Tugas 1 IF4020 Kriptografi

Semester 2 Tahun 2021/2022

Oleh:

13517079 - Fadhil Muhammad Rafi’

13518087 - Radhinansyah Hemsa Ghaida

Institut Teknologi Bandung

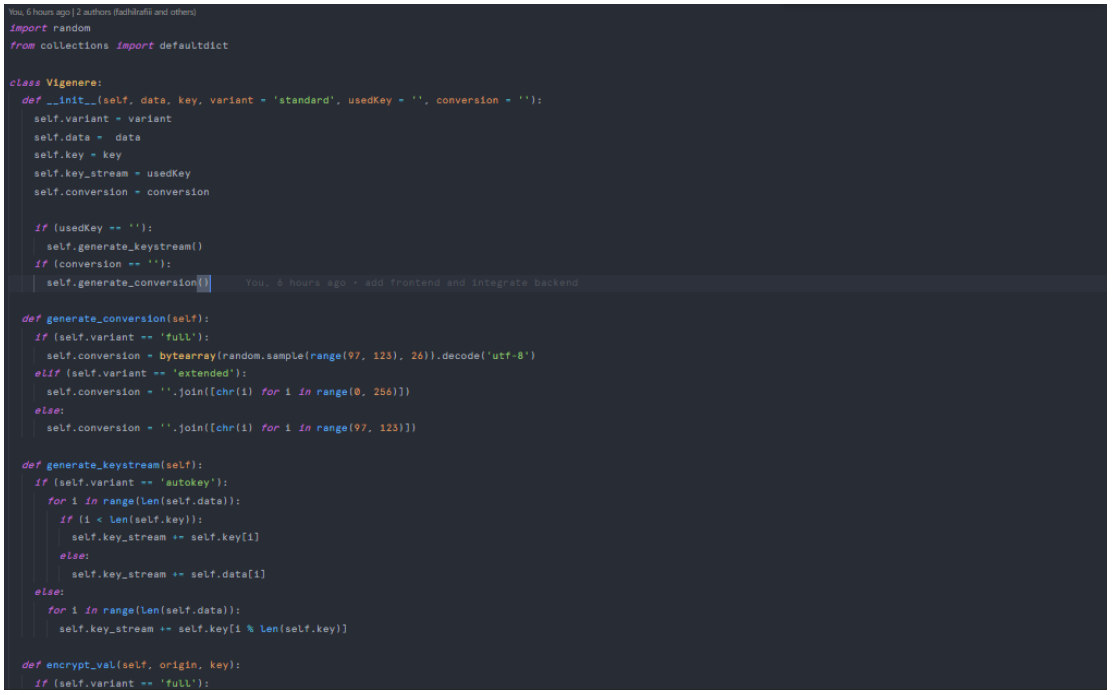
Teknik Informatika

2021

A. Source Code

1. Vigenere Cipher (seluruh varian)

   Algoritma Vigenere diimplementasikan pada *class Vigenere*. Kelas ini menerima beberapa argumen, antara lain, *data* yang merupakan teks yang ingin dienkripsi atau didekripsi, *key* yang merupakan kunci untuk menerapkan enkripsi/dekripsi, *variant* yang merupakan jenis dari Vigenere yang akan digunakan, *usedKey* yang digunakan untuk mendekripsi kembali cipher teks pada varian Autokey Vigenere (*karena keystream* dibangkitkan sesuai plain teks), dan terdapat *conversion* untuk mendekripsi kembali pada varian Full Vigenere karena urutan tabel konversi yang dibangkitkan tiap inisialisasi kelas akan berbeda.



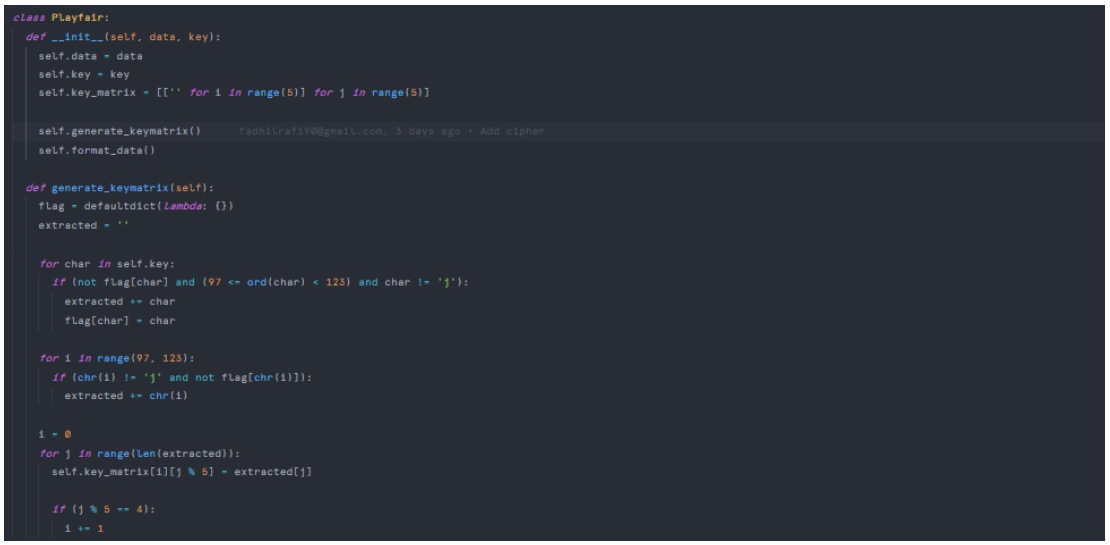
```
You, 6 hours ago | 2 authors (fadhilrafiii and others)
import random
from collections import defaultdict

class Vigenere:
  def __init__(self, data, key, variant = 'standard', usedKey = '', conversion = ''):
    self.variant = variant
    self.data =  data
    self.key = key
    self.key_stream = usedKey
    self.conversion = conversion

    if (usedKey == ''):
      self.generate_keystream()
    if (conversion == ''):
      self.generate_conversion()      You, 6 hours ago • add frontend and integrate backend

  def generate_conversion(self):
    if (self.variant == 'full'):
      self.conversion = bytearray(random.sample(range(97, 123), 26)).decode('utf-8')
    elif (self.variant == 'extended'):
      self.conversion = ''.join([chr(i) for i in range(0, 256)])
    else:
      self.conversion = ''.join([chr(i) for i in range(97, 123)])

  def generate_keystream(self):
    if (self.variant == 'autokey'):
      for i in range(len(self.data)):
        if (i < len(self.key)):
          self.key_stream += self.key[i]
        else:
          self.key_stream += self.data[i]
    else:
      for i in range(len(self.data)):
        self.key_stream += self.key[i % len(self.key)]

  def encrypt_val(self, origin, key):
    if (self.variant == 'full'):
```

```
    fadhilrafiii, 5 days ago • Add full and autokey vigenere
  def encrypt_val(self, origin, key):
    if (self.variant == 'full'):
      return self.conversion[(self.conversion.index(origin) + self.conversion.index(key)) % 26]
    if (self.variant == 'extended'):
      return self.conversion[(ord(origin) + ord(key)) % 256]

    return self.conversion[(ord(origin) + ord(key) - 194) % 26]

  def decrypt_val(self, value, key):
    if (self.variant == 'full'):
      return self.conversion[(self.conversion.index(value) - self.conversion.index(key)) % 26]
    if (self.variant == 'extended'):
      return self.conversion[(ord(value) - ord(key)) % 256]

    return self.conversion[(ord(value) - ord(key)) % 26]

  def encrypt(self):
    encrypted = ''
    for i in range(len(self.data)):
      encrypted += self.encrypt_val(self.data[i], self.key_stream[i])

    self.data = encrypted
    return self.data

  def decrypt(self):
    decrypted = ''
    for i in range(len(self.data)):
      decrypted += self.decrypt_val(self.data[i], self.key_stream[i])

    self.data = decrypted
    return self.data
```

Algoritma ini memiliki beberapa *method* untuk melakukan enkripsi dan dekripsi. Terdapat *generate_conversion* untuk membangkitkan tabel konversi karakter, *generate_keystream* untuk membangkitkan keystream dari key yang dimasukkan pengguna, *encrypt_val* dan *decrypt_val* yang berturut-turut untuk mengenkripsi dan mendekripsi karakter dari suatu string, serta *encrypt* dan *decrypt* yang berturut-turut untuk mengenkripsi dan mendekripsi keseluruhan data.

2. Playfair Cipher

Algoritma Vigenere diimplementasikan pada *class Vigenere*. Kelas ini menerima beberapa argumen, antara lain, *data* yang merupakan teks yang ingin dienkripsi atau didekripsi, *key* yang merupakan kunci untuk menerapkan enkripsi/dekripsi.



```
class Playfair:
  def __init__(self, data, key):
    self.data = data
    self.key = key
    self.key_matrix = [['' for i in range(5)] for j in range(5)]

    self.generate_keymatrix()      fadhilrafi90@gmail.com, 3 days ago • Add cipher
    self.format_data()

  def generate_keymatrix(self):
    flag = defaultdict(lambda: {})
    extracted = ''

    for char in self.key:
      if (not flag[char] and (97 <= ord(char) < 123) and char != 'j'):
        extracted += char
        flag[char] = char

    for i in range(97, 123):
      if (chr(i) != 'j' and not flag[chr(i)]):
        extracted += chr(i)

    i = 0
    for j in range(len(extracted)):
      self.key_matrix[i][j % 5] = extracted[j]

      if (j % 5 == 4):
        i += 1
```

```
def format_data(self):
  formatted = ''

  data = ''
  for i in range(len(self.data)):
    if (self.data[i] == 'j'):
      data += 'i'
    else:
      data += self.data[i]

  # Bigram musn't have same char
  i = 0
  while (i < len(data)):
    formatted += data[i]

    if (i == len(data) - 1):
      formatted += 'x'
    elif (data[i] == data[i + 1]):
      formatted += 'x'
      formatted += ' '
      i -= 1
    else:
      formatted += data[i+1]

      if (i != len(data) - 2):
        formatted += ' '

    i += 2

  self.data = formatted
```

```
def find_pos(self, val):
  i = 0
  for j in range(25):
    if (self.key_matrix[i][j % 5] == val):
      return i, j % 5

    if (j % 5 == 4):
      i += 1

def encrypt(self):
  arr_bigram = self.data.split(' ')
  arr_encrypted = []
  for bigram in arr_bigram:
    i1, j1 = self.find_pos(bigram[0])
    i2, j2 = self.find_pos(bigram[1])

    if (i1 == i2):
      arr_encrypted.append(self.key_matrix[i1][(j1 + 1) % 5] + self.key_matrix[i2][(j2 + 1) % 5])
    elif (j1 == j2):
      arr_encrypted.append(self.key_matrix[(i1 + 1) % 5][j1] + self.key_matrix[(i2 + 1) % 5][j2])
    else:
      arr_encrypted.append(self.key_matrix[i1][j2] + self.key_matrix[i2][j1])

  self.data = ''.join(arr_encrypted)
  return self.data
```

```
def decrypt(self):
  arr_bigram =  arr_bigram = self.data.split(' ')
  arr_decrypted = []
  for bigram in arr_bigram:
    i1, j1 = self.find_pos(bigram[0])
    i2, j2 = self.find_pos(bigram[1])

    if (i1 == i2):
      arr_decrypted.append(self.key_matrix[i1][(j1 - 1) % 5] + self.key_matrix[i2][(j2 - 1) % 5])
    elif (j1 == j2):
      arr_decrypted.append(self.key_matrix[(i1 - 1) % 5][j1] + self.key_matrix[(i2 - 1) % 5][j2])
    else:
      arr_decrypted.append(self.key_matrix[i1][j2] + self.key_matrix[i2][j1])

  decrypted = ''
  for i in range(len(arr_decrypted)):
    if (arr_decrypted[i][0] != 'x'):
      decrypted += arr_decrypted[i][0]
    if (arr_decrypted[i][1] != 'x'):
      decrypted += arr_decrypted[i][1]

  self.data = decrypted
  return self.data
```

Pada kelas ini, terdapat beberapa *method* tambahan dibandingkan kelas Vigenere, antara lain *format data*, untuk memformat teks yang akan di enkripsi/dekripsi, *generate_keymatrix* untuk membangkitkan matriks untuk mengkonversi karakter, dan *find_pos* untuk mendapatkan posisi karakter pada matriks.

3. Affine Cipher

   Kelas ini menerima dua argumen, yaitu *data* yang merupakan teks yang akan dienkripsi/dekripsi, serta *m* dan *b* yang merupakan kunci untuk mengenkripsi/mendekripsi teks. Untuk *method* pada kelas ini kurang lebih sama seperti kelas Vigenere, hanya terdapat satu tambahan, yaitu *set_inverse_m* untuk mendapatkan invers modulo dari kunci *m*.

```
fadhilrafi90@gmail.com, 3 days ago | 1 author (fadhilrafi90@gmail.com)
class Affine:
  def __init__(self, data, m, b):
    self.data = data
    self.m = m
    self.b = b
    self.inverse_m = -1

    self.set_inverse_m()

  def set_inverse_m(self):
    for x in range(1, 26):      fadhilrafi90@gmail.com, 3 days ago • Add cipher
      if (((self.m % 26) * (x % 26)) % 26 == 1):
        self.inverse_m = x

        return x

    return -1

  def encrypt_val(self, val):
    return chr((self.m * ((ord(val) - 97) % 26) + self.b) % 26 + 97)

  def decrypt_val(self, val):
    return chr((self.inverse_m * ((ord(val) - 97)  - self.b)) % 26 + 97)

  def encrypt(self):
    encrypted = ''
    for char in self.data:
      encrypted += self.encrypt_val(char)

    self.data = encrypted

    return self.data
```

```
def decrypt(self):
  decrypted = ''
  for char in self.data:
    decrypted += self.decrypt_val(char)

  self.data = decrypted

  return self.data
```

4. Hill Cipher

   Algoritma Hill Cipher diimplementasikan pada *class Hill.* Kelas ini menerima beberapa argumen, antara lain, *data* yang merupakan teks yang ingin dienkripsi atau didekripsi, *m* yang merupakan dimensi *key,* serta *key* yang merupakan kunci berbentuk matriks dengan dimensi *m* untuk menerapkan enkripsi/dekripsi.

```
# Hanya untuk m = 3
class Hill:
  def __init__(self, data, key, m):
    self.data = data
    self.length_data = len(data)
    self.key = key
    self.m = m
    self.inverse_key = [[0 for i in range(self.m)] for j in range(self.m)]

    self.set_inverse_key()      radhinghaida03@gmail.com, 3 days ago • add affine and hill cipher

  def get_inverse_mod(self, val):
    for x in range(1, 26):
      if (((val % 26) * (x % 26)) % 26 == 1):
        return x

    return -1

  def transpose(self, matrix):
    return [[row[i] for row in matrix] for i in range(len(matrix[0]))]

  def get_matrix_minor(self, matrix, i, j):
    return [row[:j] + row[j + 1 :] for row in (matrix[:i] + matrix[i + 1 :])]

  def determinant(self, matrix):
    if len(matrix) == 2:
      return (matrix[0][0] * matrix[1][1] - matrix[0][1] * matrix[1][0])

    determinant = 0
    for c in range(len(matrix)):
      determinant += ((-1.0) ** c) * matrix[0][c] * self.determinant(self.get_matrix_minor(matrix, 0, c))

    return determinant
```

```
def get_matrix_inverse(self, matrix):
  determinant = 1 / self.get_inverse_mod(self.determinant(matrix))

  # For 2x2 Matrix
  if len(matrix) == 2:
    return [[matrix[1][1] / determinant % 26, -1 * matrix[0][1] / determinant % 26] ,
    [-1 * matrix[1][0] / determinant % 26, matrix[0][0] / determinant % 26]]

  cofactors = []
  for r in range(len(matrix)):
    cofactorRow = []

    for c in range(len(matrix)):
      minor = self.get_matrix_minor(matrix,r,c)
      cofactorRow.append(((-1) ** (r + c)) * self.determinant(minor))

    cofactors.append(cofactorRow)

  cofactors = self.transpose(cofactors)
  for r in range(len(cofactors)):
    for c in range(len(cofactors)):
      cofactors[r][c] = cofactors[r][c] / determinant % 26

  return cofactors

def set_inverse_key(self):
  self.inverse_key = self.get_matrix_inverse(self.key)
```

Pada kelas ini terdapat *mehod get_matrix_inverse* untuk mendapatkan invers matriks dengan modulo 26, serta terdapat juga *method-method* yang dibutuhkan ntuk operasi *get_matrix_inverse* seperti *transpose, determinant, get_matrix_minor,* dan *get_inverse_mod* untuk mendapatkan inverse modulo dari suatu bilangan bulat.

```
def format_data(self):
  remain = self.m - len(self.data) % self.m
  added_data = self.data
  if (remain != 0):
    for i in range(remain):
      added_data += 'z'

  formatted = ''
  for i in range(len(added_data)):
    formatted += added_data[i]

    if (i % self.m == self.m - 1 and i != len(added_data) - 1):
      formatted += ' '

  return formatted

def crypt_mgram(self, mgram, key):
  encrypted_mgram = [0 for i in range(self.m)]
  i = 0
  j = 0
  while(j < self.m ** 2 and i < self.m):
    encrypted_mgram[i] += int(key[i][j]) * (ord(mgram[j]) - 97)
    j += 1

    if (j % self.m == 0):
      encrypted_mgram[i] = chr(encrypted_mgram[i] % 26 + 97)
      i += 1
      j = 0

  return ''.join(encrypted_mgram)
```

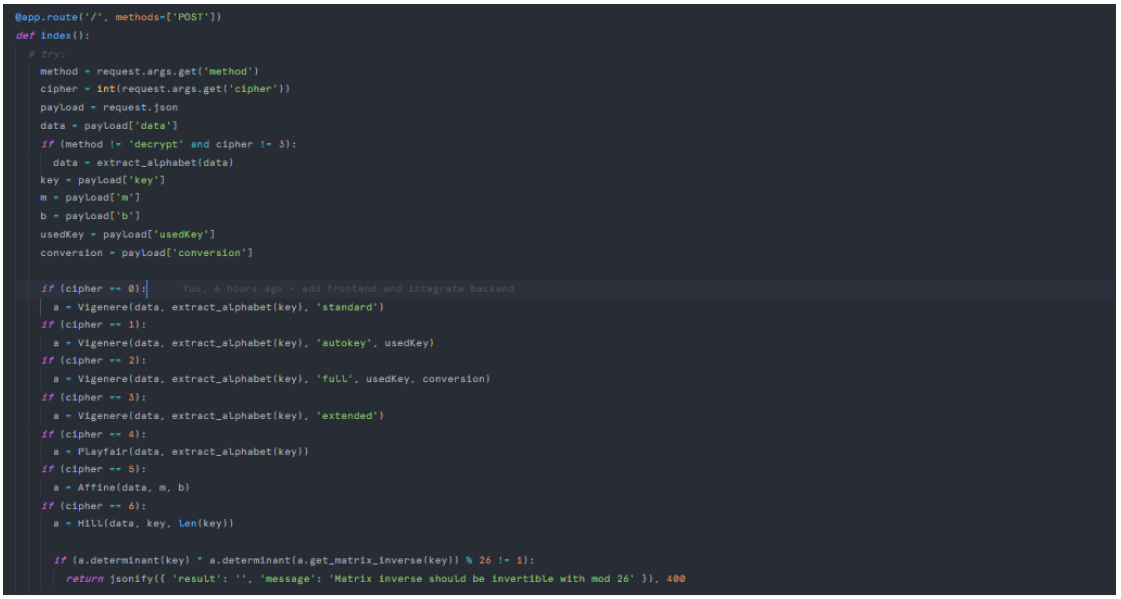
```
def encrypt(self):
  formatted_data = self.format_data().split(' ')

  encrypted = ''
  for i in range(len(formatted_data)):
    encrypted += self.crypt_mgram(formatted_data[i], self.key)

  self.data = encrypted[0 : self.length_data]
  return self.data

def decrypt(self):
  formatted_data = self.format_data().split(' ')

  decrypted = ''
  for i in range(len(formatted_data)):
    decrypted += self.crypt_mgram(formatted_data[i], self.inverse_key)

  self.data = decrypted[0 : self.length_data]
  return self.data
```

Selain itu, terdapat *format_data* untuk memisahkan teks setiap *m* karakter dan menambahkan ‘x’ pada akhir data sejumlah kurangnya panjang data untuk habis dibagi dengan *m, crypt_mgram* untuk mengonversi setiap *m* karakter pada data, serta *encrypt* dan *decrypt* untuk mengenkripsi dan mendekripsi teks utuhnya.

5. Backend API

Implementasi *backend* pada tugas ini dilakukan dengan *library Flask* pada *Python.* Pada implementasinya, dibuat satu buat *endpoint* API dengan method POST, yang menerima query *method* yang berisi metode yang dipilih (enkripsi/dekripsi) dan *cipher* yang merupakan pilihan algoritma cipher yang akan digunakan.

```
You, 39 minutes ago | 1 author (You)
from flask import Flask, jsonify, request
from flask_cors import CORS, cross_origin

# Import utils
from utils.helper import *
from utils.string import extract_alphabet, parse_n_char

# Import Algorithm
from cipher.vigenere import Vigenere
from cipher.playfair import Playfair
from cipher.affine import Affine
from cipher.hill import Hill

# INIT APP
app = Flask(__name__)

# SET CORS
cors = CORS(app)
app.config['CORS_HEADERS'] = 'Content-Type'
```



```
@app.route('/', methods=['POST'])
def index():
  # try:
    method = request.args.get('method')
    cipher = int(request.args.get('cipher'))
    payload = request.json
    data = payload['data']
    if (method != 'decrypt' and cipher != 3):
      data = extract_alphabet(data)
    key = payload['key']
    m = payload['m']
    b = payload['b']
    usedKey = payload['usedKey']
    conversion = payload['conversion']

    if (cipher == 0):      You, 6 hours ago • add frontend and integrate backend
      a = Vigenere(data, extract_alphabet(key), 'standard')
    if (cipher == 1):
      a = Vigenere(data, extract_alphabet(key), 'autokey', usedKey)
    if (cipher == 2):
      a = Vigenere(data, extract_alphabet(key), 'full', usedKey, conversion)
    if (cipher == 3):
      a = Vigenere(data, extract_alphabet(key), 'extended')
    if (cipher == 4):
      a = Playfair(data, extract_alphabet(key))
    if (cipher == 5):
      a = Affine(data, m, b)
    if (cipher == 6):
      a = Hill(data, key, len(key))

      if (a.determinant(key) * a.determinant(a.get_matrix_inverse(key)) % 26 != 1):
        return jsonify({ 'result': '', 'message': 'Matrix inverse should be invertible with mod 26' }), 400
```

API ini juga menerima beberapa *request body*, yaitu *data* yang merupakan teks yang akan dikonversi, *key* yang merupakan kunci yang diketikkan pengguna untuk enkripsi/dekripsi teks, *m* dan *b* yang merupakan kunci ketika memilih algoritma Affine, *usedKey* untuk mengembalikan *keystream* yang sebelumnya digunakan untuk mengenkripsi sehingga hasil dekripsi bisa konsisten, dan *conversion* untuk mengembalikan tabel konversi yang telah digunakan pada enkripsi sebelumnya pada algoritma Full Vigenere, hal ini karena tabel konversi pada Full Vigenere selalu acak di setiap inisiasi kelasnya.

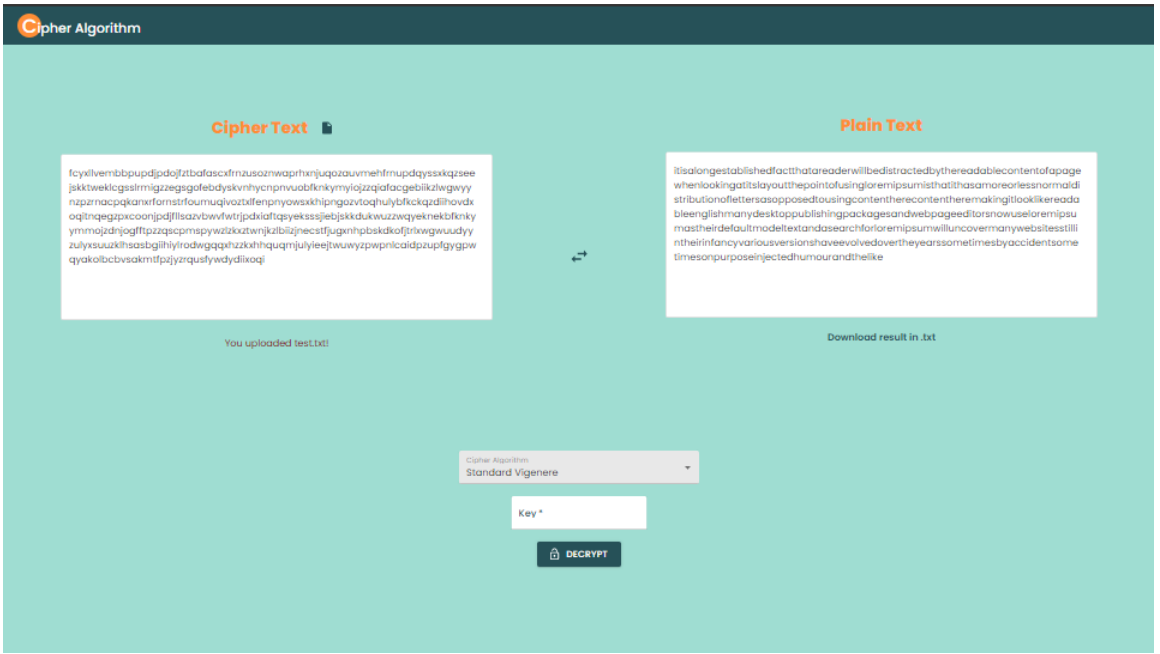
```
    if (method == 'decrypt'):
      result = a.decrypt()
    else:
      result = a.encrypt()

    if (cipher == 1):
      return jsonify({ 'result': result, 'usedKey': a.key_stream })
    if (cipher == 2):
      return jsonify({ 'result': result, 'conversion': a.conversion })

    return jsonify({ 'result': result })
  # except Exception as err:
  #   return jsonify({ 'error': str(repr(err)) }), 400

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

6. *GUI*

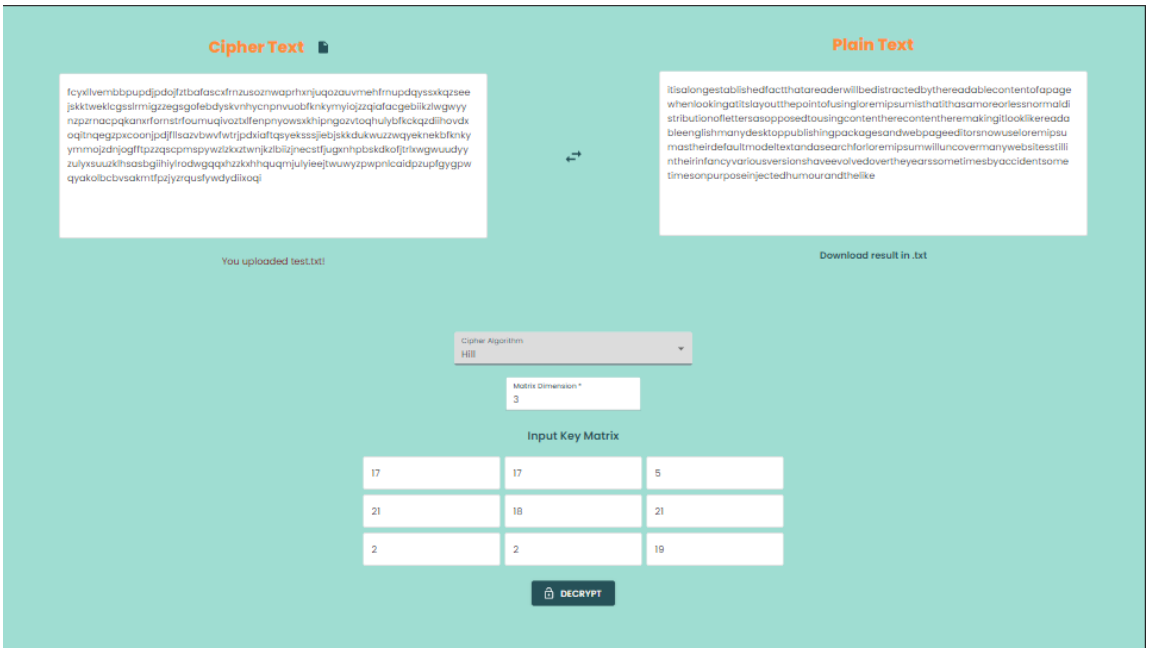
   GUI pada aplikasi ini berupa web yang dikembangkan dengan React JS. Untuk penjelasan lebih lanjut mengenai antarmuka dapat dilihat pada bagian selanjutnya.

B. Tampilan Antarmuka

Antarmuka yang digunakan dalam aplikasi ini adalah antarmuka web yang dikembangkan dengan *framework* React JS. Antarmuka terdiri dari kotak teks di sebelah kiri yang bisa diketik sendiri dan juga bisa membangkitkan teks dari file yang diunggah melalui tombol dengan ikon file di samping judul kotak teks di sebelah kiri. Kotak teks di sebelah kanan merupakan hasil enkripsi atau dekripsi teks di sebelah kiri. Kotak teks di sebelah kanan tidak bisa diketikkan tulisan (*read-only)*. Di bawah kotak teks sebelah kanan terdapat tombol unduh file untuk mengunduh teks hasil enkripsi atau dekripsi dalam file txt.



Di bagian bawah terdapat *dropdown* untuk memilih jenis algoritma yang ingin digunakan. Di bawahnya terdapat *field* untuk memasukkan key yang akan digunakan untuk mengenkripsi atau mendekripsi teks. *Field* key ini berbeda-beda bergantung pada algoritma yang dipilih. Jika algoritma yang dipilih adalah Vigenere atau Playfair, maka *field* key akan meminta input string, meminta input *m* dan *b* jika algoritma yang dipilih adalah Affine dan meminta matriks apabila yang dipilih adalah Hill.



C. Contoh Plainteks

Beberapa contoh plain teks yang digunakan, antara lain:

Plain teks	Key	Hasil enkripsi	Hasil dekripsi kembali
aku sayang kamu	aku juga *(Autokey Vigenere)*	auobueaamuayo	akusayangkamu
aku cinta rasulullah	amin *(Autokey Vigenere)*	pejmvardnrdizwshpz	akucintarasulullah
attack at dawn	17 17 5 21 18 21 2 2 19 *(Hill)*	cnjgmmaprxtf	attackatdawn
temui ibu nanti malam	jalan ganesha sepuluh *(Playfair)*	zbrsfykupglgrkvsnlqv	temuiibunantimalam
kripto	m = 7, b = 10	czolne	kripto

File Image JPG (terlalu panjang untuk ditampilkan)	aku suka kamu *(Vigenere Extended)*	(terlalu panjang untuk ditampilkan)	(terlalu panjang untuk ditampilkan)
It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using Lorem Ipsum is that it has a more-or-less normal distribution of letters, as opposed to using 'Content here, content here', making it look like readable English. Many desktop publishing packages and web page editors now use Lorem Ipsum as their default model text, and a search for 'lorem ipsum' will uncover many web sites still in their infancy. Various versions have evolved over the years, sometimes by accident, sometimes on purpose (injected humour and the like).	lemon *(Vigenere Standard)*	txugnwszurdxmpytwt sqqeohgsefoepepseh mxzophuggceohrofkh upvqoqlfxspzrfsaesro clkqkuprxcbvmzune mfgylcaigelqdbtrfcsf wubtwsdsztteiztwfvn emfvndeycepsdzrdwz cexexrvdxdwofxucaz jxsgeidgndsbdbdiphb fwubtnszhryxtsepgab gprfvrciyoxtrswgwsa yytoqfrlhmpypizuytw tanycpsfvxadcffxwfs mzuclgwotpwmbqhin dnriqrvesdgazaggrws dsztteizlwfvrtvpsslyx hzzhqzgpbfoaoeesnc gttbcpafrxmbghxauzy frocipvyoajaqpftxqgf emxzvyxtsvcmztnygk jncmaifgidgvzrevngi qjbwzqrbgidhupcqoe dwaaremysfmcmqpth qbgdsysgtqqgbytgfcz wqwauiohrolgabfvm bqelqzvvi	itisalongestablishedfa ctthatareaderwillbedi stractedbythereadable contentofapagewhenl ookingatitslayoutthep ointofusingloremipsu misthatithasamoreorl essnormaldistribution oflettersasopposedtou singcontenthereconte ntheremakingitlookli kereadableenglishma nydesktoppublishing packagesandwebpage editorsnowuseloremi psumastheirdefaultm odeltextandasearchfo rloremipsumwillunco vermanywebsitesstilli ntheirinfancyvariousv ersionshaveevolvedo vertheyearssometime sbyaccidentsometime sonpurposeinjectedhu mourandthelike

D. Prosedur Menjalankan Program

Untuk menjalankan program, ikuti langkah-langkah berikut:

1. Setup backend
   a. Masuk ke direktori backend: `cd backend`
   b. Install seluruh dependencies yang dibutuhkan: `pip install -r requirement.txt`
   c. Jalankan aplikasi: `python app.py` (Note: command '`python`' bergantung pada device masing-masing)
2. Setup frontend
   a. Masuk ke direktori frontend: `cd frontend`
   b. Install seluruh dependencies yang diperlukan: `npm install`
   c. Jalankan aplikasi: `npm start`

E. Pranala

Sumber kode program: https://github.com/fadhilrafiii/tugas-1-kripto

No.	Spek	Berhasil	Kurang Berhasil	Keterangan
1	Vigenere Standard Cipher	v		
2	Full Vigenere Cipher	v		
3	Autokey Vigenere Cipher	v		
4	Extended Vigenere Cipher	v		
5	Playfair Cipher	v		
6	Affine Cipher	v		
7	Hill Cipher	v		