Implementasi Algoritma Kriptorafi Kunci Publik Cramer-Shoup dan Perbandingannya dengan Algoritma ElGamal

Mohamad Rivai Ramandhani-13511043
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13511043@std.stei.itb.ac.id

***Abstract*— Makalah ini membahas implementasi algoritma kriptografi kunci publik Cramer-Shoup dan ElGamal serta perbandingan kinerja keduanya. Algoritma Cramer-Shoup merupakan pengembangan dari algoritma ElGamal dari sisi keamanan, yaitu algoritma ini dapat menangani serangan *Choosen Cipher Attack* yang tidak dapat ditangani oleh algoritma ElGamal biasa. Berdasar eksperimen ditujukkan bahwa untuk meningkatkan keamanannya, algoritma Cramer-Shoup memiliki waktu ekripsi-dekripsi, panjang kunci dan ukuran pesan terenkripsi yang jauh lebih besar dibanding algoritma ElGamal.**

***Index Terms*—algoritma kriptografi kunci publik, cramer-shoup, elgamal, perbandingan kinerja.**

I. PENDAHULUAN

Saat ini, sudah banyak algoritma kriptografi kunci publik yang telah dikembangkan, masing-masing dengan kinerja dan tingkat keamanan yang berbeda-beda.Salah satunya adalah algoritma ElGamal. Algoritma ElGamal dikembangkan berdasarkan masalah logaritma diskrit. Contoh lain dari algoritma kriptografi kunci publik adalah algoritma Cramer-Shoup. Algoritma Cramer-Shoup sebenarrnya merupakan pengembangan dari algoritma ElGamal dari sisi keamanan. Algoritma Cramer-Shoup sudah dibuktian dapat menangani serangan *Choosen Cipher Attack* yang tidak dapat ditangani oleh algoritma ElGamal biasa[1]. Dengan adanya penambahan fitur tersebut, perlu dianalisis dampaknya terhadap kinerja algoritma, diantaranya yaitu terhadap kecepatan ekripsi dan dekripsi algoritma serta panjang kunci dan hasil pesan terenkripsi.

II. LANDASAN TEORETIS

A. Kriptografi Kunci Publik

Kriptografi kunci publik merupakan salah satu contoh dari kriptografi modern. Disebut juga sebagai kriptografi asimetris karena berbeda dengan kebalikannya yaitu kriptografi simetris, pada algoritma yang termasuk ke dalam kriptografi kunci publik ini digunakan kunci yang berbeda (asimetris) ketika melakukan enkripsi dan dekripsi. Pada saat enkripsi, pengirim pesan perlu mengenkripsi pesannya menggunakan kunci publik sang penerima pesan. Sementara dari sisi penerima, pesan yang diterima perlu didekripsi dengan kunci privat miliknya sendiri.

Penggunaan kunci yang berbeda ini dilakukan untuk mengatasi kesulitan pada algoritma kunci simetris yang mengharuskan kedua pihak (pengirim dan penerima) sepakat dengan kunci simetris yang digunakan pada saluran komunikasinya. Akibat penggunaan kunci yang sama, kunci perlu dipertukarkan dalam saluran komunikasi sehingga rentan untuk disadap oleh penyerang.

Sudah banyak algoritma kunci publik yang dikembangkan. Banyak algoritma kunci publik digunakan dalam sistem *hybrid*. Algoritma kriptografi kunci publik cenderung lebih mahal dari segi komputasi daripada algoritma kriptografi kunci simetris, terutama untuk pesan berukuran yang besar. Pada sistem *hybrid*, pesan dienkripsi menggunakan algoritma kunci simetris. Namun seperti yang telah disebutkan sebelumnya, kunci simetris ini rawan disadap penyerang ketika dipertukarkan antara pengirim dan penerima pesan. Algoritma kriptografi kunci publik dapat digunakan untuk mengenkripsi kunci simetris (yang biasanya berukuran jauh lebih kecil daripada ukuran pesan) tersebut, sehingga walaupun berhasil disadap namun kunci simetris tersebut berada dalam keadaan terenkripsi.

B. Algoritma ElGamal

Algoritma yang dikembangkan oleh Taher El Gamal ini didasarkan pada masalah logaritma diskrit yaitu sulitnya untuk memfaktorkan bilangan prima sebagai keamanan dari algoritma ini[2].

Pembangkitan kunci pada algoritma ElGamal dilakukan sebagai berikut.

1. Dipilih sebuah bilangan prima sembarang, p.
2. Dipilih dua bilangan acak, g dan y, dengan syarat $g < p$ dan $1 <= x <= p-2$.
3. Dihitung nilai $y = g^x \bmod p$.
4. Kunci publik milik pengguna kemudian ditentukan dari tripel p, g dan y. Sementara kunci privat merupakan tupel p dan x.

Setelah kunci dibangkitkan, sebuah pesan m dienkripsi

dengan menghitung

$a = g^k \bmod p$, dan

$b = my^k \bmod p$

k merupakan sebuah bilangan acak yang memenuhi $1 <= k <= p - 2$. Pasangan a dan b merupakan cipherteks dari pesan m.

Kemudian untuk mendekripsi cipherteks berupa pasangan a dan b, dilakukan dengan menghitung

$(a^x)^{-1} = a^{p-x-1} \bmod p$, untuk kemudian mendapatkan

$m = b/a^x \bmod p = b(a^x)^{-1} \bmod p$

C. Algoritma Cramer-Shoup

Algoritma ini dikembangkan oleh Ronald Cramer dan Victor Shoup dengan tujuan untuk memberikan perlindungan dari serangan *Chosen Cipher Text* (CCA)[1]. CCA dilakukan peneyerang untuk memperoleh kunci yang sebelumnya tidak diketahui penyerang dengan memilih sebuah atau beberapa cipherteks serta mendapatkan hasil dekripsinya.

Detil algoritma Cramer-Shoup sangat mirip dengan algoritma ElGamal. Pembangkitan kunci pada algoritma Cramer-Shoup dapat dilakukan sebagai berikut.

1. Dipilih sebuah bilangan prima sembarang, p.
2. Dipilih dua bilangan acak, g_1 dan g_2, dengan syarat keduanya kurang dari p.
3. Dihitung $c = g_1^{x1} g_2^{x2}$, $d = g_1^{y1} g_2^{y2}$, dan $h = g_1^z$
4. Secara rahasia dipilih juga empat bilangan acak, x_1, x_2, y_1, y_2, dan z, dengan syarat masing-masing nilainya merupakan nilai yang sah dari hasil fungsi *hash* yang digunakan.
5. Kunci publik milik pengguna kemudian ditentukan sebagai nilai-nilai p, g_1, g_2, c, d, dan h. Sementara kunci privat merupakan nilai-nilai p, x_1, x_2, y_1, y_2, dan z.

Setelah kunci dibangkitkan, sebuah pesan m dienkripsi dengan menghitung nilai-nilai berikut.

$u_1 = g_1^k$,

$u_2 = g_2^k$,

$e = h^k m$,

$a = \mathrm{H}(u_1, u_2, e)$, dan

$v = c^k d^{ka}$

k merupakan sebuah bilangan acak dalam rentang nilai yang valid dari fungsi *hash* H(). Nilai-nilai hasil perhitungan ini merupakan cipherteks dari pesan m.

Sebelum sebuah cipherteks berupa nilai-nilai (u_1,u_2,e,v) didekripsi, perlu dilakukan validasi dengan menguji

$u_1^{x1+y1a} u_2^{x2+y2a} = v$

Jika uji tersebut gagal, cipherteks akan ditolak sebelum didekripsi. Namun jika uji tersebut valid, dilakukan dekripsi pesan m sebagai berikut.

$m = e/u^{z1}$

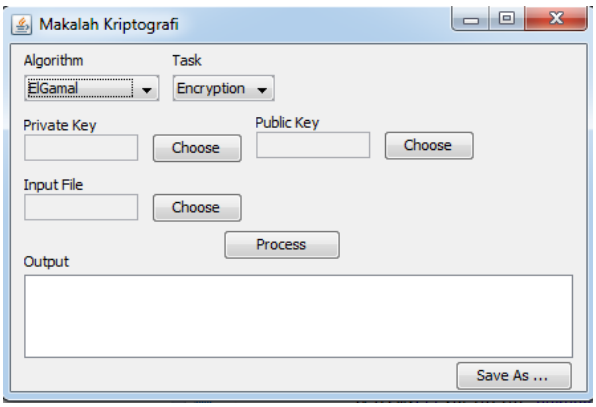
Algoritma Cramer-Shoup memiliki beberapa varian lain seperti yang dibahas pada makalah yang ditulis Cramer dan Shoup, diantaranya varian yang tidak menggunkan fungsi *hash*.

III. PEMBAHASAN

A. Implementasi Algoritma

Implementasi algoritma dilakukan dalam bahasa pemrograman Java berdasarkan teori yang telah dijabarkan pada bagian sebelumnya. Komputasi matematis dilakukan dengan bantuan *library* BigInteger yang ada pada Java. Fungsi *hash* yang digunakan untuk enkripsi pada algoritma Cramer-Shoup menggunakan fungsi SHA-1 yang ada di Java. Implementasi dilakukan pada lingkungan sistem operasi Microsoft Windows 7 64-bit dengan prosesor Intel Core i-5 dan RAM 2GB.

Antarmuka program dibuat dengan bantuan *library* Swing. Antarmuka program hasil implementasi kedua algoritma tersebut dapat terlihat pada Gambar III.1.



Gambar III.1 Antarmuka program

Implementasi tidak memasukkan bagian pembangkitan kunci secara otomatis. Kunci yang digunakan pada program dibaca dari *file* teks dengan format tertentu.

Eksperimen pada makalah ini menggunakan set kunci yang telah didefinisikan oleh penulis. Nilai-nilai pada set kunci publik dan kunci privat ini merupakan nilai-nilai BigInteger dengan panjang 64-bit yang dibangkitkan secara acak namun tetap memenuhi syarat kunci untuk masing-masing algoritma. Set kunci yang digunakan dapat dilihat pada Tabel III.1 dan Tabel III.2.

p	11699203708089531239
g1	10678113006897097894
g2	9210324319079460628
c	6213798760962502716
d	4046682160657159089
h	896838330658136232
x1	5639322931354459473
x2	11315444699813292979
y1	4789747378515615201
y2	10465133005470727572
z	2239822594638368540

Tabel III.1 Nilai-nilai kunci algoritma Cramer-Shoup

p	17152311739627521283
g	4657315399987939763
y	7902122913963261004
x	5032015948664434768

Tabel III.2 Nilai-nilai kunci algoritma ElGamal

Dari set kunci yang didefinsikan ini, terlihat bahwa isi set kunci untuk algoritma Cramer-Shoup jauh lebih banyak dibanding algoritma ElGamal. Hal ini tentunya mengakibatkan algoritma Cramer-Shoup membutuhkan ruang penyimpanan yang lebih besar untuk kuncinya.

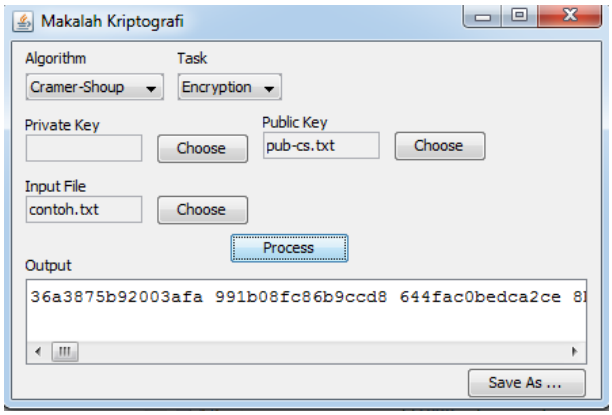
Set kunci akan digunakan untuk semua proses enkripsi dan dekripsi pada makalah ini. Hasil implementasi algoritma berhasil digunakan untuk melakukan enkripsi dan dekripsi pesan teks. Hasil enkripsi menghasilkan cipherteks yang disusun sebagai kumpulan BigInteger pada format heksadesimal yang dipisahkan oleh spasi.

Hasil enkripsi dengan algoritma Cramer-Shoup untuk pesan teks "Contoh pesan." dapat dilihat pada Gambar III.3. Sementara hasil enkripsi dengan algoritma ElGamal dapat dilihat pada gambar III.4. Dekripsi cipherteks dengan kunci privat yang tepat juga berhasil menghasilkan kembali pesan yang asli baik dengan algoritma Cramer-Shoup maupun ElGamal.

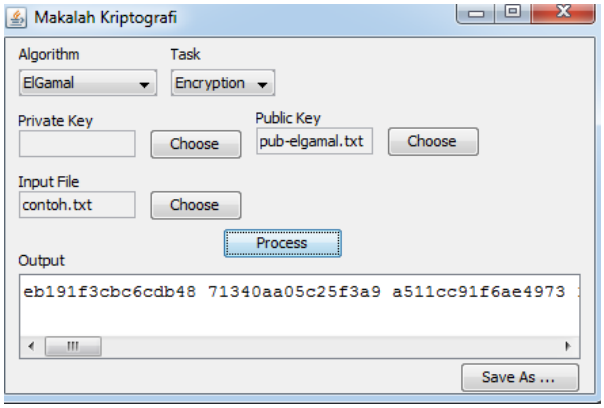
Kinerja program juga dicatat dalam *log* seperti berikut.

```
Encryption time: 40
Output size: 880 bytes
```

Log tersebut digunakan untuk mengukur kinerja setiap proses yang dilakukan dalam program yang telah diimplementasikan.



Gambar III.2 Enkripsi dengan Cramer-Shoup



Gambar III.3 Enkripsi dengan ElGamal

B. Analisis Eksperimen

Pada bagian ini dibahas mengenai hasil percobaan enkripsi dan dekripsi pesan menggunakan algoritma ElGamal dan Cramer-Shoup yang diimplementasikan seperti yang telah ditunjukkan pada bagian sebelumnya. Pada makalah ini hanya dilakukan percobaan dengan pesan berupa *file* teks. Dipilih tiga *file* teks yang dimaksud dengan ukuran berbeda-beda seperti tertera pada Tabel III.3.

Pesan1	29
Pesan2	18176
Pesan3	164700

Tabel III.3 Ukuran masing-masing pesan uji dalam *byte*

Analisis akan dilakukan dengan membandingkan hasil dari proses enkripsi-dekripsi ketiga *file* teks tersebut dengan algoritma ElGamal dan Cramer-Shoup. Kunci yang digunakan untuk enkripsi dekripsi adalah set kunci yang digunakan pada bagian sebelumnya. Perbandingan kecepatan enkripsi dan dekripsi kedua algoritma dapat dilihat pada Tabel III.4 dan Tabel III.5 secara berurutan. Sementara Tabel III.6 berisi rangkuman perbandingan ukuran hasil enkripsi menggunakan kedua algoritma.

	Cramer-Shoup	**ElGamal**
Pesan1	32	17
Pesan2	1466	577
Pesan3	9919	2886

Tabel III.4 Perbandingan kecepatan enkripsi kedua algoritma dalam milisekon

	Cramer-Shoup	**ElGamal**
Pesan1	0	0
Pesan2	1482	344
Pesan3	10296	1560

Tabel III.5 Perbandingan kecepatan dekripsi kedua algoritma dalam milisekon

	Cramer-Shoup	ElGamal
Pesan1	700	224
Pesan2	1228304	615396
Pesan3	11130502	5576419

Tabel III.6 Perbandingan ukuran hasil pesan terenkripsi kedua algoritma dalam byte

Dari segi kecepatan terlihat bahwa algoritma ElGamal memiliki kecepatan dekripsi yang lebih cepat dibanding enkripsinya. Sementara pada algoritma Cramer-Shoup ternyata waktu enkripsi dan dekripsi tidak berbeda jauh. Jika dibandingkan, algoritma Cramer-Shoup cenderung memiliki waktu yang lebih lama baik untuk enkripsi maupun dekripsi. Hal ini karena meskipun kedua algoritma ini sangat mirip, namun pada algoritma Cramer-Shoup ditambahkan beberapa perhitungan lain yang tidak ada pada algoritma ElGamal. Kinerja dari fungsi *hash* yang dipilih juga akan memengaruhi kecepatan dari enkripsi algoritma Cramer-Shoup. Pengaruh fugnsi *hash* terhadap kinerja algoritma Cramer-Shoup ini dapat dihilangkan dengan menggunakan varian algoritma yang tidak menggunakan fungsi hash.

Dari segi ukuran pesan terenkripsi, kedua algoritma menghasilkan pesan terenkripsi dengan ukuran yang jauh lebih besar dibanding pesan aslinya. Namun algoritma Cramer-Shoup terlihat menghasilkan pesan dengan ukuran yang lebih besar daripada algoritma ElGamal (hingga dua kalinya). Hal ini dikarenakan ketika sebuah pesan dienkripsi dengan algoritma Cramer-Shoup akan dihasilkan cipherteks yang memiliki empat komponen (nilai-nilai u_1, u_2, e, v). Sementara algoritma ElGamal menghasilkan cipherteks dengan jumlah komponen yang hanya setengahnya dari algoritma Cramer-Shoup (pasangan nilai a dan b).

Peningkatan ukuran pesan terlihat sangat siginifikan seiring dengan membesarnya ukuran pesan asli. Ukuran paling besar dihasilkan terlihat ketika mengenkripsi Pesan3 yang berukuran 164700 (17 KB) dengan algoritma Cramer-Shoup menjadi pesan berukuran 11130502 (11 MB). Peningkatan ukuran file ini menunjukkan bahwa kedua algoritma sebaiknya digunakan untuk pesan yang berukuran kecil.

V. KESIMPULAN DAN SARAN

Berdasarkan hasil implementasi dan analisis yang telah dibahas, dapat disimpulkan bahwa dengan meningkatkan keamanan terhadap serangan CCA, implementasi algoritma Cramer-Shoup perlu mengorbankan waktu enkripsi-dekripsi serta ukuran pesan terenkripsi. Baik waktu enkripsi-dekripsi maupun ukuran pesan terenkripsi hasil dari algorima Cramer-Shoup jauh lebih besar daripada algoritma ElGamal. Hal ini dapat dijadikan pertimbangan ketika akan memilih sebuah algoritma kriptografi diantara keduanya, apakah membutuhkan keamanan yang lebih tinggi atau waktu serta ukuran yang lebih efisien. Terlihat juga bahwa semakin besar pesan yang akan dienkripsi, waktu dan ukuran pesan terenkripsi juga meningkat dengan cukup signifikan. Hal ini merupakan karakteristik umum dari algoritma kriptografi kunci publik, dan mengakibatkan algoritma yang tergolong jenis ini lebih cocok digunakan untuk pesan-pesan berukuran kecil. Kedua algoritma yang dibahas termasuk ke dalam algoritma kriptografi kunci public karena menggunakan kunci asimetris ketika mengenkripsi dan dekripsi. Algoritma ElGamal dan Cramer-Shoup lebih cocok digunakan pada sistem hybrid sebagai algoritma yang digunakan untuk mengenkripsi kunci yang digunakan pada algoritma kriptografi simetris.

Pengembangan implementasi dapat dilakukan dengan melakukan percobaan terhadap jenis pesan lain selain teks, misalnya *file* biner. Implementasi algoritma Cramer-Shoup juga dapat dilakukan dengan menggunkanan fungsi *hash* yang lain selain SHA-1 atau bahkan menggunakan varian algoritma yang tidak menggunakan fungsi *hash* sama sekali.

REFERENSI

[1] Cramer, R., & Shoup, V. 1998. A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. Prosiding Crypto '98, LNCS 1462, pp. 13-25.

[2] Munir, R. 2015. Algoritma ElGamal (215). Bahan Kuliah IF4020 Kriptografi, Teknik Informatika, STEI, ITB.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 11 Mei 2015

Mohamad Rivai Ramandhani-13511043