IMPLEMENTASI ALGORITMA KRIPTOGRAFI RSA PADA SURAT ELEKTRONIK (*E-Mail*)

Anang Paramita Wahyadyatmika[*)], R. Rizal Isnanto, and Maman Somantri

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

[*)]*E-Mail: nankene.83@gmail.com*

Abstrak

Seiring kemajuan teknologi informasi, muncul isu-isu keamanan yang kemudian menjadi sebuah hal yang sangat penting. Karena adanya dampak negatif dari perkembangan teknologi ini yaitu terjadinya kejahatan-kejahatan *cyber* yang memanfaatkan celah keamanan yang ada untuk masuk kedalam suatu jaringan dan melakukan manipulasi terhadap data/informasi yang ditransmisikan. Oleh karena itu perlu adanya suatu aplikasi yang mampu menjaga kerahasiaan dan keamanan proses pendistribusian informasi salah satunya yaitu aplikasi *email client*. Aplikasi ini dirancang menggunakan bahasa pemrograman PHP dengan mengimplementasikan algoritma kriptografi RSA di dalamnya. Penelitian ini menghasilkan suatu aplikasi *email client* yang terhubung langsung dengan jaringan internet. Dengan mengimplementasikan algoritma kriptografi RSA pada aplikasi ini, membantu pengguna untuk melakukan proses enkripsi pesan sebelum dikirimkan dan melakukan proses dekripsi saat penerimaan pesan. Sehingga dengan menggunakan aplikasi ini, pengguna untuk dapat melakukan proses distribusi informasi secara aman.

Kata kunci: Kriptografi RSA, Bahasa Pemrograman PHP, Email, Enkripsi, Dekripsi.

Abstract

As information technology advances, emerging security issues that later became a very important thing. Because of the negative impact of this technology is the development of cyber crimes that exploit vulnerabilities that are to enter into a network and manipulate the data / information being transmitted. Hence the need for an application that is able to maintain the confidentiality and security of information distribution process one of them is an email client application. This application is designed using PHP programming language to implement the RSA cryptographic algorithm in it. This study resulted in an email client that is connected directly to the Internet network. By implementing the RSA cryptographic algorithm in this application, helps users to perform the encryption process the message before sending and do the decryption process when receiving messages. So by using this application, users can perform secure information distribution process.

Keywords: RSA Cryptography, PHP Programming Languages, Email, Encryption, Decryption.

1. Pendahuluan

Dalam proses komunikasi, terdapat sebuah metode dalam pengamanan informasi yang dikenal dengan nama kriptografi. Menurut Bruce dalam bukunya yang berjudul "Applied Cryptography 2nd", kriptografi merupakan ilmu sekaligus seni untuk menjaga keamanan pesan[1]. Dalam kriptografi terdapat dua prinsip dasar, yaitu adanya proses enkripsi dan proses dekripsi. Enkripsi merupakan proses mengubah pesan asli (plainteks) menjadi pesan yang terkodekan (chiperteks). Sedangkan dekripsi merupakan proses mengubah chiperteks menjadi plainteks.

Berdasarkan kunci yang digunakan dalam proses enkripsi dan dekripsi, kriptografi dibedakan menjadi kriptografi kunci-simetri (*symetric-key cryptography*) yang biasa disebut kriptografi kunci-privat dan kriptografi kunci-nirsimetri (*asymetric key cryptography*) yang biasa disebut kriptografi kunci-publik[3]. Pada kriptografi kunci-publik terdapat sepasang kunci, satu kunci untuk proses enkripsi dan satu kunci untuk proses dekripsi. Kunci untuk proses enkripsi bersifat umum atau tidak dirahasiakan (kunci publik), oleh karena itu semua orang dapat menggunakannya. Sedangkan kunci untuk proses dekripsi bersifat rahasia (kunci rahasia) dan hanya digunakan oleh penerima pesan. Oleh karena itu, kunci enkripsi tidak boleh sama dengan kunci dekripsi.

Salah satu algoritma kriptografi kunci-publik yang populer adalah algoritma RSA. Keamanan algoritma RSA

terletak pada sulitnya memfaktorkan bilangan yang besar menjadi faktor-faktor prima. Pemfaktoran dilakukan untuk memperoleh kunci privat. Selama pemfaktoran bilangan besar menjadi faktor-faktor prima belum ditemukan algoritma yang mangkus, maka selama itu pula keamanan algoritma RSA tetap terjamin.

Penelitian ini mengambil studi kasus pada perangkat lunak pengiriman surat elektronik (*e-mail*) sederhana yang menggunakan konsep algoritma kriptografi RSA. Penelitian ini diharapkan dapat mejelaskan lebih detail proses-proses yang ada pada algoritma kriptografi RSA dan implementasiannya dalam suatu perangkat lunak lain ataupun pada penelitian lebih lanjut tentang algoritma kriptografi RSA.

Penelitian ini bertujuan adalah mampu mengetahui cara kerja dan kinerja algoritma kriptografi RSA. Selain itu juga diharapkan mampu mengimplementasikan teknik kriptografi RSA dalam perangkat lunak menggunakan bahasa pemrograman PHP. Dan akhirnya kita mendapatkan suatu konsep teknik keamanan pada pengiriman pesan melalui pembuktian yang dilakukan.

2. Metode

2.1 Kriptografi

Kriptografi (*cryptography*) berasal dari bahasa Yunani, terdiri dari dua suku kata yaitu "*cryptos*" yang artinya "*secret*" (rahasia) dan "*graphein*" yang artinya tulisan. Ada beberapa definisi kriptografi yang telah dikemukakan dalam berbagai literatur. Kriptografi merupakan suatu seni atau ilmu untuk menjaga kerahasiaan sebuah tulisan agar tetap aman tanpa diketahui oleh pihak yang tidak berhak[1]. Namun saat ini kriptografi bukan hanya sekedar seni dan kerahasiaan tetapi juga integritas, otentikasi dan keabsahan data.

2.1.1 Tujuan Kriptografi

Kriptografi bertujuan untuk memberikan layanan keamanan atau yang biasa disebut dengan aspek-aspek kemanan, antara lain sebagai berikut[1][2] :

1. Kerahasiaan adalah layanan yang digunakan untuk menjaga isi informasi dari siapapun kecuali yang memiliki otoritas atau kunci rahasia untuk membuka/mengupas informasi yang telah dienkripsi.
2. Integritas data adalah layanan yang memberikan jaminan bahwa tiap bagian pesan tidak akan mengalami perubahan dari saat data dibuat/dikirim oleh pengirim sampai dengan saat data tersebut dibuka oleh penerima data.
3. Otentikasi adalah layanan yang berhubungan dengan identifikasi / pengenalan, baik mengidentifikasi kebenaran pihak-pihak yang berkomunikasi maupun mengidentifikasi kebenaran sumber pesan.
4. Nirpenyangkalan adalah layanan untuk mencegah entitas yang berkomunikasi melakukan penyangkalan terhadap pesan yang dikirim oleh pengirim maupun penerima pesan menyangkal telah menerima pesan.

2.2 Algoritma Kriptografi RSA

Algoritma kriptografi RSA merupakan algoritma kriptografi kunci publik (nirsimetri). Ditemukan pertama kali pada tahun 1977 oleh R. Rivest, A. Shamir, dan L. Adleman. Nama RSA sendiri diambil dari ketiga penemunya tersebut. Sebagai algoritma kunci publik, RSA mempunyai dua kunci, yaitu kunci publik dan kunci rahasia. Kunci publik boleh diketahui oleh siapa saja, dan digunakan untuk proses enkripsi. Sedangkan kunci rahasia hanya pihak-pihak tertentu saja yang boleh mengetahuinya, dan digunakan untuk proses dekripsi. Algoritma kriptografi RSA terdiri dari tiga proses, yaitu proses pembentukan kunci, proses enkripsi dan proses dekripsi.

2.2.1 Proses Pembentukan Pasangan Kunci

Berikut ini adalah proses pembentukan kunci dalam algoritma kriptografi RSA :

1) Memilih dua bilangan prima yang diberi simbol sebagai p dan q (nilai $p \neq q$).
2) Menghitung nilai $n = p.q$ ($p \neq q$, karena jika $p = q$, maka nilai $n = p^2$ sehingga nilai p dapat diperoleh dengan menarik akar pangkat dua dari n).
3) Hitung $\varphi(n) = (p - 1)(q - 1)$.
4) Memilih kunci publik e yang relatif prima terhadap $\varphi(n)$.
5) Bangkitkan kunci privat dengan persamaan $e \cdot d \equiv 1 \pmod{\varphi(n)}$ dimana $1<d<\varphi(n)$. Perhatikan bahwa persamaan $e \cdot d \equiv 1 \pmod{\varphi(n)}$ ekivalen dengan $e \cdot d = 1 + k\,\varphi(n)$, sehingga untuk mencari nilai d dapat dihitung dengan $d = \frac{1 + k\,\varphi(n)}{e}$

   Hasil dari pembentukan pasangan kunci di atas adalah
   a) Kunci publik (e, n)
   b) Kunci rahasia (d, n)

   Nilai n tidak bersifat rahasia karena diperlukan pada saat perhitungan proses enkripsi dan dekripsi.

2.2.2 Proses Enkripsi

Berikut ini adalah proses enkripsi dalam algoritma kriptografi RSA :

1) Ambil kunci publik penerima pesan *e* dan modulus *n* atau (*e*,*n*).
2) Pilih plainteks *m* dan ubah isi pesan *m* menjadi pesan dengan nilai ASCII.

3) Potong pesan menjadi blok-blok pesan m_1, m_2, m_3, ... dengan nilai setiap bloknya adalah $0 \le m \le n-1$.
4) Setiap blok m dihitung dengan rumus $c_i = m_i^e$ **mod** n.
5) Susun nilai c hasil enkripsi dengan susunan c_1, c_2, c_3,....,c_n sehingga diperoleh cipherteks dari pesan m.

2.2.3 Proses Dekripsi

Berikut ini adalah proses dekripsi dalam algoritma kriptografi RSA :
1) Ambil pesan (cipherteks) yang telah diterima.
2) Kemudian ambil kunci rahasia d dan modulus n atau (d,n).
3) Potong pesan menjadi blok-blok pesan c_1, c_2, c_3, ... dengan nilai setiap bloknya adalah $0 \le c \le n-1$.
4) Hitung $m_i = c_i^d$ **mod** n.
5) Susun nilai m hasil dekripsi dengan susunan m_1, m_2, m_3,....,m_n sehingga diperoleh plainteks (pesan asli) dari cipherteks yang diterima.

2.3 Surat Elektronik (*E-Mail*)

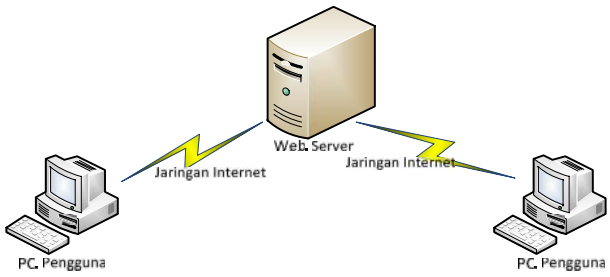
Elecktronic Mail atau *e-mail* atau surat elektronik adalah sebuah program yang diterapkan dalam sebuah jaringan komputer (termasuk *internet*) yang digunakan oleh pemakai untuk mengirimkan pesan dalam bentuk tulisan kepada pemakai lain di dalam atau di luar jaringan komputer tersebut. Fungsi dan cara kerja *e-mail* pun sama dengan surat pada umumnya, yaitu sebagai sarana untuk saling berkomunikasi dan saling mengirim sebuah informasi dokumen ataupun data dari suatu pihak ke pihak/orang lain dengan berdasarkan sebuah penamaan atau alamat *e-mail* yang dituju.

2.4 Deskripsi Sistem

Deskripsi sitem berisi tentang gambaran umum aplikasi yang akan dibangun. Aplikasi surat elektronik klien sederhana (*email client*) dibangun dengan mengimplementasikan algoritma kriptografi RSA. Aplikasi ini dirancang untuk dapat dijalankan secara *online* dengan menggunakan bahasa pemrograman PHP. Basisdata yang digunakan dalam sistem ini adalah MySQL. Sistem aplikasi yang dijalankan secara *online* dapat digambarkan seperti dalam **Gambar 2.1**.

Aplikasi yang akan dibangun sebenarnya memiliki fungsi yang sama dengan aplikasi surat elektronik pada umumnya yaitu dapat mengirimkan dan menerima pesan. Yang membedakan aplikasi ini dengan aplikasi surat elektronik lainnya adalah adanya proses enkripsi dan dekripsi pesan.

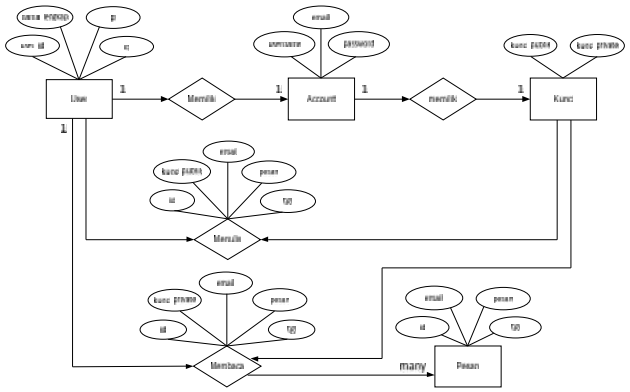
Dalam aplikasi surat elektronik klien sederhana terdapat beberapa entitas antara lain : entitas pengguna, entitas akun, entitas kunci dan entitas pesan. Setiap pengguna yang sudah terdaftar pada aplikasi ini akan menjadi klien. Dan setiap klien mempunyai akun yang berisi data pribadi dan pasangan kunci (kunci rahasia dan kunci publik) yang masing-masing berbeda satu sama lain. Selain itu, setiap klien juga dapat melakukan proses pengiriman dan proses pembacaan pesan. Pada proses inilah terjadi enkripsi dan dekripsi pesan.



Gambar 2.1 Desain Arsitektur Sistem Aplikasi *E-mail*.

2.4.1 Perancangan *Entity Relationship Diagram* (ERD)

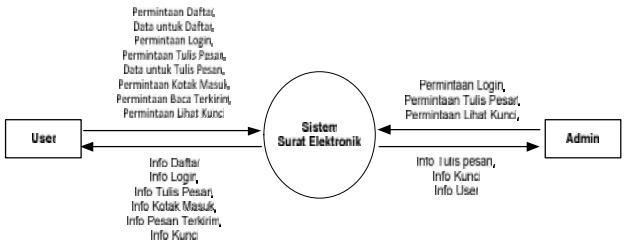
Pada proses perancangan sistem, ERD digunakan untuk menggambarkan interaksi yang terjadi antar entitas yang ada pada sistem surat elektronik dilihat dari sudut pandang pengguna (*user*). Gambar 2.2 menunjukkan ERD yang ada pada sistem.



Gambar 2.2 ERD Sistem Aplikasi Surat Elektronik.

2.4.2 Perancangan Diagram Konteks

Diagram konteks digunakan untuk menggambarkan interaksi yang terjadi antara sistem dengan dunia luar. Diagram konteks mencakup seluruh masukan/aliran data dari dan keluar sistem. Gambar 2.3 menunjukkan Diagram Konteks dari sistem.



Gambar 2.3 Diagram Konteks Sistem Aplikasi Surat Elektronik.

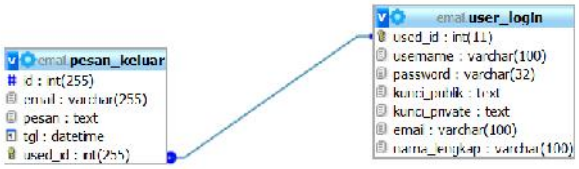
Dari diagram konteks di atas dapat diketahui kesatuan luar yang berinteraksi atau yang terlibat di dalam sistem, yaitu:

1. Admin
   Admin mempunyai hak akses untuk mengolah seluruh data, baik data pesan maupun data pengguna (*user*).
2. *User* / Pengguna
   Pengguna mempunyai hak akses untuk *login* ke dalam sistem dan mengolah data pesan (mengirim pesan dan membaca pesan masuk). Selain itu pengguna dapat melihat pasangan kunci yang dimiliki.

2.4.3 Perancangan Basisdata

Basisdata adalah kumpulan data yang secara logik berkaitan dalam merepresentasikan fakta secara terstruktur dalam domain tertentu untuk mendukung aplikasi pada sistem tertentu.

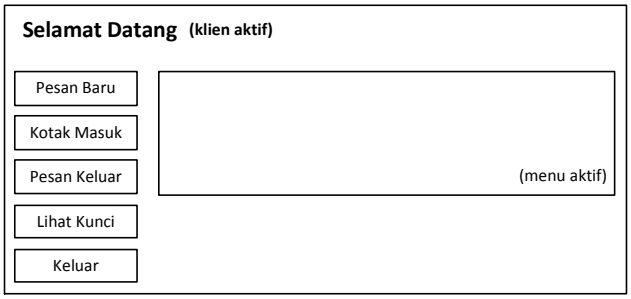
Pada sistem yang akan dibangun terdapat dua buah tabel, antara lain : pesan_keluar dan user_login. Tabel user_login digunakan untuk menyimpan data-data pribadi klien termasuk pasangan kunci yang digunakan untuk proses enkripsi dan dekripsi pesan. Sedangkan tabel pesan_keluar digunakan untuk menyimpan data pesan baik yang dikirim maupun diterima oleh masing-masing klien. Gambar 3.4 menunjukkan relasi antar tabel dalam sistem aplikasi surat elektronik sederhana :



Gambar 2.4 Relasi Antar Tabel.

2.4.4 Perancangan Antarmuka

Aplikasi surat elektronik berbasis *web* ini dibangun menggunakan bahasa pemrograman PHP. Salah satu tujuan yang perlu diperhatikan dalam membangun suatu aplikasi adalah *user friendly* atau mudah digunakan. Perancangan antarmuka terdiri dari antarmuka untuk login dan pendaftaran, antarmuka pengiriman pesan, antarmuka kotak masuk, antarmuka pesan terkirim dan antarmuka lihat pasangan kunci. Gambar 3.4 menunjukkan desain antarmuka menu utama aplikasi.



Gambar 2.5 Desain Halaman Menu Utama.

3. Hasil dan Analisa

Simulasi ini dilakukan untuk mengetahui apakah fungsi utama perangkat lunak yang dibuat sudah sesuai dengan kebutuhan sistem.

3.1 Pengujian Aplikasi

3.1.1 Pengujian Pendaftaran

Proses pendaftaran diawali dengan menekan tombol Daftar pada halaman utama aplikasi. pengguna diminta untuk mengisi dua buah masukan, yaitu nama lengkap pengguna dan alamat *email*. Pada halaman ini juga, pengguna mendapatkan nilai *p* dan *q* yang nantinya digunakan untuk membangkitkan pasangan kunci, yaitu kunci publik dan kunci private. Nilai *p* dan *q* merupakan bilangan prima yang dibangkitkan oleh sistem sehingga pengguna tidak bisa menentukan sendiri nilai *p* dan *q* yang didapat. Selain itu nilai *p* dan *q* untuk masing-masing pengguna berbeda satu sama lain. Gambar. 4.1 menunjukkan halaman daftar awal sistem.

Gambar 3.1 Halaman Daftar Awal Sistem.

Pada proses pembangkitan kunci, pengguna harus menekan tombol **Generate Kunci**. Kemudian muncul halaman baru yang menampilkan masukan yang lebih detail antara lain nama lengkap pengguna, alamat *email*,

nilai *p*, nilai *q*, kunci publik, kunci private, *username* dan *password*. Setelah semua masukan terisi, tekan tombol **Daftar**. Halaman daftar detail ditunjukkan oleh Gambar 4.2.

Gambar 3.2 Halaman Daftar Detail.

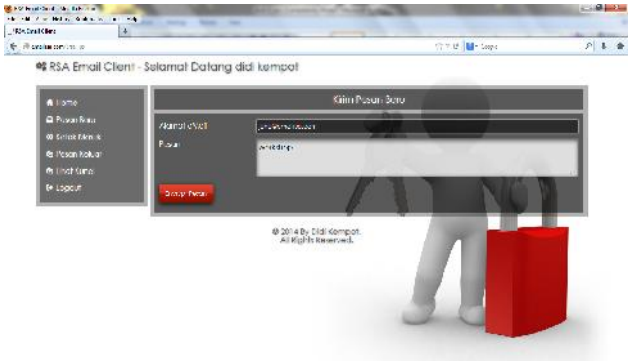
3.1.2 Pengujian *Login* Aplikasi

Jika pengguna yang sudah terdaftar atau klien akan menggunakan fasilitas yang ada pada aplikasi ini, klien harus masuk ke dalam sistem aplikasi melalui proses *login* dengan menekan tombol ***Login***. Gambar 4.3 menunjukkan proses *login* yang terjadi pada halaman awal aplikasi.

Gambar 3.3 Proses *Login*.

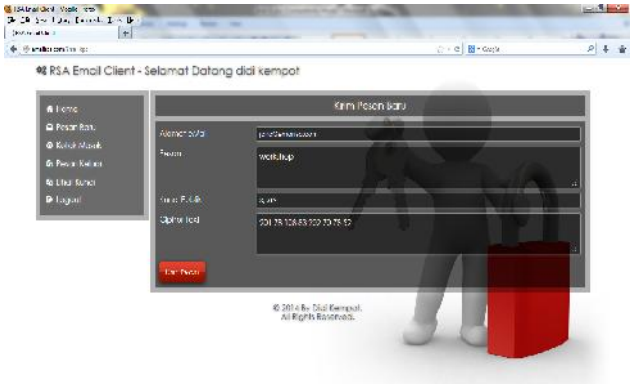
3.1.3 Pengujian Pengiriman Pesan

Pada menu **Pesan Baru**, klien diminta untuk memasukkan alamat penerima pesan dan isi pesan yang akan dikirimkan. Gambar 4.4 menunjukkan halaman awal proses penulisan pesan yang akan dikirim. Di halaman ini belum muncul tombol **Kirim Pesan** karena pesan belum melalui proses enkripsi.



Gambar 3.4 Halaman Tulis Pesan Awal.

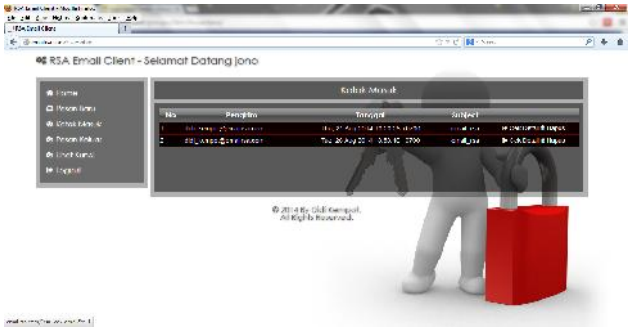
Untuk meng-enkripsi pesan/plainteks yang akan dikirimkan, klien diminta untuk menekan tombol **Encrypt Pesan**. Sehingga muncul pesan hasil enkripsi berupa cipherteks yang siap untuk dikirimkan kepada alamat tujuan seperti pada Gambar 4.5. Perlu diketahui bahwa aplikasi ini hanya bisa mengirimkan pesan kepada sesama pengguna aplikasi ini saja.



Gambar 3.5 Halaman Tulis Pesan Enkripsi.

3.1.4 Pengujian Baca Pesan Masuk

Seluruh daftar pesan yang diterima oleh klien dapat di lihat melalui menu **Kotak Masuk**. Daftar pesan yang telah diterima diurutkan berdasarkan waktu penerimaan pesan. Pada menu **Kotak Masuk** ini, terjadi proses dekripsi pesan. Gambar 4.6 menunjukkan halaman pesan masuk.



Gambar 3.6 Halaman Pesan Masuk.

Jika klien ingin melihat isi dari pesan yang diterimanya, klien bisa memilih salah satu pesan dan menekan tombol **Cek Detail**. Kemudian akan muncul halaman lihat pesan. Pada halaman ini pesan yang diterima masih berupa cipherteks. Gambar 4.7 menunjukkan halaman detail pesan masuk.

Gambar 3.7 Halaman Detail Pesan Masuk.

Untuk melihat pesan asli dari pesan yang telah diterima, klien harus menekan tombol **Decrypt Pesan** dan muncul halaman detail pesan yang menampilkan baik cipherteks maupun pesan asli atau plainteksnya. Pasangan kunci yang digunakan pada menu Pesan Masuk merupakan pasangan kunci publik dan kunci private milik penerima pesan. Gambar 4.8 menunjukkan halaman dekrip pesan masuk.

Gambar 3.8 Halaman Dekrip Pesan Masuk.

3.1.5 Pengujian Lihat Pasangan Kunci

Pada menu **Lihat Kunci** menampilkan pasangan kunci yang dimiliki oleh klien. Pasangan kunci bernilai beda untuk masing-masing klien yang terdaftar. Nilai dari kunci tersebut tidak dapat diubah, karena sudah digeneralisasi oleh sistem pada saat klien melakukan pendaftaran. Gambar 4.9 menunjukkan halaman lihat kunci.

Gambar 3.9 Halaman Lihat Kunci.

3.2 Pengujian Manual

Pengujian secara manual dilakukan dengan tujuan membandingkan hasil akhir dari proses enkripsi berupa cipherteks dan hasil akhir proses dekripsi berupa plainteks. Dari pengujian aplikasi diatas, data yang digunakan untuk pengujian adalah seperti tabel berikut.

Tabel 3.1. Data Pengujian Program.

Keterangan	Nilai
Pesan (*m*) / Plainteks	workshop
Kunci Publik (*e, n*)	(*3,319*)

3.2.1 Pengujian Enkripsi

Langkah-langkah penyelesaian proses enkripsi secara manual adalah sebagai berikut.
Diketahui :

Plainteks (*m*) : “workshop”
Kunci publik (*e*, *n*) : (3, 319)

Jawab :

1) Ambil kunci publik penerima pesan (*e*, *n*) = (3, 319).
2) Ubah plainteks (*m*) menjadi pesan dengan nilai ASCII. w = 119, o = 111, r = 114, k = 107, s = 115, h = 104, o = 111, p = 112.
3) Potong pesan menjadi blok-blok pesan m_1, m_2, m_3, ... dengan nilai setiap bloknya adalah $0 \le m \le n - 1$.
   m_1 = 119, m_2 = 111, m_3 = 114, m_4 = 107, m_5 = 115, m_6 = 104, m_7 = 111,
   m_8 = 112.

4) Setiap blok *m* dihitung dengan rumus $c_i = m_i^e$ **mod** *n*.
   $c_1 = 119^3 \bmod 319 = 201$
   $c_2 = 111^3 \bmod 319 = 78$
   $c_3 = 114^3 \bmod 319 = 108$
   $c_4 = 107^3 \bmod 319 = 83$
   $c_5 = 115^3 \bmod 319 = 202$
   $c_6 = 104^3 \bmod 319 = 70$
   $c_7 = 111^3 \bmod 319 = 78$
   $c_8 = 112^3 \bmod 319 = 52$
5) Susun nilai *c* hasil enkripsi dengan susunan c_1, c_2, c_3,...., c_n sehingga diperoleh cipherteks dari pesan *m*.
   Cipherteks : 201, 78, 108, 83, 202, 70, 78, 52

3.2.2 Pengujian Dekripsi

Langkah-langkah penyelesaian proses enkripsi secara manual adalah sebagai berikut.
Diketahui :

Cipherteks (*m*) : "201, 78, 108, 83, 202, 70, 78, 52".
Kunci rahasia (*d, n*) : (187, 319)

Jawab :
1) Ambil pesan (cipherteks) yang telah diterima.
   Cipherteks (*m*) : "201, 78, 108, 83, 202, 70, 78, 52".
2) Kemudian ambil kunci rahasia (*d*,*n*) = (187, 319).
3) Potong pesan menjadi blok-blok pesan c_1, c_2, c_3, ... dengan nilai setiap bloknya adalah $0 \le c \le n - 1$.
   $c_1 = 201$, $c_2 = 78$, $c_3 = 108$, $c_4 = 83$, $c_5 = 202$, $c_6 = 70$, $c_7 = 78$, $c_8 = 52$.
4) Hitung $m_i = c_i^d$ **mod** *n*.
   $m_1 = 201^3 \bmod 319 = 119$
   $m_2 = 78^3 \bmod 319 = 111$
   $m_3 = 108^3 \bmod 319 = 114$
   $m_4 = 83^3 \bmod 319 = 107$
   $m_5 = 202^3 \bmod 319 = 115$
   $m_6 = 70^3 \bmod 319 = 104$
   $m_7 = 78^3 \bmod 319 = 111$
   $m_8 = 52^3 \bmod 319 = 112$
5) Ubah nilai m_1, m_2, m_3,....,m_n sesuai dengan tabel ASCII sehingga diperoleh plainteks (pesan asli) dari cipherteks yang diterima.
   119 = w, 111 = o, 114 = r, 107 = k, 115 = s, 104 = h, 111 = o, 112 = p.
6) Susun nilai ASCII yang dihasilkan. Hasil penyusunan inilah yang merupakan pesan asli (plainteks) setelah melalui proses dekripsi.
   Pesan asli (plainteks) : "`workshop`"

3.3 Analisis Keamanan Algoritma Kriptografi RSA

Terdapat beberapa serangan-serangan yang dapat dilakukan terhadap algoritma kriptografi RSA. Pada subbahasan ini akan dibahas dua buah serangan yang umum dilakukan oleh kriptanalis terhadap algoritma kriptografi RSA yaitu serangan faktorisasi dan serangan *Brute-Force*.

3.3.1 Serangan Terhadap Cipherteks Saja

Dengan metode ini kriptanalis hanya memiliki cipherteks, dan akan berusaha menemukan plainteks. Dengan melihat percobaan enkripsi pesan pada pengujian diatas dapat diketahui bahwa :
Cipherteks : `2017810883202707852`
Pesan asli/plainteks-nya adalah : `workshop`

Dari hasil perbandingan antara cipherteks dan plainteks terlihat bahwa panjang cipherteks yang dihasilkan tidak sama dengan panjang plainteks. Hal ini terjadi dikarenakan pada proses enkripsi setiap karakter pesan akan diubah ke dalam bentuk ASCII. Proses tersebut membuat jumlah karakter pada cipherteks lebih banyak dibandingkan dengan jumlah plainteks, sehingga panjang pesan yang dihasilkan (cipherteks) juga akan bertambah. Perbedaan panjang tersebut akan menyulitkan menebak plainteks (pesan asli) meskipun kriptanalis memiliki cipherteks.

3.3.2 Serangan Faktorisasi

Serangan yang sering dilakukan terhadap algoritma kriptografi RSA adalah faktorisasi. Penyerangan ini bertujuan untuk memfaktorkan nilai *n* menjadi dua buah faktor primanya yaitu *p* dan *q*. Jika *p* dan *q* berhasil difaktorkan, fungsi euler $\varphi(n) = (p - 1)(q - 1)$ akan dapat dikomputasi dengan mudah dan kemudian kunci rahasia (*d*, *n*) dapat diketahui. Fungsi totien euler φ mendefinisikan $\varphi(n)$ untuk $n \ge 1$ yang menyatakan bilangan bulat positif < *n* yang relatif prima dengan *n*.

Dari pengujian diatas diketahui bahwa kunci publik (*e*, *n*) = (3, 319). Setelah diketahui nilai *n* adalah 319, maka kriptanalis akan mencari nilai $\varphi(319)$. Untuk menemukan nilai *p* dan *q*, kriptanalis harus mencoba seluruh bilangan bulat positif yang benilai lebih kecil dari *n* yang relatif prima dengan *n*.

3.3.3 Serangan Brute-Force

Serangan *Brute-Force* merupakan sebuah teknik serangan terhadap sebuah sistem keamanan komputer dengan melakukan percobaan terhadap kombinasi pasangan kunci rahasia (*d*, *n*) yang mungkin. Nilai *n* dapat diketahui karena bersifat tidak rahasia dan digunakan pada saat proses enkripsi.

Percobaan dilakukan dengan asumsi kriptanalis mengetahui cipherteks dan kunci publik (*e*, *n*). Percobaan dilakukan sebanyak tiga kali dengan mengkombinasikan nilai *d* dan *n* yang mungkin. Tabel 4.2 menunjukkan

kombinasi pasangan kunci rahasia untuk serangan *Brute-Force*.

Tabel 3.2. Data Pasangan Kunci Percobaan Serangan *Brute-Force*.

Percobaan	Kunci Publik	Kunci Rahasia	Cipherteks
1	(3, 319)	(83, 319)	201, 78, 108, 83, 202, 70, 78, 52
2	**(3, 319)**	**(89, 319)**	**201, 78, 108, 83, 202, 70, 78, 52**
3	(3, 319)	(97, 319)	201, 78, 108, 83, 202, 70, 78, 52

Percobaan dilakukan dengan dekripsi terhadap cipherteks yang telah di dapat dengan menggunakan kemungkinan kunci pribadi yang ada. Kemudian akan dibandingkan plainteks hasil dari percobaan serangan *Brute-Force* dengan plainteks asli. Berikut ini adalah proses dekripsi dari salah satu percobaan serangan *Brute-Force* menggunakan kunci rahasia (89, 319).

$m_1 = 201^{89} \mod 319 = 26$
$m_2 = 78^{89} \mod 319 = 111$
$m_3 = 108^{89} \mod 319 = 60$
$m_4 = 83^{89} \mod 319 = 310$
$m_5 = 202^{89} \mod 319 = 289$
$m_6 = 70^{89} \mod 319 = 157$
$m_7 = 78^{89} \mod 319 = 111$
$m_8 = 52^{89} \mod 319 = 315$

Tabel 3.3. Data Hasil Dekripsi Pada Percobaan Serangan *Brute-Force*.

Percobaan	Kunci Rahasia	Plainteks
1	(83, 319)	m_1 = 159, m_2 = 45, m_3 = 47, m_4 = 7, m_5 = 86, m_6 = 75, m_7 = 45, m_8 = 314
2	**(89, 319)**	**m_1 = 26, m_2 = 111, m_3 = 60, m_4 = 310, m_5 = 289, m_6 = 157, m_7 = 111, m_8 = 315**
3	(97, 319)	m_1 = 218, m_2 = 45, m_3 = 301, m_4 = 239, m_5 = 115, m_6 = 302, m_7 = 45, m_8 = 24

Dari tabel percobaan diatas, plainteks masih bernilai desimal sehingga perlu diubah melalui tabel ASCII untuk mengetahui isi dari plainteks aslinya. Tabel 4.4 menunjukkan isi plainteks setelah diubah melalui tabel ASCII.

Tabel 3.4. Data Plainteks Setelah ASCII Pada Percobaan Serangan *Brute-Force*.

Percobaan	Kunci Rahasia	Plainteks
1	(83, 319)	m_1 = (kosong), m_2 = -, m_3 = /, m_4 = BEL, m_5 = V, m_6 = K, m_7 = -, m_8 = (kosong)
2	(89, 319)	m_1 = **SUB**, m_2 = **o**, m_3 = **<**, m_4 = (kosong), m_5 = (kosong), m_6 = (kosong), m_7 = **o**, m_8 = (kosong)
3	(97, 319)	m_1 = (kosong), m_2 = -, m_3 = (kosong), m_4 = (kosong), m_5 = s, m_6 = (kosong), m_7 = -, m_8 = CANCEL

Jadi plainteks dari hasil serangan *Brute-Force* tidak ada yang sesuai dengan plainteks pesan asli, yaitu "`workshop`". Pengujian *Brute-Force* didasarkan pada jumlah karakter kunci *d*. Jumlah karakter kunci ini akan menentukan banyaknya jumlah percobaan yang harus dilakukan untuk mendapatkan plainteks.

Tabel 3.5. Jumlah Bit Kunci Pada Percobaan Serangan *Brute-Force*.

Percobaan	Kunci Rahasia	Jumlah bit Kunci
1	(**83**, 319)	2 karakter kunci atau (16 bit)
2	(**89**, 319)	2 karakter kunci atau (16 bit)
3	(**97**, 319)	2 karakter kunci atau (16 bit)

Dari tabel diatas dapat diketahui bahwa jumlah bit kunci adalah 2 karakter kunci atau (16 bit). Maka dengan menggunakan rumus jumlah kemungkinan kunci yang mungkin adalah sebagai berikut:

Jumlah kemungkinan kunci $= 2^{\text{bit kunci}}$
$= 2^{16}$
$= 65.536.$

Jadi jumlah kemungkinan kunci yang dapat dicoba pada serangan *Brute-Force* adalah sebanyak 65.536 buah kunci.

4. Kesimpulan

Algoritma kriptografi RSA bekerja berdasarkan tiga proses, yaitu proses pembentukan kunci untuk memperoleh pasangan kunci publik dan kunci rahasia kemudian proses enkripsi dan proses dekripsi terhadap data/informasi yang akan dtransmisikan. Pada aplikasi yang dibuat menunjukkan bahwa algoritma kriptografi nirsimetri RSA sangat baik untuk mengatasi masalah manajemen distribusi kunci yaitu dengan menyimpan pasangan kunci pada basisdata sedangkan kunci yang di distribusikan hanya kunci publik. Algoritma RSA termasuk algoritma yang baik (aman secara komputasi). Dengan jumlah cipherteks yang lebih banyak dari plainteks mengakibatkan faktor kerja yang dibutuhkan untuk melakukan pemecahan chiperteks membutuhkan waktu yang lebih lama. Dengan membuat panjang faktor prima dari *p* dan *q* menjadi tidak seimbang (nilai *p* jauh lebih kecil dari nilai *q*) maka waktu yang diperlukan untuk melakukan serangan faktorisasi terhadap nilai *n* akan semakin lama. Semakin panjang karakter kunci yang digunakan maka semakin lama waktu yang dibutuhkan untuk menemukan plainteks, sehingga dapat dikatakan semakin panjang karakter kunci semakin kuat juga terhadap serangan *Brute-Force*. Salah satu cara untuk menjaga keamanan dan kerahasiaan data/informasi pada saat ditransmisikan adalah dengan melakukan penyandian terhadap data/informasi, sehingga pihak lain yang tidak berkepentingan tidak dapat mengetahui isi dari data/informasi tersebut.

Referensi

[1]. Schneier, Bruce. "*Applied Cryptography 2nd*". John Wiley and Sons, 1996.

[2]. Menezes, Alfred J., Paul C van Oorschot, and Scoot A.Vanstone. "*Handbook of Applied Cryptography*". CRCPress. New Jersey, 1996.

[3]. Munir, Rinaldi. "*Kriptografi*". Informatika. Bandung, 2006.

[4]. Stalling, W. "*Cryptography and Network Security, Principle and Practice Second Edition*". Pearson Education, Inc., 1998.

[5]. Bose, Ranjan. "*Informaton Theory, Coding and Cryptography*". Tata McGraw-Hill Publishing Company Limited. New Delhi, 2002.

[6]. Nugroho, Bunafit. "*Aplikasi Pemrograman Web Dinamis dengan PHP dan MySQL*". Gava Media. Yogyakarta, 2004.

[7]. ..., "ASCII Table and Description", www.AsciiTable.com, 17 Maret 2012, 10:30