Pengamanan Removable Disk dengan Mengenkripsi Seluruh Header File

Gagarin Adhitama - 13508089
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
if18089@studetns.if.itb.ac.id

***Abstraksi*—Penggunaan *flashdisk* sebagai tempat penyimpanan data sudah sangat umum digunakan. *Flashdisk* ini dapat membantu mobilitas penggunanya karena dapat memindahkan data dari satu komputer ke komputer lain. Isu mengenai keamanan data yang disimpan dalam *flashdisk* ini menjadi penting untuk dibahas. Karena mobilitasnya yang cukup tinggi, kemungkinan isi data dapat diambil atau "dicuri" oleh pihak yang tidak bertanggung jawab. Oleh karena itu, pengamanan data diperlukan dalam *flashdisk* ini. Salah satu bentuk keamanan yang dapat dilakukan adalah dengan mengenkripsi *header* seluruh file yang disimpan dalam *flashdisk* tersebut. Enkripsi dan dekripsi data menggunakan algoritma ElGamal. ElGamal merupakan algoritma kriptografi asimetri yang menggunakan kunci publik untuk mengenkripsi dan kunci privat untuk mendekripsi. Dengan mengenkripsi *header* file, maka file tersebut akan tidak dapat dibuka jika tidak didekripsi terlebih dahulu. Ini merupakan salah satu cara untuk mengamankan *flashdisk* dari orang yang tidak bertanggung jawab.**

Kata kunci*—penerapan ElGamal, pengamanan *flashdisk

I. PENDAHULUAN

Pada era *mobile* saat ini, segala sesuatunya harus dapat dibawa, dipindahkan, dan digunakan dengan mudah. Penggunaan *removable disk* adalah salah satu cara menudukung mobilitas orang-orang jaman sekarang. Salah satu keuntungan menggunakan *removable disk* ini adalah kita dapat membawa data dalam jumlah yang besar (dilihat dari ukuran file) dalam satu benda. Jaman dahulu, untuk menyimpan satu file dokumen digunakan disket yang ukurannya hanya terbatas 1,44 Mega Byte. Lalu berkembang menggunakan *compactdisk* (CD) yang berkapasitas 800 MB. Berkembang lagi menjadi suatu DVD yang dapat menyimpan file dengan kapasitas maksimal 4 GB. Nah, saat ini yang paling sering digunakan oleh orang banyak adalah *flashdisk* atau *external harddisk*.

Jika orang sudah banyak menggunakan *flashdisk* ini, yang menjadi perhatian utama adalah bagaimana keamanan data yang disimpan dalam *flashdisk* tersebut. Semakin *flashdisk* tersebut memiliki mobilitas yang tinggi, maka semakin besar celah kemanan datanya. Contoh dari celah keamanan data yakni ketika barang tersebut dipinjam orang, atau ada orang lain yang menitipkan datanya pada *flashdisk* kita. Selain itu, kadang *flashdisk* ini berisi file yang cukup penting dan rahasia. Sering terjadi *flashdisk* yang kita punya tertinggal di suatu tempat dan atau hilang diambil orang. Jika hal ini terjadi, maka data yang kita anggap penting dan rahasia tersebut dapat saja dibuka dan "dicuri" oleh pihak yang tidak bertanggung jawab. Celah-celah keamanan ini harus dapat diantisipasi agar data yang kita simpan di *flashdisk* ini menjadi aman.

Pada makalah ini, akan dibuat suatu aplikasi yang ide dasarnya membuat pengamanan data yang kita simpan dalam *flashdisk*. Pengamanan data tersebut dengan cara mengenkripsi *header* seluruh data yang disimpan dalam *flashdisk* tersebut. Algoritma yang digunakan dalam mengenkripsi adalah algoritma ElGamal. Algoritma ini dipilih karena dia merupakan algoritma asimetris yang dirasa cukup aman dan agak susah orang lain untuk menjebol keamanannya.

Aplikasi ini dapat mengenerate kunci publik dan privat untuk mengenkripsi dan mendekripsi *header* file. Tujuan mengenkripsi *header* file ini agar file yang disimpan menjadi *corrupt* atau rusak sehingga tidak dapat dibuka. Aplikasi ini cukup mendukung untuk pengamanan data dengan metode sederhana.

II. LANDASAN TEORI

A. Algoritma ElGamal

Algoritma ElGamal merupakan algoritma enkripsi kunci asimetris yang berdasarkan pada pertukaran kunci Diffe-Hellman. Algoritma ini diusulkan Taher ElGamal pada tahun 1984. Algoritma ini disebutalgoritma diskret karena nilainya berhingga dan bergantung pada bilangan prima yang digunakan. Karena bilangan prima yang digunakan adalah bilangan prima yang besar, maka sangat sulit bahkan tidak mungkin menurunkan kunci privat dari kunci publik yang diketahui walaupun serangan dilakukan dengan menggunakan sumberdaya komputer yang sangat besar.

Algoritma ElGamal terdiri dari tiga proses, yaitu proses pembentukan kunci, proses enkripsi, dan proses dekripsi. Plainteks yang akan dienkripsi dipecah menjadi blok-blok plainteks, selanjutnya proses enkripsi pada blok-blok plainteks dan menghasilkan blok-blok cipherteks yang kemudian dilakukan proses dekripsi dan hasilnya digabungkan kembali menjadi pesan yang utuh dan dapat dimengerti.

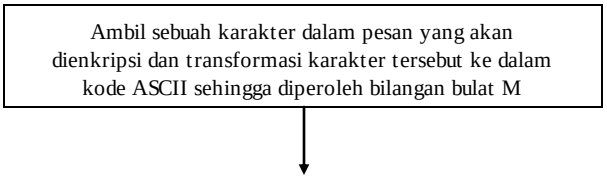
Proses pembuatan kunci pada ElGamal ini diawali dengan memasukkan bilangan prima (P), bilangan *generator* (G), dan kunci rahasia (X). Ketiga bilangan tersebut adalah bilangan pembangun. Kemudian dari ketiga bilangan tersebut akan menghasilkan satu bilangan (sebut saja bilangan Y). Untuk membangun nilai Y ini digunakan rumus berikut ini

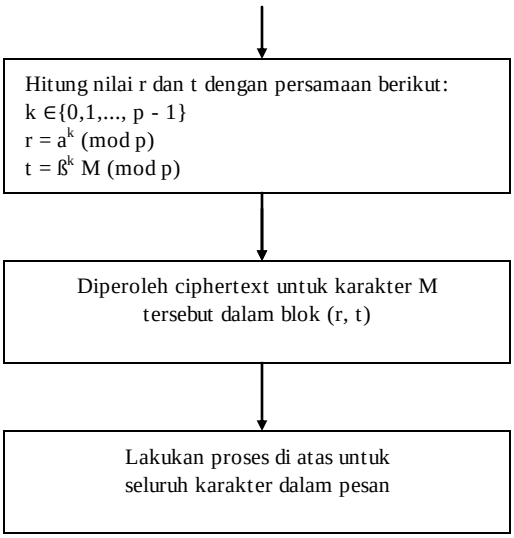
$$Y = G^X \bmod P$$

Sehingga didapatkan kunci publik yakni bilangan P, G, dan Y. Kunci publik ini dapat digunakan untuk mengenkripsi file. Sedangkan untuk melakukan dekripsi file, digunakan kunci privat yakni bilangan P dan X.

Keuntungan dari algoritma ElGamal ini adalah untuk melakukan enkripsi dan dekripsi menggunakan dua kunci yang berbeda (algoritma asimetris), sehingga lebih sulit bagi kriptanalis. Selain itu, masalah pertukaran kunci akan lebih mudah dan aman.

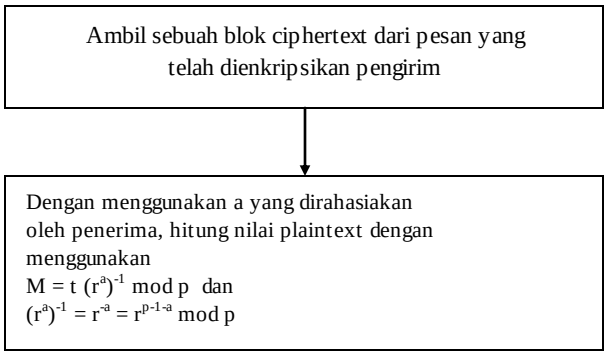
Proses enkripsi dapat dilihat pada bagan berikut ini





Gambar 1 Proses Enkripsi ElGamal [2]

Sedangkan untuk proses dekripsinya dapat dilihat pada bagan di bawah ini



Gambar 2 Proses Dekripsi ElGamal [2]

B. Header File

Setiap jenis file, .doc, .pdf, .xls, atau jenis file lainnya terbagi menjadi header dan body. Pada header disimpan bentuk (ekstensi) dari file tersebut, besar ukuran file tersebut, besarnya resolusi gambar (pada file bertipe gambar), dan lain sebagainya. Header file ini dapat disebut menyimpan meta data dari file. Jika meta data dari file tersebut disembunyikan (dienkripsi), maka yang terjadi adalah file tidak dapat dibuka atau *corrupt*. Header file dieksekusi (dicek) pertama kali saat pengiriman dan pada saat membuka file.

C. Aplikasi Microsoft .NET

Microsoft .NET yang awalnya disebut Next Generation Windows Services (NGWS) adalah suatu platform untuk membangun dan menjalankan generasi penerus aplikasi-aplikasi terdistribusi. Microsoft .NET merupakan framework (kerangka) pengembangan yang menyediakan antarmuka pemrograman baru

untuk layanan Windows dan API (Application Programming Interface). Microsoft .NET merupakan strategi Microsoft untuk menghubungkan sistem, informasi, dan alat (device), sehingga orang dapat berkomunikasi serta berkolaborasi dengan lebih efektif. Teknologi .NET terintegrasi penuh melalui produk-produk Microsoft, dan menyediakan kemampuan untuk mengembangkan solusi dengan menggunakan Web service. Platform Microsoft .NET tersusun dalam tiga lapisan (layer) yaitu:

1. Visual Studio .NET

Microsoft Visual Studio .NET merupakan kumpulan lengkap tools pengembangan untuk membangun aplikasi Web ASP.NET, XML Web Services, aplikasi dekstop, dan aplikasi mobile. Di dalam Visual Studio inilah bahasa-bahasa pemrograman .NET seperti Visual Basic, Visual C++, Visual C# (CSharp), dan Visual J# (JSharp) semuanya menggunakan lingkungan pengembangan terintegrasi atau IDE yang sama sehingga memungkinkan untuk saling berbagi tools dan fasilitas.

2. Visual Basic .NET

Visual Basic .NET (atau VB.NET) merupakan salah satu bahasa pemrograman yang bisa digunakan untuk membangun aplikasi-aplikasi .NET di platform Microsoft .NET. Tidak seperti generasi sebelumnya Visual Basic versi 6.0 ke bawah yang lebih difokuskan untuk pengembangan aplikasi desktop, Visual Basic .NET memungkinkan para pengembang membangun bermacam aplikasi, baik desktop maupun aplikasi web. Seiring dengan perkembangan aplikasi perangkat lunak yang semakin kompleks, saat ini Visual Basic .NET memasuki versi kelima (Visual Basic 2008).

3. IDE Visual Basic

IDE (Integrated Development Environment), atau juga disebut sebagai Integrated Design/Debugging Environment, adalah perangkat lunak komputer yang berfungsi untuk membantu pemrogram dalam mengembangkan perangkat lunak. Singkatnya, IDE merupakan suatu lingkungan pengembangan aplikasi yang terintegrasi; lengkap dengan beragam tools atau utilitas pendukung.

III. PENERAPAN DAN EKSPLORASI

A. Tahap Persiapan

Pada tahap persiapan ini, dilakukan beberapa hal seperti mengumpulkan segala materi (tertulis) untuk digunakan sebagai bahan pertimbangan dalam mengembangkan aplikasi ini. Materi yang perlu dipelajari salah satunya yakni tentang bagaimana standardisasi pengembangan aplikasi .NET. Aplikasi ini dikembangkan menggunakan bahasa C#. Lain daripada itu, hal-hal kecil seperti pengubahan isi file ke dalam bentuk byte ASCII. Tujuan pengubahan tersebut agar algoritma ElGamal ini dapat diterapkan.

Pada tahap persiapan ini pula disiapkan beberapa modul untuk membuat suatu program *autoplay*. Sehingga ketika *flashdisk* ini dicolokkan ke komputer kita, maka aplikasi Pengamanan Data ini dapat otomatis dijalankan atau di-*execute*. Tujuan dari *autoplay* ini adalah agar mempermudah pengguna dalam menggunakan aplikasi ini. Ketika *flashdisk* dicolokkan, aplikasi akan *auto-run* dan muncul tampilan awal aplikasi. Pengguna tinggal memilih akan melakukan enkripsi atau dekripsi. Ketika melakukan enkripsi, maka pengguna harus memasukkan kunci publik, lalu seluruh isi data di *flashdisk* tersebut akan diacak *header*-nya. Untuk mengembalikannya (mendekripsi) harus menggunakan kunci privatnya.

Aplikasi ini pada umumnya terdiri dari dua bagian penting, yaitu bagian generator kunci publik dan privat dan bagian enkripsi dekripsi. Algoritma ElGamal ini digunakan karena dianggap lebih kuat dalam mengamankan bit-per-bit.

B. Tahap Pengembangan

Pada tahap pengembangan ini, yang pertama dilakukan adalah mengubah bentuk file ke dalam byte. File-file di dalam *flashdisk* diubah bentuk menjadi *array of byte* sehingga file dapat “diutak-atik” sesuai kebutuhan. Setelah file tersebut diubah ke dalam byte, kumpulan dari byte tersebut diubah ke dalam bentuk integer. Tujuan pengubahan bentuk integer ini bertujuan agar file dapat dibagi-bagi dalam bentuk blok-blok. Selain untuk membagi ke dalam bentuk blok, integer hasil pengubahan byte digunakan untuk menjalankan enkripsi dekripsi dengan algoritma ElGamal ini. Integer hasil dari pengubahan byte tadi, digunakan untuk menghitung panjang plainteks. Panjang plainteks akan dibagi dengan suatu nilai “long” sehingga menghasilkan berapa jumlah blok hasil pembagian plainteks tersebut. File ini dibagi ke dalam beberapa blok tujuannya agar file kunci prima (P) dapat digunakan untuk melakukan enkripsi dekripsi. Pada aplikasi ini tidak digunakan batasan minimal kunci. Nilai berapapun yang dimasukkan, maka kunci dapat digenerate. Untuk mengantisipasi tidak adanya batasan tersebut, maka digunakan pembagian blok ini.

Setelah urusan translasi ke dalam byte dan pembagian blok ini selesai, maka dilanjutkan dengan pembuatan pembangkit kunci (key generator). Pada pembangkit kunci ini, yang dilakukan pertama kali adalah melakukan pengecekan apakah bilangan P benar diisi oleh bilangan prima atau tidak. Untuk lebih dapat mengamankan data, bilangan P ini seharusnya diisi

dengan nilai prima yang besar. Tujuannya agar jika ada seorang kriptanalis yang berusaha memecahkan kunci privatnya, itu akan sangat sulit. Setelah pengecekan nilai prima tersebut, kemudian dikerjakan pengecekan masukkan nilai G dan X. Nilai bilangan G tidak boleh lebih besar daripada nilai bilangan P. Kemudian nilai bilangan X harus lebih besar sama dengan 1 dan kurang dari nilai bilangan P dikurangi 2 ($1 \leq X < P-2$). Syarat-syarat nilai bilangan ini harus dipenuhi agar penghitungan nilai bilangan Y dapat dilakukan dan menghasilkan nilai Y. Nilai Y didapatkan dari penghitungan seperti rumus yang telah dituliskan sebelumnya. Setelah dihasilkan nilai Y, maka pengguna telah berhasil melakukan pembangkitan kunci publik dan kunci privat. Pengguna dapat mengingat-ingat angka tersebut untuk melakukan enkripsi dan dekripsi atau pengguna juga dapat menyimpannya ke dalam suatu bentuk file yang dapat digunakan untuk enkripsi dan dekripsi juga. Pada aplikasi ini, kunci publik disimpan ke dalam file dengan ekstensi .pub dan kunci privat disimpan dalam file berekstensi .pri. Tujuan dari penyimpanan kunci ini agar mempermudah pengguna untuk melakukan pengamanan data *flashdisk* dengan tanpa harus mengingat-ingat angka hasil pembangkitan kunci tadi.

Dalam melakukan enkripsi, blok yang telah dibagi-bagi tadi di-assign dalam bitArray. Hasil dari bitArray ini akan ditambahkan dengan nilai A dan nilai B untuk menghasilkan suatu bilangan hexadecimal. Bilangan hexadecimal ini merupakan hasil enkripsi (chipertext). Nilai A dan B didapatkan dari perpangkatan dan perkalian antara nilai P, G, dan Y. Pada penerapan algoritma ElGamal ini banyak digunakan rumus atau fungsi matematis dalam kerjanya. Perhitungan matematika seperti perpangkatan, perkalian, pembagian, dan penjumlahan ada di dalamnya. Hasil cipherteksnya pun berbentuk suatu bilangan yakni bilangan hexadecimal. Untuk mengembalikan (mendekripsi) cipherteks tersebut, dilakukan hal yang mirip dengan enkripsinya. Hanya saja nilai bilangan X digunakan dalam perpangkatan. Secara teknis, tahap enkripsi dan dekripsi ini tidak mengalami banyak kendala.

Selanjutnya, yang dilakukan adalah membuat suatu tampilan menarik dalam bahasa C#. Tampilan ini sering diremehkan dalam pengembangannya. Padahal dalam pembuatan tampilan ini perlu banyak pertimbangan agar seluruh pengguna dapat menggunakan aplikasi ini. Konsep tampilan pada aplikasi ini adalah sederhana dan mudah dipahami, baik dari segi tulisan, warna, dan tombol-tombol yang ada. Pada aplikasi ini juga dibuat program *autoplay* sehingga ketika ada *flashdisk* dicolokkan ke komputer, program ini akan otomatis dijalankan. Tampilan awal dari *autoplay* tersebut adalah tampilan untuk mendekripsi. Pertimbangan hal tersebut karena secara kebiasaan, jika *flashdisk* tersebut sudah diproteksi maka ketika dicolokkan pertama kali, yang harus dilakukan oleh pengguna adalah mendekripsinya terlebih dahulu. Oleh karena itu, program *autoplay* yang dibuat sengaja menampilkan tampilan untuk mendekripsi. Jika *flashdisk* tersebut memang belum diproteksi dengan aplikasi ini, program *autoplay* ini dapat ditutup atau *close* sehingga mudah dalam penggunaan.

C. Hambatan

Dalam pengembangan aplikasi ini, dari awal hingga dapat digunakan, terdapat beberapa hambatan yang cukup mengganjal. Salah satunya yakni dalam transformasi dari file menjadi suatu byte. Setelah itu harus dibagi ke dalam suatu blok-blok kecil. Blok tersebut nantinya dibuat dalam suatu bitArray dan dioperasikan dengan bilangan-bilangan. Yang menjadi permasalahan utama adalah permasalahan bahasa C# yang cukup rumit untuk membentuk suatu bitArray.

Kemudian hambatan yang lain yakni jika di dalam *flashdisk* tersebut terdapat berkas (folder), dimana tiap berkas tersebut memiliki upa folder (sub folder) yang cukup dalam, aplikasi ini kadang mengalami kegagalan dalam mendapatkan isi folder tersebut. Jika upa folder hanya satu atau dua kedalaman, aplikasi ini masih dapat mencari isi file dari folder tersebut.

Kesulitan yang cukup berat untuk diselesaikan adalah memisahkan antara *header* file dengan *body* file pada tiap-tiap data yang disimpan di *flashdisk* tersebut. Awalnya, transformasi file ke dalam bentuk byte hanya dilakukan pada *header* saja. Namun, ternyata tiap-tiap file memiliki struktur baku *header* yang berbeda-beda. Bahkan untuk file tertentu seperti .mp3, *header* filenya tidak dapat dipisahkan dengan *body* file. Untuk mengantisipasi hal tersebut, maka untuk file yang *header*-nya memiliki struktur bakunya tidak diketahui, mau tidak mau seluruh isi file ditransformasikan ke dalam bentuk byte. Efek sampingnya adalah waktu eksekusi menjadi cukup lama.

IV. ANALISIS HASIL

Hasil dari pengembangan aplikasi ini cukup baik. Hanya saja memang masih terdapat beberapa kutu (*bug*) yang agak mengganggu kenyamanan. Dari aplikasi yang dikembangkan ini, dapat diambil banyak pelajaran. Tujuan awal dari pengembangan aplikasi ini adalah mengamankan *removable disk* sehingga orang

yang tidak berkepentingan tidak dapat menggunakan data kita. Hal tersebut secara sederhana sudah dapat dilakukan dengan baik, tetapi ternyata dalam mengembangkan suatu aplikasi terdapat beberapa hal yang luput dari rencana awal. Seperti halnya awalnya ingin membuat suatu aplikasi yang dapat untuk mengunci *flashdisk*, tetapi ternyata aplikasi ini dapat digunakan untuk pengamanan data. Ketika file yang telah dienkripsi dipindahkan dari *flashdisk* ke komputer, maka file tersebut akan tetap dalam keadaan terenkripsi. Aplikasi ini tetap dapat digunakan untuk membuka "gembok" mendekripsi file yang sudah dipindahkan tadi. Rencana awal, direktori file yang akan dienkripsi dan dekripsi sudah otomatis ditujukan pada direktori *removable disk*. Namun, ternyata cukup sulit dalam pengaplikasiannya. Karena direktori tersebut tidak tetap, bergantung pada jumlah direktori komputer yang digunakan. Ada komputer yang hanya dua direktori (C: dan D:), ada yang hingga empat direktori. Ini yang menyebabkan rencana awal tersebut batal dieksekusi. Lalu pada pengembangannya, direktori file dipilih secara manual. Pemilihan direktori secara manual ini yang dapat digunakan untuk enkripsi dan dekripsi file lain yang berada di luar *flashdisk* kita.

Untuk sebagai pengaman data, aplikasi sederhana ini sudah cukup mumpuni karena dapat mengacak *header* file sehingga file tidak dapat dibuka jika tidak didekripsi terlebih dahulu. Hanya saja permasalahan lain yang timbul adalah masalah lama waktu eksekusi. Karena tidak semua struktur baku *header* file diketahui, maka transformasi file ke dalam bentuk byte dilakukan ke seluruh isi file. Transformasi ini memakan waktu yang cukup lama. Karena transformasi sudah membutuhkan waktu, maka enkripsi hanya dilakukan pada *header* file. Ide awal ini karena memang ditujukan untuk mempercepat waktu komputasi dalam mengeksekusi.

Aplikasi ini mungkin lebih cocok disebut aplikasi untuk mengamankan file. Karena ternyata untuk beberapa jenis file, mau disimpan dalam *flashdisk* atau tidak, file tersebut tetap dapat "diamankan" dengan mengacak *header* filenya. Namun, tidak ada salahnya aplikasi ini tetap digunakan untuk mengamankan *flashdisk*, karena pada prinsipnya memang dapat digunakan untuk hal tersebut.

V. KESIMPULAN

Pada prinsipnya, kemanan data itu sangat penting. Kita dapat melakukan banyak cara untuk melakukan pengamanan data. Salah satu metode yang dapat dilakukan adalah dengan mengamankan alat penyimpanan data berupa *removable disk* atau yang banyak dipakai yakni *flashdisk*. Untuk mengamankan data tersebut pun dapat digunakan aplikasi yang cukup sederhana. Aplikasi yang telah dikembangkan ini sudah dapat digunakan untuk salah satu cara pengamanan data. Hanya saja memang masih ada kekurangan. Kekurangan paling utama yakni permasalahan lamanya waktu eksekusi. Lamanya waktu ini disebabkan oleh pengubahan atau transformasi file ke dalam kumpulan byte. Namun, lepas daripada itu aplikasi ini sudah dapat berkontribusi dalam dunia kriptografi.

REFERENSI

[1] Tsiounis, Yianis and Yung, Moti. (1998) *On The Security of* ElGamal *based Encryption.* Public Key Cryptography, PKC'98, LNCS 1431, pp. 117-134

[2] Zulhazmi, Dimas. (2012). *Analisis Algoritma Enkripsi* ElGamal, *Grain V1, dan Aes dengan Studi Kasus Aplikasi Resep Makanan*

[3] http://en.wikipedia.org/wiki/Header_(computing)

[4] http://msdn.microsoft.com/en-us/netframework/default.aspx

[5] http://stackoverflow.com/questions/4894882/how-to-encrypt-decrypt-text-files-using-elgamal

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, 13 Mei 2012

ttd

Gagarin Adhitama 13508089