Volume 1, Oktober 2017
ISSN: 2614-1205

Efektifitas Algoritma ElGamal untuk Perlindungan Dokumen PDF

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

[1,2,3]Jurusan Teknik Informatika, FIK, Universitas Dian Nuswantoro Semarang
Email: [1]handoko@dosen.dinus.ac.id ,[2]aisyatul.karima@dsn.dinus.ac.id ,[3]arisaputro93@gmail.com

Abstrak

Berbagai macam jenis format dokumen tidak terlepas dari tindakan plagiasi, termasuk diantaranya adalah dokumen dengan ekstensi PDF. Meskipun PDF dapat dilengkapi dengan pengamanan menggunakan *password*, namun hal tersebut tidak mengurangi niat penyerang untuk mencoba membuka dokumen PDF tersebut. Tulisan ini akan membahas tentang efektifitas pemanfaatan algoritma Kriptografi ElGamal untuk memberikan alternatif maupun tambahan pengamanan yang dapat diimplementasikan terhadap dokumen PDF. Uji coba akan dilakukan dengan memberikan analisa terhadap model penyerangan terhadap dokumen PDF yang telah dienkripsi dengan menggunakan metode *brute-force attack* dengan memanfaatkan *tools* yang banyak beredar di internet. Hasil dari tulisan ini berupa bagaimana efektifitas algoritma ElGamal terhadap dokumen PDF.
Kata Kunci: Algoritma *ElGamal*, dokumen PDF, enkripsi, *brute-force attack.*

Abstract

Various types of document are not secure from plagiarism, such as a PDF document. Although the PDF has been completed with the password, but there are many hackers to attack the document. This paper discuss about the effectiveness of utilizing the ElGamal Cryptography Algorithm to provide the additional protection on PDF document. Analyzing process based on the experimental process on attack model of PDF document use the brute force attack. The result shows that how the effectiveness ElGamal algorithm against PDF document.
Keywords: *ElGamal algorithm, PDF document, encryption, brute force attack*

1. PENDAHULUAN

Kepraktisan penggunaan dokumen dengan format *(Portable Document Format)* PDF cukup membuat ekstensi file ini menjadi favorit para penggunanya. Dengan menggunakan format PDF ini, dokumen siap untuk dicetak dengan susunan dan format yang telah ditentukan sebelumnya. Kelebihan tersebut menjadikan ekstensi file PDF sangat praktis karena tidak terpengaruh oleh kondisi apapun. Selain itu, alasan lain penggunaan format ini adalah dalam segi keamanan, format dokumen ini dianggap tidak mudah diubah sehingga akan meminimalkan tindakan plagiasi yang mungkin dilakukan oleh pihak-pihak yang tidak berkepentingan. Namun, kini muncul aplikasi yang bisa merubah atau merusak pesan format PDF yang menyebabkan menurunnya tingkat keamanan dokumen itu sendiri [1].

Berdasarkan [2] sejak ditemukannya PDF pada tahun 1993 oleh Adobe System, telah ditemukan sekitar 450 juta lebih dokumen PDF yang beredar di dunia digital. PDF masih menjadi pilihan utama yang digunakan oleh khalayak ramai untuk pertukaran data dengan keunggulan yang dimilikinya yang mudah dibaca dan fleksibel, serta dapat dibuka dengan platform yang berbeda sekalipun.

Terdapat asumsi bahwa selain kemudahan yang diberikan oleh dokumen PDF, isi dari dokumen ini juga tidak mudah untuk dimodifikasi. Namun belakangan hal tersebut tidak menjadi jaminan lagi. Meskipun telah dilengkapi dengan fasilitas *password* untuk pengamanannya, ternyata masih banyak juga yang belum memahami tentang fasilitas tersebut, sehingga masih banyak dokumen-dokumen PDF yang dengan mudah dapat dijiplak dan dimodifikasi oleh pihak-pihak yang tidak berkepentingan [3]. Selain itu juga

Volume 1, Oktober 2017
ISSN: 2614-1205

telah muncul alat maupun perangkat bantu yang dapat dipergunakan untuk membongkar *password* yang dapat dengan mudah didapatkan di internet secara bebas.

Algoritma kriptografi baik yang simetris maupun asimetris dapat digunakan untuk meningkatkan keamanan dokumen PDF yang dirasa sudah tidak aman lagi. Salah satu kriptografi asimetris yang digunakan yaitu algoritma ElGamal. Tingkat kesulitan dalam menghitung logaritma diskrit menjadikan algoritma ini memiliki tingkat kemananan yang cukup baik [4].

Dalam algoritma ini menggunakan kunci asimetris yang memiliki dua kunci yang berbeda, yaitu kunci publik untuk enkripsi dan kunci pribadi untuk dekripsi [5]. Algoritma ElGamal ini tahan terhadap serangan dari luar yang mencoba memecahkan kuncinya, hal ini dikarenakan algoritma ini sangat kompleks [6].

Penelitian terkait [7] 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 terjamin.

Adapun kelebihan dari algoritma ElGamal [8] yaitu pada sisi keamanannya yang tergantung pada bilangan prima yang dipilih merupakan bilangan prima yang besar. Dengan bilangan prima yang besar ini akan mempersulit para kriptanalis untuk memecahkan kode yang disandikan. Dalam ElGamal menggunakan suatu bilangan prima p serta dua buah biangan acak g dan x, $g < p$ dan $x < p$, dengan x adalah kunci rahasia yang digunakan dalam penyandian.
Penelitian lain [6] menyebutkan bahwa algoritma ElGamal cukup aman untuk melindungi file bertipe .docx dari beberapa serangan dari luar. Pernyataan tersebut dibuktikan dengan melakukan uji coba brute force attack selama lebih dari 15 jam untuk memecahkan *ciphertext* namun password sebagai objek ujicoba belum mampu dipecahkan.

Dalam penelitian lain [9] menunjukkan bahwa perhitungan menggunakan MSB pada pesan yang sudah terenkripsi menggunakan algoritma ElGamal sangat sulit dipecahkan seperti halnya algoritma RSA. Berdasarkan beberapa kelebihan tersebut, dalam penelitian ini akan menerapakan algoritma ElGamal ini untuk melindungi pesan yang berupa dokumen PDF dari serangan *brute force*. Dengan beberapa kelebihan yang dimiliki algoritma ElGamal, diharapkan akan meningkatkan tingkat keamanan dokumen PDF tersebut.

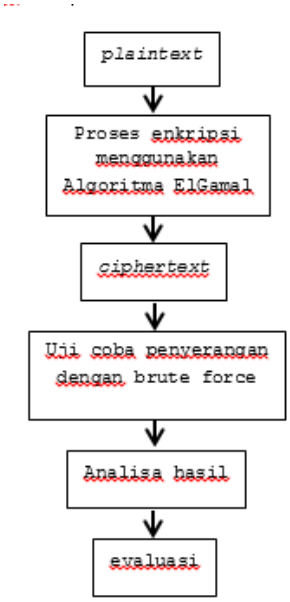
2. METODE

Metode penelitian yang digunakan dalam penelitian ini berupa metode eksperimental dengan melakukan uji coba penyerangan terhadap dokumen PDF yang sudah dienkripsi menggunakan algoritma ElGamal. Uji coba dilakukan dengan menguji dokumen yang sudah dienkripsi (*ciphertext)* tersebut dengan menggunakan aplikasi *brute force attack* yang merupakan salah satu aplikasi yang digunakan untuk menguji tingkat ketahanan sebuah algoritma kriptografi. Adapun metode penelitian bisa dilihat pada Gambar 1.

Proses diawali dengan menyiapkan pesan asli *(plaintext)* yang merupakan dokumen PDF, dilanjutkan dengan proses enkripsi menggunakan algoritma ElGamal. Hasil dari proses enkripsi tersebut berupa dokumen PDF yang sudah disandikan menggunakan kunci khusus menjadi sebuah pesan yang sudah disandikan *(ciphertext)*. *Ciphertext* tersebut akan diuji coba penyerangan menggunakan *brute force attack* untuk mengetahui tingkat ketahanan algoritma ElGamal terhadap serangan dari luar. Hasil uji coba

Volume 1, Oktober 2017
ISSN: 2614-1205

tersebut akan dianalisa dan dievaluasi bagaimana efektivitas algoritma ElGamal untuk mengamankan dokumen PDF.



Gambar 1. Desain penelitian

3. HASIL DAN PEMBAHASAN

3.1 Algoritma ElGamal

Berdasarkan desain penelitian pada Gambar 1 di atas, langkah awal yang dilakukan adalah menyiapkan pesan asli yang berupa dokumen PDF sebagai *plaintext* sebagai bahan untuk enkripsi. Untuk selanjutnya proses enkripsi menggunakan algoritma ElGamal.Dalam algoritma ElGamal terdapat beberapa persamaan yang digunakan baik untuk proses pembangkitan kunci, proses enkripsi, serta proses dekripsi [3].
Dalam proses pembangkitan kunci diambil dari bilangan prima p dan dua buah bilangan acak *(random) g* dan *x* dengan syarat $g < p$ dan $x < p$.

$$y=g^x mod\ p \tag{1}$$

Dari [3] variabel *y, g* dan *p* merupakan kunci publik, sedangkan variabel *x* dan *p* merupakan kunci privat. Adapun variabel lainnya yang dibutuhkan dalam algoritma *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.

Selain itu, dalam proses enkripsi setiap blok *plaintext m* dienkripsi dengan kedua persamaan berikut ini.

$$a = g^k mod\ p \tag{2}$$
$$b = y^x m\ mod\ p \tag{3}$$

Variabel *k* merupakan bilangan acak yang berada dalam himpunan $1 \le k \le p-2$. Sedangkan variabel *a* dan *b* merupakan pasangan hasil enkripsi yang akan dijadikan sebagai *ciphertext.*

Volume 1, Oktober 2017
ISSN: 2614-1205

Kemudian untuk proses dekripsi menggunakan kunci privat *x* dan *p* untuk mendekripsi *a* dan *b* hasil enkripsi menjadi *plaintext m* dengan persamaan (4) dan (5) sehingga *plaintext* dapat ditemukan kembali pasangan *ciphertext a* dan *b.*

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

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

3.2 Implementasi Algoritma ElGamal

Adapun untuk simulasi perhitungan manual enkripsi dengan rincian variabel sebagai berikut [3] :

Plaintext (m) : %PDF
Bilangan acak (g) : 31
Bilangan acak pengirim (k) : k = 10
Bilangan acak prima (p) : 457
Kunci *private* (x) : 439
Kunci *public* (y) : $y = g^x \bmod p$
$y = 31^{439} \bmod 457$
$y = 145$

Berdasarkan variabel yang sudah ditentukan kita ubah dulu *plaintext* (m) yang berupa karakter %PDF menjadi bilangan desimal menggunakan tabel ASCII dengan % = 37, P = 80, D = 68, dan F = 70. Adapun simulasi perhitungan manual dengan algoritma ElGamal bisa dilihat 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	%	$a = g^k \bmod p$ $= 31^{10} \bmod 457$ $= 14$	$b = y^k m \bmod p$ $= 145^{10}.37 \bmod 457$ $= 308$	*14*	*308*
2	P	$a = g^k \bmod p$ $= 31^{10} \bmod 457$ $= 14$	$b = y^k m \bmod p$ $= 145^{10}.80 \bmod 457$ $= 246$	*14*	*246*
3	D	$a = g^k \bmod p$ $= 31^{10} \bmod 457$ $= 14$	$b = y^k m \bmod p$ $= 145^{10}.68 \bmod 457$ $= 331$	*14*	*331*
4	F	$a = g^k \bmod p$ $= 31^{10} \bmod 457$ $= 14$	$b = y^k m \bmod p$ $= 145^{10}.70 \bmod 457$ $= 101$	*14*	*101*

Berdasarkan Tabel 1 tersebut, maka *ciphertext* yang dihasilkan dari proses enkripsi adalah = a1, b1, a2, b2, a3, b3

= 14, 308, 14, 246, 14, 331, 14, 101

Ciphertext inilah yang akan kita uji coba penyerangan menggunakan *brute force attack.*

3.3 Analisa pengujian penyerangan

Langkah selanjutnya adalah pengujian penyerangan menggunakan *brute force attack.* Berdasarkan hasil uji coba enkripsi menggunakan ekstensi file .txt file tersebut masih bisa dibuka yang berupa hasil dari proses enkripsi yang merupakan perubahan kode ASCII dari isi file .txt tersebut.

Namun ketika diuji coba enkripsi menggunakan ekstensi file .docx [6] file tersebut tidak terbaca hasil nilai ASCII nya ketika dibuka menggunakan Microsoft Word. Begitu juga sama halnya ketika diterapkan

pada ekstensi file PDF file juga tidak bisa terbaca seperti nampak file PDF yang rusak. Ketika melakukan uji coba file berekstensi .docx dibuka menggunakan aplikasi *brute force,* file *ciphertext* hasil enkripsi masih bisa dibuka, namun ketika menggunakan ekstensi file PDF seolah file tersebut ditolak oleh aplikasi *brute force.*

Berdasarkan hasil analisa, perbedaan format file .txt, .doc, .docx maupun PDF masing-masing dokumen memiliki format dan karakter sendiri yang khas. Untuk file txt, memang tidak ada format-nya sehingga bisa langsung dibuka dan dibaca dengan aplikasi apapun. Ini perbedaan mendasar untuk file .txt. Hal ini merupakan perbedaan mendasar untuk file .txt.

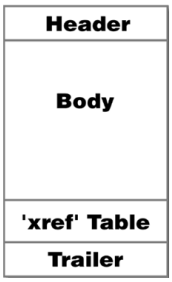
Kemudian untuk file .doc dan .docx memiliki format atau *header* tertentu yang biasanya berbasis XML yang normalnya berisi ASCII murni, namun meskipun memiliki format header tertentu, tiap aplikasi memperlakukan file .doc maupun .docx ini seperti hal-nya file .txt. Berdasarkan hal tersebut mengakibatkan header tidak diketemukan, sehingga file tersebut akan dibuka dengan model file ASCII, dan hasilnya ketika file dibuka akan menampilkan file dengan isi karakter ASCII yang sepertinya tidak beraturan.

Khusus untuk file PDF, ini sama hal-nya juga dengan .doc ataupun .docx, dimana file PDF memiliki format *header* khusus yang mana aplikasi akan memperlakukan file PDF sesuai dengan formatnya *(Portable Document File)*. Oleh karena itu, bila *header* ini tidak terbaca, maka efek-nya adalah file tidak dikenal. Hal inilah yang menyebabkan *brute force* PDF menolak, karena dia tidak bisa membaca *header* format file PDF. Sebagai penjelasan lebih lanjut, bisa dilihat pada Gambar 2 sebagai ilustrasi file PDF.

Gambar 2. Ilustrasi file PDF

Sesuai Gambar 2 tersebut, nampak kotak pertama merupakan *header* dari dokumen dengan format PDF. Adapun komposisi atau penempatan header untuk dokumen PDF bisa dilihat pada Gambar 3.

Volume 1, Oktober 2017
ISSN: 2614-1205



Gambar 3. Komposisi file PDF [10]

Selain *header*, peletakan posisi lainnya seperti 'xref' Table dan Trailer yang merupakan halaman terakhir dari ilustrasi file PDF terletak di bagian terakhir dari Gambar 3. Dengan demikian, wajar saja jika aplikasi *cracker* yang dipakai tidak bisa digunakan untuk membaca dokumen ini, hal ini dikarenakan biasanya dokumen dengan extensi PDF, minimal harus memiliki *header* yang sesuai.

Untuk header bisa dilihat pada Gambar 2 pada dua baris atau lima baris pertama. Untuk dokumen yang *plain*, biasanya hanya 2 baris sedangkan yang terenkripsi bisa sampai 5 baris karena mengandung informasi enkripsi. Gambar-gambar tersebut memang dapat dilihat secara visual, jika kita buka dengan menggunan aplikasi seperti notepad atau sejenisnya.

Jika kita menghendaki aplikasi cracker dapat membuka tanpa error, maka paling tidak yang perlu dilakukan adalah hanya mengenkrip bagian body saja yang notabene berisi dokumen yang sesungguhnya. Dengan kondisi dienkrip secara penuh seluruh file, maka extensi PDF menjadi tidak berguna dan tidak dikenal oleh aplikasi lain atau aplikasi cracker yang digunakan. Pada Gambar 4 berikut ini adalah ilustrasi file PDF yang telah dienkripsi menggunakan algoritma ElGamal. Dari gambar tersebut nampak *header* sudah berubah, sehingga tidak dikenal oleh program apapun yang biasanya bisa mengenal format file PDF.

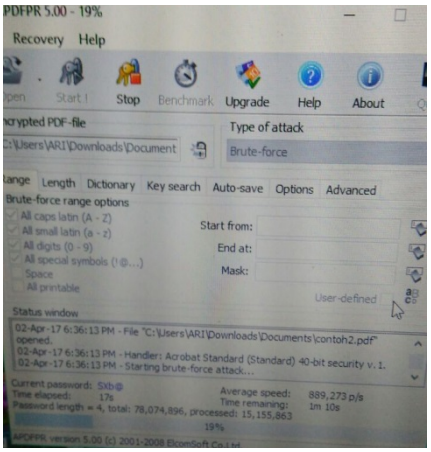
Gambar 4. Ilustrasi file PDF setelah dienkripsi dengan algoritma ElGamal

Gambar 5. Ilustrasi file PDF setelah diproteksi dengan *password* internal

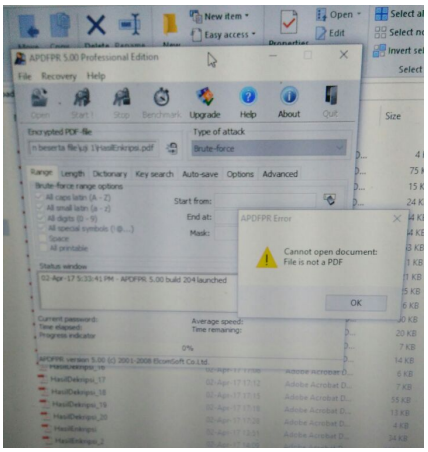
Coba kita bandingkan gambar 4 dengan gambar 5 berikut. Pada gambar 5, file PDF sudah dienkripsi secara default dengan menggunakan proteksi *password* internal dari aplikasi pembuat PDF, jika diamati *header*-nya mirip tetapi ada perbedaan sedikit.
Setelah mengamati gambar 4 dan gambar 5, kemudian perhatikan ilustrasi pada gambar 6 berikut ini.

Gambar 6. Ilustrasi file PDF dengan pengamanan sertifikat SSL

Gambar 6 tersebut merupakan file PDF dimana file tersebut bahkan kontennya diamankan dengan menggunakan pengamanan sertifikat SSL, dan tentunya ada perbedaan dengan dokumen PDF yang lain.



Gambar 7. Hasil uji coba *brute force*



Gambar 8. Uji coba membuka file PDF

Uji coba penyerangan *brute force* sesuai pada Gambar 7 tersebut terhadap dokumen PDF yang terenkripsi baru dilaksanakan sebatas pada *ciphertext* yang berupa file PDF secara utuh termasuk *header* dokumen tersebut. Oleh karena itu, analisa penyerangan masih belum bisa dilakukan secara menyeluruh. Sebagai bahan analisa awal dengan menggunakan aplikasi *cracker* yang digunakan, *ciphertext* belum dapat terpecahkan baik kunci maupun *plaintext*-nya seperti nampak pada Gambar 8 yang mencoba membuka dokumen PDF yang sudah terenkripsi namun tidak berhasil. Dokumen PDF yang terenkripsi dianggap seolah file dokumen tersebut rusak, seperti yang sudah dijelaskan di atas hal ini terjadi karena dokumen dienkripsi secara keseluruhan termasuk *header, body* dan *trailer* dari dokumen PDF.

Selanjutnya dalam proses dekripsi dilakukan menggunakan kunci privat x dan p untuk mendekripsi a dan b menjadi *plaintext* m menggunakan rumus (4) dan (5). Dalam perhitungan ini menggunakan kunci private (x,p) = (439, 457) dan bilangan prima (p) = 457. Untuk perhitungan detailnya bisa dilihat pada Tabel 2 berikut ini.

Volume 1, Oktober 2017
ISSN: 2614-1205

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 = $b.a^{p-1-x} \bmod p$ = $308.14^{457-1-439} \bmod 457$ = $61.14^{17} \bmod 457$ = 37	37	%
2	m2	m2 = $b.a^{p-1-x} \bmod p$ = $246.14^{457-1-439} \bmod 457$ = $246.14^{17} \bmod 457$ = 80	80	P
3	m3	m3 = $b.a^{p-1-x} \bmod p$ = $331.14^{457-1-439} \bmod 457$ = $331.14^{17} \bmod 457$ = 68	68	D
4	m4	m4 = $b.a^{p-1-x} \bmod p$ = $101.14^{457-1-439} \bmod 457$ = $101.14^{17} \bmod 457$ = 70	70	F

Dari hasi dekripsi pada Tabel 2 diperoleh hasil bahwa *plaintext(m) m1*=37, *m2*=80, *m3*=68, *m4*=70. Adapun hasil konversi dengan karakter ASCII menghasilkan *plaintext* berupa :

m1 = 37 = %
m2 = 80 = P
m3 = 68 = D
m4 = 70 = F

4. SIMPULAN

Berdasarkan uji coba yang telah dilakukan terhadap dokumen PDF yang terenkripsi baru dilaksanakan sebatas pada *ciphertext* yang berupa file PDF secara utuh termasuk *header* dokumen tersebut. Oleh karena itu, analisa penyerangan masih belum bisa dilakukan secara menyeluruh. Sebagai bahan analisa awal dengan menggunakan aplikasi *cracker* yang digunakan, *ciphertext* belum bisa terpecahkan baik kunci maupun *plaintext*-nya. Dokumen PDF yang terenkripsi dianggap seolah file dokumen tersebut rusak, seperti yang sudah dijelaskan di atas hal ini terjadi karena dokumen dienkripsi secara keseluruhan termasuk *header, body* dan *trailer* dari dokumen PDF. Sesuai dengan hasil uji coba tersebut, efektifitas algoritma ElGamal terhadap dokumen PDF dinyatakan cukup aman, dikarenakan penggunakan beberapa bilangan acak prima yang digunakan dalam proses enkripsi.

5. REFERENSI

[1] Sutikno, Marsela dan Karima, A. 2016. Implementasi Vernam Cipher dan Steganografi End Of File (Eof) Untuk Enkripsi Pesan PDF. *TECHNOCOM, hal:* 254-268.
[2] Vicky. 2007. Mengenal PDF dan Cara Mengconvert File Word To PDF dengan Microsoft Word. (Online), (http://belajar-komputer-mu.com/mengenal-PDF-dan-cara-mengconvert-file-word-to-PDF-dengan-microsoft-word-2007/, diakses 26 Semptember 2017).
[3] Karima, Aisyatul, Handoko, dkk. 2017. *Pemfaktoran Bilangan* Prima pada Algoritma ElGamal untuk Keamanan Dokumen PDF. *Jurnal Nasional Teknik Elektro dan Teknologi Informasi – JNTETI.*. Vol. 6.
[4] Munir, R. 2006. *Kriptografi.* Informatika Bandung, Bandung.
[5] Massandy, D. T. 2009. Algoritma Elgamal dalam Pengamanan Pesan Rahasia. *Jurnal Informatika.*

Volume 1, Oktober 2017
ISSN: 2614-1205

[6] Karima, A., Saputro, A. 2016. Pembangkitan Kunci pada Algoritma Asimetris ElGamal untuk Meningkatkan Keamanan Data bertipe .docx. *Jurnal SISFOTENIKA*. Vol. 6 (2): 170-181.

[7] Zaelvina, A. 2012. *Perancangan Aplikasi Pembelajaran Kriptografi Kunci Publik Elgamal Untuk Mahasiswa.* Jurnal Dunia Teknologi Informasi. Vol. 1(1): 56-62.

[8] Pratama, S. F. 2009. *Algoritma ElGamal untuk Keamanan Aplikasi Email.* Bandung.

[9] Kang, Z. Q., Wei, K. L. 2015. New Result on The Hardness of ElGamal and RSA bits basing Binary Expansions. *IEEE, International Conference on Information Science and Control Engineering.* Vol. 2.

[10]Unknown. (Online) (http://www.simpoPDF.com/resource/PDF-file-structure.html).