Vigenere Cipher with Dynamic Key

Andrei Dharma Kusuma / 13508009
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
andreidkusuma@yahoo.com

***Abstrak*—Makalah ini akan membahas mengenai salah satu kriptografi substitusi, yaitu vigenere cipher. Dalam makalah ini akan dibahas mengenai vigenere secara sekilas, sedikit mengenai kelebihan dan kelemahan vigenere cipher serta bagaimana cara meningkatkan keamanan vigenere cipher dari sisi keamanan kunci. Dalam makalah ini akan terdapat beberapa cara bagaimana cara memperkuat vigenere cipher dengan kunci yang sederhana.**

I. Pendahuluan

Dewasa ini, informasi atau pesan merupakan sesuatu yang sangat penting dalam kehidupan sehari-hari. Ada pesan yang tidak rahasia, dan sebaliknya pesan rahasia untuk orang-orang tertentu pun ada. Oleh karena itu, dibutuhkan sebuah metode khusus, bagaimana agar pesan tersebut tidak mungkin diketahui orang lain.

Sedikitnya ada dua metode yang digunakan untuk mengamankan pesan, yaitu kriptografi dan steganografi. Kriptografi merupakan teknik pengamanan pesan dengan mengubah pesan yang ada. Kriptografi sederhana yang dikenal ialah Caesar cipher. Caesar cipher merupakan algoritma kriptografi dengan menggeser huruf per huruf sejumlah bilangan tertentu. Sebagai contoh kata 'serang' digeser sejauh 6 karakter menjadi 'ykxgtm'.

Steganografi merupakan proses pengamanan pesan yang sering dikenal dengan "penyembunyian pesan". Cara yang digunakan misalnya sebuah kalimat 'lari , jam satu' dapat disembunyikan menjadi '**L**upakan **a**sal **r**umor **i**tu, **j**aga **a**gar **m**atamu **s**ehat **a**tau **t**urunkan **u**banmu'.

Kriptanalis merupakan ilmu yang berkembang untuk memecahkan algoritma-algoritma yang digunakan dalam kriptografi. Biasanya untuk menangkal hal tersebut, algoritma kriptografi dikembangkan serumit mungkin, hingga kriptanalis sulit memecahkannya. Usaha yang digunakan dalam mengembangkan algoritma ataupun menciptakan algoritma baru cukup besar.

Oleh karena itu terlintas dipikiran penulis mengapa tidak memaksimalkan algoritma kriptografi bukan dari sisi algoritmanya, melainkan dari sisi lain yang membuat algoritma kriptografi yang sudah ada menjadi maksimal. Sebagai contoh bagaimana dengan memaksimalkan algoritma kriptografi dari sisi kunci yang digunakan. Kunci yang baik dalam algoritma kriptografi ialah kunci yang memiliki panjang sepanjang plainteks yang akan di enkripsi. Namun hal ini membutuhkan cost yang cukup besar. Dalam makalah ini, penulis memilih algoritma sederhana yang masih bisa dipecahkan, yaitu vigenere cipher dan mengembangkan algoritma tersebut dari sisi kuncinya.

II. Vigenere Cipher

Vigenere Cipher termasuk dalam cipher abjad-majemuk (polyalphabetic substitution cipher). Algoritma ini dipublikasikan oleh seorang diplomat asal Perancis, yaitu Blaise de Vigenere pada abad 16 (1586). Namun sebenarnya Giovan Batista Belaso telah menggambarkannya pertama kali pada tahun 1553 seperti ditulis di dalam bukunya *La Cifra del Sig. Giovan Batista Belaso.*

Algoritma ini kemudian baru dikenal 200 tahun kemudian dengan nama penemunya yaitu Vigenere Cipher. Algoritma ini namun berhasil dipecahkan oleh Babbage dan Kasiski pada pertengahan abad ke 19. Vigenere Cipher digunakan oleh Tentara Konfiderasi (Confederate Army) pada Perang Sipil Amerika (American Civil War). Perang Sipil ini terjadi setelah Vigenere Cipher berhasil dipecahkan.

Bagaimana cara kerja Vigenere Cipher ?

Vigenere Cipher menggunakan Bujur sangkar Vigenere untuk melakukan enkripsi. Kemudian setiap baris di dalam bujursangkar tersebut menyatakan huruf-huruf cipherteks yang diperoleh dengan Caesar Cipher. Kunci yang digunakan terdiri dari beberapa huruf dimana

K = k1k2…km

Ki untuk 1 <= I <= m menyatakan jumlah pergeseran huruf ke-i. Algoritma selengkapnya dari Vigenere Cipher adalah sebagai berikut :

Ci(p) = (p + ki) mod 26

Jika panjang kunci yang ada lebih pendek daripada jumlah plainteks. Maka kunci diulang secara periodic. Misalkan panjang kunci = 20, maka 20 karakter pertama dienkripsi dengan persamaan diatas, dan setiap karakter ke-I menggunakan kunci ki. Hal yang sama berlaku dengan 20 karakter selanjutnya dimana 20 karakter selanjutnya dienkripsi menggunakan kunci yang sama dengan 20 karakter pertama. Karakter k-21 dienkripsi dengan kunci ke-1, dan selanjutnya.

Gambar 4.2 di bawah ini merupakan contoh bujursangkar Vigenere Cipher yang kita gunakan.

Misalkan kita memiliki contoh sebagai berikut :

Kunci = sony
Plainteks: THIS PLAINTEXT
Kunci: sony sonysonys

Maka yang harus kita lakukan ialah melihat tabel bujursangkar vigenere dan mencocokkan kunci dengan plainteks untuk mendapatkan cipherteks. (Gam.bar 4.3)

Plainteks

Kunci	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

Gambar 4.2 Bujursangkar *Vigènere*

Plainteks

KUNCI	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

Gambar 4.3 Enkripsi huruf `T` dengan kunci `f`

Dengan melihat tabel di atas maka didapatkan hasil enkripsi seluruh plainteks adalah sebagai berikut :

Pada dasarnya, setiap enkripsi huruf adalah Caesar cipher dengan kunci yang berbeda-beda.

Plainteks : THIS PLAINTEXT
Kunci : sony sonysonys
Cipherteks : **LVVQ HZNGFHRVL**

(T + s) mod 26 = L
(H + o) mod 26 = V, dst

Oleh sebab itu huruf yang sama tidak selalu dienkripsi menjadi huruf cipherteks yang sama pula. Hal ini menunjukkan karakteristik dari cipher abjad-majemuk : setiap huruf cipher teks dapat memiliki kemungkinan banyak huruf plainteks. Dimana pada cipher substitusi sederhana, setiap huruf cipherteks selalu menggantikan huruf plainteks tertentu.

Varian Vigenere Cipher

1. Full Vigenere Cipher
Setiap baris dalam tabel tidak menyatakan pergeseran huruf, tetapi merupakan permutasi huruf-huruf alfabet.

2. Auto-Key Vigenere Cipher
Jika panjang kunci lebih kecil dari panjang plainteks, maka kunci disambung dengan plainteks tersebut.

3. Running-Key Vigenere Cipher
Kunci adalah string yang sangat panjang yang diambil dari teks bermakna (misalnya naskah proklamasi, naskah Pembukaan UUD 1945, terjemahan ayat di dalam kitab suci, dan lain-lain).

Serangan Terhadap Vigenere Cipher

Friedrich Kasiski adalah orang yang pertama kali memecahkan Vigenere cipher pada tahun 1863. Metode ini membantu menemukan panjang kunci dari Vigenere cipher. Metode Kasiski ini memanfaatkan keuntungan bahwa bahasa Inggris tidak hanya mengandung perulangan huruf, tetapi juga perulangan pasangan huruf atau tripel huruf seperti TH, THE, dsb. Perulangan kelompok huruf ini dapat menghasilkan kriptogram yang berulang pula.

Contoh dibawah ini memperlihatkan bagaimana pola yang sama terulang dalam cipherteks yang dihasilkan.

Plainteks :
CRYPTO IS SHORT FOR CRYPTOGRAPHY

Kunci :
abcdab cd abcda bcd abcdabcdabcd
Cipherteks :
CSASTP KV SIQUT GQU **CSASTP**IUAQJB

Pada contoh ini, CRYPTO dienkripsi menjadi kriptogram yang sama, yaitu **CSATP**.

Berikut adalah langkah-langkah yang digunakan dalam metode kasiski :

- Cari semua pola berulang dalam cipherteks
- Hitung jarang antara kriptogram yang berulang
- Hitung semua faktor pembagi dari jarak tersebut
- Tentukan irisan dari himpunan factor pembagi tersebut
- Nilai tersebut menyatakan nilai-nilai kemungkinan dari panjang kunci

Analisis Vigenere Cipher

Metode Kasiski merupakan metode yang mencari jumlah panjang kunci dan Vigenere Cipher menjadi mudah dipecahkan. Namun, semakin panjang kunci yang diberikan, apalagi sepanjang plainteks yang ada, maka kemungkinan Vigenere Cipher dipecahkan semakin sulit.

Selanjutnya ialah jika kita menggunakan Vigenere Cipher dan key yang digunakan adalah sepanjang teks yang ada, maka hal ini akan sangat memboroskan, tidak efektif dalam jumlah biaya yang dikeluarkan. Masalah yang kemudian muncul ialah :

- Bagaimana cara menyimpan kunci yang panjang tanpa dapat dicuri orang ?
- Bagaimana cara memindahkan key / key dapat dibuat mobile tanpa dapat disadap orang ?

Oleh karena itu, penulis telah memikirkan sebuah pola yang penulis namakan dengan Dynamic Key.

II. Dynamic Key

Apa itu Dynamic Key? Dynamic key merupakan sebuah algoritma sederhana dalam mengoptimalkan jumlah panjang key yang terbatas. Caranya ialah dengan meng-*generate* key yang panjangnya terbatas tersebut menjadi key yang lebih panjang tanpa perulangan yang signifikan dari key yang terbatas tersebut.

Ada beberapa cara yang penulis sarankan dalam pembuatan dynamic key ini yang merupakan inti dari makalah ini.

1. Caesar Flowing Key Algorithm

Desain Awal

Caesar Flowing Key merupakan sebuah ide dalam meng-*generate* key mirip dengan apa yang dilakukan dalam Caesar Cipher, dan kata flowing mendefinisikan pen-*generate*-an key yang tidak monoton / satu arah, tetapi dinamik / flowing.

Sebagai contoh sebuah kunci = ‘key’ diduplikasi menjadi kunci = ‘key’ + ‘lfz’ yang merupakan hasil Caesar cipher dari kunci awalnya. Algoritma keseluruhan akan dijelaskan dibawah ini dan beberapa istilah yang akan dipakai ialah sebagai berikut :

K = Kunci
Ki = Kunci bagian ke i
P = Panjang Kunci
L = Lompatan karakter

Dimana,

Xi = X dibagi setiap P karakter

Algoritmanya ialah sebagai berikut dengan asumsi key yang digunakan berupa 26 karakter saja:

1. Cipherteks dibagi setiap P karakter dan setiap bagian dipasangkan dengan kunci tertentu untuk bagian tersebut.
2. Pengenkripsian dilakukan tiap P karakter untuk mempermurah biaya dalam enkripsi.
3. Setelah pengenkripsian dilakukan, key diupdate dengan melakukan Caesar cipher pada key dengan pergeseran sepanjang L.
4. Pergeseran tersebut dilakukan sebanyak 25 kali (jumlah karakter -1).
5. Setelah 25 kali pergeseran, maka L ditambahkan 1 dan di mod dengan 26 atau L = (L + 1) mod 26.
6. Perulangan terus dilakukan hingga semua plainteks dienkripsi.

Namun, algoritma ini masih memiliki siklus key yang berulang. Dengan hasil panjang satu siklus key yang baru ialah sebagai berikut:

Panjang 1 siklus =
Panjang key + (25 * 25 * Panjang key)

Dari persamaan diatas memperlihatkan bahwa dengan key yang sederhana, pemecahan vigenere cipher sudah menjadi 626 kali lebih sulit dibandingkan pemecahan vigenere cipher dengan algoritma dan key pada awalnya.

Algoritma di atas memiliki asumsi bahwa karakter yang digunakan dalam enkripsi ialah hanya 26 karakter. Untuk mempersulit pengenkripsian dapat digunakan algoritma Caesar Flowing Key dengan 256 karakter. Sehingga kesulitan Panjang key menjadi 65026 kali panjang key semula.

Implementasi

Fungsi Caesar Cipher

```
public String caesar(String apapun, int geser)
   {

      String hasil ="";
      int dummyint;
      char dummychar[] = new char[apapun.length()];
      String dummy;
      for (int i = 0; i < apapun.length(); i++)
      {
         dummyint = (int)apapun.charAt(i)+ geser;
         if(dummyint > 122)
         {
            dummyint -= 26;
         }
         dummychar[i] = (char)dummyint;
         dummy = Character.toString(dummychar[i]);
         hasil+=dummy;
      }
      return hasil;
   }
```

Fungsi Caesar Flowing Key

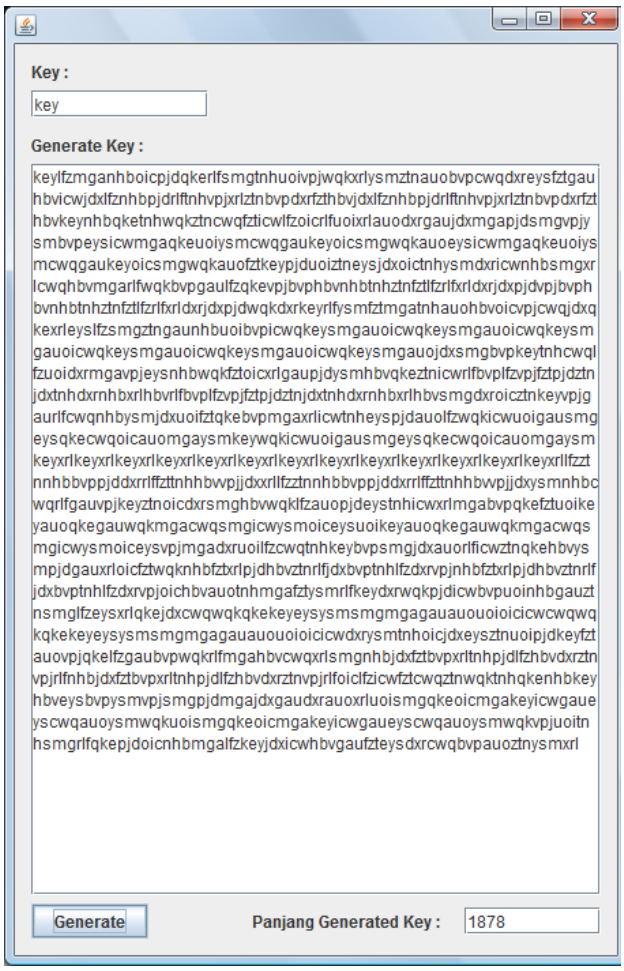
```
public String generatekey(String key){

      String genkey = key;
      int i;
      int j;
      int k = 1;

      for (i = 0; i<25;i++)
      {
         for (j = 0;j<25;j++)
         {
            key = caesar(key,k);
            genkey+=key;
         }
         k++;
      }
      return genkey;
   }
```

Uji Coba

Berikut adalah contoh CFK Algorithm dengan 26 karakter yang penulis buat.



Analisis

Setelah dilakukan pengenkripsian dengan bantuan CryptoHelper.jar dari situs http://informatika.org/~rinaldi, maka plainteks semula yaitu :

KUKU KAKIKU KAKU KAKU SEKAKU KAKUNYA KAKUNYA ITU KUKU KAKIKU KAKU KAKU SEKAKU KAKUNYA KAKUNYA ITU KUKU KAKIKU KAKU KAKU SEKAKU KAKUNYA KAKUNYA ITU

Jika dilakukan enkripsi Vigenere Cipher masih menggunakan program CryptoHelper.jar dengan kunci = "key", didapatkan hasil sebagai berikut :

UYI EOY UMI EOY UYI KOS CII KOS UEI ERW KOY UYL IEG DYI EOS UEI SOS UEI EOY UYQ OOY UYI KOS XCY UEI ERW KMR EOS UYI KOG UYI KOS UEI EWC UEI EOY UYL IEI KOS XCY SXS

Jika key yang digunakan ialah key = key yang sudah digenerate dengan bantuan algoritma Caesar Flowing Key, didapatkan hasil sebagai berikut :

UYI FPZ WOK HRB YCM PTX IOO RVZ CMQ NAF UYI FJW UQS QLV SCG JTX IEI LVZ WGQ NRJ IIS PTD GKO UZV SCG DQV LNS HRV ZDN RVN DHR VZD HRV TLR LVZ XHR PTG FBF JNR YDZ VAV

Perbedaan yang signifikan terjadi pada tingkat keamanan dari vigenere cipher ini. Pada contoh dengan key belum digenerate dengan algoritma CFK, perulangan sangat sering terjadi sehingga dapat dengan mudah ditebak bahwa panjang key adalah 3. Namun setelah digenerate, sepintas mata kita melihat bahwa UYI merupakan perulangan kata. Padahal, jika kita membandingkan dengan plainteks awal, maka dapat dengan jelas terlihat bahwa UYI yang pertama memiliki plainteks yang berbeda dengan UYI yang kedua.

Oleh karena itu, algoritma CFK ini terbukti dapat meningkatkan keamanan vigenere cipher.

2. Neighbor Key Algorithm

Neighbor Key merupakan algoritma yang lebih sederhana namun juga baik dari Caesar Flowing Cipher. Inti dari algoritma ini ialah setiap anggota dari key memiliki pasangan dengan anggota key lainnya.

Algoritma ini mengutamakan perubahan key dengan yang didapat dari perkalian antara key dengan pasangannya.

Sebagai contoh sebuah kunci = 'key' diduplikasi menjadi sebuah kunci = 'key' + '(k*e)(e*y)(y*k)' dimana masing-masing perkalian di-mod-kan dengan 26, sehingga kunci sekarang menjadi 'key' + 'cuo'. Algoritma selengkapnya akan dibahas dibawah dengan beberapa hal yang perlu diketahui sebagai berikut :

K = Key masukkan pengguna
Ki = Key ke-i
Ki-j = Key ke-i dengan indeks karakter ke-j
P = Panjang key masukkan pengguna

Algoritma yang digunakan ialah sebagai berikut dengan asumsi enkripsi 26 karakter:

1. Kunci dimasukkan dalam sebuah senarai / *list.*
2. Masing-masing karakter dalam kunci diberi indeks sesuai urutannya.
3. Key pertama atau K1 ialah key yang dimasukkan oleh pengguna. K1 ini digunakan untuk enkripsi P karakter awal dari plainteks.
4. P karakter selanjutnya dienkripsi dengan menggunakan K2. K2-1 didapatkan dengan mengalikan K1-1 dengan K1-2 kemudian di-mod-kan dengan 26 untuk mendapatkan karakternya. Dan seterusnya hingga K2-n didapatkan dengan mengalikan K1-n dengan K1-1.
5. Pengulangan terus dilanjutkan hingga semua plainteks sudah dienkripsi.
6. Seperti halnya pada Caesar Flowing Key, untuk mempermurah biaya enkripsi, dilakukan setiap P karakter plainteks.

Implementasi

Fungsi Neighbor Key

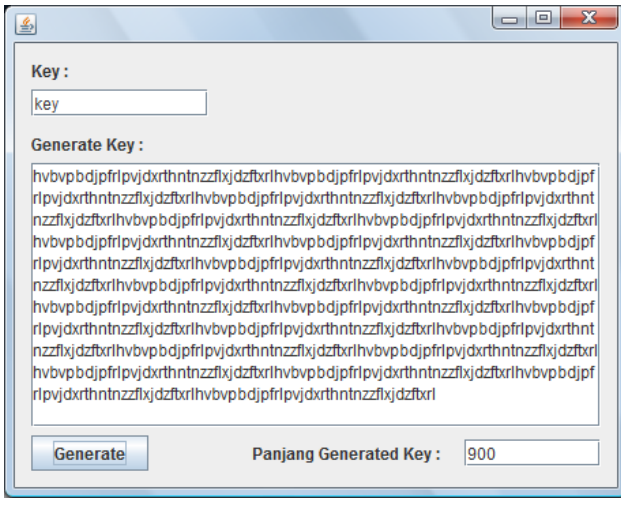
```
public String neighborkey(String key){

        String hasil = "";
        String dummy ="";
        String dummy2 = key;
        int dummyint;
        int pk = key.length();
        for (int i = 0 ; i < 300 ; i++){
           dummy = "";
           for (int j=0; j<pk;j++)
           {
                    dummyint = (int)dummy2.charAt(j) +
(int)dummy2.charAt((j+1)%(pk));
              while(dummyint>122)
              {
                 dummyint-=26;
              }
              dummy += (char)(dummyint);
           }
           dummy2 = dummy;
           hasil += dummy;
        }

        return hasil;
    }
```

Uji Coba

Berikut ini merupakan key yang di-*generate* dengan asumsi plainteks sepanjang 900 karakter.



Analisis

Pada awalnya atau hipotesis penulis awalnya, algoritma ini sangat sederhana namun memiliki kemampuan yang dapat bisa mengcover seluruh panjang key, lain halnya dengan algoritma Caesar Flowing Key. Dengan dilakukannya perkalian antar karakter dalam key, kemungkinan terjadinya seluruh karakter dalam key adalah sama, dengan asumsi saya sendiri, sangat kecil. Oleh karena itu algoritma ini merupakan algoritma yang 'bagus'.

Namun setelah dilakukan implementasi, terlihat dengan jelas bahwa ternyata key hasil *generate* dengan algoritma ini sangat-sangat tidak efektif. Kunci terduplikasi menjadi suatu pola yang sama, yaitu :

HVBVPBDJPFRLPVJDXRTHNTNZZFLXJDZFTXRL

Sehingga dapat dikatakan bahwa algoritma ini adalah algoritma yang gagal dalam memaksimalkan algoritma vigenere cipher. Oleh karena itu, proses analisis tidak saya lanjutkan ke dalam proses vigenere cipher dan analisis saya hentikan sampai sini.

IV. Kesimpulan

Pemaksimalan algoritma Vigenere Cipher dengan menggunakan kunci dinamik ternyata tidak semata-mata menyempurnakan algoritma Vigenere Cipher. Masih terdapat banyak cara untuk melakukan proses dekripsi algoritma Vigenere Cipher yang sudah dimodifikasi dengan kunci dinamik ini.

Algoritma Caesar Flowing Key buatan penulis cukup memudahkan pengguna algoritma Vigenere Cipher dengan hanya memasukkan kunci yang sederhana, namun dapat terduplikasi menjadi kunci yang benar-benar unik sepanjang ratusan hingga ribuan kali lipat panjang kunci awal. Algoritma ini bagus dalam peningkatan keamanan Vigenere Cipher tanpa harus menyulitkan pengguna dalam bertukar kunci.

Algoritma Neighbor Key buatan penulis, merupakan algoritma yang gagal dalam men-*generate* kunci dinamik. Kunci yang dihasilkan tidak lain adalah kunci berpola. Algoritma ini tidak dapat membantu meningkatkan keamanan Vigenere Cipher.

Penulis juga tidak dapat menemukan algoritma yang benar-benar sempurna dalam meng-*generate* kunci hingga benar-benar unik tanpa harus memasukkan kunci yang panjang.

Daftar Pustaka

[1] http://informatika.org/~rinaldi/

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, 23 Maret 2011

Andrei Dharma Kusuma / 13508009