Pengembangan Vigenere Cipher melalui Pergeseran Karakter

Ezra Hizkia Nathanael (13510076)[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]*ezra.hizkia@itb.ac.id*

Makalah ini disusun sebagai salah satu prasayart kelulusan mata kuliah Kriptografi (IF3058) bagi mahasiswa program studi Teknik Informatika, Institut Teknologi Bandung semester genap tahun ajaran 2012/2013. Makalah ini digunakan sebagai pengganti nilai UTS. Tema utama dari makalah ini adalah perbaikan atau pengembangan dari sebuah algoritma kriptografi klasik yakni Vigenere Cipher. Tujuan dari perbaikan algoritma ini adalah untuk mempersulit proses kriptanalisis. Dengan demikian, pada makalah ini akan mencantumkan mulai dari proses pembuatan algoritma hingga pengujian dengan kriptanalisis untuk algoritma Vigenere Cipher yang umum dan yang telah diperbaiki.Urutan penyusunan makalah ini terdiri dari Pendahuluan, Dasar Teori, Perancangan dan Implementasi, Pengujian, dan Kesimpulan.

***Index Terms*—applet, character shifting, classic cipher, kriptografi, vigenere cipher**

I. PENDAHULUAN

Dewasa ini, banyak sekali algoritma-algoritma kriptografi yang beredar, dan kebanyakan dari antaranya merupakan algoritma modern berbasiskan bit per bit. Perlu diingat bersama bahwa awal kemunculan kriptografi berasal dari algoritma-algoritma sederhana seperti Caesar Cipher, Vigenere Cipher, dan Playfair Cipher. Algoritma-algoritma kriptografi semacam ini pada umunmya telah obsolete dan sudah tidak digunakan kembali. Beberapa alasannya dikarenakan sudah banyak metode kriptanalisis yang beredar untuk memecahkan sandi-sandi tersebut, dan tentu saja dengan adanya kemunculan komputer. Dengan menggunakan komputer, perhitungan untuk menemukan baik panjang suatu kunci maupun kunci itu sendiri tidaklah membutuhkan waktu yang lama. Mungkin pada zaman dahulu di mana komputer masih sangat sederhana, dibutuhkan waktu dalam orde minggu untuk dapat memecahkan suatu sandi yang panjangnya dua halaman kertas. Kini, hanya dibutuhkan beberapa menit untuk mendekripsi total pesan dengan panjang berhalaman-halaman.

Akan tetapi, harus diakui bahwa algoritma-algoritma klasik semacam ini sederhana sehingga mudah untuk diimplementasikan. Basis yang digunakan pada algoritma klasik adalah karakter, sehingga setiap operasi yang terjadi dilakukan untuk karakter per karakter.

Salah satu tujuan dari penulis untuk kembali mengangkat topik mengenai Vigenere Cipher berawal dari Tugas Kecil 1 dan Tugas Kecil 2 yang diberikan pada mata kuliah IF3058 Kriptografi ini. Pada Tugas Kecil 1 para peserta mata kuliah diminta untuk mengimplementasikan algoritma Vigenere Cipher ke dalam suatu applet dan dapat digunakan baik untuk mengenkripsi maupun mendekripsi pesan. Pada Tugas Kecil 2, peserta diberikan suatu ciphertext dan diminta untuk melakukan kriptanalisis terhadap ciphertext tersebut. Ciphertext ini merupakan suatu plaintext yang disandikan dengan menggunakan Vigenere Cipher. Berangkat dari dua hal inilah penulis ingin mencoba untuk kembali 'mengoprek' algoritma Vigenere Cipher dengan tujuan untuk mempersulit proses kriptanalisisnya, di mana di sini digunakan metode Kasiski. Pada akhir dari penelitian singkat ini diharapkan akan diperoleh kesimpulan apa saja dampak dari perbaikan yang telah dilakukan terhadap algoritma Vigenere Cipher ini.

II. DASAR TEORI

A. Vigenere Cipher

Vigenere Cipher merupakan salah satu algoritma kriptografi klasik. Algoritma pada kategori ini berbasiskan komputer, dan termasuk ke dalam kriptografi kunci-privat atau kunci-simetri. Lebih lanjut lagi, Vigenere Cipher ini termasuk ke dalam algoritma yang menggunakan cipher substitusi, artinya penyandian pesan dilakukan dengan mengubah setiap karakter yang ada dengan menyesuaikan kunci yang digunakan. Cipher substitusi sendiri ada beberapa jenis seperti cipher abjad-tunggal (monoalphabetic cipher), cipher substitusi homofonik (homophonic substitution cipher), cipher abjad-majemuk (polyalphabetic substitution cipher), dan cipher substitusi poligram (polygram substitution cipher). Pembahasan dari keempat cipher ini tidak akan dijelaskan karena berada di luar cakupan makalah ini. Adapun Vigenere Cipher termasuk ke dalam cipher abjad-majemuk (polyalphabetic substitution cipher). Ciri khas pada cipher abjad-majemuk adalah setiap huruf akan disandikan dengan menggunakan kunci yang berbeda, sehingga belum tentu satu huruf akan bersubstitusi menjadi satu huruf yang sama secara terus-menerus. Karakter utama dari ciphertext adalah $c_i(p) = (p + k_i) \bmod 26$. Kunci yang digunakan akan diulang secara periodik.

B. Metode Kasiski

Metode ini merupakan suatu metode atau rangkaian langkah-langkah untuk melakukan kriptanalisis, yakni untuk memecahkan suatu ciphertext (mendekripsi) menjadi plaintextnya. Kriptanalisis sendiri memiliki beberapa metode, salah satu yang digunakan dalam memecahkan sandi untuk algoritma cipher abjad tunggal adalah adanya metode analisis frekuensi. Analisis frekuensi, sesuai namanya, bertujuan menemukan banyaknya kemunculan suatu karakter di dalam ciphertext. Analisis frekuensi ini tidak hanya terbatas kepada satu karakter saja (monogram), akan tetapi dapat juga dinyatakan dalam bigram, trigram, dan seterusnya.

Metode Kasiski juga menggunakan analisis frekuensi di dalam salah satu langkahnya. Nama Kasiski ini sendiri diperoleh dari nama Friedrich Kasiski, yakni orang yang pertama kali memecahkan Vigenere Cipher. Metode Kasiski ini akan bekerja lebih baik pada plaintext yang berasal dari bahasa Inggris, karena metode Kasiski banyak menggunakan analisis frekuensi untuk bigram dan trigram. Dalam bahasa Inggris banyak digunakan bigram dan trigram seperti TH, THE, dan sebagainya.

Langkah-langkah dalam menerapkan metode Kasiski adalah:

1.) Temukan kriptogram yang berulang. Bisa ditemui pada pesan-pesan yang cukup panjang

2.) Hitung jarak antara kriptogram yang berulang

3.) Hitung faktor pembagi dari jarak tersebut. Digunakan untuk memperkirakan panjang kunci

4.) Tentukan irisan dari himpunan faktor pembagi tersebut.

Metode Kasiski ini akan digunakan kemudian ketika kita akan menguji algoritma baru yang telah disusun (perbaikan Vigenere Cipher), yakni mencoba untuk mendekripsi suatu pesan yang sudah ada.

III. PERANCANGAN DAN IMPLEMENTASI

A. Skema Umum Algoritma Vigenere Cipher

Vigenere Cipher merupakan algoritma kriptografi klasik dengan skema umum. Enkripsi :

$$C_i = E_K(P_i) = (P_i + K_i) \bmod 26$$

Dengan demikian, maka skema dekripsinya adalah:

$$P_i = D_K(C_i) = (C_i - K_i) \bmod 26$$

Di mana $P = P_0,\ldots,P_n$ merupakan plaintext, $C = C_0,\ldots,C_n$ merupakan ciphertext, dan $K = K_0,\ldots,K_n$ merupakan kunci. Adapun E_K menandakan fungsi enkripsi dengan kunci K, sedangkan D_K menandakan fungsi dekripsi dengan kunci K.

Kita ambil sebuah contoh. Anggaplah terdapat suatu pesan dengan isi: “MAKAN NASI” dan kunci yang digunakan adalah “AYAM”. Maka proses yang dilakukan adalah :

$C_0 = (P_0 + K_0) \bmod 26 = (12 + 0) \bmod 26 = 12 = \text{‘M’}$
$C_1 = (P_1 + K_1) \bmod 26 = (0 + 24) \bmod 26 = 24 = \text{‘Y’}$
$C_2 = (P_2 + K_2) \bmod 26 = (10 + 0) \bmod 26 = 10 = \text{‘K’}$
$C_3 = (P_3 + K_3) \bmod 26 = (0 + 12) \bmod 26 = 12 = \text{‘M’}$

Selanjutnya, karena panjang kunci hanya 4 karakter, maka kunci digunakan kembali. Dalam artian kata AYAM ini akan diulang kembali untuk seluruh sisa plaintext. Spasi tidak digunakan atau dihiraukan di dalam perhitungan.

…
$C_9 = (P_9 + K_9) \bmod 26 = (8 + 0) \bmod 26 = 8 = \text{‘I’}$

Dan dengan demikian diperoleh ciphertext “MYKMNLAEI” sebagai hasil dari enkripsi tersebut. Biasanya penyajian ciphertext adalah dalam blok per 5 karakter (MYKMN LAEI …) akan tetapi hal ini bukanlah suatu kewajiban.

Untuk proses dekripsi sendiri cukup dengan menggunakan skema dekripsi:

P0 = (C0 – K0) mod 26 = (12 - 0) mod 26 = 12 = ‘M’
P1 = (C1 – K1) mod 26 = (24 - 24) mod 26 = 0 = ‘A’
P2 = (C2 – K2) mod 26 = (10 - 0) mod 26 = 10 = ‘K’
P3 = (C3 – K3) mod 26 = (12 - 12) mod 26 = 0 = ‘A’
…
P9 = (C9 – K9) mod 26 = (8 - 0) mod 26 = 8 = ‘I’

Maka akan diperoleh kembali plaintext bertuliskan “MAKANNASI”.

Apabila algoritma Vigenere Cipher ini dituliskan dalam bentuk notasi algoritmik maka akan menjadi:

```
function VigenereEncrypt (String pesan, String kunci) → String hasil

variabel
i,j,k,pjg_kunci, pjg_teks : int
algoritma

pjg_kunci ← GetPanjang(kunci)
repeat
  if (bukan karakter alfabet) then do nothing
  hasil[k] ← (pesan[i] + kunci[j]) mod 26
  i ← i + 1
  j ← (j + 1) mod pjg_kunci
  k ← k + 1
until i = k = pjg_teks
→ hasil
```

Kode yang ditulis dalam bahasa Java adalah sebagai berikut:

```
String pesan = jTextArea1.getText();
String kunci = jTextField1.getText();
String hasil = "";

pesan = pesan.toUpperCase();
for (int i=0,j=0 ; i <
pesan.length() ; i++) {
  char c = pesan.charAt(i);
  if (c < 'A' || c > 'Z') continue;
  hasil += (char)((c + kunci.charAt(j)
- 2 * 'A') % 26 + 'A');
  j = ++j % kunci.length();
}
jTextArea2.setText(hasil);
```

Sedangkan untuk proses dekripsi, notasi algoritmiknya adalah:

```
function VigenereDecrypt (String pesan, String kunci) → String hasil

variabel
i,j,k,pjg_kunci, pjg_teks : int
algoritma

pjg_kunci ← GetPanjang(kunci)
repeat
  if (bukan karakter alfabet) then do nothing
  hasil[k] ← (pesan[i] - kunci[j]) mod 26
  i ← i + 1
  j ← (j + 1) mod pjg_kunci
  k ← k + 1
until i = k = pjg_teks
→ hasil
```

Kode untuk dekripsi yang ditulis dalam bahasa Java adalah:

```
String pesan = jTextArea1.getText();
String kunci = jTextField1.getText();
String hasil = "";

pesan = pesan.toUpperCase();
for (int i=0,j=0 ; i <
pesan.length() ; i++) {
    char c = pesan.charAt(i);
    if (c < 'A' || c > 'Z') continue;
    hasil += (char)((c -
kunci.charAt(j) + 26) % 26 + 'A');
    j = ++j % kunci.length();
}
jTextArea2.setText(hasil);
```

B. Perubahan Algoritma dengan Penggeseran Karakter

Ide dari penggeseran karakter ini sebetulnya sederhana. Kita telah ketahui dari dasar teori bahwa metode Kasiski sebetulnya lebih berpegang kepada analisis frekuensi dari bigram dan trigram (khususnya untuk plaintext yang berasal dari bahasa Inggris). Oleh karena itu, langkah untuk 'mengacaukan' metode Kasiski ini adalah dengan melakukan pergeseran karakter sejauh satu karakter setiap karakter kelipatan dua atau kelipatan tiga. Adapun pergeseran ini sendiri dibagi menjadi dua, jika kelipatan empat maka digeser mundur sejauh satu karakter, sedangkan jika hanya kelipatan dua saja maka pergeseran dilakukan maju sejauh satu karakter. Untuk yang kelipatan tiga, apabila kelipatan enam maka akan digeser mundur satu karakter, sedangkan jika hanya kelipatan tiga saja akan digeser maju satu karakter.

Dalam notasi algoritmik, fungsi enkripsi-2 akan menjadi:

```
function VigenereEncrypt2 / VigenereEncrypt3 (String pesan, String kunci) → String hasil

variabel
i,j,k,pjg_kunci, pjg_teks : int
hasil_temp : string

algoritma

pjg_kunci ← GetPanjang(kunci)
pjg_teks ← GetPanjang(pesan)
repeat
  if (bukan karakter alfabet) then do nothing
  hasil_temp[k] ← (pesan[i] + kunci[j]) mod 26
  i ← i + 1
  j ← (j + 1) mod pjg_kunci
  k ← k + 1
until i = k = pjg_teks

hasil ← hasil_temp

repeat
  if (kelipatan 4) (kelipatan 6) then
    hasil[i] ← hasil[i] – 1
  else if (kelipatan 2) (kelipatan 3) then
    hasil[i] ← hasil[i] + 1
  i ← i + 1
until i = GetPanjang(hasil)
```

Cara membaca notasi di atas adalah, untuk fungsi Vigenere Cipher pengacak bigram (kelipatan 2) bacalah yang diberi warna merah, sedangkan untuk fungsi Vigenere Cipher pengacak trigram (kelipatan 3) bacalah yang diberi warna biru.

Adapun untuk proses dekripsi, notasi algoritmiknya adalah sebagai berikut:

```
function VigenereDecrypt2 / VigenereDecrypt3 (String pesan, String kunci) → String hasil

variabel
i,j,k,l,pjg_kunci, pjg_teks : int
algoritma

pjg_kunci ← GetPanjang(kunci)
pjg_teks ← GetPanjang(pesan)

repeat
  pesan_temp ← pesan
  if (kelipatan 4) (kelipatan 6) then
    pesan_temp[l] ← pesan_temp[l] + 1
  else if (kelipatan 2) (kelipatan 3) then
    pesan_temp[l] ← pesan_temp[l] – 1
  l ← l + 1
until l = GetPanjang(Pesan)

repeat
  if (bukan karakter alfabet) then do nothing
  hasil[k] ← (pesan_temp[i] - kunci[j]) mod 26
  i ← i + 1
  j ← (j + 1) mod pjg_kunci
  k ← k + 1
until i = k = pjg_teks
→ hasil
```

Adapun kode untuk enkripsi-2, enkripsi-3, dekripsi-2, dan dekripsi-3 secara berurutan, dinyatakan dalam bahasa Java, adalah sebagai berikut:

```
String pesan = jTextArea1.getText();
String kunci = jTextField1.getText();
String hasil = "";
String hasil2 = "";

pesan = pesan.toUpperCase();
for (int i=0,j=0 ; i <
pesan.length() ; i++) {
      char c = pesan.charAt(i);
      if (c < 'A' || c > 'Z') continue;
      hasil += (char)((c +
kunci.charAt(j) - 2 * 'A') % 26 +
'A');
      j = ++j % kunci.length();
}

for (int k=0 ; k < hasil.length() ;
k++) {
   if ((k%2 == 0)&&(k%4==0)) {
   hasil2 += (char)(hasil.charAt(k) -
1);
  } else if ((k%2 == 0)&&(k%4!=0)) {
 hasil2 += (char)(hasil.charAt(k) + 1);
 } else
 {
 hasil2 += (char)(hasil.charAt(k));
 }
}

jTextArea2.setText(hasil2);
```

```
String pesan = jTextArea1.getText();
String kunci = jTextField1.getText();
String hasil = "";
String hasil2 = "";

pesan = pesan.toUpperCase();
for (int i=0,j=0 ; i <
pesan.length() ; i++) {
  char c = pesan.charAt(i);
  if (c < 'A' || c > 'Z') continue;
  hasil += (char)((c + kunci.charAt(j)
- 2 * 'A') % 26 + 'A');
  j = ++j % kunci.length();
  }

for (int k=0 ; k < hasil.length() ;
k++) {
  if ((k%3 == 0)&&(k%6==0)) {
    hasil2 += (char)(hasil.charAt(k) -
1);
  } else if ((k%3 == 0)&&(k%6!=0)) {
    hasil2 += (char)(hasil.charAt(k) +
1);
  } else {
  hasil2 += (char)(hasil.charAt(k));
  }
}

jTextArea2.setText(hasil2);
```

```
String pesan = jTextArea1.getText();
String kunci = jTextField1.getText();
String hasil = "";
String pesan2 = "";

pesan = pesan.toUpperCase();
for (int k=0 ; k < pesan.length() ;
k++) {
  if ((k%2 == 0)&&(k%4==0)) {
  pesan2 += (char)(pesan.charAt(k) +
1);
  } else if ((k%2 == 0)&&(k%4!=0)) {
  pesan2 += (char)(pesan.charAt(k) -
1);
  } else {
  pesan2 += (char)(pesan.charAt(k));
  }
}

for (int i=0,j=0 ; i <
pesan2.length() ; i++) {
  char c = pesan2.charAt(i);
```

```
  if (c < 'A' || c > 'Z') continue;
  hasil += (char)((c - kunci.charAt(j)
+ 26) % 26 + 'A');
  j = ++j % kunci.length();
  }

jTextArea2.setText(hasil);
```

```
String pesan = jTextArea1.getText();
String kunci = jTextField1.getText();
String hasil = "";
String pesan2 = "";

pesan = pesan.toUpperCase();
for (int k=0 ; k < pesan.length() ;
k++) {
  if ((k%3 == 0)&&(k%6==0)) {
  pesan2 += (char)(pesan.charAt(k) +
1);
  } else if ((k%3 == 0)&&(k%6!=0)) {
    pesan2 += (char)(pesan.charAt(k) -
1);
    } else {
      pesan2 += (char)(pesan.charAt(k));
      }
}

for (int i=0,j=0 ; i <
pesan2.length() ; i++) {
  char c = pesan2.charAt(i);
  if (c < 'A' || c > 'Z') continue;
  hasil += (char)((c - kunci.charAt(j)
+ 26) % 26 + 'A');
  j = ++j % kunci.length();
}

jTextArea2.setText(hasil);
```

C. Perancangan Applet

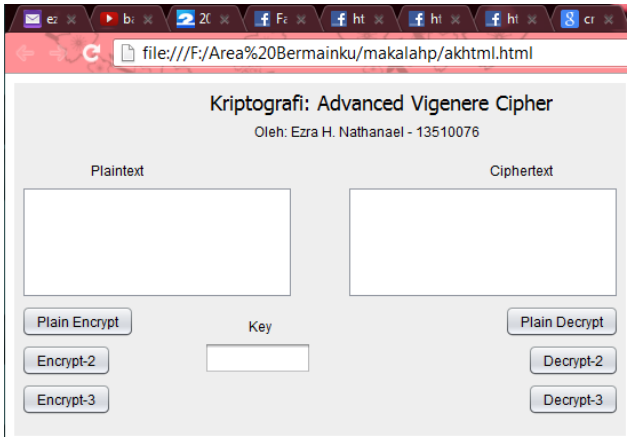
Keseluruhan rancangan kode program ini disusun ke dalam suatu applet Java, sekaligus merupakan perbaikan dan penyempurnaan dari Tugas Kecil 1 Kriptografi.

File yang ditulis dalam bahasa Java bernama appletkripto01.java, sehingga class yang terbentuk adalah appletkripto01.class. Untuk file html yang digunakan untuk memanggil applet sendiri dinyatakan sebagai berikut:

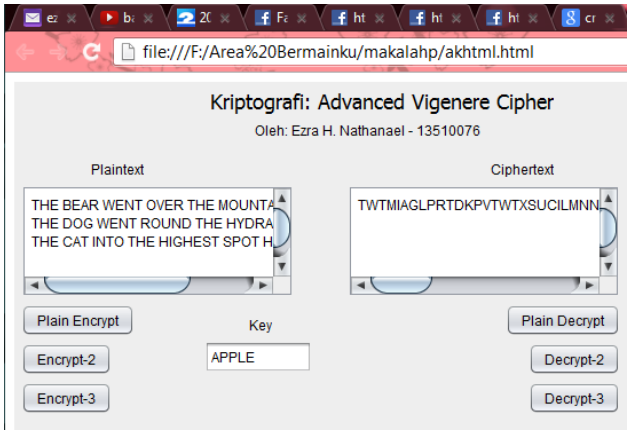
```
<html>
<head>
<title>Vigenere Cipher</title>
</head>
<body>
<applet code="appletkripto01.class"
width="535" height="308"></applet>
</body>
</html>
```

D. Implementasi Program Keseluruhan

Berikut ini merupakan hasil tampilan akhir dari applet:



Sedangkan apabila diuji coba, maka tampilannya akan menjadi:



IV. PENGUJIAN

Pada bagian pengujian ini, akan diuji apakah dengan dua mode enkripsi baru yang sudah disusun, pesan menjadi lebih sulit untuk dipecahkan dengan kriptanalisis menggunakan metode Kasiski. Pesan yang akan digunakan sepanjang pengujian ini diambil dari slide perkuliahan, yakni berbunyi:

> "THE BEAR WENT OVER THE MOUNTAIN YEAH
> THE DOG WENT ROUND THE HYDRANT
> THE CAT INTO THE HIGHEST SPOT HE COULD FIND "

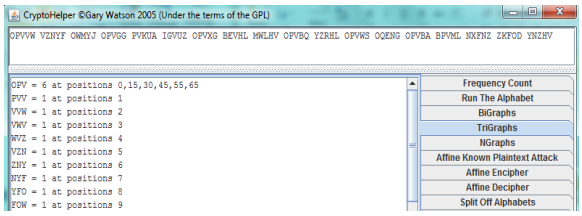
Sedangkan untuk kunci, disepakati bersama di sini bahwa kunci yang digunakan sepanjang lima karakter, yakni 'VIRUS'.

Mari kita coba untuk mengenkripsi pesan di atas. Ciphertext yang dihasilkan adalah:

OPVVW VZNYF OWMYJ OPVGG PVKUA IGVUZ
OPVXG BEVHL MWLHV OPVBQ YZRHL OPVWS
OQENG OPVBA BPVML NXFNZ ZKFOD YNZHV

Sekarang mari kita coba melakukan kriptanalisis dengan metode Kasiski.

Langkah pertama, kita cari kriptogram yang paling sering muncul. Dengan menggunakan kakas bernama Crypto Helper diperoleh:



Trigram yang paling sering muncul adalah trigram OPV, yakni sebanyak enam buah. Trigram OPV ini muncul pada posisi 0, 15, 30, 45, 55, dan 65. Dapat dilihat di sini bahwa jaraknya adalah 10 atau 15 karakter. Faktor pembagi 10 adalah {2,5,10}, sedangkan faktor pembagi 15 adalah {3,5,15}. Jadi irisannya adalah 5, sehingga kemungkinan besar panjang kunci adalah 5 karakter.

Langkah selanjutnya, kelompokkan pesan, dan cari 5 huruf paling sering muncul. Cara mengelompokkannya adalah dengan mengelompokkan menjadi satu karakter-karakter yang terletak pada posisi dengan kelipatan yang sama. Karena dugaan kita panjang kunci adalah 5, maka kita kelompokkan setiap kelipatan 5.

OVOOP IOBMO YOOOB NZY
PZWPV GPEWP ZPQPP XKN
VNMVK VVVLV RVEVV FFZ
VYYGU UXHHB HWNBM NOH
WFJGA ZGLVQ LSGAL ZDV

Di kelompok pertama, huruf paling sering muncul adalah ‘O’. Kelompok kedua adalah ‘P’. Kategori tiga adalah ‘V’. Kategori empat ‘H’. Untuk kategori lima ada dua opsi yakni ‘G’ atau ‘L’. Karena dugaan kita trigram OPV berkorelasi dengan trigram THE, maka kita substitusikan O dengan T, P dengan H, dan V dengan E. Maka kita memperoleh:

T + x = O
H + y = P
E + z = V

Maka kita duga bahwa kuncinya diawali huruf ‘V-I-R’ karena x digantikan dengan V, y digantikan dengan I, dan z digantikan dengan R. Selanjutnya kita tahu bahwa kuncinya adalah ‘VIRUS’ dan kita bisa memperoleh plaintext sesuai dengan yang sudah kita sepakati di awal.

Selanjutnya, marilah kita mencoba melakukan metode Kasiski untuk mode enkripsi-2 dan enkripsi-3.

Untuk metode enkripsi-2, kita memperoleh ciphertext:

NPWVV V[NXF PWLYK OOVHG OVLU@ IHVTZ
PPUXH BDVIL LWMHU OQVAQ ZZQHM OOVXS
NQFNF OQVAA CPUMM NWFOZ YKGOC YOZGV

Sedangkan untuk metode enkripsi-3, maka ciphertext yang diperoleh adalah:

NPVWW VYNYG OWLYJ PPVFG PWKU@ IGWUZ
NPVYG BDVHM MWKHV PPVAQ Y[RHK OPWWS
NQEOG OOVBB BPUML OXFMZ ZLFOC YN[HV

Di sini kita akan mencoba untuk melakukan metode Kasiski. Perlu diperhatikan di sini muncul beberapa karakter noise seperti ‘[‘ dan ‘@’. Ini disebabkan pada tabel ASCII, nilai ASCII ‘@’ adalah 64 (satu sebelum ‘A’) dan nilai ‘[‘ adalah 91 (satu setelah ‘Z’). Munculnya dua karakter ini dikarenakan ketidaksempurnaan pada algoritma shifting.

Kita mencoba melakukan metode Kasiski untuk hasil dari enkripsi-3. Trigram yang paling sering muncul adalah NPV (2 kemunculan pada posisi 0 dan 29) serta PPV (pada posisi 15 dan 44). Selisih di antara keduanya adalah 29, sehingga berdasarkan prinsip Kasiski, maka kemungkinan panjang kunci yang ada adalah 29 karakter. Dari tahap ini saja sudah nampak bahwa metode Kasiski sudah mulai ‘kacau’.

Untuk mode enkripsi-2, trigram yang paling sering muncul adalah OOV, OQV, dan QVA. Masing-masing dengan hanya 2 kemunculan. OOV muncul pada posisi 14 dan 53. OQV pada posisi 43 dan 63, sedangkan QVA pada posisi 44 dan 64. Selisihnya adalah 39, 30, dan 20. Di sini tidak ada faktor yang beririsan sehingga penentuan pengiraan panjang kunci menjadi mustahil untuk dilakukan.

Akan tetapi, kedua teks ciphertext ini masih bisa dikembalikan menjadi plaintext semula dengan menggunakan kunci yang bersesuaian (‘VIRUS’). Di sini fungsi dekripsi menggunakan fungsi dekripsi yang telah dirancang sebelumnya, dan dapat menampilkan plaintext yang persis sama tanpa adanya noise.

V. KESIMPULAN

Kesimpulan yang diambil setelah riset sederhana ini adalah, metode Kasiski yang ampuh untuk memecahkan Vigenere Cipher ternyata dapat ‘diakali’ dengan melakukan shifting karakter. Shifting di sini efektif untuk kelipatan 2 dan 3 karena metode Kasiski menggunakan analisis frekuensi, khususnya untuk bigram dan trigram. Secara algoritma dan kode tidak ada masalah, fungsi

enkripsi dan dekripsi berjalan dengan baik, akan tetapi perlu diperhatikan batasan dari karakter dalam ASCII karena dalam implementasi ini masih ada kekurangan sehingga muncul karakter seperti ‘@’ dan ‘[‘ (penjelasan ada pada bab IV akhir). Adapun tujuan dari penelitian ini (untuk mempersulit pemecahan Vigenere Cipher menggunakan metode Kasiski dengan cara shifting karakter) dapat dikatakan berhasil dengan mengambil contoh pada satu sample yang tersedia.

VI. UCAPAN TERIMA KASIH

Ucapan terima kasih disampaikan kepada Tuhan Yang Maha Esa atas segala bimbingan dan penyertaan sepanjang penyusunan makalah ini. Terima kasih kepada kedua orang tua penulis yang setia mendukung dalam doa dan semangat. Terima kasih kepada Bapak Rinaldi Munir selaku dosen pengampu mata kuliah IF3058 Kriptografi, teman-teman Informatika 2010, dan juga seluruh staff unit Genshiken ITB atas dukungannya di dalam segala hal. Terima kasih pula kepada keluarga di PMK ITB yang menjadi sahabat dalam setiap saat.

REFERENSI

[1] Rinaldi Munir, Slide Materi Kuliah IF3058 Kriptografi.
[2] Dewey Taylor, Classical Cryptography. Virginia Commonwealth University
[3] Budi Rahardjo, et al. Mudah Belajar Java.

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, 26 Maret 2013

Ezra Hizkia Nathanael
13510076