REKAYASA PERANGKAT LUNAK KRIPTOGRAFI MENGGUNAKAN ALGORITMA RSA PADA SISTEM KEAMANAN FILE BERBASIS JAVA

Oleh:
Busran*, Novernus Ayundha Putra**
*Dosen Jurusan Teknik Informatika
**Mahasiswa Jurusan Teknik Informatika
Fakultas Teknologi Industri, Institut Teknologi Padang

Abstract

Data security is a top priority on a data storage device, using cryptographic techniques as a method of encryption can increase system security files are confidential, data security techniques can make use of the RSA algorithm the algorithm which makes use of two keys on the encoding process and its return to the pure message back. By making use of the RSA algorithm as the algorithm to encode the data, then the algorithm can be applied with designing a software using the java programming language.

This research aims to be able to design a device that makes use of the RSA algorithm toughness as the encoding algorithm for data and can be installed by building devices that use the java programming language so that it can help the user in securing confidential data before performing the process of storage on storage media. With the creation of this device, is expected to help the user in securing confidential data, and can understand the concept of the RSA algorithm.

***Keywords** : Cryptography, RSA Algorithm, Java, Data Security System*

1.1 Latar Belakang

Penggunaan teknologi komputer sebagai salah satu media penyimpanan dan komunikasi menjadi suatu kebutuhan yang tidak dapat dipisahkan lagi disetiap kegiatan, baik dalam bidang pendidikan, dunia kerja ataupun pada bidang lainnya. Berbagai informasi yang diperoleh akan dimanfaatkan dan didistribusikan untuk berbagai kepentingan. Namun dengan perkembangan tersebut terdapat permasalahan yang perlu diperhatikan. Penggunaan teknologi sebagai media penyimpanan dan komunikasi secara luas memungkinkan pihak-pihak yang tidak memiliki kepentingan memanfaatkannya sehingga membahayakan integritas dari informasi tersebut. Oleh karena itu, dibutuhkan suatu mekanisme yang dapat mengamankan informasi tersebut dari pihak yang tidak memiliki kepentingan. Salah satu algoritma yang memanfaatkan kriptografi sebagai sistem keamanan adalah algoritma RSA (Ron Rivest, **S**hamir, dan Leonard **A**dleman).

RSA merupakan suatu algoritma yang populer digunakan karena kesederhanaan, serta memiliki kecepatan proses yang tidak kalah cepat dibandingkan dengan algoritma kriptografi lainnya. Secara umum algoritma RSA banyak digunakan pada proses penyandian yang bersifat *software* aplikasi web, contohnya adalah penggunaan algoritma RSA pada proses penyandian informasi yang bersifat rahasia secara *online*, sedangkan salah satu kelemahan sistem penyandian yang bersifat *online* adalah pengguna diharuskan terkoneksi dengan jaringan internet setiap ingin melakukan proses enkripsi dan dekripsi informasi rahasia. Maka sebagai solusi dari kelemahan diatas adalah dengan membangun *software* aplikasi kriptografi berbasis *desktop* yang dapat dijalankan tanpa harus terkoneksi ke jaringan internet.

Dari latar belakang tersebut maka penulis tertarik untuk membuat sebuah penelitian mengenai perancangan dengan mengaplikasian algoritma kriptogafi RSA pada suatu sistem keamanan file, penulispun

mencoba membuat sebuah aplikasi yang dapat membantu menangani masalah tersebut dengan judul penelitian **"REKAYASA PERANGKAT LUNAK KRIPTOGRAFI MENGGUNAKAN ALGORITMA RSA PADA SISTEM KEAMANAN FILE BERBASIS JAVA".**

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas maka penulis mencoba merumuskan suatu masalah yaitu "Bagaimana merancang sistem dengan mengimplementasikan algoritma RSA berbasis pemograman Java sebagai sistem keamanan sebuah informasi?".

1.3 Batasan Masalah

Batasan masalah pada penelitian ini adalah :

1. Mengimplementasikan algoritma RSA sebagai sistem penyandian.
2. Pengimplementasian algoritma RSA pada program aplikasi berbasis Java.
3. Format file yang digunakan adalah file *.txt.
4. Batas pembuatan kunci adalah 512-bit dan 1024-bit.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah merancang sistem kriptografi dengan mengimplementasikan algoritma RSA sebagai salah satu algoritma penyandian yang diaplikasikan pada program berbasis Java.

1.5 Manfaat Penelitian

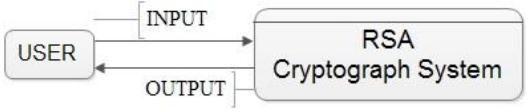
Manfaat penelitian yaitu :

1. Membantu mengamankan data file yang bersifat rahasia berbasis pemrograman Java.
2. Mengetahui pengaplikasian algoritma RSA pada proses enkripsi maupun dekripsi pada sistem kriptografi.

2. Metodologi

2.1 Context Diagram

Context diagram digunakan sebagai alat bantu dalam merancang sistem secara global dengan memperlihatkan sistem secara umum. Berikut context diagram penelitian untuk aplikasi kriptografi algoritma RSA berbasis JAVA :

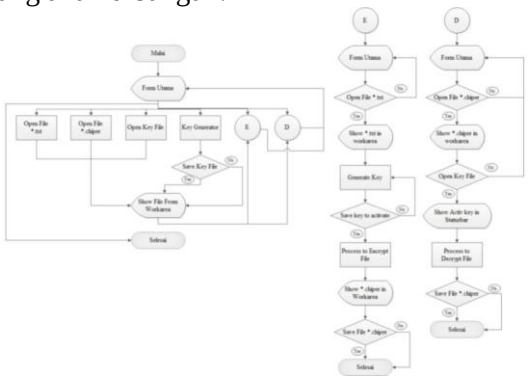


Gambar 1 Context Diagram

Pada gambar diatas dapat dijelaskan bahwa user sebagai pengguna aplikasi dapat menginputkan permintaan sesuai kebutuhan yang disediakan pada sistem RSA Cryptograph System yang dirancang.

2.2 Tahapan Perancangan Antar Muka Sistem

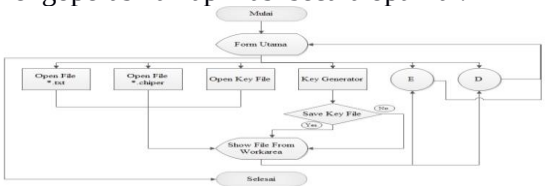
Perancangan antarmuka berisi pemaparan user interface yang digunakan dalam perancangan aplikasi. Dengan perancangan antar muka ini diharapkan dapat memudahkan pengguna dalam mengoperasikan aplikasi. Berikut rancangan flowchart program yang akan dibangun.



Gambar 2 Rancangan Flowchart Antar Muka Aplikasi

2.2.1 Tampilan Desktop

Rancangan tampilan program aplikasi yang akan dibangun terdiri dari beberapa content yang dirancang seinteraktif mungkin sehingga memungkinkan pengguna dapat mengoperasikan aplikasi secara optimal.



Gambar 3 Rancangan Flowchart Antar Muka *Desktop*

Berikut rancangan tampilan aplikasi program kriptografi RSA yang akan dibangun :

X _ RSA Cryptograph v.01
Main Menu
Toolbar
Workarea
Statusbar

Gambar 4 Rancangan antar muka *Desktop* Aplikasi

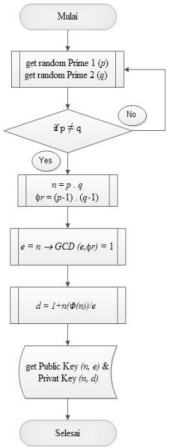
2.2.2 Tampilan Form Generate Key

Form generate key merupakan form yang digunakan untuk melakukan proses pembuatan kunci publik dan kunci privat, pada form ini disediakan interface panjang kunci 512bit dan 1024bit, generate, save, dan close, serta juga disediakan baris yang berisi product, publik key, private key dari hasil generate. Berikut rancangan tampilan form key generator,

X _ Key Generator		
Product (r)		
Public Key		
Private Key		
Key Length :	O 512-bit	O 1024-bit
Generate	Save	Close

Gambar 5 Rancangan interface Key Generator

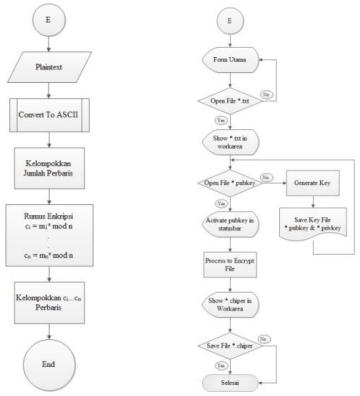
Dari rancangan tampilan form key generator diatas didapat dari proses pembuatan kunci yang divisualkan dalam bentuk flowchart berikut,



Gambar 6 Flowchart Pembuatan Kunci

2.2.3 Proses Enkripsi

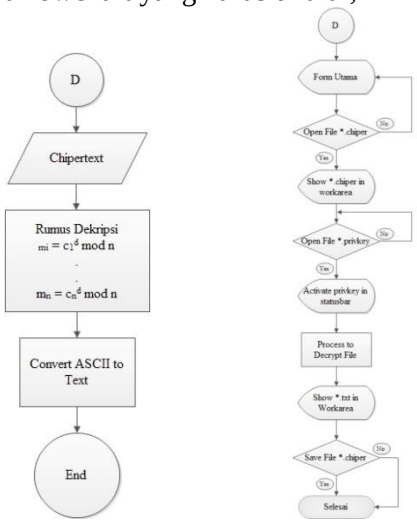
Pada proses enkripsi ada beberapa tahapan yang harus dilalui pengguna sebelum mendapatkan hasil file yang telah dienkripsi, berikut flowchart untuk melakukan proses enkripsi,



Gambar 7 Flowchart Proses Enkripsi File Dengan Algoritma RSA dan Flowchart Tahapan Enkripsi pada Aplikasi yang Dibangun

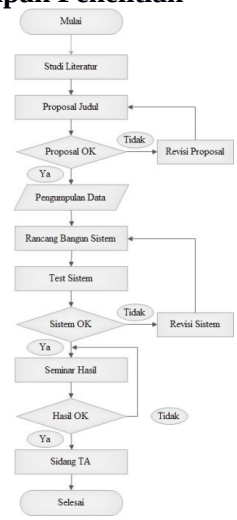
3.6.4 Proses Deskripsi

Pada proses dekripsi ada beberapa tahapan yang harus dilalui pengguna sebelum mendapatkan hasil file yang telah didekripsi, berikut flowchart yang harus dilalui,



Gambar 8 Flowchart Proses Dekripsi File dengan algoritma RSA dan Flowchart Tahapan Dekripsi pada Aplikasi yang Dibangun

2.3 Tahapan Penelitian



Gambar 9 Flowchart Penelitian

3. Pembahasan dan Analisa

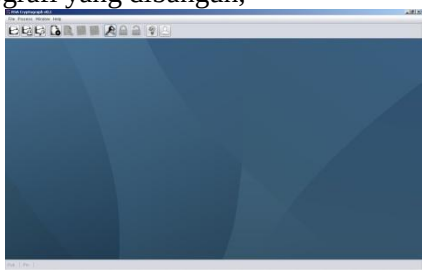
Sebelum melakukan perancangan, maka perlu mempersiapkan *tools* sebagai pembantu dalam perancangan dan pembangunan sistem, diantaranya adalah dengan menggunakan sistem operasi berbasis *Open Source* yaitu Linux Ubuntu 12.04. Dalam tahapan berikut akan dijelaskan tahapan proses instalasi *tools* sebagai penunjang pembangun program, rancangan tampilan, serta lankah-langkah menjalankan aplikasi hasil perancangan sebagai proses pengujian program.

3.1 Proses Pembangunan Perangkat Lunak Kriptografi RSA

3.1.1 Halaman Utama Aplikasi

Aplikasi Sistem Kriptografi yang berbasiskan pemrograman berorientasi objek ini dibangun dengan tujuan untuk dapat melakukan proses penyandian terhadap data file yang bersifat rahasia sebelum data tersebut dipublikasi. Sistem yang dibangun ini bersifat umum dan terbuka, yang berarti pengguna program ini bebas menggunakannya selama program yang akan dibangun ini terinstal pada perangkat yang digunakan dan dapat digunakan oleh siapa saja.

Pada aplikasi kriptografi yang dibangun ini terdiri dari beberapa Form yang berfungsi untuk menampilkan file yang akan di enkripsi, dekripsi, serta tampilan proses untuk mendapatkan private dan public key. Berikut adalah tampilan dari Halaman Utama aplikasi kriptografi yang dibangun,



Gambar 10 Tampilan Halaman Utama

Dari tampilan Halaman Utama diatas, dapat dilihat bahwa program ini pada dasarnya dibangun dengan menggunakan beberapa komponen sehingga dapat menampilkan jendela program seperti diatas, pembangun jendela utama program diatas menggunakan komponen script program diantaranya,

```
public void run() {
      FrmUtama frm = new
FrmUtama();
      frm.setVisible(true);
}
```

Pada method diatas dapat dilihat bahwa perintah untuk menampilkan form utama pada aplikasi adalah dengan menciptakan sebuah variabel FrmUtama, perintah untuk menampilkan form utama tersebut adalah dengan mengatur visible dari form tersebut menjadi true. Untuk mengatur properties dari halaman tersebut dapat dilihat pada lampiran dengan method FrmUtama. Pada pengaturan properties halaman utama aplikasi dapat diatur ukuran halaman yang akan terbuka diatur defaultnya pada posisi minimum 800x700 dan pada posisi maksimum halaman 1024x786 dengan title RSA Cryptograph v0.1. untuk menampilkan aplikasi dengan icon yang telah ditentukan adalah dengan meload resource variabel appIcon pada perintah ImageIcon dan menuju folder tempat icon disimpan yaitu pada folder /res/images/icon.png.

3.1.1.1 Komponen Halaman Utama Aplikasi

Form ini merupakan halaman utama untuk mengakses aplikasi sistem kriptografi. Pada halaman ini terdapat pilihan untuk mengakses item menu selanjutnya. Menu merupakan daftar perintah-perintah atau kumpulan baris perintah suatu perangkat lunak

(program) yang apabila di eksekusi akan menjalankan suatu tugas tertentu, apabila di klik akan muncul beberapa perintah untuk masuk ke form yang dibutuhkan pengguna. Selain menu-menu, form ini juga terdiri dari toolbar yang bisa digunakan oleh pengguna selain menggunakan menu-menu yang disediakan, icon dari toolbar yang disediakan dapat mewakili dari proses yang dijalankan ketika pengguna menggukan tombol tersebut sebagai permintaan kepada sistem.

Gambar 11 Komponen Pembangun Halaman Utama

Berikut komponen pembangun halaman utama pada script aplikasi

```
// komponen halaman utama
appIcon = new
ImageIcon(FrmUtama.class.getRes
ource("/res/images/icon.png"));
setTitle("RSA Cryptograph
v0.1");
```

Pada komponen form utama, icon diatur pada appIcon dengan mengatur tujuan pembukaan image pada folder /res/images/icon.png serta pengaturan title halaman adalah dengan mengaturnya menjadi RSA Cryptograph v0.1

```
// komponen menubar
JMenuBar menuBar = new
JMenuBar();
setJMenuBar(menuBar);
```

Komponen pembangun halaman utama lainnya adalah dengan menambahkan komponen menu yang diatur dengan perintah new JMenuBar, dan mengatur Jmenubar menjadi komponen menubar.

```
// komponen toolbar
toolbar = new RSAToolbar();
toolbar.btEncFile.addActionList
ener(new ActionListener() {
public void
actionPerformed(ActionEvent
evt) {
btEncFileActionPerformed(evt);
```

Komponen selanjutnya adalah komponen toolbar yang berfungsi sebagai tombol cepat untuk melakukan proses yang diinginkan oleh pengguna, tombol–tombol ini diatur dengan method sendiri dengan nama method RSAToolbar.

```
// komponen workarea
JPanel workarea = new JPanel();
desktopPane = new
JDesktopPane();
desktopPane.setAutoscrolls(true
);
GroupLayout gl_workarea = new
GroupLayout(workarea);
```

Workarea adalah komponen penting yang menyediahan halaman untuk melakukan proses yang dilakukan oleh sistem, workarea dibangun dengan menggunakan komponen JPanel yang diberi nama variabel JPanel workarea dengan settingan desktopPane Autoscrol menjadi true.

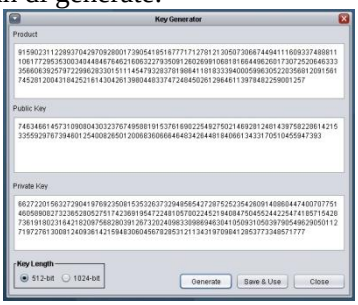
```
// komponen statusbar
JPanel statusbar = new
JPanel();
statusbar.setBorder(new
EtchedBorder(EtchedBorder.RAISE
D, null, null));
```

Statusbar adalah komponen selanjutnya yang membantu pengguna dalam menjalankan aplikasi. Untuk mengatur statusbar diperlukan sebuah panel yang diambil dari JPanel yang tersedia.

3.1.1.2 Key Generator

Form Key Generator merupakan fasilitas untuk melakukan proses enkripsi sebuah file. Key generator berfungsi melakukan proses pembuatan file kunci sebagai kunci pengunci sebuah file dan sebagai kunci pembuka dari sebuah file yang terkunci. Konsep dari algoritma RSA menjelaskan bawa algoritma RSA merupakan algoritma dengan tipe asimetris, maksudnya adalah algoritma ini memiliki dua buah kunci yang digunakan untuk mengunci dan yang digunakan untuk membuka. Dari konsep itulah penulis mencoba mempresentasikannya dalam sebuah form Key Generator. Dibawah ini merupakan interface pembuatan kunci publik dan kunci pribadi

dengan kontem tambahan berupa panjang kunci yang akan di generate.



Gambar 12 Tampilan form Key Generator

Dari tampilan diatas, berikut adalah algoritma membuatan kuncinya,

```
public static BigInteger[]
getTwoRandomPrimeBI(Random
random, int bitLength) {
BigInteger[] hasil = { new
BigInteger("0"), new
BigInteger("0") };
while (
hasil[0].equals(hasil[1]) ) {
hasil[0] =
getRandomPrimeBI(random,
bitLength);
hasil[1] =
getRandomPrimeBI(random,
bitLength);           }
return hasil;
      }
```

1. Dapatkan dua bilangan prima random direpresentasikan dengan p dan q sepanjang bit pilihan (berdasarkan pilihan pada form generate key)
   - hasil[0] = *getRandomPrimeBI*(random, bitLength);
   - hasil[1] = *getRandomPrimeBI*(random, bitLength);

```
      public static BigInteger
getProduct(BigInteger[]
twoPrime) {
            BigInteger hasil =
twoPrime[0].multiply(twoPrime[1
]);
            return hasil;
      }
```

2. Dapatkan product *n*, dimana product adalah hasil kali p dan q,
   - BigInteger hasil = twoPrime[0].multiply(twoPrime[1]);

```
      public static BigInteger
getPhi(BigInteger[] twoPrime) {
      BigInteger pMin1 =
twoPrime[0].subtract(BigInteger
.ONE);
      BigInteger qMin1 =
twoPrime[1].subtract(BigInteger
.ONE);
      BigInteger hasil =
pMin1.multiply(qMin1);
            return hasil;
      }
```

3. Dapatkan *phi*,
   - phi = (p-1).(q-1), BigInteger hasil = pMin1.multiply(qMin1);
4. Dapatkan public key random yang relatif prima dengan *phi*
   - pubKey = BigInteger.*probablePrime*(bitLength, random);
5. Dapatkan privat key $d = 1 + k\ (phi) / e$
   - BigInteger hasil = pubKey.modInverse(phi);

3.1.1.3 Proses Enkripsi File

Proses enkripsi adalah suatu proses yang dilakukan setelah dilakukannya proses pembuatan kunci. Proses ini meliputi proses pembukaan file berformat *.txt, selanjutnya adalah proses pembukaan kunci publik, dan dilanjutkan dengan proses pengenkripsian isi file.

Setelah file berhasil dienkripsi dengan menggunakan kunci publik, maka aplikasi akan menhasilkan sebuah file baru dengan format *.chipertext.

maka dapat diaplikasikan pada listing program sebagai berikut.

```
public     static     BigInteger
encrypt(BigInteger        ascii,
BigInteger  product,  BigInteger
pubKey ) {
            BigInteger   hasil   =
ascii.modPow(pubKey, product);
            return hasil;
      }
```

Dari fungsi diatas, dapat dijabarkan bahwa proses enkripsi RSA menggunakan rumus

$c_i = m_1^e \bmod n$

Kunci public didapatkan dengan memanfaatkan fungsi generate key sebelum dilakukannya proses enkripsi.

3.1.1.4 Proses Dekripsi File

Dekripsi adalah proses konversi informasi yang telah disandikan sebelumnya menjadi sebuah informasi asli dan utuh secara konsisten sehingga informasi tersebut dapat dipahami oleh pengguna dan dapat dimanfaatkan untuk keperluan lainnya. Pada proses dekripsi, ada beberapa persyaratan yang hasus dipenuhi oleh pengguna, diantaranya adalah, file yang akan didekripsi harus merupakan file dengan format *.chipertext. Dimana file dengan format ini adalah file yang akan didekripsi. Selanjutnya pengguna juga harus memiliki kunci privat yang akan digunakan oleh saplikasi sebagai kunci untuk membuka file tersebut sehingga informasi dapat dibaca oleh pengguna.

Setelah persyaratan tesebut dipenuhi, maka barulah pengguna baru dapat melakukan proses dekripsi dengan memanfaatkan aplikasi ini untuk dapat menghasilkan sebuah file dengan format *.txt.

Berikut adalah potongan fungsi dari proses dekripsi pesan dengan menggunakan rumus dekripsi menggunakan algoritma RSA yang dibangun

```
public static BigInteger decrypt(BigInteger ascii, BigInteger product, BigInteger privKey ) {
BigInteger hasil = ascii.modPow(privKey, product);
return hasil; }
```

Dari fungsi diatas, dapat dijabarkan bahwa proses dekripsi RSA menggunakan rumus

$$m_i = c_1^d \bmod n$$

Dapat dijabarkan kedalam source program dengan menggunakan method decrypt yang mana menggunakan tipe data BigInteger pada variabel ascii, product dan privkey. Untuk mendapatkan fingsi decrypt adalah dengan mengkonversikan bilangan ascci decimal kedalam bentuk karakter setelah dilakukan proses pemotongan perdigit chipertext lalu dipangkatkan dengan kunci privat dan di cari sisa hasil baginya dengan *n* atau *product* sehingga didapat bilangan *ascii*.

Kunci privat didapatkan dengan memanfaatkan fungsi generate key sebelum dilakukannya proses dekripsi.

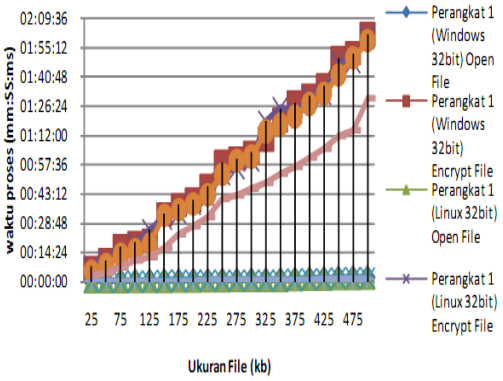
3.2 Analisa Sistem Aplikasi RSA Kriptografi

Berikut merupakan tabel hasil pengujian file dengan variabel yang telah ditentukan berikut grafik serta hasil analisa dari pengujian aplikasi berdasarkan variabel yang telah ditentukan diatas yang menyatakan bahwa berdasarkan variabel pengujian dapat diambil kesimpulan :

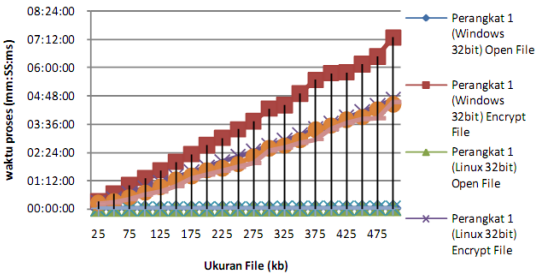
Lampiran 1

Lampiran 2

Dari tabel diatas, maka dapat dilakukan analisa dengan memanfaatkan variabel yang telah ditentukan. Berikut tampilan grafik dari proses analisa sistem.



Gambar 13 Grafik Master hasil pengujian Sistem (kunci 512-bit)



Gambar 14 Grafik Master hasil pengujian Sistem (kunci 1024-bit)

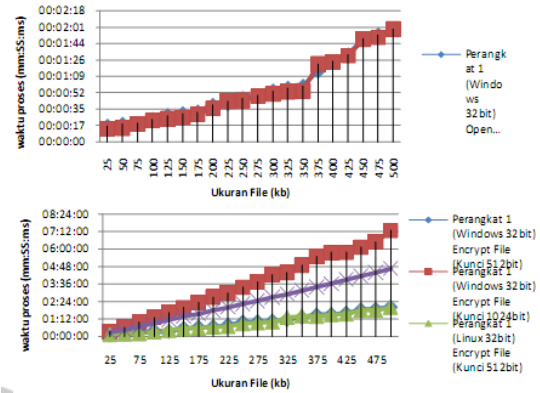
A. Ukuran File dan Waktu Proses

Setelah dilakukan pengujian file dengan menguji sistem dengan file terkecil

berukuran 25 kb dapat dilihat pada saat membuka file 1kb.txt sistem berlansung cepat dengan hanya memakan waktu 00:00:17 dan 00:06:32 untuk proses enkripsi.

Dilanjutkan dengan file-file yang semakin besar dan diakhiri dengan file 500kb.txt yang merupakan file dengan ukuran terbesar berukuran 500 kb. Pada grafik menunjukkan file 500kb.txt memakan waktu 00:01:58 untuk melakukan proses pembukaan file dan 02:01:35 untuk proses enkripsi file.

Dengan demikian dapat disimpulkan bahwa ukuran file merupakan variabel yang sangat mempengaruhi lama waktu dari proses pembukaan file dan enkripsi file pada aplikasi yang telah dibangun.



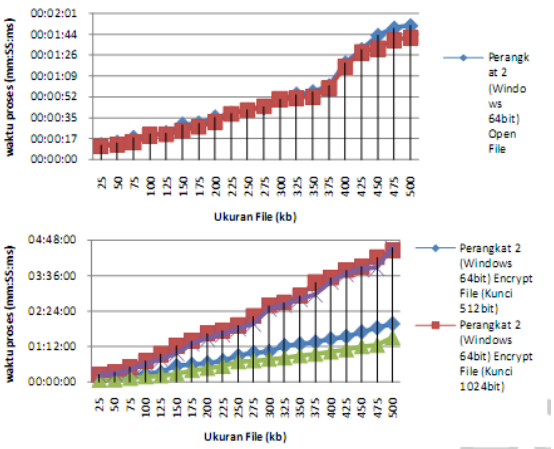
Gambar 15 Grafik perbandingan lama proses Open File dan Encypt File antar sistem operasi berdasarkan ukuran file pada perangkat 1 (Kunci 512 & 1024-bit).

B.2 Variabel Perangkat 2 berdasarkan sistem Operasi

Berdasarkan variabel Perangkat 2 dengan sistem operasi Windows 7 64bit, lama waktu yang dibutuhkan untuk membuka file 25kb.txt adalah 00:00:12, dan untuk proses enkripsi membutuhkan waktu 00:05:22. Jika dibandingkan dengan Perangkat 2 dengan sistem operasi Linux Ubuntu 12.04 64bit waktu yang dibutuhkan untuk membuka file 25kb.txt adalah 00:00:11 serta 00:03:23 untuk melakukan proses enkripsi file.

Dilanjutkan dengan file-file yang berukuran lebih besar dapat dilihat pada grafik bahwa waktu yang dibutuhkan juga semakin besar dengan rentang waktu variabel perangkat 2 berdasarkan sistem operasi tidak cukup berbeda.

Dari pengujian antar sistem tersebut dapat dilihat pada grafik bahwa sistem operasi mempengaruhi lama proses yang dibutuhkan untuk melakukan proses membuka file dan enkripsi file dengan perbandingan bahwa sistem operasi Windows 7 Ultimate 64bit lebih membutuhkan waktu cukup lama dengan rengtang waktu tidak jauh berbeda dibandingkan dengan Linux Ubuntu 12.04 LTS 64bit pada perangkat 2.



Gambar 16 Grafik perbandingan lama proses Open File dan Encrypt File berdasarkan ukuran File pada perangkat 2 (Kunci 512 & 1024-bit).

B. Perangkat dengan Waktu Proses

Dari pengujian antar perangkat berdasarkan waktu proses dapat diketahui bahwa perangkat yang digunakan pada saat menguji sistem juga pempengaruhi lama waktu proses pembukaan file dan proses enkripsi file dikarenakan spesifikasi hardware dan sistem operasi yang digunakan berbeda. Ini mengindikasikan bahwa kecepatan dari masing-masing komponen perangkat memiliki andil dalam proses Open file dan Encrypt File.

Dapat dilihat pada grafik perbandingan antara perangkat 1 dengan perangkat 2 dengan sistem operasi yang sama namun dengan spesifikasi perangkat yang sangat berbeda membutuhkan waktu proses yang berbeda pula untuk melakukan pembukaan file serta encrypt file. Berikut grafik perbandingan antar perangkat dengan sistem operasi yang sama.

Dari hasil analisa diatas, dapat disimpulkan bahwa lama waktu proses pengolahan sebuah informasi menjadi sebuah sandi dengan mengaplikasikan algoritma RSA berbasis pemrograman java sangat dipengaruhi oleh ukuran file yang akan diolah, namun aspek dari spesifikasi hardware dan sistem operasi juga memiliki pengaruh walaupun tidak sebesar ukuran file.

4. Penutup

4.1 Kesimpulan

Berdasarkan pembahasan yang dilakukan dalam penelitian, maka dapat disimpulkan bahwa dengan menggunakan algoritma RSA berbasis java sebagai algoritma untuk menyandikan data file, dapat membantu pengguna mengamankan data file yang bersifat rahasia untuk dapat disimpan pada media penyimpanan *online* ataupun tidak, dikarenakan algoritma RSA adalah algoritma yang sangat sulit untuk ditembus sistem keamanannya. Dengan memanfaatkan panjang bit pada perancangan aplikasi yang dapat didukung dengan sistem komputasi perangkat yang lebih baik, maka semakin dapat memaksimalkan algoritma RSA tersebut dalam menyandikan data yang bersifat rahasia.

4.2 Saran

Berdasarkan pengamatan yang dilakukan, maka diharapkan kepada pembaca untuk lebih memahami konsep dari pemrograman dengan mengaplikasikan bahasa pemrograman java. Dengan memahami pemrograman java, maka diharapkan dapat mempersingkat algoritma dari proses enkripsi, dekripsi, dan generate k proses penyandian informasi dengan menggunakan algoritma RSA dengan hasil komputasi yang lebih ringan sehingga dapat mempermudah proses kinerja *processor* sebagai mesin pengolah. Selain proses pembangunan aplikasi, diharapkan juga aplikasi dapat dijalannkan dengan perangkat yang lebih baik dibandingkan dengan perangkat yang digunakan untuk membangun aplikasi ini, dikarenakan proses komputasi akan memakan waktu yang cukup lama jika menggunakan data dengan ukuran yang cukup besar.

5. Daftar Pustaka

Ariyus, Dony (2008). Pengantar Ilmu Kriptografi, Teori, Analisa, dan Implementasi. Yogyakarta. Penerbit : Andi Offset

Kadir, Abdul (2005). Dasar Pemrograman Java 2. Yogyakarta. Penerbit : Andi Offset.

Murni, Sri (2010). Implementasi Algoritma RSA Berbasis Web Pada Sistem Download Jurnal ITP. Padang. Penerbit : Institut Teknologi Padang

Meidina. (2010). Visualisasi Algoritma RSA Dengan Menggunakan Bahasa Pemrograman Java. Penerbit : Universitas Gunadarma

Suarga (2009). Dasar Pemrograman Komputer / DalamBahasa Java /. Makasar. Penerbit : Andi Offset

Lampiran 1

Tabel 4.1 Tabel Pengujian File Berdasarkan Variabel Yang Telah Ditentukan dengan skunci 512-bit

NO	Nama File (*.txt)	Ukuran File (Kb)	Waktu Komputasi (mm:ss:ms)							
			Perangkat 1				Perangkat 2			
			Windows 32bit		Ubuntu 32bit		Windows 64bit		Ubuntu 64bit	
			Open	Encrypt	Open	Encrypt	Open	Encrypt	Open	Encrypt
1	25kb	25	00:00:17	00:06:32	00:00:14	00:07:32	00:00:12	00:05:22	00:00:11	00:03:23
2	50kb	50	00:00:19	00:10:42	00:00:15	00:10:30	00:00:14	00:08:26	00:00:12	00:04:21
3	75kb	75	00:00:20	00:17:12	00:00:19	00:11:12	00:00:18	00:13:25	00:00:14	00:07:32
4	100kb	100	00:00:24	00:19:23	00:00:23	00:18:23	00:00:21	00:15:32	00:00:20	00:10:45

5	125kb	125	00:00:28	00:20:23	00:00:25	00:25:23	00:00:22	00:18:34	00:00:21	00:12:34
6	150kb	150	00:00:30	00:33:14	00:00:26	00:31:14	00:00:29	00:32:46	00:00:24	00:16:42
7	175kb	175	00:00:32	00:37:52	00:00:29	00:33:52	00:00:30	00:35:24	00:00:27	00:23:24
8	200kb	200	00:00:39	00:40:23	00:00:36	00:38:23	00:00:35	00:38:24	00:00:31	00:27:23
9	225kb	225	00:00:41	00:46:53	00:00:43	00:41:53	00:00:38	00:42:55	00:00:38	00:31:43
10	250kb	250	00:00:46	00:59:00	00:00:44	00:53:00	00:00:41	00:53:00	00:00:41	00:40:53
11	275kb	275	00:00:49	01:00:43	00:00:48	00:55:43	00:00:43	00:59:47	00:00:44	00:42:45
12	300kb	300	00:00:54	01:03:34	00:00:51	00:59:34	00:00:50	01:01:35	00:00:50	00:45:52
13	325kb	325	00:00:57	01:09:53	00:00:53	01:18:53	00:00:54	01:13:51	00:00:51	00:48:31
14	350kb	350	00:00:59	01:19:25	00:00:54	01:25:25	00:00:56	01:17:22	00:00:52	00:53:00
15	375kb	375	00:01:15	01:28:13	00:01:22	01:23:13	00:01:01	01:21:14	00:0:59	00:56:23
16	400kb	400	00:01:25	01:31:32	00:01:24	01:30:32	00:01:20	01:27:32	00:01:17	01:00:41
17	425kb	425	00:01:32	01:36:24	00:01:31	01:32:24	00:01:31	01:32:22	00:01:29	01:05:28
18	450kb	450	00:01:49	01:49:43	00:01:48	01:47:43	00:01:43	01:41:44	00:01:32	01:11:39
19	475kb	475	00:01:53	01:52:14	00:01:50	01:48:14	00:01:49	01:50:13	00:01:39	01:14:28
20	500kb	500	00:01:58	02:01:35	00:01:59	02:00:55	00:01:51	01:58:25	00:01:41	01:30:00

Lampiran 2

Tabel 4.1 Tabel Pengujian File Berdasarkan Variabel Yang Telah Ditentukan dengan kunci 1024-bit

NO	Nama File (*.txt)	Ukuran File (Kb)	Waktu Komputasi (mm:ss:ms)							
			Perangkat 1				Perangkat 2			
			Windows 32bit		Ubuntu 32bit		Windows 64bit		Ubuntu 64bit	
			Open	Encrypt	Open	Encrypt	Open	Encrypt	Open	Encrypt
1	25kb	25	00:00:17	00:19:34	00:00:14	00:16:31	00:00:12	00:14:42	00:00:11	00:13:31
2	50kb	50	00:00:19	00:38:32	00:00:15	00:25:27	00:00:14	00:20:12	00:00:12	00:17:14
3	75kb	75	00:00:20	00:59:23	00:00:19	00:39:47	00:00:18	00:29:45	00:00:14	00:24:51
4	100kb	100	00:00:24	01:18:24	00:00:23	00:53:44	00:00:21	00:41:54	00:00:20	00:36:14
5	125kb	125	00:00:28	01:38:18	00:00:25	01:06:49	00:00:22	00:56:12	00:00:21	00:45:17

6	150kb	150	00:00:30	01:59:15	00:00:26	01:22:24	00:00:29	01:13:31	00:00:24	00:58:18
7	175kb	175	00:00:32	02:19:19	00:00:29	01:33:12	00:00:30	01:23:12	00:00:27	01:17:19
8	200kb	200	00:00:39	02:42:28	00:00:36	01:46:33	00:00:35	01:38:32	00:00:31	01:27:43
9	225kb	225	00:00:41	03:00:22	00:00:43	02:00:15	00:00:38	01:43:41	00:00:38	01:36:58
10	250kb	250	00:00:46	03:22:58	00:00:44	02:13:30	00:00:41	01:54:13	00:00:41	01:48:16
11	275kb	275	00:00:49	03:43:26	00:00:48	02:26:34	00:00:43	02:13:45	00:00:44	01:57:58
12	300kb	300	00:00:54	04:15:20	00:00:51	02:41:34	00:00:50	02:34:43	00:00:50	02:27:53
13	325kb	325	00:00:57	04:24:27	00:00:53	02:54:13	00:00:54	02:41:21	00:00:51	02:36:16
14	350kb	350	00:00:59	04:54:16	00:00:54	03:08:58	00:00:56	02:54:46	00:00:52	02:47:59
15	375kb	375	00:01:15	05:29:10	00:01:22	03:23:28	00:01:01	03:21:31	00:0:59	02:56:47
16	400kb	400	00:01:25	05:46:15	00:01:24	03:37:12	00:01:20	03:31:45	00:01:17	03:23:35
17	425kb	425	00:01:32	05:48:09	00:01:31	03:53:54	00:01:31	03:46:41	00:01:29	03:39:26
18	450kb	450	00:01:49	06:09:15	00:01:48	04:07:15	00:01:43	03:53:31	00:01:32	03:48:41
19	475kb	475	00:01:53	06:29:23	00:01:50	04:22:32	00:01:49	04:12:56	00:01:39	03:51:25
20	500kb	500	00:01:58	07:17:03	00:01:59	04:38:42	00:01:51	04:26:19	00:01:41	04:31:46