    for i in range(len(pftext)):
        procctext[i] = playfair_calculate(pftext[i],pfkey,encrypt)

    result = np.reshape(procctext,procctext.size)
    return result

def affine_encrypt(message,m,b):
    encryptchar = []
    for characters in message:
        encryptchar.insert(len(encryptchar),(((ord(characters)%97)*m)+b)%26
)

    return encryptchar

def vigenere_decrypt(message,key):
    keychar = keymaker(message,key)
    messagechar = []
    for characters in message:
        messagechar.insert(len(messagechar),ord(characters)%97)

    decrypted = messagechar
    for i in range(len(decrypted)):
        decrypted[i] = vigenere_matrice[(keychar[i]*-1)%26][messagechar[i]]

    return decrypted

def full_vigenere_decrypt(message,key):
    keychar = keymaker(message,key)
    messagechar = []
    for characters in message:
        messagechar.insert(len(messagechar),ord(characters)%97)

    decrypted = messagechar
    for i in range(len(decrypted)):
        for j in range(26):
            if(messagechar[i]==full_vigenere_matrice[keychar[i]][j]):
                decrypted[i] = j
                break

    return decrypted
```

```
def autokey_vigenere_decrypt(plaintext,message,key):
    keychar = autokeymaker(plaintext,key)
    messagechar = []
    for characters in message:
        messagechar.insert(len(messagechar),ord(characters)%97)

    decrypted = messagechar
    for i in range(len(decrypted)):
        for j in range(26):
            if(messagechar[i]==vigenere_matrice[keychar[i]][j]):
                decrypted[i] = j
                break

    return decrypted

def inverse_mod(base,m):
    for i in range(1,base):
        if (((i*m)%base)==1):
            return i

def affine_decrypt(message,m,b):
    decryptchar = []
    for characters in message:
        decryptchar.insert(len(decryptchar),((inverse_mod(26,m)*((ord(chara
cters)%97)-b))%26))

    return decryptchar

vigenere_matrice = \
    np.array([[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25],
              [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,0],
              [2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,0,1],
              [3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,0,1,2],
              [4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,0,1,2,3],
              [5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,0,1,2,3,4],
              [6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,0,1,2,3,4,5],
              [7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,0,1,2,3,4,5,6],
              [8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,0,1,2,3,4,5,6,7],
              [9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,0,1,2,3,4,5,6,7,8],
              [10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,0,1,2,3,4,5,6,7,8,9],
              [11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,0,1,2,3,4,5,6,7,8,9,10],
              [12,13,14,15,16,17,18,19,20,21,22,23,24,25,0,1,2,3,4,5,6,7,8,9,10,11],
              [13,14,15,16,17,18,19,20,21,22,23,24,25,0,1,2,3,4,5,6,7,8,9,10,11,12],
              [14,15,16,17,18,19,20,21,22,23,24,25,0,1,2,3,4,5,6,7,8,9,10,11,12,14],
              [15,16,17,18,19,20,21,22,23,24,25,0,1,2,3,4,5,6,7,8,9,10,11,12,14,15],
              [16,17,18,19,20,21,22,23,24,25,0,1,2,3,4,5,6,7,8,9,10,11,12,14,15,16],
              [17,18,19,20,21,22,23,24,25,0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16],
             [18,19,20,21,22,23,24,25,0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17],
              [19,20,21,22,23,24,25,0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18],
              [20,21,22,23,24,25,0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19],
              [21,22,23,24,25,0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],
```

```python
vigenere_matrice = \
    np.array([[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25],
              [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,0],
              [2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,0,1],
              [3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,0,1,2],
              [4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,0,1,2,3],
              [5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,0,1,2,3,4],
              [6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,0,1,2,3,4,5],
              [7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,0,1,2,3,4,5,6],
              [8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,0,1,2,3,4,5,6,7],
              [9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,0,1,2,3,4,5,6,7,8],
              [10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,0,1,2,3,4,5,6,7,8,9],
              [11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,0,1,2,3,4,5,6,7,8,9,10],
              [12,13,14,15,16,17,18,19,20,21,22,23,24,25,0,1,2,3,4,5,6,7,8,9,10,11],
              [13,14,15,16,17,18,19,20,21,22,23,24,25,0,1,2,3,4,5,6,7,8,9,10,11,12],
              [14,15,16,17,18,19,20,21,22,23,24,25,0,1,2,3,4,5,6,7,8,9,10,11,12,14],
              [15,16,17,18,19,20,21,22,23,24,25,0,1,2,3,4,5,6,7,8,9,10,11,12,14,15],
              [16,17,18,19,20,21,22,23,24,25,0,1,2,3,4,5,6,7,8,9,10,11,12,14,15,16],
              [17,18,19,20,21,22,23,24,25,0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16],
              [18,19,20,21,22,23,24,25,0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17],
              [19,20,21,22,23,24,25,0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18],
              [20,21,22,23,24,25,0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19],
              [21,22,23,24,25,0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],
              [22,23,24,25,0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21],
              [23,24,25,0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22],
              [24,25,0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23],
              [25,0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24]])

full_vigenere_matrice = vigenere_matrice

''' akses matriks vigenere, vigenere_matrice[kunci][plainteks]'''
```

App.py

```python
from flask import Flask, render_template, request
import src as algo

app = Flask(__name__)

@app.route('/' , methods=["GET", "POST"])
def home():
    return render_template('index.html')
```

```python
@app.route('/encrypt', methods=["GET", "POST"])
def encrypt():
    if (request.method == "POST"):
        file = request.files['fileInput']
        if file.filename == '':
            cypher = request.form['methodInput']
            text = request.form['plaintextInput'].lower().replace(" ","")
            key = request.form['keyInput'].lower().replace(" ","")
            m = int(request.form['key-m'])
            b = int(request.form['key-b'])
            if (cypher=="vigenere"):
                cytext = algo.vigenere_encrypt(text,key)
                encrypted1 = ""
                for elements in cytext:
                    encrypted1 += chr(elements+97)
                encrypted = algo.outputmaker(encrypted1)
                with open("result.txt", "w") as fo:
                    fo.write(encrypted)
                return render_template("index.html", answer = encrypted, mode = "encrypted")
            elif (cypher=="full-vigenere"):
                cytext = algo.full_vigenere_encrypt(text,key)
                encrypted1 = ""
                for elements in cytext:
                    encrypted1 += chr(elements+97)
                encrypted = algo.outputmaker(encrypted1)
                with open("result.txt", "w") as fo:
                    fo.write(encrypted)
                return render_template("index.html", answer = encrypted, mode = "encrypted")
            elif (cypher=="autokey-vigenere"):
                cytext = algo.autokey_vigenere_encrypt(text,key)
                encrypted1 = ""
                for elements in cytext:
                    encrypted1 += chr(elements+97)
                encrypted = algo.outputmaker(encrypted1)
                with open("result.txt", "w") as fo:
                    fo.write(encrypted)
                return render_template("index.html", answer = encrypted, mode = "encrypted")
            elif (cypher=="playfair"):
                playfairkey = key.replace('j','')
                pfkey = algo.playfairkeymaker(playfairkey)
                pftext = algo.playfairtextmaker(text)
                pfresult = algo.playfair(pftext,pfkey,True)
                encrypted1 = ""
                for elements in pfresult:
                    encrypted1 += chr(elements+97)
                encrypted = algo.outputmaker(encrypted1)
                with open("result.txt", "w") as fo:
                    fo.write(encrypted)
```

```
            return render_template("index.html", answer = encrypted, mode = "encrypted")
        elif (cypher=="affine"):
            cytext = algo.affine_encrypt(text,m,b)
            encrypted1 = ""
            for elements in cytext:
                encrypted1 += chr(elements+97)
            encrypted = algo.outputmaker(encrypted1)
            with open("result.txt", "w") as fo:
                fo.write(encrypted)
            return render_template("index.html", answer = encrypted, mode = "encrypted")
    else:
        filetext = file.read().decode("utf-8").split("\r\n")
        text = filetext[0].lower().replace(" ","")
        key = filetext[1].lower().replace(" ","")
        m, b = 0, 0
        if (len(filetext)>2):
            m = int(filetext[2])
            b = int(filetext[3])
        cypher = request.form['methodInput']
        if (cypher=="vigenere"):
            cytext = algo.vigenere_encrypt(text,key)
            encrypted = ""
            for elements in cytext:
                encrypted += chr(elements+97)
            with open("result.txt", "w") as fo:
                fo.write(encrypted)
            return render_template("index.html", answer = encrypted, mode = "encrypted")
        elif (cypher=="full-vigenere"):
            cytext = algo.full_vigenere_encrypt(text,key)
            encrypted = ""
            for elements in cytext:
                encrypted += chr(elements+97)
            with open("result.txt", "w") as fo:
                fo.write(encrypted)
            return render_template("index.html", answer = encrypted, mode = "encrypted")
        elif (cypher=="autokey-vigenere"):
            cytext = algo.autokey_vigenere_encrypt(text,key)
            encrypted = ""
            for elements in cytext:
                encrypted += chr(elements+97)
            with open("result.txt", "w") as fo:
                fo.write(encrypted)
            return render_template("index.html", answer = encrypted, mode = "encrypted")
        elif (cypher=="playfair"):
            playfairkey = key.replace('j','')
            pfkey = algo.playfairkeymaker(playfairkey)
            pftext = algo.playfairtextmaker(text)
            pfresult = algo.playfair(pftext,pfkey,True)
```

```
                encrypted = ""
                for elements in pfresult:
                    encrypted += chr(elements+97)
                with open("result.txt", "w") as fo:
                    fo.write(encrypted)
                return render_template("index.html", answer = encrypted, mode = "encrypted")
            elif (cypher=="affine"):
                cytext = algo.affine_encrypt(text,m,b)
                encrypted = ""
                for elements in cytext:
                    encrypted += chr(elements+97)
                with open("result.txt", "w") as fo:
                    fo.write(encrypted)
                return render_template("index.html", answer = encrypted, mode = "encrypted")
    else:
        return render_template("index.html")

@app.route('/decrypt', methods=["GET", "POST"])
def decrypt():
    if (request.method == "POST"):
        cypher = request.form['methodInput']
        encrypted = request.form['cyphertextInput'].lower().replace(" ","")
        key = request.form['keyInput'].lower().replace(" ","")
        m = int(request.form['key-m'])
        b = int(request.form['key-b'])
        if (cypher=="vigenere"):
            decryptedtext = algo.vigenere_decrypt(encrypted,key)
            decrypted = ""
            for elements in decryptedtext:
                decrypted += chr(elements+97)
            return render_template("index.html", answer1 = decrypted, mode= "decrypted")
        elif (cypher=="full-vigenere"):
            decryptedtext = algo.full_vigenere_decrypt(encrypted,key)
            decrypted = ""
            for elements in decryptedtext:
                decrypted += chr(elements+97)
            return render_template("index.html", answer1 = decrypted, mode= "decrypted")
        elif (cypher=="autokey-vigenere"):
            original = request.form['plaintextInput']
            ptext = algo.autokey_vigenere_decrypt(original,encrypted,key)
            decrypted = ""
            for elements in ptext:
                decrypted += chr(elements+97)
            return render_template("index.html", answer1 = decrypted, mode= "decrypted")
        elif (cypher=="playfair"):
            playfairkey = key.replace('j','')
            pfkey = algo.playfairkeymaker(playfairkey)
            pfdecrypt = algo.playfair(algo.playfairtextmaker(encrypted),pfkey,False)
```

```
            decrypted1=""
            for elements in pfdecrypt:
                decrypted1 += chr(elements+97)
            decrypted = decrypted1.replace("x","")
            return render_template("index.html", answer1 = decrypted, mode= "decrypted")
        elif (cypher=="affine"):
            ptext = algo.affine_decrypt(encrypted,m,b)
            decrypted = ""
            for elements in ptext:
                decrypted += chr(elements+97)
            return render_template("index.html", answer1 = decrypted, mode= "decrypted")
    else:
        return render_template("index.html")
```

```
if __name__ == '__main__':
    app.run(debug=True)
```

Index.html

```
<!DOCTYPE html>
<html lang="en" dir="ltr">

  <head>
    <meta charset="utf-8">
    <title>CypherKuy</title>
    <link href="https://fonts.googleapis.com/css?family=Open+Sans" rel="stylesheet">
    <style>
      h1 {
        font-size: 45px;
        text-align: center;
        margin: 0;
        padding: 20px 0;
        color: #00539F;
        text-shadow: 3px 3px 1px black;
      }

      .identitas {
        text-align: center;
        font-size: 10px;
      }

      p {
        text-align: center;
      }
```

```
      body {
        width: 600px;
        margin: 0 auto;
        background-color: #FFFFFF;
        padding: 0 20px 20px 20px;
        border: 5px solid black;
      }

      html {
        background-
image: url("https://preview.redd.it/lr41m72m2ms41.jpg?width=1920&format=pjpg&auto=webp&
s=d4561d11381900125c44c6e78521d4afa8a35b1f");
      }

      html {
        font-size: 16px;
        font-family: "Open Sans
      }

    </style>
  </head>

  <body>
    <h1> old Cypher </h1>
    <p class="identitas">Naufal Yahya - 13519141</p>
  </body>

  {% if mode=="encrypted" %}
    <div class="answer">
      <p>cypher text anda adalah {{answer}}</p>
    </div>
    <form action="/decrypt" method="POST">
      <div class="form-group">
        <label for="originalInput">Plaintext</label>
        <br>
        <input name ="originalInput" class="form-
control" id="plaintextInput" placeholder="Enter Plaintext">
        <small id="plaintextdesc" class="form-text text-
muted">Isi untuk AutoKey Viginere</small>
      </div>
      <div class="form-group">
        <label for="cyphertextInput">Cypher text</label>
        <br>
        <input name ="cyphertextInput" class="form-
control" id="cyphertextInput" placeholder="Enter Cyphertext">
      </div>
      <div class="form-group">
        <label for="keyInput">Key</label>
```

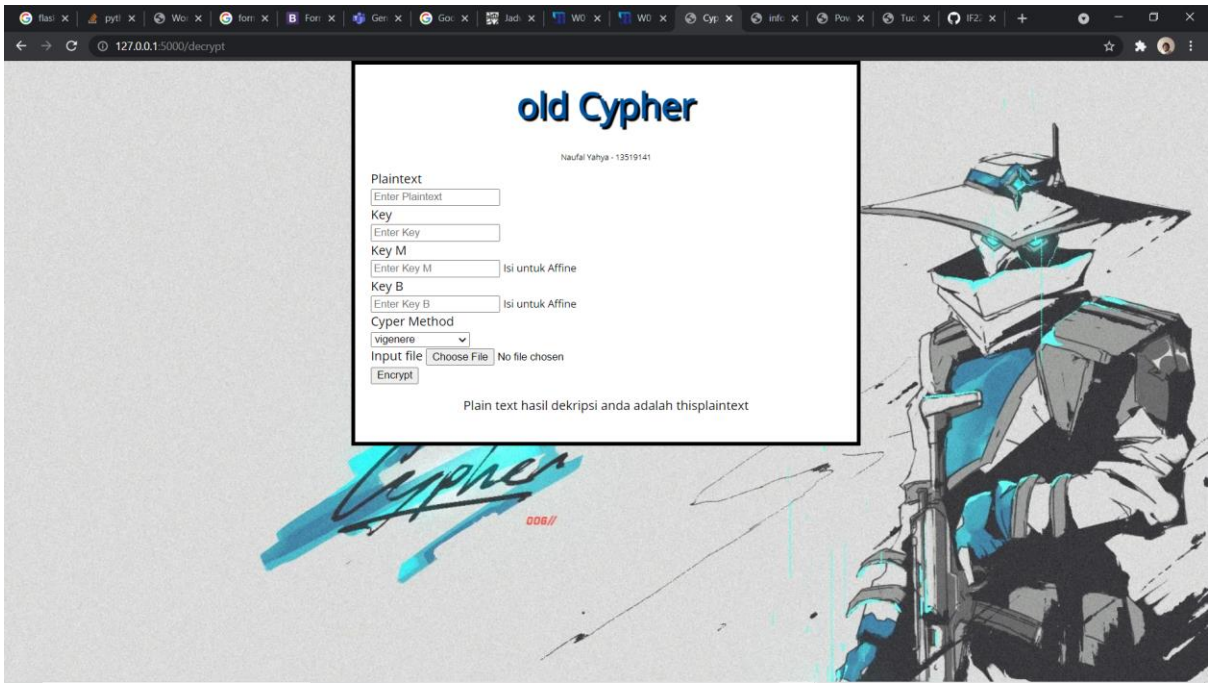
```
        <br>
        <input name ="keyInput" class="form-
control" id="keyInput" placeholder="Enter Key">
      </div>
      <div class="form-group">
        <label for="keyInput">Key M</label>
        <br>
        <input name ="key-m" class="form-
control" id="keyInput" placeholder="Enter Key M">
        <small id="additionalKey" class="form-text text-
muted">Isi untuk Affine (sama dengan enkripsi)</small>
      </div>
      <div class="form-group">
        <label for="keyInput">Key B</label>
        <br>
        <input name ="key-b" class="form-
control" id="keyInput" placeholder="Enter Key B">
        <small id="additionalKey" class="form-text text-
muted">Isi untuk Affine (sama dengan enkripsi)</small>
      </div>
      <div class="form-group">
        <label for="methodInput">Cyper Method</label>
        <br>
        <select name ="methodInput" class="form-
control" id="exampleFormControlSelect1">
          <option>vigenere</option>
          <option>full-vigenere</option>
          <option>autokey-vigenere</option>
          <option>playfair</option>
          <option>affine</option>
        </select>
        <small id="additionalKey" class="form-text text-
muted">Isi sama dengan Metode Cypher Enkripsi</small>
      </div>
      <button type="submit" class="btn">Decrypt</button>
    </form>

  {% else %}

  <form action="/encrypt" method="POST" enctype=multipart/form-data>
    <div class="form-group">
      <label for="plaintextInput">Plaintext</label>
      <br>
      <input name ="plaintextInput" class="form-
control" id="plaintextInput" placeholder="Enter Plaintext">
    </div>
    <div class="form-group">
      <label for="keyInput">Key</label>
```

```
      <br>
      <input name ="keyInput" class="form-
control" id="keyInput" placeholder="Enter Key">
    </div>
    <div class="form-group">
      <label for="keyInput">Key M</label>
      <br>
      <input name ="key-m" class="form-
control" id="keyInput" placeholder="Enter Key M">
      <small id="additionalKey" class="form-text text-muted">Isi untuk Affine</small>
    </div>
    <div class="form-group">
      <label for="keyInput">Key B</label>
      <br>
      <input name ="key-b" class="form-
control" id="keyInput" placeholder="Enter Key B">
      <small id="additionalKey" class="form-text text-muted">Isi untuk Affine</small>
    </div>
    <div class="form-group">
      <label for="methodInput">Cyper Method</label>
      <br>
      <select name ="methodInput" class="form-control" id="exampleFormControlSelect1">
        <option>vigenere</option>
        <option>full-vigenere</option>
        <option>autokey-vigenere</option>
        <option>playfair</option>
        <option>affine</option>
      </select>
    </div>
      <div class="form-group">
        <label for="FormControlFile1">Input file</label>
        <input name="fileInput" type="file" class="form-control-
file" id="FormControlFile1">
      </div>
    <button type="submit" class="btn">Encrypt</button>
  </form>
  <div class="answer">
    <p>Plain text hasil dekripsi anda adalah {{answer1}}</p>
  </div>

  {% endif %}

  <script>

  </script>

</html>
```

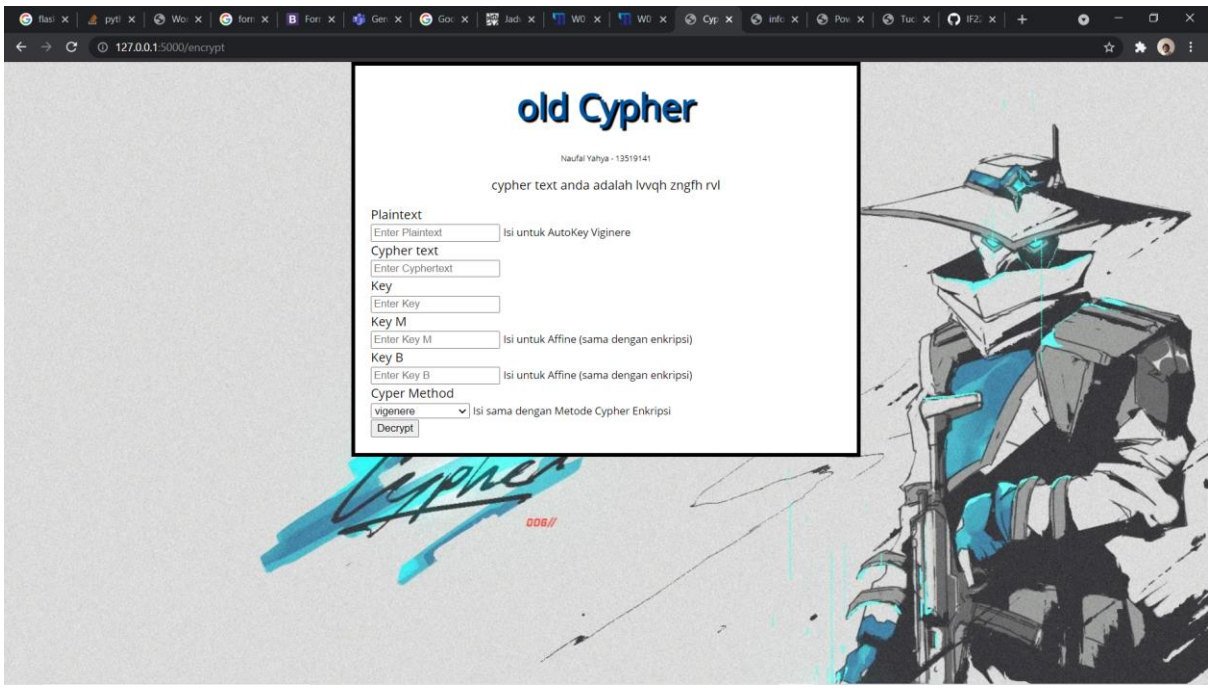
2. Tampilan Antarmuka Program

Menggunakan tampilan web based local. Index.html memiliki 2 mode yang dapat ditampilkan yaitu encrypt serta decrypt/home.

Mode Decrypt/Home

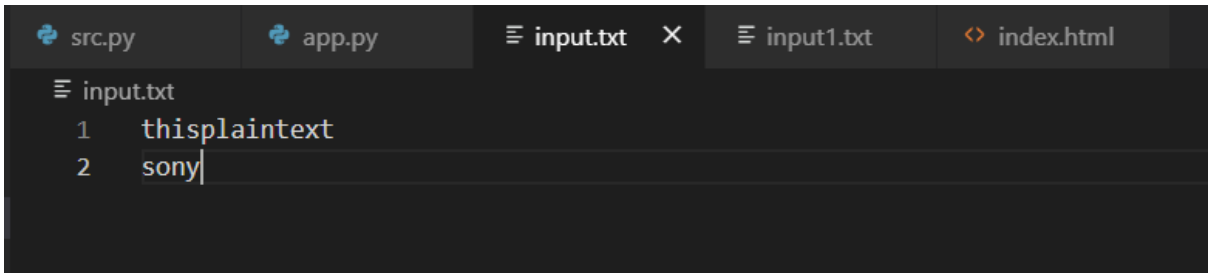


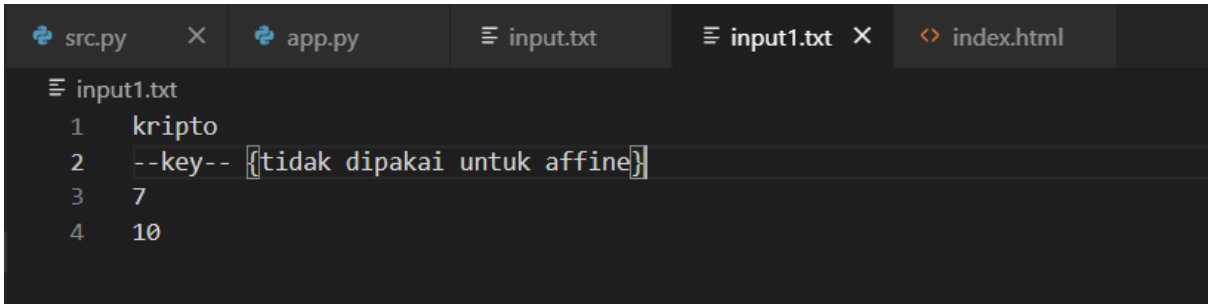
Mode Encrypt



3. Cuplikan Penggunaan

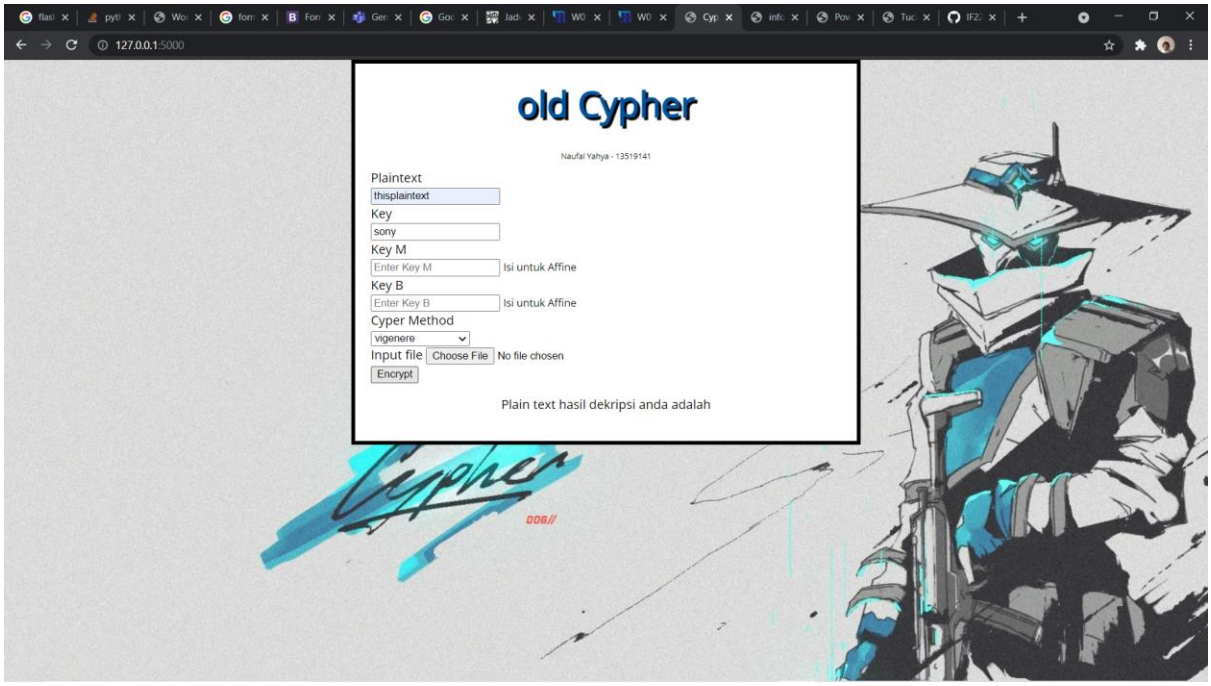
File txt yang akan digunakan ada 2 jenis, untuk affine dan untuk sisanya. Tidak ada file biner karena extended vigenere tidak selesai. Berikut 2 file txt yang ada.





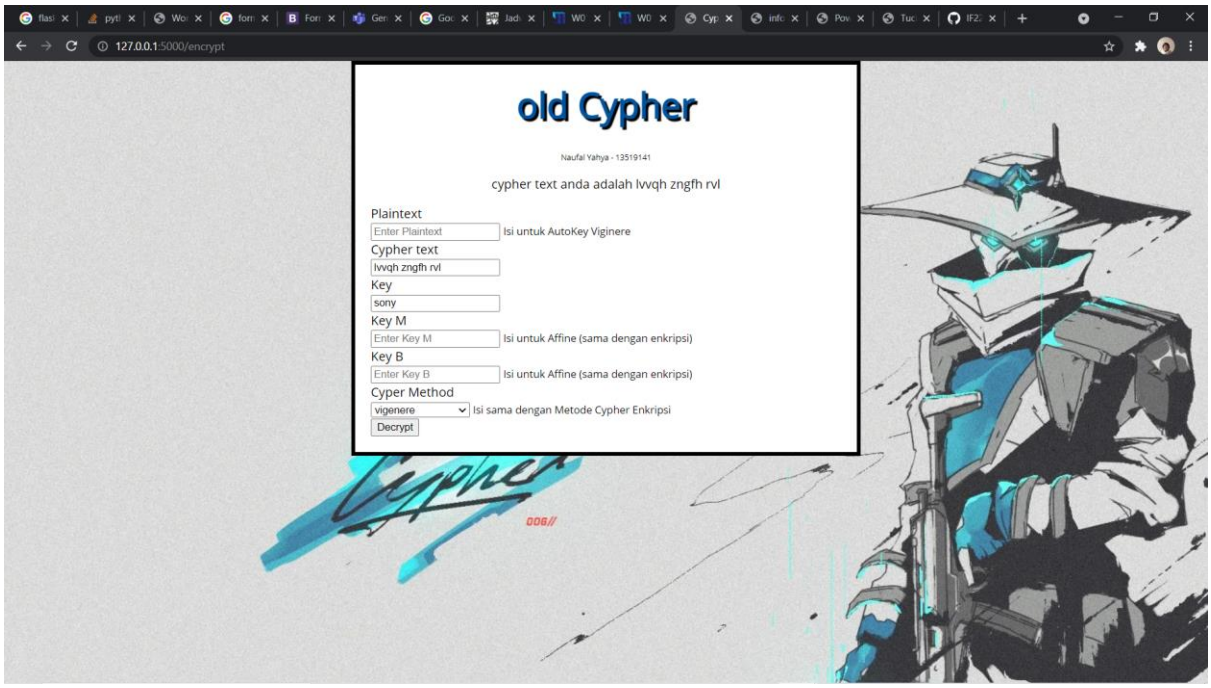
Vigenere Cypher

Sebelum Enkripsi -

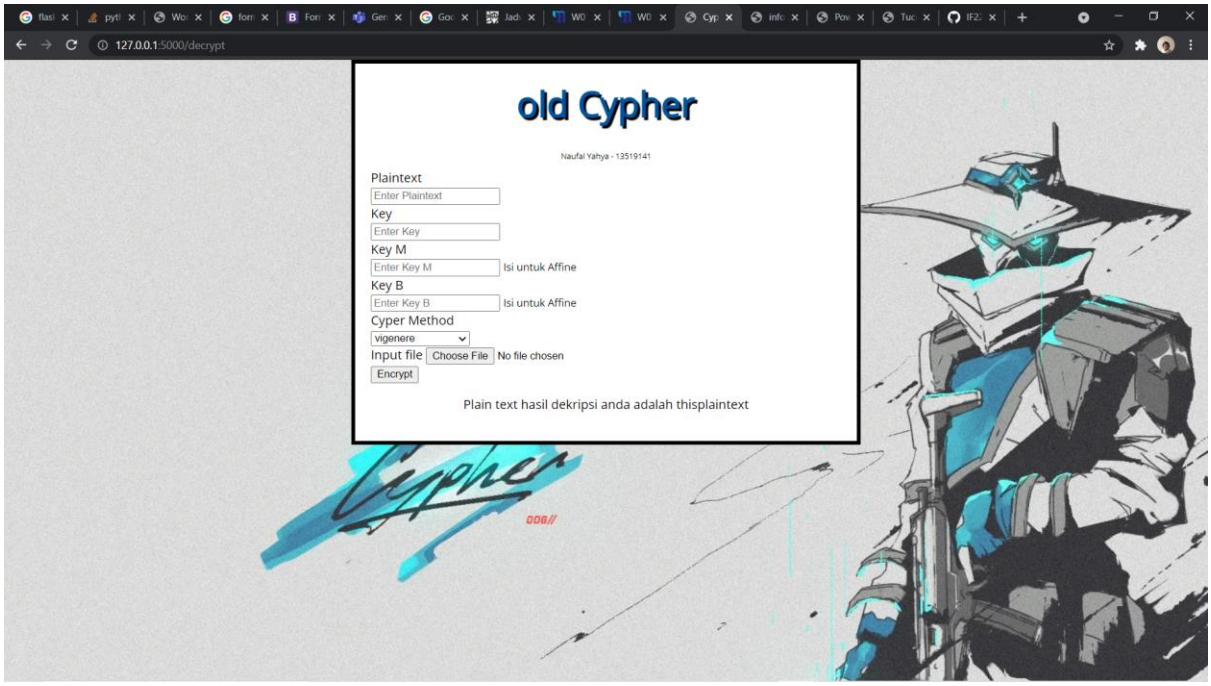


Naufal Yahya Kurnianto 13519141
IF4020 Kriptografi

Setelah Enkripsi -

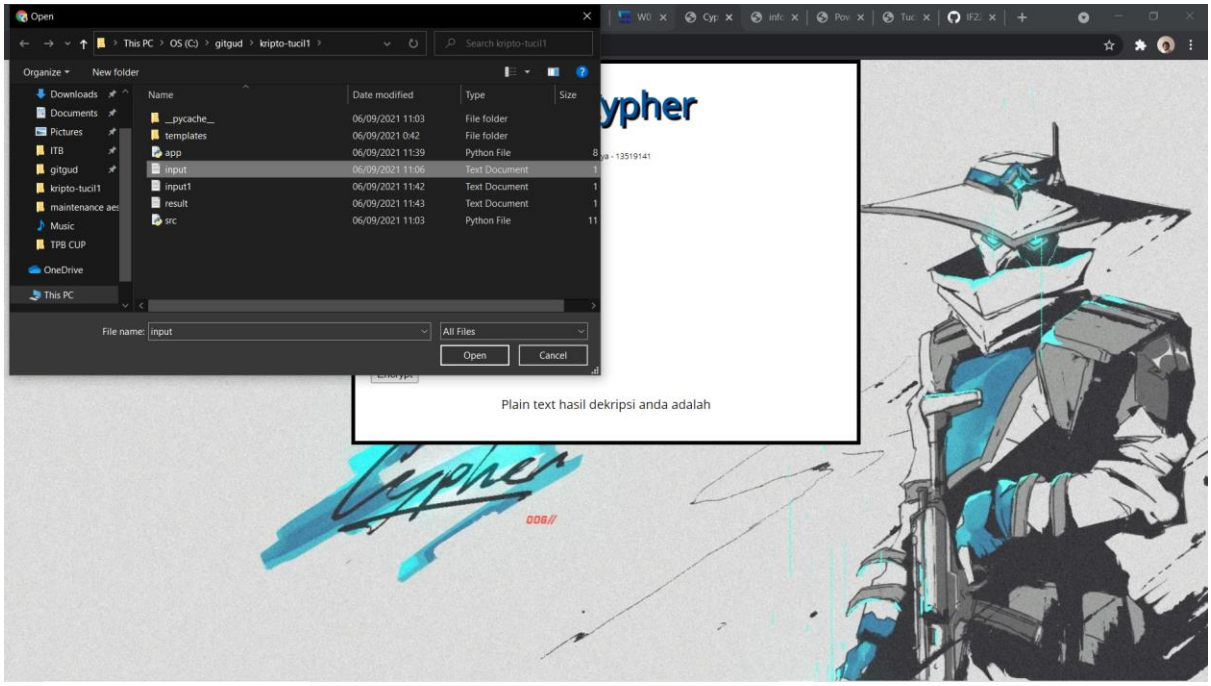


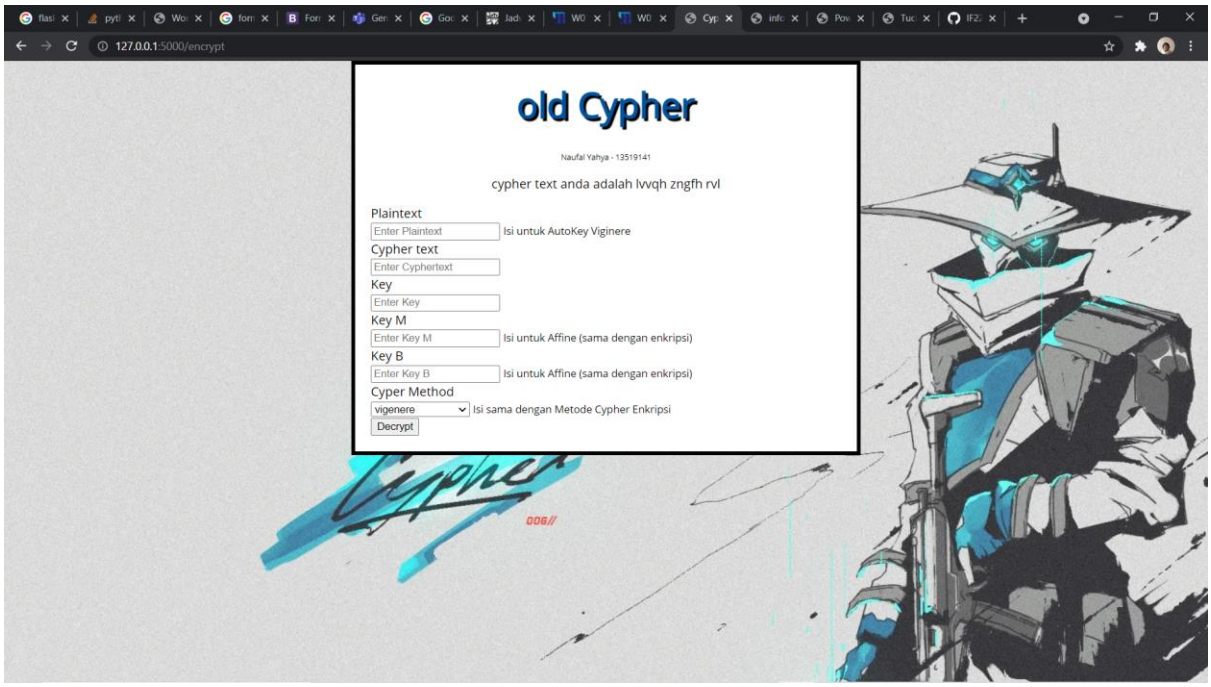
Setelah Dekripsi –



Menggunakan File

Naufal Yahya Kurnianto 13519141
IF4020 Kriptografi





Hasil keluaran dapat disimpan dalam file

```
with open("result.txt", "w") as fo:
    fo.write(encrypted)
```

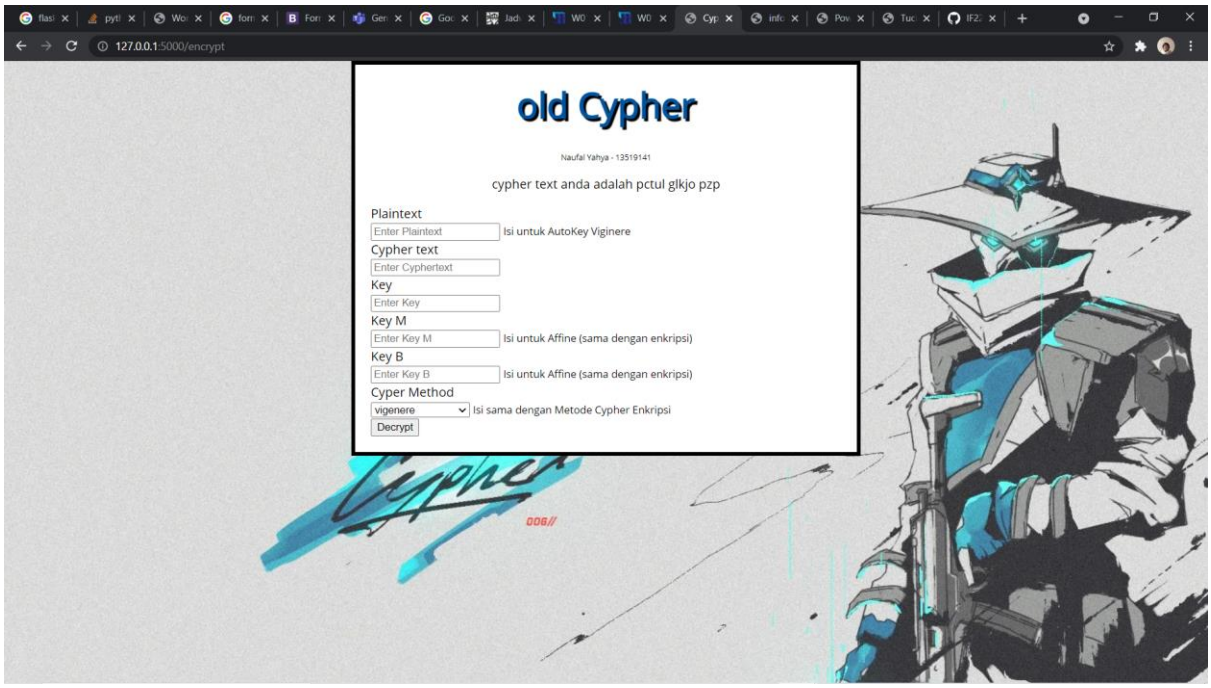
src.py | app.py | result.txt | input.txt | input1.txt | index.html

result.txt

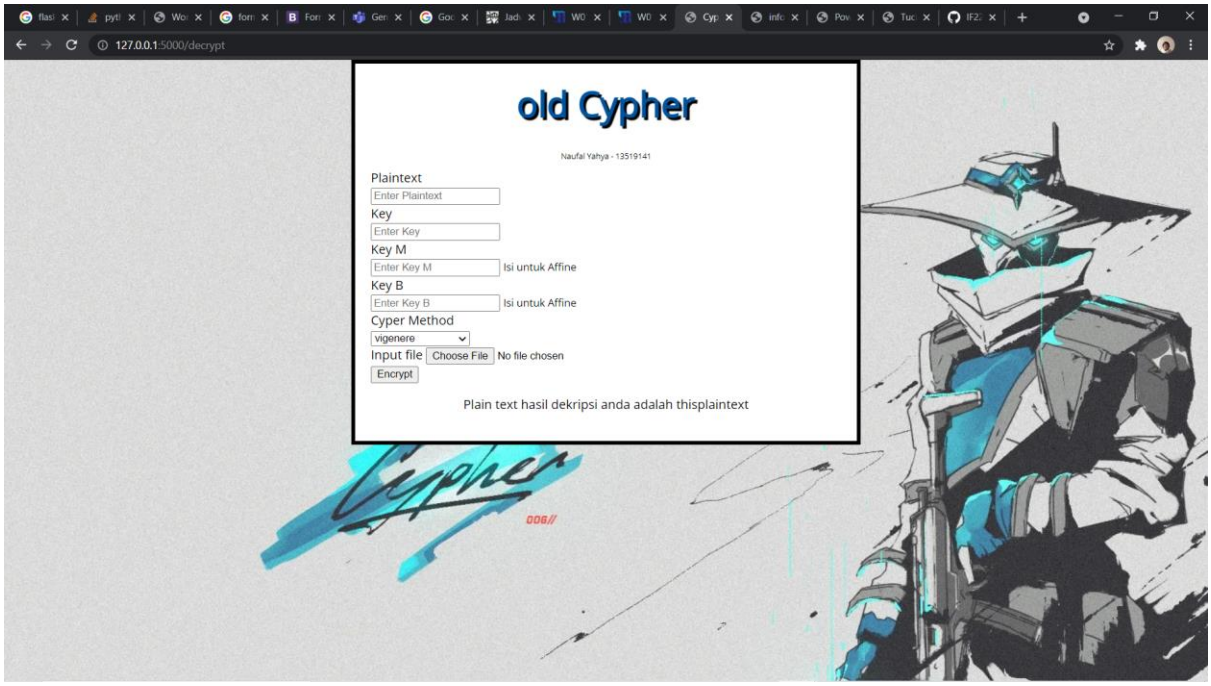
```
lvvqh zngfh rvl
```

Naufal Yahya Kurnianto 13519141
IF4020 Kriptografi

Full Vigenere Cypher

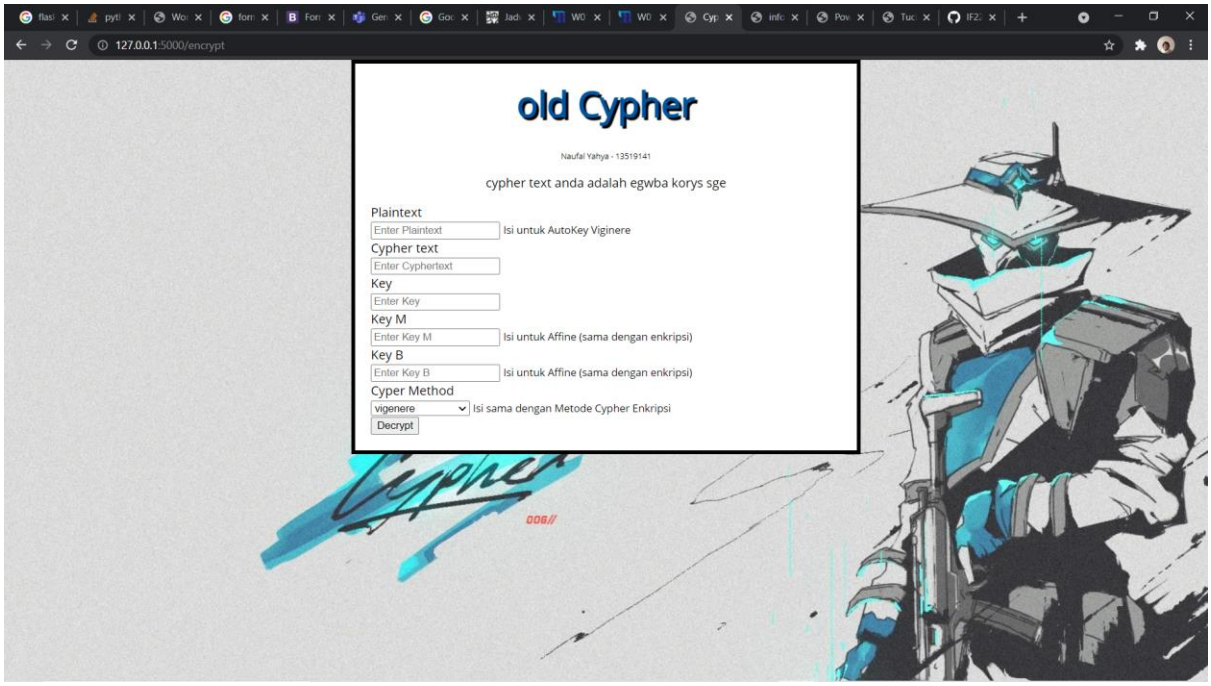


Sebelum dekripsi dan setelah dekripsi.

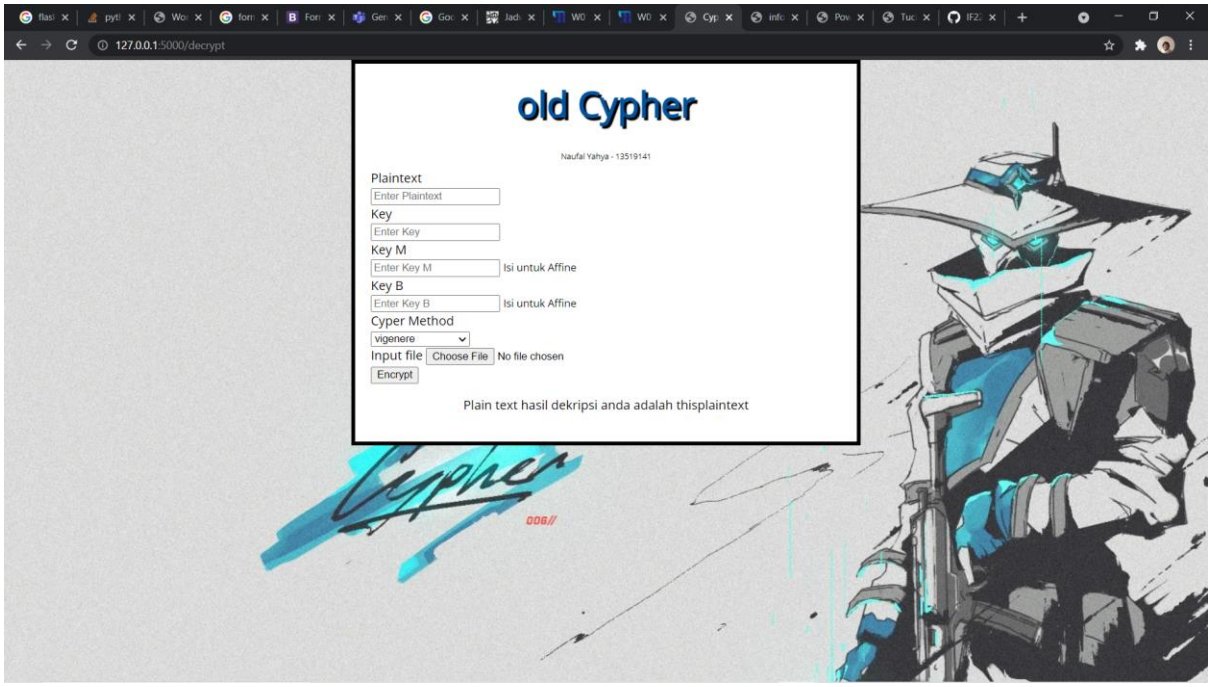


Menggunakan File

Naufal Yahya Kurnianto 13519141
IF4020 Kriptografi

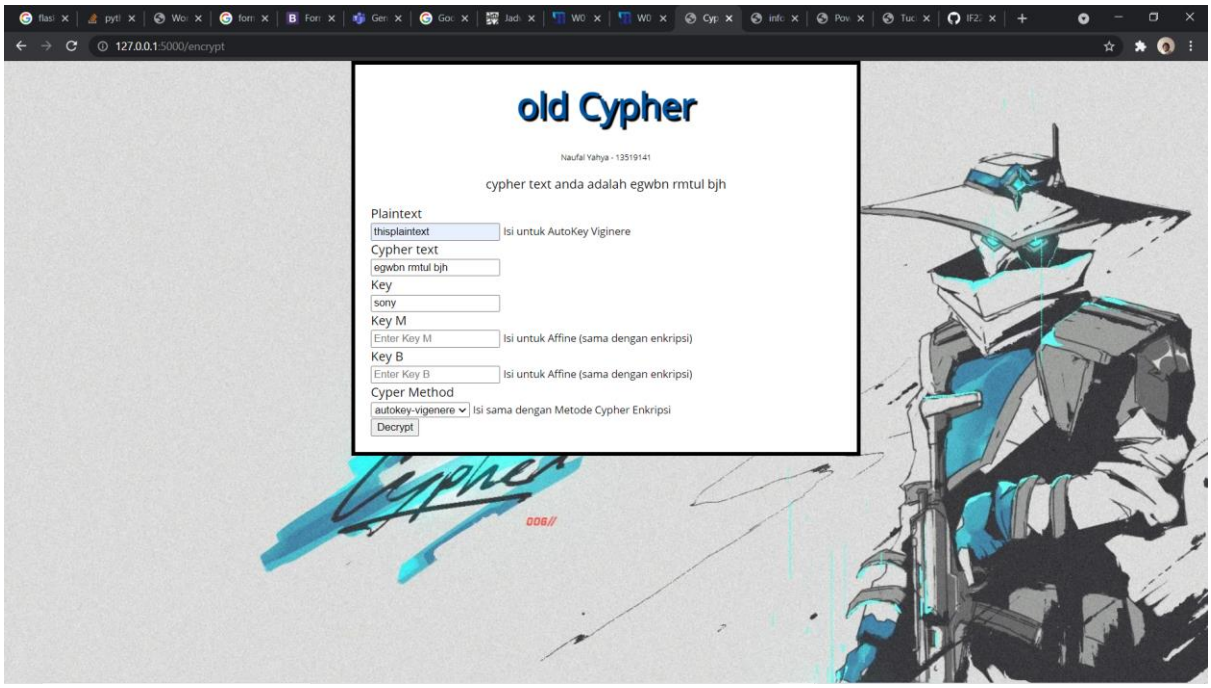


Sebelum dekripsi dan setelah dekripsi

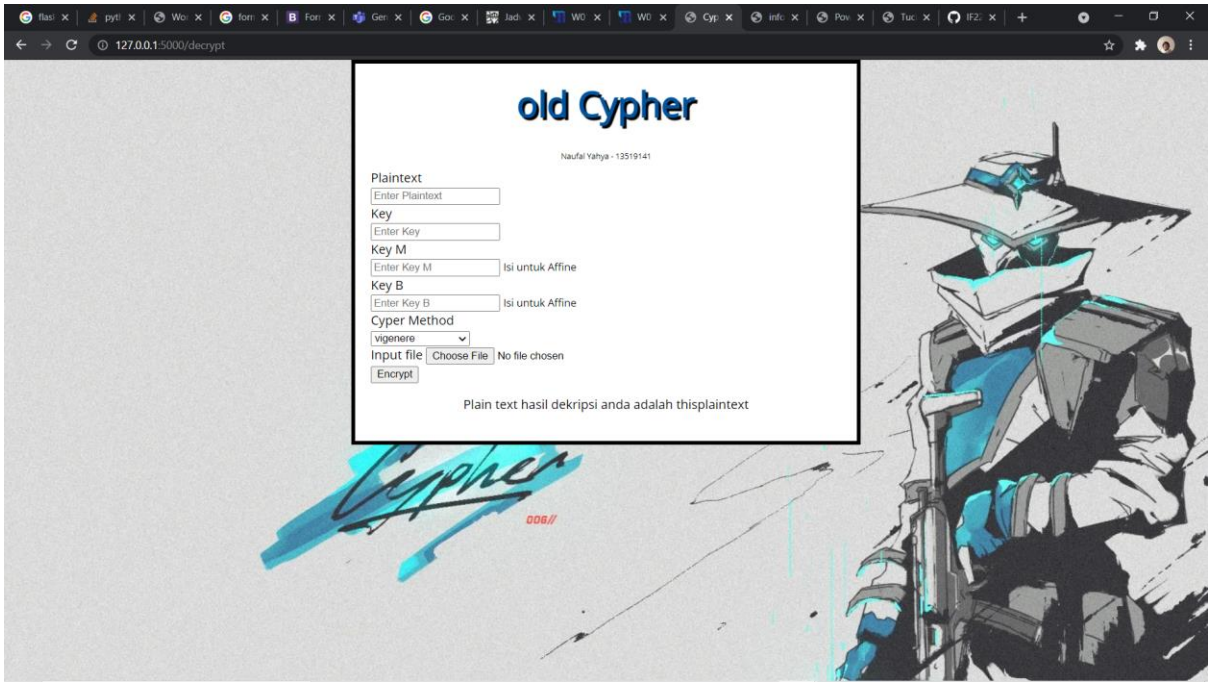


Naufal Yahya Kurnianto 13519141
IF4020 Kriptografi

AutoKey Viginere



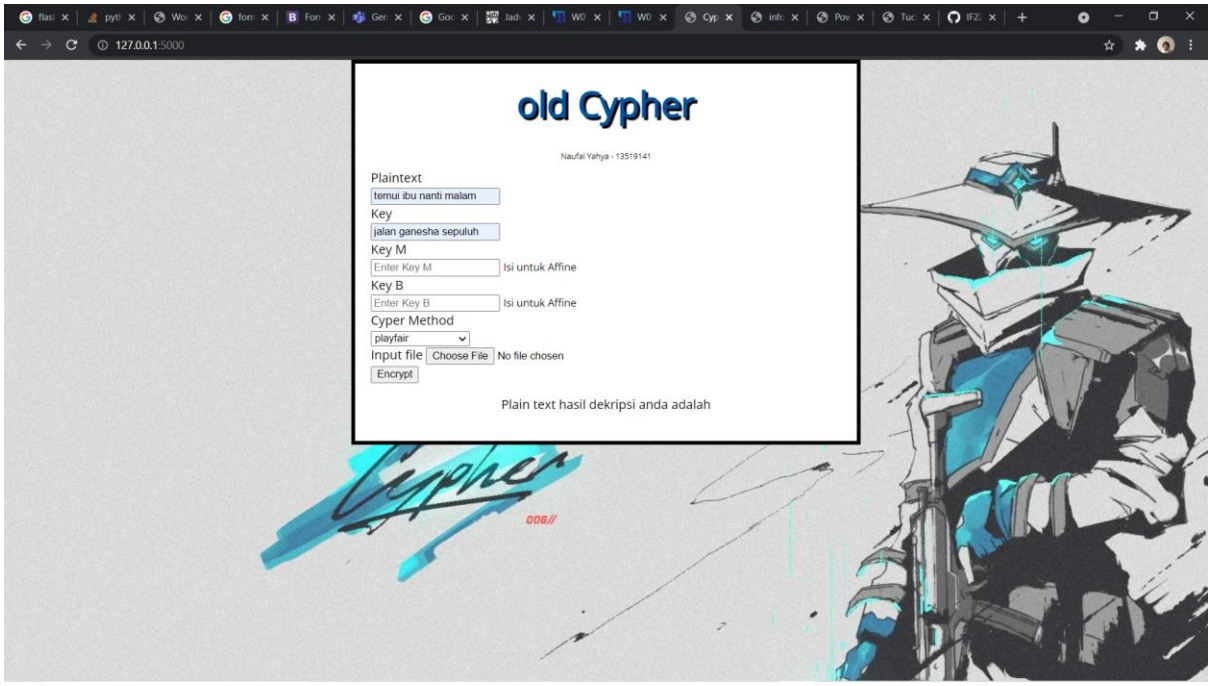
Sebelum dekripsi dan setelah dekripsi



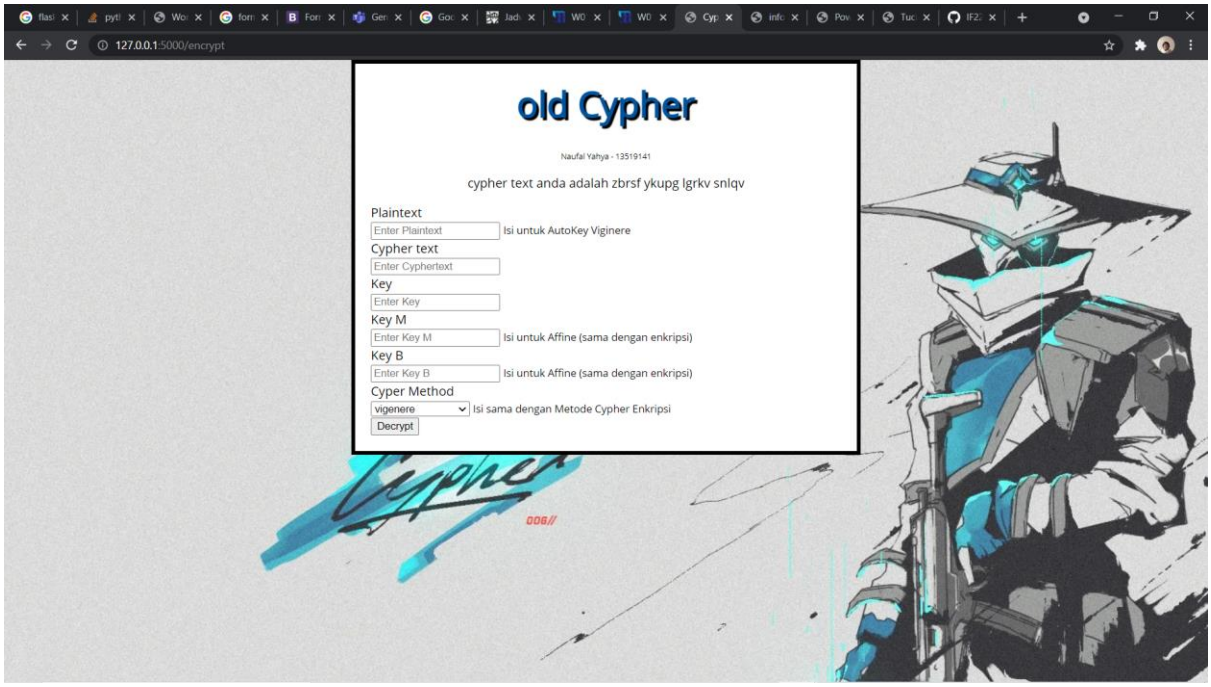
Playfair

Input playfair adalah temui ibu nanti malam, menggunakan key mentah jalan ganesha sepuluh

Naufal Yahya Kurnianto 13519141
IF4020 Kriptografi

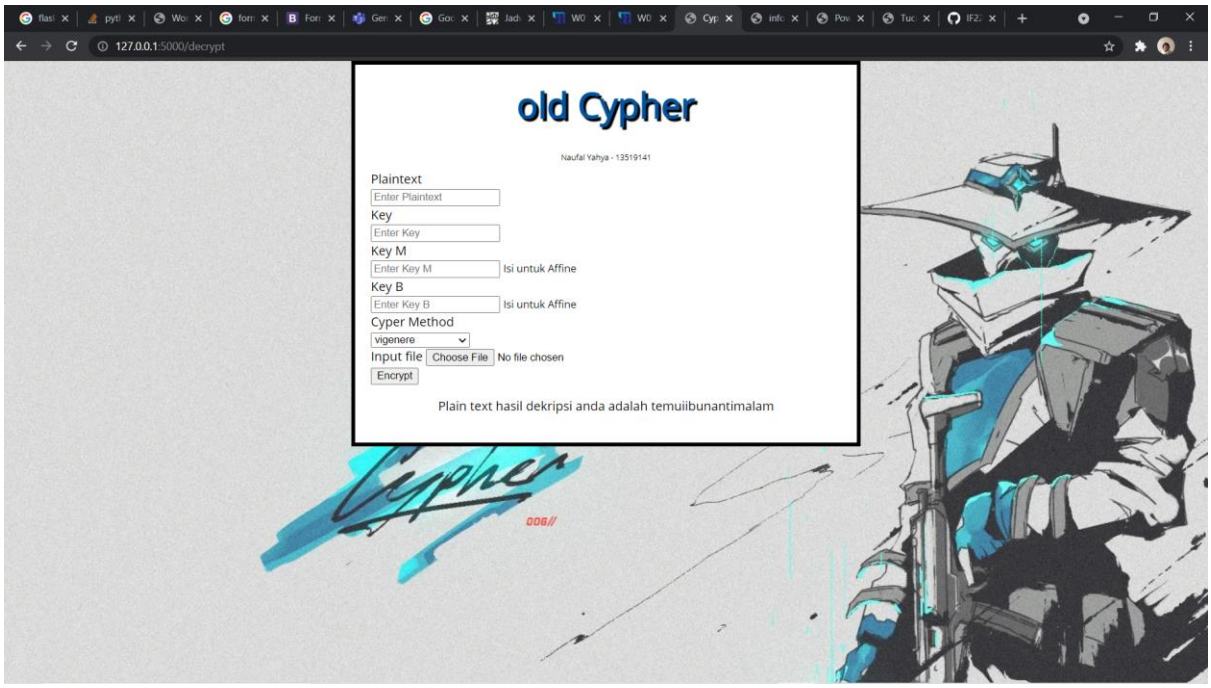


Sebelum dan Setelah Enkripsi



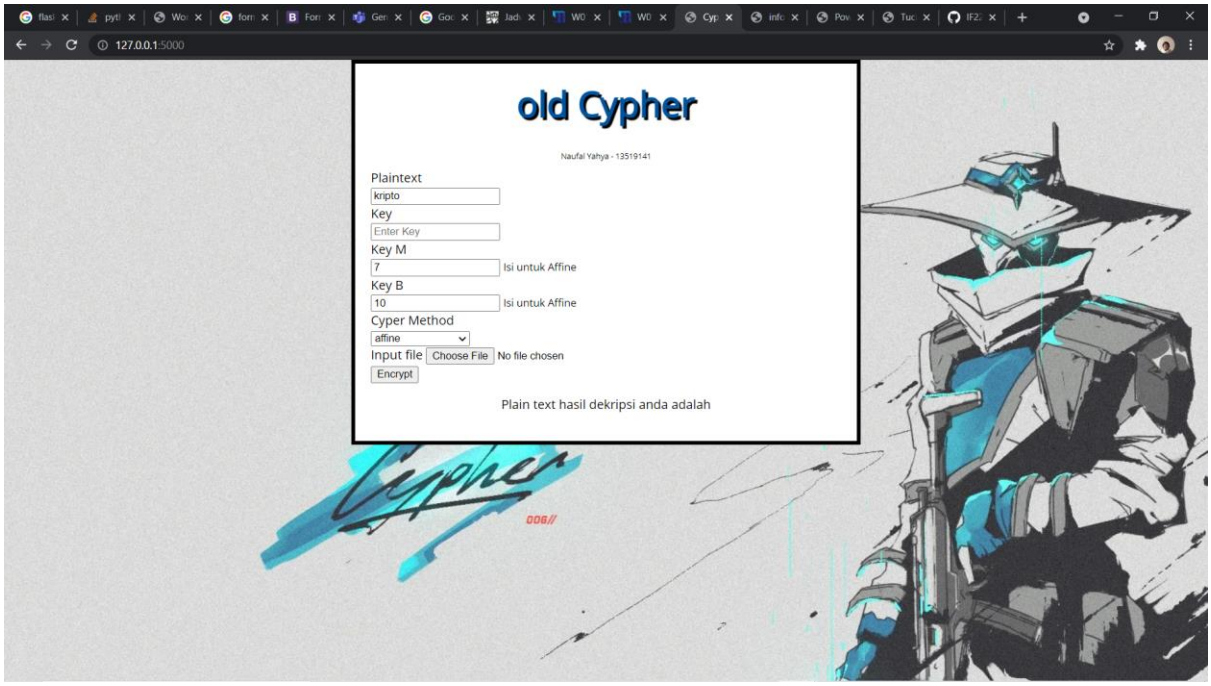
Naufal Yahya Kurnianto 13519141
IF4020 Kriptografi

Setelah Dekripsi

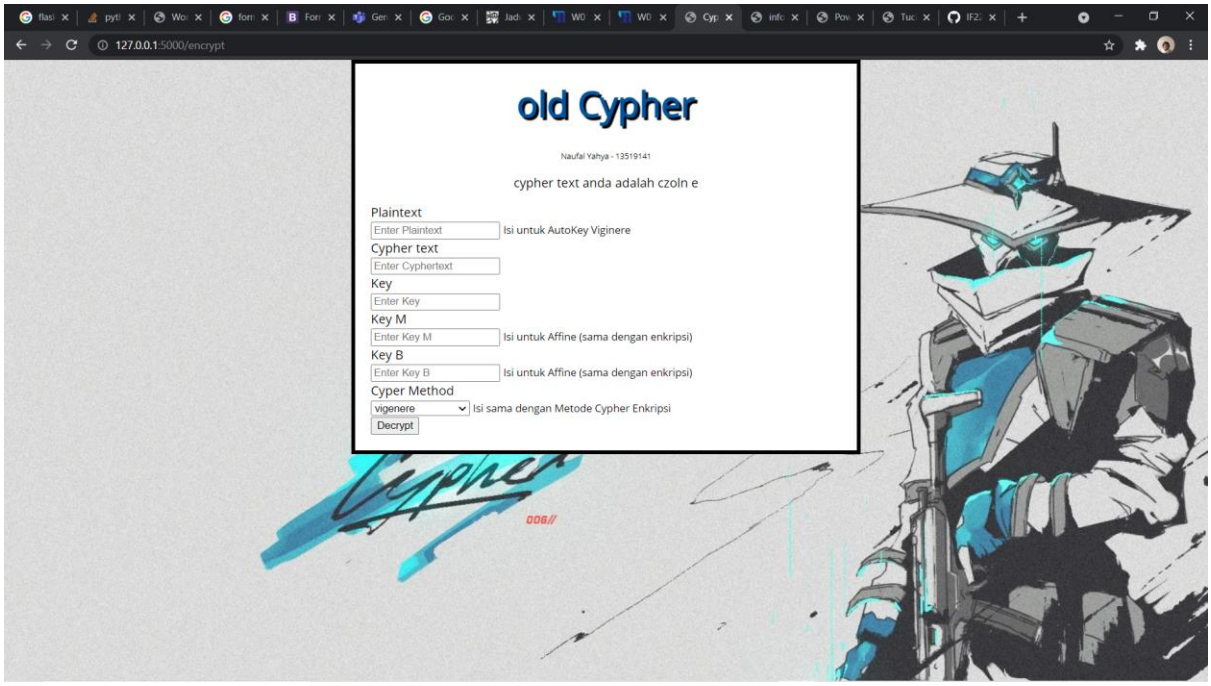


Affine

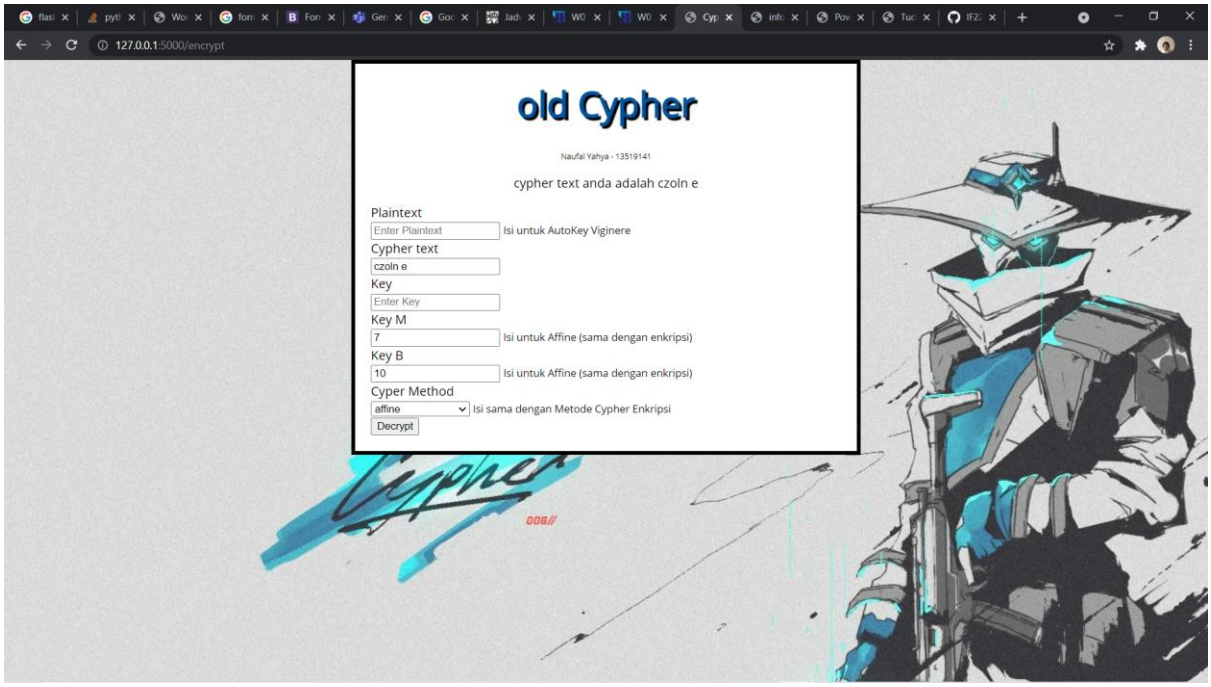
Sebelum dan Setelah Enkripsi



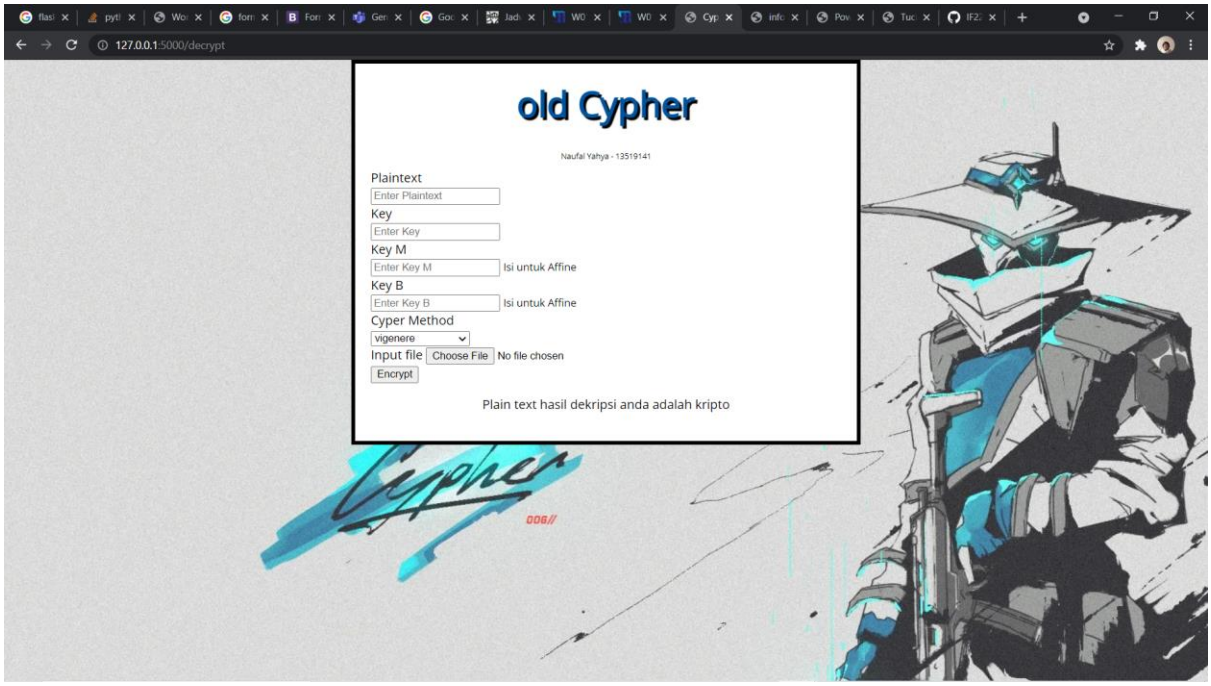
Naufal Yahya Kurnianto 13519141
IF4020 Kriptografi



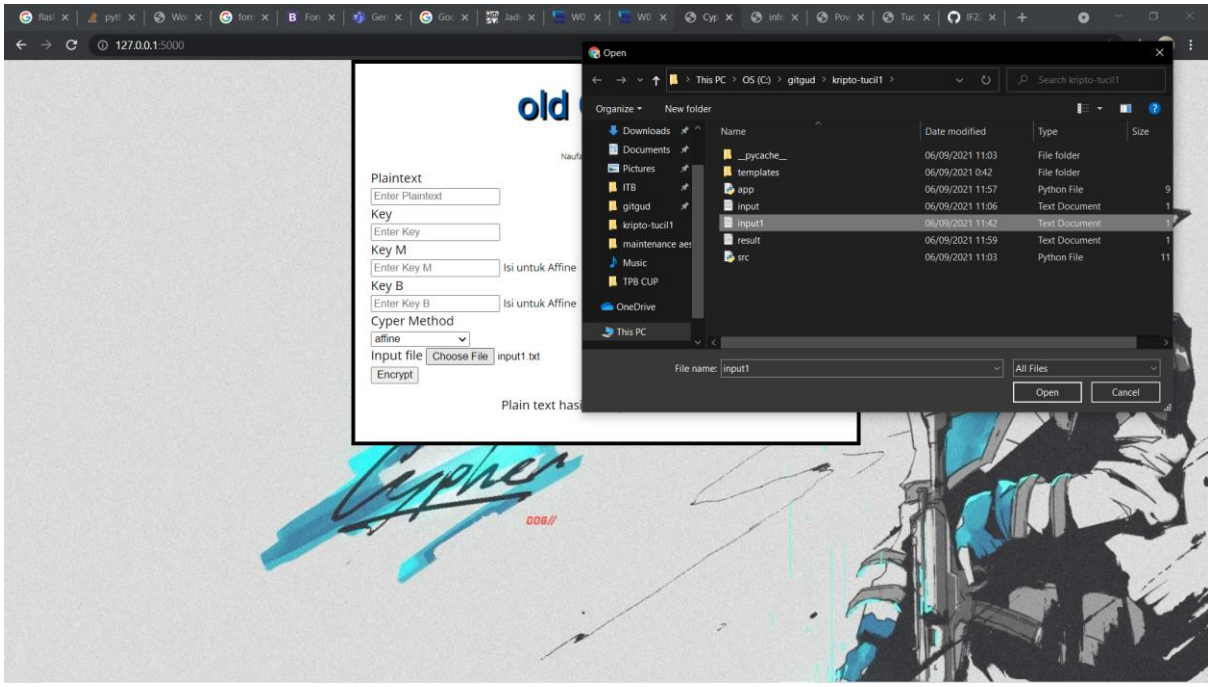
Sebelum dan Setelah Dekripsi

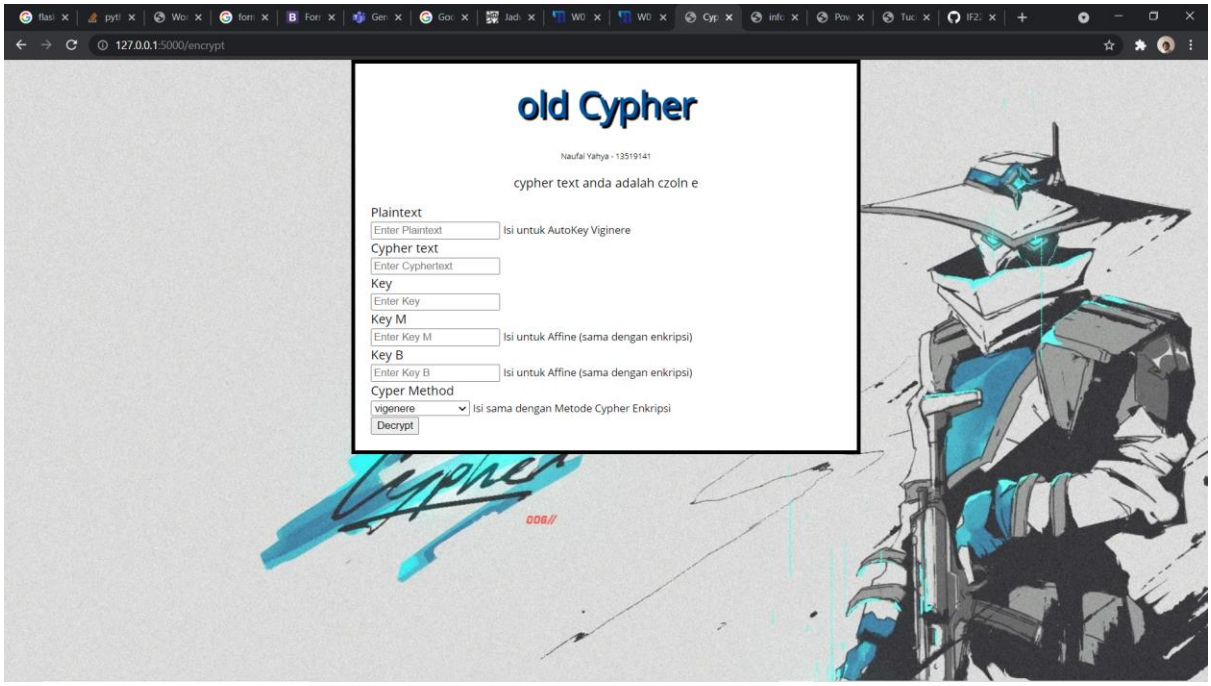


Naufal Yahya Kurnianto 13519141
IF4020 Kriptografi

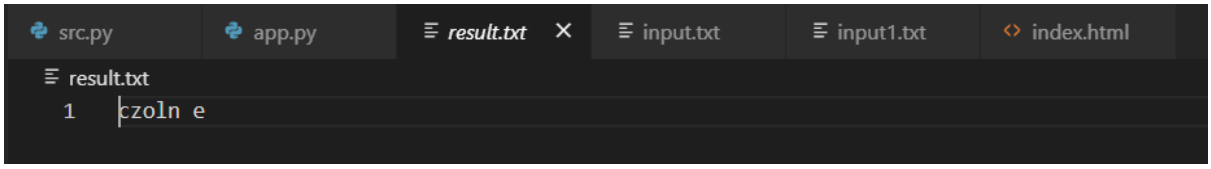


Menggunakan File





Hasil di result.txt



4. Tabel Penyelesaian

No	Spek	Berhasil	Kurang Berhasil	Keterangan
1.	Vigenere	v		
2.	Full Vigenere	v		
3.	Auto-key Vigenere	v		
4.	Extended Vigenere		v	Tidak selesai
5.	Playfair	v		
6.	Affine	v		
7.	Bonus		v	Tidak mengerjakan

5. Link Github

https://github.com/ayahyaaa/Kripto-Tucil1