1. Pendahuluan

1.1 Latar Belakang

Seiring semakin populernya bahasa pemrograman java maka semakin banyak pula bermunculan file-file executable JAR di internet. File executable JAR adalah file executable dari *class-class* java. Salah satu bentuk kegunaan dari file executable JAR ini adalah untuk aplikasi di *mobile device*. File-file tersebut dapat diakses dengan mudah dari web maupun dari wap. Karena kemudahan akses tersebutlah maka banyak file executable JAR yang telah disusupi virus ataupun telah diedit oleh orang yang tidak bertanggung jawab.

Salah satu bentuk perlindungan dari file executable JAR adalah dengan *digital signature*. Dengan menandatangani secara digital file executable JAR maka aspek keamanan berupa Integritas data, otentikasi dan nir penyangkalan dapat disediakan. Integritas data akan membuat pendownload yakin bahwa file yang didownload dari internet benar-benar asli, tidak ada pengurangan atau penambahan dari virus. Otentikasi akan membuat pendownload merasa yakin bahwa file yang didownload benar-benar dari pengupload yang benar. Sedangakan nir penyangkalan akan mencegah salah satu pihak membantah telah melakukan proses upload ataupun download.

1.2 Rumusan Masalah

Aplikasi ini digunakan untuk memberikan *digital signature* terhadap file executable JAR. Permasalahan yang timbul diantaranya :

1. Bagaimana sistem memverifikasi file executable JAR yang didownload dari sistem adalah file yang asli tidak ada penambahan sintaks virus
2. Menguji kebenaran dari *digital signature* dalam memverifikasi file melalui beberapa studi kasus pengujian.
3. Mengetahui waktu *generate key, sign dan verify* tercepat untuk beberapa bit kunci berbeda
4. Bagaimana sistem mengirimkan *public key* dan *mac key* yang digunakan dalam *digital signature.*

Permasalahan dalam tugas akhir ini memiliki batasan sebagai berikut

1. Uploader pihak pengembang sistem.
2. Kanal yang digunakan untuk pentransmisian *mac key* diasumsikan aman
3. Pengecekan digital signature hanya dapat dilakukan melalui koneksi internet, atau terhubung ke server tidak dapat dilakukan secara *offline*

1.3 Tujuan

Tujuan yang hendak dicapai dalam TA ini adalah membangun suatu aplikasi yang mampu untuk:

1. Mengetahui keaslian dari file yang berada di server.
2. Menguji kebenaran dari *digital signature* dalam memverifikasi file melalui beberapa studi kasus pengujian.
3. Mengetahui waktu *generate key, sign dan verify* tercepat untuk beberapa bit kunci berbeda
4. Menangani cara pengiriman *public key* dan *mac key* yang digunakan dalam *digital signature*

1.4 Metodologi Penyelesaian Masalah

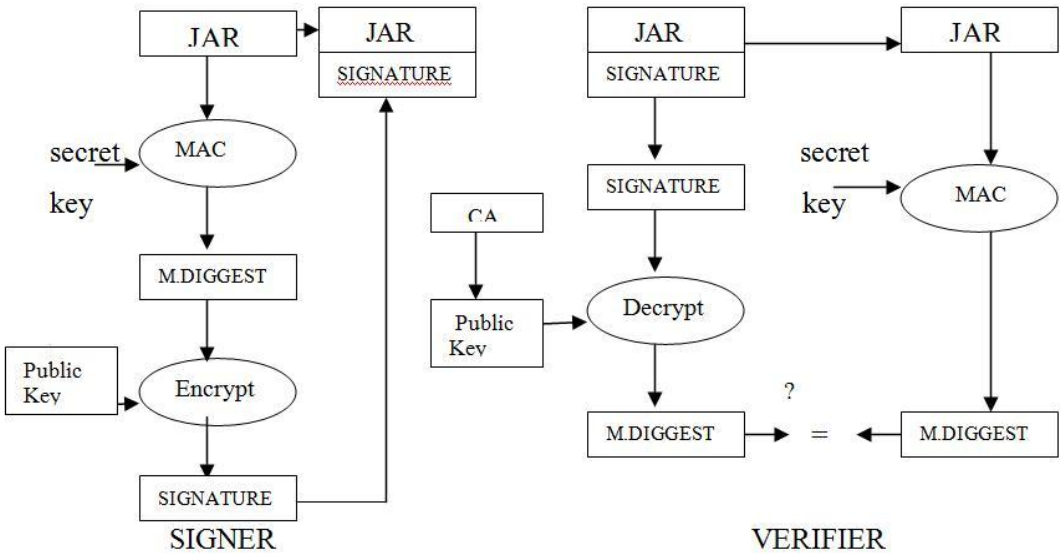
Aplikasi ini dibangun dengan metode analisis dan desain terstruktur dengan tahapan :

1. Tahap Analisis dan Definisi Kebutuhan

   Suatu hal yang dibutuhkan oleh file executable JAR yang didownload dari internet adalah keyakinan pendownload bahwa file tersebut adalah benar-benars asli, baik asli dari pembuat aplikasi maupun asli tidak ada penambahan virus di dalamnya. Untuk itulah dibutuhkan suatu cara untuk menanganinya. Hipotesis dari penulis untuk menangani masalah tersebut adalah dengan *digital signature*.

2. Tahap Desain Sistem dan Perangkat Lunak

Sistem yang akan dibangun digambarkan pada bagan berikut ini :



Gambar 1. 1 Bagan rancangan sistem digital signature

Pada proses *signer* atau pemberian tanda tangan, file executable JAR akan dihitung nilai hashnya dengan menggunakan kunci privat dengan algoritma MAC. Pendistribusian kunci privat menggunakan fasilitas email. *Message digest* yang dihasilkan akan di *encrypt* kembali menggunakan algoritma RSA, sehingga dihasilkan *signature* dari file executable JAR tersebut. Dan untuk proses *verifier* akan dibandingkan kesamaan dari *message digest* file executable JARnya dengan *message digest* signaturenya. *Message digest* JAR di dapat dari fungsi hash yang menggunakan sebuah kunci privat dengan algoritma MAC. Sedangkan message digest signature didapat dari *decrypt* signature menggunakan *public key* dengan algoritma RSA. *Private key* dan *publik key* yang digunakan dalam algoritma RSA didapat dari CA. Jika kedua message digest yang dihasilkan sama maka file executable JAR tersebut masih asli.

Rancangan pada sistem digital signature ini dapat digunakan untuk mengenerate nilai signature tipe file apapun namun pada tugas akhir ini digunakan studi kasus hanya untuk file executable JAR. Hal ini mengingat tujuan dari tugas akhir ini untuk melindungi file executable JAR dengan metode rancangan seperti gambar 1.1.

3. Tahap Implementasi

   Selama tahap ini sistem akan diimplementasikan dan diuji berdasarkan fungsi-fungsinya. Sistem akan diimplementasikan ke dalam dua aplikasi yang berbeda yaitu aplikasi berbasis web yang akan dijalankan dalam PC dan aplikasi berbasis wap yang akan dijalankan dalam *mobile device*.

   Perbedaan dari kedua aplikasi ini adalah pada fungsionalitas upload. Fungsi upload hanya ditangani oleh aplikasi berbasis web.

4. Tahap Integrasi dan Pengujian Sistem

   Tahap ini unit-unit program diintegrasikan dan diuji sebagai suatu sistem *digital signature* untuk file executable JAR secara keseluruhan, untuk memastikan bahwa sistem sudah sesuai dengan spesifikasinya. Cara pengujian yang digunakan antara lain

   a. Menambahkan isi dari file JAR (tes jika file JAR disusupi virus)
      Hipotesis hasil : Sistem akan menampilkan pesan *signature failed*

   b. Mengurangkan isi dari file JAR (tes jika file JAR corrupt)
      Hipotesis hasil : Sistem akan menampilkan pesan *signature failed*

   c. Mengganti isi dari file JAR

      Hipotesis hasil : Sistem akan menampilkan pesan *signature failed*

   d. Menambahkan signature pada server
      Hipotesis hasil : Sistem akan menampilkan pesan *signature failed*

   e. Mengurangkan signature pada server
      Hipotesis hasil : Sistem akan menampilkan pesan *signature failed*

   f. Menghapus signature pada server
      Hipotesis hasil : Sistem akan menampilkan pesan *signature failed*

   g. Mengganti signature pada server

      Hipotesis hasil : Sistem akan menampilkan pesan *signature failed*

h. Memberikan kunci publik palsu ke downloader.
   Hipotesis hasil : Sistem akan menampilkan pesan *signature failed*

i. Tidak melakukan manipulasi terhadap file JAR (Menguji kekuatan digital signature untuk penanda bahwa file JAR tersebut adalah asli dan tidak dimanipulasi)
   Hipotesis hasil : Sistem akan menampilkan pesan *signature success*

j. Membandingkan waktu generate key terhadap kunci dengan panjang digit berbeda.
   Hipotesis hasil : Kunci dengan digit lebih pendek akan lebih cepat.

k. Membandingkan waktu sign terhadap kunci dengan panjang digit berbeda.
   Hipotesis hasil : Kunci dengan digit lebih pendek akan lebih cepat.

l. Membandingkan waktu pemverifikasian terhadap kunci dengan panjang digit berbeda.
   Hipotesis hasil : Kunci dengan digit lebih pendek akan lebih cepat.