Pembangkitan Kunci pada Algoritma Asimetris ElGamal untuk Meningkatkan Keamanan Data bertipe .docx

Key Generation on ElGamal Asymmetric Algorithm To Enhance .docx Format Data Security

Aisyatul Karima[1], Ari Saputro[2]
[1,2]Teknik Informatika, Fakultas Ilmu Komputer Universitas Dian Nuswantoro ;
Jl. Imam Bonjol No. 207 Semarang, telp/fax Telp. (024) 3517261
[1]aisyatul.karima@gmail.com, [2]arisaputro93@gmail.com

Abstrak

Pencurian data melalui plagiasi dan modifikasi dokumen .docx sering terjadi dalam kehidupan digital. Hal ini dikarenakan dokumen tersebut mudah dimodifikasi oleh siapapun. Oleh karena itu dibutuhkan sebuah keamanan berupa penyandian data untuk mengamankan dokumen .docx yang bersifat rahasia dari pihak yang tidak bertanggung jawab. Teknik algoritma pengkodean yang digunakan yaitu algoritma ElGamal yang bersifat asimetris. Adapun tahapan metode penelitian terbagi dalam proses pembangkitan kunci, enkripsi dan dekripsi. Metode ini menggunakan pembangkitan kunci untuk menghasilkan kombinasi kunci yang variatif. Kunci yang dihasilkan berupa kunci privat, kunci public, bilangan prima serta bilangan acak. Berdasarkan hasil percobaan, proses pembangkitan kunci pada algoritma ElGamal mampu mengacak isi pesan berekstensi .docx secara aman. Hal ini terbukti dengan percobaan menggunakan beberapa kombinasi bilangan prima dan kunci private terhadap data yang sudah dienkripsi, kunci belum mampu terpecahkan. Pengujian penyerangan dilakukan dalam waktu lebih dari 15 jam, namun algoritma ElGamal belum mampu terpecahkan. Selain itu, sebagai dampak dari pengujian terhadap file, menghasilkan perbedaan ukuran file asli dengan file yang telah dienkripsi rata-rata 7 kali lipat. Perubahan ukuran tersebut tidak berpengaruh, sebab file akan kembali ke ukuran semula setelah proses dekripsi. Dari segi keamanan, penggunaan algoritma ElGamal ini dinyatakan cukup kuat sebagai alternatif untuk membantu meningkatkan keamanan data .docx.

Kata kunci— Pembangkitan kunci,Kunci Asimetris, Algoritma ElGamal, Keamanan data, .docx

Abstract

Thieving data through the plagiarism and modification of .docx have many occurred in digital era because of their easiness in modifying. Therefore, data encryption needs to safe the secure .docx document from third party. Encryption algorithm that used is ElGamal asymmetric algorithm. The stages of research are key generation processes, encryption and decryption. This method uses the key generation process to produce the varied key combination. Those keys are private key, public key, prime number and random number. Based on the experimental result, the key generation process on ElGamal algorithm can scramble the .docx plaintext safely. The result shows that the password is unbreakable when use the prime number and private key combination to encrypt the data. The experiment needs more than 15 hours to try break the password, but ElGamal algorithm is still unbreakable. Another impact in the size of encryption data changes 7 times from the original message, but it doesn't involve the message because the file back to the normal size after decryption process. Based on the experimental result, the ElGamal algorithm is strong to enhance .docx Format Data Security.

Keywords— Key generation process, Assymetric key, ElGamal algorithm, Data security, .docx

1. PENDAHULUAN

Saat ini pencurian data melalui plagiasi dan modifikasi sering kita temui dalam kehidupan digital. Dokumen yang banyak digunakan adalah dokumen dengan ekstensi *docx.* Hal ini dikarenakan dokumen dengan ekstensi *.docx* merupakan salah satu dokumen yang mudah dalam

proses pembuatan serta penyimpanannya. Untuk menjaga kerahasiaan, integritas, pengenalan identitas pengirim dan pencegahan penyangkalan pengiriman data dokumen, maka diperlukan sebuah alat bantu untuk melindungi dokumen tersebut. Hal ini menjadi masalah utama dalam persaingan dunia bisnis. Oleh karena itu untuk menjaga integritas data tersebut dibutuhkan sebuah sistem keamanan berupa penyandian atau pengkodean data sebelum proses pengiriman dilakukan, yang bertujuan untuk mengamankan data penting yang bersifat rahasia agar tidak dengan mudah dibaca dan diubah isi dari pesan tersebut oleh pihak yang tidak berkepentingan.

Dalam pengamanan data terdapat beberapa teknik, salah satu diantaranya adalah algoritma pengkodean atau penyandian data. Adapun teknik algoritma pengkodean atau penyandian data yang sering digunakan adalah kriptografi yang terbagi menjadi dua tipe algoritma, baik simetris maupun asimetris. Dalam paper ini, mengusulkan pengamanan data menggunakan algoritma asimetris berupa algoritma *ElGamal.* Algoritma asimetris ini memiliki kunci yang berbeda saat proses enkripsi dan dekripsinya, yaitu kunci publik untuk enkripsi dan kunci pribadi untuk dekripsi [HYPERLINK \l "Sin12" 1]. Kunci yang didistribusikan dalam algoritma asimetris adalah kunci publik yang tidak diperlukan kerahasiaanya sedangkan kunci pribadi tetap dijaga dan disimpan kerahasiannya atau tidak didistribusikan. Setiap orang yang mempunyai kunci publik bisa melakukan enkripsi data tetapi hasil dari enkripsi tersebut hanya bisa dibaca oleh orang yang memiliki kunci pribadi.

Tingkat keamanan algoritma *Elgamal* ini terletak pada kesulitan dalam menghitung logaritma diskrit2]**}**. Kelebihan dari algoritma ini adalah pembangkitan kunci yang menggunakan logaritma diskrit dan metode enkripsi dekripsi yang menggunakan proses komputasi yang besar sehingga hasil enkripsinya berukuran dua kali dari ukuran semula. Proses enkripsi pada plaintext yang sama diperoleh ciphertext yang berbeda-beda, sedangkan untuk proses dekripsi diperoleh *plaintext* yang sama [HYPERLINK \l "Dan09" 3].

Penelitian terkait menyebutkan bahwa kriptografi kunci publik *ElGamal* dapat digunakan untuk mengamankan data karena algoritma *ElGamal* dalam pembentukan salah satu kuncinya menggunakan bilangan prima dan menitikberatkan kekuatan kuncinya pada pemecahan masalah logaritma diskrit sehingga keamanan kuncinya lebih terjamin5]**}**.

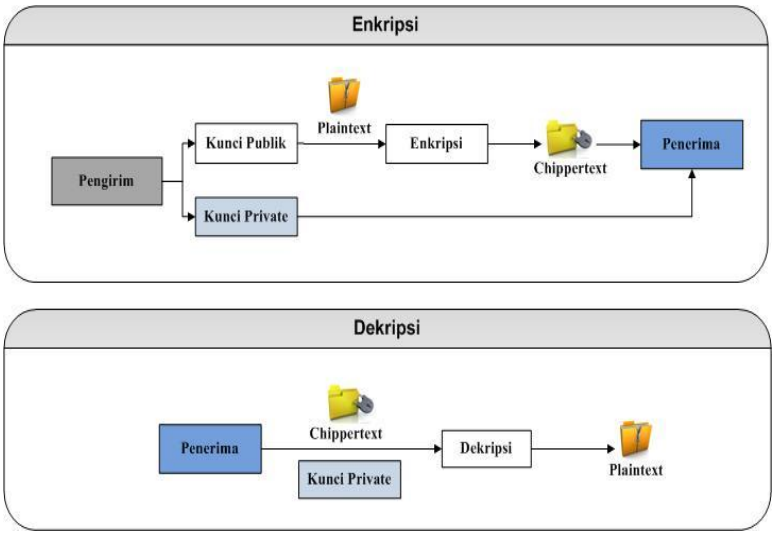
Perubahan besarnya data dari *plaintext* yang melalui proses enkripsi dan menjadi *ciphertext* adalah 554,9545 %. Waktu yang dibutuhkan untuk sebuah sistem untuk melakukan proses enkripsi dan dekripsi sistem kriptografi *ElGamal* hingga 400 karakter adalah 0.004503 detik untuk proses enkripsi dan0.00479 detik untuk proses dekripsi [HYPERLINK \l "Suc13" 7].

Berdasarkan Penyisipan Menggunakan Fungsi Linier, algoritma ElGamal memiliki kelebihan yang terletak pada tingkat kesulitan perhitungan logaritma diskrit pada modulo ketika bilangan yang dipilih adalah bilangan prima yang besar, sehingga upaya untuk menyelesaikan masalah logaritma ini menjadi sulit untuk dipecahkan 8]**}**. Algoritma *ElGamal* ditemukan oleh Taher ElGamal pada tahun 1985. Algoritma ini pada umumnya digunakan untuk *digital signature*, namun kemudian dimodifikasi sehingga bisa digunakan untuk enkripsi dan dekripsi [9].

Kelebihan algoritma *ElGamal* lainnya adalah pada proses pembangkitan kunci yang menggunakan logaritma diskrit dan metode enkripsi dekripsi yang menggunakan proses komputasi yang besar sehingga hasil enkripsinya berukuran dua kali dari ukuran semula [10]. Namun di sisi lain, algoritma *ElGamal* juga mempunyai kekurangan yaitu membutuhkan *resource* yang besar dan processor yang mampu melakukan perhitungan besar. Meskipun memiliki kelemahan tersebut, namun algoritma *ElGamal* memiliki kelebihan yang jauh lebih banyak, sehingga dalam paper ini menggunakan algoritma *ElGamal* dalam meningkatkan keamanan data.

2. METODE PENELITIAN

Dalam penelitian ini metode yang digunakan untuk menyelesaian permasalahan yang ada adalah menggunakan model seperti pada gambar 1 berikut :



Gambar 1. Metode Penelitian

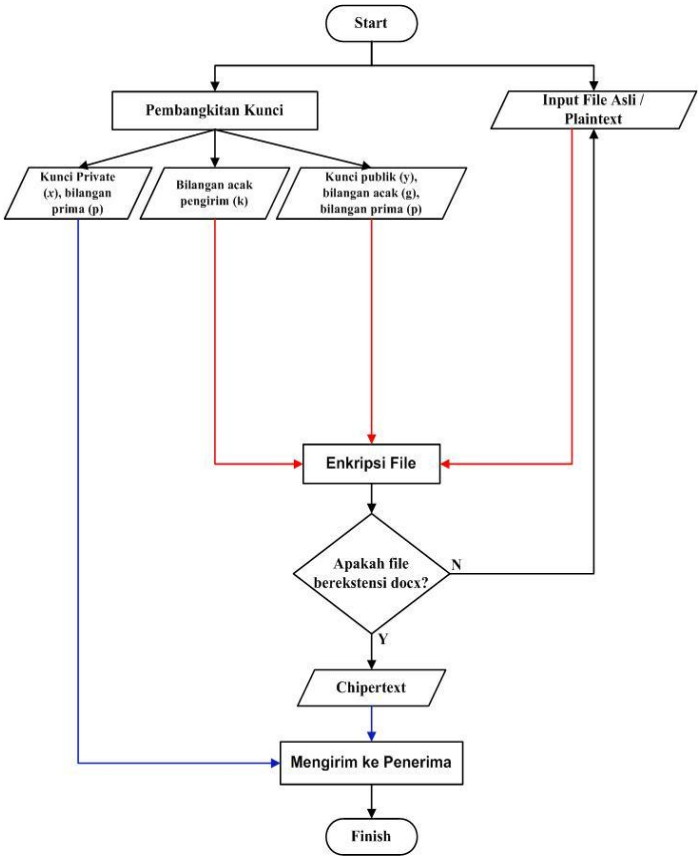
Berdasarkan gambar 1 di atas menjelaskan bahwa dalam paper ini menggunakan satu teknik algoritma kriptografi yaitu algoritma *ElGamal* yang digunakan sebagai proses enkripsi dekripsi. Adapun rincian proses enkripsi dan dekripsi sebagai berikut.

a. Langkah enkripsi
   1. Langkah awal adalah pengirim melakukan pembangkitan kunci yang akan menghasilkan kunci publik dan kunci *private.*
   2. Pengirim menggunakan kunci publik dan memasukkan data awal (*plaintext*) untuk melakukan Enkripsi file dengan menggunakan algoritma ElGamal dan hasil dari keluaran adalah *ciphertext.*
   3. Kemudian hasil dari proses enkripsi (*ciphertext*) beserta kunci private tersebut akan dikirimkan ke penerima pesan.

b. Langkah dekripsi
   1. Untuk dapat melihat atau membuka informasi dari isi pesan tersebut, penerima akan melakukan proses dekripsi file dengan langkah awal memasukkan kunci private dan *ciphertext* yang diterima, kemudian dilakukan proses dekripsi dengan menggunakan algoritma ElGamal.
   2. Hasil dari proses dekripsi tersebut adalah *plaintext* yang merupakan file asli dan isi dari pesan tersebut langsung diterima oleh penerima pesan.

3. HASIL DAN PEMBAHASAN

Proses enkripsi diawali dengan pembangkitan kunci secara acak. Adapun untuk tahapan proses enkripsi dapat dilihat pada gambar 2 berikut.



Gambar 2. Flowchart enkripsi *ElGamal*

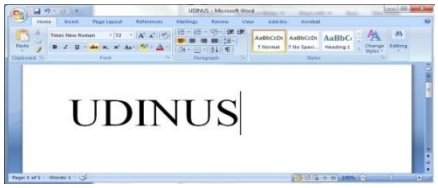
Adapun proses enkripsi diawali dengan pembangkitan kunci secara acak yang menghasilkan beberapa bilangan diantaranya bilangan prima, bilangan acak pengirim, kunci *public* serta kunci *private*. Berikut merupakan algoritma pembangkitan kunci.

a. Pilih sembarang bilangan prima p (p dapat di-share di antara anggota kelompok)
b. Pilih dua buah bilangan acak, g dan x, dengan syarat $g<p$ dan $1 \leq x \leq p-2$
c. Hitung $y = g^x \bmod p$. Hasil dari algoritma ini:
   - Kunci publik: tripel (y, g, p)
   - Kunci *privat*: pasangan (x, p)

Selanjutnya, proses enkripsi *ElGamal* dilaksanakan menggunakan bilangan acak dan kunci *public* disertai dengan bilangan prima yang sudah ditentukan sebelumnya. Berikut merupakan algoritma *ElGamal* :

a. Susun *plaintext* menjadi blok-blok m1, m2, …, (nilai setiap blok di dalam selang [0, p –1].
b. Pilih bilangan acak k, yang dalam hal ini $1 \leq k \leq p-2$.
c. Setiap blok m dienkripsi dengan rumus
   $a = g^k \bmod p$; $b = y^k m \bmod p$
   Pasangan a dan b adalah ciphertext untuk blok pesan m. Jadi, ukuran *ciphertext* dua kali ukuran plaintextnya.

Sedangkan penjelasan lebih lanjut tahapan proses enkripsi dapat dilihat pada gambar 3. Berikut merupakan proses enkripsi file .docx dengan *plaintext* UDINUS. Uji coba dilakukan pada file “UDINUS.docx” .



Gambar 3. Tampilan file asli

Untuk proses pembangkitan kunci dapat dilihat pada gambar 4 berikut :

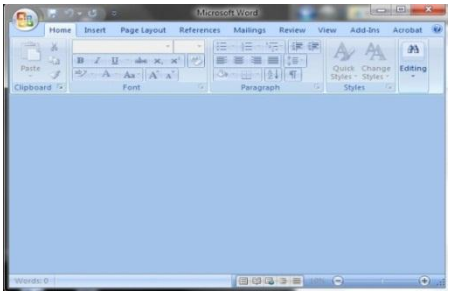
Gambar 4. Pembangkitan kunci

Proses selanjutnya dilanjutkan dengan input file asli *(plaintext)* yang akan dienkripsi sesuai pada gambar 5.

Gambar 5. Input file asli dan enkripsi

Tahapan selanjutnya adalah mengunduh file hasil enkripsi yang telah disimpan sebelumnya dengan nama file HasilEnkripsi.docx sesuai pada gambar 6. Adapun tampilan *plaintext* yang sudah dienkripsi *(ciphertext)* seperti tampak pada gambar 7.

Gambar 6. Download file hasil enkripsi



Gambar 7. Hasil file asli yang sudah dienkripsi(ChipperText).

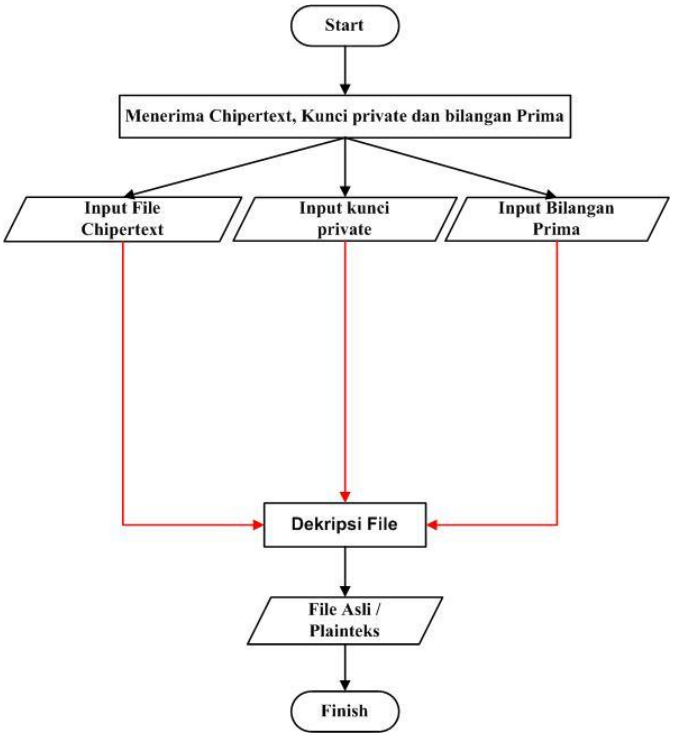
Sistem ini hanya memproses *plaintext* yang berasal dari dokumen file yang berekstensi .docx. Adapun hasil akhir enkripsi akan menghasilkan *ciphertext* yang akan diubah kembali menjadi karakter sesuai dengan tabel *ASCII* yang akan diteruskan ke sisi penerima disertai dengan kunci *private* dan bilangan prima untuk melakukan proses dekripsi.

Untuk melakukan proses dekripsi dapat dilihat pada gambar 8 yang diawali dengan proses penginputan file hasil penyandian *(ciphertext)*, kunci *private* serta bilangan prima. Dengan beberapa variabel input tersebut, algoritma *ElGamal* akan memproses dekripsi file dengan sendirinya. Berikut merupakan algoritma dekripsi *ElGamal* :

a. Gunakan kunci *private* x untuk menghitung $(a^x)^{-1} = a^{p-1-x} \bmod p$
b. Hitung *plaintext* m dengan persamaan :
   $m = b/a^x \bmod p \quad = b(a^x)^{-1} \bmod p$

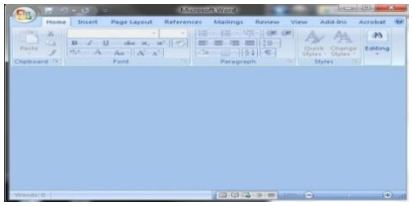
Untuk proses dekripsi, algoritma ini membutuhkan waktu yang lebih lama karena kompleksitas proses dekripsinya yang rumit. Dibutuhkan dua kali komputasi karena ukuran *ciphertext* yang lebih besar dibandingkan *plaintext* nya.

a. Masalah logaritma diskrit: Jika p adalah bilangan prima dan g dan y adalah sembarang bilangan bulat. carilah x sedemikian sehingga : gx = y (mod p)
b. Besar-besaran dalam algoritma *ElGamal* adalah sebagai berikut :
   1. Bilangan prima, *p* (tidak rahasia).
   2. Bilangan acak, *g (g<p)* (tidak rahasia).
   3. Bilangan acak, *x (x<p)* (rahasia, kunci privat).
   4. $y = g^x \bmod p$ (tidak rahasia, kunci publik).
   5. *m (plaintext)* (rahasia) Jika plaintext tersebut berbentuk karakter maka diubah kedalam bentuk ASCII.
   6. *a* dan *b (ciphertext)* (tidak rahasia).



Gambar 8. Flowchart dekripsi ElGamal

Pada gambar 9 merupakan proses dekripsi file .docx dengan *plaintext* yang ada pada file "HasilEnkrip.docx"



Gambar 9. Hasil dari proses enkripsi sebelumnya *(ciphertext)*

Proses dekripsi diawali dengan input kunci *private* meliputi bilangan prima serta input file *ciphertext* seperti pada gambar 10.

Gambar 10. Proses input kunci *private*, bilangan prima dan *ciphertext*

Langkah selanjutnya adalah dengan mengunduh file hasil dekripsi seperti pada gambar 11, kemudian dilanjutkan dengan membuka file hasil dekripsi yang berupa pesan asli (paintext) seperti tampak pada gambar 12.

Hasil dekripsi berupa file yang akan diubah kembali ke dalam bentuk karakter (teks) dimana sebelumnya teks tersebut sudah diubah menjadi karakter *ASCII.* Melalui proses dekripsi ini akan menghasikan file asli *(plaintext)* yang merupakan file berekstensi *.docx.*

Gambar 11. *Download* hasil dekripsi file

Gambar 12. File teks hasil dekripsi

Dari beberapa percobaan yang sudah dilakukan, berikut diuraikan simulasi perhitungan manual proses enkripsi algoritma *ElGamal* dengan rincian sebagai berikut :

Plaintext (*m*) : UDINUS
Bilangan acak prima (*p*) : 1031
Bilangan acak (*g*) : 514
Kunci private (*x*) : 416
Kunci public (*y*) : $y = g^x \bmod p$
$y = 514^{416} \bmod 1031$
$y = 585$
Bilangan acak pengirim (*k*) : k = 762

3.1 Proses Enkripsi

Sebelum melakukan proses enkripsi, file asli (*plaintext*) harus diubah kedalam nilai decimal ASCII terlebih dahulu, dengan U = 85, D = 68, I = 73, N = 78, U = 85, S = 83. Adapun simulasi perhitungan enkripsi algoritma *ElGamal* seperti pada tabel 1 berikut ini.

Tabel 1. Simulasi Perhitungan Enkripsi Algoritma *ElGamal* Manual

No.	Karakter ASCII	Rumus Enkripsi		Hasil	
		$a = g^k \bmod p$	$b = y^k m \bmod p$	a	b
1	U	$a = g^k \bmod p$ $= 514^{762} \bmod 1031$ $= 1000$	$b = y^k m \bmod p$ $= 585^{762}.85 \bmod 1031$ $= 1007$	*1000*	*1007*
2	D	$a = g^k \bmod p$ $= 514^{762} \bmod 1031$ $= 1000$	$b = y^k m \bmod p$ $= 585^{762}.68 \bmod 1031$ $= 187$	*1000*	*187*
3	I	$a = g^k \bmod p$ $= 514^{762} \bmod 1031$ $= 1000$	$b = y^k m \bmod p$ $= 585^{762}.73 \bmod 1031$ $= 974$	*1000*	*974*
4	N	$a = g^k \bmod p$ $= 514^{762} \bmod 1031$ $= 1000$	$b = y^k m \bmod p$ $= 585^{762}.78 \bmod 1031$ $= 730$	*1000*	*730*
5	U	$a = g^k \bmod p$ $= 514^{762} \bmod 1031$ $= 1000$	$b = y^k m \bmod p$ $= 585^{762}.85 \bmod 1031$ $= 1007$	*1000*	*1007*
6	S	$a = g^k \bmod p$ $= 514^{762} \bmod 1031$	$b = y^k m \bmod p$ $= 585^{762}.83 \bmod 1031$	*1000*	*486*

= 1000	*= 486*

Nilai karakter tersebut merupakan hasil proses enkripsi yang kemudian akan digunakan dalam proses dekripsi file.

3.2 Proses Dekripsi

Adapun proses perhitungan nilai file hasil enkripsi *ciphertext* menjadi file asli *plaintext* (m) sesuai pada table 2 berikut ini.

Tabel 2. Simulasi Perhitungan Dekripsi Algoritma *ElGamal* Manual

No.	Plaintext (m)	Rumus Dekripsi $(b.a^{p-1-x} \bmod p)$	Hasil	Konversi Karakter ASCII
1	m1	m1 = $b.a^{p-1-x} \bmod p$ = $1007.1000^{1031-1-3} \bmod 1031$ = $1007.1000^{1027} \bmod 1031$ = *85*	85	U
2	m2	m1 = $b.a^{p-1-x} \bmod p$ = $187.1000^{1031-1-3} \bmod 1031$ = $187.1000^{1027} \bmod 1031$ = *68*	68	D
3	m3	m1 = $b.a^{p-1-x} \bmod p$ = $974.1000^{1031-1-3} \bmod 1031$ = $974.1000^{1027} \bmod 1031$ = *73*	73	I
4	m4	m1 = $b.a^{p-1-x} \bmod p$ = $730.1000^{1031-1-3} \bmod 1031$ = $730.1000^{1027} \bmod 1031$ = *78*	78	N
5	m5	m1 = $b.a^{p-1-x} \bmod p$ = $1007.1000^{1031-1-3} \bmod 1031$ = $1007.1000^{1027} \bmod 1031$ = *85*	85	U
6	m6	m1 = $b.a^{p-1-x} \bmod p$ = $486.1000^{1031-1-3} \bmod 1031$ = $486.1000^{1027} \bmod 1031$ = *83*	83	S

Setelah melalui proses pembuktian kebenaran hasil dari perhitungan dekripsi yang diubah menjadi teks kembali dengan menggunakan *ASCII*, maka hasil akhir dekripsi berupa hasil perubahan nilai decimal *ASCII* kedalam karakter yang berupa U D I N U S.

3.3 Pengujian hasil

Pengujian dalam penelitian ini menggunakan metode *Brute force attack*. Cara kerja metode ini sangat sederhana yaitu mencoba semua kombinasi yang memungkinkan memecahkan tembok keamanan. Pengujian dilaksanakan dengan menggunakan *password* hasil acak dari pembangkitan kunci dan disimulasikan *password* yang digunakan untuk mengamankan sebuah file word berekstensi *.docx* dimana file tersebut akan diserang menggunakan aplikasi *infixi word password recovery* versi 1.0. untuk menguji ketahanannya. Hasil pengujian *brute force attack* dapat dilihat pada tabel 3 berikut ini.

Tabel 3. Pengujian brute force attack

No	Password	Lama password diketahui	No.	Password	Lama password diketahui
1	Bilangan prima : 859 Kunci private : 252	1 jam 31 detik masih belum bisa terpecahkan	16	Bilangan prima : 1061 Kunci private : 100	6 jam 33 menit 12 detik belum bisa terpecahkan
2	Bilangan prima : 937 Kunci private : 759	15 jam 15 menit 42 detik belum bisa terpecahkan	17	Bilangan prima : 1409 Kunci private : 487	6 jam 32 menit 19 detik belum bisa terpecahkan
3	Bilangan prima : 977 Kunci private : 208	2 jam 55 menit 55 detik belum bisa terpecahkan	18	Bilangan prima : 659 Kunci private : 658	6 jam 31 menit 12 detik masih belum bisa terpecahkan
4	Bilangan prima : 373 Kunci private : 272	4 jam 1 menit 53 detik masih belum bisa terpecahkan	19	Bilangan prima : 431 Kunci private : 103	2 jam 57 menit 39 detik masih belum bisa terpecahkan
5	Bilangan prima : 1229 Kunci private : 408	7 jam 15 menit 5 detik masih belum bisa terpecahkan	20	Bilangan prima : 1627 Kunci private : 1366	3 jam 2 menit 47 detik masih belum bisa terpecahkan
6	Bilangan prima : 2061 Kunci private : 1216	10 jam 6 menit 27 detik belum bisa terpecahkan	21	Bilangan prima : 1019 Kunci private : 544	3 jam 23 menit 43 detik belum bisa terpecahkan
7	Bilangan prima : 653 Kunci private : 343	10 jam 1 menit 55 detik belum bisa terpecahkan	22	Bilangan prima : 1009 Kunci private : 893	4 jam 11 menit 30 detik masih belum bisa terpecahkan
8	Bilangan prima : 653 Kunci private : 343	9 jam 57 menit 34 detik masih belum bisa terpecahkan	23	Bilangan prima : 607 Kunci private : 691	11 jam 52 menit 22 detik masih belum bisa terpecahkan
9	Bilangan prima : 262 Kunci private : 619	6 jam 11 menit 53 detik masih belum bisa terpecahkan	24	Bilangan prima : 133 Kunci private : 449	4 jam 19 menit 33 detik belum bisa terpecahkan
10	Bilangan prima : 620 Kunci private : 709	5 jam 49 menit 39 detik belum bisa terpecahkan	25	Bilangan prima : 389 Kunci private : 116	9 jam 59 menit 3 detik belum bisa terpecahkan
11	Bilangan prima : 2129 Kunci private : 1826	2 jam 29 menit 27 detik belum bisa terpecahkan	26	Bilangan prima : 1567 Kunci private : 397	10 jam 11 menit 48 detik belum bisa terpecahkan
12	Bilangan prima : 1979 Kunci private : 524	2 jam 30 menit 4 detik masih belum bisa terpecahkan	27	Bilangan prima : 709 Kunci private : 474	8 jam 44 menit 4 detik belum bisa terpecahkan
13	Bilangan prima : 1381 Kunci private : 685	2 jam 28 menit 20 detik masih belum bisa terpecahkan	28	Bilangan prima : 331 Kunci private : 115	1 jam 14 menit 17 detik masih belum bisa terpecahkan
14	Bilangan prima : 409 Kunci private : 353	6 jam 35 menit 35 detik belum bisa terpecahkan	29	Bilangan prima : 569 Kunci private : 195	1 jam 15 menit 22 detik masih belum bisa terpecahkan
15	Bilangan prima : 2411 Kunci private : 1533	6 jam 34 menit 2 detik belum bisa terpecahkan	30	Bilangan prima : 569 Kunci private : 547	1 jam 16 menit 8 detik belum bisa terpecahkan

Berdasarkan tabel 3 di atas dijelaskan bahwa dibutuhkan waktu tertentu untuk menunggu hasil dari *brute force attack.* Jika algoritma yang digunakan tersebut lemah maka semakin cepat waktu yang dibutuhkan untuk menemukan dan mengetahui kunci (password) dari file data enkripsi tersebut. Sebaliknya, jika algoritma yang digunakan rumit dan kuat maka semakin lama waktu yang diperlukan untuk mengetahui atau menyerang sebuah file data yang sudah terenkripsi tersebut. Pada waktu tertentu saat proses penyerangan dilakukan dan diketahui

jumlah karakter kunci (password) sudah terlewat atau melebihi dari panjangnya karakter kunci yang dibuat oleh pengirim, kemungkinan besar *bruteforce attack* tersebut gagal melakukan serangan. Dari tabel 3 nampak jelas bahwa setelah dilakukan beberapa kali percobaan dengan beberapa kombinasi bilangan prima dan kunci private data yang sudah dienkripsi belum mampu terpecahkan. Terbukti dengan pengujian kali ini dicoba diserang dalam waktu lebih dari 15 jam, namun algoritma *ElGamal* belum bisa terpecahkan.

Selain itu, sebagai dampak dari pengujian terhadap file hanya menghasilkan perbedaan ukuran file asli dengan file yang telah dienkripsi dengan rata-rata 7 kali lipat dari file asli, kemudian file tersebut akan kembali ke ukuran semula setelah proses dekripsi, hal ini dapat dilihat pada tabel 4.

Tabel 4. Perbandingan Ukuran File sebelum dan sesudah dienkripsi

No	Nama file	Ukuran File asli (kb)	Ukuran File setelah di enkripsi (kb)	Ukuran File setelah di dekripsi (kb)
1	Struktur OrganisasiI CV	13	96	13
2	Penjelasan Skema Proses Produksi Percetakan	14	106	14
3	Pernyataan Surat Perjanjian	12	87	12
4	Surat Kuasa Pengambilan Passport	13	94	13
5	Surat Kuasa Pemegang Saham Tahunan	19	140	19

4. KESIMPULAN

Berdasarkan pembangkitan kunci pada algoritma *ElGamal* yang sudah dilaksanakan, maka dapat disimpulkan sebagai berikut :

1. Dalam percobaan yang telah dilakukan, penulis berhasil membuktikan bahwa pada proses pembangkitan kunci pada algoritma *ElGamal* dapat berjalan dengan baik dengan mengacak atau merusak isi sebuahpesan berekstensi *.docx* dengan aman. Tentunya hal ini dapat digunakan untuk melindungi dan menjaga keamanan data dari pihak-pihak yang tidak bertanggung jawab. File *ciphertext* yang dihasilkan tidak menimbulkan efek yang dapat merusak atau mengganggu file asli sebelumnya. Pengujian terhadap file hanya menghasilkan perbedaan ukuran file asli dengan file yang telah dienkripsi dengan rata-rata 7 kali lipat dari file asli, kemudian file tersebut akan kembali ke ukuran semula setelah proses dekripsi.
2. Hasil akhir diperoleh dengan melakukan proses percobaan pada file yang sudah dienkripsi dengan melakukan serangan bruteforce. Dibutuhkan waktu yang cukup lama dalam proses penyerangan untuk mengetahui karakter kunci *(password)* dari sebuah file yang sudah terenkripsi, dapat dibuktikan dalam tabel pengujian pada pembahasan sebelumnya bahwa sudah lebih dari 15 jam tapi password masih belum bisa terpecahkan. Dari segi keamanan, penggunaan metode algoritma ElGamal ini dinyatakan cukup kuat dalam penerapannya untuk membantu privasi keamanan data sesuai kebutuhan dalam penelitian ini.

5. SARAN

Adapun saran-saran yang berguna untuk penelitian selanjutnya agar output yang dihasilkan lebih baik lagi adalah sebagai berikut :

1. Proses enkripsi dalam penelitian ini cukup memakan waktu. Hal ini dikarenakan algoritma ini bergantung pada penggunaan bilangan acak yang terlalu besar dan jumlah teks yang

terlalu banyak. Sehingga hal ini akan berpengaruh pada tingkat keberhasilan proses enkripsi tersebut.
2. Untuk tetap menjaga keamanan ciphertext hasil enkripsi dengan algoritma *ElGamal*, kunci rahasia harus selalu dilindungi dari upaya manipulasi oleh pihak-pihak yang tidak bertanggung jawab.
3. Melakukan pengembangan algoritma dengan menggunakan penggabungan dengan algoritma lain untuk lebih meningkatkan tingkat ketahanan algoritma.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada seluruh pihak yang telah memberi dukungan baik moril maupun materiil sehingga penulis berhasil menulis paper ini dengan baik sesuai dengan harapan. Semoga paper ini bermanfaat untuk civitas akademika pada khususnya dan masyarakat umum pada umumnya.

DAFTAR PUSTAKA

[1]. Rashmi Singh and Shiv Kumar. 2012. ElGamal Algorithm in Cryptography. vol. 3.
[2]. Munir Rinaldi, 2006. *Kriptografi*. Bandung: Informatika Bandung.
[3]. Danang Tri Massandy, 2009. Algoritma ElGamal Dalam Pengamanan Pesan Rahasia. Bandung.
[4]. Satya Fajar Pratama. 2009. Algoritma ElGamal untuk Keamanan Aplikasi Email. Bandung.
[5]. Anandia Zaelvina. 2012. Perancangan Aplikasi Pembelajaran Kriptografi Kunci Publik ElGamal Untuk Mahasiswa. Universitas Sumatera Utara, Medan, Jurnal Dunia Teknologi Informasi. Vol. 1, No. 1, pp.56-62.
[6]. Hari kurniadi, 2015. Implementasi Algoritma Kriptografi ElGamal Untuk File Citra 2 Dimensi. Malang.
[7]. Devie R Suchendra and Charel Samuel M, 2013. Penerapan Sistem Kriptografi Menggunakan Algoritma ElGamal Pada Aplikasi Email Berbasis Web. Institut Teknologi Telkom, Bandung, Jurnal LPKIA, Vol.3 No.2, Desember 2013.
[8]. Lidya Andiny Nasution, 2013. Implementasi Kriptografi Algoritma ElGamal dengan Steganografi Teknik Least Significant Bit (LSB) Berdasarkan Penyisipan Menggunakan Fungsi Linier. Universitas Sumatera Utara, Medan.
[9]. Nur Alinuddin Kaharu, 2015. Implementasi Algoritma ElGamal Untuk Enkripsi Text Pada Aplikasi E-Mail Client. Karya Ilmiah Mahasiswa Fakultas Teknik Universitas Negeri Gorontalo, vol. 3, no. 1, p.3, January 2015.
[10]. Anandia Zelvina, 2012. Perancangan Aplikasi Pembelajaran Kriptografi Kunci Publik ElGamal Untuk Mahasiswa. Jurnal Dunia Teknologi Informasi. vol. 1, no. 1, p.59.