IMPLEMENTASI STEGANOGRAFI PADA CITRA DIGITAL MENGGUNAKAN METODE *LEAST SIGNIFICANT BIT*

TUGAS AKHIR

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

Oleh :

Nama : Endah Kurnia Asih Sejati

NIM : 08 523 106

JURUSAN TEKNIK INFORMATIKA

`FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

YOGYAKARTA

2012

IMPLEMENTASI STEGANOGRAFI PADA CITRA DIGITAL MENGGUNAKAN METODE *LEAST SIGNIFICANT BIT*

TUGAS AKHIR

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

Oleh :

Nama : Endah Kurnia Asih Sejati

NIM : 08 523 106

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA

2012

LEMBAR PENGESAHAN PEMBIMBING

IMPLEMENTASI STEGANOGRAFI PADA CITRA DIGITAL MENGGUNAKAN METODE *LEAST SIGNIFICANT BIT*

TUGAS AKHIR

Oleh :

Nama : Endah Kurnia Asih Sejati

NIM : 08 523 106

Yogyakarta,

Pembimbing

(Yudi Prayudi, S.Si., M.Kom.)

LEMBAR PENGESAHAN PENGUJI

IMPLEMENTASI STEGANOGRAFI PADA CITRA DIGITAL MENGGUNAKAN METODE *LEAST SIGNIFICANT BIT*

TUGAS AKHIR

Oleh:

Nama : Endah Kurnia Asih Sejati

NIM : 08 523 106

Telah Dipertahankan di Depan Sidang Penguji sebagai Salah Satu Syarat untuk Memperoleh Gelar Sarjana Jurusan Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 23 April 2012

Tim Penguji

Yudi Prayudi, S.Si., M.Kom.	
Ketua	
Zainudin Zukhri, ST., MIT.	
Anggota I	
Affan Mahtarami, S.Kom., M.T.	
Anggota II	

Mengetahui,

Ketua Jurusan Teknik Informatika

Universitas Islam Indonesia

(Yudi Prayudi, S.Si., M.Kom.)

LEMBAR PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan dibawah ini

Nama : Endah Kurnia Asih Sejati

NIM : 08 523 106

Tugas Akhir dengan judul :

IMPLEMENTASI STEGANOGRAFI PADA CITRA DIGITAL MENGGUNAKAN METODE *LEAST SIGNIFICANT BIT*

Dengan ini saya menyatakan dengan sesungguhnya bahwa dalam tugas akhir ini tidak terdapat keseluruhan tulisan atau karya yang saya ambil dengan menyalin, meniru dalam bentuk rangkaian kalimat atau simbol atau algoritma atau program yang menunjukan gagasan atau pendapat atau pemikiran orang lain, yang saya aku seolah-olah sebagai tulisan atau karya saya sendiri.

Apabila saya melakukan hal tersebut diatas, baik sengaja atau tidak, dengan ini saya menyatakan menarik tugas akhir yang saya ajukan sebagai hasil karya saya sendiri. Bila dikemudian hari terbukti bahwa saya melakukan tindakan diatas, gelar dan ijazah yang telah diberikan oleh Universitas Islam Indonesia batal saya terima.

Yogyakarta, 23 April 2012

Yang Membuat Pernyataan

(Endah Kurnia Asih Sejati)

HALAMAN PERSEMBAHAN

Tugas akhir ini kupersembahkan untuk.

Allah SWT

Atas karunia dan berkah yang tidak terhingga

Bapak dan Ibu

Yang selalu memberi dukungan dan doa atas semuanya.

Kasih sayang yang tiada batas.

Keluarga

Yang memberikan semangat dan bantuan dalam pengerjaan tugas ini

Teman-teman

Terima kasih atas bantuan , kritik dan saran yang sangat berguna dalam membantu penyelesaian tugas ini

MOTTO

“ Ketika kita dihadapakan dengan kebuntuan, disitulah timbul jalan lain yang tidak diduga-duga arahnya”

“No sacrifice, no victory”

“ Tidak ada masalah yang tidak bisa diselesaikan selama ada komitmen bersama untuk menyelesaikannya”

“ Dan Allah tidak menjadikan pemberian bala bantuan itu melainkan sebagai kabar gembira bagi kemenanganmu, dan agar tentram hatimu karenanya. Dan kemenanganmu itu hanyalah dari Allah”

(QS.Al Israa :36)

KATA PENGANTAR

Assalamualaikum Wr. Wb.

Alhamdulillahi rabbil 'alamin. Segala puji bagi Allah SWT yang telah memberikan kesempatan bagi penulis untuk menyelesaikan Laporan Tugas Akhir ini. Sesungguhnya hanya atas izin dan kehendak-Nya penulis dapat menyelesaikan tugas akhir ini.

Tugas akhir ini merupakan syarat wajib di jurusan Teknik Informatika Universitas Islam Indonesia untuk memperoleh gelar sarjana. Untuk itu pada kesempatan baik ini, penulis ingin mengucapkan terima kasih kepada :

1. Allah SWT atas segala berkah dan rahmat-Nya sehingga Tugas Akhir ini dapat diselesaikan.
2. Bapak, Ibu, dan adik-adikku atas kasih sayang, segala limpahan doa, dan dukungan.
3. Yang saya hormati Bapak Ir. Gumbolo HS., M.Sc selaku Dekan Fakultas Teknologi Industri Universitas Islam Indonesia.
4. Ketua Jurusan Teknik Informatika FTI UII, sekaligus dosen pembimbing tunggal dalam penyusunan Tugas Akhir ini Bapak Yudi Prayudi S.Si.,M.Kom.
5. Seluruh Dosen di Jurusan Teknik Informatika Universitas Islam Indonesia. Semoga ilmu yang telah diajarkan dapat menjadi amal, Amin.
6. Teman-teman angkatan 2008 (SNIPER) di Jurusan Teknik Informatika Universitas Islam Indonesia. Rekan-rekan seperjuangan Juwita Puspasari, Dwi Rahma Yulia, Resky Putri, Susan Dayat, Galuh Iswari, M. Zulfariansyah, Ahmad Fathoni, Rif'an Raditya, Yoga Shegi, Gilang Dewantara, dan banyak lagi yang tidak bisa disebutkan satu persatu. Sukses buat kita semua ☺.

7. Specially to Juwita Puspasari, Dwi Rahma Yulia, Resky Putri, Susan Dayat yang telah menjadi sahabat-sahabatku selama ini. Aldiani Puspasari, Lukman Ikhwanurahman, Dhini Nandya Maharani, Wowox, Mas Dana, Cahyadi atas kegilaannya selama ini, anak-anak bulqis, temen-temen KKN, temen-temen TK, SD, SMP, SMA dan Sychol terima kasih banyak atas dukungan dan waktunya selama skripsi ini dikerjakan.

Semua pihak yang tidak dapat saya sebutkan satu persatu. Penulis menyadari bahwa Tugas Akhir ini masih jauh dari sempurna. Untuk itu, penulis mengharapkan kritik dan saran yang bersifat membangun agar dapat berguna di kemudian hari. Penulis berharap semoga Tugas Akhir ini bermanfaat bagi semua orang, dan diri penulis sendiri. Amin.

Wassalamu'alaikum Wr. Wb.

Yogyakarta, 23 April 2012

Penulis

ABSTRAKSI

Teknologi jaringan dan keamanan komputer berkembang tanpa mengenal batasan jarak dan waktu. Perkembangan teknologi yang pesat juga sangat mendukung transmisi data untuk pertukaran informasi. Mudahnya pertukaran informasi yang ada juga harus didukung dengan perkembangan keamananan dalam transmisi data. Data yang dikirim sangat rentan terhadap adanya kerusakan dan kejahatan oleh pihak ketiga seperti *hacker* dan *cracker*. Steganografi merupakan salah satu solusi untuk melakukan transmisi data yang aman. Steganografi merupakan suatu ilmu yang mempelajari tentang teknik penyembunyian data kedalam file lain.

Salah satu contoh implementasi steganografi adalah dengan menggunakan metode *Least Significant Bits* (LSB) untuk penyembunyian data dan dapat di kombinasikan dengan enksripsi dengan algortima DES untuk menambah sekuritas file yang disembunyikan. File yang akan di aplikasikan dalam penelitian ini adalah file berformat *bitmap* sebagai *carrier image* dan file teks (.txt) sebagai file sisipnya.

. Pengujian pada implementasi steganografi ini akan dilakukan dengan menguji hasil *encoding* file dengan melakukan rotasi, mengubah ukuran pixel, menambah tingkat *brightness* dan perbedaan ukuran pada masing-masing sampel. Bedasarkan pengujian tersebut, didapatkan kesimpulan bahwa implementasi steganografi menggunakan metode LSB dan algoritma enkripsi DES sangat rentan terhadap modifikasi, tetapi tidak menimbulkan kecurigan pihak ketiga dari sisi ukuran karena file yang digunakan merupakan file *bitmap* yang berupa file citra tidak terkompresi.

Kata kunci : Steganografi, *Least Significant Bits*, Algoritma Enkripsi DES, *bitmap*

TAKARIR

carrier image	pembawa gambar
brightness	kecerahan
flowchart	diagram alir
embedding	penyisipan
extracting	pengungkapan
input	masukan
output	keluaran
tool	alat, sarana

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

UNIVERSITAS ISLAM INDONESIA

BAB I
PENDAHULUAN

1.1 Latar Belakang Masalah

Semakin berkembang pola pikir manusia dan semakin banyaknya tantangan, semakin berkembang pula tingkat kebutuhan. Salah satunya yaitu kebutuhan dalam memperoleh informasi yang cepat yang harus didukung dengan perkembangan teknologi. Perkembangan teknologi tersebut juga didukung dengan kemajuan cara berfikir manusia untuk mengembangkan *tool* yang dapat mengatasi masalah dalam perpindahan suatu informasi. Hal ini dapat dilihat dari berkembanganya internet yang semakin pesat. Pertukaran informasi dengan media internet memiliki banyak kelebihan, terutama dari segi kecepatan. Namun masalah yang sering terjadi adalah kurangnya segi keamanan dalam pengiriman data melalui internet. Data yang dikirim melalui media internet mudah sekali untuk disadap oleh para *hacker, cracker* .

Satu hal yang harus dimengerti adalah data atau informasi tidak hanya disajikan dalam bentuk teks, tapi dapat berupa citra, audio, maupun video. Keempat macam data atau informasi ini disebut dengan multimedia. Ada beberapa faktor mengapa data-data digital atau multimedia rentan dengan aktifitas pengaksesan secara legal atau tanpa ijin, antara lain yaitu data-data digital tersebut tidak didukung dengan perangkat lunak yang berhubungan dengan sekuritas, atau bisa juga seorang pengguna tidak memahami bagaimana cara menyembunyikan atau melindungi data digital dari orang atau pihak yang tidak berhak. Oleh karena itu, suatu bentuk pengamatan pada data digital sangatlah penting artinya. Pengamanan pada data digital selain berfungsi untuk meningkatkan keamanan, juga berfungsi untuk melindungi data digital yang bersifat rahasia agar tidak terlihat atau tidak dapat dibaca dari orang atau pihak yang tidak berhak.

Terdapat berbagai cara untuk menangani masalah keamanan data rahasia yang dikirimkan melalui media internet, diantaranya menggunakan teknik kriptografi, steganografi, watermarking. Steganografi adalah cara

menyembunyikan pesan ke dalam media lain sehingga orang-orang tertentu saja yang dapat mengetahui isi pesan. Dalam steganografi terdapat beberapa metode dalam menyembunyikan pesan rahasia ke dalam berkas lain tanpa menunjukkan ciri-ciri perubahan yang terlihat dalam kualitas dan struktur berkas semula. perubahan yang nyata atau terlihat dalam kualitas dan struktur dari berkas semula. Kelebihan steganografi jika dibandingkan dengan kriptografi adalah pesan-pesannya tidak menarik perhatian orang lain, karena dalam kriptografi hanya mengubah pesan kedalam pesan berkode sehingga dapat menimbulkan kecurigaan. Seringkali, steganografi dan kriptografi digunakan secara bersamaan untuk menjamin keamanan pesan rahasianya. Steganografi digital memiliki dua proses umum, yaitu proses *embedding* (penyisipan) dan *extracting* (pengungkapan/pemekaran). Di dalam steganografi digital, ada dua properti yang harus dipenuhi yaitu media penampung dan data digital yang akan disisipkan dapat berupa teks, dokumen, citra, audio, maupun video.

Ada beberapa teknik atau metode didalam steganografi digital dan yang paling umum digunakan adalah metode *Least Significant Bit* (LSB). Metode penyisipan ini merupakan metode yang bisa dikatakan sangat populer dan umum di dalam dunia steganografi digital. Metode ini bisa dikatakan metode yang paling sederhana dari beberapa metode yang ada, hanya saja metode LSB memiliki kelemahan yang sangat vital, yaitu data digital yang sudah disisipkan data digital yang lain dapat dideteksi dengan mudah dan juga kapasitas penyisipan tidak terlalu besar.

Dengan demikian, untuk menjaga data lebih aman digunakan algoritma enkripsi DES. DES merupakan algoritma yang pernah menjadi sangat terkenal di Amerika dan pernah menjadi keamanan dasar yang digunakan diseluruh dunia. DES (*Data Encryption Standard*) adalah algoritma *cipher block* yang populer karena dijadikan standar algoritma enkripsi kunci simetri.

Melalui penelitian ini dibangun suatu aplikasi berbasis Java yang mengimplementasikan steganografi dengan menggunakan metode *Least Significant Bit* sebagai cara untuk menyisipkan suatu data yang sudah di enkripsi dengan kriptografi algortima DES ke dalam media citra digital. Citra digital yang

akan digunakan sebagai media penampung adalah citra berformat bitmap yang merupakan representasi dari citra grafis yang terdiri dari susunan titik yang tersimpan didalam memori komputer. Penggunaan teknologi steganografi ini diharapkan dapat membantu upaya dalam peningkatan pengamanan pengiriman informasi dan mempermudah perlindungan atas hak cipta hasil karya media elektronik.

1.2 Rumusan Masalah

Dalam pelaksanan penelitian ini terdapat beberapa permasalahan yang menjadi titik utama pembahasan, yaitu bagaimana mengimplementasikan penyisipan pesan yang sudah dienskripsi menggunakan algoritma DES kedalam media citra digital menggunakan metode *Least Significant Bit* .

1.3 Batasan Masalah

Dengan adanya permasalahan yang ada, maka dibangun sebuah aplikasi steganografi dengan beberapa batasan masalah sebagai berikut :

a. Metode yang digunakan dalam penyisipan adalah *Least Significant Bit* dan algortima enkripsi data yang digunakan adalah algoritma DES.
b. Citra yang digunakan sebagai penampung (*carrier image*) adalah citra digital berformat bitmap.
c. Berkas yang akan disisipkan dapat berupa berkas teks (.txt).
d. Keluaran (*output*) dari perangkat lunak adalah berkas *bitmap* (*.bmp).
e. Ukuran berkas yang akan disisipkan harus lebih kecil dari citra penampung (*carrier image*).
f. Menggunakan ASCII 8 bit.
g. Perangkat lunak dibangun menggunakan Netbeans IDE 7.1.0 .

1.4 Review Penelitian

Setelah mempelajari beberapa penelitian sebelumnya, beberapa cara yang sering digunakan dalam steganografi adalah menggunakan metode *Least Significant Bits.* Penggunaaan metode tersebut juga sering dimodifikasi menggunakan algoritma. Adapun masalah yang terjadi adalah ketahanan hasil setelah proses steganografi. Dengan demikian topik penelitian ini menjadi sangat penting untuk mengangkat issue ketahanan hasil steganografi serta memberikan gambaran tentang pentingnya kerahasiaan dalam suatu transmisi data yang mana hasilnya dapat dimanfaatkan oleh orang lain.

Sejumlah penelitian telah mencoba mengangkat issue tentang steganografi. Chen (1999) telah melakukan penelitian yang berjudul "*An Adaptive Image Steganographic Model Based on Minimum-Error LSB Replacement*" dimana metode adaptif dipakai untuk melepas pembatasan ukuran *embedding* tetap dalam *pixel*. Metode ini akan mengurangi kesalahan *embedding* dan *embedding* menyediakan kapasitas yang lebih tinggi. Selain itu, untuk mendeteksi keberadaan pesan juga lebih sulit.

Selain itu, Hakim (2007) mengungkapkan tentang cara mengatasi kelemahan metode LSB dalam menyisipkan data dalam penelitian berjudul "Studi dan Implementasi Steganografi Metode LSB dengan *Preprocessing* Kompresi Data dan Ekspansi Wadah". Metode pertama untuk mengatasi masalah tersebut adalah melakukan preprossesing terhadap berkas data yang akan disimpan yaitu dengan jalan memampatkan data tersebut, kedua adalah melakukan preprosessing terhadap berkas wadah (*cover*) dengan cara memperbesar berkas wadah (*stretching image*), dan yang ketiga adalah menggabungkan metode pertama dan kedua sekaligus.

Untuk kalangan peneliti di Indonesia, issue tentang keamanan transmisi data atau pesan belum banyak disentuh dan menarik untuk diteliti. Untuk itulah maka penelitian ini dilakukan sebagai sebuah penelitian berkelanjutan dengan topik steganografi menggunakan metode LSB menggunakan *tools* Java.

1.5 Tujuan Penelitian

Tujuan penelitian ini adalah untuk membuat sebuah perangkat lunak dengan mengimplementasikan metode penyisipan *Least Significant Bit* dan algortima *enkripsi* DES dalam mempertahankan pesan atau data steganografi yang disisipkan pada citra berformat *bitmap.*

1.6 Manfaat Penelitian

Manfaat penelitian ini adalah membuat sebuah implementasi steganografi yang membantu pengguna dalam melakukan pengiriman data yang aman menyisipkan data rahasia dalam gambar.

1.7 Metodologi Penelitian

Metode yang digunakan dalam penyusunan tugas akhir ini adalah sebagai berikut :

a. Studi Literatur

   Studi literatur dilakukan dengan mempelajari buku, jurnal ilmiah, halaman web yang berkaitan dengan kriptografi, steganografi, watermarking, metode penyisipan dan ekstraksi pesan/data steganografi, algortima enkripsi, dan citra digital.

b. Analisis Masalah

   Pada tahap ini perkerjaan yang dilakukan adalah menganalisis kebutuhan aplikasi penyisipan pesan/data steganografi terhadap citra bitmap.

c. Perancangan Perangkat Lunak

   Pada tahap ini yang akan dilakukan adalah perancangan aplikasi penyisipan pesan/data steganografi berdasar hasil analisis kebutuhan yang dilakukan sebelumnya dengan menggunakan bahasa pemrograman JAVA Netbeans 7.1.0

d. Implementasi Perangkat Lunak

   Pada tahap ini akan dilakukan pengimplemantasian aplikasi penyisipan pesan/data steganografi yang telah dirancang prototipenya.

e. Pengujian dan Analisis Hasil Impementasi

Analisis diperoleh dari implementasi sistem yang telah dibangun.

f. Kesimpulan

Setelah pengujian dan analisis dilaksanakan, maka dibuat kesimpulan mengenai program dan hasil implementasinya.

1.8 Sistematika Penulisan

Sistematika penulisan yang digunakan adalah sebagai berikut :

BAB I PENDAHULUAN

Bab ini berisi tentang latar belakang masalah, rumusan masalah, batasan masalah, review penelitian, tujuan penelitian, manfaat penelitian, metode penelitian, dan sistematika penelitian.

BAB II LANDASAN TEORI

Bab ini berisi tentang uraian teori-teori yang berhubungan dengan penelitian tugas akhir yang berisikan penjelasan mengenai citra bitmap, sejarah steganografi, steganografi digital, perbedaan digital steganografi dengan kriptografi, metode penyisipan pada digital steganografi, metode penyisipan *least significant bit* , algortitma *enkripsi* DES, serta uraian singkat mengenai perangkat lunak yang digunakan pada tugas akhir ini.

BAB III ANALISIS KEBUTUHAN DAN PERANCANGAN PERANGKAT LUNAK

Bab ini berisi tentang analisis kebutuhan, perancangan dan proses pembuatan perangkat lunak serta langkah-langkah apa saja yang dikerjakan untuk menyelesaikan masalah.

BAB IV HASIL DAN PEMBAHASAN

Berisi implementasi hasil bagaimana realisasi metodologi dan rancangan yang dibuat dengan cara atau proses pembuatan dari awal sampai akhir dan berisi alur-alur program yang dibuat serta aliran data yang digunakan, termasuk juga hasil tampilan. Hasil pembuatan software aplikasi diuji coba pada berbagai data yang akan disembunyikan.

BAB V KESIMPULAN DAN SARAN

Bab ini berisikan hasil proses pengembangan perangkat lunak dan saran-saran yang perlu diperhatikan berdasarkan keterbatasan-keterbatasan yang ditemukan.

BAB II
LANDASAN TEORI

2.1 Steganografi

Menurut modul CHFI (2011) steganografi didefinisikan sebagai "Seni dan ilmu menyembunyikan informasi dengan menyisipkannya ke dalam pesan lain, pesan yang tampaknya tidak berbahaya". Sedangkan menurut Prayudi dan Rahmadhani (2005) steganografi merupakan ilmu yang mempelajari, meneliti, dan mengembangkan seni menyembunyikan sesuatu informasi. Steganografi dapat digolongkan sebagai salah satu bagian dari ilmu komunikasi. Kata steganografi berasal dari bahasa Yunani yang berarti "tulisan tersembunyi". Pada era informasi digital, steganografi merupakan teknik dan seni menyembunyikan informasi yang sesungguhnya tidak kelihatan.

Secara teori, semua file umum yang ada didalam komputer dapat digunakan sebagai media, seperti gambar berformat JPG, GIF, BMP, atau di dalam musik MP3, atau bahkan didalam sebuah film dengan format WAV atau AVI. Semua dapat dijadikan tempat bersembunyi, asalkan file tersebut memiliki bit-bit data redundan yang dapat dimodifikasi. Setelah dimodifikasi file media tersebut tidak akan banyak terganggu fungsinya dan kualitasnya tidak akan jauh berbeda dengan aslinya (Prayudi & Rahmadhani, 2005).

Menurut Irianto (2004) ada tiga aspek yang berbeda yang mempengaruhi sifat dan sistem penyembunyian atau penyisipan pesan rahasia pada gambar seperti kapasitas, keamanan, dan ketahanan. Kapasitas merujuk pada jumlah informasi yang dapat disembunyikan dalam medium cover. Keamanan adalah ketidakmampuan pengamat untuk mendeteksi pesan yang tersembunyi, dan ketahanan yaitu jumlah modifikasi *stego medium* yang dapat bertahan sebelum fihak lain dapat merusak pesan rahasia yang tersembunyi tersebut.

2.1.1 Sejarah Steganografi

Steganography (steganografi) merupakan seni untuk menyembunyikan pesan (rahasia) ke dalam pesan lainnya sedemikian rupa sehingga membuat orang lain tidak menyadari ada sesuatu di dalam pesan tersebut. Kata *Steganography* berasal dari bahasa Yunani, yaitu gabungan dari kata *steganos* (tersembunyi atau terselubung) dan *graphein* (tulisan atau menulis), sehingga menurut Sellars (1996) kurang lebih bisa diartikan menulis tulisan yang tersembunyi atau tulisan tersembunyi (*hidden/covered writing*) menurut Johnson & Jajodia (1998). Pada jaman Yunani kuno, steganografi digunakan oleh penguasa Yunani untuk mengirimkan pesan rahasia dengan cara menuliskan pesan tersebut di kepala prajurit atau budak yang rambutnya sudah dicukur terlebih dahulu dan ketika rambut prajurit atau budak tersebut sudah tumbuh kemudian prajurit atau budak tersebut diutus untuk membawa pesan rahasia di kepalanya.

Bangsa Romawi mengenal steganografi dengan menggunakan *Invisible Ink* (tinta tak tampak) untuk menuliskan pesan rahasia. Tinta tersebut dibuat dari campuran sari buah jeruk, susu atau urine dan cuka, yang mana jika tinta tersebut digunakan untuk menulis maka tulisannya menjadi tidak terlihat dan tulisan tersebut bisa dibaca dengan cara memanaskan kertas. Pada Perang Dunia II, salah satu teknik atau metode dari steganografi, yaitu *Null Cipher* juga digunakan untuk mengirim pesan rahasia. *Null Cipher* adalah suatu cara untuk menyembunyikan pesan didalam pesan yang lain tanpa menggunakan algoritma yang rumit (Kessler, 2004). Kessler (2004) memberikan contoh klasik dari *null chiper* yang digunakan pada waktu PD II yang dibuat oleh kedutaan Jerman di Washington DC dan mengirimkannya melalui telegram ke markas besar Jerman di Berlin:

*"A**p**parently n**e**utral's p**r**otest i**s** t**h**roughly d**i**scounted a**n**d i**g**nored.*
*I**s**man h**a**rd **i**t. B**l**ockade i**s**sue a**ff**ects p**r**etext f**o**r e**m**bargo o**n** b**y**-*
*products, e**j**ecting s**u**ets a**n**d v**e**getables o**i**ls."*

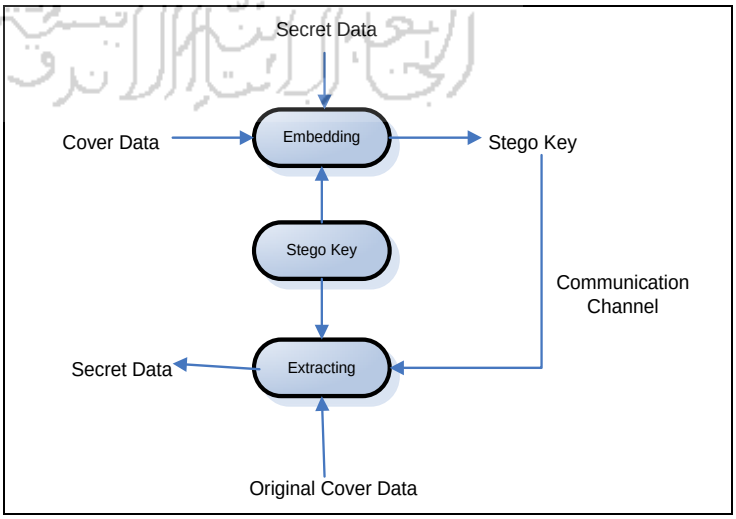
Pesan rahasia bisa diperoleh dengan cara membaca huruf ke-2 disetiap katanya, maka akan diperoleh :

"Pershing sails from NY June 1."

2.1.2 Steganografi Digital

Di dalam steganografi digital, harus memiliki dua properti yaitu media penampung (*cover data* atau *data carrier*) dan data digital yang akan disisipkan (*secret data*), dimana media penampung dan data digital yang akan disisipkan berupa multimedia (teks/dokumen, citra, audio, maupun video) tapi pada umumnya berupa data citra. Proses umum yang ada didalam steganografi digital adalah proses *embedding* atau *encoding* (penyisipan) dan *extracting* atau *decoding* (pemekaran atau pengungkapan kembali (*reveal*)). Hasil yang didapat setelah proses *embedding* atau *encoding* disebut *Stego Object* (apabila media penampung hanya berupa data citra maka disebut *Stego Image*).

Untuk menambah tingkat keamanan, *stego object* atau *stego image* diproteksi dengan kunci rahasia yang disebut juga *Stego Key*. Biasanya *stego key* menerapkan metode-metode yang ada didalam *cryptography* (kriptografi), contohnya algoritma *Rivest Code 4* atau *Rivest Code 6*. *Stego key* di dalam steganografi digital bersifat opsional, hanya saja apabila data rahasia yang akan disisipkan tidak dilengkapi dengan *stego key*, maka data tersebut mudah untuk dibongkar oleh pihak ketiga atau pihak yang tidak berhak. Berikut adalah gambaran umum dari steganografi digital adalah sebagaimana pada Gambar 2.1 :



Gambar 2. 1 Gambaran umum proses steganografi digital

Sumber : (Prayudi & Kuncoro, 2005)

Menurut gambar diatas, di dalam steganografi digital untuk melakukan proses *embedding* (penyisipan) dibutuhkan 3 materi, yaitu *cover data* (data penampung), *secret data* (data rahasia) dan *stego key* (kunci stego), kemudian hasil akhir yang didapat berupa data digital yang telah disisipkan data digital yang bersifat rahasia atau *stego object*. Untuk proses *extracting* (pengungkapan kembali), yang diperlukan hanya *stego object* dan *stego key* yang sama pada saat digunakan waktu proses *embedding* dimana hasil akhir yang didapat yaitu *secret data*. *Stego key* disini walaupun bersifat optional namun tetap dimasukkan di dalam atribut yang harus dimiliki di dalam steganografi digital, dengan alasan untuk menambah tingkat keamanan pada *stego object*. *Communication channel* adalah suatu media yang digunakan untuk mengirim *stego object* dari pengirim ke penerima, seperti *email* dan sebagainya.

Ketika proses penyembunyian data digital yang bersifat rahasia ke dalam *cover data*, secara otomatis akan mengubah kualitas dari data yang dijadikan media penampung. Menurut Munir (2006) di dalam aplikasi steganografi digital harus memiliki beberapa kriteria, adapun kriteria-kriterianya adalah sebagai berikut :

a *Imperceptibility*

Keberadaan pesan rahasia tidak dapat dipersepsi oleh inderawi. Misalnya, jika *covertext* berupa citra, maka penyisipan pesan membuat citra *stegotext* sukar dibedakan oleh mata dengan citra *covertext*-nya. Jika *covertext* berupa audio (misalnya berkas *mp3, wav, midi,* dan sebagainya), maka indera telinga tidak dapat mendeteksi perubahan pada audio *stegotext*-nya.

b *Fidelity*

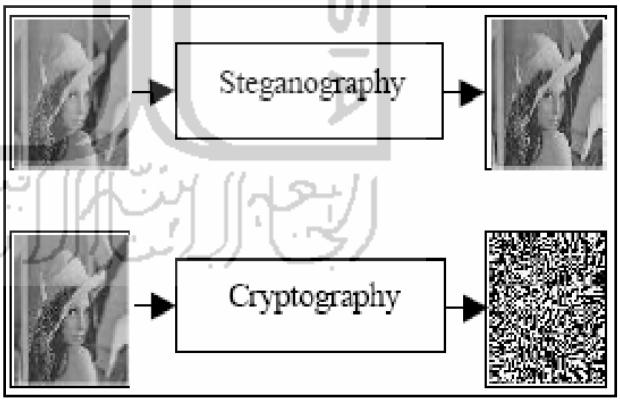
Mutu media penampung tidak berubah banyak akibat penyisipan. Perubahan tersebut tidak dapat dipersepsi oleh inderawi. Misalnya, jika *covertext* berupa citra, maka penyisipan pesan tersebut membuat citra *stegotext* sukar dibedakan oleh mata dengan citra *covertext*-nya. Jika *covertext* berupa audio (misalnya berkas *mp3, wav, midi,* dan sebagainya), maka audio *stegotext* tidak rusak dan indera telinga tidak dapat mendeteksi perubahan tersebut.

c *Recovery*

Pesan yang disembunyikan harus dapat diungkapkan kembali (*reveal*). Karena tujuan steganografi digital adalah *data hiding*, maka sewaktu-waktu pesan rahasia di dalam *stegotext* harus dapat diambil kembali untuk digunakan lebih lanjut.

2.1.3 Perbedaan Steganografi Digital dan Kriptografi

Telah dijelaskan di awal bahwa steganografi digital dan *cryptography* (kriptografi) masih dalam satu induk mengenai pengamanan data digital, hanya saja keduanya memiliki perbedaan yang mendasar. Pada steganografi digital, data digital atau informasi rahasia dibuat tidak terlihat karena informasi tersebut disembunyikan di dalam data digital yang lain, sedangkan pada kriptografi informasi rahasia dibuat menjadi tidak terbaca dan seolah-olah terlihat berantakan. Gambaran umum dari perbedaan antara steganografi digital dengan kriptografi bisa dilihat pada Gambar 2.2 :



Gambar 2. 2 Ilustrasi steganografi digital dengan kriptografi pada citra

Sumber : Prayudi & Kuncoro (2005)

Perbedaan steganografi dan kriptografi menurut modul CHFI (2011) akan dijelaskan pada Tabel 2.1 :

Tabel 2. 1 Perbedaan Steganografi dan Kriptografi

Steganografi	Kriptografi
Steganografi adalah teknik menyembunyikan informasi dengan penyisipan pesan dalam pesan lain	Kriptografi adalah teknik pengkodean isi dari pesan sedemikian rupa sehingga isinya tersembunyi dari luar
Pesan tersebut tidak terlihat, karena tersembunyi di balik pesan lain	Keberadaan pesan jelas, namun artinya dikaburkan
Hanya satu kunci private yang digunakan	Ada dua kunci yang digunakan, kunci pyblik untuk enkripsi dan kunci private untuk deksripsi

2.2 Metode Penyisipan pada Steganografi Digital

Sekarang ini sudah banyak metode-metode yang digunakan untuk menyisipkan data digital rahasia pada steganografi digital dan yang paling sering digunakan adalah metode *Least Significant Bit* (LSB).

2.2.1 Metode Penyisipan *Least Significant Bit* (LSB *Encoding*)

Metode penyisipan ini merupakan metode yang bisa dikatakan sangat populer dan umum di dalam dunia steganografi digital. Metode penyisipan LSB atau LSB *encoding* merupakan metode dengan pendekatan yang sangat sederhana dibandingkan dengan metode-metode yang lain. Sebagai wacana bahwa pada susunan bit di dalam sebuah *byte* (8 bit), memiliki bit yang paling berarti atau *Most Significant Bit* (MSB) dan bit yang paling kurang berarti atau disebut juga *Least Significant Bit* (LSB). Bit yang cocok untuk diganti adalah bit LSB, sebab penggantian hanya mengubah nilai *byte* tersebut satu lebih tinggi atau satu lebih rendah dari nilai sebelumnya. Sebagai contoh *byte* tersebut di dalam citra menyatakan warna tertentu, maka perubahan satu bit LSB tidak mengubah warna tersebut secara berarti dan lagipula sistem penglihatan manusia atau *Human Visual System* (HVS) tidak dapat membedakan perubahan yang kecil pada warna. Jadi metode LSB adalah suatu metode penyisipan pada steganografi digital, dengan mengubah bit terakhir (umumnya 1 bit) pada setiap *byte* (pada *cover data*) dengan 1 bit pada pesan rahasia. Dalam LSB, sisi kanan dalam notasi biner digantikan dengan pesan tertanam. Hanya saja metode ini sangat mudah untuk

dideteksi oleh pihak ketiga atau pihak yang tidak berwenang jika penyerang mengetahui teknik LSB subtitusi yang digunakan (CHFI, 2011).

Untuk lebih jelasnya bisa dilihat ilustrasi berikut. Sebagai media penampung adalah data citra 24 bit (*cover image*) dengan ukuran 3 piksel (*9 bytes*), kemudian diubah kebentuk data *raster* (biner) :

```
00100111 11101001 11001000
00100111 11001000 11101001
11001000 00100111 11101001
```

Data yang akan disisipkan adalah karakter ”A” yang mempunyai nilai desimal 65, kemudian diubah kebentuk biner (ASCII 8 bit) menjadi `01000001`. Nilai ketiga piksel diatas akan berubah menjadi :

```
0010011***0*** 1110100*1* 1100100*0*
0010011***0*** 1100100*0* 1110100***0***
1100100*0* 0010011*1* 1110100*1*
```

Bisa dilihat bahwa setelah dilakukan proses penyisipan, nilai yang benar-benar berubah hanya pada *byte* pertama, keempat dan keenam (LSB yang dicetak tebal), dan itu tidak menimbulkan perubahan warna yang sangat signifikan karena total perubahan yang terjadi hanya 1/255 atau 0,39% dari seluruh permukaan citra.

Mengenai hal kapasitas penyembunyian, pada metode penyisipan LSB memiliki kelemahan karena hanya bisa menyembunyikan dengan jumlah yang kecil. Jika 3 *bit* per piksel, maka jumlah total *bit* yang akan disisipkan sama dengan 3 kali jumah total piksel suatu data citra. Sebagai contoh suatu citra *24 bit* memilik resolusi atau ukuran 1024 x 768 x 3 / 8 = 294.912 *bytes*.

2.2.2 Penyisipan *Image File* dalam LSB

Bit paling kanan dalam sebuah bit adalah *least significant bit*. LSB dari setiap byte dapat diganti dengan perubahan kecil pada keseluruhan file. Data biner dari pesan rahasia diubah kemudian disisipkan kedalam LSB dari setiap pixel dalam *image file*. Berikut ini teknik menyembunyikan dan memulihkan data menurut modul CHFI (2011) :

Cara menyembunyikan data :

a. Menggunakan Red, Green, Blue (RGB) model, program stego membuat salinan sebuah palet gambar.
b. LSB dari masing-masing pixel 8 bit biner diganti dengan satu bit dari pesan tersembunyi
c. Sebuah RGB color dalam sebuah palet disalin terbentuk
d. Pixel ini lalu diubah kedalam bilangan biner 8bit RGB color yang baru.

Cara memulihkan data (*recovering the data*) :

a. Program stego menemukan nomor 8 bit biner dari masing-masing RGB color dalam setiap piksel.
b. Setiap LSB dari 8 bit bilangan biner masing-masing piksel adalah satu bit dari file data yang disembunyikan
c. Setiap LSB kemudian ditulis ke file output.

2.3 Citra Digital

Citra adalah gambar pada bidang dua dimensi. Dalam tinjauan matematis, citra merupakan fungsi kontinu dari intensitas cahaya pada bidang dua dimensi. Ketika sumber cahaya menerangi objek, objek memantulkan kembali sebagian cahaya tersebut. Pantulan ini ditangkap oleh alat-alat pengindera optik, misalanya mata manusia, kamera, scanner, dan sebagainya. Bayangan objek tersebut akan terekam sesuai intensitas pantulan cahaya. Ketika alat optik yang merekam pantulan cahaya itu merupakan mesin digital, misalnya kamera digital, maka citra yang dihasilkan merupakan citra digital. Pada citra digital, kontinuitas intensitas cahaya dikuantisasi sesuai resolusi alat perekam.

Di dalam komputer, citra digital disimpan sebagai suatu file dengan format tertentu. Format citra tersebut menunjukan cara sebuah citra digital disimpan, misalnya apakah dengan suatu kompresi atau tidak. Contoh format citra digital adalah .bmp, .jpg, .png, .tif dan sebagainya. Ukuran citra digital dinyatakan dalam pixel (picture element). Umumnya, nilai setiap pixel merupakan kuantisasi harga intensitas cahaya. Dengan demikian, suatu citra digital dapat dipandang sebagai

sebuah matriks yang elemen-elemennya menunjukkan intensitas cahaya terkuantisasi. Bedanya terletak pada urutan penyebutan angka ukuran tersebut. Citra digital dengan ukuran 92x112 pixel sebenarnya merupakan sebuah matriks dengan ukuran 112x92, dimana 112 merupakan banyaknya baris dan 92 merupakan banyaknya kolom.

Citra digital yang dimaksudkan dalam keseluruhan tugas akhir ini adalah "citra diam" (still image). Selanjutnya citra diam cukup disebut citra seperti pada Gambar 2.3 berikut ini :

Gambar 2. 3 Gambar Fruit contoh citra diam

Sumber : https://engineering.purdue.edu/~ace/sampleimages.html

2.3.1 Format Citra Digital

Citra digital disimpan dalam berkas dengan menggunakan format tertentu. Format citra yang baku dilingkungan sistem operasi Microsoft Windows dan IBM OS/2 adalah berkas Bitmap (**BMP**), *Joint Photographic Experts Group* (**JPEG**), *Graphics Interchange Format* (**GIF**) dan lain-lain. Saat ini format **BMP** memang tidak sepopuler format **JPEG** atau **GIF**, hal ini karena berkas **BMP** tidak dimampatkan sehingga ukuran datanya relatif besar daripada berkas **JPEG** maupun **GIF**. Meskipun format **BMP** memiliki kekurangan dari segi ukuran

tetapi format **BMP** memiliki kelebihan dari segi kualitas gambar, karena tidak dimampatkan sehingga tidak ada informasi yang hilang.

2.3.2 Bitmap (BMP)

Bitmap atau raster merupakan gambar yang tersusun atas titik-titik elemen gambar (pixel), masing-masing pixel memiliki informasi warna. Jumlah kemungkinan warna yang dapat ditampilkan oleh suatu pixel tergantung pada satuan bit yang dimiliki gambar tersebut. Sebagai contoh bitmap 4 bit, berarti pixel-pixel yang menyusunnya dapat menampilkan kombinasi warna sebanyak 2^4 atau 16 warna, demikian sebaliknya bitmap 8 bit yang mampu menampilkan kombinasi warna hingga 2^8 atau 256 warna.

Berkas bitmap warna 24 bit mempunyai tiga komponen warna yaitu RGB *(Red, Green, Blue)*. Tiap-tiap komponen tersebut terdiri 1 *byte* (8 bit). Karena tiap *byte* memiliki kombinasi 256 warna, maka jika terdapat 3 komponen warna maka mempunyai 2^{24} atau 16.777.216 kombinasi warna. Jika sebuah citra dengan format bitmap warna 24 bit dengan ukuran 800 x 600 maka besarnya ukuran berkas bitmap tersebut adalah (800 x 600 x 24) bit.

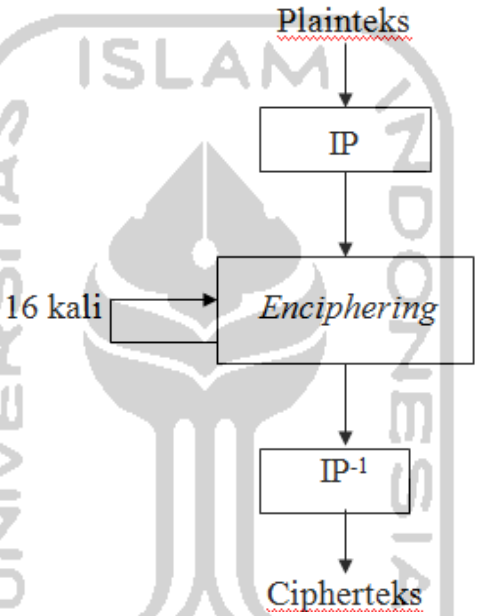
Bitmap dengan resolusi (jumlah pixel setiap satuan ukur) besar, akan terlihat lebih halus dibandingkan dengan yang memiliki resolusi rendah. Resolusi bitmap dinyatakan dalam satuan dot per inch (dpi) atau pixel per inch (ppi).

2.4 Algoritma *DES (Data Encryption Standard)*

DES (*Data Encryption Standard*) adalah algoritma *cipher block* yang populer karena dijadikan standar algoritma enkripsi kunci simetri. Sebenarnya DES adalah nama standar enkripsi simetri, nama algoritma enkripsinya sendiri adalah DEA (*Data Encryption Algorithm*), namun nama DES lebih populer daripada DEA. Algoritma DES dikembangkan di IBM dibawah kepemimpinan W. L. Tuchman pada tahun 1972. Algoritma ini didasarkan pada algoritma *Lucifer* yang dibuat oleh Horst Feistel. Algoritma ini telah disetujui oleh *National*

Bureau of Standard (*NBS*) setelah penilaian kekuatannya oleh *National Security Agency* (*NSA*) Amerika Serikat.

DES beroperasi pada ukuran blok 64 bit. DES mengenkripsikan 64 bit *Plaintext* menjadi 64 bit *ciphertext* dengan menggunakan 48 bit kunci internal (*internal key*) atau upa-kunci (*subkey*). Kunci internal dibangkitkan dari kunci eksternal (*external key*) yang panjangnya 64 bit.



Gambar 2. 4 Skema Global Algoritma DES

Sumber : Munir (2006)

Skema global dari algoritma DES adalah sebagai berikut :

a. Blok *Plaintext* dipermutasi dengan matriks permutasi awal (*initial permutation* atau IP).

b. Hasil permutasi awal kemudian di-*enciphering* sebanyak 16 kali (16 putaran). Setiap putaran menggunakan kunci internal yang berbeda.

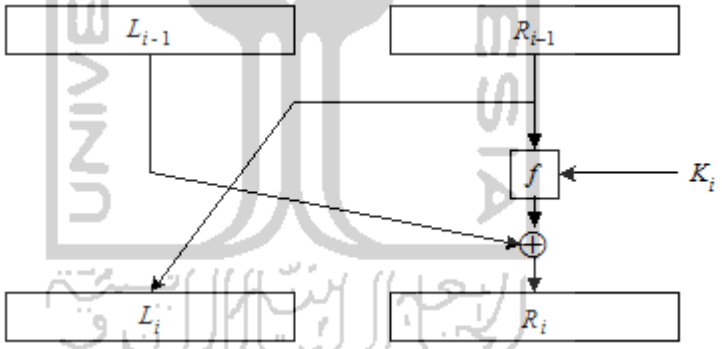
c. Hasil *enciphering* kemudian dipermutasi dengan matriks permutasi balikan (*invers Initial Permutation* atau IP^{-1}) menjadi blok *ciphertext*.

Di dalam proses *enciphering*, blok *plaintext* terbagi menjadi dua bagian, kiri (L) dan kanan (R), yang masing-masing panjangnya 32 bit. Kedua bagian ini masuk ke dalam 16 putaran DES. Pada setiap putaran *i*, blok *R* merupakan masukan untuk fungsi transformasi yang disebut *f*. Pada fungsi *f*, blok *R* dikombinasikan dengan kunci internal K_i. Keluaran dari fungsi *f* di-XOR-kan dengan blok *L* untuk mendapatkan blok *R* sebelumnya. Ini adalah satu putaran DES. Secara matematis, satu putaran DES dinyatakan sebagai :

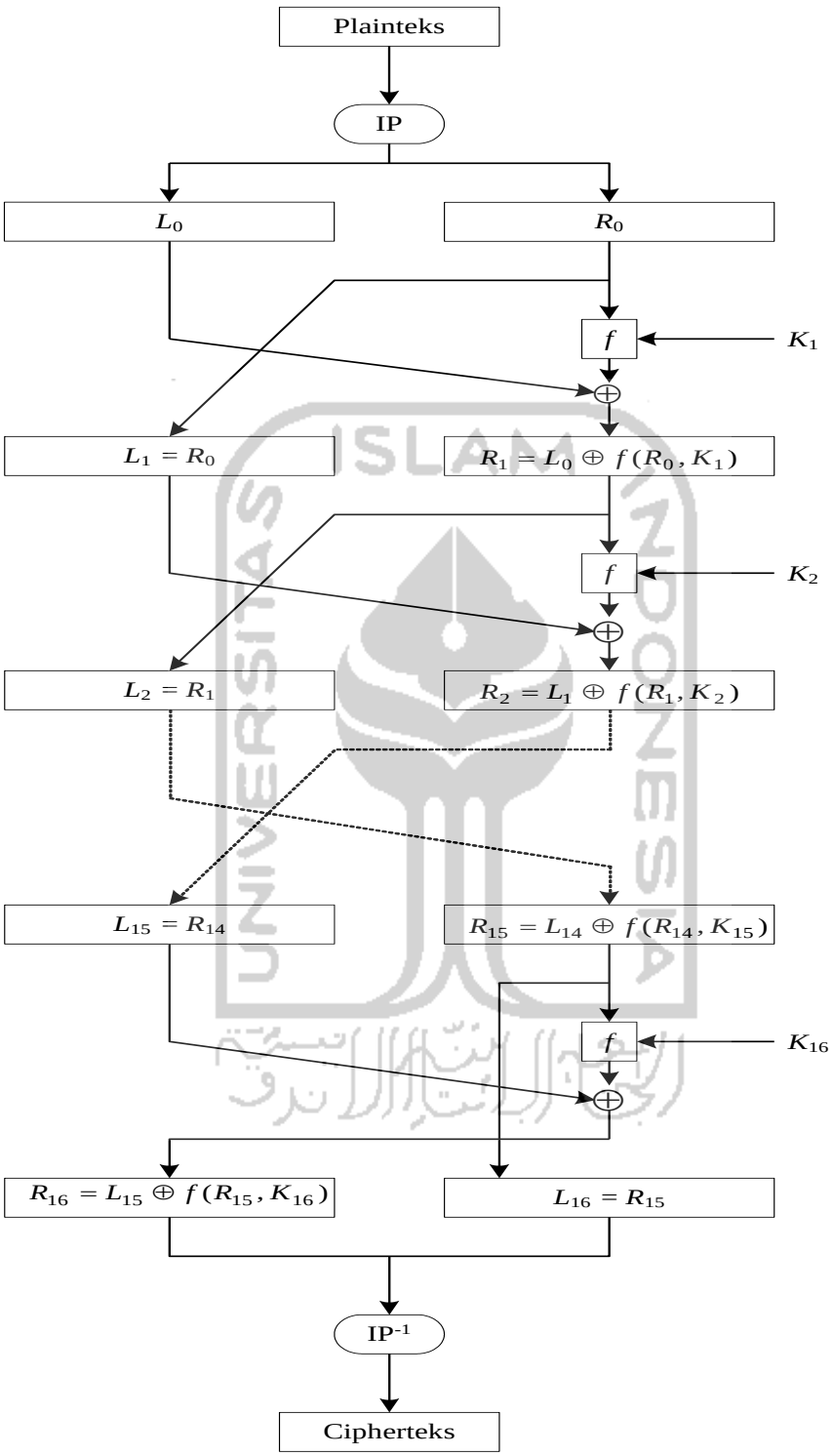
$$Li = R_{i-1} \quad (2\text{-}1)$$

$$R_i = \mathrm{L}_{i-1} \oplus f(R_{i-1}, K_i) \quad (2\text{-}2)$$

Gambar 2.6 memperlihatkan skema algoritma DES yang lebih rinci. Perlu dicatat dari Gambar 2.5 bahwa jika (L_{16}, R_{16}) merupakan keluaran dari putaran ke-16, maka (L_{16}, R_{16}) merupakan *pre-ciphertext* dari *enciphering* ini. *Ciphertext* yang sebenarnya diperoleh dengan melakukan melakukan permutasi awal balikan, IP^{-1} terhadap blok *pre - ciphertext*.



Gambar 2. 5 Jaringan *Feistel* untuk satu putaran DES
Sumber : Munir (2006)



Gambar 2. 6 Algoritma enkripsi dengan DES permutasi awal
Sumber : Munir (2006)

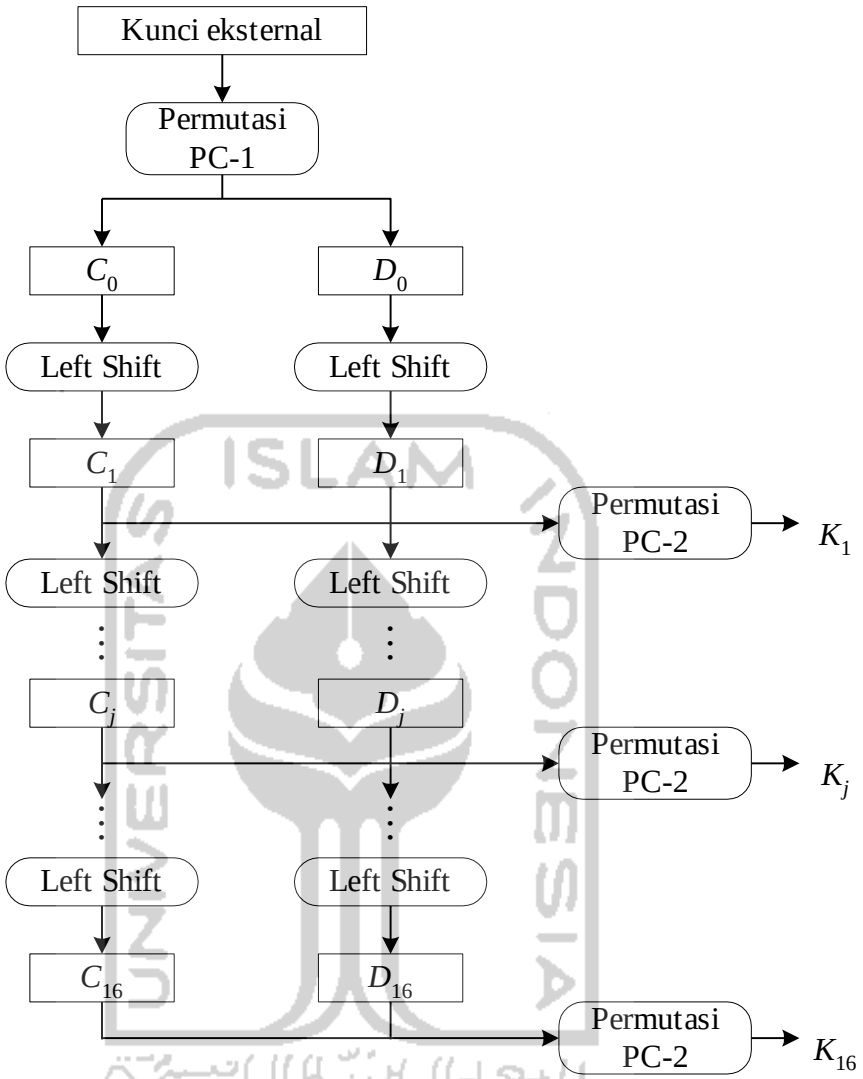
Sebelum putaran pertama, terhadap blok plainteks dilakukan permutasi awal (*initial permutation* atau IP). Tujuan permutasi awal adalah mengacak plainteks sehingga urutan bit-bit di dalamnya berubah.

Karena ada 16 putaran, maka dibutuhkan kunci internal sebanyak 16 buah. Kunci-kunci internal ini dapat dibangkitkan sebelum proses enkripsi atau bersamaan dengan proses enkripsi. Kunci internal dibangkitkan dari kunci eksternal yang diberikan oleh pengguna. Kunci eksternal panjangnya 64 bit atau 8 karakter.

Dalam permutasi ini, tiap bit kedelapan (*parity bit*) dari delapan *byte* kunci diabaikan. Hasil permutasinya adalah sepanjang 56 bit, sehingga dapat dikatakan panjang kunci DES adalah 56 bit. Selanjutnya, 56 bit ini dibagi menjadi 2 bagian, kiri dan kanan, yang masing-masing panjangnya 28 bit, yang masing-masing disimpan di dalam C0 dan D0.

Selanjutnya, kedua bagian digeser ke kiri (*left shift*) sepanjang satu atau dua bit bergantung pada tiap putaran. Operasi pergeseran bersifat *wrapping* atau *round-shift*. Misalkan (C_i,D_i) menyatakan penggabungan C_i dan D_i. (C_i+1, D_i+1) diperoleh dengan menggeser C_i dan D_i satu atau dua bit.

Jadi, setiap kunci internal K_i mempunyai panjang 48 bit. Bila jumlah pergeseran bit-bit pada Tabel 1 dijumlahkan semuanya, maka jumlah seluruhnya sama dengan 28, yang sama dengan jumlah bit pada C_i dan D_i. Karena itu, setelah putaran ke-16 akan didapatkan kembali $C_{16}=C_0$ dan $D_{16}=D_0$.



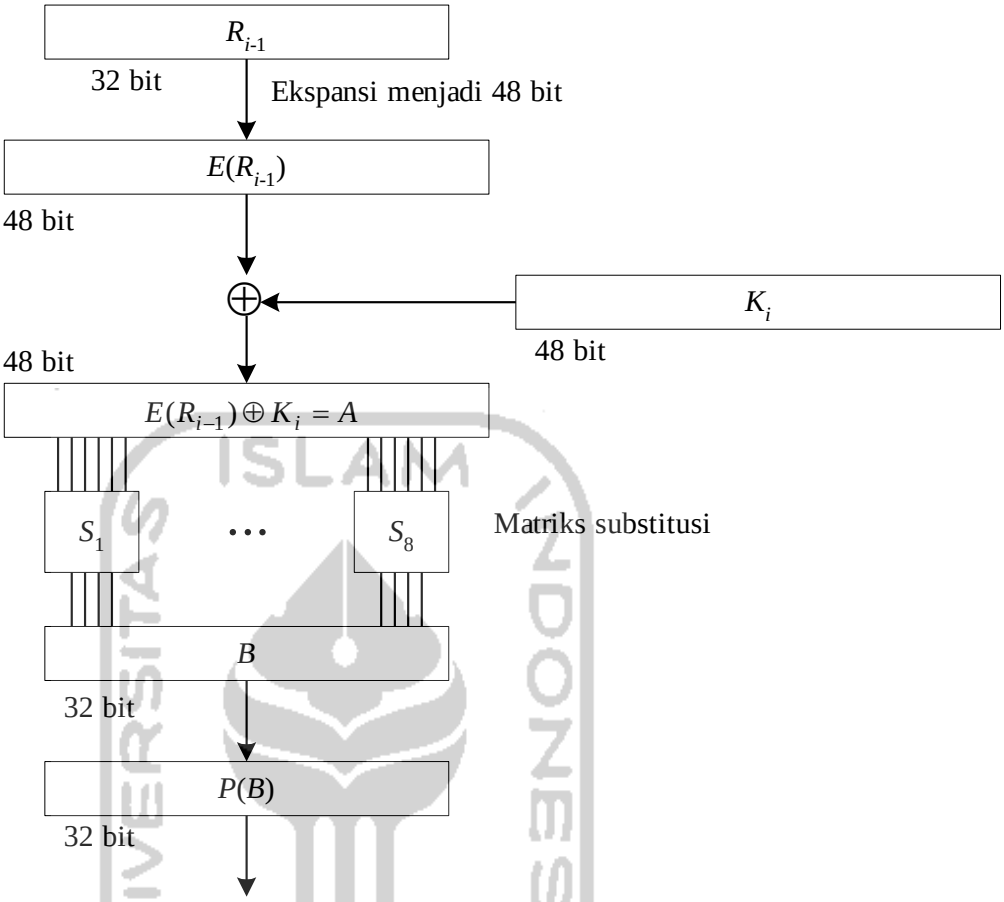
Gambar 2. 7 Proses pembangkitan kunci-kunci internal DES
Sumber : Munir (2006)

Proses *enciphering* terhadap blok plainteks dilakukan setelah permutasi awal. Setiap blok plainteks mengalami 16 putaran *enciphering*. Setiap putaran *enciphering* merupakan jaringan Feistel yang secara matematis dinyatakan sebagai

$$L_i = R_{i-1} \tag{2-3}$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i) \tag{2-4}$$

Diagram komputasi fungsi *f* diperlihatkan pada Gambar 2.8



Gambar 2. 8 Rincian komputasi *f*

Sumber : Munir (2006)

E adalah fungsi ekspansi yang memperluas blok Ri – 1 yang panjangnya 32-bit menjadi blok 48 bit. Selanjutnya, hasil ekspansi, yaitu E(Ri – 1), yang panjangnya 48 bit di-XOR-kan dengan Ki yang panjangnya 48 bit menghasilkan vektor A yang panjangnya 48-bit:

$$E(R_{i-1}) \oplus K_i = A \qquad (2\text{-}5)$$

Vektor A dikelompokkan menjadi 8 kelompok, masing-masing 6 bit, dan menjadi masukan bagi proses subtitusi. Proses subtituasi dilakukan dengan menggunakan delapan buah kotak-S (S-box), S1 sampai S8. Setiap kotak-S menerima masukan 6 bit dan menghasilkan keluaran 4 bit. Kelompok 6-bit pertama menggunakan S1, kelompok 6-bit keda menggunakan S2, dan seterusnya.

Keluaran proses subtitusi adalah vektor B yang panjangnya 48 bit. Vektor B menjadi masukan untuk proses permutasi. Tujuan permutasi adalah untuk mengacak hasil proses subtitusi kotak-S.

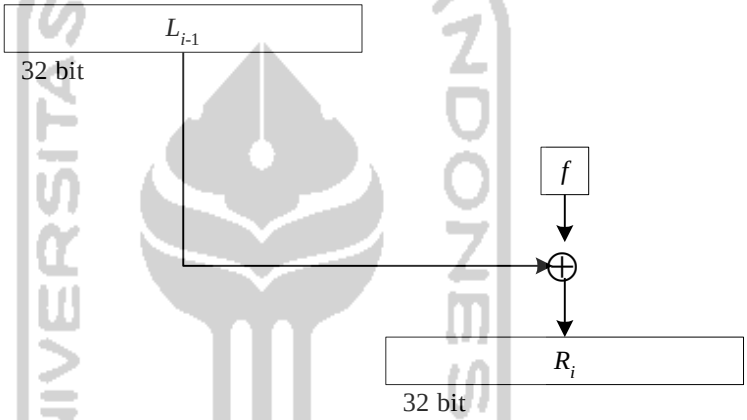
Bit-bit P(B) merupakan keluaran dari fungsi f.

Akhirnya, bit-bit P(B) di-XOR-kan dengan Li – 1 untuk mendapatkan Ri

$$R_i = L_{i-1} \oplus P(B) \quad (2\text{-}6)$$

Jadi, keluaran dari putaran ke-i adalah :

$$(L_i, R_i) = (R_{i-1}, L_{i-1} \oplus P(B)) \quad (2\text{-}7)$$



Gambar 2. 9 Skema perolehan R_1

Sumber : Munir (2006)

Proses dekripsi terhadap cipherteks merupakan kebalikan dari proses enkripsi. DES menggunakan algoritma yang sama untuk proses enkripsi dan dekripsi. Keluaran pada setiap putaran *deciphering* adalah

$$L_i = R_{i-1} \quad (2\text{-}8)$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i) \quad (2\text{-}9)$$

Dalam hal ini, R16, L16) adalah blok masukan awal untuk *deciphering*. Blok (R16, L16) diperoleh dengan mempermutasikan cipherteks dengan matriks permutasi IP-1. Pra-keluaran dari *deciphering* adalah (L0, R0). Dengan permutasi awal IP akan didapatkan kembali blok plainteks semula. Selama *deciphering*, K16 dihasilkan dari (C16, D16) dengan permutasi PC-2. Tentu saja (C16, D16) tidak

dapat diperoleh langsung pada permulaan *deciphering*. Tetapi karena (C16, D16) = (C0,D0), maka K16 dapat dihasilkan dari (C0, D0) tanpa perlu lagi melakukan pergeseran bit. Catatlah bahwa (C0, D0) yang merupakan bit-bit dari kunci eksternal K yang diberikan pengguna pada waktu dekripsi.

Selanjutnya, K15 dihasilkan dari (C15, D15) yang mana (C15, D15) diperoleh dengan menggeser C16 (yang sama dengan C0) dengan D16 (yang sama dengan C0) satu bit ke kanan. Sisanya, K14 sampai K1 dihasilkan dari (C14, D14) sampai (C1, D1). (Ci-1, Di-1) diperoleh dengan menggeser Ci dan Di dengan cara yang sama seperti sebelumnya, tetapi pergeseran kiri (*left shift*) diganti menjadi pergeseran kanan (*right shift*).

2.5 Pemrograman Berorientasi Objek

Pemrograman Berorientasi Objek atau yang biasa disebut *Object-Oriented Programming (OOP)* merupakan paradigma pemrograman yang berorientasi kepada objek. Semua data dan fungsi didalam peradigma ini dibungkus dalam kelas-kelas atau objek-objek. Bandingkan dengan logika pemrograman terstruktur. Setiap objek dapat menerima pesan, memproses data, dan mengirim pesan ke objek lainnya. Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik piranti lunak skala besar. Bahasa pemrograman yang mendukung OOP antara lain Visual Foxpro, Java, C++, Pascal, VisualBasic.NET, PHP.

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

Bahasa pemrograman Java sendiri terbagi menjadi 3, yaitu :

a. J2SE (*Standard Edition*), untuk aplikasi dekstop.
b. J2EE (*Enterprise Edition*), contoh: JSP, Servlet, EJB, XML, web service untuk aplikasi web.
c. J2ME (*Micro Edition*), contoh: Midlet; untuk aplikasi mobile.

2.5.1 J2SE (Standard Edition)

J2SE merupakan teknologi Java yang digunakan untuk membangun aplikasi dekstop. Komponen-komponen J2SE terdiri dari *Java Virtual Machine* (JVM) yang digunakan untuk menjalankan aplikasi Java pada emulator atau *Handeld device, runtime class libraries,* dan *Java application launcher* yang digunakan untuk menjalankan program yang ditulis dengan bahasa pemrograman Java.

2.5.2 *NetBeans IDE 7.0.1*

NetBeans adalah IDE (*Integrated Development Enterprise*) open source yang ditulis sepenuhnya dengan bahasa pemrograman Java. Net beans 7.1 mendukung pengembangan semua tipe aplikasi Java yang dibutuhkan untuk membuat aplikasi dekstop profesional, enterprise, web dan aplikasi mobile dengan bahasa pemrograman Java, C/C++ dan even dynamic language seperti PHP, Javasript, Groovy dan Ruby. Net Beans 7.1 dapat dijalankan di beberapa platform termasuk Windows, Linux, Mac OS dan Solaris.

BAB III
ANALISIS KEBUTUHAN DAN PERANCANGAN PERANGKAT LUNAK

3.1 Analisis Kebutuhan

Analisis merupakan identifikasi terhadap hal-hal yang berkaitan dengan detail atau struktur perancangan aplikasi ini. Dapat dibayangkan sebagai pemecahan sistem yang utuh menjadi bagian-bagian penyusunnya untuk mengetahui permasalahan yang ada. Analisis diperlukan untuk perancangan aplikasi yang akan dibuat dan berhubungan dengan hasil yang ingin dicapai oleh aplikasi itu sendiri.

Metode analisis yang akan digunakan adalah metode analisis dengan pendekatan terstruktur (*struktured approach*) yang lengkap dengan teknik yang dibutuhkan dalam pengembangan sistem. Metode ini dalam pemrosesannya membahas analisis terhadap masukan (*input*) dan keluaran (*output*) yang dinyatakan dengan pembuatan diagram alir (*flowchart*) dalam membuat sistem tersebut. Hasil akhir dari analisis terhadap metode pendekatan terstruktur yang digunakan, diharapkan akan diperoleh sistem yang strukturnya dapat didefinisikan dengan baik dan jelas.

3.2 Hasil Analisis

Dengan dilakukannya analisis tersebut diatas maka dapat diketahui apa yang menjadi kebutuhan masukan, kebutuhan keluaran, kebutuhan proses, kebutuhan antarmuka, kebutuhan perangkat keras dan perangkat lunak.

3.2.1 Analisis Kebutuhan Masukan (*Input*)

Input adalah suatu bentuk yang biasanya berupa data yang telah ada dan dibutuhkan oleh perangkat lunak untuk mencapai tujuan yang diinginkan. Masukan (*input*) yang dibutuhkan untuk meng-implementasikan *digital*

steganography menggunakan metode *Least Significant Bits* dan enkripsi algoritma DES adalah sebagai berikut :

a. Media penampung (*cover image*) berupa berkas *bitmap 24-bit* (*.bmp).
b. Data rahasia yang akan disisipkan dapat berupa teks(*.txt).

3.2.2 Analisis Kebutuhan Keluaran (*Output*)

Berdasarkan analisis kebutuhan masukan, maka sistem yang akan dibangun akan menghasilkan keluaran (*output*). *Output* adalah hasil keluaran dari suatu proses sebuah sistem yang berupa data atau informasi yang telah di olah. Adapun keluarannya berupa berkas *bitmap 24-bit* (*.bmp) yang telah disisipkan data rahasia (*stego image*).

3.2.3 Analisis Kebutuhan Proses

Berdasarkan analisis yang telah dilakukan, ditentukan proses yang akan dilakukan oleh sistem sebagai berikut :

a. Proses *input* data berupa *cover image* dan file yang akan disisipkan.
b. Proses enkripsi menggunakan algoritma DES pada data yang akan disembunyikan. Proses DES menggunakan *library* tambahan yang di *download* di http://commons.apache.org/codec/download_codec.cgi
c. Proses ekstraksi setelah file rahasia disisipkan.

3.2.4 Analisis Kebutuhan Antarmuka

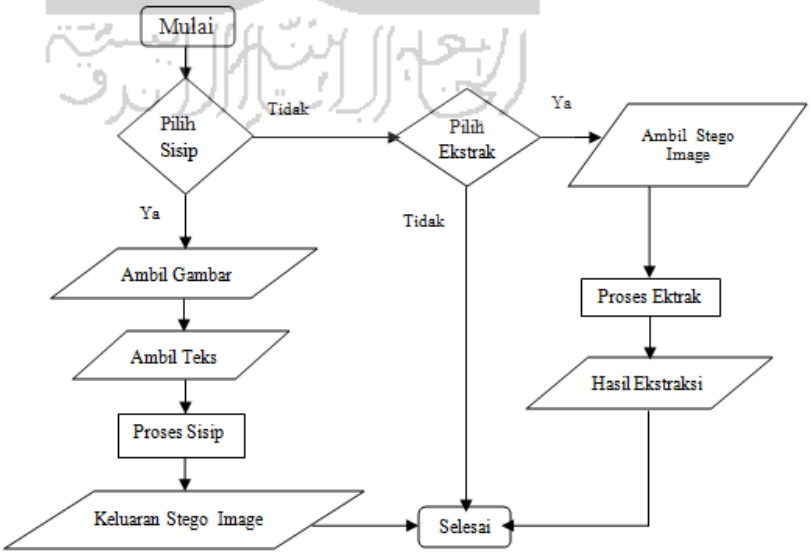
Kebutuhan antarmuka (*interface*) yang ditawarkan sebaiknya bersifat ramah pengguna (*user friendly*), dengan tujuan agar perangkat lunak yang dibangun dapat memudahan pengguna dalam mengoperasikannya. Adapun kebutuhannya sebagai berikut berupa *form* utama yang didalamnya terdiri dari 4 tombol menu, yaitu tombol sisip, ekstrak, dan *browse* dimana 2 diantaranya merupakan tombol menu utama, yaitu tombol sisip dan ekstrak dan 2 tombol lainnya yaitu tombol *browse* yang digunakan untuk mengambil file atau data.

3.3 Perancangan Perangkat Lunak

Untuk melakukan interaksi dengan pengguna atau *user*, program aplikasi yang akan dirancang haruslah mampu berkomunikasi baik dengan pengguna. Untuk merancang suatu program aplikasi tersebut diatas diperlukan perangkat keras (*hardware*), perangkat lunak (*software*) dan aspek manusia itu sendiri (*brainware*). Ketiga aspek tersebut perlu bekerja sama agar sistem komputer dapat bekerja dengan sempurna. Perangkat lunak ini dirancang berdasarkan ketiga aspek tersebut dengan harapan agar tercipta sebuah interaksi manusia dan komputer yang sempurna dan optimal. Pada subbab ini akan dijelaskan bagaimana perancangan perangkat lunak yang akan digunakan untuk membuat program aplikasi ini. Perancangan yang dilakukan adalah percancangan diagram alir sistem (*flowchart*) untuk melihat proses-proses apa saja yang akan dilakukan dan perancangan tampilan antarmuka.

3.3.1 Diagram Alir Menu Utama

Diagram alir menu utama menggambarkan urutan penggunaan program dengan menu-menu yang tersedia, yang menghubungkan ke pilihan-pilihan yang akan ditampilkan. Diagram alir menu utama program implementasi setagnografi ditunjukan pada Gambar 3.1 :

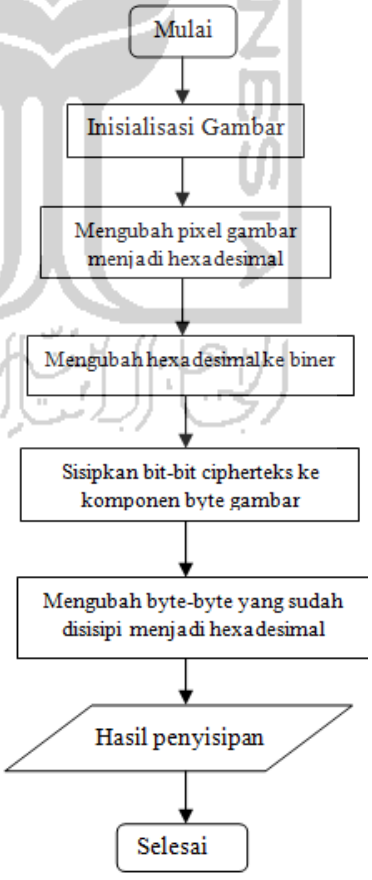


Gambar 3. 1 Diagram Alir Menu Utama

Pada Gambar 3.1 dapat dijelaskan bahwa pada saat program dijalankan akan ditampilan menu utama yang terdiri dari dua pilihan yaitu **Sisip** dan **Ekstrak**. Setelah tampilan GUI muncul, jika pengguna memilih tab menu **Sisip**, maka muncul pilihan menu **Cari** untuk mengambil file yaitu **Input File Asli** berupa file yang akan menjadi *cover image* dan **Input File Yang Akan Disisipkan** berupa file yang akan disisipkan pada gambar. Lalu menu **Sisip** digunakan untuk memproses data menjadi file steganografi. Sedangakan pada tab menu **Ekstrak** pengguna menginputkan file pada **Input File Yang Akan Diekstrak**, lalu pilih menu **Ekstrak** untuk memproses gambar.

3.3.2 Diagram Alir Proses Penyisipan

Diagram alir proses penyisipan atau *embedding* menggambarkan penggunaan menu yang tersedia setelah program dijalankan.

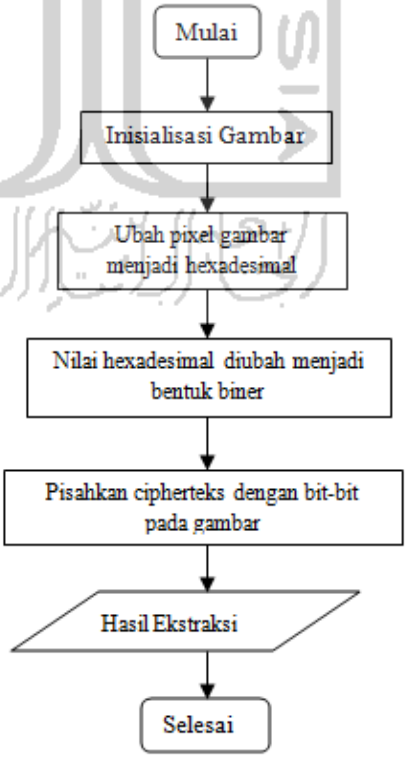


Gambar 3. 2 Diagram alir proses penyisipan citra pesan

Diagram alir pada Gambar 3.2 memperlihatkan proses penyisipan citra dengan bit-bit pesan yang telah di buat *ciphertext* dengan menggunakan algoritma DES. Proses yang pertama kali dilakukan adalah menginisialisasi gambar ,setelah itu gambar yang sudah diinisialisasi diubah dari bentuk pixel menjadi bentuk hexadesimal lalu dibinerkan. Setelah berbentuk biner, lalu disisipkan cipherteks dari file yang akan disisipkan kedalam setiap komponen byte gambar. Setelah terbentuk komponen byte-byte baru, maka diubah lagi menjadi bentuk pixel gambar dan terbentuklah *stego image.*

3.3.3 Diagram Alir Proses Ekstraksi

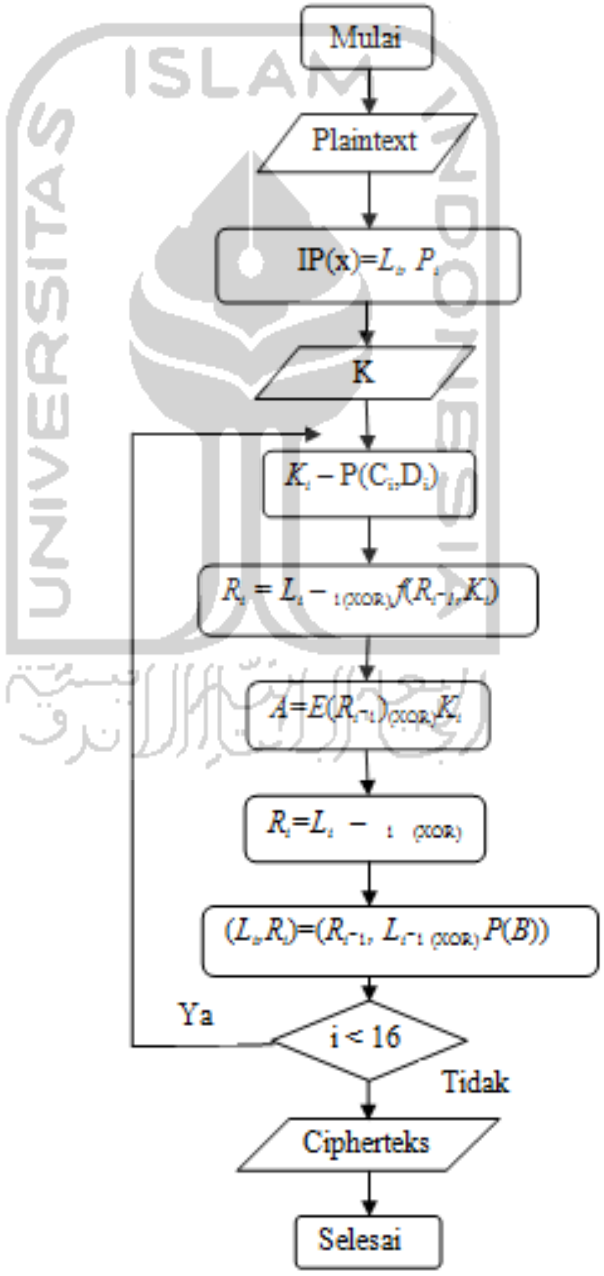
Diagram alir program *extracting* menjelaskan tentang proses yang terjadi pada pengungkapan citra yang telah disisipi pesan. Diagram alir *extracting* citra pesan yaitu proses untuk mengungkapkan citra pesan rahasia yang berbentuk *ciphertext* yang ada didalam citra penampung. Diagram alirnya ditunjukkan pada Gambar 3.3.



Gambar 3. 3 Diagram alir ekstraksi citra pesan

Diagram alir pada Gambar 3.3 memperlihatkan proses ekstraksi citra pesan. Proses ekstraksi hampir sama prosesnya dengan proses penyisipan. Awal proses ektraksi pesan adalah dengan menginisalisasi gambar, lalu gambar yang diinisialisasi diubah menjadi hexadesimal lalu dibinerkan. Setelah berbentuk biner, bit-bit *ciphertext* dipisahkan dengan bit-bit pada *stego image.*

3.3.4 Algoritma Enkripsi dan Dekripsi DES



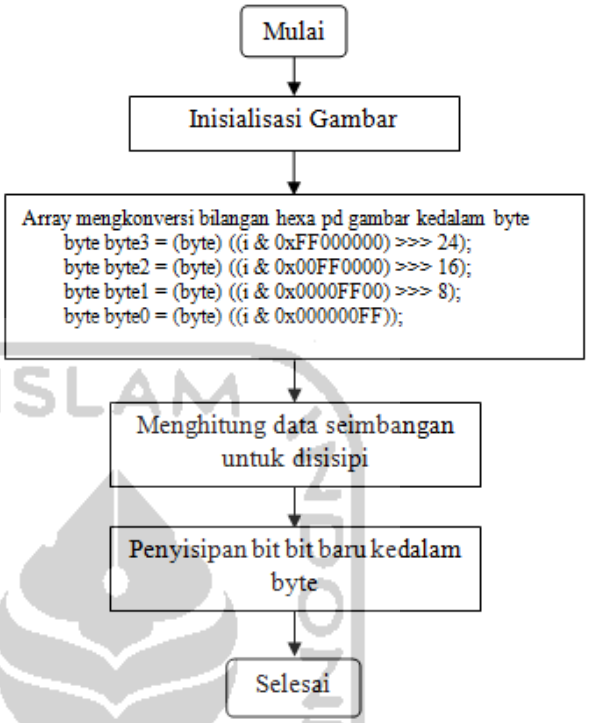
Gambar 3. 4 Diagram alir *Enkripsi* dan *Dekripsi* DES

Diagram alir pada Gambar 3.4 merupakan proses rumus-rumus berkelanjutan seperti yang telah dijelaskan pada BAB II. Proses awal algoritma DES diawali dengan permutasi . Dalam permutasi ini, tiap bit kedelapan (*parity bit*) dari delapan *byte* kunci diabaikan. Hasil permutasinya adalah sepanjang 56 bit, sehingga dapat dikatakan panjang kunci DES adalah 56 bit. Selanjutnya, 56 bit ini dibagi menjadi 2 bagian, kiri dan kanan, yang masing-masing panjangnya 28 bit, yang masing-masing disimpan di dalam C0 dan D0.

Di dalam proses *enciphering*, blok *plaintext* terbagi menjadi dua bagian, kiri (L) dan kanan (R), yang masing-masing panjangnya 32 bit. Kedua bagian ini masuk ke dalam 16 putaran DES yang diinisialisasikan dengan *i*. Pada setiap putaran *i*, blok *R* merupakan masukan untuk fungsi transformasi yang disebut *f*. Pada fungsi *f*, blok *R* dikombinasikan dengan kunci internal K_i. Keluaran dari fungsi *f* di-XOR-kan dengan blok *L* untuk mendapatkan blok *R* sebelumnya. Ini adalah satu putaran DES. Karena ada 16 putaran, maka dibutuhkan kunci internal sebanyak 16 buah. Kunci-kunci internal ini dapat dibangkitkan sebelum proses enkripsi atau bersamaan dengan proses enkripsi. Kunci internal dibangkitkan dari kunci eksternal yang diberikan oleh pengguna. Kunci eksternal panjangnya 64 bit atau 8 karakter.

E adalah fungsi ekspansi yang memperluas blok Ri – 1 yang panjangnya 32-bit menjadi blok 48 bit. Selanjutnya, hasil ekspansi, yaitu E(Ri – 1), yang panjangnya 48 bit di-XOR-kan dengan Ki yang panjangnya 48 bit menghasilkan vektor A yang panjangnya 48-bit. Bit-bit P(B) merupakan keluaran dari fungsi f. Akhirnya, bit-bit P(B) di-XOR-kan dengan Li – 1 untuk mendapatkan Ri . Setelah blok *plaintext* diproses maka proses akan berulang sampai 16 kali sehingga terbentuklah *ciphertext.*

3.3.5 Metode LSB (*Least Significant Bit*)



Gambar 3. 5 Diagram Alir Metode *LSB*

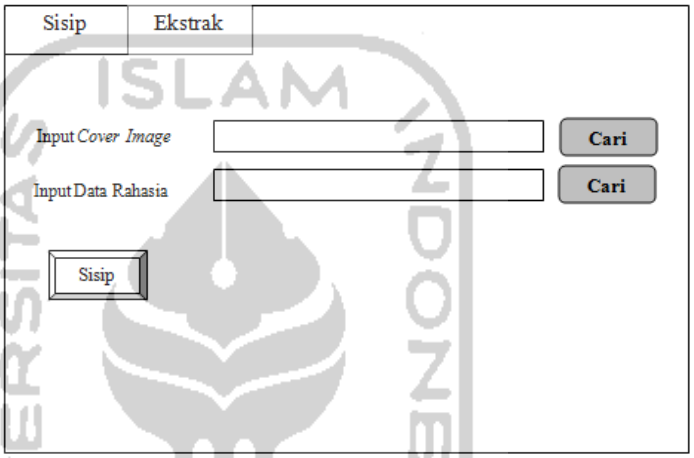
Proses diawali dengan inisialisasi gambar lalu melakukan pembacaan tinggi dari citra digital, setelah itu dilakukan pembacaan baris dan data warna yang dilanjutkan oleh pembacaan lebar citra digital. Proses utama dalam diagram alir ini terdapat pada pengambilan nilai bit-bit dari citra digital untuk diproses kemudian digabungkan dengan bit-bit data.

Proses ini dilakukan pada setiap bit gambar dalam array yang dikonversikan dari nilai hexdesimal setiap warna ke dalam byte lalu dilakukan pengecekan gambar bisa disisipi atau tidak. Jika bisa maka eksekusi dilakukan pergeseran bit untuk menyisipakan pesan pada bit bit tertentu. Untuk membaca datanya, dilakukan suatu pengambilan nilai data yang telah disembunyikan dengan metode *LSB*, langkah yang dilakukan pertama kali yaitu pembacaan *byte-byte* warna pada citra digital, kemudian dilakukan suatu proses yang menggunakan operasi AND dengan bilangan 1 yang memiliki nilai biner 00000001, hal ini bertujuan untuk mengambil nilai 1 bit dari belakang *byte* warna

yang merupakan data yang disembunyikan pada proses penyisipan data, setelah itu data yang telah berhasil dibaca dimasukan ke dalam variabel penampung yang akan dikembalikan dalam bentuk semula.

3.3.6 Rancangan Menu Sisip

Hasil rancangan *form* sisip untuk program steganografi adalah seperti pada Gambar 3.6.

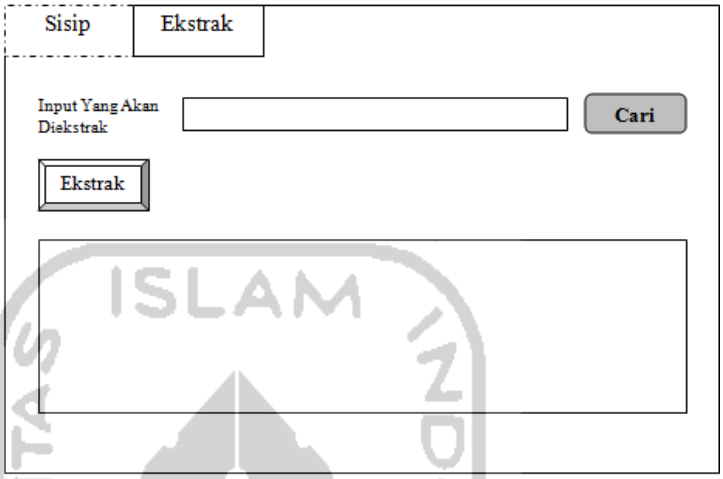


Gambar 3. 6 Rancangan *Form* Sisip

Pada *form* sisip terdapat dua tombol utama yaitu tombol **Sisip** yang merupakan tombol untuk menjalankan fungsi sisip. Terdapat juga **Cari** yang berfungsi untuk mengambil file citra dan file yang akan disisipkan dalam citra.

3.3.7 Rancangan Menu Ekstrak

Hasil rancangan *form* ekstrak untuk program steganografi adalah seperti pada Gambar 3.7.



Gambar 3. 7 Rancangan *Form* Ekstrak

Pada *form* ekstrak terdapat menu utama yaitu tombol **Ekstrak** yang merupakan tombol untuk menjalankan fungsi ekstrak. Terdapat juga **Cari** yang berfungsi untuk mengambil file citra untuk diekstrak.

3.4 Rencana Pengujian Sistem

Rencana pengujian adalah cara atau teknik untuk menguji perangkat lunak yang mempunyai kemungkinan tinggi untuk menemukan kesalahan. Rencana pengujian sistem yang akan dilakukan adalah sebagai berikut :

a. Pengujian dilakukan dengan cara menggunakan data uji untuk menguji semua elemen program. Data uji yang diinputkan untuk mengetahui struktur internal dari perangkat lunak, misalnya dengan input data error pada sistem agar memunculkan pesan error pada sistem.

b. Pengujian dilakukan dengan mengeksekusi hasil dari output sebuah sistem dengan mengubahnya seperti memodifikasi output lalu di eksekusi pada program kembali untuk mengetahui pesan error. Modifikasi yang akan dilakukan untuk menguji ketahanan program adalah dengan cara melakukan rotasi vertikal maupun horisontal dan memodifikasi lighten dan darken pada citra.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Tinjauan Perangkat Lunak

Sebelum membahas lebih lanjut tentang analisis hasil program, berikut akan diuraikan spesifikasi sistem yang akan digunakan dalam menyelesaikan dan menjalankan program ini. Spesifikasi yang akan digunakan adalah :

a. Perangkat keras yang digunakan :

- Processor : Intel(R) Core(TM) i3
- Memori RAM : 2GB
- HardDisk : 320GB
- Monitor : Generic PnP Monitor
- Keyboard
- Mouse

b. Perangkat lunak yang digunakan :

- Sistem Operasi : Windows 7 Ultimate 32-bit
- Netbeans IDE 7.0.1 : *software* utama

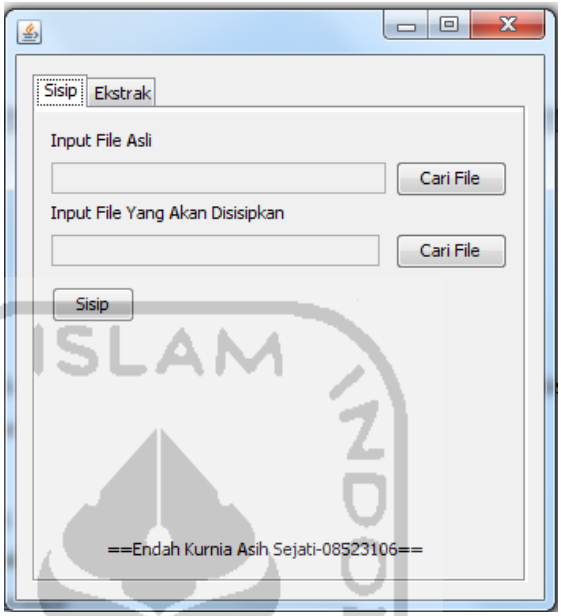
4.2 Implementasi

Implementasi antar muka dibuat sesederhana mungkin sehingga dapat mempermudah pengguna dalam menggunakan sistem. Dalam tahap implementasi ini, sistem akan dioperasikan dalam keadaan yang sesungguhnya. Tujuan dari implementasi ini adalah untuk mengetahui apakah sistem yang dibuat sesuai dengan rancangan awal sistem.

4.2.1 Halaman Depan

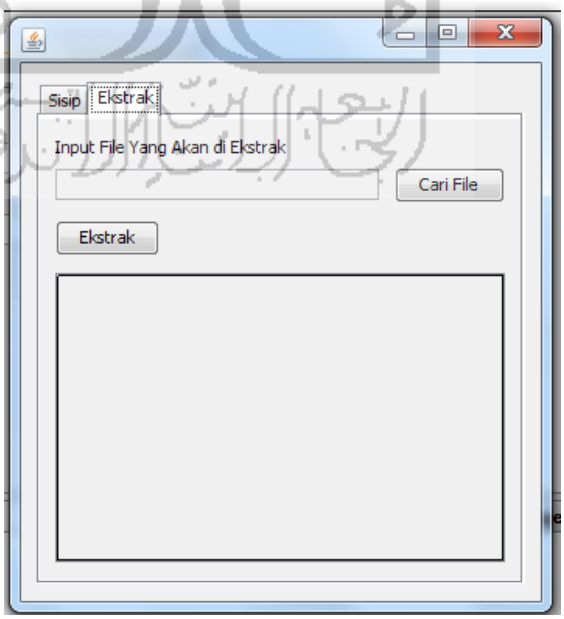
Halaman depan merupakan tampilan yang pertama kali muncul ketika aplikasi ini dijalankan. Melalui halaman depan ini, pengguna dapat memilih apa yang akan dilakukan dalam aplikasi ini. Terdapat 2 pilihan menu yang dapat

dipilih oleh pengguna, yaitu : tab **Sisip** dan tab **Ekstrak**. Implementasi halaman depan dapat dilihat pada Gambar 4.1 dan Gambar 4.2 :



Gambar 4. 1 *Form* Sisip

Pada *form* ini terdapat tab **Sisip**, pengguna meng-*input*kan pada *form* yang tersedia lalu menekan tombol **Sisip**.



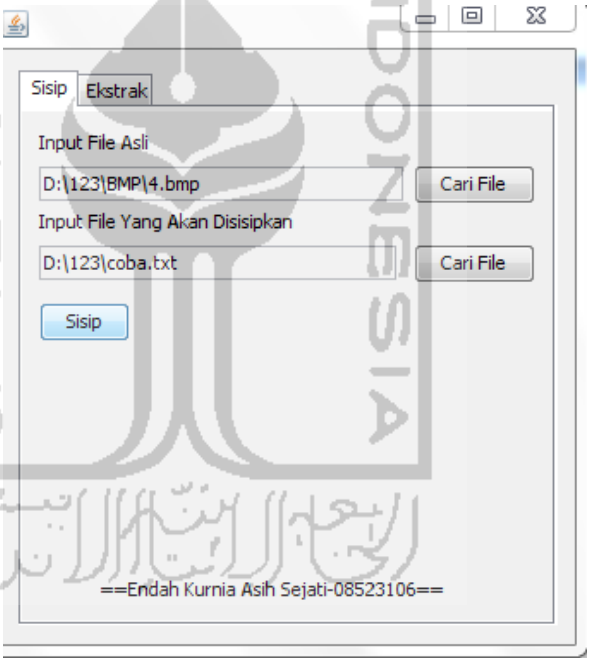
Gambar 4. 2 *Form* Ekstrak

Pada *form* ini terdapat tab **Ekstrak**, pengguna meng-*input*kan pada *form* yang tersedia lalu menekan tombol **Ekstrak**. Setelah proses selesai maka, hasil akan muncul pada Label persegi.

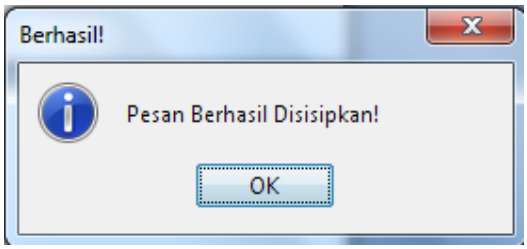
4.3 Hasil Pengujian

4.3.1 Pengujian Pada Menu Sisip

Setelah program di run dan muncul *form* **Sisip**, maka pengguna memasukan *input*-an berupa file gambar berformat *bitmap* pada **Input File Asli** misal file 4.bmp, lalu *input* file berformat .txt pada **Input File Yang Akan Disisipkan** misal file coba.txt. Lalu tekan *button* **Sisip**.



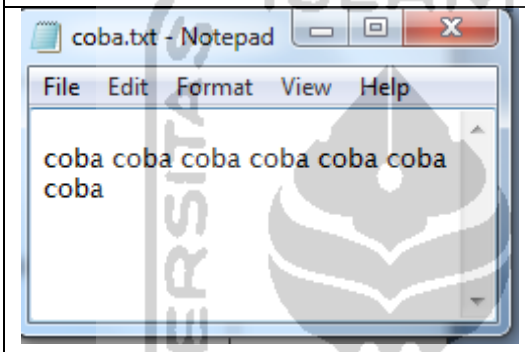
Gambar 4. 3 *Form* penyisipan



Gambar 4. 4 Pesan setelah file berhasil disisipkan

Berikut adalah tabel hasil penyisipan :

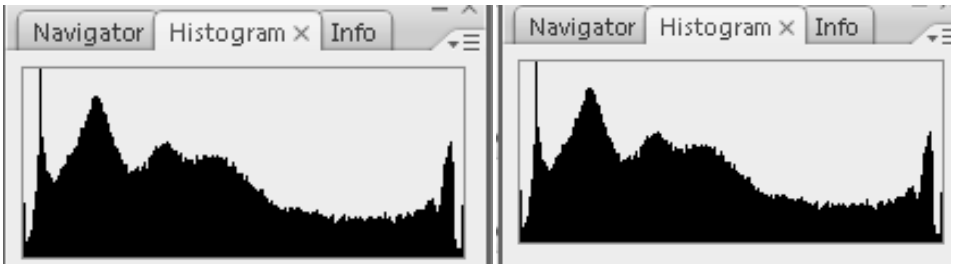
Tabel 4. 1 Penyisipan pesan

Nama File	Keterangan
	File yang akan disisipkan(388KB)
	File Sisip (1KB)
	Hasil penyisipan(388KB)

Pada proses penyisipan pesan berupa file cob.txt terjadi enkripsi menggunakan algoritma DES. Isi file coba.txt adalah " coba coba coba coba coba coba coba" dan hasil enkripsinya :

"fqrtaY+80kxRgnTFh5WOQDL1dG/7OQzTVkGQCpMxLPnUjsLJOljQg".

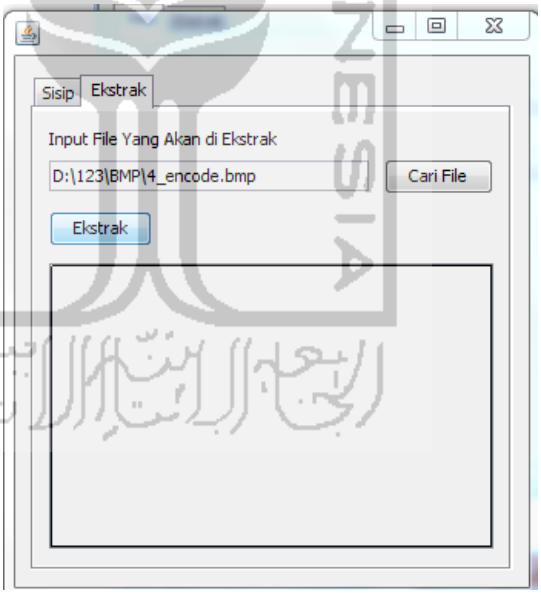
Perbandingan gambar yang asli dan yang telah disisipi terlihat pada histogram dibawah ini :



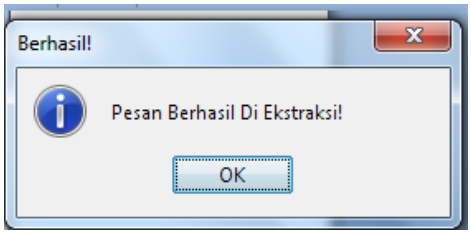
Gambar 4. 5 Histogram gambar asli dan gambar setelah disisipi

4.3.2 Pengujian Pada Menu Ekstrak

Setelah program di run dan muncul *form* **Ekstrak**, maka pengguna memasukan *input*-an berupa file gambar berformat *bitmap* pada **Input File Yang Akan Diekstrak** misal file *bitmap* 4_encode.bmp, lalu tekan *button* **Ekstrak**.



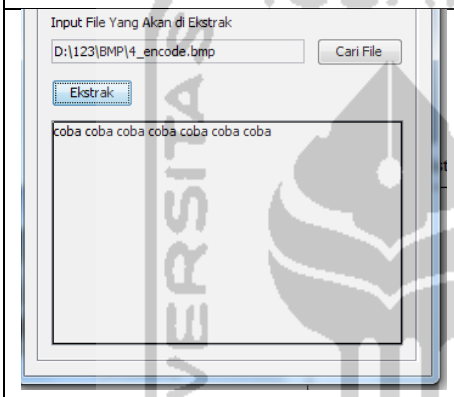
Gambar 4. 6 *Form* Ektraksi



Gambar 4. 7 Pesan Jika Pesan Berhasil Di Ekstraksi

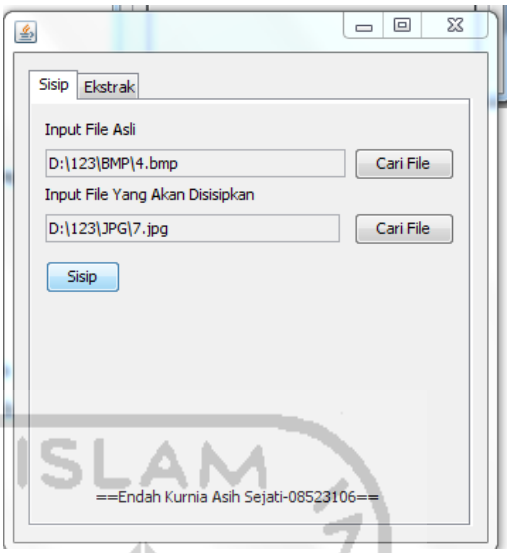
Berikut adalah tabel hasil ekstraksi file :

Tabel 4. 2 Ekstraksi File

Nama File	Keterangan
	File yang akan di ekstraksi
	Hasil ekstraksi

4.3.3 Pengujian Tidak Normal

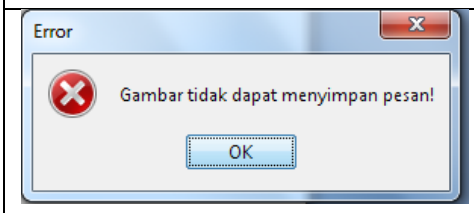
Pengujian dilakukan dengan cara melakukan prosedur pengisian masukan yang tidak benar. Untuk penanganan kesalahan dalam pengisian, perangkat lunak akan memberi keluaran berupa kotak pesan kesalahan pengisian kepada pengguna. Setelah program di run dan muncul *form* **Sisip**, maka pengguna memasukan *input*-an berupa file gambar berformat *bitmap* 4.bmp pada **Input File Asli**, lalu *input* file berformat .jpg pada **Input File Yang Akan Disisipkan**. Lalu tekan *button* **Sisip**.



Gambar 4. 8 *Form* Sisip Error

Berikut adalah tabel hasil pengujian tidak normal :

Tabel 4.3 Tabel Pengujian Tidak Normal

Nama File	Keterangan
	File asli untuk menyimpan pesan
	File yang akan disisipkan
	Hasil keluaran

Jika dilihat dari pesan error diatas dapat disimpulkan perangkat lunak akan memunculkan pesan error saat file yang dimasukan untuk disisipkan tidak sesuai dengan ekstensi yang ditentukan.

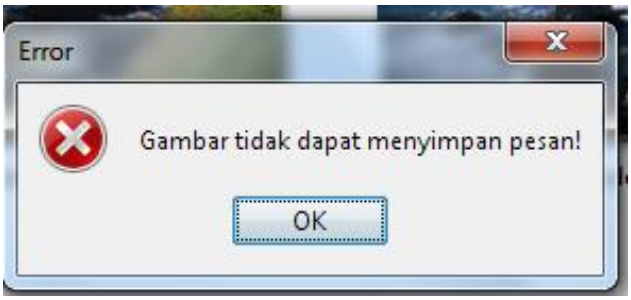
4.3.4 Pengujian Terhadap Ukuran File Sisip

Pengujian juga dilakukan terhadap beberapa ukuran file yang berbeda-beda, seperti pada tabel berikut ini :

Tabel 4. 4 Uji Coba Terhadap Ukuran File Berbeda

No.	Nama File	Ukuran File	Hasil Penyisipan
1.	4.bmp coba.txt	388 KB 1 KB	Berhasil
2.	4.bmp coba3.bmp	388 KB 35 KB	Berhasil
3.	4.bmp coba1.txt	388 KB 36 KB	Berhasil
4.	4.bmp coba4.txt	388 KB 37 KB	Gagal
5.	4.bmp coba2.txt	388 KB 87 KB	Gagal
6.	10.bmp coba2.txt	2250 KB 87 KB	Berhasil
7.	10.bmp coba5.txt	2250 KB 207 KB	Berhasil
8.	10.bmp coba6.txt	2250 KB 225 KB	Gagal
9.	10.bmp coba7.txt	2250 KB 216 KB	Berhasil

Berikut ini keluaran *error* apabila gambar tidak dapat menampung pesan karena besar file sisip terlalu besar :



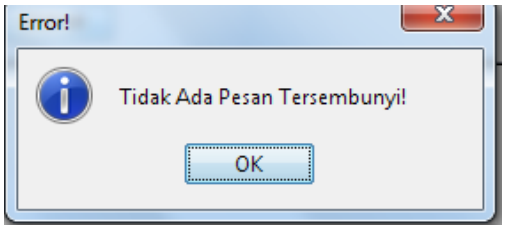
Gambar 4. 9 File Sisip Terlalu Besar

Berdasarkan percobaan-percobaan yang dilakukan, apabila besar ukuran file sisip yang terlalu besar akan memunculkan pesan error. Setelah dilakukan pengujian, didapatkan perbandingan ukuran file asli dan file sisip adalah kurang dari 10:1.

4.3.5 Pengujian Terhadap File Yang Tidak Berisi *Stego Object*

Pengujian dilakukan dengan memasukan file random untuk mengecek suatu gambar terdapat steganografi atau tidak.

Gambar 4. 10 *Form* Pesan Yang Tidak Terdapat Pesan



Gambar 4. 11 Hasil Ektraksi Tidak Ada Pesan

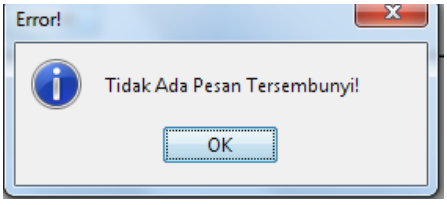
4.3.6 Pengujian Terhadap File Yang Telah Dimanipulasi

Pembahasan kali ini mengenai pengujian terhadap *stego image* yang telah dilakukan proses manipulasi, dimana teknik-teknik yang digunakan untuk memanipulasi *stego image* adalah teknik-teknik yang umum digunakan dalam olah grafis, yaitu *cropping*, rotasi, penambahan efek cerah atau gelap. Pengujian akan dilakukan pada waktu proses pengungkapan kembali dengan materi sebelumnya sudah dilakukan proses penyisipan dimana citra originalnya bernama "4.bmp".

Proses manipulasi pertama adalah *cropping*, dimana *stego image* akan dipotong pada bagian atas dan bawah. Berikut tampilan dari hasil pemotongan :

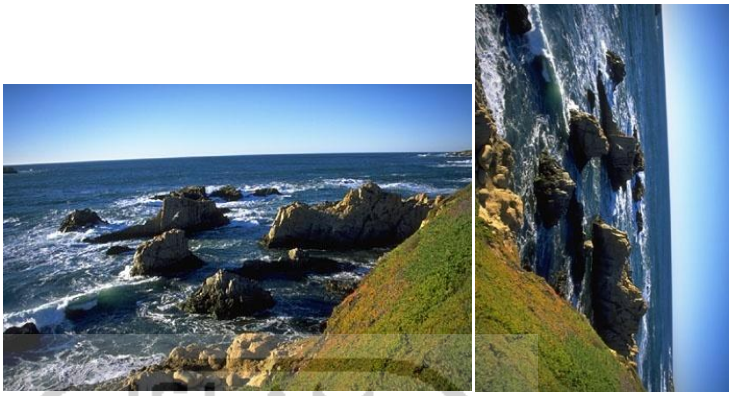
Gambar 4. 12 Gambar asli hasil encoding dan gambar yang di *crop*

Apabila dilakukan proses ekstraksi pada file yang sudah dimanipulasi dengan *cropping*, maka pesan kesalahan yang akan keluar adalah sebagai berikut:



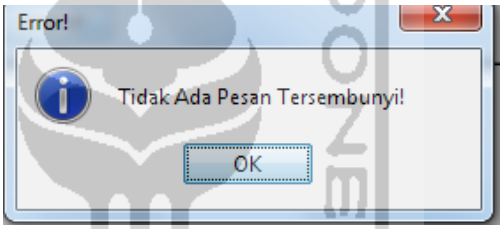
Gambar 4. 13 Pesan *error stego image* yang telah di *cropping*

Untuk manipulasi yang kedua yaitu dengan merotasi 90 derajat.



Gambar 4. 14 Gambar yang terotasi

Apabila dilakukan proses ekstraksi pada file yang sudah dimanipulasi dengan rotasi , maka pesan kesalahan yang akan keluar adalah sebagai berikut:

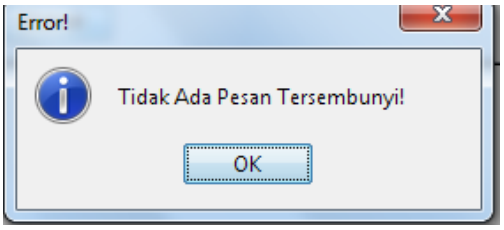


Gambar 4. 15 Pesan kesalahan tidak ada berkas yang disisipkan pada *stego image* yang telah dirotasi

Untuk manipulasi yang ketiga yaitu dengan menambah efek *brightness*.

Gambar 4. 16 Gambar dengan *brightness*

Apabila dilakukan proses ekstraksi pada file yang sudah dimanipulasi dengan menambah efek *brightness* , maka pesan kesalahan yang akan keluar adalah sebagai berikut:



Gambar 4. 17 Pesan kesalahan tidak ada berkas yang disisipkan pada *stego image* yang telah ditambah efek *brightness*

Dari ketiga proses manipulasi didapatkan pesan kesalahan yang sama. Jadi bisa ditarik kesimpulan bahwa apapu teknik manipulasi yang digunakan pada *stego image*, maka perangkat lunak akan menganggap didalam *stego image* tidak terdapat berkas yang disisipkan walaupun sebelum dilakukan proses manipulasi *stego image* tersebut berisi berkas yang disisipkan.

4.4 Pembahasan Kode Program

4.4.1 Algoritma DES

Kode program yang digunakan pada algoritma DES merupakan *reuse* dari library tambahan algoritma DES yang sudah ada pada http://commons.apache.org/codec/download_codec.cgi pada kelas Base64.java dan memanfaatkan library bawaan pada java dengan import javax.crypto.* pada kelas *cipher*.

Berikut ini beberapa kode program yang digunakan dalam algoritma DES :

a. *Class* CryptoDES

```
01. public class CryptoDES {
02.     private Cipher encryptCipher = null;
03.     private Cipher decryptCipher = null;
    // Konstruktor dari class dan mengambil key
04.     public CryptoDES(SecretKey key) throws Exception {
05.         encryptCipher = Cipher.getInstance("DES");
06.         decryptCipher = Cipher.getInstance("DES");
07.         encryptCipher.init(Cipher.ENCRYPT_MODE, key);
08.         decryptCipher.init(Cipher.DECRYPT_MODE, key);}
```

Kode program baris kelima dan keenam untuk memilih algoritma yang digunakan, dan algoritma yang digunakan adalah algoritma DES. Kode

program pada baris ketujuh dan kedelapan merupakan inisialisasi objek cipher dengan modus operasi enkrip dekrip dengan kunci.

b. Kode program yang digunakan untuk memanggil fungsi enkripsi

```
01. public String encryptBase64(String unencryptedString) throws
    Exception {
02.         byte[] unencryptedByteArray =
    unencryptedString.getBytes("UTF8");
03.         byte[] encryptedBytes =
    encryptCipher.doFinal(unencryptedByteArray
04.         byte[] encodedBytes =
    Base64.encodeBase64(encryptedBytes);
05. return new String(encodedBytes);}
```

Pada baris ketiga , pesan yang sudah dienkripsi masih berupa hexadesimal, dan pada baris keempat sudah berupa String dan sudah bisa dibaca pada baris kelima. DES merupakan enkripsi yang menggunakan blok cipher 64bit sehingga kode program menggunakan encodeBase64. Base64 mengkonversikan array byte kedalam string.

Kunci yang digunakan pada saat enkrispsi dan dekripsi sudah ditentukan pada *class* SteganoImage seperti pada baris 02 dibawah.

```
01. //konfigurasi Algoritma DES
02.     String kunci = "00110101";
03.     // convert string kunci menjadi byte
04.     DESKeySpec key = null;
05.     // deklarasi instan kriptografi adalah algoritma DES
06.     SecretKeyFactory keyFactory = null;
07.     CryptoDES kripto = null;
```

c. Kode Program Untuk Memanggil Fungsi Dekripsi

Kode program yang digunakan untuk dekripsi hampir sama dengan pada proses enkripsi, kunci yang digunakan juga sama.

```
01. public String decryptBase64(String encryptedString) throws
    Exception {
02.         byte[] decodedBytes =
    Base64.decodeBase64(encryptedString.getBytes());
03.         byte[] unencryptedByteArray =
    decryptCipher.doFinal(decodedBytes);
04.     System.gc();
05.     return new String(unencryptedByteArray, "UTF8");}
```

4.4.2 Kode Program Steganografi dengan Menggunakan Metode LSB

Berdasarkan *flowchart* pada Gambar 3.7, berikut ini adalah beberapa kode program stegaganografi menggunakan metode LSB :

a. Kode Program Untuk Memilih Gambar Berdasarkan Ekstensinya

```
01. ekstensi = Image_Filter.getExtension(g);
02.       if (ekstensi.equalsIgnoreCase("bmp") {
03. bi = ImageIO.read(g);
04. WritableRaster raster = bi.getRaster();
05. DataBufferByte buffer = (DataBufferByte)
    raster.getDataBuffer();
06. pesan = buffer.getData();}
```

Sesudah gambar dipilih pada *form* **Sisip**, lalu gambar diproses dengan cara mengecek ekstensi dari file tersebut pada *class* Image_Filter, jika file yang dipilih berformat .bmp, maka eksekusi dapat langsung dilanjutkan. Lalu gambar yang sudah sesuai formatnya diproses menggunakan *method* ImageIO.read() yang membutuhkan parameter berupa BufferedImage dan di proses dalam bentuk raster.

b. Kode Program Untuk Mengambil File Pesan

```
01. String temp = null;
02. String textPesan = "";
03. else if (ekstensi.equalsIgnoreCase("txt")) {
04.          BufferedReader br = new BufferedReader(new
    FileReader(g));
05.          while ((temp = br.readLine()) != null) {
             textPesan = textPesan + temp + "\n";}
06.       pesan = kripto.encryptBase64(textPesan).getBytes();
07. System.out.println(new String(pesan));}
```

Jika format file sisip berupa file .txt, maka eksekusi dapat dilanjutkan, lalu file .txt yang sudah diubah menjadi byte diproses menjadi sebuah enkripsi dengan menggunakan kripto.encryptbase seperti pada *coding* dibaris 06 .

c. Proses *Embedding* Pesan

```
if (model.encode(path, name, ext, stegan, pesan)) {
```

Pada cuplikan baris *coding* diatas path merupakan lokasi penyimpanan pesan, name merupakan nama file, ext merupakan ekstensi, stegan merupakan nama dari *stego image* dan pesan adalah isi dari file sisip, jika parameter tersebut sesuai dengan inputan maka pesan akan di *encoding.*

Kode program dari encode :

Berikut ini program encode secara garis besar :

```
01. public boolean encode(String path, String original, String
    ext1, String stegan, byte[] msg) {
02.       String file_name = image_path(path, original, ext1);
03.       BufferedImage image_orig = getImage(file_name);
04.       BufferedImage image = user_space(image_orig);
05.       image = add_text(image, msg);
06. return (setImage(image, new File(image_path(path, stegan,
    "bmp")), "bmp"));}
```

Pada kode add_text merupakan proses penyisipan pesan berupa teks kedalam file *image.*

```
01. private BufferedImage add_text(BufferedImage image, byte[]msg)
{
02.        byte img[] = get_byte_data(image);
03.        byte len[] = bit_conversion(msg.length);
04.        try {
05.             encode_text(img, len, 0); //0 first positiong
06.             encode_text(img, msg, 32); //4 bytes of space
    for length: 4bytes*8bit = 32 bits
07.        } catch (Exception e) {

08.  JOptionPane.showMessageDialog(null,"Gambar tidak dapat
    menyimpan pesan!",
    "Error", JOptionPane.ERROR_MESSAGE);
        }
09.        return image;}
```

Pada koding diatas bit_conversion berfungsi sebagai array yang mengkonversi bilangan hexa pada gambar kedalam byte, banyaknya tanda kurung siku (>) menandakan jumlah pergeseran bit menjadi bit-bit baru

```
01. private byte[] bit_conversion(int i) {
02. //only using 4 bytes
03.        byte byte3 = (byte) ((i & 0xFF000000) >>> 24); //0
04.        byte byte2 = (byte) ((i & 0x00FF0000) >>> 16); //0
05.        byte byte1 = (byte) ((i & 0x0000FF00) >>> 8); //0
06.        byte byte0 = (byte) ((i & 0x000000FF));
07        //{0,0,0,byte0} is equivalent, since all shifts >=8
          will be 0
08.        return (new byte[]{byte3, byte2, byte1, byte0});}
```

Pada koding berikut file sisip dicek muat atau tidaknya seperti pada koding di baris 03, lalu masing-masing bit disisipkan bit baru.

```
01. private byte[] encode_text(byte[] image, byte[] addition,
    int offset) {
02.        //check that the data + offset will fit in the
    image
03.       if (addition.length + offset > image.length) {
04.            throw new IllegalArgumentException("File not
  long enough!");}
```

```
05.   for (int i = 0; i < addition.length; ++i) {
06.            int add = addition[i];
07.            for (int bit = 7; bit >= 0; --bit, ++offset)
08.               int b = (add >>> bit) & 1;
09.                image[offset] = (byte) ((image[offset] &
      0xFE) | b);}}
10.         return image;
```

d. Proses *Extracting* Pesan

Pada umumnya, proses ekstraksi sama dengan proses penyisipan, hanya saja tanda kurung siku yang dibalik (<).

```
01.    public byte[] decode_text(byte[] image) {
02.        try {
03.            int length = 0;
04.            int offset = 32;
05.            for (int i = 0; i < 32; ++i)
06.                length = (length << 1) | (image[i] & 1);}
07.     byte[] result = new byte[length];
08.     for (int b = 0; b < result.length; ++b) {
09.                for (int i = 0; i < 8; ++i, ++offset) {
10.                    result[b] = (byte) ((result[b] << 1) |
       (image[offset] & 1)); }}
11.            return result;
12.   } catch (Exception e) {
13.            return new byte[0];}}
```

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil analisis, perancangan sistem dan pembuatan program sampai dengan tahap penyelesaian, maka dapat ditarik beberapa kesimpulan antara lain sebagai berikut :

a. Implementasi Steganografi Pada Citra Digital Menggunakan Metode *Least Significant Bits* berhasil dibangun.
b. Dengan dibuatnya aplikasi ini yang pengguna dapat menyisipkan file dengan format .txt pada file citra dengan format .bmp.
c. Berdasarkan hasil penyisipan, file asli sebelum disisipkan dan sesudah disisipkan, besar ukurannya sama, karena pada format .bmp tidak akan mengubah ukuran file karena bitmap merupakan jenis file citra tak terkompresi. Ini menunjukan jika format bitmap pada citra tak terkompresi lebih mendukung untuk menghindari kecurigaan pihak lain dalam hal perbedaan ukuran.
d. Pada program ini sangat rentan terhadap manipulasi terhadap file *stego image* seperti rotasi, *cropping*, penambahan *brightness* yang menunjukan bahwa metode LSB sangat rentan terhadap manipulasi gambar.
e. Ukuran file yang akan disisipkan harus lebih kecil daripada file *carrier image*, karena pada metode LSB, hanya bit-bit terakhir saja yang bisa disisipi oleh bit-bit baru yang berasal dari file yang akan disisipkan. Jika dilihat dari hasil pengujian, perbandingan file gambar yang akan disisipi dan file sisip adalah kurang dari 10 : 1.

5.2 Saran

Berdasarkan hasil penelitian yang telah dilakukan banyak sekali hal yang dapat dikembangkan dari aplikasi yang telah dibuat seperti :

a. Jenis file yang akan disisipkan pada program dapat juga menggunakan jenis file lain selain format teks (.txt) seperti gambar, audio maupun video.

b. Pengembangan implementasi juga dapat dicoba dibangun pada *tools* lain, sehingga dapat melengkapi kekurangan-kekurangan yang ada pada *tools* pemograman java.

c. Metode penyisipan lain selain *Least Significant Bit* juga dapat dicoba dikembangan agar kekurangan dalam ukuran penyisipan file dapat teratasi dengan metode lain dan gambar juga lebih tahan terhadap manipulasi seperti yang telah dilakukan pada implementasi ini .

DAFTAR PUSTAKA

Chen, Y.K.L. & L.H. (1999). An Adaptive Image Steganographic Model Based on Minimum-Error LSB Replacement. hal. 1-8.(diakses pada 17 Maret 2012)

CHFI. (2011). Module XX – Steganography. *EC-council*.

Hakim, M. (2007). Studi dan Implementasi Steganografi Metode LSB dengan Preprocessing Kompresi data dan Ekspansi Wadah. *Spectrum*. (http://www.eepis-its.edu/uploadta/downloadmk.php?id=1216 diakses pada 12 Desember 2011)

Irianto. (2004). Embedding Pesan Rahasia Dalam Gambar. Thesis. .(http://budi.insan.co.id/courses/ec7010/2004-2005/irianto-report.pdf diakses pada 13 Desember 2011)

Johnson, N., & Jajodia, S. (1998). Exploring steganography: Seeing the unseen. Computer , hal 26-34.(diakses pada 23 November 2011)

Kessler, G. C. (2004). An Overview of Steganography for the Computer Forensics Examiner. (www.faculty.ksu.edu.sa/ghazy/Steg/References/ref1.pdf diakses pada 23 November 2011)

Munir, R. (2006). Kriptografi. Bandung: Informatika.

Prayudi, Y., & Kuncoro, P. S. (2005). IMPLEMENTASI STEGANOGRAFI. Makalah disampaikan pada Seminar SNATI 2005,hal. 1-6.

Prayudi, Y., & Rahmadhani, B. (2005). STEGANOGRAPHY MENGGUNAKAN TEKNIK LSB PADA. Media Informatika, *x*(x), 231-243. (http://www.mendeley.com/research/steganography-menggunakan-teknik-lsb-pada/ diakses pada 30 Oktober 2011)

Sellars, D. (1996). An Introduction to Steganography. (http://www.zoklet.net/totse/en/privacy/encryption/163947.html diakses pada 23 November 2011)

http://commons.apache.org/codec/download_codec.cgi

http://www.dreamincode.net/forums/topic/27950-steganography/

LAMPIRAN

Kode Program Enkripsi Dekripsi DES

```
package steganografi;

/**
 *
 * @author Endah
 */
import org.apache.commons.codec.binary.Base64;
import javax.crypto.*;

public class CryptoDES {

    private Cipher encryptCipher = null;
    private Cipher decryptCipher = null;

// Konstruktor dari class dan mengambil key
    public CryptoDES(SecretKey key) throws Exception {
        encryptCipher = Cipher.getInstance("DES");
        decryptCipher = Cipher.getInstance("DES");
        encryptCipher.init(Cipher.ENCRYPT_MODE, key);
        decryptCipher.init(Cipher.DECRYPT_MODE, key);
    }

// method untuk proses enkripsi
    public String encryptBase64(String unencryptedString) throws
Exception {
        byte[] unencryptedByteArray = unencryptedString.getBytes("UTF8");
        byte[] encryptedBytes =
encryptCipher.doFinal(unencryptedByteArray);//pesan belum di stringkan
berupa hexa
        byte[] encodedBytes = Base64.encodeBase64(encryptedBytes);
//sudah berupa teks enkripsi
        return new String(encodedBytes);//sudah bisa dibaca
    }

// method untuk proses dekripsi
    public String decryptBase64(String encryptedString) throws Exception
{
        byte[] decodedBytes =
Base64.decodeBase64(encryptedString.getBytes());
        byte[] unencryptedByteArray =
decryptCipher.doFinal(decodedBytes);
        System.gc();
        return new String(unencryptedByteArray, "UTF8");
    }
}
```

Kode Program Steganografi

a. *Class* Image_Filter

```
package steganografi;

/**
 *
 * @author Endah
 */
import java.io.*;

public class Image_Filter extends javax.swing.filechooser.FileFilter
{

	protected boolean isImageFile(String ext)
	{
		return (ext.equals("bmp"));
	}


	public boolean accept(File f)
	{
	    if (f.isDirectory())
	    {
			return true;
	    }

	    String extension = getExtension(f);
		if (extension.equals("bmp"))
		{
			return true;
		}
		return false;
	}


	public String getDescription()
	{
		return "Supported Image Files";
	}

	protected static String getExtension(File f)
	{
		String s = f.getName();
		int i = s.lastIndexOf('.');
		if (i > 0 &&  i < s.length() - 1)
		  return s.substring(i+1).toLowerCase();
		return "";
	}
}
```

b. *Class* SteganoImage

```
package steganografi;

import java.awt.Image;
import java.awt.image.BufferedImage;
import java.awt.image.DataBufferByte;
import java.awt.image.WritableRaster;
import java.io.BufferedReader;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.imageio.ImageIO;
import javax.swing.*;
import javax.swing.filechooser.FileNameExtensionFilter;
import javax.crypto.*;
import javax.crypto.spec.DESKeySpec;
import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;

public class SteganoImage extends javax.swing.JFrame {

    /** Creates new form Stegano2 */
    //JFileChooser JFc1 = new JFileChooser();
    JFileChooser JFc = new JFileChooser();
    JFileChooser chooser = new JFileChooser("./");
    String ekstensi = "";
    int returnVal = 0;
    File f = null;
    File g = null;
    File h = null;
    Image i = null;
    BufferedImage bi = null;
    byte[] pesan = null;
    String stegan = "";
    //konfigurasi Algoritma DES
    String kunci = "00110101";
    // convert string kunci menjadi byte
    DESKeySpec key = null;
    // deklarasi instan kriptografi adalah algoritma DES
    SecretKeyFactory keyFactory = null;
    CryptoDES kripto = null;
    Steganography model = new Steganography();
    //decode variable
    String stat_path = "";
    String stat_name = "";
    String name = "";
    String path = "";
    String ext = "";

    public SteganoImage(Steganography aModel) {
        try {
            initComponents();
            setLocationRelativeTo(null);

            key = new DESKeySpec(kunci.getBytes());
            keyFactory = SecretKeyFactory.getInstance("DES");
            kripto = new CryptoDES(keyFactory.generateSecret(key));
            jTextField1.setEditable(false);
            jTextField2.setEditable(false);
```

```
            jTextField3.setEditable(false);

            FileNameExtensionFilter          filter          =          new
FileNameExtensionFilter("File Text (*.txt)", "txt");

            JFc.addChoosableFileFilter(filter);
            filter  =  new  FileNameExtensionFilter("FileImage  (*.bmp)",
"bmp");

            JFc.addChoosableFileFilter(filter);
        } catch (Exception ex) {

Logger.getLogger(SteganoImage.class.getName()).log(Level.SEVERE,     null,
ex);
        }
    }

    @SuppressWarnings("unchecked")

    public String toBinary(byte b) {
        StringBuilder sb = new StringBuilder("00000000");
        for (int j = 0; j < 8; j++) {
            if (((b >> j) & 1) > 0) {
                sb.setCharAt(7 - j, '1');
            }
        }
        return sb.toString();
    } //merubah byte ke bit

private void tbBrowse1ActionPerformed(java.awt.event.ActionEvent evt) {
    chooser.setFileSelectionMode(JFileChooser.FILES_AND_DIRECTORIES);
    chooser.setFileFilter(new Image_Filter());
    returnVal = chooser.showOpenDialog(this);
    if (returnVal == JFileChooser.APPROVE_OPTION) {
        f = chooser.getSelectedFile();
        jTextField1.setText(f.getPath());
    }
}

private void tbBrowse2ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
    int x = JFc.showOpenDialog(this);
    if (x == JFileChooser.APPROVE_OPTION) {
        g = JFc.getSelectedFile();
        jTextField2.setText(g.getAbsolutePath());
    }
}

private void tbEmbedActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        ekstensi   =   Image_Filter.getExtension(g);   //untuk   mengecek
ekstensi
        String temp = null;
        String textPesan = "";
        if              (ekstensi.equalsIgnoreCase("bmp")              ||
ekstensi.equalsIgnoreCase("jpg")  ||  ekstensi.equalsIgnoreCase("png")  ||
ekstensi.equalsIgnoreCase("gif")) {
            bi = ImageIO.read(g);
            WritableRaster raster = bi.getRaster();
            DataBufferByte        buffer        =        (DataBufferByte)
raster.getDataBuffer();
            pesan = buffer.getData();
        } else if (ekstensi.equalsIgnoreCase("txt")) {
```

```
            BufferedReader br = new BufferedReader(new FileReader(g));
            while ((temp = br.readLine()) != null) {
                textPesan = textPesan + temp + "\n";
            }
            pesan   =   kripto.encryptBase64(textPesan).getBytes();//untuk
memanggil fungsi algoritma enkripsi DES
            System.out.println(new  String(pesan));  //untuk  menampilkan
hasil enkripsi txt
        } else if (ekstensi.equalsIgnoreCase("wav")) {
            wavData = new byte[(int) g.length()];
            ais = AudioSystem.getAudioInputStream(g);
            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            byteRead = ais.read(wavData);
            baos.write(wavData, 0, byteRead);
            wavData = baos.toByteArray();
            pesan = new byte[wavData.length];
            System.arraycopy(wavData, 0, pesan, 0, wavData.length);
        } else {
            JOptionPane.showMessageDialog(this,   "Format   file   tidak
diijinkan!",
                    "Error!", JOptionPane.INFORMATION_MESSAGE);
        }

        ext = Image_Filter.getExtension(f);
        name = f.getName();

        path = f.getPath();

        path = path.substring(0, path.length() - name.length() - 1);
        name = name.substring(0, name.length() - 4);

        stegan = name + "_encode";//nama output setelah disisipkan
        stat_path = path;
        stat_name = stegan;
        if (model.encode(path, name, ext, stegan, pesan)) {
            JOptionPane.showMessageDialog(this,     "Pesan      Berhasil
Disisipkan!",
                    "Berhasil!", JOptionPane.INFORMATION_MESSAGE);

            tbDecode.setEnabled(true);
        } else {
            JOptionPane.showMessageDialog(this,   "Pesan   Tidak   Dapat
Disisipkan!",
                    "Error!", JOptionPane.INFORMATION_MESSAGE);
        }
    } catch (Exception ex) {
        ex.printStackTrace();
    }
}

private void tbDecodeActionPerformed(java.awt.event.ActionEvent evt) {
    try {

        System.out.println("ext: " + ext);
        System.out.println("name1: " + name);
        System.out.println("path1: " + path);
        System.out.println("path2: " + path);
        System.out.println("name2: " + name);
        System.out.println("stegan: " + stegan);
        System.out.println("stat path: " + stat_path);
        System.out.println("stat name: " + stat_name);

        // TODO add your handling code here:
```

```
        if (ekstensi.equalsIgnoreCase("txt")) {
            String message = kripto.decryptBase64(model.decode(stat_path,
stat_name));//proses pemanggilan dekripsi des
            System.out.println(stat_path + ", " + stat_name);
            if (!message.equals("")) {
                JOptionPane.showMessageDialog(this,  "Pesan  Berhasil  Di
Ekstraksi!",
                        "Berhasil!", JOptionPane.INFORMATION_MESSAGE);
                jLabel3.setText(message);
                tbBrowse1.setEnabled(true);
                tbBrowse2.setEnabled(true);
                tbEmbed.setEnabled(true);
                //tbDecode.setEnabled(false);
            } else {
                JOptionPane.showMessageDialog(this, "Pesan Tidak Dapat Di
Ekstraksi!",
                        "Error!", JOptionPane.INFORMATION_MESSAGE);
            }
        }      else      if      (ekstensi.equalsIgnoreCase("bmp")      ||
ekstensi.equalsIgnoreCase("jpg")
                ||          ekstensi.equalsIgnoreCase("png")          ||
ekstensi.equalsIgnoreCase("gif")) {
            String message = kripto.decryptBase64(model.decode(stat_path,
stat_name));
            System.out.println(stat_path + ", " + stat_name);
            if (!message.equals("")) {
                JOptionPane.showMessageDialog(this,  "Pesan  Berhasil  Di
Ekstraksi!",
                        "Berhasil!", JOptionPane.INFORMATION_MESSAGE);
                jLabel3.setText(message);
                tbBrowse1.setEnabled(true);
                tbBrowse2.setEnabled(true);
                tbEmbed.setEnabled(true);
                //tbDecode.setEnabled(false);
            } else {
                JOptionPane.showMessageDialog(this,   "Tidak   Ada   Pesan
Tersembunyi!",
                        "Error!", JOptionPane.INFORMATION_MESSAGE);
            }
        } else {
            JOptionPane.showMessageDialog(this,   "Format   file   tidak
diijinkan!",
                    "Error!", JOptionPane.INFORMATION_MESSAGE);
        }
    } catch (Exception ex) {
        Logger.getLogger(SteganoImage.class.getName()).log(Level.SEVERE,
null, ex);
    }

    System.gc();
}

private void tbBrowse3ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
    chooser.setFileSelectionMode(JFileChooser.FILES_AND_DIRECTORIES);
    chooser.setFileFilter(new Image_Filter());
    returnVal = chooser.showOpenDialog(this);
    if (returnVal == JFileChooser.APPROVE_OPTION) {
        h = chooser.getSelectedFile();
        jTextField3.setText(h.getPath());
    }
```

```
    if ((int) returnVal == 0) {
        try {
            ekstensi = Image_Filter.getExtension(h);

            ext = Image_Filter.getExtension(h);
            name = h.getName();
            path = h.getPath();
            path = path.substring(0, path.length() - name.length() - 1);
            name = name.substring(0, name.length() - 4);

            stegan = name + "_encode";
            stat_path = path;
            stat_name = name;

        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
}

private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
}

    public static void main(String args[]) throws Exception {

        try {

javax.swing.UIManager.setLookAndFeel(javax.swing.UIManager.getSystemLookA
ndFeelClassName());
        } catch (Exception e) {
        }
        java.awt.EventQueue.invokeLater(new Runnable() {

            @Override
            public void run() {
                new SteganoImage(new Steganography()).setVisible(true);
            }
        });
    }
    // Variables declaration - do not modify
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JPanel jPanel4;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTabbedPane jTabbedPane1;
    private javax.swing.JTextField jTextField1;
    private javax.swing.JTextField jTextField2;
    private javax.swing.JTextField jTextField3;
    private javax.swing.JButton tbBrowse1;
    private javax.swing.JButton tbBrowse2;
    private javax.swing.JButton tbBrowse3;
    private javax.swing.JButton tbDecode;
    private javax.swing.JButton tbEmbed;
    // End of variables declaration
}
```

c. *Class* Steganography

```
package steganografi;

import java.io.File;
import java.awt.Graphics2D;
import java.awt.image.BufferedImage;
import java.awt.image.WritableRaster;
import java.awt.image.DataBufferByte;

import javax.imageio.ImageIO;
import javax.swing.JOptionPane;

public class Steganography {
    public Steganography() {
    }

    public boolean encode(String path, String original, String ext1,
String stegan, byte[] msg) {
        String file_name = image_path(path, original, ext1);
        BufferedImage image_orig = getImage(file_name);
        BufferedImage image = user_space(image_orig);
        image = add_text(image, msg);

        return (setImage(image, new File(image_path(path, stegan,
"bmp")), "bmp"));
    }


    public String decode(String path, String name) throws Exception {
        byte[] decode;
        try {
            System.gc();
            BufferedImage image = user_space(getImage(image_path(path,
name, "bmp")));
            decode = decode_text(get_byte_data(image));

            image.flush();
            System.gc();
            return (new String(decode));
            //return String.valueOf(decode);
        } catch (Exception e) {
            System.gc();
            e.printStackTrace();
            JOptionPane.showMessageDialog(null,
                    "Tidak ada pesan tersembunyi dalam gambar ini!",
"Error",
                    JOptionPane.ERROR_MESSAGE);
            return "";
        }
    }

    public String image_path(String path, String name, String ext) {
        return path + "\\" + name + "." + ext;
    }
    public BufferedImage getImage(String f) {
        BufferedImage image = null;
        File file = new File(f);

        try {
            image = ImageIO.read(file);
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(null,
```

```
                    "Gambar        tidak        terbaca!",        "Error",
JOptionPane.ERROR_MESSAGE);
        }
        return image;
    }

    private boolean setImage(BufferedImage image, File file, String ext)
{
        try {
            file.delete(); //delete resources used by the File
            ImageIO.write(image, ext, file);
            return true;
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null,
                    "Tidak        dapat        disimpan!",        "Error",
JOptionPane.ERROR_MESSAGE);
            return false;
        }
    }


    private BufferedImage add_text(BufferedImage image, byte[] msg) {
        //convert  all  items  to  byte  arrays:  image,  message,  message
length
        byte img[] = get_byte_data(image);
        //byte msg[] = text.getBytes();
        byte len[] = bit_conversion(msg.length);
        try {
            encode_text(img, len, 0); //0 first positiong
            encode_text(img,  msg,  32);  //4  bytes  of  space  for  length:
4bytes*8bit = 32 bits
        } catch (Exception e) {

            JOptionPane.showMessageDialog(null,
                    "Gambar  tidak  dapat  menyimpan  pesan!",  "Error",
JOptionPane.ERROR_MESSAGE);
        }
        return image;
    }

    public BufferedImage user_space(BufferedImage image) {
        //create new_img with the attributes of image
        BufferedImage  new_img  =  new  BufferedImage(image.getWidth(),
image.getHeight(), BufferedImage.TYPE_3BYTE_BGR);
        Graphics2D graphics = new_img.createGraphics();
        graphics.drawRenderedImage(image, null);
        graphics.dispose(); //release all allocated memory for this image
        new_img.flush();
        System.gc();
        return new_img;
    }

    public byte[] get_byte_data(BufferedImage image) {
        WritableRaster raster = image.getRaster();
        DataBufferByte buffer = (DataBufferByte) raster.getDataBuffer();
        return buffer.getData();
    }


    private byte[] bit_conversion(int i) {
        //originally integers (ints) cast into bytes
        //byte byte7 = (byte)((i & 0xFF00000000000000L) >>> 56);
        //byte byte6 = (byte)((i & 0x00FF000000000000L) >>> 48);
```

```
        //byte byte5 = (byte)((i & 0x0000FF0000000000L) >>> 40);
        //byte byte4 = (byte)((i & 0x000000FF00000000L) >>> 32);

        //only using 4 bytes
        byte byte3 = (byte) ((i & 0xFF000000) >>> 24); //0
        byte byte2 = (byte) ((i & 0x00FF0000) >>> 16); //0
        byte byte1 = (byte) ((i & 0x0000FF00) >>> 8); //0
        byte byte0 = (byte) ((i & 0x000000FF));
        //{0,0,0,byte0} is equivalent, since all shifts >=8 will be 0
        return (new byte[]{byte3, byte2, byte1, byte0});
    }

    private byte[] encode_text(byte[] image, byte[] addition, int offset)
{
        if (addition.length + offset > image.length) {
            throw new IllegalArgumentException("File not long enough!");
        }
        for (int i = 0; i < addition.length; ++i) {
            int add = addition[i];
            for (int bit = 7; bit >= 0; --bit, ++offset) //ensure the new
offset value carries on through both loops
            {

                int b = (add >>> bit) & 1;

                image[offset] = (byte) ((image[offset] & 0xFE) | b);
            }
        }
        return image;
    }

    public byte[] decode_text(byte[] image) {
        try {
            int length = 0;
            int offset = 32;

            for (int i = 0; i < 32; ++i) //i=24 will also work, as only
the 4th byte contains real data
            {
                length = (length << 1) | (image[i] & 1);
            }

            byte[] result = new byte[length];

            for (int b = 0; b < result.length; ++b) {
                for (int i = 0; i < 8; ++i, ++offset) {
                    result[b] = (byte) ((result[b] << 1) | (image[offset]
& 1));
                }
            }
            return result;

        } catch (Exception e) {
            return new byte[0];
        }
    }
}
```