# Generator Key Vigenere Cipher dengan Menggunakan Randomisasi dari Key tertentu

Abdurrosyid Broto Handoyo and 13510107[1]
*Program Studi Teknik Informatika*
*Sekolah Teknik Elektro dan Informatika*
*Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia*
*[1]13510107@stei.itb.ac.id*

***Abstrak*—Kriptografi klasik seperti vigenere cipher merupakan salah satu algoritma kriptografi yang implementasinya sangat sederhana namun cukup kuat di jamannya. Setelah ditemukannya motede kasiski, maka vigenere cipher sangat mudah untuk dianalisis oleh kriptanalis sehingga mendapatkan kunci kriptografinya. Hal ini disebabkan karena masih ada keterhubungan yang erat antara cipher, plainteks dan kunci kriptografi. Algoritma randomisasi pseudorandom ternyata dapat digunakan untuk membuat keterhubungan antara cpher, plainteks dan kunci kriptografi menjadi sangat semu sehingga sulit untuk dipecahkan oleh kriptanalis secara mangkus.**

***Kata kunci*—vigenere cipher, kunci kriptografi, pseudorandom, kriptanalis**

## I. PENDAHULUAN

Kriptografi merupakan sebuah ilmu yang mempelajari penjagaan keamanan dari sebuah data atau pesan yang kita miliki. Ilmu kriptografi sudah berkembang sejak masa Julius Caesar untuk mengamankan pesan dalam sebuah peperangan. Banyak sekali algoritma kriptografi yang telah dikembangkan baik secara klasik maupun secara modern.

Salah satu algoritma kriptografi klasik yang terkenal adalah Vigenere Cipher. Vigenere Cipher merupakan bentuk polyalphabetic substitution seperti ide yang diungkapkan Caesar Cipher namun dengan menambahkan kunci sehingga cipher yang dibentuk lebih aman. Ide ini sudah diterapkan berkali-kali, namun pada tahun 1553 metode ini baru dikemukakan oleh Giovan Battista Bellaso di bukunya *La cifra del. Sig. Giovan Battista Bellaso.* Algoritma ini akhirnya terkenal dengan nama Vigenere Cipher karena pada tahun 1586 Blaise de Vigenere mempublikasikan ide yang sama namun dengan generator kunci yang lebih kuat, yaitu autokey cipher.

Vigenere Cipher sangat terkenal karena mudah untuk dimplementasikan. Metode ini juga cukup kuat untuk menghindari kriptanalis yang menggunakan analisis frekuensi.Pada masa kejayaannya, sistem ini disebut sebagai le chiffre indechifferable. Hingga akhirnya seseorang bernama Frederich Kasiski dapat membuat metode yang efisien dalam memecahkan cipher vigenere.

Metode kasiski ini dipublikasikan pada tahun 1863. Metode ini mengharuskan seorang kriptanalis untuk mendeduksi panjang kunci yang digunakan. Dengan trik ini, maka cipher yang rumit ini dapat dipecahkan dalam waktu yang cukup *reasonable* pada saat itu.

Algoritma kriptografi klasik seperti Vigenere Cipher sudah ditinggalkan oleh orang-orang karena cukup mudah dianalisis oleh kriptanalis. Apalagi dengan teknologi yang sudah ada saat ini dan ditemukannya metode kasiski, maka bila ingin menggunakan Vigenere Cipher ini dengan aman, kita perlu melakukan modifikasi dari cara sistem ini bekerja. Cara yang diusulkan kali ini adalah meng-generate sebuah urutan kunci secara random dari kunci yang telah disepakati bersama.

## II. DASAR TEORI

### 2.1. Keamanan Pesan

Dalam proses pengamanan pesan, terdapat beberapa komponen penting yang menjadi inti dari keamanan sebuah pesan, yaitu:

- Data Confidentiality

Aspek kerahasiaan pesan yang dimiliki. Dalam kriptografi, hal ini menjadi fokus utamanya. Kriptografi mencoba untuk menjaga dan melindungi kerahasiaan pesan sehingga pesan tersebut tidak dapat diketahui oleh semua orang. Selain itu, dalam aspek ini juga harus memperhitungkan kemudahan bagi orang yang memang diperbolehkan membuka pesan tersebut.

- Authentication

Aspek ini berkaitan dengan autentifikasi siapa pengirim pesan tersebut. Salah satu permasalahan keamanan adalah bisa saja kita mendapatkan pesan dari orang ketiga yang seolah-olah menjadi orang yang ingin mengirim pesan kepada kita. Pesan palsu ini sangat berbahaya karena dapat merusak komunikasi yang terjalin antara dua pihak yang ingin berkomunikasi. Hal ini telah menjadi salah satu *concern* utama dalam keamanan informasi.

- Data Integrity

Aspek ini berkaitan mengenai bagaimana keabsahan pesan yang dikirim. Bisa saja ketika pesan tersebut

dikirim, di tengah jalan ada orang yang mengintercept kemudian mengubah isi pesan tersebut sehingga pesan gagal disampaikan dengan baik. Oleh karena itu, kriptografi harus memberikan layanan penjamin keaslian pesan yang dikirimkan dengan berbagai cara, antara lain bila teks cipher telah diubah, maka pesan sudah tidak dapat lagi dibuka. Hal ini dapat mengindikasikan terdapat interceptor yang mengubah isi pesan tersebut.

- Nonrepudiation

Aspek ini adalah bagaimana mencegah entitas yang berkomunikasi menyangkal bahwa pesan tersebut tidak dikirim oleh mereka. Aspek ini membahas juga bagaimana kita dapat membuktikan pesan tersebut benar-benar dikirim oleh orang yang mengklaim tersebut. Bila hal ini tidak diperhatikan,maka bisa saja dalam sebuah transasksi pembelian, orang yang membeli mengklaim bahwa transaksi itu bukan darinya meskipun sebenarnya dia yang melakukan pembelian.

2.2. Vigenere Cipher

Pada Caesar Cipher sebagai pencetus ide alphabetic substitution, setiap huruf teks pada plainteks akan diubah menjadi alphabet lain dengan sebuah pergeseran tertentu yang konstan. Sedangkan pada Vigenere Cipher, pergeseran huruf yang terjadi dilakukan dengan nilai geser yang berbeda. Berikut ini adalah tabel Vigenere Cipher:

| | 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 Vigenere Cipher*

Untuk melakukan enkripsi, kita perlu menentukan terlebih dahulu kunci yang akan digunakan. Panjang kunci ini dapat bervariasi tergantung dari kesepakatan anatara kedua belah pihak yang saling berkomunikasi. Setelah menentukan kunci yang digunakan, maka substitusi dilakukan menggunakan Tabel Vigenere Cipher (Gambar 1). Pada setiap step enkripsi dari alphabet, cipher yang dihasilkan menggunakan indeks alphabet yang berbeda-beda sesuai kunci. Bila kunci < panjang plainteks, maka kunci tersebut akan digunakan berulang-ulang.

Proses dekripsi dilakukan merujuk pada baris kunci ke-i, kemudian cari pada kolom apa cipher ke-i tersebut muncul. Indeks kolom tersebut merupakan plainteks ke-i.

Sebagai contoh proses enkripsi-dekripsi, misalkan plainteks yang akan dienkripsi adalah `MAKALAH KRIPTOGRAFI` dengan kunci yang telah ditetapkan bersama, yaitu `VIGENERE`.

Kunci dibuat berulang-ulang sesuai dengan plainteks terlebih dahulu seperti berikut:

Plainteks : `MAKALAH KRIPTOGRAFI`

Key : `VIGENER EVIGENEREVI`

Dari susunan tersebut, maka kita akan melakukan operasi penjumlahan sebagai berikut

$$C[i] = (P[i] + K[i])\ mod\ 26$$
$$\text{dengan } i \leq length(P)$$

$C$: cipher teks
$P$: plainteks
$K$: kunci

Sehingga didapatkan cipher seperti berikut:
Cipher : `HIQEYEY OMQVXBKIEAQ`

Sedangkan proses dekripsi dilakukan dengan melakukan operasi pengurangan sebagai berikut

$$P[i] = (C[i] - K[i])\ mod\ 26$$
$$\text{dengan } i \leq length(P)$$

2.3. Metode Kasiski

Kekuatan dari algoritma kriptografi Vigenere adalah tidak mudah dilakukan analisis frekuensi pada cipher karena keterkaitan antara plainteks dan cipher sudah cukup dikurangi dengan penggunaan kunci. Kasiski menemukan sebuah metode analisis dengan mendeduksi panjang dari kunci yang tidak diketahui.

Bila panjang kunci telah diketahui dengan pasti misalkan n, maka cipher dibariskan secara terurut menjadi n baris, kemudian masing-masing baris kita lakukan analisis frekuensi karena masing-masing baris tersebut adalah monoalphabetic substitution seeprti Caesar Cipher.

Namun, permasalahannya di sini adalah bagaimana kita dapat dengan cukup akurat mendeduksi panjang dari kunci yang belum kita ketahui? Metode kasiski memanfaatkan analisis frekuensi namun tidak dengan analisis frekuensi alphabet tunggal. Tetapi menggunakan analisis frekuensi beberapa karakter berurutan yang sering ditemui pada bahasa plainteks tersebut dibuat. Bila kita memiliki urutan karakter yang sama berulang-ulang maka jarak-jarak tersebut harus kita rekam. Kemudian dari semua jarak-jarak karakter beurutan yang sering muncul tersebut, kita mencari factor persekutuan terbesar dari

selisih-selisih tersebut.

Angka inilah yang dapat dijadikan dasar yang kuat dalam mendeduksi panjang kunci yang digunakan. Sehingga proses selanjutnya menjadi lebih mudah karena seperti menyelesaikan Caesar Cipher yang sudah terbukti cukup mudah.

Setelah menggunakan Caesar Cipher dengan dalam proses tersebut, maka kunci tersebut harus disusun ulang agar menjadi kunci yang utuh, namun terkadang kunci yang dihasilkan dari analisis frekuensi tidak akurat sehingga kita harus mencoba-coba dari kunci yang dihasilkan pada cipher teks, apakah benar-benar menjadi plainteks bermakna. Bila masih belum ditemukan plain teks yang benar-benar bermakna, maka kita harus melakukan perubahan pada kunci sesuai dengan kondisi plainteks yang dihasilkan saat ini dengan kemungkinan plainteks yang seharusnya. Dalam hal ini, penguasaan kata-kata dalam bahasa yang digunakan pada plainteks harus cukup baik agar deduksinya benar.

Namun, mengapa cara dalam menentukan kunci tersebut dapat digunakan adalah bila sebuah substring yang diulang muncul di plainteks dan jarak antar karakter adalah kelipatan dari panjang kunci, maka plainteks yang berulang tersebut dienkripsi dengan urutan kunci yang sama.

Sebagai contoh, perhatikan contoh berikut

Plainteks : AKU ADALAH PELAKUNYA

Key : ABC ABCABC ABCABCABC

Cipher : ALW AECLBJ PFNALWNZC

Dari data tersebut diketahui jarak antara perulangan cipher merupakan 9 yang berupakan perkalian dari 3 dengan bilangan bulat. Namun, data cipher yang sedikit dapat membiaskan hasil deduksi kita dari data FPB. Data cipher yang lebih banyak dapat membuat tebakan kita pada panjang kunci menjadi lebih baik.

Metode kasiski ini dapat berjalan dengan baik karena masih ada keterkaitan antara plainteks dan kunci dalam hal perulangan kata pada plainteks. Hal ini merupakan kelemahan utama dari metode Vigenere karena masih belum menciptakan dimensi confusion dan diffusion.

### 2.4. Pseudorandom

Dalam kondisi kenyataan, membangkitkan sebuah bilangan truly random merupakan hal yang mustahil, karena agar komputer secara otomatis menghasilkan bilangan random, maka komputer harus memiliki fungsi yang dapat menghasilkan nilai random. Niliai random ini pun tidak semata-mata random begitu saja, namun merupakan sebuah fungsi dengan parameter tertentu yang dapat menghasilkan bilangan dengan harapan memiliki pattern kerandoman.

Fungsi random ini biasanya melibatkan banyak factor agar menghasilkan kerandoman yang lebih baik. Sebagai contoh, Linux menggunakan variabel seperti keystrokes, I/O, voltase Least Significant Bit untuk menghasilkan sebuah angka yang random.

Namun, pada kasus kali ini kita akan memakai fungsi pseudorandom yang diimpelmentasikan oleh sebuah bahasa pemrograman. Fungsi pseudorandom ini akan menghasilkan output yang sama bila diberikan seed yang sama. Seed yang dipakai pada proses enkripsi dan dekripsi akan dihasilkan dari fungsi yang dirancang oleh penulis pada makalah ini.

## III. RANDOMISASI KEY

Kelemahan utama dari algoritma Vigenere Cipher adalah masih ada keterkaitan antara plainteks, cipher dan kunci. Hal ini terbukti dengan masih berjalannya metode kasiski yang memanfaatkan celah ini dengan baik. Apalagi dengan teknologi saat ini, metode Vigenere Cipher dapat dianalisis dengan cepat menggunakan bantuan computer.

Selain itu, kita harus dapat memasukkan dua factor yang membuat kriptanalis semakin sulit dalam melakukan bruteforce, yaitu confusion dan diffusion. Confusion adalah keterhubungan antara cipher teks dan kunci menjadi sangat kompleks sehingga sulit sekali dicari ketehbungan secara umum. Sedangkan factor diffusion adalah membuat perubahan kunci sedikit, dapat membuat cipher yang dibuat menjadi sangat berbeda, hal ini tentu saja menyulitkan kriptanalis. Bila kedua factor ini dimiliki maka cipher yang dihasilkan akan sangat rumit untuk dipecahkan.

Dalam Vigenere Cipher ini, desain kunci harus dibuat sehingga kita menipiskan hubungan antara kunci dan plainteks sehingga sulit ditemukan metode yang mangkus dalam memecahkan cipher kriptografi yang dirancang.

Dengan menggunakan randomisasi kunci kita dapat mendapatkan rangkaian kunci yang cukup sulit untuk ditebak oleh kriptanalis, selain itu kriptanalis yang masih engira bahwa vigenere tersebut digunakan dengan kunci standar yang berulang akan merasa sangat kesulitan dalam memecahkan Vigenere Cipher tersebut.

Desain kunci random yang diusulkan oleh penulis adalah pertama-tama antara pengirim dan penerima mengkomunikasikan kunci (K) yang akan digunakan bersama sebagai kunci utama generator kunci. Selain itu, pengirim dan penerima harus memilih sembarang pasangan bilangan bulat A dan B. Pasangan bilangan inilah yang akan membantu merandomkan kunci yang dihasilkan.

Prinsip dasar enkripsi pun masih sama dengan vigenere biasa, hanya saja kita mengganti kunci enkripsi menjadi sebuah fungsi dari K, A dan B.

$$C[i] = (P[i] + Q[i])\ mod\ 26$$
dengan $i \leq length(P)$

$C$: cipher teks

$P$: plainteks
$Q$: hasil fungsi A, B dan K
$Q = f(A, B, K)$

Sedangkan proses dekripsi juga dilakukan dengan melakukan operasi pengurangan sebagai berikut

$$P[i] = (C[i] - Q[i]) \ mod \ 26$$
dengan $i \leq length(P)$

Dengan hasil tersebut, kita mulai dapat memisahkan keterhubungan antara plainteks dan key yang dihasilkan. Karena fungsi F inilah yang akan menggenerate kunci dengan algoritma pseudorandom yang diharapkan cukup sulit untuk dianalisis. Hal ini diharapkan agar sulit untuk membuat sebuah metode yang mangkus sehingga kriptanalis harus melakukan brute-force kepada K, A dan B agar mendapatkan plainteks yang seharusnya.

Permasalahan selanjutnya adalah bagaimana kita mengimplementasikan fungsi F tersebut agar didapatkan hasil yang maksimal. Pada makalah ini selain penulis mengusulkan metode fungsi yang membuat keterkaitan antara cipher, plainteks dan kunci menjadi sangat semu, namun penulis juga mengusulkan salah satu cara untuk mengimplementasikan fungsi F tersebut.

Fungsi F tersebut akan memanfaatkan fungsi randomisasi yang disediakan oleh library sebuah bahasa pemrograman seperti C++, C#, Java, dsb agar mendapatkan angka yang cukup random dengan implementasi pseudorandom yang dibuat masing-masing bahasa.

Pada proses awal pembangkitan kunci, kita mendefinisikan sebuah fungsi $g(x)$. Fungsi ini akan menghasilkan sebuah bilangan bulat yang akan menjadi masukan dari fungsi pseudorandom yang akan dipakai. Fungsi $g(x)$ yang akan dipakai adalah seperti berikut.

$$g(x) = \sum_{i=1}^{length(x)} x[i]$$

$x$ dalam fungsi tersebut merupakan sebuah string yang nantinya adalah kunci yang disepakati bersama. $x$ ini dalam bentuk string agar mudah diingat oleh kedua belah pihak yang saling berkomunikasi.

Hasil dari fungsi $g(x)$ ini adalah menjadi input dari fungsi pseudorandom dari implementasi bahasa pemrograman masing-masing. Misalkan sebuah fungsi random pada sebuah bahasa yang dibangkitkan dengan seed tertentu adalah $rand(y, i)$.

$$rand(y, i) = Random_i(y)$$

Dengan i adalah urutan diambilnya kunci tersebut dari fungsi pseudorandom. Misalkan i adalah 3, maka kita akan memanggil $random(y)$ sebanyak 3 kali untuk mendapatkan hasilnya. $y$ yang akan dipakai dalam proses ini nantinya adalah berasal dari $g(Key)$

$$y = g(Key) + A$$

Fungsi $Random$ di atas adalah implementasi pseudorandom yang dimiliki oleh sebuah library bahasa pemrograman tertentu yang mana akan selalu berbeda pada setiap bahasa. Hal ini juga dapat menyulitkan kriptanalis untuk mencari implementasi pada bahasa pemrograman apakah yang digunakan oleh kedua pihak yang sedang berkomunikasi.

Fungsi $rand(y, i)$ tersebut sudah dapat membuat urutan kunci yang akan digunakan dalam Vigenere Cipher dengan baik. Namun, dengan cara demikian cipher masih kurang rumit untuk serangan bruteforce. Karena hanya melibatkan dua variabel. Selain itu fungsi pseudorandom yang ada, masih akan memperlihatkan sebuah pola meski hal ini pun sudah cukup rumit.

Sehingga ditambahkan sebuah operasi baru yang akan selalu menggeser posisi *y* bila telah dilakukan ekstraksi bilangan random sebanyak *B* kali. Dengan kata lain, setiap *B* kali ekstraksi kunci Vigenere, variabel *y* sebagai parameter dari fungsi *rand*, akan berubah sesuai persamaan seperti berikut.

$$y_{n+1} = y_n + rand(y_n, B + 1)$$

Perubahan variabel *y* semakin membuat rumit cipher yang dihasilkan. Perubahan kecil pada suatu parameter, dapat membuat cipher yang dihasilkan sama sekali berbeda dengan bantuan fungsi pseudorandom.

Sekuens yang dihasilkan tersebut, akan langsung digunakan untuk proses enkripsi dan dekripsi Vigenere Cipher. Namun, pada key ini, hubungan antara cipher, plainteks dan kunci menjadi sangat semu.

## IV. REALISASI

Berikut ini adalah rancangan algoritma generator key

```
ALGORITMA GENERATOR KEY

DEKLARASI FUNGSI
  Function g(x:string):integer;
  Function rand(y:integer, i:integer)
      :integer;

DEKLARASI VARIABEL
  Key: string
  GeneratedKey: string
  A,B:integer
  i,k: integer

ALGORITMA
k = 0
y = g(Key) + A
For i = (1 to length(Key)) do
  If (k == B) then
    k = 0
    y = y + Rand(y, B + 1)
  GeneratedKey[i] = (Rand(y, k)) mod
```

```
26
  k = k + 1
```

Variabel Key, A dan B sudah diinisialisasi dari awal. Variabel GeneratedKey adalah variabel yang akan digunakan sebagai key pada proses pembuatan cipher seuai dengan prinsip Vigenere Cipher. Sementara realisasi dari fungsi *rand* adalah sebagai berikut.

```
DEKLARASI FUNGSI
rand(y:integer, i:integer)
     :integer;
VARIABEL LOKAL
  j, Ret:integer
ALGORITMA
  For j = (1 to i) do
    Ret = Random(y)
  EndFor
  → Ret
```

Library random di mana algoritma ini direalisasikan menjadi pilhan dari pengguna dan kedua belah pihak yang akan bertukar informasi.

## V. TESTING

Penulis mengimplementasikan algoritma tersebut pada bahasa C++. Dengan menggunakan library rand() yang disediakan oleh C++.

Berikut ini adalah beberapa hasil tes pasangan kunci dan hasil kalkulasi fungsi pseudorandom yang telah dirancang sebelumnya.

```
Kunci  : algoritmakriptografivigenere
A      : 10
B      : 5
Hasil  : kogwvlclniddvmwcmrdenxczsvip
```

Dari hasil tersebut dapat dilihat bahwa antara kunci yang disepakati dan kunci yang dihasilkan, tidak memiliki pola yang dapat dipecahkan dengan mudah.

Contoh berikut ini merupakan contoh kunci yang dihasilkan bila panjang key kurang dari panjang plainteks

```
PlainTeks :
saya mengerjakan makalah kriptografi
Kunci     : vigenere
A         : 11
B         : 6
Hasil generate kunci :
aqcc dhrwyfixlvn vbustef iyneiizimpj
Cipher yang dihasilkan:
sqac pleccwrxvva hbeseem spvtbwfzmur
```

Dari contoh kedua, dapat dilihat antara cipher, plainteks dan kunci tidak memiliki kesamaan yang umum. Hal ini akan menyulitkan kriptanalis dalam memecahkan cipher yang ada karena mereka harus menemukan tiga variabel yang digunakan dalam proses pengiriman, yaitu kunci utama, variabel A dan variabel B.

Bila kriptografi ini cukup aman dari serangan analisis yang selama ini mangkus, maka perlu dicoba pula ketahanan algoritma ini terhadap serangan bruteforce attack. Serang bruteforce attack atau disebut complete-search attack ini adalah serangan kepada sebuah sistem kriptografi dengan mencoba semua kemungkinan kunci yang dapat dilakukan.

Dalam serangan bruteforce ini, penyerang harus menentukan variabel kunci utama, A dan B. Untuk menebak semua kemungkinan kunci utama, maka penyerang harus mencoba $26^{\text{length(key)}}$ kemungkinan. Sedangkan untuk menebak A dan B, penyerang harus menebak masing-masing sebesar batas atas nilai tersebut. Dalam perhitungan ini saya membatasi A dan B menjadi 0 hingga $10^6$ karena terhitung masih mudah diingat untuk menghafalkan 6 digit.

Ternyata kompleksitas dari serangan bruteforce kali ini mencapai $O(10^{12}.26^{length(key)})$, yang mana kompleksitas ini eksponensial atau dapat dikatakan non-polynomial sehingga membutuhkan waktu yang sangat lama untuk memecahkannya.

## VI. KESIMPULAN

- Generator kunci Vigenere Cipher dengan memanfaatkan randomisasi cukup baik dalam memisahkan keterkatian antara cipher, plainteks dan kunci yang digunakan.
- Penggunaan fitur pseudorandom dari bahasa pemrograman dalam proses menghasilkan key cukup mangkus karena proses enkrip dan dekripsi dapat dilakukan dengan cepat
- Proses bruteforce attack secara naif terhadap struktur algoritma ini akan menghasilkan kompleksitas

  $$O(A.B.26^{length(key)})$$

  Bila implementasi menggunakan C++ dan bilangan A dan B adalah angka diantara 0 hingga $10^6$ (agar masih mudah diingat), maka kompleksitas menjadi sangat besar.

## VII. UCAPAN TERIMA KASIH

Makalah ini dibuat untuk memenuhi tugas pengganti Ujian Tengah Semester dari mata kuliah IF3058-Kriptografi. Penulis mengucapkan terima kasih kepada Bapak Rinaldi Munir sebagai dosen mata kuliah ini yang telah membimbing kami dalam menyelesaikan makalah

ini.

Semoga makalah yang dibuat oleh penulis ini dapat bermanfaat pada bidang keilmuan kriptografi dan berguna bagi yang membaca.

Ucapan terima kasih juga kami berikan kepada semua pihak yang tidak dapat disebutkan satu per satu dan telah membantu penulis dalam pembuatan makalah ini sehingga dapat diselesaikan dengan baik.

## DAFTAR PUSTAKA

[1] Rinaldi Munir. IF3058-Kriptografi. http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2012-2013/kripto12-13.htm

[2] Wade Trappe and Lawrence C. Washington, Introduction to Cryptography with Coding Theory. Second edition. Pearson Prentice Hall, 2006.

[3] Cestus. "Basic Cryptanalysis". Diambil dari halaman http://www.bbc.co.uk/dna/h2g2/alabaster/A613135 pada tanggal 25 Maret 2013 pukul 20.32

[4] http://www.cryptool-online.org/index.php?option=com_content&view=article&id=105&Itemid=128&lang=en

## 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

Abdurrosyid Broto Handoyo
13510107