Vigènere Cipher Homofonik

Yudi Haribowo - 13504111

Program Studi Teknik Informatika
Institut Teknologi Bandung
Jl. Ganesha 10 Bandung 40132
E-mail : if14111@students.if.itb.ac.id

Abstraksi

Seiring meningkatnya perkembangan dunia teknologi, sistem pengaman yang canggih terhadap suatu data semakin dibutuhkan. Hal ini juga didorong oleh semakin maraknya kejahatan di dunia cyber. Salah satu sektor yang rawan mengundang kejahatan adalah pengiriman data. Oleh karena itu, pengguna teknologi semakin beramai-ramai mengembangkan suatu sistem pengamanan terhadap data yang biasa disebut kriptografi. Salah satu metode kriptografi yang dikenal adalah algoritma kriptografi klasik dimana metode ini sudah dikenal sejak lama. Algoritma kriptografi klasik memiliki berbagai macam bentuk, dimana yang akan dibahas dalam makalah ini adalah vigènere cipher.

Dalam makalah ini, penulis akan mencoba mengembangkan algoritma vigènere cipher dengan cara menggabungkannya dengan cipher substitusi homofonik. Selain variasi dengan penambahan tipe homofon sebagai bagian dari algoritma, penulis juga menambahkan dasar permutasi dalam penggunaan kunci yang digunakan untuk lakukan enkripsi. Dasar ide pengusulan metode ini adalah kompleksitas yang cukup tinggi pada kasus permutasi.

Isi dari makalah melipui konsep dasar, alasan, implementasi dan tingkat keamanan algoritma yang disertai pengujian. Konsep dasar meliputi landasan teori, implementasi meliputi penjelasan mengenai algoritma yang digunakan, dan disertai pengujian untuk berbagai macam kasus untuk menguji tingkat keamanannya.

***Kata kunci**: analisis frekuensi, enkripsi, dekripsi, cipherteks,key streaming, Known Plaintext Attack, kode*

1 Pendahuluan

Perkembangan dunia digital saat ini membuat lalu lintas pengiriman data elektronik semakin ramai dan sensitif. Hampir setiap orang melakukan transaksi data digital setiap hari. Data yang dipertukarkan pun juga bervariasi baik dari jenisnya maupun tingkat kerahasiaannya. Mulai dari data pribadi, data organisasi sampai dengan data negara yang sangat rahasia. Hal inilah yang menuntut adanya pengamanan terhadap proses pengiriman data tersebut sehingga tidak diketahui dan kemudian dimanfaatkan untuk kepentingan pihak ketiga. Telah banyak ditemukan teknik-teknik dalam pengamanan data, baik teknik klasik maupun modern.

Dalam dunia kriptografi, suatu algoritma dikatakan aman jika untuk memecahkannya dibutuhkan waktu dan biaya yang relatif besar. Di samping itu, persamaan matematis yang menggambarkan operasi algoritmanya cukup rumit. Metode vigènere cipher yang klasik sudah bisa dilakukan kriptanalisis terhadapnya dengan menentukan panjang kunci melalui metode Kasiski. Atas dasar itulah penulis mencoba mengembangkan metode vigènere cipher dengan lebih lanjut dengan memodifikasi operasi terhadap kunci di samping menggabungkannya dengan cipher substitus homofonik.

Penggabungan vigènere cipher dengan cipher substitusi homofonik bertujuan mempersulit kriptanalis melakukan pemecahan kunci karena menghilangkan jejak panjang kunci yang ditinggalkan oleh metode vigènere cipher klasik.

Vigènere cipher sendiri adalah bagian dari cipher subsitusi. Cipher substitusi adalah tipe enkripsi pesan yang intinya adalah mengubah isi dari pesan dengan teks lain. Cipher subsitusi dapat dikelompokkan berdasarkan jumlah karakter hasil enkripsi relatif dibandingkan plainteksnya. Klasifikasi tersebut antara lain :

1. **Cipher abjad-tunggal** (*monoalphabetic cipher* atau cipher substitusi sederhana-*simple substitution cipher*)
   - Satu karakter dalam plainteks diganti dengan satu karakter yang bersesuaian sehingga fungsi enkripsi-dekripsinya satu ke satu.
   - Jika plainteks terdiri dari huruf abjad, maka jumlah kemungkinan susunan huruf-huruf cipherteks yang dapat dibuat adalah sebanyak 26!.
   - Sedangkan jika terdiri dari karakter ASCII maka kemungkinannya menjadi 256!.
   - Caesar cipher adalah kasus khusus dari cipher abjad tungal di mana susunan huruf cipherteks diperoleh dengan menggeser huruf-huruf alfabet sejauh tiga karakter.
   - Jumlah kunci di dalam cipher abjad-tunggal sama dengan jumlah cara menyususn ke-26 huruf abjad tersebut, yaitu sebanyak 26! yang juga menyatakan jumlah kunci untuk menyusun huruf-huruf alfabet ke dalam tabel substitusi.
   - Contoh tabel substitusi :
     P_i : A B C D E F G H I J K L M
     C_i : D I Q M T B Z S Y K V O F

     P_i : N O P Q R S T U V W X Y Z
     C_i : E R J A U W P X H L C N G
   - Sehingga untuk plainteks "**TUGASMAKALAH**" dienkripsi menjadi "**PXZDWFDVDOS**".
2. **Cipher substitusi homofonik** (*homophonic substitution cipher*)
   - Seperti cipher abjad tunggal tetapi setiap karakter plainteks dapat dipetakan menjadi salah satu karakter cipherteks yang mungkin.
   - Fungsi enkripsi-dekripsi memetakan satu ke banyak.
   - Cipher substitusi homofonik pertama kali ditemukan pada tahun 1401 oleh wanita bangsawan Mantua.
   - Cipher substitusi homofonik lebih sulit dipecahkan daripada cipher abjad tunggal. Namun, dengan *known-plaintext attack* dapat dipecahkan sedangkan dengan *ciphertext-only attack* lebih sulit.
3. **Cipher abjad majemuk** (*Polyabathetic substitution cipher*)
   - Merupakan cipher substitusi ganda yang melibatkan kunci berbeda.
   - Cipher abjad majemuk dibuat dari sejumlah cipher abjad tunggal, masing-masing dengan kunci yang berbeda. Kebanyakan cipher abjad majemuk adalah cipher subsitusi periodik yang didasarkan pada periode m.
   - Contoh cipher substitusi periodik adalah vigènere cipher yang ditemukan kriptologi Prancis, Blaise de Vigènere pada abad 16.

2 Vigènere Cipher

2.1 Konsep Dasar

Vigènere cipher mungkin contoh terbaik dari cipher abjad majemuk "manual". Algoritma ini ditemukan oleh diplomat sekaligus kriptologis dari Prancis, Blaise de Vigènere pada abad 16. Vigènere cipher dipublikasikan pada tahun 1856, tetapi algoritma ini baru dikenal luas 200 tahun kemudian dan berhasil dipecahkan oleh Babbage dan Kasiski pada pertengahan abad 19.

Vigènere cipher digunakan oleh tentara konfederasi (*Confederate Army*) pada Perang Sipil Amerika (American Civil War). Perang Sipil terjadi setelah Vigènere Cipher berhasil dipecahkan. Hal ini diilustrasikan oleh kutipan pernyataan Jenderal Ulysess S. Grant: "*It would sometimes take too log to make translation of intercepted dispatches for us to receive any benefit from them, but sometimes they gave useful information*".

Vigènere cipher menggunakan Bujursangkar vigènere untuk melakukan enkripsi (l**ihat tabel 1**). Kolom paling kiri menyatakan huruf-huruf kunci sedangkan baris paling atas menyatakan huruf-huruf plainteks. Setiap baris dalam bujursangkar menyatakan huruf-huruf cipheteteks yang diperolehdengan *Caesar Cipher*, yang mana jauh pergeseran huruf plainteks ditentukan oleh nilai decimal oelh huruf kunci tersebut (a = 0, b = 1,…, z = 25). Sebagai contoh, huruf kunci c menyatakan huruf plainteks digeser sejauh 2 huruf ke kanan dari susunan alfabetnya.

	A	B	C	D	E	F	G	H	I
A	A	B	C	D	E	F	G	H	I
B	B	C	D	E	F	G	H	I	J
C	C	D	E	F	G	H	I	J	K
D	D	E	F	G	H	I	J	K	L
E	E	F	G	H	I	J	K	L	M

F	F	G	H	I	J	K	L	M	N
G	G	H	I	J	K	L	M	N	O
H	H	I	J	K	L	M	N	O	P
I	I	J	K	L	M	N	O	P	Q
J	J	K	L	M	N	O	P	Q	R
K	K	L	M	N	O	P	Q	R	S
L	L	M	N	O	P	Q	R	S	T
M	M	N	O	P	Q	R	S	T	U
N	N	O	P	Q	R	S	T	U	V
O	O	P	Q	R	S	T	U	V	W
P	P	Q	R	S	T	U	V	W	X
Q	Q	R	S	T	U	V	W	X	Y
R	R	S	T	U	V	W	X	Y	Z
S	S	T	U	V	W	X	Y	Z	A
T	T	U	V	W	X	Y	Z	A	B
U	U	V	W	X	Y	Z	A	B	C
V	V	W	X	Y	Z	A	B	C	D
W	W	X	Y	Z	A	B	C	D	E
X	X	Y	Z	A	B	C	D	E	F
Y	Y	Z	A	B	C	D	E	F	G
Z	Z	A	B	C	D	E	F	G	H

	J	K	L	M	N	O	P	Q	R
A	J	K	L	M	N	O	P	Q	R
B	K	L	M	N	O	P	Q	R	S
C	L	M	N	O	P	Q	R	S	T
D	M	N	O	P	Q	R	S	T	U
E	N	O	P	Q	R	S	T	U	V
F	O	P	Q	R	S	T	U	V	W
G	P	Q	R	S	T	U	V	W	X
H	Q	R	S	T	U	V	W	X	Y
I	R	S	T	U	V	W	X	Y	Z
J	S	T	U	V	W	X	Y	Z	A
K	T	U	V	W	X	Y	Z	A	B
L	U	V	W	X	Y	Z	A	B	C
M	V	W	X	Y	Z	A	B	C	D
N	W	X	Y	Z	A	B	C	D	E
O	X	Y	Z	A	B	C	D	E	F
P	Y	Z	A	B	C	D	E	F	G
Q	Z	A	B	C	D	E	F	G	H
R	A	B	C	D	E	F	G	H	I
S	B	C	D	E	F	G	H	I	J
T	C	D	E	F	G	H	I	J	K
U	D	E	F	G	H	I	J	K	L
V	E	F	G	H	I	J	K	L	M
W	F	G	H	I	J	K	L	M	N
X	G	H	I	J	K	L	M	N	O
Y	H	I	J	K	L	M	N	O	P
Z	I	J	K	L	M	N	O	P	Q

	S	T	U	V	W	X	Y	Z
A	S	T	U	V	W	X	Y	Z
B	T	U	V	W	X	Y	Z	A
C	U	V	W	X	Y	Z	A	B
D	V	W	X	Y	Z	A	B	C
E	W	X	Y	Z	A	B	C	D
F	X	Y	Z	A	B	C	D	E
G	Y	Z	A	B	C	D	E	F
H	Z	A	B	C	D	E	F	G
I	A	B	C	D	E	F	G	H
J	B	C	D	E	F	G	H	I
K	C	D	E	F	G	H	I	J
L	D	E	F	G	H	I	J	K
M	E	F	G	H	I	J	K	L
N	F	G	H	I	J	K	L	M
O	G	H	I	J	K	L	M	N
P	H	I	J	K	L	M	N	O
Q	I	J	K	L	M	N	O	P
R	J	K	L	M	N	O	P	Q
S	K	L	M	N	O	P	Q	R
T	L	M	N	O	P	Q	R	S
U	M	N	O	P	Q	R	S	T
V	N	O	P	Q	R	S	T	U
W	O	P	Q	R	S	T	U	V
X	P	Q	R	S	T	U	V	W
Y	Q	R	S	T	U	V	W	X
Z	R	S	T	U	V	W	X	Y

Tabel 1 Bujursangkar Vigènere

Bujursangkar vigener digunakan untuk memperoleh cipherteks dengan menggunakan kunci yang sudah ditentukan. Jika panjang kunci lebih pendek daripada panjang plainteks, maka kunci diuklang penggunaannya (system periodik). Bila panjang kunci adalah m, maka periodenya dikatakan m.

Sebagai contoh, jika plainteks adalah "TUGASMAKALAH" dan kunci adalah "KEY", maka penggunaan kunci secara periodic adalah sebagai berikut:

Plainteks : T UG A S M A K A L A H
Kunci : K E Y K E Y K E Y K E Y

setiap huruf plainteks dienkripsi dengan setiap huruf kunci di bawahnya.

Untuk melakukan enkripsi dengan vigener cipher, lakukan pada bujursangkar vigènere

sebagai berikut: tari garis vertical dari huruf plainteks ke bawah, lalu tarik garis mendatar dari huruf kunci ke kanan. Perpotongan kedua garis tersebut menyatakan huruf cipherteksnya. Sebagai contoh di atas, tarik garis vertical dari huruf T dan tarik garis mendatar dari huruf K, perpotongannya adalah kotak yang berisi huruf D (lihat tabel 2).

	S	T	U	V	W	X	Y	Z
A	S	T	U	V	W	X	Y	Z
B	T	U	V	W	X	Y	Z	A
C	U	V	W	X	Y	Z	A	B
D	V	W	X	Y	Z	A	B	C
E	W	X	Y	Z	A	B	C	D
F	X	Y	Z	A	B	C	D	E
G	Y	Z	A	B	C	D	E	F
H	Z	A	B	C	D	E	F	G
I	A	B	C	D	E	F	G	H
J	B	C	D	E	F	G	H	I
K	C	D	E	F	G	H	I	J
L	D	E	F	G	H	I	J	K
M	E	F	G	H	I	J	K	L
N	F	G	H	I	J	K	L	M
O	G	H	I	J	K	L	M	N
P	H	I	J	K	L	M	N	O
Q	I	J	K	L	M	N	O	P
R	J	K	L	M	N	O	P	Q
S	K	L	M	N	O	P	Q	R
T	L	M	N	O	P	Q	R	S
U	M	N	O	P	Q	R	S	T
V	N	O	P	Q	R	S	T	U
W	O	P	Q	R	S	T	U	V
X	P	Q	R	S	T	U	V	W
Y	Q	R	S	T	U	V	W	X
Z	R	S	T	U	V	W	X	Y

Tabel 2 Enkripsi huruf T dengan kunci K

Hasil enkripsi seluruhnya adalah sebagai berikut:

Plainteks : T UG A S M A K A L A H
Kunci : K E Y K E Y K E Y K E Y
Cipherteks: D Y E K W K K P Y V E F

Dapat diamati bahwa sebuah huruf dapat dienkripsi menjadi huruf yang berbeda. Hal ini merupakan karakteristik dari cipher abjad majemuk. Pada cipher substitusi sederhana, setiap huruf dalam cipherteks selalu menggantikan huruf plainteks tertentu sedangkan pada cipher abjad majemuk setiap huruf cipherteks dapat memiliki kemungkinan banyak huruf plainteks.

Jadi, dengan menggunakan Vigener Cipher, kita dapat mencegah frekuensii huruf-huruf di dalam cipherteks yang mempunyai pola tertentu yang sama sebagaimana yang diperlihatkan pada cipher substitusi sederhana.

Dekripsi pada Vigener Cipher dilakukan dengan cara yang berkebalikan, yaitu menarik garis mendatar dari huruf kunci sampi ke huruf cipherteks yang dituju, lalu dari huruf cipherteks tarik garis vertical ke atas sampai huruf plainteks.

Untuk memecahkan Vigènere Cipher, cukup menemukan kuncinya. Jika periode kunci diketahui dan tidak terlalu panjang, maka kunci dapat ditentukan dengan menulis program untuk melakukan *exhaustive key search*. Sebagai contoh jika diberikan cipherteks yang dihasilkan dengan vigènere cipher:

ASGERAEWEOHPH

dan diperoleh informasi bahwa panjang kunci adalah 3 huruf dan plainteks ditulis dalam bahasa Inggris, maka *running* program dengan mencoba semua kemungkinan kunci yang panjangnya 3 huruf, lalu periksa apakah hasilnya memiliki arti dalam bahasa Inggris. Cara ini membutuhkan percobaan sebanyak 26^n kali. Semakin panjang kunci (semakin besar periode), semakin banyak usaha percobaan yang harus dilakukan.

2.2 Varian

2.2.1 The Full Vigènere Cipher

The full vigènere cipher mirip dengan vigener cipher biasa yang menggunakan kunci dengan menggunakan kata atau frase. Namun, t*he full vigènere cipher* menggunakan keyword yang merupakan permutasi dari rangkaian alphabet daripada deretan huruf saja. Bujursangkarnya dapat dilihat dalam tabel 3.

	A	B	C	D	E	F	G	H	I
A	F	W	Y	G	B	D	Z	I	X
B	G	A	Y	O	M	X	C	W	H
C	L	Y	B	O	N	I	Z	C	K
D	F	D	I	V	Z	H	E	G	U
E	Q	T	G	S	A	R	Z	P	B
F	M	X	C	P	O	N	F	W	E
G	F	E	P	Z	D	Y	O	I	C
H	O	B	Z	M	N	Y	A	L	U
I	N	F	Y	D	Z	H	O	E	A
J	S	A	U	M	E	K	O	N	J
K	E	W	N	D	L	X	U	K	O

L	M	B	L	T	A	S	N	X	J
M	J	I	O	C	W	H	U	M	B
N	E	S	C	Y	G	Z	R	U	D
O	B	Z	K	J	W	P	U	Y	L
P	Z	Y	O	U	M	W	N	B	V
Q	I	V	E	H	Q	J	F	D	K
R	C	B	U	Y	T	G	N	P	E
S	V	E	R	D	S	Q	W	O	G
T	W	B	R	A	P	O	D	F	T
U	R	B	O	M	A	N	T	C	D
V	C	Z	B	N	G	L	O	Y	F
W	A	S	P	Y	Q	R	G	F	D
X	P	Q	O	Z	M	X	Y	W	S
Y	M	Y	X	O	A	N	V	C	L
Z	Q	P	W	O	Y	Z	N	X	H

	J	K	L	M	N	O	P	Q	R
A	V	H	A	L	K	J	U	E	T
B	Z	N	B	S	T	E	V	P	D
C	M	J	X	H	G	A	E	T	Q
D	Y	B	T	K	P	W	C	S	N
E	H	X	F	J	O	Y	K	U	D
F	V	I	Q	B	D	G	H	L	Z
G	W	B	Q	X	J	S	N	H	A
H	R	D	C	K	P	H	Q	F	X
I	G	P	W	C	V	M	I	J	T
J	F	C	P	T	H	Y	V	L	G
K	F	V	M	T	C	S	R	I	P
L	W	D	U	V	O	C	K	Q	P
M	V	G	N	Y	F	P	K	L	Y
N	P	O	F	A	H	T	V	K	Q
O	A	X	H	V	R	M	I	F	Q
P	D	G	P	K	T	A	R	H	C
Q	U	Z	G	R	A	T	P	C	S
R	S	D	Q	Z	O	A	M	F	L
S	F	C	P	Y	J	U	N	H	L
T	C	M	X	Y	G	U	E	Q	N
U	V	L	Q	J	Z	E	S	K	U
V	X	K	M	W	H	R	D	P	J
W	E	Z	H	O	T	V	I	B	X
X	L	N	U	K	V	T	I	J	D
Y	U	W	B	I	T	G	K	Q	J
Z	M	S	J	L	I	U	A	G	C

	S	T	U	V	W	X	Y	Z
A	C	N	R	P	S	M	O	Q
B	K	Q	U	L	F	R	I	J
C	F	V	D	W	P	R	S	U
D	Q	J	M	O	A	L	X	R
E	W	I	M	V	C	N	L	E
F	U	K	R	Y	J	T	A	S
G	R	T	G	L	K	V	M	U
H	J	E	S	T	G	I	W	V
I	R	B	Q	L	K	S	U	X
J	Q	Z	D	X	I	R	B	W
K	Z	G	Q	J	Y	H	A	B
L	I	F	Z	G	R	E	Y	H
M	D	X	E	R	Q	S	Z	A
N	I	M	B	X	J	L	W	N
O	G	O	S	N	C	T	E	D
P	X	J	I	E	L	Q	S	F
Q	Y	M	W	O	L	B	X	N
R	W	K	I	R	X	J	H	V
S	X	I	K	Z	T	B	A	M
T	I	Z	V	L	S	H	K	J
U	I	W	Y	P	H	F	X	G
V	S	A	I	Q	U	E	V	T
W	N	U	J	L	K	W	C	M
X	G	B	R	E	A	F	C	H
Y	P	Z	H	R	S	E	D	F
Z	T	E	F	V	D	K	B	R

Tabel 3 Bujursangkar *The Full Vigènere Cipher*

Untuk melakukan enkripsi dan dekripsi caranya sama dengan menggunakan bujursangkar vigènere yang sebelumnya (tabel 1 dan 2). Keunggulan *The Full Vigènere Cipher* disbanding vigener cipher sederhana terletak pada kebutuhan akan frekuensi relatif secara keseluruhan yang diperlukan untuk memecahkan pesan.

2.2.2 The Auto-key Vigènere Cipher

Vigènere Cipher adalah contoh awal tentang cipher aliran. Cipher aliran adalah metode enkripsi yang mengenkripsi huruf berdasarkan posisinya dalam teks. Idealnya, kunci yang digunakan tidak pernah berulang. Beberapa cipher aliran menggunakan palinteks dan/atau cipherteks sebagai bagian dari prosesnya, seperti The Auto-key Vigènere Cipher ini.

Enkripsi bermula dengan kunci primer $k_0,k_1,\ldots,k_{l-1}$. Enkripsi karakter sampai karakter ke l mengunakan cara yang sama dengan vigènere cipher sederhana. Setelah itu, enkripsi dilanjutkan dengan menggunakan karakter dari plainteks.

Sebagai contoh untuk mempermudahnya kita mempunya pesan "TUGASMAKALAH" yang akan dienkripsi dengan kunci "KEY". Maka hasilnya adalah sebagai berikut:

Plainteks: **TUGASMAKA**LAH

Key: KEY**TUGASMAKA**

Cipherteks: DYETMSACMLKH

Untuk mendekripsikannya kita cukup mengetahui kuncinya. Jika sudah mendekripsi dengan kunci, plainteks hasilnya dekripsinya digunakan sebagai kunci selanjutnya.

Kehati-hatian sangat diperlukan saat proses enkripsi, karena kesalahan satu huruf membuat error kepada kesuluruhan teks. Hal ini juga berlaku saat pengiriman cipherteks.

2.2.3 The Running-key Vigènere Cipher

Variasi lain dari vigènere cipher adalah *running-key vigènere cipher* yang menggunakan teks yang memiliki arti atau cukup dikenal dalam masyarakat. Teks ini bisa berupa buku yang dimiliki oleh pengirim dan penerima pesan.

Sebagai contoh kita memiliki sebuah teks yang ingin dienkripsi "TUGASMAKALAHKU". Lalu kita menggunakan kunci dari lirik lagu SNADA yang berjudul "Demi Masa".

Plainteks: T U G A S M A K A L A H K U

Kunci: D E M I M A S A S E S U N G

Cipherteks: WY S I E M S K S P S B X A

Untuk mendekripsinya kit hanya perlu melakukan dekripsi seperti halnya pada metode vigènere cipher sederhana dengan kunci berupa teks yang dimiliki kedua pihak yang terlibat.

2.2.4 One Time Pad (OTP)

OTP mengunakan kunci sepanjang ukuran teks yang ingin dienkripsi secara acak. Setelah dipakai, kertas note tersebut dibuang untuk menjaga kerahasiaan dan juga randomisasi kunci selanjutnya. OTP ditemukan pada tahun 1917 oleh Major Joseph Mauborgne.

Aturan enkripsi yang digunakan sama persis dengan vigener cipher biasa. Pengirim pesan mengenkripsi sebuah karakter plainteks dengan sebuah karakter dari kunci. Perhatikan bahwa panjang kunci sama dengan panjang plainteks sehingga tidak terjadi pengulangan/periodisasi kunci.

Sebagai contoh kita ingin mengenkripsi

Plainteks: TUGASMAKALAH dengan

Kunci : ADAFYMAGOEAG maka

Cipherteks: TXGFQYAQOPAN

Kekuatan OTP terletak pada kunci yang acak sehingga menghasilkan cipherteks yang seluruhnya acak juga. Selain itu cipherteks yang sama mungkin menghasilkan teks berbeda yang sama-sama memiliki arti sehingga membingungkan kriptanalis. Contoh

Cipherteks:	NEOHRTA	NEOHRTA
Kunci:	MAXGNQA	UEEPRHA
Plainteks:	BERBEDA	TAKSAMA

2.3 Kelemahan

Selain OTP yang bisa dikatan tak dapat dipecahkan, semua variasi dari vigènere cipher di atas memiliki kelemahan jika dianalisi secara frekuensi meski auto-key dan running-key menghilangkan jejak pengulangan. Hal ini disebabkan plainteks yang digunakan sebagai kunci. Meski plainteks tidak pernah berulang, tetap saja menyediakan informasi. Hal ini disebabkan huruf dengan frekuensi tinggi di kunci akan sering mencipher huruf dengan frekuensi yang tinggi juga di plainteks.

OTP sendiri meskipun bisa dikatakan tdiak dapat dipecahkan pada kenyataannya tidak digunakan secara umum sebagai algoritma kriptografi. Hal ini disebabkan:

1) Karena panjang kunci harus sama dengan panjang plainteks, maka OTP hanya cocok untuk mengenkripsi pesan yang berukuran kecil. Hal ini menjadi problem dalam penyimpanan dan pendistribusian kunci.
2) Karena kunci dibangkitkan secara acak, tidak mungkin pengirim membangkitkan kunci yang sama secara simultan.

2.4 Metode Kasiski untuk Analisis Panjang Kunci

Metode Kasiski adalah metode untuk mendapatkan panjang kunci yang digunakan untuk melakukan dekripsi. Metode ini mengambil keuntungan dari fakta bahwa suatu bahasa tidak hanya mengandung sebuah karakter tunggal yang sering muncul tetapi juga pasangan karakter yang membentuk biplet, triplet dan sebagainya yang juga seing muncul. Kita dapat menggunakan fakta ini untuk mendapatkan posisi triplet yang sering muncul di cipherteks. Hal ini bisa terjadi jika frekuentif triplet diencipher oleh bagian yang sama dari kunci. Dengan mengetahui posisi-posisi dari triplet tersebut kita dapat menebak panjang kunci.

Mari kita perhatikan contoh berikut ini. Diberikan cipherteks sebagai berikut dalam bahasa Inggris:

```
LJVBQ STNEZ LQMED LJVMA MPKAU
FAVAT LJVDA YYVNF JQLNP LJVHK
VTRNF LJVCM LKETA LJVHU YJVSF
KRFTT WEFUX VHZNP
```

Jika kita menggunakan metode Kasiski, kita perhatikan bahwa triplet LJV terus berulang. Jarak antara kemunculan triplet tersebut sebagai berikut:

Kemunculan ke	Jarak
2	15
3	15
4	15
5	10
6	10

Tabel 4 Jarak kemunculan-kemunculan LJV

Dari tabel ini dapat ditarik kesimpulan bahwa panjang kunci adalah 5. Sekarang kita pisahkan cipherteks ke dalam 5 kategori dan kita lakukan analisis frekuensi pada tiap kategori. Seperti dapat dilihat pada tabel berikut:

K	Karakter				Tersering muncul
1	LSLLM	FLYJL	VLLLY	KWV	L
2	JTQJP	AJYQJ	TJKJJ	REH	J
3	VNMVK	VVVLV	RVEVV	FFZ	V
4	BEEMA	ADNNH	NCTHS	TUN	N
5	QZDAU	TAFPK	FMAUF	TXP	A

Dalam setiap kategori, huruf paling sering muncul mungkin berkoresponden dengan huruf E,T,I,N, atau R dalam plainteks. Akan lebih mudah jika memiliki lebih banyak teks, karena sepertinya E lebih banyak muncul dari huruf manapun di plainteks. Akan tetapi, kita memiliki informasi tambahan lagi : trplet yang paling serng muncul dalam bahasa Inggris adalah THE, dan mungkin saja ini berkoresponden dengan triplet LJV di cipherteks. Dengan sedikit informasi ini kita bisa mencoba beberapa kemungkinan seperti dapat dilihat dalam tabel berikut ini:

Kategori	P	C	K
1	T	L	18 = S
2	H	J	2 = C
3	E	V	17 = R
4	N	N	0 = A
5	O	A	12 = M

Tabel 5 Pencarian nilai kunci

Kita dapatkan kunci SCRAM sehingga bisa kita dapatkan plainteksnya:

```
THEBE ARWEN TOVER THEMO UNTAI
NYEAH THEDO GWENT ROUND THEHY
DRANT THECA TINTO THEHI GHEST
SPOTH ECOUL DFIND
```

3 Vigènere Cipher Homofonik

3.1 Konsep Umum

Vigènere cipher homofonik adalah paduan antara permutasi, vigènere cipher dan cipher substitusi homofonik. Seperti dijelaskan di atas, vigènere cipher memiliki kelebihan dan kekurangan sebagai algoritma kriptografi. Dengan vigènere cipher homofonik, penulis berusaha mengambil sisi positif dari vigènere cipher dan menutupi kekurangannya dengan memadukannya dengan metode tersebut di atas.

Vigènere cipher homofonik bekerja pada rangkaian huruf ASCII (256 karakter). Hal ini bertujuan agar variasi kunci yang memungkinkan

menjadi semakin banyak sehuingga mempersulit kriptanalis untuk memecahkan cipherteks (dalam makalah ini semua karakter akan ditulis dalam format heksadesimal untuk mengatasi ketidakmampuan teks editor untuk menulis karakter-karakter *nullspace*).

Fungsi permutasi digunakan untuk memvariasikan kunci tanpa berdasarkan plainteks yang membuat kriptanalis mampu menganalisis frekuensi, melainkan dari kombinasi dari kunci yang rahasia denga plainteks yang rahasia juga. Sehubungan dengan penggunaan fungsi permutasi ini, kunci yang digunakan berupa deretan angka.

Komponen lain yang digunakan dalam algoritma ini adalah cipher substitusi homofonik. cipher substitusi homofonik berfungsi menghilangkan jejak frekuensi yang mungkin muncul meskipun kecil kemungkinan terjadinya. Hal ini sesuai dengan pendapat Murphy dalam Murphy's Law: "*If something can go wrong, it will*". Akan tetapi, berbeda denga cipher substitusi homofonik biasa yang setiap hurufnya dipetakan ke jumlah karakter yang sama (missal satu ke dua), dalam algoritma yang dirangkai penulis dalam makalah ini, tidak semua huruf dipetakan ke karakter lain dengan jumlah yang sama. Lebih jelasnya akan dijelaskan dalam algorimta umu dalam **subbab 3.3.**

3.2 Lingkungan Pengembangan Aplikasi

Program aplikasi algoritma transposisi ini dikembangkan pada lingkungan sebagai berikut:

- Intel(R) Pentium(R) 4 2.0 GHz Core Speed 1999.8 MHz Bus Speed 400.0 MHz
- Data Cache
  - L1 4-ways 8 KB
  - L2 8-ways 512 KB
- Memory DDR-SDRAM 640 MB 133.3 MHz
- Nvidia GeForce4 MX4000 128 MB
- Windows XP SP2 dengan .Net Framework 2.0
- Kakas pemrograman Microsoft Visual Studio .NET berbasis C# language

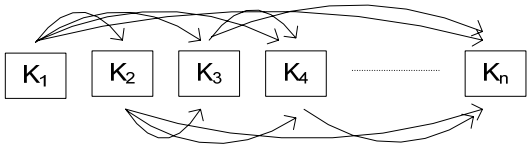
3.3 Algoritma Umum

3.3.1 Fungsi Permutasi

Penggunaan fungsi permutasi dimaksudkan untuk memvariasikan kunci tanpa berdasarkan plainteks yang membuat kriptanalis mampu menganalisis frekuensi melainkan dari kunci itu sendiri. Sebagaimana telah dijelaskan sebelumnya, kunci yang digunakan dalam algoritma ini berupa deretan angka.

Secara garis besar penggunaan fungsi permutasi adalah sebagai berikut:

Kunci $K = K_1\ K_2 \ldots\ldots K_n$
Kunci Cipher K_c:
$K_{c1} = K_1 * K_2$ .....(i)
$K_{c2} = K_1 * K_3$ .....(ii)
$K_{c3} = K_1 * K_4$ .....(iii)
.
.
.
$K_{cm} = K_2 * K_3$ .....(iv)
$K_{cm+1} = K_2 * K_4$
.
.
.
$K_{cn} = K_{n-1} * K_n$



Gambar 1 Skema perkalian permutasi

Akan tetapi dalam setiap langkahnya, dua digit angka terdepan dari hasil perkalian yang dikalikan lagi dengan nilai ASCII dari karakter plainteks, digunakan untuk menggantikan kunci dengan indeks terkecil dari dua buah kunci yang dikalikan. Dengan kata lain, hasil dari perkalian (i)*P_1 mensubstitusi K_1, hasil dari perkalian (ii)*P_2 mensubstitusi K_1, hasil dari perkalian (iii)*P_3 mensubstitusi K_1, hasil dari perkalian (iv)*P_{cm} mensubstitusi K_2.

Agar lebih jelasnya diberikan contoh sebagai berikut:
K = 41513287, sehingga
$K_1 = 41$
$K_2 = 51$
$K_3 = 32$
$K_4 = 87$

Untuk memperoleh pengelompokan dua digit angka-angka ini digunakan fungsi sebagai berikut:

```
function Classified(
  key:String
) → arrayOfInteger;
{Mengelompokkan kunci proses dari
keyWord(kunci dari user)}

Deklarasi
  pjg  : integer;
  arr  : arrayOfInteger;
  i,j  : integer;

Algoritma
  begin
    pjg := kunci.Length;
    arr := new int[pjg / 2];
    j := 0;
    for i := 0 to pjg / 2 do
    begin
      arr[i] := TakeTwo(kunci, j);
      j += 2;
    end;
    {end for}

  return arr;
  end.
{endfunciton}
```

Dalam fungsi di atas digunakan fungsi untuk mengambil dua digit dari kunci masukan sebagai berikut:

```
function TakeTwo(
  key : String,
  indeks : integer
) → integer
{Mengembalikan dua digit awal
dari key}

Deklarasi
  d1 : integer;
  d2 : integer;

Algoritma
  begin
    d1 := (key[indeks] – 48) * 10;
    d2 := key[indeks + 1] – 48;
    return d1 + d2;
  end.
{endfunction}
```

Setelah dilakukan pengelompokan digit-digit kunci, algoritma dilanjutkan dengan mencipher secara vigènere tiap karakter dalam plainteks dengan hasil kali permutasi dari kelompok-kelompok kunci yang sudah disusun di atas.

$C_i = (P_i + \text{factor}) \bmod 256$, dengan faktor adalah hasil perkalian secara permutasi

faktor = arr[i] * arr[i+1].

Kemudian faktor yang didapatkan di atas digunakan untuk menggantikan kunci dengan indeks terkecil dari dua buah kunci yang dikalikan (i).

3.3.2 Cipher Substitusi homofonik

Peran cipher substitusi homofonik terletak pada saat sebuah karakter plainteks sudah dienkripsi. Sebuah karakter hasil enkripsi akan mendapat tambahan karakter didepannya jika

faktor div 256 merupakan bilangan genap

sedangkan jika hasilnya ganjil, karakter tersebut hanya dipetakan ke satu karakter.

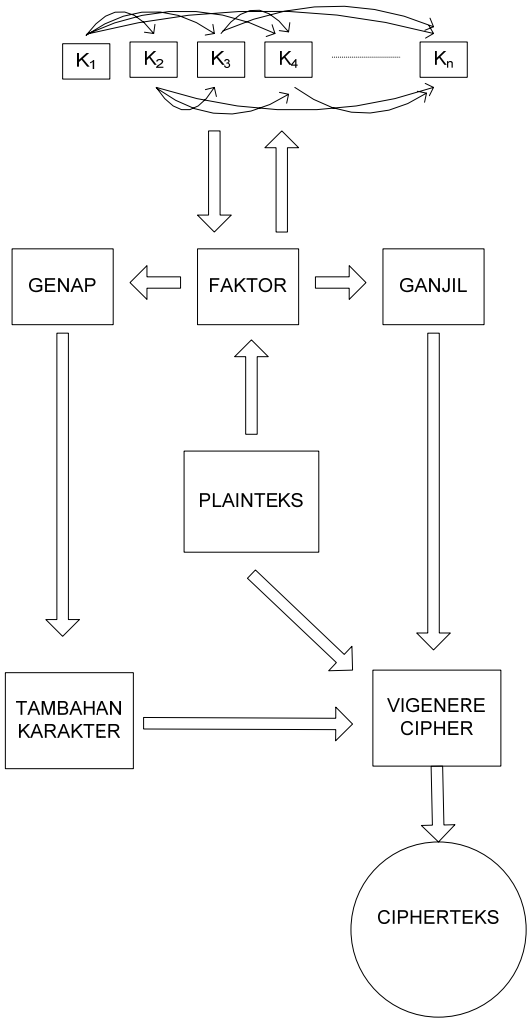
Maksud dari penggunaan cipher substitusi homofonik ini adalah untuk mengaburkan frekuensi kemunculan tiap-tiap huruf karena karakter plainteks yang dienkripsi menjadi dua karakter cipherteks memiliki karakter awal yang hanya merupakan *dummy* / ilusi. Selain itu, cipher substitusi homofonik juga bertujuan membingungkan kriptanalis dalam mencari pola kata dalam cipherteks.

3.3.3 Proses Enkripsi

Enkipsi dilakukan dengan mengikuti aliran data seperti sudah dijelaskan di atas. Algoritma diawali dengan melakukan pengelompokkan digit-digit kunci yang kemudian dilakukan perkalian antar elemen kunci. Setiap perkalian dilakukan, hasilnya merupakan jumlah pergeseran karakter plainteks. Hasil perkalian juga menentukan apakah karakter plainteks tersebut dipetakan menjadi satu atau dua karakter. Dan kembali, hasil perkalian tersebut menggantikan kunci sebelumnya.

Sekarang timbul pertanyaan, bagaimana jika semua elemen kunci sudah selesai dipermutasi? Hal ini ditangani dengan mengulang perkalian permutasi dari elemen pertama dari kunci. Pengulangan ini tidak seperti pengulangan yang terjadi pada vigènere cipher biasa yang bisa menimbulkan pengulangan pula pada cipherteksnya. Hal tersebut dimungkinkan karena kunci mengalami pergantian setiap dilakukan perkalian sehingga bisa dijamin bahwa kunci terus berubah-ubah.

Diagram aliran data pada proses enkripsi digambarkan sebagai berikut:



Gambar 2 Diagram aliran data proses Enkripsi

Sedangkan algoritma untuk proses enkripsi dijabarkan sebagai berikut:

```
procedure Encrypt(
  key : String,
  pesan : arrayOfInteger,
  cipher : arrayOfInteger
);
{Melakukan proses enkripsi terhadap
pesan dengan kunci key dan
menghasilkan cipherteks}
{arrayOfInteger merepresentasikan
nilai ASCII dari pesan yang akan
dienkripsi dan cipherteks hasil
enkripsi}
```

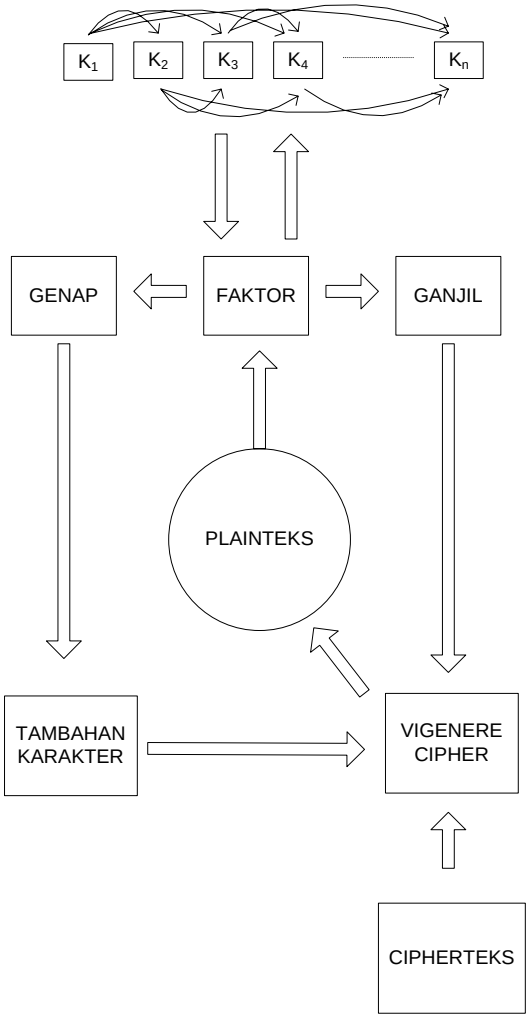
```
Deklarasi
  i,j,k,l     : integer;
  bt          : integer;
  faktor      : integer;
  arr         : arrayOfInteger;

Algoritma
  begin
    arr := Classified(key);
    j := 0;
    k := 0;
    l := 1;
    for i := 0 to pesan.Length do
    begin
      faktor := arr[k] * arr[l];
      bt = pesan[i];
      if ((faktor div 256)mod2=0)
      then
      begin
        cipher[j]:=(bt-faktor/256+
                256) % 256;
        j := j + 1;
      end;
      cipher[j] := (bt+faktor%256)
              % 256;
      arr[k]=TakeTwo(faktor*bt,0);
      if (l = arr.Length - 1) then
      begin
        if (k = arr.Length-2)  then
        begin
          k = 0;
          l = 0;
        end;
        else
        begin
          k := k + 1;
          l = k;
        end;
      end;
      l ;= l + 1;
    end;
    {endfor}
  end.
  {end procedure}
```

3.3.4 Proses Dekripsi

Untuk melakukan dekripsi, kita cukup melakukan kebalikan dari proses enkripsi di atas. Hanya perlu diperhatikan di sini bahwa jika kita dapatkan faktor yang merupakan bilangan genap, kita cukup memajukan indeks pembacaan sebanyak satu indeks tanpa perlu melakukan apapun terhadap cipherteks di indeks tersebut. Hal ini dikarenakan karakter tersebut memang hanya ditujukan untuk membingungkan kriptanalis memecahkan kode. Dekripsi sudah bisa dilakukan terhadap satu karakter cipherteks saja.

Berikut digambarkan diagram aliran data pada proses dekripsi:



Gambar 3 Diagram aliran data proses dekripsi

Algoritma proses dekripsi secara Pascal-*like* dijabarkan sebagai berikut:

```
procedure Decrypt(
  key : String,
  cipher : arrayOfInteger,
  plain : arrayOfInteger
);
{Melakukan prosew dekripsi terhadap
cipherteks dengan kunci key dan
menghasilkan plainteks}
{arrayOfInteger di sini
merepresentasikan nilai ASCII dari
pesan yang akan dienkripsi dan
cipherteks hasil enkripsi}
```

```
Deklarasi
  i : integer;
  j : integer;
  k : integer
  l : integer;
  bt  : integer;
  faktor : integer;
  arr : arrayOfInteger;

Algoritma
  begin
    arr := Classified(key);
    j := 0;
    k := 0;
    l := 1;
    for i := 0 to pesan.Length do
    begin
      faktor := arr[k] * arr[l];
      if ((faktor div 256)mod2=0)
      then
      begin
        i := i + 1;
      end;
      bt = cipher[i];
      plain[j] := (bt-faktor%256)
                          % 256;
      arr[k] = TakeTwo(faktor*
               plain[j],0);
      if (l = arr.Length - 1) then
      begin
        if (k = arr.Length - 2)
        then
        begin
          k = 0;
          l = 0;
        end;
        else
        begin
               k := k + 1;
               l = k;
        end;
      end;
      l ;= l + 1;
    end;
    {endfor}
  end.
  {end procedure}
```

3.3.5 Contoh Permasalahan

Diberikan sebuah plainteks (dalam heksadesimal) sebagai berikut:

P = 21 AF 01 AB 3F F1

yang akan diekripsi dengan kunci 154295.

Langkah-langkahnya adalah sebagai berikut::

1. Kunci dikelompokkan dua-digit-dua digit menjadi

   K_1 = 15

   K_2 = 42

   K_3 = 95

2. Kalikan K_1 dengan K_2 untuk mendapatkan faktor.

   faktor = 15*42 = 630

   Gunakan hasil ini untuk mensubstitusi K_1 sehingga kunci untuk enkripsi selanjutnya menjadi 204295.

3. Cek hasil div faktor dengan 256.

   flag = 630 div 256 = 2 (genap)

4. Karena genap, tambahkan sebuah karakter pada cipherteks dengan persamaan

   C = (33-flag+256) mod 256 = 31 = 1F

5. Dekripsi 21 dengan persamaan vigènere cipher

   C = (33+ 630 mod 256) mod 256 = 151 = 97

   Maka karakter 21 dienkripsi menjadi 1F 97.

6. Lakukan hal yang sama dengan karakter-karakter selanjutnya:

   a. P = AF = 175
      - Kunci lama = 204295
      - Faktor = 1900
      - Kunci baru = 334295
      - Flag = 7 (ganjil)
      - Hasil = 1B

   b. P = 01 = 1
      - Kunci lama = 334295
      - Faktor = 3990
      - Kunci baru = 333995
      - Flag = 15 (ganjil)
      - Hasil = 97

   c. P = AB = 171
      - Kunci lama = 333995
      - Faktor = 1287
      - Kunci baru = 223995
      - Flag = 5 (ganjil)
      - Hasil = B2

   d. P = 3F = 63
      - Kunci lama = 223995
      - Faktor = 2090
      - Kunci baru = 133995
      - Flag = 8 (genap)
      - Hasil = 37 69

   e. P = F1 = 241
      - Kunci lama = 133995
      - Faktor = 3705
      - Kunci baru = 138995
      - Flag = 14 (genap)
      - Hasil = E3 6A

Maka cipherteks hasil enkripsinya adalah

1F 97 1B 97 B2 37 69 E3 6A

Untuk mendekripsinya dilakukan langkah-langkah sebagai berikut:

1. Kunci dikelompokkan dua-digit-dua digit menjadi

   K_1 = 15

   K_2 = 42

   K_3 = 95

2. Kalikan K_1 dengan K_2 untuk mendapatkan faktor.

   faktor = 15*42 = 630

3. Cek hasil div faktor dengan 256.

   flag = 630 div 256 = 2 (genap)

   Karena genap, karakter yang sedang dieksekusi diganti menjadi karakter berikutnya karena karakter tersebut hanya merupakan dummy sehingga C = 97 = 151

4. Dekripsi 21 dengan persamaan invers vigènere cipher

   P = (151- 630 mod 256) mod 256 = 33 = 21

   Gunakan hasil ini untuk mensubstitusi K_1 sehingga kunci untuk enkripsi selanjutnya menjadi 204295.

5. Lakukan hal yang sama dengan karakter-karakter selanjutnya:

a. C = 1B = 27
   - Kunci lama = 204295
   - Faktor = 1900
   - Flag = 7 (ganjil)
   - Hasil = AF
   - Kunci baru = 333995

b. C = 97 = 151
   - Kunci lama = 334295
   - Faktor = 3990
   - Flag = 15 (ganjil)
   - Hasil = 01
   - Kunci baru = 333995

c. C = B2 = 178
   - Kunci lama = 333995
   - Faktor = 1287
   - Flag = 5 (ganjil)
   - Hasil = AB
   - Kunci baru = 223995

d. C = 37 = 55
   - Kunci lama = 223995
   - Faktor = 2090
   - Flag = 8 (genap), C = 69 = 105
   - Hasil = 3F
   - Kunci baru = 133995

e. C = E3 = 227
   - Kunci lama = 133995
   - Faktor = 3705
   - Flag = 14 (genap), C = 6A
   - Hasil = F1
   - Kunci baru = 138995

Maka plainteks hasil dekripsinya adalah

21 AF 01 AB 3F F1

3.4 Kekuatan algoritma

Kekuatan algoritma vigènere cipher homofonik terletak pada penggunaan fungsi perkalian permutasi yang dipadukan dengan variasi cipher substitusi homofonik. Hal ini disebabkan karena:

1. Perkalian dengan permutasi membuat kunci memegang peranan yang sangat penting dan plainteks digunakan untuk semakin mengaburkan jejak frekuensi karena perkalian kunci yang rahasia dengan plainteks yang rahasia.
2. Substitusi kunci dengan perkalian antara kunci sebelumnya dengan plainteks sebelumnya membuat kedua elemen yang rahasia, yaitu kunci dan plainteks memegang peranan yang sangat penting
3. Perkalian dengan permutasi juga membuat kesalahan satu elemen kunci saja menjadi sangat fatal, karena akan mempengaruhi encipher selanjutnya.
4. Penggunaan kunci berupa sebuah bilangan membuat jumlah kemungkinan kunci yang harus dicoba jika ingin melakukan *exhaustive key search* menjadi sangat banyak. Jika diketahui panjang kunci adalah n, maka jumlah kemungkinan kunci adalah $100^{n/2}$.
5. Variasi cipher substitusi homofonik membuat jejak frekuensi dan hubungan antara plainteks dan cipherteks menjadi kabur sehingga mempersulit kriptanalis melakukan kriptanalisis.
6. Algoritma sederhana sehingga proses encipher-decipher dapat dilakukan dengan cepat.

3.5 Kelemahan Algoritma

Di samping memiliki kelebihan seperti sudah dijelaskan di atas, algoritma ini memiliki sedikit kelemahan-kelemahan sebagai berikut:

1. Jika kunci yang digunakan pendek, pencarian dengan *exhaustive key search* dapat menghasilkan kunci yang tepat dalam waktu singkat.
2. Panjan kunci harus merupakan bilangan genap. Jika ganjil, harus dilakukan padding dengan menambahkan *dummy* di akhir kunci.
3. Fungsi perkalian dengan permutasi kurang kompleks sehingga kompleksitas algoritma hanya n^2 yang berarti percobaan satu kunci dapat dilakukan dengan cepat.

4 Serangan-serangan Kriptanalis terhadap Vigènere Homofonik

4.1 Ciphertext-only Attack

Algoritma ini sangat kuat terhadap serangan berjenis ini kareena penggunaan perkalian permutasi dan variasi cipher substitusi homofonik. Hal ini disebabkan antara lain:

1. Perkalian dengan permutasi membuat kunci memegang peranan yang sangat penting dan plainteks digunakan untuk semakin mengaburkan jejak frekuensi karena perkalian kunci yang rahasia dengan plainteks yang rahasia.
2. Substitusi kunci dengan perkalian antara kunci sebelumnya dengan plainteks sebelumnya membuat kedua elemen yang rahasia, yaitu kunci dan plainteks memegang peranan yang sangat penting
3. Perkalian dengan permutasi juga membuat kesalahan satu elemen kunci saja menjadi sangat fatal, karena akan mempengaruhi encipher selanjutnya.
4. Penggunaan kunci berupa sebuah bilangan membuat jumlah kemungkinan kunci yang harus dicoba jika ingin melakukan *exhaustive key search* menjadi sangat banyak. Jika diketahui panjang kunci adalah n, maka jumlah kemungkinan kunci adalah $100^{n/2}$.
5. Variasi cipher substitusi homofonik membuat jejak frekuensi dan hubungan antara plainteks dan cipherteks menjadi kabur sehingga mempersulit kriptanalis melakukan kriptanalisis.

4.2 Known-plainteks Attack

Algoritma ini rawan terhadap serangan dari *known-plaintext attack* tidak seperti terhadap *ciphertext-only attack*, karena dengan sedikit metode coba-coba terhadap kemungkinan karakter dummy, kriptanalis bisa memperoleh persamaan yang bisa membantunya melakukan kriptanalisis. Misalkan seperti contoh kasus di bab sebelumnya:

P = 21 AF 01 AB 3F F1

C = 1F 97 1B 97 B2 37 69 E3 6A

Terlihat bahwa tidak ada petunjuk tentang karakter mana yang merupakan dummy dan karakter mana yang bukan. Akan tetapi dalam analisisnya, kriptanalis bisa menggunakan teknik spekulasi terhadap cipherteks dengan melakukan perkiraan karakter mana yang merupakan karakter dummy.

Misalkan kriptanalis menganggap karakter pertama merupakan dummy sehingga dia mempunyai persamaan

$31 = (33 - K_1 * K_n \text{ div } 256) \bmod 256$, dan

$151 = (33 + K_1 * K_n \bmod 256) \bmod 256$

Dengan keterbatasan range perkalian dua digit bilangan yang hanya berkisar antara 0-9801, kriptanalis bisa melakukan *exhaustive key search* hanya sebanyak 9801 kemungkinan. Jika tidak ditemukan, kriptanalis melanjutkan metodenya ke karakter selanjutnya.

4.3 Chosen-plaintext Attack

Seperti halnya terhadap metode *known-plaintext attack*, algoritma ini juga rawan terhadap serangan dengan metode *chosen-plaintext attack.* Bahkan lebih mudah lagi bagi kriptanalis karena dia bisa memilih pasangan plainteks-cipherteks yang meburutnya lebih mudah untuk dikriptanalisis.

4.4 Exhaustive Attack

Seperti dijelaskan di bab sebelumnya, penggunaan kunci yang panjang bisa mempersulit pencarian dengan *exhaustive key search* karena semakin panjang kunci semakin lama waktu yang dibutuhkan untuk melakukan *exhaustive key search.* Jumlah kemungkinan kunci adalah $100^{n/2}$.

5 Kesimpulan dan Saran

Berdasarkan percobaan dan analisis yang sudah dituangkan di atas, penulis bisa menarik beberapa kesimpulan terkait algoritma vigènere cipher homofonik.

- Algoritma vigènere cipher homofonik lebih baik dari vigènere cipher yang biasa dan variannya karena menggunakan perkalian permutasi dan cipher substitusi homofonik untuk menghilangkan jejak periodisasi dan frekuensi serta hubungan antara plainteks dengan cipherteks.
- Kompleksitas algoritma ini adalah n^2 karena menggunakan permutasi.
- Jumlah kemungkinan kunci algoritma ini adalah $100^{n/2}$
- Algoritma vigènere cipher homofonik kuat terhadap serangan cipherteks-only attack dan metode analisis frekuensi

- Algoritma vigènere cipher homofonik lemah terhadap serangan bertipe *known-plaintext attack* dan *chosen-plaintext attack*.
- Saran bagi pengguna yang ingin menggunakan algoritma ini adalah usahakan untul memakai kunci yang panjang.
- Saran bagi yang ingin mengembangkan algoritma ini, kompleksitas permutasi dan jenis kunci dipersulit lagi.

DAFTAR PUSTAKA

[1] Tanenbaum, Andrew S., *Modern Operating System*, Prentice Hall, 2001

[2] Bishop,David, *Introduction to Cryptography with Java Applet*, Jones and Bartlet Computer Science, 2003

[3] Munir, Rinaldi, *Diktat Kuliah IF5054 Kriptografi*, Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung, 2006

[4] http://www.cacr.math.uwaterloo.ca/hac/