# Modifikasi Vigenere Cipher dengan Mengkombinasikan Vigenere 26 dan 256 Karakter

Yudhistira - 13508105
*Program Studi Teknik Informatika*
*Sekolah Teknik Elektro dan Informatika*
*Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia*
*[1]author@itb.ac.id*

***Abstract*—Vigenere Cipher merupakan salah satu algoritma kriptografi klasik abjad tunggal (lebih dikenal dengan nama *Polyalphabetic Cipher*) yang sudah lama terlahir dan banyak digunakan untuk menyamarkan pesan dan untuk autorisasi.**

**Ada banyak pemrogram atau Software Engineer yang menggunakan Vigenere Cipher karena cipher ini relatif mudah untuk digunakan. Namun, algoritma ini sudah lama bisa dipecahkan dengan metode kasiski. Untuk mengantisipasi hal tersebut, dalam makalah ini, dikombinasikan algoritma Vigenere 26 karakter dan Vigenere 256 karakter menjadi sebuah algoritma Vigenere kombinasi baru yang lebih kuat. Algoritma ini tentu saja lebih sulit untuk dipecahkan dengan metode kasiski.**

***Index Terms*—Vigenere,simple,extended,combination,26 character, 256 character**

## I. PENDAHULUAN

Kriptografi telah mengalami perkembangan yang sangat pesat dari tahun ke tahun. Semakin lama, algoritma yang digunakan untuk perlindungan data semakin kompleks dan semakin sulit untuk dipecahkan. Hal ini juga didukung oleh kemajuan kecepatan prosesor, dan perkembangan cara berfikir manusia.

Berdasarkan perkembangannya, algoritma kriptografi terbagi ke dalam 2 bagian. Kedua bagian tersebut adalah algoritma kriptografi modern dan algoritma kriptografi klasik. Karakteristik algoritma kriptografi modern yaitu beroperasi dalam mode bit. Biasanya, algoritma kriptografi modern memiliki operasi operasi *xor*, bitwise, pengubahan ke heksadesimal, dan pengelompokan rangkaian bit menjadi blok-blok tertentu. Secara umum, algoritma ini terbagi menjadi 2, yaitu:

a. *Stream Cipher* (Cipher Aliran)
b. *Block Cipher* (Block Cipher)

*Stream Cipher* memiliki karakteristik beroperasi pada bit tunggal, dan enkripsi dilakukan secara bit per bit. Sedangkan *Block Cipher* memiliki karakteristik beroperasi pada blok bit, dan enkripsi dilakukan secara blok per blok. Besar tiap blok bisa didefinisikan sendiri oleh desainer algoritma.

Algoritma kriptografi klasik memiliki karakteristik berbasis karakter. Secara umum, algoritma kriptografi klasik terbagi atas 2 bentuk umum. Bentuk tersebut adalah *cipher* substitusi, dan *cipher* transposisi.

Disebut c*ipher* substitusi karena *cipher* tersebut mengganti (menyubstitusi) suatu huruf pada *plaintext* menjadi huruf lain pada *ciphertext*. Ada 4 jenis *cipher* substitusi yang dikenal dalam dunia kriptografi. *Cipher-cipher* tersebut adalah:

a. *Monoalphabetic Cipher* (Cipher Abjad Tunggal)
b. *Homophonic Substitution Cipher* (Cipher Substitusi Homofonik)
c. *Polyalphabetic Cipher* (Cipher Abjad Majemuk)
d. *Polygram Substitution Cipher* (Cipher Substitusi Poligram)

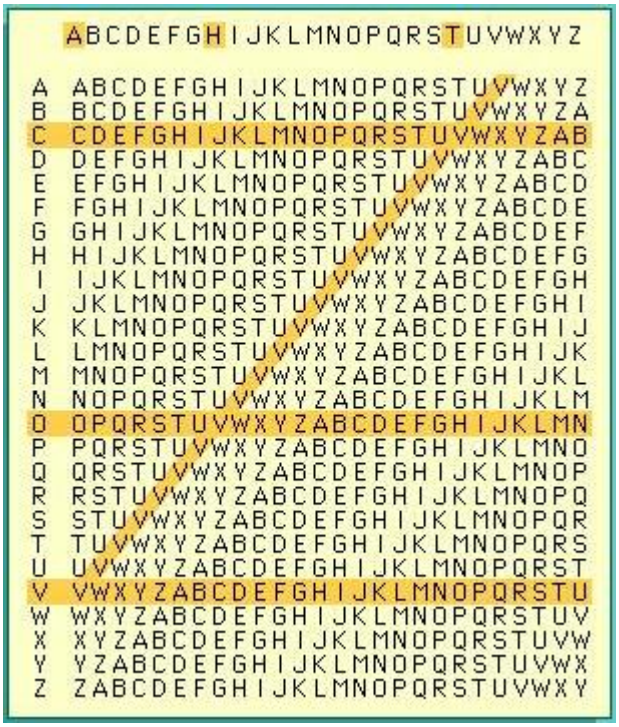
Pada Monoalphabetic *Cipher*, satu huruf pada *plaintext* diganti dengan satu huruf lain yang bersesuaian pada *ciphertext*. Contohnya, huruf a diganti dengan huruf n, huruf b diganti dengan huruf o, dan seterusnya. Pada *homophonic substitution cipher*, satu huruf pada *plaintext* diganti dengan 1 huruf lain yang mungkin pada *ciphertext.* Contohnya, huruf a diganti dengan pasangan huru hj atau ki, huruf b diganti dengan pasangan huruf zg atau fc, dan seterusnya.

Pada *Polyalphabetic Cipher*, setiap huruf pada *plaintext* dienkripsi dengan menggunakan kunci yang berbeda, dan setiap kunci diselesaikan dengan *monoalphabetic cipher*. Terakhir, pada *Polygram Substitution Cipher*, blok huruf pada *plaintext* diubah menjadi blok huruf lain pada *ciphertext.* Contohnya, blok huruf av diubah menjadi gj, blok huruf gj diubah menjadi zp, dan seterusnya.

Contoh dari *Polyalphabetic Cipher* yang terkenal adalah *Vigenere Cipher.* Tidak ada yang tau siapa penemu sebenarnya dari *Vigenere Cipher*. Algoritma ini dinamai *Vigenere Cipher* karena Blaise de Vigenere adalah orang paling terkenal yang menggunakan algoritma ini (1586). Namun faktanya, deskripsi algoritma sejenis *Vigenere Cipher* telah muncul pada tahun 1553 dalam buku *La cifra del. Sig. Giovan Battista Bellaso* karangan Giovan Battista Bellaso.

*Vigenere Cipher* menggunakan sebuah tabel tertentu untuk melakukan enkripsi dan dekripsi. Tabel tersebut diberi nama Bujursangkar Vigenere. Setiap baris di dalam

bujursangkar menyatakan huruf-huruf ciphertext yang dapat dibuat sendiri dengan memanfaatkan Caesar 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: Bujursangkar *Vigenere*

Jauhnya pergeseran nilai yang dilakukan oleh *Vigenere Cipher* bergantung pada kuncinya. Masing-masing kunci memiliki nilai tertentu yang masing-masing nilai untuk tiap hurufnya berurut dimulai dari a=0, sampai z=25. Contohnya, huruf b dienkripsi dengan kunci c. Nilai dari c adalah 2. Karena itu, huruf b digeser sebanyak 2 kali ke kanan. Maka pada *ciphertext*, huruf b berubah menjadi huruf d.

Pada *Vigenere Cipher*, kunci digunakan secara berulang-ulang sampai panjang ciphertext habis. Contohnya, apabila kita memiliki *plaintext* "Yudhistira", sedangkan kuncinya adalah "tujuh", maka kunci tersebut akan diulang sampai memenuhi panjang dari *plaintext*. Dalam kasus ini, kuncinya memanjang menjadi "tujuhtujuh".

Untuk memecahkan *cipher* ini, caranya terbilang sangat mudah. Kriptanalis terkenal yang memecahkan *Vigenere Cipher* adalah seorang petinggi militer Jerman bernama Friedrich Kasiski. Karena itu, metode untuk memecahkan *Vigenere Cipher* dikenal dengan metode Kasiski.

Cara melakukan metode Kasiski ini ini sangat mudah sekali. Intinya adalah melakukan analisis frekuensi trigram dan jarak antar trigram, lalu menebak panjang kunci, dan melakukan analisis frekuensi terhadap potongan *ciphertext* yang sudah dipotong-potong berdasarkan panjang kunci. Dengan metode ini, *Vigenere Cipher* standar akan dengan mudah dapat dipecahkan oleh banyak orang.

Oleh karena itu, diperlukan sebuah kombinasi tertentu untuk membuat *Vigenere Cipher* menjadi lebih kuat. Dalampaper ini, dikembangkan sebuah algoritma baru yang lebih kuat yang mengombinasikan *Vigenere Standar 26 huruf* dengan *Vigenere Extended 256 karakter*.

## II. Perbandingan Antara Vigenere Cipher Standar, Vigenere Cipher Extended, dan Vigenere Cipher Kombinasi

### 2.1 *Vigenere Cipher Standar*

Untuk melakukan enkripsi dan dekripsi menggunakan *Vigenere Cipher* standar, yang dibutuhkan hanyalah *plaintext* dan kunci. Ada 2 metode yang bisa dilakukan untuk mengenkripsi dan dekripsi menggunakan *Vigenere Cipher* Standar. Metode pertama adalah menggunakan Bujursangkar Vigenere seperti sudah dijelaskan di bab sebelumnya. Metode kedua adalah dengan memanfaatkan operasi matematika sederhana. Tiap-tiap *plaintext*

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| a | b | c | d | e | f | g | h | i | j | k | l | m |

| 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n | o | p | q | r | s | t | u | v | w | x | y | z |

memiliki nilai tersendiri, terurut dari a=0 sampai z=25. Dalam versi ini, seluruh tanda baca dan spasi diabaikan (dianggap tidak ada).

Gambar 2: Distribusi Nilai Karakter

Nilai *plaintext* tersebut kemudian dijumlahkan dengan nilai kunci pada karakter tersebut, lalu hasilnya dimodulo dengan 26 (jumlah karakter alfabet). Misalkan $p_i$ adalah *plaintext* pada karakter ke-i, $c_i$ adalah *ciphertext* pada karakter ke-i, dan $k_i$ adalah kunci maka bentuk umum dari vigenere cipher adalah:

$c_i = (p_i + k_i) \ mod \ 26$
$p_i = (c_i - k_i) \ mod \ 26$
(1)

Contohnya, misalkan String "kriptografi" ingin

| Plaintext | k | r | i | p | t | o | g | r | a | f | i |
|---|---|---|---|---|---|---|---|---|---|---|---|
| + | + | | | | | | | | | | |
| Kunci | a | w | a | n | a | w | a | n | a | w | a |
| mod 26 | mod 26 | | | | | | | | | | |
| Ciphertext | k | n | i | c | t | k | g | e | a | b | i |

dienkripsi dengan menggunakan kunci "awan". Maka operasi yang terjadi adalah sebagai berikut:

Gambar 3: Operasi pada *Vigenere Cipher* Standar

### 2.2 *Vigenere Cipher Extended*

*Vigenere Cipher Extended* dibuat untuk memperkuar algoritma *Vigenere Cipher* Standar. Dalam *cipher* ini, tabel yang digunakan bukan tabel distribusi nilai karakter seperti pada *Vigenere Cipher* Standar, melainkan tabel ASCII Extended 256 karakter (www.asciitable.com). Dalam algoritma ini, spasi dan tanda baca tidak diabaikan, karana spasi dan tanda baca terdapat dalam tabel ASCII Extended.

Untuk melakukan enkripsi dan dekripsi dengan *cipher* ini, formula yang digunakan sama dengan yang digunakan pada *Vigenere Cipher* Standar. Yang menjadi pembeda adalah nilai modulo yang digunakan. Karena pada *cipher* ini digunakan 256 karakter dalam proses enkripsinya, maka formula untuk melakukan enkripsi dan dekripsinya adalah:

$$c_i = (p_i + k_i) \ mod \ 256$$
$$p_i = (c_i - k_i) \ mod \ 256 \qquad (2)$$

Terlihat bahwa operasi yang digunakan pada *Vigenere Cipher Extended* sama dengan operasi yang digunakan pada *Vigenere Cipher* Standar. Namun, pada versi *Extended*, lebih sulit melakukan analisis frekuensi karena karakternya pun akan lebih bervariasi. Namun, tetap saja *cipher* ini dapat dipecahkan dengan mudah dengan menggunakan bantuan computer yang semakin lama semakin cepat dalam melakukan komputasi. Untuk meningkatkan lagi aspek kompleksitas pada *Vigenere Cipher*, dibuatlah algoritma *Vigenere Cipher* Kombinasi.

2.3 *Vigenere Cipher* Kombinasi

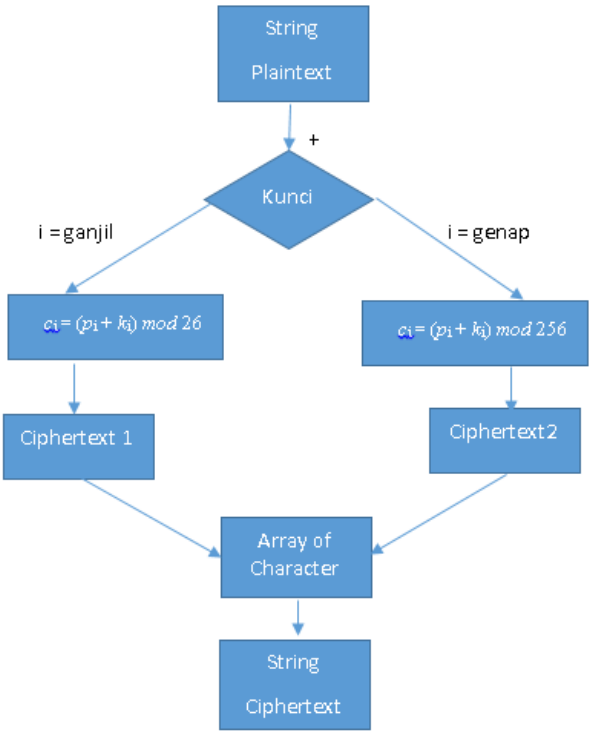
*Vigenere Cipher* Kombinasi merupakan gabungan dari *Vigenere Cipher* Standard dan *Vigenere Cipher Extended*. Ide dari algoritma ini adalah menggabungkan kedua algoritma tersebut sehingga analisis frekuensi menjadi tidak mungkin untuk dilakukan.

Untuk melakukan enkripsi dengan *cipher* ini, digunakan formula sebagai berikut:

Kalau i bilangan ganjil:
$$c_i = (p_i + k_i) \ mod \ 26$$
$$p_i = (c_i - k_i) \ mod \ 26$$

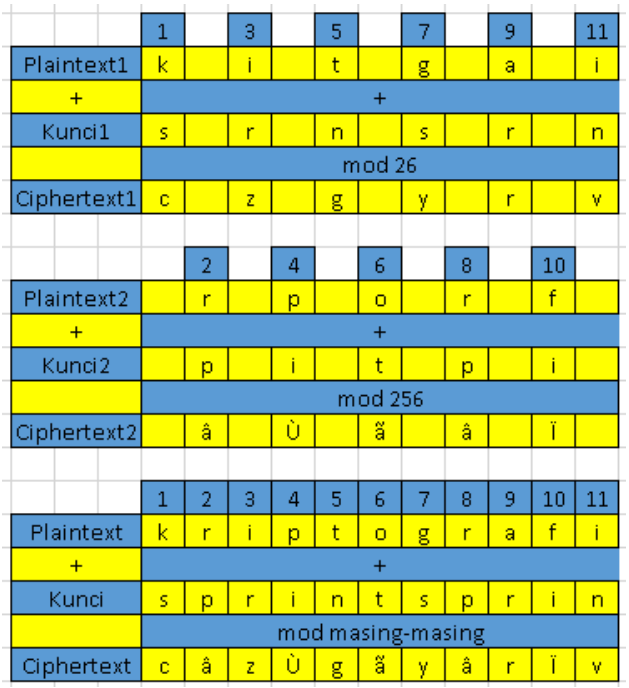
Kalau i bilangan genap:
$$c_i = (p_i + k_i) \ mod \ 256$$
$$p_i = (c_i - k_i) \ mod \ 256 \qquad (3)$$

Misalkan String yang akan dienkripsi adalah "kriptografi" dan kunci yang akan digunakan untuk mengenkripsi adalah "sprint". String "kriptografi" akan diparse. Kemudian, dilakukan operasi enkripsi terhadap karakter berindeks ganjil pada String tersebut. Hasil operasi tersebut kemudian disimpan ke string ciphertext1. Setelah itu, dilakukan operasi enkripsi terhadap karakter berindeks genap. Hasil operasi tersebut kemudian disimpan ke string ciphertext2. Setelah itu, kedua potongan ciphertext tersebut disatukan lagi. Alur algoritmanya adalah sebagai berikut:




Gambar 4: Alur Algoritma *Vigenere Cipher* Kombinasi

Contoh kasus, misalkan sebuah String "kriptografi" ingin dienkripsi dengan menggunakan kunci "sprint", maka proses enkripsinya adalah sebagai berikut:




Gambar 5: Operasi pada *Vigenere Cipher* Kombinasi

Karena merupakan gabungan dari 2 buah *cipher*, maka *source code cipher* ini jauh lebih panjang dibandingkan dengan *cipher* sebelumnya. Berikut adalah algoritma enkripsi yang diimplementasikan dengan menggunakan bahasa pemrograman Java:

```
        String plaintext = "";
        String ciphertext = "";
        String ciphertext1 = "";
        String ciphertext2 = "";
        String kunci = "";
        String kuncidummy = "";
        int p_kunci,p_kunciawal;
        int p_plaintext, p_ciphertext;
        int i,j,k,l,m,dummy;
        int spasi = 0;
        int[] arraykunci;

        kunci = jTextField1.getText();
        p_kunci = kunci.length();
        arraykunci = new int[p_kunci];

        l = 0; m = 0;
        kuncidummy = "";
        for (i=0;i<p_kunci;i++){

if((kunci.charAt(i)>=97)&&(kunci.charAt(i)<=122)){
                    arraykunci[l]=kunci.charAt(i)-97;
                    l++;
                    kuncidummy+=kunci.charAt(i);
                }
                else
if((kunci.charAt(i)>=65)&&(kunci.charAt(i)<=90)){
                    arraykunci[l]=kunci.charAt(i)-65;
                    l++;
                    kuncidummy+=kunci.charAt(i);
                }
                else{
                    m++;
                }
        }
p_kunciawal=p_kunci;
```

*Source code* di atas merupakan bagian deklarasi variabel dan pengolahan kunci. Setelah ini, algoritma masuk ke bagian enkripsi.

```
    if (pilihan==1){ //enkripsi
            plaintext = jTextArea1.getText();
            p_plaintext = plaintext.length();
            p_kunci-=m;
            for (i=0;i<p_plaintext;i=i+2){
                j = (i-spasi) % p_kunci;

if((plaintext.charAt(i)>=97)&&(plaintext.charAt(i)<=122)){
                    dummy = (arraykunci[j]+plaintext.charAt(i)-
97)%26;
                    dummy += 97;
                    ciphertext1 += (char)dummy;
                }
                else
if((plaintext.charAt(i)>=65)&&(plaintext.charAt(i)<=90)){
                    dummy = (arraykunci[j]+plaintext.charAt(i)-
65)%26;
                    dummy += 65;
                    ciphertext1 += (char)dummy;
                }
                else if(plaintext.charAt(i)==32){
                    spasi++;
                    ciphertext1 += plaintext.charAt(i);
                }
                else{
                    ciphertext1 += plaintext.charAt(i);
                }
            }
            p_kunci=p_kunciawal;

            for (i=0;i<p_kunci;i++){
                arraykunci[i]=kunci.charAt(i);
            }
            for (i=1;i<p_plaintext;i=i+2){
                j = (i-spasi) % p_kunci;
                if((plaintext.charAt(i)==32)){
                    spasi++;
                    ciphertext2 += plaintext.charAt(i);
                }
                else{
                    dummy                                    =
(arraykunci[j]+plaintext.charAt(i))%256;
                    ciphertext2 += (char)dummy;
                }
            }

            if((p_plaintext%2)==0)
            {
                char cipher[] = new char[p_plaintext];
                for(i=0;i<p_plaintext;i++)
                {
                    cipher[i]=ciphertext1.charAt(i/2);
                    i++;
                    cipher[i]=ciphertext2.charAt(i/2);
                }
                String ciphertext = new String(cipher);
                jTextArea2.setText(ciphertext);
            }
            else
            {
                char cipher[] = new char[p_plaintext];
                for(i=0;i<p_plaintext;i=i+2)
                {
                    cipher[i]=ciphertext1.charAt(i/2);
                }
                for(i=1;i<p_plaintext;i=i+2)
                {
                    cipher[i]=ciphertext2.charAt(i/2);
                }
                String ciphertext = new String(cipher);
                jTextArea2.setText(ciphertext);
            }
    }
```

Pada potongan *source code* yang dicetak merah, terdapat *Vigenere Cipher* Standar yang mengubah plaintext pada index ganjil menjadi potongan ciphertext yang disimpan pada String ciphertext1. Pada potongan *source code* yang dicetak biru, terdapat *Vigenere Cipher Extended* yang mengubah plaintext pada index genap menjadi potongan ciphertext yang disimpan pada String ciphertext2.

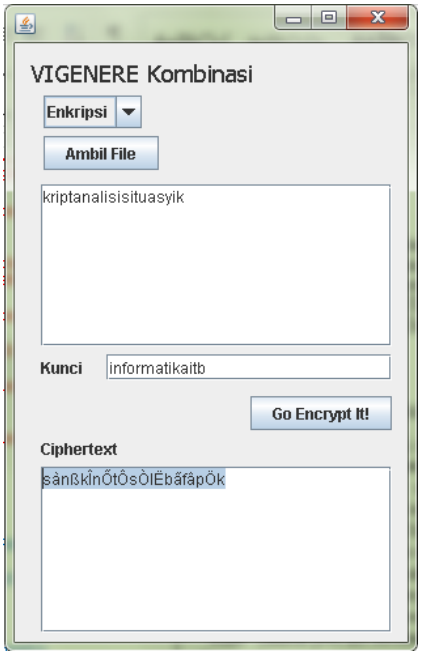
Kemudian, pada *source code* yang berwarna hijau, string ciphertext1 dan ciphertext2 digabungkan di variabel "cipher" yang bertipe array karakter dengan ukuran sepanjang plainteks. Array karakter tersebut kemudian diubah menjadi string, dan jadilah String ciphertext.

Proses dekripsi algoritma ini juga relatif sama dengan proses enkripsinya. Pertama adalah dekripsi sebagian dengan *Vigenere Cipher* Standar, lalu dekripsi sebagian lagi dengan *Vigenere Cipher Extended*. Terakhir, kedua hasil dekripsi disatukan menjadi. Berikut algoritmanya:

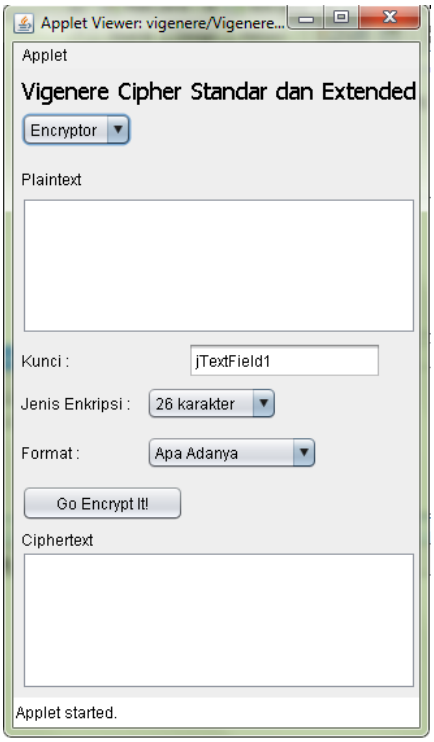
```
    else if (pilihan==2){ //dekripsi
        ctext = jTextArea1.getText();
        p_ciphertext = ctext.length();
        p_kunci-=m;
        for (i=0;i<p_ciphertext;i=i+2){
          j = (i-spasi) % p_kunci;
          if((ctext.charAt(i)>=97)&&(ctext.charAt(i)<=122)){
            dummy = (ctext.charAt(i)-97-arraykunci[j]+26)%26;
            dummy += 97;
            ciphertext1 += (char)dummy;
          }
          else
if((ctext.charAt(i)>=65)&&(ctext.charAt(i)<=90)){
            dummy = (ctext.charAt(i)-65-arraykunci[j]+26)%26;
            dummy += 65;
            ciphertext1 += (char)dummy;
          }
          else if(ctext.charAt(i)==32){
              spasi++;
              ciphertext1 += ctext.charAt(i);
          }
          else{
              ciphertext1 += ctext.charAt(i);
          }
        }

        p_kunci=p_kunciawal;
        for (i=0;i<p_kunci;i++){
            arraykunci[i]=kunci.charAt(i);
        }
        for (i=1;i<p_ciphertext;i=i+2){
            j = (i-spasi) % p_kunci;
            if((ctext.charAt(i)==32)){
                spasi++;
                ciphertext2 += ctext.charAt(i);
            }
            else{
                dummy                =                (ctext.charAt(i)-
arraykunci[j]+256)%256;
                ciphertext2 += (char)dummy;
            }
        }

        if((p_ciphertext%2)==0)
        {
            char cipher[] = new char[p_ciphertext];
            for(i=0;i<p_ciphertext;i++)
            {
                cipher[i]=ciphertext1.charAt(i/2);
                i++;
                cipher[i]=ciphertext2.charAt(i/2);
            }
            String plain = new String(cipher);
            jTextArea2.setText(plain);
        }
        else
        {
            char cipher[] = new char[p_ciphertext];
            for(i=0;i<p_ciphertext;i=i+2)
            {
                cipher[i]=ciphertext1.charAt(i/2);
            }
            for(i=1;i<p_ciphertext;i=i+2)
            {
                cipher[i]=ciphertext2.charAt(i/2);
            }
            String plain = new String(cipher);
            jTextArea2.setText(plain);
        }
    }
}
```

## III. PENGUJIAN DAN ANALISIS

Untuk membandingkan kekuatan antara *Vigenere Cipher* Standard dan *Vigenere Cipher* Kombinasi, maka diimplementasikan 2 macam program *Vigenere Cipher* sederhana dalam bahasa java. Diberikan beberapa testcase berikut:




Gambar 6: Program *Vigenere Cipher Kombinasi*




Gambar 7:
Program *Vigenere Cipher Standard an Extended*

Berikut ini berturut turut merupakan ciphertext *Vigenere Cipher* Standar, *Vigenere Cipher Extended* dan *Vigenere Cipher* Kombinasi.

| | | |
|---|---|---|
| Plaintext | : | kriptanalisisituasyik |
| Kunci | : | informatikaitb |
| Ciphertext Std | : | sendkmnttssqljbhfgpuk |
| Ciphertext Ext | : | ÔàÏßæÎÏÕÕÔÔÒçËÝãÇâëÖÌ |
| Ciphertext Kom | : | sànßkÎnÕtÔsÒlËbãfâpÖk |

Dari ketiga *versi ciphertext tersebut*, versi Std sangat mudah dipecahkan dengan melakukan analisis frekuensi. Semakin panjang jumlah teksnya, analisis frekuensi akan semakin akurat. Versi Ext juga bisa dipecahkan dengan teknik analisis frekuensi. Namun, dalam versi ini karakter unik dan spasi seluruhnya dilibatkan, sehingga perlu ketelitian lebih untuk memecahkan *cipher* ini.

Sedangkan pada versi Kom, analisis frekuensi sangat sulit sekali untuk dilakukan, karena ada 2 macam algoritma yang digunakan untuk mengenkripsi *plaintext*. Sebagai contoh:

| | |
|---|---|
| Plaintext | : dandandandandandandandandan |
| Kunci | : informatikaitb |
| Ciphertext | : lÏsÓrÛdÕvÏa×wÃvÒfÝuÎnØiÙ |

Pada *ciphertext* di atas, terlihat bahwa kata "dan" bisa memiliki lebih banyak variasi dibandingkan dengan versi Std dan Ext. Pada kasus di atas, kata "dan" bisa berubah menjadi lÏs, ÓrÛ, dÕv, Ïa×, wÃv, ÒfÝ, uÎn, dan ØiÙ. Tentu saja sangat sulit melakukan analisis frekuensi dan memperkirakan panjang kunci dengan kunci yang terlalu bervariasi.

Berdasarkan panjang kompleksitas kuncinya, versi Kom juga memiliki kompleksitas yang lebih tinggi. Untuk kasus *plaintext* "kriptanalisisituasyik", maka untuk mem-*bruteforce ciphertext*nya, harus mencoba kemungkinan kunci sebanyak $256^{21}$ kunci, sama dengan versi Ext, dan lebih baik dari versi Std yang hanya $26^{21}$ kunci.

## IV. KESIMPULAN

Berdasarkan kompleksitas algoritmanya, *Vigenere Cipher* Kombinasi merupakan algorima yang lebih kuat daripada *Vigenere Cipher* Standar maupun *Vigenere Cipher Extended*. Pada *Vigenere Cipher* Kombinasi, sangat sulit untuk melakukan teknik analisis frekuensi, dan lebih lagi untuk menebak panjang kunci seperti bisa dengan mudah dilakukan dalam versi-versi sebelumnya, terutama pada *Vigenere Cipher Standar*. Tapi, *Vigenere Cipher Kombinasi* berdasarkan kemungkinan kuncinya ternyata tidak lebih kuat daripada versi *Extended*, walaupun masih lebih kuat daripada versi Standar. Secara keseluruhan, *Vigenere Cipher* Kombinasi lebih kuat dari kedua *cipher* yang membentuk *cipher* ini.

## REFERENSI


[1] Sinaga,Y.A.,Pergeseran Kemiringan Pada Vigenere Cipher http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2007-2008/Makalah1/MakalahIF5054-2007-A-075.pdf

[2] http://www.cryptomuseum.com/crypto/vigenere/ (diakses pada tanggal 26 Maret 2013)

[3] http://global.britannica.com/EBchecked/topic/628637/Vigenere-cipher (diakses pada tanggal 27 Maret 2013)

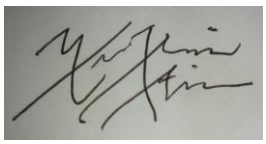
[4] www.asciitable.com (diakses pada tanggal 25 Maret 2013)

[5] Slide Bahan kuliah IF3058 Kriptografi – Semester II 2012-2013


## 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, 29 Maret 2013



Yudhistira
13508105