Algoritma Kriptografi Kamus dengan Memanfaatkan Teorema Kecil Fermat

Damiann Muhammad Mangan, 13510071
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13510071@std.stei.itb.ac.id

***Abstrak*—Dalam Makalah ini, penulis akan memperkenalkan algoritma kriptografi yang memanfaatkan kamus, seperti halnya *one-time pad*, setelah dibangkitkan dengan memanfaatkan sifat bilangan prima pada teorema kecil Fermat. Algoritma ini dapat diaplikasikan baik dalam bentuk klasik maupun modern. Walaupun algoritm ini akan membentuk kata sandi yang memiliki ukuran lebih besar dari informasi yang dibawa, mampu memiliki kunci yang *generic* (tidak spesifik). Penulis juga akan memaparkan hasil implementasi algoritma ini.**

***Kata Kunci*—Algoritma kriptografi simetris, bilangan prima, kamus, kunci *generic*, teorema kecil Fermat.**

I. PENDAHULUAN

Kriptografi merupakan praktik dan pembelajaran cara mengamankan pertukaran informasi walaupun adanya keberadaan pihak ketiga; adapula yang berpendapat bahwa kriptografi adalah ilmu dan seni untuk menjaga kerahasiaan pesan dengan cara menyandikan ke bentuk yang dapat dimengerti lagi maknanya.

Di lain pihak, kriptanalisis merupakan praktik dan pembelajaran cara membongkar sandi kembali menjadi bentuk yang dapat diambil informasinya tanpa kepemilikan kunci; atau ilmu dan seni untuk membongkar pesan sandi rahasia ke dalam bentuk yang dapat dimengerti lagi maknanya.

Oleh karena adanya ilmu kriptanalisis, teknik penyandian informasi terus berkembang. Ilmu matematika pun dimanfaatkan agar informasi yang disandikan semakin sulit dibongkar artinya.

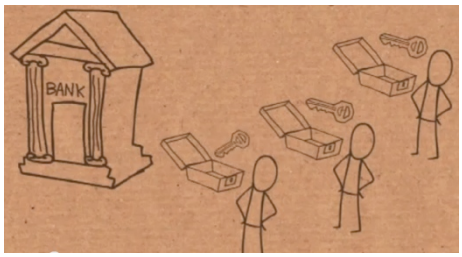
Dalam matematika, terdapat bilangan unik yang hanya dapat habis dibagi oleh bilangan itu sendiri atau angka satu; bilangan ini biasa disebut bilangan prima. Salah satunya adalah teorema kecil Fermat yang pada bagian selanjutnya akan dijelaskan. Sebelum adanya kriptografi, bilangan ini hanya memiliki sifat unik dan tidak memiliki fungsi dalam kehidupan sehari-hari. Contoh algoritma yang memanfaatkan sifat bilangan prima dan teorema kecil Fermat adalah algoritma RSA, algoritma kriptografi asimetris yang dikembangkan oleh Ron Rivest, Adi Shamir, dan Leonard Adleman pada tahun 1977.

II. KRIPTOGRAFI SIMETRIS DAN ASIMETRIS

Kriptografi simetris merupakan kriptografi yang menggunakan kata kunci yang sama baik pada dekripsi maupun enkripsi; berbeda dengan kriptografi asimetris yang menggunakan kata kunci berbeda pada dekripsi dan enkripsi, biasa disebut kunci privat-publik.

Kriptografi simetris sampai sekarang masih dipakai dalam pengamanan berkas untuk disimpan, walaupun sudah sangat jarang dipakai dalam bertukar rahasia. Kriptografi jenis ini berkurang popularitasnya karena diperlukan daftar kunci yang tidak *scalable* dengan banyaknya lawan bertukar pesan, karena tanpa keunikan kata kunci pada setiap lawan bertukar pesan akan muncul. Hal ini akan mudah ditangkap setelah dianalogikan dalam bentuk kunci dan gembok.

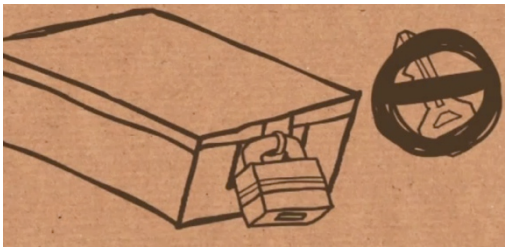
Pada sebuah Bank X digunakan kriptografi simetris untuk mengamankan data-data klien-kliennya, sebut saja A dan B. Karena digunakan kriptografi simetris, dapat dianalogikan bahwa A dan B menerima kunci K untuk membuka data milik masing-masing; tetapi kenyatannya kunci tersebut dapat membuka seluruh berkas yang diamankan dengan kunci yang sama, sehingga A dapat melihat informasi berkas milik B apabila ia mencurinya, begitupun sebaliknya. Sehingga keamanan kurang terjamin karena hal ini menyebabkan berkas tersebut bisa dibuka selain klien yang merupakan pemilik dan bank yang menyediakan fasilitas pengamanan tersebut.



Gambar 1. Bank X yang memberikan A dan B kunci-kunci yang sama.

Gambar 2. A dapat membuka berkas B karena mereka memiliki kunci yang sama.

Lain halnya dengan Bank Y yang menggunakan kriptografi asimetris; fasilitas bank ini tidak akan memberikan klien-kliennya kunci untuk membuka berkas, melainkan gembok Pb. Dengan tidak diberikannya kunci Pv, C dan D yang merupakan klien-klien bank ini hanya akan mampu menutup berkasnya; sehingga C tidak dapat mendapatkan makna berkas milik D walaupun ia sudah mendapatkan berkasnya, hanya pihak bank yang mampu membuka berkas masing-masing karena hanya kunci Pv yang mampu membuka gembok Pb. Gembok Pb tidak mampu membuka gembok Pb lainnya.



Gambar 3. Bank memberikan klien-kliennya gembok, bukan kunci.

Gambar 4. C tidak dapat membuka berkas D, karena tiap klien hanya diberikan gembok, bukan kunci.

Gambar 5. Hanya bank yang mampu membuka berkas klien-kliennya, karena hanya bank yang memiliki kunci yang mampu membuka gembok.

Dari kedua contoh diatas, dapat disimpulkan bahwa algoritma asimetris jauh lebih efisien dalam kehidupan nyata. Dari n orang, apabila seluruhnya saling bertukar ke seluruh lainnya, dengan algoritma kriptografi simetris akan diperlukan,

$$\sum_{i=1}^{n}\sum_{j=1}^{i} j = \frac{1}{6}n(n+1)(n+2),$$

buah kunci unik agar tidak ada satupun dari anggoya tersebut yang m 100, diperlukan 171700 kunci unik. Sedangkan dengan algoritma kriptografi asimetris, cukup digunakan n buah kunci unik untuk n orang agar tidak ada satupun dari anggota kelompok orang tersebut yang mampu membuka berkas anggota lain. Dengan kata lain, algoritma kriptografi asimetris sangat *scalable*, berbeda dengan algoritma kriptografi simetris.

Sesuai dengan abstrak yang dipaparkan pada awal, algoritma yang diperkenalkan pada makalah ini membangkitkan kamus, sehingga memiliki konsep dasar yang sama dengan algoritma kriptografi simetris.

III. One-Time Pad

Dalam kriptografi simetris, terdapat teknik enkripsi agar pesan yang sudah menjadi sandi sama sekali tidak bisa dipecahkan oleh kriptanalisis tanpa memiliki penerjemah. Penerjemah tersebut adalah kamus; dalam kamus ini, tidak terdapat pola-pola sehingga tidak dapat diambil kesimpulan setelah sandi tersebut dianalisis.

Walaupun teknik kriptografi ini sangat kokoh dalam teori, teknik ini sangat lemah dalam kehidupan nyata; hal ini berkaitan dengan diperlukannya waktu yang cukup lama dalam membentuk kamus, mengenkripsi pesan, maupun mendekripsikan pesan.

Sebaliknya, teknik-teknik konvensional yang lemah secara teori, kokoh dalam kehidupan nyata; karena tidak perlu digunakan kunci dalam mendekripsikan pesan, cukup digunakan kunci.

Algoritma yang akan diperkenalkan pada makalah ini memiliki kunci yang tidak konvensional, kunci yang *generic*, sehingga akan sangat sulit dilakukan kriptanalisis terhadapnya; walaupun pada akhirnya algoritma ini memiliki kelemahan yang dimiliki algoritma simetris, yaitu tidak *scalable*. Kelemahan lain yang terdapat pada algoritma ini adalah hasil enkripsi seringkali lebih besar dari dekripsi.

IV. Kriptografi Klasik dan Modern

Pada awalnya kriptografi digunakan untuk menyembunyikan informasi pesan pada teks dengan cara mengganti huruf alfabet ke alfabet lainnya ataupun kombinasi huruf-huruf alfabet ke kombinasi huruf-huruf lainnya; hal ini juga dapat memanfaatkan matematika dengan merepresentasikan huruf-huruf kedalam angka. Kriptografi yang berkisar pada teks alfabet disebut dengan kriptografi klasik.

Setelah dunia sudah terekspos oleh era dijital, kriptografi pun berkembang. Penggunaan bit digunakan untuk merepresentasikan semua hal, termasuk huruf dan bahkan lambang-lambang dalam literatur. Kriptografi modern lah kriptografi yang memanfaatkan dan merekayasa bit dalam mengubah informasi ke dalam bentuk yang tidak dapat diartikan.

Sesuai dengan abstrak yang dipaparkan di awal, algoritma yang diperkenalkan pada makalah ini dapat bersifat klasik maupun modern karena teknik-teknik yang digunakan untuk membangkitkan kamus memanfaatkan sifat-sifat bilangan prima dalam ilmu matematika.

VI. Bilangan Prima dan Teorema Kecil Fermat

A. Bilangan Prima

Definisi bilangan prima adalah bilangan bulat positif yang bukan merupakan bilangan kuadrat dan hanya bisa habis dibagi oleh bilangan itu sendiri atau angka satu. Tetapi, ternyata sifat bilangan prima bukan hanya itu; sampai sekarang, belum ditemukan rumus umum untuk mendapatkan bilangan prima, hanya ada teknik untuk memeriksa apakah suatu bilangan termasuk bilangan prima. Sehingga menggunakan bilangan prima sebagai kunci akan mempersulit kriptanalis untuk mendekripsikan sandi yang sudah dienkripsi.

Sampai sekarang, bilangan prima terbesar yang sudah ditemukan adalah $2^{57,885,161} - 1$, ditemukan pada 25 Januari 2013 oleh GIMPS (*Great Internet Mersenne Prime Search*); GIMPS sendiri adalah proyek kolaboratif yang dapat dijalankan oleh sukarelawan secara bebas untuk mencari bilangan prima Mersenne, bilangan prima dengan bentuk $2^n - 1$. Hal yang menarik disini adalah beda nilai pangkat pada dua, n, antara bilangan prima terbesar sekarang dengan bilangan prima terbesar kedua sekarang adalah 14,772,552. Hal ini dapat memberi kita bayangan bahwa dari satu bilangan prima, bilangan prima selanjutnya yang memiliki bentuk serupa belum tentu berdekatan nilainya.

Pada bilangan prima pun berlaku teorema kecil Fermat, yang dinamakan dari penemunya yaitu Pierre de Fermat dan ditemukan pada 1640. Sub-bagian selanjutnya akan membahas teorema ini.

B. Teorema Kecil Fermat

Pada tahun ditemukannya—teorema ini hanyalah hiburan, hanyalah fakta menarik pada bilangan prima—sama sekali tidak ada aplikasi dalam kehidupan nyata. Hanya setelah muncul internetlah teorema ini berguna, bahkan digunakan dimana-mana, digunakan di berbagai pertukaran pesan, dengan adanya algoritma RSA yang memanfaatkan teorema ini. Pada paragraf selanjutnya, teorema ini akan dijelaskan.

Teorema kecil Fermat mengklaim bahwa untuk setiap bilangan bulat a, dan bilangan prima p, maka $a^p - a$ habis dibagi oleh p. Dalam notasi modulo, ekspresi ini dapat dituliskan sebagai,

$$a^p \equiv a \pmod{p}.$$

Jika a tidak habis dibagi oleh p, maka bisa didapatkan ekspresi ini,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Walaupun makalah ini memanfaatkan dan menjelaskan teorema ini, pembuktiannya diserahkan kepada pembaca untuk dibaca dan dipahami karena sangat menarik.

Selain teorema ini, ada teorema Euler yang merupakan generalisasi dari teorema kecil Fermat. Teorema ini bermakna sebagai berikut; untuk modulus n apapun dan bilangan a yang saling prima dengan n, terdapat,

$$a^{\phi(n)} \equiv 1 \pmod{n},$$

dengan $\phi(n)$ sebagai *Euler's totient function*, yang berarti banyaknya bilangan yang saling prima dengan bilangan tersebut dan lebih kecil dari bilangan tersebut. Pembuktian teorema ini pun diserahkan juga ke pembaca.

Dalam pembentukan kamus oleh algoritma ini, cukup dimanfaatkan persamaan,

$$a^{p-1} \equiv 1 \pmod{p}.$$

VII. Algoritma

Agar memiliki hasil enkripsi yang sulit dipecahkan, algoritma ini memiliki kunci yang *generic*. Ada tiga bagian pada kunci, yaitu X, Y, dan Z[], dengan X adalah bilangan bulat tidak negatif, Y adalah bilangan asli, dan Z adalah senarai bilangan prima yang terus membesar.

Untuk menjelaskan manfaat bagian-bagian pada kunci tersebut, perlu dijelaskan terlebih dahulu cara kerja algoritma yang mampu membangkitkan kamus dari kunci yang diberikan.

Algoritma ini memanfaatkan kongruensi yang diberikan oleh teorema kecil Fermat, $a^{p-1} \equiv 1 \pmod{p}$. Untuk menjalankan algoritma ini, dimanfaatkan *lemma* sebagai berikut,

> "Untuk $a_i \neq a_j$ dengan $i, j \in \{1, 2, \ldots, n\}$ dan $n < p$, selama $a_i < p$, ${a_i}^{p-2} \pmod{p}$ akan menghasilkan nilai x_i yang bersifat unik."

Lemma diatas dapat dengan mudah didapat kebenarannya. Karena ${a_i}^{p-1} \equiv 1 \pmod{p}$, ${a_i}^{p-2} \equiv$

$a_i^{-1} \pmod p$; perhatikan bahwa $a_i^{p-2} \equiv x_i \pmod p$, sehingga $x_i \equiv a_i^{-1} \pmod p$. Dari kongruensi terakhir didapat bahwa $x_i \cdot a_i \equiv 1 \pmod p$ untuk setiap $i \in \{1, 2, 3, \ldots, n\}$. Sehingga $x_i \cdot a_i \equiv x_j \cdot a_j \pmod p$, untuk $j \in \{1, 2, 3, \ldots, n\}$. Pada saat $i \neq j$, $a_i \neq a_j$ sehingga $x_i = x_j$ tidak mungkin terjadi. Secara ringkas, pembuktian diatas memberikan kesimpulan bahwa,

$$a_i^{p-2} \equiv x_i \pmod p$$

dengan x_i unik oleh a_i yang unik.

Dengan adanya *lemma* tersebut, dapat dilakukan pembangkitan kamus yang menggunakan a_i sebagai sebuah ataupun kombinasi huruf, bahkan sebuah simbol yang dapat direpresentasikan dalam bentuk biner maupun kombinasinya, asalkan $i \in \{1,2, \ldots, n\}$ dan $n < p$. Contohnya, untuk A = 1, B = 2, …, dan Z = 26, dan p adalah bilangan prima terkecil yang melebihi 26 (jumlah huruf), yaitu 29. Sesuai dengan *lemma* yang penulis ajukan, bilangan unik yang dipangkatkan dengan bilangan prima yang sudah dikurangi angka dua akan menghasilkan nilai yang unik setelah dimodulo dengan prima itu sendiri; maka akan terbentuk tabel sebagai berikut,

Huruf (a_i)	Hasil (x_i)	Huruf (a_i)	Hasil (x_i)
A = 0	0 = A	N = 13	9 = J
B = 1	1 = B	O = 14	27 = BB
C = 2	15 = P	P = 15	2 = C
D = 3	10 = K	Q = 16	20 = U
E = 4	22 = W	R = 17	12 = M
F = 5	6 = G	S = 18	21 = V
G = 6	5 = F	T = 19	26 = BA
H = 7	25 = Z	U = 20	16 = Q
I = 8	11 = L	V = 21	18 = S
J = 9	13 = N	W = 22	4 = E
K = 10	3 = D	X = 23	24 = Y
L = 11	8 = I	Y = 24	23 = X
M = 12	17 = R	Z = 25	7 = H

Dari hasil diatas, terlihat bahwa *lemma* yang disampaikan sudah benar; masalah yang didapat adalah adanya BB dan BA dalam hasil enkripsi.

Dengan pembentukan kamus satu huruf seperti ini akan sangat mudah dilakukan kriptanalisis, teknik yang efektif untuk perubahan dari satu huruf ke satu huruf lainnya adalah dengan serangan statistik; untuk itu, penulis membuat bagian kunci tambahan, yaitu Y. Y bermakna sebagai jumlah simbol yang akan dienkripsi. Berdasarkan makna tersebut, Y pada tabel diatas adalah 1, karena hanya dilakukan pemetaan dari satu huruf.

Bagian kunci pertama pada algoritma ini, yaitu X, adalah nilai awal dari seluruh simbol; berdasarkan makna X, nilai X pada tabel diatas adalah 0. Sedangkan, bagian terakhir kunci, yaitu Z[], adalah senarai bilangan prima yang terus membesar dan seluruhnya lebih besar dari jumlah simbol ataupun kombinasi simbol; sehingga nilai Z[] agar tabel diatas dapat dihasilkan adalah [29].

Untuk Z[] yang memiliki anggota lebih dari satu, hasil enkripsi dari kamus tersebut akan dienkripsi lagi ke bilangan prima selanjutnya yang lebih besar hingga dicapai bilangan prima terakhir yang merupakan anggota Z. Dengan kata lain, x_i dari a_i akan dipangkatkan lagi dengan $p[1] - 2$, agar hasil enkripsi semakin sulit dianalisis dan memanfaatkan kunci yang *generic*.

Dengan fungsi X, Y, dan Z[], sebagai bagian-bagian dari kunci, sesuai dengan yang dijelaskan, kita dapat membentuk kamus yang sangat kompleks. Cukup dipilih X dan Y nilai yang cukup besar dan Z[] terdiri dari beberapa bilangan prima yang banyak dan cukup besar.

VIII. IMPLEMENTASI

Penulis membuat program dengan kode dalam bahasa python dengan antarmuka *command line* sederhana. Program ini mampu mengenkripsi berkas tulisan sederhana, yaitu dalam bentuk ASCII (*American Standard Character for Information Interchange*). Berikut ini adalah kodenya. Hasil enkripsi disimpan dalam bentuk murni angka, agar mudah diimplementasi dan untuk selanjutnya, *encoding* dapat dilakukan dengan bebas oleh pengguna.

```
print 'This program will make dictionary,'
print 'that could be used either to encrypt or decrypt ASCII text file.'

do = ''
while do != 'y' or do != 'n':
    do = raw_input('Encrypt [y] or decrypt [n]? ')

file_name = raw_input('Input textfile name: ')
x = raw_input('Input X value: ')
y = raw_input('Input Y value: ')
z = []
SIZE = raw_input('Input Z size: ')
for i in range(SIZE):
    z.append(int(raw_input('Input Z[{0}] value: '.format(i))))
dictionary = {}

def base_ASCII(string):
    length = len(string)
    temp = 0
    for i in string:
        temp += ord(i) * (128 ** (length - 1))
        length -= 1
    return temp

def mod_exp(base, mod):
    exp = mod - 2
    temp = base % mod
    for i in range(exp - 1):
        temp = (temp * base) % mod
    return temp

def algoXYZ(x, y, z):
    DICT_SIZE = 128 ** y
    start = x
    dictionary = {}
    for i in range(len(z)):
        for j in range(x, DICT_SIZE + x):
            dictionary[j] = mod_exp(j, z[i])
    return dictionary

dictionary = algoXYZ(X, Y, Z)

if do == 'y':
    with open(file_name, 'rb') as plain_file:
        with open('encrypted {0}'.format(file_name), 'wb') as encrp_file:
            while True:
                comb = plain_file.read(y)
                if not comb:
                    break
                temp = base_ASCII(comb)
                encrp_file.write(str(dictionary[temp]) + '\n')
##masih terdapat kesalahan dalam dekripsi
##else:
##  with open(file_name, 'rb') as encrp_file:
##    with open('decrypted {0}'.format(file_name), 'wb') as decrp_file:
##      while True:
```

```
##         comb = encrp_file.readline().strip()
##         if not comb:
##           break
##         for i in dictionary:
##           if i == comb:
##             decrp_file.write(str(i))
##           break
```

Setelah dilakukan enkripsi dengan kunci X = 0, Y = 1, dan Z[] = [157, 223], data *plaintext* sebagai berikut,

nama berkas: plain_text.txt

```
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
```

berubah menjadi berkas sandi sebagai berikut,

nama berkas: encrypted plain_text.txt (‘\n’ sesungguhnya merupakan perintah ke baris selanjutnya; tetapi demi penghematan spasi, penulis mengubahnya menjadi simbol tersebut.)

```
179\n221\n45\n53\n178\n7\n17\n2\n64\n61\n178\n7\n29\n221\n159\n221\n45\n7\n64
\n17\n25\n7\n23\n178\n53\n25\n147\n7\n214\n221\n148\n64\n53\n214\n25\n53\n25\
n61\n45\n7\n23\n29\n17\n2\n17\n64\n17\n214\n17\n148\n13\n7\n53\n159\n17\n25\n
147\n7\n64\n53\n29\n7\n29\n221\n7\n53\n17\n61\n64\n178\n221\n29\n7\n25\n53\n1
78\n2\n221\n45\n7\n17\n148\n214\n17\n29\n17\n29\n61\n148\n25\n7\n61\n25\n7\n1
59\n23\n66\n221\n45\n53\n7\n53\n25\n7\n29\n221\n159\n221\n45\n53\n7\n178\n23\
n13\n148\n23\n7\n23\n159\n17\n75\n61\n23\n160\n7\n21\n25\n7\n53\n148\n17\n178
\n7\n23\n29\n7\n178\n17\n148\n17\n178\n7\n206\n53\n148\n17\n23\n178\n147\n7\n
75\n61\n17\n64\n7\n148\n221\n64\n25\n45\n61\n29\n7\n53\n210\n53\n45\n214\n17\
n25\n23\n25\n17\n221\n148\n7\n61\n159\n159\n23\n178\n214\n221\n7\n159\n23\n66
\n221\n45\n17\n64\n7\n148\n17\n64\n17\n7\n61\n25\n7\n23\n159\n17\n75\n61\n17\
n2\n7\n53\n210\n7\n53\n23\n7\n214\n221\n178\n178\n221\n29\n221\n7\n214\n221\n
148\n64\n53\n75\n61\n23\n25\n160\n7\n82\n61\n17\n64\n7\n23\n61\n25\n53\n7\n17
\n45\n61\n45\n53\n7\n29\n221\n159\n221\n45\n7\n17\n148\n7\n45\n53\n2\n45\n53\
n208\n53\n148\n29\n53\n45\n17\n25\n7\n17\n148\n7\n206\n221\n159\n61\n2\n25\n2
3\n25\n53\n7\n206\n53\n159\n17\n25\n7\n53\n64\n64\n53\n7\n214\n17\n159\n159\n
61\n178\n7\n29\n221\n159\n221\n45\n53\n7\n53\n61\n7\n129\n61\n13\n17\n23\n25\
n7\n148\n61\n159\n159\n23\n7\n2\n23\n45\n17\n23\n25\n61\n45\n160\n7\n181\n210
\n214\n53\n2\n25\n53\n61\n45\n7\n64\n17\n148\n25\n7\n221\n214\n214\n23\n53\n2
14\n23\n25\n7\n214\n61\n2\n17\n29\n23\n25\n23\n25\n7\n148\n221\n148\n7\n2\n45
\n221\n17\n29\n53\n148\n25\n147\n7\n64\n61\n148\n25\n7\n17\n148\n7\n214\n61\n
159\n2\n23\n7\n75\n61\n17\n7\n221\n129\n129\n17\n214\n17\n23\n7\n29\n53\n64\n
53\n45\n61\n148\n25\n7\n178\n221\n159\n159\n17\n25\n7\n23\n148\n17\n178\n7\n1
7\n29\n7\n53\n64\n25\n7\n159\n23\n66\n221\n45\n61\n178\n160\n
```

IX. SIMPULAN

Algoritma dengan kunci X, Y, dan Z[] ini memiliki kunci yang cukup dinamis (atau *generic*) sehingga cukup sulit dipecahkan. Nilai X digunakan untuk mengaburkan nilai-nilai hasil enkripsi; sedangkan nilai Y digunakan untuk membuat hasil enkripsi lebih kompleks, walaupun memiliki konsekuensi yang sangat besar terhadap ukuran dan waktu pembentukan kamus.

Senarai Z memiliki ukuran yang dinamis, karena apabila Z hanya memiliki satu bilangan prima, kriptanalis dapat menerka bilangan yang mungkin berdasarkan angka terbesar hasil enkripsi; dengan Z memiliki anggota lebih dari satu, kriptanalis akan dipersulit dengan adanya bilangan prima diantaranya, bahkan banyaknya bilangan prima yang berada diantaranya akan sulit DITERKA.

Kelemahan algoritma ini terdapat pada waktu perhitungannya, karena menggunakan perpangkatan bilangan besar. Sifat algoritma kriptografi simetris pun tidak dapat dihindari, sehingga dianjurkan pemakaian algoritma kriptografi asimetris untuk bertukar kunci dahulu.

Algoritma ini juga memiliki kelemahan dalam ukuran berkas sandi, karena hasil terjemahan dari simbol awal oleh kamus seringkali terdapat nilai yang lebih besar dari nilai terbesar yang terdapat pada simbol awal. Contoh pada tabel alfabet diatas pun pada akhirnya diperlukan simbol ‘BA’ dan ‘BB’ karena 26 dan 27 sudah diluar dari batas nilai alfabet.

VII. ACKNOWLEDGMENT

The preferred spelling of the word “acknowledgment” in American English is without an “e” after the “g.” Use the singular heading even if you have many acknowledgments. Avoid expressions such as “One of us (S.B.A.) would like to thank ... .” Instead, write “F. A. Author thanks ... .” Sponsor and financial support acknowledgments are placed in the unnumbered footnote on the first page.

REFERENCES

[1] Setya Budi, Wono, “Langkah Awal Menuju Olimpiade Matematika,” 2005.

[2] Munir, Rinaldi. (2004). Bahan Kuliah IF5054. Kriptografi. Departemen Teknik Informatika, Institut Teknologi. Bandung.

[3] http://www.youtube.com/watch?v=M7kEpw1tn50, diakses pada 18 Maret 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, 25 Maret 2013

Damiann Muhammad Mangan, 13510071