Implementasi Algoritma Kriptografi Kunci Publik Okamoto-Uchiyama

Ezra Hizkia Nathanael (13510076)[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]*ezra.hizkia@itb.ac.id*

Makalah ini disusun sebagai salah satu prasayart kelulusan mata kuliah Kriptografi (IF3058) bagi mahasiswa program studi Teknik Informatika, Institut Teknologi Bandung semester genap tahun ajaran 2012/2013. Makalah ini digunakan sebagai pengganti nilai UAS. Tema utama dari makalah ini adalah eksperimen pengujian implementasi dari algoritma kriptografi kunci publik Okamoto-Uchiyama. Tujuannya adalah mencoba untuk mengimplementasikan algoritma tersebut dan melihat perbandingannya dengan algoritma ElGamal yang cukup mirip. Urutan penyusunan makalah ini terdiri dari Pendahuluan, Dasar Teori, Perancangan dan Implementasi, Pengujian, dan Kesimpulan.

***Kata Kunci*—algoritma, El Gamal, kriptografi kunci publik, Okamoto-Uchiyama cryptosystem**

I. PENDAHULUAN

Saat ini, dunia IT berkembang dengan sangat pesat. Tuntutan yang muncul menjadi sangat beragam, mulai dari kemangkusan algoritma hingga keamanan informasi. Untuk bidang keamanan sendiri, terdapat metode kriptografi untuk mengamankan pesan. Tujuan dari kriptografi sendiri adalah untuk menyamarkan suatu pesan-jelas (plaintext) menjadi sebuah pesan-sandi (ciphertext) sehingga tidak sembarang orang dapat mengetahui isi pesan tersebut tanpa memiliki kunci yang bersesuaian. Dalam dunia kriptografi sendiri dikenal dua macam jenis kriptografi, yakni kriptografi kunci privat dan kriptografi kunci publik. Untuk kriptografi kunci privat sendiri telah banyak dibahas sebelumnya melalui makalah pengganti nilai UTS.

Dalam kriptografi kunci publik sendiri, banyak sekali algoritma yang sudah beredar, sebut saja algoritma ElGamal, Diffie-Helman, RSA, Lamport, Rabin, Knapsack, dan masih banyak lagi. Salah satu algoritma yang tergolong baru dalam keluarga kriptografi kunci publik adalah algoritma Okamoto-Uchiyama yang dikembangkan oleh T.Okamoto dan S.Uchiyama pada tahun 1998. Meskipun saat ini sudah berumur 15 tahun, namun belum banyak dokumentasi dan implementasi terhadap algoritma ini sehingga penulis memutuskan untuk melakukan riset sederhanan terhadap algoritma ini.

Secara garis besar, algoritma Okamoto-Uchiyama memiliki kesamaan yang cukup besar dengan algoritma ElGamal. Kekuatan algoritma ini bersandar kepada sulitnya melakukan faktorisasi untuk bilangan prima dengan jumlah digit yang sangat besar. Keunggulan algoritma ini dibandingkan dengan algoritma ElGamal adalah jumlah bilangan prima yang lebih besar serta menggunakan dua buah bilangan prima sebagai kunci privat (algoritma ElGamal hanya menggunakan satu). Dengan demikian, diharapkan akan semakin sulit atau bahkan mustahil untuk melakukan faktorisasi tanpa memiliki kunci secara keseluruhan sehingga mempersulit proses kriptanalisis. Skema yang ditawarkan juga cukup sederhana namun membutuhkan tingkat ketelitian implementasi yang tidak sedikit.

II. DASAR TEORI

A. Kriptografi Kunci Publik

Kriptografi kunci publik juga dikenal sebagai kriptografi kunci asimetrik. Hal ini dikarenakan terdapat dua macam kunci yang digunakan, yakni kunci privat dan kunci publik. Sesuai namanya, kunci privat hanya dapat diakses oleh pemiliknya sedangkan kunci publik diletakkan di tempat umum dan siapa saja dapat mengaksesnya. Ciri khasnya terdapat pada pemrosesan bilangan yang berdasarkan kepada faktorisasi dan logaritma diskrit.

Kriptografi kunci publik ini mulai dikembangkan pada periode tahun 1970-an. Tujuan awal dari adanya kriptografi kunci publik ini adalah untuk memperkecil jumlah kunci yang dibutuhkan untuk melakukan proses kriptografi. Bayangkan saja, dengan menggunakan kriptografi kunci privat (kunci simetrik), maka jika kita memiliki seribu orang yang akan kita ajak berkomunikasi, maka kita harus menyimpan seribu kunci yang berbeda! Hal ini tentu saja tidak praktis, dan jumlah pertumbuhan kuncinya meningkat dengan sangat pesat. Apabila menggunakan kriptografi kunci publik maka kita cukup menyimpan kunci privat milik kita saja – dengan sangat hati-hati tentunya – dan mencari kunci publik dari orang/pihak yang berkomunikasi dengan kita.

Selain menekan jumlah kunci, keunggulan lainnya dari kriptografi kunci publik terkait dengan keamanan pengiriman kunci. Pada mode kriptografi kunci privat kita harus betul-betul menjamin kunci kita terkirim melalui jaringan yang aman, di mana jaringan yang tidak aman semakin memperbesar peluang adanya penyadapan di

tengah jalan. Bahkan, kita dapat mengirimkan kunci simetrik kita ini melalui jalur aman yang dibentuk berdasarkan mekanisme kriptografi kunci publik (bandingkan dengan mekanisme pertukaran kunci Diffie-Helman yang tidak menjadi cakupan dari makalah ini). Pasangan kunci privat dan publik ini tidak perlu diganti dalam waktu yang lama.

Mekanisme yang pada umumnya dilakukan adalah sebagai berikut: sebutlah dua orang bernama Alice dan Bob yang akan saling berkirim pesan. Apabila Alice akan mengirimkan pesan kepada Bob, maka ia akan mengunci pesan tersebut dengan menggunakan kunci publik Bob yang tersedia secara umum. Dengan demikian, pesan tersebut hanya dapat dibuka (didekripsi) dengan menggunakan kunci privat yang hanya dimiliki Bob, demikian pula sebaliknya.

Meskipun demikian, terdapat beberapa kelemahan pula dari model kriptografi kunci publik ini, di antaranya adalah waktu pemrosesan yang memakan waktu cukup lama mengingat dibutuhkan komputasi yang cukup rumit untuk melakukan faktorisasi atau logaritma diskrit yang menjadi ciri khas kriptografi ini. Selain waktu pemrosesan yang lama, ciphertext yang dihasilkan juga biasanya berukuran jauh lebih besar dibandingkan dengan plaintext semula. Isu mengenai otentikasi pengirim pesan juga menjadi persoalan tersendiri mengingat kunci publik dapat diakses oleh siapa saja.

Kriptografi kunci publik digunakan untuk berbagai keperluan, di antaranya adalah untuk mengenkripsi/dekripsi pesan, penandatanganan digital, dan mekanisme pertukaran kunci.

B. Algoritma Okamoto-Uchiyama

Algoritma kriptografi kunci publik ini tergolong ke dalam algoritma baru. Dikembangkan oleh T.Okamoto dan S.Uchiyama pada tahun 1998, algoritma berumur 15 tahun ini mengambil beberapa prinsip yang sangat mirip dengan algoritma kriptografi kunci publik lainnya, yakni algoritma ElGamal. Algoritma Okamoto-Uchiyama menyandarkan kekuatannya pada sulitnya melakukan faktorisasi dan operasi logaritma diskrit pada bilangan prima dengan digit yang sangat besar. Di sini, kunci privat dinyatakan dalam dua buah bilangan prima besar yakni p dan q. Kunci publik merupakan tuple tiga buah bilangan yakni n , g , dan h yang masing-masing memiliki prasyarat dan kegunaan tersendiri (akan dijelaskan pada bagian skema umum pembangkitan kunci). Bukti kekuatan algoritma ini adalah dalam pemilihan nilai n yang sangat besar, yakni $p^2 \times q$. Bandingkan dengan algoritma lain yang rata-rata nilai n hanya diperoleh dari hasil perkalian p dan q saja. Domain untuk algoritma ini terletak pada domain bilangan bulat $(\mathbb{Z}/n\mathbb{Z})^*$.

Skema algoritma Okamoto-Uchiyama memiliki dua properti yakni *homomorphic* dan *malleable*. Homomorphic artinya proses dekripsi dapat dilakukan pada bagian-bagian ciphertext dan akan menghasilkan keluaran yang bersesuaian dengan plaintext aslinya. Hal ini dapat dimanfaatkan untuk menjaga keamanan dari proses dekripsi/enkripsi, di mana proses tersebut dapat dilakukan oleh banyak orang (misalnya dalam rantai kerja suatu perusahaan) akan tetapi seseorang tidak mengetahui gambaran keseluruhan baik dari plaintext atau ciphertext (prinsipnya adalah satu tahu beberapa, bukan satu tahu semua). Malleable berarti adanya suatu kesempatan untuk mengubah file ciphertext menjadi file ciphertext lain yang memiliki keserupaan hasil dekripsi dengan file plaintext semula. Hal ini sebetulnya memiliki beberapa kelemahan seperti kemungkinan adanya pengubahan pesan, namun hal tersebut berada di luar cakupan makalah ini. Perlu diingat pula bahwa memang kebanyakan algoritma kriptografi memiliki properti malleable ini.

C. Algoritma ElGamal

Algoritma ElGamal cukup banyak disinggung di dalam makalah ini karena kedekatannya dengan algoritma Okamoto-Uchiyama, oleh karena itu pada bagian ini akan dijelaskan beberapa konsep mendasar terkait dengan algoritma kriptografi kunci-publik ElGamal.

Algoritma ini dibuat oleh Taher El-gamal pada tahun 1985. Beliau pertama kali menjabarkannya di dalam makalah yang berjudul "*A public-key cryptosystem and a signature scheme based on discrete logarithms*". Dapat dilihat di sini bahwa kekuatan algoritma ini juga bersandar kepada sulitnya melakukan faktorisasi dan menghitung logaritma diskrit. Beberapa elemen penting dari algoritma ini adalah sebuah bilangan prima p dan bilangan acak g yang bersifat publik, sebuah bilangan acak x yang menjadi salah satu bagian kunci privat, dan nilai y sebagai hasil komputasi p,g, dan x.

Pembangkitan kunci untuk algoritma ini adalah dengan membangkitkan tiga buah bilangan acak p, g, dan x di mana p haruslah merupakan bilangan prima. Syarat untuk g dan x adalah $g < p$ dan $1 \leq x \leq p-2$. . Kemudian, hitung nilai y yakni hasil dari komputasi $y = g^x \bmod p$. Yang dihasilkan dari proses ini adalah sebuah kunci publik yang merupakan tuple dari nilai $<y, g, p>$ dan kunci privat yang berupa pasangan (x, p).

Proses enkripsi sendiri dilakukan dengan cara membagi plaintext ke dalam blok-blok pesan. Setiap blok ini akan dienkripsi dengan rumus:

$$a = g^k \bmod p$$
$$b = y^k m \bmod p$$

Di mana k merupakan sebuah bilangan acak yang memenuhi syarat $1 \leq k \leq p-2$. Ciphertext dinyatakan dalam pasangan a dan b untuk setiap blok pesan. Hal ini menyebabkan ukuran ciphertext menjadi jauh lebih besar dibandingkan plaintextnya.

Untuk melakukan dekripsi, kita menggunakan kunci privat x yang kita miliki untuk melakukan komputasi terhadap persamaan $(a^x)^{-1} = a^{p-1-x} \bmod p$. Selanjutnya plaintext dapat diperoleh melalui persamaan $m = b(a^x)^{-1} \bmod p$.

III. PERANCANGAN DAN IMPLEMENTASI

A. Skema Umum Pembangkitan Kunci

Seperti yang sudah dijelaskan pada bagian dasar teori, terdapat dua buah kunci dalam algoritma Okamoto-Uchiyama, yakni kunci privat dan kunci publik. Langkah-langkah pembangkitan kunci adalah sebagai berikut:

1) Bangkitkan dua buah bilangan prima p dan q, kemudian hitung nilai n di mana $n = p^2q$. Semakin besar nilai p dan q tentunya akan membuat proses perhitungan faktorisasi dan logaritma diskrit menjadi semakin sulit.
2) Pilih sebuah bilangan $g \in (\mathbb{Z}/n\mathbb{Z})^*$ yang memenuhi persyaratan $g^p \neq 1 \ mod \ p^2$.
3) Hitung nilai $h = g^n \ mod \ n$.

Setelah mengikuti ketiga tahapan di atas maka diperoleh tuple kunci publik sebagai $< n, g, h >$ dan kunci privat adalah dua buah bilangan prima p dan q.

B. Skema Umum Proses Enkripsi/Dekripsi

Untuk melakukan proses enkripsi, kita ambil pesan yang akan dienkripsi. Perlu diingat sebelumnya bahwa domain kita adalah dalam bilangan bulat, maka pesan m yang akan kita enkripsi haruslah merupakan elemen pada $\mathbb{Z}/n\mathbb{Z}$. Selanjutnya, kita membangkitkan bilangan random r yang merupakan elemen dari bilangan bulat. Untuk menghasilkan ciphertext, maka lakukan komputasi sebagai berikut:

$$C = g^m h^r \ mod \ n$$

Adapun untuk melakukan proses dekripsi, kita sebelumnya harus mendefinisikan sebuah fungsi bantu yang dinyatakan sebagai $L(x) = \frac{x-1}{p}$. Dengan menggunakan bantuan fungsi ini maka proses dekripsi dinyatakan sebagai berikut:

$$m = \frac{L(C^{p-1} \ mod \ p^2)}{L(g^{p-1} \ mod \ p^2)} \ mod \ n$$

C. Keamanan Sistem

Keamanan dari keseluruhan pesan terdapat pada proses faktorisasi nilai n. Semakin besar nilai p dan q yang kita ambil tentunya akan menghasilkan nilai n yang semakin besar pula. Keamanan secara semantik dihasilkan dari asumsi yang menyatakan bahwa akan sulit untuk menentukan apakah sebuah elemen x pada domain bilangan bulat merupakan upa-bagian dari perpangkatan p. Hal ini menyatakan masalah yang serupa dengan *quadratic residuosity problem* dan *higher residuosity problem*.

D. Implementasi Rancangan

Seluruh rancangan di atas akan diimplementasikan ke dalam sebuah aplikasi sederhana yang ditulis dengan menggunakan bahasa pemrograman C# dan bantuan IDE Microsoft Visual Studio C# 2010. Untuk mempermudah digunakan library bantuan yakni Microsoft Solver Foundation untuk mempermudah beberapa bagian komputasi.

Khusus untuk Prime Number Generator menggunakan aplikasi sederhana yang ditulis dalam bahasa Java yang sudah pernah dibuat oleh penulis sebelumnya. Dilakukan sedikit modifikasi untuk membentuk dua buah bilangan prima besar dengan panjang 16-bit (untuk p dan q) sekaligus mengkomputasi nilai n.

Untuk aplikasi yang mengimplementasikan algoritma ElGamal diambil dari hasil Tugas Kecil 3 untuk mata kuliah IF3058 Kriptografi ini. Aplikasi ElGamal merupakan hasil karya dari penulis bersama dengan Sdr. Mufi Yanuar (13510106).

Skema dari program yang disusun dalam bahasa C# sendiri direncanakan akan terbagi ke dalam 3 bagian, yakni main program yang akan menjalankan program utama, kemudian sebuah form yang menyatakan layout aplikasi, dan sebuah file OUScheme yang menyatakan skema dari algoritma Okamoto-Uchiyama. OUScheme ini akan menyatakan proses enkripsi, dekripsi, dan functionL sebagai fungsi bantu untuk melakukan dekripsi.

Yang diimplementasikan adalah sebuah aplikasi yang terbagi ke dalam tiga bagian utama, yakni parameter, controller, dan ruang pesan. Pada bagian parameter terdapat lima buah tempat yang dapat digunakan untuk mengisi nilai $p, q, n, g, dan \ h$. Akan tetapi, di sini yang perlu user masukkan hanyalah nilai p dan q. Nilai n, g, dan h akan dibangkitkan secara otomatis oleh komputer dengan menekan tombol ‘Generate’.

Bagian controller merupakan tempat untuk memilih file yang akan dijadikan sebagai file input untuk proses enkripsi/dekripsi. Pemilihan file dilakukan dengan menekan tombol ‘Browse…’. Isi dari file akan ditampilkan pada textBox yang terletak pada bagian Okamoto-Uchiyama Cryptosystem bagian atas (rtb_Domain). Pada bagian controller juga terdapat dua buah checkbox yakni cb_Enkripsi dan cb_Dekripsi. User memilih checkbox yang bersesuaian dengan proses yang ingin dilakukan. Setelah mengisi checkbox maka selanjutnya yang perlu dilakukan adalah menekan tombol ‘Execute!’ untuk menjalankan proses enkripsi/dekripsi. Hasil dari proses tersebut akan dinyatakan pada kolom textBox bagian kanan bawah (rtb_Result). Terdapat juga progress bar untuk memeriksa progress dari proses enkripsi/dekripsi dan juga slot untuk menuliskan lamanya waktu eksekusi yang dibutuhkan untuk menjalankan proses. Hasil eksekusi yang ditampilkan pada rtb_Result juga dapat disimpan ke dalam file eksternal melalui button Save to File…

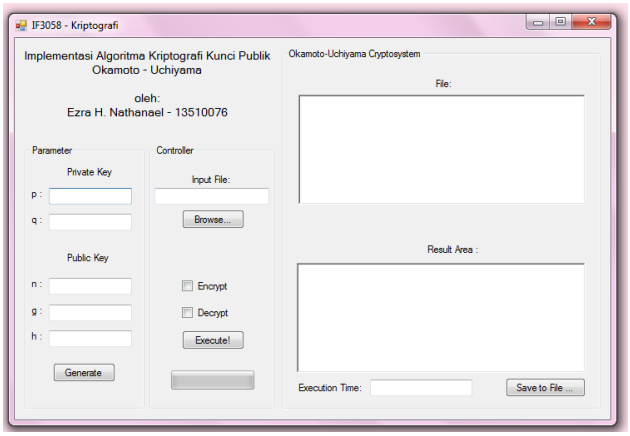
Untuk hasil eksekusi sendiri, dinyatakan dalam deretan bilangan-bilangan besar (BigInteger) yang dipisahkan oleh simbol ‘x’. Adapun untuk penulisan ke dalam file eksternal, simbol ini dihilangkan dan diganti menjadi penulisan per baris.

Adapun alur proses menjalankan program adalah sebagai berikut:

1) User memasukkan dua buah bilangan prima p dan q.
2) User bisa mendapatkan kedua bilangan prima tersebut dengan mengeksekusi program RPG001.java
3) Setelah memasukkan kedua bilangan prima, user menekan tombol Generate untuk menghasilkan bilangan n, g, dan h secara otomatis.
4) Setelah itu user memilih file input melalui tombol Browse yang terdapat pada panel controller.
5) Apabila berhasil, isi dari file input akan ditampilkan pada kolom File di bagian panel Okamoto-Uchiyama Cryptosystem.
6) User memilih mode yang akan dilakukan, apakah melakukan proses enkripsi atau dekripsi.
7) Setelah memilih proses dengan mengklik checkbox yang tersedia, user menjalankan program dengan menekan tombol Execute!
8) Tunggu hingga hasil selesai dengan melihat progress bar pada panel controller
9) Apabila proses sudah selesai (progress bar full), hasil akan ditampilkan pada bagian Result Area. Lamanya waktu eksekusi juga akan ditampilkan dalam satuan milisekon.
10) User dapat menyimpan file hasil proses baik ciphertext maupun plaintext ke dalam file eksternal melalui tombol Save to File

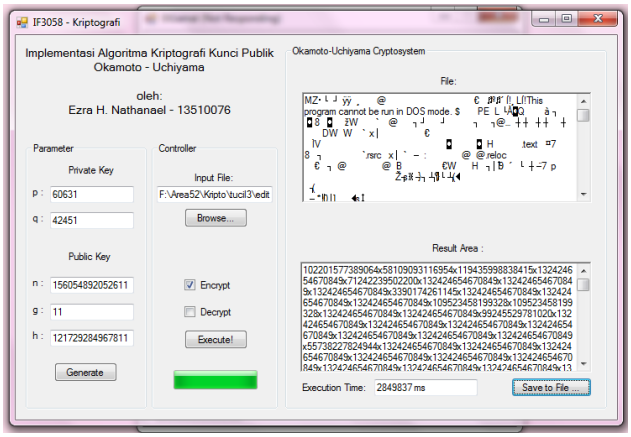
E. Implementasi Program Keseluruhan

Berikut ini merupakan hasil tampilan akhir dari program yang dibuat:



Gambar 1: Tampilan Awal Program

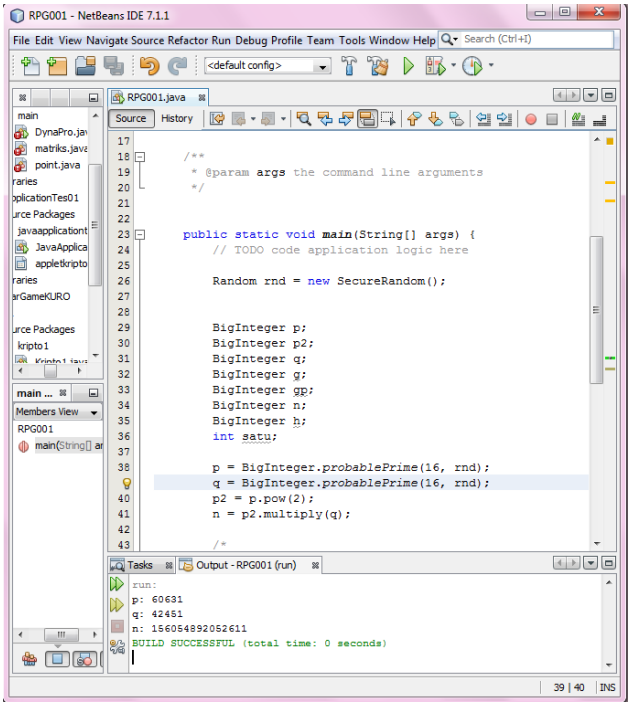
Sedangkan setelah dijalankan, contoh tampilannya adalah sebagai berikut:



Gambar 2: Tampilan Program Pasca Eksekusi

Dapat dilihat bagian-bagiannya di sini. Bagian paling kiri merupakan grup parameter, bagian tengah adalah controller, dan ruang di sebelah kanan merupakan tempat untuk membuka isi file dan menampilkan hasil eksekusi, serta menunjukkan lamanya waktu yang dibutuhkan untuk mengeksekusi proses. Penjelasan dari tiap-tiap bagian dapat dilihat pada bagian *D. Implementasi Rancangan*.

Adapun untuk tampilan program Prime Number Generator yang dibangun dalam bahasa Java adalah sebagai berikut:



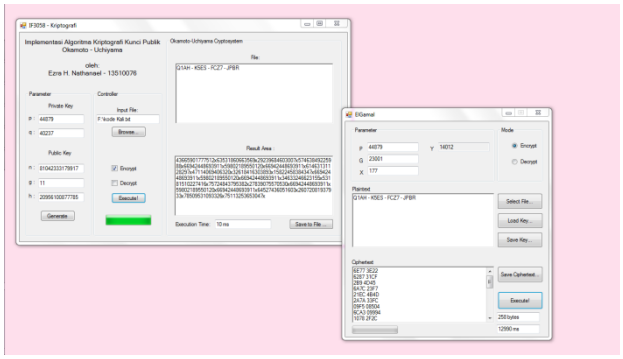
Gambar 3: Tampilan Program Prime Number Generator

IV. PENGUJIAN

Pada bagian pengujian ini, akan dilakukan proses enkripsi dua buah file dengan ekstensi .txt dan .doc

dengan menggunakan dua buah aplikasi yang sudah dibuat sebelumnya, yakni implementasi algoritma ElGamal dan Okamoto-Uchiyama. Faktor yang akan diperhatikan di sini adalah ukuran file awal, ukuran file hasil enkripsi (ciphertext), kunci yang digunakan, dan lamanya waktu eksekusi. Sebagai tambahan akan dicoba juga dengan sebuah file berekstensi .exe (executable files). Semua kunci yang digunakan untuk kunci privat (kedua bilangan prima) memiliki panjang maksimal 16 bit.

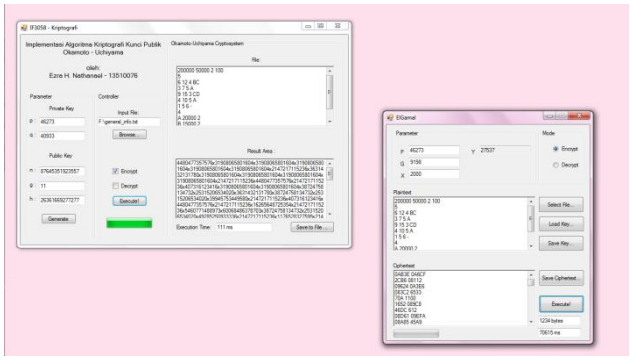
Pengujian pertama melibatkan sebuah file dengan ekstensi .txt berukuran 1KB. Hasil perbandingan adalah sebagai berikut:



Gambar 4: Perbandingan File Teks-1

Dapat dilihat bahwa untuk algoritma Okamoto-Uchiyama (selanjutnya disingkat dengan OU) membutuhkan waktu 10 ms, sedangkan algoritma ElGamal (selanjutnya disingkat dengan EG) membutuhkan waktu 12990 ms. Hasil file keluaran yang dihasilkan adalah sebesar 2KB baik untuk algoritma OU maupun EG.

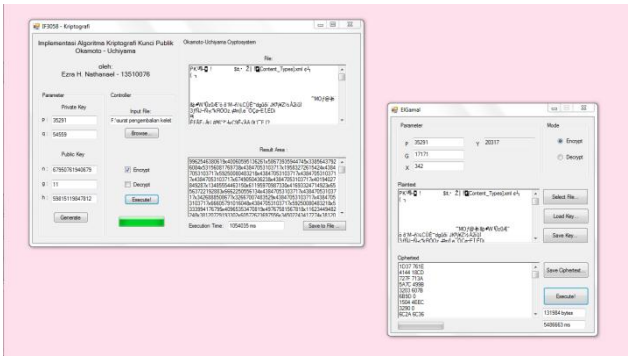
Apabila file teks sedikit diperbesar (meskipun masih berukuran 1KB), maka hasilnya adalah:



Gambar 5: Perbandingan File Teks-2

Waktu untuk algoritma OU adalah 111 ms sedangkan algoritma EG membutuhkan 70615ms. Hasil keluaran file juga berukuran 2KB untuk kedua algoritma.

Untuk pengujian dengan file berekstensi .docx, diambil sebuah file dengan besar 13.5KB, dan hasil perbandingannya adalah sebagai berikut:



Gambar 6: Perbandingan File Document

Waktu eksekusi yang dibutuhkan untuk algoritma OU adalah sebesar 1054035 ms atau sekitar 17.5 menit, sedangkan algoritma EG membutuhkan waktu 5486663 ms atau sekitar 91 menit. Ukuran file keluaran untuk algoritma OU adalah sebesar 215 KB, sedangkan algoritma EG sebesar 143 KB.

Adapun pengujian dengan executable file (ekstensi .exe) hanya berhasil dilakukan pada algoritma OU saja, di mana hasil yang diperoleh sama dengan pada gambar 2 di atas. Waktu eksekusi yang dibutuhkan adalah sebesar 2849837 ms atau sekitar 47 menit. Untuk algoritma EG hingga 120 menit masih belum mengeluarkan hasil.

Analisis yang dilakukan terhadap setiap hasil ini adalah bahwa dengan mengambil empat sampel, waktu pemrosesan untuk algoritma OU jauh lebih cepat dibandingkan algoritma EG. Hal ini dapat disebabkan beberapa hal seperti nilai random g yang menghasilkan nilai sangat kecil, bandingkan dengan algoritma EG di mana untuk setiap parameternya mengambil bilangan yang besar. Hal ini juga dapat dipengaruhi skema pemrosesan pada algoritma OU yang lebih sederhana dibandingkan dengan algoritma EG yang menghasilkan keluaran berupa pasangan dua nilai. Untuk ukuran file ciphertext yang dihasilkan, pada satu sampel terlihat bahwa hasil dari algoritma OU lebih besar ketimbang algoritma EG. Hal ini dipengaruhi oleh penulisan hasil, di mana untuk algoritma EG keluaran dinyatakan dalam pasangan bilangan berbasis hexadecimal, sedangkan untuk algoritma OU menggunakan bilangan besar (BigInteger). Nilai random yang dihasilkan pada algoritma OU (untuk nilai g dan r) juga belum sempurna sehingga mengurangi kepercayaan terhadap hasil yang diperoleh dalam pengujian.

V. KESIMPULAN

Kesimpulan yang diambil dari riset sederhana yang dilakukan ini adalah bahwa algoritma Okamoto-Uchiyama cukup praktis dibandingkan dengan algoritma ElGamal. Waktu yang digunakan untuk melakukan

pemrosesan jauh lebih singkat dibandingkan dengan algoritma ElGamal, namun ukuran file hasil eksekusi lebih besar. Kekuatan masih belum terbukti karena belum diuji untuk melakukan kriptanalisis dan pemfaktoran atau penghitungan logaritma diskrit. Secara keseluruhan, algoritma Okamoto-Uchiyama ini menarik, namun sayang masih belum banyak dokumen yang beredar terkait algoritma ini sehingga peluang pengembangan algoritma ini ke depannya masih sangat luas.

VI. UCAPAN TERIMA KASIH

Ucapan terima kasih disampaikan kepada Tuhan Yang Maha Esa atas segala bimbingan dan penyertaan sepanjang penyusunan makalah ini. Terima kasih kepada kedua orang tua penulis yang setia mendukung dalam doa dan semangat. Terima kasih kepada Bapak Rinaldi Munir selaku dosen pengampu mata kuliah IF3058 Kriptografi, teman-teman Informatika 2010, dan juga seluruh staff unit Genshiken ITB atas dukungannya di dalam segala hal. Terima kasih pula kepada keluarga di PMK ITB yang menjadi sahabat dalam setiap saat.

REFERENSI

Rinaldi Munir, Slide Materi Kuliah IF3058 Kriptografi.

http://www.stackoverflow.com

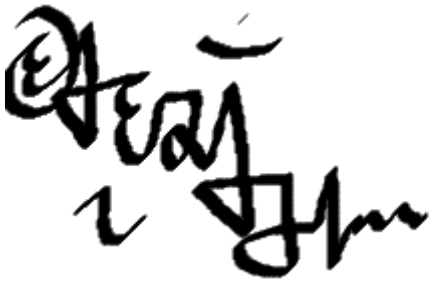
Uchiyama, Okamoto: A new public-key cryptosystem as secure as factoring

Budi Rahardjo, et al. Mudah Belajar Java.

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, 20 Mei 2013



Ezra Hizkia Nathanael
13510076