Laporan Kriptosistem Tradisional

Tugas 1 IF4020 Kriptografi Semester II Tahun 2022/2023

Dibuat oleh:
Mochammad Fatchur Rochman (13519009)
M. Abdi Haryadi. H (13519156)

TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2023

Bagian A: Kriptografi

Tabel A.I Rangkuman Penyelesaian Tugas Bagian A

No.	**Spek**	**Berhasil**	**Kurang berhasil**
1	*Standard Vigènere Cipher*	V	
2	*Auto-Key Vigènere Cipher*	V	
3	*Extended Vigènere Cipher*	V	
4	*Affine Cipher*	V	
5	*Playfair Cipher*	V	
6	*Hill Cipher*	V	
7	*Enigma Cipher*	V	

Tabel I menunjukkan rangkuman penyelesaian tugas bagian A. Satu-satunya yang menerima masukan ASCII adalah *extended Vigènere cipher*. Untuk *hill cipher* implementasi dibatasi dengan matriks kunci yang dimasukkan diasumsikan pasti memiliki balikan/*inverse*. *Cipher* ini melakukan enkripsi per 3 huruf plaintext, dan memiliki batasan plaintext dan ciphertext yang akan di-enkripsi atau di-deskripsi harus dipastikan memiliki jumlah huruf sesuai kelipatan matriks kunci yang dipilih, jika matriks kunci 2x2 maka jumlah huruf *plaintext/ciphertext* harus kelipatan 2, jika matriks kunci 3x3 maka jumlah huruf *plaintext/ciphertext* harus kelipatan 3. Untuk *enigma cipher*, implementasi dibatasi dengan pengaturan cincin yang bernilai A saja. Namun, posisi awal, susunan huruf rotor, banyak rotor, dan susunan huruf reflektor dapat diatur. Aplikasi web yang dibuat diasumsikan pengguna memasukkan input dengan benar.

Subbab-subbab berikutnya meliputi contoh masukan dan keluaran, termasuk *plaintext* dan *ciphertext*, serta kode sumber dari masing-masing *cipher*. Kode sumber yang digunakan dapat diakses melalui tautan `https://github.com/AbdiHaryadi/classic-cryptology`.

Hasil

Bagian ini menunjukkan contoh hasil dari setiap *cipher* beserta rujukan kode sumbernya. Kode sumber dilampirkan pada sebagai subbab lain.

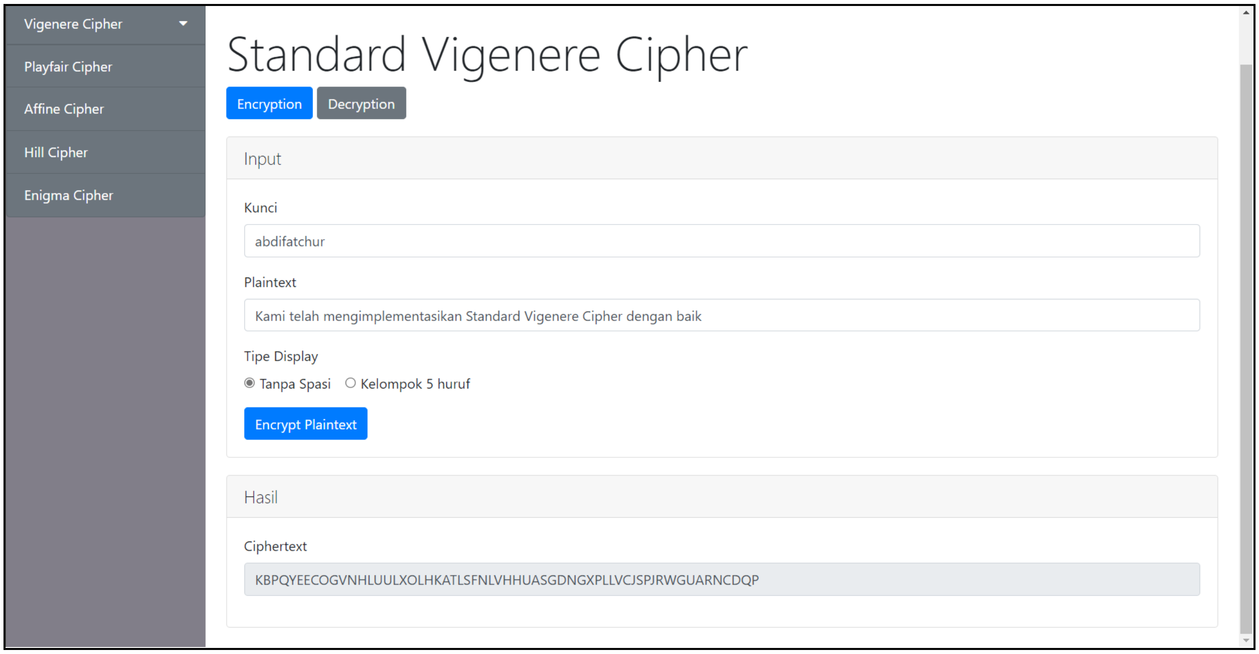
Standard Vigènere Cipher

Cipher ini menerima kunci dan *plaintext* alfabetik. Karakter yang bukan abjad akan diabaikan. Kode-kode sumber yang digunakan adalah `standard_vigenere_cipher.py` sebagai kode utama, `letters.py` untuk melakukan iterasi pada alfabet saja, serta `standard_vigenere_table_generator.py` sebagai pembangkit tabel untuk *cipher* ini.

Contoh masukan:
Kunci: `abdifatchur`
Plaintext:
`Kami telah mengimplementasikan Standard Vigenere Cipher dengan baik.`

Contoh keluaran enkripsi:
Ciphertext:
`KBPQYEECOGVNHLUULXOLHKATLSFNLVHHUASGDNGXPLLVCJSPJRWGUARNCDQP`

Tampilan keluaran enkripsi (pada aplikasi web):



Standard Vigenere Cipher

Input

Kunci
abdifatchur

Plaintext
Kami telah mengimplementasikan Standard Vigenere Cipher dengan baik

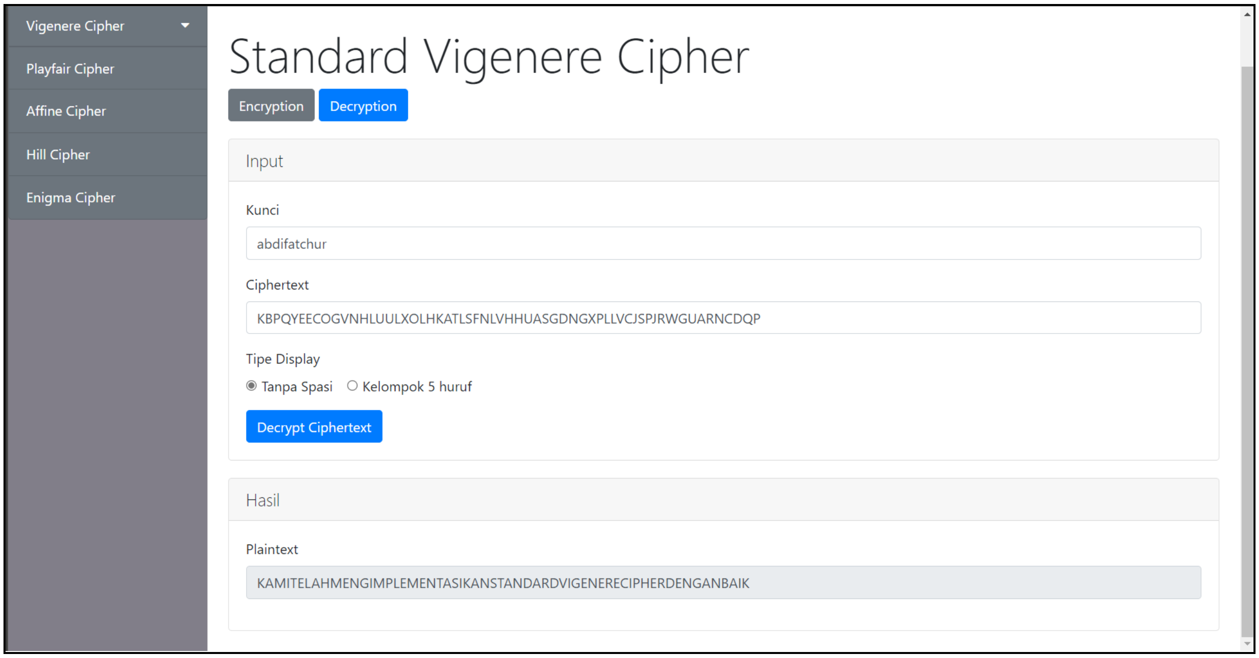
Tipe Display
● Tanpa Spasi ○ Kelompok 5 huruf

Hasil

Ciphertext
KBPQYEECOGVNHLUULXOLHKATLSFNLVHHUASGDNGXPLLVCJSPJRWGUARNCDQP

Contoh keluaran dekripsi semula:
Plaintext:
`KAMITELAHMENGIMPLEMENTASIKANSTANDARDVIGENERECIPHERDENGANBAIK`

Tampilan keluaran dekripsi semula (pada aplikasi web):



Standard Vigenere Cipher

Input

Kunci

abdifatchur

Ciphertext

KBPQYEECOGVNHLUULXOLHKATLSFNLVHHUASGDNGXPLLVCJSPJRWGUARNCDQP

Tipe Display

● Tanpa Spasi ○ Kelompok 5 huruf

Decrypt Ciphertext

Hasil

Plaintext

KAMITELAHMENGIMPLEMENTASIKANSTANDARDVIGENERECIPHERDENGANBAIK

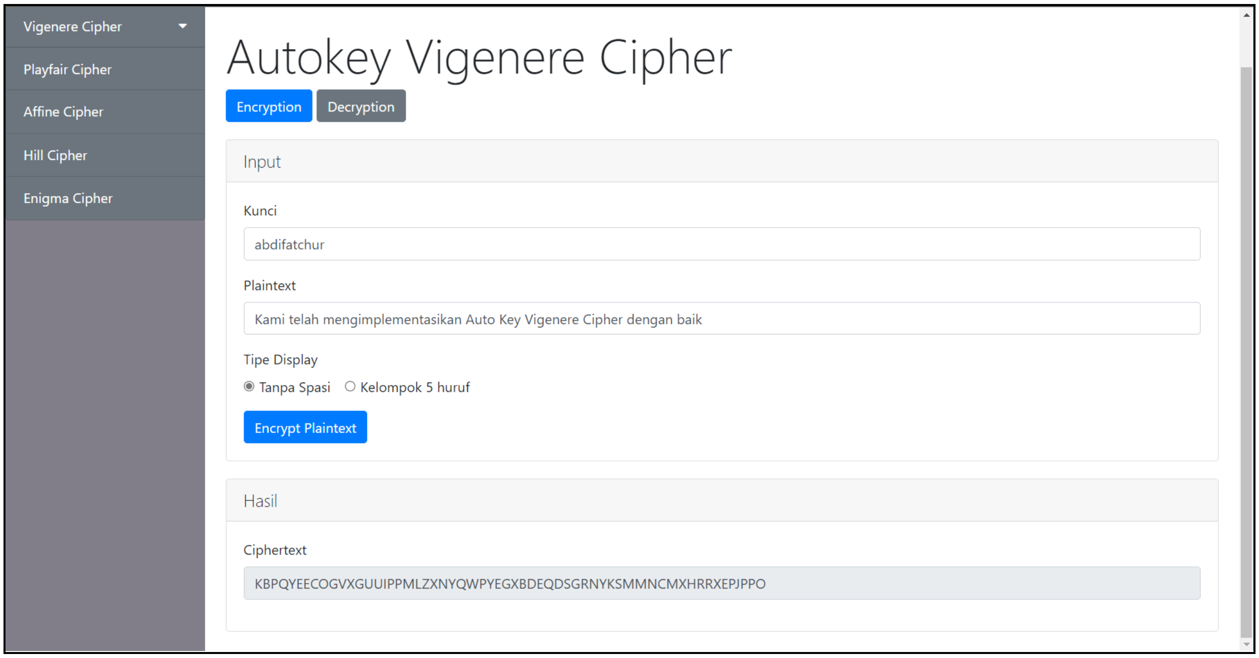
Auto-Key Vigènere Cipher

Cipher ini menerima kunci dan *plaintext* alfabetik. Karakter yang bukan abjad akan diabaikan. Kode sumber yang digunakan adalah `auto_key_vigenere_cipher.py` sebagai kode utama, `letters.py` untuk melakukan iterasi pada alfabet saja, serta `standard_vigenere_table_generator.py` sebagai pembangkit tabel untuk *cipher* ini..

Contoh masukan:
Kunci: `abdifatchur`
Plaintext:
`Kami telah mengimplementasikan Auto Key Vigenere Cipher dengan baik.`

Contoh keluaran enkripsi:
Ciphertext:
`KBPQYEECOGVXGUUIPPMLZXNYQWPYEGXBDEQDSGRNYKSMMNCMXHRRXEPJPPO`

Tampilan keluaran enkripsi (pada aplikasi web):



Autokey Vigenere Cipher

Input

Kunci: abdifatchur

Plaintext: Kami telah mengimplementasikan Auto Key Vigenere Cipher dengan baik

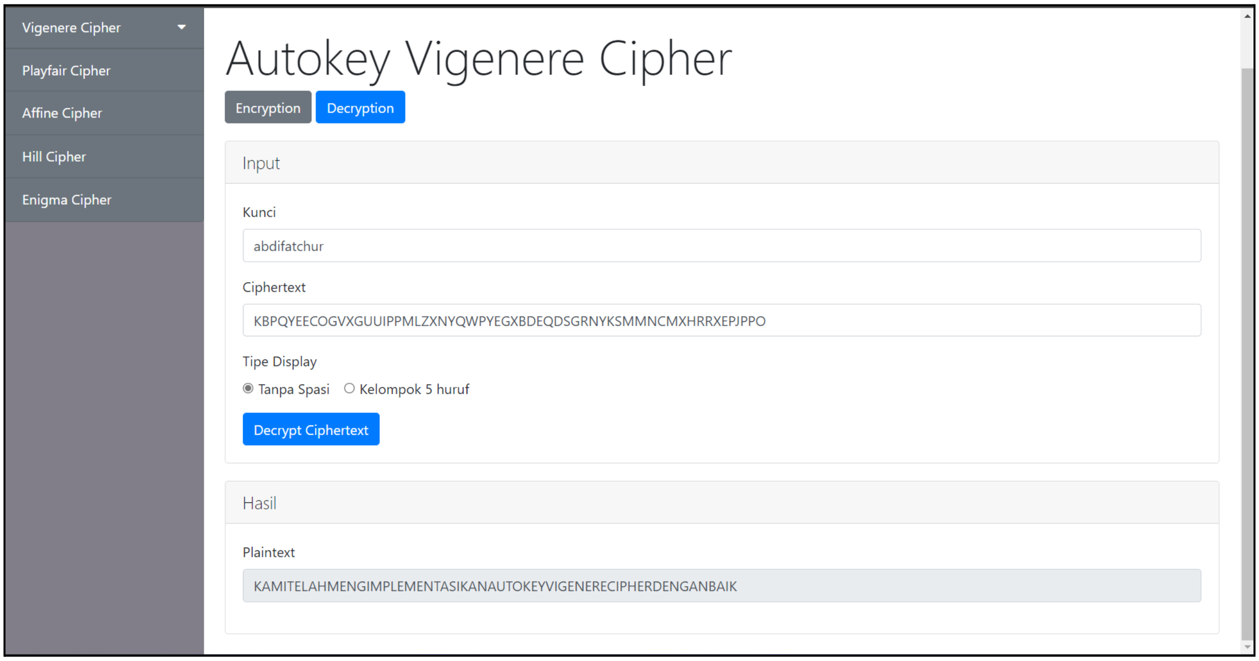
Tipe Display: ● Tanpa Spasi ○ Kelompok 5 huruf

Hasil

Ciphertext: KBPQYEECOGVXGUUIPPMLZXNYQWPYEGXBDEQDSGRNYKSMMNCMXHRRXEPJPPO

Contoh keluaran dekripsi semula:
Plaintext:
`KAMITELAHMENGIMPLEMENTASIKANAUTOKEYVIGENERECIPHERDENGANBAIK`

Tampilan keluaran dekripsi semula (pada aplikasi web):



Autokey Vigenere Cipher

Input

Kunci: abdifatchur

Ciphertext: KBPQYEECOGVXGUUIPPMLZXNYQWPYEGXBDEQDSGRNYKSMMNCMXHRRXEPJPPO

Tipe Display: ● Tanpa Spasi ○ Kelompok 5 huruf

Hasil

Plaintext: KAMITELAHMENGIMPLEMENTASIKANAUTOKEYVIGENERECIPHERDENGANBAIK

Extended Vigènere Cipher

Cipher ini menerima kunci dan *plaintext* yang terdiri dari *byte*. Kode sumber yang digunakan adalah `extended_vigenere_cipher.py`. Alasan kode `standard_vigenere_cipher.py` tidak digunakan ulang adalah algoritme tersebut tidak mendukung untuk tipe data *byte*.

Contoh masukan:
Kunci (ASCII): `abdifatchur`
Plaintext (ASCII):

```
Kami telah mengimplementasikan Extended Vigenere Cipher dengan baik.
```

Contoh keluaran enkripsi:
Ciphertext (heksadesimal):

```
ac c3 d1 d2 86 d5 d9 cf c9 dd 92 ce c7 d2 d0 cf ce e4 cf cd e2 d7 cf
d6 c5 dc cf cc d5 d1 88 ba ea d5 c7 d2 cd cb c5 94 b9 d1 dc d7 cf c7
d6 ce 86 a4 dd d3 d0 da e4 81 c6 c9 d7 cd c2 e2 83 ca d6 db cc 90
```

Contoh keluaran dekripsi semula:
Plaintext (ASCII):

```
Kami telah mengimplementasikan Extended Vigenere Cipher dengan baik.
```

Affine Cipher

Cipher ini menerima kunci dan *plaintext* alfabetik. Kode sumber yang digunakan adalah `affine_cipher.py`.

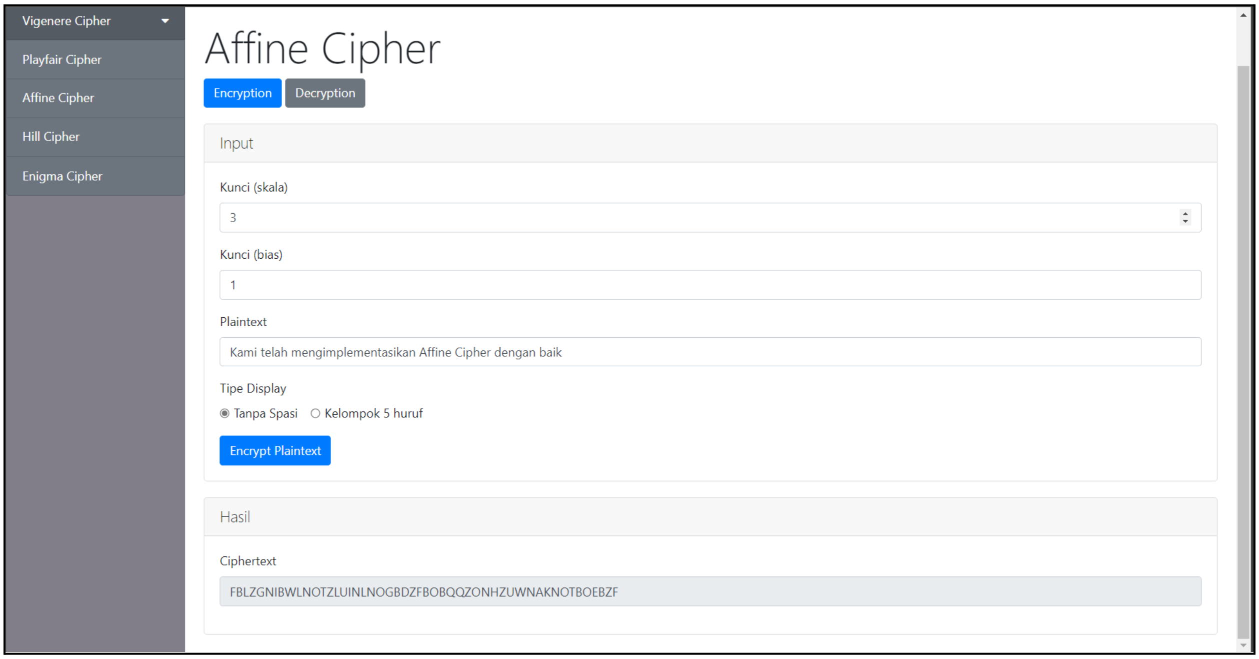
Contoh masukan:
Kunci: skala = 3; bias = 1
Plaintext:

```
Kami telah mengimplementasikan Affine Cipher dengan baik.
```

Contoh keluaran enkripsi:
Ciphertext:

```
FBLZGNIBWLNOTZLUINLNOGBDZFBOBQQZONHZUWNAKNOTBOEBZF
```

Tampilan keluaran enkripsi (pada aplikasi web):



Affine Cipher

Input

Kunci (skala): 3

Kunci (bias): 1

Plaintext: Kami telah mengimplementasikan Affine Cipher dengan baik

Tipe Display: ● Tanpa Spasi ○ Kelompok 5 huruf

Hasil

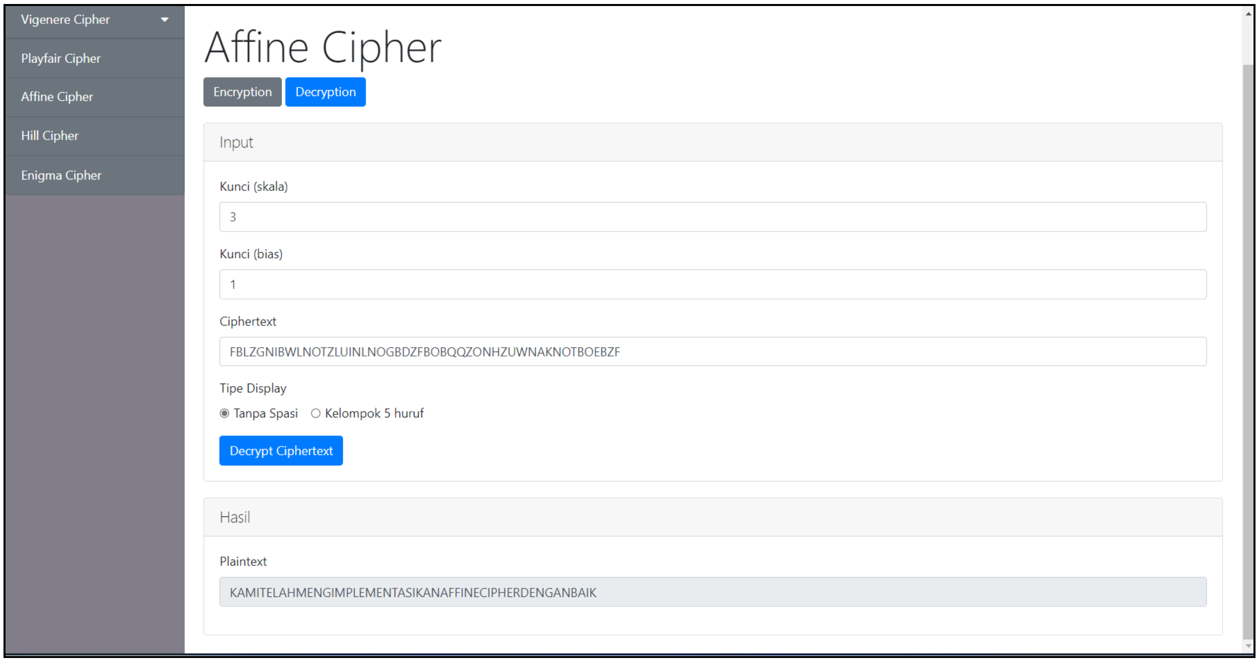
Ciphertext: FBLZGNIBWLNOTZLUINLNOGBDZFBOBQQZONHZUWNAKNOTBOEBZF

Contoh keluaran dekripsi semula:

Plaintext:

KAMITELAHMENGIMPLEMENTASIKANAFFINECIPHERDENGANBAIK

Tampilan keluaran dekripsi semula (pada aplikasi web):



Affine Cipher

Input

Kunci (skala): 3

Kunci (bias): 1

Ciphertext: FBLZGNIBWLNOTZLUINLNOGBDZFBOBQQZONHZUWNAKNOTBOEBZF

Tipe Display: ● Tanpa Spasi ○ Kelompok 5 huruf

Hasil

Plaintext: KAMITELAHMENGIMPLEMENTASIKANAFFINECIPHERDENGANBAIK

Playfair Cipher

Cipher ini menerima kunci dan *plaintext* alfabetik. Kode sumber yang digunakan adalah `playfair_cipher.py`. Huruf yang digunakan jika terdapat huruf yang berulang dalam bigram yang sama adalah X. Huruf yang sama berlaku untuk akhiran *plaintext* yang ganjil. Namun, jika huruf yang berulang adalah X, huruf yang digunakan adalah Q. Kunci akan disusun secara sekuensial dari kiri ke kanan, atas ke bawah pada tabel Playfair.

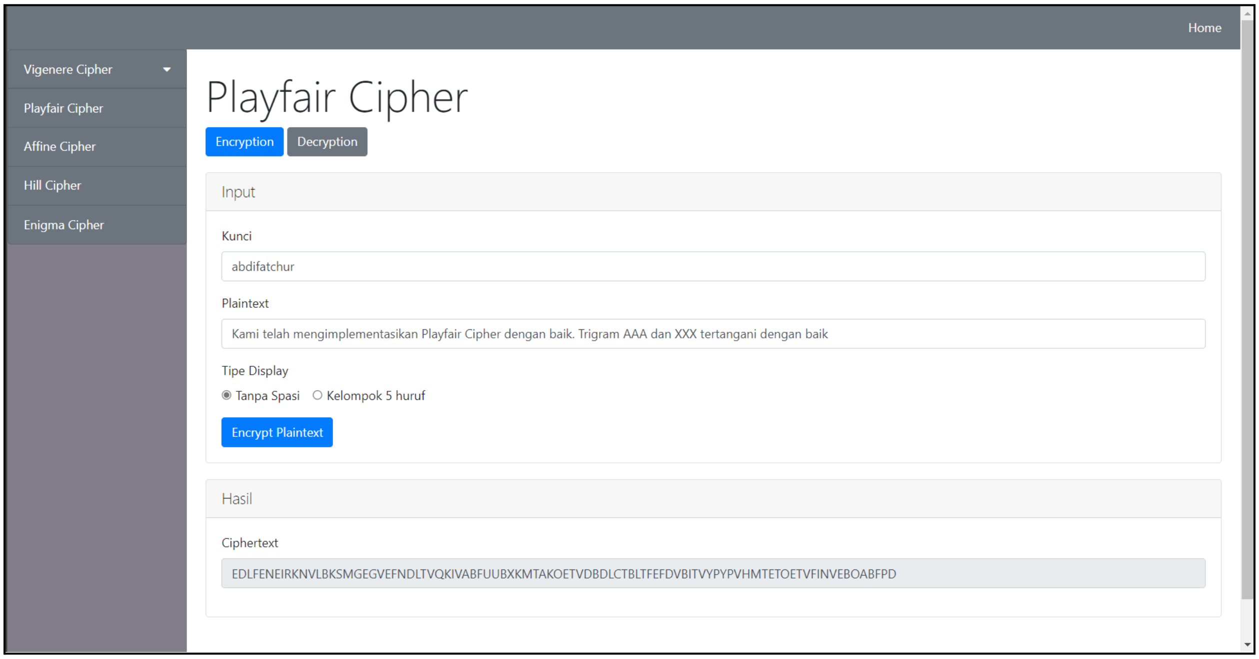
Contoh masukan:
Kunci: `abdifatchur`
Plaintext:

```
Kami telah mengimplementasikan Playfair Cipher dengan baik. Trigram AAA dan XXX tertangani dengan baik.
```

Contoh keluaran enkripsi:
Ciphertext:

```
EDLFENEIRKNVLBKSMGEGVEFNDLTVQKIVABFUUBXKMTAKOETVDBDLCTBLTFEFDVBITVYPYP
VHMTETOETVFINVEBOABFPD
```

Tampilan keluaran enkripsi (pada aplikasi web):



Playfair Cipher

Input

Kunci: abdifatchur

Plaintext: Kami telah mengimplementasikan Playfair Cipher dengan baik. Trigram AAA dan XXX tertangani dengan baik

Tipe Display: ● Tanpa Spasi ○ Kelompok 5 huruf

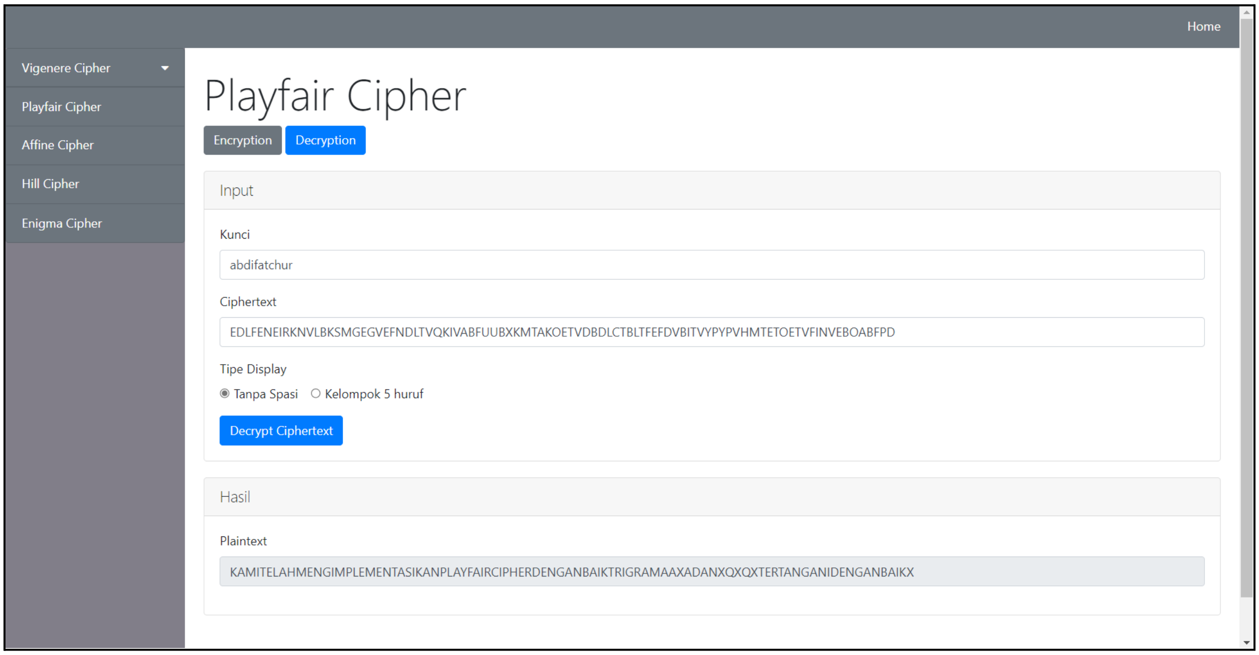
Hasil

Ciphertext: EDLFENEIRKNVLBKSMGEGVEFNDLTVQKIVABFUUBXKMTAKOETVDBDLCTBLTFEFDVBITVYPYPVHMTETOETVFINVEBOABFPD

Contoh keluaran dekripsi semula:
Plaintext:

```
KAMITELAHMENGIMPLEMENTASIKANPLAYFAIRCIPHERDENGANBAIKTRIGRAMAAXADANXQXQ
XTERTANGANIDENGANBAIKX
```

Tampilan keluaran dekripsi semula (pada aplikasi web):



Playfair Cipher

Input

Kunci

abdifatchur

Ciphertext

EDLFENEIRKNVLBKSMGEGVEFNDLTVQKIVABFUUBXKMTAKOETVDBDLCTBLTFEFDVBITVYPYPVHMTETOETVFINVEBOABFPD

Tipe Display

● Tanpa Spasi ○ Kelompok 5 huruf

Decrypt Ciphertext

Hasil

Plaintext

KAMITELAHMENGIMPLEMENTASIKANPLAYFAIRCIPHERDENGANBAIKTRIGRAMAAXADANXQXQXTERTANGANIDENGANBAIKX

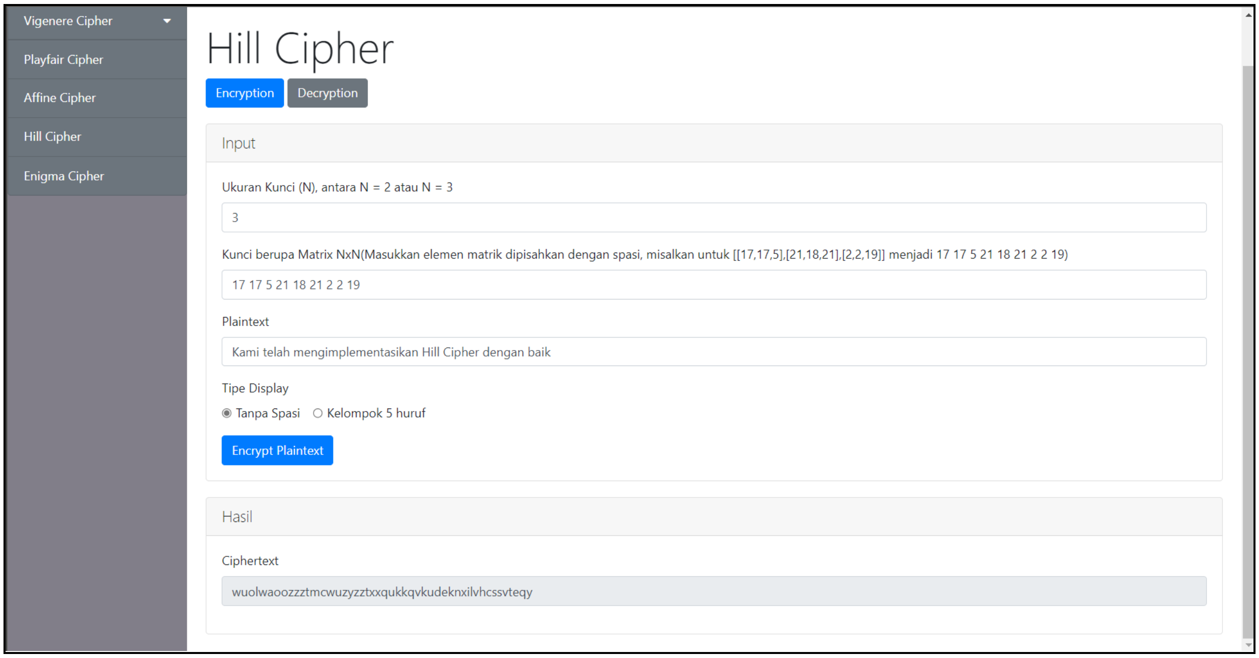
Hill Cipher

Cipher ini menerima kunci sebuah matriks 3x3 atau 2x2 dan *plaintext* alfabetik. Kode sumber yang digunakan adalah `hillCipher.py`.

Contoh masukan:
Kunci: 17 17 5 21 18 21 2 2 19, yang sama dengan [[17,17,5], [21,18,21],[2,2,19]]
Plaintext:
`Kami telah mengimplementasikan Hill Cipher dengan baik.`

Contoh keluaran enkripsi:
Ciphertext:
`wuolwaoozzztmcwuzyzztxxqukkqvkudeknxilvhcssvteqy`

Tampilan keluaran enkripsi (pada aplikasi web):



Hill Cipher

Input

Ukuran Kunci (N), antara N = 2 atau N = 3

3

Kunci berupa Matrix NxN(Masukkan elemen matrik dipisahkan dengan spasi, misalkan untuk [[17,17,5],[21,18,21],[2,2,19]] menjadi 17 17 5 21 18 21 2 2 19)

17 17 5 21 18 21 2 2 19

Plaintext

Kami telah mengimplementasikan Hill Cipher dengan baik

Tipe Display

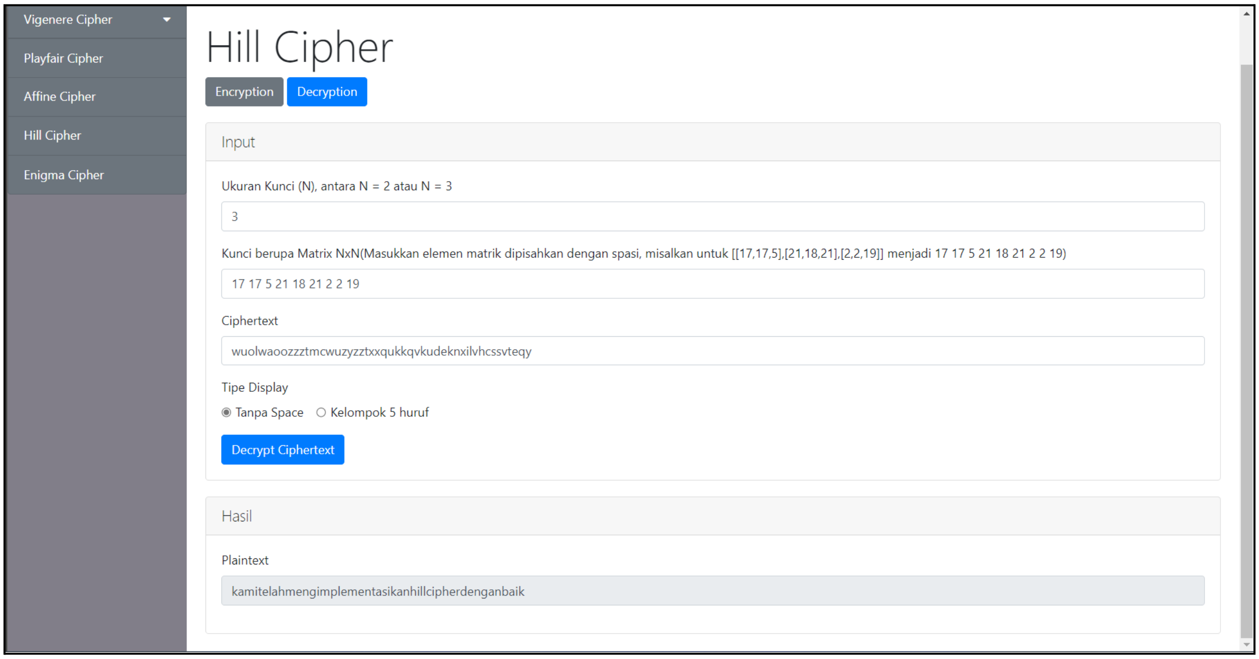
● Tanpa Spasi ○ Kelompok 5 huruf

Hasil

Ciphertext

wuolwaoozzztmcwuzyzztxxqukkqvkudeknxilvhcssvteqy

Contoh keluaran dekripsi semula:
Plaintext:
`kamitelahmengimplementasikanhillcipherdenganbaik`
Tampilan keluaran dekripsi semula (pada aplikasi web):



Hill Cipher

Input

Ukuran Kunci (N), antara N = 2 atau N = 3

3

Kunci berupa Matrix NxN(Masukkan elemen matrik dipisahkan dengan spasi, misalkan untuk [[17,17,5],[21,18,21],[2,2,19]] menjadi 17 17 5 21 18 21 2 2 19)

17 17 5 21 18 21 2 2 19

Ciphertext

wuolwaoozzztmcwuzyzztxxqukkqvkudeknxilvhcssvteqy

Tipe Display

● Tanpa Space ○ Kelompok 5 huruf

Hasil

Plaintext

kamitelahmengimplementasikanhillcipherdenganbaik

Enigma Cipher

Cipher ini menerima kunci dan *plaintext* alfabetik. Kunci yang digunakan adalah posisi awal rotor. Selain itu, konfigurasi algoritme dapat ditentukan yang meliputi susunan karakter rotor, susunan karakter reflektor, serta karakter rotor khusus untuk memutar rotor sebelumnya (disebut juga *notch*). Kode sumber yang digunakan adalah `enigma_cipher.py`. Pada antarmuka web, konfigurasi rotor yang digunakan berasal dari [2] sehingga kunci yang diberikan harus memiliki panjang tepat tiga karakter.

Contoh masukan:
Konfigurasi: (sesuai dengan [2])

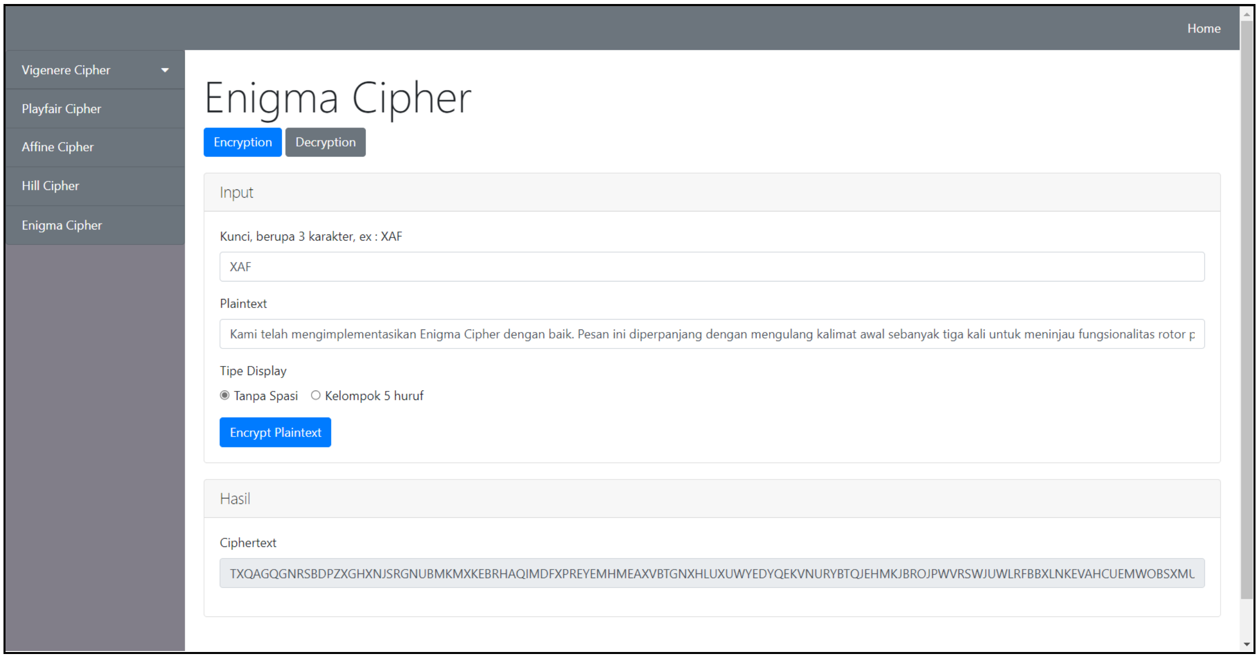
- Roda pertama: (rotor I, Enigma I)
  - Susunan huruf = EKMFLGDQVZNTOWYHXUSPAIBRCJ
  - *Notch* = Q
- Roda kedua: (rotor II, Enigma I)
  - Susunan huruf = AJDKSIRUXBLHWTMCQGZNPYFVOE
  - *Notch* = E
- Roda ketiga: (rotor III, Enigma I)
  - Susunan huruf = BDFHJLCPRTXVZNYEIWGAKMUSQO
  - *Notch* = V
- Reflektor: (Reflector B)
  - Susunan huruf = YRUHQSLDPXNGOKMIEBFZCWVJAT

Kunci: posisi awal = XAF
Plaintext:
Kami telah mengimplementasikan Enigma Cipher dengan baik. Pesan ini diperpanjang dengan mengulang kalimat awal sebanyak tiga kali untuk meninjau fungsionalitas rotor pertama. Kami telah mengimplementasikan Enigma Cipher dengan baik. Kami telah mengimplementasikan Enigma Cipher dengan baik. Kami telah mengimplementasikan Enigma Cipher dengan baik.

Contoh keluaran enkripsi:
Ciphertext:

```
TXQAGQGNRSBDPZXGHXNJSRGNUBMKMXKEBRHAQIMDFXPREYEMHMEAXVBTGNXHLUXUWYEDYQ
EKVNURYBTQJEHMKJBROJPWVRSWJUWLRFBBXLNKEVAHCUEMWOBSXMUTCTNKTZKBDQWMURPU
ZKSZAJKRXEFTEWWNPOYQLWFSOOXNOKMGNPGCQQIIUBMWDRDWTGDWJKDXXJTQLQHSJRLFSS
RJGLKTQKVLWWNLJSHBJRGDEZMRHBZDSHZYNJKRDUOOTBCGQKHLORIALIWQHZTQDXLVOLKH
EWRBSEERMCTLNXKYAQOKD
```

Tampilan keluaran enkripsi (pada aplikasi web):



Enigma Cipher

Input

Kunci, berupa 3 karakter, ex : XAF

XAF

Plaintext

Kami telah mengimplementasikan Enigma Cipher dengan baik. Pesan ini diperpanjang dengan mengulang kalimat awal sebanyak tiga kali untuk meninjau fungsionalitas rotor p

Tipe Display

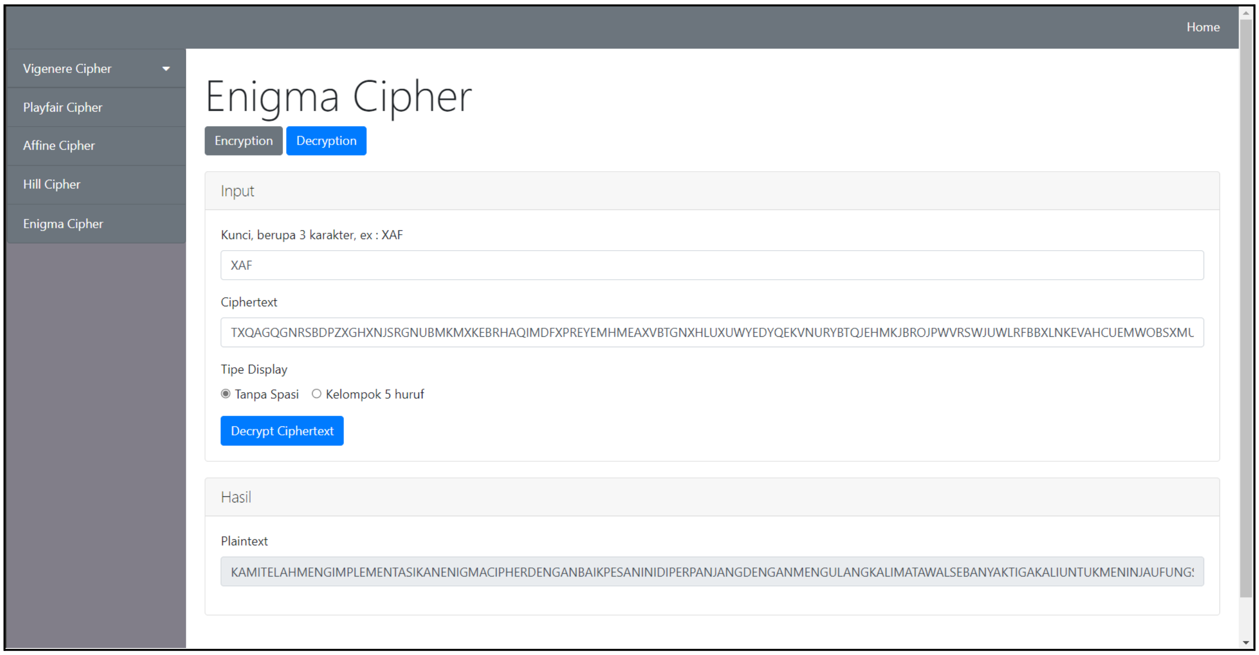
● Tanpa Spasi ○ Kelompok 5 huruf

Hasil

Ciphertext

TXQAGQGNRSBDPZXGHXNJSRGNUBMKMXKEBRHAQIMDFXPREYEMHMEAXVBTGNXHLUXUWYEDYQEKVNURYBTQJEHMKJBROJPWVRSWJUWLRFBBXLNKEVAHCUEMWOBSXML

Contoh keluaran dekripsi semula:
Plaintext:
KAMITELAHMENGIMPLEMENTASIKANENIGMACIPHERDENGANBAIKPESANINIDIPERPANJANGDENGANMENGULANGKALIMATAWALSEBANYAKTIGAKALIUNTUKMENINJAUFUNGSIONALITASROTORPERTAMAKAMITELAHMENGIMPLEMENTASIKANENIGMACIPHERDENGANBAIKKAMITELAHMENGIMPLEMENTASIKANENIGMACIPHERDENGANBAIKKAMITELAHMENGIMPLEMENTASIKANENIGMACIPHERDENGANBAIK
Tampilan keluaran dekripsi semula (pada aplikasi web):



Enigma Cipher

Input

Kunci, berupa 3 karakter, ex : XAF

XAF

Ciphertext

TXQAGQGNRSBDPZXGHXNJSRGNUBMKMXKEBRHAQIMDFXPREYEMHMEAXVBTGNXHLUXUWYEDYQEKVNURYBTQJEHMKJBROJPWVRSWJUWLRFBBXLNKEVAHCUEMWOBSXML

Tipe Display

● Tanpa Spasi ○ Kelompok 5 huruf

Hasil

Plaintext

KAMITELAHMENGIMPLEMENTASIKANENIGMACIPHERDENGANBAIKPESANINIDIPERPANJANGDENGANMENGULANGKALIMATAWALSEBANYAKTIGAKALIUNTUKMENINJAUFUNGS

Kode Sumber

Pada bagian ini, kode dan bagian kode yang dilampirkan adalah kode yang berkaitan langsung dengan kriptografi dan dirujuk pada bagian Hasil. Kode yang lengkap dapat ditemukan pada tautan https://github.com/AbdiHaryadi/classic-cryptology.

affine_cipher.py

```
from letters import Letters

class AffineCipher:
    def __init__(self, scale_key: int, bias_key: int):
        if self._gcd(scale_key, 26) != 1:
            raise ValueError("Module key must be relatively prime to
bias key.")

        self._scale_key = scale_key
        self._bias_key = bias_key
        self._determine_inverse_scale_key()

    def _gcd(self, a, b):
        if b == 0:
            return a
        else:
            return self._gcd(b, a % b)

    def _determine_inverse_scale_key(self):
        for key in range(1, 26):
            if (self._scale_key * key) % 26 == 1:
                # Found it!
                self._inverse_scale_key = key
                break

    def encrypt(self, message: str):
        ciphertext = ""
        for m_letter in Letters(message):
            m_letter_index = self._get_letter_index(m_letter)
            c_letter_index = (self._scale_key * m_letter_index +
self._bias_key) % 26
            ciphertext += self._get_letter_by_index(c_letter_index)

        return ciphertext

    def _get_letter_index(self, letter):
        return ord(letter) - ord("A")

    def _get_letter_by_index(self, letter_index):
        return chr(letter_index + ord("A"))

    def decrypt(self, ciphertext: str):
        message = ""
        for c_letter in Letters(ciphertext):
            c_letter_index = self._get_letter_index(c_letter)
            m_letter_index = ((c_letter_index - self._bias_key) *
self._inverse_scale_key) % 26
            message += self._get_letter_by_index(m_letter_index)

        return message
```

auto_key_vigenere_cipher.py

```
import string
from itertools import chain
from letters import Letters
from standard_vigenere_table_generator import
StandardVigenereTableGenerator

class AutoKeyVigenereCipher:
    def __init__(self, key):
        if key == "":
            raise ValueError("Key cannot be empty.")
        self._key = key
        self._generate_table()

    def _generate_table(self):
        self._table = StandardVigenereTableGenerator().generate()

    def _index_to_upper_letter(self, c_char_num):
        return chr(c_char_num + ord('A'))

    def encrypt(self, message):
        ciphertext = ""
        key_iterator = chain(Letters(self._key), Letters(message))
        for m_char, k_char in zip(Letters(message), key_iterator):
            ciphertext += self._table[k_char][m_char]

        return ciphertext

    def decrypt(self, ciphertext):
                                            ciphertext_fragments      =
self._make_fragments_with_key_length(ciphertext)

        message = ""
        current_key = self._key

        for fragment in ciphertext_fragments:
            next_key = ""
                       for  c_char,  k_char  in  zip(Letters(fragment),
Letters(current_key)):
                for m_char in string.ascii_uppercase:
                    if self._table[k_char][m_char] == c_char:
                        break # Found it!

                # table[k_char][m_char] == char

                message += m_char
                next_key += m_char
            current_key = next_key

        return message

    def _make_fragments_with_key_length(self, ciphertext):
        ciphertext_fragments = []
        start_index = 0
        key_length = len(self._key)

        while start_index < len(ciphertext):

ciphertext_fragments.append(ciphertext[start_index:start_index+key_len
gth])
            start_index += key_length
        return ciphertext_fragments
```

enigma_cipher.py

```
import string
from letters import Letters

class Rotor:
    def __init__(self, letters: str, notch="Z"):
        if notch not in string.ascii_uppercase:
            raise ValueError("Notch should be an uppercase letter.")

        self._r_letters = letters
        self._notch = notch
        self._position_index = 0

    def encipher(self, i_letter):
        if i_letter not in string.ascii_uppercase:
            raise ValueError("I-letter should be an uppercase letter.")

        i_letter_index = self._get_letter_index(i_letter)
        r_letter_index = (i_letter_index + self._position_index) % 26
        l_letter = self._r_letters[r_letter_index]
        l_letter_index = (self._get_letter_index(l_letter)) % 26
        o_letter_index = (l_letter_index - self._position_index) % 26
        return self._get_letter_by_index(o_letter_index)

    def decipher(self, o_letter):
        if o_letter not in string.ascii_uppercase:
            raise ValueError("O-letter should be an uppercase letter.")

        o_letter_index = self._get_letter_index(o_letter)
        l_letter_index = (o_letter_index + self._position_index) % 26
        l_letter = self._get_letter_by_index(l_letter_index)

        for r_letter_index in range(26):
            if self._r_letters[r_letter_index] == l_letter:
                break # Found it!

        i_letter_index = (r_letter_index - self._position_index) % 26
        return self._get_letter_by_index(i_letter_index)

    def _get_letter_index(self, m_letter):
        return ord(m_letter) - ord("A")

    def _get_letter_by_index(self, m_letter_index):
        return chr(m_letter_index + ord("A"))

    def advance(self):
        self._position_index = (self._position_index + 1) % 26

    @property
    def notch(self):
        return self._notch

    @property
    def position(self):
        return self._get_letter_by_index(self._position_index)

    @position.setter
    def position(self, new_position):
        if new_position not in string.ascii_uppercase:
            raise ValueError("Position should be an uppercase letter.")

        self._position_index = self._get_letter_index(new_position)

class Reflector:
    def __init__(self, letters: str):
        for letter_index in range(26):
            reflected_letter = letters[letter_index]
                                            reflected_letter_index    =
self._get_letter_index(reflected_letter)

                               if   letters[reflected_letter_index]   !=
self._get_letter_by_index(letter_index):
                raise ValueError("Letters is not reflexive.")

        self._letters = letters

    def reflect(self, letter):
        if letter not in string.ascii_uppercase:
            raise ValueError("M-letter should be an uppercase letter.")

        reflected_letter_index = self._get_letter_index(letter)
        return self._letters[reflected_letter_index]

    def _get_letter_index(self, m_letter):
        return ord(m_letter) - ord("A")

    def _get_letter_by_index(self, m_letter_index):
        return chr(m_letter_index + ord("A"))

class EnigmaCipher:
       def  __init__(self,  rotors:  list[Rotor],  reflector:  Reflector,
initial_positions="AAA"):
        if len(rotors) != len(initial_positions):
               raise  ValueError("Length  of  rotors  and  initial  position
does not match.")
```

```
         if any([(position not in string.ascii_uppercase) for position
in initial_positions]):
              raise ValueError("Each positions should only an uppercase
letter.")

        self._rotors = rotors
        self._reflector = reflector
        self._initial_positions = initial_positions

    def _reset_rotors(self):
                  for rotor, initial_position in zip(self._rotors,
self._initial_positions):
            rotor.position = initial_position

    def encrypt(self, message: str):
        self._reset_rotors()
        ciphertext = ""

        for m_letter in Letters(message):
            self._advance_rotors()

            current_letter = m_letter
            for rotor in self._rotors[::-1]:
                current_letter = rotor.encipher(current_letter)

            current_letter = self._reflector.reflect(current_letter)

            for rotor in self._rotors:
                current_letter = rotor.decipher(current_letter)

            ciphertext += current_letter

        return ciphertext

    def _advance_rotors(self):
        advance_rotor = True
        rotor_index = 2
        while advance_rotor:
            rotor = self._rotors[rotor_index]
            prev_rotor_position = rotor.position
            rotor.advance()

            if prev_rotor_position == rotor.notch and rotor_index > 0:
                rotor_index -= 1
            else:
                advance_rotor = False

    def decrypt(self, ciphertext):
        return self.encrypt(ciphertext)
```

extended_vigenere_cipher.py

```
from itertools import cycle

class ExtendedVigenereKeyCycler:
    def __init__(self, key: bytes):
        if len(key) == 0:
            raise ValueError("Key cannot be empty.")

        self._key = key

    def __iter__(self):
        self._key_iter = cycle(self._key)
        return self

    def __next__(self):
        return next(self._key_iter)

class ExtendedVigenereCipher:
    def __init__(self, key: bytes):
        if len(key) == 0:
            raise ValueError("Key cannot be empty.")

        self._key = key
        self._generate_table()
    def _generate_table(self):
        table = {}
        for k_index in range(256):
            table[k_index] = {}
            for m_index in range(256):
                c_index = (m_index + k_index) % 256
                table[k_index][m_index] = c_index

        self._table = table

    def encrypt(self, message: bytes):
        ciphertext_indices = []
        for m_index, k_index in zip(message,
ExtendedVigenereKeyCycler(self._key)):
            ciphertext_indices.append(self._table[k_index][m_index])

        return bytes(ciphertext_indices)

    def decrypt(self, ciphertext: bytes):
        message_indices = []
        for c_index, k_index in zip(ciphertext,
ExtendedVigenereKeyCycler(self._key)):
            for m_index in range(256):
                if self._table[k_index][m_index] == c_index:
                    break # Found it!

            # table[k_index][m_index] == c_index

            message_indices.append(m_index)

        return bytes(message_indices)

    @property
    def table(self):
        return self._table.copy()
```

hillCipher.py

```
import numpy as np
from numpy.linalg import inv
import string
from algorithm.letters import Letters

class HillCipher:
    def __init__(self, plain=None, cypher=None, key=None):
        self.plain = np.array([letter for letter in Letters("".join(plain))] if plain else [])
        self.cypher = np.array([letter for letter in Letters("".join(cypher))] if cypher else [])
        self.key = key
        self.maxPartisi = 3 if (len(self.key)==3) else 2

    def setCypher(self, cypher):
        self.cypher = cypher

    def getPlain(self):
        return self.plain

    def getKey(self):
        return self.key

    def getCypher(self):
        return self.cypher

    def encrypt(self, partialPlain):
        return np.matmul(self.key, partialPlain.transpose())%26

    def doEncryptAll(self):
        # mengencrypt per maxPartisi huruf
        count = 0
        partisi = np.array([])
        cypherNum = np.array([])
        idx = 0

        while ((len(self.getPlain())>idx)):
            # ubah alpha ke num dan append ke per partisi
            partisi = np.append(partisi, self.getPlain()[idx])
            count+=1
            if (count == self.maxPartisi):
                # lakukan encrypt partial dan append ke array encrypt
                partialPlainNum = self.convertAllAlphaToNum(partisi)
                                    cypherNum  =  np.append(cypherNum,
self.encrypt(partialPlainNum))
                # hapus count dan kosongkan per3char
                count = 0
                partisi = np.array([])

            idx+=1

        # ubah num ke alpha lalu tambahkan ke self.cyper
        cypherAlpha = self.convertAllNumToAlpha(cypherNum.astype(int))
        self.cypher = np.append(self.cypher, cypherAlpha)



    def decrypt(self, partialCypher):
            return  np.matmul(self.matrixInverseModulo(matrix=self.key,
divisor=26), partialCypher.transpose())%26

    def doDecryptAll(self):
        # mendecrypt per maxPartisi huruf
        count = 0
        partisi = np.array([])
        plainNum = np.array([])
        idx = 0

        while ((len(self.getCypher())>idx)):
            # ubah alpha ke num dan append ke per partisi
            partisi = np.append(partisi, self.getCypher()[idx])
            count+=1
            if (count == self.maxPartisi):
                # lakukan decrypt partial dan append ke array derypt
```

```
                                                   partialCypherNum   =
self.convertAllAlphaToNum(partisi).astype(int)
                                      plainNum   =   np.append(plainNum,
self.decrypt(partialCypherNum))
                # hapus count dan kosongkan partisi
                count = 0
                partisi = np.array([])

            idx+=1

        # ubah num ke alpha lalu tambahkan ke self.plain
        plainAlpha = self.convertAllNumToAlpha(plainNum.astype(int))
        self.plain = np.append(self.plain, plainAlpha)

    def convertAllAlphaToNum(self, alpha):
        num = np.array([])
        for i in range(len(alpha)):
            num = np.append(num, self.alphaToNum(alpha[i]))
        return num

    def convertAllNumToAlpha(self, num):
        alpha = np.array([])
        for i in range(len(num)):
            alpha = np.append(alpha, self.numToAlpha(num[i]))
        return alpha

    def alphaToNum(self, alpha):
        # Increment character
        if alpha.isupper():
            # Use ascii_uppercase if character is uppercase
            letters = string.ascii_uppercase
        else:
            # Use ascii_lowercase if character is lowercase
            letters = string.ascii_lowercase

        # Find index of character in letters
        index = letters.index(alpha)
        return index

    def numToAlpha(self, num):
        ch = 'a'
        # Increment character
        if ch.isupper():
            # Use ascii_uppercase if character is uppercase
            letters = string.ascii_uppercase
        else:
            # Use ascii_lowercase if character is lowercase
            letters = string.ascii_lowercase

        # Find index of character in letters
        index = letters.index(ch)

        # Increment index and retrieve next character from letters
        str = letters[index + num]

        return str

    def moduloInverse(self, dividend, divisor):
        return pow(dividend,-1,divisor)

    def matrixInverseModulo(self, matrix, divisor):
        # inisiasi
        inverseDet = None
        res = np.array([])
        # cek panjang matrix (buat penyesuaian method invers)
        if (len(matrix) == 2):
            # dapatkan det(matrix)
                                 det   =   matrix[0][0]*matrix[1][1]   -
matrix[1][0]*matrix[0][1]
            # inverse Det dengan divisor
            inverseDet = self.moduloInverse(det, divisor)
            # matriks
                     res  =  np.array([  [matrix[1][1],  -matrix[0][1]],
[-matrix[1][0], matrix[0][0]] ])
```

```
        if (len(matrix) == 3):
            # method crammer
            A = matrix[1][1]*matrix[2][2] - matrix[2][1]*matrix[1][2]
                                 B   =   -(matrix[1][0]*matrix[2][2]   -
matrix[1][2]*matrix[2][0])
            C = matrix[1][0]*matrix[2][1] - matrix[1][1]*matrix[2][0]
                                 D   =   -(matrix[0][1]*matrix[2][2]   -
matrix[2][1]*matrix[0][2])
            E = matrix[0][0]*matrix[2][2]-matrix[0][2]*matrix[2][0]
                                 F   =   -(matrix[0][0]*matrix[2][1]   -
matrix[0][1]*matrix[2][0])
            G = matrix[0][1]*matrix[1][2] - matrix[1][1]*matrix[0][2]
                                 H   =   -(matrix[0][0]*matrix[1][2]   -
matrix[0][2]*matrix[1][0])
            I = matrix[0][0]*matrix[1][1] - matrix[0][1]*matrix[1][0]
            # dapatkan det(matrix)
            det = matrix[0][0]*A + matrix[0][1]*B + matrix[0][2]*C
            # inverse Det dengan divisor
            inverseDet = self.moduloInverse(det, divisor)
            # matriks
            res = np.array([ [A, D, G], [B, E, H], [C, F, I] ])


        return (inverseDet*res).astype(int)
```

letters.py

```
import string

class Letters:
    def __init__(self, text):
        self._letters = ""
        for character in text.upper():
            if character in string.ascii_uppercase:
                self._letters += character

    def __iter__(self):
        self._letters_iter = iter(self._letters)
        return self

    def __next__(self):
        return next(self._letters_iter)
```

playfair_cipher.py

```
from itertools import chain
import string
from letters import Letters

class PlayfairTableGenerator:
    def generate(self, key: str):
        used_letters = set()
        table_sequence = []

        for letter in chain(Letters(key), string.ascii_uppercase):
            if letter in used_letters or letter == "J":
                continue # Skip

            table_sequence.append(letter)
            used_letters.add(letter)

        return [
            table_sequence[0:5],
            table_sequence[5:10],
            table_sequence[10:15],
            table_sequence[15:20],
            table_sequence[20:25],
        ]

class PlayfairBigrams:
    def __init__(self, text):
        self._letters = Letters(text)

    def __iter__(self):
        self._letters_iter = iter(self._letters)
        self._next_first_letter = None
        return self

    def __next__(self):
        first_letter = self._get_first_letter()
        second_letter = next(self._letters_iter, "X").replace("J", "I")

        if first_letter != second_letter:
            return first_letter + second_letter
        else:
            self._next_first_letter = second_letter
            if first_letter != "X":
                return first_letter + "X"
            else:
                return first_letter + "Q"

    def _get_first_letter(self):
        if self._next_first_letter is None:
            return next(self._letters_iter).replace("J", "I")
        else:
            first_letter = self._next_first_letter
            self._next_first_letter = None
            return first_letter

class PlayfairCipher:
    def __init__(self, key: str):
        self._table = PlayfairTableGenerator().generate(key)

    def encrypt(self, message: str):
        ciphertext = ""
        for bigram in PlayfairBigrams(message):
            m_first_row, m_first_col = self._get_position(bigram[0])
            m_second_row, m_second_col = self._get_position(bigram[1])

            if m_first_row == m_second_row:
                  ciphertext += self._table[m_first_row][(m_first_col +
1) % 5]
                 ciphertext += self._table[m_first_row][(m_second_col +
1) % 5]
```

```
            elif m_first_col == m_second_col:
                        ciphertext += self._table[(m_first_row + 1) %
5][m_first_col]
                       ciphertext += self._table[(m_second_row + 1) %
5][m_first_col]

            else: # different row, different column
                ciphertext += self._table[m_first_row][m_second_col]
                ciphertext += self._table[m_second_row][m_first_col]

        return ciphertext

    def decrypt(self, ciphertext: str):
        message = ""
        for bigram in PlayfairBigrams(ciphertext):
            m_first_row, m_first_col = self._get_position(bigram[0])
            m_second_row, m_second_col = self._get_position(bigram[1])

            if m_first_row == m_second_row:
                  message += self._table[m_first_row][(m_first_col - 1)
% 5]
                 message += self._table[m_first_row][(m_second_col - 1)
% 5]

            elif m_first_col == m_second_col:
                          message += self._table[(m_first_row - 1) %
5][m_first_col]
                         message += self._table[(m_second_row - 1) %
5][m_first_col]

            else: # different row, different column
                message += self._table[m_first_row][m_second_col]
                message += self._table[m_second_row][m_first_col]

        return message

    def _get_position(self, character):
        result = (-1, -1) # Initialize with dummy
        for row in range(5):
            for col in range(5):
                if self._table[row][col] == character:
                    result = (row, col)

        return result
```

standard_vigenere_cipher.py

```
from itertools import cycle
import string
from letters import Letters
from standard_vigenere_table_generator import
StandardVigenereTableGenerator

class VigenereKeyCycler:
    def __init__(self, key):
        if key == "":
            raise ValueError("Key cannot be empty.")

        self._key_letters = ""
        for character in key.upper():
            if character in string.ascii_uppercase:
                self._key_letters += character

    def __iter__(self):
        self._key_iter = cycle(self._key_letters)
        return self

    def __next__(self):
        return next(self._key_iter)

class StandardVigenereCipher:
    def __init__(self, key):
        if key == "":
            raise ValueError("Key cannot be empty.")
        self._key = key
        self._table = StandardVigenereTableGenerator().generate()

    def encrypt(self, message):
        ciphertext = ""
        for m_char, k_char in zip(Letters(message),
VigenereKeyCycler(self._key)):
            ciphertext += self._table[k_char][m_char]

        return ciphertext

    def decrypt(self, ciphertext):
        message = ""
        for c_char, k_char in zip(Letters(ciphertext),
VigenereKeyCycler(self._key)):
            for m_char in string.ascii_uppercase:
                if self._table[k_char][m_char] == c_char:
                    break # Found it!

            # table[k_char][m_char] == char

            message += m_char

        return message
```

standard_vigenere_table_generator.py

```
import string

class StandardVigenereTableGenerator:
    def generate(self):
        table = {}
        for k_char_index, k_char in enumerate(string.ascii_uppercase):
            table[k_char] = {}
            for m_char_index, m_char in
enumerate(string.ascii_uppercase):
                c_char_index = (m_char_index + k_char_index) % 26
                c_char = self._index_to_upper_letter(c_char_index)
```

```python
                table[k_char][m_char] = c_char

        return table

    def _index_to_upper_letter(self, c_char_num):
        return chr(c_char_num + ord('A'))
```

Bagian B: Kriptanalisis

Tabel B.I Rangkuman Penyelesaian Tugas Bagian B

No.	**Spek**	**Berhasil**	**Kurang berhasil**	**Keterangan**
1	Kriptanalisis *Cipher* Abjad Majemuk	V		
2	Metode Kasiski	V		
3	Kriptanalisis *Playfair Cipher*		V	Kesulitan pada penyusunan papan setelah ditemukan aturan translasi *bigram*
4	Kriptanalisis *Hill Cipher*		V	Kesulitan menemukan kunci matriks yang cocok

Tabel II menunjukkan rangkuman penyelesaian tugas bagian B. Pada kriptanalisis *playfair cipher*, metode statistik dengan meninjau *bigram* dan *quadgram* telah digunakan. Akan tetapi, penyusunan papan dari translasi menjadi kesulitan dalam pengerjaan tugas. Selain itu, pencarian kunci dengan algoritme *simulated annealing* telah dilakukan. Namun, kunci juga belum ditemukan. Penggabungan validasi translasi pada *simulated annealing* tidak *feasible* karena membangkitkan kunci dengan kondisi tersebut memerlukan waktu jika menggunakan *brute-force*, atau algoritme tertentu yang belum dieksplorasi.

Kriptanalisis *Cipher* Abjad Majemuk

Mula-mula, tinjau karakter terbanyak terlebih dahulu. Didapatkan `W` sebagai karakter terbanyak sehingga diduga itu adalah `E` yang juga merupakan karakter terbanyak dalam bahasa Inggris [1]. Kemudian, tinjau trigram terbanyak. Didapatkan lima trigram terbanyak sebagai berikut:

1) `CZW` (102)
2) `NJP` (51)
3) `FJY` (36)
4) `WKW` (26)
5) `SNJ` (23)

Didapatkan `CZW` sebagai trigram terbanyak sehingga diduga itu adalah `THE` sebagai trigram terbanyak dalam bahasa Inggris [1].

Selanjutnya, tinjau pola yang memenuhi `CZ.C` (tanda titik menunjukkan karakter apapun). Mengingat penggalan *string* `CZFC` yang paling banyak, diduga itu adalah `THAT`. Maka, jika itu benar, `FJY` dapat diduga sebagai `AND` karena merupakan trigram yang terbanyak yang diawali dengan huruf A, dilanjutkan dengan NJP sebagai ING karena merupakan trigram terbanyak kedua [1] (karena AND sudah ditentukan sebagai ketiga). Selain itu, WKW dianggap sebagai ERE karena merupakan trigram yang cukup banyak dengan karakter awal dan akhir yang sama.

Langkah selanjutnya hanya berupa tebakan. Huruf nonkapital adalah huruf yang telah di-*decipher*.

- Xriginated -> originated. Maka, X -> o.
- norQegian -> norwegian. Maka, Q -> z.
- Ltrength -> strength. Maka, L -> s.
- regardVess -> regardless. Maka, V -> l (huruf L nonkapital).
- UarioHs -> various. Maka, U -> v, H -> u.
- thehistoriA -> thehistoric; whileothersAlaiE -> whileothersclaim. Maka, A -> c, E -> m.
- OrutalitR -> brutality; OerememOered -> beremembered. Maka, O -> b, R -> y.
- viSing -> viking. Maka, S -> k.
- islamicemGire -> islamicempire. Maka, G -> p.
- dwellersoBviken -> dwellersofviken. Maka, B -> f.
- Tuickly -> quickly. Maka, T -> q.
- wereeIperiencing -> wereexperiencing, foreIample -> foreexample. Maka, I -> x.

Pengubahan tersebut cukup untuk mendekripsi pesan berikut:

```
CZWKWFKWUFKNXHLCZWXKNWLFLCXQZWKWCZWCWKEUNSNJPXKNPNJFCWYXJWV
XXSLCXCZWBWENJNJWGKWBNIUNSEWFJNJPNJVWCXKOFRQZNVWXCZWKLAVFNE
CZWZNLCXKNAJXKQWPNFJLWCCVWEWJCXBUNSWJNLQZWKWCZWJFEWYWKNUWLZ
WJAWUNSNJPLQWKWCZWXKNPNJFVYQWVVWKLXBUNSWJFVCWKJFCNUWVRKWAXP
JNLWYWCREXVXPNLCLLHAZFLFJFCXVRVNOWKEFJGXNJCCXCZWXVYJXKLWQXK
YUNSFEWFJNJPLWFENVWCZWLGFAWVWBCOWCQWWJCQXKXQNJPOXFCLNJAXJUX
RQZNVWXCZWKLKWBWKCXCZWCZAWJCHKRFJPVXLFIXJGXWEQNYLNCZQZNAZKW
BWKLCXLAFJYNJFUNFJGNKFCWLFLQNANJPLKWPFKYVWLLPNUWJCZWLAFJYNJ
FUNFJLYXENJFJAWXBCZWLWFYHKNJPCZNLWFKVRGWKNXYNCQFLJCVXJPHJCN
VAXJCKFLCNJPJFEWLQWKWOWNJPXBBWKWYHGORUFKNXHLXCZWKAHVCHKWLFA
KXLLCZWGVFJWCCZWFKFOLLVFULFJYORDFJCNJWLBXKWIFEGVWSJWQXBCZWL
WKFNYWKLFLKHLXKKZXLKWVFCNJPCXKXQNJPQZNVWCZWPWKEFJLVFOWVVWYC
ZWEFLFLAXEFJJNFLZEWJFVVHYNJPCXCZWNKFLZQXXYOXFCLXCZWKJFCNXJL
LHAZFLCZWWJPVNLZFJYAWVCLLWCCVWYEWKWVRXJYFJWLZWFCZWJLXKGFPFJ
LQZNVLCCZWNKNLZKWBWKKWYCXCZWEFLYHOPFNVFJYBNJJPFNVYFKSFJYBFN
KBXKWNPJWKLXKTHNCWLHNCFOVRJXKCZEWJPNUWJCZWAXEEXJAHVCHKFVGKF
ACNAWXBCXKEWJCNJPAXFLCFVLWCCVWEWJCLFJYEXJFLCWKNWLNCQFLJCVXJ
PHJCNVCZWUNSNJPLBWFKLXEWKWGHCFCNXJLGKWFYCXFVEXLCFVVAXKJWKLX
BWHKXGWFJYEWLXGXCFENFCZWKWNLWUWJWUNYWJAWCZFCCZWUNSNJPLKWFAZ
WYOFPZYFYCZWAWJCKWXBCZWNLVFENAWEGNKWFCCZWCNEWCZWUNSNJPFPWFL
AXEEXJVRKWBWKKWYCXVFLCWYBKXECZWWFKVRLCXCZWJXKEFJAXJTHWLCXBW
```

```
JPVFJYNJCZKXHPZXHCCZNLGWKNXYCZWUNSNJPLHLWYCZWJXKCZWKJFJYOFV
CNALWFLCXCWKKXKNLWJWNPZOXHKNJPSNJPYXELWICWJYNJPCZWNKNJBVHWJ
AWCZKXHPZAXEOFCFJYAHVCHKWHJCNVWUWJCHFVVRUNSNJPLAXHVYJXVXJPW
KOWEWKWVRYWLAKNOWYFLAXFLCFVKFNYWKLAXJLNYWKCZWBFACLCQXUNSNJP
SNJPLLQWRJBXKSOWFKYFJYAJHCCZWPKWFCQXHVYFLAWJYCZWWJPVNLZCZKX
JWVWNBWKNSLXJFJWFKVRNAWVFJYWKQXHVYLWCCVWLZXKCNUWYAXVXJNWLNJ
JXKCZFEWKNAFLAFJYNJFUNFJLQXHVYWUWJLWKUWFLEWKAWJFKNWLBXKCZWO
RDFJCNJWWEGNKWNJLZXKCCZWLWQWKWJXEWKWGNKFCWLOHCCZWBXKWBFCZWK
LXBFGFCAZQXKSTHNVCAHVCHKWCZWEXCNUFCNXJLBXKLHAZWIGFJLNXJFKWL
HOMWACCXYWOFCWBXKEXYWKJZNLCXKNFJLCZXHPZCZWKWFKWAVWFKNJAWJCN
UWLFLCXQZRCZWGXGHVFCNXJXBLAFJYNJFUNFENPZCZFUWFACWYNJCZWQFRC
ZFCCZWRYNYYHKNJPCZNLRWFKGWKNXYXJWKWVFCNUWVRFGGFKWJCKWFLXJNL
FLAFKANCRXBKWLXHKAWLCZHLBXKANJPCZWUNSNJPLCXVXXSBHKCZWKFBNWV
YWUWJKXOONJPFJYSNVVNJPAVFLLWLXBGWXGVWOVWLLWYQNCZFEXKWOXHJCN
BHVZXEWVFJYFJXCZWKGXLLNOVWLCNEHVHLNLCZWKHVWXBAZFKVWEFPJWFJY
CZWKWVNPNXHLGWKLWAHCNXJCZFCQWJCZFJYNJZFJYQNCZNCQNCZAZKNLCNF
JNJBVHWJAWLWWGNJPWUWKBHKCZWKNJCXYWJEFKSLQWYWJFJYJXKQFRNCEFS
WLVXPNAFVLWJLWCZFCCZWUNSNJPLQWKWVXXSNJPCXGKXCWACCZWNKGFPFJO
WVNWBLRLCWEKWLNLCMHYWXAZKNLCNFJUFVHWLFJYWUWJCFSWKWUWJPWBXKC
ZXLWLWCCVWEWJCLFVKWFYRVXLCCXFEXJXCZWNLCNAYWUXCNXJCZNLNLJXCL
GWAHVFCNXJCZWNJCKXYHACNXJXBAZKNLCNFJNCRQXHVYAXEWCXYNUNYWJXK
QFRBXKFVEXLCZFVBFAWJCHKRAFHLNJPHJCXVYOVXXYLZWYFJYAHVCHKFVCK
FJLBXKEFCNXJNCLZXHVYFVLXOWJXCWYCZFCYHKNJPCZWUNSNJPFPWLAFJYN
JFUNFLAVXLWLCJWNPZOXHKLQWKWWIGWKNWJANJPUFKRNJPVWUWVLXBNJJWK
CHKEXNVCZHLPKFJCNJPCZWUNSNJPLFJFYUFJCFPWQZWJWIGVXNCNJPCZWLW
VFJYLBXKQWFVCZLVFUWLXKCWKKNCXKRCZWLWFKXHCWLHLWYORCZWUNSNJPL
QWKWFVEXLCWJCNKWVRBKWWXBXGGXLNCNXJVWFUNJPCZWKFNYWKLHJNEGWYW
YFLCZWRCKFUWVVWYBKXEXJWYWLCNJFCNXJXBGVHJYWKCXCZWJWICCZNLOKW
FSYXQJNJQZFCZFYXJAWOWWJFGKXBNCFOVWJWCQXKSXBCKFYWKXHCWLBXKWH
KXGWFJSNJPYXELAFJOWZWKFVYWYOFASFLBFKFLCZWAXVVFGLWXBCZWKXEFJ
WEGNKWNJCZWCZAWJCHKRFJYVFCWKCXCZWKFGNYCZAWJCHKRWIGFJLNXJXBN
LVFENAGZNVXLXGZRCZWWJYXBCZWUNSNJPFPWAFJOWGNJJWYYXQJCXFJHEOW
KXBBFACXKLBNKLCXBFVVCZWBFVVXHCCZFCXAAHKKWYBXVVXQNJPCZWAZKNL
CNFJNLFCNXJXBLAFJYNJFUNFQXHVYZFUWHJCXVYWBBWACLXJCZWKWPNXJLY
XEWLCNAFJYBXKWNPJGXVNARORCZWCZAWJCHKRYWJEFKSJXKQFRFJYLQWYWJ
QWKWWBBWACNUWVRAXJCKXVVWYORYNXAWLWLVWPNCNENLWYORCZWAFCZXVNA
AZHKAZFJYZFYBNKEVRWLCFOVNLZWYCZWELWVUWLFLLWGFKFCWSNJPYXELCZ
NLEWFJCFJWJXKEXHLAHVCHKFVLZNBCNJCZWGKNXKNCNWLXBLAFJYNJFUNFL
```

```
VWFYWKLZNGNJCZFCLWJLWCZWUNSNJPLQWKWJXCYWBWFCWYOHCFKPHFOVREF
YWCXOWZFUWNJFEFJJWKCZFCBNCCZWANUNVNCRXBCZWNKTHNASVRCKFJLBXK
ENJPZXEWVFJYLBXKWIFEGVWCZWEWYNWUFVAZHKAZEFYWNCBXKONYYWJCXCF
SWBWVVXQAZKNLCNFJLFLLVFUWLPNUWJCZWBFACCZFCLVFUWCKFYNJPQFLCZ
WJHEOWKXJWLXHKAWXBGKXBNCBXKCZWUNSNJPLCZNLKWEXUWYFPKWFCYWFVX
BCZWWAXJXENANJAWJCNUWCXCKFUWVFJYKFNYXUWKLWFLCZWJWQVWFYWKLZN
GFVLXAZXLWCXKWBXAHLCZWNKENVNCFKRFCCWJCNXJBKXECZWSNJPYXELXBC
ZWQWLCFJYNJLCWFYGFKCFSWNJLHAZAFEGFNPJLFLCZWOFVCNAQFKLFJYCZW
FCCWEGCWYAXJTHWLCXBMWKHLFVWEBKXEZWKWXJXHCNCLWWEWYCZWUNSNJPL
QWKWJXVXJPWKFKWAXPJNLWYBXKAWNJCZWQXKVYCZXHPZCZWNKOKHCFVNCRO
KFUWKRFJYLCKWJPCZQXHVYVXJPOWKWEWEOWKWYORCZXLWQZXZFYXJAWBWVC
CZWLZFKGWYPWXBCZWNKOFCCVWFIW
```

Hasilnya berasal dari tautan https://guidetoiceland.is/history-culture/vikings-and-norse-gods-in-iceland dengan isi sebagai berikut:

> The Viking Age, as commonly referred to, lasted from the early 790s to the Norman Conquest of England in 1066. Throughout this period, the Vikings used the Northern and Baltic seas to terrorise neighbouring kingdoms, extending their influence through combat and culture until, eventually, Vikings could no longer be merely described as coastal raiders.
>
> Consider the facts; two Viking kings, Sweyn Forkbeard and Cnut the Great, would ascend the English throne. Leif Erikson (an early Icelander) would settle short-lived colonies in North America. Scandinavians would even serve as mercenaries for the Byzantine Empire. In short, these were no mere pirates, but the forefathers of a patchwork-quilt culture.
>
> The motivations for such expansion are subject to debate for modern historians, though there are clear incentives as to why the population of Scandinavia might have acted in the way that they did during this 200 year period. One relatively apparent reason is a scarcity of resources, thus forcing the Vikings to look further afield, even robbing and killing classes of people blessed with a more bountiful homeland.
>
> Another possible stimulus is the rule of Charlemagne and the religious persecution that went hand-in-hand with it. With Christian influence seeping ever further into Denmark, Sweden and Norway, it makes logical sense that the Vikings were looking to protect their pagan belief system, resist Judeo-Christian values and even take revenge for those settlements already lost to a monotheistic devotion. This is not speculation; the introduction of Christianity would come to divide Norway for almost half a century, causing untold bloodshed and cultural transformation.

It should also be noted that during the Viking Age, Scandinavia's closest neighbours were experiencing varying levels of inner turmoil, thus granting the Vikings an advantage when exploiting these lands for wealth, slaves or territory. The sea routes used by the Vikings were almost entirely free of opposition, leaving the raiders unimpeded as they travelled from one destination of plunder to the next.

This breakdown in what had once been a profitable network of trade routes for European kingdoms can be heralded back as far as the collapse of the Roman Empire in the 5th Century, and later, to the rapid 7th Century expansion of Islamic philosophy.

The end of the Viking Age can be pinned down to a number of factors. First of all, the fallout that occurred following the Christianisation of Scandinavia would have untold effects on the region's domestic and foreign policy.

By the 12th Century, Denmark, Norway and Sweden were effectively controlled by dioceses legitimised by the Catholic Church and had firmly established themselves as separate Kingdoms. This meant an enormous cultural shift in the priorities of Scandinavia's leadership; in that sense, the Vikings were not defeated, but arguably, made to behave in a manner that fit the civility of their quickly transforming homelands.

For example, the medieval church made it forbidden to take fellow Christians as slaves. Given the fact that slave-trading was the number one source of profit for the Vikings, this removed a great deal of the economic incentive to travel and raid overseas. The new leadership also chose to refocus their military attention from the kingdoms of the west and, instead, partake in such campaigns as the Baltic Wars and the attempted conquest of Jerusalem.

From here on out, it seemed, the Vikings were no longer a recognised force in the world, though their brutality, bravery and strength would long be remembered by those who had once felt the sharp edge of their battleaxe.

Metode Kasiski

Mula-mula, prediksi panjang kunci dicari terlebih dahulu. Pada *ciphertext*, *substring* yang panjang dan berulang ditemukan. *Substring* beserta indeks karakter pertama (dimulai dari nol) yang ditinjau adalah sebagai berikut:

- `LMEQWZMCY`, indeks 118 dan 158;
- `RBFWREYPWY`, indeks 209 dan 349;
- `ECHYYEGK`, indeks 36 dan 386;
- `ISHIEBTQ`, indeks 260 dan 500; serta
- `GULBJWLMGS`, indeks 2.027 dan 2.087.

Perbedaan kedua posisi pada setiap *substring* yang ditinjau memiliki faktor persekutuan terbesar 10. Maka, panjang kunci yang diduga adalah 10.

Langkah selanjutnya adalah membagi *ciphertext* sehingga menjadi 10 karakter secara kontigu. Kemudian, hitung trigram pada setiap partisi beserta *offset* awalnya. Gabungkan hasilnya sehingga diperoleh lima trigram teratas sebagai berikut:

- XVP-4, frekuensi 4
- HSS-5, frekuensi 4
- LME-8, frekuensi 3
- PNI-2, frekuensi 3
- ZFR-8, frekuensi 3

Trigram yang tidak ditampilkan memiliki frekuensi kurang dari 3. Untuk indeks yang dapat melewati *substring* asal (contoh: LME-8), bagian yang terlewat menggunakan *substring* selanjutnya.

Pada peninjauan, XVP-4 (trigram XVP pada posisi 4) tidak membuahkan hasil yang baik sehingga yang selanjutnya ditinjau adalah HSS-5. Mengingat THE adalah trigram dengan frekuensi tertinggi pada bahasa Inggris [1], peninjauan pada HSS-5 pada salah satu bagian diilustrasikan sebagai berikut:

```
?????THE?? # Plaintext
?????OLO?? # Kunci (peninjauan plaintext dan ciphertext)
RLETHHSSLM # Ciphertext
```

Selanjutnya, kata THE digunakan lagi untuk PNI-2. Didapatkan sebagai berikut:

```
??THESPR??
??WGEOLO??
HXPNIGAFKY
```

Sisanya adalah tebak-tebakan. Partisi *ciphertext* yang digunakan selanjutnya adalah EMOREBOHNC sehingga diperoleh sebagai berikut:

```
??SLANDT??
??WGEOLO??
EMOREBOHNC
```

Dengan mencoba menyambungkan kata THE pada *plaintext*, diperoleh sebagai berikut:

```
??SLANDTHE
??WGEOLOGY
```

```
EMOREBOHNC
```

Kemudian, tinjau partisi *ciphertext* GLAVSDFZGR:

```
??EPOPULAT
??WGEOLOGY
GLAVSDFZGR
```

Gunakan lagi kata THE, diperoleh sebagai berikut:

```
THEPOPULAT
NEWGEOLOGY
GLAVSDFZGR
```

Maka, diberikan *ciphertext* sebagai berikut:

```
FSIKTSZDRCZEUGPFPOJWXRKCXVPVOQGSNESTECHYYEGKPCNOZCQMJTSFEVYSZEPXEDCCBG
AGAHYHQCXRUSOKSTJCAUUSZURCEYTMJXDKWHZFEZRLETHHSSLMEQWZMCYCLJNOAZSPLHNG
FXESIHSSXCVWOUQSTBLMEQWZMCYHNYGAERPPPROQPYOYIRNIXGBYOGWKSOZPREZOXRZKTR
BFWREYPWYMAIKLXVPZGPTIOZPOVSYGAWKAXVPOYRNWEGEBOWYYISHIEBTQRYXIETXVPAOB
QPAZLSCSOQNREYPOYRIYYPAJWOXCYGEMOREBOHNCZEFUVWEMUDGLAVSDFZGRVSJGVCFBJR
BFWREYPWYZNXWQXFTPKGGMOKWHTAGRRHPNEHECHYYEGKAODTUPZIZJYFTBMYAITVPCDWUL
BJWHSIEMKYEWWMSKSWIFVWWTIFFDZGBROATSCJUJPEJUWIASXTBPYGRCEVGRVWIUYBEHUZ
NAETHPWCCLISHIEBTQGQULWYWDCSGBGSDGPTEVKCNVPNJFZAIFVRWZSGZIZFNJNOGOPJKL
DYEZIGFFVPVWETKPPQGSFIEZXICBYMHXPNIGAFKYQSBZLSOIYRGSXKVSNCXBRHQVXCEVKL
BVPNTCWSZFRINATHTCTMPGQXVSOTUPBRACISVOTBGLAJYGEPAPFXNKEQSSJIVPKSIHPFYY
OSRKWSLZKTRPPNISGWJCAGALSIYRGJFSNKMBQCXARWPNIBZHOMAXDGXHSSLMEGAUJHSSKP
HTPOSBLBJGGWKIIOYKGTRWYUYZOOTLVLEREHPZODRMJGXZLBZGFXDOWWYQOBRRPIEIDSJK
NWOJIOEVLMYPKCIRMMZFRITZMBNHOMASBYSAPGVCPMAYEQNCXBVRCZSRYOKTVHATGSEVOQ
RVQVXWZBGJFSONVOYYZFRRQSFSCCLNRSLRIHZOHMHXKLXVPHURNPDAQOYDUNHPWZMCYCLR
UIAGVHSOZRUEPZMAPOHMHXIOPZTCTNRSLRIOQHKPGLAKVIAHOMAEYGPRPFGUNWBUVAPRCF
VGDZLSYTOJYIZCMHSKGRRVWTHPPQGKRADGXWDBUUXRKCRODHUZNPWQIIAKGPQTNKWGFFKZ
LXDKQORAGRUEPNEGYCZWRXYUQSZIZANYOKWHSSKKRVCKRQPCLQNQKYMFTGRYAHPNIFPWYY
YWKLSZVZUPREXUYHECHYYEGKMGDOOBUIOGMRLHOKRADKRHSSXCJEOGJOCAKPAEIKHHZPGU
USSKRHQWYFVRCZSHSSXGIINZSUPHLGFLPUIOEHUZNKAZWOMWMYAHXKEIEWLSYJEYLPFHNC
VWOAVDCWYCQXDGXHSSLGFLYGRHLZQYCTWXIBEZERUIBOWVTGZFRMJIEFYOZGBRKLEDCWTA
RWOCLCHOYAHVOKHTZFBGBPWZMBRHNCEYHKWCQHNCXMJMHCXOYYGLWTOMZILMELWBMBRFKJ
REOKHVPFLPBQPNIQFFYCGLAVVWYQKQFAWYQOCFERBFWNSKPJKPGLAXIWDCTCCVKSMGPHNY
GLWYFSPBGEEIAJYDZBZFRCONSIWRTMGXARPOYMULRXDGXHSSVPVRYKWGTGGDVWDLVCXHNC
ZENXMORSCYFFKXROMCELNQAJWOXCYGEWWSSGTFMPRAETXCLJKPLLWTHGZAKYAHOZVCYUHM
LFQZXVPFKYEIDGFWEGZFNXWSENPSBCECKTIVPORUNCOLISWGNSAKNEEBOBKTRVOGXWDTOC
```

```
QEHRXVPTUMQVWZMCYGGPREHCEMDRKTBYNKHKTHNMHXPNIFPGZSAXERSBPRGWFEIUWWCOYQ
VKJKHHZRKJVZAXJCZRLMELEYJOEVKPVRPNITTSRBOYPZLSQCUBAIRKVQLAKRBFWGPGZOVN
ESWILSOGGKBWEXEBOOYIRHSNIFPHNCSSKJJCCVOKOYPYEAZGOPNHIOXHPRZFNXPNITZCJG
FEHXIOOMKYGIJZSPLKGQIINEEBRFEYAHQTOBZKOLTPUHVSLYYFVWLXSATGKZLWWEMBRGUL
BJWLMGSGGKBWEXMAXSJGNXARCQZAVJNMJKHHZVOQZSPNIFTTNCPEHRIRLGULBJWLMGSHNC
CVETGSDGCYFHEYEDACOLGIZHIQLIYCUINNYGMOTBUEOHVCVSTRUILXSATGKQUIYXMSOORM
GEJJXCWRYYZSOOVHZFALGSPNIVTUNFVPHYYROSTJLZAXCVPOBWEEETGOXSJMJRSOXVLHKP
EMXRIZTUNRAMJMXVPKGRRVKBIFQZUURHPUHFZKTRUIATXWCSBGYPWMIHSSVSQHHKXICBKB
VRPUEZLYKRUEPOWBZKIYYPAJXCMORYXIPNIBEVKGFPWTHKSSXCFEIUWWCGNCYXAXMGNORJ
RHOGQCDWXGFPWTHHSOZQGLANMGECXWBJPUFOWOQCGLWZWGZITGDYAXMUSHODLSQBMGTHZM
OEHGOSJCAANRBKIZEVKABSHGXAZGVFRVAUJHSSRYXIWIGCXDGLVIZHCPPOARVJQRZWPKYM
SWWSSGTFOQYEJJ
```

Gunakan kunci NEWGEOLOGY sehingga menghasilkan suatu *plaintext*. *Plaintext* ini diduga berasal dari tautan https://pradiptadh.wordpress.com/2020/06/25/history-of-lake-toba/#more-192 dengan isi sebagai berikut:

> Some people may already know the history of Toba Lake located in North Sumatra Province. But for those of you who do not know the story behind the formation of Lake Toba, this time there is some information that will be discussed. Curious as what? Read it down!
>
> Toba Lake is one of the largest lakes in Southeast Asia, and is a volcanic lake in the middle there is an island called Samosir Island. The majority of the population around Toba Lake is Batak tribe.
>
> It is estimated that Toba Lake was formed during an explosion of about 73,000-75,000 years ago which is an eruption Supervolcano (super volcano) that is Mount Toba. Wind-blown volcanic ash has spread to half the earth, from China to South Africa. Even quite surprising because it turns out the spread of the dust to be recorded up to the North Pole. The eruption occurred for one week and the dust burst reached 10 kilometers above sea level. The evidence found also reinforces the notion that the force of the eruption and its ocean waves could annihilate life in Atlantis.
>
> This incident caused mass death followed by the extinction of some species. According to DNA evidence, this eruption also shrank the number of people to about 60% of the total human population of the earth at that time, about 60 million people.
>
> After the eruption, a caldera was formed which then filled with water and became what is now known as Toba Lake. Upward pressure by the magma that has not yet come out causes the emergence of Samosir Island.

> There is also folklore about Toba Lake is, said he said a time when there was a farmer named Toba who went fishing to the river to get fish to eat. Toba gets a big and beautiful fish, but he is surprised that the fish can talk. Apparently the fish is the incarnation of a princess who was cursed for violating the rules of the kingdom. As a thank you for having released her from the curse, the princess was marry Toba. However, there is one promise that has been agreed upon, they should not tell anyone that the princess is a fish.
>
> From the marriage was born a boy named Samosir. Samosir grew into a very handsome and strong boy, but there are habits that amaze everyone. He always feels hungry and never satisfied, all the food rations are always devoured without the rest. Until one day Samosir assigned to deliver food for his father in the field, but the food never came. Toba also approached Samosir and asked where the food for him, but Samosir admitted that the food is already eaten. Toba was very angry and unknowingly breaks his promise by saying “Son of a fish!”
>
> Samosir immediately complained to his mother if he called a Son of a fish, the princess was disappointed because her husband has broken the promise. She cried a lot and told Samosir to run to the high hill. Suddenly very heavy rain came down with a terrible lightning, the water overflowed to drown the entire village. The puddle turned into a lake that is now called Toba Lake, then the island where Samosir shelter is called Samosir Island.
>
> That’s the history of Toba Lake, that’s so unique right? If you visit Toba Lake you can feel the cool atmosphere of the lake accompanied by beautiful views of Samosir Island.

Kami merasa puas saat menemukan pesan tersebut.

Kriptanalisis *Playfair Cipher*

Kami belum berhasil memecahkan *ciphertext* yang terenkripsi dengan *Playfair cipher*. Percobaan yang telah dilakukan adalah analisis statistik secara manual dan menggunakan *simulated annealing*. Analisis statistik secara manual didasari oleh frekuensi bigram dari [1] dan [3], serta frekuensi kuadgram dari [3]. Selain itu, *simulated annealing* didasari oleh frekuensi kemunculan kuadgram yang diperoleh dari [3]. Kesulitan yang ditemukan terdapat pada penyusunan papan Playfair berdasarkan pemetaan bigram yang telah ditemukan. Hasil analisis manual tidak dapat digabungkan dengan *simulated annealing* karena pencarian kunci yang diimplementasikan bersifat *random brute-force*. Akibatnya, pencarian kunci dengan kondisi analisis manual tidak dapat ditemukan atau sangat lambat untuk ditemukan. Skrip *simulated annealing* dapat dilihat pada `algorithm/simulated_annealing_playcipher` di repositori.

Sebagai kelengkapan, berikut adalah *ciphertext* untuk masalah ini:

```
QUKAROQULALPKHBUSHPLIWIDCSCYGRBAUXSHBUSHAGCFHZQCQBWUZCBKECI
VDGFQDGFAEALASHBPKNPOBLHZFXFMBCFBMEALALXDUGWUZHXDFQFTLUSHKN
LVCSANSHXDUGWUVCMOCLCSENMLKFHEQUVFUGZDGDMBZSCZEMZHXDFQFTIDP
WPCGRDQRUQCBLCZGROWVCRVBLHZUQZOSHXKDKFAILKBKGFQKBXDBLBLFBKZ
CHHAFTLUIBKZCHUPQMQCTHPWWOEAIVDTQPBUSHQBWUFTLUSHBKIDPWPCCHX
KABNVROQUBLCZLGBAQBWUFTHOSHBLWCHLRVUMSHBPAHCYWCDIBLLAGLCLCK
LGDIEMZDLRACPZQBWUEMBKLCZEDMWOFTLCZEMLKFEMBLCZPWWOQCBLCZGRO
WYCTKSAMPQUCEBANULEBMSHLELCDGRVAMCHCTHLBAUXDSVCMEOZLGBAEMQB
WUFTLCPBCHEMTHPWWOFTILKBLEPKVCMEZEMLKFEMHEQUVFUGZSCZEMBLKBA
EPUAKLMHAIDLRAMHZUQXYZDLRACUGZDFBBLAKLGDIVCZEPUBUSHONKZFBIG
CHCEBATKZFKTLEIOEMZHLUSHIPFQUMPCUMLGOMAHECUQBLKBXDEAZSCZEME
TACRUGKTDUWVLBTRUXKWCLCUGLENDLBVFUGZSAKWCDIUQUTACNULEALFNQU
KZAHZHZSWCDIOBKXEMZSKBGREHCTNVZSHSCLCVQUHCNKTDDMPCGRIYHDEMM
EUGZDFBZSHZUMWOGRKNQUUMKGALUGDMPCCHLASHBPAHKGRGALUGMEGIDMPC
GRIYHDEMMHPWPCGRIYHDEMMPBUSHVCIVDVDSGRBLHPMLUGHPBLAEKLBKMBQ
CLMMASHDMPCGRIYHDEMMEBQZCBKLUDSCWONKZFBKTRVBLAEKLBKMBQCLMMA
SHBQSHVCPZMERLQEHKGCPUZSAFBFVULERVBLAHPWPCGRIYHDEMPZMQPCABS
HUNRUSERUQBWUZDLRCZEMINWUGRDNMKUGINCZOPBKFTLOPUHEACRUGKTDFT
OUSHBQSHVCPZZHZAFNACBPMDHOSHVCMEZEACRUGKTDEMBLCHFVFTLOPUHEA
CRUGKTDBLFTBLFSBCFBSHBPAHECUGQUKAZDFBBLFSBCFBEMFAILKBDMPCGR
IYHDEMMHPWPCGRIYHDEMMHPWPCGRIYHDEMMPBUSHVCIVDVDSGRBLHPMLUGH
PBLAEKLBKMBQCLMDGALBQZCBKLUDSTCNUECPKKBBLFTLMDVDSKTHSKBAHCL
NYBCFBTRAVPUHEACRUGKTDMBGDDGPZQBWUZDLRACPHZCFBBLFBBLFSBCFBV
CPZZSWDFALCUGURNBCHZFIVDMVCUGINKZFBXDBLFBWZKNPOBLLADSRNZDFB
AZKBZSAHPWPCCHLAQBWUZDLRHSKBSHKTBLACZAIGBGKBTNACMBZSHSLACHB
LFHPKHMPOCDLEUGFBWZFTIOKPHCINCZLGHLKBFTILGHSHKPBGKBSHKTAMPI
ACZCCHMPBLLAZSHSCLUQSKCKCHFVPURGHLBMSHKCIVHALCDBBUMOFTFTHOK
LLUCFABSHEDGDLCVCXGBUMUSHCQBKMBDXGDGTMDQUHFBILTHSKBGRQVPHHL
RAGDDLKBFTHUKPFQSATEKFSHBIACNUEMLAPCFBBLHSRBBMVLUQEHZFLCUGU
MPCSHTKVCZNRUZDFBSHLEIWIUCVBGHPFRGKACUGVCMEUSRUUMXPKBKAQBWU
ZDLRHZGRUQKABUSHKTBMSHEHLFBUWUKZKSFTHOKLLCLIACBIRBBMVLTGACZ
ASAQVDGEADXGDITRBCVFVGRUMBLCZGDGCPUHZALUGBFZAUKPOHSPOACLFGR
FTLCPZZSCZEMMPEHCTRLCNMASHDBBUMOFTLEIOLRWQBGKBFTHOKLIURBCVF
VBLFBWQROPWPUEDGDLUSHXDDIEHEMBMXDIVHMPWQULEZAHCWMRNCHWQRUDS
NDLBSHFTGWABLRZHASCHEMGKEMUMBFLEHUKZFBUPIVHAGCUPSLHPSHKTBMG
RKALRVCQBMDQUHFNBKZCHCNGAPULGVAPCGDDLKBFTLCMEWQRBHNMCKPBORV
BLFBKCSANRRYRGFQNZSHMBETEHIGALUGDCUMRULRZSACNUFTLUPWPCLRUMB
LFSBCFBEMXKPWQCBLLABLAKLGDIVCOHACLUSHDWPCLELUPMLMRVBLHPFTLU
```

```
SHXDKFCHONKZFBLAKHALUGBLWCHLBMSHLEHCLAFBBLHSACNUQCQBWUZDLRH
ERUUQVUSHQPPUEDGDOUEMLMMLDMBFGRBLFHBKLACQBKMBFTHOMUKBKPKGCH
BLLAZSAHPWPCCHLAAZKBVGACCQEAZANXUNKZACUGIGGUMUFRWUZDLAUMOWB
LFBZSHEFHFTUMNRTFUGTKUQZABPLEKXMBQUHZFTLUIBWCHLBLPUBQFTLUSH
XDDIEHEMBMTKRQEMBLKBTRULGISHNBSDKTOMSHTKTFMHQULFZSKBEAABUWT
DIHWCBUGDLRVCMEMVPMBAZARGFQBLFBZSHZCVLMLRAMAKMUALHFFTOUFBRL
BUPDRVUMBKMBSHBPAHEAZANXBLGRHGBUSHDWRVALHFVCPZMOLMHTRUBFLAK
BFTODHAFTKBBLFTKCBTABMDLUSHBKVCMEALTHCSUYLGEDVCMEZDLRAKMWKB
FTHOAZFBDGRVAMKSMLBAMEUGQUACEDMILAACDGRVAMHZQPQUOPBKMWKBBLK
BCZKBHZAZFBDGRVAMAFRGZAMEUGCQKIMEUGZHZATGLRAMASNDMELFGRMBBL
CZLMHMUMWOZHZATGLRAMHECHPZPHHCNZPHHCFTODMWKBCHXDBLQBWUZDLRA
KWCBUKBMKBFMHLRAMHZUQXYZDLRCZEMMPDBFQNXFTLULGBPNVBLLAZSACBA
MPSAZAHFALTHHZFTHUBKCLKNPOACEDMILAACUGIUFTFCIUPQQCXDBFWCUME
ACKLSNDCQBPSHIGGUMUXDZSUMALFALMGAUPOPBFQUGVBLFBEMZSCZEMMPRL
BKHOSHMLGISHBPAHCGWUUQUQBFQCBLAHLRAMHECHZERUUQBFQCBLASOBLRC
HSHUPBFVCMBPWMDQULFUQBLKBUMPMDSNUTRALAMEMBMSHCQRBLULERVBLHZ
AZFBBLQUCZEMKPHLALUGMPZSKBASKLBFIGRGBMKLLCXKFVKBFTOUFBRVUMM
HKZKSZDFBLRVCMSRDBALCKIBLCSYUKBMPANSHDWRVALHFKCSATEKFBLFZZC
BKZHZAUGVCNKOZDSOCLFUGHCMTCHGRBLASWCBUKLBQCQUPBKBLFZDGRVBLK
HBPAZFBKCUGHAMEUGEMLRVCQBWUDGHLUVLRLGBUSHDWRVALHFBLFZMEVULE
BMPOCSBFLEHUKZFBBLKBAEPUAZKBVNLGFQVCQBMUGRBLHZLVCSUIKBGRZDD
KLBSHZGCLHAANLRBLAEPKMBMEGIZDSDEMEHFBPHRVRLFQQUQVEDKLNUSHZH
DWUGZDSDEMEHFBACRLFQQCAQXQMLBABLABLGUIZDSDEMEHFBALIPFQMPSAQ
CQVTNACLUSHCQRNBLZDSDEMEHFBACRLFQQVZAHFALTHHZBLAEGKBLZDSDEM
EHFBBOGRLQNGPUARSHPHVRDZDSLEZEKZFBDCRVRLFQXDBLQVFRGKABSHZHZ
ABUDZDSLEZEKZFBIUGRIPFQWCUMAQVQNDBLFBBLAEPKMBDMPCCHUPRBLCUG
MEVULERVBLKBCZEMLMLRAMHSMDCSBGHPEHLCUGSHMEGIZDSDEMEHFBHFRVR
LFQGRUMQVEHLUSHMUSHBPKCSAROFTHCCKDELMHAOWRUMPSAUSGWLEZEKZFB
GVGRHGTQZNCHUMSURUBLHZAZFBBLZDFBSHDMPCCHLALGZECHMEVDLRAMHZU
QXYZDLRCZDSVCMHXGFQEMHAKSBLKBKHTFUGSHKCIVHALUSHMUSHBPZDWDLV
KBRBRGFQROFTLUSHWFKPCHWCUVLRLEMBQUHPLSFBALUGTNWCHLRVBLKHBPA
ZFBDGRVAMAKFTIDHZFTIDABBLAHGHLBECIVDMTGLRAMHZUQXYZDLRCSIZSH
LTFBMPIZSHLTFBOEKPCHBLFZZDLALMMWAHWFLGDIFTLUSHZXKBHZQDMLLUL
ERVBLFZIGGUMUVCFASHNRSRRUHABLKBMHKZKSQUGRBLHECHFTLUSHZHZABU
LIVCNKMHKSUVLRIZLGOEPOMKRGQUMPBFDSRNXDBLACEDFTHOQDMUBLUPNDL
BSHRGHLUVSHRGUVEMOPNBGRHDLRAMHZUQXYZDLRACUWFAFTOUEMKNGHLBFB
CHZDFBZSHZCVBLHZAZFBOUKTEQTRRVBLFZZCBKKNHKUGGVFTLCPECHSHUXL
EBLKBTFSAVCMONFLVKHPZQBWUZDLRHZSHFTOZKBCHDSZDLTFZWCDWSAUMWC
```

```
BIWCZHMEGIBLCHVCNKMBSHBQSHUMDIBLASBLLASHBPBAQOMUSHBICLXDZSC
HUMLEZASHKIGDHAHURUBLLABLHSRBBMVLBGCLMQKTCHSHTIGKACUGBLLAZS
HSCLNRFGPUBLCZDSHALCWNBUGDLAMLMBZSHSCLCQRBLUSHPKOUAHDGFQBLC
HVCNKPZMEGIGKWQCUGDMACFAKKTCSCQRNDSBOAKPWPCVLFABLHECHOZEMSZ
CZFTICRGALUGGKWQCUGDIAKZKSAZKBVNLGFQBFLAFTODHAFTWQCDFTMBFVX
DBLBOFTGWWCZSLMDVFTNAPUUQBLGRQNHZMEGIQBWUZDLRCZLRLEIVMVZGAC
NUFTHOSHMBFVCHXDBLBLASZSAFKSRVBLHSWCZHGRPUHCNKPUBLASGRBLASP
URGFQMBSHZXFBRVUMBLASWCBUKLBQFTIDPWPCCHCQUKPQSKTKUGDQDIGRBL
KZAZFBGRHQBLFZKCSATEKFFTLUSHBUSHPKOBXRSKBICLUMEHBKCLVNSHHGT
IVCMEDMBFBLCZDSHALCZQMUSHDQWOOUKTEQVCNBCHSHTKUGMEGIEHVCBKQC
WQRNMBKSOPBLKBZSCZGDHMPWQUFRWUBLLAWQTCBKSHBKEHOBBKUMHAUBPWQ
UAKPOKHBTRUBLFSBCFBEHDGAZGRLNLEBMSHLEHCLAFBQBWUZDLRHZSHKTUA
WCDIUQUTRUBLGRARLEBMSHVCMELFGRBLAEPKMBFTLOPMUTACRUGKTDQCLMM
LUGZSCZFBRVUMSHTIPWPCGRIYHDEMMEUGMEGIDMPCGRIYHDEMMHPWPCGRIY
HDEMMPBUSHVCIVDVDSGRBLHPMLUGHPBLAEKLBKMBQCLMMLUGBLFHMLPZMEB
QZCBKOWGSBCFBBLWCKTNAKLBKMBQCLMDGZHHERUMWKBBLHSGDHMZDKBABSH
ZHZAUGVCNKOHZCIVHMUQXYZDLRKHPZMBGDMLDGZAFTOUAHMLUGUQBFHPMPE
CPKEMZSABSHBQSHVCMEMBWCUGCHCQBPSHFRCZBLLABLAHPWPCGRIYHDEMQB
AZKBMQPCAELMZHGVFTHOSHFRCZBLLABLHSRBBMVLBGCLEHUNFVCHSHTKUGB
LLADGRVAMHZUQXYZDLRCZEMMBGDMLDGZAFTHOPMSHBLWCHLALUGBLWCHLAL
LFGRDSZOSHPLHLRAGDDLKBCQBPMDQULFPZZSCZEMUQRVBLAEKLBKMBGRBLC
ZDSHAMLUGFBWZHABLKBLEZAUQBKMBFTOUSHBQSHLELCAMEMRVBLWCHLUMEQ
POABLGFQUMOWZSKSKLBUCHSHNKCKACUGIUHZZHILKBZHGADGFAFTMDIOCHH
AUXPOFTFTGUPWQUAKWCDILEZAFRWUBGKBGRBLHPIGQHRDZHZSCZFBUMZANU
SHZHZATQWCBUKLBQUMBLHZAZFBOUKTEQFTICUQKFCHLABLCHPWPUFTODKPC
HXKABNVBLGRHQUMZHIVMTKBWFSHKMZANXEDACMILRAMHZUQXYZDLRAHPWPC
CHWCUMANSHXDUGWUFTODLMHAOWRUDQWOLCVQNQPWWOUWVLUYLEBLLTFZWCU
MZHIVDHPWWOBLGRHQXKABNVBLGRHQZSACBQZCBKHOALVGCKHZQCLMDAMDWC
BPFTHOSHORIVHAOWRUQUCZLGEDVCZETKGICHKNPOZFIVDMXUCHFTHUDTHZG
DIPVLVGABBLCZPUBLWQDIUWVLRGBUDSNDLBQBWUZDLRACUGZSCBBTMDBALU
SHOWPUFTHUWCHLRVBLKSBKRVNVMLKAOELGDIMEGIBLCSDIUWVLUYLEALKNG
HLBWQTDPWPCDWSADXGDIVMLKAWQROUPSKTINGPUQUKAQBWUZDLRHSCLUQOB
MQDKDPBTRUMBPWWOEHCQBKSHTKUGHABLKBZHGAEHMLKAOULRLBSHBFVDCKH
ZTRRVBLCSDIUWVLTGCKCHMPONDKAIVFUGMPRLHLAMVNLEBMUQXYZDLRAHPM
BLKBTNACBLFTGCAHDIWUTFPHCGACGUWULKVASHOPMBEHCTRLCNHMKLLUSHO
NKZFBUMSHBPAHECUGTKTFVCNFMUDMFQCEBAUXKTHOGRBLKZAZFBGRLNSHZH
ZAUGVCNKOELVHSPOHERUDSZODSKFCHBLFZZCBKZDFBBLFZMEVUSHPKHCKTD
GRVAMHZUQXYZDLRAHXGFQQUBLFHUPRBLCUGBLLAZSFBKHBLKBMBPKBKGUPU
```

```
OPZALCUGZHASCHUMEHHCCLBLFZDGANCHSHNRMKUGEMBLFZMEVULEBMSHVCM
HCKCHUMMUGHLBGVBLFZDCRVBLAHCKHZBLFBZSHEFHFTUMTNACBLACDGRVAM
ACUGCEBATKZDGDAKLVABMDGKACLFGRZDFBBLCHVCNKZSACUIZDLALEILKMS
KBFLUSHZQKLLUSHMDIOCHHAUXPOFTVCQBPWQUKZAHZHBLFTBLCZDKFAGOBC
FBALFAKCBKFTIDABUQQUAKPOKHUYSHUYACBKUQUVLRGZWCTRRVBLCZDKFAO
UPOFTZDFBZSHSCLBKCKSHILPOCZFBRLFGUPBUQCBLFHMLPZMEUGEMFAIDPW
PCGRIYHDEMMHPWPCGRIYHDEMMPBUSHVCIVDVDSGRBLHPMLUGHPBLAEKLBKM
BQCLMMLUGLRFTOZKBCHEMEHCQBKSDONKZFBBLWCKTNAKLBKMBQCLMDGZHHE
RUMWKBBLHSGDHMZDKBABSHZHZAUGVCNKOHZCIVHMUQXYZDLRKHPZMBGDMLD
GZAFTOUAHMLUGUQBFHPMPECPKEMZSCZSHBQSHSHKTLULEALIVDLKBTHPWWO
NRZSCHUMSHBIACNUXDBLEAKTCQBPSHMEXOMLGRGVBLLADGRVAMHZUQXYZDL
RCZEMFLKLTFDGZATRUBWUZSHZKLIGXDIVMALGRFQCMPSABLGRLNLEBMLEIV
IMRUFTFBLUQWWCFTHUVNSHSHIMQCXDUADETKANZDDKSZSHRBHCBPUMWOZSA
SCLACQPHPQUWCOEPOTUSHBQSHIGQHRDZHILKBZHGAFTLUPWPCBLHZLESKQC
FTMUSHUPDIUWVLBQPMSHZCBUMWKBBLHZAZFBOPRBALGRMBMUSHZHZAUGVCN
KPEUQKFCHLABLCHPWPUFTODKPCHDQWOBLGRHQUMZHIVDASHKMEDACMILRAM
HZUQXYZDLRAHPWPCCHWCALUGMEGIDQCVFVDKFTUQAMABFTWQBFDWSAGRHPR
OQPZHWQCDFTDMPCMEGIBLCSDIUWVLTFUGORIVHALUSHQPHPQUWCOEPOUSRU
FTILAHIGURXPRMHAEMCHBLFHPKHMWUAHMALEBMSHHABLKBZHGAEHEHDNGDC
HFTOWSKBFLUSHOWPUZDFBBLFZLELOCLACTENIKLBUSHMDOUPOFTMEGIUQXD
XDIVDAPONZWCXKPQKBGVCQUPUFKSPWPUDGRVAMHZUQXYZDLRHSCLUQOBMQD
KDPTFUGHARVBLCSDIUWVLUGMCPZZSKSHAEMCHTRBLKTIDZGCLZSKSRUBLAK
PORHBGKBLEPKBLFTBLKSPDMPRGRGRVUMPCAEEAAUFTLUSHHGTIEAIVDIWUB
QFBZHHAPZQZWCMKTKDQUQENACRUZQKLLUSHXDKFCHUWVLFNWCKTCHQUAEPU
UQVCUGZSCZFBALVCZNRUCQNURBLAAHXGUVEMLMOPMBAZFBGRDYSHBUSHZHZ
AUGVCNKOELVHSPOCZSHBUSHZQCVQBWUZDLRAHXGFQEMGKHCCLROQUBLFHUP
RBLUSHVFUMUFHZBOSKAUCHBLHZBAQOMUSHTKUGBLFZDMPCCHFTGCWCUGBLK
SPDMPBFODPOEZKCUKAHZGCLBLFZALFARGUMRUZDFBQBWUZDLRAKLVABSDKB
ZHGAFTLUMDLUSHOVLEBLCLLEXRSKBFLUSHBUSHZXKTBFILKBQUKAOPBKUME
HROQUSHNITCUIFTLUPWPQFBBLCHPWPUUMUQQUABSHONKZFBLADSSAZCBUGR
KNQUUMANSHHDEMMEUGMEGIDMPCGRIYHDEMMHPWPCGRIYHDEMMPBUSHVCIVD
VDSGRBLHPMLUGHPBLAEKLBKMBQCLMMASHRGALBQZCBKLCZEAEPUCSGSBCFB
BLWCKTNAKLBKMBQCLMDGZHHERUMWKBBLHSGDHMZDKBABSHZHZAUGVCNKOHZ
CIVHMUQXYZDLRKHPZMBGDMLDGZAFTOUAHMLUGUQBFHPMPECPKEMZSCZSHBQ
SHSHKTLUSHHDEMPZMQACARDBPZZSABBKSTHALCUGZSPWPCXDBLTKHFQBWUZ
DLRHZLEIVDIHKZSAKKPCHAZFBGKLRDWMBMOASVGGKABSHBKROQUZSCZFBRL
BUMCONLRHZAKBKAMQUAHXNPWPOZDKBFBPWQUKZAZKBKCSAFTLUSHBKZSASC
LACZANXQPHPQUWCMEXRMICSRUPHHCLRDMPCCHXKABNVZDLRCZLRLEBKODSH
```

```
AFMPBLLAAZKBWQBFZDPMCVLRDMFQCHCQKPUTRUZDSCZATKBAKMHKKAQCLRO
TMBOBBKGVIGCZSHBUSHKMMICZEMBKCLZQSHMKGRBAILKBECKAFTIWIUHZZH
ILKBZHGAROEMCKWCBUNXUWVLTFUGMPZSCZFBUMZANUSHZHZATQWCBUKLBQU
MBLHZAZFBOUKTEQZSAFUQKFCHLABLCHPWPUQBWUZDLRKSRUSHBIACOWRUQC
BLCZGROWVCUGMEGIDKFTUQAMABFTWQBFGRBLHZAZFBOUKTEQLEZACQNTGIH
CTQKHRLMEIVMASHMESAUMSAFTOZKBCHBLCZPOFTHPLEIVHMPWQUHFUNGIQC
QVKMMIHZBLKBKHXDIVDHLXFZWCQUFBPMKLHOUQXYZDLRKHLCBKUQRVALFAF
TVNLGFQKTFZWCCETKGIQCQPHPQUMEGIBLCSDIUWVLTGPWPCDXGDDARUBLAC
XRMIKHBUUWXPAKHZWQBCLABLKBCHEDKZAFFTIGXDIVHALABLCZLGBABLACX
RMICZEMMPDCRYRGFQGVVLHCBLLAQUGVBLKBCHEDKZAFVCMQPDMPBFHOUQXY
ZDLRAHQUHFGCPUBLAEGRACXRMIACUGZDFBZSHZCVBLLABLCZPOFTLAKSKTU
MKGBMSHDWTDIUHZHPUBMDQUHFTKUGMBBKUASHOWRUSHBIFTLCUGUMPCBLKS
PDMPUQBOLEGATRBLKTIDZGCLZSACRHUMKGRLBGKBOPRULBLEBQSHEAIVDIW
UUGACLUSHBUSHONKZFBDMPCCHLASHUXLRLEIUACGCTDDMPCFTIDCTHLCHLM
WCLCUGMEGIZDLRACPZQBWUBKLCZEDMWOTHCKFCZHUSFNUWWOBLHPRLSABLC
HVCNKOEFTUQUVCFFZWCROFLKLTFUGZDFBZSACPECHQCBLAHPWPCGRIYHDEM
MEBLPOAHPWPCGRIYHDEMMHPWPCGRIYHDEMMPBUSHVCIVDVDSGRBLHPMLUGH
PBLAEKLBKMBQCLMDGALBQZCBKLCAMEMUMGSBCFBGRBLHPMLUGBLWCKTNAKL
BKMBQCLMMASHBGKBFBXLWCZSACNULEIUHZBMQCKTEMFTFBXLWCZSACNUKCB
GLTKBHZRVBLCHVCNKOZSHBUSHWFLVHSPOKHBUSHAZFBGRQGWCUGQBWUZDLR
AHXGFQROQUBLFHUPRBHOSHTNACBLCHUQDMFQKBFTOUEMHCCLBLFZDGANCHS
HNRMIPWPCCHUMZHCZSHBLKBBLFZDWTDGCGRLCFNBLGRIQPDMPUQBORBMLKA
ILKBDWSTCHSHBIKLUXEMSHILKBXDBLVCBATKUGXDBFTRRLUVEMLMDMGFWCZ
HBLKSPWPUEDGDOUEMHCCLFTIUASKLBFIWHCCLBLFZMLGISHNRQPTFRHKBFT
LCIVHMAZFBQCBLASMEUNWCUGLRFTOUKSNAPUSHTKUGZCMRBLBKCHFVMHQUL
NSHBUSHZXKBFHPDFQUMTRNXSHNTRUZSHZRLIVIVDMPCCHEMGKZSCZKBAHLX
GRLFUGMBGDDLCLSHKXBKRVNVBKODSHAFMBSHZQKLOUAKWCDIUQUTRNZGKBG
RBLCHKTFIUPRBLCUGBLFZLELCUNFTMQKTFBUAQCKGUQGQMLPZMOCLHZMULE
BMSHDWTDHUHZKZFBKNPOLMHMGIHZFTLUSHVGKLILKBGRLRFTOUUPBASHNBL
VCBQPRGRLFQMDHCTGABBABPFTLULEBMSHVCMEIPFQOHCTHLBANUSHBUSHXQ
RUBLAKQCKGUQRUROQUBLASWCBUKLTFUGQUCSANSHVLDVFVPZMBFVCHNZLRF
TOULAEDCHLRFTHUPKHOKCSAUMMCUGZCMRCQBPUQXYZDLRAEPKMBFTWUMASH
TFTKZATFUGMLMBCLMWACUGUQZOUQXYZDLRAHFVLMQUHDQULNLPKHBUSHDWK
YKGTFUGZSCHGIUQUALEFQHERUDMPCCHEMGKZSCZKBACMHKZKSCQBPSHVCME
OZLGBAEMQBWUEMBKLCZEDMWOFTILKBLEPKVCMEZEMLKFEMHEQUXGBLKMSKB
FILWUAZKBBLLACFGRHQMPUFLVKHBUMUSHCQBKMBFTOUFBRVUMBLCHVCNKZS
WCZHUMMQFBLUSHRGHLBLHZCVBLAKQCKGUQBUSHOPRBALGRFTLUSHEHCTRLC
NHMUQXYZDLRCZLRLGRGALUGBKCLZDLAVCOZKPRVBABRQPRGRLFQMDHCTGAB
```

```
BABPBLFBSHMEGIUMBLCHVCNKMHABSALEZABLAKQCKGRGXDIVDHLXFZWCZDL
AAZKBWQCUFTNAPULRTRRVBLCHVCNKMEBQZCBKOUCZGDGTMUMKNUXDBLLRCQ
TKIVMASHDQDIGRBLCZPUDIBLFBSHMEGIHAAVHSLTKHALMEHGANCQKPKCRYU
QAMLXCZLRDSRUZHKHFQQBWUZDLRKHXDIVDLQUWCTKUGXKHXHSKBEMQVHCKT
HZRMPMZHPZPHQUEMSHMQPCKHBULGOZFVBLFHPWWOOUKTEQUMPCXPNVROQUL
GVLUGLFZALGVASHDWKYKGTFUGUQVUSHIPFQPZMPBGCLLRKCKRKPCHCVFVNZ
LGPZZHTXFTBMQUBLKHBPDSTDHCBPFTIGBLKMSKBFLULERVBLFZMBTOTHCHM
WKBLABKHZUROQFTOULRLBSHZSWDARSHQPHPQUWCMQHKKAQCKMMICZLGEDQB
WUZDLRHSCLRHRVBAUQKYEKLVCSRUQCSHNUIBMCALUGEHCQBKDMFQZSCSSKB
FILKBFZHZDGANCHROBLAHGIQCBLAKQCKGBQLAROFTOUEMQUKAOPBKLMLXCS
IZSHLTFBOZSHBKLVHPSHKUHKLUSHIPFQPZMPFGTDDMKGQWMEGIWQTCBKXDB
LSAFTLUMDILKBZDLALEILKMSKBFLCUGMEGIGDMWFZWCOPBKBLFTAZKBVNLG
FQGRBLCZPUDIDWSAXDBLSAUMQVECBLKBMQLMCKFZWCZSLMHTASZXGKACUGQ
BWUZDLRCZEMXDIVDGFQFTOUFBUVLRIZLGVLUGBLKHUXCHIGFQVCZSAHOULR
LHBKLAZSWUFTHOMIFBOWRNTRBMUQXYZDLRHZXDKFCHMBKSOPBLKBVCMEHMS
UGIHCBUMUSHEAEMUVSHBQSHLELCKRTKZFILKBZHGAGREHCTRLCNDADMBLHZ
ZSCZFBUTAEPUABSHDMPCGRIYHDEMMEUGMEGIDMPCGRIYHDEMMHPWPCGRIYH
DEMMPBUSHVCIVDVDSGRBLHPMLUGHPBLAEKLBKMBQCLMMASHHDEMMEBQZCBK
OWGSBCFBQCLMDLKBABSHECPKHZALNUBLWCTRRVBLFZWCFQONKZFBHPECPKK
BNZECTKPHUNWUBLFBBLCZDKFAOUPOFTRUBABKLCDCBPACUGVCPZMPXUABED
CHMPRUBATIZXBKUASHLULEBMSHFRCZUQUVLERVUMOWLAKGBPBMSHUWTDGUM
UDQUMBLCZCHIGFQLALMHTRUZSHSCLUQSKCKACUGOTMBDQUMZHABSHWQRBQY
BCFBFTOUSHBQSHZCBUGRZSAFBFZOUQXYZDLRACUGZSHZUMWOMBGDDVLRIBF
LACUGEAKTFTODWCDIUQBMRLNTRUPKQUMHPXSKBPLELCITACGWEHFBORURQP
BUSHKGBKFTLUSHZXKBHEUPNDLBGRXDBLUMFQMEUGZHUTAEPUHSKBBLFBZSC
ZEMCQUKCHUMORUMBUSHBKILMUZSSCMEUGLCUFCBBUGDZSCHUPXRSKIWOWYU
HCCLNZINLPPLCKSUFTOUGDSHMV
```

Kriptanalisis *Hill Cipher*

Kami belum berhasil memecahkan *ciphertext* yang terenkripsi dengan *Hill cipher*. Percobaan yang adalah dengan melakukan ***known-plaintext attack*** dari pasangan plaintext dan ciphertext yang diketahui. Proses yang dilakukan mula-mula mengubah plaintext yang diketahui menjadi angka (A = 0, B= 1, .. Z= 26).

Plaintext “HelloCaptainHaddock” menjadi [7, 4, 11, 11, 14, 2, 0, 15, 19, 0, 8, 13, 7, 0, 3, 3, 14, 2, 10] dan Ciphertext “TFJOXUPOUXYTTRDSXQM” menjadi [19, 5, 9, 14, 23, 20, 15, 14, 20, 23, 24, 19, 19, 17, 3, 18, 23, 16, 12].

Plaintext “Tintin” menjadi [19 8 13 19 8 13] dan Ciphertext “HDWHBB” menjadi [7 3 22 7 1 1].

Kemudian dari kedua pasangan plaintext-ciphertext dilakukan percobaan pencarian kunci matriks. Diketahui enkripsi dilakukan setiap 3 karakter sehingga dapat diketahui bahwa ukuran dari matriks kunci adalah 3x3. Rumus yang digunakan untuk mencari kunci adalah $K = CP^{-1}$, dengan K adalah matriks kunci, C adalah matriks ciphertext dan P^{-1} adalah inverse matriks dari matriks plaintext, yang masing-masing berukuran 3x3. Percobaan yang dilakukan adalah sebagai berikut:

Percobaan 1

P = [7, 4, 11], C = [19, 5, 9]

P = [11, 14, 2], C = [14, 23, 20]

P = [0, 15, 19], C = [15, 14, 20]

P^{-1} = [[16, 5, 7],[23,11,5],[2,11,16]], didapatkan K = [[7,11,0],[4,14,15],[11,2,19]], ketika kunci tersebut digunakan untuk mendeskripsi tidak menunjukkan hasil yang diharapkan

Percobaan 2

P = [0, 8, 13], C = [23, 24, 19]

P = [7, 0, 3], C = [19, 17, 3]

P = [3, 14, 2], C = [18, 23, 16]

P^{-1} , det(P) tidak memiliki modulo invers dengan 26

Percobaan 3

P = [4, 11, 11], C = [5, 9, 14]

P = [14, 2, 0], C =[23, 20, 15]

P = [15, 19, 0], C=[14, 20, 23]

P^{-1} , det(P) tidak memiliki modulo invers dengan 26

Percobaan 4

P = [11, 11, 14], C = [9, 14, 23]

P = [2, 0, 15], C = [20, 15, 14]

P = [19, 0, 8], C = [20, 23, 24]
P^{-1} , det(P) tidak memiliki modulo invers dengan 26

Percobaan 5

P = [11, 14, 2], C = [14, 23, 20]

P = [0, 15, 19], C = [15, 14, 20]

P = [0, 8, 13], C = [23, 24, 19
P^{-1} , det(P) tidak memiliki modulo invers dengan 26

Percobaan 6

P = [14, 2, 0], C= [23, 20, 15]

P = [15, 19, 0], C = [14, 20, 23]

P = [8, 13, 7], C = [24, 19, 19]

P^{-1} , det(P) tidak memiliki modulo invers dengan 26

Percobaan 7

P = [2, 0, 15], C = [20, 15, 14]

P = [19, 0, 8], C = [20, 23, 24]

P = [13, 7, 0], C = [19, 19, 17]

P^{-1} = [[2, 0, 5],[19,13,20],[0,15,0]], didapatkan K = [[4,25,6],[25,12,15],[16,21,4]], ketika kunci tersebut digunakan untuk mendeskripsi tidak menunjukkan hasil yang diharapkan. Skrip yang digunakan dalam percobaan terdapat pada repository algorithm/hillCipherKriptaAnalisis.py.

Sebagai kelengkapan, berikut adalah *ciphertext* untuk masalah ini:

TFJOXUPOUXYTTRDSXQMONIYPEUFJDQUBGIMOCJQTNBEHCZEKROVBNTWLMVXMO
WZLUCHOXYGSKBQGUAOBQZKIXYJIETSWVXHVKCUAOTOFYIZAKJGXKAWGQTRVFDZA
JNQDUIWZCMYWNFIUPYMCZXIAKYUCQIAZPIQMGAMGUAKKKHMWKDUXQDUAAKYOWE
HLJPWYFKXSARBLLHGAJKTQNTRTPWSCIZASCGSLKVDHTUZSWBNBTJGYYUPQMFSYZ
AUTOQCDNGQMFSRLRTUWEMKADIVYLTJKFHLKJUWTSSHMHJFGTRIBYIDAHQEPMPIQ
CROWDYRYZNSPNOJHQVKKTOCBPNFAJNLYJZNVBAYJWRGMCHJPWBDHHTPOXSIJVQ
WDMSIGMTRVEVXDILKVAYTNUNJXEZLAPGYETRVZNVHSVWLGICDXQFOALDVPASUSYX
PFHUWTILUQHTJQVGWFSPAEKBRBNIINYKHNTNUKJVDHVLXQKUZNVQXUOZZOJZYNPI
VYSVFVTZMMUUPWTGHRIOWCBKZYAGUMRCKHIQZSIGISPGBXPYXMOAWGAGHQVUW
TEIGPBMOMBWIOPQEVKMRQATNBMILHHLVUXGMOUWTZCLBKGWIJHFRNGOSCMUHD
WHBB

Daftar Pustaka

[1] Keating, Barry. 2008. “Letter Frequencies in the English Language,” diperoleh dari https://www3.nd.edu/~busiforc/handouts/cryptography/Letter%20Frequencies.html pada tanggal 1 Februari 2023, pukul 8.58 WIB

[2] 101Computing.net. 2019. “Enigma Machine - Beta version,” diperoleh dari https://www.101computing.net/enigma/enigma-instructions.html pada tanggal 3 Februari 2023, pukul 21.04 WIB

[3] Lyons, James. 2020. “English Letter Frequencies.” *Practical Cryptography*, diperoleh dari http://practicalcryptography.com/cryptanalysis/letter-frequencies-various-languages/english-letter-frequencies/ pada tanggal 3 Februari 2023, pukul 21.13 WIB