Modifikasi Algoritma Vigenere Cipher Dengan Prinsip Jam

Okharyadi Saputra (13510072)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
okharyadi@gmail.com

***Abstrak*—Algoritma Vigenere Cipher merupakan salah satu algoritma kriptografi klasik yang sudah dikenal luas saat ini. Algoritma ini memanfaatkan prinsip bujursangkar vigenere untuk melakukan enkripsi. Sebenarnya algoritma ini sudah cukup baik dan pernah diterapkan selama beberapa dekade, namun sayangnya pada abad ke 19 algoritma ini berhasil dipecahkan oleh Kasiski dengan metode yang disebut metode Kasiski.**

Dalam makalah kali ini penulis mencoba untuk melakukan modifikasi terhadap algoritma Vigenere Cipher dengan memanfaatkan prinsip jarum jam. Prinsip jarum jam yang dimaksud disini adalah prinsip perputaran jarum jam yang akan berperan sebagai kunci dalam proses enkripsi.

***Kata Kunci*— Kriptografi, Vigenere Cipher, Algoritma, Metode Kasiski**

I. PENDAHULUAN

Vigenere Cipher merupakan sebuah algoritma kriptografi klasik yang cukup mudah untuk diimplementasikan. Algoritma ini bekerja dengan menggunakan sebuah tabel sebagai panduan dalam proses enkripsi.

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 1
Tabel Enkripsi Vigenere Cipher

Misalkan saja untuk mengenkripsi kalimat seperti berikut:

APA KABAR

Jika digunakan kata HAI sebagai kuncinya, maka dengan menggunakan algoritma Vigenere Cipher kalimat tersebut akan dienkripsi menjadi kalimat berikut:

IPJ RAJHR

Jika sekilas dilihat sebenarnya algoritma ini dapat menyembunyikan teks awal dengan cukup baik, namun seiring dengan perkembangan algoritma kriptografi dan proses kriptanalisis, algoritma inipun dianggap sudah tidak lagi aman karena sudah berhasil dipecahkan.

Algoritma Vigenere Cipher dapat dipecahkan dengan menggunakan metode Kasiski. Pemecahan dengan metode Kasiski mengandalkan kelemahan dalam algoritma Vigenere Cipher, yaitu akan terbentuk pengulangan hasil enkripsi setelah selang tertentu, tergantung dari panjang kunci yang digunakan. Hal inilah yang dimanfaatkan dalam metode Kasiski untuk memecahkan algoritma Vigenere Cipher, seperti yang pernah penulis lakukan pada tugas yang sebelumnya.

Dalam kesempatan kali ini, penulis mencoba untuk mengembangkan sebuah algoritma kriptografi baru hasil modifikasi algoritma Vigenere Cipher yang diharapkan dapat menutupi kelemahan algoritma Vigenere Cipher. Algoritma ini penulis sebut sebagai algoritma jam.

II. USULAN ALGORITMA

Algoritma jam adalah algoritma hasil modifikasi algoritma Vigenere Cipher. Berbeda dengan algoritma Vigenere Cipher biasa yang mengandalkan sebuah kata atau kalimat sebagai kunci utamanya, algoritma jam bekerja dengan menggunakan sebuah satuan jam sebagai kunci utamanya.

Seperti yang diketahui, bahwa jam analog memiliki 12 angka didalamnya. Setiap angka tersebut akan mewakili huruf A hingga L. Angka 1 mewakili huruf A, angka 2 mewakili huruf B dan seterusnya. Keunikan dari algoritma ini adalah kunci yang dihasilkan akan otomatis diproses oleh program, sehingga yang perlu dilakukan pembuat pesan adalah menentukan jam serta pola loncatan.

Untuk lebih jelasnya berikut adalah gambaran umum bagaimana algoritma ini akan bekerja:

1. Pembuat pesan haruslah menentukan plain teks yang akan dienkripsi disertai jam sebagai kunci

utama. Misalkan saja dilakukan proses enkripsi untuk kalimat:

TEMUI SAYA DI TAMAN

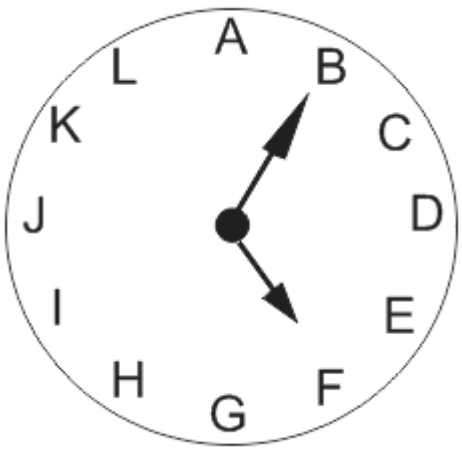
Misalkan pembuat pesan menentukan jam(kunci) yang digunakan adalah 05:05.

2. Selain menentukan jam, seorang pembuat pesan juga harus menentukan seperti apa pola loncatan waktu yang akan terjadi. Hal ini jugalah yang akan menjadi keunikan dari algoritma jam ini. Seperti yang diketahui sebuah jam akan berputar terus menerus. Terdapat dua jarum jam yang berputar (dengan mengabaikan jarum detik). Jarum pendek yang merupakan representasi jam, serta jarum panjang yang merupakan representasi menit.
   Pola loncatan yang ditentukan oleh pembuat pesan akan sangat berpengaruh dalam proses enkripsi.

   Misalkan sang pembuat pesan menentukan pola loncatan adalah 25 menit.

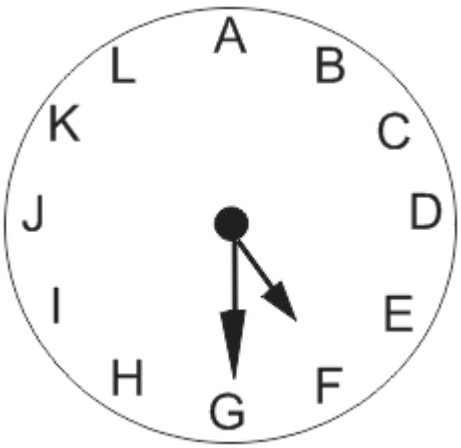
3. Dengan menggunakan jam awal 05:15 serta pola loncatan 25 menit, program secara otomatis akan menentukan kunci dalam bentuk huruf yang akan digunakan dalam proses enkripsi. Setiap jarum jam dan jarum menit akan menghasilkan satu buah huruf. Sehingga setiap kali proses perputaran jarum jam, akan diperoleh dua buah huruf sebagai kunci.

   Misalkan dalam kasus ini kunci awal yang digunakan adalah 05:05. Hal ini akan diterjemahkan oleh program sebagai huruf E dan huruf B. Untuk lebih jelasnya dapat dilihat pada ilustrasi dibawah ini:



Gambar 1
Ilustrasi Algoritma Jam

Untuk proses selanjutnya, karena pola lompatan yang ditentukan adalah 25 menit, maka selanjutnya jarum jam akan menunjukkan pukul 05:30 yang diterjemahkan sebagai huruf E dan huruf G seperti dapat dilihat pada ilustrasi dibawah ini:



Gambar 2
Ilustrasi Algoritma Jam

Proses ini akan terus berulang sampai diperoleh kunci yang sama panjang dengan panjang dari plain teks. Jika setelah sekian kali proses putaran ternyata jarum jam kembali menunjukkan waktu yang sama, maka pola lompatan akan bertambah 5 menit.

Karena dalam contoh kali ini digunakan teks TEMUI SAYA DI TAMAN, maka diperlukan kunci sepanjang 16 karakter, hal ini sama dengan delapan kali proses perputaran jam. Sehingga untuk kasus kali ini akan diperoleh kunci sebagai berikut:

EBEGL FEFJ GC GHHAI

4. Setelah kunci diperoleh, proses selanjutnya yang dilakukan adalah memanfaatkan tabel enkripsi seperti layaknya pada algoritma Vigenere Cipher klasik.

   Dalam contoh kali ini proses enkripsi awal akan menmberikan hasil sebagai berikut:

   XFQAT XEDJ JK ZHTAV

5. Dalam proses enkripsi yang terakhir, demi meningkatkan keamanan, dilakukan proses translasi ke dalam bentuk ASCII.

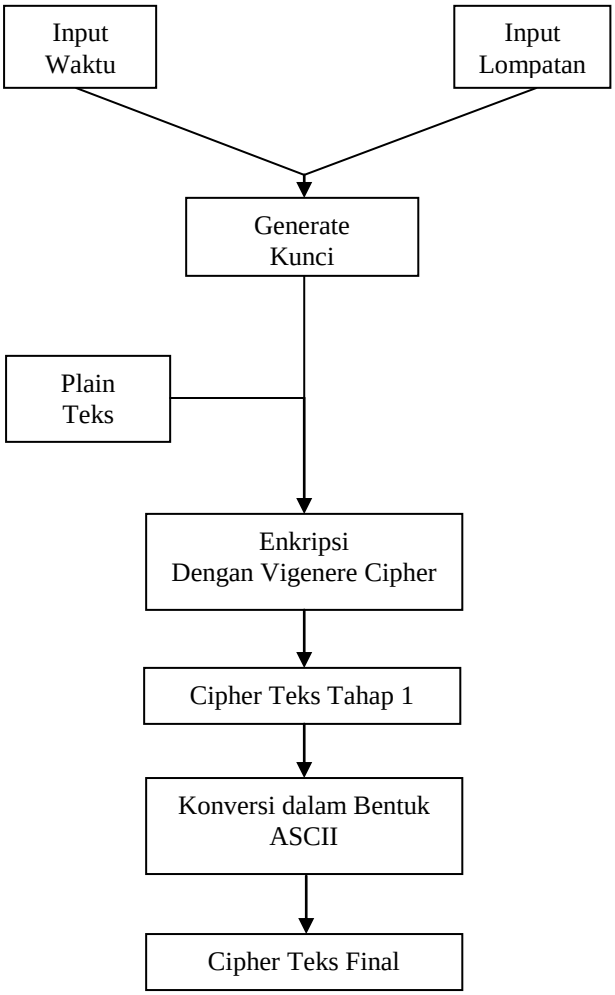
   Dalam contoh kali ini proses translasi akan memberikan hasil sebagai berikut:

   088069081090077032066070066070032076079
   3208507083072089

   Dengan penerapan algoritma seperti ini,

diharapkan kelemahan yang terdapat dari algoritma Vigenere Cipher dapat tertutupi.

Jika digambarkan dalam bentuk diagram, algoritma ini akan bekerja sebagai berikut:

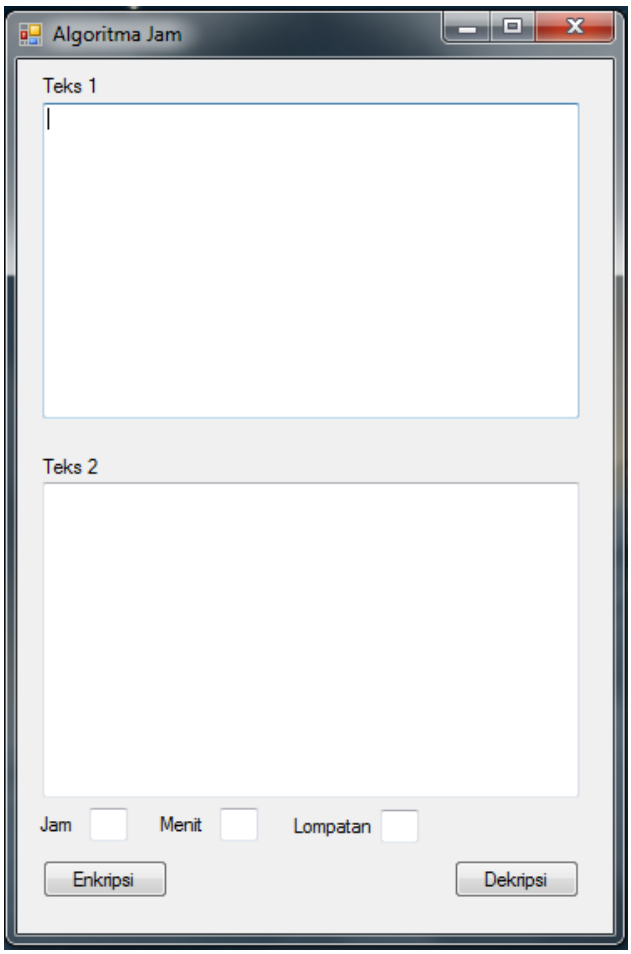


III. IMPLEMENTASI ALGORITMA

Untuk mencari tahu keberhasilan dari algoritma ini, selanjutnya penulis mencoba untuk menerapkan algoritma ini dalam sebuah program dengan menggunakan bahasa C#.

Program ini merupakan sebuah program yang melakukan proses enkripsi dari teks. Dalam program ini user diminta untuk menginputkan data berupa plain teks atau cipher teks yang akan dienkripsi atau didekripsi disertai kunci berupa jam, menit dan lompatannya. Teks awal haruslah diinputkan pada teks boks berlabel Teks 1 dan hasil akhirnya akan tampil pada teks boks berlabel Teks 2.

Berikut adalah hasil implementasi dari program tersebut:

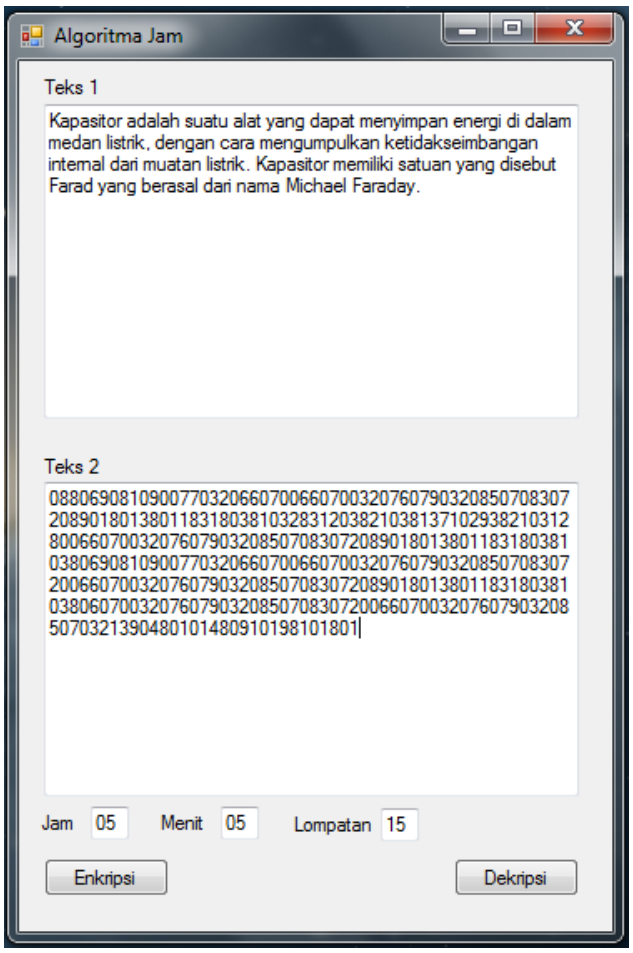


Gambar 3
Tampilan Awal Program

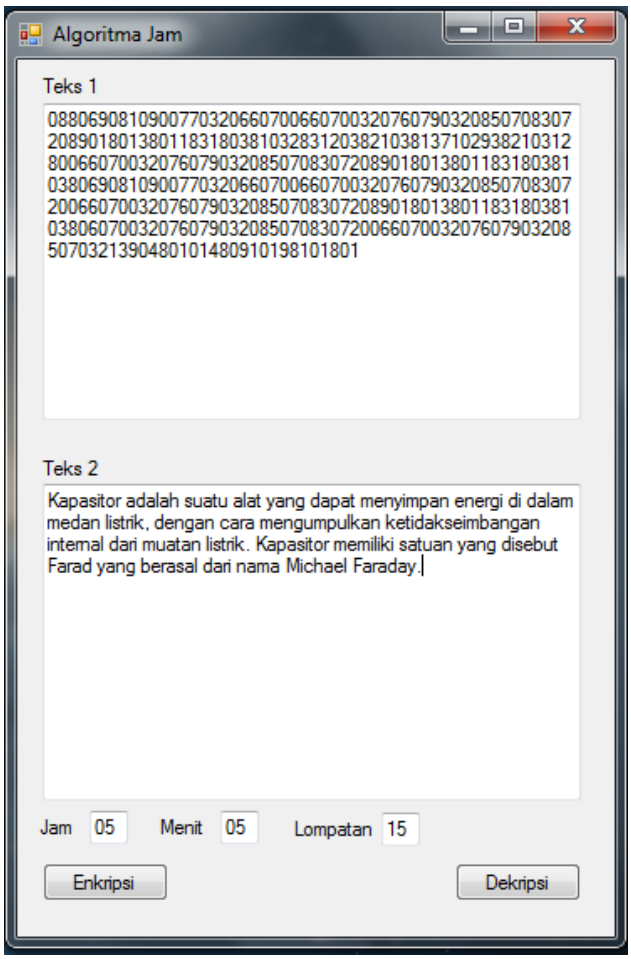
Untuk menguji keberhasilan dari algoritma ini, penulis mencoba melakukan enkripsi untuk sebuah teks seperti berikut:

Kapasitor adalah suatu alat yang dapat menyimpan energi di dalam medan listrik, dengan cara mengumpulkan ketidakseimbangan internal dari muatan listrik. Kapasitor memiliki satuan yang disebut Farad yang berasal dari nama Michael Faraday.

Berikut adalah *screenshot* yang merupakan hasil dari proses enkripsi dengan menggunakan kunci 05:05 dan lompatan 15 menit:



Gambar 4
Hasil Enkripsi Pada Program



Gambar 5
Hasil Dekripsi Pada Program

*Cipher*Teks:

0880690810900770320660700660700320760790320850708307208901801380118318038103283120382103813710293821031280066070032076079032085070830720890180138011831803810380690810900770320660700660700320760790320850708307200660700320760790320850708307208901801380118318038103806070032076079032085070830720066070032076079032085070830720066070032076079032085070321390480101480910198101801

Selanjutnya teks pada hasil enkripsi diatas dicoba untuk dikembalikan (didekripsi) kembali. Proses dekripsi dalam algoritma ini dapat dilakukan dengan membalik algoritma yang digunakan pada proses enkripsi. Pada program kali ini user cukup menginputkan data berupa *cipher* teks pada teks boks berlabel Teks 1 dan mengklik tombol berlabel dekripsi.

Berdasarkan hasil uji coba pada *cipher* teks diatas, dan dengan menggunakan kunci yang sama, diperoleh hasil dekripsi sebagai berikut:

Sebagai tambahan berikut adalah algoritma yang penulis gunakan untuk implementasi program ini:

```
Procedure EnkripsiJam()
{
    TerimaInputTeks();
    //program menerima input teks yang akan dienkripsi

    TerimaInputJam();
    //program menerima input jam sebagai kunci

    TerimaPolaLompatan();
    //program menerima pola lompatan yang akan digunakan

    While(i=0;i<=PanjangTeksInput;i++)
    {
        GenerateKunci(Jam,Lompatan);
        //proses generate kunci dengan berdasarkan pada jam dan pola lompatan serta panjang teks input

        If(JamSekarang == JamAwal)
        {
```

```
                    Lompatan = Lompatan+5;
                    //jika terjadi perulangan waktu
                    jam, maka pola lompatan akan
                    ditambah 5 menit
                }

                EnkripsiSecaraVigenere(Teks,Kunci);
                //melakukan enkripsi secara Vigenere
                Cipher berdasarkan kunci yang sudah
                dihasilkan

                KonversiKeASCII(teksenkripsi1);
                //melakukan konversi ke dalam ASCII
                berdasarkan hasil enkripsi pada proses
                pertama (EnkripsiSecaraVigenere)
        }
    }
```

IV. UJI COBA

Setelah dilakukan implementasi pada program, selanjutnya penulis mencoba untuk melakukan analisa serta uji coba pada algoritma jam ini.

Seperti dapat dilihat dalam hasil enkripsi diatas, algoritma jam ini mampu untuk memberikan *cipher* teks yang berbeda dibandingkan jika proses enkripsi dilakukan dengan menggunakan algoritma Vigenere Cipher biasa.

Dalam pembahasan sebelumnya penulis telah membahas bahwa kelemahan algoritma Vigenere Cipher adalah dapat dipecahkan dengan menggunakan metode Kasiski, karena itu untuk uji coba kali ini penulis mencoba untuk menggunakan analisa berdasarkan metode Kasiski demi mengetahui apakah algoritma ini berhasil atau tidak.

1. UJI COBA DENGAN METODE KASISKI

A. Metode Uji Coba

Uji coba dilakukan dengan menggunakan sebuah *cipher* teks yang sudah diperoleh dalam uji coba sebelumnya. Dengan *cipher* teks ini penulis akan melakukan analisa mengenai frekuensi kemunculan trigraphsnya (sama dengan metode analisa Kasiski untuk algoritma Vigenere Cipher biasa). Penulis akan meninjau bagaimana hasil dari analisa menggunakan metode Kasiski tersebut. Hasil dari analisa TriGraphs ini akan digunakan untuk analisa menggunakan metode Kasiski.

B. Uji Coba

Dengan menggunakan metode pada penjelasan diatas, ternyata diperoleh hasil bahwa analisa TriGraphs tidak memberikan hasil yang berarti dari *cipher* teks yang dihasilkan dari algoritma ini. Misalkan ditemukan beberapa perulangan untuk *cipher* teks 900.

Namun perulangan tersebut tidak memiliki pola yang teratur karena memang perulangan tersebut hanyalah sebuah kebetulan, dan bukan dampak dari proses enkripsi dengan kunci yang berulang.

Dengan gagalnya dilakukan analisa TriGraphs untuk algoritma ini, menyebabkan analisa metode Kasiski tidak dapat dilakukan lebih lanjut. Hal lain yang menyebabkan analisa menggunakan metode Kasiski tidak akan dapat dilakukan adalah panjang kunci yang digunakan untuk enkripsi ini adalah sama dengan panjang teks semula, sehingga kemungkinan perulangan pola dari kunci yang berulang nyaris tidaklah mungkin.

2. UJI COBA DENGAN METODE BRUTE FORCE

Uji coba selanjutnya yang mungkin untuk dilakukan adalah dengan menggunakan metode Brute Force. Seperti yang diketahui, algoritma Brute Force adalah algoritma yang naif namun dapat digunakan untuk menyelesaikan hampir seluruh permasalahan algoritma.

Sebenarnya, algoritma yang penulis rancang dalam tulisan kali ini dapat dipecahkan dengan menggunakan dua buah kunci yang berbeda. Yang pertama adalah menggunakan kunci berupa jam serta pola lompatan, sementara yang kedua adalah dengan menggunakan kunci hasil generate dari kunci jam. Sebenarnya cara yang kedua cenderung mustahil untuk dilakukan dalam kondisi normal, namun cara ini cukup memberi peluang untuk dipecahkan dengan menggunakan metode Brute Force.

Jika metode Brute Force dicoba untuk diterapkan dengan menggunakan cara pertama, yaitu dengan kunci berupa jam, maka akan dibutuhkan jumlah percobaan seperti berikut:

Variasi Jam	: 12
Variasi Menit	: 12
Variasi Lompatan	: 1435

Ket. angka variasi lompatan diperoleh dengan mengasumsikan jumlah lompatan terbesar yang mungkin adalah (24x60)-5 = 1435, atau dengan kata lain sama dengan nilai 24 jam kurang lima menit, karena jika kunci yang digunakan adalah 24 jam, maka hal ini menyebabkan algoritma jam tidak akan berguna.

Sehingga variasi kemungkinan kuncinya adalah 206.640 kemungkinan kunci.

Namun sebuah keuntungan dari algoritma ini adalah proses *generate* kunci yang tidak lazim, sehingga besar kemungkinan proses kriptanalisis akan dilakukan dengan menggunakan variasi kunci-kunci dalam bentuk huruf. Hal ini akan sama dengan proses Brute Force pada algoritma Vigenere Cipher biasa dengan sebelumnya sang pembobol pesan sudah terlebih dahulu mengkonversi kode ASCII yang digunakan kedalam teks padanannya.

Karena panjang kunci yang digunakan dalam algoritma ini pastilah sama panjang dengan *plain* teksnya, maka jika dilakukan proses Brute Force akan dibutuhkan jumlah percobaan sebanyak:

$$26^{\text{panjangteks}}$$

Dapat disimpulkan jika proses Brute Force ini benar-benar dilakukan oleh pembobol pesan, maka hal tersebut akan membutuhkan waktu yang sangat lama dan membutuhkan sumber daya komputer yang handal.

V. KESIMPULAN

Berdasarkan kepada implementasi yang telah penulis lakukan, maka dapat disimpulkan bahwa algoritma jam yang penulis kembangkan dalam tulisan kali ini dapat berfungsi dengan baik sebagai algoritma kriptografi alternatif.

Algoritma ini mampu menutupi kekurangan yang dimiliki oleh algoritma Vigenere Cipher klasik, hal ini dipengaruhi oleh panjang kunci yang digunakan untuk proses enkripsi adalah sama dengan panjang teks. Hal ini berdampak pada tidak dimungkinkannya dilakukan proses analisa menggunakan metode Kasiski.

REFERENSI

[1] Sadikin, Rifki. 2012. Kriptografi Untuk Keamanan Jaringan. Yogyakarta : Penerbit Andi.
Rasheed, Faraz. 2006. C# School. Programmers Heaven.

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, 25 Maret 2013

Okharyadi Saputra
13510072