Institut Teknologi Bandung

Program Studi Matematika

MA4151 - Kriptografi

Laporan Tugas 2

Kriptanalisis Sandi Vigenere untuk Teks Berbahasa Indonesia

Kelompok 1

Jihan Navitri	10118010
Sulthan Bimo Rizqullah	10119028
Ibrahim Naufal Mangaraja	10119031

Dosen Pengampu

Prof. Edy Tri Baskoro, M.Sc., Ph.D.

October 9, 2022

1 Distribusi Huruf dalam Bahasa Indonesia

Dari hasil pemrograman menghitung frekuensi kemunculan huruf dalam bahasa Indonesia, didapatkan hasil distribusi sebagai berikut.

A	19.61109506762196
B	2.849915220467118
C	0.6474220084763399
D	3.4252221933618676
E	7.953626966828697
F	0.19150010567192016
G	4.324966160785877
H	2.552904525823475
I	7.57111074901225
J	0.8890961098213581
K	5.231002044065997
L	3.420866251615328
M	4.682476046353674
N	10.069646668591885
O	1.5581364958545956
P	2.9165449960715857
Q	0.004678604098134528
R	4.976582779832958
S	4.087486670011599
T	5.063701614763739
U	5.589479916688581
V	0.056949905056603044
W	0.43236755113794945
X	0.014197143470201325
Y	1.8440153393681948
Z	0.03500886514811009

Figure 1: Distribusi Huruf dalam Bahasa Indonesia (%)

2 Sandi Geser dan Vigenere

Sandi geser mengenkripsi dengan cara menggeser setiap huruf dalam kata asal sejumlah langkah yang hanya diketahui oleh pengirim dan penerima yang diberi kewenangan *(authorized)*. Sandi ini disebut sebagai sandi monoalfabet karena untuk setiap kunci yang dipilih maka setiap karakter dipetakan pada satu karakter yang tunggal. Sementara sandi Vigenere merupakan suatu sandi polialfabet. kita dapat mengasosiasikan kunci K dalam sandi Vigenere dengan string panjang m, yang disebut sebagai katakunci. Sandi Vigenere mengenkripsi m huruf sekaligus dengan menggunakan kata kunci K.

3 Kriptanalisis Sandi Vigenere

3.1 Kata asal

Surainya yang indah menggelombang perlahan bagai tarian Kuda itu menderap melebihi kecepatan angin Lihat Ada kuda Kuda Kuda Dari atas bukit anak anak itu bisa melihat bahwa cahaya putih yang meluncur di padang rumput membentang itu adalah seekor kuda Itu kuda putih Mereka memerhatikan bagaimana kuda itu berlari dengan indah Di mata anak anak yang setiap hari pergi menggembalakan kambing bercakap dengan daun daun dan sengaja mendengarkan sungai bernyanyi laju kuda itu bisa dicermati begitu rupa seolah kuda itu bergerak begitu lamban bagaikan tarian yang terjaga Mereka memerhatikan kuda itu dengan bertanya tanya Kuda bukanlah binatang yang asing bagi mereka begitu pula kuda yang berlari lepas di padang padang terbuka Namun laju kuda ini bukanlah laju kuda biasa bukan hanya karena lebih cepat namun mengapakah seekor kuda harus berlari secepat itu Kitab Omong Kosong Kuda itu berlari ke arah desa mereka Apakah orang orang akan me nangkapnya Satya melihat kesibukan yang luar biasa di gerbang desa Orang orang berlarian membawa tombak bambu runcing pentungan kayu bahkan juga alu Apakah yang akan mereka lakukan Apakah mereka akan membunuh kuda gagah perkasa yang melaju dengan kecepatan angin itu Tapi untuk apa membunuh kuda indah yang tidak bersalah Apa yang mereka lakukan Satya bertanya Tak ada satu anak pun bisa menjawabnya Seingat Satya tidak ada sesuatu yang istimewa belakangan ini yang harus mereka perhatikan se perti misalnya jika orang orang desa harus melakukan upacara Kalau ada sesuatu yang harus diketahuinya anak anak akan mendapat penjelasan dari orangtuanya

3.2 Kunci dan Hasil Enkripsi

Dipilih kata kunci k = *inikatakunci* dengan hasil enkripsi sebagai berikut:

y = ahzki gyksn poqal khfex atgtw zjknz polyc piajk gtidu ekivx cnabt egrpl meizm xlovv jqsri bcxpk nnpiv tqxlb hknnf ishlk kndke hfiln zsama cvhmq bnvkk tnkev vcjva kmxls bnvji uekrv arulc xcgqr ytnqg rncvp cbdbp kxnpo zhuzu mmogo gvbnv qimuk xnnip fmokh ruoqc qbhse dtrzo gkpur zoktm ogrtp igqua gbkan kuiai uuwas nhdmz yibiw exanp qvqir dbmkn ncvix ixady khtta mgqkp aabcc gzovu onzgo gocti xixkt mlcai jmekk ktpny aiivq ienwa ehqcv arvqa cawya fmvnz qakkk hfwvo nqlek niuaa qtnre kndkc gwjqf inive bgnvq jrost nrejn umwyi rkndk cgwjm eoort kbvri qbhtk muaxv niiqx ixttr suaai vtbor caquz gzmxi wefeb bnvqs nvuuw asnhf mvtix bxrdu aaizg ixytk exndc snvva abshn vivtg knzac caiji tqwek euuog oqgcz ueauo qcgia olekl klvnm xnani kpkxn poxnl knzto lowsi aiwug lkdhm clnqx iuuuu anipy itudu nuoki anjek tnrua aisnz ontrv yokpk rxktg awoao mvtiz adarm rgswe sedth klhuj metkr bsowr ribvb ekbtk vbowv tsysh nqlxw livbe bxrvu eksmn zkhwe cuzgz mxikp tkkbb tivtw baggk enpum aixgd azhlc aiggk mxlsb nvzsr asbnk khlcv oyckr uikmn fqorz laggn yfcwz nvqok axaog ztnzs agmog oceig wwbtk luzdc zecxc bnqjr pbcao kndai oocps nvtuz akfhc xixir ytnqu xcvur zoktl kehmi vnxkk thwye gsiei uagmo gowvc usedt gkanj xmesk stykh tomtn redxn quamm krxkt tnkht kvqgc daiie hgwsi cibmx mloaw pshlk igdkb lcvog qnadb olfct iuiza raxaz gzmxi vaduu uauib lilek tkhlc bixin akskn hcvix xenui cuzgv rnekb gykmr kvonb camyk nvfis nlksx seugw giaos smiwy jcjmy iuagg khekv qlixg aabof omzrs kpxrr ugksi aaopx rdczk aiyvi aciuu btivt wbagg nyfcp ieccm xlkeh mivec zavab uxcti hinal econv cgnvq htrem qksmg irubn iunpi snvkk tkkhz gvlnx ktiex drnia nvnak ibiec vogck nrab

3.3 Mencari Panjang Kunci dengan Indeks Koinsidensi

Misalkan $\mathbf{x} = x_1x_2....x_n$ suatu string dengan n karakter alfabet. Indeks koinsidensi dari $\mathbf{x}$, dinotasikan $I_c(x)$, didefinisikan sebagai peluang dua buah unsur acak di $\mathbf{x}$ identik.
Notasikan frekuensi munculnya $A, B, \ldots, Z$ di $\mathbf{x}$ dengan $f_0, f_1, \ldots, f_{25}$.

Kita dapat memilih dua elemen di $\mathbf{x}$ sebanyak $\binom{n}{2}$ cara.
Untuk setiap $i, 0 \leq i \leq 25$ ada $\binom{f_i}{2}$ cara agar kedua elemen menjadi i. Maka diperoleh formula

$$I_c = \frac{\sum_{i=0}^{n} \binom{f_i}{2}}{\binom{n}{2}} = \frac{\sum_{i=0}^{n} f_i(f_i - 1)}{n(n-1)}$$

Misalkan x adalah suatu string dari teks Bahasa Indonesia. Notasikan ekspektasi kemungkinan munculnya alfabet a,b....,z pada Figure 1 dengan $p_0, p_1, \ldots, p_{25}$. maka

$$I_c \approx \sum_{i=0}^{n} p_i^2 = 0.082$$

Sekarang misalkan kata sandi $y = y_1, y_2, \ldots, y_n$ yang dikonstruksi dengan sandi vigenere.
Definisikan m buah substring dari $\mathbf{y}$ dimana

$$\mathbf{y}_1 = y_1 y_{m+1} y_{2m+1} \cdots$$

$$\mathbf{y}_2 = y_2 y_{m+2} y_{2m+2} \cdots$$

$$\mathbf{y}_3 = y_3 y_{m+3} y_{2m+3} \cdots$$

$$\vdots$$

$$\mathbf{y}_m = y_m y_{2m} y_{3m} \cdots$$

Jika $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_m$ memang dikonstruksi dengan cara seperti ini dan m memanglah panjang kunci, maka setiap $I_c(\mathbf{y}_i) \approx 0.827$. Jika m bukan panjang kunci, maka substring $\mathbf{y}_i$ akan terlihat lebih random karena $\mathbf{y}_i$ diperoleh dari geser dengan kunci yang berbeda.
Perhatikan suatu string acak akan memiliki

$$I_c \approx 26 \left(\frac{1}{26}\right)^2 = \frac{1}{26} = 0.038$$

Dua nilai 0.082 dan 0.038 cukup jauh sehingga seringkali dapat ditentukan panjang kunci yang benar dengan metode ini.
Jika kita terapkan metode ini kepada kata sandi pada bagian 3.2 diperoleh panjang kunci $m = 12$.

3.4 Mencari Kata Kunci

Diasumsikan kita sudah mendapatkan panjang kunci m yang benar, bagaimana cara menentukan kunci $k = (k_1, k_2, \ldots, k_m)$?. Misal $1 \leq i \leq m$ dan misal $f_0, \ldots, f_{25}$ menyatakan frekuensi dari $A, B, \ldots, Z$ pada string $\mathbf{y}_i$. Misalkan juga $n' = n/m$ menyatakan panjang dari string $\mathbf{y_i}$. Maka peluang disktribusi dari 26 huruf di $\mathbf{y}_i$ adalah

$$\frac{f_0}{n'}, \ldots, \frac{f_{25}}{n'}.$$

Ingat bahwa substring $\mathbf{y}_i$ diperoleh dengan menggeser subset teks asal sejauh k_i. Sehingga diharapkan peluang distribusi

$$\frac{f_{k_i}}{n'}, \ldots, \frac{f_{25+k_i}}{n'}.$$

akan "dekat" dengan distribusi peluang $p_1, \ldots, p_{25}$, dimana subskrip pada formula diatas dievaluasi pada modulo 26.
Misalkan $\mathbf{x} = x_1 x_2 .... x_n$ dan $\mathbf{y} = y_1 y_2 .... y_n$ adalah string n dan n' karakter berturut-turut. Indeks koinsiden mutual dari $\mathbf{x}$ dan $\mathbf{y}$, dinotasikan dengan $MI_c(x, y)$, adalah peluang bahwa elemen random dari $\mathbf{x}$ identikal dengan elemen random dari $\mathbf{y}$. jika kita nyatakan frekuensi

dari A, B, C, ..., Z dalam **x** dan **y** dengan $f_0, f_1, ..., f_{25}$ dan $f'_0, f'_1, ..., f'_{25}$ berturut-turut, maka $MI_c(x, y)$ adalah:

$$MI_c(x, y) = \frac{\sum_{i=0}^{25} f_i f'_i}{nn'}$$

Misal $0 \leq g \leq 25$, dan definisikan

$$M_g = \sum_{i=0}^{25} \frac{p_i f_{i+g}}{n'}$$

jika $g = k_i$ maka

$$M_g \approx \sum p_i^2 = 0.082$$

jika $g \neq k_i$, maka M_g akan jauh lebih kecil dibandingkan 0.082.

Dengan menggunakan metode diatas didapatkan $k = inikatakunci$ yang memang merupakan kata kunci yang digunakan untuk mengenkripsi kata asal pada bagian 3.1 dan 3.2

4 Lampiran

4.1 Pemrograman Enkripsi Sandi Vigenere

```
from string import ascii_lowercase as letter
import string
import random

# fungsi yang mengubah string menjadi angka
def letter_to_number(l):
    l_new = []
    for i in l:
        l_new.append((ord(i) - 97))
    return l_new

# fungsi yang mengubah angka menjadi string
def number_to_letter(a):
    letter = []
    for i in a:
        letter.append(chr(int(i) + 97))
    return "".join(letter)

# fungsi yang mengekripsi kata asal x dengan enkripsi vigenere dengan kunci k
def e_vigenere(x, k):
    n = len(k)
    key_list = letter_to_number(k)
    plain_list = letter_to_number(x)
    for i in range(len(plain_list)):
        plain_list[i] = (plain_list[i] + key_list[i % n]) % 26
    return number_to_letter(plain_list)

# fungsi yang inputnya string a dengan output string a tanpa spasi dan huruf kecil
def clean(a):
    return a.replace(" ", "").lower()

# kata asal x
x = input(str("Plaintext: "))
#membersihkan kata asal x
x = clean(x)

#generate kata kunci random sepanjang N
N = 7
k = "".join(random.choices(string.ascii_lowercase, k=N))
k = "inikatakunci"
# mengenkripsi x dengan kunci k
y = e_vigenere(x,k)
b = list(y)
c = []
for i in range(len(b)):
    c.append(b[i])
    if (i+1) % 5 == 0:
        c.append(" ")
```

```
print("".join(c))

```

4.2 Pemrograman Dekripsi Sandi Vigenere

4.2.1 Pemrograman untuk Mencari Panjang Kata Kunci dengan Indeks Koinsidensi

```
#fungsi yang menerima input string y dengan output array yang berisi frekuensi huruf di string
     y
def count(y):
    fi = []
    for i in letter:
        n = 0
        for j in y:
            if i == j:
                n += 1
        fi.append([i, n])
    return fi

#fungsi yang menerima input array y dan integer m dengan output
def block(y, m):
    temp_y = []
    for j in range(m):
        temp_y.append([])
    for i in range(len(y)):
        temp_y[i % m].append(str(y[i]))
    return temp_y

#fungsi dengan input array y, integer m dengan output float indeks koinsidensi
def indeks_koinsidensi(y, m):  # input frekuensi tiap huruf di block ciperteks dan output
    rata2 index koinsidensi tiap block
    out = 0
    Hasil = block(y, m)
    for j in range(m):
        n = 0
        sum = 0
        con_arr = count(Hasil[j])
        for i in range(len(con_arr)):
            sum += (int(con_arr[i][1]) * (int(con_arr[i][1]) - 1))
            n += int(con_arr[i][1])
        out += sum / (n * (n - 1))
    return out / m

#fungsi yang menerima string y dengan output tebakan panjang kunci k yang digunakan
def len_k(y):
    a = True
    m = 1
    while a:  # mecari panjang k dengan indeks koinsidensi
        ic = indeks_koinsidensi(y, m)
        if ic >= Ic * 0.85:
            a = False
        else:
            m += 1
    return m

#fungsi dengan input string y, integer g, dan array pi (peluang tiap huruf pada bahasa
    Indonesia)
# dengan output nilai mutual koinsidensi string y yang digeser sejauh g terhadap pi
def mutual_koinsiden(y, g, pi):
    sum = 0
    n = 0
    for i in range(26):
        sum += float(pi[i][1]) * int(y[(i + g) % 26][1]) / 100
        n += int(y[i][1])
    return sum / n

#tabel distribusi bahasa indonesia
pi = [['a', '19.61109506'], ['b', '2.849915220'], ['c', '0.647422008'], ['d', '3.425222193'],
    ['e', '7.953626966'],
      ['f', '0.191500105'], ['g', '4.324966160'], ['h', '2.552904525'], ['i', '7.571110749'],
    ['j', '0.889096109'],
      ['k', '5.231002044'], ['l', '3.420866251'], ['m', '4.682476046'], ['n', '10.06964666'],
    ['o', '1.558136495'],
      ['p', '2.916544996'], ['q', '0.004678604'], ['r', '4.976582779'], ['s', '4.087486670'],
    ['t', '5.063701614'],
```

```
      ['u', '5.589479916'], ['v', '0.056949905'], ['w', '0.432367551'], ['x', '0.014197143'], ['y', '1.844015339'],
      ['z', '0.035008865']]
Ic = 0.08267724949016449  # indeks koinsiden Bahasa Indonesia

#menebak panjang kunci k
m = len_k(y)

# menghitung frekuensi tiap huruf di y1,y2,...,ym
Y = []
temp = block(y, m)
for i in temp:
    Y.append(i)
fi = []
for i in Y:
    fi.append(count(i))

Hasil = []
hasil = []

# menghitung mutual koinsidensi untuk setiap y1,y2,...ym
for g in range(26):
    for i in range(m):
        Hasil.append([i, g, mutual_koinsiden(fi[i], g, pi)])

```

4.2.2 Pemrograman Mencari Kata Kunci

```
#fungsi dengan input matriks y dengan output matriks y yang terurut berdasarkan kolom pertama
def Sort(y):
    return sorted(y, key=lambda tup: tup[0])
key = number_to_letter(hasil)
print("kunci yang digunakan:", k)
print("kunci yang ditebak:", key)

```

4.2.3 Pemrograman Mendekripsi Sandi Vigenere

```
#fungsi yang mendekripsi kata cypher y dengan kunci k
def d_vigenere(y, k):
    n = len(k)
    key_list = letter_to_number(k)
    cypher_list = letter_to_number(y)
    for i in range(len(cypher_list)):
        cypher_list[i] = (cypher_list[i] - key_list[i % n]) % 26
    return number_to_letter(cypher_list)
print("kata asal:", d_vigenere(y, key))

```