Pengembangan Vigenere Cipher menggunakan Deret Fibonacci

Jaisyalmatin Pribadi (13510084)[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13510084@std.stei.itb.ac.id

Algoritma kriptografi sudah banyak digunakan untuk menyamarkan plain text menjadi cipher text sehingga tidak dapat dipahami oleh pihak yang tidak ditentukan dalam mengambil informasi tersebut. Berdasarkan kemajuan dalam dunia kriptografi, terdapat dua jenis kriptografi klasik dan kriptografi modern. Kriptografi klasik merupakan langkah enkripsi terhadap setiap karakter menjadi karakter berbeda sehingga susunan kata yang dimiliki tidak terlihat memiliki arti. Namun kriptografi klasik sudah dapat dikenali dan dipecahkan oleh berbagai metode kriptanalis. Sedangkan kriptografi modern adalah metode enkripsi karakter menjadi susunan *binary* sehingga langkah penyamaran karakter menjadi lebih kompleks dan sulit untuk dipecahkan.

Algoritma kriptografi klasik yang banyak diimplementasikan yaitu *viginere cipher*. Algoritma kriptografi ini merupakan pengembangan lebih lanjut dari algoritma kriptografi yang muncul pertama kali, yaitu *Caesar cipher*. Algortima *vigenere cipher* memiliki keunikan dalam melakukan enkripsi menggunakan *key* yang ada. Namun kriptanalisis Kasiski dapat memecahkan *vigenere cipher*. Pada makalah ini akan dibahas mengenai implementasi deret *Fibonacci* supaya *key* yang dimiliki tidak dapat diprediksi.

***Key words* — kriptografi, vigenere cipher, fibonacci**

I. PENDAHULUAN

Kriptografi merupakan suatu langkah manipulasi *plain text* menjadi *cipher text* sehingga susunan huruf yang memiliki informasi dapat berubah menjadi kumpulan huruf yang tidak memiliki arti. Langkah ini sering digunakan untuk menjaga pesan agar tidak dapat diketahui oleh pihak yang tidak diinginkan. Secara prosedural, pihak yang ingin membaca *cipher text* tersebut harus mengetahui bahwa *cipher text* ini dienkripsi dengan algoritma kriptografi *vigenere cipher* dan mengetahui *key*.

Keberadaan kriptografi klasik hanya dipandang sebagai dasar dari adanya kriptografi modern. Saat ini, algoritma kriptografi klasik sudah tidak lagi digunakan karena alasan keamanan. Seluruh algoritma klasik telah dapat dipecahkan oleh kriptanalisis yang ada, salah satunya adalah kriptanalisis Kasiski yang dapat memecahkan algoritma *vigenere cipher*. Didukung kemunculuan komputer yang dapat melakukan analisis dalam tingkat yang sangat cepat dan menerima *input* yang sangat banyak.

Pada makalah ini, penulis ingin mengangkat kembali keberadaan algoritma kriptografi *vigenere cipher* dengan melakukan manipulasi pada penggunaan *key*. Alasan penulis memiliki fokus pada penggunaan *key* karena pada kriptanalisis yang dilakukan Kasiski, langkah pertama dalam menghancurkan *vigenere cipher* adalah dengan mengetahui *key* yang digunakan. *Key* yang digunakan dalam algoritma ini akan terus digunakan pada huruf berikutnya sehingga terlihat perulangan metode enkripsi di setiap huruf ke-n.

Usulan yang penulis ajukan adalah dengan melakukan enkripsi dengan langkah pergeseran huruf yang tidak membuat suatu perulangan dalam susunan *cipher text*. Karena itulah dibutuhkan suatu aturan lain dalam menentukan langkah pergeseran yang diinginkan. Penulis menggunakan aturan yang ada pada deret matematika *Fibonacci*. Adanya deret yang memang telah ditentukan, pihak yang menerima *cipher text* dapat membaca isi pesan tersebut.

II. DASAR TEORI

A. Vigenere Cipher

*Vigenere Ciph*er merupakan salah satu algoritma kriptografi klasik yang merupakan pengembangan lebih lanjut dari *Caesar cipher*. Algoritma ini termasuk ke dalam algoritma yang menggunakan *cipher substitution*, yaitu suatu langkah enkripsi dengan cara melakukan pergeseran setiap karakter yang ada pada *plain text*. Jumlah pergeseran karakter yang dilakukan ditentukan oleh *key* yang dimiliki.

Berdasarkan jenisnya, *vigenere cipher* merupakan *polyalphabetic substitution cipher* (*cipher* abjad-majemuk). Ciri algoritma kriptografi jenis ini adalah menggunakan kunci yang berbeda sehingga setiap karakter yang sama pada *plain text* tidak akan disubstitusi menjadi karakter yang sama pada *cipher text*.

Pada saat pertama kali ditemukan, langkah dalam melakukan kriptografi *vigenere cipher* harus menggunakan bujursangkar *vigenere cipher* untuk menentukan karakter yang akan dienkripsi atau didekripsi.Namun terdapat cara lain dalam melakukan kriptografi *viginere cipher*, yaitu dengan rumus matematika.

Berikut rumus untuk melakukan:

$C_{(p)} = (p + k_i) \bmod 26$

Keterangan,
$C_{(p)}$: karakter *cipher*
p : karakter *plain*
k_i : karakter ke-i pada *key*

Sedangkan rumus untuk melakukan dekripsi sebagai berikut:

$p = (C_{(p)} - k_i) \bmod 26$

Jika panjang *key* lebih pendek daripada panjang *plain text*, maka *key* akan digunakan secara periodik.

B. Metode Kasiski

Metode ini merupakan suatu langkah dalam melakukan kriptanalis yang ditemukan oleh Kasiski untuk memecahkan kriptografi *vigenere cipher*. Secara umum, metode kriptanalis yang dilakukan Kasiski adalah pengembangan dari analisis frekuensi yang sebelumnya digunakan untuk memecahkan kriptografi *Caesar cipher*. Pegembangan yang dilakukan Kasiski adalah dengan melanjutkan analisis frekuensi yang tidak hanya sebatas monogram (satu karakter), namun hingga bigram, trigram, dan seterusnya.

Langkah analisis frekuensi merupakan sala satu tahapan yang ada pada metode Kasiski dengan mengambil keuntungan pada bigram *plain text* yang sering muncul dalam pesan bahasa Inggris, yaitu bigram TH dan HE. Selanjutnya denga melakukan perpaduan pada analisis frekuensi monogram yang sering muncul pada bahasa Inggris, yaitu monogram E, T, A, O, I, dan N.

Langkah yang dilakukan Kasiski adalah sebagai berikut,

1) Menemukan kriptogram berulang pada *cipher text*.
2) Hitung faktor pembagi dari jarak setiap kriptogram berulang yang ditemukan. Nilai yang ditemukan akan mengindikasikan panjang *key* yang digunakan.
3) Kelompokkan *cipher text* berdasarkan panjang *key* yang telah ditemukan.
4) Tentukan frekuensi monogram terbanyak yang ada di setiap kelompok. Analisis kemungkinan huruf yang dapat disubstitusi dengan analisis kemunculan monogram terbanyak pada *plain text* berbahasa inggris.

Metode Kasiski ini akan digunakan dalam makalah ini untuk menentukan kekuatan dari kriptografi pengembangan *vigenere cipher* yang dirancang. Langkah dekripsi menggunakan metode Kasiski akan digunakan terhadap *cipher text* yang dihasilkan dari pengembangan *vigenere cipher*.

C. Deret Fibonacci

Deret Fibonacci merupakan deret yang ada pada bidang keilmuan matematika. angka selanjutnya dari deret ini merupakan penjumlahan dari dua angka yang ada pada deret sebelumnya. Namun ada pengecualian untuk beberapa kondisi.

Rumus dalam menentukan deret *Fibonacci* sebagai berikut,

$$F(n) = \begin{cases} 0, & \text{jika } n = 0; \\ 1, & \text{jika } n = 1; \\ F(n-1) + F(n-2) & \text{jika tidak.} \end{cases}$$

Deret *fibonacci* yang dihasilkan dari aturan di atas adalah:

0, 1, 1, 2, 3, 5, 8, 13, ..

Namun, pada implementasi deret *Fibonacci* yang akan digunakan, hanya akan menggunakan aturan yang ada pada saat n tidak sama dengan 0 dan 1.

III. PERANCANGAN DAN IMPLEMENTASI

A. Skema Awal Algoritma Vigenere Cipher

Vigenere Cipher memiliki skema umum seperti yang dijelaskan pada sub bab sebelumnya sebagai berikut,

Enkripsi:

$C_{(p)} = (p + k_i) \bmod 26$

Dekripsi:

$p = (C_{(p)} - k_i) \bmod 26$

Dimana p merupakan karakter *plain*, $C_{(p)}$ merupakan karakter *cipher* yang dienkripsi dari karakter *plain*, dan k_i merupakan karakter ke-i pada *key* yang digunakan sebagai aturan pada pergeseran karakter enkripsi. Kedua fungsi tersebut menggunakan modulus 26 karena jumlah karakter alfabet yang berjumlah 26. Setiap karakter yang digunakan dalam skema ini, sebelumnya dilakukan substitusi dengan nilai yang terurut berdasarkan pada karakter alfabet. Representasi nilai dimulai dari karakter A yang memiliki nilai 0 dan karakter Z yang memiliki nilai 25.

Contoh langkah melakukan enkripsi dapat ditunjukkan sebagai berikut. Sebagai contoh, terdapat *plain text* yang berisi “TIDUR”. Kemudian *key* yang digunakan adalah ’”ADA”. Maka proses yang dilakukan adalah,

$C_0 = (P_0 + K_0) \bmod 26 = (19 + 0) \bmod 26 = 19 =$ ‘T’
$C_1 = (P_1 + K_1) \bmod 26 = (8 + 4) \bmod 26 = 12 =$ ‘M’
$C_2 = (P_2 + K_2) \bmod 26 = (3 + 0) \bmod 26 = 3 =$ ‘D’

Namun *plain text* belum seluruhnya terenkripsi. Selanjutnya akan dilakukan enkripsi pada sisa karakter *plain text* dengan *key* yang diulang dari awal. Sehingga dapat dilanjutkan langkah enkripsi sebagai berikut,

$C_3 = (P_3 + K_0) \bmod 26 = (20 + 0) \bmod 26 = 20 = \text{'U'}$
$C_4 = (P_4 + K_1) \bmod 26 = (17 + 4) \bmod 26 = 21 = \text{'V'}$

Berdasarkan contoh langkah enkripsi yang telah dijabarkan, diperoleh *cipher text* berupa "TMDUV". Sebagai catatan, skema ini tidak melakukan proses terhadap karakter spasi, sehingga pada proses enkripsi dan dekripsi, seluruh isi *plain text* dan *cipher text* akan tersambung menjadi satu. Namun pada langkah akhir enkripsi, umumnya terdapat pilihan untuk membagi *cipher text* menjadi jumlah blok yang diinginkan untuk lebih menghilangkan jejak makna yang terkandung di dalam *cipher text* tersebut.

Contoh berikutnya adalah langkah untuk melakukan dekripsi. Berdasarkan skema umum di atas, langkah dekripsi terhadap *cipher text* contoh dapat dilihat sebagai berikut,

$P_0 = (C_0 - K_0) \bmod 26 = (19 - 0) \bmod 26 = 19 = \text{'T'}$
$P_1 = (C_1 - K_1) \bmod 26 = (12 - 4) \bmod 26 = 8 = \text{'I'}$
$P_2 = (C_2 - K_2) \bmod 26 = (3 - 0) \bmod 26 = 3 = \text{'D'}$
$P_3 = (C_3 - K_3) \bmod 26 = (20 - 0) \bmod 26 = 20 = \text{'U'}$
$P_4 = (C_4 - K_4) \bmod 26 = (21 - 4) \bmod 26 = 17 = \text{'R'}$

Sehingga setelah dilakukan proses dekripsi terhadap *cipher text* yang ada sebelumnya, dapat dilihat sebuah *plain text* yang berisi "TIDUR".

Skema umum *vigenere cipher* dapat ditulis ke dalam notasi algoritma sebagai berikut:

```
function VigenereCipher (String plain, String key) → String result

variabel i, j, key_length : integer

algoritma
key_length ← GetLength(key)
i = 0
j = 0

do
  if (not karakter alfabet) then do nothing

  result[i] ← (plain[i] + key[j]) mod 26
  i ← i + 1
  j ← (j + 1) mod 26
until (i = key_length)

→ result
```

Sedangkan notasi algoritma untuk langkah dekripsi dapat dilihat sebagai berikut:

```
function VigenereDecrypt (String cipher, String key) → String result

variabel i, j, key_length : integer

algoritma
key_length ← GetLength(key)
i = 0
j = 0

do
  if (not karakter alfabet) then do nothing

  result[i] ← (cipher[i] - key[j]) mod 26
  i ← i + 1
  j ← (j + 1) mod 26
until (i = key_length)

→ result
```

B. Pengembangan Algoritma vigenere cipher dengan deret Fibonacci

Ide dari pengembangan algoritma *vigenere cipher* ini dapat dilihat kembali pada pendahuluan, yaitu melakukan pengembangan terhadap *key* yang digunakan dalam melakukan pergeseran karakter saat melakukan enkripsi dan dekripsi. Jika melihat pada pola metode Kasiski, algoritma enkripsi ini dapat dipecahkan karena mengetahui adanya suatu pola secara periodik sehingga dapat ditebak panjang karakter *key*. Dibutuhkan suatu cara yang membuat *key* yang dilakukan untuk melakukan pergeseran tidak dapat ditebak, bahkan tidak dapat diketahui panjang *key* tersebut.

Maka penulis memiliki ide untuk menggunakan deret matematika *Fibonacci* sehingga *key* yang digunakan dalam melakukan pergeseran tidak dapat diketahui panjangnya. Aturan deret *Fibonacci* kedua menjadi ide dari pengembangan ini, yaitu dengan menjumlahkan dua bilangan sehingga menghasilkan *key* yang digunakan dalam melakukan pergeseran. Namun tidak seperti aturan deret *Fibonacci*, penjumlahan yang dilakukan adalah jumlah bilangan pada urutan tersebut dan bilangan setelahnya. Jika telah sampai pada karakter terakhir, penjumlahan akan dilakukan antara bilangan tersebut dengan bilangan yang pertama. Bilangan pertama akan menjadi *key* dalam proses *vigenere cipher* dengan pengembangan deret *Fibonacci*. Alasan tidak menggunakan rumus pada deret *Fibonacci* sercara keseluruhan karena karakter yang akan dienkripsi atau dekripsi dapat berjumlah sangat banyak. Jika *key* yang digunakan untuk melakukan pergeseran karakter mengikut jumlah pada deret *Fibonacci* maka dibutuhkan proses

komputasi yang tinggi. Sehingga dibutuhkan jumlah pergeseran yang tidak bernilai besar, lebih dari seribu pergeseran. Terdapat sedikit perbedaan dalam melakukan enkripsi dan dekripsi pada metode pengembangan *vigener cipher* dengan deret *Fibonacci*.

Terdapat sedikit perbedaan pada langkah yang diperlukan untuk melakukan enkripsi dengan metode pengembangan *vigenere cipher* dengan deret *Fibonacci*. Perbedaan tersebut dapat dilihat bahwa langkah enkripsi tidak membutuhkan *key* dari luar. Berikut rangkaian langkah dalam melakukan enkripsi:

1) Pesan *plain text* dibuat. Bentuk pesan tanpa tanda baca. Pada pengembangan *vigenere cipher* ini, pengguna tidak perlu menyiapkan *key*.
2) Kemudian *plain text* tersebut akan melakukan proses pembuatan *key* yang akan digunakan sebagai dasar dalam melakukan jumlah pergeseran karakter *plain* menjadi karakter *cipher*. Berdasarkan usulan pengembangan *vigenere cipher* yang telah dijelaskan di atas, *key* pada urutan ke-*i* adalah jumlah nilai dari karakter *plain* ke-*i* dan (*i* + 1).
3) Setelah *key* pada urutan ke-*i* telah didapat, maka pergeseran karakter *plain* ke-*i* akan diproses.
4) Pembuatan *key* dan pergeseran pada karakter berikutnya juga melakukan hal yang sama seperti langkah 2 dan 3. Hal tersebut dilakukan hingga mencapai karakter terakhir.
5) Pada karakter terakhir, *key* didapat dengan cara menjumlahkan nilai pada karakter *plain* ke-*i* dan nilai pada karakter pertama pada *plain text*. Setelah itu dilakukan pergeseran karakter tersebut berdasarkan *key* yang didapat.
6) Nilai pada karakter pertama akan dipegang sebagai *key* dalam melakukan dekripsi.

Sedangkan pada langkah dekripsi, metode pengembangan ini tetap membutuhkan masukan *key* untuk melakukan prosesnya. Bentuk dari langkah ini sedikit berbeda dengan langkah enkripsi. Berikut rangkaian langkah dalam melakukan dekripsi:

1) Pertama siapkan *cipher text* yang ingin dilakukan proses dekripsi dan *key* yang akan digunakan. Proses dekripsi pada pengembangan *vigenere cipher* akan dilakukan dari karakter paling akhir hingga mencapai karakter pertama.
2) Langkah dekripsi dimulai dengan mengambil nilai pada karakter *cipher* paling akhir. Kemudian nilai tersebut akan menjadi dasar dalam melakukan pergeseran ke arah sebaliknya dari langkah enkripsi.
3) Nilai yang dihasilkan adalah karakter *plain*. Sebelum memulai proses berikutnya, nilai pada karakter *plain* ini akan disimpan sebagai *key* untuk melakukan dekripsi selanjutnya.
4) Proses dekripsi pada karakter berikutnya dilakukan proses yang sama seperti langkah 2 dan 3. Langkah ini terus dilakukan hingga seluruh *cipher text* berhasil didekripsi.

Bentuk notasi algoritma enkripsi dari ide pengembangan *vigenere cipher* dengan deret *Fibonacci* dapat dilihat sebagai berikut:

```
function NewVigenereCipher (String plain) → String result

variabel
i, key_length : integer

algoritma
key_length ← GetLength(key)
i = 0

do
   if (not karakter alfabet) then do nothing

   if (not i = key_length) then
      result[i] ← (plain[i] + plain[j]) mod 26
   else
      result[i] ← (plain[i] + plain[0]) mod 26

   i ← i + 1

until (i = key_length)

→ result
```

Sedangkan notasi algoritma dari ide pengembangan *vigenere cipher* dengan deret *Fibonacci* untuk langkah dekripsi dapat dilihat sebagai berikut:

```
function NewVigenereDecrypt (String cipher, String key)
→ String result

variabel
i, j: integer

algoritma
i ← GetLength(key)
j = key

do
   if (not karakter alfabet) then do nothing

   result[i] ← (cipher[i] - j) mod 26

   j ← result[i]
   i ← i   - 1

until (i = 0)

→ result
```

IV. IMPLEMENTASI PADA PROGRAM

Pengujian akan penulis lakukan untuk melihat kekuatan pada algoritma pengembangan *vigenere cipher* dengan deret *Fibonacci*. Pengujian akan dilakukan dengan melakukan implementais menggunakan program yang berisi algoritma pengembangan. Selanjutnya pengujian kekuatan dari hasil enkripsi akan dijelaskan pada sub bab berikutnya.

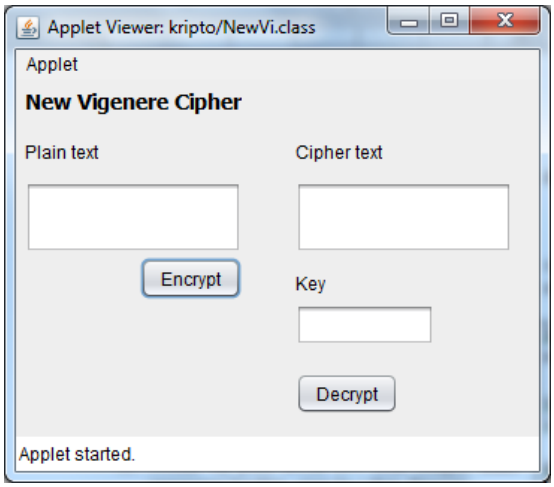
Berikut merupakan implementasi prosedur enkripsi dalam bahasa pemrograman *java*,

```
public static String NewViCi(String p){

        int i, key_length;
        String result="";

        key_length=p.length();
        i=0;

        do{
            if (p.charAt(i) < 'A' || p.charAt(i) > 'Z')
continue;

            if(i!=key_length-1){

result+=IntChar(geser(CharInt(p.charAt(i)),CharInt(p.char
At(i+1))));
            }else{

result+=IntChar(geser(CharInt(p.charAt(i)),CharInt(p.char
At(0))));
            }

            i++;
        }while(key_length!=i);

        return result;
    }
```

Kemudian di bawah ini merupakan implementasi prosedur dekripsi dalam bahasa pemrograman *java*,

```
public static String NewViDe(String p, int key){

        int i,j,k, key_length;
        String result="";

        i=p.length()-1;
        j=key;
        k=0;

        do{
            //if (p.charAt(i) < 'A' || p.charAt(i) > 'Z')
continue;

            k=geserN(CharInt(p.charAt(i)),j);
//System.out.print(i);
            result=IntChar(k)+result;
            j=k;
            i-=1;
        }while(i!=-1);

        return result;
    }
```

Berikut merupakan *screenshot* dari hasil implementasi pengembangan metode *vigenere cipher*,



Untuk menguji implementasi proses enkripsi yang telah dibangun, maka penulis akan memberikan input *plain text* ke dalam program tersebut.

```
THE EVENTUAL EXPLOSION WILL BE THE
FREQUENT BUT LOW LEVEL ACTIVITY OF
VESUVIUS IN RECENT CENTURIES HAS
RELIEVED THE BUILD UP OF PRESSURE IN THE
MAGMA CHAMBER
```

Setelah *plain text* tersedia, penulis akan melakukan proses enkripsi sehingga dihasilkan *cipher text* sebagai berikut,

```
ALIZZRGNULPBMAZGAWBJETWMFXALJWVUKY
RGUVNEZKHPZZPLCVBDDBRMTAZWMPDCMAVE
VGGRGVGRGNLZMWZHSJVPTMZZHWALFVCTOX
JDTUGVWKMLVMVGALQMGSMCJHMNFVK
```

Kemudian penulis melakukan langkah dekripsi menggunakan *cipher text* yang dihasilkan dari langkah di atas.

```
THEEVENTUALEXPLOSIONWILLBETHEFREQUEN
TBUTLOWLEVELACTIVITYOFVESUVIUSINRECEN
TCENTURIESHASRELIEVEDTHEBUILDUPOFPRES
SUREINTHEMAGMACHAMBER
```

Berdasarkan hasil uji coba proses enkripsi dan dekripsi di atas, dapat dikatakan bahwa metode pengembangan *vigenre cipher* menggunakan deret *Fibonacci* berhasil diimplementasi. Algoritma ini dapat dijadikan sebagai algoritma kriptografi alternatif.

V. PENGUJIAN

Terdapat beberapa metode kriptananalisis yang akan dilakukan untuk melakukan pembuktian mengenai kekuatan kriptografi pengembangan ini. Dimulai dari kriptanalisis yang paling sederhana, hingga kriptanalisis yang digunakan dalam memecahkan *vigenere cipher*.

A. Bentuk substitusi huruf

Pengujian pertama akan melihat kembali apakah pengembangan algoritma *vigenere cipher* akan menghilangkan kelebihan *vigenere cipher* yaitu tidak melakukan substitusi karakter *plain* dan *cipher* yang sama.

Dapat dilihat bahwa karakter ketiga dan keempat pada *plain text* merupakan huruf yang sama, yaitu huruf E. namun hasil enkripsi yang dilakukan menunjukkan bahwa huruf E pada karakter ketiga disubstitusi menjadi huruf I dan huruf E pada karakter keempat disubstitusi menjadi huruf Z. Hal ini membuktikan bahwa pengembangan *vigenere cipher* tidak menghilangkan keunggulan utama dari *vigenere cipher*.

B. Panjang key

Langkah awal dalam memecahkan *vigenere cipher* adalah dengan mengetahui *key* yang digunakan. Cara yang biasa digunakan adalah mencari faktor dari kriptogram yang berulang. Nilai faktor tersebut akan mengindikasikan panjang *key*.

Penulis kembali menggunakan CryptoHelper.jar untuk menghitung kriptogram yang berulang. Namun tidak banyak kriptogram yang berulang. Paling tinggi hanya menyimpan dua hingga tiga kriptogram berulang. Hal ini membuktikan bahwa metode Kasiski tidak dapat digunakan dalam memecahkan langkah pengembangan *vigenere cipher*.

C. Bentuk n-gram

Pada bentuk n-gram ini, penulis akan melakukan pengujian pada n dengan nilai 2 dan 3. Kedua nilai tersebut umumnya dijadikan acuan dalam melakukan analisisi n-gram. Umumnya, kriptogram yang berulang mengindikasikan kriptogram yang sering muncul dalam kalimat.

Hasil pada perhitungan jumlah bigram terbanyak adalah bigram RG yang muncul sebanyak empat kali. Menurut teori yang diajarkan, bigram terbanyak umumnya merupakan *cipher text* dari bigram pada *plain text* berupa TH atau HE. Namun sebenarnya dapat dilihat bahwa RG yang muncul pada karakter keenam dan tujuh memiliki *plain text* EN.

Sedangkan untuk trigram dapat dilihat bahwa RGN yang muncul dua kali. Jika dilihat bahwa susunan RGN pada karakter 6, 7, dan 8 memiliki arti *plain* yaitu ENT. Tidak seperti yang dijealskan sebelumnya bahwa trigam terbanyak adalah THE.

VI. KESIMPULAN

Kesimpulan yang diambil setelah riset sederhana yang dilakukan penulis adalah metode Kasiski yang dapat memecahkan *vigenere cipher* dapat diatasi dengan melakukan manipulasi terhadap *key* yang digunakan untuk melakukan pergeseran. Manipulasi *key* ini dapat berjalan efektif karena panjang *cipher text* yang dihasilkan tidak dapat memperlihatkan perulangan periodik yang sebelumnya dignasalakan untuk memecahkan *key* yang digunakan. Hal efektif yang dapat dimanfaatkan pada algoritma pengembangan ini adalah tidak dibutuhkannya *key* dalam melakukan enkripsi. Namun setelah melakukan enkripsi, pengguna harus menyimpan *key*. *Key* tersebut akan digunakan dalam melakukan dekripsi.

Tujuan dari makalah ini untuk mengatasi kriptanalis Kasiski dapat diatasi dengan baik menggunakan algoritma pengembangan *vigenere cipher* menggunakan deret *Fibonacci*. Hal tersebut dapat dilihat pada tahap pengujian di sub bab sebelumnya.

VII. UCAPAN TERIMA KASIH

Ucapan terima kasih penulis sampaikan kepada Allah SWT atas segala rahmat-Nya Makalah ini dapat diselesaikan. Terima kasih kepada kedua orang tua penulis yang setia mendukung dalam doa dan semangat. Terima kasih kepada Bapak Rinaldi Munir selaku dosen mata kuliah IF4020 Kriptografi.

REFERENSI

[1] Rinaldi Munir, Slide Materi Kuliah IF4020 Kriptografi.
[2] Dewey Taylor, Classical Cryptography. Virginia Commonwealth University

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, 17 Maret 2014

Jaisyalmatin Pribadi
13510084