Modifikasi Full Vigenere Chipher dengan Pengacakan Susunan Huruf pada Bujur Sangkar Berdasarkan Kunci

Setia Negara B. Tjaru
13508054
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
setia.negara.91@gmail.com

Abstrak – Vigènere cipher merupakan salah satu algoritma klasik yang digunakan untuk menyembunyikan pesan berupa teks dari pihak yang tidak berhak dengan menggunakan teknik substitusi dimana tiap huruf pada plainteks akan disubstitusi menjadi huruf lain berdasarkan kunci yang digunakan. Berbeda dengan Caesar cipher, Vigènere cipher adalah algoritma substitusi jamak dimana suatu huruf plainteks tidak selalu disubstitusi menjadi huruf yang sama, namun disubstitusi berdasarkan kunci yang digunakan. Algoritma Vigènere cipher klasik ini merupakan algoritma enkripsi yang cukup mudah dipecahkan dengan menggunakan teknik analisis frekuensi terhadap cipherteksnya.

Kelemahan utama pada algoritma Vigènere cipher ini yaitu jika panjang kunci lebih pendek dari panjang plainteksnya akan menghasilkan perulangan kunci yang digunakan untuk mengenkripsi plainteks tersebut. Kunci yang berulang tersebut menimbulkan celah berupa jumlah pergeseran yang sama untuk setiap plainteks yang disubstitusi oleh huruf pada kunci yang sama sehingga huruf-huruf pesan atau plainteks dapat dikelompokkan berdasarkan kunci yang digunakan. Karena terdapat kelompok huruf-huruf plainteks yang disubstitusi dengan huruf kunci yang sama karena perulangan kunci, maka tiap kelompok huruf-huruf tersebut dapat dikenakan metode analisis frekuensi terhadapnya.

Salah satu varian dari Vigenere Chipher untuk memperkuat keamanannya dari analisis frekuensi ialah Full Vigenere Chipher. Chipher ini memakai permutasi dari huruf alfabet tanpa pola untuk ditempatkan pada bujur sangkar vigenere, berbeda dengan vigenere biasa yang hanya merupakan permutasi hasil pergeseran huruf pada isi bujur sangkarnya. Masalah muncul ketika kita ingin mengirimkan pesan kepada seseorang yang belum mengetahui kombiansi bujur sangkarnya. Oleh akrena itu perlu dibuat metode untuk membangkitkan isi dari bujur sangkar vigenere berdasarkan kunci yang ada.

Kata Kunci—Vigenere Chipher, Full Vigenere Chipher, Auto-key Vigenere Cipher, enkripsi, dekripsi, Pseudo-random Generator, .

I. PENDAHULUAN

Dewasa ini perkembangan teknologi informasi dan komunikasi sudah berkembang sangat pesat. Hampir di setiap bidang kehidupan telah menggunakan teknologi ini sebagai saran pendukung maupun sarana utama. Sehubungan dengan hal ini, aspek keamanan dalam teknologi informasi dan komunikasi tentunya tidak bisa diabaikan. Dalam kegiatan kirim-terima pesan, aspek keamanan yang perlu diperhatikan antara lain kerahasiaan, integritas data, otentikasi, dan nirpenyangkalan. Aspek keamanan tersebut bisa dijaga dengan memanfaatkan kriptografi.

Pada umumnya, algoritma kriptografi bisa dibagi menjadi dua, yakni kriptografi klasik dan kriptografi modern. Kriptografi klasik biasanya menggunakan algoritma yang sederhana dan berbasiskan karakter. Sedangkan kriptografi modern biasanya menggunakan algoritma yang kompleks dan beroperasi dalam mode bit, sehingga lebih susah untuk dipecahkan.

Algoritma kriptografi klasik terdiri dari dua macam, yaitu chiper substitusi dan chiper transposisi. Chiper substitusi menyandikan plainteks dengan cara mengganti setiap karakter dengan karakter lain dalam susunan abjad. Jenis-jenis chiper substitusi ini antara lain chiper abjad-tunggal, chiper abjad-majemuk, chiper substitusi homofonik, dan chiper substitusi poligram. Sedangkan chiper transposisi menyandikan plainteks dengan cara melakukan transpose terhadap rangkaian karakter di dalam plainteks.

II. VIGENERE CHIPHER

A. Konsep Dasar

Vigènere chiper merupakan salah satu contoh chiper abjad-majemuk (polyalphabetic substitution chiper). Chiper abjad-majemuk akan mengganti setiap karakter pada plainteks dengan karakter lain yang mungkin berbeda-beda pada chiperteksnya. Vigènere chiper menggunakan bujur sangkar Vigènere untuk melakukan enkripsi. Setiap baris di dalam bujur sangkar menyatakan huruf-huruf chiperteks yang diperoleh dengan Caesar chiper, di mana jauh pergeseran huruf plainteks ditentukan oleh nilai desimal dari huruf kunci tersebut (A = 0, B = 1, C = 3, ... , Z = 25).

Untuk melakukan enkripsi dengan Vigènere chiper, lakukan pada bujur sangkar Vigènere sebagai berikut:tarik garis vertikal dari huruf plainteks ke bawah, lalu tarik garis mendatar dari huruf kunci ke kanan. Perpotongan kedua garis tersebut menyatakan huruf chiperteksnya.

	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. Bujur Sangkar Vigenere

Pada Vigènere chiper, jika panjang kunci lebih pendek daripada panjang plainteks, maka kunci tersebut akan diulang penggunaannya.

Contoh penggunaan Vigènere chiper:

P : SETIANEGARABTJARU
K : SETIASETIASETIASE
C : KIMQAFIZIRSFMRAJY

Pada contoh di atas, plainteks "SETIANEGARABTJARU" dienkripsi dengan kunci "SETIA" menghasilkan chiperteks "KIMQAFIZIRSFMRAJY".

Perhatikan bahwa huruf A pada plainteks disubstitusi dengan huruf yang berbeda-beda pada chiperteks, yakni A, I, S. Hal inilah yang menyebabkan Vigènere chiper termasuk chiper abjad-majemuk.

Aturan enkripsi pada Vigènere chiper bisa dinyatakan juga sebagai penjumlahan modulo 26 dari satu karakter plainteks dengan satu karakter kunci.

$$C_i \equiv P_i + K_i \pmod{26}$$

di mana
Pi : karakter plainteks
Ki : karakter kunci
Ci : karakter chiperteks

Dekripsi pada Vigènere chiper dilakukan dengan cara yang berkebalikan, yaitu dengan cara menarik garis horizontal dari huruf kunci sampai ke huruf chiperteks yang dituju, lalu dari huruf cipherteks tarik garis vertikal ke atas sampai ke huruf plainteks. Atau, bisa juga dinyatakan dalam persamaan:

$$P_i \equiv C_i - K_i \pmod{26}$$

B. Kekuatan

Kekuatan algoritma Vigènere chiper ini adalah dapat mencegah frekuensi huruf-huruf di dalam chiperteks yang memiliki pola tertentu yang sama, seperti yang terjadi pada chiper abjad-tunggal. Pada chiper abjad-tunggal, huruf yang paling sering muncul di chiperteks merupakan substitusi dari huruf yang paling sering muncul di plainteks. Karena itu, dengan teknik analisis frekuensi, kriptanalis bisa dengan mudah menebak huruf tersebut. Namun, pada Vigènere chiper hal tersebut tidak bisa dilakukan karena satu macam huruf pada plainteks mungkin dienkripsi menjadi beberapa macam huruf pada chiperteks, seperti pada contoh sebelumnya.

C. Kelemahan

Vigènere chiper memungkinkan perulangan huruf atau pasangan huruf pada plainteks terjadi juga pada chiperteksnya. Hal ini dikarenakan kunci yang digunakan untuk melakukan enkripsi juga diulang. Akibatnya, bagian plainteks dan bagian kunci tertentu bisa "berpasangan" lebih dari satu kali. Contoh:

P : SETIANEGARSETIA
K : SETIASETIASETIA
C : KIMQAFIZIRKIMQA

Terlihat pada contoh di atas, SETIA dienkripsi menjadi kriptogram yang sama, yaitu KIMQA. Namun, perlu diperhatikan bahwa kasus seperti ini tidak selalu demikian, misalnya pada contoh berikut ini:

P : SETIANEGARASETIA
K : SETIASETIASETIAS
C : KIMQAFIZIRSWXBIS

Pada contoh di atas, SETIA tidak dienkripsi menjadi kriptogram yang sama.

Sifatnya yang mungkin untuk menghasilkan kriptogram yang sama terhadap bagian plainteks yang sama ini menjadi kelemahan Vigènere chiper.

D. Metode Kasiski

Metode Kasiski memanfaatkan kelemahan Vigènere chiper yang mungkin menghasilkan kriptogram yang sama untuk bagian plainteks yang sama. Pada contoh di atas, di mana SETIA dienkripsi menjadi kriptogram yang sama,

yaitu KIMQA, secara intuitif menunjukkan bahwa jika jarak antara dua buah string yang berulang pada plainteks merupakan kelipatan dari panjang kunci, maka string yang sama tersebut akan muncul menjadi kriptogram yang sama pula pada chiperteks. Pada contoh pertama di atas, jarak antara string SETIA adalah 10, dan panjang kunci SETIA adalah 5.

Sedangkan pada contoh kedua, jarak antara string SETIA adalah 10, dan panjang kunci SETIA adalah 5.

Dengan metode Kasiski, kriptanalis bisa memperoleh panjang kunci dari suatu Vigènere chiper. Caranya adalah:

1. Temukan semua kriptogram yang berulang di dalam cipherteks (pesan yang panjang biasanya mengandung kriptogram yang berulang).
2. Hitung jarak antara kriptogram yang berulang
3. Hitung semua faktor (pembagi) dari jarak tersebut (faktor pembagi menyatakan panjang kunci yang mungkin).
4. Tentukan irisan dari himpunan faktor pembagi tersebut. Nilai yang muncul di dalam irisan menyatakan angka yang muncul pada semua faktor pembagi dari jarak-jarak tersebut . Nilai tersebut mungkin adalah panjang kunci. Hal ini karena string yang berulang dapat muncul bertindihan (coincidence) .

Contoh:

ASDFBDKDFGRWASDFFGRW

Kriptogram yang berulang adalah ASDF dan FGRW.

Jarak antara dua perulangan ASDF adalah 12. Jarak antara dua perulangan FGRA adalah 4. Dengan demikian, kemungkinan besar panjang kunci adalah 4.

Jika panjang kunci telah diketahui, maka kunci dapat ditentukan dengan beberapa cara, antara lain:

1. Exhaustive key search, yakni dengan membangkitkan semua kemungkinan kunci. Jika panjang kunci adalah p, maka cara ini membutuhkan 26^p kali percobaan.
2. Mengelompokkan huruf-huruf pada chiperteks ke sejumlah p kelompok (p = panjang kunci). Lalu, melakukan teknik analisis frekuensi pada tiap-tiap kelompok tersebut.

Dengan demikian, Vigènere chiper merupakan chiper yang bisa terpecahkan (breakable chiper).

III. FULL VIGENERE CHIPHER

Full vigenere chipher adalah varian dari vigenere cipher dimana urutan huruf pada baris-baris dalam bujur sangkar vigenere cipher tidak lagi berurut sesuai alfabet. Namun merupakan urutan yang acak.

Full vigenere chipher sedikit lebih aman daripada cipher vigenere biasa. Karena vigenere cipher biasa dapat dipecahkan dengan menggunakan metode frekuensi hanya dengan mengetahui huruf-huruf yang sering muncul pada suatu bahasa tertentu , sementara dengan full vigenere diperlukan seluruh frekuensi distribusi relatif dari satu bahasa untuk memecahkannya.

Kelemahan dari full vigenere cipher ini adalah karena urutan alfabet dari bujur sangkar ini acak dan tidak berpola. Sehingga sulit ketika ingin mengirimkan suatu pesan terenkripsi kepada orang yang belum memiliki urutan bujur sangkarnya.

Untuk itu pada makalah ini penulis mencoba merancang modifikasi cipher full vigenere dengan meng-*generate* bujur sangkar sesuai dengan kunci yang dimasukkan.

IV. AUTO-KEY VIGENERE CIPHER

Full vigenere chipher adalah varian dari vigenere cipher yang berguna untuk menghilangkan salah satu kelemahan dari vigener cipher biasa, yaitu adanya kemungkinan munculnya kriptogram yang berulang jika dienkripsi menggunakan kunci yang lebih pendek dari plaintext. Hal itu disebabkan karena adanya perulangan kunci yang bertujuan agar panjang kunci sama dengan panjang plaintext agar setiap huruf pada plaintext mendapat pasangan.

Perbedaan dari auto key vigenere cioher dengan vigenere cipher biasa adalah jika pada vigenere cipher biasa panjang kunci lebih pendek daripada panjang plaintext, maka kunci akan diulang sehinga setiap huruf dari plaintext akan mendapatkan pasangan kunci.

Sementara untuk Auto-key vigenere cipher, jika panjang kunci lebih pendek daripada panjang plaintext, maka plaintext akan disambungkan dengan kunci hingga panjang kunci yang baru akan sama dengan panjang plaintext.

Contoh:

P: SETIANEGARABTJARU
K: NAMA

Jika menggunakan vigenere cipher biasa, maka akan menjadi:

P: SETIANEGARABTJARU
K: NAMANAMANAMANAMAN

Jika menggunakan auto-key vigenere cipher, maka akan menjadi:

P: SETIANEGARABTJARU
K:NAMASETIANEGARABT

V. PSEUDO-RANDOM GENERATOR

Pseudo-random generator adalah algoritma untuk membangkitkan urutan bilangan yang mendekati ciri-ciri bilangan acak. Urutannya tidak akan benar-benar acak karena ditentukan oleh nilai awal yang biasa disebut seed.

Terdapat beberapa algoritma pseudo-random generator.seperti:

1. Blum Blum Shub
2. Complementary-multiply-with-carry
3. Inversive congruential generator
4. ISAAC (cipher)
5. Lagged Fibonacci generator
6. Linear congruential generator

Pada beberapa bahasa pemrograman menyediakan fungsi atau method untuk membangkitkan bilangan acak sesuai Pseudo-random Generator. Misalnya method "Random (long seed)" pada java yang sudah menerapkan algoritma pseudo-random generator.

VI. PERANCANGAN MODIFIKASI FULL VIGENERE CIPHER

Modifikasi full vigenere cipher akan kita lakukan dengan mengacak susunan huruf-huruf pada bujur sankar vigenere berdasarkan dengan kunci masukan. Sehingga untuk menerjemahkan pesan yang terdekripsi, sang penerima pesan hanya harus mengetahui kunci tanpa harus mengetahui susunan huruf pada bujur sangkar vigenere, karena hal tersebut dapat dibangkitkan dengan menggunakan kunci yang ada.

Agar urutan huruf pada bujur sangkar vigenere saat enkripsi dan dekripsi sama, maka metode pengacakannya menggunakan metode pseudo-random generator.

Adapun metode pengacakannya adalah sebagai berikut:

```
public int[][] generaterandom() {
        System.out.println(key);
        int[] keyArray = new
int[key.length()];
        int tempseed = 0;

        for (int i = 0; i <
key.length(); i++) {
            keyArray[i] =
key.charAt((int) i);
            tempseed += keyArray[i];
        }

        int seed = tempseed;
        int[][] bujursangkarViginere =
new int[26][26];
        boolean[] flagHuruf = new
boolean[26];


        for (int i = 0; i < 26; i++)
{// untuk kolom

            //reinisialisasi flaghuruf
            for (int k = 0; k <
flagHuruf.length; k++) {
                flagHuruf[k] = false;
            }

            for (int j = 0; j < 26;
j++) {//untuk baris
                seed += tempseed +
(keyArray[i % keyArray.length]) *
(j+1) * (i+1);
                seed %= 26;
                while
(flagHuruf[seed]) {
                    ++seed;
                    seed %= 26;
                }

bujursangkarViginere[i][j] = seed;
                flagHuruf[seed] =
true;
                seed += seed;
            }
        }

        return bujursangkarViginere;
    }
```

Pertama-tama kita akan menentukan seed untuk dijadikan bilangan dasar pengacakan. Untuk itu kita menambahkan semua nilai dari huruf-huruf dari kunci.

```
for (int i = 0; i < key.length(); i++)
{
            keyArray[i] =
key.charAt((int) i);
            tempseed += keyArray[i];
        }
```

Maka untuk menentukan huruf-huruf di dalam bujur sangkar vigenere maka seed awal ditambahkan dengan seed kemudian di-*mod*-kan dengan 26 (26 alfabet).

```
seed += tempseed + (keyArray[i %
keyArray.length]) * (j+1) * (i+1);
seed %= 26;
```

Masalah yang ditemukan penulis saat ini ialah bagaimana membedakan kunci yang kombinasi huruf pembentuknya sama tapi urutan hurufnya berbeda.

Contohnya kata kunci “SUSU” dan “USUS”. Untuk itu urutan huruf harus terlibat dalam metode pengacakan.

Oleh karena itu ditambahkanlah `(keyArray[i % keyArray.length])` agar huruf ke-i pada kunci akan berperan dalam menentukan huruf ke-i pada satu baris bujur sangkar vigenere. Sementara `(j+1) * (i+1)` adalah variasi agar bujursangkar benar-benar teracak, karena tanpa adanya penambahan dari variabel i, penulis menemukan adanya kecenderungan pengulangan pola deretan huruf pada kolom terakhir.

Seringkali dalam melakukan pengacakan huruf yang sebelumnya sudah muncul akan muncul kembali oleh akrena itu ditangani dengan menggeser huruf yang berulang sebanyak satu kali hingga huruf yang muncul belum pernah muncul sebelumnya.

```
while (flagHuruf[seed]) {
                    ++seed;
                    seed %= 26;
                }
```

Selain itu untuk lebih memperkuat cipher yang baru ini kita akan mengadapatasi algoritma Auto-key vigenere cipher. Yaitu tidak memakai pengulangan kunci tapi menambahkan huruf-huruf plaintext pada kunci jika panjang kunci yang ada lebih pendek daripada panjang plaintext.

Perbedaannya adalah untuk cipher ini tidak ditambahkan plaintext jika kunci lebih pendek, tapi menambahkan hasil ciphernya sebagai kunci.

Sebelum diterapkan Auto-key:

P: ASDF GHJK GHJK ASDF
K: SETI
C: NYIZ EBKM EBKM NYIZ

Setelah diterapkan Auto-key:

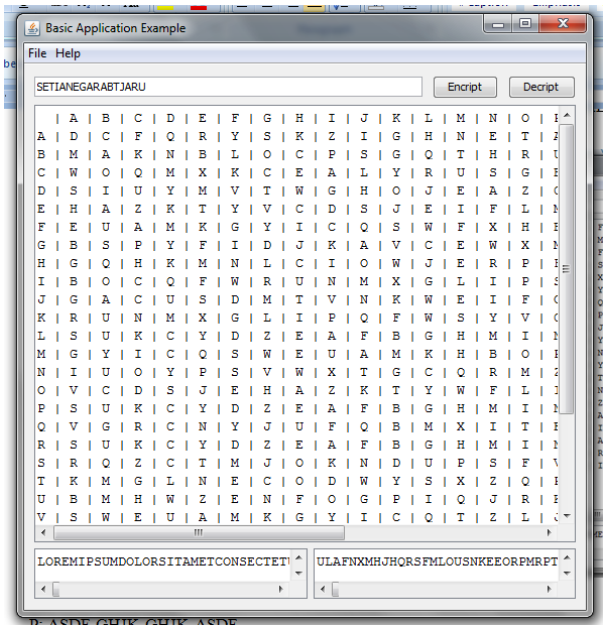
P: ASDF GHJK GHJK ASDF
K: SETI
C: NYIZ PDHJ ELUX IUHC

Terlihat sudah tidak nampak perulangan kriptogram meskipun kunci lebih pendek dan perulangan deretan huruf pada plaintext jaraknya sama dengan panjang kunci.

Secara ringkas dapat dituliskan algoritma hasil modifikasi full vigenere cipher adalah sebagai berikut:

1. Menambahkan semua nilai dari semua huruf yang ada pada kunci dan menjadikannya sebagai seed. (Huruf A-Z berkorespondensi dengan angka 0-25)
2. Untuk setiap huruf ke-j pada baris ke-i, seed akan ditambahakan dengan hasil kali dari nilai huruf kunci ke-i dengan (i+1)*(j+1) kemudian di-mod dengan 26.
3. Jika nilai yang muncul belum pernah muncul sebelumnya, maka nilai ini akan dipakai namun jika sudah pernah, maka nilai akan ditambah satu hingga nilai tersebut belum pernah muncul sebelumnya.
4. Dalam pengenkripsian, jika plainteks lebih panjang daripada kunci, maka plainteks sejumlah panjang kunci dienkripsi menggunakan kunci yang ada terlebih dahulu, baru kemudian sisa plainteks yang belum dienkripsi akan dienkripsi dengan kunci yang merupakan hasil enkripsi sebelumnya. (ciphertext)

VII. Analisis Keamanan Cipher



Analisis keamanan cipher, dengan plaintext:

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem. Investigationes demonstraverunt lectores legere me lius quod ii legunt saepius. Claritas est etiam processus dynamicus, qui sequitur mutationem consuetudium lectorum. Mirum est notare quam littera

gothica, quam nunc putamus parum claram, anteposuerit litterarum formas humanitatis per seacula quarta decima et quinta decima. Eodem modo typi, qui nunc nobis videntur parum clari, fiant sollemnes in futurum.

Kunci: SETIANEGARABTJARU

A. Analisis Frekuensi

Frekuensi Plaintext

Freq.	Letter
108	I
101	E
87	T
81	U
79	A
65	S
61	M
60	O
60	N
53	R
53	L
35	C
34	D
23	P
16	Q
13	G
10	F
10	V
7	B
6	H
4	Y
3	Z
2	X
1	W

Frekuensi Ciphertext

Freq.	Letter
50	L
49	M
47	A
47	Z
46	I
45	Q
44	T
43	G
43	B
43	P
42	O
41	H
39	D
36	R
36	Y
36	S
35	N
34	W
32	V
29	J
29	E
28	U
27	X
26	K
23	F
22	C

Dari perhitungan frekuensi kemunculan huruf dapat terlihat bahwa tidak ada kolerasi antara frekuensi huruf pada plaintext dengan ciphertext.
Bahkan frekuensi huruf yang muncul pada ciphertext cenderung setara, meskipun pada plaintext terdapat huruf-huruf yang sangat tinggi frekuensinya dan ada juga yang sangat rendah. Oleh karena itu metode analisis frekuensi tidak dapat diterapkan untuk memecahkan cipher ini.

B. Metode Kasiski

ULA: occurs 2 times, at pos 0, 115
distance(s): 115
AGU: occurs 2 times, at pos 52, 448
distance(s): 396
MQJ: occurs 2 times, at pos 82, 220
distance(s): 138
MOA: occurs 2 times, at pos 119, 269
distance(s): 150
OAP: occurs 2 times, at pos 120, 611
distance(s): 491
UGW: occurs 2 times, at pos 130, 381
distance(s): 251
WPU: occurs 2 times, at pos 148, 833
distance(s): 685
MZI: occurs 2 times, at pos 158, 797
distance(s): 639
QSN: occurs 2 times, at pos 169, 916
distance(s): 747
LMF: occurs 2 times, at pos 185, 912
distance(s): 727
HTM: occurs 2 times, at pos 218, 418
distance(s): 200
YJD: occurs 2 times, at pos 227, 786
distance(s): 559
JQM: occurs 2 times, at pos 236, 267
distance(s): 31
QMS: occurs 2 times, at pos 237, 312
distance(s): 75
IET: occurs 2 times, at pos 254, 484
distance(s): 230
STY: occurs 2 times, at pos 309, 327
distance(s): 18
VZH: occurs 2 times, at pos 416, 875
distance(s): 459
TWC: occurs 2 times, at pos 486, 931
distance(s): 445
AQN: occurs 2 times, at pos 510, 762
distance(s): 252

IUG: occurs 2 times, at pos 562, 665
distance(s): 103
UGM: occurs 2 times, at pos 563, 666
distance(s): 103
BOA: occurs 3 times, at pos 610, 621, 660
distance(s): 11, 50, 39
FZQ: occurs 2 times, at pos 614, 914
distance(s): 300
BWZ: occurs 2 times, at pos 627, 637
distance(s): 10
TPL: occurs 2 times, at pos 701, 713
distance(s): 12

Jika dilakukan serangan dengan metode kasiski didapatkan hasil seperti diatas. Dimana banyak ditemukan kriptogram tapi sering kali tidak akan berulang dan bahkan ketika berulang pun hanya sekali.
Selain itu, jarak antara perulangan kriptogram tidak akan konsisten atau memiliki faktor pembagi. Terlihat pada data di atas bahkan ada perulangan pada bilangan prima yang cukup besar seperti 31 atau 103.

Dengan demikian penyerangan dengan metode kasiski untuk mencari panjang kunci dengan melihat perulangan kriptogram yang muncul tidak akan diterapkan kepada cipher ini.

VIII. SIMPULAN

1. Modifikasi pada vigenere cipher dapat dilakukan dengan menggabungkan algoritma dari full vigenere cipher, auto-key vigenere cipher, dan pseudo-random generator.
2. Modifikasi dari full vigenere cipher ini akan lebih kuat dari varian vigenere cipher seperti auto-key dan full vigeere cipher karena menggabungkan kekuatan keduanya.
3. Modifikasi full vigenere cipher ini lebih praktis karena penerima pesan tidak perlu memiliki bujur sangkar vigenere karena bujur sangkarnya akan dibangkitkan berdasarkan kunci yang ada. Hal ini juga menambah kekuatan dari cipher karena bujur sangkar akan selalu berubah-ubah sesuai kunci.

REFERENCES

[1] Munir, Rinaldi, 2005. *Diktat Kuliah IF5054 Kriptografi*. Departemen Teknik Informatika, Institut Teknologi Bandung.
[2] http://en.wikipedia.org/wiki/Vigenère_cipher diakses pada Senin, 12 Maret 2012.
[3] http://en.wikipedia.org/wiki/Pseudorandom_number_generator diakses pada Senin, 19 Maret 2012.
[4] http://www.informatika.stei.itb.ac.id/~rinaldi/Kriptografi/2009-2010/CryptoHelper.jar
[5] http://www.csgnetwork.com/documentanalystcalc.html diakses pada Senin, 19 Maret 2012.
[6] http://pages.central.edu/emp/lintont/classes/spring01/cryptography/java/kasiski.html diakses pada Senin, 19 Maret 2012.

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, 19 Maret 2012

ttd

Setia Negara B. Tjaru/13508054