Studi dan Implementasi Algoritma Inverse Generator Cipher

Muhamad Fajrin Rasyid [1)]
1) Program Studi Teknik Informatika ITB, Bandung 40132, email: if14055@students.if.itb.ac.id

Abstract –*Vigenere Cipher merupakan cipher yang dapat dipecahkan dengan menggunakan beberapa teknik seperti pendugaan panjang kunci dan pemeriksaan pengulangan karakter-karakter. Oleh karena itu, penggunaan cipher tersebut dalam mengenkripsikan pesan tidak dianjurkan karena bersifat kurang aman. Apabila keamanan menjadi faktor utama, penggunaan unbreakable cipher seperti one-time pad merupakan hal yang disarankan. Namun, metode ini merupakan metode yang tidak praktis terutama untuk pesan berukuran besar. Untuk mengurangi permasalahan tersebut, makalah ini akan memperkenalkan dan membahas algoritma inverse generator cipher dan implementasinya. Algoritma ini merupakan pengembangan dari vigenere cipher dengan ketentuan bahwa untuk setiap pengulangan kunci, kunci semula tidak digunakan lagi, melainkan dibangkitkan kunci baru yang merupakan balikan modulo suatu bilangan dari elemen-elemen kunci semula. Bilangan yang dimaksud dibentuk dari dua hal, yaitu faktor kunci yang dipilih oleh pengguna (bilangan asli berapapun) dan kelipatan persekutuan terkecil elemen-elemen kunci. Bilangan ini merupakan bilangan yang relatif prima dengan seluruh elemen kunci sehingga balikan seluruh elemen kunci tersebut akan selalu terdefinisi. Balikan modulo digunakan dalam algoritma ini karena balikan modulo merupakan bilangan yang tidak dapat dirumuskan secara umum. Dengan cara ini, akan sangat sulit untuk mengetahui pola umum kunci apabila kunci tidak diketahui.*

Kata Kunci*: Vigenere Cipher, Balikan, Modulo, Faktor Kunci, Kelipatan Persekutuan Terkecil (KPK)*

1. PENDAHULUAN

Vigenere Cipher merupakan salah satu algoritma kriptografi klasik yang menerapkan *cipher*-abjad majemuk (*polyalphabetic substitution cipher*). Vigenere Cipher ditemukan pada abad 16 oeh seorang diplomat Perancis, Blaise de Vigenere, dan digunakan oleh tentara konfederasi pada Perang Sipil Amerika. Saat ini, penggunaan Vigenere Cipher tidak terlalu banyak. Hal ini disebabkan Vigenere Cipher ternyata dapat dipecahkan dengan teknik analisis frekuensi dan metode Kasiski untuk menentukan panjang kunci yang digunakan [1].

Teknik analisis frekuensi dapat digunakan untujk memecahkan Vigenere Cipher karena Vigenere Cipher menggunakan kunci yang sama berulang-ulang. Dengan demikian, besar kemungkinan tedapat istilah-istilah yang sama di dalam pesan yang dienkripsi menjadi istilah-istilah yang sama pula sehingga hubungan kemunculan antara istilah-istilah tersebut dapat ditelusuri. Oleh karena itu, apabila ingin mengembangkan Vigenere Cipher sehingga sulit dipecahkan, hal yang perlu ditekankan adalah bagaimana mengembangkan dan memodifikasi kunci yang mendasari algoritma sehingga tidak hanya sekadar digunakan secara berulang-ulang.

Algoritma inverse generator cipher menggunakan dasar teori bilangan bulat dalam memodifikasi kunci yang digunakan. Dalam algoritma ini, selain kunci berupa string, digunakan pula kunci berupa bilangan asli yang untuk selanjutnya disebut faktor kunci dan disimbolkan dengan *k*. Jika Vigenere Cipher menggunakan kunci yang sama berulang-ulang, maka inverse generator akan bekerja berdasarkan prinsip sebagai berikut. Setelah selesai menggunakan kunci, bangkitkan bilangan $g(n,k) = kn + 1$ (1), dengan *k* adalah faktor kunci yang disebutkan sebelumnya dan *n* adalah kelipatan persekutuan terkecil elemen-elemen kunci. Setelah itu, kunci berikutnya didapat dengan cara menentukan balikan setiap elemen kunci modulo $g(n,k)$. Karena balikan modulo tidak memiliki rumus umum, kunci berikutnya akan terlihat tidak berhubungan dengan kunci semula apabila kunci string dan faktor kunci tidak diketahui. Langkah ini dilakukan terus-menerus sehingga didapatkan barisan kunci yang sangat acak. Akan sangat sulit untuk merumuskan barisan kunci tersebut. Dengan demikian, proses kriptanalisis akan sulit untuk dilakukan.

Makalah ini akan membahas algoritma inverse generator cipher, termasuk di dalamnya cara kerja dan teori yang mendasarinya, aplikasinya, contoh kasus, tingkat keamanannya – yaitu pengujian akan seberapa sulit kunci ini dapat dipecahkan termasuk apabila salah satu elemen kunci (faktor kunci atau kunci string) diketahui, serta implementasinya. Dengan demikian, diharapkan algoritma inverse generator cipher dapat menjadi pertimbangan untuk digunakan sebagai salah satu alternatif cipher substitusi.

2. DASAR TEORI

2.1. Vigenere Cipher

Vigenere Cipher merupakan pengembangan dari Caesar Cipher yang melakukan pergeseran sejaun *n* (*n* lebih besar atau sama dengan 0 dan kurang dari atau sama dengan 25 sesuai dengan banyaknya huruf) terhadap setiap huruf [6]. Vigenere Cipher menggunakan bujursangkar Vigenere untuk

melakukan enkripsi. Dalam hal ini, setiap baris di dalam bujursangkar menyatakan huruf-huruf cipherteks yang diperoleh dengan Caesar Cipher dengan pergeseran ditentukan oleh kuruf kunci yang diasosiasikan oleh baris tersebut (misalnya $a = 0$, $b = 1$, dan seterusnya).

2.1.1. Contoh Penggunaan Vigenere Cipher

Contoh penggunaan Vigenere cipher adalah sebagai berikut. Akan dilakukan enkripsi terhadap pesan "tujuh puluh tujuh" dengan kunci "empat". Hasil enkripsi dapat dilihat pada tabel berikut:

Tabel 1 Contoh Aplikasi Vigenere Cipher

P	t	u	j	u	h	p	u	l
K	e	m	p	a	t	e	m	p
E	4	12	15	0	19	4	12	15
C	x	g	y	u	a	t	g	a

P	u	h	t	u	j	u	h
K	a	t	e	m	p	a	t
E	0	19	4	12	15	0	19
C	u	a	x	g	y	u	a

Keterangan:
P = Plainteks
K = Kunci
C = Cipherteks
E = pergeseran

Dengan demikian, cipherteks dari pesan plainteks (tanpa menggunakan spasi) "tujuhpuluhtujuh" adalah "xgyuatgauaxgyua".

2.1.2. Karakteristik Vigenere Cipher

Beberapa karakteristik Vigenere Cipher akan dijelaskan sebagai berikut. *Pertama*, setiap huruf pada cipherteks dapat memiliki banyak kemungkinan makna plainteks [5]. Sebagai contoh, pada kasus diatas, huruf 'a' yang merupakan huruf kelima pada cipherteks berarti 'h' pada plainteks, sedangkan huruf 'a' yang merupakan huruf kedelapan pada cipherteks berarti 'l' pada plainteks. Dengan demikian, teknik analisis frekuensi tunggal yang dapat digunakan untuk memecahkan Caesar Cipher tidak dapat digunakan untuk menyerang vigenere cipher.

Kedua, penggunaan kunci yang berulang-ulang dalam vigenere cipher dapat memunculkan string yang berulang-ulang [5]. Dalam contoh diatas, pada cipherteks kita menemukan string "xgyua" berulang. Hal ini memang tidak selalu terjadi karena bergantung pada panjang kunci yang digunakan. Apabila jarak antara dua istilah yang berulang dalam plainteks sama dengan panjang kunci atau kelipatannya, akan ditemukan string berulang. Jarak antara huruf pertama string "tujuh" pertama dan string "tujuh" kedua pada kasus diatas adalah 10 (sama dengan kelipatan dari panjang kunci yang digunakan), sedangkan kunci yang digunakan memiliki panjang 5.

2.1.3 Kelemahan Vigenere Cipher

Karakteristik kedua tersebut mendasari metode Kasiski, yaitu metode penentuan panjang kunci dengan memprediksi berdasarkan jarak tiap string yang berulang [1]. Dalam hal ini, panjang kunci merupakan faktor pembagi bersama jarak-jarak string yang berulang. Pada contoh di atas, tanpa mengetahui kunci yang digunakan, dapat diduga bahwa panjang kunci adalah 2, 5, atau 10. Adapun panjang kunci sebenarnya adalah 5. Apabila panjang kunci sudah diprediksi, cipherteks dapat dibagi sesuai dengan sisa pembagian modulo panjang kunci, dan kemudian teknik analisis frekuensi dapat diterapkan untuk setiap kelompok kunci untuk memprediksi kunci. Hal inilah yang menjadi kelemahan Vigenere Cipher, yaitu dapat dipecahkan berdasarkan karakteristik string-string yang berulang.

2.2. Teori Bilangan Bulat

2.2.1. Keterbagian

Suatu bilangan bulat a disebut membagi b jika terdapat bilangan bulat k sehingga $b = ka$ [3]. Sebagai contoh, 3 habis membagi 12 karena $12 = 4.3$. Akan tetapi, 5 tidak habis membagi 12 karena tidak ada bilangan bulat k sehingga $12 = 5k$.

Teorema berikut (**Teorema (1)**) menyatakan bahwa apabila a membagi b dan a membagi c, maka a membagi $bx + cy$ untuk berapapun bilangan bulat x dan y [3]. Bukti teorema tersebut adalah sebagai berikut. Apabila a membagi b dan a membagi c, maka $b = ak$ dan $c = al$. Dengan demikian, $bx + cy = akx + aly = a(kx + ly)$. Karena $kx + ly$ adalah bilangan bulat, berarti a membagi $bx + cy$.

Teorema (1) membawa akibat bahwa apabila $a > 1$ dan a membagi b, maka a tidak membagi $b + 1$. Sebagai bukti, jika a membagi b, maka $b = ka$ untuk suatu bilangan bulat k, dan $-b = (-k)a$, artinya a juga membagi $-b$. Andaikan a membagi $b + 1$, maka haruslah a membagi $(b + 1) + (-b) = 1$, suatu kemustahilan. Artinya a tidak membagi $b + 1$.

2.2.2. Teori Modulo

Bilangan bulat a dan b yang memenuhi $a = b \bmod m$ untuk suatu bilangan bulat m mengandung arti bahwa a dan b bersisa sama jika dibagi m. Istilah $a = b \bmod m$ dapat juga berarti $a = km + b$ [2]. Sebagai contoh, $7 = 12 \bmod 5$ karena 7 dan 12 sama-sama bersisa 2 jika dibagi dengan 5. Dengan demikian, apabila a membagi b, maka $b = 0 \bmod a$.

Teorema berikut (**Teorema (2)**) menyatakan bahwa jika $ac = bc \bmod m$ dan $\text{FPB}(c,m) = 1$, maka $a = b \bmod m$ [4]. Bukti teorema ini adalah sebagai berikut. $ac = bc \bmod m$ berarti $ac = bc + km$ atau $km = ac - bc = c(a - b)$. Karena $\text{FPB}(c,m) = 1$, maka c tidak habis dibagi oleh seluruh faktor m. Oleh karena itu, seluruh faktor m membagi $a - b$, atau m membagi $a - b$. Dengan kata lain, $a - b = pm$ untuk suatu bilangan bulat p, atau $a = pm + b$, atau $a = b \bmod m$.

Untuk suatu bilangan bulat a dan b, bilangan bulat x

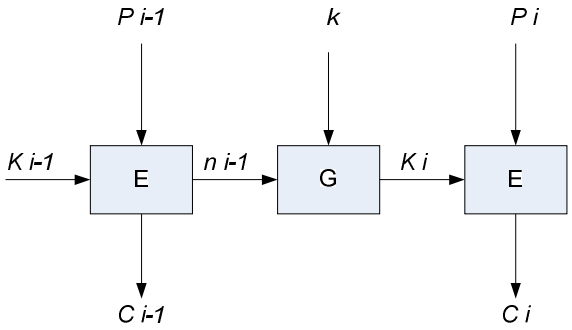
disebut sebagai balikan *a* modulo *b* jika $ax = 1 \bmod b$. Sebagai contoh, balikan dari 7 modulo 5 adalah 3, karena $7.3 = 21 = 1 \bmod 5$. Teorema berikut (**Teorema (3)**) menyatakan bahwa *a* dan *b* relatif prima (memiliki faktor persekutuan terbesar/FPB sama dengan 1) jika dan hanya jika *x* terdefinisi [4]. Bukti untuk teorema ini adalah sebagai berikut. Apabila FPB(*a*,m) = 1; jika $0 < r < m$, $ar = s \bmod m$ untuk suatu $0 < s < m$. Selain itu, setiap nilai *r* berbeda akan menghasilkan nilai *s* berbeda karena jika $r_1a = s \bmod m$ dan $r_2a = s \bmod m$ maka $r_1a = r_2a \bmod m$ atau $r_1 = r_2 \bmod m$ berdasarkan **Teorema (2)**. Oleh karena itu, $1a.2a.\ldots.(m-1)a = 1.2.\ldots.(m-1) \bmod m$ atau $a^{m-1} = 1 \bmod m$. Dengan demikian, untuk setiap bilangan bulat *a*, kita dapat memilih balikan a^{m-2}. Sebaliknya, apabila FPB(*a*,*m*) > 1; andaikan terdapat *x* sehingga $ax = 1 \bmod m$ berarti $ax = km + 1$ untuk suatu bilangan bulat *k*, atau $ax - km = 1$. Karena FPB(*a*,*m*) > 1, maka $a = nb$ dan $m = nc$ untuk suatu $n > 1$. Dengan demikian, $nbx - knc = 1$, atau $n(bx - kc) = 1$. Akan tetapi, ini berarti bahwa *n* membagi 1, suatu kemustahilan. Dengan demikian pengandaian tersebut salah dan tidak terdapat *x* yang dimaksud.

3. HASIL DAN PEMBAHASAN

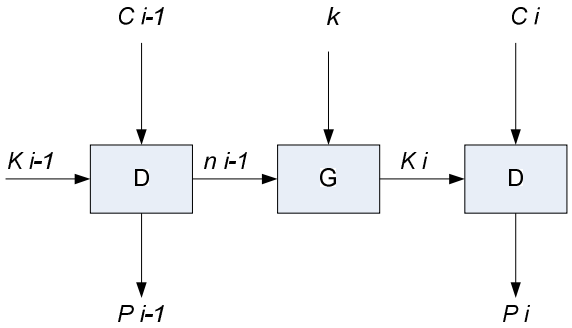
Algoritma inverse generator cipher merupakan algoritma yang penulis kembangkan dari vigenere cipher. Algoritma ini termasuk dalam cipher-abjad majemuk (*polyalphabetic substitution cipher*), yaitu cipher-abjad yang menggunakan kunci dengan panjang *m* suatu bilangan asli. Plainteks digeser berdasarkan kunci tersebut dengan mengacu pada bujursangkar Vigenere. Namun, tidak seperti vigenere cipher yang menggunakan kunci yang sama berulang-ulang (periodik) dengan periode sesuai dengan panjang kunci *m*, inverse-generator cipher membangkitkan kunci baru setelah menggunakan kunci semula dengan berdasarkan suatu nilai pembangkit dan suatu faktor kunci yang telah ditentukan. Proses pembangkitan kunci baru ini berulang hingga akhir pesan. Dengan demikian, pemecahan dengan metode Kasiski dan analisis frekuensi seperti pada vigenere cipher menjadi tidak dapat dilakukan karena tidak menggunakan kunci yang sama berulang-ulang. Hal ini menjadikan inverse generator cipher sukar untuk dipecahkan.

3.1. Cara Kerja Inverse Generator Cipher

Inverse generator cipher bekerja secara paralel yaitu ketika melakukan enkripsi terhadap plainteks pada periode ke-i, ketika itu pula dibangun kunci dibangkitkan bilangan yang diperlukan untuk membangun kunci ke-i+1. Setelah itu, proses enkripsi terhadap plainteks periode ke-i+1 siap dilakukan. Untuk lebih jelasnya, proses enkripsi pada inverse generator cipher dapat dilihat pada Gambar 1, sedangkan proses dekripsi pada Gambar 2.



Gambar 1 Proses Enkripsi Inverse Generator Cipher



Gambar 2 Proses Dekripsi Inverse Generator Cipher

Pada proses enkripsi, untuk setiap string sepanjang kunci *m* pada periode ke-i-1, terdapat satu kunci yang bersesuaian (K_{i-1}). Proses enkripsi yang dilakukan sama seperti proses pada vigenere cipher, yaitu karakter-karakter pada P_{i-1} digeser sejauh karakter pada kunci menjadi karakter-karakter pada C_{i-1}. Setelah itu, kunci akan digunakan untuk membangkitkan bilangan n_{i-1} yang diperlukan untuk menghasilkan bilangan $g_{i-1} = k\, n_{i-1} + 1$. Bilangan g_{i-1} inilah yang akan menjadi basis modulo untuk elemen kunci ke-i-1 dalam menghasilkan balikan yang akan menjadi elemen kunci ke-i.

Dalam membangiktkan barisan kunci tersebut, balikan modulo dipilih untuk menghasilkan kunci selanjutnya sebab untuk suatu bilangan bulat *a* dan *b*, balikan modulo *b* merupakan suatu hal yang tidak dapat dinyatakan secara eksplisit dalam bentuk *a*. Misalnya, balikan dari 3 modulo 7 adalah $7n + 5$ untuk seluruh bilangan bulat *n*. Dalam hal ini, 5 tidak dapat diduga berdasarkan 7 maupun 3 saja. Kita harus mengetahui keduanya, yang berarti bahwa kita harus mengetahui kunci maupun faktor kuncinya untuk bisa membangkitkan kunci berikutnya. Karena balikan bersifat simetris, ini berlaku pula untuk sebaliknya. Apabila kita menemukan suatu kunci ke-i namun tidak mengetahui faktor kunci, sulit untuk mengetahui kunci ke-i-1.

Dalam menghasilkan g_{i-1}, *k* merupakan faktor kunci (bersama-sama dengan kunci string menjadi kunci yang mendasari inverse generator cipher) dan n_{i-1} adalah kelipatan persekutuan terbesar elemen-elemen

kunci. Dalam hal ini, tiap elemen kunci yang merupakan suatu huruf diasosiasikan dengan bilangan antara 1-26, yaitu $a = 1$, $b = 2$, … , $z = 26$. Digunakan bilangan antara 1-26 dan bukan 0-25 seperti pada pergeseran vigenere cipher disebabkan oleh fakta bahwa seluruh kelipatan dari 0 adalah 0, yaitu tidak mungkin terdapat suatu bilangan asli p sehingga $p = q.0$ untuk suatu bilangan bulat q. Sebagai contoh, jika kunci = "rune", maka $r = 18$, $u = 21$, $n = 14$, dan $e = 5$, sehingga n = KPK(18,21,14,5) = 630.

Bilangan g_{i-1} merupakan bilangan yang relatif prima terhadap seluruh elemen kunci ke-i-1 berdasarkan akibat **Teorema (1)** karena $k\ n_{i-1}$ pasti habis dibagi dengan seluruh elemen kunci ke-i-1. Dengan demikian, berdasarkan **Teorema (3)**, balikan modulo g_{i-1} untuk seluruh elemen kunci ke-i-1 akan selalu terdefinisi. Dalam hal ini, faktor kunci diperlukan untuk memperkuat kunci itu sendiri. Dalam hal ini, faktor kunci dapat membuat nilai bilangan g_{i-1} lebih acak sehingga balikan elemen-elemen kunci pun akan semakin acak.

Dalam melakukan proses dekripsi, yang dilakukan kurang lebih sama, yaitu ketika melakukan dekripsi pada periode ke-i terhadap karakter-karakter pada C_{i-1} dengan cara menggeser terbalik berdasarkan kunci pada periode tersebut, kunci dan faktor kunci akan membangkitkan kunci pada periode berikutnya yang menjadi dasar untuk pergeseran pada periode itu.

Berikut ditampilkan pseudocode untuk enkripsi dengan inverse generator cipher :

```
program encryptInverseGeneratorCipher (input
PT:String, output CT:String, input
initKey:String, input factorKey:integer)
kamus
  key   : String
  i,j,k : integer
  GCD   : integer
  generated : integer
algoritma
  i ← 0
  j ← 0
  key <-- initKey
  // inisiasi key dengan nilai key semula

  while (i < Length(PT)) do
    CT(i) ← Encrypt(PT(i),key(j))
    // geser dengan aturan vigenere biasa

    if (j = Length(key)-1) then
        k ← 0
        GCD ← CountKPK(key)
        // menghitung nilai KPK
        generated ← GCD * factorKey + 1
        //    menghitung    nilai    bilangan
pembangkit
        while (k < Length(key)) do
          key[k] ← inverse(key[k],generated)
          // menentukan balikan dari tiap
elemen kunci untuk menjadi kunci berikutnya
          k ← k + 1
        endwhile
       j ← 0
    endif
  i ← i + 1
  j ← j + 1
  endwhile
endprogram
```

Pseudocode untuk dekripsi sama dengan untuk enkripsi, hanya saja fungsi `CT(i) ← Encrypt(CT(i),key(j))` diganti dengan fungsi `PT(i) ← Decrypt(CT(i),key(j))`

3.2. Aplikasi Inverse Generator Cipher

Berikut akan dilakukan enkripsi terhadap pesan "tujuh puluh tujuh" dengan kunci string "runee" dan faktor kunci 1 dengan menggunakan inverse generator cipher.

Tabel 2 Contoh Aplikasi Inverse Generator Cipher

P	t	u	j	u	h	p	u	l
K	r	u	n	e	e	w	c	n
E	17	20	13	4	4	22	2	13
E+1	18	21	14	5	5	23	3	14
C	k	o	w	y	l	l	w	y

P	u	h	t	u	j	u	h
K	k	k	y	m	n	o	o
E	10	10	24	12	13	14	14
E+1	11	11	25	13	14	15	15
C	e	r	r	g	w	i	v

Keterangan:
P = Plainteks
K = Kunci
C = Cipherteks
E = pergeseran
E + 1 = pergeseran + 1 (untuk membangkitkan bilangan)

Dalam contoh di atas, kunci "runee" memiliki n_1 = KPK (18,21,14,5) = 630. Dengan demikian, g_1("runee",1) = 1.630 + 1 = 631. Oleh karena itu, kunci berikutnya merupakan balikan modulo 631. Balikan dari 18 modulo 631 adalah 596 = 23 mod 26. Balikan dari 21 modulo 631 adalah 601 = 3 mod 26. Balikan dari 14 modulo 631 adalah 586 = 14 mod 26. Balikan dari 5 modulo 631 adalah 505 = 11 mod 26. Dengan demikian, kunci berikutnya adalah "wcnkk". Dapat dilihat bahwa sulit untuk menelusuri keterhubungan "wcnkk" dan "runee" kecuali KPK elemen-elemen "rune" diketahui – yang berarti bahwa kunci harus diketahui.

Apabila kita meninjau "wcnkk", kunci baru tersebut memiliki n_2 = KPK(23,3,14,11,11) =10626. Dengan demikian, proses pembangkitan balikan elemen-elemen kunci baru ini akan menghasilkan kunci berbeda dari sebelumnya. Dalam hal ini, g_2("wcnkk",1) = 1.10626 + 1 = 10627. Oleh karena itu, kunci berikutnya merupakan balikan modulo 10627. Balikan dari 23 modulo 10627 adalah 10165 = 25 mod 26. Balikan dari 3 modulo 10627 adalah 7085 = 13 mod 26. Balikan dari 14 modulo 10627 adalah 9868 = 14 mod 26. Balikan dari 11 modulo 10627 adalah 9661 = 15 mod 26. Dengan demikian, kunci berikutnya adalah "ymnoo".

Dengan demikian, diperoleh bahwa cipherteks dari

pesan plainteks (tanpa menggunakan spasi) "tujuhpuluhtujuh" dengan menggunakan inverse generator cipher dengan kunci "runee" dan faktor kunci 1 adalah "kowyllwyerrgwiv". Dapat dilihat bahwa tidak ditemukan kembali string yang berulang seperti pada vigenere cipher sehingga tidak analisis frekuensi dan Kasiski tidak dapat dilakukan.

3.3. Analisis dan Pengujian Inverse Generator Cipher

Seperti disebutkan sebelumnya, penggunaan balikan modulo dan faktor kunci akan sangat menyulitkan dalam memecahkan ciphertext karena tidak adanya pola yang dapat dikenali seperti pada contoh pada **bab 3.2.** Berikut akan diberikan beberapa contoh kasus yang melibatkan inverse generator cipher untuk mendiskusikan lebih lanjut akan hal ini. Dalam hal ini, akan ditinjau bermacam-macam kemungkinan kunci dan kemungkinan pemecahan ciphertext apabila bagian-bagian algoritma diketahui.

3.3.1. Kasus Kunci String Tunggal dan Faktor Kunci Satu

Dalam hal ini, dipilih kunci "b" dan faktor kunci 1, maka E+1('b') = 2. Dengan demikian, n = KPK(2) = 2. Oleh karena itu, $g = kn + 1 = 1.2 + 1 = 3$. Balikan dari 2 modulo 3 adalah 2. Dengan demikian, akan diperoleh kunci yang sama, yaitu "b". Hal ini disebabkan balikan dari $n - 1$ modulo n adalah $n - 1$ itu sendiri karena $(n - 1)(n - 1) = n^2 - 2n + 1 = 1 \bmod n$. Dengan demikian, kunci string tunggal dengan faktor kunci 1 akan menghasilkan barisan kunci karakter yang sama, dalam contoh ini yaitu "bbbbb…..", atau serupa dengan Caesar Cipher.

3.3.2. Kasus Kunci String Tunggal dan Faktor Kunci Lebih Dari Satu

Dalam hal ini, dipilih kunci "b" dan faktor kunci 2, maka E+1('b') = 2. Dengan demikian, n = KPK(2) = 2. Oleh karena itu, $g = kn + 1 = 2.2 + 1 = 5$. Balikan dari 2 modulo 5 adalah 3. Dengan demikian, akan diperoleh kunci yang berbeda, yaitu "c". Jika dilanjutkan, E+1('c') = 3. Dengan demikian, n = KPK(3) = 3. Oleh karena itu, $g = kn + 1 = 2.3 + 1 = 7$. Balikan dari 3 modulo 7 adalah 5. Hal ini akan menghasilkan kunci yang berbeda kembali, yaitu "e". Jika dilanjutkan, E+1('e') = 5. Dengan demikian, n = KPK(5) = 5. Oleh karena itu, $g = kn + 1 = 2.5 + 1 = 11$. Balikan dari 5 modulo 11 adalah 9. Hal ini akan menghasilkan kunci yang berbeda kembali, yaitu "i". Dengan demikian, akan diperoleh barisan kunci acak "bcei….". Dapat dilihat bahwa kunci string tunggal dan faktor kunci lebih dari satu sudah membentuk suatu proses enkripsi yang sulit dipecahkan.

3.3.3. Kasus Kunci Mengandung Huruf 'a'

Perhatikan bahwa E+1('a') = 1. Untuk nilai g berapapun, balikan dari 1 modulo g adalah 1 itu sendiri. Dengan demikian, setiap kunci yang mengandung 'a' akan menghasilkan kunci yang tetap mengandung 'a' ($\ldots x_1\text{'a'}y_1\ldots \rightarrow \ldots x_2\text{'a'}y_2\ldots$). Oleh karena itu, penggunaan karakter 'a' amat disarankan untuk dihindari untuk memperkuat kunci itu sendiri.

3.3.4. Kasus Kunci Mengandung Huruf Yang Sama

Perhatikan kembali aplikasi inverse generator cipher pada **bab 3.2.** Dapat dilihat bahwa kunci "runee" menghasilkan kunci "wcnkk", dan kunci "wcnkk" menghasilkan kunci "ymnoo". Karena karakter keempat dan kelima kunci tersebut sama, maka untuk seterusnya kedua karakter itu akan tetap sama. Sebagai contoh tambahan, tinjau kunci "bbbbb" dan faktor kunci 2. Karena E+1('b') = 2. Dengan demikian, n = KPK(2) = 2. Oleh karena itu, $g = kn + 1 = 2.2 + 1 = 5$. Balikan dari 2 modulo 5 adalah 3. Dengan demikian, akan diperoleh kunci "ccccc". Dengan cara yang sama, E+1('c') = 3. Dengan demikian, n = KPK(3) = 3. Oleh karena itu, $g = kn + 1 = 2.3 + 1 = 7$. Balikan dari 3 modulo 7 adalah 5. Hal ini akan menghasilkan kunci "eeeee". Dapat disimpulkan bahwa kunci yang dihasilkan akan selalu merupakan lima karakter yang sama, dalam hal ini "bbbbbccccceeeee…..".

Hal ini disebabkan meskipun inverse generator bekerja untuk suatu string sesuai panjang kunci, pada hakikatnya perubahan dari satu kunci ke kunci berikutnya merupakan perubahan per elemen-elemen kunci. Dengan kata lain, terdapat korespondensi antara elemen pertama kunci ke-1 dengan elemen pertama kunci ke-2, antara elemen kedua kunci ke-1 dengan elemen kedua kunci ke-2, dan seterusnya.

Tiga hal yang mempengaruhi perubahan elemen kunci tersebut adalah elemen kunci itu sendiri, elemen-elemen kunci keseluruhan dalam bentuk KPK, dan faktor kunci. Apabila terdapat dua atau lebih elemen yang sama dalam satu kunci, dapat disimpulkan bahwa elemen-elemen tersebut akan berubah ke arah yang sama seperti terlihat pada contoh diatas.

3.3.5. Pengujian Pemecahan Algoritma Apabila Kunci String Diketahui namun Faktor Kunci Tidak Diketahui

Untuk kasus ini, perhatikan contoh pada **bab 3.2**. Dalam kondisi ini, setiap faktor kunci akan menghasilkan kunci berbeda pula. Sebagai contoh, g_1("runee",2) = 2.630 + 1 = 1261. Oleh karena itu, kunci berikutnya merupakan balikan modulo 1261. Balikan dari 18 modulo 1261 adalah 1191 = 21 mod 26. Balikan dari 21 modulo 1261 adalah 1201 = 5 mod 26. Balikan dari 14 modulo 1261 adalah 1171 = 1 mod 26. Balikan dari 5 modulo 1261 adalah 1009 = 21 mod 26. Dengan demikian, kunci berikutnya adalah "ueauu". Dapat dilihat bahwa kunci ini berbeda dengan "wcnkk". Oleh karena itu, kasus ini serupa dengan One-Time Pad, yaitu mungkin saja faktor kunci tertentu yang berbeda dapat menghasilkan istilah bermakna.

3.3.6. Pengujian Pemecahan Algoritma Apabila

Faktor Kunci Diketahui namun Kunci String Tidak Diketahui

Kondisi ini tidak akan banyak membantu. Faktor kunci dibuat untuk memperkaya kemungkinan kunci sehingga lebih kuat. Namun, yang menjadi inti kunci adalah kunci string itu sendiri. Dengan demikian, misal untuk kasus pada bab **3.2.**, andaikan diketahui bahwa faktor kunci = 1, maka info tersebut bersama ciphertext "kowyllwyerrgwiv" akan tetap sulit dipecahkan.

3.3.7. Pengujian Pemecahan Algoritma Secara Brute Force

Terdapat suatu kesulitan dalam memecahkan secara brute force karena algoritma inverse generator cipher bekerja berdasar 2 variabel, kunci string dan faktor kunci. Oleh karena itu, pencarian secara *exhaustive search* memakan waktu lama. Secara umum, banyaknya kemungkinan kunci adalah $f(k,m) = k.26^m$ dengan k adalah banyaknya kemungkinan faktor kunci (merupakan barisan bilangan asli tak hingga – 1,2, …), dan m merupakan panjang kunci (1,2, … , panjangplainteks). Sebagai contoh, untuk panjang kunci 5 dan pencarian dibatasi hingga faktor kunci 10^9, maka terdapat $10^9.26^5$ atau sekitar $1{,}2.10^{16}$.

3.4. Implementasi Inverse Generator Cipher

Pada makalah ini hanya disinggung implementasi inverse generator cipher pada enkripsi pesan teks huruf. Namun, bukan berarti penggunaannya hanya sebatas itu. Karena inverse generator cipher pada dasarnya berbasiskan bilangan bulat dan bukan huruf-huruf, algoritma ini dapat diperluas untuk diaplikasikan tidak hanya ke dalam karakter-karakter huruf (A..Z yang disimbolkan oleh 0…25 untuk pergeseran dan 1…26 untuk perhitungan KPK dan balikan modulo), tetapi juga ke dalam seluruh karakter ASCII (0…255 untuk pergeseran dan 1…256 untuk perhitungan KPK dan balikan modulo). Dengan demikian, pergeseran yang digunakan bukanlah berdasarkan bujursangkar vigenere biasa, melainkan berdasarkan bujursangkar ASCII-vigenere yang berukuran 256 x 256 dan setiap baris mengandung seluruh karakter ASCII. Dengan cara seperti ini, kunci yang dapat digunakan pun tidak hanya string huruf saja, tetapi juga seluruh karakter ASCII. Dapat disimpulkan bahwa pengembangan vigenere cipher menjadi inverse generator cipher dapat diimplementasikan untuk mengenkripsi pesan dalam bentuk apapun (suara, gambar, dan sebagainya).

4. KESIMPULAN

Dari pembahasan di atas, dapat ditarik kesimpulan sebagai berikut.

1. Vigenere cipher merupakan salah satu contoh cipher-abjad majemuk yang kurang aman karena dapat dipecahkan dengan menggunakan gabungan metode Kasiski dan teknik analisis frekuensi.
2. Vigenere cipher memiliki karakteristik yaitu suatu karakter ciphertext dapat memiliki banyak kemungkinan makna plaintext dan dapat memunculkan string yang berulang-ulang.
3. Teori bilangan bulat terutama terkait dengan keterbagian dan modulo dapat digunakan untuk memodifikasi Vigenere Cipher dengan mengasosiasikan karakter A…Z ke dalam bilangan bulat 0…25 atau 1…26
4. Inverse Generator Cipher merupakan pengembangan dari vigenere Cipher dengan menggunakan bilangan yang dibangkitkan oleh elemen kunci dan faktor kunci
5. Inverse Generator Cipher berbeda dengan vigenere Cipher dalam penggunaan kunci; vigenere cipher menggunakan kunci yang sama berulang-ulang, sedangkan Inverse Generator Cipher membangkitkan suatu barisan kunci
6. Beberapa contoh kunci yang sebaiknya dihindari dalam penggunaan inverse generator cipher yaitu kunci string tunggal dengan faktor kunci satu, kunci dengan menggunakan karakter 'a', dan kunci yang menggunakan karakter-karakter yang sama
7. Apabila hanya salah satu bagian kunci (faktor kunci atau kunci string) yang diketahui, inverse generator cipher tetap sulit untuk dipecahkan. Sementara itu, pengujian dengan brute force akan memakan waktu amat lama
8. Inverse generator cipher dapat diperluas untuk diimplementasikan tidak hanya pada pesan teks huruf tetapi juga pesan apapun (suara, gambar, dan sebagainya).

DAFTAR REFERENSI

[1] R. Munir, "*Diktat Kuliah IF5054 Kriptografi*", Program Studi Teknik Informatika Institut Teknologi Bandung, 2006.

[2] R. Munir, "*Diktat Kuliah IF2151 Matematika Diskrit*", Departemen Teknik Informatika Institut Teknologi Bandung, 2005.

[3] G. Jones, M. Jones, "*Elementary Number Theory*", Springer Verlag London, 1998.

[4] G. Gamble, "*Number Theory*", University of Western Australia, 1997.

[5] T. Leary, "Cryptology in the 16th and 17th Centuries", http://home.att.net/~tleary/cryptolo.htm, 1997.

[6] A. Menezes, P. Oorschot, S. Vanstone, "Handbook of Applied Cryptography", http://www.cacr.math.uwaterloo.ca/hac/, 2001.