Vigenere Cipher 3 Dimensi

Puanta Della Maharani - 13507135
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
if17135@students.if.itb.ac.id

Algoritma enkripsi klasik masih sering digunakan saat ini walaupun telah tersedia algoritma-algoritma yang lebih modern. Alasannya, algoritma klasik mudah digunakan secara manual dan jarang membutuhkan program khusus untuk memecahkan kodenya.

Algoritma vigenere cipher merupakan algoritma klasik yang sering digunakan. Algoritma ini menggunakan tabel yang berisi pemecahan masing-masing huruf sehingga pengguna cukup mencari pasangan huruf yang dibutuhkan saja. Akan tetapi, dengan analisis frekuensi kemunculan huruf dan metode Kasiski, algoritma ini menjadi sangat mudah untuk dipecahkan.

Algoritma ini membutuhkan modifikasi dalam penggunaan metodenya. Tabel sederhana seharusnya diubah menjadi tabel 3 dimensi yang memakai 2 buah kata kunci agar tidak mudah dipecahkan.

Pada makalah ini, akan dijelaskan tentang penggunaan algoritma vigenere serta algoritma caesar yang melengkapi metodenya, analisis pemecahan vigenere, dan metode vigenere dengan tabel 3 dimensi dengan 2 kata kunci.

Kata kunci: algoritma vigenere, algoritma caesar, tabel 3 dimensi, 2 kata kunci.

I. PENDAHULUAN

Dalam berkomunikasi, tidak semua informasi dapat diketahui oleh orang selain pengirim dan penerima informasi. Agar informasi tidak langsung diketahui, pesan yang dikirim harus dienkripsi terlebih dahulu.

Banyak sekali algoritma enkripsi yang bisa dipergunakan. Algoritma klasik yang cukup popular adalah algoritma substitusi. Dilihat dari jumlah alphabet yang disubstitusi, algoritma substitusi terbagi menjadi 2 macam, yaitu *polyalphabetic substitution* dan *monoalphabetic substitution*.

Polyalphabetic substitution adalah teknik pengenkripsian suatu data dimana satu alphabet dapat dienkripsi menjadi alfabet yang berbeda walaupun menggunakan teknik eknripsi yang berbeda. Hal ini sangat baik dimana dapat mengurangi kemungkinan pemecahan informasi yang kita enkrip menggunakan analisis frekuensi. Contoh dari *polyaphabetic substitution* adalah Vigere Cipher. *Monoalphabetic substitution* adalah teknik pengenkripsian suatu informasi dimana sebuah huruf atau alphabet akan menjadi alphabet yang sama jika menggunakan teknik enkripsi yang sama. Contoh *monoalphabetic substitution* yang sangat terkenal adalah teknik Caesar Cipher.

II. LANDASAN TEORI

A. *Caesar Cipher*

Dalam kriptografi, sandi Caesar, atau sandi geser, kode Caesar atau Geseran Caesar adalah salah satu teknik enkripsi paling sederhana dan paling terkenal. Sandi ini termasuk sandi substitusi dimana setiap huruf pada teks terang (plaintext) digantikan oleh huruf lain yang memiliki selisih posisi tertentu dalam alfabet. Misalnya, jika menggunakan geseran 3, W akan menjadi Z, I menjadi L, dan K menjadi N sehingga teks terang "wiki" akan menjadi "ZLNL" pada teks tersandi. Nama Caesar diambil dari Julius Caesar, jenderal, konsul, dan diktator Romawi yang menggunakan sandi ini untuk berkomunikasi dengan para panglimanya.

Langkah enkripsi oleh sandi Caesar sering dijadikan bagian dari penyandian yang lebih rumit, seperti sandi Vigenère, dan masih memiliki aplikasi modern pada sistem ROT13. Pada saat ini, seperti halnya sandi substitusi alfabet tunggal lainnya, sandi Caesar dapat dengan mudah dipecahkan dan praktis tidak memberikan kerahasiaan bagi pemakainya.

Cara kerja sandi ini dapat diilustrasikan dengan membariskan dua set alfabet; alfabet sandi disusun dengan cara menggeser alfabet biasa ke kanan atau ke kiri dengan angka tertentu (angka ini disebut kunci). Misalnya sandi Caesar dengan kunci 3, adalah sebagai berikut:

```
Alfabet Biasa:
ABCDEFGHIJKLMNOPQRSTUVWXYZ
Alfabet Sandi:
DEFGHIJKLMNOPQRSTUVWXYZABC
```

Untuk menyandikan sebuah pesan, cukup mencari setiap huruf yang hendak disandikan di alfabet biasa, lalu tuliskan huruf yang sesuai pada alfabet sandi. Untuk memecahkan sandi tersebut gunakan cara sebaliknya. Contoh penyandian sebuah pesan adalah sebagai berikut.

```
teks terang:   kirim pasukan ke sayap kiri
teks tersandi: NLULP SDVXNDQ NH VDBDS NLUL
```

Proses penyandian (enkripsi) dapat secara matematis menggunakan operasi modulus dengan mengubah huruf-huruf menjadi angka, A = 0, B = 1,..., Z = 25. Sandi (E_n) dari "huruf" $x$dengan geseran n secara matematis dituliskan dengan,

$$E_n(x) = (x + n) \mod 26. \quad (1)$$

Sedangkan pada proses pemecahan kode (dekripsi) adalah:

$$D_n(x) = (x - n) \mod 26. \quad (2)$$

Berikut ini adalah gambar model Caesar Cipher yang digunakan zaman Roma.

Figure 1 Caesar Cipher

Setiap huruf yang sama digantikan oleh huruf yang sama di sepanjang pesan, sehingga sandi Caesar digolongkan kepada, substitusi monoalfabetik, yang berlawanan dengan substitusi polialfabetik.

B. Vigenere Cipher

Sandi Vigenère adalah metode menyandikan teks alfabet dengan menggunakan deretan sandi Caesar berdasarkan huruf-huruf pada kata kunci. Sandi Vigenère merupakan bentuk sederhana dari sandi substitusi polialfabetik. Kelebihan sandi ini dibanding sandi Caesar dan sandi monoalfabetik lainnya adalah sandi ini tidak begitu rentan terhadap metode pemecahan sandi yang disebut analisis frekuensi. Giovan Batista Belaso menjelaskan metode ini dalam buku La cifra del. Sig. Giovan Batista Belaso (1553); dan disempurnakan oleh diplomat Perancis Blaise de Vigenère, pada 1586. Pada abat ke-19, banyak orang yang mengira Vigenère adalah penemu sandi ini, sehingga, sandi ini dikenal luas sebagai "sandi Vigenère".

Sandi ini dikenal luas karena cara kerjanya mudah dimengerti dan dijalankan, dan bagi para pemula sulit dipecahkan. Pada saat kejayaannya, sandi ini dijuluki *le chiffre indéchiffrable* (bahasa Prancis: 'sandi yang tak terpecahkan').

Sandi Vigenère sebenarnya merupakan pengembangan dari sandi Caesar. Pada sandi Caesar, setiap huruf teks terang digantikan dengan huruf lain yang memiliki perbedaan tertentu pada urutan alfabet. Misalnya pada sandi Caesar dengan geseran 3, A menjadi D, B menjadi E and dan seterusnya. Sandi Vigenère terdiri dari beberapa sandi Caesar dengan nilai geseran yang berbeda.

Untuk menyandikan suatu pesan, digunakan sebuah tabel alfabet yang disebut tabel Vigenère (gambar).

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Figure 2 Tabel Vigenere Cipher

Tabel Vigenère berisi alfabet yang dituliskan dalam 26 baris, masing-masing baris digeser satu urutan ke kiri dari baris sebelumnya, membentuk ke-26 kemungkinan sandi Caesar. Setiap huruf disandikan dengan menggunakan baris yang berbeda-beda, sesuai kata kunci yang diulang

Misalnya, teks terang yang hendak disandikan adalah perintah "Serbu Berlin" sedangkan kata kunci antara pengirim dan tujuan adalah "Pizza". "PIZZA" diulang sehingga jumlah hurufnya sama banyak dengan teks terang:

PIZZAPIZZAP

Huruf pertama pada teks terang, S, disandikan dengan menggunakan baris berjudul P, huruf pertama pada kata kunci. Pada baris P dan kolom S di tabel Vigenère, terdapat huruf H. Demikian pula untuk huruf kedua, digunakan huruf yang terletak pada baris I (huruf kedua kata kunci) dan kolom E (huruf kedua teks terang), yaitu huruf M. Proses ini dijalankan terus sehingga

Teks terang:	serbuberlin
Kata kunci:	PIZZAPIZZAP
Teks bersandi:	HMQAUQMQKIC

Proses sebaliknya (disebut dekripsi), dilakukan dengan mencari huruf teks bersandi pada baris berjudul huruf dari kata kunci. Misalnya, pada contoh di atas, untuk huruf pertama, kita mencari huruf H (huruf pertama teks tersandi) pada baris P (huruf pertama pada kata kunci), yang terdapat pada kolom S, sehingga huruf pertama adalah S. Lalu M terdapat pada baris I di kolom E, sehingga diketahui huruf kedua teks terang adalah E, dan seterusnya hingga didapat perintah "serbuberlin".

Enkripsi (penyandian) dengan sandi Vigenère juga dapat dituliskan secara matematis, dengan menggunakan penjumlahan dan operasi modulus, yaitu:

$$C_i \equiv (P_i + K_i) \mod 26 \quad (3)$$

atau C = P + K kalau jumlah dibawah 26 & - 26 kalau hasil jumlah di atas 26 dan dekripsi,

$$P_i \equiv (C_i - K_i) \mod 26 \quad (4)$$

atau P = C - K kalau hasilnya positif & + 26 kalau hasil pengurangan minus

Keterangan: $C_i$adalah huruf ke-i pada teks tersandi, P_i adalah huruf ke-i pada teks terang, $K_i$adalah huruf ke-i pada kata kunci, dan modadalah operasi modulus (sisa pembagian).

C. Analisis Pemecahan Sandi Vigenere

Sama seperti teknik cipher substitusi lainnya, teknik vigenere cipher mencoba untuk memalsukan atau mengubah sebuah karakter pada sebuah plainteks menjadi huruf lain pada cipherteks. Namun kelemahan utama pada vigenere cipher adalah kuncinya yang terkadang memiliki panjang tidak sama dengan plainteksnya harus diulang sehingga memiliki panjang yang sama dengan plainteks. Sehingga jika kita mampu menebak berapa panjang karakter dari sebuah kunci maka kita akan semakin mudah untuk memecahkan cipherteks tersebut. Berikut adalah beberapa teknik yang sering digunakan untuk membantu menentukan panjang kunci.

1. Analisis Kasiski

Analisis kasiski menggunakan fakta bahwa pada beberapa kata terkadang dienkripsi menggunakan huruf kunci yang sama sehingga menghasilkan blok-blok yang berulang. Misal untuk kasus seperti ini:

Key:
ABCDABCDABCDABCDABCDABCDABCD
Plaintext:
CRYPTOISSHORTFORCRYPTOGRAPHY
Ciphertext:
CSASTPKVSIQUTGQUCSASTPIUAQJB

Pada teks di atas dapat dilihat bahwa terdapat blok kata yang berulang dua kali yaitu CSASTP. Asumsikan bahwa blok yang berulang adalah blok plaintext yang sama. Maka, kita dapat menebak bahwa panjang kunci adalah salah satu diantara bilangan pembagi jarak dua kata itu, yaitu 16. Berarti kemungkinan panjang kunci adalah 16, 8, 4, 2, atau 1. Panjang kunci 2 dan 1 hampir tidak mungkin untuk digunakan karena terlalu pendek, maka menyisakan kemungkinan panjang kunci adalah 16, 8, atau 4. Karena kita telah mengetahui kemungkinan panjang kunci tersebut maka kita akan semakin mudah untuk memecahkan cipherteks tersebut

2. Friedman Test

SFriedman test pertama kali ditemukan oleh William F. Friedman pada tahun1920-an. Metode ini menggunakan *index of coincidence*. Teknik ini membutuhkan dua variable yaitu Kp dan Kr dimana Kp adalah probabilitas dua huruf yang dipilih secara acak pada sebuah bahasa adalah sama. Untuk bahasa inggris nilai ini adalah 0.067. Sedangkan Kr adalah kemungkinan sebuah pemilihan random dari sebuah alfabet (1/26) Maka panjang kunci yang dimaksud dapat diperkirakan sebagai:

$$\frac{\kappa_p - \kappa_r}{\kappa_o - \kappa_r} \quad (5)$$

Dengan K0 adalah

$$\kappa_o = \frac{\sum_{i=1}^{c} n_i(n_i - 1)}{N(N-1)} \quad (6)$$

Dimana :
C : panjang alphabet (26 untuk bahasa inggris)
N : panjang teks
Ni : Frekuensi kemunculan huruf

III. VIGENERE DENGAN TABEL 3 DIMENSI DAN 2 KATA KUNCI

Salah satu kelemahan yang dimiliki oleh vigenere cipher biasa adalah kemungkinan terjadinya perulangan pada cipher teks serta pada kunci. Kemungkinan panjang kunci sama dengan panjang plain teks juga sangat sulit diterapkan pada informasi-informasi yang berukuran sangat besar. Oleh karena itu, untuk mempersulit pemecahan vigenere cipher ini maka akan ditambahkan satu dimensi lagi untuk menambah kemungkinan huruf yang tersedia pada table vigenere ini.

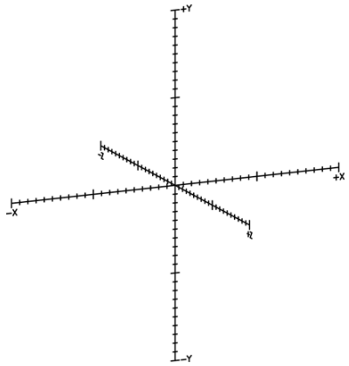

Figure 3 Model 3 dimensi

A. Tabel 3 Dimensi

Table vigenere yang digunakan dalam teknik ini adalah table vigenere 3 dimensi. Sumbu-sumbu yang

terdapat pada vigenere cipher yang menggunakan model 3 dimensi ini:
Sumbu X : Berisi 26 Point dari A-Z
Sumbu Y : Berisi 26 Point dari A-Z
Sumbu Z : Berisi 26 karakter A-Z

B. Dua Kunci pada Vigenere 3 Dimensi

Kunci yang digunakan pada teknik enkripsi ini ada dua kunci. Kunci pertama adalah kunci yang digunakan untuk menentukan ketinggian pada sumbu Z, serta menjadi pengenkrip untuk plainteks sedangkan kunci yang satu lagi akan digunakan untuk mengenkrip plainteks sama seperti vigenere cipher biasa.

Proses enkripsi yang terjadi sebanyak dua kali dapat dilihat pada contoh berikut.

Plain teks	: puantadellamaharani
Kunci 1	: satusatusatusatusatusat
Hasil 1	: huthlawydltgshtlsnb
Plain teks 2	: huthlawydltgshtlsnb
Kunci 2	: duaduaduaduaduaduad
Hasil 2	: kotkfazsdongvbtomne

Dari contoh di atas, dapat dilihat bahwa proses enkripsi yang terjadi dua kali sebenarnya dapat dipermudah dengan table vigenere 3 dimensi. Dengan table 3 dimensi, proses yang terjadi hanya berlangsung sebanyak 1 kali proses.

C. Teknik Enkripsi

Untuk mengenkripsi sebuah plainteks maka dibutuhkan dua kunci. Kunci pertama akan digunakan untuk menentukan ketinggian dari table vigenere cipher yang akan dipakai. Selain itu kan ditambahkan karakter spasi. Karakter spasi ini akan menjadi point ke 26 pada sumbu Z. Dengan adanya karakter spasi ini maka jika kunci pertama mengandung spasi , maka spasi tersebut tidak akan dibuang Kunci kedua akan digunakan untuk pengenkripsi sama seperti vigenere cipher biasa. Plainteks awalnya akan dienkrip melalui vigenere cipher biasa dengan kunci 2, lalu dicari ketinggiannya dengan menggunakan kunci pertama. Maka itulah hasil enkripsi dari plain teks tersebut.

D. Teknik Dekripsi

Untuk mendekripsi sebuah cipherteks, pertama-tama harus dilihat berada pada ketinggian berapa cipherteks tersebut berdasarkan kunci. Jika ternyata ketinggian yang didapat adalah ketinggian 0, maka ambil satu huruf lalu didekrip. Jika ketinggian besar dari 0 maka ambil dua huruf lalu mendekrip. Teknik dekrip yang digunakan adalah sebagai berikut. Ambil satu atau dua karakter (tergantung pada ketinggian yang didapatkan). Setelah mendapatkan ketinggian tersebut, maka pendekripsian sama dengan menggunakan vigenere cipher biasa yaitu mencari pasangan kunci (menggunakan kunci kedua) dan cipherteks yang didapatkan. Maka itulah plainteks yang akan didapatkan.

IV. HASIL DAN ANALISIS

Teknik Vigenere Cipher dengan 2 buah kata kunci diterapkan ke dalam pembuatan suatu program dengan Java Applet. Kode program yang digunakan adalah sebagai berikut.

```
  static char[] alphabet =
{'A','B','C','D','E','F','G','H','I','J',
'K','L','M','N','O','P','Q','R','S','T','
U','V','W','X','Y','Z'};


  private static char[][][]
createTable(char[] alphabet) {
       char[][] table = new
char[alphabet.length][alphabet.length];
       int a;

       for(int i = 0; i <
   alphabet.length; i++){
              for(int j = 0; j <
   alphabet.length; j++){
                     a = i+j;
                     if(a >=
   alphabet.length){
                            a = a -
alphabet.length;
                     }
                     table[i][j] =
alphabet[a];
              }
       }

       char[][][] table3d= new
char[alphabet.length][alphabet.length][al
phabet.length];

       for( i = 0; i < alphabet.length;
i++){
              for( j = 0; j <
alphabet.length; j++){

       table3d[i][j][0]=table[i][j];
              }
       }

       for(i = 0; i < alphabet.length;
i++){
              for(j = 0; j <
alphabet.length; j++){
                     for(int k = 1; k <
alphabet.length; k++){

       table3d[i][j][k]=alphabet(k-1);
                     }
              }
       }
       return table3d;
  }

  private static void
printTable(char[][][] table) {
```

```
      for(int i = 0; i < table.length;
i++){
             for(int j = 0; j <
table.length; j++){
                    for(int k = 0; k <
table.length; k++){

       System.out.print(table[i][j][k]);
                    }
             }
             System.out.println();
       }
  }

  public char enkripsi1(char p, char key)
      {
          int a=Integer.valueOf(p);
          int b=Integer.valueOf(key);
          int tot;

          if(a>=65&&a<=90)
          {
              a+=32;
          }
          tot=a;
          if(b>=65&&b<=90)
          {
              b+=32;
          }
          b-=96;
          if(a>=97&&a<=122)
          {
              a-=97;
              tot=(a+b)%26;
              tot+=97;
          }
          return (char)tot;
      }

      public char enkripsi2(char p, char
key)
      {
          int a=Integer.valueOf(p);
          int b=Integer.valueOf(key);
          int tot;
          tot=(a+b)%256;
          return (char)tot;
      }

      public String enkripsi(String p,
String key)
      {
          String hasil="";
          int index1=0;
          int index2=0;
          if(mode==3)
          {
              key=key+p;
          }
          while(index1<p.length())
          {
              if(index2==key.length())
              {
                  index2=0;
              }
              if(mode==1)
              {

hasil+=enkripsi1(p.charAt(index1),
key.charAt(index2));
              }
              else
              {

hasil+=enkripsi2(p.charAt(index1),
key.charAt(index2));
              }
              index1++;
              index2++;
          }
          return hasil;
      }

      public char dekripsi1(char p, char
key)
      {
          int a=Integer.valueOf(p);
          int b=Integer.valueOf(key);
          int tot;
          if(a>=65&&a<=90)
          {
              a+=32;
          }
          tot=a;
          if(b>=65&&b<=90)
          {
              b+=32;
          }
          b-=96;
          if(a>=97&&a<=122)
          {
              a-=96;
              tot=(a-b)%26;
              tot+=96;
          }
          return (char)tot;
      }

      public char dekripsi2(char p, char
key)
      {
          int a=Integer.valueOf(p);
          int b=Integer.valueOf(key);
          int tot;
          tot=(a-b)%256;
          return (char)tot;
      }

      public String dekripsi(String p,
String key)
      {
          String hasil="";
          String temp="";
          int index1=0;
          int index2=0;
          if(mode!=3)
          {
              while(index1<p.length())
              {

if(index2==key.length())
                  {
                      index2=0;
                  }
                  if(mode==1)
                  {

hasil+=dekripsi1(p.charAt(index1),
```

```
key.charAt(index2));
                    }
                    else
                    {

hasil+=dekripsi2(p.charAt(index1),
key.charAt(index2));
                    }

                    index1++;
                    index2++;
                }
            }
            else
            {
                while(index1<p.length())
                {

if(index2==key.length())
                    {
                        index2=0;
                        key=temp;
                        temp="";
                    }
  temp+=dekripsi2(p.charAt(index1),
key.charAt(index2));

hasil+=dekripsi2(p.charAt(index1),
key.charAt(index2));
                    index1++;
                    index2++;
                }
            }
            return hasil;
        }

        public String delspace(String text)
        {
            String hasil="";
            int a=0;
            while (a<text.length())
            {
                if(text.charAt(a)!=' ')
                    hasil+=text.charAt(a);
                a++;
            }
            return hasil;
        }

        public String parser1(String p,
String teks) //nambahin spasi
        {
            int a=0;
            while (a<p.length())
            {
                if(p.charAt(a)==' ')
                    teks=teks.substring(0,
a)+" "+teks.substring(a);
                a++;
            }
            return teks;
        }

        public String parser2(String
teks)//memberi spasi tiap 5 huruf
        {
            int a=0;
            String hasil="";
            while (a<teks.length())
            {
                hasil+=teks.charAt(a);
                if((a+1)%5==0)
                    hasil+=" ";
                a++;
            }
            return hasil;
        }

        public String hasil(String p,
String key)
        {
            String hasil="";
            String temp=delspace(p);
            if(modeed==1)
                hasil=enkripsi(temp, key);
            else
                hasil=dekripsi(temp, key);
            if(modec==1)
            {
                hasil=parser1(p, hasil);
            }
            else if(modec==2)
            {
                //do nothing
            }
            else
            {
                hasil=parser2(hasil);
            }
            return hasil;
    }
```

Kode program di atas terbagi atas 3 bagian utama, yaitu pembuatan tabel 3 dimensi, proses enkripsi, dan proses dekripsi teks. Pembuatan tabel 3 dimensi diawali dengan pembuatan table 2 dimensi yang berupa table vigenere kemudian ditambahkan 1 dimensi array lagi untuk menampung sumbu z yang berisi huruf hasil enkripsi berdasarkan kata kunci ke-2. Proses enkripsi dilakukan dengan cara mengenkripsi huruf dengan kata kunci pertama di table 2 dimensi. Setelah itu mengenkripsinya lagi dengan kata kunci kedua menggunakan sumbu z di table 3 dimensi. Proses dekripsi berlaku sebaliknya.

Setelah diimplementasikan di Java Applet, hasil dari penggunaan teknik ini dapat dilihat pada gambar-gambar di bawah ini.

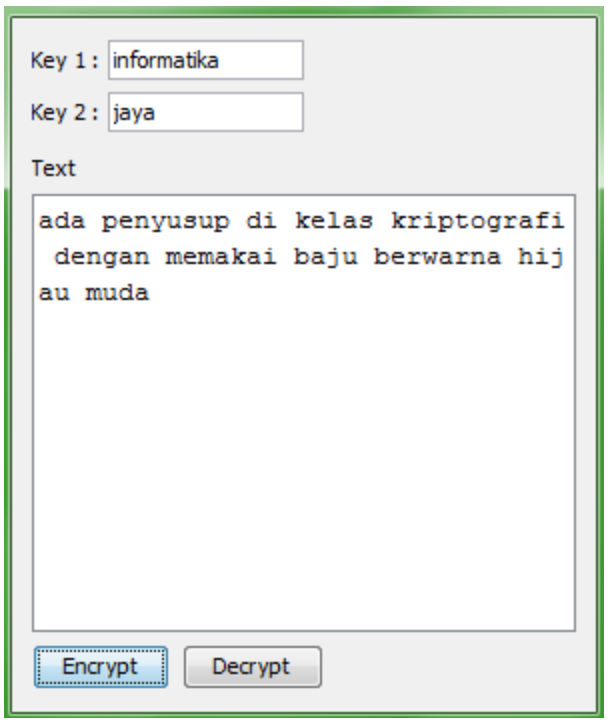

Figure 4 Halaman Awal

Halaman awal aplikasi ini memuat 2 buah kotak penulisan 2 kata kunci yang dipakai. Di bawahnya, terdapat kotak teks besar yang menampung teks yang akan dienkripsi atau didekripsi.

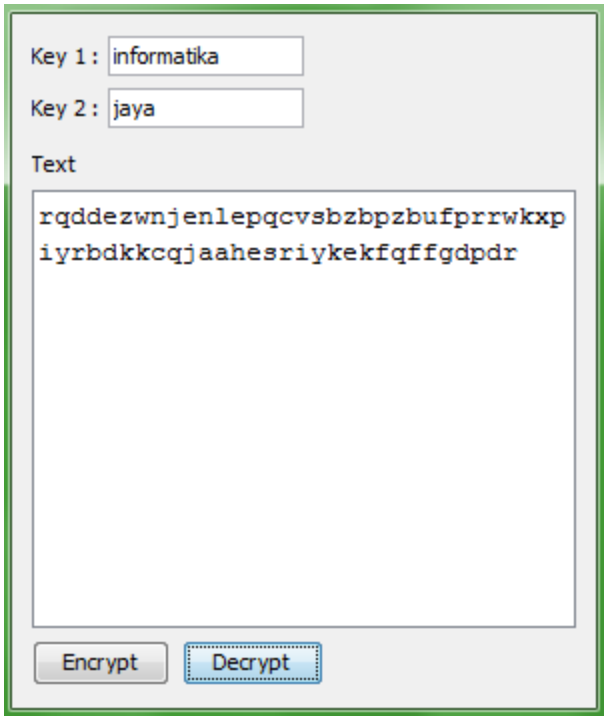

Figure 5 Enkripsi Teks

Setelah itu, teks dienkripsi dengan 2 kata kunci yang telah dimasukkan pengguna. Teks yang dienkripsi akan membuang tanda baca dan spasi sehingga hasilnya menjadi seperti pada gambar 4. Hal ini dilakukan agar kriptanalis tidak mudah dalam memecahkan kodenya.

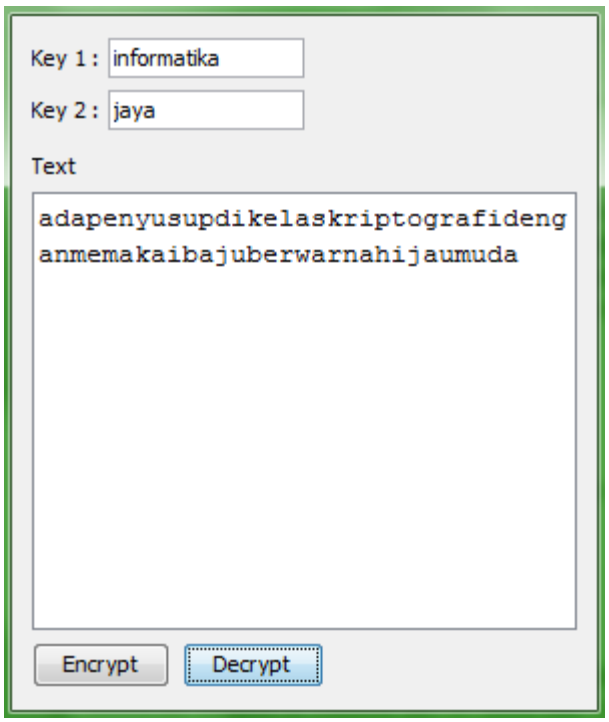

Figure 6 Dekripsi Teks

Untuk mendekripsi, pengguna dapat menekan tombol dekripsi lalu mendapat hasil dekripsi seperti pada gambar 5. Karena teks cipher yang dimasukkan tidak memakai tanda baca, maka hasil dekripsi juga tidak menggunakan tanda baca.

Penggunaan teknik vigenere cipher dengan menggunakan model tiga dimensi akan menguatkan pengenkripsian dari vigenere cipher biasa. Hal ini dapat dilihat dari panjang ciphertext yang tidak memiliki panjang sama dengan plain teks. Hal ini dapat menyebabkan para kriptanalis bingung untuk memilih bagian mana dari cipherteks yang akan di dekrip, apakah hanya satu huruf atau dua huruf.

Penggunaan dua kunci juga akan memiliki nilai kurang dan lebih. Nilai lebihnya adalah jika salah satu kunci berhasil dicuri, belum tentu cipherteks tersebut dapat dipecahkan. Namun dengan penggunaan dua kunci mengharuskan pengguna untuk ekstar hati-hati dimana jika pengguna kehilangan kunci tersebut maka cipherteks tersebut akan menjadi senjata makan tuan. Selain itu penggunaan model ini akan mengacaukan teknik analisis frekuensi dimana seringnya muncul karakter-karakter yang justru tidak ada hubungannya dengan teks atau informasi asli. Penggunaan teknik ini juga memperbanyak kemungkinan kombinasi huruf sehingga semakin tidak memungkinkan penggunaan teknik bruteforce.

V. KESIMPULAN

Vigenere cipher adalah salah satu algoritma kriptografi klasik yang masih sering digunakan karena kekuatan dan kemudahan penggunaannya.

Namun, dengan menggunakan beberapa metode, seperti analisis kasiski menjadikan algoritma ini menjadi dapat dipecahkan. Oleh karena itu, algoritma ini memerlukan suatu pengembangan dimana dengan tidak menghilangkan identitas vigenere cipher, algoritma ini dapat menjadi semakin kuat.

Salah satu teknik yang dapat digunakan untuk memperkuat algoritma ini adalah dengan menggunakan vigenere cipher yang menggunakan model tiga dimensi sebagai table vigenerenya. Dengan menggunakan algoritma ini, maka teknik analisis frekuensi dan metode kasiski akan semakin sulit menebak sebuah cipherteks karena cipher teks yang dihasilkan lebih panjang dari cipherteks. Hal ini mengakibatkan munculnya karakter-karakter yang sebenarnya tidak memiliki hubugan langsung dengan informasi asli. Pengguna yang ingin mengenkrip dan mendekrip secara manual juga dapat melakukannya secara manual.

REFERENCES

Munir, Rinaldi. 2011. "Bahan Kuliah IF3054 Kriptografi". Departemen Teknik Informatika, Institut Teknologi Bandung

Ans, Muhammad. 2010. "3D Model Vigenere Cipher". Departemen Teknik Informatika, Institut Teknologi Bandung

http://id.wikipedia.org/wiki/Sandi_Caesar
waktu akses: 19 Maret 2012 pukul 17.00 WIB

http://id.wikipedia.org/wiki/Sandi_Vigen%C3%A8
waktu akses: 19 Maret 2012 pukul 17.00 WIB

http://sharkysoft.com/misc/vigenere/
waktu akses: 19 Maret 2012 pukul 17.00 WIB

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Maret 2012

Puanta Della Maharani
13507135