ALGORITMA ELGAMAL UNTUK KEAMANAN APLIKASI E-MAIL

Satya Fajar Pratama – NIM : 13506021

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if16021@students.if.itb.ac.id

Abstrak

Aplikasi email saat ini sudah banyak digunakan di seluruh belahan dunia. Banyak hal yang dapat diselesaikan hanya dengan mengirimkan sebuah email, seperti undangan suatu acara. Aplikasi ini menjadi banyak digemari oleh seluruh kalangan karena tidak memerlukan biaya yang mahal dan memiliki efektifitas serta efisiensi yang tinggi.

Dengan pesatnya laju perkembangan email di dunia, maka dibutuhkan suatu mekanisme yang dapat menjamin keamanan email yang dikirim. Saat ini, pesan yang dikirimkan melalui email tidak hanya pesan biasa, tetapi juga pesan yang bersifat penting dan rahasia. Karena proses pengiriman email akan meninggalkan salinan pada kedua belah pihak, maka kemungkinan terjadinya serangan terhadap email tersebut akan semakin besar. Oleh karena itu, mekanisme keamanan pada email menjadi suatu hal yang *mandatory*.

Salah satu metode yang dapat digunakan untuk mengamankan pesan yang dikirim melalui email adalah enkripsi dengan algoritma ElGamal. Algoritma ElGamal termasuk ke dalam kriptografi kunci publik. Kekuatan algoritma ElGamal terletak pada *discrete logarithm problem* dan cara kerjanya yang nirsimetri. Penerapan algoritma ElGamal pada aplikasi email diharapkan dapat meningkatkan keamanan pada pesan yang dikirimkan sehingga para pengguna aplikasi email tersebut tidak perlu khawatir akan terjadinya serangan (*attack*).

***Kata kunci** : email, keamanan, El Gamal, kriptografi kunci public*

1. Pendahuluan

Dewasa ini, teknologi di dunia terus berubah dan berkembang. Aplikasi internet merupakan aplikasi yang sangat umum digunakan saat ini. Namun, semua perkembangan teknologi tersebut ternyata membawa beberapa masalah. Masalah terbesar dari aplikasi internet adalah keamanan. Banyak algoritma dan metode baru yang dikembangkan untuk membuat aplikasi ini lebih aman. Para pengembang perangkat lunak tentunya harus mencoba metode baru tersebut terlebih dahulu. Jika metode baru tersebut berjalan sesuai dengan harapan, barulah para pengembang perangkat lunak dapat menerapkan metode ini pada aplikasinya. Aplikasi e-mail, yang saat ini banyak digunakan di seluruh dunia, mulai digunakan pertama kali di Departemen Pertahanan Amerika pada tahun 1970. Banyak proses yang dapat dilakukan hanya dengan mengirimkan e-mail, seperti hubungan antar perusahaan. Selain itu, biaya, kecepatan dan efektivitas merupakan fitur penting lainnya dari sebuah e-mail. Untuk melindungi aplikasi e-mail dari segala jenis serangan, metode yang dapat digunakan adalah menggunakan algoritma enkripsi yang kuat. Algoritma enkripsi tersebut dapat diderivasi atau diturunkan dari algoritma-algoritma enkripsi yang lama. Dengan adanya sebuah algoritma enkripsi yang kuat, permasalahan keamanan pada aplikasi e-mail dapat teratasi.

Dalam kriptografi, terdapat banyak sekali algoritma-algoritma yang dapat digunakan untuk melakukan enkripsi pada sebuah pesan. Pada makalah ini, algoritma enkripsi ElGamal akan dipilih untuk melakukan enkripsi pada suatu data, sedangkan sebuah fungsi SHA

(*Secure Hashing Algorithm*) yang unik akan digunakan untuk melakukan autentikasi identitas.

Enkripsi merupakan sebuah proses transformasi yang akan mengubah pesan menjadi suatu bentuk yang tidak dapat dimengerti dengan menggunakan sebuah fungsi nonlinear. Bentuk pesan yang belum dienkripsi disebut sebagai "plaintext" dan bentuk pesan yang telah dienkripsi disebut sebagai "ciphertext". Proses pengubahan ciphertext menjadi plaintext kembali dengan menggunakan fungsi balikan (*inverse*) dari fungsi pengenkripsi disebut dekripsi. Model matematis untuk proses enkripsi adalah :

F (P) = C ,

sedangkan model matematis untuk proses dekripsi adalah :

$F^{-1}(C) = P$.

Kriptografi dahulu hanya digunakan untuk komunikasi politik atau militer. Akan tetapi, seiring dengan area penggunaan kriptografi yang semakin luas, kriptografi telah menjadi sebuah standar. Algoritma enkripsi ElGamal merupakan sebuah algoritma kunci-publik yang diperkenalkan pada tahun 1985 oleh T. ElGamal. Belum ada serangan (*attack*) yang berhasil memecahkan algoritma ini.

Panjang kunci ElGamal dapat berkisar antara 256-bit sampai tak terhingga. Panjang kunci sebesar 1024 bit - 2048 bit dapat dianggap aman untuk 10 tahun ke depan. Tentu saja, prediksi ini didasarkan pada daya komputasi saat ini dan perkiraan kasar kemajuan hardware dalam waktu dekat. Panjang *private key* sendiri dapat berkisar antara 160 bit sampai 240 bit.

2. Infrastruktur Algoritma ElGamal

Algoritma ElGamal merupakan sebuah algoritma kunci-publik yang didasarkan pada permasalahan *discrete logarithm*. Algoritma ElGamal terdiri atas algoritma enkripsi dan tanda tangan digital (*digital signature*). Algoritma enkripsi ElGamal sendiri serupa dengan Diffie-Hellman key agreement protocol.

Parameter algoritma ElGamal terdiri dari sebuah bilangan prima p dan sebuah bilangan bulat g. Penerima (*receiver*) akan memiliki sebuah *private key* a dan sebuah *public key* y, dimana $y = g^a \pmod p$. Apabila seorang pengirim (*sender*) ingin mengirimkan pesan m kepada penerima, pengirim tersebut haruslah menentukan sebuah bilangan acak k yang kurang dari p terlebih dahulu. Setelah bilangan acak k diperoleh, barulah pengirim tersebut dapat melakukan komputasi sebagai berikut :

$y_1 = g^k \pmod p$ dan $y_2 = m \text{ xor } y^k$

dimana xor menunjukkan bitwise eksklusif-or. Pengirim kemudian akan mengirimkan (y1, y2) kepada penerima. Setelah menerima ciphertext dari pengirim, penerima dapat melakukan komputasi sebagai berikut :

$m = (y_1^a \bmod p) \text{ xor } y_2$

Setelah komputasi di atas selesai dilakukan, barulah penerima memiliki plaintext yang sama dengan pengirim.

2.1 Proses enkripsi dalam aplikasi

Langkah pertama yang akan dilakukan dalam penerapan algoritma ElGamal adalah proses pembagian sebuah plaintext ke dalam blok-blok. Enkripsi akan diaplikasikan terhadap blok-blok tersebut. Untuk menentukan panjang dari setiap blok, perhitungan atau kalkulasi berikut dapat dilakukan.

Block=int(log(p)/log(strlen(alph)))

Alph merepresentasikan abjad (alphabet). Abjad yang digunakan dalam aplikasi ini meliputi simbol-simbol dasar sebanyak 96 karakter. Strlen merupakan fungsi yang digunakan untuk menghitung jumlah karakter dalam abjad. Nilai P adalah kunci yang akan dipakai. Logaritma dari nilai P dan logaritma dari jumlah karakter dalam alfabet akan sama dengan jumlah karakter dalam setiap blok. Jika panjang abjad yang digunakan berkurang atau nilai P bertambah, maka panjang blok akan meningkat. Seperti yang telah diketahui, suatu algoritma enkripsi akan menghasilkan sebuah ciphertext dan sebuah karakter

plaintext. Misalnya, jika sebuah karakter dienkripsi, pada akhir enkripsi akan terdapat dua buah karakter yang dihasilkan. Hal ini berarti apabila sebuah blok hanya memiliki satu karakter, akan dihasilkan sebuah ciphertext hasil enkripsi dengan panjang dua kali dari panjang plaintext-nya. Akan tetapi, apabila plaintext dibagi ke dalam sejumlah blok, permasalahan ini dapat dicegah. Setelah langkah di atas selesai dilakukan, karakter-karakter yang terdapat pada plaintext akan dipindai satu per satu. Karakter pertama akan diambil. Sebuah bilangan acak k akan ditentukan. Lalu, bilangan acak k tersebut akan dimodulasi sehingga k lebih kecil dari p. Kemudian, nilai y_1 akan dikalkulasi sesuai dengan parameter nilai k, nilai p, dan nilai g. Nilai y_1 hasil kalkulasi tersebut akan dimodulasi dengan panjang dari abjad yang digunakan.

$y_1 = y_1 \text{ (mod strlen (alph))}$
cipher1 [i] = alph [y1]

Abjad ke-y_1 dalam alphabet yang digunakan akan di-*assign* ke dalam array cipher1 (indeks i menandakan karakter yang dienkripsi). Array cipher1 akan berisi karakter-karakter yang telah dienkripsi. Nilai y_2 akan dikalkulasi sesuai dengan parameter nilai indeks, nilai k, nilai p, dan nilai y. Nilai y_2 hasil kalkulasi tersebut akan dimodulasi dengan panjang abjad yang digunakan.

$y_2 = y_2 \text{ (mod strlen (alph))}$

cipher2[i] = alph [y_2]

Abjad ke-y_2 dalam alphabet yang digunakan akan di-*assign* ke dalam array cipher2. Array cipher2 ini akan berperan sebagai kunci bagi karakter-karakter yang telah dienkripsi. Setelah semua proses selesai dilakukan, data-data yang terdapat pada array cipher1 dan cipher2 akan digabungkan menjadi sebuah array cipher.

cipher[i]=cipher1[i]+cipher2[i]

Array cipher inilah yang akan berisi ciphertext.

2.2 Proses dekripsi dalam aplikasi

Hal pertama yang harus dilakukan dalam proses dekripsi adalah menentukan jumlah blok dalam ciphertext dengan menggunakan rumus (5). Kemudian, sebuah loop yang akan melakukan perhitungan sebanyak *Strlen (cipher) / 2 *blok+2* kali dibentuk. Angka numerik yang setara dengan karakter di-*assign* ke dalam array plain1. Nilai tersebut adalah indeks dari karakter dalam abjad. Dengan nilai ini, y_1 dapat dikalkulasi. Angka numerik yang setara dengan karakter berikutnya akan di-*assign* ke dalam array plain2. Dengan nilai tersebut, nilai y_2 dapat dikalkulasi. Selanjutnya, dengan menggunakan nilai p dan nilai a yang telah diberikan, fungsi (y_1^a mod p) xor y_2 dapat dikalkulasi. Fungsi tersebut akan mengembalikan sebuah indeks. Karakter pada indeks tersebut dalam abjad yang digunakan merupakan karakter yang telah didekripsi. Karakter tersebut kemudian di-*assign* ke dalam array plain. Setelah semua karakter selesai diproses, array plain akan berisi plaintext.

3. Algoritma Hash

Para pengguna yang akan memanfaatkan atau menggunakan aplikasi haruslah mendaftar (*register*) terlebih dahulu. Pengguna akan diminta untuk memasukkan password ketika proses pendaftaran berlangsung. Sandi ini tidak akan langsung disimpan ke dalam database. Namun, sandi tersebut akan diubah menjadi bentuk lain terlebih dahulu dengan fungsi hashing, barulah kemudian disimpan ke dalam database. Hal ini dilakukan untuk mempersulit upaya orang-orang yang tidak berwenang dalam memperoleh password para pengguna aplikasi.

Berikut penjelasan algoritma hash yang digunakan dalam aplikasi. Password yang dimasukkan oleh para pengguna akan dibaca sebagai array of character. Kemudian, indeks dari masing-masing karakter tersebut akan dipangkatkan dengan dua. Hasil pemangkatan tersebut akan dikalikan dengan nilai ASCII dari masing-masing karakter. Selanjutnya, semua hasil perkalian tersebut akan saling ditambahkan. Apabila hasil akhir lebih besar dari sebuah bilangan acak epsilon, maka hasil

akhir tersebut dipecah menjadi 2 bagian, yaitu bagian kiri dan bagian kanan. Lalu, kedua bagian tersebut ditambahkan. Proses ini akan dilakukan berulang-ulang sampai hasil akhir yang diperoleh sama atau lebih kecil dari bilangan acak epsilon.

Misalnya, password yang dimasukkan oleh pengguna adalah “fajar”. Seperti yang telah diketahui, nilai ASCII untuk *lower case* dan *upper case letters* adalah berbeda. Password tersebut kemudian akan dibaca sebagai array of character sehingga karakter f memiliki indeks 0, a 1, j 2, a 3, dan r 4. Nilai ASCII dari masing-masing karakter tersebut adalah sebagai berikut :
f:102
a:97
j:106
r:114

Selanjutnya, fungsi berikut akan dikalkulasi berdasarkan password yang dimasukkan oleh penguna :

$102*2^0+97*2^1+106*2^2+97*2^3+114*2^4$

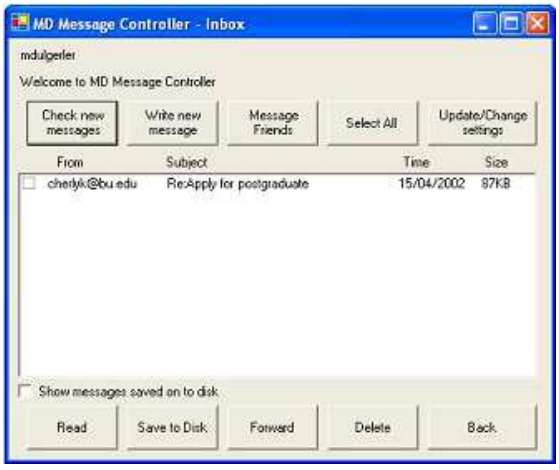
Hasil dari perhitungan di atas adalah 3320. Apabila bilangan acak epsilon adalah 1000, maka hasil perhitungan tersebut harus dipecah menjadi 2 bagian karena melebihi nilai epsilon. Angka 33 akan menjadi bagian pertama dan angka 20 akan menjadi bagian kedua. Kedua bagian tersebut kemudian akan ditambahkan (33+20=53) sehingga diperoleh hasil sebesar 53. Karena 53 lebih kecil dari bilangan acak epsilon, maka hasil tersebut akan disimpan ke dalam database sebagai password pengguna.

Ketika para pengguna melakukan login untuk mengakses aplikasi, pengguna tersebut akan memasukkan password pada form yang telah disediakan. Apabila pengguna menekan tombol OK, maka prosedur atau mekanisme yang telah dijelaskan sebelumnya akan dilakukan. Selanjutnya, hasil yang diperoleh dari prosedur atau mekanisme tersebut akan dibandingkan dengan hasil pembacaan dari database. Jika kedua nilai tersebut sama, maka pengguna tersebut dapat mengakses aplikasi.

4. Aplikasi e-mail berbasis algoritma ElGamal

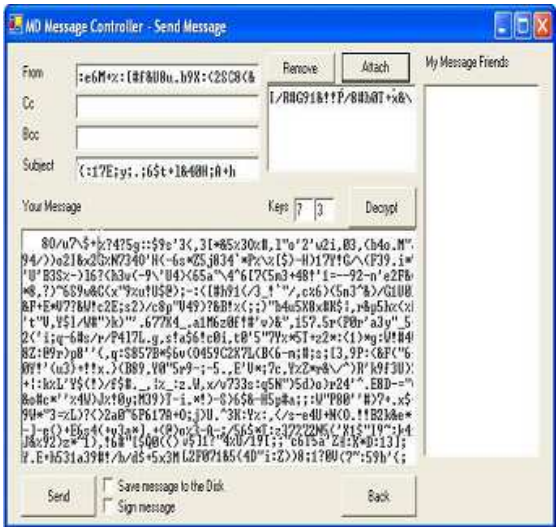
Berikut beberapa screenshot dari aplikasi e-mail yang menggunakan algoritma ElGamal dalam sistem keamanannya.

Gambar 1 User Login Window



Gambar 2 Received Message Window

Gambar 3 Send Message Window



Gambar 4 Encrypted Message

5. Kesimpulan dan rencana pengembangan

Aplikasi ini dibangun dengan memanfaatkan algoritma enkripsi ElGamal dan sebuah algoritma hash yang unik. Tingkat keamanan dari aplikasi ini dapat ditingkatkan dengan menambah jumlah bit yang digunakan dalam proses. Berdasarkan analisis yang telah dilakukan, algoritma RSA dan ElGamal akan memiliki tingkat keamanan yang setara untuk panjang kunci yang sama. Pengembangan lebih lanjut yang dapat diterapkan pada aplikasi e-mail ini adalah meningkatkan performansi dari aplikasi dengan menggunakan algoritma kompresi yang sudah ada.

6. Daftar Pustaka

[1] R. Munir, "*Diktat Kuliah IF3058 Kriptografi*", Program Studi Teknik Informatika Institut Teknologi Bandung, 2006.

[2] http://www.idssoftware.com/jsecure.html

[3] http://www.soriac.de

[4] http://www.cryptome.org