Permutasi 7 Huruf Pada Confusing Vigenere Cipher

7th Sword Vigenere

Robert Gunawan 13508038
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
gabriel_robert_gunawan@yahoo.com

Makalah ini akan membahas pengenai pengembangan algoritma Vigenere Cipher yang memungkinkan penyembunyian pesan sekaligus mengacak pesan yang ada sehingga mampu menyembunyikan pola yang terbentuk dari penggunaan Vigenere Cipher untuk menyembunyikan pesan. Dalam makalah ini akan dibahas mengenai celah keamanan di Vigenere Cipher yang biasa, dasar algoritma 7th Sword Vigenere, sekaligus dokumentasi dari pembuatan applikasi dalam bahasa java yang memanfaatkan algoritma ini untuk mengamankan data yang ada. Makalah ini sekaligus merupakan bentuk dokumentasi dari pembuatan program 7th Sword Vigenere.

Kata kunci: Algoritma, Vigenere Cipher, Caesar Cipher, enkripsi, dekripsi, kunci, permutasi, confusing, ASCII.

I. Pendahuluan

Algoritma Vigenere Cipher adalah algoritma pengamanan standar enkripsi dekripsi yang sudah sangat umum digunakan untuk mengenkripsi data yang sederhana. Vigenere Cipher merupakan turunan dari Caesar Cipher yang menyembunyikan pesan dengan mengganti huruf-huruf yang akan disembunyikan dengan huruf yang memiliki selisih rentang huruf tertentu dengan huruf yang akan diganti. Caesar Cipher diperkenalkan oleh Julius Caesar dan digunakan untuk menyembunyikan pesan pada saat zaman kerajaan Romawi. Cara ini kurang aman karena pola yang terbentuk masih mungkin dikenali sebagai suatu kata / kalimat karena masing-masing huruf hanya diganti dengan suatu huruf yang memiliki selisih rentang huruf yang sama untuk seluruh pergantian.

Dari kasus di atas, munculah turunan dari Caesar Cipher yang diperkenalkan oleh Blaise de Vigenere berdasarkan algoritma yang dibuat oleh Giovann Battista Bellaso. Vigenere Cipher memanfaatkan suatu kunci yang berupa sejumlah karakter yang akan dipergunakan untuk menyembunyikan pesan yang ada. Masing-masing huruf kunci digunakan untuk menyembunyikan huruf yang ada dalam pesan dan terus diulang-ulang sampai seluruh pesan disembunyikan. Cara ini juga masih memiliki kelemahan dimana kemunculan huruf masih dapat dianalisa dan digunakan untuk membuka pesan yang ada berdasarkan pola kemunculan huruf dan pencarian kunci lemah. Kunci lemah adalah suatu kunci dimana jika kunci tersebut digunakan maka hasil penyembunyian pesan sama dengan pesan asli yang ada.

Berdasarkan 2 kasus di atas, penulis menemukan cara untuk menyamarkan pola yang ada dengan permutasi huruf dan prinsip confusing. Permutasi adalah suatu cara matematis untuk mengurutkan dengan memperhatikan letak urutan dan confusing adalah teknik untuk membingungkan pencuri data dalam pengamanan data. Dengan cara ini penulis dapat menghapus jejak pola yang timbul sehingga mempersulit proses pencarian data dengan analisa kemunculan pola yang ada.

II. Landasan Teori

2.1. Vigenere Cipher

Vigenere Chiper merupakan perkembangan lebih lanjut dari Caesar Cipher yang memiliki kunci. Yang membedakan Vigenere Cipher dengan Caesar Cipher adalah Vigenere Cipher memiliki kunci yang diulang dengan pola sesuai dengan pola kunci sebanyak / sepanjang teks yang akan dienkripsi ataupun didekripsi. Vigenere ditemukan oleh Giovan Battista Bellaso pada tahun 1953 dan disempurnakan oleh Blaise de Vigenere pada tahun 1958. Nama Vigenere sendiri dibuat untuk menghormati Blaise de Vigenere.

Berikut ini diilustrasikan cara kerja Vigenere Cipher

i. Misal teks yang akan di enkripsi (mode 26 karakter) adalah “serbuberlin”

ii. Kunci yang digunakan adalah “pizza”

iii. Berikut adalah gambar ilustrasi

Teks : serbuberlin
Kata kunci : pizzapizzap
Teks bersandi: hmqauqmqkic

iv. Teks yang dienkripsi menjadi hmqauqmqkic sehingga musuh akan kebingungan jika tidak tahu kalau pesan sudah di enkripsi

v. Untuk dekripsi, teks bersandi akan dipecahkan dengan kunci menjadi teks asli sesuai dengan teks yang sebenarnya jika kuncinya tepat.

Rumus Vigenere Cipher adalah sebagai berikut (mode 256 karakter ASCII)

Enkripsi: Ci = (Pi+Ki) mod 256
Dekripsi: Pi = (Ci-Ki) mod 256; untuk Ci >= Ki
Pi = (Ci+256-Ki) mod 256; untuk Ci < Ki

2.2 Permutasi

Permutasi adalah penyusunan kembali suatu kumpulan objek dalam urutan yang berbeda dari urutan yang semula. Sebagai contoh, kata-kata dalam kalimat sebelumnya dapat disusun kembali sebagai "*adalah Permutasi suatu urutan yang berbeda urutan yang kumpulan semula objek penyusunan kembali dalam dari.*" Proses mengembalikan objek-objek tersebut pada urutan yang baku (sesuai ketentuan) disebut sorting.

Rumus untuk menentukan banyaknya permutasi

$$P(n,r) = \frac{n!}{(n-r)!}$$

Dengan n adalah banyak objek dan r adalah jumlah objek yang akan diurutkan.

2.3 Confusing

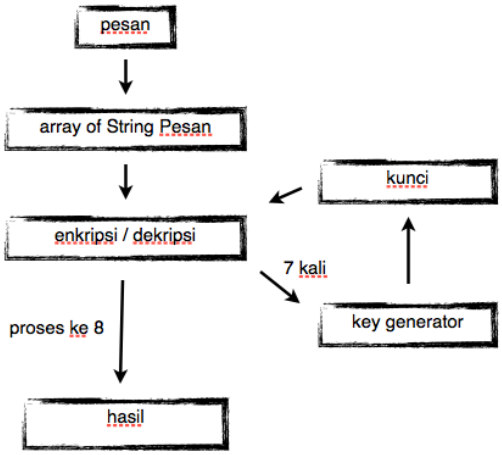
Pada dasarnya confusing adalah cara menyembunyikan keterkaitan antara hasil pesan yang disembunyikan dengan pesan sebelumnya. Berbagai macam metode confusing telah dikenal dan dipakai di dalam algoritma penyembunyian pesan salah satunya adalah Pengacakan (Randomness).

III. Perancangan Algoritma

Algoritma 7th Sword Vigenere dibuat dengan menggunakan algoritma Vigenere Cipher dengan menggabungkan permutasi 7 huruf dan prinsip confusing. Berikut ini adalah urutan proses algoritmanya

1. Membaca masukan berupa teks
2. Menyimpan dalam sebuah variabel array of String (kumpulan dari kumpulan character)
3. Melakukan prosedur enkripsi / dekripsi dengan kunci awal.
4. Melakukan prosedur enkripsi / dekripsi sebanyak 7 kali dengan kunci yang dibuat berdasarkan permutasi kunci sebelumnya. Di sinilah letak confusing dan permutasi 7 hurufnya
5. Menyimpan dalam array of String
6. Menampilkan ke pengguna hasil enkripsi / dekripsi yang telah dilakukan.

Untuk lebih jelasnya, dapat dilihat di gambar berikut ini



Gambar 1

Perancangan Algoritma

Inti dari algoritma ini terletak pada proses dekripsi dan enkripsi yang berulang dengan memanfaatkan kunci yang dibuat berdasarkan kunci sebelumnya. Dalam pembuatan kunci, algoritma harus mampu menyajikan suatu keterkaitan antara kunci sekarang dengan kunci sebelumnya, sehingga makin mempersulit pencuri data untuk menebak kuncinya.

IV. Lingkungan Implementasi

Sistem Operasi	: Mac OSX Snow Leopard 10
Kakas pemrograman	: Netbeans 6.9.1
Bahasa Pemrograman	: java JDK 6
Implementasi	: Java Swing JFrame
Library / engine	: N/A

V. Algoritma

5.1 Pengolah Character

Secara umum, terdapat 2 hal yang dibuat dalam pengolah character yaitu enkripsi dan dekripsi. Di sini, dibuatlah suatu prosedur untuk enkripsi dan dekripsi per character untuk memudahkan proses enkripsi dan dekripsi kumpulan character (String) yang ada dengan metode Caesar Cipher.

Berikut ini adalah rumus umum enkripsi dan dekripsi karakter 256 ASCII
Enkripsi: Ci = (Pi+Ki) mod 256
Dekripsi: Pi = (Ci-Ki) mod 256; untuk Ci >= Ki
Pi = (Ci+256-Ki) mod 256; untuk Ci < Ki
Berikut ini adalah implementasinya

```
public static char Enkripsi (char
a,char b)
    {
        //kamus lokal
        int Nilai1 = (int) a;
        int Nilai2 = (int) b;
        int NilaiAkhir;
```

```
        //algoritma

        //mulai enkripsi 256
            NilaiAkhir =
(Nilai1+Nilai2)%256;
            return (char) NilaiAkhir;
    }
```

```
public static char Dekripsi (char
a,char b)
    {
        //kamus lokal
        int Nilai1 = (int) a;
        int Nilai2 = (int) b;
        int NilaiAkhir;

        //algoritma
           if(Nilai1<Nilai2)
           {
               Nilai1 = Nilai1+256;
           }

           //mulai dekripsi
           NilaiAkhir = (Nilai1-
Nilai2)%256;
           return (char)NilaiAkhir;

    }
```

5.2 Pengolah String

Secara umum, Pengolah String menangani enkripsi dan dekripsi yang berhubungan dengan string sekaligus sebagai pembaca dan penyimpan file yang ada.

Pada proses enkripsi dan dekripsi, dilakukan proses perulangan untuk menghasilkan satu per satu character dengan Pengolah Character 5.1.

Berikut ini adalah algoritma untuk enkripsi dan dekripsi string

```
public static String
PenangananEnkripsiBaris(String a,
String b)
    {
        //kamus lokal
        String Hasil;
        Hasil = "";
        int Pa = a.length();
        int Pb = b.length();
        int i;
        String temp;
        char lala;
        //algoritma

        for(i=0;i<Pa;i++) //direpetisi
           {
               Hasil = Hasil +
(Character.Enkripsi(a.charAt(i),
b.charAt(i%Pb)));
           }
        return Hasil;
    }
```

```
public static String
PenangananDekripsiBaris(String a,
String b)
            //ArrayString berisi array
byte 8 bit dan b adalah sebuah kunci
            //Keluaran berupa string
yang terenkripsi
    {
        //kamus lokal
        String Hasil;
        Hasil = "";
        int Pa = a.length();
        int Pb = b.length();
        int i;
        String temp;
        char lala;
        //algoritma

        for(i=0;i<Pa;i++) //direpetisi
           {
               Hasil = Hasil +
(Character.Dekripsi(a.charAt(i),
b.charAt(i%Pb)));
           }
        return Hasil;
    }
```

Dalam kelas ini juga dibuat prosedur untuk mengenkripsi dan dekripsi menggunakan generate kunci yang akan dijelaskan di 5.3. Berikut ini adalah algoritmanya.

```
public static ArrayList<String>
Enkripsi7Kali (ArrayList<String>
Masukan, String Kunci)
    {
        //kamus lokal
        ArrayList<String> Hasil =
Masukan;
        String tempkunci = Kunci;
        int i;
        //algoritma
        for(i=0;i<7;i++)
        {
            tempkunci =
PenangananKunci.GenerateKunci(tempkunc
i);
            Hasil =
Enkripsi(Hasil,tempkunci);
        }
        return Hasil;
    }
```

```
public static ArrayList<String>
Dekripsi7Kali (ArrayList<String>
Masukan, String Kunci)
    {
        //kamus lokal
        ArrayList<String> Hasil =
Masukan;
        String tempkunci = Kunci;
        int i;
        //algoritma
```

```
        for(i=0;i<7;i++)
        {
            tempkunci =
PenangananKunci.GenerateKunci(tempkunc
i);
            Hasil =
Dekripsi(Hasil,tempkunci);
        }
        return Hasil;
    }
```

5.3. Pengolah Kunci

Pengolah kunci hanya terdiri dari 1 prosedur di mana prosedur itu akan membuat sebuah kunci baru dari kunci lama yang ada. Berikut ini adalah gambaran umum prosedur pembuatan kunci

1. Mengecek jika panjang masukan < 7 maka tambahkan character tambahan di belakangnya. Character yang ditambahkan adalah character terdepan lalu mundur ke belakang sampai panjang masukan >= 7
2. Membuat permutasi
3. Memilih permutasi dengan rumus

   Terpilih: ((character ke length mod 7) * ASCII char 1 * ASCII char 7) mod banyak permutasi.
4. Memulai rekursif

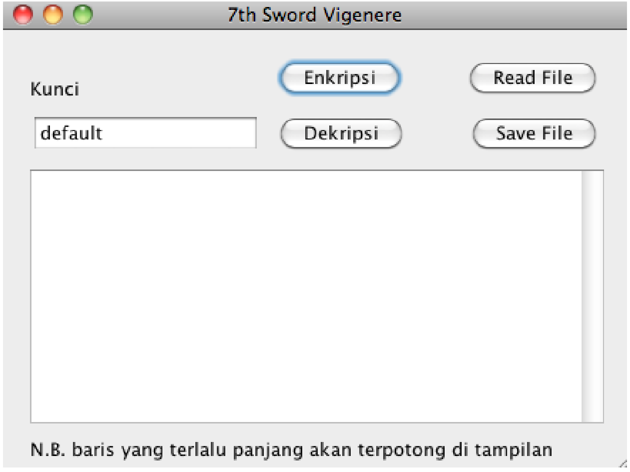
   Basis: jika character = 7, hasil = masukan

   Rekurens: Jika > 7, hasil = masukan + rekurens sisanya.

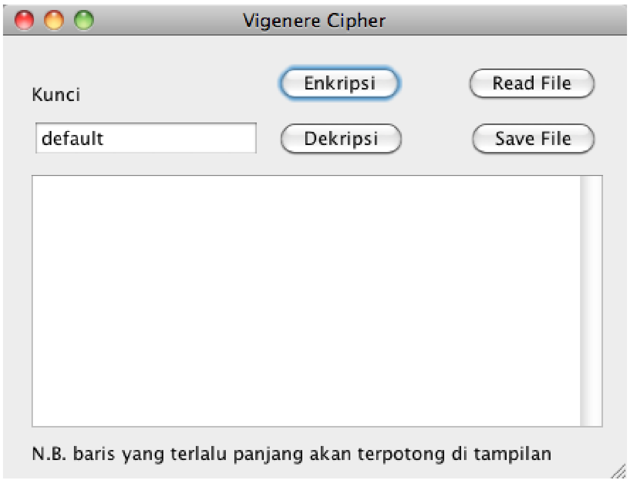
Berikut ini adalah algoritmanya

```
public static String GenerateKunci(String masukan)
   //masukan adalah kunci yang akan digenerate
        //keluaran adalah versigenerate dari kunci yang
ada
   {
      //kamus lokal
      int i;
      int a,b,c,d,e,f,g;
      String Hasil = "";
      int j =0;
      //algoritma
      if(masukan.length() < 7)
      {
         for(i=masukan.length();i<7;i++)
         {
            masukan = masukan + masukan.charAt(j);
//dummy char di belakang
            j++;
         }
      }

      //kunci sudah min 7 char
      //mulai proses pengacakan

      //permutasi
      ArrayList<String> Permutasi = new
ArrayList<String>();
      for(a=0;a<7;a++)
      {
         for(b=0;b<7;b++)
         {
            for(c=0;c<7;c++)
            {
               for(d=0;d<7;d++)
               {
                  for(e=0;e<7;e++)
                  {
                     for(f=0;f<7;f++)
                     {
                        for(g=0;g<7;g++)
                        {
                           if((a!=b) && (a!=c) && (a!=d)
&& (a!=e) && (a!=f) && (a!=g)
                              && (b!=c) && (b!=d) && (b!
=e) && (b!=f) && (b!=g)
                              && (c!=d) && (c!=e) && (c!
=f) && (c!=g)
                              && (d!=e) && (d!=f) && (d!
=g)
                              && (e!=f) && (e!=g)
                              && (f!=g))
                           {
                              String kodok = "";
                              kodok = kodok +
masukan.charAt(a) + masukan.charAt(b) +
masukan.charAt(c) + masukan.charAt(d) +
masukan.charAt(e) + masukan.charAt(f) +
masukan.charAt(g);
                              Permutasi.add(kodok);
                           }
                        }
                     }
                  }
               }
            }
         }
      }
      //array permutasi sudah terisi
      int selected = ((int)
masukan.charAt(masukan.length()%7)) *
((int)masukan.charAt(6)) * ((int)masukan.charAt(0))
%Permutasi.size();

      if(masukan.length()== 7)
      {
         Hasil = Permutasi.get(selected);
      }
      else
      {
         Hasil = Permutasi.get(selected) +
GenerateKunci(masukan.substring(7,masukan.length()))
;
      }

      return Hasil;
   }
```

VI. Tampilan Antarmuka



Gambar 2
Antarmuka 7th Sword Vigenere



Gambar 3
Antarmuka Vigenere Cipher

VII. Pengujian dan Pembandingan

Sebagai salah satu algoritma turunan Vigenere Cipher yang berarti turunan ke-2 Caesar Cipher, tentu saja tidak langkap jika belum dibandingkan performanya dengan algoritma Vigenere Cipher. Perbandingan di sini merupakan gabungan dari kombinasi kunci dan masukan untuk setiap analisa kasus. Vigenere Cipher yang terpilih juga dalam bentuk 256 Vigenere Cipher yang dibuat dengan primitif dasar yang sama.

7.1 String pendek (Lebih kecil dari 100 character)
Masukan

ini coba bandingkan antara vigenere cipher dengan 7th sword vigenere

Kunci: test

Keluaran 7th Sword Vigenere

√æ¬´¬®√±¬ß√ª¬ø√≥√´¬≥√µ¬†

√ù¬≤ √ª√æ√¨¬∑√£e √∏√Ø¬≥√∂¬©√æ¬ø√•¬¥√π¬´√Ω √æ√µ √≤¬¶√∞e√ë√π¬™¬∏¬¶ √Ä√≤¬≠√∏¬´√π√∞

Keluaran Vigenere Cipher

√ù√¨√ú¬î√ó√î√ï√¨¬î√á√Ö√¢√ò√ë√º√õ√ú√Ü√º √ï√Ô√ß√Ï√¶√Ü¬ì√™√Ù√Ñ√ò√¢√ô√ó√ò¬î√ó√Ñ√£√ú√ô√ó √ò√ä√º√õ√ï√ì¬ì¬´√®√ç¬ì√ß√ ´√î√•√ò¬î√õ√ú√õ√ô√ì√ò√¶√ô

7.2 String Panjang (Lebih dari 100 character)
Masukan:

PT. Bursa Berjangka Jakarta was established on August 19, 1999 by 4 palm plantations, 7 refineries, 8 coffee exporters, 8 securities companies and 2 general traders. Jakarta Futures Exchange has been licensed to operate and held a soft opening on December 15, 2000. The Exchange is designed as a multi-commodity Futures Exchange. The first two contracts: Olein Futures Contract and Robusta Futures Contract will be followed in due time by Cocoa Futures, Pepper Futures, Rubber Futures, and Plywood Futures. Options on Futures and Financial Futures will be considered later.

Kunci: test

Keluaran 7th Sword Vigenere

√™√§√ç¬´¬á√∑¬∑ ¬±√å¬™¬∫√∫ √ß¬•¬±¬ê√¨ √Ω¬™√±ƒ √±¬´¬™√µ¬π√ª√Ω√≥¬∏√π¬ú√Ω¬ø √™d√í¬ª√≤√øV√Å√Ü¬∞√ê√Ñ~¬ªe√º¬±¬æe¬ò √Ä√¨¬∞√≤¬•√ø√ø√•¬•√æ¬π√Ü¬∞√ñ¬´¬∑√ß¬´ √∂√º¬Æ√∂¬™√ñ¬ø√ò¬ú¬ß¬¨√± ¬´õ¬∂ √∞∑√µq¬∫√ó¬±√Ω¬™√•¬¨ √•¬© ƒ√Æ √ª¬ó√æ¬Ø√ø¬ø√¨¬≥√¶e√å¬ø√∏√Ø¬≥√∂¬©√∫ √Ä√∞¬∂√≤¬™√∞ ¬πV√ö¬ß √± √ø¬¶¬¢¬ã √º¬™ W√û√§¬•√ø¬≠√∞¬æ√¨¬©¬∞¬®√ø√µ ¬´¬±√´¬®√ø √Ø¬©¬±¬´¬ø √¨¬©¬ß√ø√Ä√¨¬ß√•ƒ√µ √Øe√£e √∑√æe¬ß√æ √™¬´¬±¬µ√π¬æ√§√∞¬ô√µ¬≥√º√µ ¬´v¬∑q¬∫√ë√Å¬∫u¬øW√≠

¬ú¬â ¬©√≥√ø
√≤¬õ¬∞¬Ø
¬∞√∞¬∏√´¬¨
√µ¬™¬¶
W√∫¬ø
√±¬∞
¬Ø¬∏
√∏¬£√ø¬™
¬´¬ã√∑¬π

√∂√Ωe√ñ¬Ø√º
√™¬´√∂t¬´√≤√∞V√∂¬Ø
¬´¬π√π¬¥¬∫√∏¬π¬ò√º¬∂d√†¬≤√∞

¬´|
¬∫

√æe√Ö¬¥√´¬®
W√∫

¬ú¬ñ¬®
√¨V√ñ¬ª

√∞¬∏¬¢¬à

√º¬¶√¥¬´¬π √®¬∞¬±¬®√∞¬æ
√∫¬¢√º¬µ
√µ¬´¬Æ√∞e√æ√∂¬™¬π√∫¬§√æ¬ø√µd√î¬µ√Æ
¬´|
¬∫

√æq¬¢¬ï√ø
√Ø¬∑¬±}
√Æ¬©
r¬´√∞√≠¬ò√µ¬∏¬∫√ñ√ø¬∫√¥¬™
√ã¬±√´¬≥√µW√©
√≥¬≥¬™¬´√§√ø¬´¬´
¬æ¬ø√ö¬µ√∂¬Æ

¬™¬¥√øW√ü√±¬∂√∂¬π¬´√ø
√ØV√ñ¬Ø√±
√Æ¬Æ√£¬±¬∫√•
√æ¬∫¬ú
¬ø√•¬∞√Ωf√≠√Ä√Æ¬•√æ¬π√¥
√Ω¬™√¶e

√Ø¬∑¬ø

Keluaran Vigenere Cipher

√Ñ¬π¬º¬î¬∂√ö√•√ß√ï¬Ö¬µ√ô√¶√è√î√
¢√õ√ê√î¬æ√Ü√ûú√¶√ô√î¬î√
´√Ü√¶¬î√ô√ò√ß√ï√ñ√ëú√ç√ú√ä√ó¬î√£√ì¬ì¬µ√©√å√
®√ß√®¬Ö¬§¬≠¬†¬Ö¬§¬≠¬≠¬û¬ì√ñ√≠¬Ö¬ß¬î√§√Ü√ü
√º¬î√ï√ü√ï√¢√ô√î√®√ù√î√º√ß¬†¬Ö¬™¬î√¶√ä√ô√ù√
¢√ä√•√ù√ô√ò¬ü¬î¬®¬Ö√ñ√£√ö√ã√ò√ô¬î√ä√
´√§√£√ó√ç√ô√¶√ò¬ü¬î¬¨¬Ö√¶√ô√ó√ö√•√ù√®√©√ò√ß¬
î√à√¢√º√§√Ü√º√ù√ô√ò¬ì√ï√¢√â¬ì¬¶¬î√Ö√ò√

¢√ô√ó√î√†¬î√ô√•√ö√ï√ä√ß¬
¢¬Ö¬Ω√ï√ü√Ü√•√®√ï√ï¬π√©√®√ö√ô√ß√ô√ß¬Ö¬∏√¨√
ó√ç√î√¢√ã√∂√ä√ë√ô√ã√ì¬î√ñ√ã√ª¬î√Ü¬ì√ß√ü√ô√
¢√ò√ò√ì¬î√ô
¢¬î√£√ï√¶√ù√ò¬î√ï√ì√ó¬î√ú√ä√ü√ò¬î√Ü¬ì√ß√£√ã
√ß¬î√£√ï¬ì¬∏√ô√à√ò√º√ñ√ä√•¬î¬ï¬ö¬ü¬î¬¶¬ï¬
£¬ß¬¢¬Ö√á√ú√ô¬Ö¬∏√¨√ó√ç√î
¢√õ√ä¬ì√ù√ß¬Ö√ó√ô√ß√é√ö
¢√ô√ä¬ì√ï√ß¬Ö√ï¬î√º√ö√ü√®√ù¬ì√ñ√£√º√ú√
¢√ò√ù√ô√¨¬î¬∫√ö√ß√©√¶√ä√¶¬î¬π√ù√ñ√ú√ó√ö√ò¬
¢¬Ö√Ü√ú√ô¬Ö√ô√ù√¶√ò√ß¬î√®√ú
¢¬î√ó√î√º√®√¶√Ü√ñ√®√ß¬ü¬ì√É√†√ä√ú√¢¬î¬
´√®√®√©√ó√ò√ß¬î¬®√¢√¢√®√ó√î√ó√®¬Ö√Ô√î
¢√ò¬Ö√Ö√£√ñ√¶√ï√®√ñ¬π√©√®√ö√ô√ß√ô√ß¬Ö¬∂√£
√¢√ô√•√ï√ì¬ì√´√ù√ë√ü¬î√ñ√ä¬ì√ö√£√ë√ü√£√
´√Ü√ó¬î√ú√ì¬ì√ò√©√Ü¬ì√®√ú√í√ò¬î√ñ√û¬ì¬∑√£√Ü√
¢√ï¬î¬´√®√®√©√ó√ò√ß¬†¬Ö√É√ô√§√ï√ò√¶¬î
´√®√®√©√ó√ò√ß¬†¬Ö√Ö√©√ñ√Ü√ò√¶¬î
´√®√®√©√ó√ò√ß¬†¬Ö√î√¢√ò¬Ö√É√†√≠√ú√
¢√£√ò¬Ö¬π√©√®√ö√•√ô√ß¬ì√É√§√ô√ú√£√
¢√ò¬ì√£√¢¬Ö¬π√©√®√ö√•√ô√ß¬Ö√î√¢√ò¬Ö¬π√ù√
¢√Ü√º√ó√ù√Ü√ü¬î¬∫√ö√ß√©√¶√ä√¶¬î√
´√é√ü√†¬î√ä√ò¬î√ó√î√º√ß√ù√ä√ò√¶√ô√Ç¬ì√†√ï√ô√ò√
¶¬¢

7.3 Kasus khusus (String berulang)

Masukan:

hahahahahahahahahahahahahahahahahaha

Kunci: test

Keluaran 7th Sword Vigenere

√±
√¨¬≠√£¬≠√ª
√≤√≤¬¶√π¬ò√ù¬¨√≤¬Æ√¨
√≥¬ó√∏¬ß√±
√¨¬≠√£¬≠√ª

Keluaran Vigenere Cipher

√ú√Ü√õ√ï√ú√Ü√õ√ï√ú√Ü√õ√ï√ú√Ü√õ√ï√ú√Ü√õ√ï√ú√
Ü√õ√ï√ú√Ü√õ√ï√ú√Ü√õ√ï√ú√Ü√õ√ï

7.4 Kasus Khusus (Kunci berulang)

Masukan:

ini coba bandingkan
antara vigenere cipher
dengan 7th sword vigenere

Kunci: hahahahaha

Keluaran 7th Sword Vigenere

x}¬õY¬é¬ö¬õi6j¬ºvs¬õ}v¬ù¬ö¬ô
p}¬¶¬ö¬ù¬åY~o¬•vt¬§t/¬ï¬¢¬õ¬ì¬ûz
st¬†¬†¬å¬ôY?¬äp`{¬Ü¬º¬ÅsR¬Ø¬î¬í¬ûv{z¬•

Keluaran Vigenere Cipher

√ú√Ü√õ√ï√ú√Ü√õ√ï√ú√Ü√õ√ï√ú√Ü√õ√ï√ú√Ü√õ√ï√ú√
Ü√õ√ï√ú√Ü√õ√ï√ú√Ü√õ√ï√ú√Ü√õ√ï

7.5 Kasus Khusus (Kunci adalah String dengan character yang sama)

Masukan:

```
ini coba bandingkan
antara vigenere cipher
dengan 7th sword vigenere
```

Kunci: aaaa

Keluaran 7th Sword Vigenere

```
qvq(kwji(jivlqvosiv
iv|izi(~qomvmzm(kqxpmz
lmvoiv(?|p({ƀwzl(~qomvmzm
```

Keluaran Vigenere Cipher

```
√ä√è√ä¬Å√Ñ√ê√É√Ç¬Å√É√Ç√è√Ö√ä√è√à√å√Ç√è
√Ç√è√ï√Ç√ì√Ç¬Å√ó√ä√à√Ü√è√Ü√ì√Ü¬Å√Ñ√ä√ë√â√
Ü√ì
√Ö√Ü√è√à√à√Ç√è¬Å¬ò√ï√â¬Å√Î√ò√ê√ì√Ö¬Å√ó√ä√à√Ü
√è√Ü√ì√Ü
```

VIII. Analisa Kasus

8.1 String Pendek (lebih kecil dari 100 character)

Dalam kasus ini, terlihat bahwa pola yang dibentuk oleh 7th Sword Vigenere lebih acak dibandingkan dengan pola Vigenere Cipher sehingga 7th Sword Vigenere lebih baik daripada Vigenere Cipher dalam kasus ini.

8.2 String Panjang (lebih dari 100 character)

Dalam kasus ini juga terlihat bahwa pola yang dibentuk oleh 7th Sword Vigenere lebih acak dibandingkan dengan pola Vigenere Cipher sehingga 7th Sword Vigenere lebih baik daripada Vigenere Cipher dalam kasus ini.

8.3 Kasus Khusus (String berulang)

Kasus ini dibuat untuk melihat dengan jelas pola berulang yang dibentuk oleh Vigenere Cipher. Dalam kasus ini, pola yang dibentuk jelas mengingat masukan yang berupa string berulang. Dan di dalam kasus ini 7th Sword Vigenere cukup baik dalam menangani kasus ini

8.4 Kasus Khusus (Kunci berulang)

Kasus ini untuk menguji kunci yang berulang terhadap kinerja 7th Sword Vigenere dan Vigenere Cipher. Dalam kasus ini, pola yang dibentuk kedua algoritma kelihatan memiliki suatu pola yang mirip.

8.5 Kasus Khusus (Kunci adalah String dengan character yang sama)

Dalam kasus ini, terlihat bahwa pola yang dibentuk oleh 7th Sword Vigenere dan Vigenere Cipher sama. Hal itu terjadi karena tidak adanya penanganan untuk kunci yang sama di dalam 7th Sword Vigenere sehingga pola yang terbentuk sama.

IX. Kesimpulan

7th Sword Vigenere memiliki kelebihan untuk penyembunyian pola dengan teknik permutasi dan prinsip confusing dibandingkan dengan Vigenere Cipher, namun hal itu tidak berlaku jika kunci adalah suatu string dengan character yang sama.

X. Daftar Pustaka

1. www.informatika.org/~rinaldi (tanggal akses: 13 Maret 2011) sub HTML: Kriptografi
2. http://java.sun.com/javase/technologies/desktop/javahelp/ (tanggal akses: 14 Maret 2011)

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, 16 Maret 2011

Robert Gunawan / 13508038