IMPLEMENTASI ALGORITMA KRIPTOGRAFI ELGAMAL UNTUK KEAMANAN PESAN (*MESSAGE SECURITY*)

Nur Rochmat[*)], R.Rizal Isnanto, and Maman Somantri

Jurusan Teknik Elektro, Universitas Diponegoro Semarang
Jl. Prof. Sudharto, SH, Kampus UNDIP Tembalang, Semarang 50275, Indonesia

[*)]*E-mail: whitenbluesky@yahoo.com*

Abstrak

Berdasarkan konsep algoritma kriptografi ElGamal, pada penelitian ini akan mengimplementasikan algoritma tersebut kedalam suatu aplikasi simulasi surat elektronik sederhana. Program aplikasi diawali dengan proses pendaftaran, yang didalamnya terdapat proses pembuatan kunci. Setelah proses pendaftaran selesai, kunci akan disimpan pada basis data. Kunci tersebut terdiri dari kunci publik dan kunci pribadi. Kunci publik digunakan untuk mengenkripsi pesan pada proses tulis pesan. Kunci ini dapat diakses secara bebas. Kunci pribadi digunakan untuk mendekripsi pesan pada proses baca pesan. Kunci ini hanya bisa diakses oleh penerima pesan yang dituju. Pada program aplikasi dilakukan proses pembuatan kunci, pengujian enkripsi dan dekripsi terhadap suatu pesan agar langkah-langkah managemen kunci dan hasil yang terjadi pada proses enkripsi dan dekripsi dengan algoritma ElGamal dapat diketahui. Pada percobaan lain dilakukan pengujian dengan model serangan seperti ciphertext only attack, know-plain attack dan brute force attack untuk mengetahui kekuatan algoritma ElGamal dari model serangan tersebut. Berdasarkan hasil pengujian yang telah dilakukan diperoleh kesimpulan bahwa algoritma ElGamal sangat baik untuk mengatasi masalah manajemen distribusi kunci. Selain itu, algoritma ini juga memiliki ketahanan yang baik terhadap metode serangan seperti ciphertext-only attack, know-plain attack dan brute force attack.

Kata kunci: kriptografi, algoritma asimetris, elgamal, enkripsi, dekripsi, *serangan kriptografi*

Abstract

Based on the concept of ElGamal cryptographic algorithms, the research will be to implement the algorithm into a simple electronic mail simulation application. The application program begins with the registration process, there are key generating process in it. Once the registration process is completed, the key will be stored in the database. The key consists of a public key and a private key. The public key is used to encrypt messages in the process of writing a message. This key is publicly accessible.The private key is used to decrypt messages on the message read. This key can only be accessed by the intended recipient. In the application program the process of making key be done, encryption and decryption of a message so the key management measures and outcomes that occur in the process of encryption and decryption with ElGamal algorithm can be known. In another experiment conducted tests with models such as the ciphertext only attack, know-plain attack and brute force attack to determine the strength of the ElGamal algorithm from those kind of attack models.Based on the results of testing that has been done can be concluded that the ElGamal algorithm is very good to solve the problem of key distribution management. In addition, the algorithm also has good resistance to attack methods such as ciphertext-only attack, know-plain attack and brute force attack.

Keywords: Cryptography, Asymmetric algorithms, ElGamal, Encryption, Decryption, Cryptographic attacks.

1. Pendahuluan

Pendistribusian informasi sangat rentan terhadap serangan dan gangguan dari pihak lain. Sebagai contoh keberhasilan KPK (Komisi Pemberantasan Korupsi) dalam melakukan penyadapan informasi pribadi milik beberapa pejabat negara, menunjukkan bahwa pendistribusian pesan yang berisi informasi masih sangat rentan dari gangguan pihak lain.

Salah satu cara pengamanan dalam penyampaian informasi adalah dengan melakukan pengkodean terhadap informasi tersebut. Teknik pengkodean ini dikenal dengan nama kriptografi. Kriptografi merupakan seni dan ilmu untuk menjaga kerahasiaan berita. Prinsip dasar

pengamanan kriptografi adalah proses enkripsi dan dekripsi. Enkripsi mengubah pesan asli (*plaintext*) menjadi pesan terkodekan (*chipertext*).

Kriptografi berdasarkan kunci terdiri dari kriptografi kunci simetri (mennggunakan kunci yang sama untuk enkripsi maupun untuk dekripsi) dan kriptografi kunci asimetri (menggunakan kunci yang berbeda untuk enkripsi maupun untuk dekripsi). Masalah distribusi kunci yang muncul pada kriptografi simetri dapat diatasi dengan penggunaan kriptografi asimetris. Kunci pada kriptografi asimetris terdiri dari kunci publik (digunakan pada proses enkripsi, kunci ini tidak bersifat rahasia sehingga setiap orang boleh melihat dan menggunakannya) sedangkan kunci yang lainnya disebut kunci pribadi (digunakan pada proses dekripsi, kunci ini bersifat rahasia dan hanya digunakan oleh penerima pesan).

Algoritma kriptografi ElGamal merupakan salah satu algoritma kunci asimetris yang didasarkan pada logaritma diskrit. Masalah logaritma diskrit adalah dengan memperhatikan hal berikut. Jika diberikan suatu bilangan a, maka menghitung $b \equiv \alpha^a$ (*mod p*) adalah mudah, tetapi jika diberikan suatu bilangan b, maka untuk menemukan a sehingga $b \equiv \alpha^a$ (*mod p*) adalah permasalahan yang sulit. Algoritma ini dikembangkan pertama kali oleh ilmuan Mesir Taher ElGamal pada tahun 1985.

Dalam penelitian ini diambil studi kasus pada suatu perangkat lunak simulasi surat elektronik sederhana yang menggunakan konsep algoritma kriptografi ElGamal. Pada perangkat lunak ini pembuatan kunci dilakukan pada proses pendaftaran, enkripsi dilakukan pada proses tulis pesan sedangkan deskripsi dilakukan pada proses baca pesan. Pembahasan ini diharapkan dapat digunakan untuk lebih mejelaskan proses yang ada pada algoritma kriptografi ElGamal dan pengimplementasiannya dalam suatu perangkat lunak lain ataupun penelitian lebih lanjut tentang algoritma kriptografi ElGamal.

2. Metode

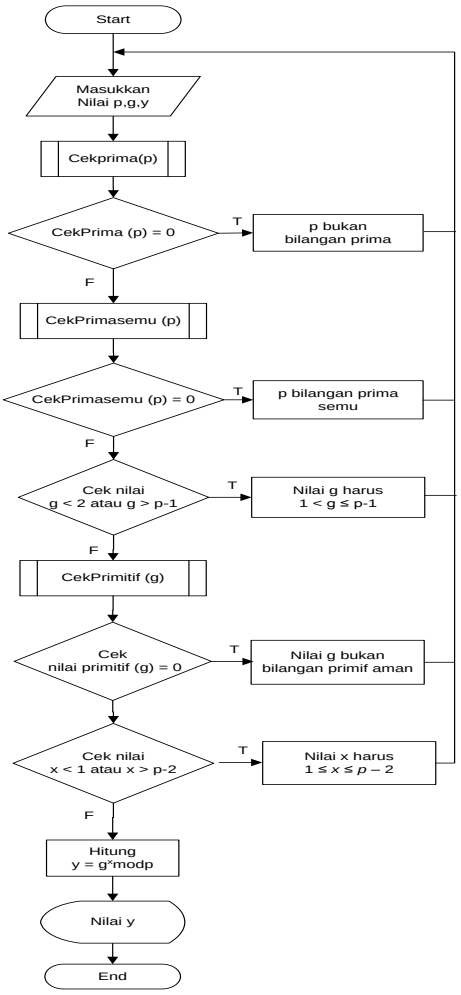
Algoritma ElGamal ditemukan oleh ilmuwan Mesir, yaitu Taher ElGamal pada tahun 1985, merupakan algoritma kriptografi kunci publik. Algoritma ElGamal terdiri atas tiga proses, yaitu proses pembentukan kunci, enkripsi, dan dekripsi. Algoritma ElGamal mendasarkan kekuatannya pada fakta matematis kesulitan menghitung logaritma diskret.

2.1 Proses Pembentukan Kunci

Proses pembentukan kunci merupakan proses penentuan suatu bilangan yang kemudian akan digunakan sebagai kunci pada proses enkripsi dan dekripsi pesan. Kunci untuk enkripsi terdiri dari nilai *p, g, y* sedangkan kunci untuk dekripsi terdiri dari nilai *x, p*. Masing-masing nilai mempunyai persaratan yang harus dipenuhi. Langkah-langkah dalam pembuatan kunci adalah sebagai berikut.

i. Pilih sembarang bilangan prima *p*, dengan syarat *p* >255.
   a. Cek bilangan prima.
   b. Cek prima semu (cek bilangan komposit). $\frac{p-1}{2} = x$, jika $x \in$ bilangan prima maka $p \in$ bilangan prima
ii. Pilih bilangan elemen primitif *g* dengan syarat $1 < g \leq p\text{-}1$.
   $g^2 \bmod p \neq 1$ dan $g^{\frac{p-1}{2}} \bmod p \neq 1$.
iii. Pilih bilangan acak *x*, dengan syarat $1 \leq x \leq p-2$.
iv. Hitung $y = gx \bmod p$.

Dari langkah-langkah diatas akan diperoleh kunci publik yang digunakan untuk enkripsi adalah nilai *p, g, y* dan kunci pribadi yang digunakan untuk dekripsi adalah nilai *x, p*. Nilai *p, g, y* bersifat tidak rahasia sedangkan nilai *x* bersifat rahasia. Gambar 1 menunjukkan bagan alir (*flowchart*) pembentukan kunci.



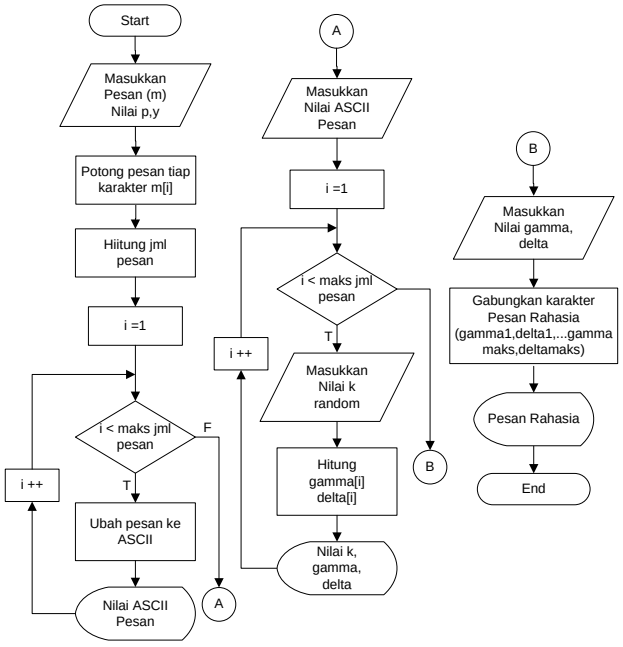
Gambar 1. *Flowchart* pembentukan kunci

2.2 Proses Enkripsi

Proses enkripsi merupakan proses mengubah pesan asli (*plaintext*) menjadi pesan rahasia (*ciphertext*). Pada proses ini digunakan kunci publik (*p, g, y*). Langkah-langkah dalam mengenkripsi pesan adalah sebagai berikut.

i. Potong *plaintext* menjadi blok-blok m_1, m_2, …, nilai setiap blok di dalam selang $[0, p-1]$.
ii. Ubah nilai blok pesan ke dalam nilai ASCII.
iii. Pilih bilangan acak k, dengan syarat $1 \leq k \leq p-2$.
iv. Setiap blok m dienkripsi dengan rumus sebagai berikut.
$gamma\ (\gamma) = g^k \bmod p.$ (1)
$delta\ (\delta) = y^k m \bmod p.$ (2)
v. Susun *ciphertext* dengan urutan $\gamma_1,\delta_1,\ \gamma_2,\delta_2,...,\gamma_n,\delta_n$.

Pasangan γ dan δ adalah *cipherteks* untuk blok pesan *m*. Hasil yang didapat dari proses enkripsi berupa pesan rahasia (*ciphertext*). Gambar 2 menunjukkan bagan alir (*flowchart*) proses enkripsi pesan.



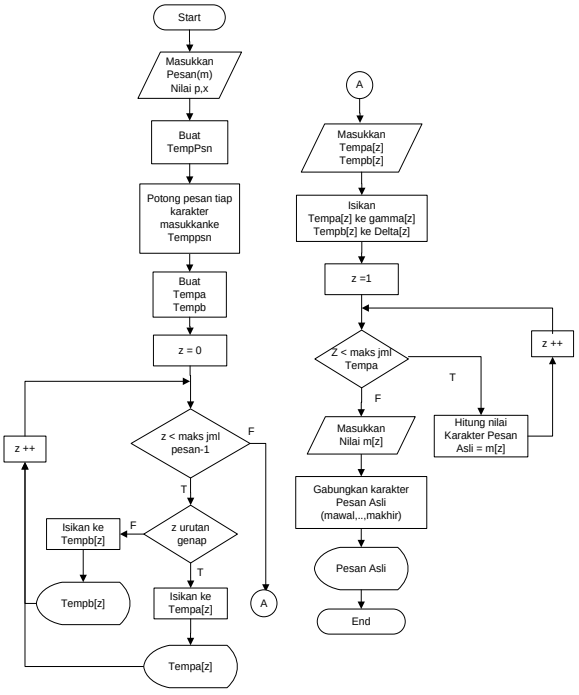
Gambar 2. *Flowchart* enkripsi pesan

2.3 Proses Dekripsi

Proses dekripsi merupakan proses mengubah pesan rahasia (*ciphertext*) menjadi pesan asli (*plaintext*). Pada proses ini digunakan kunci pribadi (*x, p*). Langkah-langkah dalam mendekripsi pesan adalah sebagai berikut.

i. Penentuan nilai *gamma* dan *delta*. Nilai *gamma* (γ) diperoleh dari *ciphertext* dengan urutan ganjil sedangkan *delta* (δ) dengan urutan genap.
ii. Hitung *plaintext m* dengan persamaan rumus berikut.
$m = \delta.\ \gamma^{(p-1-x)} \bmod p$ (3)
iii. Ubah nilai *m* yang didapat kedalam nilai ASCII.
iv. Susun *plaintext* dengan urutan $m_1,m_2,...,m_n$.

Hasil yang didapat dari proses enkripsi berupa pesan asli (*plaintext*). Gambar 3 menunjukkan bagan alir (*flowchart*) proses dekripsi pesan.



Gambar 3. *Flowchart* dekripsi pesan

3. Hasil dan Analisa

3.1 Pengujian Proses Enkripsi dan Dekripsi

Pengujian dilakukan dengan cara membandingkan hasil proses enkripsi dan dekripsi dari program aplikasi yang telah dibuat dengan hasil penghitungan enkripsi dan dekripsi secara manual. Data yang digunakan untuk pengujian ini adalah seperti pada Tabel 1.

Tabel 1. Data Pengujian Program.

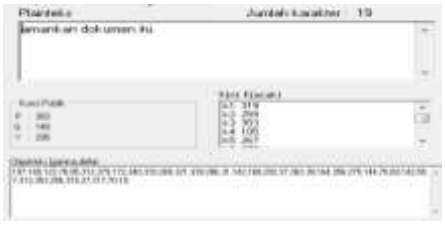
Keterangan	Nilai
Pesan	amankan dokumen itu
nilai (*p,g,y,x*)	(383, 148, 295, 338)
nilai *k*	k_1 = 319, k_2 = 259, k_3 = 353, k_4 = 105, k_5 = 267, k_6 = 279, k_7 = 190, k_8 = 252, k_9 = 60, k_{10} = 87, k_{11} = 360, k_{12} = 139, k_{13} = 48, k_{14} = 376, k_{15} = 116, k_{16} = 20, k_{17} = 38, k_{18} = 330, k_{19} = 211

3.1.1 Pengujian Proses Enkripsi

A. Proses enkripsi dengan program aplikasi.

Hasil dari proses ini akan berupa pesan rahasia (*ciphertext*). Gambar 4 menunjukkan proses enkripsi yang ada pada form Tulis pesan. Sedangkan Tabel 2 untuk

lebih memperjelas data proses enkripsi yang terjadi pada program aplikasi.



Gambar 4 Proses enkripsi pesan pada program aplikasi

Tabel 2. Proses enkripsi pesan pada program aplikasi

Keterangan	Nilai
Plaintext	amankan dokumen itu
Kunci publik (*p,g,y*)	(383, 148, 295)
nilai *k*	k_1 = 319, k_2 = 259, k_3 = 353, k_4 = 105, k_5 = 267, k_6 = 279, k_7 = 190, k_8 = 252, k_9 = 60, k_{10} = 87, k_{11}= 360, k_{12} = 139, k_{13} = 48, k_{14}= 376, k_{15} = 116, k_{16} = 20, k_{17} = 38, k_{18} = 330, k_{19} = 211
Hasil Enkripsi (ciphertext)	197,140,122,78,85,312,379,172,340,310,269, 321,339,286,31,142,168,292,37,362,38,164, 356,275,144,70,69,142,50,7,313,353,256,374,27,2 17,70,10,

B. Proses Enkripsi dengan cara manual

Langkah-langkah penyelesaian proses enkripsi secara manual adalah sebagai berikut:
Diketahui :
Plaintext : "amankan dokumen itu"
Nilai p = 383, g = 148 dan y = 295.
Nilai k_1 = 319, k_2 = 259, k_3 = 353, k_4 = 105, k_5 = 267, k_6 = 279, k_7 = 190, k_8 = 252, k_9 = 60, k_{10} = 87, k_{11} = 360, k_{12} = 139, k_{13} = 48, k_{14} = 376, k_{15} = 116, k_{16} = 20, k_{17} = 38, k_{18} = 330, k_{19} = 211.

Jawab :

i. Ubah pesan asli (*plaintext*) ke dalam *ASCII*
   a = 97, m = 109, a = 97, n = 110, k =107, a =97, n =110, spasi = 32, d = 100, o = 111, k = 107, u = 117, m = 109, e = 101, n = 110, spasi = 32, i = 105, t = 116, u = 117.
   sehingga nilai pesan *ASCII* adalah sebagai berikut :
   m_1 = 97, m_2 = 109, m_3 = 97, m_4 = 110, m_5 = 107, m_6 = 97, m_7 = 110, m_8 = 32, m_9 = 100, m_{10} = 111, m_{11} = 107, m_{12} =117, m_{13} =109, m_{14} = 101, m_{15} = 110, m_{16} =32, m_{17} =105, m_{18} =116, m_{19} =117.
ii. Hitung *gamma* (γ) dengan rumus $\gamma = g^k \bmod p$.
   $\gamma_1 = 148^{319} \bmod 383 = 197$
   $\gamma_2 = 148^{259} \bmod 383 = 122$
   $\gamma_3 = 148^{353} \bmod 383 = 85$
   Dst...
   Hasil nilai γ_1 = 197, γ_2 = 122, γ_3 = 85, γ_4 = 379, γ_5 = 340, γ_6 = 269, γ_7 = 339, γ_8 = 31, γ_9 = 168, γ_{10} = 37, γ_{11} = 38, γ_{12} = 356, γ_{13} = 144, γ_{14} = 69, γ_{15} = 50, γ_{16} = 313, γ_{17} = 256, γ_{18} = 27, γ_{19} = 70.
iii. Hitung *delta* dengan rumus $\delta = y^k . m \bmod p$
   $\delta_1 = 295^{319} . 97 \bmod 383 = 140$
   $\delta_2 = 295^{259} . 109 \bmod 383 = 78$
   $\delta_3 = 295^{353} . 97 \bmod 383 = 312$
   Dst...
   Hasil nilai δ_1 = 140, δ_2 = 78, δ_3 = 312, δ_4 = 172, $\delta_5 = 3_{10}$, δ_6 = 321, δ_7 = 286, δ_8 = 142, δ_9 = 292, δ_{10} = 362, δ_{11} = 164, δ_{12} = 275, δ_{13} = 70, δ_{14} = 142, δ_{15} = 7, δ_{16} = 353, δ_{17} = 374, $\delta_{18} = 2_{17}$, δ_{19} = 10.
iv. Susun hasil perhitungan *gamma* (γ) dan *delta* (δ)
   Ciphertext : 197, 140, 122, 78, 85, 312, 379, 172, 340, 310, 269, 321, 339, 286, 31, 142, 168, 292, 37, 362, 38, 164, 356, 275, 144, 70, 69, 142, 50, 7, 313, 353, 256, 374, 27, 217, 70, 10,

3.1.2 Pengujian Proses Dekripsi

A. Proses dekripsi dengan menggunakan aplikasi.

Gambar 5 menunjukkan proses dekripsi yang ada pada form Tulis pesan. Tabel 3 untuk lebih memperjelas data proses dekripsi yang terjadi pada program aplikasi.

Gambar 5 Proses dekripsi pesan pada program aplikasi

Tabel 3. Proses dekripsi pesan pada program aplikasi

Keterangan	Nilai
Pesan rahasia (ciphertext)	197,140,122,78,85,312,379,172,340, 310,269,321,339,286,31,142,168,292, 37,362,38,164,356,275,144,70,69,142, 50,7,313,353,256,374,27,217,70,10,
Kunci publik (*p,x*)	(383,338)
Hasil Dekripsi (plaintext)	amankan dokumen itu

B. Proses dekripsi dengan cara manual

Langkah-langkah penyelesaian proses dekripsi secara manual adalah sebagai berikut :
Diketahui :
Ciphertext : 197, 140, 122, 78, 85, 312, 379, 172, 340, 310, 269, 321, 339, 286, 31, 142, 168, 292, 37, 362, 38, 164, 356, 275, 144, 70, 69, 142, 50, 7, 313, 353, 256, 374, 27, 217, 70, 10,
Nilai p = 383, x = 338.
Jawab :

i. Pisahkan nilai *gamma* dan *delta* pada pesan rahasia (*ciphertext*).
   γ = Ciphertext urutan ganjil.
   δ = Ciphertext urutan genap.

Nilai gama γ1 = 197, γ2 = 122, γ3 = 85, γ4 = 379, γ5 = 340, γ6 = 269, γ7 = 339, γ8 = 31, γ9 = 168, γ10 = 37, γ11 = 38, γ12 = 356, γ13 = 144, γ14 = 69, γ15 = 50, γ16 = 313, γ17 = 256, γ18 = 27, γ19 = 70. Nilai delta δ1 = 140, δ2 = 78, δ3 = 312, δ4 = 172, δ5 = 310, δ6 = 321, δ7 = 286, δ8 = 142, δ9 = 292, δ10 = 362, δ11 = 164, δ12 = 275, δ13 = 70, δ14 = 142, δ15 = 7, δ16 = 353, δ17 = 374, δ18 = 217, δ19 = 10

ii. Hitung *m* (pesan asli) dengan rumus :
$m = \delta . \gamma^{(p-1-x)} \bmod p$
$m_1 = 140 . 197^{(383-1-338)} \bmod 383$
$= 140 . 197^{44} \bmod 383$
$= 97$
$m_2 = 78 . 122^{(383-1-338)} \bmod 383$
$= 78 . 122^{44} \bmod 383$
$= 109$
$m_3 = 312 . 85^{(383-1-338)} \bmod 383$
$= 312 . 85^{44} \bmod 383$
$= 97$
Dst..

Sehingga Hasil nya :
$m_1 = 97$, $m_2 = 109$, $m_3 = 97$, $m_4 = 110$, $m_5 = 107$, $m_6 = 97$, $m_7 = 110$, $m_8 = 32$, $m_9 = 100$, $m_{10} = 111$, $m_{11} = 107$, $m_{12} = 117$, $m_{13} = 109$, $m_{14} = 101$, $m_{15} = 110$, $m_{16} = 32$, $m_{17} = 105$, $m_{18} = 116$, $m_{19} = 117$

iii. Ubah *m* kedalam *ASCII*.
97 = a, 109 = m, 97 = a, 110 = n, 107 = k, 97 = a, 110 = n, 32 = spasi, 100 = d, 111 = o, 107 = k, 117 = u, 109 = m, 101 = e, 110 = n, 32 = spasi, 105 = i, 116 = t, 117 = u.

iv. Hasil dari penyusunan inilah yang merupakan pesan asli (*plaintext*) yang dihasilkan pada proses dekripsi. *plaintext*: "amankan dokumen itu".

Hasil proses perhitungan enkripsi dekripsi dengan program aplikasi dan secara manual adalah sama. Selain itu *plaintext* setelah dekripsi sama dengan nilai *plaintext* sebelum di enkripsi.

3.2 Pembahasan analisis keamanan Algoritma ElGamal

3.2.1 *Cipher only attack*

Dengan melihat percobaan enkripsi pesan pada Tabel 4.2. Dari hasil itu terlihat bahwa jumlah *ciphertext* yang dihasilkan tidak sama dengan jumlah *plaintext*. Hal ini terjadi dikarenakan *ciphertext* terdiri dari nilai *gamma* dan *delta*. Akibat proses tersebut menjadikan jumlah *ciphertext* lebih banyak dari jumlah *plaintext*. Dengan adanya perbedaan jumlah tersebut tentunya akan menyulitkan menebak *plaintext* sebenarnya meskipun kriptanalis memiliki *ciphertext*.

3.2.1 *Known-plain attack*

Untuk mengetahui keamanan ElGamal dari jenis serangan ini akan dilakukan tiga kali percobaan proses enkripsi pesan dengan *plaintext* dan kunci yang sama. Nilai *plaintext*: "amankan dokumen itu", dengan kunci publik (*p, g, y*) = (383, 148, 295).

Pada tiga kali percobaan dihasilkan data seperti terlihat pada Tabel 4.4. Dari tersebut dapat dilihat bahwa dengan *plaintext* dan kunci yang sama menghasilkan *ciphertext* yang berbeda. Hal ini dikarenakan adanya nilai *k* yang acak. Nilai *k* yang acak membuat nilai *gamma* dan *delta* selalu berubah sehingga *ciphertext* yang dihasilkan untuk setiap percobaan selalu berubah. Hal tersebut akan menyulitkan kriptanalis dalam mengkorelasikan (menemukan hubungan) antara plaintext dengan *ciphertext*, sehingga akan menyulitkan penemuan algoritma alternatif.

Tabel 4 Hasil Pengujian *known-plain attack*

No	*Plaintext*	Kunci (p, g, y)	*Ciphertext*
1.	amankan dokumen itu	383, 148, 295	197,140,122,78,85,312,379, 172,340,310,269,321,339, 286,31,142,168,292,37,362, 38,164,356,275,144,70,69, 142,50,7,313,353,256,374, 27,217,70,10,
2.	amankan dokumen itu	383 ,148, 295	118,39,275,178,233,302,244,19 6,252,287,283,379,126, 274,96,65,273,23,54,5,356, 78,242,60,170,125,170,288, 268,207,187,224,193,307, 154,243,205,198,
3.	amankan dokumen itu	383, 148, 295	142,127,292,271,86,341,39, 119,185,109,132,247,233, 149,239,263,174,28,258,47, 352,20,207,257,230,311,378,12, 119,17,115,102,134,308, 377,31,13,85,

3.2.2 Exhaustive *Attack* atau *Bruteforce attack*

Pengujian *bruteforce* dilakukan dengan mencoba semua kombinasi kunci yang digunakan untuk melakukan dekripsi (*p* dan *x*).

Percobaan akan dilakukan sebanyak tiga kali dengan panjang kunci berbeda pada *plaintext* yang sama. Pada setiap panjang kunci yang sama akan dilakukan sebanyak empat kali percobaan penyerangan dengan nilai *x* acak. *Plaintext* yang digunakan adalah: "Amankan dokumen rahasia itu dan jaga jangan sampai hilang, Serahkan pada Bp.Wardoyo hari senin besok jam 9 pagi di Auditorium Jl Prof Sudarto.".Data nilai kunci percobaan pada Tabel 5.

Tabel 5 Data nilai kunci percobaan *bruteforce attack.*

Percobaan	Nilai (*p, g, y*)	Nilai *x* Acak
1.	*p* : 383	x_1 : 9
	g : 148	x_2 : 40

	y : 295	x_3 : 250
		x_4 : 370
2.	p : 21599	x_1 :125
	g : 20543	x_2 :1897
	y : 17305	x_3 :17500
		x_4 : 21111
3.	p : 472254227	x_1 :1234
	g : 12618948	x_2 :1234567
	y : 168465791	x_3 : 213333737
		x_4 : 413373733

Dari percobaan dapat dibuat tabel waktu rata-rata untuk setiap satu kali percobaan serangan dengan *bruteforce attack.*,seperti dapat dilihat pada Tabel 6

Tabel 6 waktu rata rata untuk setiap satu kali percobaan serangan

Percobaan	Nilai x acak	Waktu serangan (dt)	Waktu rata-rata serangan Waktu serang ($x_1 + x_2 + x_3 + x_4$)/4
1.	x_1 : 9	0,050	42,5 milidetik
	x_2 : 40	0,040	
	x_3 : 250	0,030	
	x_4 : 370	0,050	
2.	x_1 : 125	0,040	37,5 milidetik
	x_2 : 1897	0,030	
	x_3 : 17500	0,040	
	x_4 : 21111	0,040	
3.	x_1 : 1234	0,031	49,25 milidetik
	x_2 : 1234567	0,040	
	x_3 : 213333737	0,040	
	x_4 : 413373733	0,086	

Pengujian *bruteforce* didasarkan pada jumlah karakter kunci *x*. Jumlah karakter kunci *x* ini akan menentukan banyaknya jumlah percobaan yang harus dilakukan untuk mendapatkan hasil (*plaintext*).

Tabel 7. Jumlah bit kunci *x* pada percobaan *bruteforce attack.*

Percobaan	Kunci x	Jumlah bit Kunci
1.	338	3 karakter kunci atau (24bit)
2.	20123	5 karakter kunci atau (40bit)
3.	293839738	9 karakter kunci atau (72bit)

Dengan menggunakan rumus maka jumlah kemungkinan kunci mungkin adalah :

$$\text{Jumlah kemungkinan kunci} = 2^{\text{bit kunci}}$$

i. Percobaan 1
Dengan data pada Tabel 7 maka jumlah bit kunci percobaan 1 adalah 24. Dengan rumus jumlah kemungkinan kunci adalah :
Jumlah kemungkinan kunci = 2^{24} = 16.777.216.
Apabila mencoba seluruh kemungkinan kunci, maka jumlah percobaan untuk mendapatkan hasil sama dengan jumlah kemungkinan kunci. Pada percobaan *bruteforce* untuk mendapatkan hasil setidaknya dilakukan separuh dari percobaan junlah kemungkinan kunci. Sehingga untuk mendapatkan hasil setidaknya dilakukan sebanyak : 0,5 x Jumlah kemungkinan kunci.
0,5 x 16.777.216 = 8.388.608 percobaan
Lihat Tabel 6, pada percobaan 1 tercatat waktu rata-rata yang diperlukan untuk melakukan setiap satu kali serangan adalah 42,5 milidetik atau 0,0425 detik. Dengan menggunakan waktu tersebut sebagai waktu yang dibutuhkan untuk setiap kali percobaan serangan, maka untuk mendapatkan hasil diperlukan waktu 0,0425 x banyak percobaan. Dengan demikian waktu yang diperlukan adalah :
0,0425 x 8.388.608 = 356.515,84 detik.

ii. Percobaan 2
Jumlah kemungkinan kunci = 2^{40} = 1.099.511.627.776
Banyak percobaan= 0,5 x Jumlah kemungkinan kunci.
= 0,5 x 1.099.511.627.776
=549.755.813.888 percobaan
waktu yang diperlukan= 0,0375 x banyak percobaan.
= 0,0375 x 549.755.813.888
= 20.615.843.020,8 detik.

iii. Percobaan 3
Jumlah kemungkinan kunci = 2^{72}
= 4.722.366.482.869.645.213.696
Banyak percobaan= 0,5 x Jumlah kemungkinan kunci.
= 0,5 x 4.722.366.482.869.645.213.696
= 2.361.183.241.434.822.606.848 percobaan
waktu yang diperlukan : 0,04925 x banyak percobaan
= 0,04925 x 2.361.183.241.434.822.606.848
= 116.288.274.640.665.013.387,264 detik.

Untuk lebih jelas data hasil perhitungan lama waktu yang diperlukan untuk melakukan *bruteforce* dapat dilihat pada Tabel8

Tabel 8 Data percobaan lama waktu *bruteforce attack.*

Kunci x	Jumlah bit	Lama waktu *bruteforce* (detik)
338	24	356.515,84
20123	40	20.615.843.020,8
293839738	72	116.288.274.640.665.013.387,264

Dari percobaan yang dilakukan semakin besar jumlah karakter nilai *x* yang diambil maka akan semakin lama dalam proses pemecahan secara *bruteforce*.

Hasil percobaan terlihat jumlah *ciphertext* adalah dua kali jumlah *plaintext*. Hal tersebut mengakibatkan panjang *ciphertext* lebih besar dibanding panjang *plaintext*. *Ciphertext* yang panjang akan membutuhkan waktu yang lebih lama saat proses pengiriman. Dengan melihat hal tersebut maka algoritma jenis ini kurang sesuai digunakan untuk jenis aplikasi yang berjalan *real time*.

4. Kesimpulan

Algoritma asimetri ElGamal sangat baik untuk mengatasi masalah manajemen distribusi kunci. Algoritma ElGamal termasuk algoritma yang baik (aman secara komputasi). Dengan jumlah *ciphertext* yang lebih banyak dari *plaintext* mengakibatkan faktor kerja (*work factor*) yang dibutuhkan untuk melakukan pemecahan *chipertext* menjadi lebih lama dibandingkan dengan algoritma lain yang hanya memiliki jumlah *ciphertext* yang sama dengan jumlah *plaintext*nya. Algoritma ini mempunyai ketahanan yang baik terhadap metode serangan *Ciphertext only attack* dan *Know plain attack*. Semakin panjang karakter kunci yang dipilih semakin lama waktu yang dibutuhkan dalam model serangan *brute force attack*.. Semakin panjang *plaintext* akan meningkatkan panjang *chipertext* yang dihasilkan. Dengan *chipertext* yang panjang akan memperlambat saat proses pengiriman, sehingga algoritma ElGamal ini kurang sesuai untuk jenis aplikasi yang berjalan *real time*.

Referensi

[1]. Ardhian. A, A. Achmad, M .Z. Riyanto., "Ancaman Keamanan Komunikasi dan Serangan Terhadap Kriptografi", http://sandi.math.web.id, 2008

[2]. A.Menezes, P.van Oorschot, and S.Vanstone, "Handbook of Applied Cryptography", CRCPress, New Jersey, 1996

[3]. Author1 A, Author2 B. Judul Buku. City: Publisher. Year.

[4]. Munir, Rinaldi, "Kriptografi",Informatika,Bandung, 2006.

[5]. Riyanto, M.Z," Pengamanan pesan rahasia menggunakan algoritma kriptografi ElGamal atas grup pergandaan Zp*". Jurusan Matematika FMIPA Universitas Gadjah Mada, Yogyakarta, 2007.

[6]. ..., "Penyadapan, Jurus Andalan KPK Jerat Pejabat Korup",Edisi 14 Desember 2009, JawaPos, Surabaya, 2009.