RANCANG BANGUN APLIKASI UNTUK PENYISIPAN *TEXT* DAN *FILE* KE DALAM *IMAGE* DAN *AUDIO FILE* DENGAN METODE *LEAST SIGNIFICANT BIT* (LSB)

LAPORAN TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana
Teknik Pada Jurusan Teknik Informatika

Oleh :

ISMAIL MARZUKI

10751000195

FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI SULTAN SYARIF KASIM RIAU
PEKANBARU

2011

APPLICATION DESIGN TO EMBED TEXT AND FILE INTO IMAGE AND AUDIO FILE WITH LEAST SIGNIFICANT BIT (LSB) METHOD

ISMAIL MARZUKI

10751000195

Final Exam Date: July 8th, 2011

Graduation Ceremony Period: October 2011

Information Engineering Department

Faculty of Sciences and Technology

State Islamic University of Sultan Syarif Kasim Riau

ABSTRACT

Steganography is a technique to secure the communications of data or message. Message is secured by embedding bits message into bits carrier file. A method to embed bits message into bits carrier is Least Significant Bit (LSB) Method that embeds bits message on small bits value of carrier file. Generally, this report discussed about designing of steganography applications that modified 2 bits of carrier file using LSB Method. Bitmap file (.bmp) and wave file (*.wav) as carrier files recommendation for this application. There were two types message to be embedded to the carrier as text and file. This application was developed with Object Oriented Programming approach (Java). As result, this application was successfully to embed and retrieve messages as text and files without modifying carrier capacity before or after message embedded to the carrier, avarage quality* 50.316 dB *(bitmap) and* 53.507 dB *(wave), but it was still unrobust from a visual attacking.*

Key words: *Carrier file, Embedding, Least Significant Bit (LSB), Object Oriented Programming, Retrieving, Steganography, Stegofile.*

RANCANG BANGUN APLIKASI UNTUK PENYISIPAN *TEXT* DAN *FILE* KE DALAM *IMAGE* DAN *AUDIO FILE* DENGAN METODE *LEAST SIGNIFICANT BIT* (LSB)

ISMAIL MARZUKI

10751000195

Tanggal Sidang: 8 Juli 2011

Periode Wisuda: Oktober 2011

Jurusan Teknik Informatika

Fakultas Sains dan Teknologi

Universitas Islam Negeri Sultan Syarif Kasim Riau

ABSTRAK

Steganografi merupakan sebuah teknik untuk mengamankan komunikasi suatu data. Pesan diamankan dengan cara menyisipkan bit-bit pesan ke dalam bit-bit *carrier file.* Salah satu metode untuk menyisipan pesan pada *carrier file* adalah metode *Least Significant Bit* yang menyisipkan bit-bit pesan pada bit-bit terkanan *carrier file*. Laporan penelitian ini membahas tentang rancang bangun aplikasi steganografi dengan metode LSB yang memodifikasi 2 bit LSB *carrier file* berupa *file* gambar berformat bitmap dan *file* suara berformat wave dengan pesan yang akan disisipkan berupa teks dan *file.* Pengujian yang dilakukan meliputi pengujian pada aspek kecepatan, kapasitas, ketahanan, dan kualitas dari *stegofile* yang dihasilkan. Rancang bangun aplikasi dikembangkan menggunakan bahasa pemrograman berorientasi objek (Java). Hasilnya adalah penyisipan *(embedding)* dan pengambilan *(retrieving)* pesan berupa teks maupun *file* berhasil dilakukan tanpa mengubah kapasitas *file* sebelum maupun setelah pesan disisipkan. Rata-rata kualitas *stegofile* yang dihasilkan sebesar 50.316 dB pada bitmap dan 53.507 dB pada wave, namun masih cenderung sangat rentan terhadap penyerangan visual *(unrobust)*.

Kata kunci: *Carrier file, Embedding, Least Significant Bit* (LSB), Pemrograman Berorientasi Objek*, Retrieving,* Steganografi, *Stegofile.*

DAFTAR ISI

DAFTAR ISTILAH

Carrier File	:	Calon media pembawa pesan
Cryptography	:	Teknik tentang penyamaran data yang bertujuan untuk keamanan data
Embedding	:	Proses penyisipan pesan ke dalam medianya
Least Significant Bit	:	Merupakan bit-bit terkanan yang memiliki nilai paling rendah atau dapat dikatakan sebagai metode steganografi yang bekerja dengan cara memodifikasi bit-bit terakhir media pembawa pesan dengan bit-bit pesannya yang memiliki kelebihan, seperti mudah dan cepat dalam hal algoritma, kapasitas pesan yang disisipkan relatif besar tanpa harus menurangi kualitas secara signifikan
Retrieving	:	Proses pengambilan kembali (ekstraksi) pesan dari media pembawanya
Steganography	:	Sebuah teknik dan seni tentang cara penyembunyian pesan rahasia pada sebuah media
Stegofile	:	Sebuah *file* yang telah disisipi atau membawa pesan rahasia
Watermarking	:	Cabang dari keamanan data yang bertujuan untuk pencatatan data, seperti lisensi, *copyright,* dan lain sebagainya

BAB I

PENDAHULUAN

1.1. Latar Belakang

Steganografi merupakan sebuah ilmu, teknik, dan seni tentang penyembunyian atau penyisipan suatu informasi atau pesan pada suatu media atau wadah penyisipan *(carrier file)*. Hal ini erat kaitannya dengan keamanan *(security)* data. Dalam teknologi informasi dan komunikasi, steganografi merupakan suatu teknik dan seni penyembunyian yang memanfaatkan format-format digital, artinya suatu informasi digital disembunyikan di balik informasi digital lain, sehingga informasi digital yang sesungguhnya tidak terlihat.

Secara teori, semua *digital file* secara umum yang ada di dalam komputer dapat digunakan sebagai media penyembunyian *(carrier file)* atau yang disembunyikan, seperti file gambar, audio, teks, video dan lain sebagainya. File-file tersebut memiliki bit-bit data redundan sebagai karakteristik sebuah file digital yang dapat dimodifikasi *(Suyono, 2004)*.

Berbagai penelitian terhadap steganografi ini sebenarnya telah dilakukan dan tetap dikembangkan oleh para peneliti dengan menggunakan beragam metode steganografi. Metode yang paling banyak digunakan dan ditemui adalah metode *Least Significant Bit (LSB)*. Hal ini disebabkan karena LSB merupakan metode yang memiliki kelebihan-kelebihan seperti, mudah dan cepat secara algoritma, semua jenis file digital dapat dijadikan *sampling* dan pesan yang bisa disembunyikan adalah pesan-pesan yang ukuranya relatif besar yang dapat disisipkan pada semua jenis file digital. Namun, kebanyakan penelitian yang secara umum dilakukan masih terpisah atau terbatas pada salah satu jenis file digital saja, bahkan hanya pada satu jenis digital format file saja, meskipun penelitian pada jenis file dan format file digital lainnya tetap dilakukan. Hal ini

tentu akan menyusahkan *user* yang ternyata ingin menyisipkan informasi atau pesan dengan file atau format file yang berbeda.

Beberapa contoh jurnal internasional dan penelitian yang dipublikasikan pada tahun 2009 - 2011 ini diantaranya adalah: Kriti Saroha dan Pradeep Kumar Singh (2010) meneliti steganografi khusus file audio dengan menggunakan LSB, Sujay Narayana dan Gaurav Prasad (2010) meneliti steganografi dengan LSB pada *file image* saja, Pradeep Kumar Singh dan R.K.Aggrawal (2010) menggunakan LSB pada *file image* ke audio, Dian Dwi Hapsari dan Lintang Yuniar Banowosari (2009) menggunakan LSB pada gambar, Saurabh Singh dan Gaurav Agarwal (2010) menggunakan LSB pada video, Rahul Rishi, dkk (2011) steganografi pada gambar dengan menggunakan metode *Mode and Multiple Technique* yang masih dikembangkan dari LSB, dan masih banyak jurnal-jurnal dan penelitian lainnya yang masih menggunakan LSB dalam penelitian steganografi.

Penjelasan yang diterangkan di atas merupakan hal yang melatarbelakangi penulis melakukan penelitian tugas akhir tentang steganografi ini. Dengan kata lain, penulis akan membuat sebuah rancang bangun aplikasi steganografi yang dapat melakukan penyembunyian atau penyisipan dan tentu pengambilan kembali suatu informasi yang tidak hanya dapat menyembunyikan teks atau file saja, tapi keduanya. Dan media penyisipan yang digunakan tidak hanya pada file gambar saja, tapi juga audio yang dapat dijadikan alternatif pilihan jika teks atau file yang akan disembunyikan cukup besar dengan menggunakan metode *Least Significant Bit (LSB)* dengan harapan dapat memberikan kemudahan bagi *user*.

1.2. Rumusan Masalah

Berdasarkan penjelasan yang telah dijelaskan dibagian latar belakang di atas, maka dapat ditarik sebuah rumusan masalah yang akan dijelaskan lebih lanjut pada laporan tugas akhir ini, yaitu bagaimana menyembunyikan dan mengambil kembali suatu informasi berformat digital, yaitu teks dan file dengan menggunakan metode *least significant bit* (LSB) pada file-file digital berupa file

gambar dan audio sehingga informasi yang disembunyikan tersebut tidak diketahui atau tidak terlihat.

1.3. Batasan Masalah

Agar tidak terjadi kesalahan persepsi dalam laporan tugas akhir ini, maka berikut dijelaskan beberapa hal yang menjadi batasan masalah laporan ini:

1. Media penampung yang akan digunakan hanya file-file digital yang bersifat *uncompressed*, khususnya file gambar berformat bitmap (*.bmp), dan file audio berformat wave (*.wav). Hal ini disebabkan karena kedua format file ini memililiki *raw data* dasar (belum mengalami perubahan *raw data* seperti, pemampatan file/ kompresi).
2. Bit-bit *carrier file* yang akan dimodifikasi adalah hanya pada 2 bit LSB-nya saja.
3. Penelitian tidak menambahkan kunci keamanan *(key)* pada aplikasi sebagai pengaman pesan. Karena faktor penyisipan pesan pada 2 bit LSB *carrier file* merupakan fokus utama penelitian. Namun, tidak mengesampingkan faktor sekuritas pesan.

1.4. Tujuan Penelitian

Tujuan yang ingin dicapai pada penelitian terhadap kasus yang dibahas dalam laporan ini, yaitu: untuk merancang sebuah rancang bangun aplikasi steganografi yang memiliki kemampuan dalam menyembunyikan *(embed)* dan mengambil kembali *(retrieve)* suatu informasi digital berupa *text* dan *file* ke dalam file gambar dan suara dengan menggunakan metode *Least Significant Bit* (LSB).

1.5. Sistematika Penulisan

Berikut merupakan rencana susunan sistematika penulisan laporan tugas akhir yang akan dibuat. Sistematika penulisan laporan tugas akhir ini meliputi:

1. **Bab I Pendahuluan**

   Bab I ini merupakan bagian yang akan menguraikan hal-hal seperti; latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian, dan sistematika penulisan laporan tugas akhir

2. **Bab II Landasan Teori**

   Bab ini berisi tentang teori-teori tentang steganografi dan metode *least significant bit* (LSB) serta menjelaskan tentang proses kerja metode LSB dalam menyembunyikan informasi digital pada file digital lainnya.

3. **Bab III Metodologi Penelitian**

   Bab ini berisi tentang cara-cara atau hal-hal yang dilakukan dalam menyelesaikan kasus tugas akhir ini.

4. **Bab IV Analisa dan Perancangan**

   Bab ini berisi tentang analisa dari penelitian yang dilakukan dalam tugas akhir ini sekaligus menerangkan perancangan rancang bangun aplikasi steganografi yang dibangun.

5. **Bab V Implementasi dan Pengujian**

   Bab ini berisi tentang langkah-langkah pembangunan rancang bangun aplikasi steganografi dan menguji hasil dari rancangan yang telah dibangun.

6. **Bab VI Penutup**

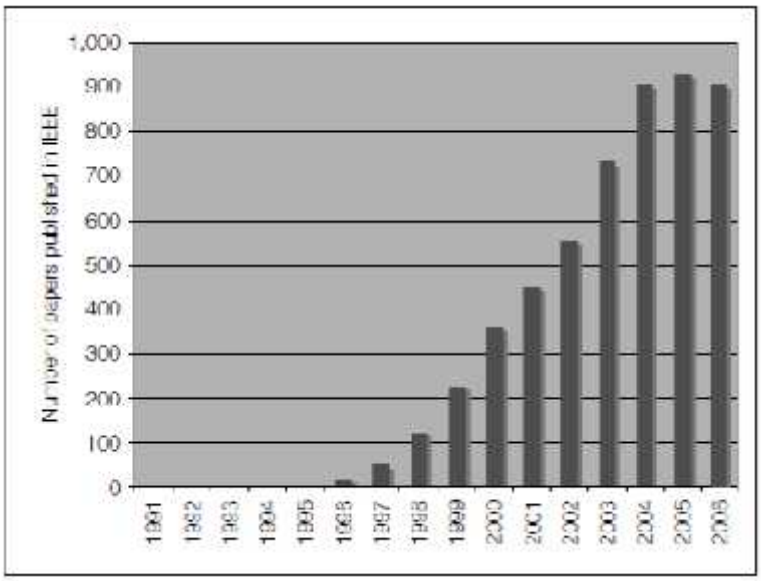
   Bab ini berisi kesimpulan dan saran mengenai hasil analisa, perancangan, hasil implementasi dan hasil pengujian yang telah dilakukan terhadap rancang bangun aplikasi steganografi yang telah dibangun.

BAB II

LANDASAN TEORI

2.1. Pengenalan Steganografi

Steganografi pada dasarnya merupakan sebuah teknik dan seni penyembunyian suatu informasi pada suatu informasi lainnya dengan tujuan untuk merahasiakan informasi yang disembunyikan tersebut. Steganografi yang merupakan sebuah kegiatan yang diadopsi dari kegiatan bangsa Yunani dulu dikembangkan secara ilmu pengetahuan dan teknologi dimana pengembangan ini dapat dibuktikan dengan banyaknya penelitian dibidang steganografi yang terus dipublikasikan setiap tahunnya. Hasil publikasi ini dapat dilihat pada gambar 2.1 berikut.



Gambar 2.1. *Annual Number* Publikasi Penelitian tentang *steganography* oleh IEEE *(Sumber: Pradeep dkk, 2010)*

2.1.1. Pengertian Steganografi

Steganografi (dalam bahasa Yunani disebut *Steganos,* "tersembunyi/ disembunyikan", dan *grapphein,* "menulis") adalah sebuah seni dan ilmu *(science)* tentang berkomunikasi dengan cara menyembunyikan eksistensi informasi dari komunikasi tersebut. Steganografi menyembunyikan eksistensi suatu pesan dengan cara menanamkan/ melekatkan *(embedding)* pesan ke dalam sebuah media yang disebut *carrier file (Sujay dkk, 2010).*

Pada prinsipnya, *steganogrphy* memiliki fungsi yang sama dengan *cryptographic* dan *watermarking,* yaitu sama-sama berperan dalam keamanan suatu data, tapi ketiganya memiliki maksud yang berbeda. Pada c*ryptographic,* pesan dikodekan sedemikian rupa sehingga orang lain tidak mengerti atau mengenali pesan tersebut *(Alfebra dkk, 2008), watermarking* merupakan teknik peluasan informasi dan menjadikannya suatu atribut dari sebuah media digital yang dapat berupa *copyright*, kepemilikan, atau lisensi *(Suyono, 2004),* sementara *steganography* merupakan teknik, seni, dan ilmu berkomunikasi dengan cara menyembunyikan informasi pada informasi yang lainnya sehingga orang lain tidak mengetahui keberadaan informasi dari komunikasi tersebut *(Kriti dkk, 2010).*

Beberapa literatur menjelaskan bahwa istilah steganografi merupakan pengembangan cara berkomunikasi bangsa Yunani dulu atau zaman Romawi Kuno yang menyembunyikan sebuah pesan rahasia dengan menggunakan tinta yang tidak terlihat. Tinta tersebut dibuat dari campuran air sari buah jeruk, *urine*, atau susu *(Alfebra dkk, 2008).* Prinsip dasarnya adalah bagaimana cara merahasiakan komunikasi antara orang pertama dengan orang kedua yang mengandung suatu informasi rahasia dalam komunikasi tersebut, sehingga orang ketiga tidak tahu eksistensi suatu informasi yang ada dan tidak memiliki kecurigaan terhadap komunikasi yang dirahasiakan tersebut *(Neil dkk, 1998).*

Secara teori, semua jenis *file* atau format digital yang mengandung sebuah informasi dapat dijadikan media penyisipan, seperti *text file, image, audio, video* dan lain sebagainya. Hal ini disebabkan jenis-jenis *file* atau *format* digital tersebut memiliki bit-bit data redundan yang dapat dimodifikasi sebagai karakteristik dari

sebuah *digital file (Suyono, 2004).* Informasi digital akan ditanamkan atau dilekatkan pada media digital lain yang berfungsi sebagai media penampung informasi dengan cara memanipulasi bit-bit yang dimiliki oleh media penampung dengan informasi yang ingin disembunyikan, sehingga terjadi perubahan bit-bit pada media penampungnya, namun tidak terjadi perbedaan signifikan terhadap media aslinya dan perubahan bit tersebut tidak akan terdeteksi oleh insting manusia.

2.1.2. Konsep Steganografi

Secara umum steganongrafi mengindikasikan adanya dua kebutuhan inti pada sebuah steganografi, yaitu pesan atau data digital yang akan disembunyikan dan media penampung atau data digital yang berfungsi untuk menyembunyikan pesan. Agar sebuah apalikasi steganografi dapat menyisipkan pesan ke dalam media penampungnya tentu membutuhkan suatu algoritma yang dapat memodifikasi objek digital menjadi objek digital baru. Algoritma ini disebut dengan algoritma *embedding*. Sementara algoritma yang digunakan untuk mengambil kembali pesan yang telah disisipkan pada medianya disebut dengan algoritma *retrieving.*

Sebuah steganografi memiliki tiga aspek berbeda yang dapat menentukan berhasil-tidaknya atau baik-tidaknya sebuah steganografi dalam melakukan pekerjaannya *(Ermadi dkk, 2004),* yaitu:

1. Kapasitas *(capacity)* yang mengacu pada jumlah atau ukuran informasi yang dapat disembunyikan pada sebuah media penampungnya *(payload capacity)* dan kemampuan media penampung dalam menyembunyikan informasi *(imperceptibility)*,
2. Keamanan *(security)* yang mengacu pada pencegahan *(prevention)* bagi orang biasa yang tidak mampu mendeteksi informasi yang tersembunyi di dalam sebuah media penampung, dan
3. Ketahanan *(robustness)* yang berfungsi untuk modifikasi media stego sehingga dapat bertahan terhadap suatu *attack* yang dapat menghancurkan informasi tersembunyi.

Pada aspek pertama, yaitu kapasitas *(capacity),* memiliki dua hal yang juga dianggap sangat penting dan dapat dijadikan bahan pertimbangan untuk pemilihan suatu metode sebuah steganografi, yaitu: kapasitas penyisipan informasi *(payload capacity)* dan kemampuan menyembunyikan informasi atau *imperceptibility (YANG, 2009).* Kedua hal ini berbanding terbalik satu sama lain dalam menjalankan fungsinya, artinya adalah jika kapasitas suatu informasi ditingkatkan, maka kemampuan untuk menyembunyikan informasi akan berkurang dan begitu sebaliknya. Untuk itu pemilihan metode steganografi dapat disesuaikan dengan kebutuhan, apakah kapasitas penyisipan atau kemampuan penyembunyian informasi yang diutamakan. Ada beberapa penelitian steganografi yang terkait dengan peningkatan *payload capacity* dan *security data,* yaitu:

a. (Habes, 2006), menggunakan penyisipan bit LSB lebih dari satu, namun tidak lebih dari 4 bit yang bermanfaat untuk meningkatkan *payload capacity*, dan
b. (Prasad dkk, 2009), menggabungkan teknik kompresi dan kriptografi untuk meningkatkan *payload capacity* dan *security data*.

2.1.3. Metode-Metode Steganografi

Ada beberapa metode yang dapat digunakan sebagai teknik penyembunyian suatu informasi digital ke dalam informasi digital lainnya *(steganography)*, diantaranya adalah:

1. *Least Significant Bit Insertion* (LSB)

   LSB merupakan sebuah metode yang lazim digunakan oleh para peneliti pada sebuah steganografi. Hal ini disebabkan karena LSB merupakan sebuah metode steganografi yang paling sederhana, cepat, dan mempunyai kapasitas penyisipan suatu informasi digital yang cukup besar. LSB menyisipkan sebuah informasi rahasia pada bit rendah atau bit yang paling kanan dari sebuah data pixel yang menyusun sebuah informasi digital yang menjadi media penampung suatu informasi rahasia.

2. *Masking and Filtering*

   Metode ini biasanya dibatasi pada image 24 bit warna dan *image grayscale*. Beberapa literatur menyatakan bahwa metode ini mirip dengan *watermark*, dimana suatu image diberi tanda *(marking)* untuk menyembunyikan pesan rahasia. Hal ini dapat dilakukan dengan memodifikasi *luminance image* dibeberapa bagiannya. Metode ini memiliki ketahanan *(robustness)* terhadap kompresi, dan *cropping*. Namun, memiliki batasan kapasitas tertentu pada informasi yang akan disembunyikan.

3. *Algorithm and Transformation*

   Metode ini merupakan metode steganografi yang jauh lebih kompleks dari metode-metode sebelumnya, artinya tingkat kesulitan dalam penerapan metode ini lebih tinggi. Untuk menyembunyikan sebuah informasi digital pada media penampungnya dilakukan dengan memanfaatkan *Discerete Cosine Transformation* (DCT) dan *Wavelet Compression*. DCT digunakan pada *file-file* terkompresi, seperti JPEG. Metode ini terjadi di domain frekuensi dari sebuah *file* digital, bukan pada domain spasial.

4. *Spread Spectrum Methode*

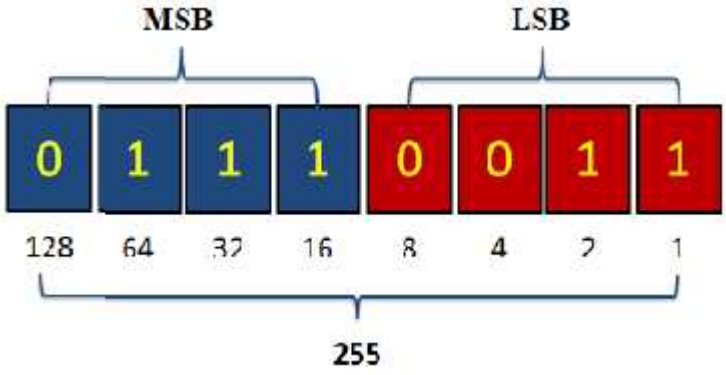
   Teknik metode ini dalam menyembuyikan suatu informai digital adalah dengan mengkodekan informasi rahasia dan disebarkan ke setiap spektrum frekuensi yang memungkinkan. Namun, metode ini masih mudah diserang, yaitu dengan cara penghancuran atau pengrusakan dari kompresi dan proses image (gambar).

2.2. Konsep Kerja LSB *(Least Significant Bit)*

Bilangan biner merupakan dasar dari terciptanya komputer, karena sebenarnya komputer bekerja berdasarkan dua bilangan saja, yaitu 0 dan 1. Kedua bilangan ini sering disebut dengan istilah *bit.* Kemudian bit-bit ini akan terus berangkai dan bersusun membentuk suatu struktur biner yang menjadi sebuah

rangkaian informasi. Bentuk yang paling umum digambarkan untuk serangkaian bit berjumlah 8-bit atau sering disebut dengan istilah 1 byte *(Alfebra dkk, 2008).*

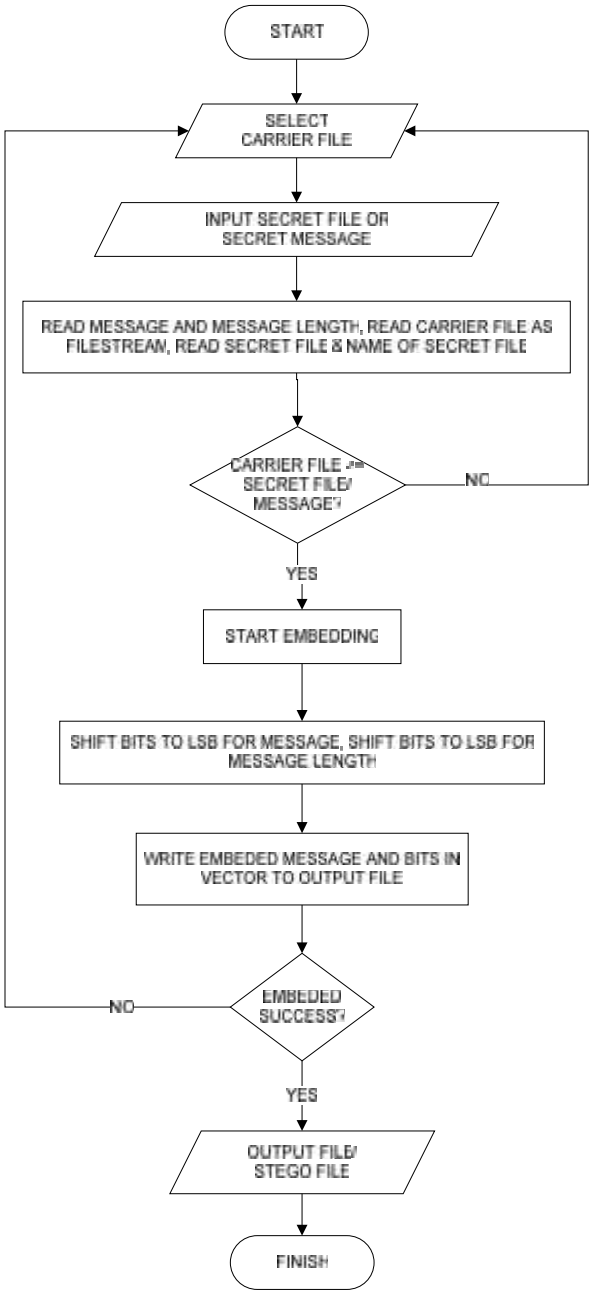
Pada sebuah rangkaian informasi terdapat penggolongan-penggolongan bit berdasarkan urutan dan pengaruhnya dalam byte. Secara garis besar, dalam rangkaian informasi terdapat 2 golongan bit, yaitu *Most Significant Bit* (MSB) dan *Least Significan Bit* (LSB), seperti yang ditunjukkan pada gambar 2.2. *Most Significant Bit* merupakan reprsentasi 4-bit yang memiliki pengaruh besar pada sebuah rangakain informasi, artinya adalah akan terjadi perubahan yang drastis apabila bit-bit ini dimodifikasi. Sementara *Least Significant Bit* merupakan representasi 4-bit yang paling sedikit memiliki pengaruh apabila bit-bit tersebut dimodifikasi dan tidak akan terjadi perubahan yang drastis, sehingga kemungkinan terjadinya kecurigaan manusia terhadap bit-bit LSB yang dimodifikasi sangat kecil. Dengan demikian, semakin kekanan, bit-bit tersebut semakin kecil pengaruhnya terhadap keutuhan data yang dikandung. Oleh sebab itu, 4-bit terakhir tersebut yang dimodifikasi dan dijadikan tempat pelekatan sebuah informasi digital steganografi.



Gambar 2.2. Representasi Biner *(Sumber: Agus dkk, 2010)*

Teknik LSB dilakukan dengan memodifikasi bit-bit yang tergolong bit-bit LSB pada tiap byte dalam sebuah *file* yang digunakan sebagai *carrier file,* atau dengan kalimat yang lain dengan cara mengganti bit-bit LSB dengan bit-bit informasi yang ingin dilekatkan. Proses penggantian bit ini disebut dengan proses *encoding/ embedding.* Setelah semua bit informasi tersebut menggantikan bit LSB *carrier file* tersebut, maka informasi telah berhasil dilekatkan pada *carrier file* dan

output-nya disebut dengan *Stegofile.* Apabila suatu informasi yang dilekatkan tersebut ingin dibuka *(ekstract)* kembali, maka bit-bit LSB yang ada pada *stegofile* akan diambil satu per satu dan dikembalikan lagi atau disatukan kembali sehingga menjadi sebuah informasi atau disebut dengan *decoding/ retrieving (Alfebra dkk, 2008).* Standar *flowchart* dari metode LSB yang digunakan untuk menyisipkan informasi dan mengekstraknya dapat dilihat pada gambar 2.3.



Gambar 2.3. Flowchart LSB *(Sumber: Alfebra dkk, 2008)*

Berdasarkan gambar 2.3 di atas dapat dijelaskan alur steganografi dalam menyembunyikan sebuah data digital dengan metode LSB:

1. Pertama sekali yang perlu dilakukan adalah menginputkan sebuah data digital yang berfungsi sebagai media penampung penyembunyian data digital lain *(carrier file).*
2. Kemudian pilih *file* atau teks yang akan disembunyikan dalam *carrier file* dengan cara mengonversikannya ke bentuk biner.
3. Lakukan pengukuran ukuran *file* atau teks yang akan disembunyikan dimana *file* dan teks harus >= dari *file* atau teks yang akan disembunyikan.
4. Selanjutnya, proses *embedding* dilakukan.
5. Proses *embedding* dilakukan dengan menukar bit-bit terakhir (LSB) pada *carrier file* diganti dengan bit-bit pada teks atau *file* yang akan disembunyikan.
6. Konversikan *carrier file* yang telah dimodifikasi dengan bit-bit *file* atau teks ke bentuk vektor.
7. Jika sukses, maka akan dihasilkan *file* steganografi *(stegofile)*.
8. Selesai.

Berikut ini adalah contoh proses steganografi. Pada contoh, dimisalkan bilangan biner pada kotak berikut adalah *carrier file* untuk pesan yang akan disembunyikan. Anggap pesan yang akan disembunyikan adalah *text.*

01001101 00101110 10101110 10001010 10101111 10100010 00101011 10101011

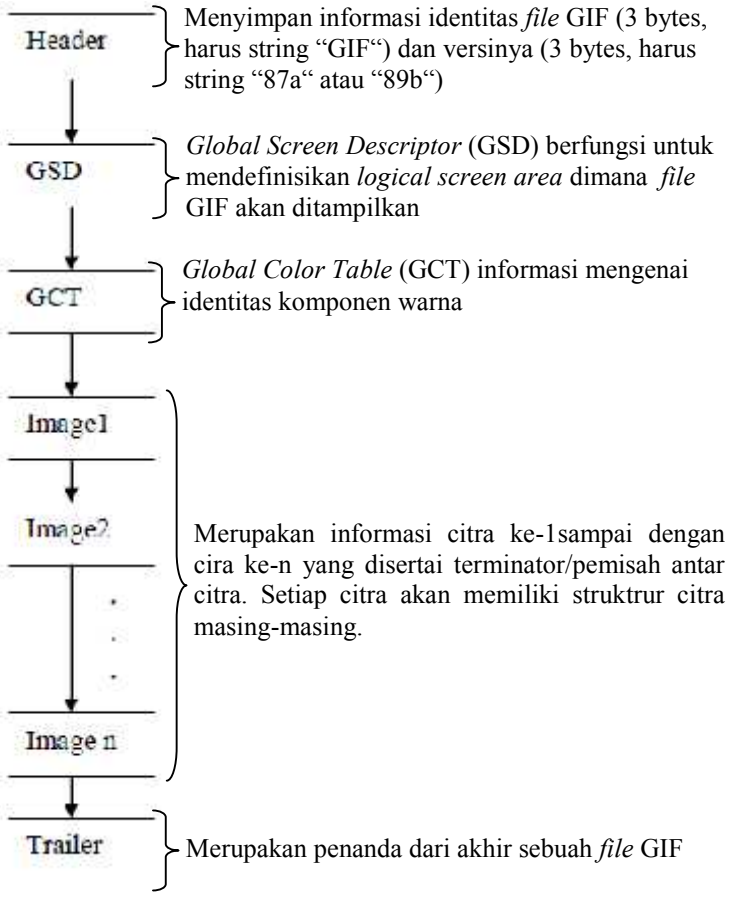
Biner-biner di atas digunakan untuk menyimpan karakter '**H**' yang memiliki nilai biner **(01001000)**, maka pixel – pixel wadah tersebut akan dirubah menjadi:

0100110**0** 00101111 10101110 10001010 10101111 10100010 0010101**0** 1010101**0**

2.3. Media-Media Steganografi

Berdasarkan penjelasan sebelum sub-bab ini dapat diketahui bahwa semua *file* digital dapat dijadikan sebagai media penampung suatu informasi rahasia digital. Oleh sebab itu, berikut akan dijelaskan beberapa *file* digital yang paling sering dijadikan sebagai sampel penelitian dan dari kedua *file* digital ada beberapa format yang paling sering dijadikan sampel penelitian steganografi.

2.3.1. Media Steganografi pada *File GIF (*.gif)*



Gambar 2.4. Struktur Format GIF *(Sumber: http://digilib.its.ac.id)*

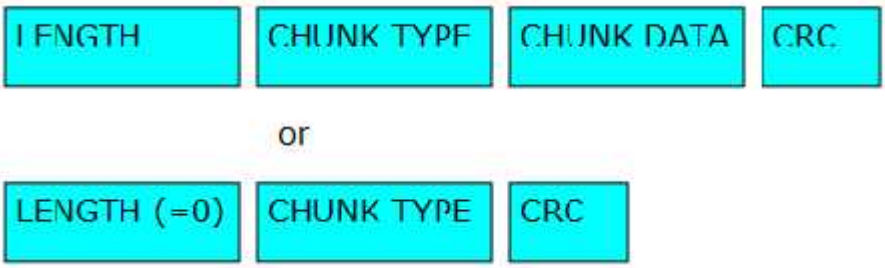
GIF *(Graphic Interchange Format)* dibuat oleh Compuserve pada tahun 1987 untuk menyimpan berbagai *file* bitmap manjadi *file* lain yang mudah diubah dan ditransmisikan pada jaringan komputer. GIF merupakan format citra web yang tertua yang mendukung kedalaman warna sampai 8 bit (256 warna),

menggunakan 4 langkah *interlacing*, mendukung *transparency*, dan mampu menyimpan banyak *image* dalam 1 *file (Steve, 1993).* Secara umum, struktur penyusun *file* GIF ini dapat dilihat pada gambar 2.4.

2.3.2. Media Steganografi pada *File PNG (*.png)*

PNG *(Portable Network Graphics)* merupakan format gambar yang dirancang untuk dunia internet yang merupakan pengganti dari gambar berformat GIF. Kelebihan yang dimiliki oleh PNG adalah dapat menyimpan data *gamma* dan *Kromatisitas* warna telah disesuaikan pada *platform heterogen (http://www.w3.org).* Format penamaan *file* PNG diatur ke dalaam urutan blok-blok biner yang disebut dengan *chunk.* Secara umum blok-blok biner *(chunk)* ini dapat dilihat pada gambar 2.5 yang merupakan ilustrasi dari struktur format PNG ini.

Berdasarkan gambar 2.5 dapat dijelaskan bahwa *Length* merupakan informasi ukuran *file* yang panjangnya adalah 4 byte. CRC *(Cyclic Redundancy Check)* merupakan bagian gambar yang berfungsi untuk pendeteksian *(error checking)* pada saat transmisi data.



Gambar 2.5. Struktur Format PNG *(Sumber: http://www.w3.org)*

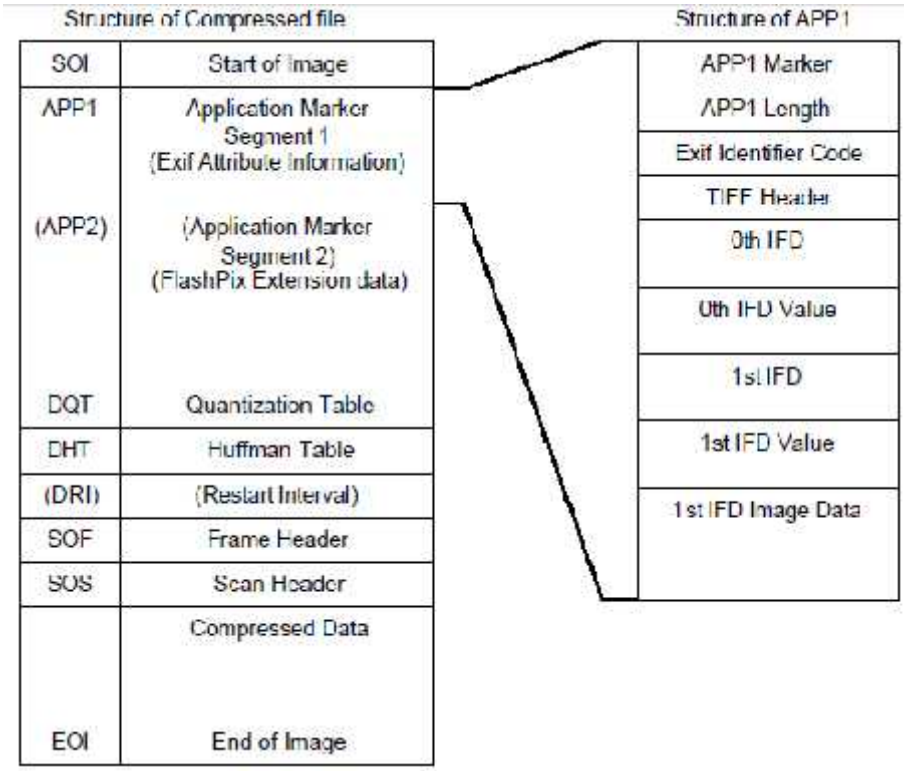
Chunk Type merupakan informasi nama *chunk* yang besarnya adalah 4 byte yang terdiri dari 4 ASCII dengan spesifikasi *(http://digilib.its.ac.id)*:

1. Karakter ke-1,2, dan 4 boleh *uppercase/lowercase.*
2. Jika karakter ke-1 *uppercase*, maka disebut *critical chunk* (harus valid), contohnya: IHDR, PLTE, IDAT, dan IEND dimana:
    a. PNG *Signature*: tanda *file* PNG
    b. IHDR *chunk*: menyimpan *dimension*, *depth*, dan *color type*

c. PLTE *chunk*: untuk PNG yang menggunakan *color palette type*
d. IDAT *chunk* 1, IDAT *chunk* 2, IDAT *chunk* 3, ... IDAT *chunk*-n
e. IEND *chunk*: *end of PNG image*

3. Jika karakter ke-1 *lowercase*, berarti *non-critical chunk* (contohnya: bKGD, cHRM, gAMA, hIST, pHYs, sBIT, tEXt, tIME, tRNS, zTXt).
4. Jika karakter ke-2 *uppercase*, berarti *public* (PNG Standard)
5. Jika karakter ke-2 *lowercase*, berarti *private* PNG
6. Jika karakter ke-4 *lowercase*, berarti *save-to-copy*
7. Jika karakter ke-4 *uppercase*, berarti *unsave-to-copy*
8. Karakter 3 harus *uppercase*

2.3.3. Media Steganografi pada *File JPEG (*.jpeg)*



Gambar 2.6. Struktur Kompresi Format JPEG *(Sumber: http://www.exif.org)*

JPEG *(Joint Photograpic Experts Group)* merupakan citra gambar yang bersifat kompresi dan *lossy* (menghilangkan). Berbeda dengan 2 format gambar sebelumnya, yaitu GIF dan PNG yang juga bersifat kompresi, tapi *losless* (tidak

menghilangkan). *Lossy* maksudnya adalah bahwa ketika sebuah citra dikompresi menjadi *file* JPEG, maka ada kemungkinan terjadinya penghilangan beberapa bit pada citra semula, sehingga akan berakibat menurunkan kualitas citra. Namun, karena JPEG mendukung indeks warna sebanyak 16 juta sehingga penurunan kualitas terhadap citra JPEG tidak akan terlihat secara jelas oleh mata manusia.

JPEG merupakan nama teknik kompresi, sedangkan nama format *file*-nya adalah JFIF *(JPEG File Interchange Format)*. Secara umum, struktur *file* kompresi jenis ini dapat dilihat pada gambar 2.6. Struktur JPEG ini memiliki bagian yang menyimpan berbagai informasi yang masing-masing diawali oleh sebuah "marker" untuk penanda yang berukuran 2 bytes. Byte pertama selalu bernilai FF16 sedangkan bit kedua bisa berupa *(http://digilib.its.ac.id)* :

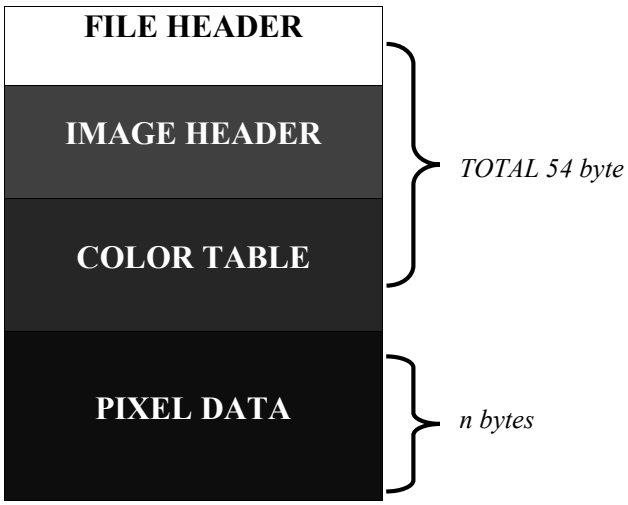
1. APPn: untuk meng-*handle application specific data*, misalnya informasi tambahan yang ada dalam JPEG
2. COM *(Comment)*: untuk memberikan komentar *plain text string*, seperti *copyright*.
3. DHT *(Define Huffman Table)*: menyimpan tabel kodekode Algoritma Huffman
4. DRI *(Define Resart Interval)*: sebagai tanda *restart interval*
5. DQT *(Define Quantization Table)*: mendefinisikan tabel kuantisasi yang digunakan dalam proses kompresi
6. EOI *(End of Image)*: tanda akhir *file* JPEG
7. RSTn: *restart marker*
8. SOI *(Start of Image)*: tanda awal image
9. SOFn: *start of frame*
10. SOS: *start of scan*

2.3.4. Media Steganografi pada *File Bitmap (*.bmp)*

Bitmap (*.bmp) merupakan format *file* gambar yang pertama sekali berhasil diteliti sebagai *carrier file* pada steganografi. Bitmap merupakan *carrier file* dimana setiap pixel biasanya bersifat independen yang dapat diubah dan dimodifikasi, atau dalam kalimat lain dapat dikatakan bahwa bitmap tidak akan

mengalami perubahan kualitas yang besar jika terjadi modifikasi pada bit-bitnya *(Walaa dkk, 2010).*

File gambar berformat *.bmp secara umum dibentuk oleh bit-bit penyusun. Setiap bit penyusun memiliki maksud dan fungsi masing-masing. Secara umum, sebuah *file* gambar berformat bitmap tersusun atas *file header, image header, color table,* dan *pixel data.*



Gambar 2.7. Struktur Format Bitmap *(Sumber: Muhammad, 2008)*

Pada gambar 2.7 di atas dapat diketahui bahwa *file header* merupakan bagian yang merepresentasikan jenis atau format *file* atau sebagai bagian identifikasi *file.* Letaknya di blok pertama yang berfungsi untuk memastikan apakah *file* ini benar *file* bitmap atau tidak. Identifikasi ini dapat diketahui dengan menemukan kode bit yang menuliskan BM (bmp). *Image header* merupakan bagian yang berisi informasi ukuran panjang dan lebar *file* dalam satuan pixel, format warna (jumlah bidang warna/ bits-per-pixel), informasi apakah bitmap terkompresi atau tidak serta tipe kompresinya, jumlah data bitmap dalam byte, resolusi, dan jumlah warna yang digunakan. *Color Table* merupakan bagian yang berisi informasi intensitas RGB untuk setiap komponen warna pada *pallete*. Dan *pixel data* berisi pixel-pixel data atau gambar.

Perlu diperhatikan, bagian *File Header, Image Header*, dan *Color Palette* terdiri atas informasi-informasi yang penting untuk menampilkan citra, apabila

terjadi kehilangan data atau kerusakan data pada bagian-bagian ini maka hal tersebut akan mengakibatkan citra rusak atau bahkan tidak bisa ditampilkan, keculi pada *pixel data.* Oleh sebab itu, pesan akan disisipkan pada bagian *pixel data* ini *(Ferri, 2009)*. Selain itu, hal-hal lain yang perlu diperhatikan adalah:

1. Format *file* ini menyimpan datanya secara terbalik, yaitu dari bawah ke atas
2. Citra dengan resolusi warna 8-bit, lebar citra harus merupakan kelipatan dari 4, bila tidak maka pada saat penyimpanan akan ditambahkan beberapa *byte* pada data hingga merupakan kelipatan dari 4.
3. Citra dengan resolusi warna 24-bit, urutan penyimpanan tiga warna dasar adalah merah, hijau, biru (RGB). Lebar citra dikalikan dengan 3 harus merupakan kelipatan dari 4, bila tidak maka pada saat penyimpanan akan ditambahkan beberapa *byte* pada data hingga merupakan kelipatan dari 4.
4. BMP mendukung pemampatan *run-length encoding* (RLE).

2.3.5. Media Steganografi pada *File WAVE (*.wav)*

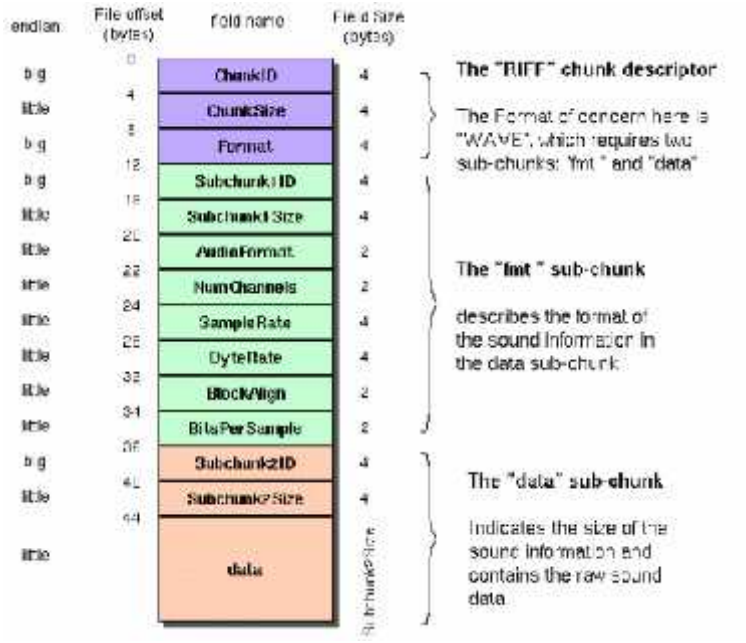
File WAV adalah *file* audio standar yang digunakan oleh Windows. Suara yang berupa digital audio dalam *file* wav disimpan dalam bentuk gelombang, karena itulah *file* ini memiliki ekstensi *.wav (wave). Pada metode LSB, cara ini sebenaranya sama ketika pesan disisipkan pada *file* gambar, yaitu menyisipkan bit-bit pesan ke bit-bit LSB *carrier file*, kemudian bit-bit LSB *carrier file* diganti dengan bit-bit pada pesan.

Sebuah *file* wav memiliki struktur standar yang disebut dengan RIFF *(Resource Interchange File Format)* yang merupakan struktur untuk data-data multimedia dalam Windows. Struktur ini mengatur data dalam *file* ke dalam bagian-bagian yang masing-masing memiliki *header* dan *size* yang disebut dengan *Chunk* yang memungkinkan bagi suatu program untuk mengenali bagian-bagian tertentu dari *file*. Contoh *file* yang menggunakan struktur RIFF adalah *file* wav dan avi. Struktur RIFF yang telah dijelaskan di atas dapat dilihat pada gambar 2.8.

Sesuai dengan struktur *file* RIFF, *file* wav diawali dengan 4 byte yang berisi “RIFF”, lalu diikuti oleh 4 byte yang menyatakan ukuran dari *file* tersebut,

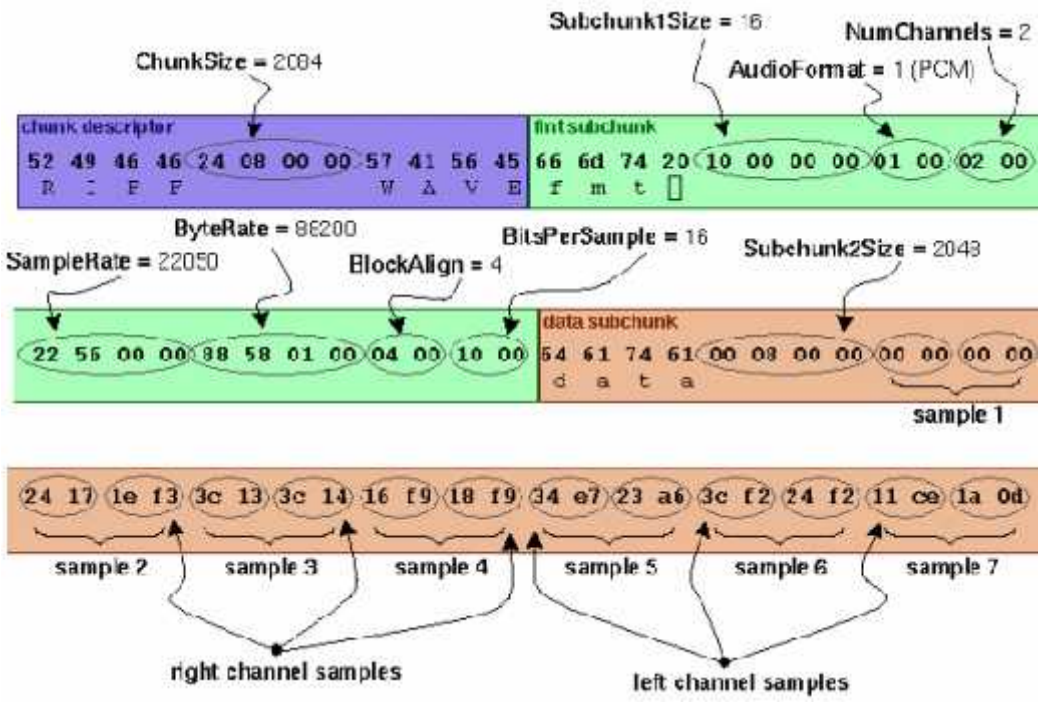
dan 4 byte berikutnya adalah “WAVE” yang menyatakan bahwa *file* tersebut adalah *file* wav dan peyembuyian pesan dapat dilakukan di data bagian dari RIFF. Selanjutnya adalah bagian data RIFF berisi informasi dari format sample atau disebut sebagai subbagian-subbagian dari RIFF, lalu diikuti dengan sub-bagian data audio-nya yang dapat dilihat pada gambar 2.9.

Gambar 2.8. Struktur RIFF *(Sumber: Dziki, 2007)*



Gambar 2.9. Detail struktur *Chunk (Sumber: http://www.codeproject.com)*

Pada gambar 2.9 dapat diketahui bahwa sebuah *file* “Wave” secara umum terdiri dari tiga bagian dasar, yaitu deskripsi *chunk* RIFF, format sub-*chunk,* dan data sub-*chunk.* Deskripsi *chunk* RIFF dan format sub-*chunk* telah dijelaskan sebelumnya di atas. Sementara data sub-*chunk* terdiri dari *header data sub-chunk* berupa “ID”, kemudian *size* dari data sub-*chunk*, dan selanjutnya bagian dari *data audio* yang di dalamnya berisikan bit-bit *digital sample audio*. Bit-bit *digital sample audio* inilah yang dijadikan sebagai media penyembunyian sebuah data atau *file* rahasia pada *file audio* berformat *.wav. Struktur data sub-*chunk* ini dapat dilihat dengan jelas pada gambar 2.10.



Gambar 2.10. Struktur Data sub-c*hunk (Sumber: http://www.codeproject.com)*

Sebuah data digital audio dapat memiliki beragam kulitas suara yang ditentukan dari *bitrate, samplerate,* dan jumlah *channel. Bitrate* merupakan ukuran bit tiap sampelnya, yaitu 8-bits, 16-bits, 24 bits, atau 32 bits. *Samplerate* menyatakan banyaknya jumlah sampel yang dimainkan setiap detikanya. *Samplerate* yang umum dipakai adalah 8000Hz, 11025Hz, 22050Hz, dan 44100Hz. 8000Hz (banyaknya sampel suara yang dimainkan tiap detiknya adalah 8000 kali) memiliki kualitas seperti suara di telephone, 11025Hz biasa digunakan

untuk merekam suara, kemudian untuk menhasilkan suara yang lebih bagus *samplerate* ini dinaikkan menjadi 22050Hz. Sementara 44100Hz biasanya digunakan untuk suara-suara berkualitas CD. *Channel* pada dasarnya terbagi atas dua macam, yaitu mono dan stereo diamana untuk 1 *channel* (mono) akan membutuhkan kapasitas sebesar 2 byte untuk memainkan sekian kali *sampelerate*. Mono berarti hanya memiliki 1 *channel,* sedangkan stereo 2 *channel* yang artinya memakan tempat 2 kali lebih banyak daripada mono atau sama dengan membutuhkan kapasitas sebesar 2 kali 2 byte (4 byte) untuk memainkan 1 sampel suara.

2.4. Pengujian Kelayakan Steganografi

Penelitian terhadap sekuriti data khususnya di bidang steganografi telah banyak dipublikasikan dalam bentuk laporan penelitian. Disetiap tulisan penelitian selalu menempatkan pengujian sebagai akhir dari sebuah penelitian. Pada steganografi, ada beberapa hal yang sering dijadikan objek penelitian yang berkaitan dengan aspek dasar dari steganografi itu sendiri, yaitu: kapasitas *(capacity)*, kemanan *(security)*, dan ketahanan *(robustness)*. Ketiga aspek ini selalu menjadi objek pengujian pada steganografi yang dapat dilakukan dengan menggunakan *tools* tambahan untuk mendukung hasil penelitian, seperti: pada aspek keamanan dapat diuji dengan aplikasi *steganalyst,* misalnya *StegSpy* atau *StegDetect* yang dapat diperoleh secara bebas di site-site tertentu, dan pada aspek ketahanan dapat memanfaatkan aplikasi-aplikasi gambar dan audio editor, seperti photoshop, paint, Microsoft Office Picture Managaer, Cool Edit Pro 2.0, dan lain sebagainya. Hal ini sangat wajar dilakukan, karena steganografi telah memberikan tiga aspek yang sangat jelas dalam mengidentifikasikan fungsi dan manfaatnya di bidang sekuriti data. Oleh sebab itu, beragam formulasi, cara, teknik, bahkan aplikasi atau *tools* tambahan yang berfungsi sebagai alat pengujian digunakan untuk menguji berhasil-tidaknya suatu aplikasi steganografi.

Pengujian lainnya adalah dari aspek kualitas dimana objeknya adalah *file* input dan output yang dihasilkan oleh aplikasi steganografi, yaitu dengan cara membandingkan *file* inputan yang akan dijadikan media penampung *(carrier file)*

dengan *file* output yang dihasilkan yang telah disisipi pesan *(stegofile).* Pengujian ini mengggunakan formulasi yang disebut dengan *Peak Signal to Noise Ratio* (PSNR) untuk *file* gambar dan *Signal to Noise Ratio* (SNR) untuk *file* suara yang keduanya memiliki satuan dalam decibel (dB). Menurut Cole (2003) nilai PSNR dikatakan baik jika berada diatas nilai 20 dB, artinya di bawah nilai 20 dB distorsi yang terjadi sangat besar antara *carrier file* dan *stegofile*. Adapun formulasi PSNR ini adalah sebagai berikut *(Cole, 2003):*

$$\text{PSNR} - 10\,Log_{10}\left(\frac{255^2}{MSE}\right) \tag{2.1}$$

Dimana nilai 255 merupakan nilai tertinggi intensitas suatu piksel dan MSE *(Mean Sequare Error)* merupakan nilai rata-rata dari jumlah kuadrat *Absolute Error* antara *carrier file* dengan *stegofile.* Nilai ini dapat diperoleh dengan menggunakan formulasi berikut:

$$MSE - \frac{1}{m\,n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2 \tag{2.2}$$

Dimana m = Jumlah baris/ lebar gambar

n = Jumlah kolom/ tinggi gambar

I*(i,j)* = Nilai pixel dari gambar asli

K*(i,j)* = Nilai pixel dari gambar yang mengandung pesan *(stegofile)*

Sementara untuk formulasi SNR yang digunakan untuk mengetahui besarnya perbandingan nilai distorsi yang terjadi setelah pesan disisipkan pada *carrier*-nya memiliki persamaan sebagai berikut *(Parthasarathy, 2005 – 2009):*

$$\text{SNR} = 10\,Log_{10}\left\{\frac{\sum_n \mathbf{X}^2(\mathrm{n})}{\sum_n [\mathbf{X}^2(\mathrm{n}) - \mathbf{Y}^2(\mathrm{n})}\right\} \tag{2.3}$$

Dimana nilai X(n) menyatakan nilai rata-rata RMS *(Root Mean Sequare)* dari *file* asli *(carrier file)* dan Y(n) merupakan nilai rata-rata RMS dari *stegofile*. Nilai

RMS sendiri sebenarnya menyatakan jumlah atau banyaknya (besar-kecilnya) sampel suara yang dikeluarkan oleh sebuah suara yang dapat diperoleh dari beberapa aplikasi *audio editor* yang menyediakan informasi mengenai kualitas RMS sebuah audio, misalnya Cool Edit Pro 2.0.

2.5. Pendekatan terhadap Pemrograman Berorientasi Objek *(Object Oriented Programming – OOP)*

Object Oriented Programming (selanjutnya disebut OOP) merupakan salah satu cara baru dalam teknologi informasi yang bertujuan untuk menyelesaikan permasalahan dengan cara menimbulkan kerangka berpikir serta logika ke dalam bentuk objek-objek tertentu *(Adi, 2002).* Secara mendasar OOP berbeda dengan pemrograman tersetruktur yang merupakan pendahulunya. Pada OOP permasalahan dinyatakan seperti sebuah objek yang memiliki kombinasi struktur data dan fungsi tertentu atau disebut dengan entitas tunggal. Sementara pada pemrograman terstruktur, permasalahan didefinisikan secara terpisah antara struktur data dan fungsi pembangunnya yang menggambarkan tidak adanya hubungan yang erat diantara keduanya.

Objek pada OOP adalah konsep abstraksi atau sesuatu yang memiliki arti bagi aplikasi yang akan dikembangkan yang secara umum merupakan kata benda, namun tidak harus sebuah kata benda yang dapat dikenali oleh panca indra manusia pada umumnya *(Adi, 2002).* Selain objek, OOP juga mengenal istilah "identitas" yang merupakan sesuatu hal yang dapat membedakan sebuah objek yang satu dengan objek yang lainnya *(James dkk, 1991).*

2.5.1. Metode USDP *(Unified Software Development Process)*

USDP merupakan salah satu metode rekayasa perangkat lunak OOP yang berfungsi untuk memberikan kerangka berfikir dalam melakukan analisis terhadap pengembangan perangkat lunak berorientasi objek. USDP menjadikan UML *(Unified Modeling Language)* sebagai *tools* analisis OOP dimana *tools* ini telah digunakan oleh banyak vendor dalam melakukan analisis. USDP sebagai metode

analisis rekayasa perangkat lunaka memiliki beberapa karakteristik *(Adi, 2010)*, yaitu:

1. ***Use-case Driven***. Dengan metode USDP ini, *use-cases* merupakan urutan-urutan interaksi antara *actor* dengan system/ perangkat lunak yang sedang dikembangkan. Kemudian dilanjutkan oleh Ivar Jacobson (1992) dalam bukunya dijelaskan bahwa *use-cases* merupakan diagram yang menjadi pedoman-pedoman bagi diagram lainnya. Dengan kata lain, bahwa diagram-diagram yang terdapat pada *tools* UML akan cenderung terhadap analisis *use-cases* yang ada.
2. ***Architecture Centric***. Dijelaskan bahwa *use-cases* saja belum cukup untuk mengembangkan perankat lunak secara utuh. Oleh sebab itu, dibutuhkan sebuah arsitektur dianamis yang dapat memodelkan elemen-elemen perangkat lunak, seperti memodelkan subsistem-subsistem, kebergantungan-kebergantungan, antarmuka-antarmuka, kolaborasi-kolaborasi, simpul-simpul, serta kelas-kelas aktif.
3. ***Iterative and Incremental***. Pada dasarnya sebuah perangkat lunak dibangun dan dikerjakan secara *iterative* untuk menghasilkan perangkat lunak yang terintegrasi secara *incremental*. Iteratif pada pengembangan perangkat lunak berlangsung pada sejumlah *use-cases* yang secara keseluruhan memperluas fungsionalitas system yang lebih besar.

Telah dijelaskan di atas bahwa USDP merupakan metode yang memiliki karakteristik pada pengembangannya, yaitu bersifat terpusat (sentral) pada *use-cases* analisis yang ada. Selain karakteristik, USDP menjadi kerangka analisis terhadap model-model analisis pada UML *(Adi, 2010)*, seperti:

1. **Model Analisis** memiliki 2 kegunaan, yaitu memperhalus dan merinci definisi-definisi masing-masing *use-cases*.
2. **Model Perancangan** mendefinisikan struktur statis sistem, seperti subsistem, kelas-kelas, antarmuka dan hubungannya yang masing-masing berada dalam kerangka sistem yang dikembangkan.

3. **Model Implementasi** memuat komponen-komponen (kode-kode) bahasa pemrograman tertentu dan melakukan pemetaan kelas-kelas ke komponen-komponen.
4. **Model *Deployment*** mendefinisikan simpul-simpul komputer secara fisik dan melakukan pemetaan masing-masing komponen ke setiap simpul komputer yang ada.
5. **Model Pengujian** mendeskripsikan kasus-kasus dan prosedur-prosedur pengujian yang tujuannya adalah melakukan verifikasi terhadap perangkat lunak yang dihasilkan dengan cara melihat dan memastikan apakah masing-masing *use case* telah diimplementasikan dengan cara yang sesuai dengan fungsionalitas utama yang tercakup di dalamnya.

2.5.2. UML *(Unified Modeling Language)*

Seperti yang telah dijelaskan di atas bahwa *Unified Software Development Process* (USDP) merupakan salah satu metode pemodelan objek dalam pengembangan sistem/ perangkat lunak berbasis OOP yang menggunakan UML *(Unified Modeling Language)* sebagai *tool* utamanya. UML adalah "bahasa" pemodelan untuk sistem atau perangkat lunak yang berparadigma " berorientasi objek" *(Adi, 2010).* Pada dasarnya pemodelan digunakan untuk menyederhanakan permasalahan-permasalahan yang sifatnya kompleks sehingga lebih mudah dipelajari dan dipahami. Namun tidak tertutup juga untuk memodelkan sebuah permasalahan sederhana dan UML digunakan untuk menyederhanakan permasalahan-permasalahan tersebut yang bertujuan untuk menggantikan metodologi pengembangan perangkat lunak sebelumnya, seperti ERD dan DFD.

Berdasarkan tabel 2.1 di atas dapat diketahui bahwa UML dibagi menjadi beberapa *view* dimana *view* merupakan sejumlah konstruksi pemodelan UML yang merepresentasikan suatu aspek tertentu dari sistem/ perangkat lunak yang sedang dikembangkan yang dilihat berdasarkan 3 area utama atau *major area* yang terdiri dari: klasifikasi struktural *(structural classification),* prilaku dinamis *(dynamic behavior),* serta *model management* atau pengelolaan/ managemen model *(Adi, 2010). Diagram* merupakan gambaran yang bermanfaat untuk

memodelkan abstraksi dan perancangan program aktual sebuah objek untuk menyederhanakan masalah, memudahkan memahami masalah, akurat, dan baik diterapkan dalam kenyataan yang meliputi: *Class diagram, Use case diagram, Component diagram, Deployment diagram, Statechart diagram,* dan *Collaboration diagram (Adi, 2010).*

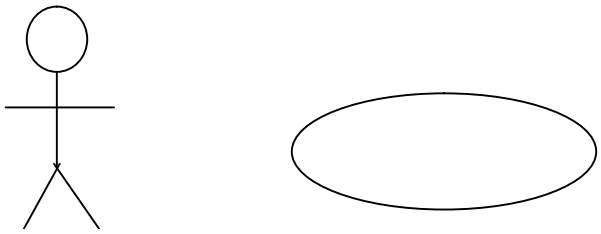
Tabel 2.1 *View* dan Diagram dalam UML

Major Area	***View***	***Diagram***	***Main Concept***
Structural classification	*Static view*	*Class diagram*	*Class, association generalization, dependency, realization, interface*
	Use case view	*Use case diagram*	*Use case, actor, association, extend, include, use case generalization*
	Implementation view	*Component diagram*	*Component, interface, dependency, realiation*
	Deployment view	*Deployment diagram*	*Node, component, dependency, location*
Dynamic behaviour	*State machine view*	*Statechart diagram*	*State, event, transition, action*
	Activity view	*Activity diagram*	*State, activity, completion transition, fork, join*
	Interaction view	*Sequence diagram*	*Interaction, object, message, activition*
		Collaboration diagram	*Collaboration, interaction, collaboration role, message*
Model management	*Model management view*	*Class diagram*	*Package, subsystem, model*
Extensibility	*All*	*All*	*Constraint, sterotype, tagged values*

(Sumber: Adi, 2010)

a. *Use Case Diagram*

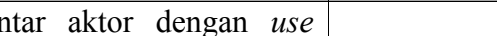
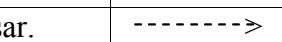
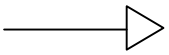
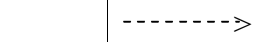
Use cases dan aktor merupakan model fungsionalitas sistem/ perangkat lunak yang dilihat dari sisi pengguna yang ada diluar sistem, namun memiliki koherensi terhadap sistem/ perangkat lunak tersebut yang diekspresikan sebagai transaksi-transaksi yang terjadi antara aktor dan sistem *(Adi, 2010).* Dalam buku lain yang berjudul *Object-Oriented Software Engineering (A Use Case Driven Approach)* dijelaskan *use case* merupakan sekumpulan aksi dari sebuah sistem yang menspesifikasikan cara yang digunakan sebuah sistem untuk mengeksekusi aksi dari aktor bahwa aktor dan *use cases* merupakan transformasi utama yang dibuat dari sisi spesifikasi kebutuhan sebuah model sistem/ perangkat lunak yang terdiri dari sebuah model *use case,* deskripsi terhadap *interface,* domain masalah model *(Jacobson, 1992).* Berikut ini adalah gambar komponen-komponen yang harus ada pada sebuah *use case.*



Gambar 2.11. *Use case model* terdiri dari aktor dan *use cases*
(Sumber: Jacobson, 1992)

Seperti yang ditunjukkan pada gambar 2.11 di atas, model *use case* menggunakan aktor dan *use cases* untuk mendefinisikan hal-hal apa yang terjadi di luar sistem (aktor) dan apa aksi atau pengaruh dari sistem *(use cases)*, dimana *use case* bersifat mandiri satu dengan yang lainnya, meskipun implementasinya suatu *use case* memungkinkan membuat kebergantungan implisit antarobjek-objek yang dibagikan. *(Jacobson, 1992)*.

Tabel 2.2. Relasi-relasi dalam *use case*

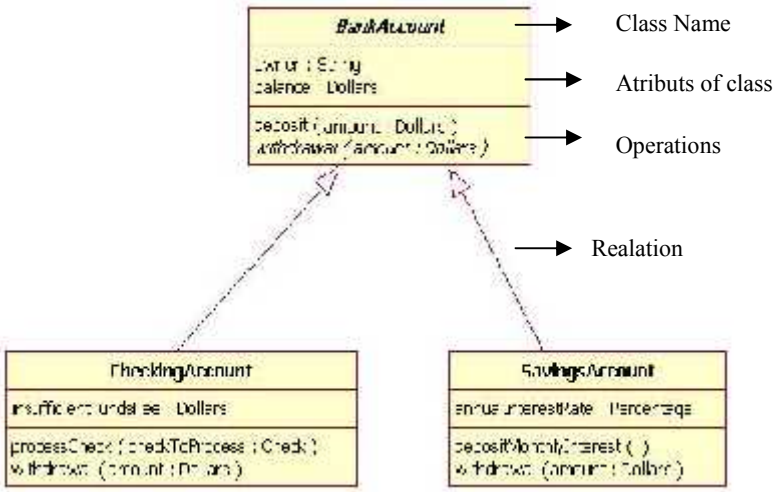
Relasi	Fungsi	Notasi
Asosiasi *(association)*	Lintasan komunikasi antar aktor dengan *use case.*	
Extend	Penambahan prilaku ke suatu *use case* dasar.	
Generalisasi *use case*	Menggambarkan hubungan antara *use case* yang bersifat umum dengan *use case* yang bersifat lebih spesifik.	
Include	Penambahan prilaku ke suatu *use case* dasar dan mendeskripsikan penambahan tersebut.	

(Sumber: Adi, 2010)

b. *Class Diagram*

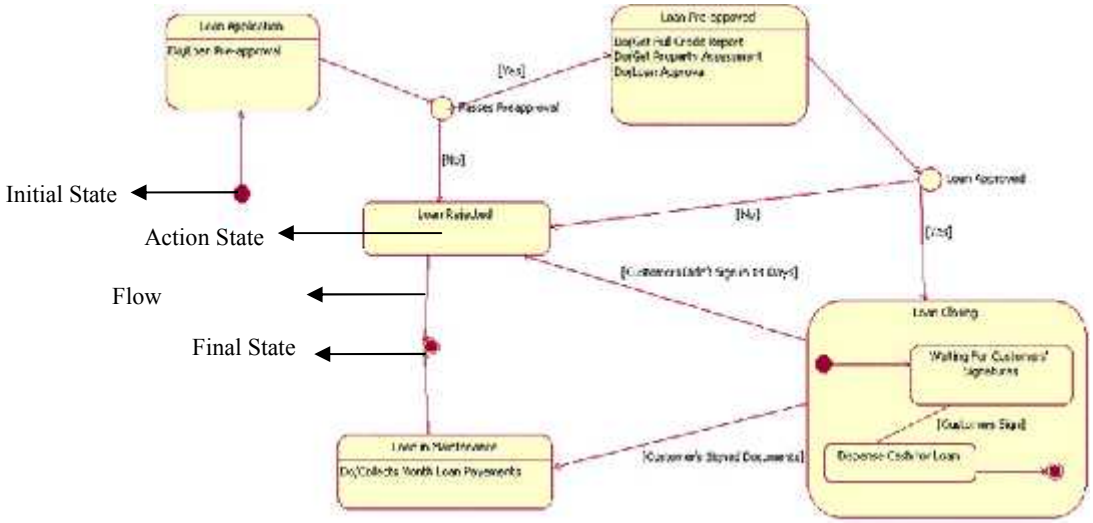
Sebuah kelas *(class)* merupakan konsep diskrit dalam model yang terdiri dari identitas *(identity), state,* prilaku *(behaviour)*, serta relasi dengan kelas yang lainnya *(relationship).* Beberapa jenis pengelasan mencakup di dalamnya kelas, *interface*, serta tipe data *(Adi, 2010).* Kelas diagram pada umumnya terdiri dari nama kelas, atribut-atribut pembangunnya, operasi dan relasi antar kelas. Hal ini dapat dilihat pada gambar 2.12.

Kelas umumnya mendefinisikan sejumlah objek yang memiliki state dan prilaku tertentu. State bisa dideskipsikan menggunakan atribut-atribut dan asosiasi kelas. Atribut merupakan data-data tanpa identitas tertentu, sementara asosiasi digunakan untuk menghubungkan objek-objek tertentu. Prilaku dalam kelas merupakan operasi-operasi untuk kelas/ objek yang bersangkutan, sementara metode adalah impelementasi dari suatu operasi dalam bahasa pemrograman berorientasi objek tertentu untuk pengembangan sebuah sistem/ perangkat lunak. Sebuah metode bisa bersifat private, public, protected, maupun friend/ default *(Adi, 2010).*



Gambar 2.12. Ilustrasi sebuah *class diagram (Sumber: http://www.ibm.com)*

c. *State Chart Diagram*



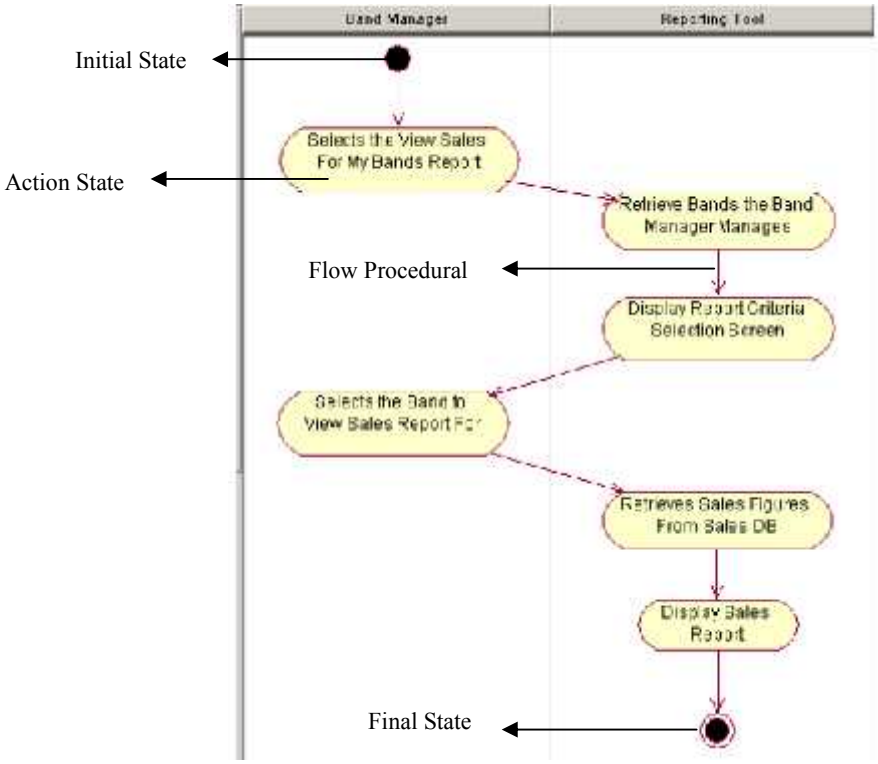
Gambar 2.13. Ilustrasi sebuah *state chart diagram*
(Sumber: http://www.ibm.com)

State mendeskripsikan suatu perjalanan pengklasifikasi dalam perjalanan waktu. Dengan kata lain *state* berfungsi untuk memberikan berbagai cara/jalan kepada model untuk berbagai kejadian yang mungkin terjadi dalam sebuah objek. *State* digunaan untuk menggambarkan prilaku objek yang sifatnya dinamis dalam

sebuah sistem. Sebuah *state chart diagram* terdiri dari *initial state, final state, action state,* dan alur (transisi). Hal ini dapat dilihat pada gambar 2.13.

d. *Activity Diagram*

Berikut ini adalah gambar 2.14 yang mengilustrasikan sebuah *activity diagram* perangkat lunak yang pada dasarnya merupakan perluasan dari *state chart*. *Activity diagram* merupakan model aliran yang sifatnya prosedural, namun pada prakteknya sebuah *activity diagram* berfokus pada urutan tindakan eksekusi dan kondisi yang membutuhkan aksi langsung secara internal pada sistem yang bertujuan untuk memodelkan aliran tindakan prosedural yang merupakan bagian dari kegiatan yang lebih besar pada pengembangan perangkat lunak *(Bell, 2003)*.



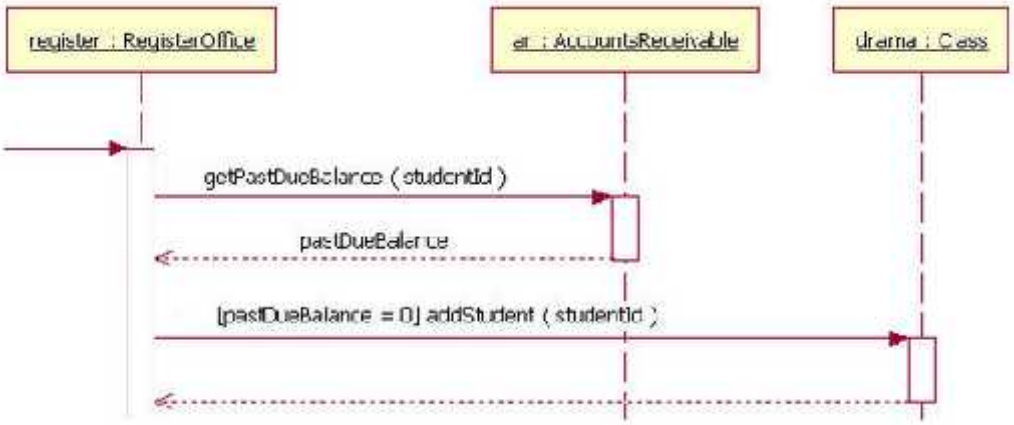
Gambar 2.14. Ilustrasi sebuah *activity diagram (Sumber: http://www.ibm.com)*

Activity diagram terdiri dari *state* yang menyatakan *state* dari komputasi yang dieksekusi dan dilaksanakan tanpa adanya interupsi-interupsi eksternal berbasis *event* yang terjadi pada sistem *(action state)* yang merepresentasikan

eksekusi pernyataan dalam suatu prosedur suatu aktivitas dalam suatu aliran kerja *(Adi, 2010).* Karena *activity diagram* menunjukan sebuah *sequence* dari aksi-aksi sistem, maka *activity diagram* harus mengindikasikan adanya *state* awal yang disebut dengan *initial state* dan *state* akhir atau disebut dengan *final state* (lihat gambar 2.11) *(Bell, 2003).*

e. *Sequence Diagram*

Secara umum *sequence diagram* merupakan gambaran interaksi sebagai diagram dua dimensi (matra) yang terdiri dari matra vertikal dan horizontal. Matra vertikal merupakan sumbuh waktu yang berubah dan bertambah dari atas ke bawah, sementara matra horizontal merupakan peran pengklasifikasi yang merepresentasikan objek-objek mandiri yang terlibat dalam sebuah kolaborasi. Masing-masing pengklasifikasi direpresentasikan sebagai kolom-kolom vertikal dalam *sequence diagram* dan sering disebut sebagai garis waktu *(lifeline)*.

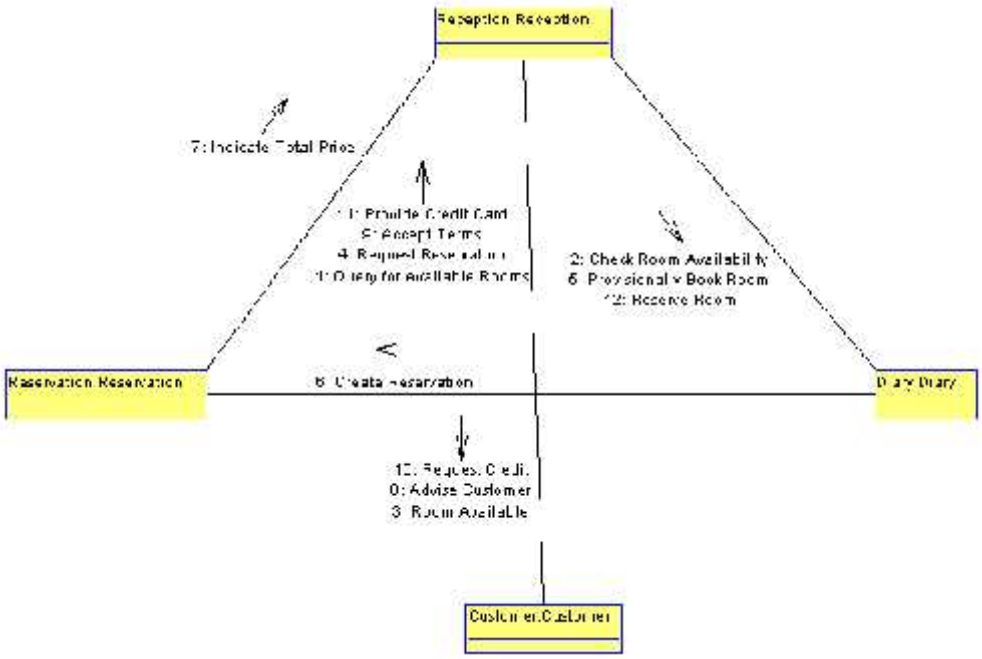


Gambar 2.15. Ilustrasi sebuah *sequence diagram (Sumber: http://www.ibm.com)*

Selama objek ada, peran digambarkan menggunakan garis tegas. Selama aktivasi ada prosedur pada objek aktif, garis waktu digambarkan sebagai garis ganda. Pesan-pesan digambarkan sebagai suatu tanda panah dari garis waktu objek ke garis waktu objek lain, dan panah-panah menggambarkan aliran pesan antarperan pengklasifikasian yang digambarkan dalam urutan waktu kejadiannya dari atas ke bawah. Untuk lebih jelasnya dapat dilihat pada gambar 2.15.

f. ***Collaboration Diagram***

Merupakan sebuah diagram interaksi yang menunjukkan penyampaian pesan-pesan atau interaksi (komunikasi) yang terjadi antar operasi-operasi atau transaksi-transaksi dalam kurun waktu tertentu pada sebuah sistem. Sebuah kolaborasi diagram terdiri dari objek, link yang menghubungkan tiap objek, dan pesan yang disampaikan disetiap transaksi objek. Masing-masing digram kolaborasi membangun sebuah *view* interaksi atau relasi struktural antara objek dan entitas atau objek lainnya yang mememiliki pengaruh langsung pada sistem. Pada dasarnya diagram ini adalah sama dengan *sequence diagram,* sehingga dijadikan sebagai alternatif diagram dalam pemodelan sistem berorientasi objek. Hal ini dapat dilihat pada gambar 2.16 berikut.



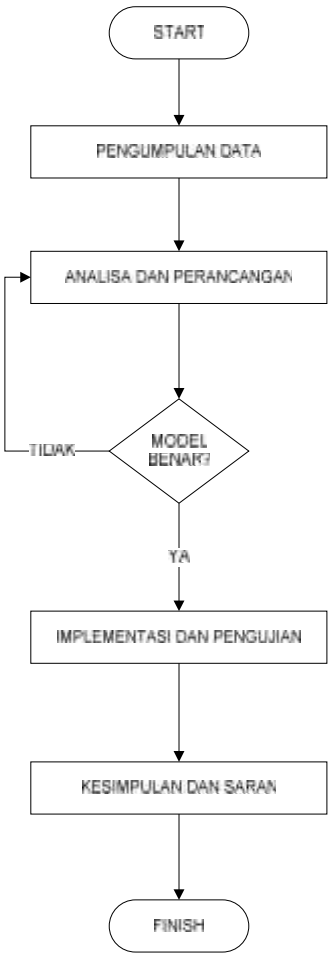
Gambar 2.16. Ilustrasi sebuah *collaboration diagram*

(Sumber: http://www.ibm.com)

BAB III

METODOLOGI PENELITIAN

3.1. Alur Metodologi Penelitian



Gambar 3. 1 Tahapan Metodologi Penelitian

Gambar 3.1 di atas merupakan metodologi penelitian yang akan dilakukan oleh penulis. Metodologi penelitian bertujuan untuk menguraikan seluruh kegiatan yang dilaksanakan selama kegiatan penelitian berlangsung. Dari gambar di atas, dapat diketahui bahwa ada tiga tahapan yang akan dilakukan untuk menyelesaikan kasus pada penelitian tugas akhir ini yang meliputi: pengumpulan data, analisa dan perancangan, implementasi dan pengujian aplikasi, dan selanjutnya kesimpulan dan saran.

3.2. Pengumpulan Data

Pengumpulan data merupakan metode yang difungsikan untuk memperoleh informasi-informasi atau data-data terhadap kasus yang menjadi permasalahan dalam laporan tugas akhir ini. Hal yang paling perlu dibutuhkan oleh penulis adalah informasi-informasi mengenai metode yang digunakan dalam penelitian kasus ini, yaitu *least significant bit (LSB).* Ada dua pendekatan yang penulis lakukan untuk memperoleh informasi-informasi atau pengumpulan data ini diantaranya adalah:

a. Studi Pustaka

   Studi pustaka merupakan metode yang dilakukan untuk menemukan dan mengumpulkan data atau informasi kasus dari referensi-referensi terkait. Referensi-referensi ini dapat berupa buku-buku tentang steganografi, jurnal-jurnal atau tulisan penelitian steganografi, atau artikel-artikel yang membahas kasus yang sama dengan kasus dalam laporan ini.

b. Diskusi

   Merupakan metode yang dimaksudkan untuk berdiskusi dalam menyelesaikan permasalahan yang dibahas dalam laporan ini dengan orang-orang yang memahami tentang kasus pembahasan atau berdiskusi tentang masalah perancangan aplikasi yang akan dibangun.

3.3. Analisa dan Perancangan

Analisa dan perancangan merupakan metode yang dilakukan setelah pengumpulan terhadap data-data atau informasi mengenai kasus yang diangkat pada penelitian tugas akhir ini. Analisa berarti metode yang khusus untuk menganalisis masalah yang dapat dimulai dari analisa terhadap alur-alur proses steganografi, kemudian menganalisa model hingga rancang bangun aplikasi steganografi itu sendiri. Sementara perancangan berarti metode yang khusus digunakan untuk merancang hal-hal yang telah dianalisa dengan tujuan untuk memberikan kemudahan dan menyederhanakan suatu proses atau jalannya aliran data, perancangan terhadap model, dan merancang rancang bangun aplikasi ini. Perancangan ini meliputi perancangan model sistem yang teridiri dari: *Use Case Diagram, Class Diagram, State Chart Diagram, Activity Diagram, Sequence Diagram,* dan *Collaboration Diagram,* serta perancangan *interface* sistem yang terdiri dari *Prototype* dan struktur menu.

3.4. Implementasi dan Pengujian

Implementasi dan pengujian merupakan metode terakhir yang digunakan setelah analisa dan perancangan rancang bangun aplikasi selesai dilakukan. Metode ini akan menjelaskan tentang penerapan jalannya rancang bangun yang telah dianalisa dan dirancang. Aplikasi yang telah dirancang dan dianalisa selanjutnya diimplementasikan dan dilakukan pengujian untuk mengetahui tingkat keberhasilan aplikasi yang telah ada. Implementasi pengembangan aplikasi ini akan dikembangkan pada spesifikasi *hardware* dan *software* berikut:

1. Perangkat keras
   Processor : *Pentium(R) Dual-Core CPU T4400 @ 2.20GHz*
   Memori (RAM) : 1.00 GB
2. Perangkat Lunak
   Sistem operasi : *Windows 7 Profesional 32-bit Operating System*
   Bahasa pemrograman : Java
   Tools perancangan : Netbeans 6.9.1

Tools Audio Player : Windows Media Player
Tools Image Viewer : Windows Photo Viewer

Sementara untuk tahapan pengujian yang akan dilakukan pada aplikasi steganografi yang telah dibangun meliputi:

1. Pengujian *blackbox* untuk pengujian tingkah laku aplikasi yang telah dirancang.
2. Pengujian terhadap kualitas *stegofile* yang akan menggunakan formulasi *Peak Signal to Noise Ratio* (PSNR) pada persamaan (2.1) dan *Mean Sequare Error* (MSE) pada persamaan (2.2) untuk *file* gambar (bitmap), serta *Signal to Noise Ratio* (SNR) pada persamaan (2.3) untuk *file* suara (wave).
3. Pengujian lainnya terdiri dari pengujian terhadap tiga aspek penting steganografi, yaitu: kapasitas, kemanan, dan ketahanan. Pengujian terhadap ketiga aspek ini akan menggunakan *tools* tambahan, seperti:
   a. Menguji kapasitas *stegofile* akan dilakukan dengan cara membandingkan kapasitas (ukuran) *carrier file* dan *stegofile.*
   b. Untuk menguji kemanan *stegofile* akan menggunakan *free steganalist tools,* yakni StegSpy 2.1.
   c. Sementara untuk ketahanan *stegofile* akan menggunakan *tools* tambahan*,* yakni Microsoft Office Picture Manager *image stegofile* dan Cool Edit Pro 2.0 pada *audio stegofile*.

3.5. Kesimpulan dan Saran

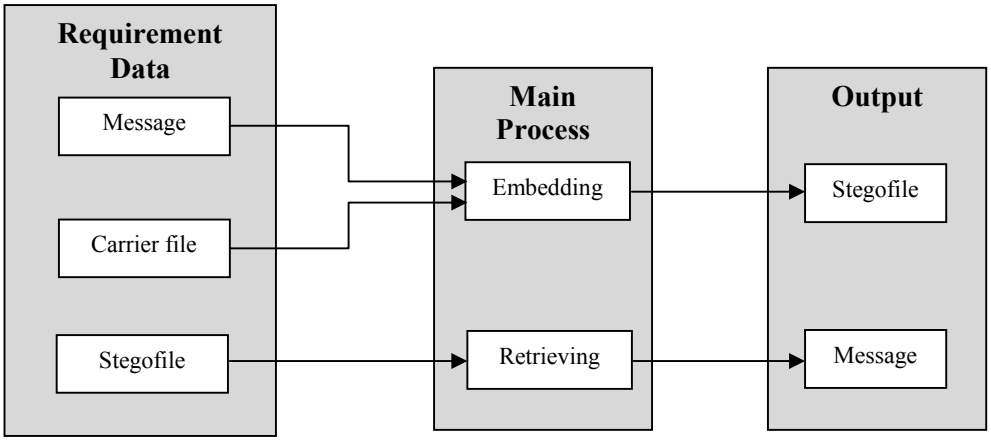
Tahapan kesimpulan dan saran merupakan akhir dari penelitian tugas akhir ini. Tahapan ini berisi tentang kesimpulan dari hasil-hasil penelitian dan pengujian yang telah dilakukan pada penelitian tugas akhir ini, yaitu perancangan rancangan bangun aplikasi steganografi dan berisi saran-saran membangun yang dapat dijadikan bahan penelitian ulang untuk meneliti dan merancang aplikasi steganografi yang lebih baik.

BAB IV

ANALISA DAN PERANCANGAN

4.1. Gambaran Umum Sistem

Pada dasarnya sebuah aplikasi steganografi memiliki proses yang dapat merahasiakan pesan (proses *embedding*) dan tentu cara untuk mengambil kembali pesan yang telah disembunyikan (proses *retrieving*). Begitu juga dengan rancang bangun aplikasi steganografi untuk penyisipan *text* dan *file* ke dalam *image* dan *audio file* dengan metode *least significant bit* (LSB) yang dibahas dalam laporan ini akan memiliki dua proses utama steganografi tersebut. Kebutuhan akan data-data inputan *(requirement data)* pada sebuah aplikasi steganografi seperti yang telah dijelaskan pada landasan teori juga menjadi kebutuhan utama pada rancang bangun aplikasi steganografi ini. Untuk lebih jelasnya dapat dilihat pada gambar 4.1 berikut.



Gambar 4.1. Gambaran Umum Sistem

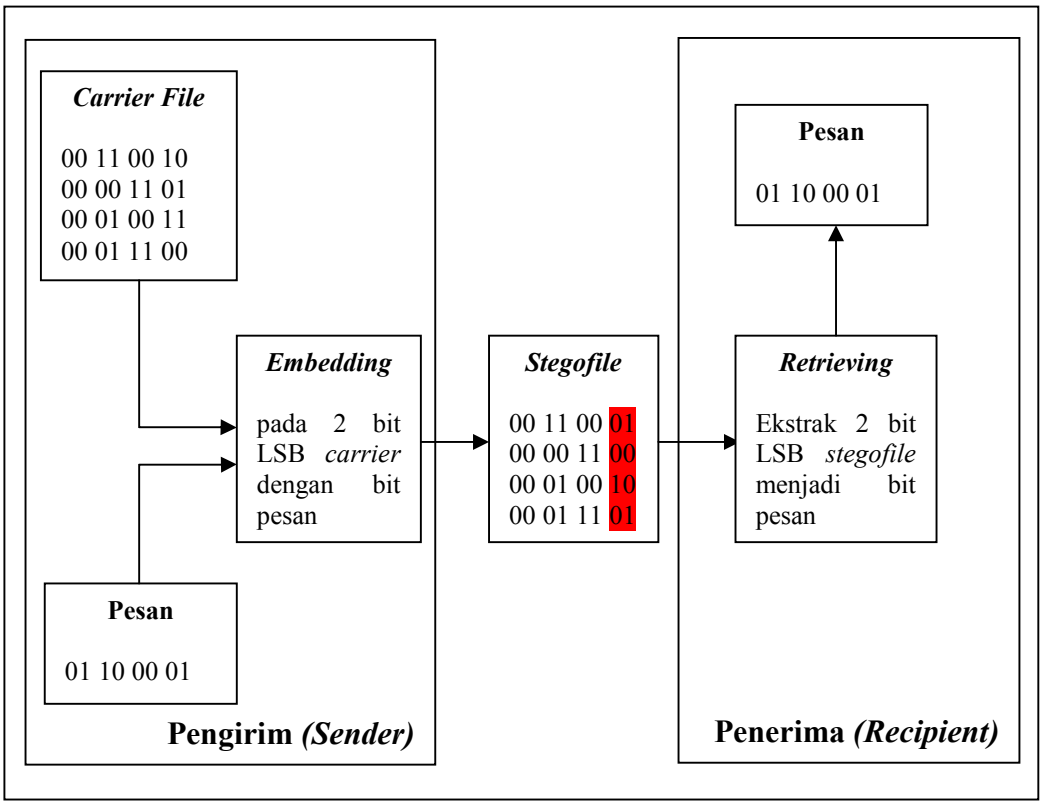
Berdasarkan gambar 4.1 dapat diketahui bahwa rancang bangun aplikasi steganografi ini memiliki proses, kebutuhan inputan data, dan output yang sama seperti aplikasi steganografi pada umumnya:

1. *Requirement data* merupakan kebutuhan inputan data yang dibutuhkan oleh sistem yang terdiri dari tiga jenis data yang dibedakan berdasarkan proses-proses utama *(main process)* steganografi, yaitu pesan rahasia *(message)* yang akan disisipkan dan media penampung *(carrier file)* yang menyatakan tempat dimana pesan rahasia akan disisipkan untuk proses *embedding*, serta data yang telah membawa pesan *(stegofile)* untuk proses *retrieving.*
2. *Main process,* menyatakan proses-proses utama yang terdapat pada rancang bangun aplikasi steganografi, yaitu proses penyembunyian atau penyisipan *(embedding)* dan proses pengambilan kembali pesan *(retrieving).* Pada proses inilah penerapan metode LSB itu terjadi yang dimulai dengan mengonversi *requirement data* dalam biner, kemudian dilakukan penyisipan bit-bit pesan ke bit-bit LSB medianya.
3. *Output* merupakan hasil dari *main process* yang terjadi pada sistem. Output dari e*mbedding* disebut dengan *stegofile*, sementara output dari *retrieving* adalah pesan rahasia yang tersembunyi di dalam *stegofile.*

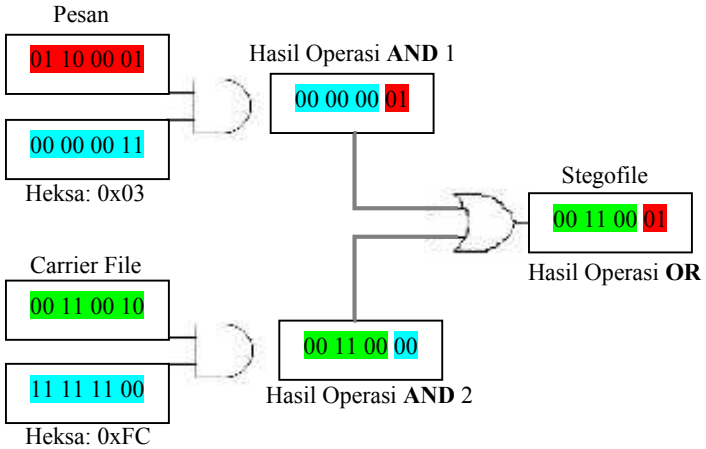
4.1.1. Gambaran Umum Analisis terhadap Metode LSB

Pada batasan masalah disebutkan bahwa penelitian ini akan dilakukan pada 2 bit LSB *carrier file* atau dengan kalimat lain dapat disebutkan bahwa bit-bit *carrier* yang akan dimodifikasi adalah bit ke-8 dan bit ke-7. Tujuannya adalah agar aplikasi steganografi yang akan dirancang ini dapat memberikan peningkatan *payload capacity* (jumlah pesan yang akan disisipkan pada sebuah *carrier file*). Karena semakin banyak jumlah bit LSB yang di modifikasi pada setiap byte-nya, maka akan semakin besar kapasitas pesan yang dapat disisipkan pada medianya.

Sebagai ilustrasi, gambar 4.2 merupakan gambaran umum proses metode LSB yang akan diterapkan pada sistem. Sementara untuk proses pertukaran atau modifikasi 2 bit LSB dengan bit-bit pesan dapat dilihat pada gambar 4.3.



Gambar 4.2. Gambaran Umum Metode LSB pada Sistem



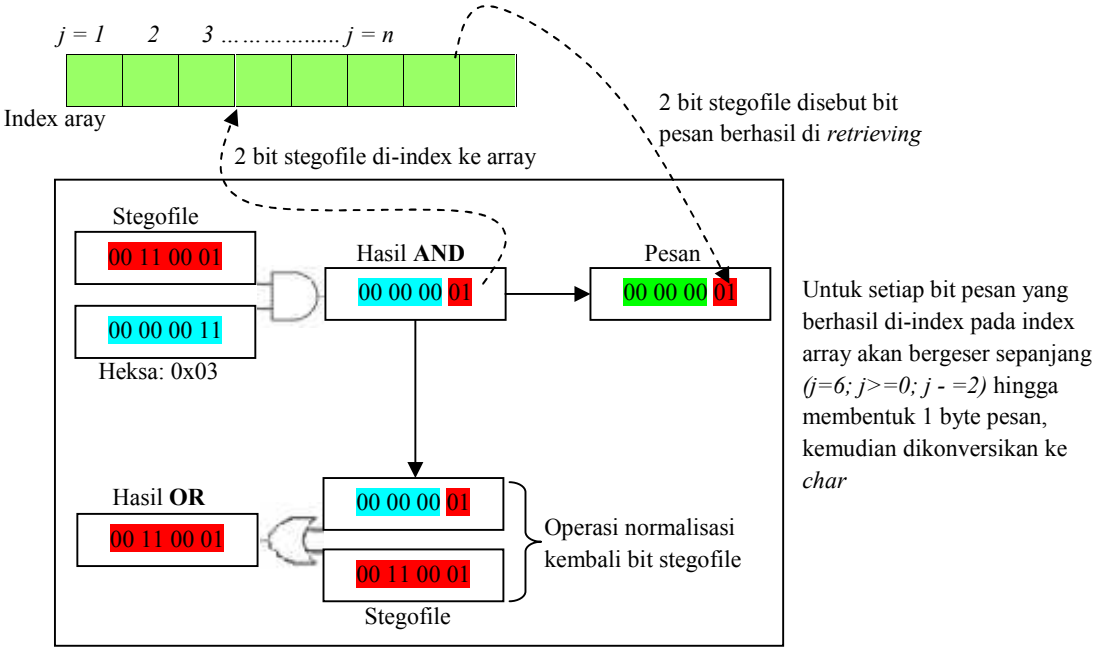
Gambar 4.3. Blok Diagram Metode LSB pada Sistem untuk Proses *Embedding*

Gambar 4.3 merupakan blok diagram metode LSB pada sistem untuk proses *embedding*. Proses pertukaran bit antara bit-bit pesan dengan 2 bit LSB *carrier*-nya dapat digambarkan seperti gambar 4.3 tersebut. Berdasarkan gambar 4.3 di atas dapat dijelaskan bahwa:

1. Hal yang pertama sekali perlu dilakukan untuk menyisipkan pesan pada 2 bit LSB *carrier*-nya adalah dengan mengonversikan pesan dan *carrier file* ke dalam bentuk biner.
2. Beberapa tahapan yang dibutuhkan sebelum pada akhirnya 2 bit LSB *carrier* dimodifikasi dengan bit pesan adalah:
    a. Pesan yang telah terkonversi dalam biner akan di operasikan dengan operasi AND terhadap heksa 0x03 untuk memberikan nilai 0 pada 6 bit pesan terkiri dan tidak mengubah 2 bit pesan terkanannya. Karena 2 bit pesan terkanan yang akan disisipkan pada 2 bit LSB *carrier*-nya terlebih dahulu, maka 6 bit pesan lainnya harus di beri nilai 0 agar tidak mengganggu nilai 6 bit pesan sesungguhnya.
    b. Agar 2 bit pesan terkanan tepat benar dapat disisipkan pada 2 bit LSB *carrier*-nya, maka bit *carrier* harus dioperasikan dengan operasi AND terhadap heksa 0xFC terlebih dahulu untuk memberikan nilai 0 pada 2 bit LSB *carrier* dan tidak memodifikasi 6 bit lainya.
    c. Karena 6 bit terkiri *carrier* tidak diharapkan untuk termodifikasi pada proses pertukaran bit, sementara 6 bit pesan terkanan sementara memiliki nilai 0, maka yang harus dilakukan selanjutnya adalah melakukan operasi OR antara hasil operasi AND bit pesan pada poin (b) dengan hasil operasi AND bit *carrier* pada poin (c) untuk memodifikasi 2 bit LSB *carrier* dengan 2 bit pesan terkanan pertama. Sehingga dapat diperoleh 6 bit *carrier* terkiri yang tidak mengalami modifikasi, sementara 2 bit LSB-nya termodifikasi oleh bit pesan.

3. Setelah empat tahapan di atas selesai dan berhasil memodifikasi atau menyisipkan 2 bit pesan terkanan pertama pada 2 bit LSB *carrier* pertama, maka selanjutnya adalah memodifikasi 2 bit pesan terkanan kedua dengan cara menambahkan nilai 0 di awal bit pesan. Karena setelah disisipkannya 2 bit pesan terkanan pertama pada *carrier*-nya, 6 bit pesan lainnya akan bergeser ke kanan dan memberikan 2 ruang kosong tak bernilai pada awal bit. Oleh sebab itu, perlu ditambahkan nilai 0 pada ruang kosong tersebut yang indikasinya sama dengan ruang kosong, tapi bernilai. Begitu terus tahapannya proses pertukaran bit ini terjadi hingga semua bit pesan berhasil disisipkan pada 2 bit LSB *carrier*-nya *(looping)*.

Sementara untuk gambaran umum proses pengambilang kembali pesan *(retrieving)* dapat dilihat pada gambar 4.4. Proses *retrieving* ini dimulai dengan membuat *index aray* yang panjangnya sebanyak n index sepanjang jumlah bit pesan yang disisipkan pada medianya dan berfungsi untuk tempat pengumpulan bit-bit pesan yang diambil dari medianya *(stegofile)*. Untuk lebih jelasnya, berikut adalah gambar 4.4 yang menggambarkan secara umum proses *retrieving* ini.



Gambar 4.4. Blok Diagram Metode LSB pada Sistem untuk Proses *Retrieving*

Berdasarkan gambar 4.4 dapat dijelaskan kembali bahwa proses *retrieving* yang akan terjadi pada rancang bangun aplikasi steganografi yang sedang dibahas ini adalah sebagai berikut:

1. *Stegofile* yang merupakan *requirement data* pada proses *retrieving* ini dikonversi terlebih dahulu ke dalam biner.
2. Kemudian, membuat *index aray* yang panjangnya sama dengan jumlah kapasitas bit pesan yang disisipkan pada medianya. *Index aray* ini berfungsi untuk tempat pengumpulan bit-bit pesan yang diambil dari medianya *(stegofile).*
3. Bit *stegofile* selanjutnya akan dioperasikan dengan operasi AND terhadap nilai heksa 0x03 untuk memberikan nilai 0 pada 6 bit *stegofile* terkiri dan tidak mengubah 2 bit *stegofile* terkanannya. Karena 2 bit *stegofile* terkanan merupakan bit-bit yang dianggap sebagai bit-bit pesan yang sebelumnya telah disisipkan pada proses *embedding,* maka 6 bit *stegofile* lainnya harus di beri nilai 0 agar tidak mengganggu nilai 6 bit *stegofile* sesungguhnya.
4. Selanjutnya, 2 bit tersebut akan diindex ke index array sesuai dengan posisinya. Proses atau langkah ini akan terus berulang sepanjang jumlah bit pesan belum terkumpul pada *index array.*
5. Bit-bit yang terkumpul akan dikonversikan ke-*char* hingga menjadi susunan karakter
6. Selanjutnya, bit hasil operasi AND sebelumnya harus dinormalisasi kembali menjadi bit *stegofile* agar bit-bit yang terubah sebelumnya tidak merusak *stegofile* itu sendiri.

4.1.2. Kebutuhan Data Inputan *(Requirement Data)*

Dari gambaran umum sistem atau aplikasi steganografi di atas telah dijelaskan bahwa rancang bangun aplikasi steganografi ini membutuhkan tiga jenis data inputan yang dibedakan berdasarkan proses-proses utama sistem, yaitu pesan rahasia yang akan disisipkan, media penampungnya *(carrier file),* dan data atau *file* yang telah membawa pesan *(stegofile).* Adapun pesan rahasia yang akan disembuyikan pada medianya ada dua jenis, yaitu pesan berupa teks dan pesan

berupa *file.* Kemudian media atau *carrier* yang direkomendasikan untuk rancang bangun aplikasi steganografi ini juga ada dua jenis, yaitu *file* gambar berformat bitmap (*.bmp) dan *file* suara berformat wave (*.wav). Sementara *stegofile* akan bergantung pada jenis *carrier file* yang digunakan.

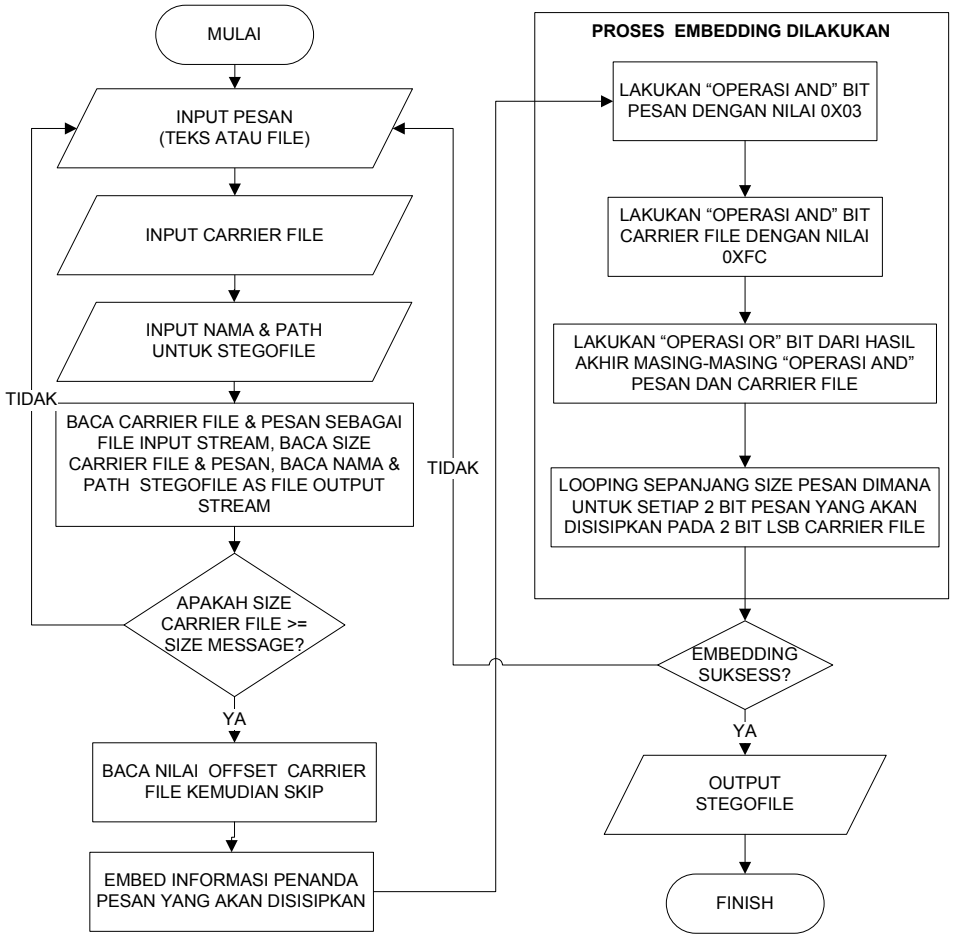
Beberapa alasan pemilihan rekomendasi jenis *carrier file* ini dapat dijelaskan sebagai berikut:

1. Sebelumnya telah dijelaskan pada latar belakang bahwa penambahan *file* suara sebagai *carrier file* pada rancang bangun aplikasi ini diharapkan dapat memberikan alternatif baru kepada *user* yang ingin merahasiakan pesan dengan ukuran besar. Hal ini disebabkan karena kebanyakan penelitian dan aplikasi steganografi hanya meneliti atau membatasi pada satu jenis *carrier* saja. Selain itu, pada umumnya *file* suara memiliki kapasitas yang lebih besar dari pada *file* gambar.
2. Pada dasarnya kedua jenis *carrier* ini merupakan *file* digital yang tidak mengalami kompresi dimana secara kapasitas memiliki kapasitas yang relatif besar. Berbeda dengan *file* digital yang mengalami kompresi yang cenderung berkapasitas lebih kecil. Namun, pada intinya pemilihan *carrier* pada *file* digital *uncompressed* ini lebih disebabkan karena jenis *carrier* ini memiliki struktur data yang hampir sama, sehingga mudah dalam mengimplementasikan keduanya pada sebuah perancangan aplikasi.
3. Adanya kecenderungan pemilihan metode steganografi yang tepat pada *file-file* terkompresi yang disesuaikan dengan struktur data masing-masing *file* kompresi itu sendiri.

4.1.3. Proses Penyembunyian Pesan *(Embedding)*

Metode LSB pada proses penyembunyian pesan *(embedding)* terjadi apabila *file-file* yang dibutuhkan telah terinput. Pada saat itu, proses *embedding* telah siap dijalankan dengan cara membaca bit-bit pertama pada *carrier file*. Bit-bit pertama ini adalah bit-bit yang termasuk bit-bit penting, seperti bit-bit *file header, image header, color table* (pada *file* gambar bitmap) dan bit-bit yang termasuk bit-bit RIFF atau *Chunk* (pada *file* WAVE atau baca hal. II-13 sampai II-

17 pada bab landasan teori). Bit-bit ini sering dideklarasikan sebagai nilai *OFFSET carrier file.* Secara umum, proses *embedding* untuk rancang bangun aplikasi steganografi ini dapat dilihat pada gambar 4.5.



Gambar 4.5. *Flowchart* Proses *Embedding*

Berdasarkan *flowchart* proses *embedding* yang ditunjukkan pada gambar 4.4 dapat dijelaskan bahwa:

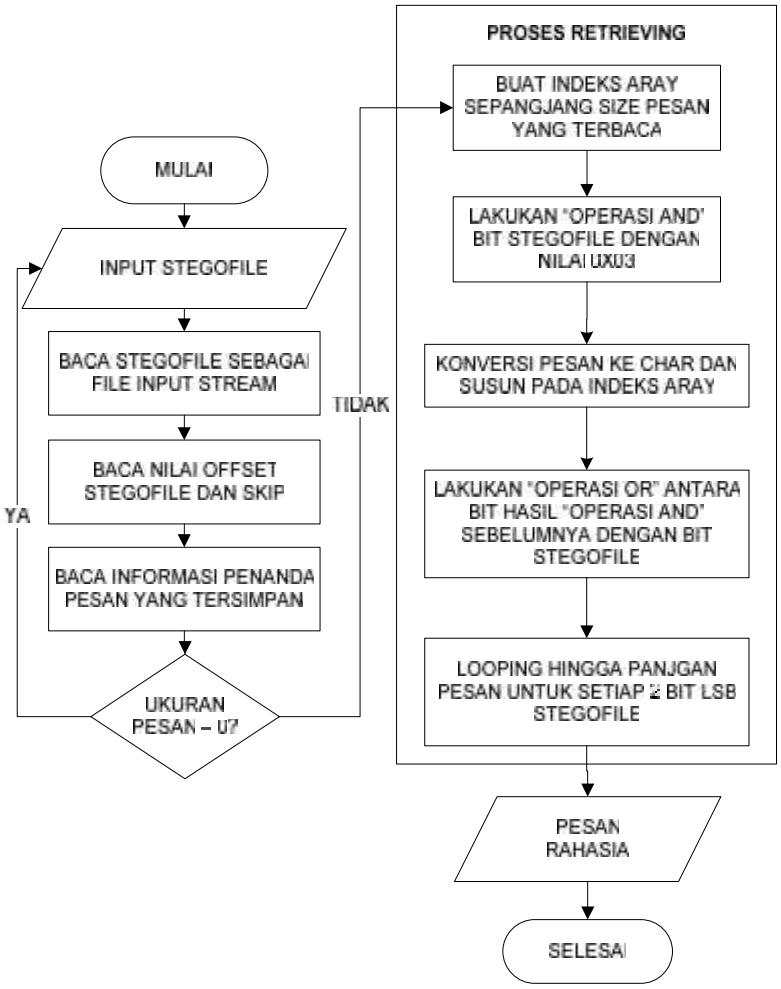
1. Penyisipan pesan rahasia dimulai dengan menginputkan *requirement data* ke dalam sistem, yang terdiri dari pesan rahasia yang akan disembunyikan, *carrier file,* dan direktori atau posisi dimana output ingin disimpan.
2. Selanjutnya, aplikasi akan membaca pesan rahasia dan ukurannya, kemudian membaca *data stream* dari *carrier file* dan pesan yang

dideklarasikan sebagai *file input stream,* serta membaca *data stream* dari nama dan *path stegofile* yang dideklarasikan sebagai *file output stream.* Kemudian dilakukan validasi terhadap ukuran *carrier file* dan pesan yang akan disembunyikan. Jika ukuran *carrier file* lebih besar atau sama dengan ukuran pesan rahasia, maka proses akan berlanjut pada proses pembacaan nilai *OFFSET carrier file bits* kemudian melangkahinya *(skip).* Namun, jika ukuran *carrier file* lebih kecil dari ukuran pesan rahasia atau pesan lebih besar dari medianya, maka proses akan kembali ke langkah pertama dan meminta inputan *carrier file* yang lebih besar.

3. *Embed* informasi penanda pesan yang akan disisipkan pada *carrier.* Bagian ini dibedakan berdasarkan pada tipe pesan yang akan disisipkan pada *carrier.* Kalau tipe pesan yang disisipkan adalah teks, maka yang dijadikan penanda adanya pesan di dalam sebuah *carrier* adalah hanya berupa ukuran dari pesan yang akan disisipkan, sementara pada tipe pesan berupa file, maka yang dijadikan penanda selain ukuran pesan yang akan disisipkan adalah *filename* dari pesan termasuk tipe file yang disisipkan dan data atau file itu sendiri. Bagian ini menjadi bagian yang sangat penting ketika penerima ingin mengambil *(retrieving)* pesan.
4. Proses *embedding* dilakukan pada bit-bit setelah nilai OFFSET dengan cara:
    a) Melakukan "Operasi AND" bit-bit pesan dengan nilai heksa 0x03 (dalam biner: 00000011).
    b) Langkah selanjutnya adalah melakukan "Operasi AND" bit-bit *carrier file* dengan nilai heksa 0xFC (dalam biner: 11111100).
    c) Lakukan "Operasi OR" untuk masing-masing hasil akhir "Operasi AND" bit pesan dan bit *carrier file.* "Operasi OR" ini merupakan langkah atau proses pertukaran bit-bit pesan dengan 2 bit LSB *carrier*-nya.
5. Diperoleh output berupa *stegofile* (*file* baru yang telah membawa pesan).
6. Selesai.

4.1.4. Proses Ekstraksi Pesan *(Retrieving)*

Proses *retrieving* berfungsi untuk memperoleh kembali pesan rahasia yang telah disembunyikan. Secara detil, proses ini ditunjukan pada gambar 4.6.



Gambar 4.6. *Flowchart* Proses *Retrieving*

Berdasarkan alur proses *retrieving* yang ditunjukkan pada gambar 4.6 dapat dijelaskan bahwa:

1. *Retrieving* terjadi apabila *requirement data* berupa *stegofile* telah diinputkan pada sistem.
2. Sistem akan membaca *data stream* dari *stegofile* yang diinputkan.
3. Kemudian membaca *OFFSET* dari *stegofile* dan melompati nilai *OFFSET*.

4. Membaca informasi penanda pesan yang disisipkan pada *stegofile.*
5. Membaca barisan bit-bit *stegofile* untuk menemukan informasi ukuran pesan yang tersimpan. Jika ukuran pesan adalah 0, maka hal ini mengindikasikan bahwa *requirement data* yang diinputkan tidak mengandung pesan (bukan *stegofile).* Namun, jika ditemukan informasi ukuran pesan yang tersimpan pada *stegofile,* maka:
    a) Buat indeks aray sepanjang informasi ukuran pesan yang diperoleh.
    b) Lakukan operasi AND bit *stegofile* dengan nilai heksa 0x03 (dalam biner: 00000011).
    c) indekskan ke indeks aray yang telah disusun sebelumnya dan susun bit-bit yang diperoleh hingga genap menjadi 1 byte (8 bit). Setelah itu, konversikan ke dalam char.
    d) Lakukan operasi OR antara bit hasil dari operasi AND sebelumnya dengan bit *stegofile* itu sendiri yang bertujuan untuk proses normalisasi bit *stegofile.*
6. Output yang dihasilkan adalah pesan rahasia.
7. Selesai.

4.2. Perancangan Sistem

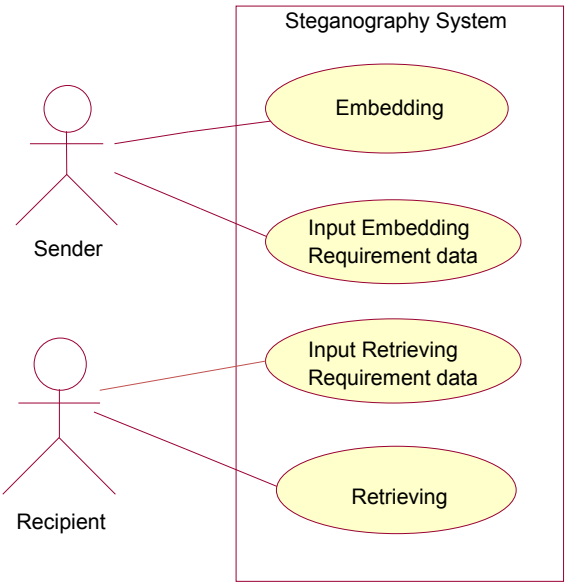
Setelah dilakukan beberapa tahapan dalam analisa sistem, maka dapat dilakukan beberapa perancangan sistem steganografi ini. Perancangan-perancangan yang akan dijelaskan dalam laporan ini meliputi perancangan model dalam bentuk UML *(Unified Modeling Language)* dengan menggunakan metode USDP *(Unified Software Development Process)* yang terdiri dari *Use Case Diagram, Class Diagram, State Chart Diagram, Activity Diagram, Sequence Diagram,* dan *Collaboration Diagram* dan perancangan *interface* sistem yang terdiri dari perancangan *prototype* dan struktur menu.

4.2.1. Perancangan Model Sistem

Seperti yang telah dijelaskan sebelumnya bahwa aplikasi steganografi yang akan dirancang nantinya akan menggunakan bahasa pemrograman berbasis objek. Oleh sebab itu, ada beberapa diagram yang akan dimodelkan yang menggambarkan rancangan sistem ini, yang terdiri dari *use case diagram, class diagram, state chart diagram, activity diagram, sequence diagram,* dan *collaboration diagram*.

4.2.1.1. Perancangan *Use Case Diagram*

Use case diagram merupakan diagram terpenting dalam pemodelan sistem berbasis objek yang pemodelannya dilakukan dengan pendekatan metode USDP yang akan menjadi pedoman terhadap perancanaan model-model sistem yang lainnya. Berikut ini adalah gambar 4.7 yang merupakan visualisasi dari *use case diagram* untuk sistem steganografi ini yang kemudian akan dijelaskan beberapa penjelasan mengenai *use case* secara detil pada table-tabel *use case specification*, yaitu dari tabel 4.1 sampai dengan tabel 4.4.



Gambar 4.7. *Use Case Diagram* Rancang Bangun Aplikasi Steganografi

Dari gambar 4.7 dapat diketahui bahwa rancang bangun aplikasi steganografi ini hanya terdiri dari 4 *use cases* dan 2 *actors*. Secara berurut ke-4 *use cases* yang terdapat pada aplikasi steganografi ini adalah dimulai dari proses *input embedding requirements data*, *input retrieving requirement data,* kemudian proses penyisipan pesan *(Embedding),* dan proses pengambilan kembali pesan *(Retrieving)*. Sementara aktor-aktor yang terlibat langsung terhadap sistem adalah pengirim pesan *(sender)* dan penerima pesan *(Recipient).*

Tabel 4.1. Spesifikasi *Use Case Input Embedding Requirement Data*

Aktor utama	Pengirim *(Sender)*
Kondisi awal	-
Kondisi Akhir	Sistem memvalidasi ukuran *carrier file* adalah minimal 4 kali lebih besar dari pesannya
Main success scenario	1. Dimulai ketika aktor memilih proses *embedding.* 2. Sistem akan menampilkan dua pilihan jenis pesan yang akan disisipkan, teks atau *file.* 3. Kebutuhan dasar kedua jenis pesan adalah sama, yaitu pesan yang akan disisipkan *(message),* media penampungya *(carrier file),* dan direktori penyimpanan output *(stegofile)*. 4. Sistem memvalidasi *size* antara *message* yang akan disisipkan dengan *carrier*-nya. Kondisinya adalah ukuran *carrier* harus minimal 4 kali lebih besar dari pesannya. 5. Jika *carrier file* memenuhi kondisi, maka sistem dapat melakukan proses *embedding.*
Exception	1. Jika *size* pesan lebih besar dari *carrier file,* maka muncul *message box* yang menyatakan bahwa pesan membutuhkan *carrier* yang lebih besar.

Tabel 4.2. Spesifikasi *Use Case Input Retrieving Requirement Data*

Aktor utama	Penerima *(Recipient)*
Kondisi awal	-
Kondisi Akhir	Data yang terinput adalah data *stegofile*
Main success scenario	1. Dimulai ketika aktor memilih proses *retrieving.* 2. Sistem akan menampilkan dua pilihan jenis pesan yang akan dapat diambil kembali, teks atau *file.* 3. Kebutuhan dasar kedua jenis pesan adalah sama, yaitu data yang mengandung atau membawa pesan *(stegofile).* 4. Sistem memvalidasi data yang terinput. 5. Jika *stegofile* yang terinput, maka sistem dapat melakukan proses *retrieving.*
Exception	1. Jika yang terinput bukan *stegofile,* maka akan muncul *message box* yang berisi "Error".

Tabel 4.3. Spesifikasi *Use Case Embedding*

Aktor utama	Pengirim *(Sender)*
Kondisi awal	Ukuran *requirement data* yang terinput telah tervalidasi
Kondisi Akhir	Output berupa *file* baru pembawa pesan *(Stegofile)*
Main success scenario	1. Dimulai ketika aktor melakukan aksi *embedding* terhadap sistem. 2. Sistem akan mengonversi data-data inputan dalam bentuk *data stream.* 3. Kemudian membaca nilai *OFFSET* dan melewatinya *(skip)*. 4. Sistem akan melakukan proses penyisipan pesan ke *carrier*-nya dengan cara mengganti 2 bit LSB *carrier* dengan bit-bit pesan secara bertahap, yaitu melakukan operasi AND antara bit pesan dengan

	nilai nilai heksa 0x7F, hasilnya di lakukan operasi AND lagi dengan nilai heksa 0x03, melakukan operasi AND bit *carrier* dengan nilai heksa 0xFC, hasilnya akan di operasikan secara OR dengan hasil akhir operasi AND pada bit pesan, begitu seterusnya hingga seluruh bit pesan berhasil disisipkan pada *carrier*-nya. 5. Jika berhasil, maka akan menghasilkan output berupa *stegofile.*
Exception	1. Jika pertukaran bit tidak berhasil, maka akan muncul pesan bahwa *embedding* terhadap pesan gagal.

Tabel 4.4. Spesifikasi *Use Case Retrieving*

Aktor utama	Penerima *(Recipient)*
Kondisi awal	*Requirement data* yang terinput telah tervalidasi
Kondisi Akhir	Pesan diterima oleh *recipient*
Main success scenario	1. Dimulai ketika aktor melakukan aksi *retrieving* terhadap sistem. 2. Sistem akan mengonversi data-data inputan dalam bentuk *data stream.* 3. Kemudian membaca nilai *OFFSET* dan melewatinya *(skip)*. 4. Sistem membaca informasi ukuran pesan yang disisipkan. 5. Buat indeks aray sepanjang informasi ukuran pesan yang diperoleh. 6. Lakukan operasi AND bit *stegofile* dengan nilai heksa 0x03, susun bit-bit yang diperoleh hingga genap menjadi 1 byte (8 bit). Setelah itu,

	konversikan ke dalam char dan indekskan ke indeks aray yang telah disusun sebelumnya. Lakukan operasi OR antara bit hasil operasi AND sebelumnya dengan bit *stegofile* kembali. 7. Output yang dihasilkan adalah pesan rahasia yang berhasil di ambil kembali.
Exception	-

4.2.1.2. Perancangan *Class Diagram*

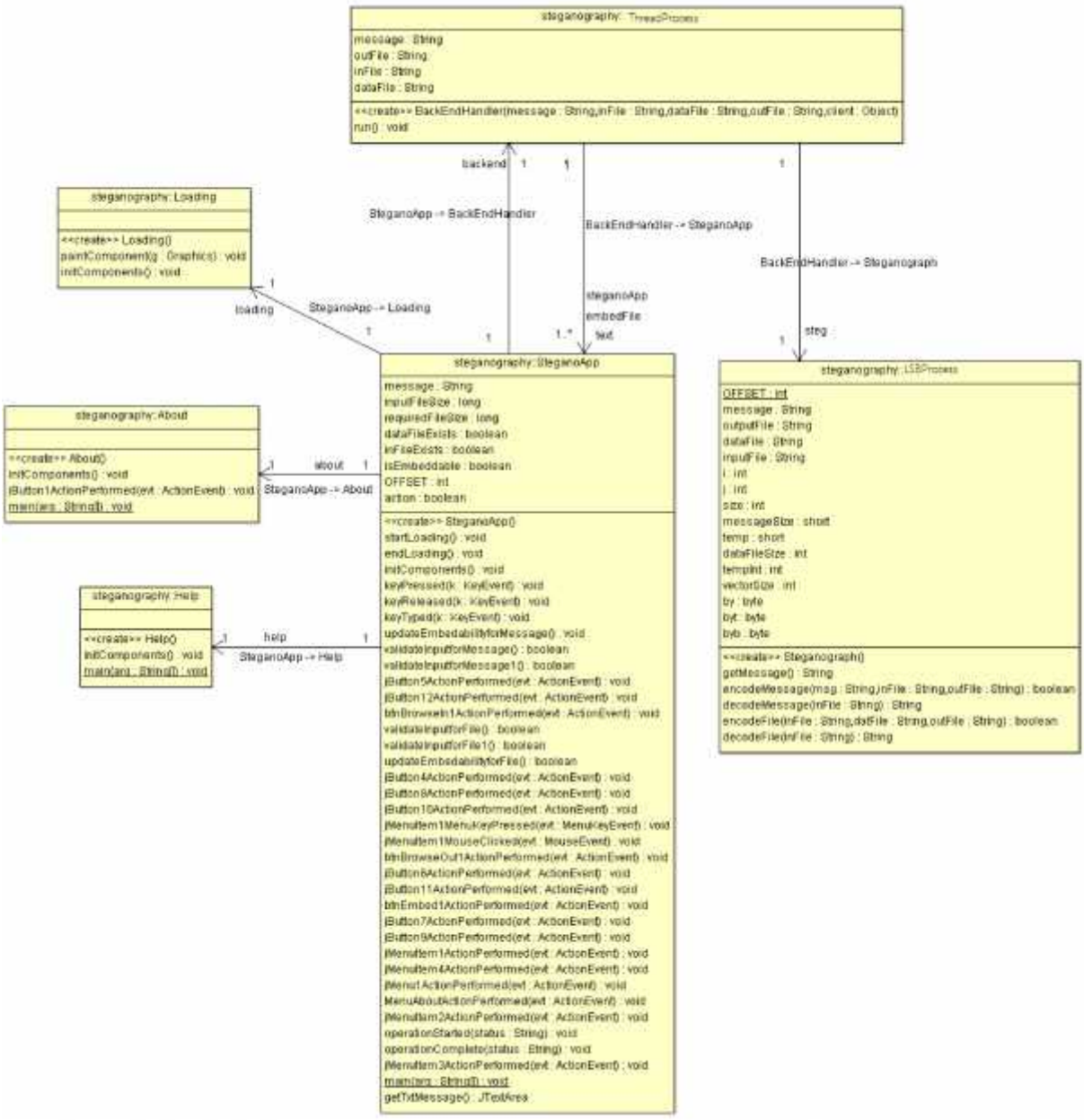
Class diagram digunakan untuk mendeskripsikan jenis-jenis objek dalam sistem dan berbagai macam hubungan statis yang terdapat dalam sistem tersebut. Gambar 4.8 merupakan rancangan *class diagram* steganografi yang dibahas dalam laporan ini. Berdasarkan gambar 4.8 dapat diketahui bahwa aplikasi steganografi ini akan dibangun dari enam kelas rancangan, yaitu kelas *SteganoApp,* kelas *Loading,* kelas *ThreadProcess,* kelas *LSBProcess,* kelas, *Help,* dan kelas *About.*

Namun, dari keenam rancangan kelas yang ada, pada prinsipnya hanya akan ada tiga kelas saja yang menjadi kelas-kelas utama dari pembangunan aplikasi steganografi ini. Adapun ketiga kelas utama yang menjadi kelas-kelas pembangun rancang bangun aplikasi steganografi ini, yaitu:

1. Kelas *SteganoApp* yang merupakan kelas dimana tempat perancangan *interface* sistem secara umum. Kelas ini terdiri dari beberpa fungsi, seperti: sebagai kelas pendirian *interface* sistem yang akan berhadapan langsung dengan pengguna *(user)* baik pengirim maupun penerima pesan, untuk menginisialisasikan semua kegiatan sistem, serta menginisialisasi dan memvalidasi data-data inputan untuk diproses selanjutnya oleh sistem.
2. Kelas *ThreadProcess* yang berfungsi sebagai *Thread* sistem. *Thread* ini menjadi penanda terjadinya atau berjalannya kegiatan sistem. Adapun beberapa aktivitas yang akan di lakukan adalah memberikan informasi awal dan berakhirnya operasi *(embedding* atau *retrieving)* dan memberian

informasi-informasi tentang berhasil tidaknya sistem/apliasi melakukan aktivitasnya.

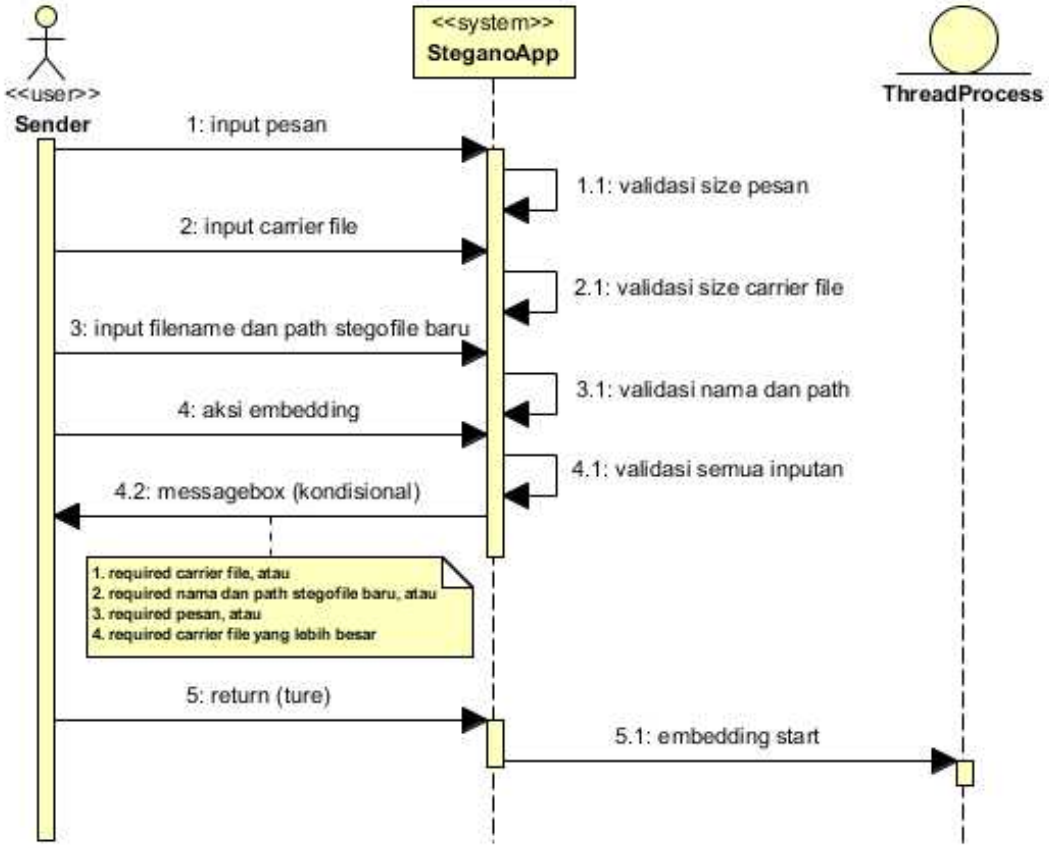
3. Kelas *LSBProcess* yang berfungsi sebagai tempat terjadinya proses pertukaran-pertuaran bit (kelas implementasi metode LSB). Aktivitas yang ditangani ialah mengangani pertukaran bit pada proses *embedding* dan *retrieving.*



Gambar 4.8. *Class Diagram* Rancang Bangun Aplikasi Steganografi

4.2.1.3. Perancangan *Sequence Diagram*

Gambar 4.9 berikut merupakan diagram *sequence diagram* yang dilakukan oleh pengirim terhadap sistem, tepatnya pada proses penginputan *embedding requirement data*. Adapun data atau *file* yang perlu diinputan oleh aktor meliputi: pesan yang akan disembunyikan, yaitu inputan berupa huruf untuk menyembunyikan pesan dan inputan berupa *file* untuk menyembunyikan *file carrier file, path* dan nama baru untuk *stegofile*. Sementara untuk diagram-diagram lainnya, seperti *state chart diagram, activity diagram, collaboration diagram*, termasuk *sequence diagram* lainnya akan dijelaskan pada **Lampiran A.**



Gambar 4.9. *Sequence Diagram* Rancang Bangun Aplikasi Steganografi

4.2.2. Perancangan *Pseudocode* Sistem

Berdasarkan kelas-kelas rancangan yang telah dirancang untuk aplikasi steganografi ini, maka dapat disimpulkan bahwa proses-proses utama dari aplikasi ini akan terpusat pada satu kelas, yaitu pada kelas *LSBProcess.* Algoritma 4.1 berikut adalah rancangan algoritma *(pseudocode)* aplikasi steganografi ini untuk proses *embedding* (penyisipan pesan).

```
PROGRAM Embedding
{Program untuk menyisipkan pesan}
KONSTANTA
      OFFSET = 64;
DEKLARASI
      message, outputfile, inputfile : string;
      in, data : DataInputStream;
      out : DataOutputStream;
      i, j, size,n, dataFileSize, tempInt, vectorSize: integer;
      messageSize, temp: short;
      by,byt, byb: byte;
ALGORITMA
{inisialisasi nilai i, j, size sebagai 0}
      i ← 0
         j ← 0
            size ← 0
{konversi data-data inputan ke dalam data stream}
      read (in, data)
            read (out)
for i ← 0 to n do i <= OFFSET  {skip nilai offset carrier file}
      writeByte out readByte in
      messageSize ← (short) message.length
endfor
{Proses inti Embedding pesan}
for i ← 0 to n do i<messageSize
      byt ← (byte) message.charAt(i)
      byt& ← 0x7F
            for j ← 6 to j ← -2 do j>=0
            by ← byt
            by>> ← j
            by AND 0x03
            {menulis sisa byte pesan ke output file}
            byb ← readByte in
            byb& ← 0xFC
            byb OR by
            writeByte(byb)out return endfor endfor
```

Algoritma 4.1. Algoritma Proses *Embedding*

Hal yang perlu diingat kembali adalah bahwa untuk memulai proses *embedding* dan proses *retrieving* pada aplikasi ini adalah dibutuhkannya proses penginputan *reuirement data* dimasing-masing proses. Sementara rancangan *pseudocode* untuk proses *retrieving* dapat dilihat pada rancangan Algoritma 4.2.

```
PROGRAM Retrieving
{Program untuk pengambilan pesan kembali}
KONSTANTA
      OFFSET = 64;
DEKLARASI
      message, outputfile, inputfile, inFile : string;
      in, data : DataInputStream; out : DataOutputStream;
      i, j, size,n, dataFileSize, tempInt, vectorSize: integer;
      messageSize, temp: short; by,byt,byb: byte;
      flag: Boolean; mesg: char;
ALGORITMA
{inisialisasi nilai i, j, size sebagai 0}
      i ← 0
         j ← 0
            size ← 0
{inisialisasi variable baru khusus retrieving}
      inputFile ← inFile
          flag ← true
              mesg ← null
{konversi data-data inputan ke dalam data stream}
      read (in)
      messageSize ← 0{diperoleh ukuran pesan}
{skip nilai offset carrier file}
for i ← 0 to n do i <= OFFSET
      readByte in
endfor
if messageSize <= 0 return endif
{ciptakan karakter aray baru bersasarkan ukuran messagesize
      mesg ← messageSize
            for i ← 0 to n do i<messageSize
                  by= 0;
            for j ← 6 to j ← -2 do j>=0
                  byt ← readByte in
                  byt AND 0x03
                  byt<<= j
                  by OR byt;
                  mesg [i] ← (char) (((char) by)& 0x00FF)
                  return endfor endfor endfor
```

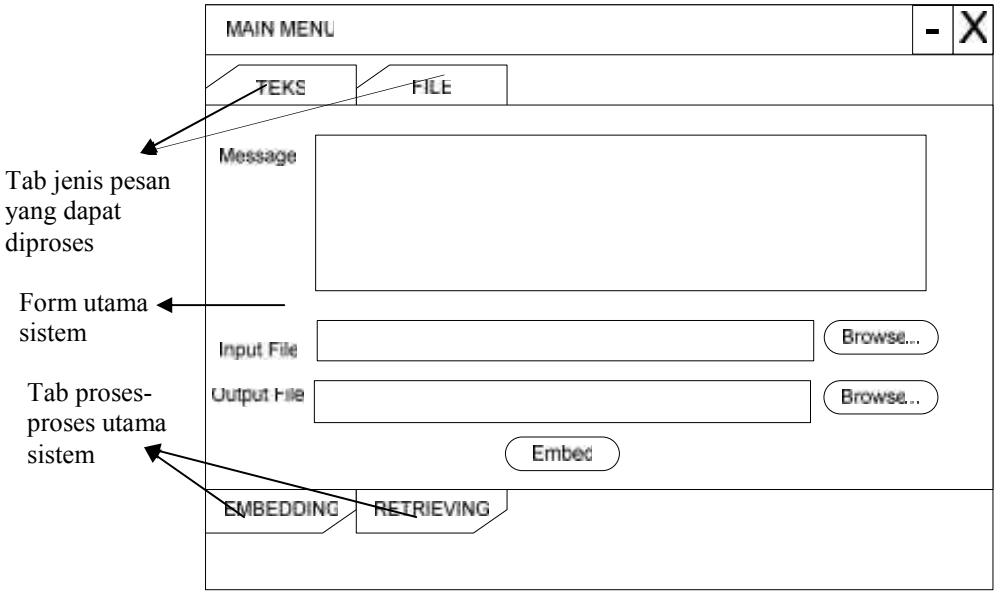
Algoritma 4.2. Algoritma Proses *Retrieving*

4.2.3. Perancangan *Interface* Sistem

Perancangan terhadap *interface* sistem merupakan sebuah rancangan pada sisi antarmuka sistem yang sedang dikembangkan. Disebabkan karena sebuah steganografi memiliki dua proses utama, yaitu *embedding* dan *retrieving,* kemudian rancang bangun aplikasi steganografi ini akan memiliki kemampuan menyisipkan dua jenis pesan, yaitu pesan berupa teks dan pesan berupa *file,* maka rancangan *interface* sistem akan dibangun dengan cara mengelompokkan kedua proses dan jenis pesan ke dalam tab-tab menu.

4.2.3.1. Perancangan *Interface* pada Proses *Embedding*

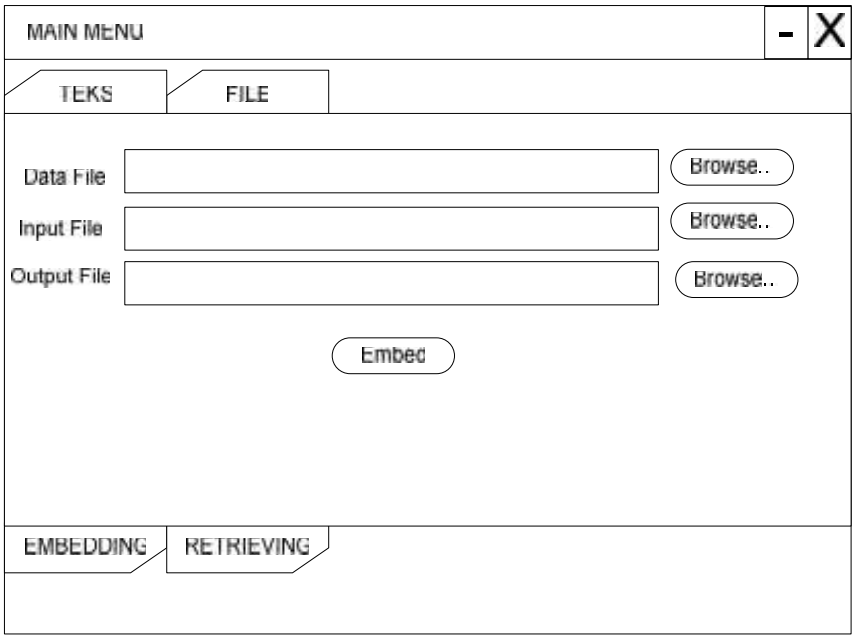
Sebelumnya telah dijelaskan bahwa *interface* sistem ini akan dibagi ke dalam tab-tab proses utama dan jenis pesan yang akan diproses. Hal tersebut dapat lebih jelas ditunjukkan pada gambar 4.10 berikut yang merupakan gambar terhadap rancangan *interface* pada aplikasi steganografi ini.



Gambar 4.10. *Interface* untuk *Embedding* Pesan berupa Teks

Berdasarkan gambar 4.10 dapat dijelaskan bahwa untuk memulai penggunaan sistem ini seorang pengguna akan dihadapkan pada tab *embedding* tepatnya pada form utama dari jenis pesan yang dapat disembunyikan, yaitu teks.

Dengan demikian, jika pengguna ingin menyembunyikan atau meng-*embed* pesan berupa *file,* maka pengguna harus memilih *Tab File* pada tab jenis pesan. *Form* untuk jenis pesan berupa *file* dapat dilihat pada gambar 4.11. Alur utamanya adalah, seorang pengguna akan diminta untuk menginputkan data-data yang dibutuhkan untuk diproses dengan menekan *Button Browse.* Selanjutnya, untuk memulai proses *embedding,* maka pengguna harus menekan *Button Embed.* Hal yang sama juga dilakukan pada proses *embedding* untuk *file* sebagai jenis pesannya.

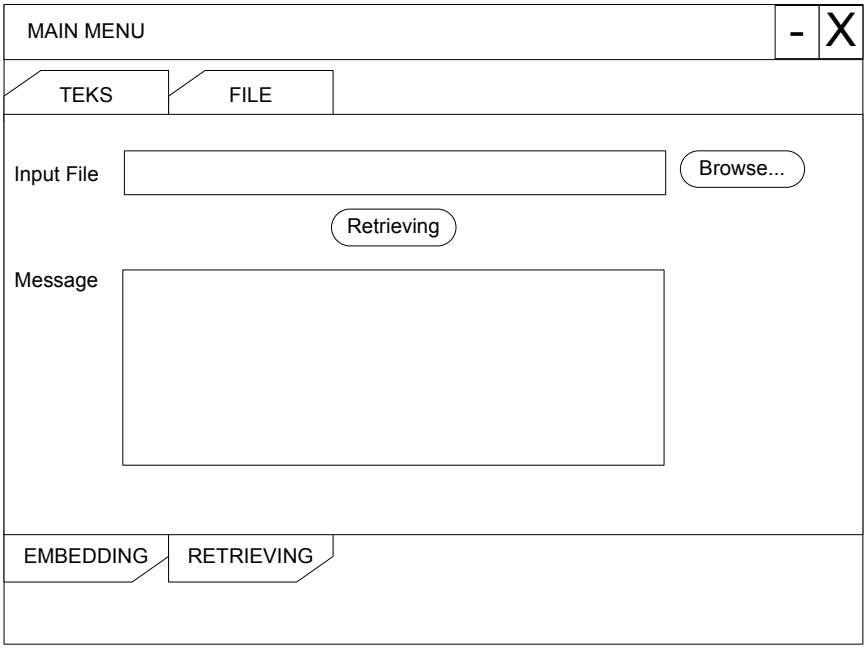


Gambar 4.11. *Interface* untuk *Embedding* Pesan berupa *File*

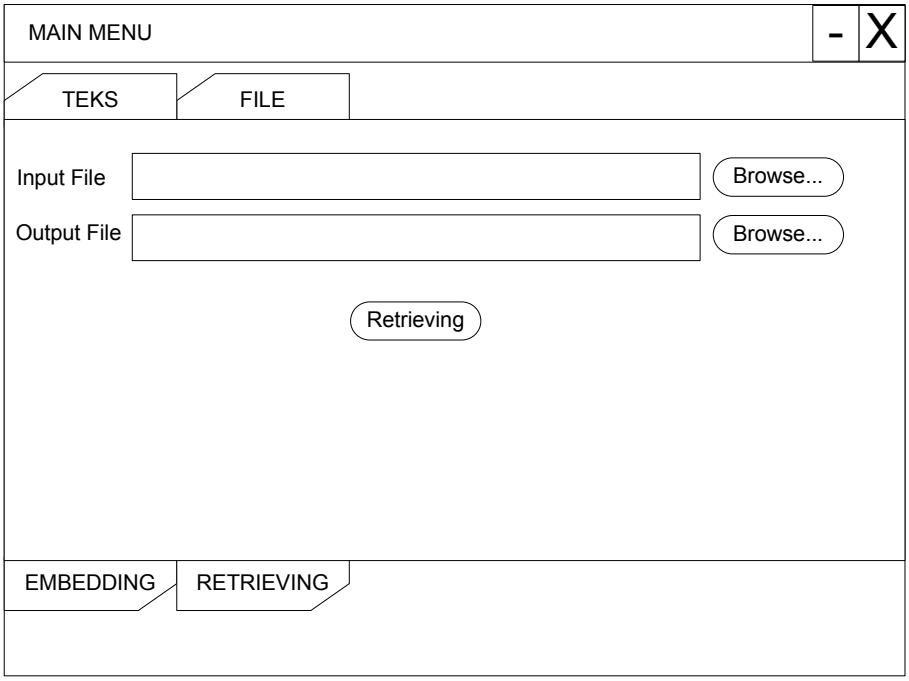
4.2.3.2. Perancangan *Interface* pada Proses *Retrieving*

Berikut ini merupakan ilustrasi dari rancangan *interface* untuk proses *retrieving.* Tidak berbeda dengan proses *embedding* sebelumnya, pada proses *retrieving* ini seorang pengguna akan diminta untuk memilih *Tab Retrieving* untuk memulai prosesnya. Kemudian nantinya seorang pengguna akan dihadapkan pada *form* utama dari jenis pesan, tepatnya *form* jenis pesan berupa teks dan ketika

pengguna ingin mengekstrak pesan berupa *file* maka pengguna harus memilih *Tab File.* Hal ini dapat dilihat pada gambar 4.12 dan 4.13.



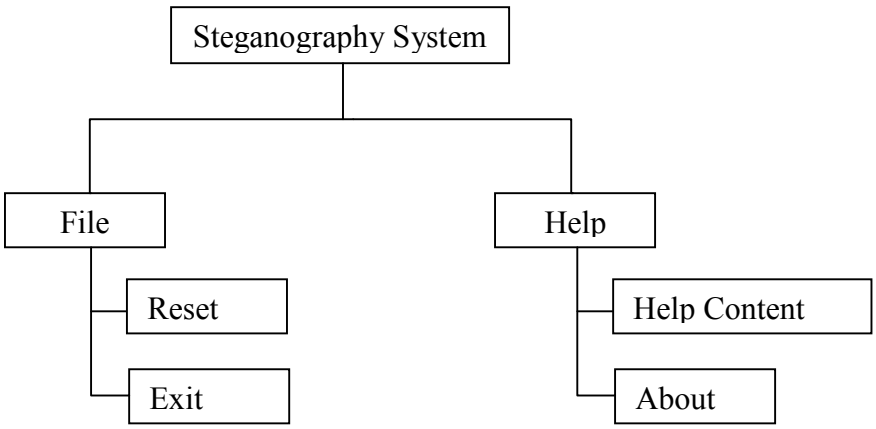
Gambar 4.12. *Interface* untuk *Retrieving* Pesan berupa Teks



Gambar 4.13. *Interface* untuk *Retrieving* Pesan berupa *File*

4.3.4 Perancangan Struktur Menu

Aplikasi ini dirancang terdiri dari dua menu utama, yaitu File Menu, dan Help Menu dimana keduanya memiliki dua submenu untuk mendukung fungsinya, yaitu submenu Reset dan submenu Exit (pada menu File), serta submenu Help dan submenu About (pada menu Help). Untuk lebih jelasnya, struktur menu ini dapat dilihat pada gambar 4.14.



Gambar 4.14. Struktur Menu Rancang Bangun Aplikasi Steganografi

Submenu *Reset* berfungsi untuk me-reset atau menghapus semua kegiatan yang dilakukan oleh sistem, seperti menghapus semua data-data inputan, dan menghapus semua validasi terhadap ukuran data inputan. Sementara submenu *Exit* berfungsi untuk keluar dari sistem. Submenu *Help Content* berisi tentang tata cara penggunaan sistem dan deskripsi cara penggunaan sistem, sementara submenu *About* berisi tentang perancang aplikasi

BAB V

IMPLEMENTASI DAN PENGUJIAN

5.1. Tahapan Implementasi

Tahapan implementasi merupakan tahapan dimana sistem informasi yang telah dirancang, dianalisa, dan dibangun, lalu diuji kelayakannya untuk selanjutnya dioperasikan sebagaimana mestinya sesuai dengan fungsi dan kelayakannya. Berikut ini akan dijelaskan tentang pengimplementasian dari analisis dan perancangan yang telah dilakukan terhadap aplikasi steganografi ini.

5.1.1. Batasan Implementasi

Mengacu pada penjelasan yang telah dijelaskan pada bab pendahuluan bahwa rancangan aplikasi steganografi ini pada dasarnya untuk meneliti kemampuan *file* gambar berformat bitmap (*.bmp) dan *file* suara berformat (*.wav) dalam menyembunyikan teks dan *file* digital pada 2 bit LSB medianya. Karena banyaknya *file* digital yang ada saat ini tidak memungkinkan bagi penulis untuk melakukan implementasi dan pengujian pada semua *file* digital tersebut. Oleh sebab itu, penulis memberikan beberapa batasan implementasi terhadap rancang bangun aplikasi steganografi ini sebagai berikut:

1. Banyaknya file digital yang ada saat ini, maka implementasi hanya akan dilakukan pada beberapa file digital, seperti:
    a. File dokumen: *.doc, *.ppt, *.xls, *.pdf
    b. File gambar: *.jpeg, *.png, *.gif, *.bmp
    c. File kompres: *.rar, *.zip
    d. File suara: *.wav, *mp3
    e. File installer: *.exe
    f. File video: *.avi

2. Pada tahapan pengujian, akan dilakukan juga pengujian pada beberapa jenis format file digital berikut sebagai media penampungnya *(carrier file),* seperti: *.jpeg, *.gif. *.png (pada file gambar) dan *.mp3 (pada file suara).

5.1.2. Lingkungan Operasional

Komponen-komponen yang dibutuhkan untuk menerapkan aplikasi ini antara lain berupa komponen *hardware* dan *software.* Seperti yang telah dijelaskan pada analisa kebutuhan sebelumnya, maka berikut ini adalah lingkungan operasional yang merupakan lingkungan tempat aplikasi ini diimplementasikan dan digunakan oleh penulis dengan spesifikasi sebagai berikut:

1. Perangkat keras

   Processor : *Pentium(R) Dual-Core CPU T4400 @ 2.20GHz*

   Memori (RAM) : 1.00 GB

2. Perangkat Lunak

   Sistem Operasi : *Windows 7 Profesional 32-bit Operating System*

   Bahasa Pemrograman : Java

   Tools Perancangan : Netbeans 6.9.1

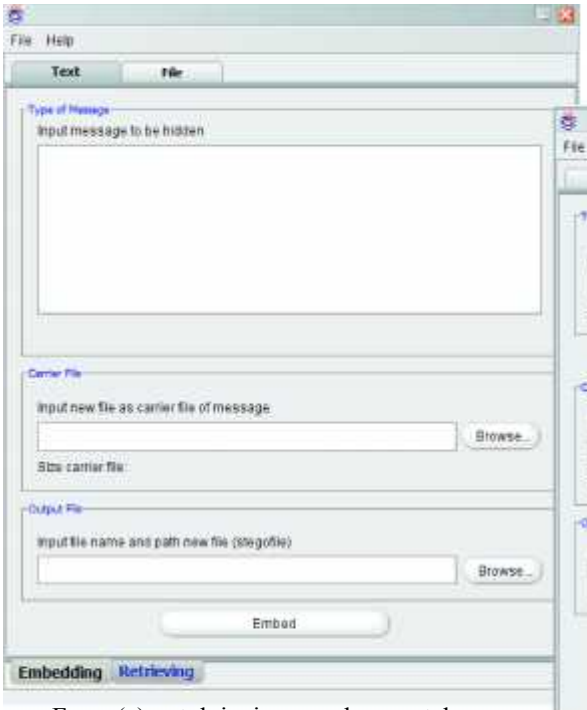
   Tools Audio Player : Windows Media Player

   Tools Image Viewer : Windows Photo Viewer

   Pemodelan UML : Visual Paradigm 8.1 Enterprise Edision

5.1.3. Implementasi *Interface* Sistem

Setelah tahap analisa dan perancangan selesai dilakukan, maka dilanjutkan dengan tahap implementasi sistem dari hasil analisa yang telah diperoleh dan mengimplementasikan hasil perancangan *interface* yang telah dibuat. Berikut ini akan dijelaskan mengenai hasil implementasi dari rancang bangun aplikasi steganografi ini, yaitu implementasi *interface* pada proses *embedding* dan *retrieving.*



Form (a) untuk jenis pesan berupa teks

Form (b) untuk jenis pesan berupa file

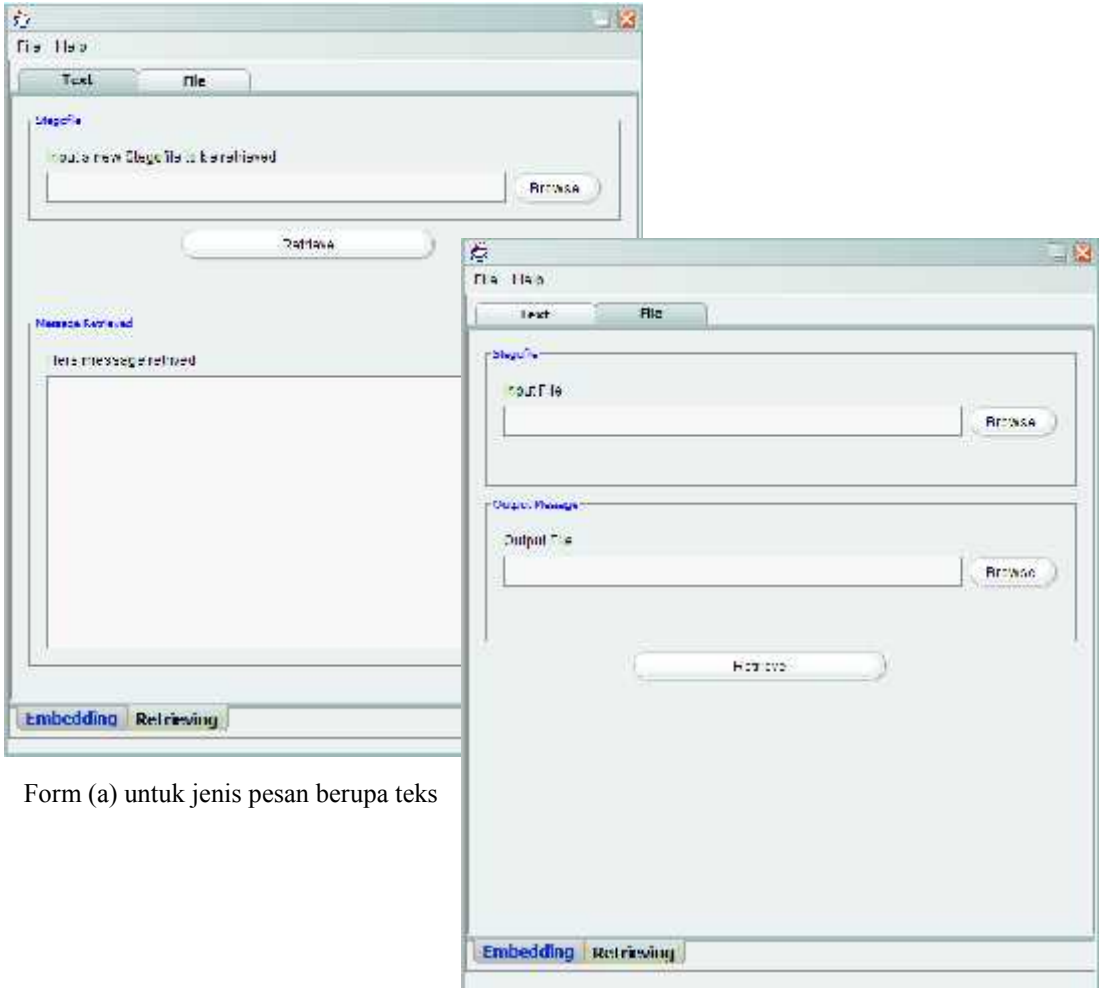
Gambar 5.1. Hasil Implementasi *Interface* Proses *Embedding*

Dari gambar 5.1 di atas dapat dilihat bahwa aplikasi ini memiliki perancangan *interface* yang dikelompokkan berdasarkan tab-tab proses utama dari steganografi dan tab-tab jenis pesan yang dapat diproses oleh aplikasi dimana rancangan ini telah dijelaskan sebelumnya pada bab analisa dan perancangan sistem. *Interface* pada gambar 5.1 tersebut yang akan muncul pertama sekali ketika *user* menjalankan sistem ini.

Pada gambar 5.1 dapat dilihat ada beberapa form yang harus diisi oleh *user* sebelum melakukan proses *embedding.* Seorang *user* diminta untuk menginputkan data-data yang dibutuhkan untuk disisipkan, yaitu menginputkan pesan berupa teks atau *file*, *carrier file,* dan nama serta *path/ directory* peletakkan output. Caranya adalah dengan menekan tombol ”Browse”. Apabila semua data telah terinput dengan benar, maka *user* dapat memulai proses penyisipan pesannya dengan menekan tombol ”Embed”. Selanjutnya akan muncul *message box* yang

menginformasikan kepada *user* bahwa *carrier file* telah berhasil disisipi pesan yang tersimpan pada *path* dimana *user* telah menginputkannya sebelumnya.

Hal-hal yang dijelaskan di atas tidak berbeda dengan yang terjadi pada proses *retrieving.* Untuk memulai proses *retrieving* seorang *user* harus menginputkan data-data atau *file-file,* seperti *file* yang telah disisipi pesan *(stegofile)* dengan cara menekan tombol ”Browse”. Dan untuk memulai proses, maka *user* dapat menekan tombol ”Retrieve”. Namun, pada pesan berupa *file*, *user* dapat mengarahkan dimana pesan akan terekstraksi (lihat gambar 5.2).



Form (a) untuk jenis pesan berupa teks

Form (b) untuk jenis pesan berupa file

Gambar 5.2. Hasil Implementasi *Interface* Proses *Retrieving*

5.2. Tahapan Pengujian

Pada bab Landasan Teori telah dijelaskan bahwa pada steganografi ada tiga aspek dasar steganografi berbeda yang sering dijadikan objek penelitian dan pengujian pada kasus steganografi. Adapun ketiga aspek steganografi tersebut ialah: kapasitas *(capacity)*, keamanan *(security)*, dan ketahanan *(robustness)*. Oleh sebab itu, penulis juga akan melakukan pengujian yang sama pada kasus ini dan ditambah beberapa pengujian lain, seperti pengujian fungsionalitas sistem dengan metode *blackbox* dan pengujian terhadap kualitas sistem.

5.2.1. Pengujian *Blackbox* pada Sistem Steganografi

Pengujian sistem dilakukan untuk memeriksa kekompakan atau kinerja antar komponen sistem yang diimplementasikan. Tujuan utama dari pengujian sistem adalah untuk memastikan bahwa elemen-elemen atau komponen-komponen dari sistem telah berfungsi sesuai dengan yang diharapkan. Salah satu metode pengujian jenis ini dikenal dengan pengujian *blackbox.* Adapun hasil dari pengujian ini dapat dilihat pada tabel 5.1.

Tabel 5.1. Hasil Pengujian Sistem dengan Metode *Blackbox*

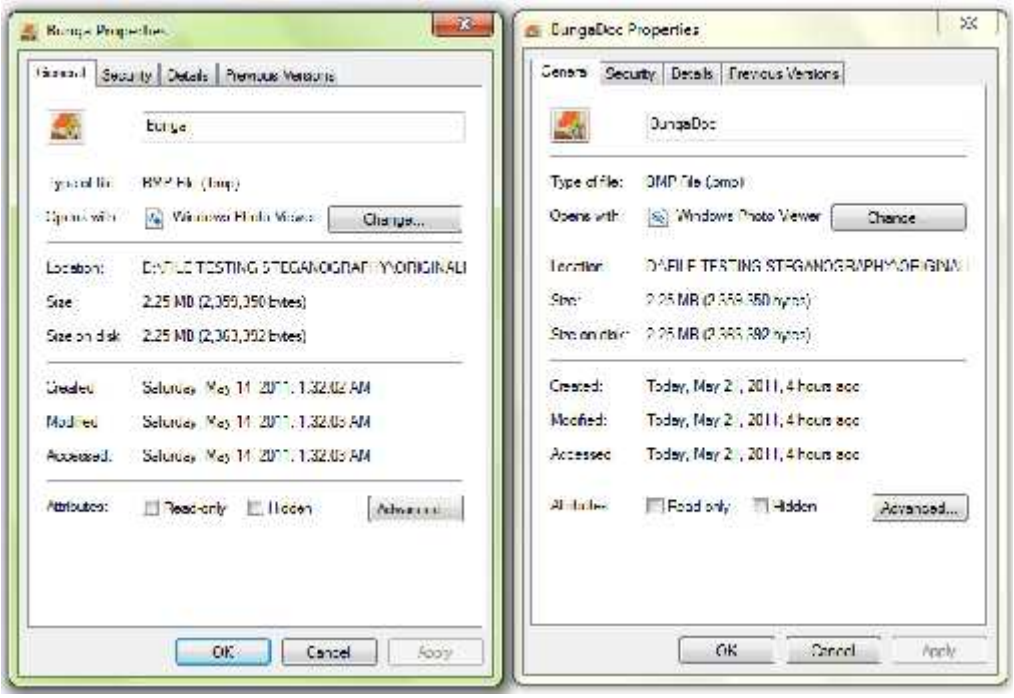
No	Objek Pengujian	Hasil yang diharapkan	Hasil
1.	*Button Browse* pada seluruh form sistem untuk penginputan data (kondisi untuk open/ save)	Memanggil jendela *filechooser* untuk proses penginputan data baik pada form inputan untuk menyisipkan maupun mengambil pesan	Benar
		Textfield dapat menuliskan inputan data	Benar
2.	*Button Embed* baik untuk menyembunyikan teks maupun file	Muncul *Message box* apabila salah satu data inputan belum diinputkan yang meminta *user* untuk menginputkan data yang belum terinput	Benar

Tabel 5.1. Hasil Pengujian Sistem dengan Metode *Blackbox* (lanjutan)

No	Objek Pengujian	Hasil yang diharapkan	Hasil
		Muncul *message box* yang menginformasikan *user* untuk menginputkan *carrier file* yang memiliki ukuran lebih besar dari pesan yang akan disisipkan	Benar
3.	*Button Retrieving* baik teks maupun file	Muncul *message box* yang menginformasikan bahwa file yang diinputkan bukan *stegofile* yang membawa pesan	Benar
		Muncul *message box* yang menginformasikan bahwa file yang dibutuhkan belum diinputkan	Benar
4.	*File Menu*	Submenu *Reset* (Ctrl+R) dapat menghapus isi form sebelumnya	Benar
		Submenu *Exit* (Alt+F4) keluar dari sistem	Benar
5.	*Help Menu*	Menampilkan jendela *help content* ketika submenu *Help Content* dipilih	Benar
		Menampilkan jendela *about* ketika submenu *About* dipilih	Benar

5.2.2. Pengujian Sistem pada Aspek Kapasitas Steganografi

Pada dasarnya, pengujian ini dilakukan untuk mengetahui kapasitas atau ukuran dari sebuah *file* sebelum membawa pesan *(carrier file)* dengan *file* setelah membawa pesan *(stegofile).* Caranya adalah dengan membadingkan ukuran atau kapasistas dari kedua *file.* Kondisi yang baik untuk sistem steganografi adalah apabila ukuran atau kapasitas *file* baik sebelum ataupun setelah disisipi/ membawa pesan adalah sama (tidak berubah). Ilustrasi pengujian ini terlihat pada gambar 5.3 berikut ini.



Gambar 5.3. Ilustrasi Pengujian Kapasitas, *file* sebelum disisipi pesan (kiri) dan *file* setelah disisipi pesan (kanan)

Gambar 5.3 di atas merupakan cara penulis dalam menguji aspek kapasitas sistem. Pada gambar kiri yang merupakan *file* sebelum disisipi pesan (*file* asli) dengan *filename* "Bunga.bmp" berkapasitas 2.25 MB memiliki ukuran yang sama dengan *file* setelah disisipi pesan berupa *file* berformat *.doc (kanan) dengan *filename* "BungaDoc.bmp" dimana ukuran pesan yang disisipkan adalah sebesar 91 Kb. Hal ini mengindikasikan bahwa aplikasi ini secara kapasitas merupakan aplikasi yang baik karena tidak terjadi perubahan kapasitas pada kedua file. Hasil pengujian terhadap *stegofile* lainnya secara detil dapat dilihat pada tabel 5.2.

Berdasarkan tabel 5.2 dapat dikutahui bahwa selain *file* gambar bitmap dan *file* suara wave, pengujian sistem pada aspek kapasitas ini juga dilakukan terhadap format-format gambar dan suara lainnya, seperti pada format gambar *.jpeg, *.gif, dan *.png, serta pada format suara *.mp3. Kemudian dari hasil yang ditunjukkan pada tabel tersebut dapat diketahui bahwa rancang bangun aplikasi steganografi ini baik dari aspek kapasitas yang tidak mengubah kapasitas asli *carrier file* baik sebelum disisipi pesan maupun setelah disisipi pesan.

Tabel 5.2. Hasil Pengujian pada Aspek Kapasitas Steganografi

<table>
<tr><th rowspan="3">NAMA FILE
(carrier file)</th><th colspan="2" rowspan="2">UKURAN
Carrier File</th><th colspan="16">UKURAN PESAN YANG DISISIPKAN</th></tr>
<tr><th colspan="14">Jenis Pesan Berupa File (kB)</th><th colspan="2">Pesan (Teks)</th></tr>
<tr><th>Sebelum</th><th>Sesudah</th><th>doc</th><th>xls</th><th>ppt</th><th>pdf</th><th>jpg</th><th>gif</th><th>png</th><th>bmp</th><th>exe</th><th>avi</th><th>mp3</th><th>wav</th><th>zip</th><th>rar</th><th>Uji I</th><th>Uji II</th></tr>
<tr><td>Bunga.bmp</td><td>2.25 MB</td><td>2.25 MB</td><td>91</td><td>116</td><td>493</td><td>424</td><td>514</td><td>380</td><td>354</td><td>495</td><td colspan="3">Kartini.bmp</td><td>86</td><td>297</td><td>488</td><td>3.04 Kb</td><td>6.82 Kb</td></tr>
<tr><td>Kartini.bmp</td><td>24.8 MB</td><td>24.8 MB</td><td colspan="8">Di uji pada Bunga.bmp, hasil OK</td><td>3076</td><td>6092</td><td>4567</td><td colspan="5">Bunga.bmp (OK)</td></tr>
<tr><td>Klip suara.wav</td><td>2.23 MB</td><td>2.23 MB</td><td>149</td><td>116</td><td>248</td><td>278</td><td>514</td><td>380</td><td>354</td><td>495</td><td colspan="3">Bismillah.wav (OK)</td><td>112</td><td>501</td><td>488</td><td>3.04 Kb</td><td>6.82 Kb</td></tr>
<tr><td>Bismillah.wav</td><td>37.1 MB</td><td>37.1 MB</td><td colspan="8">Di uji pada Klip suara.wav, hasil OK</td><td>3076</td><td>6092</td><td>4175</td><td colspan="5">Klip suara.wav (OK)</td></tr>
<tr><td>IMG_2595.JPG</td><td>2.36 MB</td><td>2.36 MB</td><td>136</td><td>28</td><td>167</td><td>132</td><td>36</td><td>77</td><td>61</td><td>31</td><td>496</td><td rowspan="8">Tidak dilakukan pengujian</td><td>293</td><td>545</td><td>5</td><td>9</td><td rowspan="2">3.04 Kb</td><td rowspan="2">5.42 Kb</td></tr>
<tr><td>IMG_2595.JPG</td><td>2.36 MB</td><td>2.36 MB</td><td>19.5</td><td>8.54</td><td>12</td><td>4.29</td><td>6</td><td>3.4</td><td>4.27</td><td>2.13</td><td></td><td>19.7</td><td>75.8</td><td>2.88</td><td>3.35</td></tr>
<tr><td>Earth-icon.png</td><td>354 KB</td><td>354 KB</td><td>19.5</td><td>8.54</td><td>12</td><td>4.29</td><td>5.41</td><td>3.4</td><td>4.27</td><td>2.13</td><td></td><td>19.7</td><td>75.8</td><td>2.88</td><td>3.35</td><td>3.04 Kb</td><td>165 B</td></tr>
<tr><td>Foto0113.png</td><td>4.60 MB</td><td>4.60 MB</td><td>19.5</td><td>8.54</td><td>12</td><td>4.29</td><td>5.41</td><td>3.4</td><td>4.27</td><td>2.13</td><td>496</td><td>19.7</td><td>75.8</td><td>2.88</td><td>3.35</td><td>3.04 Kb</td><td>165 B</td></tr>
<tr><td>earth.gif</td><td>380 KB</td><td>380 KB</td><td>19.5</td><td>8.54</td><td>12</td><td>4.29</td><td>5.41</td><td>3.4</td><td>4.27</td><td>2.13</td><td></td><td>19.7</td><td>75.8</td><td>2.88</td><td>3.35</td><td>3.04 Kb</td><td>165 B</td></tr>
<tr><td>animasi.gif</td><td>5.89 MB</td><td>5.89 MB</td><td>19.5</td><td>8.54</td><td>12</td><td>4.29</td><td>5.41</td><td>3.4</td><td>4.27</td><td>2.13</td><td>496</td><td>19.7</td><td>75.8</td><td>2.88</td><td>3.35</td><td>3.04 Kb</td><td>165 B</td></tr>
<tr><td>Tompi.mp3</td><td>3.82 MB</td><td>3.82 MB</td><td>19.5</td><td>8.54</td><td>12</td><td>4.29</td><td>5.41</td><td>3.4</td><td>4.27</td><td>2.13</td><td>496</td><td>19.7</td><td>75.8</td><td>2.88</td><td>3.35</td><td>3.04 Kb</td><td>6.82 Kb</td></tr>
</table>

Keterangan:

*** Daftar nama-nama file yang disisipkan pada masing-masing *carrier file* dapat dilihat pada **Lampiran G**

Tidak dilakukan pengujian

Stegofile baik (file masih dapat ditampilkan)

Stegofile rusak

Dari hasil implementasi dan pengujian pada aspek kapasitas steganografi sistem yang dilakukan pada beberapa *carrier file* yang menjadi media pembawa beberapa jenis *file* dan teks serta ukuran dimasing-masing pesan yang berbeda-beda, maka diperoleh kesimpulan seperti berikut:

1. Tidak terjadi perubahan kapasitas atau ukuran *carrier file* baik sebelum dan sesudah pesan disisipkan.
2. Secara kualitas yang seharusnya akan diuji dan dijelaskan pada subbab pengujian terhadap kulitas steganografi, pada pengujian pada aspek kapasitas ini dapat disimpulkan sementara bahwa untuk *carrier file* dengan format *file* *.bmp dan *.wav yang direkomendasikan sebagai media penampung pesan pada penelitian ini mampu membawa pesan baik berupa teks maupun pesan berupa *file* dimana terbukti mampu menyisipkan beragam *file* pesan.
3. Pada beberapa pengujian yang dilakukan terhadap *carrier file* dengan format *.jpeg, *.png, *.gif, dan *.mp3 dapat disimpulkan secara kapasitas terbukti baik, namun tidak dari segi kualitas. Hal ini dapat dilihat pada tabel 5.2 bahwa *carrier-carrier file* dengan format ini mengalami kerusakan visual setelah pesan berhasil disisipkan yang disebabkan karena berbedanya struktur data (format file) penyusun dengan *carrier file* bitmap dan wav.
4. *Carrier file* yang dijelaskan pada poin 3, masih mampu mengembalikan pesan yang tersimpan di dalamnya tanpa merusak sedikitpun pesan tersebut.
5. Khusus untuk *carrier file* berformat *.jpeg dan *.mp3 sebenarnya bisa menjadi rekomendasi penampung pesan, namun kapasitas yang dapat disisipkan sangat terbatas, terbukti pada tabel 5.2.

5.2.3. Pengujian Sistem pada Aspek Ketahanan Steganografi

Pengujian pada aspek ini bertujuan untuk mengetahui tingkat ketahanan *(robustness) stegofile* apabila dilakukan atau terjadi proses *editing* (penyerangan visual) terhadap *stegofile.* Pengujian pada aspek ini dilakukan dengan melakukan

beberapa perubahan *(editing)* terhadap *file* pembawa pesan, yaitu dengan modifikasi secara *visual* terhadap *stegofile*.

Pada *stegofile* berupa gambar dapat dilakukan pengujian dengan menggunakan aplikasi Microsoft Office Picture Manager, dimana aksi yang akan dilakukan terhadap *stegofile,* seperti: mengubah *brightness* gambar, melakukan *cropping* dan *rotating* terhadap gambar, serta *resize* gambar. Sementara pada *stegofile* berupa suara, pengujian dilakukan dengan menggunakan aplikasi Cool Edit Pro 2.0 dengan aksi pengujian, seperti *cut, inverse,* dan *reverse.* Dari aksi-aksi pengujian yang dilakukan terhadap *stegofile* tersebut, dapat diambil kesimpulan bahwa pesan yang tersembunyi pada *stegofile* akan cenderung rusak apabila terjadi perubahan bit pada *stegofile* tersebut. Untuk lebih jelasnya berikut adalah tabel 5.4 hasil pengujian ini.

Berdasarkan tabel 5.4 dapat diambil beberapa kesimpulan dari hasil pengujian pada aspek ketahanan ini diantaranya adalah:

1. Pengujian pada *stegofile* berupa citra dilakukan dengan beberapa aksi penyerangan, seperti *bright*, *crop, rotate,* dan *resize.* Sementara pada *stegofile* berupa suara aksi yang dilakukan adalah *cut, reverse,* dan *inverse.* Pernyataan "rentan terhadap serangan visual" pada point pertama dapat diketahui dari level pengujian yang dilakukan terhadap masing-masing *stegofile.* Pada *file* citra, level pengujian pada pixel (L) yang dilakukan bahkan sangat kecil, sebagai contoh pada aksi *crop* level pengujian yang dilakukan pada *file "*Bungatext.bmp" hanya sebesar 1 pixel pada sisi kiri citra saja dari 4 sisi yang ada sehingga hanya mengurangi pixel citra sebanyak 1 pixel dimana pixel semula adalah 1024 x 768 pixel menjadi 1023 x 768 pixel, tapi pesan yang yang disisipkan rusak ketika proses *retrieving* dilakukan. Kemudian pada *file* audio, contoh penyerangan visual yang dilakukan berupa *cut* terhadap *stegofile "*Klip suaratext.wav" mengakibatkan pesan yang tersembunyi pada *stegofile* juga rusak. Padahal panjang suara yang dipotong *(Length Cut Posisition* atau LCP) hanya pada 20 detik dari total *Length* suara yang ada.

Tabel 5.4. Hasil Pengujian pada Aspek Ketahanan Steganografi

Nama File	Tipe File *Carrier File*		Tipe Pesan			Aksi yang Dilakukan													
						Bright				Crop				Rotate				Compress			
	Image		F	T	UP	I		II		I		II		I		II		I		II	
	P	UF				L	P^1	L	P^1	L	P^1	L	P^1	L	P^1	L	P^1	L	UF^1	L	UF^1
Bungarar.bmp	1024 x 768	2.25 MB	✓		488 Kb	2	1024 x 768	1	1024 x 768	15	994 x 738	8	1020 x 764	+90	768 x 1024	-90	1024 x 768	26.2 Kb	120 Kb	5.01 Kb	25.8 Kb
Bungatext.bmp	1024 x 768	2.25 MB		✓	3.04 Kb	2	1024 x 768	1	1024 x 768	8	1020 x 764	1	1023 x 768	+90	768 x 1024	-90	1024 x 768	26.2 Kb	123 Kb	5.01 Kb	25.8 Kb
IMG_2595pdf2.jpeg	3888 x 2592	2.36 MB	✓		4.29 Kb	2	3888x2592	1	3888x2592	8	3882x2592	1	3887x2592	+90	2592x3882	-90	3882x2592	354 Kb	246 Kb	67.8 Kb	29.1 Kb
IMG_2595text.jpeg	3888 x 2592	2.36 MB		✓	3.04 Kb	2	3888x2592	1	3888x2592	8	3882x2592	2	3887x2592	+90	2592x3882	-90	3882x2592	354 Kb	246 Kb	67.8 Kb	29.1 Kb
Earth-iconteks2.png	512 x 512	354 KB		✓	165 B	1	512 x 512	2	512 x 512	8	508 x 508	4	510 x 510	+90	512 x 512	-90	512 x 512	19.6 Kb	44.8 Kb	3.76 Kb	20.7 Kb
earthteks2.gif	1024 x 768	380 KB		✓	165 B	2	1024 x 768	1	1024 x 768	8	1020 x 764	4	1022 x 766	+90	768 x 1024	-90	1024 x 768	84.4 Kb	380 Kb	16.1 Kb	94.6

Nama File	Tipe File *Carrier File*						Tipe Pesan			Aksi yang Dilakukan					
	Audio						F	T	UP	Cut				Reverse	Inverse
										I		II			
	SR	CH	RB	BR	Length	UF				LCP	$Lenght^1$	LCP	$Lenght^1$	UF^1	UF^1
Tompipdf.mp3	44100 Hz	Stereo	16 bit	128 Kbps	00:04:09:08	3.82 Mb	✓		4.29 Kb	00:00:02:14	00:04:06:24	00:00:00:20	00:04:08:18	3.80 Mb	3.80 Mb
Tompiteks2.mp3	44100 Hz	Stereo	16 bit	128 Kbps	00:04:09:08	3.82 Mb		✓	6.82 Kb	00:00:02:14	00:04:06:24	00:00:00:20	00:04:08:18	3.80 Mb	3.80 Mb
Klip suarabmp.wav	8000 Hz	Mono	32 bit	128 Kbps	00:02:26:56	2.23 Mb	✓		495 Kb	00:00:02:14	00:04:06:24	00:00:00:20	00:04:08:18	2.23 Mb	2.23 Mb
Klip suaratext.wav	8000 Hz	Mono	32 bit	128 Kbps	00:02:26:56	2.23 Mb		✓	3.04 Kb	00:00:02:14	00:04:06:24	00:00:00:20	00:04:08:18	2.23 Mb	2.23 Mb

Keterangan:

- Pesan berhasil diambil, tapi tidak terbaca (tulisan rusak)
- Pesan rusak total
- Pesan berhasil diambil dan masih dalam keadaan baik

P: Pixel gambar sebelum aksi
P^1: Pixel gambar setelah aksi
UF: Ukuran file sebelum aksi
UF^1: Ukuran file setelah aksi
UP: Ukuran pesan yang disisipkan
F: Tipe pesan berupa *File*
T: Tipe pesan berupa *Text*
I: Pengujian pertama
II: Pengujian kedua
L: Level pengujian pada pixel
SR: Sample Rate audio
CH: Chanel audio
RB: Resolusion bit audio
BR: Bitrate audio
Length: Length audio sebelum aksi
LCP: Length Cut Position
$Length^1$: Length audio setelah aksi

2. Secara umum, *stegofile* yang dihasilkan oleh aplikasi yang dirancang ini belum tahan *(unrobust)* terhadap serangan dari segi visual bahkan cenderung sangat rentan terhadap serangan visual. Hal ini disebabkan karena proses penyisipan pesan atau pemodifikasian bit antara bit pesan dan bit *carrier* dilakukan pada *stegofile* dari aplikasi ini adalah *sequential,* sehingga ketika terjadi penyerangan terhadap visual *stegofile* tentu akan cenderung merusak pesan yang ada di dalam *stegofile* itu sendiri.
3. Ketidakmampuan metode LSB dalam menangani penyerangan visual *(unrobust)* ternyata merupakan salah satu kelamahan yang lain setelah kelemahan pada aspek keamanan *(unsecure)* yang juga dimiliki oleh beberapa metode steganografi yang ada, seperti metode *Algorithm and Transformation* dan metode *Spread Spectrum Methode.* Ketiga metode ini memiliki kelemahan yang sama pada sebuah aplikasi steganografi, kecuali pada metode *Masking and Filtering* yang masih bisa tahan terhadap penyerangan visual. Namun, kelebihan yang dimiliki oleh metode LSB, seperti kelebihan pada aspek kecepatan dan kemudahan algoritma, aspek kapasitas, serta aspek kualitas menjadi kelemahan metode *Masking and Filtering.* Oleh sebab itu, seperti yang telah dijelaskan pada landasan teori pemilihan metode pada sebuah steganografi sangat bergantung dari kebutuhan yang ingin dicapai.

5.2.4. Pengujian Sistem pada Aspek Kualitas Steganografi

Pengujian sistem pada aspek kualitas dilakukan dengan menguji dan membandingkan jumlah bit-bit yang *error* antara *carrier file* dengan *stegofile.* Cara yang dilakukan, yaitu dengan menggunakan persamaan 2.1 untuk menghitung nilai *Mean Sequare Error* (MSE) dan persamaan 2.2 untuk menghitung nilai *Peak Signal to Noise Error* (PSNR) pada *file* gambar, serta persamaan 2.3 untuk menghitung nilai *Signal to Noise Error* (SNR) pada *file* audio dimana hasil dari dari ketiga persamaan ini memiliki satuan *decibel* (dB).

Apabila hasil nilai hitung PSNR dan SNR semakin besar, maka semakin baik kualitas *stegofile* yang dihasilkan atau mengindikasikan tingkatan jumlah

error pada sebuah *stegofile* semakin rendah. Tabel 5.5 adalah tabel yang merupakan hasil pengujian kualitas PSNR pada *file* gambar dan tabel 5.6 merupakan tabel hasil pengujian kualitas SNR pada *file* audio. Selanjutnya, hasil dari pengitungan ini dapat dilihat secara visual yang ditunjukkan secara pada gambar 5.6 dan gambar 5.7.

Tabel 5.5. Hasil Pengujian Aspek Kualitas PSNR pada *Image*

Nama Carrier	*Size Carrier* (bit)	Tipe Pesan	*Size* Pesan (bit)	MSE (dB)	PSNR (dB)
Bunga.bmp	18,874,800	*.bmp	4,055,040	1.2220	47.2600
		*.pdf	3,473,408	1.0212	48.0399
		*.doc	745,472	0.3865	52.2598
		*.gif	3,112,960	0.9269	48.4604
		text	24,903.68	0.0369	62.4591
		*.xls	950,272	0.4708	51.4026
		*.ppt	4,038,656	1.4035	46.6587
		*.jpg	4,210,688	1.1791	47.4153
		*.png	351,168	0.8317	48.9313
		*.wav	704,512	0.2752	53.7335
		*.zip	2,433,024	0.7093	49.6226
		*.rar	3,997,696	1.1429	47.5507

Tabel 5.6. Hasil Pengujian Aspek Kualitas SNR pada *Audio*

Nama Carrier	*Size Carrier* (bit)	RMS *Carrier*	Tipe Pesan	*Size* Pesan (bit)	RMS Pesan (dB)	SNR (dB)
Klip suara.wav	18,776,416	-14.57 dB	*.bmp	4,055,040	-14.52	49.2898
			*.pdf	3,473,408	-14.54	53.7268
			*.doc	745,472	-14.55	57.2486
			*.gif	3,112,960	-14.53	51.2280
			text	24,903.68	-14.56	63.2692
			*.xls	950,272	-14.55	57.2486
			*.ppt	4,038,656	-14.54	53.7268
			*.jpg	4,210,688	-14.52	49.2898
			*.png	351,168	-14.53	51.2280
			*.wav	704,512	-14.55	57.2486
			*.zip	2,433,024	-14.52	49.2898
			*.rar	3,997,696	-14.52	49.2898

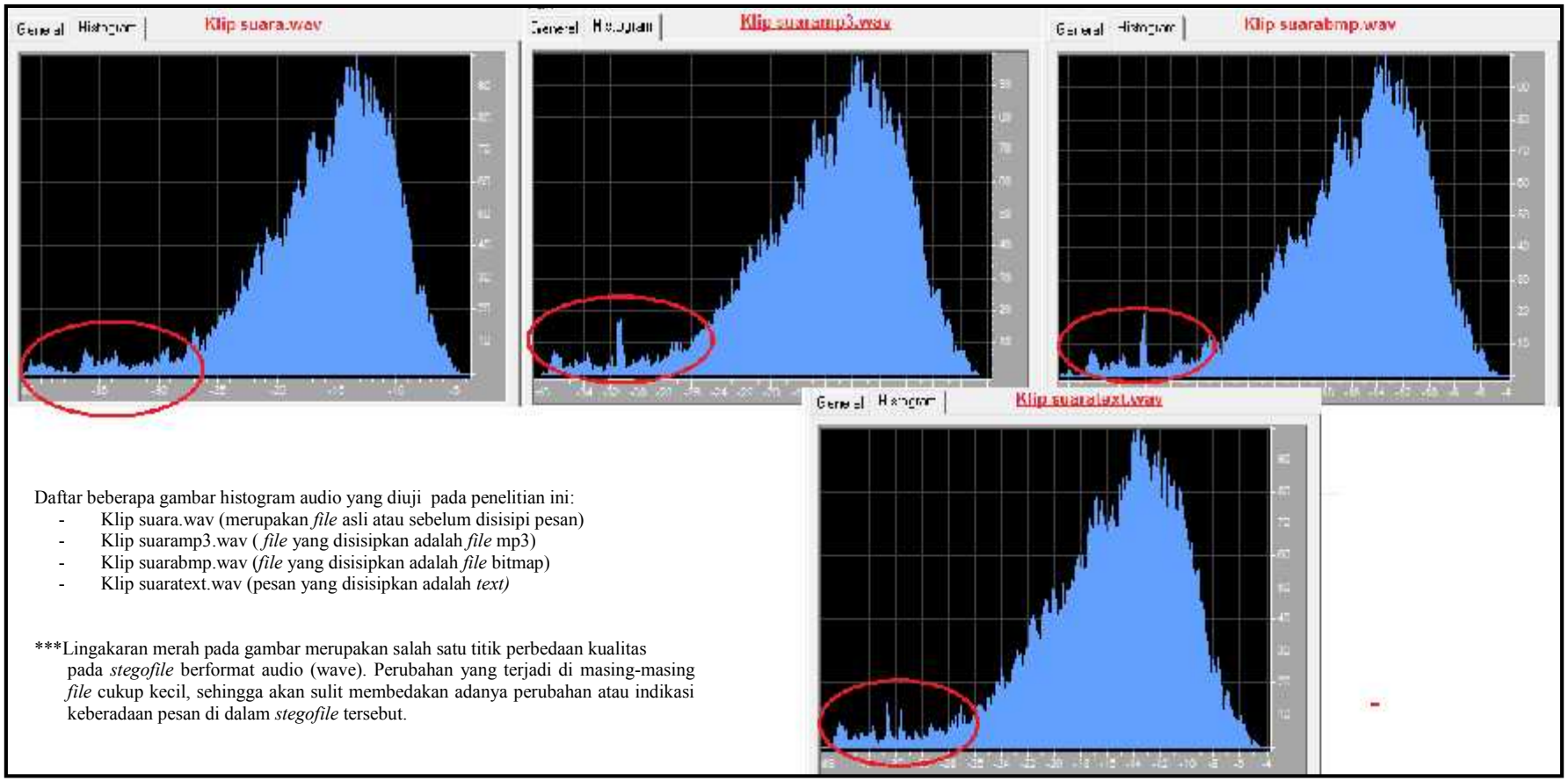
Tabel 5.7. *Waveform Statistic* untuk Memperoleh Nilai Rata-Rata RMS

Channel:	Mono	Channel:	Mono	Channel:	Mono	Channel:	Mono
Min Sample Value:	-32767	Min Sample Value:	-32767	Min Sample Value:	-32768	Min Sample Value:	-32767
Max Sample Value:	32767	Max Sample Value:	32767	Max Sample Value:	32767	Max Sample Value:	32767
Peak Amplitude:	0 dB	Peak Amplitude:	0 dB	Peak Amplitude:	0 dB	Peak Amplitude:	0 dB
Possibly Clipped:	4	Possibly Clipped:	4	Possibly Clipped:	2	Possibly Clipped:	4
DC Offset:	-.001	DC Offset:	-.002	DC Offset:	.004	DC Offset:	-.001
Minimum RMS Power:	-41.4 dB	Minimum RMS Power:	-38.8 dB	Minimum RMS Power:	-37.29 dB	Minimum RMS Power:	-41.4 dB
Maximum RMS Power:	-4.88 dB	Maximum RMS Power:	-4.88 dB	Maximum RMS Power:	-4.89 dB	Maximum RMS Power:	-4.88 dB
Average RMS Power:	**-14.57dB**	**Average RMS Power:**	**-14.56dB**	**Average RMS Power:**	**-14.52dB**	**Average RMS Power:**	**-14.58dB**
Total RMS Power:	-13.52 dB	Total RMS Power:	-13.52 dB	Total RMS Power:	-13.49 dB	Total RMS Power:	-13.52 dB
Actual Bit Depth:	16 Bits	Actual Bit Depth:	16 Bits	Actual Bit Depth:	16 Bits	Actual Bit Depth:	16 Bits
Using RMS Window of 50 ms		Using RMS Window of 50 ms		Using RMS Window of 50 ms		Using RMS Window of 50 ms	

Dari hasil pengujian terhadap aspek kualitas yang ditunjukkan pada tabel 5.5 (untuk *file* gambar) dan tabel 5.6 (untuk *file* suara) dapat disimpulkan bahwa aplikasi steganografi ini dapat mengahasilkan rata-rata kualitas *stegofile* sebesar 50.316 dB pada *file* bitmap dan 53.507 dB pada *file* wave. Rata-rata hasil pengujian pada aspek kualitas ini dapat dikatakan cukup baik karena memilki rata-rata kualitas di atas dari standar nilai toleransi, yaitu 20 dB. Untuk lebih jelasnya lihat gambar 5.6 (untuk *file* gambar) dan gambar 5.7 (untuk *file* audio) yang merupakan output *(stegofile)* yang dihasilkan oleh aplikasi steganografi ini.

Adapun nilai-nilai RMS *(Root Mean Sequare)* dapat diperoleh dengan bantuan aplikasi Cool Edit Pro 2.0 dimana RMS yang diambil adalah nilai rata-rata dari RMS yang keluar atau yang dihasilkan oleh masing-masing *audio file.* Untuk lebih jelasnya lihat tabel 5.7 berikut yang merupakan *waveform statistic* yang dilakukan oleh aplikasi Cool Edit Pro 2.0 ini untuk memperoleh nilai RMS.

Gambar 5.6. Pengujian Kualitas Gambar Secara Visual



Daftar beberapa gambar histogram audio yang diuji pada penelitian ini:

- Klip suara.wav (merupakan *file* asli atau sebelum disisipi pesan)
- Klip suaramp3.wav (*file* yang disisipkan adalah *file* mp3)
- Klip suarabmp.wav (*file* yang disisipkan adalah *file* bitmap)
- Klip suaratext.wav (pesan yang disisipkan adalah *text)*

***Lingakaran merah pada gambar merupakan salah satu titik perbedaan kualitas pada *stegofile* berformat audio (wave). Perubahan yang terjadi di masing-masing *file* cukup kecil, sehingga akan sulit membedakan adanya perubahan atau indikasi keberadaan pesan di dalam *stegofile* tersebut.

Gambar 5.7. Perbandingan Kualitas Suara Secara Visual

BAB VI

PENUTUP

6.1. Kesimpulan

Setelah menyelesaikan serangkaian tahapan-tahapan terhadap pembangunan rancang bangun aplikasi steganografi yang dimulai dari pengumpulan data tentang steganografi hingga pada tahapan pengujian, maka dapat diambil beberapa kesimpulan diantaranya adalah sebagai berikut:

1. Secara umum, rancang bangun aplikasi steganografi ini berhasil melakukan penyisipan *(embedding)* dan pengambilan kembali pesan *(retrieving)* baik pesan berupa teks maupun *file* dengan rekomendasi *carrier file* pada laporan penelitian ini, yaitu *file* bitmap (*.bmp) dan *file* wave (*.wav).
2. Hasil pengujian yang ditunjukkan pada tabel 5.2 membuktikan bahwa secara kapasitas *carrier* sebelum dan setelah disisipkan pesan adalah sama besar. Steganografi yang baik adalah steganografi yang dapat mengahasilkan *stegofile* yang harus sama besar dengan *carrier file* sebelum disisipi pesan.
3. Aspek kualitas yang ditunjukan pada tabel 5.5 membuktikan bahwa metode LSB yang digunakan pada aplikasi steganografi ini masih menunjukkan kualiatas yang baik dengan masing-masing nilai rata-rata PSNR *carrier file* berformat bitmap adalah 50.316 dB dan SNR *carrier file* berformat wave adalah 53.507 dB walaupun pada aplikasi ini bit-bit *carrier* yang dimodifikasi sebanyak 2 bit disetiap byte *carrier*-nya.
4. Kekurangan dari aplikasi ini adalah tidak tahan terhadap penyerangan secara visual bahkan cenderung sangat rentan. Kekurangan lainnya dari

metode LSB ini adalah dari sisi keamanan yang terbukti masih dapat dideteksi oleh aplikasi steganalisis, yaitu StegSpy 2.1. Hal ini dapat dibuktikan pada tabel 5.3 dan tabel 5.4.

6.2. Saran

Adapun saran-saran yang diajukan oleh penulis untuk perbaikan rancang bangun aplikasi steganografi ini adalah:

1. Berdasarkan kesimpulan yang dijelaskan di atas bahwa rancang bangun aplikasi ini masih kurang tahan terhadap modifikasi visual *stegofile* yang dihasilkan. Oleh sebab itu, penulis menyarankan untuk penelitian selanjutnya dapat menguji penggunaan dua metode steganografi sekaligus, yaitu menyandingkan metode LSB (untuk mempertahankan kelebihan aplikasi dari aspek kecepatan, kemudahan, kapasitas, dan kulaitas) dengan metode *Masking and Filtering* (untuk memperbaiki aspek ketahanan aplikasi steganografi). Metode *Masking and Filtering* secara teori merupakan satu-satunya metode dari empat metode steganografi yang telah dikenal yang dapat tahan terhadap penyerangan visual (modifikasi visual), namun hanya terbatas pada 24-bit dan *grayscale image*. Oleh sebab itu, metode LSB harus dipertahankan yang baik secara kecepatan dan kemudahan algoritmanya, aspek kapasitas dan kualitas *stegofile* yang dihasilkan sehingga bisa menutupi kelemahan metode *Masking and Filtering*.
2. Kekurangan lainnya adalah pada aspek keamanan. Oleh sebab itu, penulis menyarankan untuk penelitian selanjutnya dapat dilakukan dengan menyandingkan metode-metode steganografi seperti yang dijelaskan pada saran poin pertama dengan kriptografi untuk mengamankan data atau pesan yang disisipkan pada medianya.

DAFTAR PUSTAKA

Abu-Marie, Walaa, dkk. *Image Based Steganography Using Truth Table Based and Determinate Array on RGB Indicator.* International Journal of Signal and Image Processing (Vol.1-2010/Iss.3), May 2010.

Adli, Dziki. *Steganografi pada Berkas Wav dengan Metode Spread Spectrum dan Modifikasi LSB (Least Significant Bit).* Laporan Tugas Akhir Sarjana, Jurusan Teknik Informatika, Universitas Islam Negeri Sultan Syarif Kasim Riau. 2007.

Amarullah, Sandi. *Development Of Multimedia Steganography Applications Using End Of File Algorithm With Java Programming Language.* Undergraduate Program, Faculty of Industrial Engineering, 2010.

Ardhyana, Alfebra Stavia., Asep Juarna. 2008. *Aplikasi Steganografi pada MP3 Menggunakan Teknik LSB*. Teknik Informatika, Teknik Industri Universitas Gunadarma, 2008.

Arubusman, Yusrian Roman. *Audio Steganografi*. Laporan Skripsi Sarjana, Jurusan Teknik Informatika, Fakultas Teknologi Industri, Universitas Gunadarma, Agustus 2007.

Eric, Cole. *Hiding in Plain Sight : steganography and the Art of Covert Communication*. Wiley Publishing, Inc , Indiana, USA. 2003.

Gadicha, Vijay B., Gadicha, Ajay B. *Implementation Of Audio Wave Steganography By Replacing 4th Bit LSB Of Audio Wave File.* (IJCSIS) International Journal of Computer Science and Information Security,Vol. 9, No. 3, March 2011.

Habes, Alkhraisat. *Information Hiding in BMP Image Implementation, Analysis and Evaluation.* Saint Petersburg Institute for Informatics and Automation, Russian Academy of Sciences, Saint Petersburg, Russia, 26 February 2006.

Hapsari, Dian Dwi., Banowosari, Lintang Yuniar. *Aplikasi Video Steganography Dengan Metode Least Significant Bit (LSB).* Konferensi Nasional Sistem dan Informatika 2009; Bali, 14 November 2009.

Jacobson, Ivar, dkk. *Object-Oriented Software Engineering (A Use Case Driven Approach).* ACM Press, Addison-Wesley, Edinburgh Gate, England. 1992.

Jhonson, Neil F., Jajodia, Shusil. *Exploring Steganography: Seeing the Unseen,* IEEE Computer Magazine, 1998.

Narayana, Sujay., Prasad, Gaurav. *Two New Approaches for Secured Image Steganography Using Cryptographic Techniques and Type Conversions.* Signal & Image Processing : An International Journal(SIPIJ) Vol.1, No.2, December 2010.

Nugroho, Adi. *Analisis dan Perancangan Sistem Informasi dengan Metodologi Berorientasi Objek.* Penerbit Informatika Bandung, Bandung. 2002.

Nugroho, Adi. *Rekayasa Perangkat Lunak Berorientasi Objek dengan Metode USDP (Unified Software Development Process).* Penerbit ANDI Yogyakarta, Yogyakarta. 2010.

Parthasarathy, C., Srivatsa, S.K. *Increased Robustness Of LSB Audio Steganography by Reduced Distortion LSB Coding.* Journal of Theoretical and Applied Information Technology © 2005 - 2009 JATIT.

Prasad, M. Sitaram, dkk. *A Novel Information Hiding Technique for Security by Using Image Steganography.* Journal of Theoretical and Applied Information Technology 2005 - 2009 JATIT. All rights reserved. www.jatit.org.

Prihanto, Agus, dkk. *Peningkatan Kapasitas Informasi Tersembunyi pada Image Steganografi Menggunakan Teknik Hybrid.* Seminar Nasional Pascasarjana X – ITS, Surabaya, ISBN No. 979-545-0270-1. 4 Agustus 2010

R, Shreelekshmi., Wilscy, M. *Preprocessing Cover Images for More Secure LSB Steganography.* International Journal of Computer Theory and Engineering, Vol. 2, 1793-8201, No. 4, August 2010.

Rimmer, Steve. *Windows Bitmapped Graphics.* Windcrest Books, McGraw-Hill.Inc, United State of America. 1993.

Rishi, Rahul, dkk. *Mode and Multiple Technique: A New Image Steganography Method for Capacity Enhancement of Message in Image.* International Journal of Computer Applications (0975 – 8887) Volume 13– No.4, January 2011.

Rumbaugh, James, dkk. *Object Modeling and Design.* Prentice Hall, Englewood Cliffs, New Jersey. 1991.

Saroha, Kriti., Singh, Pradeep Kumar. 2010. *A Variant of LSB Steganography for Hiding Images in Audio.* International Journal of Computer Applications (0975 – 8887), Volume 11– No.6, December 2010.

Singh, Pradeep Kumar., Aggrawal, R.K. *Enhancement of LSB Based Steganography for Hiding Image in Audio.* (IJCSE) International Journal on Computer Science and Engineering Vol. 02, 1652-1658, No. 05, 2010.

Singh, Saurabh., Agarwal, Gaurav. 2010. *Hiding image to video: A new approach of LSB replacement.* International Journal of Engineering Science and Technology, Vol. 2, 6999-7003, December 2010.

Suyono. *Penyerangan pada Sistem Steganografi dengan Menggunakan Metode Visual Attacks dan Statistical Attacks.* Tugas Akhir Keamanan Sistem Lanjut (EC 7010), Bidang Khusus Teknologi Informasi, Program Studi Teknik Elektro, Program Pascasarjana, Institut Teknologi Bandung, 2004.

Wijaya, Ermadi Satria., Prayudi. Yudi. 2004. *Konsep Hidden Message Menggunakan Teknik Steganografi Dynamic Cell Spreading.* Media Informatika, Vol. 2, 23-38 ISSN: 0854-4743, No. 1, Juni 2004.

YANG, Hengfu, dkk. 2009. *A High-Capacity Image Data Hiding Scheme Using Adaptive LSB Substitution.* Radioengineering, Vol. 18, No. 4, December 2009.

Bell, Donald. 2003. *UML basics Part II: The activity diagram.* [Online] available. http://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep03/f_umlbasics_db.pdf, 30 April 2011.

Bell, Donald. 15-16- Sep 2004. *UML basics: The class diagram (An introduction to structure diagrams in UML 2).* [Online] available. http://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/, 30 April 2011.

Darkoman. 25 Sep 2008. *CWave - A Simple C++ Class to Manipulate WAV Files*. [Online] available. http://www.codeproject.com/KB/audio-video/CWave.aspx, 30 Maret 2011.

_____.*Data Hiding Steganograph Pada File Image Menggunakan Metode Least Significant Bit.* [Online] available. http://digilib.its.ac.id/public/ITS-NonDegree-7491-7406030017-bab2.pdf, 11 Juni 2011.

_____.1998. *Digital Still Camera Image File Format Standard (Exchangeable image file format for Digital Still Cameras: Exif) Version 2.1.* [Online] available. http://www.exif.org/Exif2-1.PDF, 11 Juni 2011.

_____.2003. *Portable Network Graphics (PNG) Specification (Second Edition) Information technology — Computer graphics and image processing — Portable Network Graphics (PNG): Functional specification. ISO/IEC 15948:2003 (E)*. [Online] available. http://www.w3.org/TR/PNG/, 11 Juni 2011.

_____. *ASCII Table.* [Online] available. http://ascii-table.com/conversions.php, 31 Mei 2011.

_____. *ASCII Table*. [Online] available. http://www.aubraux.com/design/ascii-table.php, 31 Mei 2011.

_____. *COMPUTERS Measurements for Memory & Storage*. [Online] available. http://www.athropolis.com/popup/c-comp2.htm, 31 Mei 2011.

_____. *Binary Truth Table*. Dari: http://www.tutorvista.com/math/binary-truth-table, 11 Juni 2011.

_____.http://wwweng.uwyo.edu/electrical/research/FroshECE/Microp/MicroprocessorsPORTHandout.pdf, 11 Juni 2011.