Pemfaktoran Bilangan Prima pada Algoritme ElGamal untuk Keamanan Dokumen PDF

Aisyatul Karima[1], L. Budi Handoko[2], Ari Saputro[3]

***Abstract*— Utilization of document files does not guarantee the security of those documents. There are acts of plagiarism of txt and doc files. Converting files into PDF can help secure the files, because PDF can not be easily plagiarized and a password can be embedded into the files. But, nowadays, PDF files also can be modified by other parties, thus, reducing the security of the files. This paper utilizes factor of prime and random numbers in cryptography using ElGamal algorithm to encrypt PDF plaintext documents into cipher text. This research uses experimental research with some experiments of encryption and decryption. The data analysis type is quantitative with non-parametric statistic. This technique is implemented to analyze the final result of experiments. The research is conducted by doing 20 times encryption using random keys. The result shows the successfulness of encryption process. The process begins by converting each character from the plaintext using ASCII table to a decimal number. ElGamal algorithm calculation is then applied using prime and random numbers to generate the keys. This process makes ElGamal algorithm superior to other algorithm. The result is character value used in the decryption process. The output of the encryption process is an encrypted PDF document (ciphertext). The results show that the combinations of prime and random numbers are successfully generated in the encryption process using ElGamal algorithm.**

***Intisari*— Pemanfaatan *file* dokumen *txt* dan *doc* ternyata tidak menjamin dokumen tersebut aman. Terdapat pihak yang melakukan tindakan plagiat terhadap file *txt* dan *doc*, sehingga muncul inisiatif untuk mengamankan *file* tersebut dengan melakukan konversi file ke PDF untuk alasan keamanan. Keunggulan *file* PDF yang tidak mudah diplagiat serta sudah dilengkapi dengan *password* ternyata juga dinyatakan sudah tidak aman lagi, karena masih mudah dimodifikasi oleh pihak lain. Dalam makalah ini, faktor bilangan prima dan bilangan acak pada kriptografi dengan algoritme ElGamal digunakan untuk mengenkripsi dokumen *plaintext* PDF menjadi dokumen *ciphertext* yang tidak mudah dimodifikasi. Penelitian menggunakan jenis eksperimental dengan beberapa percobaan proses enkripsi maupun dekripsi dengan variabel bilangan prima dan bilangan acak. Analisis data yang digunakan adalah analisis kuantitatif dengan statistik nonparametrik. Penerapan teknik ini berupa analisis hasil akhir *file* percobaan proses enkripsi serta dekripsi. Dari hasil uji coba sebanyak 20 kali proses enkripsi menggunakan kunci secara acak, dinyatakan *file* berhasil melalui proses enkripsi. Proses enkripsi diawali dengan konversi masing-masing karakter dari *plaintext* dengan menggunakan tabel ASCII menjadi bilangan desimal. Proses selanjutnya berupa perhitungan algoritme ElGamal dengan memanfaatkan bilangan acak prima serta bilangan acak lainnya untuk membangkitkan kunci. Hal inilah yang meyebabkan algoritme ElGamal lebih unggul dibanding algoritme lainnya. Hasil proses enkripsi berupa nilai karakter yang digunakan sebagai proses dekripsi. Keluaran dari proses enkripsi adalah dokumen PDF yang sudah terenkripsi menjadi sebuah pesan yang telah disandikan *(ciphertext)*. Hasil dari pengujian ini menunjukkan kombinasi bilangan prima dan bilangan acak berhasil dibangkitkan dalam proses enkripsi dengan algoritme ElGamal.**

***Kata Kunci*— Algoritme *ElGamal*, Enkripsi, Dekripsi, Bilangan Prima, Bilangan Acak, Dokumen PDF, Kriptografi.**

I. PENDAHULUAN

Dalam memanfaatkan dokumen digital, masyarakat mayoritas menggunakan *file* bertipe *.txt* maupun *.doc* untuk menyampaikan sebuah informasi data dari pihak pengirim kepada pihak penerima. Tipe file tersebut dipilih karena mudah dalam proses pembuatan *file*, yaitu hanya dengan menggunakan fasilitas perangkat lunak Notepad maupun Microsoft Word yang merupakan bawaan Windows. Ketika proses pengiriman *file* tersebut dilakukan melalui *email*, file hanya perlu dilampirkan dalam badan *email*. Selain kemudahan dalam proses pengiriman *file* bertekstensi *.txt* maupun *.doc* melalui *email*, *file* tersebut juga dapat dengan mudah digandakan serta diplagiat oleh khalayak umum. Hal ini memang legal dan wajar dilakukan karena tidak ada aturan resmi yang mengatur perlindungan *file* tersebut. Saat ini, semua orang bebas dan mudah mendistribusikan berbagai media digital, baik citra, audio maupun video [1]. Hal tersebut menyebabkan beberapa *file* penting yang merupakan properti intelektual milik seseorang yang harusnya menjadi konsumsi pribadi dapat dengan mudah digandakan, dimodifikasi, serta disalahgunakan oleh orang yang tidak berhak. Berdasarkan faktor tersebut, proteksi hak cipta terhadap media digital menjadi hal yang sangat vital.

Semakin lama, tren penggunaan *file* berekstensi *.txt* maupun *.doc* mulai bergeser dengan mengonversi *file* tersebut ke bentuk *Portable Document Format* (PDF). Salah satu tujuan konversi ke bentuk PDF adalah untuk alasan keamanan. Terdapat pihak yang melakukan tindakan plagiat terhadap file *.txt* dan *.doc*, sehingga muncul inisiatif untuk mengamankan *file* tersebut dengan melakukan konversi *file* ke PDF. PDF pertama kali muncul pada tahun 1993, dipelopori oleh Adobe System. Untuk keperluan pertukaran dokumen dengan tampilan yang mudah terbaca dan fleksibel dicetak

[1,2] *Dosen, Fakultas Ilmu Komputer, Universitas Dian Nuswantoro, Jl.Imam Bonjol no.207 Semarang 50131 INDONESIA (telp:024-3517261; fax:024-3520165; email: [1]aisyatul.karima@gmail.com, [1]aisyatul.karima@dsn.dinus.ac.id, [2]handoko@dosen.dinus.ac.id)*

[3] *Mahasiswa, Fakultas Ilmu Komputer, Universitas Dian Nuswantoro, Jl.Imam Bonjol no.207 Semarang 50131 INDONESIA (telp:024-3517261; fax:024-3520165; e-mail: arisaputro93@gmail.com)*

dalam sistem operasi yang berbeda, banyak masyarakat menggunakan aplikasi PDF ini. Ini terbukti dengan 450 juta lebih dokumen PDF yang sudah beredar di dunia digital sejak dikenalkan pada 1993 silam [2]. Angka ini berbeda jauh dengan 75 juta dokumen *file .doc* yang sudah beredar jauh lebih dahulu, di tahun 1980.

Masyarakat terbiasa dengan kemudahan yang diberikan aplikasi PDF yang tidak mudah dimodifikasi sehingga format ini dianggap mampu melindungi *file* dari penjiplakan dan pembajakan. Namun, ternyata dikeluhkan juga bahwa *file* PDF sudah tidak aman lagi. Selain itu, meskipun aplikasi PDF sudah dilengkapi dengan fasilitas *password* untuk mengamankan *file*, ternyata terdapat masyarakat yang belum memahami fasilitas tersebut sehingga *file* PDF masih bisa dengan mudah dijiplak dan dimodifikasi oleh pihak yang tidak berkepentingan. Di lain pihak, sudah bermunculan trik dan cara untuk membuka atau membongkar *password file* PDF. Oleh karena itu, *file* PDF juga dapat dengan mudah dibuka serta diubah maupun dimodifikasi menggunakan berbagai cara, di antaranya adalah fasilitas *"open with Microsoft Word"*.

Salah satu penyebab mudahnya peristiwa modifikasi maupun plagiat serta bentuk kejahatan digital lainnya adalah lemahnya manajemen informasi yang diterapkan dalam sebuah organisasi maupun instansi. Sebagai contoh, ketika seorang direktur maupun *general manager* kehilangan *notebook* maupun telepon selulernya, maka pencuri dapat dengan mudah mengakses seluruh data apapun yang ada di dalam perangkat digital tersebut. Akibatnya, seluruh data penting yang bersifat pribadi dapat dengan mudah disebarluaskan. Kasus lain yang sering terjadi adalah ketika seorang karyawan sebuah instansi mendapat promosi naik jabatan, dan di posisi yang baru diperoleh fasilitas *notebook* maupun laptop baru. Tidak ada aturan yang mengharuskan karyawan tersebut membersihkan data pada perangkat yang lama, sehingga dapat kebocoran data. Format data yang ada di dalam sebuah *notebook* ataupun laptop tentunya bervariasi, tidak hanya *file* bertipe *.doc* namun juga *file* PDF.

Terdapat beberapa algoritme kriptografi, baik yang simetris maupun asimetris, yang bisa digunakan untuk meningkatkan keamanan *file* PDF yang dirasa sudah tidak aman lagi. Salah satu algoritme kriptografi asimetris adalah algoritme ElGamal. Tingkat kesulitan dalam menghitung logaritma diskrit menjadikan algoritme ini memiliki tingkat kemananan yang cukup baik [3]. Algoritme ini menggunakan kunci asimetris yang memiliki dua kunci yang berbeda, yaitu kunci publik untuk enkripsi dan kunci privat untuk dekripsi [4]. Penerapan algoritme ElGamal ini biasanya diimplementasikan pada proses enkripsi pada tanda tangan digital. Selain itu, disebutkan bahwa salah satu kelebihan algoritme ElGamal adalah pada sisi keamanannya [5]. Keamanan algoritme ElGamal tergantung pada bilangan prima yang dipilih, yang merupakan bilangan prima yang besar. Penggunaan bilangan prima yang besar ini akan mempersulit para kriptanalis untuk memecahkan kode yang disandikan. Dalam ElGamal digunakan suatu bilangan prima p serta dua buah bilangan acak g dan x, $g < p$ dan $x < p$, dengan x adalah kunci rahasia yang digunakan dalam penyandian.

Berdasarkan penelitian sebelumnya, algoritme ElGamal diimplementasikan pada aplikasi *email* yang bertujuan untuk melindungi pesan yang dikirim melalui *email* dari berbagai serangan dari luar [6]. Penelitian lainnya menyebutkan bahwa algoritme ElGamal cukup aman untuk melindungi *file* bertipe *.docx* dari beberapa serangan dari luar [7]. Pernyataan tersebut dibuktikan dengan melakukan uji coba *brute force attack* selama lebih dari 15 jam untuk memecahkan *ciphertext*, tetapi *password* sebagai objek uji coba belum mampu dipecahkan. Dalam makalah ini, diterapkan algoritme ElGamal untuk melindungi pesan yang berupa *file* PDF dari serangan dari luar. Dengan beberapa kelebihan yang dimiliki algoritme ElGamal, diharapkan keamanan *file* PDF tersebut akan meningkat.

II. ALGORITME DALAM KRIPTOGRAFI

Penelitian sebelumnya menyatakan bahwa tingkat keamanan pesan rahasia *file* PDF sudah tidak aman lagi, dikarenakan sudah mulai bermunculan aplikasi yang bisa digunakan untuk memodifikasi pesan *file* PDF [8]. Cara konvensional yang digunakan untuk mengamankan dokumen adalah dengan melakukan konversi dokumen bertipe apapun ke dalam *file* berekstensi PDF. Namun, ternyata langkah ini tidak menyelesaikan masalah keamanan yang merupakan kebutuhan seluruh pengguna *file* PDF. Algoritme yang digunakan untuk mengamankan pesan PDF adalah algoritme kriptografi Vernam *cipher* yang bersifat simetris [8]. Namun, ternyata algoritme tersebut memiliki kelemahan, yaitu hasil enkripsi masih tampak dalam pandangan secara kasat mata manusia. Hal ini menyebabkan pesan hasil penyandian mudah dikenali oleh orang lain. Untuk mengantisipasi hal tersebut, dilakukan penggabungan antara algoritme Vernam *cipher* dengan teknik Steganografi. Kelemahan dari teknik tersebut adalah *file* hasil enkripsi mengalami perubahan ukuran, yaitu bertambah besar dikarenakan proses penggabungan dua *file* yang berasal dari proses enkripsi yang dilakukan sebanyak dua kali proses. Oleh karena itu, digunakan sebuah algoritme lain yang bertujuan untuk meningkatkan keamanan yang lebih baik dibanding metode Vernam *Cipher* dan *End of File.*

Ilmu kriptografi yang sudah teruji keandalannya untuk melindungi beberapa dokumen penting terbagi menjadi simetris dan asimetris. Masing-masing algoritme tersebut memiliki kelebihan dan kelemahan. Dalam makalah ini digunakan kriptografi dengan kunci asimetris yang memiliki dua kunci yang berbeda, yaitu kunci publik dan kunci privat. Penggunaan kunci asimetris bertujuan untuk memberikan keamanan ganda dikarenakan kunci yang digunakan untuk mengirim pesan (proses enkripsi) berbeda dengan kunci untuk membuka pesan (proses dekripsi). Salah satu algoritme yang menggunakan kunci asimetris yaitu algoritme ElGamal.

Algoritme ElGamal dapat digunakan untuk meningkatkan keamanan dokumen dengan beberapa ekstensi *file*. Algoritme ElGamal juga dapat diimplementasikan pada aplikasi *email* dengan tujuan untuk melindungi pesan yang dikirim berbagai serangan dari luar [6]. Tingkat kesulitan dalam menghitung logaritma diskrit menjadikan algoritme ini memiliki tingkat kemananan yang cukup baik [3]. Algoritme ElGamal dapat

digunakan untuk meningkatkan keamanan *file* bertipe *.docx* [7]. Proses pembangkitan kunci algoritme ElGamal berhasil menyandikan pesan berekstensi *.docx* dengan aman. *Ciphertext* yang dihasilkan tidak merusak *file* sebelumnya, hanya mengubah ukuran *file* rata-rata 7 kali lipat dari *file* asli. Namun, *file* tersebut akan kembali ke ukuran semula setelah proses dekripsi. Hasil pengujian membuktikan bahwa setelah dilakukan penyerangan *brute force*, dibutuhkan waktu lebih dari 15 jam untuk mencoba memecahkan *password*, dan algoritme ElGamal mampu bertahan, sehingga *password* tidak mampu dipecahkan.

Dalam penelitian lain diterangkan tentang probabilitas algoritme untuk para *hacker* yang mencoba memecahkan pesan tersembunyi ketika Oracle yang digunakan tidak sempurna dengan menggunakan prediksi *Most Significant Bit* (MSB) pada pesan tersembunyi [9]. Hasilnya menunjukkan bahwa perhitungan menggunakan MSB pada pesan yang sudah terenkripsi menggunakan algoritme ElGamal sangat sulit dipecahkan seperti halnya algoritme RSA. Algoritme ElGamal dapat menggunakan algoritme RSA untuk menghasilkan kunci publiknya. Hal ini dikarenakan enkripsi RSA dilakukan berdasarkan tingkat kesulitan pemfaktoran bilangan yang besar serta dikarenakan enkripsi ElGamal dilakukan berdasarkan pada tingkat kesulitan perhitungan logaritma diskrit pada modulo bilangan prima yang besar [4].

Selain itu, karakter utama dari algoritme ElGamal terletak pada proses enkripsi, yaitu ukuran *ciphertext* menjadi dua kali lipat daripada *plaintext* [10]. Proses enkripsi akan menghasilkan kunci K acak pada *ciphertext*, sehingga jika pada *plaintext* yang sama dilakukan enkripsi sebanyak dua kali, akan dihasilkan *ciphertext* yang berbeda. Hal tersebut menjadi salah satu keunggulan algoritme ElGamal. Namun, di sisi lain, sebagai akibat dari proses tersebut, *ciphertext* yang dihasilkan menjadi lebih panjang dan waktu yang dibutuhkan dalam prosesnya menjadi lebih lama [4].

Dalam proses, algortima ElGamal terbagi dalam tiga tahapan. Proses pertama yaitu pembentukan kunci acak, dilanjutkan proses enkripsi serta proses dekripsi untuk membuka pesan. Pada proses enkripsi maupun dekripsi, masing-masing *plaintext* maupun *ciphertext* dipecah menjadi blok-blok terlebih dahulu sebelum proses berjalan [11]. *Digital signature* sering menggunakan algoritme ElGamal dalam prosesnya. Sulitnya perhitungan logaritma menjadi keunggunalan algoritme ElGamal, terlebih ketika bilangan prima yang dipilih adalah bilangan prima yang besar. Sistem *ElGamal* memilih suatu bilangan prima p dan dua bilangan acak g dan x, g < p dan x < p, jika x adalah kunci rahasia [5].

Terdapat beberapa persamaan yang digunakan dalam proses pembangkitan kunci dalam algoritme ElGamal di antaranya adalah sebagai berikut [3].

A. Algoritme Generate Key

Sepasang kunci yang dibangkitkan diambil dari bilangan prima p dan dua buah bilangan acak *g* dan *x* dengan syarat $g < p$ dan $x < p$.

$$y=g^x \bmod p. \quad (1)$$

Kunci publik disimbolkan dengan variabel *y, g* dan *p*, sedangkan kunci privat disimbolkan dengan variabel *x* dan *p* [2]. Beberapa parameter yang digunakan dalam proses perhitungan algoritme ElGamal adalah sebagai berikut.

- Bilangan prima *p* bersifat tidak rahasia.
- Bilangan acak *g* ($g < p$) bersifat tidak rahasia
- Bilangan acak *x* ($x < p$) bersifat rahasia.
- Bilangan y bersifat tidak rahasia.
- *m (plaintext)* bersifat rahasia merupakan pesan asli yang digunakan untuk proses enkripsi.
- *a* dan *b (ciphertext)* bersifat tidak rahasia.

B. Algoritme Proses Enkripsi

Proses enkripsi dilakukan dengan memilih bilangan acak k yang berada dalam himpunan $1 \leq k \leq p-2$. Setiap blok *plaintext m* dienkripsi dengan (2) dan (3).

$$a = g^k \bmod p \quad (2)$$

$$b = y^x m \bmod p. \quad (3)$$

C. Algoritme Proses Dekripsi

Proses dekripsi menggunakan kunci privat x dan p untuk mendekripsi a dan b menjadi *plaintext* m dengan persamaan sebagai berikut.

$$(ax)^{-1} = a^{p-1-x} \bmod p \quad (4)$$

$$m = b * a^{x} \bmod p \quad (5)$$

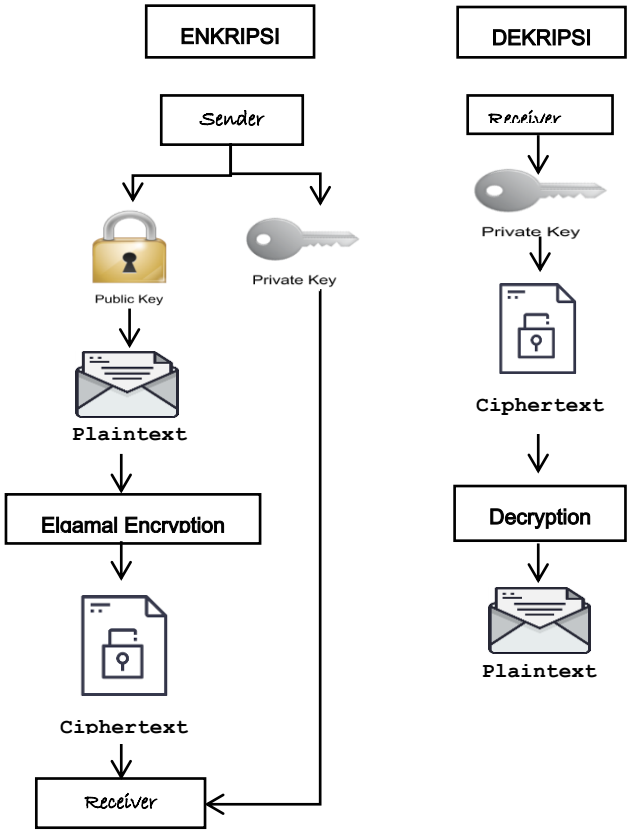
sehingga *plaintext* dapat ditemukan kembali dari pasangan *ciphertext a* dan *b.*

III. METODOLOGI

Penelitian ini menggunakan jenis penelitian eksperimental dengan melakukan beberapa percobaan baik dalam proses enkripsi maupun dekripsi. Analisis data yang digunakan adalah analisis kuantitatif dengan statistik nonparametrik. Penerapan teknik analisis kuantitatif berupa analisis hasil akhir *file* percobaan proses enkripsi serta dekripsi. Sedangkan data yang digunakan adalah berupa dokumen *file* berektensi PDF yang dijadikan sebagai pesan asli *(plaintext)* yang akan disandikan menggunakan algoritme ElGamal. Dalam pengumpulan data, dilakukan *literature review* yang sumbernya diperoleh dari jurnal, buku, internet, artikel, serta seluruh sumber informasi yang berkaitan dengan topik algoritme ElGamal ini.

Pengujian yang dilakukan adalah dengan mencoba proses enkripsi untuk mengetahui tingkat keberhasilan perangkat lunak yang digunakan untuk menyandikan sebuah pesan PDF. Selain itu, percobaan selanjutnya adalah mencoba proses dekripsi untuk mengetahui kemampuan perangkat lunak tersebut mengembalikan *file* PDF yang sudah disandikan menjadi *file* PDF semula.

Gbr. 1 memberikan penjelasan tahapan langkah dalam penelitian ini, yang ditunjukkan dengan desain model penelitian secara keseluruhan.



Gbr. 1 Metode penelitian.

Berdasarkan Gbr. 1, penelitian terbagi menjadi dua proses utama, yaitu proses enkripsi dan dekripsi. Adapun tahapan proses enkripsi diawali dengan pembangkitan kunci secara acak di sisi pengirim yang akan menghasilkan kunci privat (x,p), kunci publik (y,g,p), bilangan prima (p), serta bilangan acak (g), (x), dan (k). Sebelum proses enkripsi, *plaintext* yang berupa *file* dokumen PDF disiapkan terlebih dahulu, dilanjutkan dengan proses perhitungan rumus algoritme ElGamal. Proses enkripsi tersebut menghasilkan *ciphertext* yang akan dikirim ke pihak penerima untuk proses selanjutnya.

Untuk proses dekripsi, sisi penerima menerima *ciphertext,* kunci privat, serta bilangan prima dari pihak pengirim pesan, yang dilanjutkan dengan proses dekripsi menggunakan rumus dekripsi algoritme ElGamal. Hasil dari proses dekripsi berupa *file* dengan karakter ASCII yang nantinya diubah ke bentuk teks yang menjadi dokumen asli *(plaintext).*

IV. HASIL DAN PEMBAHASAN

Proses pengamanan dokumen PDF diawali dengan proses enkripsi menggunakan algoritme ElGamal. Tahapan prosesnya adalah sebagai berikut.

A. Proses Enkripsi

Proses enkripsi diawali dengan proses pembangkitan kunci secara acak yang terdiri atas variabel p, g, dan x. Sepasang kunci yang dibangkitkan diambil dari bilangan prima *p* dan dua buah bilangan acak *g* dan *x* dengan syarat $g < p$ dan $x < p$. Dengan menggunakan variabel awal, kunci publik (y,g,p) dan bilangan acak pengirim (k) digunakan untuk proses enkripsi menggunakan algoritme ElGamal.

Algoritme enkripsi ElGamal diimplementasikan oleh fungsi enkripsi yang memiliki masukan teks awal yang akan dikonversi ke dalam nilai angka desimal dari ASCII. Selain itu, kelas teks sandi ElGamal mempunyai dua nilai yaitu *a* dan *b*, yang mempunyai rumus masing-masing seperti pada (2) dan (3).

Kedua persamaan tersebut, jika diimplementasikan dalam sebuah bahasa pemrograman, akan tampak sebagai fungsi seperti pada Gbr. 2.

```
//Enkripsi Plaintext
public function _enkripsi($plainText){
    $ascii = $this->_toAscii($plainText);
    $chipperText = "";
    for ($i=0;$i<(strlen($ascii));$i+=3){
        $tmp = substr($ascii,$i,3);
        if (strlen($tmp)==1) $tmp = "00".$tmp;
        if (strlen($tmp)==2) $tmp = "0".$tmp;
        $a = $this->_rekursifMod($this->g,$this->k,$this->p);
        $b = ((($this->_rekursifMod($this->y,$this->k,$this->p))*
             ($this->_rekursifMod($tmp,1,$this->p))) % $this->p);
        $chipperText .= $a." ".$b." ";
    }
    return $chipperText;
}
```

Gbr. 2 Implementasi rumus a dan rumus b algoritme ElGamal.

Dari hasil uji coba enkripsi sebanyak 20 kali proses menggunakan kunci secara acak, dinyatakan proses enkripsi berhasil dilalui dengan baik. Hal ini seperti ditunjukkan pada Gbr. 3.

Gbr. 3 Uji pembangkitan kunci dan proses enkripsi.

Adapun proses perhitungan rumus, seperti yang disebutkan pada Tabel I menggunakan algoritme *ElGamal,* dilakukan dengan memilih bilangan acak yang berada dalam himpunan $1 \leq k \leq p-2$. Setiap blok *plaintext m* dienkripsi dengan (2) dan (3).

Sebagai simulasi perhitungan, digunakan *plaintext (m)*=SECRET. Berdasarkan *plaintext* tersebut, dengan menggunakan tabel ASCII, masing-masing karakter dikonversi terlebih dahulu menjadi bilangan desimal sebelum proses enkripsi, yaitu sebagai berikut.

S = 83
E = 69
C = 67
R = 82
E = 69
T = 84.

Adapun nilai variabel lainnya adalah sebagai berikut.

Bilangan acak *(g)* = 31
Bilangan acak pengirim *(k)* = 10
Bilangan acak prima *(p)* = 457
Kunci private *(x)* = 439
Kunci public *(y)* berdasarkan (1)

maka,

$y = 31^{439} \bmod 457$
$y = 145$

Perhitungan selengkapnya disajikan pada Tabel I.

TABEL I
PERHITUNGAN RUMUS ENKRIPSI ALGORITME ELGAMAL

No	Karakter ASCII	Rumus Enkripsi		Hasil	
		$a = g^k \bmod p$	$b = y^k m \bmod p$	a	b
1	S	$a = g^k \bmod p$ $= 31^{10} \bmod 457$ $= 14$	$b = y^k m \bmod p$ $= 145^{10}.83 \bmod 457$ $= 61$	14	61
2	E	$a = g^k \bmod p$ $= 31^{10} \bmod 457$ $= 14$	$b = y^k m \bmod p$ $= 145^{10}.69 \bmod 457$ $= 315$	14	315
3	C	$a = g^k \bmod p$ $= 31^{10} \bmod 457$ $= 14$	$b = y^k m \bmod p$ $= 145^{10}.67 \bmod 457$ $= 286$	14	286
4	R	$a = g^k \bmod p$ $= 31^{10} \bmod 457$ $= 14$	$b = y^k m \bmod p$ $= 145^{10}.82 \bmod 457$ $= 275$	14	275
5	E	$a = g^k \bmod p$ $= 31^{10} \bmod 457$ $= 14$	$b = y^k m \bmod p$ $= 145^{10}.69 \bmod 457$ $= 315$	14	315
6	T	$a = g^k \bmod p$ $= 31^{10} \bmod 457$ $= 14$	$b = y^k m \bmod p$ $= 145^{10}.84 \bmod 457$ $= 304$	14	304

Berdasarkan Tabel I, maka nilai hasil enkripsi dengan penulisan adalah

= a1, b1, a2, b2, a3, b3, a4, b4, a5, b5, a6, b6
= 14, 61, 14, 315, 14, 286, 14, 275, 14, 315, 14, 304

Nilai karakter ini merupakan hasil proses enkripsi yang kemudian akan digunakan sebagai proses dekripsi *file* menggunakan rumus dekripsi: $b.a^{p-1-x} \bmod p$.

Keluaran proses enkripsi dengan perhitungan algoritme ElGamal tersebut menghasilkan sebuah dokumen PDF yang sudah terenkripsi menjadi sebuah pesan *ciphertext.*

Hasil akhir enkripsi diubah kembali menjadi karakter yang disesuaikan dengan daftar karakter yang terdapat pada tabel ASCII. Setelah proses enkripsi berakhir, maka *ciphertext* akan dikirim ke penerima untuk proses selanjutnya.

Proses enkripsi dengan algoritme ElGamal ini memanfaatkan bilangan acak prima serta beberapa bilangan acak lainnya untuk membangkitkan kunci. Hal inilah yang meyebabkan algoritme ElGamal lebih unggul dibanding dengan algoritme lainnya. Seperti yang disebutkan pada penelitian yang menggunakan algoritme Vernam *cipher* yang bersifat simetris untuk mengamankan pesan yang berekstensi PDF, tetapi ternyata algoritme tersebut memiliki kelemahan yaitu hasil enkripsi masih tampak dalam pandangan secara kasat mata manusia, sehingga masih mudah dikenali oleh orang lain [8]. Lain halnya dengan algoritme ElGamal ini, pada *file* hasil enkripsi atau *ciphertext* secara kasat mata tidak tampak ada perubahan yang berarti, sehingga orang lain tidak mengetahui bahwa *file* PDF tersebut sudah terenkripsi dengan kunci khusus.

Pada dasarnya, dalam menghasilkan kunci publik, algoritme ElGamal dapat menerapkan algoritme RSA. Hal ini dikarenakan enkripsi RSA memanfaatkan tingkat kesulitan pemfaktoran bilangan yang besar. Hal ini sama dengan algoritme ElGamal yang juga memanfaatkan tingkat kesulitan perhitungan logaritma diskrit pada modulo bilangan prima yang besar [4].

B. Proses Dekripsi

Proses dekripsi dilakukan menggunakan *ciphertext* hasil proses enkripsi dilengkapi dengan kunci privat serta bilangan prima dari pihak pengirim.

Algoritme dekripsi ElGamal diimplementasikan oleh fungsi dekripsi pada Gbr. 4. Fungsi dekripsi memiliki masukan sebuah objek teks sandi ElGamal *(ciphertext)* yang mempunyai dua nilai variabel yaitu *a* dan *b*.

Proses dekripsi dilakukan menggunakan kunci privat *x* dan *p* untuk mendekripsi *a* dan *b* menjadi *plaintext m* menggunakan (4) dan (5).

```
//Dekripsi Chippertext
public function _dekripsi($chipperText){
    $t = explode(" ",$chipperText);
    $ascii = "";
    for ($i=0;$i<(count($t));$i+=2){
        $pkt = $this->p - 1 - $this->x;
        $a = $this->_rekursifMod($t[$i],$pkt,$this->p);
        $b = (($t[$i+1]*$a) % $this->p);
        if (strlen($b)==1) $b = "00".$b;
        if (strlen($b)==2) $b = "0".$b;
        $ascii .= $b;
    }
    return $this->_toText($ascii);
}
```

Gbr. 4 Implementasi rumus dekripsi algoritme ElGamal.

Dalam proses ini, sebagai contoh digunakan kunci privat *(x,p)* = (439, 457) serta bilangan prima *(p)* = 457. Dalam proses dekripsi menggunakan algoritme ElGamal, digunakan kunci privat untuk mendekripsi *a* dan *b* menjadi *plaintext* dengan memanfaatkan (4) dan (5).

Adapun *file* hasil enkripsi adalah

S = a1, b1 (14,61)
E = a2, b2 (14,315)
C = a3, b3 (14,286)
R = a4, b4 (14,275)
E = a5, b5 (14,315)
T = a6, b6 (14,304)

Berdasarkan *file* hasil enkripsi tersebut, langkah selanjutnya adalah proses perhitungan nilai *file* enkripsi menjadi *plaintext* (m) sesuai dengan Tabel II.

TABEL II
PERHITUNGAN RUMUS DEKRIPSI ALGORITME ELGAMAL

No	*Plaintext* (m)	Rumus Dekripsi ($b.a^{p-1-x} \bmod p$)	Hasil	Konversi Karakter ASCII
1	m1	$m1 = b.a^{p-1-x} \bmod p$ $= 61.14^{457-1-439} \bmod 457$ $= 61.14^{17} \bmod 457$ $= 83$	83	S
2	m2	$m2 = b.a^{p-1-x} \bmod p$ $= 315.14^{457-1-439} \bmod 457$ $= 315.14^{17} \bmod 457$ $= 69$	69	E
3	m3	$m3 = b.a^{p-1-x} \bmod p$ $= 286.14^{457-1-439} \bmod 457$ $= 286.14^{17} \bmod 457$ $= 67$	67	C
4	m4	$m4 = b.a^{p-1-x} \bmod p$ $= 275.14^{457-1-439} \bmod 457$ $= 275.14^{17} \bmod 457$ $= 82$	82	R
5	m5	$m5 = b.a^{p-1-x} \bmod p$ $= 315.14^{457-1-439} \bmod 457$ $= 315.14^{17} \bmod 457$ $= 69$	69	E
6	m6	$m6 = b.a^{p-1-x} \bmod p$ $= 304.14^{457-1-439} \bmod 457$ $= 304.14^{17} \bmod 457$ $= 84$	84	T

Sesuai Tabel II tersebut, proses dekripsi telah menghasilkan *plaintext(m) m1*=83, *m2*=69, *m3*=67, *m4*=82, *m5*=69, *m6*=84. Dari perhitungan proses dekripsi tersebut diperoleh *file* yang akan diubah kembali ke bentuk karakter atau teks yang sebelumnya berupa karakter ASCII menggunakan tabel ASCII. Adapun hasil konversi dari *m1, m2, m3, m4, m5, dan m6* tersebut ke dalam bentuk karakter sebelumnya adalah sebagai berikut.

m1 = 83 = S
m2 = 69 = E
m3 = 67 = C
m4 = 82 = R
m5 = 69 = E
m6 = 84 = T

Hasil akhir dari proses dekripsi ini berupa *file* dokumen asli *(plaintext)* sebagaimana yang dikirimkan di awal proses enkripsi yang merupakan *file* berkestensi PDF. *File* tersebut dapat langsung dibaca oleh penerima pesan.

V. KESIMPULAN

Dokumen PDF yang semula mudah diplagiat meski sudah dilengkapi fasilitas *password*, setelah dienkripsi menggunakan algoritme ElGamal menjadi tidak mudah diplagiat oleh pihak lain. Hal ini dikarenakan implementasi algoritme ElGamal menggunakan kunci asimetris yang berupa kunci privat dan kunci publik yang berasal dari pemfaktoran bilangan prima serta bilangan acak. Kelebihan algoritme ElGamal yang terletak pada tingkat kesulitan perhitungan logaritma diskrit pada modulo bilangan prima yang besar inilah yang menjadikan dokumen PDF lebih aman serta tidak mudah dimodifikasi oleh pihak lain. Berdasarkan hasil uji coba yang telah dilaksanakan baik dalam proses pembangkitan kunci, enkripsi, maupun dekripsi, algoritme ElGamal berhasil menyandikan dokumen PDF menjadi *ciphertext* yang tidak mudah diketahui oleh pihak lain. Selain itu, variasi bilangan serta kunci privat dan kunci publik yang digunakan menjadikan algoritme ini tidak mudah dipecahkan oleh pihak yang tidak bertanggung jawab.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada seluruh pihak yang telah memberi dukungan baik morel maupun materiel sehingga penulis berhasil menulis makalah ini dengan baik sesuai dengan harapan. Semoga makalah ini bermanfaat untuk civitas akademika dalam bidang keamanan data pada khususnya dan masyarakat umum pada umumnya.

REFERENSI

[1] Rani Septia and Harjoko Agus, “Skema Proteksi Hak Cipta untuk Citra Warna Digital Menggunakan Visual Cryptography”. *Jurnal Nasional Teknik Elektro dan Teknologi Informasi (JNTETI)*, Vol.5, No.4, November 2016.

[2] Vicky. [Online] den 26 September 2012. [Citat:] http://belajar-komputer-mu.com/mengenal-pdf-dan-cara-mengconvert-file-word-to-pdf-dengan-microsoft-word-2007/.

[3] Munir, Rinaldi. "*Kriptografi*". Bandung : Informatika Bandung, 2006.

[4] Singh, Rasmi and Kumar, Shiv, "ElGamal’s Algorithm in Cryptography", *International Journal of Scientific & Engineering Research (IJSER)*, vol.3, issue 12, 2012.

[5] Widyartono, Agustinus "Algoritma ElGamal untuk Enkripsi Data Menggunakan GPUNG", *Jurnal Teknologi Dan Informatika (TEKNOMATIKA)*, s. 31., 2011.

[6] Pratama, Satya Fajar, "*Algoritma ElGamal Untuk Keamanan Aplikasi Email*". Bandung : u.n., 2009.

[7] Karima Aisyatul and Saputro Ari. "Pembangkitan Kunci pada Algoritma Asimetris ElGamal untuk Meningkatkan Keamanan Data bertipe .docx". *Sisfotenika*, 2016.

[8] Dibiyo Marsela Sutikno and Karima Aisyatul. "Implementasi Vernam Cipher dan Steganografi End of File (EoF) untuk Enkripsi Pesan PDF", *Techno.Com*, ss. 66-71, 2016.

[9] Kang, Zheng-Qi and Wei Lv, Ke. "New Result on The Hardness of ElGamal and RSA bits basing Binary Expansions". *IEEE, 2nd International Conference on Information Science and Control Engineering*. s. 336, 2015.

[10] Wu, Zengqiang, Su, Di and Ding, Gang. Jinzhou, "ElGamal Algorithm for Encryption Data Transmission". *International Conference on Mechatronics and Control (ICMC).* ss. 1464 - 1467, 2014.

[11] Zelvina, Anandia, Efendi, Syahril and Arisandi, Dedy, "Perancangan Aplikasi Pembelajaran Kriptografi Kunci Publik ElGamal untuk Mahasiswa". *Jurnal Dunia Teknologi Informasi.* ss. 56-62, 2012.