Perbandingan Penggunaan Bilangan Prima Aman Dan Tidak Aman Pada Proses Pembentukan Kunci Algoritma Elgamal

Yudhi Andrian
STMIK Potensi Utama
E-mail: **yudhi.andrian@gmail.com**

Abstrak

Algoritma ElGamal merupakan algoritma dalam kriptografi yang termasuk dalam kategori algoritma asimetris. Keamanan algoritma ElGamal terletak pada kesulitan penghitungan logaritma diskret pada bilangan modulo prima yang besar sehingga upaya untuk menyelesaikan masalah logaritma ini menjadi sangat sukar. Algoritma ElGamal terdiri dari tiga proses, yaitu proses pembentukan kunci, proses enkripsi dan proses dekripsi. Proses pembentukan kunci kriptografi ElGamal terdiri dari pembentukan kunci privat dan pembentukan kunci public. Pada proses ini dibutuhkan sebuah bilangan prima aman yang digunakan sebagai dasar pembentuk kunci public sedangkan sembarang bilangan acak digunakan sebagai pembentuk kunci privat. Pada penelitian sebelumnya digunakan bilangan prima aman pada proses pembentukan kunci namun tidak dijelaskan alasan mengapa harus menggunakan bilangan prima aman tersebut. Penelitian ini mencoba membandingkan penggunaan bilangan prima aman dan bilangan prima tidak aman pada pembentukan kunci algoritma elgamal. Analisa dilakukan dengan mengenkripsi dan dekripsi sebuah file dengan memvariasikan nilai bilangan prima aman dan bilangan prima tidak aman yang digunakan untuk pembentukan kunci public dan kunci privat. Dari hasil analisa dapat disimpulkan bahwa dengan menggunakan bilangan prima aman maupun bilangan prima tidak aman, proses pembentukan kunci, enkripsi dan dekripsi tetap dapat berjalan dengan baik, semakin besar nilai bilangan prima yang digunakan, maka kapasitas cipherteks juga semakin besar.

Kata Kunci *— Kriptografi, Enkripsi, Dekripsi, Elgamal, Bilangan Prima Aman*

Abstract

Elgamal algorithm is an algorithm in cryptography that is included in the category of asymmetric algorithms. The security of Elgamal algorithm lies in the difficulty in calculating the discrete logarithm on large number of prime modulo that attempts to solve this logarithm problem becomes very difficult. Elgamal algorithm is consists of three processes, that are the key generating, encryption and decryption process. Key generation of elgamal cryptography process is consisted of the formation of the private key and public key. In this process requires a secure prime number is used as the basis for forming public key while any random number used as forming of the private key. In the previous research is used secure prime number on key generating process but does not explain the reasons of using the secure primes. This research tried to compare using secure and unsecure primes in elgamal key generating algorithm. The analysis is done by encrypting and decrypting a file by varying the value of secure and unsecure of prime numbers that are used on generating of a public and a private key. From the analysis it can be concluded that using secure and unsecure of prime numbers, the process of key generating, encryption and decryption can run well, the greater value of prime numbers are used, the greater the capacity of the ciphertext.

Keywords *— Cryptography, Encryption, Decryption, Elgamal, Secure Prime Value*

1. PENDAHULUAN

Seiring dengan perkembangan teknologi saat ini, dimana pertukaran data melalui media internet sudah menjadi kebutuhan, maka keamanan data saat melakukan pertukaran data atau transaksi melalui media internet menjadi sangat penting. Salah satu cara untuk mengamankan data saat melakukan pertukaran data adalah dengan menyandikan data tersebut.

Kriptografi merupakan ilmu tentang penyandian data. Dengan melakukan penyandian data maka hanya orang yang mengetahui penyandian tersebut yang dapat membaca data yang disandikan. Banyak metode/algoritma yang digunakan dalam proses penyandian data, salah satunya adalah Algoritma Elgamal. Algoritma ElGamal merupakan algoritma dalam kriptografi yang termasuk dalam kategori algoritma asimetris. Keamanan algoritma ElGamal terletak pada kesulitan penghitungan logaritma diskret pada bilangan modulo prima yang besar sehingga upaya untuk menyelesaikan masalah logaritma ini menjadi sangat sukar.[7]

Algoritma ElGamal terdiri dari tiga proses, yaitu proses pembentukan kunci, proses enkripsi dan proses dekripsi. Proses pembentukan kunci kriptografi ElGamal terdiri dari pembentukan kunci *privat* (rahasia) dan pembentukan kunci *public* (umum). Pada proses ini dibutuhkan sebuah bilangan prima *p* yang digunakan untuk membentuk grup Zp* dan *elemen primitif* α (*primitive root*) sebagai dasar pembentuk kunci *public* (umum) sedangkan sembarang bilangan acak $a \in \{0, 1, p\text{-}2\}$ digunakan sebagai pembentuk kunci *privat* (rahasia). [4]

Dalam penelitian sebelumnya, Mukhammad Ifanto melakukan penelitian metode enkripsi dan dekripsi menggunakan algoritma elgamal, dimana pada penelitian ini proses pembentukan kunci menggunakan bilangan prima aman besar dan elemen primitive. [4]. Danang Tri Massandy menggunakan algoritma elgamal dalam pengamanan pesan rahasia. [3]. Eko aribowo membuat aplikasi pengamanan dokumen office dengan algoritma kunci asimetris elgamal. [1]. M. Taufiq Tamam, et. Al menerapkan kriptografi elgamal untuk mengamankan file citra. [5]. Agustinus widyartono menggunakan algoritma elgamal untuk enkripsi data menggunakan GNUPG. [6].

Yudhi Andrian melakukan penelitian tentang analisis penggunaan elemen primitif dan non primitif pada algoritma elgamal.[2]. Khairul ummi melakukan penelitian tentang analisis penggunaan bilangan prima aman besar pada algoritma elgamal. [7]

Dari penelitian sebelumnya, belum dijelaskan alasan digunakannya bilangan prima aman pada proses pembentukan kunci yang akan digunakan pada proses enkripsi dan dekripsi metode elgamal. Berdasarkan uraian sebelumnya, penelitian ini mencoba membandingkan pengaruh digunakannya bilangan prima aman pada proses pembentukan kunci, apakah ada pengaruhnya jika yang digunakan adalah bilangan prima tidak aman pada pembentukan kunci algoritma elgamal, dan bagaimana perbandingannya antara penggunaan bilangan prima aman dengan bilangan prima tidak aman pada proses pembentukan kunci algoritma elgamal.

1.1. Penentuan Bilangan Prima Aman dan Bilangan Prima Tidak Aman

Bilangan prima adalah bilangan asli yang lebih besar dari 1, yang faktor pembaginya adalah 1 dan bilangan itu sendiri. Sebuah bilangan prima dikatakan aman jika memenuhi persamaan berikut :

$$p = 2.q + 1 \qquad (1)$$

Dari persamaan (1) dapat dijelaskan bahwa bilangan prima p disebut sebagai bilangan prima aman jika q juga merupakan bilangan prima. Namun jika q bukan bilangan prima, maka p bukan merupakan bilangan prima aman atau p adalah bilangan prima tidak aman.

Rumus : p = 2 x q + 1
Contoh 1 : 7 = 2 x 3 + 1
Contoh 2 : 13 = 2 x 6 + 1

Pada contoh 1, nilai p adalah 7 yang merupakan bilangan prima dan nilai q adalah 3 yang juga merupakan bilangan prima, maka nilai 7 merupakan bilangan prima aman. Pada contoh 2, nilai p adalah 13 yang merupakan bilangan prima, akan tetapi nilai q adalah 6 yang bukan merupakan bilangan prima, maka nilai 13 bukan merupakan bilangan prima aman atau 13 adalah bilangan prima tidak aman.

Langkah penentuan bilangan prima aman dan tidak aman dinyatakan sebagai berikut:

a. Menentukan bilangan prima p dimana p ≥ 5,
b. Menghitung nilai q dengan "persamaan (1)" ,
c. Jika nilai q merupakan bilangan prima, maka nilai p merupakan bilangan prima aman,
d. Jika nilai q bukan merupakan bilangan prima, maka nilai p bukan merupakan bilangan prima aman atau nilai p merupakan bilangan prima tidak aman. [4]

1.2. Algoritma ElGamal

Algoritma ElGamal terdiri dari tiga proses, yaitu proses pembentukan kunci, proses enkripsi dan proses dekripsi. Pembentukan kunci terdiri dari kunci *private* (rahasia) dan kunci *public* (umum). Pada proses ini dibutuhkan sebuah bilangan prima *p* yang digunakan untuk membentuk grup Zp*, *elemen primitif* α dan sembarang $a \in \{1,..., p - 2\}$. Kunci publik algoritma ElGamal berupa pasangan 3 bilangan, yaitu (p, α, β), dengan :

$$\beta = \alpha^a \bmod p \quad (2)$$

Algoritma membangkitkan Pasangan Kunci:

a. Penentuan bilangan prima aman yang bernilai besar,
b. Penentuan elemen primitif,
c. Pembentukan kunci berdasarkan bilangan prima aman dan elemen primitive. [4].

Hasil algoritma ini adalah kunci publik: triple (β, α, p) dan kunci privat: pasangan (a, p). Pihak yang membuat kunci publik dan kunci rahasia adalah pengirim, sedangkan pihak penerima hanya mengetahui kunci publik yang diberikan oleh pengirim, dan kunci publik tersebut digunakan untuk mengenkripsi pesan.

Setelah proses pembentukan kunci, maka proses berikutnya adalah proses enkripsi. Proses enkripsi pada algoritma ElGamal dilakukan dengan menghitung:

$$\gamma = \alpha^k \bmod p \text{ dan} \quad (4)$$
$$\delta = \beta^k . m \bmod p \quad (5)$$

dengan $k \in \{1, ..., p\text{-}2\}$ acak rahasia. Algoritma enkripsi pesan:

a. Susun plainteks menjadi blok-blok, m_1, m_2, …, sedemikian sehingga setiap blok merepresentasikan nilai di dalam rentang 0 sampai $p - 1$ (dengan mengubah nilai m menjadi kode ASCII).
b. Pilih bilangan acak k, yang dalam hal ini $0 < k < p - 1$, sedemikian sehingga k relatif prima dengan $p - 1$.
c. Setiap blok m dienkripsi dengan rumus :
   $\gamma = \alpha^k \bmod p$ *dan*
   $\delta = \beta^k m \bmod p$.
   Pasangan γ dan δ adalah cipherteks untuk blok pesan m.

Jika diberikan cipherteks (γ, δ), maka untuk mendekripsi γ dan δ digunakan kunci rahasia a, dan plainteks m diperoleh kembali dengan persamaan :

$$m = \delta / \gamma^a \bmod p. \quad (6)$$
$$m = \delta . (\gamma^{\alpha})^{-1} \bmod p \quad (7)$$

Karena Zp* merupakan grup siklik yang mempunyai order p -1 dan a ϵ {1,..., p - 2}, maka $(\gamma^a)^{-1} = \gamma^{-a} = \gamma^{p-1-a}\ mod\ p$.

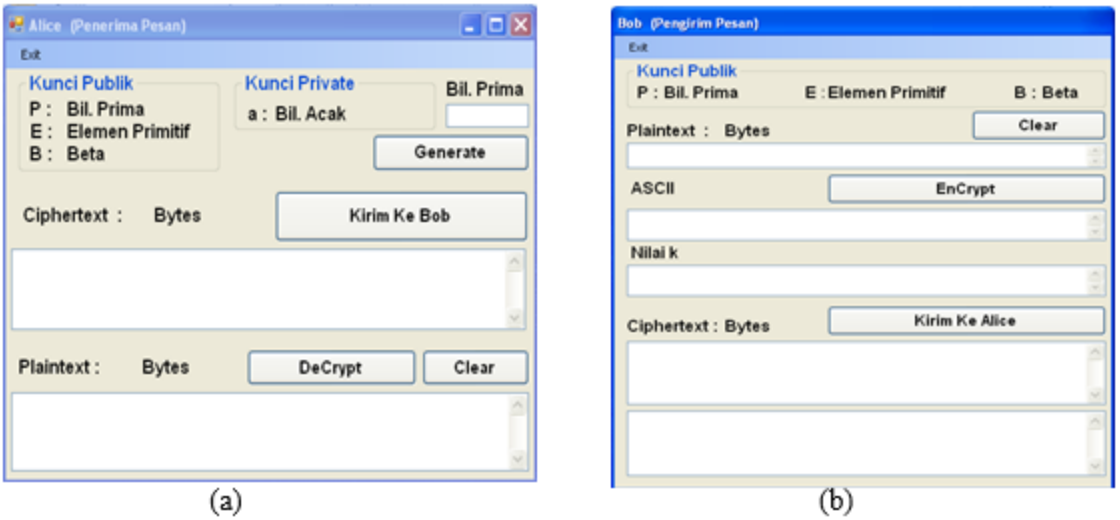
Algoritma Dekripsi pesan:

a. Untuk i dari 1 sampai n :
   Hitung γ^{p-1-a} mod p
   Hitung $m_i = \delta . (\gamma^a)^{-1}$ mod p
b. Diperoleh plainteks m_1 m_2 , ,..., m_n.
c. Konversikan masing-masing bilangan m_1, m_2,..., m_n ke dalam karakter sesuai dengan kode ASCII-nya, kemudian hasilnya digabungkan kembali.

2. METODE PENELITIAN

Penelitian ini bertujuan untuk membandingkan pengaruh digunakannya bilangan prima aman dan bilangan prima tidak aman pada proses pembentukan kunci pada algoritma elgamal. Apakah ada pengaruhnya jika yang digunakan adalah bilangan prima tidak aman pada pembentukan kunci algoritma elgamal, dan bagaimana perbandingannya antara penggunaan bilangan prima aman dengan bilangan prima tidak aman pada proses pembentukan kunci algoritma elgamal.

Untuk mencapai tujuan tersebut, penulis membuat aplikasi enkripsi dan dekripsi pesan dengan menggunakan algoritma elgamal, dimana nilai dari bilangan primanya dapat divariasikan. Pada aplikasi yang dibuat ini terdapat 2 buah *form*, yaitu: *form* pengirim pesan dan *form* penerima pesan. Tampilan *form* pengirim dan penerima dapat dilihat pada gambar 1.

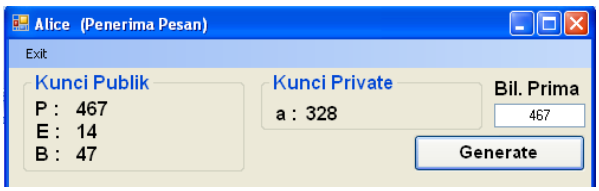


Gambar 1. (a) *Form* penerima dan (b) *form* pengirim aplikasi enkripsi dan dekripsi pesan dengan menggunakan algoritma elgamal

Proses kerja dari aplikasi yang dibuat adalah ketika Bob akan mengirimkan pesan kepada penerima (Alice) secara rahasia, maka penerima pesan (Alice) harus membuat kunci publik dan kunci *private,* kemudian penerima membangkitkan pasangan kuncin *publicn*ya. Kunci *public* inilah yang diberikan penerima pesan (Alice) kepada pengirim pesan (Bob), kunci *private* tetap dipegang oleh penerima pesan. Pengirim pesan menerima kunci *public* dari penerima. Dengan kunci *public* tersebut pengirim pesan mengenkripsi pesan tertentu dengan menggunakan algoritma elgamal untuk dikirimkan ke penerima pesan. Penerima pesan memperoleh pesan yang telah dienkripsi dengan menggunakan algoritma elgamal dari pengirim pesan. Karena penerima pesan (Aice) yang memegang kunci *private,* maka hanya Alice yang dapat mendekrip pesan tersebut agar dapat dibaca.

3. HASIL DAN PEMBAHASAN

Untuk mengetahui apakah aplikasi yang dibuat telah berjalan dengan baik, maka dilakukan beberapa pengujian. Pengujian pertama dilakukan dengan cara memasukkan sebuah bilangan prima aman, selanjutnya membangkitkan pasangan kuncinya. Sebagai contoh bilangan prima aman yang dimasukan adalah 467, hasil pembentukan pasangan kuncinya ditunjukkan pada gambar 2.



Gambar 2. Hasil pembentukan pasangan kunci.

Pada gambar 2 dapat dilihat bahwa hasil pembentukan pasangan kunci yaitu bilangan prima (p) = 467, elemen *primitive* (α) = 14, dan bilangan acak (a) = 328. Hasil ini dapat dibuktikan secara teori sebagai berikut: 467 merupakan bilangan prima aman, dibuktikan oleh persamaan (1).

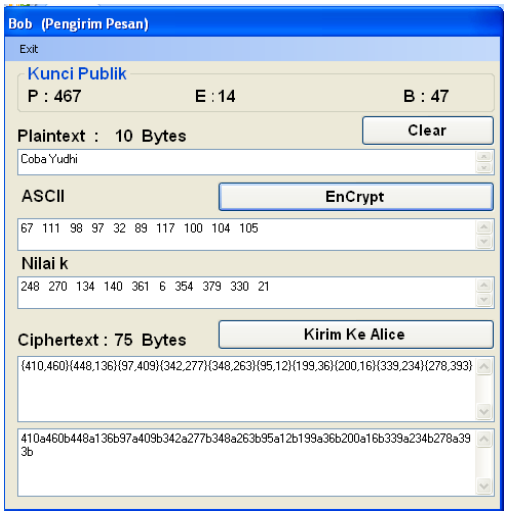
$$p = 2.q + 1$$
$$467 = 2 \times 233 + 1$$

Dari persamaan di atas, 233 merupakan bilangan prima, maka 467 merupakan bilangan prima aman. Kemudian menghitung berdasarkan persamaan (2) sebagai berikut:

$$\beta = \alpha^a \; mod \; p = \; 14^{328} \bmod 467 = 47$$

Diperoleh kunci publik (p , α , β) = (467, 14, 47) dan kunci *private*nya a = 328. Hasil perhitungan secara manual menunjukkan bahwa proses pembentuan kunci telah sesuai secara teori.

Pengujian berikutnya adalah pengujian enkripsi pesan dengan menggunakan kunci *public* yang telah diperoleh. Kunci *public* yang diperoleh adalah (p , α , β) = (467, 14, 47). Dengan kunci *public* tersebut, pengirim pesan akan mengenkripsi pesan "Coba Yudhi" untuk dikirimkan ke penerima pesan. Hasil enkripsi pesan ditunjukkan pada gambar 3.



Gambar 3. Hasil enkripsi pesan

Pada gambar 3 dapat dilihat bahwa hasil enkripsi pesan adalah {410,460} {448,136} {97,409}{342,277}{348,263}{95,12}{199,36}{200,16}{339,234}{278,393}. Hasil ini dapat dibuktikan secara teori sebagai berikut: pertama mengonversi pesan tersebut dalam kode ASCII.

Tabel 1. Konversi karakter ke ASCII

i	Karakter	ASCII
1	C	67
2	o	111
3	b	98
4	a	97
5	<spasi>	32
6	Y	89
7	u	117
8	d	100
9	h	104
10	i	105

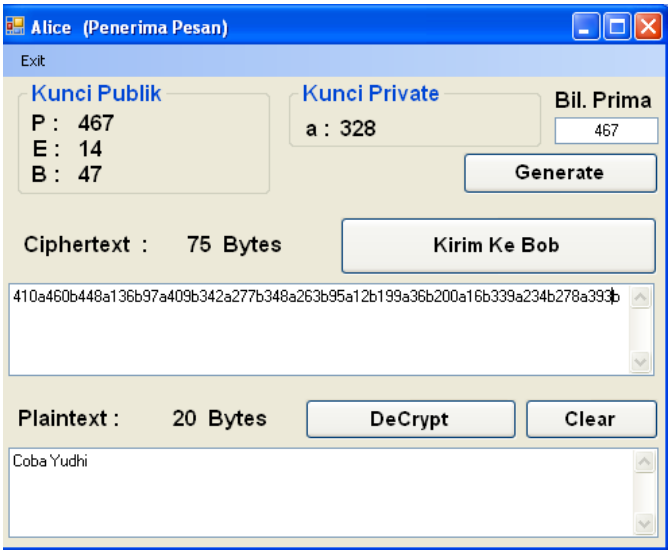
Kemudian pemberi pesan menentukan bilangan acak k (k $\in${0,1,..., 467-1}) untuk setiap *plaintext* m dan mengenkripsi *plaintext* tersebut dengan menghitung nilai γ dan δ dengan "persamaan (4)" dan "persamaan (5)".

Tabel 2. Enkripsi *plaintext* ke *ciphertext* dengan bilangan prima aman

i	m_i	k_i	$\gamma_i = 14^k \ mod\ 467$	$\delta_i = 47^k.m \ mod\ 467$
1	67	248	410	460
2	111	270	448	136
3	98	134	97	409
4	97	140	342	227
5	32	361	348	263
6	89	6	95	12
7	117	354	199	36
8	100	379	200	16
9	104	330	339	243
10	105	21	278	393

Berdasarkan Tabel 2, diperoleh *ciphertext* (γ_i, δ_i) , i = 1,2,…,10, sebagai berikut: {410,460} {448,136} {97,409} {342,277} {348,263} {95,12} {199,36} {200,16} {339,234} {278,393}. Hasil perhitungan secara manual menunjukkan bahwa proses enkripsi pesan telah sesuai secara teori.

Pengujian berikutnya adalah pengujian dekripsi pesan dengan kunci *private* (a = 328) yang dimiliki oleh penerima pesan. Hasil dekripsi pesan ditunjukkan pada gambar 4.



Gambar 3. Hasil dekripsi pesan

Pada gambar 4 dapat dilihat bahwa hasil dekripsi pesan adalah “Coba Yudhi”. Pesan hasil dekripsi sesuai dengan pesan yang dikirim. Hasil ini dapat dibuktikan secara teori dengan menggunakan “persamaan (7)” dilakukan perhitungan untuk tiap blok *ciphertext*.

Tabel 3 Proses Dekripsi *Ciphertext* ke *Plaintext* dengan bilangan prima aman

i	γ_i	δ_i	$m_i = \delta.(\gamma^{\alpha})^{-1} \ mod \ p$	m_i
1	410	460	67	C
2	448	136	111	o
3	97	409	98	b
4	342	227	97	a
5	348	263	32	<spasi>
6	95	12	89	Y
7	199	36	117	u
8	200	16	100	d
9	339	243	104	h
10	278	393	105	i

Berdasarkan perhitungan pada Tabel 3, didapatkan pesan rahasia yang dikirimkan oleh pengirim, yaitu ”Coba Yudhi”. Hasil perhitungan secara manual menunjukkan bahwa proses dekripsi pesan telah sesuai secara teori.

Berikutnya dilakukan pengujian enkripsi dan dekripsi menggunakan variasi bilangan prima aman dengan *plaintext* yang sama yaitu “Coba Yudhi”. Dari pengujian yang dilakukan didapatkan hasil sebagaimana terlihat pada tabel 4.

Tabel 4 Hasil enkripsi dan dekripsi dengan variasi bilangan prima aman.

No	(p , α , β)	a	Hasil enkripsi	Kapasitas *ciphertext*	Hasil Dekripsi
1	(347,17,159)	244	{27,126}{263,337}{144,178}{4,187}{287,155}{241,73}{322,120}{77,33}{274,6}{86,202}	71 byte	Coba Yudhi (berhasil)
2	(467,14,47)	328	{410,460}{448,136}{97,409}{342,277}{348,263}{95,12}{199,36}{200,16}{339,234}{278,393}	75 byte	Coba Yudhi (berhasil)
3	(719,22,310)	506	{280,407}{4,384}{523,461}{700,430}{374,163}{435,710}{24,600}{343,191}{670,52}{281,565}	76 byte	Coba Yudhi (berhasil)
4	(839,22,611)	591	{563,376}{587,807}{224,381}{232,25}{290,87}{31,448}{89,560}{207,697}{322,775}{694,754}	76 byte	Coba Yudhi (berhasil)
5	(983,15,55)	692	{887,851}{312,336}{910,536}{851,701}{809,195}{862,25}{497,394}{508,333}{519,918}{12,508}	78 byte	Coba Yudhi (berhasil)
6	(1019,13,135)	718	{218,624}{752,930}{419,566}{872,621}{243,595}{905,814}{561,212}{357,73}{66,36}{340,198}	77 byte	Coba Yudhi (berhasil)
7	(1283,18,224)	904	{699,828}{725,398}{696,795}{631,385}{1174,516}{691,149}{1208,1069}{510,867}{1083,1072}{323,1044}	86 byte	Coba Yudhi (berhasil)
8	(1523,13,540)	1073	{1256,324}{16,358}{234,1434}{902,1036}{433,1216}{313,960}{520,740}{285,643}{953,406}{873,1394}	84 byte	Coba Yudhi (berhasil)
9	(2027,15,699)	1429	{1679,1678}{1078,237}{1766,1281}{1287,659}{1806,851}{762,839}{1774,1255}{344,194}{961,288}{291,1809}	90 byte	Coba Yudhi (berhasil)
10	(8039,26,297)	5671	{1541,2123}{240,350}{5489,5423}{7614,352}{1376,3817}{2514,6025}{1927,3134}{6770,3632}{3898,4766}{2191,4977}	97 byte	Coba Yudhi (berhasil)

Dari tabel 4 dapat dilihat bahwa proses dekripsi untuk bilangan prima aman semuanya berhasil dilakukan dengan baik. Dari tabel 4 juga dapat dilihat bahwa semakin besar bilangan prima aman yang digunakan, maka hasil *ciphertext* juga cenderung semakin besar.

Berikutnya dilakukan pengujian enkripsi dan dekripsi menggunakan variasi bilangan prima tidak aman dengan *plaintext* yang sama yaitu “Coba Yudhi”. Dari pengujian yang dilakukan didapatkan hasil sebagaimana terlihat pada tabel 5.

Tabel 5 Hasil enkripsi dan dekripsi dengan variasi bilangan prima tidak aman.

No	(p , α , β)	a	Hasil enkripsi	Kapasitas *ciphertext*	Hasil Dekripsi
1	(307,23,292)	215	{238,256}{204,239}{71,222}{285,189}{271,253}{164,160}{181,200}{231,283}	61 byte	Coba Yudhi (berhasil)
2	(401,15,47)	282	{211,182}{388,224}{68,379}{329,395}{233,69}{282,14}{25,271}{254,89}{304,378}{247,68}	74 byte	Coba Yudhi (berhasil)
3	(701,3,303)	603	{353,83}{53,572}{162,122}{216,7}{126,159}{551,298}{493,315}{160,346}{230,438}{14,43}	74 byte	Coba Yudhi (berhasil)
4	(809,3,346)	478	{441,287}{625,52}{333,277}{278,173}{351,676}{616,508}{405,190}{5,191}{398,256}{191,433}	77 byte	Coba Yudhi (berhasil)
5	(907,3,741)	825	{367,92}{518,343}{500,453}{653,185}{228,746}{376,528}{605,573}{10,51}{358,83}{349,126}	76 byte	Coba Yudhi (berhasil)
6	(1009,26,44)	711	{936,752}{418,784}{986,844}{382,33}{666,997}{972,794}{991,657}{466,714}{450,212}{905,488}	78 byte	Coba Yudhi (berhasil)
7	(1201,26,1115)	846	{36,295}{947,728}{681,826}{506,766}{444,881}{335,1142}{489,415}{594,890}{387,909}{572,150}	80 byte	Coba Yudhi (berhasil)
8	(1511,33,471)	1065	{708,676}{487,1019}{1283,574}{346,1085}{370,708}{536,668}{728,1233}{1263,1175}{1020,293}{1313,806}	88 byte	Coba Yudhi (berhasil)
9	(2003,18,1706)	1412	{376,347}{915,1261}{272,1334}{1799,1859}{1795,1477}{1635,1007}{1137,1616}{1850,1007}{17,396}{1351,992}	92 byte	Coba Yudhi (berhasil)
10	(8009,27,2071)	5650	{6858,3845}{6749,6229}{1282,7041}{709,4812}{2183,3954}{4534,7268}{7596,4861}{5558,5957}{7523,490}{1516,1564}	98 byte	Coba Yudhi (berhasil)

Dari tabel 5 dapat dilihat bahwa proses dekripsi untuk bilangan prima tidak aman semuanya berhasil dilakukan dengan baik. Ini menunjukkan bahwa dengan menggunakan bilangan prima aman maupun bilangan prima tidak aman, proses pembentukan kunci, enkripsi dan dekripsi tetap dapat berjalan dengan baik. Dari tabel 5 juga dapat dilihat bahwa semakin besar bilangan prima tdak aman yang digunakan, maka hasil *ciphertext* juga cenderung semakin besar.

4. KESIMPULAN

Dari hasil dan pembahasan dapat disimpulkan beberapa hal sebagai berikut:

1. Pada algoritma elgamal, kunci yang dibentuk dari bilangan prima tidak aman, tidak mempengaruhi proses enkripsi dan dekripsinya. Hasil enkripsinya tetap dapat didekripsikan dengan baik.
2. Pada Algorima elgamal dengan menggunakan bilangan prima aman maupun bilangan prima tidak aman, proses pembentukan kunci, enkripsi dan dekripsi tetap dapat berjalan dengan baik.
3. Semakin besar nilai bilangan prima yang digunakan, maka kapasitas *ciphertext* juga cenderung semakin besar.

5. SARAN

Pada penelitian ini pesan yang dienkripsikan dengan algoritma elgamal adalah pesan dalam bentuk teks, pada penelitian berikutnya pesan yang digunakan dapat dikembangkan dengan berbagai jenis *file*.

DAFTAR PUSTAKA

[1] Aribowo, E., 2008, Aplikasi Pengamanan Data Office dengan Algoritma Kriptografi Kunci Asimetris Elgamal, *Jurnal Informatika*, Vol 2, No.2, hal 209-219.

[2] Andrian, Y., 2013, Analisis Penggunaan Elemen Primitif dan Non Primitif pada Algoritma Elgamal, *Prosiding Konferensi Nasional Sistem Informasi*, Mataram, 14-16 Februari 2013.

[3] Massandi, D. T., 2009, Algoritma Elgamal Dalam Pengamanan Pesan Rahasia, *Jurnal Informatika*.

[4] Ifanto, M., 2009, Metode Enkripsi dan Dekripsi dengan Menggunakan Algoritma Elgamal, *Makalah IF2091 Struktur Diskrit Tahun 2009*.

[5] Taufiq, M., Dwiono, W., Hartanto, T., 2010, Penerapan Algoritma Kriptografi ElGamal untuk Pengaman File Citra, *EECCIS*, Vol IV, No 1, hal 8-11.

[6] Widyartono, A., 2011, Algoritma Elgamal Untuk Enkripsi Data Menggunakan Gnupg, *Jurnal Teknologi dan Informatika (Teknomatika)*, Vol 1, No 1, hal 29-35.

[7] Ummi, K., 2013, Analisis Penggunaan Bilangan Prima Aman Besar Pada Algoritma Elgamal, *Prosiding Konferensi Nasional Sistem Informasi*, Mataram, 14-16 Februari 2013.