SISTEM PENGAMANAN DATA PADA MEDIA PENYIMPANAN BERBASIS *CLOUD* MENGGUNAKAN ALGORITMA AES (*ADVANCED ENCRYPTION STANDARD*) DAN RSA (*RIVEST-SHAMIR-ADLEMAN*)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Danang Januarko
NIM: 115060807111006

PROGRAM STUDI INFORMATIKA/ILMU KOMPUTER
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

PENGESAHAN

SISTEM PENGAMANAN DATA PADA MEDIA PENYIMPANAN BERBASIS *CLOUD* MENGGUNAKAN ALGORITMA AES (*ADVANCED ENCRYPTION STANDARD*) DAN RSA (*RIVEST-SHAMIR-ADLEMAN*)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Danang Januarko
NIM: 115060807111006

Skripsi ini telah diuji dan dinyatakan lulus pada
18 Januari 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Sabriansyah Rizqika Akbar, S.T, M.Eng	Aswin Suharsono, S.T, M.T
NIP: 19820809 201212 1 004	NIK: 840919 06 1 1 0251

Mengetahui
Ketua Program Studi Informatika/Ilmu Komputer

Drs. Marji, M.T
NIP: 19670801 199203 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 1 Januari 2016

Danang Januarko

NIM: 115060807111006

KATA PENGANTAR

Puji syukur kami ucapkan kehadirat Tuhan Yang Maha Esa atas rahmat dan hidayahNya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul "Sistem Pengamanan Data Pada Media Penyimpanan Berbasis *Cloud* Menggunakan Algoritma AES (*Advanced Encryption Standard*) dan RSA (*Rivest-Shamir-Adleman*)".

Penulis menyadari bahwa penyusunan makalah ini tidak akan berjalan lancar tanpa berbagai masukan yang bermanfaat dari dosen pembimbing serta teman-teman penulis. Oleh karena itu penulis mengucapkan terima kasih kepada pihak-pihak tersebut yang telah bersedia untuk memberikan arahan dan saran demi kelancaran penyusunan makalah ini diantaranya:

1. Sabriansyah Rizqika Akbar, S.T, M.Eng, selaku dosen pembimbing I yang telah banyak memberikan ilmu, saran, dan motivasi untuk menyelesaikan laporan ini.
2. Aswin Suharsono, S.T, M.T, selaku dosen pembimbing II yang telah memberikan ilmu, saran dan semangat untuk menyelesaikan laporan ini.
3. Kedua orang tua penulis, Suwadi dan Salama yang telah memberi motivasi, kasih sayang, serta dukungan baik moril maupun materil.
4. Kakak penulis, Dewi Ramandhari yang telah memberi motivasi, masukan, kasih sayang, serta dukungan moril maupun materil.
5. Sarwo Hadi Wibowo, Ariotomo Satyo Wicaksono, dan seluruh teman-teman TIF-A 2011 yang telah menemani dan memberi saran dan masukan kepada penulis hingga terselesaikanya skripsi ini.
6. Teman-teman seperjuangan skripsi laboratorium jaringan komputer, terima kasih telah memberikan semangat dan dukungan dalam pengerjaan skripsi ini.
7. Semua teman-teman PTIIK, khususnya Informatika/Ilmu Komputer 2011 terima kasih atas segala bantuan dan dukungannya selama ini.
8. Segenap dosen dan karyawan PTIIK Universitas Brawijaya yang telah membantu pelaksanaan skripsi ini.
9. Semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat baik secara langsung maupun tidak langsung demi terselesaikannya tugas akhir ini.

Semoga jasa dan amal baik mendapatkan balasan dari Allah SWT Kritik dan saran yang membangun sangat kami harapkan dari para pembaca mengingat makalah yang kami susun ini masih jauh dari kata sempurna.

Akhir kata, kami ucapkan terima kasih dan semoga makalah ini dapat memberikan manfaat bagi para pembaca terutama mahasiswa PTIIK Universitas Brawijaya.

Malang, 1 Januari 2016

Danang Januarko
danang.januarko@gmail.com

ABSTRAK

Cloud merupakan suatu layanan yang digunakan sebagai penyimpanan data melalui sebuah jaringan internet, sehingga pengguna dapat melakukan akses yang lebih cepat terhadap data yang dimiliki dimanapun mereka berada. Keamanan merupakan salah satu isu utama yang ada dalam teknologi *cloud* antara lain pencurian data dan *loss of privacy*. Metode yang digunakan untuk mengatasi masalah kemanan tersebut adalah dengan menggunakan enkripsi sebelum data dikirim ke sebuah layanan *cloud* dengan menggunakan gabungan antara algoritma AES (*Advanced encryption Standard*) dan algoritma RSA (*Rivest-Shamir-Adleman*). Algortima AES digunakan untuk melakukan enkripsi terhadap file dan algoritma RSA digunakan sebagai media untuk melakukan pertukaran kunci AES. Selain itu penggunaan algoritma RSA juga dapat digunakan sebagai algoritma *digital signature* yang digunakan sebagai proses integritas dan untuk menangani isu *non-repudiation*. Pada tugas akhir ini akan dibangun sebuah sistem yang mampu melakukan enkripsi sebelum data diunggah ke penyedia layanan *cloud* sehingga data yang dikirim aman saat berada pada layanan *cloud*. Selain itu, sistem juga dapat melakukan *sharing* terhadap *file* yang sudah terenkripsi pada layanan *cloud* sehingga pengguna dapat melakukan pertukaran file dengan aman. Oleh karena itu, penulis melakukan penelitian dengan judul "Sistem Pengamanan Data Pada Media Penyimpanan Berbasis *Cloud* Menggunakan Algoritma AES (*Advanced Encryption Standard*) dan RSA (*Rivest-Shamir-Adleman*)". Dari hasil pengujian dan analisis yang dilakukan pada penelitian ini menunjukkan bahwa sistem mampu berjalan dengan baik yang ditinjau dari segi fungsionalitas dan aspek kemanan. Dari pengujian performansi *upload file* dengan membandingkan sistem yang menggunakan enkripsi dan *non*-enkripsi didapatkan hasil dengan perbedaan waktu hanya 3 detik pada masing-masing *file*.

Kata kunci: keamanan *Cloud*, Enkripsi, pertukaran kunci, algoritma kriptografi, AES, RSA

ABSTRACT

Cloud is a service that used as a data storage through internet network, that allows user a faster access to the their data from any location. Security is a main issue in cloud technology which is theft of information and loss of privacy. Method used to overcome said security issue is with the use of encryption before data sent to a cloud service with the combine of AES algorithm (Advanced encryption Standard) and RSA algorithm (*Rivest-Shamir-Adleman*). *AES algorithm used as file encryption and RSA algorithm used as a media to exchange the AES key. Furthermore, RSA algorithm can also be used as digital signature algorithm as an integrity process and solve non-repudiation issue. In this final project will be built a system that can encrypt a data before it was uploaded into cloud service provider so that data will be secure in the cloud. Moreover, system can also share the encrypted file on the cloud so user will be able to do file exchange securely. With that in mind, writer do a research "Data Securing System on Cloud Based Storage Using AES (Advanced Encryption Standard) and RSA (Rivest-Shamir-Adleman) Algorithm". The testing and analysis result shows that the system could run well based on its the functionality and security. From file upload performance testing with comparing system that utilize encryption and non-encryption achieve results in time difference of merely 3 seconds on each files.*

Keyword: *Cloud security*, *Encryption*, *Key exchange*, *Cryptographic Alogrithm*, AES, RSA

DAFTAR ISI

PENGESAHAN ii
PERNYATAAN ORISINALITAS iii
KATA PENGANTAR iv
ABSTRAK vi
ABSTRACT vii
DAFTAR ISI viii
DAFTAR TABEL x
DAFTAR GAMBAR xi
DAFTAR *SOURCE CODE* xii
BAB 1 PENDAHULUAN 1
1.1 Latar belakang 1
1.2 Rumusan masalah 2
1.3 Tujuan 2
1.4 Manfaat 3
1.5 Batasan masalah 3
1.6 Sistematika pembahasan 3
BAB 2 LANDASAN KEPUSTAKAAN 5
2.1 Kajian Pustaka 5
2.2 *Cloud Computing* 5
2.2.1 *Cloud Storage* 7
2.3 Java 8
2.3.1 Maven *project* 8
2.4 Kriptografi 8
2.4.1 Algoritma Simetrik 9
2.4.2 Algoritma Asimetrik 12
2.5 *Digital signature* 14
BAB 3 METODOLOGI 15
3.1 Perumusan Masalah 15
3.2 Studi Literatur 16
3.3 Perancangan Sistem 16

3.3.1 Lingkungan sistem 16
3.3.2 Perancangan Aplikasi 17
3.4 Implementasi 23
3.5 Pengujian sistem 23
3.6 Analisis dan hasil pengujian 23
3.7 Kesimpulan 24
BAB 4 Implementasi dan pengujian 25
4.1 Implementasi 25
4.1.1 Konfigurasi API 25
4.1.2 Implementasi fungsi Aplikasi 26
4.2 Pengujian 38
4.2.1 Pengujian Validasi Fitur Sistem 39
4.2.2 Pengujian Performansi Sistem 39
4.2.3 Pengujian keamanan data 39
BAB 5 Hasil pengujian dan analisis 40
5.1 Hasil Pengujian 40
5.1.1 Hasil pengujian fungsional sistem 40
5.1.2 Hasil pengujian performansi sistem 41
5.1.3 Hasil pengujian keamanan jaringan 43
5.2 Analisis 46
BAB 6 Penutup 49
6.1 Kesimpulan 49
6.2 Saran 49
DAFTAR PUSTAKA 51

DAFTAR TABEL

Tabel 5.1 Tabel Hasil Pengujian Fungsional Sistem 40
Tabel 5.2 Percobaan *Upload file* dengan ukuran 5KB..... 41
Tabel 5.2 Percobaan *Upload file* dengan ukuran 5KB (Lanjutan) 42
Tabel 5.3 Percobaan *Upload file* dengan ukuran 1MB 42
Tabel 5.4 Percobaan *Upload file* dengan ukuran 13MB 43

DAFTAR GAMBAR

Gambar 2.1 Pemodelan Saas 6
Gambar 2.2 Pemodelan PaaS 6
Gambar 2.3 Pemodelan IaaS 7
Gambar 2.4 Cara kerja Algortima Simetrik 10
Gambar 2.5 CBC pada waktu enkripsi 11
Gambar 2.6 CBC pada waktu dekripsi 11
Gambar 2.7 Diagram alir enkripsi AES 12
Gambar 3.1 Diagram Alir Metodologi 15
Gambar 3.2 Arsitektur Sistem *Upload* 18
Gambar 4.1 *Form login* 27
Gambar 4.2 *Form* verifikasi Dropbox 27
Gambar 4.3 antarmuka *listfile* 28
Gambar 4.4 *form* input email *share* 31
Gambar 5.1 hasil *file* dibuka tanpa ijin 44
Gambar 5.2 verifikasi kunci gagal 44
Gambar 5.3 verifikasi *file* gagal 44
Gambar 5.4 kunci tidak valid 45
Gambar 5.5 verifikasi kunci gagal 45
Gambar 5.6 verifikasi *file* gagal 46
Gambar 5.7 Perbandingan waktu *upload* enkripsi *file* 46
Gambar 5.8 Perbandingan waktu *upload* tanpa enkripsi *file* 47
Gambar 5.9 perbandingan waktu *upload* enkripsi dan *non*-enkripsi 47

DAFTAR *SOURCE CODE*

Source code 4.1 Kode untuk menambahkan API dropbox 25
Source code 4.2 kode verifikasi pengguna 26
Source code 4.3 Fungsi *Showlist*() 27
Source code 4.4 Fungsi uploadFiles() 28
Source code 4.5 Fungsi downloadFiles() 29
Source code 4.6 Fungsi *shareFiles*() 30
Source code 4.7 Fungsi *generateKey* AES 31
Source code 4.8 Fungsi *generateKey* RSA 32
Source code 4.9 Fungsi *generateKey* RSA_sign 32
Source code 4.10 Fungsi enkripsi *file* 33
Source code 4.11 Fungsi enkripsi kunci AES 34
Source code 4.12 Fungsi dekripsi *file* 34
Source code 4.13 Fungsi dekripsi kunci AES 35
Source code 4.14 Fungsi *signature* 36
Source code 4.15 Fungsi *verify signature file* 37
Source code 4.16 Fungsi *verify signature key* 38

BAB 1 PENDAHULUAN

1.1 Latar belakang

Cloud merupakan suatu layanan yang digunakan sebagai penyimpanan data melalui sebuah jaringan internet, sehingga pengguna dapat melakukan akses yang lebih cepat terhadap data yang dimiliki dimanapun mereka berada. Selain itu dengan memanfaatkan teknologi *cloud* pengguna juga dapat menggunakannya sebagai sebuah *storage* yang dimana tergolong lebih murah dibandingkan dengan biaya membeli *harddisk* tambahan yang tergolong lebih mahal. *Cloud storage* juga dapat kita gunakan sebagai sebuah media penyimpanan cadangan jika media penyimpanan yang kita miliki mengalami kerusakan.

Keamanan merupakan salah satu isu utama yang ada dalam teknologi *cloud computing*. Isu berikutnya adalah apabila kita mengirimkan sebuah data dan pada waktu pengiriman ada orang yang dengan sengaja melihat isi dari data kita yang sering disebut dengan *Man in the Middle Attack*, isu lainnya adalah ada kemungkinan data yang sudah berada di dalam *cloud storage* bisa dibaca oleh penyedia layanan *cloud* itu sendiri, dan hilangnya privasi pengguna ketika data berada di dalam layanan *cloud storage* atau yang lebih dikenal dengan istilah *loss of privacy* (Ramadhan, 2011). Untuk mengatasi masalah tersebut, terdapat suatu proses yang digunakan untuk melakukan keamanan terhadap data yang akan dikirim pada *cloud* dengan melakukan sebuah enkripsi sebelum data tersebut dikirim. Enkripsi dilakukan dengan menggunakan kunci simetrik. Setelah data di enkripsi dengan kunci simetrik timbul permasalahan baru, data yang sudah di enkripsi tidak bisa kita bagi ke orang lain kecuali orang tersebut memiliki kunci untuk melakukan dekripsi data. Untuk mengatasi permasalahan dalam hal berbagi data yang sudah di enkripsi, perlu dilakukan suatu proses pertukaran kunci menggunakan algoritma asimetrik dengan aman untuk menghindari terjadinya pencurian terhadap kunci yang di *sharing*.

Enkripsi merupakan suatu proses yang digunakan untuk membantu dalam memastikan suatu keamanan tertentu seperti kerahasiaan, intergritas, otentikasi, dan identifikasi data. Untuk mencapai keamanan yang telah disebutkan diatas, kombinasi algoritma kriptografi harus sesuai (M.Barukab, et al., 2012). Pada penelitian ini kriptografi yang digunakan merupakan gabungan antara algoritma simetrik AES (*Advanced Encryption Standard*) dan algoritma asimetrik RSA (*Rivest-Shamir-Adleman*).

AES adalah algortima yang menggunakan satu kunci dalam pengaplikasiannya. AES memiliki keunggulan melakukan proses enkripsi yang lebih cepat dibandingan dengan algoritma asimetrik. Disisi lain, algoritma simetrik memliki kelemahan pada pendistribusian kuncinya. *Man In The Middle Attack* menjadi salah satu masalah ketika melakukan suatu pendistribusian data atau kunci dikarenakan kunci yang digunakan antara pengirim dan penerima sama (KetuWare, 2004).

RSA adalah algoritma yang menggunakan 2 kunci dalam melakukan enkripsi dan dekripsi atau sering disebut dengan algoritma kunci publik. RSA digunakan

dalam proses pertukaran data dan juga dalam melakukan proses *signature* terhadap data. Kelemahan dari algoritma ini adalah kunci RSA sangat terbatas. Jika, AES melakukan enkripsi pesan dengan 256 bits maka RSA membutuhkan 15.360 bits untuk mendapatkan tingkat keamanan yang sama. Hal ini tentu saja berdampak terhadap tingkat kinerja CPU *speed* dan *memory* (KetuWare, 2004).

Pada penelitian ini AES digunakan untuk melakukan enkripsi terhadap *file* dan RSA digunakan untuk melakukan enkripsi terhadap kunci simetrik atau *secret-key* sehingga *secret-key* hanya akan bisa digunakan oleh pemilik. Selain itu, RSA juga akan digunakan sebagai *digital signature* yang digunakan untuk pengecekan integritas data dan untuk menangani isu *non-repudiation*.

Pada penelitian sebelumnya oleh Manpreet Kaur dkk (2013) dengan judul "*Implementing Encryption Algorithms to Enhance Data Security of Cloud in Cloud Computing*" penggunaan algoritma kriptografi untuk meningkatkan pengamanan data pada *cloud*. Pada penelitian ini dibahas mengenai mengamankan *file* melalui jalur yang terenkripsi menggunakan HTTPS dan data akan di dekripsi lagi di sisi *server*. Setelah itu pengguna akan melakukan enkripsi lagi di sisi *server* menggunakan algoritma simetrik atau algoritma asimetrik yang dipilih oleh pengguna tersebut. Penelitian ini menghasilkan sistem yang dapat meningkatkan keamanan data pada *cloud* dengan menggunakan algoritma simetrik atau algoritma asimetrik tergantung dengan besar *file* dan pengguna dapat menentukan algoritma yang akan dipakai (Kaur & Singh, 2013).

Penelitian ini dilakukan untuk mengatasi masalah keamanan pada *cloud storage*. Sistem yang akan dibuat dapat melakukan proses enkripsi secara otomatis sebelum data diunggah pada layanan *cloud storage* sehingga data aman ketika proses pengiriman. Sistem ini juga dapat melakukan proses *berbagi* data dengan aman melalui proses pertukaran kunci menggunakan RSA sehingga data dapat dibaca oleh pengguna lainnya.

1.2 Rumusan masalah

Berdasarkan latar belakang yang telah dijelaskan, maka rumusan masalah yang menjadi acuan dari penelitian ini adalah:

1. Bagaimana merancang sebuah sistem yang dapat mencegah *loss of privacy* dan *theft of information* pada media penyimpanan berbasis *cloud*?
2. Bagaimana merancang sebuah sistem yang dapat melakukan pertukaran kunci simetrik pada media penyimpanan berbasis *cloud*?
3. Bagaimana performansi sistem pengamanan data terhadap proses *upload* pada layanan media penyimanan berbasis *cloud*?
4. Bagaimana hasil pengujian terhadap keamanan yang dilakukan pada sistem?

1.3 Tujuan

Membangun sebuah aplikasi yang dapat melakukan enkripsi secara otomatis sebelum data diunggah agar dapat mengatasi masalah privasi data pada sebuah

layanan *cloud storage* dan aplikasi ini juga diharapkan dapat melakukan *sharing* data terhadap sesama pengguna dengan aman.

1.4 Manfaat

Menyediakan sistem yang dapat meningkatkan keamanan data mulai dari proses pengiriman sampai data berada pada layanan media penyimpanan *cloud* dengan melakukan enkripsi data sebelum data dikirim.

1.5 Batasan masalah

Batasan - batasan masalah pada penelitian ini adalah:

1. Menggunakan Dropbox sebagai objek penelitian
2. Menggunakan bahasa pemrograman Java sebagai sarana untuk membangun aplikasi
3. Pada penelitian ini, berbagi hanya bisa dilakukan oleh 2 pengguna
4. Pada proses pengenkripsian peneliti menggunakan AES 128bit dan RSA 2048bit pada proses enkripsi kunci AES dan RSA 2048bit pada proses *signature.*

1.6 Sistematika pembahasan

Untuk mencapai tujuan yang diharapkan, maka sistematika penulisan yang disusun adalah sebagai berikut:

BAB I PENDAHULUAN

Pada bab ini telah dijabarkan latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah, sistematika penulisan serta rencana kegiatan.

BAB II LANDASAN KEPUSTAKAAN

Pada bab ini dilakukan pengkajian pustaka terhadap teori-teori yang berkaitan dan menunjang dalam penyelesaian masalah ini. Aplikasi enkripsi pada *cloud storage* dengan menggunakan algoritma simetrik dan algoritma asimetrik menjadi fokus pada penelitian ini. kriptografi digunakan sebagai metode untuk mengamankan data pada *cloud storage*, dan Java digunakan sebagai bahasa pemrograman untuk membuat aplikasi. Untuk mewujudkan aplikasi ini diperlukan metode yang dapat diimplementasikan pada penelitian ini. Teori-teori yang diambil berasal dari jurnal, buku, dan sumber referensi lainnya yang berhubungan dengan topik yang diteliti.

BAB III METODOLOGI PENELITIAN DAN PERANCANGAN

Pada bab ini membahas tentang metode yang digunakan dalam penulisan yang terdiri dari perumusan masalah, studi literatur, perancangan, implementasi, pengujian, dan analisis serta pengambilan kesimpulan dan saran. Selain itu, pada bab III juga dijelaskan mengenai rancangan dari sistem pengamanan data dengan menggunakan algoritma simetrik dan algoritma asimetrik pada layanan *cloud*

storage. Sebuah skenario pengujian mencakup tujuan penelitian juga disampaikan pada bab ini.

BAB IV IMPLEMENTASI DAN PENGUJIAN

Pada bab ini membahas tentang penerapan dari arsitektur sistem secara terperinci dengan langkah-langkah pengerjaan serta menampilkan gambar-gambar dari implementasi yang dilakukan berupa potongan-potongan kode dari hasil implementasi. Implementasi dilakukan berdasarkan sistematika yang dibuat pada bab sebelumnya. Berikutnya pengujian terhadap sistem yang dilakukan menggunakan skenario yang telah dibuat pada bab sebelumnya.

BAB V HASIL PENGUJIAN DAN ANALISIS

Pada bab ini membahas tentang hasil pengujian dan analisis terhadap sistem yang telah dibangun. Hasil pengujian didapatkan dari pengujian yang telah dilakukan sesuai dengan skenario pengujian yang sudah ditentukan. Dari hasil tersebut dapat dilakukan analisis terhadap kinerja sistem, serta mengetahui bagaimana sistem mencapai tujuan.

BAB VI PENUTUP

Memuat kesimpulan dari penelitian yang telah dilakukan serta saran untuk pengembangan lebih lanjut.

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab ini berisi kajian pustaka dan dasar teori yang digunakan untuk menunjang penyusunan skripsi. Kajian pustaka berisi tentang pembahsan peneitian sebelumnya yang berkaitan dengan skripsi ini. Sedangkan dasar teori berisi mengenai pembahasan teori yang digunakan untuk menyusun penelitian. Dasar teori yang dibahas antara lain *Cloud*, kriptografi, aspek-aspek keamanan, algoritma simetrik, algoritma asimetrik, *Digital signature*, AES *(Advanced Encryption Standard)* dan RSA (*Rivest-Shamir-Adleman*).

2.1 Kajian Pustaka

Kajian pustaka pada penelitian ini membahas tentang penelitian sebelumnya dengan mengimplementasikan penggunaan algoritma kriptografi untuk mengamankan data pada *cloud*. Pada penelitian sebelumnya oleh Manpreet Kaur dkk (2013) dengan judul "*Implementing Encryption Algorithms to Enhance Data Security of Cloud in Cloud Computing*" mengenai penggunaan algoritma kriptografi untuk meningkatkan pengamanan data pada *cloud*. Pada penelitian ini dibahas mengenai mengamankan *file* melalui jalur yang terenkripsi menggunakan HTTPS dan data akan di dekripsi lagi di sisi *server*. Setelah itu pengguna akan melakukan enkripsi lagi di sisi *server* menggunakan algoritma simetrik atau algoritma asimetrik yang dipilih oleh pengguna tersebut (Kaur & Singh, 2013).

Penelitian ini menghasilkan sistem yang dapat meningkatkan keamanan data pada *cloud* dengan menggunakan algoritma simetrik atau algoritma asimetrik tergantung dengan besar *file* dan pengguna dapat menentukan algoritma yang akan dipakai.

2.2 *Cloud Computing*

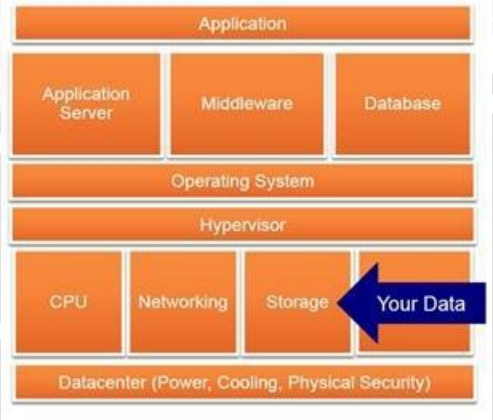
Cloud computing merupakan model komputasi yang memungkinkan pengguna untuk menggunakan *resource* yang ada dalam sebuah jaringan *cloud* (internet) sehingga dapat digunakan untuk melakukan aktifitas berbagi dan dapat digunakan bersama-sama (Pratiwi, 2011). *Cloud computing* dapat menghemat pengeluaran baik itu organisasi ataupun *personal*. Karena pengguna tidak perlu mengalokasikan biaya untuk membeli *hardware* maupun *software* baru. Pengguna cukup menyewa sebuah layanan *cloud computing* pada penyedia layanan yang tersedia. Sehingga dengan melakukan penerapan terhadap teknologi *cloud computing* ini pengguna mampu menghemat pengeluaran yang seharusnya tidak perlu terjadi.

Model layanan *cloud computing* dibagi menjadi 3 antara lain (Adrianus, 2010):

- **Software as a Service (Saas)**

  SaaS merupakan Teknologi *Cloud Computing* yang di dalam melakukan aksesnya pengguna menggunakan *Web Browser*. *End User* menjadi fokus pada layanan ini yang bertugas sebagai pengatur jasa layanan SaaS. Pada tahap awal

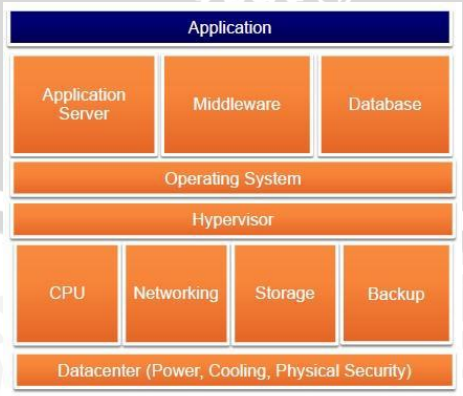
pengguna tidak mengeluarkan biaya untuk pembelian server dan lisensi perangkat lunak. Sedangkan penyedia layanan mengeluarkan biaya relatif lebih rendah dibandingkan dengan biaya untuk melakukan *Hosting* secara konvensional. Saas menjadi sebuah layanan untuk model pengiriman yang digunakan untuk banyak aplikasi bisnis, termasuk aplikasi *office* dan pengiriman pesan, aplikasi DBMS, virtualisasi dan lain-lain.



Gambar 2.1 Pemodelan Saas
Sumber : (Adrianus, 2010)

- **Platform as a Service (PaaS)**

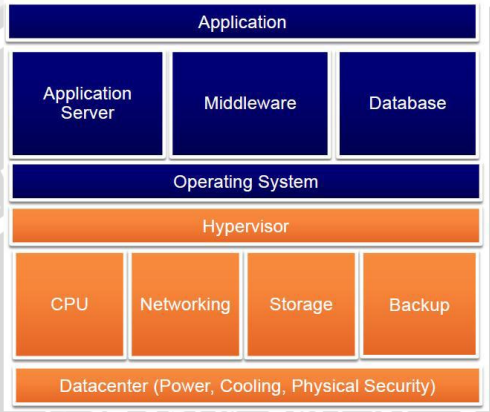
Platform-as-a-Service (PaaS) adalah sebuah servis dalam *cloud computing* yang memungkinkan pengguna untuk membuat, menjalankan dan mengelola tanpa memerlukan sebuah bangunan yang komplek dan pemeliharaan infrasturktur yang biasanya terkait dalam pembangunan sebuah aplikasi. Aplikasi penyedia Layanan ini menawarkan antarmuka pemrograman (API) yang memungkinkan para pengembang untuk memanfaatkan fungsi melalui Internet, daripada menyediakan aplikasi yang lengkap secara lokal. Variasi Teknologi *Cloud Computing* ini menyediakan lingkungan pengembangan untuk *programmer*, Analis, dan *Software* sebagai instruktur pelayanan. Contoh Aplikasi yang menggunakan Teknologi *Cloud Computing* sebagai PaaS adalah *Google App Engine*.



Gambar 2.2 Pemodelan PaaS
Sumber : (Adrianus, 2010)

- **Infrastructure as a Service (IaaS)**

Infrastruktur as a Service (IaaS) adalah penyedia infrastruktur yang menyediakan pelayanan di lingkungan virtualisasi. Pada contoh pemodelan IaaS dapat dikatakan bahwa pengguna menyewa sebuah komputer virtual. Sehingga, pengguna dapat melakukan konfigurasi sistem, menginstall OS sesuai yang dibutuhkan. Pengguna juga memiliki hak akses untuk mengontrol OS, media penyimpanan, memori yang digunakan dan komponen jaringan.



Gambar 2.3 Pemodelan IaaS
Sumber : (Adrianus, 2010)

2.2.1 *Cloud Storage*

Cloud Storage adalah sebuah model penyimpanan data dimana data disimpan di internet yang penggunanya dapat mengelola data dimana saja selama penggunanya terhubung ke *cloud storage*. Konsep *cloud storage* sama seperti konsep *file server* pada suatu kantor perusahaan, hanya saja infrastruktur media *storage* tersebut dikelola oleh *provider cloud* dan pemanfaatannya dijadikan layanan penyimpanan *file* yang dapat diakses dari internet. Sebenarnya *cloud storage* bukan layanan teknologi baru dalam dunia internet, sebelum tren *cloud computing* sepopuler saat ini, layanan *cloud storage* lebih dikenal dengan istilah virtual drive, namun memasuki era *cloud computing* ini istilah tersebut berubah menjadi lebih dikenal dengan sebutan *cloud storage* (Assagaf, 2012).

2.2.1.1 Dropbox

Dropbox adalah sebuah perusahaan IT yang memiliki produk *powerful* baik untuk perorangan ataupun bisnis (Dropbox, 2015). Dropbox merupakan salah satu penyedia layanan *cloud storage* yang cukup populer dan termasuk yang pertama dalam bidang ini. Dropbox bisa dikatakan sebuah layanan *cloud storage* pribadi yang sering digunakan untuk *file berbagi*dan kolaborasi. Dropbox tidak hanya mengakomodasi perorangan, tetapi juga terdapat layanan yang dapat digunakan untuk satu tim. Data pengguna disimpan pada Amazon *Simple Storage Service* (S3) dan dilindungi dengan *Secure Socket Layer* (SSL) dan *Advanced Encryption Standard* (AES) dengan enkripsi 256-bit. Untuk berbagi *file*, pengguna dapat

mengirimkan link dari *website* Dropbox sehingga orang lain dapat melihatnya, atau mengirimkan sebuah undangan untuk bergabung pada sebuah *shared folder* (Rouse, 2011).

2.3 Java

Java merupakan bahasa pemrograman yang berorientasi objek (OOP) dan dapat dijalankan pada berbagai *platform* sistem operasi. Perkembangan Java tidak hanya terfokus pada satu sistem operasi, tetapi dikembangkan untuk berbagai sistem operasi dan bersifat *open source* (Avestro, 2007).

Sebagai sebuah bahasa pemrograman yang berkembang pesat, Java tidak murni dibuat dari awal, melainkan banyak mengakomodasi dari bahasa pemrograman yang telah ada sebelumnya. SUN sebagai perusahan pengembang resmi juga menyediakan berbagai macam tools bagi penggunanya, yaitu: *compiler, interpreter,* penyusun dokumentasi, paket kelas dan sebagainya.

Kelebihan dari Java sendiri adalah hampir dapat digunakan untuk mengembangkan seluruh bentuk aplikasi, mulai dari desktop, web dan bahkan *socket programming*, sebagaimana dibuat dengan menggunakan bahasa pemrograman konvensional lainnya. Tidak hanya itu, karena sifatnya yang *open source,* banyak tersedia *plugin, library* dan *framework* yang bertujuan untuk memudahkan pengguna dalam membangun sebuah aplikasi berbasis Java.

2.3.1 Maven *project*

Maven adalah sebuah sebuah perangkat lunak untuk memanejemen proyek dan pemahaman alat. Berdasarkan *Project Object Model* (POM). Maven mendefinisikan bagaimana *file* .java dapat dikompilasi menjadi *file* .class, dikemas menjadi *file* .jar, mengelola *classpath*, dam segala macam tugas yang dibutuhkan untuk membangun proyek berbasis Java. Perangkat lunak ini menyederhanakan proses dalam membangun seperti halnya *ANT* dan *GRADLE* tetapi Maven lebih unggul karena Maven benar-benar tidak membutuhkan *tools* tambahan atau *script* yang memberikan perintah untuk mengunduh & menginstal *library* yang diperlukan (Anon., 2015). Tujuan utama Maven adalah untuk memungkinkan pengembang aplikasi memahami keadaan lengkap dari upaya pembangunan dalam periode waktu terpendek.

2.4 Kriptografi

Kriptografi adalah sebuah ilmu mengenai teknik enkripsi dimana data/informasi diacak menggunakan kunci enkripsi menjadi sesuatu yang sulit dibaca oleh seseorang yang tidak memiliki kunci dekripsi (Kromodimoeljo, 2009). Kriptografi digunakan untuk menyembunyikan informasi yang penting seperti: komunikasi e-mail, transaksi bank, keamanan rekening bank, PIN, password dan transaksi kartu kredit di web. Kriptografi juga digunakan untuk berbagai masalah keamanan informasi lainnya termasuk tanda tangan elektronik, yang digunakan untuk membuktikan siapa yang mengirim pesan (Schaefer, n.d.).

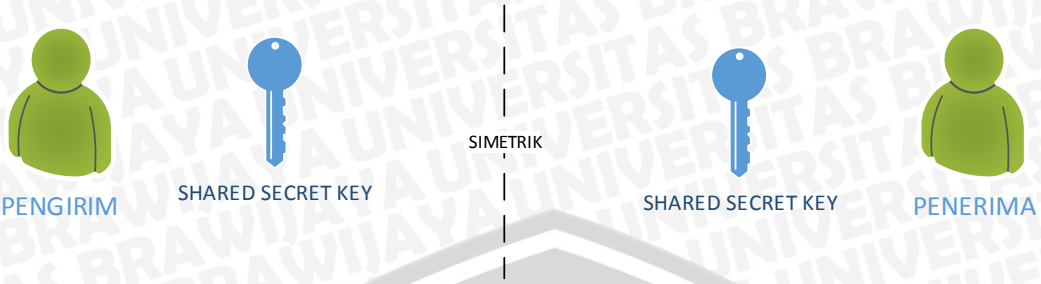
Kriptografi berasal dari bahasa Yunani yaitu dari kata *Crypto* dan *Graphia* yang berarti penulisan rahasia. Kriptografi adalah suatu ilmu yang mempelajari penulisan secara rahasia. Kriptografi merupakan bagian dari suatu cabang ilmu matematika yang disebut *Cryptology*. Kriptografi bertujuan menjaga kerahasiaan informasi yang terkandung dalam data sehingga informasi tersebut tidak dapat diketahui oleh pihak yang tidak sah.

Tujuan mendasar dari ilmu kriptografi antara lain:

- *Confidentiality*, adalah proses yang digunakan untuk menjamin kerahasiaan data atau informasi dari siapapun kecuali yang memiliki wewenang atau kunci rahasia untuk membuka/mendekrip informasi yang telah di enkripsi.
- *Integrity*, adalah proses yang digunakan untuk menjamin keaslian data dan menjamin bahwa data tidak dirubah. Untuk menjaga integritas data, sistem harus memiliki kemampuan untuk mendeteksi manipulasi data oleh pihak-pihak yang tidak berhak.
- *Authentication*, adalah proses yang berhubungan dengan identifikasi/pengenalan terhadap data atau informasi. Informasi yang dikirimkan harus diotentikasi keaslian, isi datanya, waktu pengiriman, dan lain sebagainya. Pihak yang berkomunikasi harus dapat memastikan bahwa pihak lain yang diajak berkomunikasi adalah benar-benar pihak yang bersangkutan.
- *Non-repudiation*, adalah usaha untuk mencegah terjadinya penyangkalan terhadap pengiriman suatu informasi oleh si pengirim. Pembuktian korespondensi antara pihak yang mengirimkan suatu informasi dengan yang dikirimkan juga perlu dilakukan sehingga identitas pengirim suatu informasi dapat dipastikan dan penyangkalan pihak tersebut atas informasi yang telah dikirimnya tidak dapat dilakukan.

2.4.1 Algoritma Simetrik

Algoritma simetrik adalah suatu metode yang hanya menggunakan satu kunci dalam melakukan enkripsi ataupun dekripsi. Keamanan dari algoritma ini terletak pada kuncinya, jika kunci diberitahukan atau dibocorkan maka siapa saja dapat mengenkrip dan mendekrip data, jadi kunci harus benar-benar rahasia dan aman (Dafid, 2006). Berikut merupakan contoh cara kerja algoritma simetrik yang ada pada gambar 2.4.



Gambar 2.4 Cara kerja Algortima Simetrik

Algoritma simetrik terbagi menjadi 2 jenis yaitu:

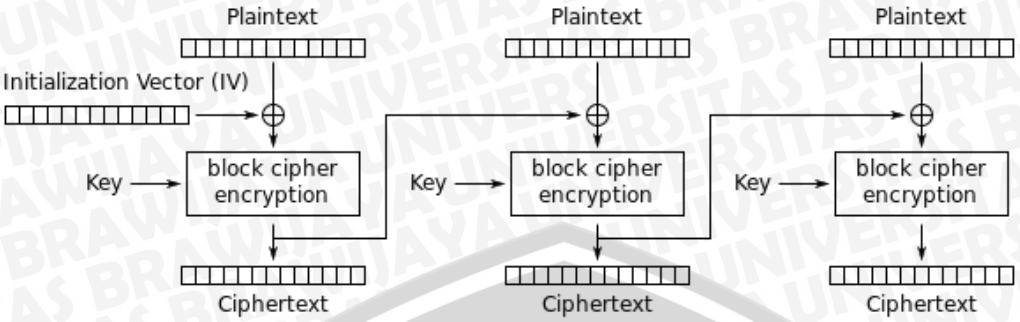
- *Block Cipher*

  Block Cipher adalah algoritma enkripsi yang akan membagi-bagi *plaintext* yang akan dikirimkan dengan ukuran tertentu (disebut blok) dengan panjang tertentu, dan setiap blok dienkripsi dengan menggunakan kunci yang sama. Pada umumnya, *block cipher* memproses *plaintext* dengan blok yang relatif panjang yang lebih dari 64 bit, untuk mempersulit penggunaan pola-pola serangan yang ada untuk membongkar kunci. Contoh *block cipher* diantaranya: DES, 3DES, AES, Blowfish, IDEA.

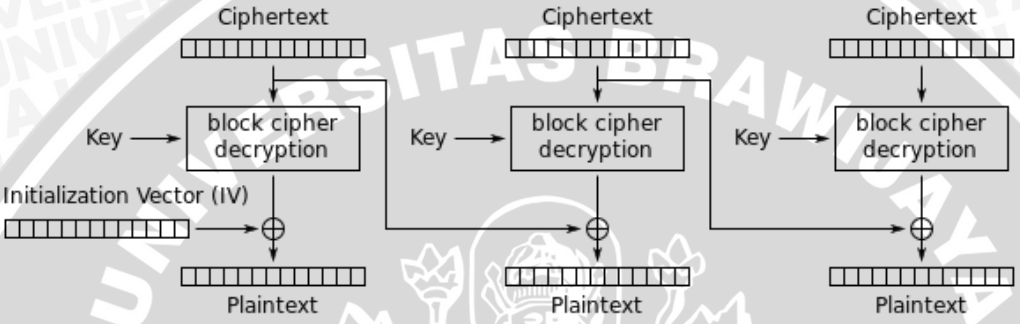
  Block cipher memiliki beberapa pengoperasian yang diantaranya: ECB (*Electronic code Book*), CBC (*Cipher Block Chaining*), OFB (*Output Feed Back*), CFB (*Cipher Feed Back*). Pada penelitian ini, peneliti menggunakan CBC sebagai proses mode pengenkripsiannya.

  - CBC (*Cipher Block Chaining*)

    Mode operasi *Cipher Block Chaining* (CBC) merupakan salah satu mode operasi *block cipher* yang menggunakan vektor inisialisasi (*initialitation vector*/IV). Pada mode operasi ini plaintext dibagi menjadi beberapa blok terebih dahulu. Setelah itu, masing-masing blok dienkripsi dengan ketentuan blok plaintext pertama dienkripsi lebih dahulu. Sebelum dienkripsi, plaintext di-XOR dengan IV. Lalu, hasil XOR tersebut dienkripsi hingga menghasilkan *ciphertext*. Selanjutnya, *ciphertext* tersebut digunakan sebagai IV untuk proses penyandian blok plaintext selanjutnya. Gambar 2.5 merupakan skema CBC pada waktu enkripsi dan gambar 2.6 merupakan skema CBC pada waktu dekripsi.



Gambar 2.5 CBC pada waktu enkripsi



Gambar 2.6 CBC pada waktu dekripsi

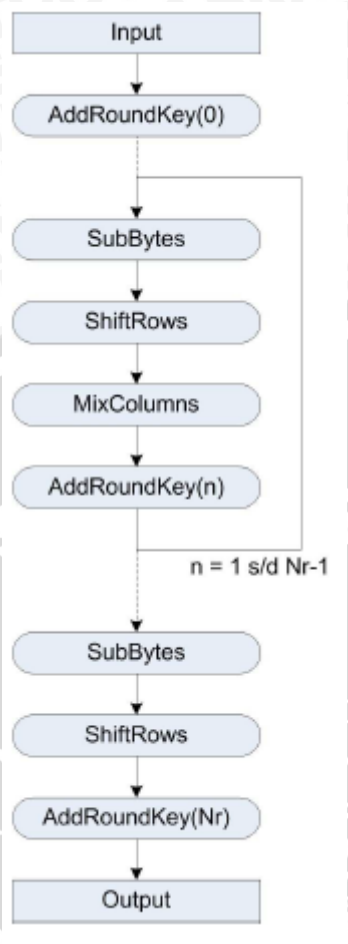
- *Stream Cipher*

  Stream Cipher adalah algoritma enkripsi yang mengenkripsi data persatuan data, seperti bit, byte, nible atau per 5 bit. Setiap mengenkripsi satu satuan data digunakan kunci yang merupakan hasil pembangkitan dari kunci sebelumnya. Kriptografi yang menggunakan *Stream Cipher* diantaranya: RC4 *(Rivest Cipher* 4), SEAL, WAKE, Cellular Automaton.

2.4.1.1 AES (*Advanced Encryption Standard*)

AES (*Advanced Encryption Standard*) adalah teknik enkripsi yang dijadikan standard FIPS oleh NIST tahun 2001. AES dimaksudkan akan secara bertahap, menggantikan DES sebagai standard enkripsi di Amerika Serikat untuk abad ke 21. Pada 2006, AES merupakan salah satu algoritma terpopuler yang digunakan dalam kriptografi kunci simetrik (Kromodimoeljo, 2009).

AES merupakan algoritma *block cipher* dengan menggunakan sistem permutasi dan substitusi (P-Box dan S-Box). Jenis AES terbagi 3, yaitu : AES-128, AES-192, AES-256. Pengelompokkan jenis AES ini adalah berdasarkan panjang kunci yang digunakan. Angka-angka di belakang kata AES menggambarkan panjang kunci yang digunakan pada tipa-tiap AES. Selain itu, hal yang membedakan dari masing-masing AES ini adalah banyaknya round yang dipakai. AES-128 menggunakan 10 *round*, AES-192 sebanyak 12 *round*, dan AES-256 sebanyak 14 *round* (Kromodimoeljo, 2009). Secara garis besar diagram alir enkripsi AES digambarkan dalam Gambar 2.7.



Gambar 2.7 Diagram alir enkripsi AES

Sumber : (Kromodimoeljo, 2009)

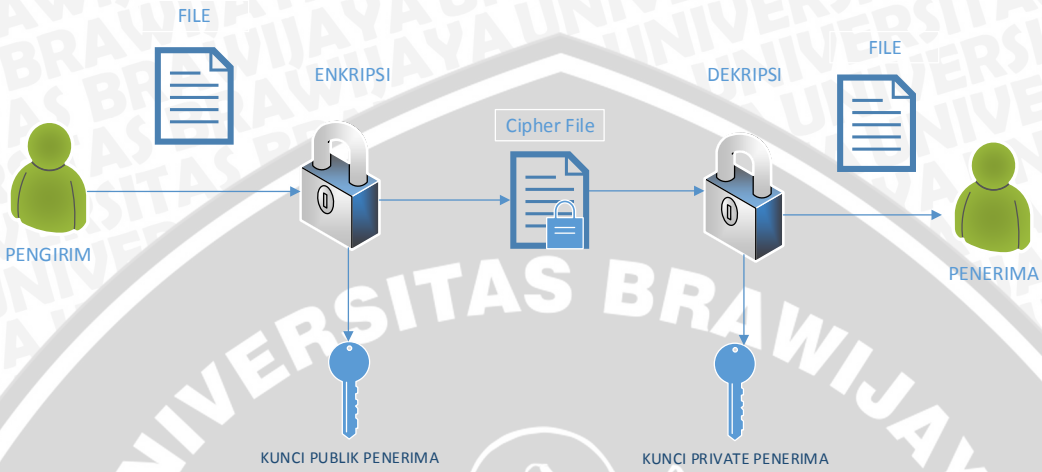
2.4.2 Algoritma Asimetrik

Algoritma asimetrik adalah algoritma yang menggunakan dua buah kunci yang berbeda, yaitu kunci publik dan kunci privat. Kedua kunci ini berisi *cipher* yang berguna untuk merubah *text* menjadi *ciphertext*. Pada proses kriptografi asimetrik salah satu kunci digunakan untuk melakukan *encipher* pada data, dan kunci lainnya digunakan untuk *dechipher* data. Sistem ini juga sering dikenal dengan istilah sistem kunci publik. Kunci yang kita gunakan untuk melakukan *encipher* pada data dapat disebar luaskan. Tetapi kunci yang digunakan untuk melakukan *decipher* data adalah rahasia atau tidak boleh ada orang yang tau. Sebagai contoh, banyak orang dapat melakukan *encipher* data menggunakan kunci publik milik anda, untuk memastikan hanya anda yang memiliki kunci privat yang dapat melakukan *decipher* terhadap data (IBM, 2015).

Algoritma kriptografi kunci publik biasanya digunakan dalam proses untuk mendistribusikan sebuah kunci rahasia atau kunci simetris, menjamin integritas data dan memberikan *non-repudiation* lewat penggunaan *digital signature*.

Algoritma asimetrik dikenal luas dan telah diuji dengan menggunakan kunci yang relatif besar. Hasilnya, waktu dalam pemrosesan membuat algortima ini

kurang ideal untuk digunakan dalam enkripsi data yang membutuhkan tingkat transaksi yang tinggi. Oleh karena itu, sistem kunci publik biasanya digunakan untuk *digital signature* atau sistem pendistribusian kunci. Algoritma ini cukup cepat jika digunakan untuk melakukan *digital signature* pada umumnya. Pada gambar 2.8 dijelaskan bagaimana cara kerja algoritma asimetrik.



Gambar 2.8 Cara kerja algortima asimetrik

2.4.2.1 RSA (*Rivest-Shamir-Adleman*)

RSA merupakan teknik kriptografi yang memiliki 2 kunci atau yang sering disebut algoritma PKA (*public key algorithm*), dimana kunci yang digunakan untuk proses enkripsi berbeda dengan kunci yang digunakan pada proses dekripsi. Kunci yang digunakan untuk melakukan enkripsi disebut kunci publik, sedangkan kunci untuk melakukan dekripsi disebut kunci privat. Orang yang mempunyai kunci publik dapat melakukan enkripsi tetapi yang dapat melakukan dekripsi hanyalah orang yang memiliki kunci privat. Oleh karena itu, kunci publik dapat dimiliki oleh sembarang orang, dalam artian kunci publik dapat disebar luaskan dengan bebas. Akan tetapi, kunci privat hanya boleh dimiliki oleh orang tertentu saja atau bisa dibilang tidak boleh dibagikan atau rahasia (Rahajoeningroem & Aria, 2011).

Untuk pembangkitan pasangan kunci RSA, algoritmanya adalah sebagai berikut:

1. Dipilih dua buah bilangan prima sembarang yang besar, *p* dan *q*. Nilai *p* dan *q* harus dirahasiakan.
2. Dihitung $n = p \times q$. Besaran *n* tidak perlu dirahasiakan.
3. Dihitung $m = (p - 1)(q - 1)$
4. Dihitung *e*, sehingga *e* tidak bisa membagi rata nilai dari *m*.
5. Dihitung kunci privat, $(d \times e) \bmod m = 1$.

Maka hasil dari algoritma tersebut diperoleh :

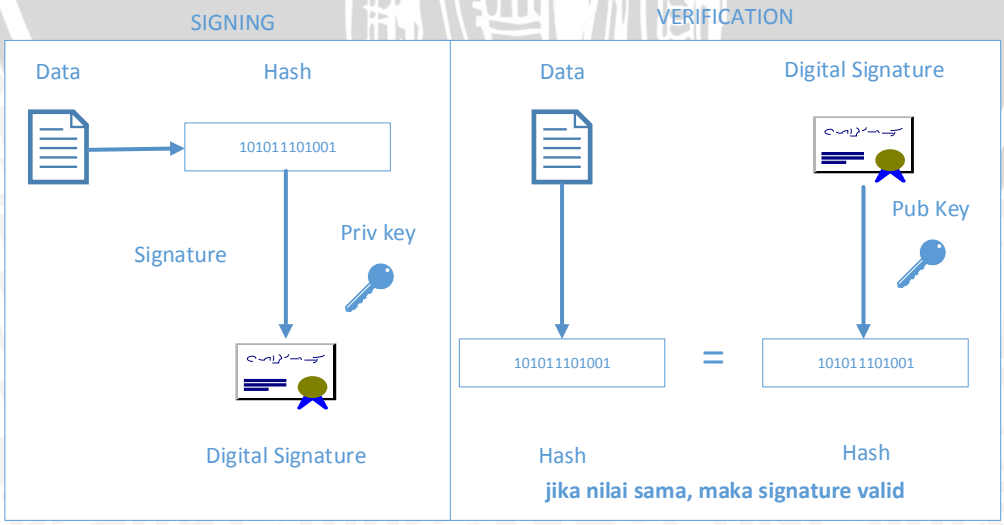
- kunci publik adalah pasangan *(e,n)*
- kunci privat adalah pasangan *(d,n)*
- *n* tidak bersifat rahasia, namun ia diperlukan pada perhitungan enkripsi/dekripsi.

Keamanan algoritma RSA terletak pada tingkat kesulitan dalam memfaktorkan bilangan non prima menjadi faktor primanya, yang dalam hal ini $n = p \times q$. Jika n berhasil difaktorkan menjadi p dan q, maka $m = (p - 1)(q - 1)$ dapat dihitung. Dan karena kunci enkripsi e telah diumumkan (tidak dirahasiakan), maka kunci dekripsi d dapat dihitung melalui persamaan $(d \times e) \bmod m = 1$. Selama belum ditemukan cara untuk memfaktorkan bilangan besar menjadi faktor-faktor primanya, maka selama itu pula keamanan algoritma RSA terjamin.

2.5 *Digital signature*

Digital signature adalah sebuah skema matematis yang digunakan untuk mendemonstrasikan otentikasi ataupun integritas pada pesan digital atau dokumen. Sebuah *digital signature* yang sah, sudah cukup menjadi alasan bagi penerima untuk percaya bahwa sebuah pesan atau dokumen yang diterima adalah berasal dari pengirim yang telah diketahui (M.Barukab, et al., 2012). Dengan begitu, *Digital signature* dapat memenuhi setidaknya tiga syarat keamanan jaringan, yaitu *authentication*, *integrity* dan *Non-repudiation*. *Digital signature* akan diimplementasikan dengan algortima asimetrik dan dikombinasikan dengan algoritma *hashing*.

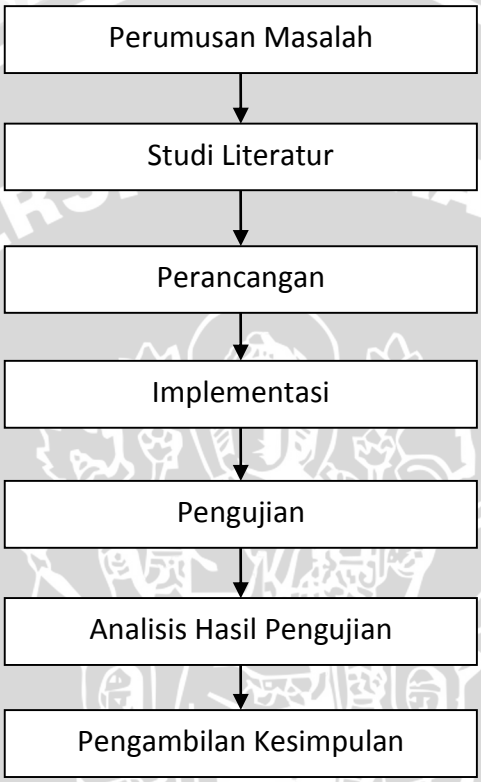
Digital signature berbasis pada PKA (*public key algorithm*), atau yang lebih dikenal dengan istilah algoritma asimetrik. Cara kerja *digital signature* adalah dengan cara melakukan tanda tangan terhadap hash yang sudah didapatkan sebelumnya dari data yang akan di tanda tangani menggunakan kunci privat, ini digunakan untuk memastikan bahwa data atau informasi ini berasal dari si pengirim (*authentication*) dan ini juga dapat digunakan agar si pengirim tidak dapat melakukan penyangkalan terhadap data yang dikirimnya (*Non-Repudiation*). Untuk melakukan pengecekan si penerima dapat dapat menggunakan kunci publik si pengirim untuk menyatakan bahwa data ini berasal dari si pengirim. Gambar 2.9 merupakan diagram alir dari *digital signature*.



Gambar 2.9 diagram alir *Digital signature*

BAB 3 METODOLOGI

Dalam bab ini dibahasa metodologi yang digunakan dalam penelitian rancang bangun sistem pengamanan data menggunakan algoritma simetrik dan asimetrik pada media penyimpanan berbasis *cloud*. Tahapan-tahapan yang akan dilakukan dalam penelitian ini diilustrasikan dalam diagram alir metode penelitian pada Gambar 3.1.



Gambar 3.1 Diagram Alir Metodologi

3.1 Perumusan Masalah

Permasalahan yang dihadapi dalam penelitian ini adalah bagaimana membangun sistem pengamanan data yang akan diimplementasikan pada layanan *cloud storage*. Berdasarkan penjelasan sebelumnya bahwa *loss of privacy* dan *Man in The Middle Attack* ada kemungkinan terjadi. Salah satu cara yang kita bisa lakukan untuk mengatasi permasalahan ini adalah dengan membuat aplikasi yang dapat mengamankan data atau *file* kita baik dalam pengiriman maupun pada saat data berada pada *cloud storage* secara otomatis saat data diunggah dengan menggunakan metode pengenkripsian data menggunakan teknik kriptografi simetrik dan asimetrik sebagai meotde pertukaran kunci simetrik.

3.2 Studi Literatur

Studi literatur mempelajari mengenai dasar teori yang digunakan untuk mennunjang penulisan serta pengerjaan tugas akhir. Teori-teori pendukung penulisan serta pemahaman tentang penelitian diperoleh dari forum, jurnal, *e-book* yang berkaitan dengan topik penelitian ini. Referensi utama yang diperlukan untuk menunjang penulisan ini adalah forum *Crypto, java programming,* algoritma simetrik*,* algoritma asimetrik*,* dan *digital signature*.

3.3 Perancangan Sistem

Pada penelitian ini, proses perancangan sistem dibagi menjadi dua bagian, yaitu tahap pertama dilakukan analisis linkungan sistem, kemudian tahap selanjutnya dilakukan perancangan aplikasi sistem. Analisis kebutuhan sistem ini biasanya yang akan menunjang fungsi utama sistem, diantaranya adalah bagaimana proses otentikasi pengguna pada layanan *cloud* dan bagaimana cara mengetahui *file* apa saja yang telah diunggah dari tiap pengguna serta membuat daftar kebutuhan dari perangkat lunak dan perangkat keras yang digunakan pada penelitian ini. Sedangkan pada tahap kedua terdiri dari perancangan aplikasi sistem secara umum, dijelaskan juga proses/alur kerja sistem. Bagaimana sistem melakukan pengamanan data atau enkripsi pada saat melakukan pengiriman data pada layanan *cloud storage* dan bagaimana melakukan dekripsi saat pengguna mengunduh data yang sudah dienkripsi sebelumnya. Selain itu bagaimana melakukan pendistribusian kunci sehingga pengguna lain juga dapat membaca *file* tersebut.

3.3.1 Lingkungan sistem

Setelah studi literatur dilakukan, perlu dilakukan analisis kebutuhan fungsional dari sistem. Kebutuhan tersebut diantaranya:

1. Diperlukan fungsi *login* pada sistem yang digunakan untuk menghubungkan pengguna dengan layanan *cloud storage*.
2. Diperlukan fungsi yang mampu melakukan *upload* dan *download file* pada layanan *cloud storage*.
3. Diperlukan fungsi yang dapat melakukan *generate key* yang akan digunakan pengguna dalam proses enkripsi dan dekripsi secara otomatis.
4. Diperlukan fungsi yang dapat melakukan enkripsi dan dekripsi *file* yang akan diunggah oleh pengguna secara otomatis.
5. Diperlukan fungsi yang dapat melakukan *signature* pada *file* untuk proses *integrity* dan mengatasi isu *non-repudiation* yang dikirim pengguna.
6. Diperlukan fungsi yang dapat melakukan distribusi kunci yang akan digunakan untuk proses berbagi *file* kepada pengguna lain.

Dari beberapa kebutuhan fungsional sistem yang telah didapat, diperlukan juga beberapa komponen *non*-fungsional yang dapat menunjang pengerjaan sistem ini, yaitu:

1. Kebutuhan perangkat keras (*hardware*)
    a. 1 unit PC/laptop
    b. Jaringan internet
2. Kebutuhan perangkat lunak (*software*)
    a. NetBeans IDE 8.0.2
    b. jdk1.8.0_60
    c. dropbox-java-sdk-1.7.7
    d. Vmware Workstation 10
3. Kebutuhan fitur
    a. Login
    b. *Generate key*
    c. Enkripsi dan dekripsi
    d. *Upload* dan *download*
    e. Melihat *file* yang telah diunggah/*listfile*
    f. *Signature* and *verify*
    g. Share

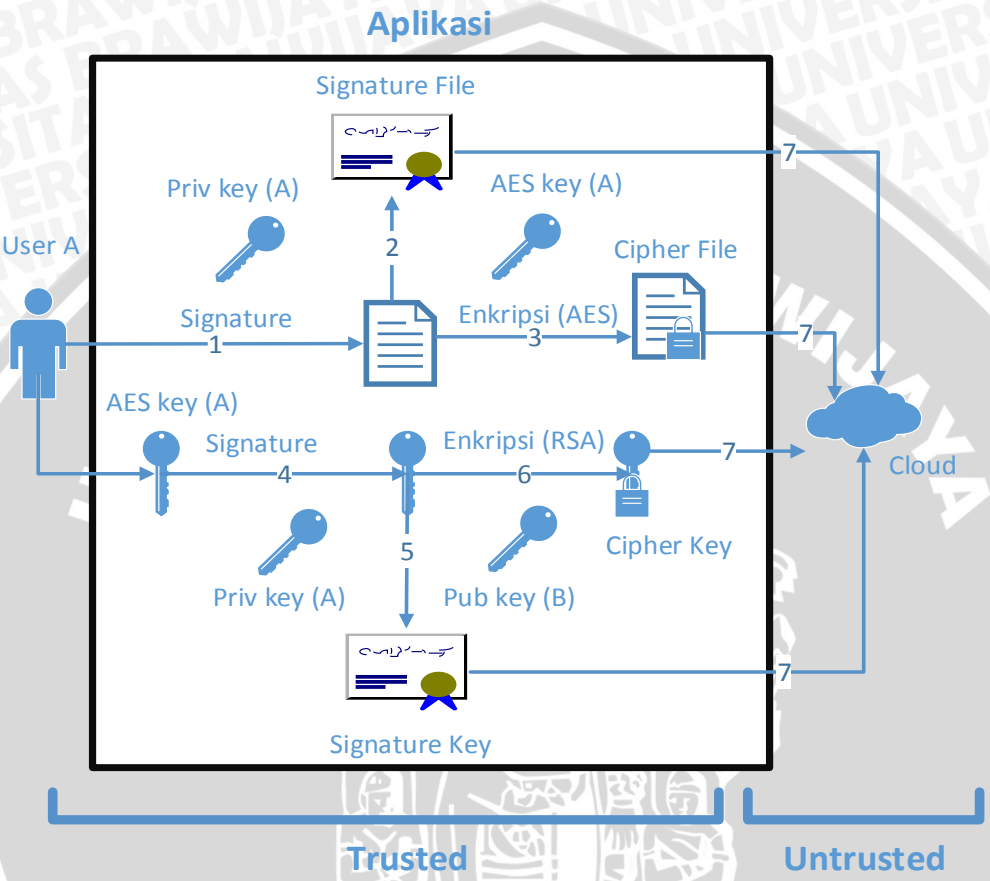
3.3.2 Perancangan Aplikasi

Pada penelitian ini, proses perancangan aplikasi didasarkan pada tiga fungsi utama, yaitu: fungsi *upload* yang akan digunakan untuk mengunggah *file*, kunci dan *signature* ke *cloud storage* dari komputer pengguna*,* fungsi *download* yang akan digunakan untuk mengunduh *file*, kunci dan *signature* dari ke *cloud storage* ke komputer pengguna yang akan digunakan *&* fungsi *share* yang akan digunakan untuk melakukan berbagi *file* antar *user* sehingga *user* dapat bertukar *file*.

Pada gambar 3.2 menjelaskan tentang arsitektur kerja pengiriman data dari jaringan yang aman (*trusted*) ke jaringan yang tidak aman (*untrusted*) pada saat pengguna melakukan *upload* ke layanan *cloud storage* (Dropbox). Pengguna akan memilih *file* yang akan diunggah ke dropbox. Setelah pengguna selesai memilih *file* sistem akan bekerja secara otomatis yang langkah-langkahnya dijelaskan sebagai berikut:

1. *File* yang dipilih akan dilakukan proses *signature file* yang digunakan untuk integritas dan *non-repudiation* nantinya menggunakan kunci privat *signature*.
2. Setelah proses *signature file* selesai, sistem akan menyimpan hasil *signature* kedalam *file*.
3. Setelah proses *signature* selesai, proses selanjutnya yang dilakukan adalah proses enkripsi menggunakan kunci AES yang akan menghasilkan *cipher file*.
4. Setelah proses enkripsi selesai, langkah selanjutnya ialah melakukan *signature* terhadap kunci meggunakan kunci privat *signature* yang sama yang digunakan untuk melakukan *signature file*.
5. Setelah proses *signature* kunci selesai, akan didapati *signature* kunci yang disimpan dalam *file*.

6. Setelah proses *signature* selesai, proses selanjutnya yang dilakukan adalah melakukan enkripsi kunci AES menggunakan kunci publik RSA pengguna lainnya yang akan menghasilkan kunci dalam bentuk *cipher.*
7. Setelah semua proses selesai, sistem akan melakukan pengunggahan terhadap *signature file* dan kunci serta *file* dan kunci yang sudah dalam bentuk *cipher*.



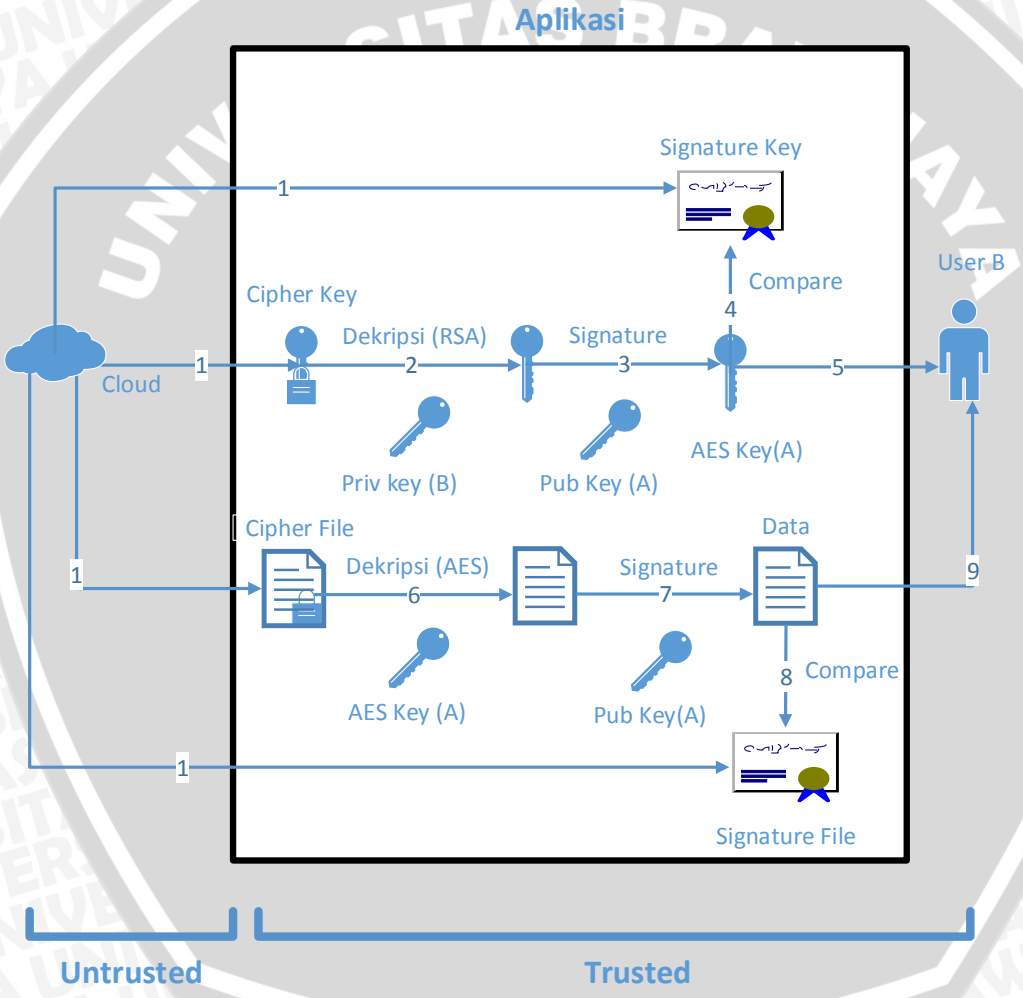
Gambar 3.2 Arsitektur Sistem *Upload*

Pada gambar 3.3 menjelaskan tentang arsitektur kerja sistem pada saat melakukan *download* dari layanan *cloud storage* (Dropbox). Setelah pengguna selesai memilih *file* yang akan diunduh melalui Dropbox sistem akan bekerja secara otomatis untuk melakukan dekripsi data yang langkah-langkahnya dijelaskan sebagai berikut:

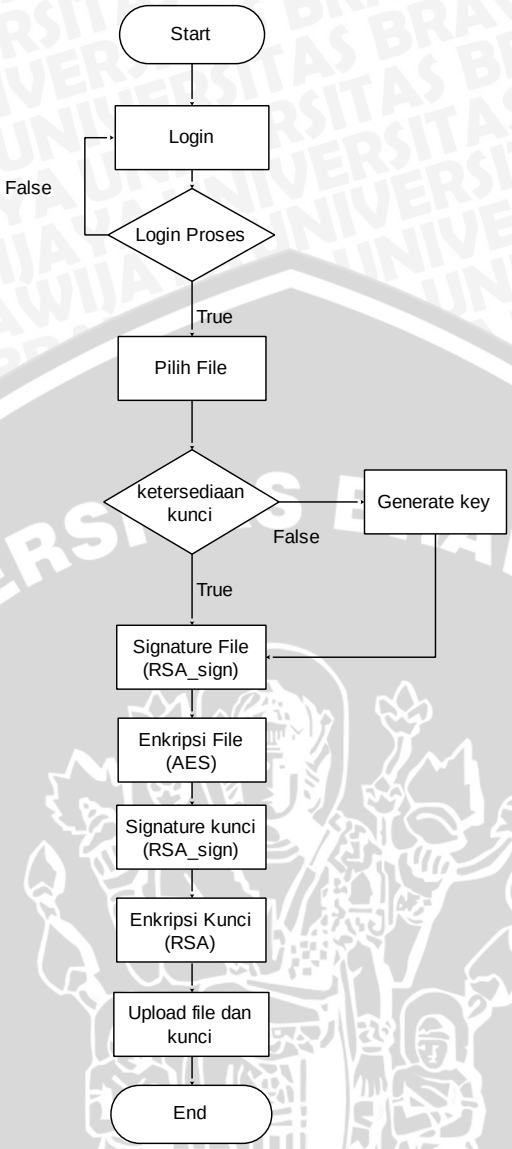
1. *File* akan diunduh sesuai dengan *file* yang dipilih pengguna, pada waktu pengunduhan tidak hanya *file* yang diunduh melainkan kunci beserta *signature file* dan kunci juga diunduh pada awal proses *download*.
2. Setelah proses unduh selesai, *cipher key* yang sudah diunduh akan di dekripsi menggunakan kunci privat RSA.
3. Setelah proses dekripsi kunci selesai, selanjutnya sistem akan melakukan verifikasi *signature* menggunakan kunci publik *signature*.
4. Setelah proses verifikasi selesai, data akan dibandingkan dengan nilai dari *signature key*. Jika nilai sama maka, bisa dipastikan bahwa data yang kita

inginkan benar. Jika nilai berbeda, ada kemungkinan data yang kita unduh sudah dirubah.

5. Setelah proses selesai, pengguna akan mendapatkan kunci AES yang akan digunakan untuk melakukan dekripsi terhadap *cipher file*.
6. *Cipher file* di dekripsi menggunakan kunci AES yang telah di dekripsi sebelumnya.
7. Setelah proses dekripsi *file* selesai, selanjutnya sistem akan melakukan verifikasi *signature* menggunakan kunci publik *signature*.
8. Setelah proses verifikasi selesai, data akan dibandingkan dengan nilai dari *signature file*.
9. Setelah proses selesai, barulah pengguna B dapat melihat isi *file* yang sudah diunduh.



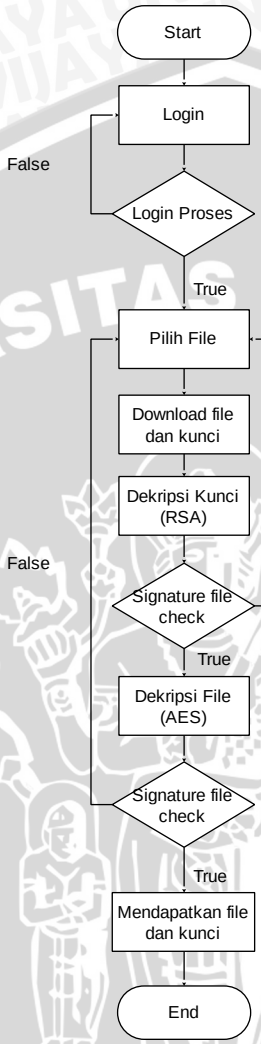
Gambar 3.3 Arsitektur Sistem *Download*



Gambar 3.4 Diagram Alir Proses *Upload*

Diagram alir pada gambar 3.4 menjelaskan bagaimana proses *upload* pada sistem. Ketika aplikasi dijalankan, pengguna harus melakukan *login* pada layanan *cloud storage* pada penelitian ini yang digunakan adalah dropbox. Jadi, pengguna akan melakukan proses otentikasi pada dropbox jika akan menggunakan aplikasi ini. Untuk melakukan *upload*, pengguna harus terlebih dahulu memilih *file* yang akan diunggah. Pada aplikasi ini pengguna dapat mengunggah tipe *file* apapun. Setelah *file* dipilih, sistem akan melakukan pengecekan ketersediaan kunci, jika kunci belum tersedia maka sistem akan secara otomatis melakukan *generate key* terlebih dahulu sebelum melakukan *signature* terhadap *file* menggunakan kunci private RSA_sign. Jika, pengguna telah menggunakan aplikasi ini sebelumnya pengguna akan dapat langsung melakukan *signature file* tanpa melakukan generate key terlebih dahulu. Setelah proses *signature file* selesai aplikasi akan melakukan operasi enkripsi terhadap *file* menggunakan kunci AES. Setelah proses enkripsi *file* selesai sistem akan melakukan *signature* terhadap kunci terlebih

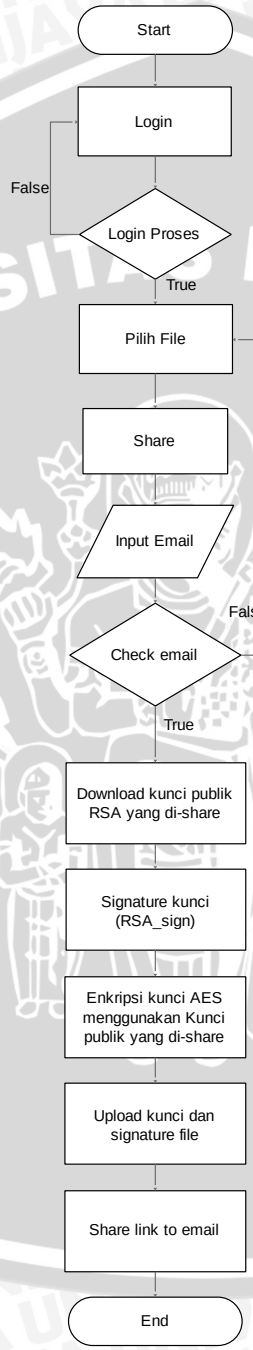
dahulu sebelum melakukan enkripsi terhadap kunci AES dengan menggunakan kunci publik RSA. Kemudian sistem akan melakukan *rename file* dengan format nama_pengguna#nama_*file* ini akan digunakan untuk memudahkan sistem mengetahui kepemilikan *file* tersebut dan *upload* ke dropbox bersamaan dengan kunci publik enkripsi dan kunci publik *signature*.



Gambar 3.5 Diagram Alir Proses *Download*

Diagram alir pada gambar 3.5 menjelaskan bagaimana proses *download file* pada sistem. Ketika aplikasi dijalankan, pengguna harus melakukan *login* pada layanan *cloud storage* pada penelitian ini yang digunakan adalah dropbox. Jadi, pengguna akan melakukan proses otentikasi pada dropbox jika akan menggunakan aplikasi ini. Jika otentikasi berhasil, fungsi-fungsi pada aplikasi akan dapat dijalankan. Pada proses *download* pengguna harus memlih salah satu dari *listfile* yang pernah kita *upload* pada layanan *cloud storage* melalui aplikasi ini. Setelah pengunduhan *file* selesai, *file* tidak bisa langsung kita baca, sistem akan secara otomatis melakukan dekripsi terhadap kunci AES terlebih dahulu menggunakan RSA kunci privat. Setelah kunci AES terdekripsi sistem akan melakukan pengecekan keaslian kunci AES dengan mencocokkan *signature*. Setelah itu kunci AES akan digunakan untuk melakukan proses dekripsi. Setelah

proses dekripsi selesai sistem akan melakukan pengecekan *file* dengan cara mencocokkan *file* yang telah kita unduh dengan *signature file* yang telah dibuat pada saat proses *upload*. Hal ini digunakan untuk pengecekan keaslian *file* dan untuk menghindari isu *non-repudiation* sehingga pengguna bisa mengetahui *file* ini telah dirubah. Setelah proses ini selesai *file* akan dirubah namanya menjadi nama *file* tanpa disertai nama pengguna.



Gambar 3.6 Diagram Alir Proses berbagi *file*

Diagram alir pada gambar 3.6 menjelaskan bagaimana proses *berbagifile* pada sistem. Ketika aplikasi dijalankan, pengguna harus melakukan *login* pada layanan *cloud storage* pada penelitian ini yang digunakan adalah dropbox. Jadi, pengguna

akan melakukan proses otentikasi pada dropbox jika akan menggunakan aplikasi ini. Pada proses *berbagi*pengguna harus memlih salah satu dari *listfile* yang pernah kita *upload* pada layanan *cloud storage* melalui aplikasi ini. Setelah itu pengguna menekan tombol sharing, kemudian pengguna diharuskan menginputkan nama email pengguna yang akan di-*share*. Sistem akan melakukan pengecekan apakah alamat email tersebut pernah menggunakan aplikasi ini atau belum. Kemudian sistem akan melakukan *download* kunci publik RSA pengguna yang akan di-*share.* Selanjutnya, sistem akan melakukan proses enkripsi kunci AES menggunakan kunci publik RSA pengguna yang di-*share* sehingga pengguna yang menerima akan dapat melakukan dekripsi *file* nantinya. Setelah selesai maka sistem akan *upload file*, kunci dan *signature*. Disini, *signature* difungsikan agar pengguna yang dikirimi *file* dapat mengetahui bahwa *file* ini benar-benar dari pengguna tersebut.

3.4 Implementasi

Implementasi akan dilakukan sesuai dengan perancangan sistem yang telah dibuat sebelumnya. Pada bagian ini dibagi menjadi beberapa tahap implementasi, diantaranya:

a. Implementasi perangkat keras (*hardware*). Implementasi ini hanya memerlukan konfigurasi untuk koneksi internet.

b. Implementasi perangkat lunak (*software*). Diperlukan beberapa perangkat lunak yang dapat menunjang jalannya sistem. Perangkat lunak tersebut meliputi: Java, sebagai basis untuk mengembangkan sistem, NetBeans IDE sebagai *compiler* program, JDK (*Java Development Kit*) sebagai alat untuk pengembangan sistem, serta *dropbox-java-sdk-1.7.7* sebagai penghubung aplikasi dengan layanan *cloud*.

Pada saat melakukan implementasi diharapkan sistem dapat berjalan sesuai rancangan. Sehingga dari sistem tersebut bisa didapatkan sebuah aplikasi yang mampu memberikan layanan enkripsi & dekripsi *file* serta *upload, download* dan *share file* pada *cloud storage*.

3.5 Pengujian sistem

Pegujian sistem ini dilakukan untuk memastikan bahwa kinerja sistem bersesuaian dengan spesifikasi kebutuhan yang telah dirancang sebelumnya. Pengujian juga dapat dilakukan untuk menguji otomasi sistem apakah mampu memenuhi kebutuhan-kebutuhan yang telah ditentukan. Pengujian dilakukan dengan menjalankan fungsi sistem dan membandingkannya dengan kebutuhan fungsional yang ada, melakukan pengujian terhadap performansi sistem dan pengujian dilakukan dengan melihat aspek keamanan yang diberikan oleh sistem.

3.6 Analisis dan hasil pengujian

Untuk mengetahui kinerja dari sistem secara keseluruhan, dilakukan analisa dari hasil pengujian yang didapat. Dari analisa hasil akan diketahui juga kelayakan

dari sistem yang dibuat. Analisa hasil juga diperlukan untuk menarik kesimpulan dari penelitian yang telah dilakukan.

3.7 Kesimpulan

Tahap ini merupakan tahapan terakhir dalam penelitian dimana ditarik kesimpulan dari hasil analisa yang didapat dari sistem. Kesimpulan dituliskan sebagai rangkuman dari proses pengerjaan penelitian dan saran dituliskan untuk pengembangan sistem kedepannya ataupun untuk memperbaiki kesalahan-kesalahan yang ada pada sistem saat ini.

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Dalam bab ini akan dijelaskan mengenai tahapan untuk mengimplementasikan rancang bangun sistem pengamanan data menggunakan algoritma simetrik dan asimetrik, serta melakukan pengujian terhadap sistem tersebut. Pada tahap implementasi, langkah-langkah yang dilakukan adalah membuat fungsi enkripsi dengan menggunakan AES yang digunakan untuk mengenkripsi *file* dan RSA yang digunakan untuk mengenkripsi kunci AES, sertamembuat fungsi *verifikasi file* dengan menggunakan *Digital signature* dengan menggunakan algoritma RSA dan menggabungkan semuanya menjadi satu sistem.

4.1 Implementasi

Implementasi dilakukan sesuai dengan perancangan aplikasi yang telah dibuat. Beberapa hal yang dilakukan untuk menerapkan perancangan tersebut diantaranya adalah mempersiapkan perangkat keras dan instalasi perangkat lunak seperti: instalasi NetBeans, konfigurasi API dropbox dan instalasi VMware.

4.1.1 Konfigurasi API

4.1.1.1 API library

Pada penelitian ini, hal yang perlu dilakukan pertama kali adalah dengan menambahkan API *library* dropbox sesuai bahasa pemrograman yang akan digunakan. Pada kasus ini, peneliti menggunakan bahasa pemrograman Java sebagai pengembangan aplikasi. API dropbox yang dipakai pada aplikasi ini adalah API versi 1.7.7. Untuk menambahkan *library* pada aplikasi ini dapat dilakukan dengan dua cara, pertama menggunakan *file pom.xml* yang berisi seluruh *dependencies library* yang diperlukan pada aplikasi ini. Cara kedua adalah dengan mengunduh *file library* yang sudah disediakan di website resmi dropbox. Pada tabel 4.1 merupakan potogan kode *pom.xml* yang akan untuk menambahkan *library* API dropbox.

***Source code* 4.1 Kode untuk menambahkan API dropbox**

```
.....
<dependency>
     <groupId>com.dropbox.core</groupId>
     <artifactId>dropbox-core-sdk</artifactId>
     <version>[1.7,1.8)</version>
</dependency>
.....
```

4.1.1.2 Registrasi API

Sebelum aplikasi dapat menggunakan API dropbox, kita wajib mendaftarkan aplikasi yang akan kita buat pada dropbox. Sehingga aplikasi dapat melakukan full access kepada dropbox dengan melakukan otentikasi. Ketika proses registrasi selesai, pengembang akan mendapatkan sebuah *App key* dan *App secret* yang

dimana ini akan digunakan dalam melakukan proses otentikasi aplikasi dengan dropbox.

4.1.2 Implementasi fungsi Aplikasi

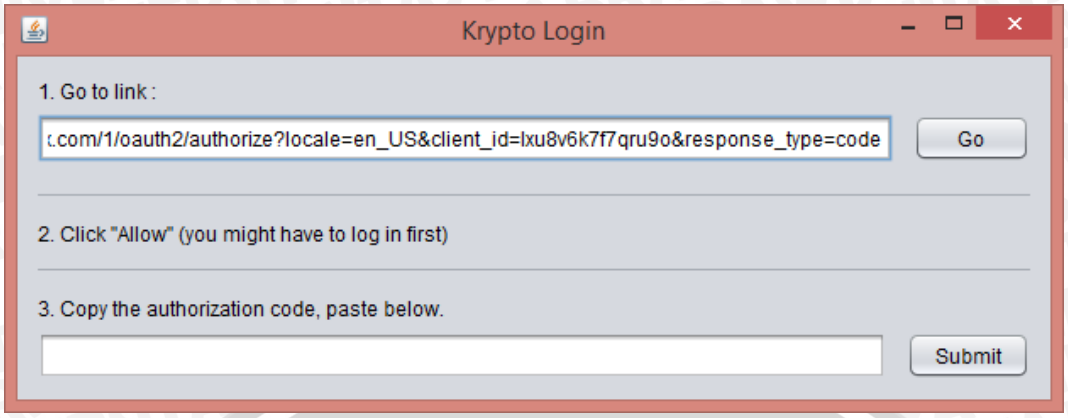
Berdasarkan rancangan sistem yang telah dijelaskan sebelumnya, aplikasi sistem pengamanan data harus mampu memenuhi kebutuhan yang telah dirancang pada bagian sebelumnya.

4.1.2.1 Implementasi Fungsi *Login*

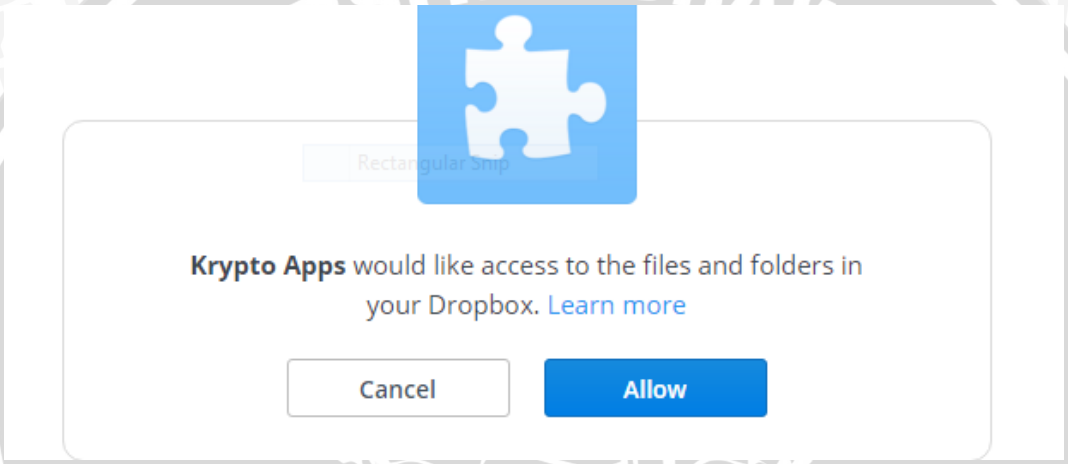
Pada implementasi *login*, sistem ini sebenarnya tidak membuat fitur *login* sendiri. Tetapi, aplikasi ini hanya digunakan sebagai jembatan antara pengguna dan *cloud storage.* Pengguna akan diarahkan ke website resmi *cloud storage*, pada penelitian ini yang digunakan adalah dropbox. Jadi, pengguna akan diarahkan ke *website* dropbox untuk melakukan verifikasi bahwa pengguna akan menggunakan aplikasi ini untuk digunakan sebagai penghubung pengguna dengan *file-file* yang ada di dalam dropbox. Setelah pengguna selesai melakukan verifikasi, pengguna akan mendapatkan kode yang akan dimasukkan kedalam form verifikasi bahwa pengguna tersebut menyetujui untuk menggunakan aplikasi ini. *Source code* 4.2 merupakan potongan kode program verifikasi pengguna. Gambar 4.1 merupakan tampilan antarmuka login yagn digunakan pada aplikasi ini. Gambar 4.2 merupakan tampilan form verifikasi dropbox.

***Source code* 4.2 kode verifikasi pengguna**

```
1  final String APP_KEY = "clientid";
2  final String APP_SECRET = "clientsecret";
3
4  DbxAppInfo appInfo = new DbxAppInfo(APP_KEY, APP_SECRET);
5
6  DbxRequestConfig config = new
7  DbxRequestConfig("JavaTutorial/1.0",
8     Locale.getDefault().toString());
9  DbxWebAuthNoRedirect webAuth = new
10    DbxWebAuthNoRedirect(config, appInfo);
11
12 // Have the user sign in and authorize your app.
13 String authorizeUrl = webAuth.start();
14 DbxClient klien;
15
16 public long DropboxAuth(String authCode) throws DbxException
17 {
18    String code = authCode;
19    DbxAuthFinish authFinish = webAuth.finish(code);
20    String accessToken = authFinish.accessToken;
21    DbxClient client = new DbxClient(config, accessToken);
22    String user = client.getAccountInfo().displayName;
23    klien = client;
24    System.out.println("Dropox linked Account : " + user);
25    System.out.println("--------------------------------");
26    long uid = client.getAccountInfo().userId;
27    return uid;
28 }
```



Gambar 4.1 *Form login*



Gambar 4.2 *Form* verifikasi Dropbox

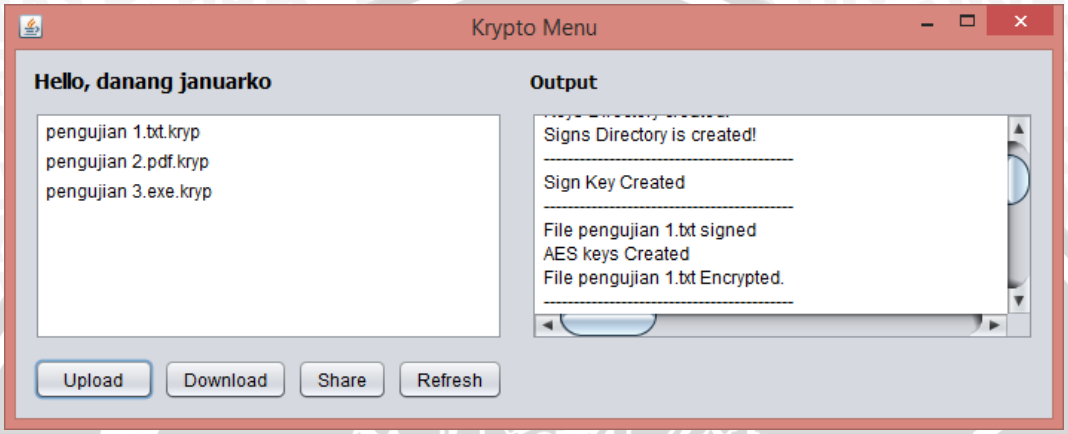
4.1.2.2 Implementasi Fungsi *Listfile*

Fungsi *listfile* ini digunakan untuk menampilkan *file-file* yang diunggah pengguna pada dropbox. Fungsi yang digunakan pada aplikasi ini adalah fungsi *showlist()*. *Source code* 4.3 merupakan potongan kode program fungsi *showlist()*. Gambar 4.3 merupakan hasil implementasi antarmuka *listfile*.

***Source code* 4.3 Fungsi *Showlist*()**

```
1   public void showList() {
2       DefaultListModel listModel = new DefaultListModel();
3       try {
4           DbxEntry.WithChildren listing =
5   klien.getMetadataWithChildren("/");
6           //btnDeleteFile.setEnabled(true);
7           for (DbxEntry anak : listing.children) {
8               if (anak.isFile()) {
9                   String file = anak.name;
10                  if (file.contains(".kryp")) {
11                      String[] parts = file.split("#", 2);
12                      String part1 = parts[0];
13                      String part2 = parts[1];
14                      listModel.addElement(part2);
```

```
15                      }
16                  }
17                  //System.out.println(anak.name);
18              }
19              listFile.setModel(listModel);
20
21          } catch (Exception e) {
22          }
23  }
```



Gambar 4.3 antarmuka *listfile*

4.1.2.3 Implementasi Fungsi *Upload*

Fungsi *upload* ini digunakan untuk mengunggah *file* yang terenkripsi, kunci publik RSA dan kunci publik RSA_sign ke layanan *cloud storage* atau dropbox. Fungsi yang digunakan pada aplikasi ini adalah fungsi uploadFiles*()*. *Source code* 4.4 merupakan potongan kode program fungsi uploadFiles*()*.

***Source code* 4.4 Fungsi uploadFiles()**

```
1   public void uploadFiles(String filepath, String filename) {
2   ......
3   File inputFile = new File(filepath + ".kryp");
4   FileInputStream inputStream = new
5   FileInputStream(inputFile);
6   //Upload Sign Key RSA
7   File keyPublic = new File(AESFileEncryption.dir +
8   "/public.key");
9   FileInputStream streamPublic = new
10  FileInputStream(keyPublic);
11  //Upload Public Public Key
12  File PublicSign = new File(GenSig.dir + "/public.sign");
13  FileInputStream streamPublicSign = new
14  FileInputStream(PublicSign);
15  try {
16      DbxEntry.File uploadedFile = klien.uploadFile("/" + user
17  + "#" + filename + ".kryp",
18              DbxWriteMode.force(), inputFile.length(),
19  inputStream);
20      DbxEntry.File uploadedPublic = klien.uploadFile("/Keys/"
21  + user + "/public.key",
22
```

```
            DbxWriteMode.force(), keyPublic.length(),
streamPublic);
    DbxEntry.File uploadedPublicSign =
klien.uploadFile("/Signs/" + user + "/public.sign",
            DbxWriteMode.force(), PublicSign.length(),
streamPublicSign);
}
......
}
```

4.1.2.4 Implementasi Fungsi *Download*

Fungsi *download* ini digunakan untuk mengunduh *file* dan kunci AES, RSA dan RSA_sign dari dropbox yang kemudian akan dilakukan proses dekripsi sehingga pengguna bisa melihat isi dari file tersebut. Fungsi yang digunakan pada aplikasi ini adalah fungsi downloadFiles*()*. *Source code* 4.5 merupakan potongan kode program fungsi downloadFiles*().*

***Source code* 4.5 Fungsi downloadFiles()**

```
public void downloadFiles() {
......
FileOutputStream outputStream = new
FileOutputStream(selected);
try {
    DbxEntry.File downloadedFile = klien.getFile("/" +
selected, null,
            outputStream);
    File keyFile = new File("Keys/" + part1 + "/AES.key");
    File ivFile = new File("Keys/" + part1 + "/" + part2 +
".iv");
    File skFile = new File("Signs/" + part1 +
"/AES.key.sign");
    File signFile = new File("Signs/" + part1 + "/" + part2
+ ".sign");
    File pubFile = new File("Signs/" + part1 +
"/public.sign");
    if (!keyFile.exists()) {
        new File("Keys/" + part1).mkdirs();
        FileOutputStream keyStream = new
FileOutputStream("Keys/" + part1 + "/AES.key");
        DbxEntry.File downloadedKey = klien.getFile("/Keys/"
+ part1 + "/" + user + "#AES.key", null,
                keyStream);
        keyStream.close();
    }
    if (!ivFile.exists()) {
        FileOutputStream ivStream = new
FileOutputStream("Keys/" + part1 + "/" + part2 + ".iv");
        DbxEntry.File downloadedIv = klien.getFile("/Keys/"
+ part1 + "/" + part2 + ".iv", null,
                ivStream);
        ivStream.close();

    }
    if (!skFile.exists()) {
        new File("Signs/" + part1).mkdirs();

```

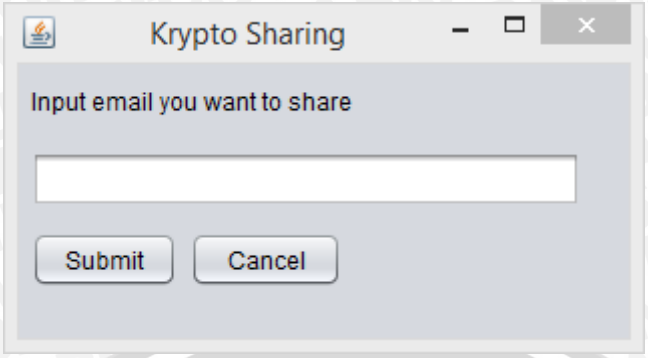
```
40          FileOutputStream skStream = new
41  FileOutputStream("Signs/" + part1 + "/AES.key.sign");
42          DbxEntry.File downloadedSk = klien.getFile("/Signs/"
43  + part1 + "/" + user + "#AES.key.sign", null,
44                  skStream);
45          skStream.close();
46      }
47      if (!signFile.exists()) {
48          FileOutputStream signStream = new
49  FileOutputStream("Signs/" + part1 + "/" + part2 + ".sign");
50          DbxEntry.File downloadedSign =
51  klien.getFile("/Signs/" + part1 + "/" + part2 + ".sign",
52  null,
53                  signStream);
54          signStream.close();
55      }
56      if (!pubFile.exists()) {
57          FileOutputStream pubStream = new
58  FileOutputStream("Signs/" + part1 + "/public.sign");
59          DbxEntry.File downloadedSign =
60  klien.getFile("/Signs/" + part1 + "/public.sign", null,
61                  pubStream);
62          pubStream.close();
63      }
64  ......
65  }
```

4.1.2.5 Implementasi Fungsi *Share*

Fungsi *share* ini digunakan untuk melakukan *berbagifile* ke pengguna lainnya. Pengguna A diharuskan mengisi *email* pengguna B. Fungsi yang digunakan pada aplikasi ini adalah fungsi shareFiles*()*. Pada fungsi ini sistem akan melakukan enkripsi ulang terhadap kunci AES dengan menggunakan kuci publik pengguna B. Sehingga hanya pengguna yang di-*berbagi*yang dapat membuka isi *file* tersebut. *Source code* 4.6 merupakan potongan kode program fungsi shareFiles*().* Gambar 4.4 form input email pengguna yang ingin di-*sharing*.

***Source code* 4.6 Fungsi *shareFiles*()**

```
1  public void ShareFiles(String email) {
2  .....
3  String selected = (String) list File.getSelectedValue();
4  String share = klien.createShareableUrl("/" + selected);
5  .....
6  }
```



Gambar 4.4 *form* input email *share*

4.1.2.6 Implementasi Fungsi *Generate Key*

Fungsi ini akan digunakan untuk melakukan *generate key* AES, RSA dan RSA_sign yang nantinya akan digunakan untuk melakukan enkripsi dan dekripsi serta digunakan untuk proses *signature*. *Source code* 4.7 merupakan potongan kode program fungsi *generateKey* AES. *Source code* 4.8 merupakan potongan kode program fungsi *generateKey* RSA. *Source code* 4.9 merpuakan potongan kode program fungsi *generateKey* RSA_sign.

***Source code* 4.7 Fungsi *generateKey* AES**

```
1   public static void generateKey() {
2           // password to encrypt the file
3           String key = UUID.randomUUID().toString();
4           SecureRandom secureRandom = new SecureRandom();
5           secureRandom.nextBytes(salt);
6           String garam = Arrays.toString(salt);
7
8           try (PrintWriter writer = new PrintWriter(dir +
9   "/AES.tmp")) {
10              writer.println(key);
11              writer.println(garam);
12              writer.close();
13              System.out.println("--------------------------
14  ---------------");
15              System.out.println("AES keys Created");
16          } catch (FileNotFoundException ex) {
17
18  Logger.getLogger(AESFileEncryption.class.getName()).log(Lev
19  el.SEVERE, null, ex);
20          }
21      }
```

***Source code* 4.8 Fungsi *generateKey* RSA**

```
1  public static void generateKeys() {
2      KeyPairGenerator kpg =
3  KeyPairGenerator.getInstance("RSA");
4      kpg.initialize(2048);
5      KeyPair kp = kpg.genKeyPair();
6      PublicKey publicKey = kp.getPublic();
7      PrivateKey privateKey = kp.getPrivate();
8      System.out.println("RSA keys created");
9
10     KeyFactory fact = KeyFactory.getInstance("RSA");
11     RSAPublicKeySpec pub = fact.getKeySpec(publicKey,
12             RSAPublicKeySpec.class);
13     RSAPrivateKeySpec priv = fact.getKeySpec(privateKey,
14             RSAPrivateKeySpec.class);
15
16     saveToFile(AES.dir + "/public.key", pub.getModulus(),
17 pub.getPublicExponent());
18     saveToFile(AES.dir + "/private.key", priv.getModulus(),
19 priv.getPrivateExponent());
20 }
```

***Source code* 4.9 Fungsi *generateKey* RSA_sign**

```
1  public static void generateKeys(String filepath, String
2  filename) {
3      /* Generate a key pair */
4      System.out.println("Sign Key Created");
5      KeyPairGenerator keyGen =
6  KeyPairGenerator.getInstance("RSA");
7      SecureRandom random =
8  SecureRandom.getInstance("SHA1PRNG");
9      keyGen.initialize(2048, random);
10     KeyPair pair = keyGen.generateKeyPair();
11     PrivateKey priv = pair.getPrivate();
12     PublicKey pub = pair.getPublic();
13     byte[] keyP = priv.getEncoded();
14     FileOutputStream keyfosP = new FileOutputStream(dir +
15 "/private.sign");
16     keyfosP.write(keyP);
17     keyfosP.close();
18     /* Save the public key in a file */
19     byte[] key = pub.getEncoded();
20     FileOutputStream keyfos = new FileOutputStream(dir +
21 "/public.sign");
22     keyfos.write(key);
23 keyfos.close();
24     /* Create a Signature object and initialize it with the
25 private key */
26     Signature rsa = Signature.getInstance("SHA1withRSA");
27 }
```

4.1.2.7 Implementasi fungsi Enkripsi

fungsi enkripsi ini akan digunakan untuk melakukan enkripsi terhadap *file* dan enkripsi kunci simetrik/AES. Jadi, pada fungsi ini ada 2 tahapan yaitu: melakukan

enkripsi *file* menggunakan kunci AES dan mengenkripsi kunci AES menggunakan Kunci publik RSA. Untuk melakukan enkripsi *file* digunakan fungsi dari AESFileEncryption(). Sedangkan untuk melakukan enkripsi terhadap kunci AES/simetrik digunakan fungsi rsaEncrypt(). *Source code* 4.10 merupakan potongan kode program fungsi enkripsi *file* menggunakan AES. *Source code* 4.11 merupakan potongan kode program fungsi enkripsi kunci AES menggunakan kunci publik RSA.

Source code* 4.10 Fungsi enkripsi *file

```
1  public class AESFileEncryption {
2  .....
3  public static void main(String filepath, String filename,
4  String user) {
5  .....
6  inFile = new FileInputStream(filepath);
7  // encrypted file
8  FileOutputStream outFile = new FileOutputStream(filepath +
9  ".kryp");
10 SecretKeyFactory factory =
11 SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1");
12 KeySpec keySpec = new PBEKeySpec(kunci.toCharArray(),
13 slt.getBytes(), 65536, 128);
14 SecretKey secretKey = factory.generateSecret(keySpec);
15 SecretKey secret = new SecretKeySpec(secretKey.getEncoded(),
16 "AES");
17 //
18 Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
19 cipher.init(Cipher.ENCRYPT_MODE, secret);
20 AlgorithmParameters params = cipher.getParameters();
21     byte[] iv =
22 params.getParameterSpec(IvParameterSpec.class).getIV();
23     ivOutFile.write(iv);
24     ivOutFile.close();
25 }   //file encryption
26 byte[] input = new byte[64];
27 int bytesRead;
28 while ((bytesRead = inFile.read(input)) != -1) {
29     byte[] output = cipher.update(input, 0, bytesRead);
30     if (output != null) {
31         outFile.write(output);
32     }
33 }
34 byte[] output = cipher.doFinal();
35 if (output
36         != null) {
37     outFile.write(output);
38 }
39 inFile.close();
40 outFile.flush();
41 outFile.close();
42 .....
43 }
44 }
```

***Source code* 4.11 Fungsi enkripsi kunci AES**

```
1  public void rsaEncrypt(String file_loc, String file_des,
2  String user) {
3  .....
4  Key pubKey = readKeyFromFile(AES.dir + "/public.key", null);
5              Cipher cipher = Cipher.getInstance("RSA");
6              cipher.init(Cipher.ENCRYPT_MODE, pubKey);
7
8              fileIn = new FileInputStream(file_loc);
9              fileOut = new FileOutputStream(file_des);
10             cipherOut = new CipherOutputStream(fileOut,
11 cipher);
12
13             // Read in the data from the file and encrypt it
14             while ((i = fileIn.read(data)) != -1) {
15                 cipherOut.write(data, 0, i);
16             }
17
18             // Close the encrypted file
19             cipherOut.close();
20             fileIn.close();
21 .....
22 }
```

4.1.2.8 Implementasi fungsi Dekripsi

fungsi dekripsi ini akan digunakan untuk melakukan dekripsi terhadap *file* dan dekripsi kunci simetrik/AES. Jadi, pada fungsi ini ada 2 tahapan yaitu: melakukan dekripsi *file* menggunakan kunci AES dan mendekripsi kunci AES menggunakan kunci privat RSA. Untuk melakukan dekripsi terhadap *file* digunakan fungsi dari AESFileDecryption(). Sedangkan untuk melakukan dekripsi terhadap kunci AES/simetrik digunakan fungsi rsaDecrypt(). *Source code* 4.12 merupakan potongan kode program fungsi dekripsi *file* menggunakan AES. *Source code* 4.13 merupakan potongan kode program fungsi enkripsi kunci AES menggunakan kunci privat RSA.

Source code* 4.12 Fungsi dekripsi *file

```
1  public class AESFileDecryption {
2  .....
3  public boolean main(String selected, DbxClient klien) {
4  .....
5  SecretKeyFactory factory =
6  SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1");
7  KeySpec keySpec = new
8  PBEKeySpec(AESFileEncryption.kunci.toCharArray(),
9  AESFileEncryption.slt.getBytes(), 65536,128);
10 SecretKey tmp = factory.generateSecret(keySpec);
11 SecretKey secret = new SecretKeySpec(tmp.getEncoded(),
12 "AES");
13
14 // file decryption
15 Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
16 cipher.init(Cipher.DECRYPT_MODE, secret, new
17 IvParameterSpec(iv));
```

```
18  FileInputStream fis = new FileInputStream(selected);
19  String outFile = FilenameUtils.removeExtension(selected);
20  FileOutputStream fos = new FileOutputStream(outFile);
21  byte[] in = new byte[64];
22  int read;
23  while ((read = fis.read(in)) != -1) {
24      byte[] output = cipher.update(in, 0, read);
25      if (output != null) {
26          fos.write(output);
27      }
28  }
29
30  byte[] output = cipher.doFinal();
31  if (output != null) {
32      fos.write(output);
33  }
34  fis.close();
35  fos.flush();
36  fos.close();
37  .....
38  }
39  .....
40  }
```

***Source code* 4.13 Fungsi dekripsi kunci AES**

```
1   public boolean rsaDecrypt(String file_loc, String file_des,
2   String selected, String user) {
3   .....
4   Key priKey = readKeyFromFile(dir + "/private.key", null);
5               Cipher cipher = Cipher.getInstance("RSA");
6               cipher.init(Cipher.DECRYPT_MODE, priKey);
7
8               fileIn = new FileInputStream(file_loc);
9               cipherIn = new CipherInputStream(fileIn,
10  cipher);
11              fileOut = new FileOutputStream(file_des);
12
13              // Write data to new file
14              while ((i = cipherIn.read()) != -1) {
15                  fileOut.write(i);
16              }
17              // Close the file
18              fileIn.close();
19              cipherIn.close();
20              fileOut.close();
21  .....
22  }
```

4.1.2.9 Implementasi fungsi *Signature*

fungsi *signature* ini akan digunakan untuk melakukan *signature* terhadap *file* dan *signature* terhadap kunci simetrik/AES. Jadi, pada fungsi ini ada 2 tahapan yaitu: melakukan *signature file* dan *signature* kunci AES menggunakan kunci privat RSA_sign. Untuk melakukan *signature* terhadap *file* digunakan fungsi dari SignFile(). Sedangkan untuk melakukan *signature* terhadap kunci AES/simetrik

juga digunakan fungsi dari SignFile(). *Source code* 4.14 merupakan potongan kode program fungsi *signature*.

Source code* 4.14 Fungsi *signature

```
1  public static void SignFile(Signature rsa, String filepath,
2  String filename) {
3      FileInputStream fis = null;
4      try {
5          /* Update and sign the data */
6          fis = new FileInputStream(filepath);
7          BufferedInputStream bufin = new
8  BufferedInputStream(fis);
9          byte[] buffer = new byte[1024];
10         int len;
11         while (bufin.available() != 0) {
12             len = bufin.read(buffer);
13             rsa.update(buffer, 0, len);
14         }
15         bufin.close();
16         /* Now that all the data to be signed has been read
17 in,
18          generate a signature for it */
19         byte[] realSig = rsa.sign();
20         /* Save the signature in a file */
21         FileOutputStream sigfos = new FileOutputStream(dir +
22 "/" + filename + ".sign");
23         sigfos.write(realSig);
24         System.out.println("File " + filename + " signed");
25         sigfos.close();
26     } catch (FileNotFoundException ex) {
27
28 Logger.getLogger(GenSig.class.getName()).log(Level.SEVERE,
29 null, ex);
30     } catch (IOException | SignatureException ex) {
31         JOptionPane.showMessageDialog(null, "Invalid
32 Signature!!!", "Error", JOptionPane.ERROR_MESSAGE);
33     } finally {
34         try {
35             fis.close();
36         } catch (IOException ex) {
37
38 Logger.getLogger(GenSig.class.getName()).log(Level.SEVERE,
39 null, ex);
40         }
41     }
42 }
```

4.1.2.10 Implementasi fungsi verify *Signature*

fungsi *verify signature* ini akan digunakan untuk melakukan *verifikasi signature* terhadap *file* dan *verifikasi signature* terhadap kunci simetrik/AES. fungsi ini akan digunakan untuk *authentikasi* pengguna terhadap *file* dan kunci. Jadi, pada fungsi ini ada 2 tahapan yaitu: melakukan *verifikasi signature file* dan *verifikasi signature* kunci AES menggunakan kunci publik RSA_sign. Untuk melakukan *verify signature* terhadap *file* digunakan fungsi checkFile(). Sedangkan untuk melakukan *verify*

signature terhadap kunci AES/simetrik juga digunakan fungsi dari checkKey(). *Source code* 4.15 merupakan potongan kode program fungsi *verify signature* terhadap *file*. *Source code* 4.16 merupakan potongan kode program fungsi *verify signature terhadap* kunci.

Source code* 4.15 Fungsi *verify signature file

```
1  public static void checkFile(String selected, DbxClient
2  klien) {
3  .....
4      FileInputStream keyfis = new FileInputStream(dir +
5  "/public.sign");
6      byte[] encKey = new byte[keyfis.available()];
7      keyfis.read(encKey);
8
9      keyfis.close();
10
11     X509EncodedKeySpec pubKeySpec = new
12 X509EncodedKeySpec(encKey);
13
14     KeyFactory keyFactory = KeyFactory.getInstance("RSA");
15     PublicKey pubKey =
16 keyFactory.generatePublic(pubKeySpec);
17
18     /* input the signature bytes */
19     FileInputStream sigfis = new FileInputStream(dir + "/" +
20 part2 + ".sign");
21     byte[] sigToVerify = new byte[sigfis.available()];
22     sigfis.read(sigToVerify);
23
24     sigfis.close();
25
26     /* create a Signature object and initialize it with the
27 public key */
28     Signature sig = Signature.getInstance("SHA1withRSA");
29     sig.initVerify(pubKey);
30
31     /* Update and verify the data */
32     FileInputStream datafis = new FileInputStream(nFile);
33     BufferedInputStream bufin = new
34 BufferedInputStream(datafis);
35
36     byte[] buffer = new byte[1024];
37     int len;
38     while (bufin.available() != 0) {
39         len = bufin.read(buffer);
40         sig.update(buffer, 0, len);
41     }
42
43     bufin.close();
44
45     boolean verifies = sig.verify(sigToVerify);
46 .....
47 }
```

Source code* 4.16 Fungsi *verify signature key

```
1  public static void checkKey(String selected, DbxClient
2  klien) {
3  .....
4      FileInputStream keyfis = new FileInputStream(dir +
5  "/public.sign");
6      byte[] encKey = new byte[keyfis.available()];
7      keyfis.read(encKey);
8
9      keyfis.close();
10
11     X509EncodedKeySpec pubKeySpec = new
12 X509EncodedKeySpec(encKey);
13
14     KeyFactory keyFactory = KeyFactory.getInstance("RSA");
15     PublicKey pubKey =
16 keyFactory.generatePublic(pubKeySpec);
17
18     /* input the signature bytes */
19     FileInputStream sigfis = new FileInputStream(dir + "/" +
20 "/AES.key.sign");
21     byte[] sigToVerify = new byte[sigfis.available()];
22     sigfis.read(sigToVerify);
23
24     sigfis.close();
25
26     /* create a Signature object and initialize it with the
27 public key */
28     Signature sig = Signature.getInstance("SHA1withRSA");
29     sig.initVerify(pubKey);
30
31     /* Update and verify the data */
32     FileInputStream datafis = new FileInputStream(nFile);
33     BufferedInputStream bufin = new
34 BufferedInputStream(datafis);
35
36     byte[] buffer = new byte[1024];
37     int len;
38     while (bufin.available() != 0) {
39         len = bufin.read(buffer);
40         sig.update(buffer, 0, len);
41     }
42
43     bufin.close();
44
45     boolean verifies = sig.verify(sigToVerify);
46 .....
47 }
```

4.2 Pengujian

Dalam penelitian ini, pengujian dibedakan menjadi 3 tahap. Tahap pertama adalah pengujian validasi sistem dari setiap fitur kebutuhan yang sudah di definisikan sebelumnya pada aplikasi sistem pengmanan data. Tahap kedua adalah pengujian performa sistem pada proses *upload* dengan menggunakan

enkripsi dan non-*enkripsi*. Performa yang diuji adalah seberapa lama waktu yang dibutuhkan sistem untuk melakukan enkripsi pada saat *upload*. Pengujian ketiga adalah pengujian keamanan data, ini dilakukan untuk mengetahui sistem sudah memenuhi standar yang sudah ada ditinjau dari segi keamanan jaringan.

4.2.1 Pengujian Validasi Fitur Sistem

Pengujian validasi digunakan untuk menguji apakah fitur-fitur yang digunakan oleh sistem sudah berjalan dengan baik atau belum. Fitur-fitur yang telah dirancang sesuai daftar kebutuhan merupakan acuan yang digunakan dalam melakukan *validation testing.* Pada pengujian validasi aplikasi sistem pengamanan data, pengujian dilakukan dengan menjalankan semua fitur-fitur yang telah dibuat dan memastikan semua fitur yang telah dibuat bekerja dengan baik.

4.2.2 Pengujian Performansi Sistem

Pengujian perfomansi dilakukan untuk mengetahui seberapa efektif sistem pengamanan data. pengujian ini dilakukan dengan cara menghitung berapa lama kinerja sistem dengan melakukan enkripsi *file* sebelum diunggah dengan proses unggah tanpa melakukan enkripsi. Pengujian ini dilakukan menggunakan beberapa *file* dengan ukuran berbeda sebagai bahan uji performansi.

4.2.3 Pengujian keamanan data

Pengujian ini dilakukan untuk mengetahui apakah sistem sudah memenuhi aspek-aspek kemanan jaringan, yang diantaranya: *integrity*, *confidentiality*, *authentication* dan *non-repudiation*. Pengujian akan dilakukan dengan menjalankan sistem dengan skenario yang telah ditentukan. Untuk *confidentiality* akan dilakukan pengujian terhadap proses enkripsi yang dilakukan oleh sistem dalam mengamankan *file*. sedangkan untuk pengujian *integrity*, *authentication* dan *non-repudiation* dapat dilakukan dengan memanfaatkan proses kerja *digital signature*.

BAB 5 HASIL PENGUJIAN DAN ANALISIS

Bab ini membahas tentang hasil pengujian dan analisis terhadap sistem yang telah dibangun. Hasil pengujian didapatkan dari pengujian yang telah dilakukan sesuai dengan skenario pengujian yang sudah ditentukan pada bab sebelumnya. Dari hasil tersebut dapat dilakukan analisis terhadap kinerja sistem, serta mengetahui bagaimana sistem mencapai tujuan.

5.1 Hasil Pengujian

Pengujian yang dilakukan dalam penelitian ini meliputi beberapa macam pengujian, meliputi pengujian fungsional dan pengujian performansi pada waktu *upload* dan *download* dari sistem.

5.1.1 Hasil pengujian fungsional sistem

Pengujian fungsional dilakukan untuk memastikan bahwa semua kebutuhan-kebutuhan yang telah didefinisikan pada bab perancangan telah dipenuhi dalam sistem aplikasi. Pada pengujian fungsional, aplikasi sistem pengamanan data akan diuji tiap fiturnya, dimana dilakukan pengujian sesuai skenario dengan hasil pengujian seperti yang disajikan pada tabel 5.1.

Tabel 5.1 Tabel Hasil Pengujian Fungsional Sistem

Fitur Utama	**Status**
Login	Terpenuhi
Listfile	Terpenuhi
Upload file	Terpenuhi
Download file	Terpenuhi
Berbagifile	Terpenuhi

Fungsi *Generate Key*	**Status**
Generate Key AES	Terpenuhi
Generate Key RSA	Terpenuhi
Generate Key RSA_sign	Terpenuhi

Fitur Enkripsi	**Status**
Enkripsi *file* menggunakan kunci AES	Terpenuhi
Enkripsi kunci AES menggunakan kunci publik RSA	Terpenuhi

Fitur Dekripsi	Status
Dekripsi *file* menggunakan kunci AES	Terpenuhi
Dekripsi kunci AES menggunakan kunci publik RSA	Terpenuhi

Fungsi *Signature*	Status
Signature file menggunakan kunci privat RSA_sign	Terpenuhi
Signature kunci AES menggunakan kunci privat RSA_sign	Terpenuhi

Fungsi *Verify Signature*	Status
Verifiy Signature file menggunakan kunci publik RSA_sign	Terpenuhi
Verify Signature kunci AES menggunakan kunci publik RSA_sign	Terpenuhi

Fitur *Share*	Status
Enkripsi *file* menggunakan AES	Terpenuhi
Enkripsi kunci AES menggunakan kunci publik RSA pengguna yang akan dikirimi *file*	Terpenuhi
Share link *file* via *email*	Terpenuhi

Dari data pada tabel 5.1 dapat disimpulkan bahwa untuk tiap fitur dari sistem pengamanan data dapat berjalan baik serta semua fitur pada tabel tersebut telah memenuhi semua kebutuhan yang telah dirancang pada bab perancangan.

5.1.2 Hasil pengujian performansi sistem

Pengujian perfomansi dilakukan untuk mengetahui waktu yang dibutuhkan oleh sistem pengamanan data. pengujian ini dilakukan dengan cara menghitung berapa lama kinerja sistem dengan melakukan enkripsi *file* sebelum diunggah dengan proses *upload* tanpa melakukan enkripsi. Pengujian ini dilakukan pada Laboratorium Kecerdasan Buatan Universitas Brawijaya. Pengujian ini juga dilakukan dengan menggunakan beberapa *file* dengan ukuran berbeda sebagai bahan uji performansi. Pengujian ini menggunakan 3 *file* dengan ukuran 5KB, 1MB, 13MB. Pengujian dilakukan pada Laboratorium Kecerdasan Buatan.

Tabel 5.2 Percobaan *Upload file* dengan ukuran 5KB

Percobaan ke -	Hasil	
	Menggunakan sistem (enkripsi)	**Tanpa sistem (*non* enkripsi)**
1	5647 milisecond	2521 milisecond
2	4383 milisecond	1500 milisecond
3	4264 milisecond	1516 milisecond

Tabel 5.2 Percobaan *Upload file* dengan ukuran 5KB (Lanjutan)

Percobaan ke -	Hasil	
	Menggunakan sistem (enkripsi)	Tanpa sistem (*non* enkripsi)
4	4545 milisecond	1515 milisecond
5	4964 milisecond	2269 milisecond
6	4507 milisecond	1739 milisecond
7	4817 milisecond	1540 milisecond
8	4857 milisecond	1590 milisecond
9	5001 milisecond	1472 milisecond
10	4537 milisecond	1493milisecond
Rata-rata	4752.2 milisecond	1715.5 milisecond

Tabel 5.2 merupakan hasil dari sistem pada proses *upload* dengan menggunakan enkripsi dan tanpa enkripsi yang dilakukan sebanyak 10 kali. Dari tabel diatas dapat disimpulkan bahwa aplikasi sistem pengamanan data membutuhkan waktu kurang lebih 3 detik lebih lama dibandingkan tanpa menggunakan enkripsi dengan estimasi rata-rata waktu *upload* menggunakan *file* sebesar 500KB 4.7 detik.

Tabel 5.3 Percobaan *Upload file* dengan ukuran 1MB

Percobaan ke -	Hasil	
	Menggunakan sistem (enkripsi)	Tanpa sistem (*non* enkripsi)
1	8134 milisecond	4038 milisecond
2	7400 milisecond	3989 milisecond
3	8865 milisecond	4128 milisecond
4	7666 milisecond	4230 milisecond
5	7021 milisecond	5621 milisecond
6	8365 milisecond	4374 milisecond
7	8082 milisecond	4586 milisecond
8	7951 milisecond	3352 milisecond
9	7738 milisecond	3383 milisecond
10	6305 milisecond	3358 milisecond
Rata-rata	7752.7 milisecond	4105.9 milisecond

Tabel 5.3 merupakan hasil dari sistem pada proses *upload* dengan menggunakan enkripsi dan tanpa enkripsi yang dilakukan sebanyak 10 kali. Dari

tabel diatas dapat disimpulkan bahwa aplikasi sistem pengamanan data membutuhkan waktu kurang lebih 3.5 detik lebih lama dibandingkan tanpa menggunakan enkripsi dengan estimasi *rata-rata* waktu *upload* dengan file sebesar 1MB sebesar 7.7 detik.

Tabel 5.4 Percobaan *Upload file* dengan ukuran 13MB

Percobaan ke -	Hasil	
	Menggunakan sistem (enkripsi)	Tanpa sistem (non enkripsi)
1	17233 milisecond	13334 milisecond
2	16438 milisecond	13810 milisecond
3	16537 milisecond	12455 milisecond
4	16378 milisecond	15863 milisecond
5	17253 milisecond	13349 milisecond
6	16299 milisecond	13521 milisecond
7	17230 milisecond	13321 milisecond
8	16914 milisecond	12608 milisecond
9	17715 milisecond	13032 milisecond
10	17050 milisecond	15232 milisecond
Rata-rata	16905 milisecond	13653 milisecond

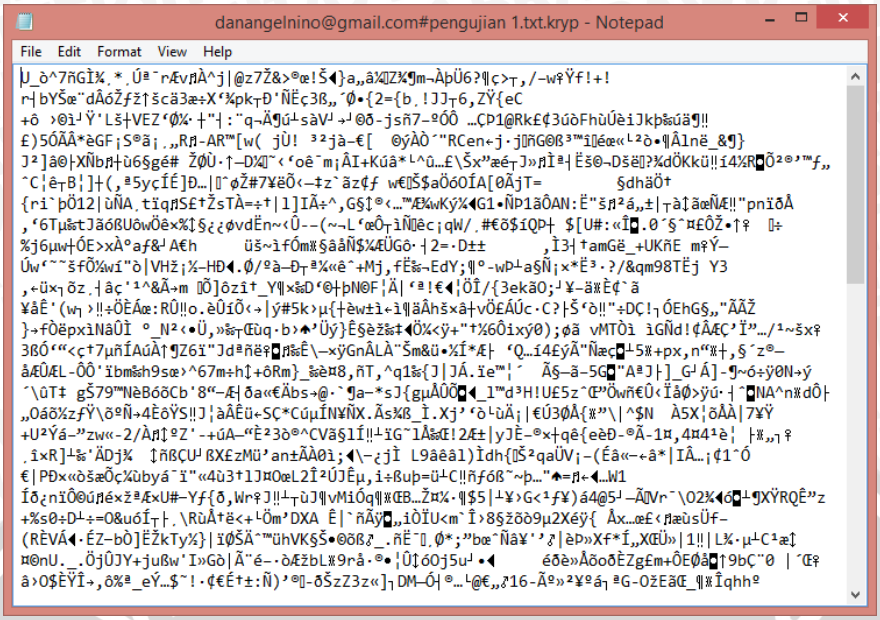
Tabel 5.4 merupakan hasil dari sistem pada proses *upload* dengan menggunakan enkripsi dan tanpa enkripsi yang dilakukan sebanyak 10 kali. Dari tabel diatas dapat disimpulkan bahwa aplikasi sistem pengamanan data membutuhkan waktu kurang lebih 3 detik lebih lama dibandingkan tanpa menggunakan enkripsi dengan estimasi *rata-rata* waktu *upload* dengan file sebesar 13MB sebesar 16.9 detik.

5.1.3 Hasil pengujian keamanan jaringan

Pengujian kemanan jaringan dilakukan untuk mengetahui tingkat keamanan sebuah data yang dilakukan oleh sistem. Untuk itu dilakukan percobaan dalam fungsi enkripsi dan fungsi dekripsi untuk mengetahui keamanan jaringan pada aspek *confidentiality, integrity, authentication dan non-repudiation*.

5.1.3.1 *Confidentiality*

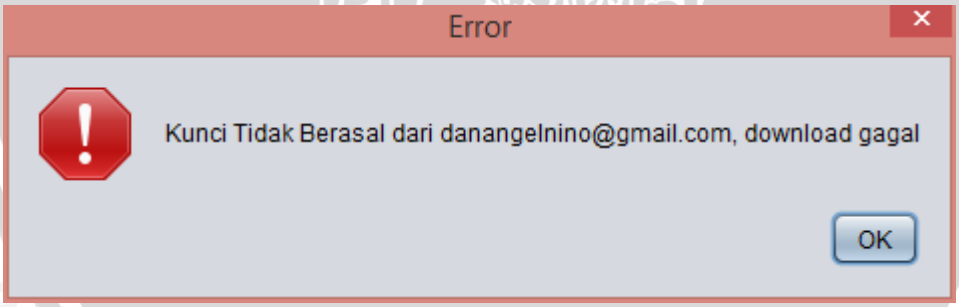
Pada pengujian aspek keamanan jaringan *confidentiality*, skenario pengujian yang dilakukan adalah membuka *file* langsung dari dropbox tanpa melakukan dekripsi *file* terlebih dahulu. Dengan membuka *file* secara paksa, ini sudah dapat dikategorikan sebagai percobaan pencurian data. Hasil *file* yang dibuka tanpa hak akses tidak dapat dibaca. Gambar 5.1 menampilkan hasil *file* yang dibuka tanpa ijin.



Gambar 5.1 hasil *file* dibuka tanpa ijin

5.1.3.2 Integrity

Pada pengujian aspek keamanan *integrity,* skenario pengujian yang dilakukan adalah dengan mencoba mengganti kunci publik *signature* atau menggunakan kunci publik *signature* yang bukan milik pengirim. Dengan melakukan perubahan kunci diibaratkan ada orang yang dengan sengaja merubah *file* atau mengganti kunci AES. Respon yang dihasilkan oleh sistem akan menunjukkan bahwa data telah diubah. Gambar 5.2 dan gambar 5.3 menampilkan respon sistem ketika menemukan bahwa data telah diubah.

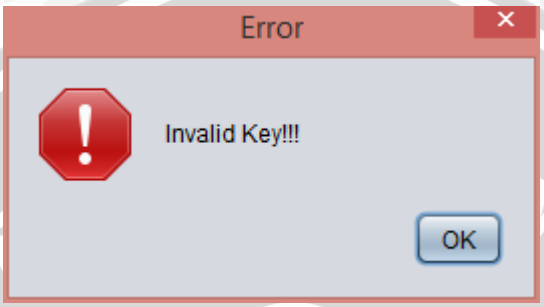


Gambar 5.2 verifikasi kunci gagal

Gambar 5.3 verifikasi *file* gagal

5.1.3.3 *Authentication*

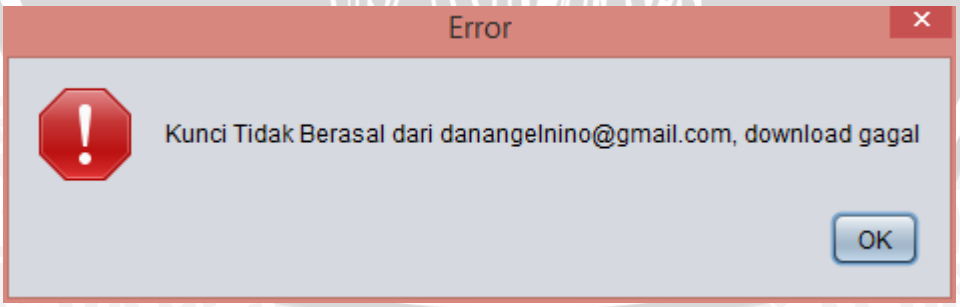
Pada pengujian aspek keamanan *authentication,* skenario pengujian yang dilakukan adalah ketika pengirim mengirimkan *file* kepada penerima dengan memanfaatkan mekanisme pertukaran kunci dan pengirim menggunakan kunci publik penerima untuk melakukan enkripsi kunci *file*. Penerima yang tidak diijikan melakukan dekripsi kunci tidak akan bisa membuka *file*, maka sistem akan menampilkan pesan *error*. Gambar 5.4 menampilkan pesan error bahwa kunci tidak valid atau proses otentikasi gagal.



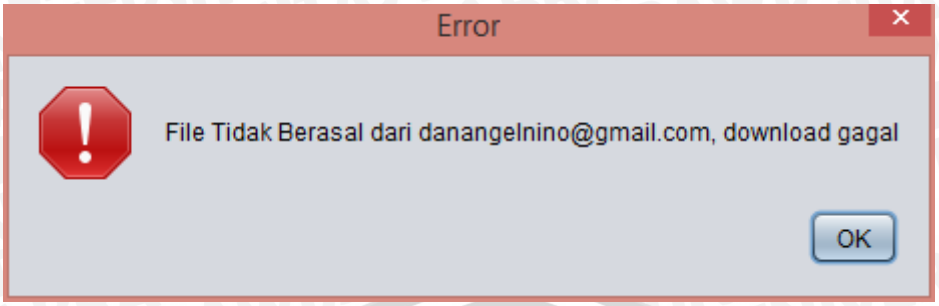
Gambar 5.4 kunci tidak valid

5.1.3.4 *Non-Repudiation*

Pada pengujian aspek keamanan *non-repudiation,* dilakukan untuk menguji bahwa tidak ada penyangkalan terhadap data nantinya. Skenario yang dilakukan adalah dengan mengganti kunci publik *signature* atau menggunakan kunci publik *signature* yang bukan milik pengirim. Pengujian ini diibaratkan bahwa ada pengguna A yang menggunakan komputer milik pengguna B dan pengguna B mengirimkan *file* ke pengguna C dengan berpura-pura menjadi pengguna A. sistem akan melakukan validasi bahwa *file* bukan berasal dari pengguna A dikarenakan *signature* yang ditangkap oleh sistem berbeda. Respon yang dihasilkan oleh sistem akan menunjukkan bahwa data bukan berasal dari pengguna aslinya. Gambar 5.5 dan gambar 5.6 menampilkan respon sistem ketika menemukan bahwa data telah diubah.



Gambar 5.5 verifikasi kunci gagal

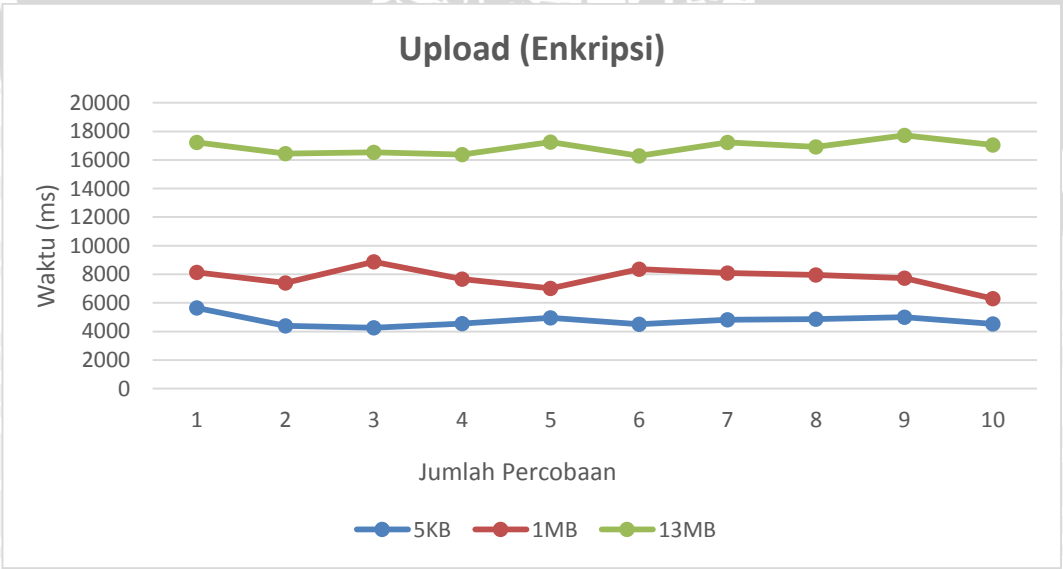


Gambar 5.6 verifikasi *file* gagal

5.2 Analisis

Pada penelitian ini, analisis aplikasi sistem pengamanan data diuji dengan melakukan validasi terhadap fitur-fitur yang dibutuhkan untuk memastikan fitur tersebut berjalan dengan normal dan dapat memenuhi kebutuhan yang telah dirancang sebelumnya. Selain itu, sistem juga diuji terhadap kelayakan terhadap aspek keamanan jaringan. Pada pengujian aspek keamanan jaringan sistem dapat berjalan dengan baik dan sistem juga memberikan respon ketika *file* atau kunci dirubah atau bukan *file* dan kunci yang sebenarnya.

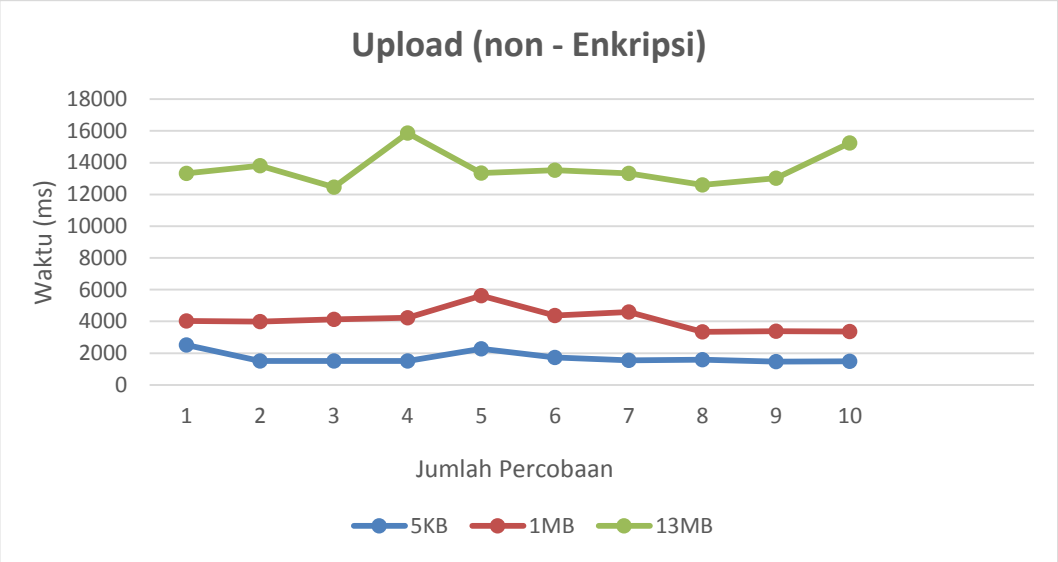
Pada pengujian performansi, pengujian yang dilakukan adalah pada saat sistem melakukan proses *upload* data. Pengujian sistem yang dilakukan adalah dengan melakukan perbandingan terhadap sistem dengan menggunakan enkripsi dan tanpa menggunakan enkripsi yang masing-masing dilakukan sebanyak 10 kali pada tiap-tiap ukuran *file* yang berbeda. Hasil pengujian performansi sistem dengan melakukan enkripsi tersebut dapat dilihat pada gambar 5.7. Hasil pengujian performansi sistem tanpa melakukan enkripsi tersebut dapat dilihat pada gambar 5.8.



Gambar 5.7 Perbandingan waktu *upload* enkripsi *file*

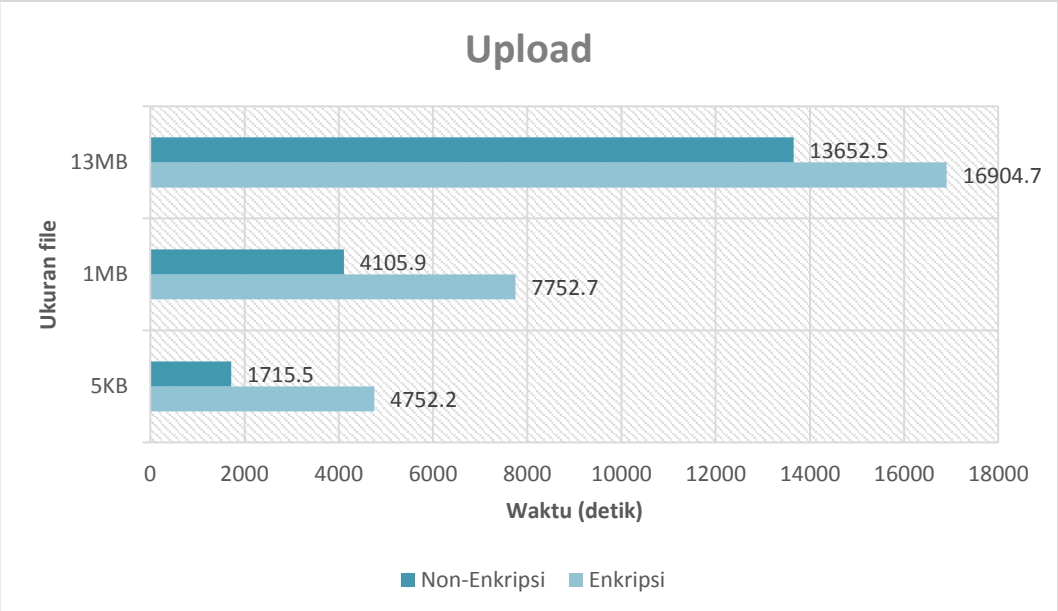
Berdasarkan grafik pada gambar 5.7 pada proses *upload file* dengan menggunakan proses enkripsi keseluruhan yang telah diuji pada tiga ukuran *file*

yang berbeda-beda. Dalam 10 kali percobaan pada ketiga *file* menunjukkan waktu yang cukup stabil. *File* dengan ukuran hampir 13MB hanya membutuhkan waktu paling lama sekitar 17 detik.



Gambar 5.8 Perbandingan waktu *upload* tanpa enkripsi *file*

Berdasarkan grafik pada gambar 5.8 pada proses *upload file* tanpa menggunakan proses enkripsi keseluruhan yang telah diuji pada tiga ukuran *file* yang berbeda-beda. Pada proses *upload* tanpa menggunakan enkripsi sistem mampu melakukan pengiriman data paling lama 16 detik pada *file* yang berukuran 13MB.



Gambar 5.9 perbandingan waktu *upload* enkripsi dan *non*-enkripsi

Berdasarkan grafik pada gambar 5.9 pada proses *upload file* secara keseluruhan yang telah diuji dengan menggunakan proses enkripsi dan *non*-enkripsi. Dari hasil pengujian rata-rata yang telah dilakukan dengan menggunakan tiga *file* yang berbeda ukuran didapatkan bahwa proses enkripsi masih menunjukkan waktu yang cukup stabil. Dari kesemua proses *upload* dengan ukuran *file* yang berbeda-beda didapatkan perbedaan tiap-tiap *file* memiliki perbedaan waktu selama 3 detik.

BAB 6 PENUTUP

Bab ini memuat kesimpulan dari penelitian yang telah dilakukan serta saran untuk pengembangan lebih lanjut.

6.1 Kesimpulan

Dari hasil implementasi, pengujian, dan analisis dari penerapan aplikasi sistem pengamanan data menggunakan algoritma simetrik dan algoritma asimetrik, dapat disimpulkan bahwa:

1. Untuk mencegah adanya *loss of privacy & theft of information* pada media penyimpanan dirancang sebuah sistem pengamanan data menggunakan enkripsi. Sistem bekerja sebelum data dikirim ke media penyimpanan *cloud,* sehingga data yang dimiliki aman dari ancaman *theft of information* dan *loss of privacy* ketika data berada pada media penyimpanan *cloud*. Algoritma yang digunakan dalam melakukan enkripsi terhadap data adalah algoritma simetrik AES. AES berfungsi sebagai algoritma yang berguna untuk merubah *file* menjadi sebuah *cipher file* sehingga file sulit dibaca tanpa menggunakan kunci AES yang sama.
2. Dalam melakukan berbagi*file* antar pengguna, aplikasi sistem pengamanan data memanfaatkan kinerja algoritma asimetrik RSA dengan melakukan pertukaran kunci publik pengguna A yang akan digunakan untuk mengenkripsi kunci simetrik oleh pengguna B dengan menggunakan kunci privat mikik B. pertukaran kunci simetrik ini digunakan untuk membuka *file* yang dibagikan oleh pengguna A.
3. Berdasarkan hasil pengujian performansi sistem dengan melakukan *upload* file dengan menggunakan enkripsi dan non-enkripsi didapatkan hasil dengan perbedaan waktu selama 3 detik yang diuji menggunakan 3 file dengan ukuran yang berbeda.
4. Berdasarkan hasil pengujian aspek keamanan jaringan yang diantaranya: *confidentiality*, *integrity*, *authentication* dan *non-repudiation*. Sistem dapat berjalan dengan baik dan sistem juga memberikan respon ketika file atau kunci dirubah atau bukan file dan kunci yang sebenarnya.

6.2 Saran

Saran yang dapat disampaikan penulis untuk pengembangan aplikasi sistem pengamanan data adalah:

1. Perlu dilakukan penelitian lebih lanjut mengembangkan sistem pengamanan data dengan cara melakukan *splitting file* terlebih dahulu sebelum melakukan enkripsi.

2. Perlu dilakukan penelitian lebih lanjut untuk mengembangkan sistem pengamanan data pada *cloud storage* dengan menggunakan algoritma yang berbeda.

DAFTAR PUSTAKA

Adrianus, E., 2010. Theft of Information di dalam Teknologi Cloud Computing dan Teknik Mengamankan Data sebagai Pencegahan Pencurian Data, Bandung: s.n.

Anon., 2015. What is Maven? | The Javabook. [Online] Available at: http://www.thejavabook.com/2015/09/what-is-maven/ [Accessed 13 Desember 2015].

Assagaf, N., 2012. Layanan Cloud Storage | Cloud Indonesia. [Online] Available at: http://www.cloudindonesia.or.id/layanan-cloud-storage.html [Accessed 11 Desember 2015].

Avestro, J., 2007. JENI Pengenalan Pemrograman 1. 1.2 ed. s.l.:s.n.

Dafid, 2006. Kriptografi Kunci Simetris Dengan Menggunakan Algoritma Crypton. Jurnal Ilmiah STMIK Gi MPD, 2(3), pp. 20-17.

Dropbox, 2015. Dropbox. [Online] Available at: https://www.dropbox.com/about [Accessed 14 December 2015].

IBM, 2015. IBM Knowledge Center. [Online] Available at: https://www-01.ibm.com/support/knowledgecenter/SSLTBW_2.1.0/com.ibm.zos.v2r1.csfb300/csfb3za212.htm%23wq21 [Accessed 14 12 2015].

Kaur, M. & Singh, R., 2013. Implementing Encryption Algorithms to Enhance Data Security of Cloud in Cloud Computing. International Journal of Computer Applications, 70(18), pp. 17-21.

KetuWare, 2004. Symmetric vs Asymmetric Encryption. KetuFile White Paper, pp. 1-7.

Kromodimoeljo, S., 2009. Teori dan Aplikasi Kriptografi. s.l.:SPK IT Consulting.

M.Barukab, O., Khan, A. I., Shaik, M. S. & Murthy, M. R., 2012. Secure Communication using Symmetric and Asymmetric Cryptographic Techniques. I.J. Information Engineering and Electronic Business, pp. 36-42.

Pratiwi, O. N., 2011. Analisis Keamanan Aplikasi Penyimpanan Data Pada Sistem Cloud Computing. e-Indonesia Initiative 2011, pp. 127-139.

Rahajoeningroem, T. & Aria, M., 2011. Studi dan Implementasi Algoritma RSA Untuk Pengamanan Data Transkip Akademik Mahasiswa. Majalah Ilmiah UNIKOM, 8(1), pp. 77-90.

Ramadhan, Z., 2011. Aspek Keamanan Pada Cloud Computing. Jurnal Ilmiah Abdi Ilmu, 4(2), pp. 646-651.

Rouse, M., 2011. What is Dropbox? | Techtarget. [Online] Available at: http://searchmobilecomputing.techtarget.com/definition/Dropbox [Accessed 14 Desember 2015].

Schaefer, E., n.d. An introduction to cryptography and cryptanalysis, Santa Clara University: s.n.