STEGANOGRAFI *CIPHERTEXT* PADA CITRA DIGITAL MENGGUNAKAN LSB

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar Sarjana dalam bidang Ilmu Komputer

Oleh:
M ZIKI ELFIRMAN
0410963033-96

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG

2010

UNIVERSITAS BRAWIJAYA

LEMBAR PENGESAHAN SKRIPSI

STEGANOGRAFI *CIPHERTEXT* PADA CITRA DIGITAL MENGGUNAKAN LSB

Oleh :
M ZIKI ELFIRMAN
0410963033-96

Setelah dipertahankan di depan Majelis Penguji
pada tanggal 6 Desember 2010
dan dinyatakan memenuhi syarat untuk memperoleh gelar Sarjana Komputer dalam bidang Ilmu Komputer

Pembimbing I,	**Pembimbing II,**
Edy Santoso, SSi., M.Kom	**Muhammad Tanzil Furqon, S.Kom**
197404142003121004	198209302008011004

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya

Dr. Agus Suryanto, MSc
196908071994121001

LEMBAR PERNYATAAN

STEGANOGRAFI *CIPHERTEXT* PADA CITRA DIGITAL MENGGUNAKAN LSB

Saya yang bertanda tangan di bawah ini :

Nama	**:** M Ziki Elfirman
NIM	**:** 0410963033-96
Jurusan	**:** Matematika
Program Studi	**:** Ilmu Komputer
Penulis skripsi berjudul	**:** Steganografi *Ciphertext* Pada Citra Digital Menggunakan LSB

Dengan ini menyatakan bahwa :

1. **Isi dari skripsi yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain, selain nama-nama yang termaktub di isi dan tertulis di daftar pustaka dalam skripsi ini.**
2. **Apabila dikemudian hari ternyata skripsi yang saya tulis terbukti hasil jiplakan, maka saya akan bersedia menanggung segala resiko yang akan saya terima.**

Demikian pernyataan ini dibuat dengan segala kesadaran

Malang, 6 Desember 2010
Yang menyatakan,

M Ziki Elfirman
NIM. 0410963033-96

STEGANOGRAFI *CIPHERTEXT* PADA CITRA DIGITAL MENGGUNAKAN LSB

ABSTRAK

Perkembangan teknologi memberikan kemudahan dalam hal berkomunikasi dan bertukar informasi baik jarak jauh maupun jarak dekat. Seiring dengan perkembangan ini diperlukan suatu teknik pengamanan data yang mempunyai tingkat keamanan yang tinggi. Teknik keamanan data yang umum digunakan merupakan pengembangan dari teknik kriptografi atau teknik steganografi yang memiliki kelemahan dan kelebihan masing – masing. Penggabungan dua teknik keamanan data kriptografi dan steganografi diharapkan akan menutupi kelemahan pada masing – masing metode.

3DES merupakan salah satu algoritma kriptografi yang mempunyai tingkat keamanan yang tinggi. Kekuatan algoritma 3DES terletak pada kunci yang digunakan yaitu 168 bit, sehingga jumlah seluruh kombinasi kemungkinan kunci yang harus dicoba untuk memecahkan *ciphertext* adalah 2^{168}. Kelemahan metode kriptografi akan tertutupi dengan penggunaan metode steganografi yang menyembunyikan pesan ke dalam sebuah *file* citra. Penggunaan teknik LSB-*insertion* karena penurunan kualitas citra yang dihasilkan tidak tampak secara kasat mata.

Pengimplementasian 3DES pada teks kemudian dilanjutkan dengan penggunaan teknik steganografi LSB-*insertion* menghasilkan citra steganografi yang memiliki nilai rata – rata perbedaan *pixel* diatas 50% dibandingkan dengan citra aslinya. Hal ini dapat dilihat pada grafik nilai PSNR yang memiliki nilai rata – rata yang cukup tinggi yaitu 64,5 dari enam kali pengujian. Semakin besar persentase bit yang tidak terpakai maka semakin tinggi nilai PSNR yang dihasilkan. Setelah dilakukan penelitian, diketahui ketahanan citra steganografi yang dihasilkan terhadap perubahan sementara *format* citra ke *format* citra yang lain.

STEGANOGRAPHY CIPHERTEXT AT DIGITAL IMAGE USING LSB

ABSTRACT

Technological development provides simplicity in terms of communicating and exchanging information on both distance and close range. Along with this development required a data security technique that has a high level of security. Data security techniques commonly used is the development of *cryptography* techniques or *steganography* techniques which have an advantage and disadvantage respectively. Combining two techniques *cryptography* and *steganography* of data security is expected to cover up weaknesses on each method.

3DES is one of the *cryptography* algorithms that have a high level of security. The strength lies in the key 3DES algorithm used is 168 bits, so that the sum of all possible key combinations to try to solve the *ciphertext* is 2^{168}. The weakness of *cryptography* method will be covered by the use of *steganography* is to hide *message* into an *image* file. The use of LSB-*insertion* technique because *image* quality degradation produced is not visible by naked eye.

3DES implementation on the text and then continued with the use of *steganography* LSB-*insertion* technique produces an *image steganography* that has an average value of pixel difference above 50% compared with the original *image*. It can be seen on the graph of PSNR average value is quite high at 64.5 on six times test. The greater percentage of unused bits, the higher result of PSNR value. After doing the research, known *steganography images* generated resistance to the temporary *image* format change to another *image* format.

KATA PENGANTAR

Puji syukur ke hadirat Allah SWT yang telah melimpahkan rahmat dan hidayah-Nya kepada penulis, sehingga penulis dapat menyelesaikan tugas akhir ini sebagai salah satu syarat untuk memperoleh gelar Sarjana dalam bidang Ilmu Komputer.

Skripsi yang berjudul "Steganografi *Ciphertext* Pada Citra Digital Menggunakan LSB" ini dibuat sebagai syarat kelulusan tingkat sarjana.

Selama melaksanakan Skripsi ini, penulis mendapat bantuan dan dukungan dari banyak pihak. Untuk itu, penulis ingin memberikan terima kasih kepada:

1. Edy Santoso, S.Si., M.Kom., selaku pembimbing utama dalam penulisan tugas akhir.
2. Tanzil Furqon, S.Kom., selaku pembimbing pendamping dalam penulisan tugas akhir.
3. Djoko Pramono, S.T., selaku dosen penasehat akademik, yang mengantarkan perjalanan panjang penulis selama masa studi di Ilmu Komputer.
4. Drs. Marji, MT., selaku Ketua Program Studi Ilmu Komputer, Jurusan Matematika, FMIPA Universitas Brawijaya.
5. Dr. Agus Suryanto, MSc., selaku Ketua Jurusan Matematika, FMIPA Universitas Brawijaya.
6. Segenap bapak dan ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada penulis selama menempuh pendidikan di Program Studi Ilmu Komputer Jurusan Matematika FMIPA Universitas Brawijaya.
7. Segenap staf dan karyawan di Jurusan Matematika FMIPA Universitas Brawijaya yang telah banyak membantu penulis dalam pelaksanaan penyusunan skripsi ini.
8. Ayah, ibu dan kakak penulis, yang mendukung serta mencurahkan kasihnya.
9. Dina Inshirna yang terus mensuport penulis untuk menyelesaikan skripsi ini.
10. Rekan-rekan di Program Studi Ilmu Komputer FMIPA Universitas Brawijaya yang telah banyak memberikan bantuannya demi kelancaran pelaksanaan penyusunan skripsi ini.
11. Dan semua pihak yang telah membantu dalam penyusunan skripsi ini yang tidak dapat penulis sebutkan satu per satu.

Semoga Allah SWT membalas budi baik semua pihak dengan rahmat-Nya yang meliputi segala sesuatu.

Akhir kata, penulis menyadari bahwa Skripsi ini bukanlah tanpa kelemahan, untuk itu kritik dan saran sangat diharapkan.

Malang, 6 Desember 2010

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR *SOURCECODE*

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Berkat perkembangan teknologi yang begitu pesat memungkinkan manusia dapat berkomunikasi dan saling bertukar informasi atau data secara jarak jauh. Seiring dengan kemudahan yang ada tuntutan akan sekuritas (keamanan) terhadap kerahasiaan informasi yang saling dipertukarkan tersebut semakin meningkat. Begitu banyak pengguna seperti departemen pertahanan, suatu perusahaan atau bahkan individu – individu tidak ingin informasi yang disampaikannya diketahui oleh orang lain atau kompetitornya. Oleh karena itu dikembangkanlah cabang ilmu yang mempelajari tentang cara – cara pengamanan data.

Dalam dunia keamanan data, istilah kriptografi sudah sangat dikenal. Kriptografi merupakan ilmu sekaligus seni untuk menjaga keamanan pesan (Schneier, 1996). Karena kriptografi merubah pesan menjadi tidak dimengerti lagi maknanya maka teknik kriptografi juga biasa disebut teknik penyandian. Salah satu algoritma teknik penyandian pesan yang sering digunakan adalah algoritma *triple data encryption standart* (3DES). Algoritma 3DES dikembangkan dari algoritma DES yang pernah menjadi standard untuk melindungi data dan informasi setelah disetujui oleh *National Bureau of Standard* (NBS) dan dinilai kekuatannya oleh *National Security Agency* (NSA). Kekuatan algoritma 3DES terletak pada kunci yang digunakan yaitu 168 bit. Karena ada 168 posisi pengisian bit yang masing – masing mempunyai dua nilai kemungkinan, yaitu 0 dan 1, maka jumlah seluruh kombinasi kemungkinan kunci yang harus dicoba untuk memecahkan *ciphertext* (pesan yang sudah disandikan) adalah 2^{168} kali (Akik Hidayat, 2008). Penggunaan kunci dengan panjang 168 bit ini membuat tingkat keamanan 3DES sangat tinggi.

Meskipun demikian, kriptografi saja tidak cukup untuk memenuhi kebutuhan akan pengamanan data. Pengamanan data dengan kriptografi memiliki kelemahan pada bentuk pesan yang tersandi dan keberadaan pesan yang dapat terdeteksi langsung oleh indera manusia.

Berbeda dengan kriptografi, teknik pengamanan data steganografi adalah ilmu dan seni menyembunyikan pesan sehingga

keberadaan pesan tidak terdeteksi oleh indera manusia (Rinaldi, 2004). Dalam penggunaannya, steganografi memerlukan sebuah media penampung untuk menyembunyikan pesan rahasia. Media penampung yang paling sering digunakan adalah media dalam bentuk citra digital (gambar). Terdapat banyak metode dalam steganografi salah satunya metode LSB-*insertion*. Metode LSB-*insertion* merupakan metode steganografi yang paling sering digunakan, hal ini disebabkan karena kemudahannya dalam implementasi.

Seiring dengan perkembangan dan kemajuan teknologi teknik pengamanan data kriptografi maupun steganografi saja tidak dapat memenuhi kebutuhan akan keamanan data yang bersifat pribadi. Dengan memanfaatkan kemajuan teknologi, pengambilan data secara ilegal semakin sering terjadi dan sangat merugikan. Banyak pihak yang tidak bertanggung jawab berusaha mengakses data orang lain yang bersifat rahasia. Permasalahan ini menjadi dasar pemikiran untuk menggabungkan dua teknik pengamanan data, yaitu teknik kriptografi dan steganografi. Dalam prosesnya pesan akan disandikan terlebih dahulu untuk kemudian disisipkan ke dalam media penampung sehingga keberadaan pesan tidak dapat terdeteksi oleh indera manusia. Dengan penggabungan dua teknik ini diharapkan tingkat keamanan data semakin tinggi sehingga dapat memenuhi kebutuhan akan keamanan data.

Berdasarkan latar belakang yang telah dikemukakan, judul yang diambil dalam tugas akhir ini adalah "Steganografi *Ciphertext* Pada Citra Digital Menggunakan LSB".

1.2 Rumusan Masalah

Beberapa permasalahan yang menjadi titik utama pembahasan dalam pelaksanaan skripsi ini adalah:

1. Bagaimana mengimplementasikan teknik pengamanan data kriptografi algoritma 3DES dan teknik pengamanan data steganografi pada sebuah perangkat lunak pengamanan data.
2. Bagaimana kualitas citra yang dihasilkan setelah disisipi data rahasia jika dibandingkan dengan citra aslinya.
3. Bagaimana ketahanan citra terhadap manipulasi citra yang umum dilakukan.

1.3 Batasan Masalah

Untuk menghindari melebarnya pembahasan yang akan diselesaikan, diberikan batasan masalah sebagai berikut :

1. *Ciphertext* yang digunakan merupakan *ciphertext* dari proses pengenkripsian menggunakan algoritma 3DES.
2. Citra digital yang digunakan dalam pengimplementasian steganografi hanya citra dengan *format bitmap* 24 bit (*.bmp).
3. Teknik steganografi yang digunakan hanya dapat menyimpan pesan rahasia yang berupa data teks (*.txt).
4. Data yang dihasilkan hanya dapat berupa citra dengan *format bitmap* 24 bit (*.bmp).

1.4 Tujuan Penelitian

Berdasarkan pada masalah yang telah diidentifikasikan, maka tujuan yang hendak dicapai skripsi ini adalah:

1. Menghasilkan perangkat lunak untuk pengamanan data yang mengimplementasikan dua teknik pengamanan data kriptografi dengan algoritma 3DES dan steganografi dengan metode LSB *insertion.*
2. Citra steganografi yang dihasilkan dari proses penyisipan mempunyai tingkat kemiripan yang tinggi dengan citra asli.
3. Mengetahui ketahanan citra terhadap proses manipulasi citra yang umum dilakukan.

1.5 Manfaat Penelitian

Dengan adanya aplikasi kriptografi dan steganografi ini diharapkan informasi rahasia yang akan dipertukarkan melalui berbagai media hanya diketahui dan dapat diakses oleh yang berkepentingan saja.

1.6 Metodologi

Penyusunan skripsi ini menggunakan metodologi sebagai berikut :

1. Studi Kepustakaan
   Studi ini dilakukan dengan cara mencari sekaligus mempelajari beberapa literatur dan artikel mengenai Kriptografi, Steganografi dan metode-metode yang akan digunakan, dalam makalah ini digunakan algoritma 3DES pada kriptografi dan LSB *insertion* pada Steganografi.
2. Perancangan
   Mengumpulkan data yang diperlukan, melakukan analisa dan perancangan untuk tahap implementasi.
3. Implementasi
   Membuat rancangan model perangkat lunak dengan analisis terstruktur dan mengimplementasikan hasil rancangan tersebut yaitu membuat piranti lunak enkripsi dan dekripsi data teks serta penyisipan dan penguraian data teks pada citra.
4. Pengujian
   Melakukan pengujian dari sistem yang telah dibangun pada tahap implementasi.
5. Pembuatan Laporan
   Membuat laporan tertulis mengenai skripsi ini.

1.7 Sistematika Penulisan Skripsi

Pembuatan skripsi ini dilakukan dengan pembagian bab sebagai berikut:

BAB I : PENDAHULUAN

Menguraikan mengenai latar belakang dari pembuatan perangkat lunak, rumusan masalah, batasan dari masalah yang timbul, tujuan, manfaat, metodologi dan sistematika penulisan yang digunakan.

BAB II : TINJAUAN PUSTAKA

Bab ini berisi ulasan publikasi atau teori yang erat hubungannya dan mendukung pembuatan Tugas akhir

BAB III : METODOLOGI DAN PERANCANGAN

Menguraikan bentuk dan model sistem.

BAB IV : HASIL DAN PEMBAHASAN

Dalam bab ini dijelaskan mengenai implementasi program, uji coba dan analisanya.

BAB V : PENUTUP

Berisi kesimpulan dari pembahasan dan saran yang diharapkan bermanfaat untuk pengembangan skripsi ini selanjutnya.


UNIVERSITAS BRAWIJAYA

BAB II
TINJAUAN PUSTAKA

2.1 Kriptografi

Kriptografi berasal dari bahasa Yunani, yang terdiri dari dua kata, *kripto* dan *graphia*. *Kripto* berarti rahasia (*secret*) sedangkan *graphia* berarti tulisan (*writing*). Sehingga kriptografi bisa diartikan sebagai "tulisan yang dirahasiakan". Secara umum kriptografi adalah ilmu dan seni untuk menjaga kerahasiaan berita (Schneier, 1996). Selain itu kriptografi juga bisa diartikan sebagai ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan data, keabsahan data, integritas data, serta autentifikasi data. Tidak semua aspek keamanan informasi ditangani oleh kriptografi seperti yang disebutkan oleh Menezes, Alfred J., Paul C. Van dan Scott A. Vanstone (1996).

Tujuan utama dari kriptografi tentu saja untuk mengamankan pesan. Pengamanan pesan yang dilakukan mencakup beberapa aspek yang termasuk dalam aspek keamanan informasi, yaitu:

1. Kerahasiaan, adalah layanan yang digunakan untuk menjaga isi dari informasi dari siapapun kecuali yang memiliki otoritas atau kunci rahasia untuk membuka atau mengupas informasi yang telah disandi.
2. Integritas data, adalah berhubungan dengan penjagaan dari perubahan data secara tidak sah. Untuk menjaga integritas data sistem harus memiliki kemampuan untuk mendeteksi manipulasi data oleh pihak-pihak yang tidak berhak antara lain penyisipan, penghapusan dan pensubsitusian data lain kedalam data yang sebenarnya.
3. Autentikasi, adalah berhubungan dengan identifikasi atau pengenalan baik secara kesatuan sistem maupun informasi itu sendiri.
4. Non-repudiasi, adalah usaha untuk mencegah terjadinya penyangkalan terhadap pengiriman atau terciptanya suatu informasi oleh yang mengirimkan atau yang membuat.

Dalam ilmu kriptografi sering juga disebutkan istilah enkripsi dan dekripsi. Enkripsi sendiri adalah proses mengubah

pesan sebenarnya menjadi sandi rahasia. Sedangkan dekripsi merupakan kebalikan dari enkripsi yaitu proses mengembalikan sandi rahasia kembali pesan yang sebenarnya.

Secara umum, dalam sebuah algoritma kriptografi terdapat tiga unsur yaitu:

a. Enkripsi, yaitu proses mengubah *plaintext* menjadi *chipertext.*
b. Dekripsi, yaitu proses mengubah *chipertext* menjadi *plaintext.*
c. Kunci, merupakan kunci yang digunakan untuk proses enkripsi maupun proses dekripsi.

Proses enkripsi dan dekripsi secara umum ditunjukkan oleh gambar 2.1.

Gambar 2.1 Diagram Enkripsi Secara Umum

2.1.1 DES (*Data Encryption Standard*)

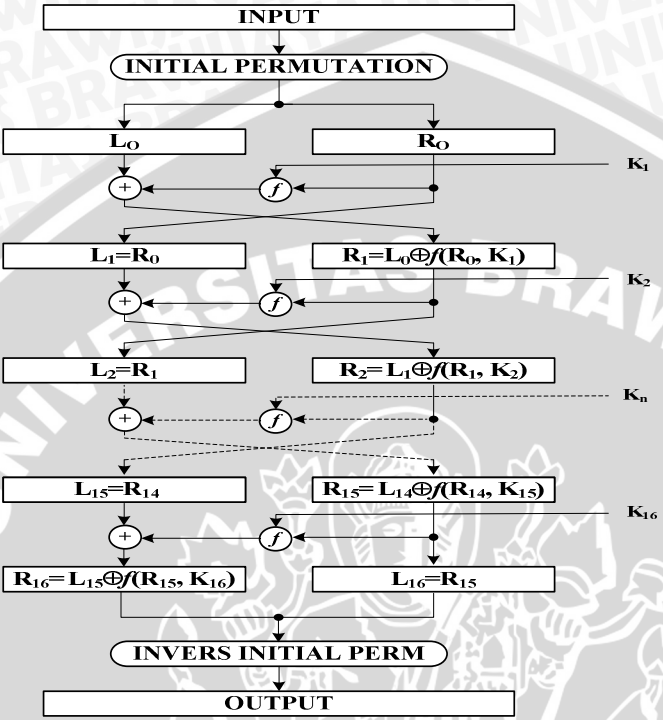
DES merupakan salah satu algoritma kunci simetris yang diadopsi oleh NIST dari algoritma Lucifer yang sudah dimodifikasi. DES beroperasi pada ukuran blok 64-bit, mengenkripsikan 64-bit *plaintext* menjadi 64-bit *chipertext* dengan menggunakan 56-bit kunci internal yang dibangkitkan dari kunci eksternal yang panjangnya 64-bit. Kunci eksternal yang diinputkan akan diproses untuk mendapatkan 16 kunci internal. Pertama, Kunci eksternal yang panjangnya 64-bit disubstitusikan pada matriks permutasi kompresi PC-1. Dalam permutasi ini, setiap bit kedelapan (*parity bit*) dari delapan byte diabaikan. Hasil permutasi panjangnya menjadi 56-bit, yang kemudian dibagi menjadi dua bagian, yaitu kiri (*C0*) dan kanan (*D0*) masing-masing panjangnya 28-bit. Kemudian, bagian kiri dan kanan dilakukan pergeseran bit pada setiap putaran sebanyak satu atau dua bit tergantung pada tiap putaran. Setelah mengalami pegeseran bit, *Ci* dan *Di* digabungkan dan disubstitusikan pada matriks permutasi kompresi dengan menggunakan matriks PC-2,

sehingga panjangnya menjadi 48-bit. Proses tersebut dilakukan sebanyak 16 kali secara berulang-ulang.

Plaintext yang diinputkan pertama akan disubtitusikan pada matriks permutasi awal (*initial permutation*) atau IP panjangnya 64-bit. Kemudian dibagi menjadi dua bagian, yaitu kiri (*L*) dan kanan (*R*) masing-masing panjangnya menjadi 32-bit. Kedua bagian ini masuk ke dalam 16 putaran DES. Satu putaran DES merupakan model jaringan Feistel. Secara matematis jaringan Feistel dinyatakan dalam persamaan 2.1 dan 2.2.

- $L_i = R_{i-1}$; $1 \leq i \leq 16$ (2.1)
- $R_i = L_{i-1} \oplus f(R_{i-1}, k_i)$ (2.2)

Bagian *R* disubtitusikan pada fungsi ekspansi panjangnya menjadi 48-bit kemudian di-XOR-kan dengan kunci internal yang sudah diproses sebelumnya pada proses pembangkitan kunci (pada putaran pertama menggunakan kunci internal pertama, dan seterusnya). Hasil XOR kemudian disubstitusikan pada *S-box* yang dikelompokkan menjadi 8 kelompok, masing-masing 6-bit hasilnya menjadi 4-bit. Kelompok 6-bit pertama menggunakan *S1*(S-BOX 1), kelompok 6-bit kedua menggunakan *S2*, dan seterusnya. Setelah proses *S-box* tersebut panjangnya menjadi 32-bit. Kemudian disubstitusikan lagi pada matriks permutasi *P-box*, kemudian di-XOR-kan dengan bagian *L*. Hasil dari XOR tersebut disimpan untuk bagian *R* selanjutnya. Sedangkan untuk bagian *L* diperoleh dari bagian *R* yang sebelumnya. Proses tersebut dilakukan 16 kali. Setelah 16 putaran selesai, bagian *L* dan *R* digabungkan dan disubstitusikan pada matriks permutasi awal balikan (*invers initial permutation*) atau IP-1, hasilnya merupakan *ciphertext* 64-bit. Proses – proses pada algoritma DES ditunjukkan oleh gambar 2.2.



Gambar 2.2 Proses Enkripsi Algoritma DES

Proses dekripsi pada DES merupakan kebalikan dari proses enkripsi DES. Perbedaannya terletak pada urutan kunci yang digunakan pada 16 putaran jaringan *feistel*. Apabila pada DES kunci internal yang digunakan adalah K= $k_1,k_2...,k_{16}$ maka pada proses dekripsi kunci yang digunakan adalah K= $k_{16},k_{15}...,k_1$. Sehingga model jaringan *feistel untuk* proses dekripsi secara matematis dinyatakan oleh persamaan 3.3 dan 3.4.

$$\mathbf{R_{i-1} = L_i} \quad ;16 \leq n \leq 1 \quad (2.3)$$

$$\mathbf{L_{i-1} = R_i \oplus f(R_{i-1}, K_n)} \quad ; 1 \leq i \leq 16 \quad (2.4)$$

2.1.2 3DES (*Triple Data Encryption Standard*)

3DES (*Triple Data Encryption Standard*) merupakan suatu algoritma pengembangan dari algoritma DES (*Data Encryption*

Standard). Pada dasarnya algoritma yang digunakan sama, hanya pada 3DES dikembangkan dengan melakukan enkripsi dengan implementasi algoritma DES sebanyak tiga kali. 3DES memiliki tiga buah kunci yang berukuran 168-bit (tiga kali kunci 56-bit dari DES). Pada algoritma 3DES dibagi menjadi tiga tahap, setiap tahapnya merupakan implementasi dari algoritma DES. Tahap pertama, *plaintext* yang diinputkan dioperasikan dengan kunci eksternal pertama (K1) dan melakukan proses enkripsi dengan menggunakan algoritma DES. Sehingga menghasilkan pra-*cipherteks* pertama. Tahap kedua, pra-*cipherteks* pertama yang dihasilkan pada tahap pertama, kemudian dioperasikan dengan kunci eksternal kedua (K2) dan melakukan proses enkripsi atau proses dekripsi (tergantung cara pengenkripsian yang digunakan) dengan menggunakan algoritma DES. Sehingga menghasilkan pra-*cipherteks* kedua. Tahap terakhir, pra-*cipherteks* kedua yang dihasilkan pada tahap kedua, dioperasikan dengan kunci eksternal ketiga (K3) dan melakukan proses enkripsi dengan menggunakan algoritma DES, sehingga menghasilkan *ciphertext* (C).

Deskripsi pada 3DES merupakan kebalikan dari enkripsi 3DES. Dekripsi 3DES mempunyai langkah-langkah yang sama seperti proses enkripsi tetapi membalik urutan proses yang dijalankan dan urutan kunci yang digunakan. Apabila pada proses enkripsi dilakukan proses dekripsi maka pada proses dekripsi akan dilakukan proses enkripsi.

2.2 Steganografi

Steganografi (*steganography*) adalah ilmu dan seni menyembunyikan pesan rahasia (*hiding message*) sedemikian sehingga keberadaan pesan tidak terdeteksi oleh indera manusia (Morkel T, 2005).

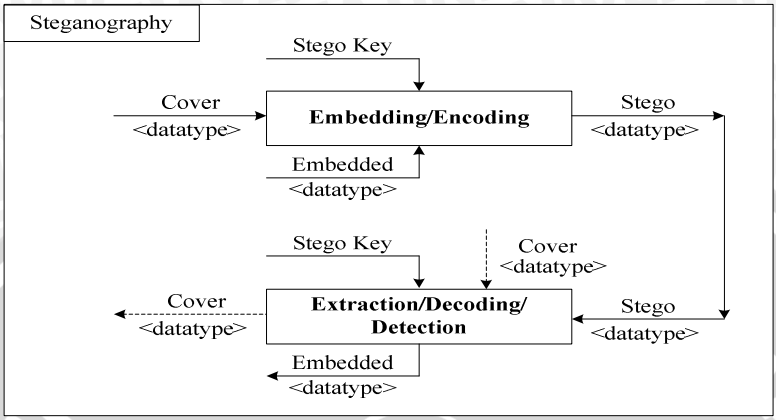
Steganografi sudah dikenal oleh bangsa Yunani. Penguasa Yunani dalam mengirimkan pesan rahasia menggunakan kepala budak atau prajurit sebagai media. Dalam hal ini, rambut budak dibotaki, lalu pesan rahasia ditulis pada kulit kepala budak. Ketika rambut budak tumbuh, budak tersebut diutus untuk membawa pesan rahasia di kepalanya.

Bangsa Romawi mengenal steganografi dengan menggunakan tinta tak-tampak (*invisible ink*) untuk menuliskan

pesan. Tinta tersebut dibuat dari campuran sari buah, susu, dan cuka. Jika tinta digunakan untuk menulis maka tulisannya tidak tampak. Tulisan di atas kertas dapat dibaca dengan cara memanaskan kertas tersebut.

Sesungguhnya steganografi merupakan metode lama yang sudah dipakai oleh orang-orang Yunani kuno. Kata Steganografi berasal dari bahasa Yunani yang artinya tulisan tertutup atau tersembunyi (*covered letter*), yang meliputi berbagai cara komunikasi yang menyembunyikan pesan dengan sangat efisien. Pada awalnya metode ini berupa penggunaan tinta yang tidak nampak, pengaturan karakter, tanda tangan digital, saluran yang dikacaukan dan spektrum komunikasi yang disebar. Dengan adanya teknologi digital maka muncul cara baru untuk sistem steganografi ini, yaitu dengan penyembunyian pesan dalam gambar digital.

Ada dua proses utama dalam steganografi yaitu penyisipan (*embedding*) dan penguraian (*extraction*) pesan atau informasi dalam media cover. *Embedding* merupakan proses menyisipkan pesan atau informasi ke dalam media cover, sedangkan *extraction* adalah proses menguraikan pesan yang tersembunyi dalam gambar stego. Pesan yang akan disembunyikan dalam sebuah gambar membutuhkan dua file. Pertama adalah gambar asli yang belum dimodifikasi yang akan menangani pesan tersembunyi, yang disebut gambar cover (*cover image*). File kedua adalah informasi pesan yang disembunyikan. Suatu pesan dapat berupa *plaintext*, *chipertext*, gambar lain, atau apapun yang dapat ditempelkan ke dalam bit *stream*. Ketika dikombinasikan, *cover image* dan pesan yang ditempelkan membuat gambar stego (*stego image*). Proses steganografi secara umum ditunjukkan oleh gambar 2.3.



Gambar 2.3 Steganografi (Mohanty, 1999)

Pihak yang terkait dengan steganografi antara lain *embeddor*, *extractor*, dan *stegoanalyst* (Mohanty, 1999). *Embeddor* adalah orang yang melakukan *embedding* dengan menggunakan aplikasi steganografi, *extractor* adalah orang yang melakukan *extract stego image* dengan menggunakan aplikasi steganografi. Sedangkan stegoanalyst adalah orang yang melakukan steganalisis. Steganalisis merupakan ilmu dan seni untuk mendeteksi pesan yang tersembunyi dalam steganografi.

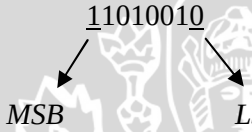
Steganografi dapat dikatakan baik jika memenuhi empat kriteria sebagai berikut (Rinaldi, 2004) :

1. *Fidelity.*
   Mutu citra penampung tidak jauh berubah. Setelah penambahan data rahasia, citra hasil steganografi masih terlihat dengan baik. Pengamat tidak mengetahui kali di dalam citra tersebut terdapat data rahasia.
2. *Robustness.*
   Data yang disembunyikan harus tahan terhadap manipulasi yang dilakukan pada citra penampung (seperti pengubahan kontras, penajaman, pemampatan, rotasi, perbesaran gambar, pemotongan (*cropping*), enkripsi dan sebagainya). Bila pada citra dilakukan operasi pengolahan citra, maka data yang disembunyikan tidak rusak.

3. *Recovery*.
   Data yang disembunyikan harus dapat diunkapkan kembali (*recovery*). Karena tujuan steganografi adalah *data hiding*, maka sewaktu – waktu data rahasia di dalam citra penampung harus dapat diambil kembali untuk digunakan lebih lanjut.

2.2.1 *Least Significant* Bit (LSB)

Least Significant Bit adalah bagian dari barisan data biner (basis dua) yang mempunyai nilai paling tidak berarti atau paling kecil. Letaknya paling kanan dari barisan bit



LSB = Least Significant Bit
MSB = Most Siginificant Bit

2.2.2 Teknik *Least Significant* Bit (LSB) *Insertion*

Penyisipan LSB dilakukan dengan memodifikasi bit terakhir dalam satu *byte* data. Bit yang diganti adalah LSB karena perubahan pada LSB hanya menyebabkan perubahan nilai *byte* satu lebih tinggi atau satu lebih rendah. Misalkan data yang diubah adalah warna hijau, maka perubahan pada LSB hanya menyebabkan sedikit perubahan yang tidak dapat dideteksi oleh mata manusia.

Seperti yang sudah diketahui, untuk file *bitmap* 24 bit maka setiap *pixel* (titik) pada gambar tersebut terdiri dari susunan tiga warna merah, hijau dan biru (RGB) yang masing-masing disusun oleh bilangan 8 bit (byte) dari 0 sampai 255 atau dengan *format* biner 00000000 sampai 11111111. Dengan demikian pada setiap *pixel* file *bitmap* 24 bit dapat disisipkan 3 bit data. Contohnya huruf A dapat disisipkan ke dalam 3 *pixel*, misalnya terdapat susunan 3 *pixel*:

(00100111 11101001 11001000)
(00100111 11001000 11101001)
(11001000 00100111 11101001)

Sedangkan representasi biner huruf A adalah 10000011. Dengan menyisipkan-nya pada data *pixel* diatas maka akan dihasilkan:

(00100111 11101000 11001000)
(00100110 11001000 11101000)
(11001001 00100111 11101001)

Terlihat hanya empat bit rendah yang berubah, untuk mata manusia maka tidak akan tampak perubahannya. Secara rata-rata dengan metode ini hanya setengah dari data bit rendah yang berubah, sehingga bila dibutuhkan dapat digunakan bit rendah kedua bahkan ketiga.

2.3 Citra Digital

Citra digital memiliki informasi berupa gambar dan terdiri dari elemen terkecil yaitu piksel. Citra digital direpresentasikan dalam bentuk matriks dua dimensi yang setiap elemen merepresentasikan piksel pada gambar.

Terdapat beberapa jenis pewarnaan pada citra digital yaitu *duotone* (dua warna), *grayscale* dan citra berwarna. Citra berwarna dapat memiliki sistem pewarnaan *RGB*, *indexed color* atau *256 color*. Pada citra digital dengan pewarnaan *duotone*, warna pada piksel hanya memiliki dua kemungkinan warna, pada umumnya hitam-putih. Dengan penggunaan warna 1-bit, maka kualitas gambar pada citra digital tidak begitu bagus. Pewarnaan *grayscale* memiliki kualitas lebih baik. Pada *grayscale*, warna yang tersedia hanyalah warna-warna yang ada diantara hitam dan putih, meliputi warna abu-abu yang beragam.

Citra *RGB* adalah yang paling populer saat ini, dimana setiap piksel direpresentasikan dengan intensitas warna merah, hijau dan biru. Citra *indexed color* hanya memiliki 256 warna yang telah didefinisikan pada tabel warna, namun memiliki ukuran *file* yang lebih kecil.

2.3.1 *Bitmap*

Matriks adalah struktur data yang tepat untuk merepresentasikan citra digital (Rinaldi, 2004). Elemen-elemen matriks dapat diakses secara langsung melalui indeksnya (baris dan kolom).

Citra digital yang berukuran N x M (tinggi = N, lebar = M) lazim dinyatakan dengan matriks N baris dan M kolom ditunjukkan pada gambar 2.4.

$$f(x,y) \approx \begin{bmatrix} f(0,0) & f(0,1) & \ldots & f(0,M) \\ f(1,0) & f(1,1) & \ldots & f(1,M) \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ f(N-1,0) & f(N-1,1) & \ldots & f(N-1,M-1) \end{bmatrix}$$

Gambar 2.4 Representasi *pixel* pada citra digital

Untuk citra dengan 256 derajat keabuan, harga setiap elemen matriks adalah bilangan bulat di dalam selang [0,255].

Citra dalam *format* BMP lebih bagus daripada citra dalam *format* yang lainnya, karena citra dalam *format* BMP umumnya tidak dimampatkan sehingga tidak ada informasi yang hilang. Nilai intensitas *pixel* di dalam citra dipetakan ke sejumlah bit tertentu. Peta bit yang umum adalah 8, artinya setiap *pixel* panjangnya 8 bit. Delapan bit ini merepresentasikan nilai intensitas *pixel*. Dengan demikian ada sebanyak 2^8 = 256 derajat keabuan, mulai 0 sampai 255.

Citra dalam *format* BMP ada tiga macam : citra biner, citra berwarna, dan citra hitam-putih (*grayscale*). Citra biner hanya

mempunyai dua nilai keabuan, 0 dan 1. Oleh karena itu, 1 bit sudah cukup untuk merepresentasikan nilai *pixel*. Citra berwarna adalah citra yang lebih umum. Warna yang terlihat pada citra *bitmap* merupakan kombinasi dari tiga warna dasar, yaitu merah, hijau, dan biru. Setiap *pixel* disusun oleh tiga komponen warna : R (*red*), G (*green*), dan B (*blue*). Kombinasi dari ketiga warna RGB tersebut menghasilkan warna yang khas untuk *pixel* yang bersangkutan. Pada citra 256 warna, setiap *pixel* panjangnya 8 bit, tetapi komponen warna RGB-nya disimpan di dalam tabel RGB yang disebut palet. Setiap komponen panjangnya 8 bit, jadi ada 256 nilai keabuan untuk warna merah, 256 nilai keabuan untuk warna hijau, dan 256 warna keabuan untuk warna biru. Nilai setiap *pixel* tidak menyatakan derajat keabuannya secara langsung, tetapi nilai *pixel* menyatakan indeks tabel RGB yang memuat nilai keabuan merah (R), nilai keabuan hijau (G), dan nilai keabuan biru (B) untuk *pixel* yang bersangkutan. Pada citra hitam-putih, nilai R = G = B untuk menyatakan bahwa citra hitam-putih hanya mempunyai satu kanal warna.

Citra yang lebih kaya warna adalah citra 24-bit. Setiap *pixel* panjangnya 24 bit, karena setiap *pixel* langsung menyatakan komponen warna merah, komponen warna hijau, dan komponen warna biru. Masing-masing komponen panjangnya 8 bit. Citra 24-bit disebut juga citra 16 juta warna, karena mampu menghasilkan 2^{24}= 16.772.216 kombinasi warna.

Saat ini beredar tiga versi berkas *bitmap*:

1. berkas *bitmap* versi lama dari Microsoft Windows atau IBM OS/2.
2. berkas *bitmap* versi baru dari Microsoft Windows.
3. berkas *bitmap* versi baru dari IBM OS/2.

Yang membedakan ketiga versi berkas tersebut adalah panjang header-nya. Header adalah data yang terdapat pada bagian awal berkas citra. Data di dalam header berguna untuk mengetahui bagaimana citra dalam *format bitmap* dikodekan dan disimpan. Data di dalam header misalnya ukuran citra, kedalaman *pixel*, ofset ke data *bitmap*, dan sebagainya.

Setiap berkas *bitmap* terdiri atas header berkas, header *bitmap*, informasi palet dan data *bitmap* (gambar 2.5).

Header berkas	Header *bitmap*	Informasi palet	Data *bitmap*

Gambar 2.5 *Format* berkas *bitmap* 8 bit

Informasi palet warna terletak sesudah header *bitmap*. Informasi palet warna dinyatakan dalam suatu tabel RGB. Setiap entry pada tabel terdiri atas tiga buat field, yaitu R(*red*), G(*green*), dan B(*blue*).

Data *bitmap* diletakkan sesudah informasi palet. Penyimpanan data *bitmap* di dalam berkas disusun terbalik dari bawah ke atas dalam bentuk matriks yang berukuran Height x Width. Baris ke-0 pada matriks data *bitmap* menyatakan data *pixel* di citra baris terbawah, sedangkan baris terakhir pada matriks menyatakan data *pixel* di citra baris teratas.

Format citra 4-bit (16 warna) serupa dengan *format* citra 8-bit. Pada citra 4-bit dan citra 8-bit, warna suatu *pixel* diacu dari tabel informasi palet pada entry ke-k (k merupakan nilai dengan rentang 0 – 15 untuk citra 16 warna dan 0-255 untuk citra 256 warna). Sebagai contoh pada gambar 2.4, *pixel* pertama bernilai 2; warna *pixel* pertama ini ditentukan oleh komponen RGB pada tabel palet warna entry ke-2, yaitu R = 14, G = 13, dan B = 16. *Pixel* kedua serupa dengan *pixel* pertama. *Pixel* ketiga bernilai 1, warnanya ditentukan oleh komponen RGB pada tabel warna entry ke-1, yaitu R = 20, G = 45, dan B = 24. Demikian seterusnya untuk *pixel – pixel* lainnya. Khusus untuk citra hitam-putih, komponen R, G, dan B suatu *pixel* bernilai sama dengan data *bitmap pixel* tersebut. Jadi, *pixel* dengan nilai data *bitmap* 129, memiliki nilai R =129, G = 129, dan B = 129.

<header berkas>
<header *bitmap*>
<palet warna RGB>

	R	G	B
1	20	45	24
2	14	13	16
3	12	17	15
…			
256	46	78	25

<data *bitmap*>
2 2 1 1 1 3 5 . . .

Gambar 2.6 *Format* citra 8-bit

Berkas citra 24-bit (16,7 juta warna) tidak mempunyai palet RGB, karena nilai RGB langsung diuraikan dalam data *bitmap*. Setiap elemen data *bitmap* panjangnya 3 byte, masing-masing byte menyatakan komponen R, G, dan B. Contoh *format* citra 24-bit (16,7 juta warna) kira-kira seperti pada gambar 2.7.

<header berkas>
<header *bitmap*>
<data *bitmap*>

20 19 21 24 24 23 24 . . .

Gambar 2.7 *Format* citra 24-bit (16,7 juta warna)

Pada contoh *format* citra 24-bit tersebut *pixel* pertama mempunyai R = 20, G = 19, B = 21, *pixel* kedua mempunyai R = 24, G = 24, B = 23, demikian seterusnya.

2.3.2 *Peak Signal to Noise Ratio* (PNSR)

Peak Signal to Noise Ratio (PSNR) adalah perbandingan antara nilai maksimum dari sinyal yang diukur dengan besarnya derau yang berpengaruh pada sinyal tersebut. PSNR biasanya diukur

dalam satuan desibel. PSNR digunakan untuk mengetahui perbandingan kualitas citra sebelum dan sesudah disisipkan pesan. Untuk menentukan PSNR. Perhitungan PSNR dinyatakan pada persamaan 2.5.

$$\text{PSNR} = 20xLog10\left(255 \Big/ \sqrt{\frac{1}{MN}\sum_{Y=1}^{M}\sum_{X=1}^{N}[I(x,y)-I'(x,y)]^2}\right) \quad (2.5)$$

Dimana :
m = panjang citra tersebut (dalam *pixel*)
n = lebar citra tersebut (dalam *pixel*)
(x,y) = koordinat masing-masing *pixel*
I = nilai bit pada citra asli
I’ = nilai bit pada citra steganografi
Nilai PSNR yang besar akan lebih baik. (Sigit L, 2009).

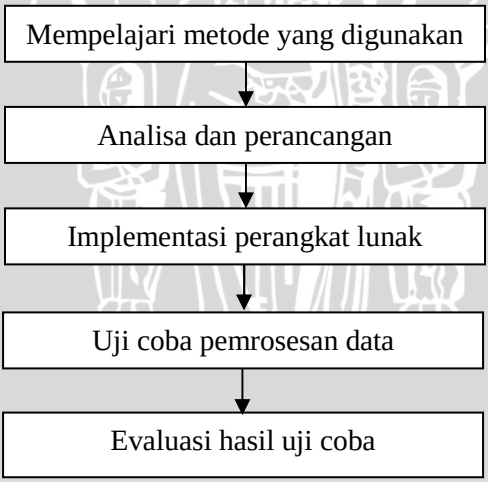
BAB III
METODOLOGI DAN PERANCANGAN

Pada bab metodologi dan perancangan ini akan dibahas metode, rancangan yang digunakan dan langkah – langkah yang dilakukan dalam pembuatan aplikasi kriptografi dan steganografi pada citral digital menggunakan algoritma 3DES.

Penelitian dilakukan dengan tahapan-tahapan sebagai berikut:

1. Mempelajari metode yang digunakan dari berbagai sumber seperti yang telah dijelaskan pada bab 2.
2. Menganalisa dan merancang perangkat lunak dengan metode yang akan digunakan.
3. Membuat perangkat lunak berdasarkan analisis dan perancangan yang dilakukan.
4. Uji coba pengamanan data dan penguraian data menggunakan perangkat lunak yang telah dibuat.
5. Evaluasi hasil uji coba.

Langkah – langkah penelitian ini ditunjukkan oleh Gambar 3.1.



Gambar 3.1 Langkah – langkah penelitian

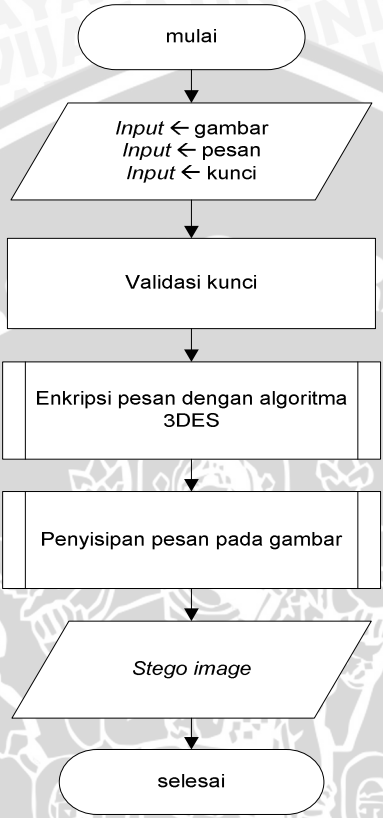
3.1 Gambaran Umum Sistem

Aplikasi perangkat lunak pengamanan data yang akan dibuat merupakan implementasi dari dua teknik pengamanan data yaitu kriptografi dengan algoritma 3DES dan steganografi dengan LSB *insertion*. Pada aplikasi ini terdapat dua proses utama yaitu proses pengamanan data dan proses penguraian data. Proses pengamanan data menjalankan prosedur enkripsi dengan algoritma 3DES dan proses *embedding* (penyisipan pesan) dengan metode LSB *insertion.* Pada proses penguraian data dijalankan prosedur *extraction* (penguraian pesan) dan dekripsi pesan dengan algoritma 3DES. Algoritma 3DES menerima masukan berupa data pesan dalam bentuk *plaintext* dan menghasilkan data pesan dalam bentuk *ciphertext* (pesan yang sudah teracak). Metode steganografi LSB *insertion* akan menterjemahkan *ciphertext* ke bentuk biner untuk kemudian disisipkan pada bit terakhir dari *image* (gambar). Hal ini akan menghasilkan gambar yang merupakan *stego – image* yaitu gambar yang mengandung pesan rahasia.

Proses – proses untuk pengamanan data adalah :

1. Memasukkan data pesan dalam bentuk *file* teks, gambar *bitmap* 24 bit dan kunci.
2. Validasi kunci.
3. Pengenkripsian pesan menggunakan algoritma 3DES.
4. Penyisipan pesan pada *image*.

Diagram langkah – langkah pengamanan data ditunjukkan pada gambar 3.2.

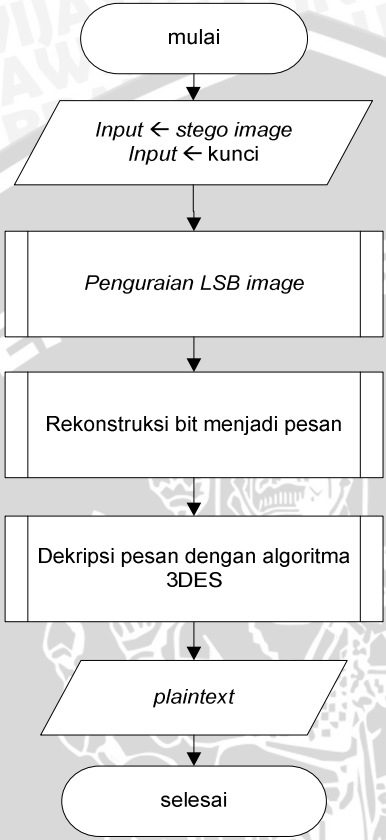


Gambar 3.2 *Flowchart* Pengamanan Data

Selain proses pengamanan data, aplikasi perangkat lunak yang akan dibangun juga dapat menguraikan dan mendekripsi pesan yang terdapat pada *stego image*. Tahapan yang dilakukan untuk penguraian data adalah :

1. Memasukkan *file* citra *stego image* dan kunci.
2. Autentifikasi kunci.
3. Menguraikan LSB dari *stego image*.
4. Menyusun bit-bit LSB menjadi *ciphertext.*
5. Mendekripsi *ciphertext*.

Diagram langkah – langkah penguraian pesan ditunjukkan pada gambar 3.3.



Gambar 3.3 *Flowchart* Penguraian Data

3.2 Perancangan Perangkat Lunak

3.2.1 Perancangan Proses Pengamanan Data

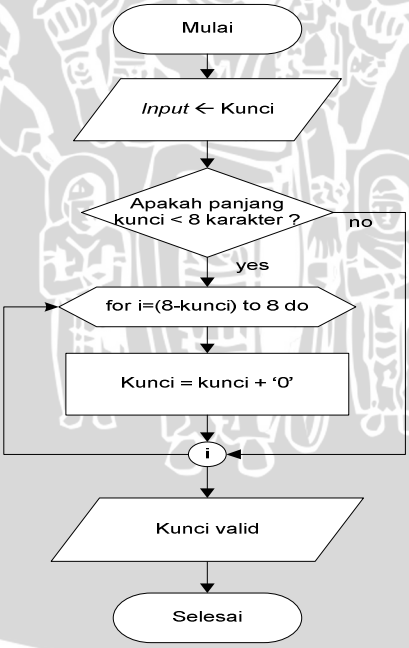
Pada subbab ini akan diuraikan proses pengamanan data yang mengimplementasikan teknik penyandian 3DES dan teknik steganografi. Karena algoritma 3DES merupakan algoritma kunci simetris yang memerlukan sebuah kunci untuk proses enkripsi dan dekripsi maka akan dilakukan proses validasi kunci yang diinputkan oleh *user*.

Sebelum prosedur pengamanan data dijalankan, akan dilakukan proses pemeriksaan terhadap kapasitas penampung (gambar). Apabila pesan yang akan disisipkan melebihi kapasitas penampung akan diberikan peringatan.

3.2.1.1 Proses Validasi Kunci

Agar kunci yang diinputkan sesuai dengan kriteria kunci yang diperlukan untuk proses pengenkripsian pesan maka perlu dilakukan proses validasi kunci. Langkah – langkah dalam proses validasi kunci adalah sebagai berikut:

1. Inputan berupa kunci.
2. Cek panjang kunci, apabila kurang dari 8 karakter maka dilakukan penambahan karakter '0'.
3. Setelah dilakukan proses penambahan karakter, panjang kunci di cek lagi. Proses penambahan selesai apabila kunci mempunyai panjang 8 karakter.



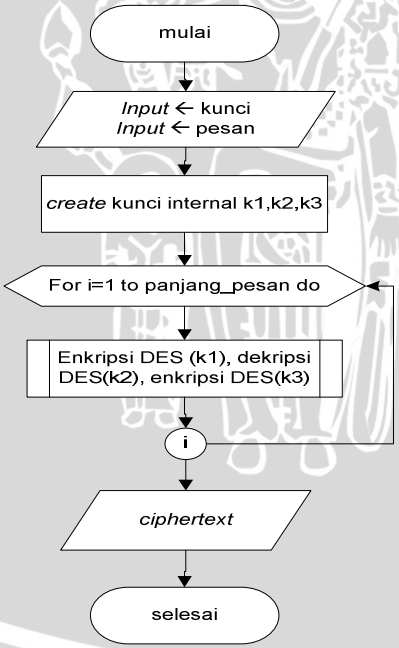
Gambar 3.4 *Flowchart* Validasi Kunci

3.2.1.2 Proses Enkripsi

Proses enkripsi menggunakan algoritma 3DES yang akan dijalankan bertujuan untuk menyandikan pesan yang berupa data teks sebelum disisipkan ke dalam *image* (gambar). Pada proses ini diperlukan sebuah kunci dengan panjang 8 karakter atau 64 bit. Karena 3DES melakukan tiga kali proses enkripsi-dekripsi DES maka diperlukan tiga kunci dalam pelaksanaannya. Tiga kunci ini dibangkitkan dari kunci eksternal yang diinputkan secara manual. Berikut adalah langkah – langkah yang dilakukan dalam proses enkripsi:

1. Inputan berupa kunci dan pesan.
2. Bangkitkan tiga kunci internal dari kunci yang telah diinputkan.
3. Lakukan tiga kali proses enkripsi-dekripsi DES pada tiap karakter pesan dengan masing – masing kunci internal.

Proses enkripsi 3DES ditunjukkan oleh gambar 3.5.



Gambar 3.5 Proses Enkripsi dengan 3DES

3.2.1.2.1 Enkripsi DES

Data Encryption Standard (DES) merupakan *block cipher* (penyandian blok) yang berarti DES menyandikan tiap – tiap blok pesan. Ukuran pesan awal (*plaintext*) dan pesan yang sudah disandikan (*ciphertext*) dengan algoritma DES tidak berbeda. DES bekerja pada bits (bilangan biner) dengan panjang 64 bits atau setara dengan 16 bilangan hexadecimal. Pesan yang akan di enkripsi harus mempunyai panjang 8 karakter (64 bits) atau kelipatannya. Apabila pesan yang akan di enkripsi tidak sama dengan 8 karakter atau kelipatannya maka akan dilakukan penambahan karakter, pada skripsi ini akan ditambahkan karakter " " (spasi) sesuai dengan kekurangannya. Algoritma DES memerlukan sebuah kunci yang memiliki panjang 64 bits. Akan tetapi, setiap bit ke-8 pada kunci akan diabaikan oleh algoritma DES sehingga panjang kunci yang efektif digunakan adalah 56 bits.

Terdapat beberapa tahapan pada algoritma DES, berikut adalah tahapan-tahapan pada algoritma DES:

1. Melewatkan 64 bits kunci melalui tabel permutasi 1 (**PC-1)** sehingga didapatkan kunci 56 bits. Tabel PC-1 ditunjukkan oleh gambar 3.6.

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	4	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Gambar 3.6 *Permutation Choice 1* (**PC-1**)

2. Memecah kunci 56 bit menjadi dua bagian kiri (C_0) dan kanan (D_0), masing-masing bagian mempunyai panjang 28 bits.

3. Melakukan *shift left* (penggeseran bit ke kiri) sebanyak satu atau dua bit dari subkunci 28 bit. Jumlah bit yang akan digeser sesuai dengan tabel *shift left* yang ditunjukkan oleh tabel 3.1.

Tabel 3.1 *Shift Left*

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Banyak bit	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

4. Membuat 48 bit subkey K_n dimana $1 \leq n \leq 16$. K_n didapatkan dengan cara menggabungkan C_n dan D_n untuk kemudian dilewatkan melalui tabel permutation Choice 2 (**PC-2**).

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Gambar 3.7 *Permutation Choice 2* (**PC-2**)

5. Melewatkan tiap 64 bit blok pesan ke tabel *initial permutation* (**IP**) kemudian dipecah menjadi 2 (dua) bagian kiri (L_0) dan (R_0) dimana tiap bagian mempunyai panjang 32 bit.

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Gambar 3.8 *Initial Permutation* (**IP**)

6. Kedua bagian pesan $\mathbf{L_0}$ dan $\mathbf{R_0}$ dimasukkan ke dalam 16 putaran DES dimana satu putaran DES merupakan model jaringan *Feistel* (*f*) dan ⊕ merupakan operator **XOR**. Proses ini akan menghasilkan final blok untuk n = 16 yaitu $L_{16}R_{16}$. Jaringan *feistel* terdiri dari 3 proses dan setiap proses akan dilewatkan melalui tabel-tabel yang bersesuaian. Proses-proses pada jaringan *feistel* sebagai berikut:

- Untuk $f(R_{n-1}, K_n)$, R_{n-1} yang mempunyai panjang 32 bit akan dilewatkan melalui tabel yg disebut ***E-bit selection table*** sehingga panjangnya menjadi 48 bit dan kemudian di-**XOR**-kan dengan subkunci $\mathbf{K_n}$. Tabel *E-bit selection* ditunjukkan oleh gambar 3.9.

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Gambar 3.9 E-bit *selection*

- Melewatkan $K_1 \oplus E(R_0)$ melalui tabel **S-BOX**. $K_1 \oplus E(R_0)$ terlebih dulu dibagi menjadi 8 blok yaitu $B_1B_2B_3B_4B_5B_6B_7B_8$ dimana masing – masing blok mempunyai panjang 6 bit. Pada tiap blok (B_1-B_8) akan diambil bit terdepan dan bit terakhir (diurutkan dari kiri ke kanan), dua bit ini digabung dan dicari nilai desimalnya untuk kemudian menjadi **indeks baris** pada **S-BOX**. Selanjutnya 4 bit yang tersisa dari blok tersebut dicari nilai desimalnya untuk kemudian menjadi **indeks kolom** pada **S-BOX**. Nilai yang didapatkan kemudian diubah ke bentuk biner. Tabel 3.2 berikut merupakan tabel **S-BOX**.

Tabel 3.2 S-BOX

S1

Kolom / Baris	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S2

Kolom / Baris	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S3

Kolom / Baris	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S4

Kolom / Baris	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S5

Kolom / Baris	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	2	12	4	1	7	10	11	6	7	5	3	15	13	0	14	9
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S6

Kolom / Baris	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S7

Kolom / Baris	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S8

Kolom / Baris	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	1	15	13	8	10	3	7	4	12	4	5	11	0	14	9	2
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

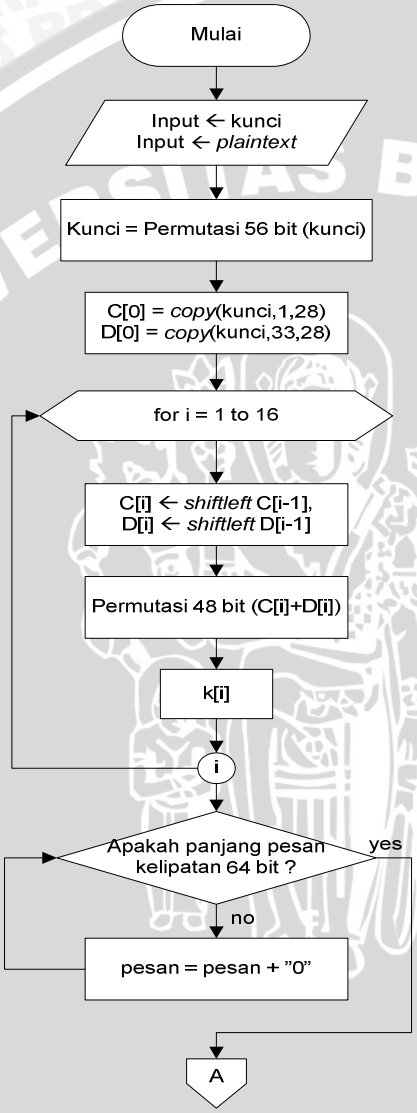
- Melewatkan 32 bit nilai akhir dari S-BOX $S1(B_1)S2(B_2)S3(B_3)S4(B_4)S5(B_5)S6(B_6)S7(B_7)S8(B_8)$ melalui tabel *Permutation* P. **P**$(S1(B_1)S2(B_2)S3(B_3)S4(B_4)S5(B_5)S6(B_6)S7(B_7)S8(B_8))$ merupakan nilai akhir dari satu putaran jaringan *feistel*. Proses dilanjutkan sampai putaran ke 16.

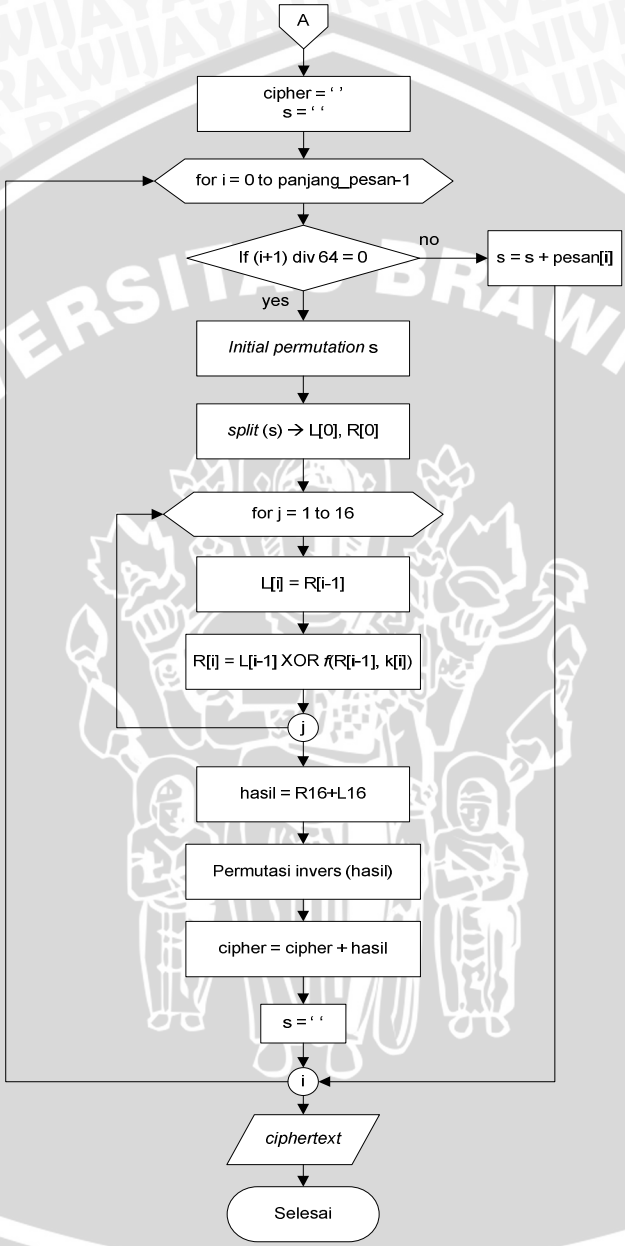
16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Gambar 3.10 *Permutation* P

Setelah 16 putaran selesai diproses akan didapatkan final blok L_{16} dan R_{16}.

7. Melewatkan 64 bit final blok $R_{16}L_{16}$ melalui tabel *invers permutation* (**IP**$^{-1}$).

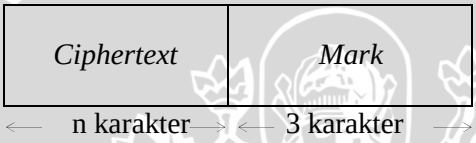




Gambar 3.11 Proses Enkripsi DES

3.2.1.3 Proses Penyisipan Pesan (*Embedding*)

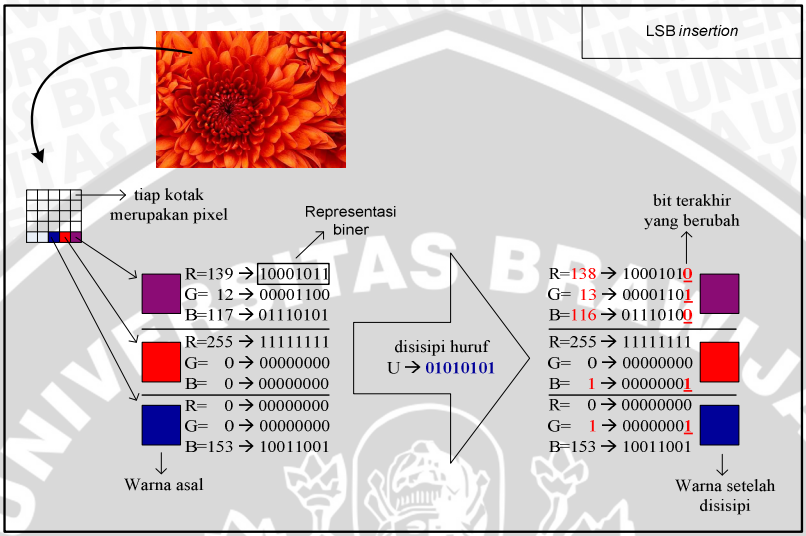
Metode yang dipakai dalam proses penyisipan pesan ke dalam bit – bit gambar yaitu teknik LSB *insertion* atau penyisipan LSB. Sebelum disisipkan ke dalam *image* (gambar) pesan yang berupa *ciphertext* akan ditambahkan *mark* (tanda) sebagai penanda akhir pesan. Penanda (*mark)* yang digunakan yaitu 3 karakter '*#$'. Penggunaan 3 karakter sebagai penanda dilakukan agar tidak terjadi kesamaan penanda dengan isi pesan. Struktur pesan pesan yang akan disisipkan terbagi ke dalam beberapa bagian seperti pada gambar 3.12.



Gambar 3.12 Struktur Pesan

Karena proses penyisipan pesan menggunakan LSB yaitu melakukan modifikasi bit terakhir dari tiap-tiap nilai RGB pada gambar maka pesan akan dirubah ke dalam bentuk biner terlebih dahulu.

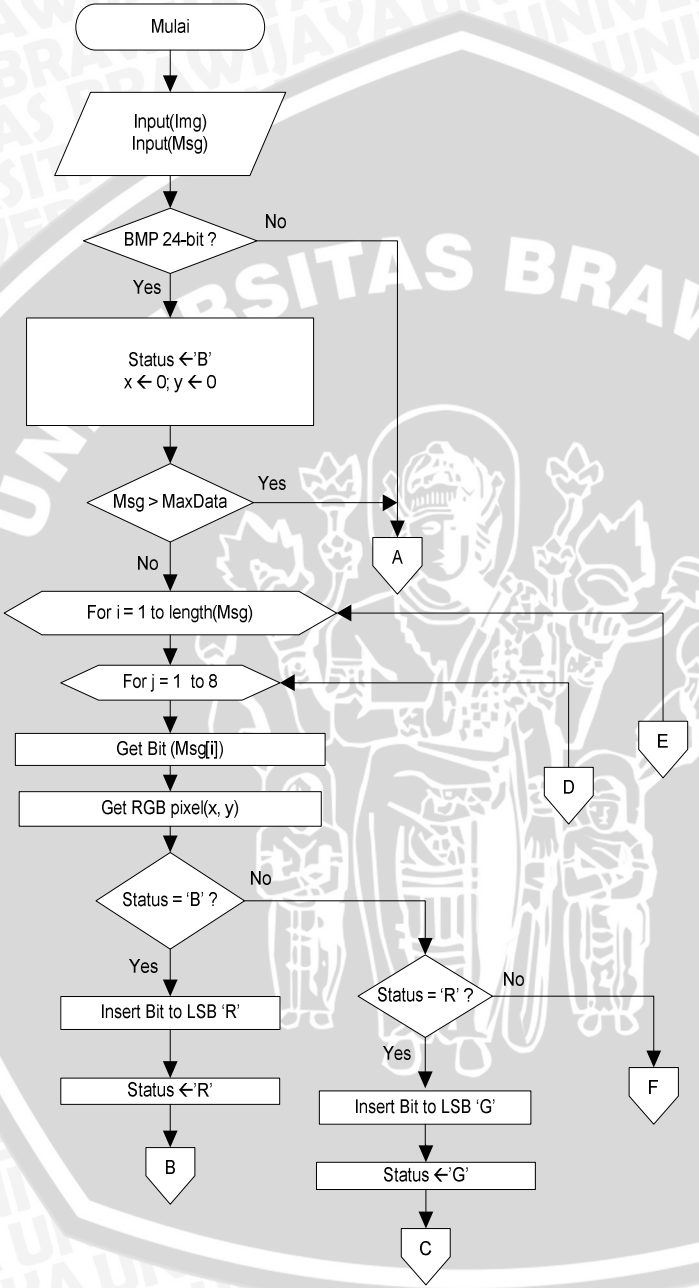
Misalkan akan disisipkan huruf U dengan representasi biner 01010101 ke dalam gambar. Maka dapat disisipkan pada data *pixel* seperti pada gambar 3.13.

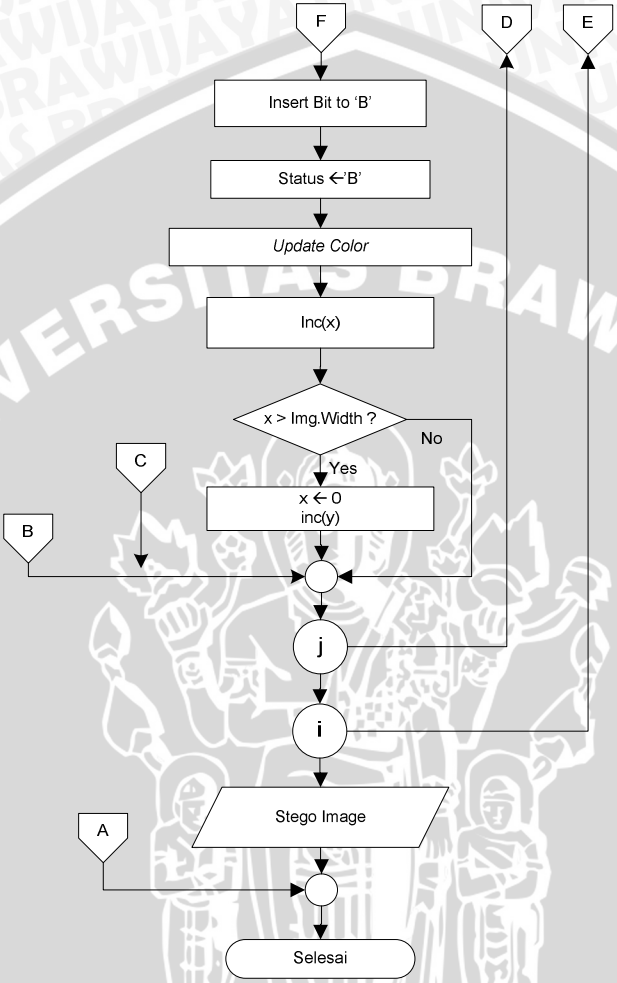


Gambar 3.13 Proses Penyisipan Pesan

Dapat dilihat penyisipan pesan hanya pada bit terakhir dan hanya bit yang tidak bersesuaian yang dirubah. Oleh karena itu perubahan warna yang terjadi tidak tampak secara kasat mata.

Proses penyisipan pesan dilakukan secara *selected pixel* artinya bit – bit data rahasia mengganti *pixel* gambar secara berurutan dimulai dari *pixel* pertama pada *file* citra. Proses penyisipan pesan dalam gambar ditunjukkan oleh gambar 3.14.


Mulai
Input(Img)
Input(Msg)
BMP 24-bit ?
No
Yes
Status ←'B'
x ← 0; y ← 0
Msg > MaxData
Yes
No
A
For i = 1 to length(Msg)
For j = 1 to 8
Get Bit (Msg[i])
Get RGB pixel(x, y)
Status = 'B' ?
No
Yes
Insert Bit to LSB 'R'
Status ←'R'
B
Status = 'R' ?
No
Yes
Insert Bit to LSB 'G'
Status ←'G'
C
D
E
F



Gambar 3.14 *Flowchart* Penyisipan Pesan

Berikut ini adalah penjelasan *flowchart* penyisipan pesan dalam gambar :

1. Data yang dimasukkan secara manual, yaitu gambar (media gambar yang akan digunakan untuk menyembunyikan pesan), pesan teks yang akan disisipkan dalam gambar (dalam bentuk *ciphertext*), dan *password* yang berlaku sebagai kunci enkripsi dalam aplikasi ini.

2. Gambar yang dipilih harus BMP 24 bit.
3. Jika seleksi gambar terpenuhi, kemudian dilakukan inisialisasi x dan y yang merepresentasikan koordinat dalam gambar, dan status warna awal.
4. *Mark* atau penanda akhir yaitu karakter '*#$' sebagai penanda akhir pesan ditambahkan ke dalam pesan (Msg) yang telah dienkripsi.
5. Banyaknya pesan maksimum yang dapat disisipkan ke dalam gambar adalah ((*bitmap*.width * *bitmap*.height) * 3 : 8).
6. Kemudian dilakukan penelusuran atau pembacaan karakter dimulai dari karakter pertama sampai length(teks).
7. Karakter pertama adalah pesan dengan indeks ke 1 atau Msg[1]. Untuk mendapatkan bit-bit pesan, maka untuk setiap karakter dilakukan perulangan sebanyak 8 kali.
8. Dilanjutkan dengan membaca *pixel* gambar f(x,y), dimulai dari koordinat awal f(0,0). Periksa status byte *pixel* yang baru ditempati, apakah R atau G atau B? Ini menunjukkan posisi terakhir penempatan bit karakter. Nilai awal status adalah B, dan selanjutnya berubah sesuai kondisi.
9. Urutan penyisipan pada tiap *pixel* adalah R – G – B, maka jika status = 'B' berarti penyisipan untuk satu *pixel* selesai, sehingga dilakukan update warna.
10. Jika indeks bit lebih besar dari 8 (j > 8), berarti satu karakter telah tersisipkan. Setelah itu dilanjutkan membaca karakter selanjutnya. Proses ini berjalan terus hingga karakter terakhir.

Setelah proses pesan selesai, maka akan dihasilkan sebuah gambar stego (gambar yang mengandung pesan tersembunyi).

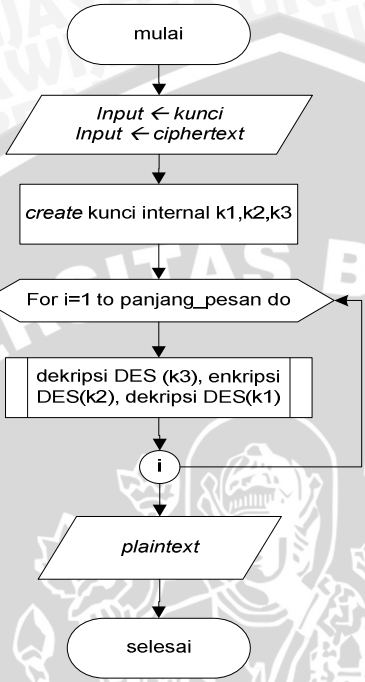
3.2.2 Perancangan Proses Penguraian Data

Pada subbab ini akan diuraikan proses penguraian data yang terdiri dari proses penguraian pesan dari gambar dan proses pendekripsian pesan. Sebelum proses penguraian dijalankan, terlebih dulu dilakukan proses pencocokan kunci. Apabila kunci yang di-input-kan oleh *user* sesuai dengan kunci pada proses enkripsi maka hasil proses penguraian data berupa *plaintext*. Apabila kunci tidak sesuai hasil penguraian akan berupa *ciphertext*.

3.2.2.1 Proses Dekripsi

Proses dekripsi pada algoritma DES mempunyai tahapan – tahapan yang sama dengan proses enkripsi pada algoritma DES. Perbedaan proses enkripsi dan proses dekripsi DES terletak pada urutan subkunci yang digunakan. Pada proses enkripsi subkunci digunakan yaitu K_1, K_2, K_3 ,..., K_{16} sedangkan pada proses dekripsi urutan subkunci yang digunakan yaitu K_{16}, K_{15}, K_{14} ,..., K_1. Hal ini disebabkan karena pengimplementasian jaringan *feistel*. Dengan membalikkan urutan kunci yang digunakan akan didapatkan kembali nilai dari biner plainteks sebelum dilakukan 16 putaran jaringan *feistel* pada proses enkripsi. Hasil dari 16 putaran jaringan feistel pada proses dekripsi merupakan L_0 dan R_0.

Proses dekripsi pesan dijalankan setelah pesan diuraikan dari dalam gambar. Proses dekripsi pesan memerlukan sebuah kunci yang identik dengan kunci untuk proses enkripsi. Dari kunci yang di-input-kan oleh *user* dibangkitkan tiga kunci internal seperti pada proses enkripsi. Pada proses enkripsi dilakukan proses enkripsi dengan kunci internal pertama kemudian dilanjutkan dengan proses dekripsi dengan kunci internal kedua dan dienkripsi lagi dengan kunci ketiga. Sedangkan pada proses dekripsi akan dilakukan proses dekripsi dengan kunci internal ketiga kemudian dilanjutkan proses enkripsi dengan kunci kedua dan didekripsi dengan kunci pertama.

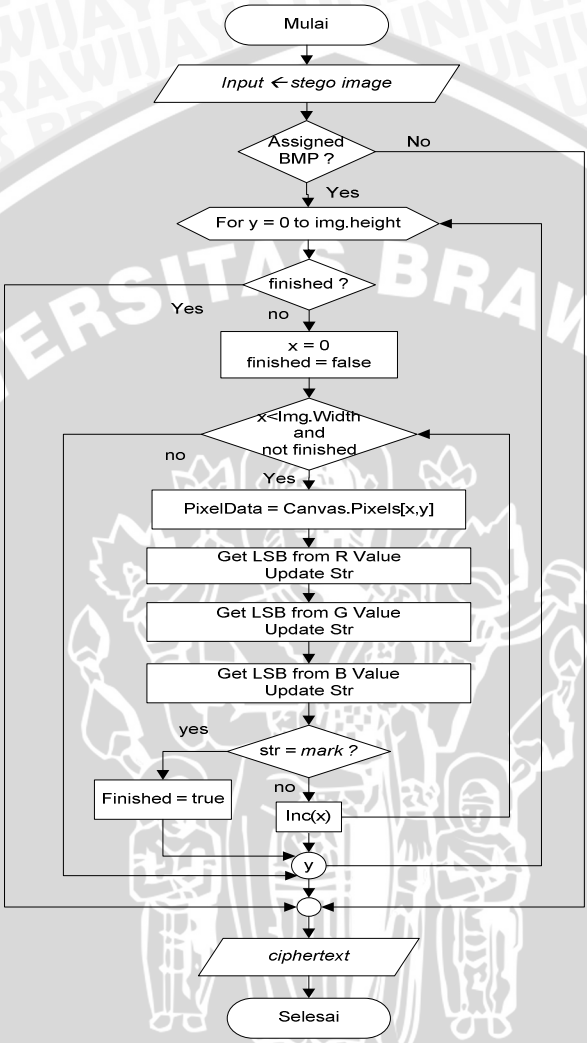


Gambar 3.15 Proses dekripsi 3DES

3.2.2.2 Proses Penguraian Pesan (*Extraction*)

Penguraian pesan dalam gambar berarti mengambil bit – bit karakter yang tersebar dalam *pixel* gambar. Dalam satu *pixel* terdapat tiga bit pesan yaitu pada LSB dari tiap – tiap RGB. Setelah minimal tiga karakter berhasil diuraikan dari dalam gambar akan dilakukan proses pemeriksaan terhadap pesan. Apabila tiga karakter terakhir pesan identik dengan karakter *mark* (penanda) maka proses penguraian pesan telah selesai.

Proses penguraian pesan dalam gambar ditunjukkan oleh gambar 3.16.



Gambar 3.16 *Flowchart* Penguraian Pesan

Berikut ini adalah penjelasan *flowchart* penyisipan pesan dalam gambar :

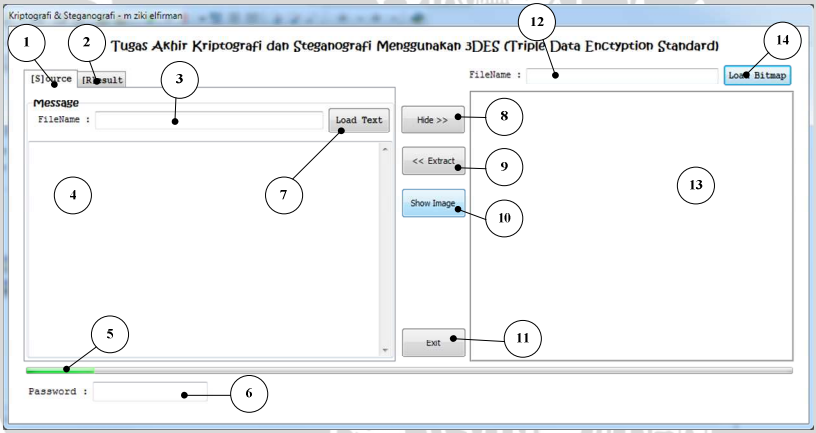
1. Data yang dimasukkan berupa gambar yang mengandung pesan.
2. Gambar yang dipilih harus bmp 24-bit.
3. Lakukan penelusuran pada *pixel – pixel* yang ditentukan.

4. Satu *pixel* langsung dipecah kedalam tiga komponen warna R,G,B. Untuk tiap komponen warna, ambil bit terakhir kemudian hasilnya digunakan untuk update karakter.

5. Setelah minimal tiga karakter berhasil diuraikan dilakukan proses pencocokan tiga karakter terakhir dengan *mark*. Apabila terdapat *mark* pada pesan yang diuraikan maka proses penguraian telah selesai.
6. Keluaran adalah pesan dalam bentuk *ciphertext*.

3.3 Perancangan Antar Muka

Perancangan antar muka untuk kriptografi dan steganografi pada citra digital dapat dilihat pada gambar 3.17.



Gambar 3.17 Rancangan Form Utama

Keterangan Gambar :

1. Tab menu data asal.
2. Tab menu untuk menampilkan teks yang teracak (*ciphertext*).
3. Menunjukkan alamat *file* teks di *harddisk.*
4. Memo untuk menampilkan *file* teks asli.
5. *Progress bar.*
6. Inputan kunci.
7. Tombol untuk memuat *file* teks.

8. Tombol untuk menyandikan pesan dan menyisipkan ke dalam gambar.
9. Tombol untuk menguraikan pesan dari gambar dan mendekripsi pesan.
10. Tombol untuk menampilkan informasi gambar dan ukuran asli gambar.
11. Tombol keluar aplikasi.
12. Menunjukkan alamat gambar di *harddisk*.
13. *Frame* untuk menampilkan gambar.
14. Tombol untuk memuat gambar.

3.4 Perancangan Uji Coba Dan Evaluasi Hasil

Pada sub bab ini akan dilakukan perancangan pengujian perangkat lunak pengamanan data kriptografi dan steganografi, baik pengujian dari perangkat lunak maupun *output* dari perangkat lunak yang telah dibuat. Pengujian terhadap perangkat lunak dibagi menjadi tiga, yaitu pengujian fungsionalitas perangkat lunak, pengujian kinerja perangkat lunak dan pengujian ketahanan citra steganografi terhadap manipulasi citra.

3.4.1 Bahan Pengujian

Bahan pengujian yang digunakan yaitu *filetext* yang berisi pesan rahasia yang akan disipkan dan *file* citra dengan *format bitmap* 24 bit dengan ukuran 100x128, 400x300 dan 512x512. Untuk proses enkripsi dan dekripsi *key* yang akan digunakan adalah qwerty.

3.4.2 Pengujian Fungsionalitas Perangkat Lunak

Untuk menguji fungsionalitas perangkat lunak, pengujian yang dilakukan adalah mencoba menyisipkan pesan ke dalam *file* citra dan mencoba memperoleh kembali pesan yang telah disisipkan. Tabel 3.3 merupakan tabel pengujian perangkat lunak untuk proses steganografi.

Tabel 3.3 Pengujian fungsionalitas perangkat lunak pada proses steganografi

No	Proses Steganografi			
	File Citra	Pesan	Penyisipan (*Embedding*)	Penguraian (*Extract*)

Karena sebelum disisipkan pesan di enkripsi terlebih dahulu maka pengujian juga dilakukan pada hasil enkripsi dan dekripsi pesan. Untuk proses kriptografi ini, pengujian yang dilakukan adalah mencoba mengenkripsi pesan dan mendekripsi *ciphertext* untuk mendapatkan kembali isi pesan. Pada proses dekripsi pesan akan digunakan dua *key* yang berbeda. *Key* pertama merupakan *key* yang sama dengan proses enkripsi, sedangkan *key* yang kedua merupakan *key* yang berbeda. Tabel 3.4 merupakan tabel pengujian fungsionalitas perangkat lunak untuk proses kriptografi.

Tabel 3.4 Pengujian fungsionalitas perangkat lunak pada proses kriptografi

No	Proses Kriptografi			
	Pesan Asli	*Ciphertext*	*Key* untuk Dekripsi	Hasil Dekripsi

3.4.3 Pengujian Kinerja Perangkat Lunak

Skenario pengujian kinerja perangkat lunak antara lain:

1. Memasukkan *filetext* dan *file* citra yang berbeda – beda.
2. Menghitung nilai PSNR (*Peak Signal to Noise Ratio*). Nilai PSNR yang semakin besar menyatakan sinyal keluaran semakin mirip dengan sinyal asli.

Dari proses penyisipan pesan ke dalam *file* citra tentunya akan ada perbedaan kualitas citra sebelum dan sesudah proses penyisipan pesan, untuk mengetahui seberapa besar penurunan kualitas citra maka akan dilakukan perhitungan nilai PSNR seperti yang telah dijelaskan pada bab sebelumnya. Tabel 3.5 adalah tabel pengujian kinerja perangkat lunak dengan menggunakan *filetext* dan *file* citra yang berbeda.

Tabel 3.5 Pengujian Kinerja Perangkat Lunak dengan Variasi *Filetext* dan *File* Citra

No	Pesan	*File* citra asal	Citra Steganografi	Hasil Ekstraksi	Nilai PSNR

3.4.4 Pengujian Ketahanan Citra Steganografi

Pengujian ketahanan citra steganografi (*robustness*) dilakukan untuk mengetahui ketahanan citra steganografi terhadap manipulasi citra yang umum dilakukan. Manipulasi citra yang dilakukan yaitu proses pemotongan citra (*cropping*), manipulasi ukuran citra (*zoom*), penambahan efek sephia, rotasi 90^0 dan perubahan *format* citra. Manipulasi citra dilakukan pada citra steganografi yang sama. Tabel 3.6 adalah tabel hasil pengujian ketahanan perangkat lunak.

Tabel 3.6 Pengujian Ketahanan Citra Steganografi

No	Jenis Serangan	*File* Citra *modified*	Hasil Ekstraksi

3.5 Contoh Perhitungan Manual

Berikut ini merupakan beberapa contoh perhitungan kinerja perangkat lunak yang akan dibuat.

3.5.1 Proses enkripsi dan dekripsi.

Sebagai contoh perhitungan manual pada proses enkripsi 3DES disediakan sebuah pesan dan kunci sebagai berikut:

- **Pesan** = Enkripsi → 01100101-01101110-01101011-01110010-01101001-01110000-01110011-01101001
- **Kunci** = abcdefgh →01100001-01100010-01100011-01100100-01100101-01100110-01100111-01101000

3.5.1.1 Kunci internal

Dari kunci yang diinputkan oleh *user* dibangkitkan tiga kunci internal **k1**, **k2** dan **k3**.

- **Kunci** = abcdefgh
- **K1** = abcdefgh → 01100001-01100010-01100011-01100100-01100101-01100110-01100111-01101000
- **K2** = hgfedcba → 01101000-01100111-01100110-01100101-01100100-01100011-01100010-01100001
- **K3** = abcdefgh → 01100001-01100010-01100011-01100100-01100101-01100110-01100111-01101000

3.5.1.2 *Permutation Choice 1* (PC-1)

Setelah didapatkan nilai biner dari pesan dan kunci, maka kunci akan dilewatkan melalui tabel PC-1 seperti pada gambar 3.6. Melewatkan 64 bit kunci melalui tabel permutasi bertujuan untuk mengacak indeks kunci sesuai dengan urutan pada tabel.
Sebagai contoh 64 bit pesan adalah $\mathbf{K_1}$ = 01100001-01100010-01100011-01100100-01100101-01100110-01100111-01101000.
Setelah dilewatkan melalui tabel PC-1 akan didapatkan $\mathbf{K_{56}}$ = 0000000-0111111-1111111-1110000-0110011-0011110-0010000-0000000 yang mempunyai panjang 56 bit.

3.5.1.3 *Shift left*

Kunci K56 akan dibagi menjadi dua bagian yaitu C_0 dan D_0 yang mempunyai panjang masing – masing 28 bit.

$\mathbf{K_{56}}$ = 0000000-0111111-1111111-1110000 –
0110011-0011110-0010000-0000000
$\mathbf{C_0}$ = 0000000-0111111-1111111-1110000
$\mathbf{D_0}$ = 0110011-0011110-0010000-0000000

Dilakukan pergeseran bit sebanyak satu atau dua bit ke kiri sesuai pada tabel 3.1 pada tiap bagian sebanyak 16 kali sehingga dihasilkan $\mathbf{C_1..C_{16}}$ dan $\mathbf{D_1..D_{16}}$ sebagai berikut:

$\mathbf{C_1}$ = 00000001111111111111111100000

$\mathbf{D_1}$ = 11001100111100010000000000
$\mathbf{C_2}$ = 00000011111111111111111000000
$\mathbf{D_2}$ = 10011001111000100000000000001
$\mathbf{C_3}$ = 00001111111111111111100000000
$\mathbf{D_3}$ = 01100111100010000000000000110
$\mathbf{C_4}$ = 00111111111111111110000000000
$\mathbf{D_4}$ = 10011110001000000000000011001
$\mathbf{C_5}$ = 11111111111111110000000000000
$\mathbf{D_5}$ = 01111000100000000000001100110
$\mathbf{C_6}$ = 11111111111111000000000000011
$\mathbf{D_6}$ = 11100010000000000000110011001
$\mathbf{C_7}$ = 11111111111110000000000001111
$\mathbf{D_7}$ = 10001000000000000011001100111
$\mathbf{C_8}$ = 11111111110000000000000111111
$\mathbf{D_8}$ = 00100000000000011001100111110
$\mathbf{C_9}$ = 11111111100000000000001111111
$\mathbf{D_9}$ = 01000000000000011001100111100
$\mathbf{C_{10}}$ = 11111110000000000000111111111
$\mathbf{D_{10}}$ = 00000000000011001100111100001
$\mathbf{C_{11}}$ = 11111000000000000011111111111
$\mathbf{D_{11}}$ = 00000000001100110011111000100
$\mathbf{C_{12}}$ = 11100000000000001111111111111
$\mathbf{D_{12}}$ = 00000000110011001111100010000
$\mathbf{C_{13}}$ = 10000000000000111111111111111
$\mathbf{D_{13}}$ = 00000011001100111110001000000
$\mathbf{C_{14}}$ = 00000000000111111111111111110
$\mathbf{D_{14}}$ = 00001100110011111000100000000
$\mathbf{C_{15}}$ = 00000000011111111111111111000
$\mathbf{D_{15}}$ = 00110011001111000010000000000
$\mathbf{C_{16}}$ = 00000000111111111111111110000
$\mathbf{D_{16}}$ = 01100110011110001000000000000

3.5.1.4 *Permutation Choice 2* (PC-2)

Membuat subkunci $\mathbf{K_1}$..$\mathbf{K_{16}}$. $\mathbf{K_1}$ didapatkan dari menggabungkan $\mathbf{C_1}$ dan $\mathbf{D_1}$ kemudian melewatkannya melalui tabel PC-2 seperti pada gambar 3.7.

$\mathbf{K_1}$ = 111000001011111001100110000100110010101010000010

$\mathbf{K_2}$ = 111000001011011001110110000100000010001100000111
$\mathbf{K_3}$ = 111001001101011001110110101101100000000010000100
$\mathbf{K_4}$ = 111001101101001101110010010000000010001111000011
$\mathbf{K_5}$ = 101011101101001101110011001101101010000000001001
$\mathbf{K_6}$ = 101011110101001101011011011000100001010101000010
$\mathbf{K_7}$ = 001011110101001111011001000011001010000100101010
$\mathbf{K_8}$ = 000111110101100111011001011001000101110001000000
$\mathbf{K_9}$ = 000111110100100111011001010010101001100001000000
$\mathbf{K_{10}}$ = 000111110110100110011101110000001100010100111000
$\mathbf{K_{11}}$ = 000111110010110110001101000010010001111000001000
$\mathbf{K_{12}}$ = 010110110010110010101101110110000101000000110000
$\mathbf{K_{13}}$ = 110110011010110010101100000000010100101000101100
$\mathbf{K_{14}}$ = 110100001010111010101110100100000011100010010000
$\mathbf{K_{15}}$ = 111100001011111000100110101000010000001000110101
$\mathbf{K_{16}}$ = 111100001011111000100110101000110100001010000000

3.5.1.5 *Initial Permutation* (IP)

Pada tahapan ini 64 bit pesan dilewatkan melalui tabel IP seperti pada gambar 3.8 kemudian membagi hasil permutasi ke dalam dua bagian masing – masing mempunyai panjang 32 bit.

Pesan = 01100101-01101110-01101011-01110010-01101001–
01110000-01110011-01101001
Pesan(IP) = 11111111-01101000-00000011-11010101 –
00000000-11111111-10010110-01001110
$\mathbf{L_0}$ = 11111111-01101000-00000011-11010101
$\mathbf{R_0}$ = 00000000-11111111-10010110-01001110

3.5.1.6 Jaringan *Feistel*

Dengan mengimplementasikan persamaan 3.1 dan 3.2 dilakukan proses perhitungan sebagai berikut:
$f(\mathbf{R_0}, \mathbf{K_1})$ = (111000001011111001100110000100110010101010000010) ⊕ (000000000001011111111111110010101100001001011100) = 111000001010100110011001110110011111010001101111 0.
$f(\mathbf{R_0}, \mathbf{K_1})$ = 111000-001010-100110-011001-110110-011110-100011-011110. Maka $\mathbf{B_1}$ = 111000, $\mathbf{B_2}$ = 001010, $\mathbf{B_3}$ = 100110, $\mathbf{B_4}$ = 011001, $\mathbf{B_5}$ = 110110, $\mathbf{B_6}$ = 011110, $\mathbf{B_7}$ = 100011, $\mathbf{B_8}$ = 011110.

Untuk $\mathbf{B_1}$ = 111000, dicari nilai desimal dari biner **10** yaitu 2 (dua), nilai **2** akan menjadi **indeks baris** pada S1. Selanjutnya biner **1100** = 12, nilai **12** akan menjadi **indeks kolom** pada S1.

S1

Kolom / Baris	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Pada tabel S1 perpotongan baris ke 2 dan kolom ke 12 adalah 3 dan nilai biner dari 3 adalah 0011. Jadi S1(111000) = 0011, S2(111000) = 0011, S3(111000) = 0011, S4(111000) = 0011, S5(111000) = 0011, S6(111000) = 0011, S7(111000) = 0011, S8(111000) = 0011. Hasil dari tiap – tiap putaran jaringan *feistel* merupakan gabungan dari S1+S2+S3+S4+S5+S6+S7+S8 yang kemudian dilewatkan melalui tabel *Permutation P*.

R_1L_1=
00001001010001100110110011111011000000000111111111001011
001001110
R_2L_2=
01011000000010001011010001011111000010010100011001101100
11110110
R_3L_3=
01011101100110000101011111100010010110000000100010110100
01011111
R_4L_4=
11000110101100010100010000000011010111011001100001010111
11100010
R_5L_5=
01001000101011001100110011101110110001101011000101000100
00000011
R_6L_6=
00000001101001110011100111111010001001000101011001100110
11101110

R_7L_7=
00101110100111111101111010110110000000011010011100111001111110100

R_8L_8=
01001001101011011001100101010001001011101001111111011110101101 10

R_9L_9=
10010001001111001010010101110101010010011010110110011001010100 01

$R_{10}L_{10}$=
10111111100111001011001000111011100100010011110010100101011101 01

$R_{11}L_{11}$=
00111001111001100000111111111011010111111100111001011001000111011

$R_{12}L_{12}$=
01010010011111011101010000010011001110011110011000001111111101 10

$R_{13}L_{13}$=
01000011110011000111001100000010010100100111111011101010000010011

$R_{14}L_{14}$=
01110101000100100100100001010100010000111100110001110011000000010

$R_{15}L_{15}$=
11111000010001011010111111011011011101010001001001001000010100

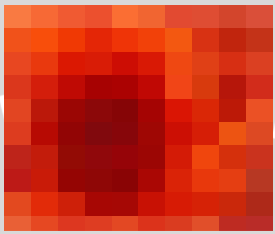
$R_{16}L_{16}$=
11110010100100101011101100001011111110000100010110101111110110 11

Setelah nilai dari final blok $\mathbf{R_{16}L_{16}}$ didapatkan, nilai $\mathbf{R_{16}L_{16}}$ tersebut akan dilewatkan melalui tabel *invers permutation*. Hasil dari permutasi invers merupakan *ciphertext* dari proses enkripsi DES.

Ciphertext = 00101111 01011111 00101000 10001111 11010110 11001100 11100010 11011110

3.5.2 Proses penyisipan pesan.

Dimisalkan gambar citra 8 x 8 ditunjukkan oleh gambar 3.18.



Gambar 3.18 Citra 8 x 8 *pixel*

Dengan nilai RGB yang ditunjukkan oleh tabel 3.7.

Tabel 3.7 Nilai RGB Citra 8 x 8

Pixel	**R**	**G**	**B**
Pixel ke - 1	249	123	69
Pixel ke - 2	240	82	31
Pixel ke - 3	225	64	36
Pixel ke - 4	222	67	37
Pixel ke - 5	214	57	34
Pixel ke - 6	192	36	29
Pixel ke - 7	216	56	29
Pixel ke - 8	236	102	59
Pixel ke - 9	247	101	50
Pixel ke - 10	241	62	7
Pixel ke - 11	215	28	9
Pixel ke - 12	178	14	7
Pixel ke - 13	168	9	3
Pixel ke - 14	176	20	7
Pixel ke - 15	207	31	7
Pixel ke - 16	229	67	35
Pixel ke - 17	237	82	47
Pixel ke - 18	229	36	3
Pixel ke – 19	191	9	2
Pixel ke - 20	145	4	6
Pixel ke - 21	133	8	10

Pixel ke - 22	138	5	6
Pixel ke - 23	163	10	4
Pixel ke - 24	217	58	36
Pixel ke - 25	249	102	49
Pixel ke - 26	228	34	5
Pixel ke - 27	182	3	2
Pixel ke - 28	136	3	6
Pixel ke - 29	134	7	11
Pixel ke - 30	142	4	7
Pixel ke - 31	150	3	3
Pixel ke - 32	218	65	37
Pixel ke - 33	242	102	49
Pixel ke - 34	238	57	8
Pixel ke - 35	209	22	8
Pixel ke - 36	178	8	4
Pixel ke - 37	164	6	4
Pixel ke - 38	167	7	4
Pixel ke - 39	193	14	4
Pixel ke - 40	219	54	31
Pixel ke - 41	228	81	52
Pixel ke - 42	235	76	19
Pixel ke - 43	237	74	21
Pixel ke – 44	225	36	7
Pixel ke - 45	212	21	4
Pixel ke - 46	229	49	10
Pixel ke - 47	220	34	6
Pixel ke - 48	223	70	39
Pixel ke - 49	213	71	48
Pixel ke - 50	208	45	18
Pixel ke - 51	203	43	15
Pixel ke - 52	195	31	6
Pixel ke - 53	228	64	13
Pixel ke - 54	232	64	15
Pixel ke - 55	219	48	11
Pixel ke - 56	204	60	39
Pixel ke - 57	213	82	60
Pixel ke - 58	205	58	31
Pixel ke - 59	211	52	35
Pixel ke – 60	221	71	38
Pixel ke - 61	223	80	39
Pixel ke - 62	199	59	37
Pixel ke - 63	184	53	33
Pixel ke - 64	186	52	48

Misalkan pesan yang akan disisipkan berupa teks:
Pesan = Enkripsi → 01100101-01101110-01101011-01110010-01101001-01110000-01110011-01101001.
Pesan mempunyai panjang 64 bit sehingga diperlukan (64 div 3) *pixel* karena setiap *pixel* dapat menyimpan 1 (satu) bit pesan.
Karakter "E" dengan biner **01100101** dapat disisipkan ke dalam 3 (tiga) *pixels* yaitu:
Pixel ke-1 → R = 249 - 11111001, G = 123 - 01111011, B = 69 - 01000101
Pixel ke-2 → R = 240 - 01110000, G = 82 - 01010010, B = 31 - 00011111
Pixel ke-3 → R = 225 - 11100001, G = 64 - 01000000, B = 36 - 00100100
Sehingga susunan *pixel* berubah menjadi:
Pixel ke-1 → R = 249 - 1111100**0**, G = 123 - 0111101**1**, B = 69 - 0100010**1**
Pixel ke-2 → R = 240 - 0111000**0**, G = 82 - 0101001**0**, B = 31 - 0001111**1**
Pixel ke-3 → R = 225 - 1110000**0**, G = 64 - 0100000**1**, B = 36 - 00100100

3.5.3 Proses penguraian pesan.

Untuk menguraikan pesan dari dalam gambar diambil LSB terakhir dari RGB pada tiap – tiap *pixel*. Setelah nilai – nilai LSB didapatkan akan dilakukan proses pencocokan nilai – nilai LSB tersebut terhadap penanda akhir pesan ('***#$**'). Akhir dari pesan akan selalu sama dengan nilai berikut:

***#$ → 00101010-00100011-00100100**

Misalnya didapatkan susunan RGB *pixel* gambar stego berupa tabel 3.8.

Tabel 3.8 RGB Gambar Steganografi

Pixel	**R**	**G**	**B**
Pixel ke – 1	10011010	00110011	10010101
Pixel ke – 2	01010110	10101010	01101101
Pixel ke – 3	01010100	00000111	00011110
Pixel ke – 4	00101011	01101101	01010100
Pixel ke – 5	01010101	01010101	00010001
Pixel ke – 6	01101010	01010100	00101011
Pixel ke – 7	10011011	01101100	01010101
Pixel ke – 8	00101010	01010101	10011011

Pixel ke – 9	10011010	01010101	01101101
Pixel ke – 10	10011011	01010100	00101010
Pixel ke – 11	01101101	01101100	00101010
Pixel ke – 12	00011111	00010011	00101010
Pixel ke – 13	01101011	00101010	00011110
Pixel ke – 14	01101101	00000010	00101011
Pixel ke – 15	00000111	00000111	01000110
Pixel ke – 16	00011000	00101010	01000000
Pixel ke – 17	01001000	01001001	00100001
Pixel ke – 18	00101011	01101100	01100010
Pixel ke – 19	01010101	01111001	00101010
Pixel ke – 20	01010011	01010101	01100100
Pixel ke – 21	01010101	01000110	00101010
Pixel ke – 22	01010101	01100110	01010110
Pixel ke – 23	01001111	01111110	00101011
Pixel ke – 24	00111110	01111111	11001010
Pixel ke – 25	00101010	10001110	10011111
Pixel ke – 26	11000000	10101010	01010010
Pixel ke – 27	00101011	00110111	01010100
Pixel ke – 28	10011000	00101011	10011110
Pixel ke – 29	10001000	01001111	00101010
Pixel ke – 30	0110111**0**	10001110	10011111

Uraikan LSB dari tiap – tiap RGB. Setelah minimal 3 (tiga) karakter atau 24 bit LSB didapatkan, dilakukan proses pencocokkan dengan nilai penanda **00101010-00100011-00100100**. Apabila 24 bit terakhir mempunyai nilai yang sama dengan nilai penanda maka proses penguraian LSB tidak akan dilanjutkan. Dari tabel 3.5 didapatkan nilai LSB:

01100101-01101110-01101011-01110010-01101001-01110000-
01110011-01101001-**00101010-00100011-0010010****0**

Penguraian LSB hanya sampai nilai R pada *pixel* ke – 30 karena 24 bit terakhir mempunyai nilai yang sama dengan nilai penanda akhir pesan. Setelah diuraikan nilai bit – bit pesan akan dikonversi ke dalam bentuk *char* (karakter) sehingga didapatkan teks pesan asli (*plaintext*).

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

Pada bab ini akan dijelaskan seluruh proses yang sudah dirancang pada bab sebelumnya, tampilan antarmuka dan bagian – bagian *source code* yang dibuat, serta analisa terhadap data yang dihasilkan sistem.

4.1 Lingkungan Implementasi

Implementasi merupakan proses transformasi representasi rancangan ke dalam bahasa pemrograman yang dapat dimengerti oleh komputer. Pada bab ini, lingkungan implementasi yang akan dijelaskan meliputi lingkungan implementasi perangkat keras dan perangkat lunak.

4.1.1 Lingkungan perangkat keras

Perangkat keras yang digunakan dalam pengembangan dan pengujian sistem kriptografi dan steganografi ini adalah sebuah Notebook dengan spesifikasi sebagai berikut :

1. Monitor : 13,3”
2. CPU : Intel Core 2 Duo P8600, 2.40 GHz
3. Hard Disk : 320 GB
4. Memori : 3 GB
5. Sound Card : IDT
6. Perangkat Masukan : Keyboard, Mouse

Perangkat keras ini akan difungsikan sebagai tempat perangkat lunak untuk penelitian dijalankan.

4.1.2 Lingkungan Perangkat lunak

Perangkat lunak yang digunakan dalam pengembangan sistem kriptografi dan steganografi ini adalah :

1. Sistem operasi *Windows 7*™ sebagai tempat aplikasi dijalankan.
2. *Embarcadero Delphi 2010* sebagai *programming software development* dalam pembuatan sistem kriptografi dan steganografi.

4.2 Implementasi Program

Berdasarkan analisa dan perancangan proses yang telah dipaparkan pada Bab III, maka pada bab ini akan dijelaskan proses-proses implementasinya.

4.2.1 Implementasi *load image format bitmap*

Tahap awal yang dilakukan dari pemrosesan pengamanan data kriptografi dan steganografi adalah memanggil file citra *format bitmap* ke dalam perangkat lunak. Pada saat pemanggilan file citra, file citra akan diseleksi terlebih dahulu apakah sudah memenuhi syarat kemudian akan dihitung jumlah data yang bisa ditampung oleh file citra yang sudah memenuhi seleksi tersebut. Prosedur yang digunakan ditunjukkan pada *sourcecode* 4.1.

```
procedure TFormMain.bt_loadbmpClick(Sender: TObject);
var
  len: integer;
begin
  ImageWall.Picture.Bitmap.FreeImage;
  if OpenPictureDialog1.Execute then
ImageSource.Picture.LoadFromFile(OpenPictureDialog1.FileName);

  if ImageSource.Picture.Bitmap.PixelFormat <> pf24bit then
  begin
    ShowMessage('Gambar Bitmap harus mempunyai kedalama 24-
bit! Silahkan Load gambar lagi.');

    ImageSource.Picture:= ImageClear.picture;
  end
  else
  begin
    ed_bmpfilename.Text:= OpenPictureDialog1.FileName;
    Form_Enlarge.ImageEnlarge.Picture:= ImageSource.Picture;
    FormImage.ImgSource.Picture:= ImageSource.Picture;
  end;

  bt_loadtxt.enabled:= true;
  ImageWall.Visible:= false;
  len:= ((imagesource.Width*imagesource.Height)*3)-32;
  label8.caption:= inttostr(len);
end;
```

***Sourcecode* 4.1** Prosedur *Load Bitmap*

4.2.2 Implementasi *load text format filetext*

Setelah file citra dimasukkan ke dalam gambar dan diketahui jumlah data yang dapat disimpan maka *user* dapat memanggil data teks yang akan dienkripsi. Pada proses pemanggilan data teks akan dilakukan perbandingan antara ukuran data teks dengan ukuran file citra. Apabila ukuran data teks lebih besar dari ukuran file citra maka jumlah data harus dikurangi agar dapat diproses. Implementasi *load text* dapat dilihat pada *sourcecode* 4.2.

```
procedure TFormMain.bt_loadtxtClick(Sender: TObject);
var
  msglength,maxlength: Integer;
begin
  MemoSource.Lines.Clear;
  if OpenDialog1.Execute then
    MemoSource.lines.LoadFromFile(OpenDialog1.FileName);
  ed_txtfilename.Text:= OpenDialog1.FileName;
  maxlength:= (imagesource.Width*imagesource.Height*3) div 8;
  msglength:= (length(MemoSource.Lines.text)*8);
  if msglength>maxlength then
  begin
    showmessage('Teks terlalu panjang!');
    memosource.Lines.clear;
  end;
end;
```

***Sourcecode* 4.2** Prosedur *Load Text*

4.2.3 Implementasi enkripsi algoritma 3DES

Langkah selanjutnya adalah mengimplementasikan algoritma 3DES untuk mengenkripsi data teks. Pada prosedur ini dilakukan pemanggilan prosedur *encrypt* dan *decrypt* yang masing – masing merupakan implementasi dari algoritma DES.

Prosedur ini akan menghasilkan nilai biner dari data teks yang dimasukkan untuk kemudian disisipkan ke dalam file citra. Untuk mengetahui hasil dari enkripsi data dikonvesi ke dalam bentuk heksadesimal dan ditampilkan pada MemoResult. Prosedur enkripsi 3DES dapat dilihat pada *sourcecode* 4.3.

```
procedure encrypt3DES(msgkey,msgsource: string; var msgresult:
string);
var
  i,len: integer;
```

```
  str,output,mem1,hex: string;
begin
  createintenalkey(msgkey);
  str:= msg.isi;
  len:= length(str) div 8;
  if (length(str)-(len*8))>0 then
  begin
    for i:= 1 to (8-(length(str)-(len*8))) do
      str:= str+' ';
  end;
  formmain.Memo1.Lines.Clear;
  formmain.Memo2.Lines.Clear;
  mem1:= '';
  for i:= 1 to length(str) do
  begin
    mem1:= mem1+chrtobin(str[i]);
  end;
  formmain.Memo1.Lines.Text:= mem1;
  encrypt(formmain.Memo1.Lines.Text,inKey.k123[1]);
  decrypt(formmain.Memo2.Lines.Text,inKey.k123[2]);
  encrypt(formmain.Memo1.Lines.text,inKey.k123[3]);
  hex:= '';
  for i:= 1 to length(formmain.Memo2.Lines.Text) div 8 do
  begin
    hex:=
hex+IntToHex(bintodes(copy(formmain.Memo2.Lines.Text,i*8-
7,8)),2);
  end;
  Formmain.MemoResult.Text:= hex;
end;
```

***Sourcecode* 4.3** Prosedur Enkripsi 3DES

4.2.4 Implementasi dekripsi algoritma 3DES

Implementasi dari dekripsi algoritma 3DES mempunyai kemiripan dengan implementasi enkripsi 3DES, perbedaannya terletak pada urutan pemanggilan prosedur enkripsi dan dekripsi DES. Prosedur dekripsi 3DES dapat dilihat pada *sourcecode* 4.4.

```
procedure decrypt3DES(msgkey: string; var msgresult: string);
var
  i: integer;
  output,res: string;
begin
  createintenalkey(msgkey);
  decrypt(formmain.Memo2.Lines.Text,inKey.k123[3]);
  encrypt(formmain.Memo1.Lines.Text,inKey.k123[2]);
  decrypt(formmain.Memo2.Lines.text,inKey.k123[1]);
  formmain.MemoResult.Lines.Clear;
  res:= '';
```

```
  for i:= 1 to length(formmain.Memo1.Lines.Text) div 8 do
  begin
    output:= copy(formmain.Memo1.Lines.Text,i*8-7,8);
    res:= res+bintostr(output);
  end;
  formmain.MemoResult.Lines.Text:=
formmain.MemoResult.Lines.Text+'Hasil decrypt : '+res;
end;
```

***Sourcecode* 4.4** Prosedur Dekripsi 3DES

4.2.5 Implementasi Enkripsi DES

Metode 3DES merupakan metode DES yang dijalankan sebanyak tiga kali dengan penggunaan kunci yang berbeda dan urutan enkripsi dan dekripsi yang berbeda seperti yang sudah dijelaskan pada bab sebelumnya. Prosedur enkripsi DES ditunjukkan oleh *sourcecode* 4.5.

```
procedure encrypt(pesan,nokunci: string);
var
  pjg,i,n: integer;
  sbin,LR,f,RL,IP_1,str: string;
  L,R: array[0..16] of string[32];
begin
  formmain.ProgressBar1.Max:=
10+(length(formmain.memo1.Lines.text) div 64);
  create16subkey(nokunci);
  FormMain.ProgressBar1.Position:=
FormMain.ProgressBar1.Position+10;
  pjg:= length(pesan);
  str:= '';
  for i:= 1 to (pjg div 64) do
  begin
    sbin:= copy(pesan,i*64-63,64);
    LR:= initpermut(sbin);
    L[0]:= copy(LR,1,32);
    R[0]:= copy(LR,33,32);
    for n:= 1 to 16 do
    begin
      L[n]:= R[n-1];
      f:= feistel(R[n-1],K[n]);
      R[n]:= fxor(L[n-1],f);
    end;
    IP_1:= '';
    RL:= R[16]+L[16];
    for n:= 1 to 64 do
      IP_1:= IP_1+copy(RL,IP1[n],1);
    str:= str+ip_1;
    FormMain.ProgressBar1.Position:=
FormMain.ProgressBar1.Position+1;
```

```
  end;
  formmain.Memo2.Lines.Text:= str;
  formmain.Memo1.Lines.clear;
end;
```

***Sourcecode* 4.5** Prosedur Enkripsi DES

4.2.6 Implementasi Dekripsi DES

Prosedur dekripsi DES mempunyai proses – proses yang sama dengan prosedur enkripsi DES. Perbedaannya terletak pada urutan penggunaan kunci pada 16 putaran jaringan *feistel*. Prosedur dekripsi DES ditunjukkan pada *sourcecode* 4.6.

```
procedure decrypt(pesan,nokunci: string);
var
  pjg,i,n,x: integer;
  sbin,LR,f,RL,IP_1,str: string;
  L,R: array[0..16] of string[32];
begin
  create16subkey(nokunci);
  pjg:= length(pesan);
  str:= '';
  for i:= 1 to (pjg div 64) do
  begin
    sbin:= copy(pesan,i*64-63,64);
    LR:= initpermut(sbin);
    L[0]:= copy(LR,1,32);
    R[0]:= copy(LR,33,32);
    x:= 16;
    for n:= 1 to 16 do
    begin
      L[n]:= R[n-1];
      f:= feistel(R[n-1],K[x]);
      R[n]:= fxor(L[n-1],f);
      x:= x-1;
    end;
    IP_1:= '';
    RL:= R[16]+L[16];
    for n:= 1 to 64 do
      IP_1:= IP_1+copy(RL,IP1[n],1);
    str:= str+ip_1;
  end;
  formmain.Memo1.Lines.Text:= str;
  formmain.Memo2.Lines.Clear;
end;
```

***Sourcecode* 4.6** Prosedur Dekripsi DES

4.2.7 Implementasi Penyisipan Pesan

Pada tahap pengamanan pesan, setelah pesan dienkripsi maka pesan yang berupa biner akan disipkan ke dalam bit – bit terakhir dari *file* citra yang sudah diinputkan. Proses penyisipan atau *embedding* akan menghasilkan *file* citra steganografi. Prosedur penyisipan pesan ke dalam *file* citra ditunjukkan oleh *sourcecode* 4.7.

```
procedure embedded(var img: TImage);
var
  tinggi,lbr,i: integer;
  pix: tColor;
  r,g,b,r1,g1,b1: byte;
  teks,biner,nbin: string;
  now: char;
begin
  teks:=
FormMain.Memo2.Lines.Text+'00101010001000110010010 0';
FormProcess.Memo1.Lines.Add('teks: '+teks);
  FormProcess.Memo1.Lines.Add(' ');
  FormProcess.Memo1.Lines.Add('Jumlah Biner:
'+inttostr(length(teks)));
  lbr:= 0;
  tinggi:= 0;
  now:= 'B';
  for i:= 1 to length(teks) do
  begin
    pix:= img.picture.Bitmap.Canvas.Pixels[lbr,tinggi];
    FormProcess.Memo2.Lines.Add('Biner['+inttostr(i)+'] =
'+teks[i]);
    if now='B' then
    begin
      r1:= GetRValue(pix);
      biner:= destobin(r1);
      nbin:= Copy(biner,1,7)+teks[i];
      r:= bintodes(nbin);
      now:= 'R';
      FormProcess.Memo2.Lines.Add('  Disisipkan ke
pixel['+inttostr(lbr)+','+inttostr(tinggi)+']');
      FormProcess.Memo2.Lines.Add('  RGB: '+now);
      FormProcess.Memo2.Lines.Add('  '+now+' asal=
'+inttostr(r1)+ ', '+now+' hasil= '+inttostr(r));
      if i=length(teks) then
      begin
        g:= GetGValue(pix);
        b:= GetBValue(pix);
        img.picture.Bitmap.Canvas.Pixels[lbr,tinggi]:=
rgb(r,g,b);
        img.Picture.Bitmap.Canvas.Refresh;
      end;
    end
    else if now='R' then
```

```
    begin
      g1:= GetGValue(pix);
      biner:= destobin(g1);
      nbin:= Copy(biner,1,7)+teks[i];
      g:= bintodes(nbin);
      now:= 'G';
      FormProcess.Memo2.Lines.Add('  Disisipkan ke
pixel['+inttostr(lbr)+','+inttostr(tinggi)+']');
      FormProcess.Memo2.Lines.Add('  RGB: '+now);
      FormProcess.Memo2.Lines.Add('  '+now+' asal=
'+inttostr(g1)+ ', '+now+' hasil= '+inttostr(g));
      if i=length(teks) then
      begin
        b:= GetBValue(pix);
        img.picture.Bitmap.Canvas.Pixels[lbr,tinggi]:=
rgb(r,g,b);
        img.Picture.Bitmap.Canvas.Refresh;
      end;
    end
    else if now='G' then
    begin
      b1:= GetBValue(pix);
      biner:= destobin(b1);
      nbin:= Copy(biner,1,7)+teks[i];
      b:= bintodes(nbin);
      now:= 'B';
      img.picture.Bitmap.Canvas.Pixels[lbr,tinggi]:=
rgb(r,g,b);
      img.Picture.Bitmap.Canvas.Refresh;
      FormProcess.Memo2.Lines.Add('  Disisipkan ke
pixel['+inttostr(lbr)+','+inttostr(tinggi)+']');
      FormProcess.Memo2.Lines.Add('  RGB: '+now);
      FormProcess.Memo2.Lines.Add('  '+now+' asal=
'+inttostr(b1)+ ', '+now+' hasil= '+inttostr(b));
      inc(lbr);
      if lbr=img.picture.Width-1 then
      begin
        inc(tinggi);
        lbr:= 0;
      end;
    end;
  end;
end;
```

***Sourcecode* 4.7** Prosedur Penyisipan Pesan

4.2.8 Implementasi Penguraian Pesan

Proses penguraian pesan dari citra steganografi dilakukan dari *pixel* pertama citra steganografi dan pada tiap – tiap komponen warna pada *pixel* tersebut dengan urutan RGB atau *red-green-blue*. Penguraian akan terus dilakukan sampai karakter penanda

ditemukan. Prosedur penguraian pesan ditunjukkan oleh *sourcecode* 4.8.

```
procedure extract(img: TImage);
var
  lbr,tinggi,len: integer;
  hasil,mrk,biner,s,last: string;
  now: char;
  pix: tColor;
  r,g,b: byte;
begin
  formmain.Memo2.Lines.Clear;
  mrk:= '*#$';
  hasil:= '';
  s:= '';
  for tinggi:= 0 to img.Height-1 do
  begin
    lbr:= 0;
    if finished(hasil,mrk) then break;
    while (lbr<Img.Width-1) and not(finished(hasil,mrk)) do
    begin
      Pix:= img.picture.Bitmap.Canvas.Pixels[lbr,tinggi];
      r:= GetRValue(Pix);
      biner:= destobin(r);
      s:= s+copy(biner,8,1);
      g:= GetGValue(Pix);
      biner:= destobin(g);
      s:= s+copy(biner,8,1);
      b:=GetBValue(Pix);
      biner:= destobin(b);
      s:= s+copy(biner,8,1);
      len:= length(s);
      if len>23 then
      begin
        if ((len mod 8)=0) or ((length(s) mod 8)=1) or
((length(s) mod 8)=2) then
        begin
          if len mod 8= 1 then
            len:= len-1;
          if len mod 8= 2 then
            len:= len-2;
          last:= copy(s,len-23,24);
          hasil:= bintostr(last);
          if StrComp(pchar(hasil),pchar(mrk)) = 0 then
            break;
        end;
      end;
      hasil:= bintostr(s);
      inc(lbr);
    end;
  end;
  formmain.Memo2.Lines.text:= copy(s,1,length(s)-24);
```

***Sourcecode* 4.8** Prosedur Penguraian Pesan

4.2.9 Implementasi Perhitungan *Peak Signal to Noise Ratio*

PSNR digunakan untuk menghitung galat error yaitu perbedaan intensitas *pixel* antara citra steganografi dengan citra asli. Semakin besar nilai PSNR maka semakin bagus citra steganografi tersebut. Prosedur PSNR dapat dilihat pada *sourcecode* 4.9.

```
procedure TFormImage.FormShow(Sender: TObject);
var
  i,j: integer;
  psnr,GapPixel,GapPixelR,GapPixelG,GapPixelB,mse: double;
begin
  ed_psnr.Text:= '';
  GapPixelR:=0;
  GapPixelG:=0;
  GapPixelB:=0;
  GapPixel:=0;
  mse:= 0;
  for i:=0 to ImgSource.Height do
  begin
    for j:=0 to ImgSource.Width do
    begin
      GapPixelR:= GapPixelR +
Power(GetRValue(ImgSource.Canvas.Pixels[i,j])-
      GetRValue(ImgStegano.Canvas.Pixels[i,j]),2);
      GapPixelG:= GapPixelG +
Power(GetGValue(ImgSource.Canvas.Pixels[i,j])-
GetGValue(ImgStegano.Canvas.Pixels[i,j]),2);
      GapPixelB:= GapPixelB +
Power(GetBValue(ImgSource.Canvas.Pixels[i,j])-
GetBValue(ImgStegano.Canvas.Pixels[i,j]),2);
    end;
  end;
  GapPixel:= (GapPixelR+GapPixelG+GapPixelB)/3;
  mse:= GapPixel / (ImgSource.Height*ImgSource.Width);
  psnr:= 20*Log10(255 /Sqrt(mse));
  ed_psnr.Text:= FloatToStrF(psnr,ffFixed,4,2);
end;
```

***Sourcecode* 4.9** Prosedur Perhitungan PSNR

4.2.10 Implementasi Penyimpanan *File* Citra

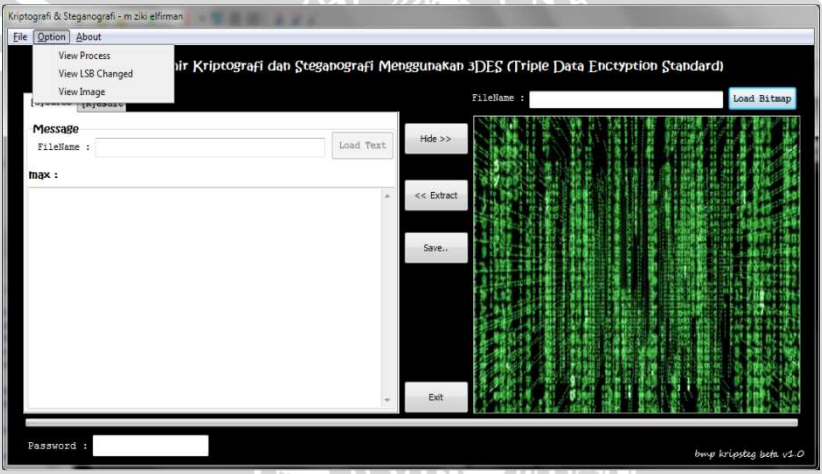
Perubahan LSB tidak secara langsung mengubah bit – bit pada citra asli sehingga untuk menyimpan citra yang sudah disisipi pesan harus menggunakan prosedur penyimpanan. Prosedur untuk menyimpan citra yang sudah disisipi pesan dapat dilihat pada *sourcecode* 4.10.

```
procedure TFormMain.bt_saveClick(Sender: TObject);
begin
  if SavePictureDialog1.Execute then
  begin
imagesource.Picture.Bitmap.SaveToFile(SavePictureDialog1.FileName+
'.bmp');
  end;
end;
```

***Sourcecode* 4.10** Prosedur Penyimpanan *File* citra

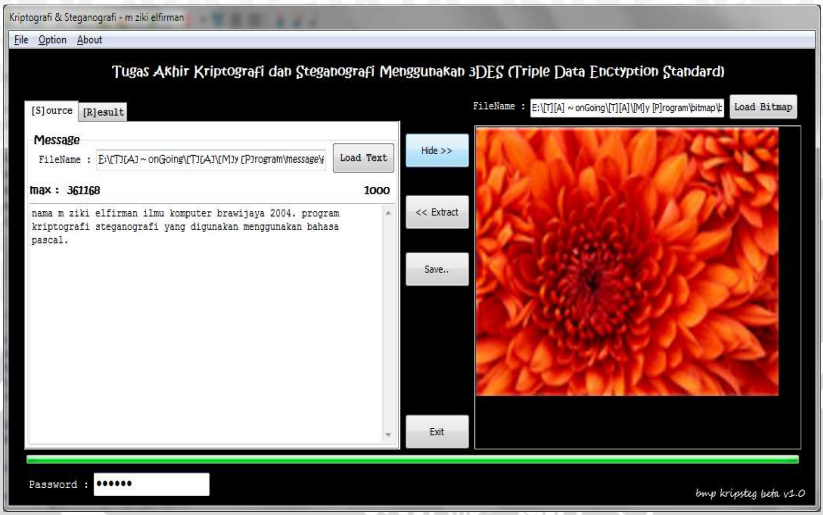
4.3 Implementasi Antarmuka

Berdasarkan rancangan antarmuka yang telah dikemukakan pada Bab 3 maka dihasilkan antaramuka pada gambar 4.1.



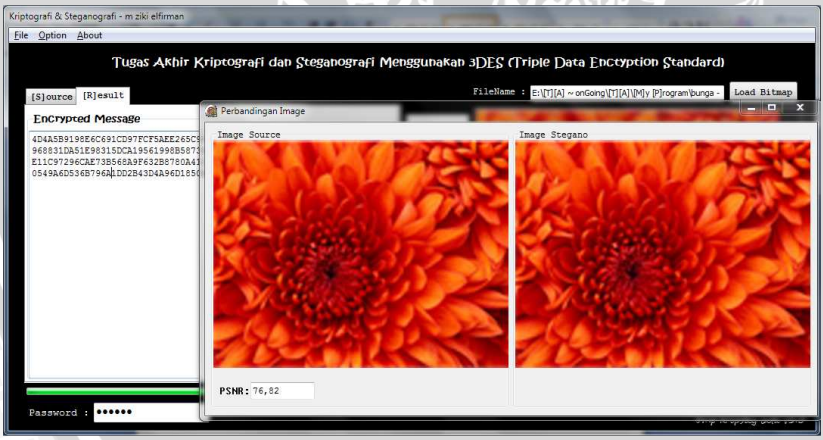
Gambar 4.1 Antarmuka utama

Untuk melakukan proses pengamanan data atau penguraian data *user* harus menginputkan *file* citra, pesan berupa teks dan kunci yang berupa sejumlah karakter. Setelah data – data diinputkan maka untuk melakukan proses penyisipan *user* harus menekan tombol Hide. Proses penyisipan pesan ditunjukkan oleh gambar 4.2.



Gambar 4.2 Antarmuka proses penyisipan pesan

Pada gambar 4.3 merupakan tampilan antarmuka dari hasil proses penyisipan pesan yang berupa *ciphertext* dan citra steganografi.



Gambar 4.3 Antarmuka hasil penyisipan pesan dan perbandingan citra

Sedangkan untuk menguraikan pesan dari citra steganografi *user* harus menginputkan citra steganografi dan *key* sesuai dengan *key* yang digunakan sewaktu menyisipkan pesan ke citra steganografi

yang diinputkan. Pada gambar 4.4 ditunjukkan proses penguraian pesan dari citra steganografi.

Gambar 4.4 Antarmuka proses penguraian pesan

4.4 Implementasi Uji Coba

Pada subbab ini akan dilakukan pembahasan mengenai pengujian yang telah dilakukan pada sistem dan hasil evaluasi dari ringkasan hasil sistem.

4.4.1 Skenario Pengujian

Pengujian terhadap perangkat lunak dibagi menjadi tiga, yaitu pengujian fungsionalitas perangkat lunak, pengujian kinerja perangkat lunak, dan pengujian ketahanan citra steganografi yang dihasilkan oleh perangkat lunak. Dalam pengujian, semua *file* citra mempunyai *format bitmap* dengan kedalaman 24 bit. Penjelasan lebih lengkap mengenai berkas citra yang akan di uji dapat dilihat pada table 4.1.

Tabel 4.1 Daftar Berkas Citra *Bitmap* 24 bit

No	Nama File	Citra	Resolusi
1	Bunga.bmp		40 x 30
2	Coffee.bmp		100 x 128
3	Lena.bmp		512 x 512

Pada penelitian ini ada beberapa teks yang akan disisipkan ke dalam *file* citra. Setiap *file* citra akan di uji menggunakan dua teks yang berbeda. Tabel 4.2 menunjukkan teks yang akan digunakan dalam penelitian kriptografi dan steganografi ini.

Tabel 4.2 Daftar Teks untuk Penelitian Kriptografi dan Steganografi

No	Nama	Pesan
1	file1.txt	nama m ziki elfirman ilmu komputer brawijaya 2004. program kriptografi steganografi yang digunakan menggunakan bahasa pascal.
2	file2.txt	Berkat perkembangan teknologi yang begitu pesat memungkinkan manusia dapat berkomunikasi dan saling bertukar informasi atau data secara jarak jauh. Dengan kemudahan yang ada tuntutan akan sekuritas terhadap kerahasiaan informasi yang saling dipertukarkan tersebut semakin meningkat. Begitu banyak pengguna seperti departemen pertahanan, suatu perusahaan atau bahkan individu – individu tidak ingin informasinya diketahui kompetitornya.

4.5 Hasil Pengujian

Berdasarkan perancangan pengujian yang telah dijelaskan pada bab 3.4, maka hasil pengujian akan dijelaskan menjadi beberapa subbab pengujian, yaitu hasil pengujian fungsionalitas perangkat lunak, hasil pengujian kinerja perangkat lunak dan pengujian ketahanan citra steganografi yang dihasilkan oleh perangkat lunak.

4.5.1 Hasil Pengujian Fungsionalitas Perangkat Lunak

Pengujian fungsi penyisipan dinyatakan berhasil apabila proses penyisipan teks ke dalam *file* citra tidak mengalami pesan kegagalan eksekusi dari perangkat lunak. Pengujian untuk fungsi penguraian kembali teks pesan dinyatakan berhasil jika pesan dapat diperoleh kembali, walaupun teks hasil penguraian belum tentu sama dengan teks pesan yang asli. Pada tabel 4.3 merupakan hasil pengujian fungsionalitas perangkat lunak untuk proses steganografi.

Tabel 4.3 Hasil Pengujian Fungsionalitas Perangkat Lunak Proses Steganografi

No	Proses Steganografi			
	***File* Citra**	**Pesan**	**Penyisipan (*Embedding*)**	**Penguraian (*Extract*)**
1.	Bunga.bmp	File1.txt	Berhasil	Berhasil
2.	Bunga.bmp	File2.txt	Berhasil	Berhasil
3.	Coffee.bmp	File1.txt	Berhasil	Berhasil
4.	Coffee.bmp	File2.txt	Berhasil	Berhasil
5.	Lena.bmp	File1.txt	Berhasil	Berhasil
6.	Lena.bmp	File2.txt	Berhasil	Berhasil

Pengujian fungsi enkripsi pesan dinyatakan berhasil apabila didapatkan *ciphertext* yang berupa karakter *hexadecimal* dan tidak terdapat pesan kegagalan eksekusi dari perangkat lunak. Sedangkan pada pengujian untuk fungsi dekripsi pesan dengan perangkat lunak dinyatakan berhasil jika pesan dapat didekripsi menjadi karakter *ASCII*, walaupun belum tentu sama dengan pesan asli. Pada tabel 4.4

merupakan hasil pengujian fungsionalitas perangkat lunak untuk proses kriptografi.

Tabel 4.4 Hasil Pengujian Fungsionalitas Perangkat Lunak Proses Kriptografi

No	Proses Kriptografi			
	Pesan Asli	***Ciphertext***	***Key* untuk Dekripsi**	**Hasil Dekripsi**
1.	File1.txt	65F8FFA8AD4AD D2368FCB8DD35 8E03B381C95DA 6EAD0C264D969 7BE32A9C76EB2 D477702FA4AFF 6D7B2CB2467323 162993BAB7DB9 0D3C283F12A3F0 7117B3AC72A860 D4C6EE6E28F939 9910DB9B1B3103 8C4DC0BFADC0 6441B3865BB3FC 924B2371CD0CA 28A927C15537FF E67541548CF854 FD35910D3A5B2 6164C92420D690 2	qwerty	nama m ziki elfirman ilmu komputer brawijaya 2004. program kriptografi steganografi yang digunakan menggunakan bahasa pascal.
2.	File1.txt	65F8FFA8AD4AD D2368FCB8DD35 8E03B381C95DA 6EAD0C264D969 7BE32A9C76EB2 D477702FA4AFF 6D7B2CB2467323 162993BAB7DB9 0D3C283F12A3F0 7117B3AC72A860 D4C6EE6E28F939 9910DB9B1B3103 8C4DC0BFADC0	ytrewq	ÆQë¥ñãîd¯$ {õN4v? ñãîdÀKµ1'Ê ˆ×9X£ô !F1½ àZCÝV8Ö/“-¹ V)ƒ8X¡ªpñãîd K í”¥BÚ7Ç ™È;ƒ»Ýèüâò F;2òÛÖ _‘Èñ+ P-sƒV/yi:³ ;ƒ»Ýèüâò F;2ò ÛÖ _‘ ®Ê âÊNÜÚa‹œ

		6441B3865BB3FC924B2371CD0CA28A927C15537FFE67541548CF854FD35910D3A5B26164C92420D6902		
3.	File2.txt	83E2AE16D28546F9AF2E87CDA4DD46487EFA6269D8BF48B0B4D403D6E750AD044526C464A6BC3A899D4C5EFD1B3712BA77CCFCF7C0BAF8EF49FACE370C2D1CB378789B01C9315A73EAA537915556123F698A60069D63BC71393094DDE656ED56B4D09866283B13DE27519DB3B94039C00EB051500D5A3D7F27CC584DEB4BEE886DC5942D70EE4E1E2F684DC8E24972C1618208FEC4252D69FFD7AE0D02C28DB682D554DA0DE56F32F933B3777C2DA61529ED3753E186EF40066E2BD35470306C46C7897EB6ED1279E1A8FA788216350697D8F8E69B03D661608A3E689EAC977F12622CA358BB94531E23686C4C80BB506CA060	qwerty	Berkat perkembangan teknologi yang begitu pesat memungkinkan manusia dapat berkomunikasi dan saling bertukar informasi atau data secara jarak jauh. Dengan kemudahan yang ada tuntutan akan sekuritas terhadap kerahasiaan informasi yang saling dipertukarkan tersebut semakin meningkat. Begitu banyak pengguna seperti departemen pertahanan, suatu perusahaan atau bahkan individu – individu tidak

		F8639288AB9410 CF9A9F4ABD8A 0CF503C98EF0F8 330F10B253CE58 232C0AEE7BDCF 51028330B8CE01 E897323B23330E F001C7C275450E B04CAEBDED1A 4DF42DA2048E2 ED2DD38909EC6 ADC04552A9776 7397CF8EAD32E 486E7347031C0F C85027B548D79A 8A1B7FA604B7D 1C2F61183B3949 6AC023D5CB524 8FC2A32C98F22F 6E6EF82203A24C F08AD168CF6931 60472CFEB1E32 A0798FDB43FC4 83B4D1CF57C867 37079EC1573604 B6542785F95EE4 56D6D391FB0DB 5EDC2333051BA 6A136175F4E517 995		ingin informasinya diketahui kompetitornya.
4.	File2.txt	83E2AE16D28546 F9AF2E87CDA4D D46487EFA6269 D8BF48B0B4D40 3D6E750AD04452 6C464A6BC3A89 9D4C5EFD1B371 2BA77CCFCF7C0 BAF8EF49FACE3 70C2D1CB378789 B01C9315A73EA A537915556123F6 98A60069D63BC7	abcdef	ËJ´¬1□i¹¢¹XÂ-Ôw,³GF `„ Ñgx,XBCô7#F [ÜbCW†ùÌÔy pÊ'w^Ã¸•¥Kêÿ »cÂy □} L™Ê^Æ V„û:-- '^ÝûJ□^:j§

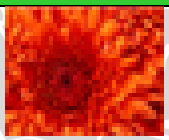
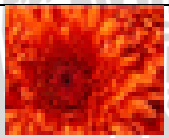
		1393094DDE656E D56B4D09866283 B13DE27519DB3 B94039C00EB051 500D5A3D7F27C C584DEB4BEE88 6DC5942D70EE4 E1E2F684DC8E24 972C1618208FEC 4252D69FFD7AE 0D02C28DB682D 554DA0DE56F32 F933B3777C2DA 61529ED3753E18 6EF40066E2BD35 470306C46C7897 EB6ED1279E1A8 FA788216350697 D8F8E69B03D661 608A3E689EAC9 77F12622CA358B B94531E23686C4 C80BB506CA060 F8639288AB9410 CF9A9F4ABD8A 0CF503C98EF0F8 330F10B253CE58 232C0AEE7BDCF 51028330B8CE01 E897323B23330E F001C7C275450E B04CAEBDED1A 4DF42DA2048E2 ED2DD38909EC6 ADC04552A9776 7397CF8EAD32E 486E7347031C0F C85027B548D79A 8A1B7FA604B7D 1C2F61183B3949 6AC023D5CB524 8FC2A32C98F22F 6E6EF82203A24C		

		F08AD168CF6931 60472CFEB1E32 A0798FDB43FC4 83B4D1CF57C867 37079EC1573604 B6542785F95EE4 56D6D391FB0DB 5EDC2333051BA 6A136175F4E517 995		

4.5.2 Hasil Pengujian Kinerja Perangkat Lunak

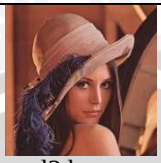
Tabel 4.5 menunjukkan hasil pengujian kinerja perangkat lunak dengan kombinasi pesan dan citra yang berbeda.

Tabel 4.5 Hasil Pengujian Kinerja Perangkat Lunak

No	Pesan	*File* citra asal	Citra Stegano	Hasil Ekstraksi	Nilai PSNR
1.	File1.txt	Bunga.bmp	 b1.bmp	nama m ziki elfirman ilmu komputer brawijaya 2004. program kriptografi steganografi yang digunakan menggunakan bahasa pascal.	57,17
2.	File2.txt	Bunga.bmp	 b2.bmp	Berkat perkembangan teknologi yang begitu pesat memungkinkan manusia dapat berkomunikasi dan saling bertukar informasi atau data secara jarak jauh. Dengan	52,13

				kemudahan yang ada tuntutan akan sekuritas terhadap kerahasiaan informasi yang saling dipertukarkan tersebut semakin meningkat. Begitu banyak pengguna seperti departemen pertahanan, suatu perusahaan atau bahkan individu – individu tidak ingin informasinya diketahui kompetitornya.	
3.	File1.txt		 c1.bmp	nama m ziki elfirman ilmu komputer brawijaya 2004. program kriptografi steganografi yang digunakan menggunakan bahasa pascal.	61,52
4.	File2.txt		 c2.bmp	Berkat perkembangan teknologi yang begitu pesat memungkinkan manusia dapat berkomunikasi dan saling	61,49

				bertukar informasi atau data secara jarak jauh. Dengan kemudahan yang ada tuntutan akan sekuritas terhadap kerahasiaan informasi yang saling dipertukarkan tersebut semakin meningkat. Begitu banyak pengguna seperti departemen pertahanan, suatu perusahaan atau bahkan individu individu tidak ingin informasinya diketahui kompetitornya.	
5.	File1.txt		 l1.bmp	nama m ziki elfirman ilmu komputer brawijaya 2004. program kriptografi steganografi yang digunakan menggunakan bahasa pascal.	79,80

6.	File2.txt		 l2.bmp	Berkat perkembangan teknologi yang begitu pesat memungkinkan manusia dapat berkomunikasi dan saling bertukar informasi atau data secara jarak jauh. Dengan kemudahan yang ada tuntutan akan sekuritas terhadap kerahasiaan informasi yang saling dipertukarkan tersebut semakin meningkat. Begitu banyak pengguna seperti departemen pertahanan, suatu perusahaan atau bahkan individu individu tidak ingin informasinya diketahui kompetitornya.	74,69

Berdasarkan hasil pengujian yang ditampilkan pada tabel 4.5 dapat dijelaskan bahwa setelah citra asli disisipkan teks yang telah ditransformasi menjadi *ciphertext* dalam bentuk biner tidak terjadi penambahan atau pengurangan nilai. Citra sebelum dan sesudah

disisipi pesan memiliki ukuran *file* yang sama dan hasil penguraian pesan sesuai dengan pesan asli.

Jumlah karakter pada pesan berpengaruh pada kualitas. Hal ini dapat dilihat pada tabel 4.5, terjadi penurunan nilai PSNR seiring dengan penambahan karakter pesan. Tingginya nilai PSNR dikarenakan hanya lsb pada citra yang akan dirubah apabila berbeda dengan nilai bit pada *ciphertext*.

Peak Signal Noise Ratio (PSNR) dilakukan untuk mengetahui seberapa besar *Noise* dari citra steganografi dengan membandingkan citra aslinya. Semakin tinggi nilai PSNR maka akan semakin mirip citra steganografi dengan citra aslinya.

4.5.3 Hasil Pengujian Ketahanan Citra Steganografi

Untuk melakukan proses pengujian ketahanan citra steganografi terhadap manipulasi citra maka dipilih citra b1.bmp. Seperti yang sudah dijelaskan pada subbab 3.4.4, manipulasi yang akan dilakukan yaitu *cropping*, penambahan dan pengurangan resolusi citra, pemberian efek sephia, rotasi 90^0 searah jarum jam dan perubahan *format* citra. Tabel 4.6 adalah tabel hasil pengujian ketahanan perangkat lunak.

Tabel 4.6 Pengujian Ketahanan Citra Steganografi

No	Jenis Serangan	*File* Citra *modified*	Hasil Ekstraksi	*Error* (%)
1.	*Cropping*		ûLÇ 7WHJ? ØÖ□vd”YXP` šor Ò§„J§’ h?Ö-¬§)ÿ´mD9¾Pßû¤ào+Ï© ²ü®® §Ö÷4 <µ=VÊøy×¶ MÆ h?Ö-¬§)ÿ´mD9ÇYÇ{œ[¨h? Ö-¬§)ÿ´mD9Üpý •qÄ÷ÆF T†+1°i©□‰¾ h?Ö-¬§)ÿ´mD9®5ƒ´2ëåt]öŠ [ê›MÍ {KõÙ,ôÍÅ Ùç½fšs.à ¼šoÏôL¡²Ÿ`□JyJiˆ/J¤-£Œîâ* õ$ qšBÆ6NqšutÂ=ŠÅ¼(´|Gï†‘Æ9iË1„ê{Ä^g2+›ŠQ □Cª]:ÿ ñþÿ`·¼ÈË¯ÏÒSSƒ	100%

			h?Ö-¬§)ÿ´mD92çŒ’	
2.	*Zoom* 110%		Ã' ð4U□t*™ÊÎ›Yq,Â2Ð®iÆ- ù3 ùwæ’…E LŠdtàî¯7‡)/ïX¡³v -_®.tË”’Ü«hqšeì ‰ fñ”É ´>÷;…!÷Z— (ÞYÍþ Ã-\§5‰â_5ÙL˜Í ¤ ¶F‘ÙT¬— ÝðšÝ«K¦uÏ#2ÕÓýÞ- CjÞn@¨ºÒ	100%
3.	*Zoom 90%*		fG3- >Ÿª©Þ ^1›,x{F 5+]~û– úrW¶¬`Óóo"¥ M*Y„`{ïw(y ráÖ ÊeæÀ6 ">[Ýñ8Ý&ÙÞ'– _æÚ ¦Æª~· Ë¿¼VXÏF1PN;Z· !UhÜwê \]$Ð Lø3G¼×Û² ?¢K šoÏôL¡²Ÿ`□JyJi^/J¤-£Œîâ*% °Ü.Ù ¡f`=ß‹ šoÏôL¡²Ÿ`□JyJi^/J¤-£Œîâ*h? Ö-¬§)ÿ´mD9¶p"'0A/Đ/ Ã°F…W)šzKÌŸ#B=Í,÷íF3œ -ŠþJ,³l 3Aó h?Ö-¬§)ÿ´mD9 œËñP¥ÉØ Y h?Ö-¬§)ÿ´mD9Í¼TNÉ‰}® ·þÍ¿˜- £nê9c)3ñ0{#àîLÊæ†^=ƒ¢µ. gx ¥OÀÊ6Ë¬Àx šoÏôL¡²Ÿ`□JyJi^/J¤-£Œîâ*h? Ö-¬§)ÿ´mD9	100%
4.	Efek sephia		h?Ö-¬§)ÿ´mD9 h?Ö-¬§)ÿ´mD9>íÃØnÞ$öX ÿ³ìU\%ç6?é□œ…W)šzKÌ Ÿ#B=Í ¶êUCãmÖÛ}FØCÏ÷‰ŒA `#| iíÿ ¦ãw‡\cÓ‰{…W)šzKÌŸ #B=Í q	100%

			õÐÚKo‹™ãJÂ«|sÄ ËÉ4N ßiQ` W®S;mT Éz*Ýû¯ … ÈW™å·=.ÌO§'Ìm2_ Áqe9°! °à!æféÉÔá”×Pà³Â¹'ý Ü®’×Z‹'ÛH- áÆçPâÚ ¡ÊÒÙ0s¼ ¾‡WÌ" ¯Jê Åè% TFÞÞBËJTk ®6 _@Ìf¦ t n©-} ^>×-nàÛF RÓ;o×·1 Zû§´»'?òÙÊ8ß,`ï¶ÉõWÜÎå0 Ò[ªõõYcöð®«ùumûŸÔ“}Ðl ÝG~D¾ÐËÎîê «ƒBQ^ù~ŠB`oêìR¶á8 Z|p Ï¡¶¥2÷ÇÍ&Dyqw§À¥å)43¼v @ñæ¥A%rÍ¡` ñ!ÕqìŸb Ó?Âw,Š×ÝÇö l‘u]éWCao ”ïA¡™ÔØÁK|Ô6÷’ÇSvÃç	
5.	Rotasi 90^0		º¦¨»]~lÛå / \/› (A|sÑ‡’2ép¼^)]i7<5ß ç8- œËMû$Ý%ukaf?Á£…W)šz KÌŸ#B=Í]Põ¼¿ÛWâHÔ@is »Æ×q-:Ábw- @¯Ë·ÛÐ%´ Éo(= á7n<õ"?è”aµµ.ú1¯ïPï3ÕD¨‘t ½V×@…Psïªa °Gƒ8ge¥9Ç ~R¬!M¾7¿ WÓƒ ãE7^Ò-_o \<wBüR }ëÄ„¤¸Ñ"sØ°G Í\DêŸErú(R wÖ¯ «òz1g¨fç- V}‰´IúSÄ®ÉŒ=` 2ò±:- oÑÙÒRÇø/©sÎÞ@°k EÂ £Ìa†OØƒÒ¯Ììao…çR8 V¶º•Ì~Eû¬©‘í“Õ£} ëPçiY“ ò®;ß— $:/Óü8oQ{4– kedƒ£«ŸYš+º,Å.Ÿ0ÄzóÝ¤	100%

			÷>5Þï°×îL¼„½å§Î 4{b†(.	
6.	Perubahan *format* ke png dan kembali ke bmp		nama m ziki elfirman ilmu komputer brawijaya 2004. program kriptografi steganografi yang digunakan menggunakan bahasa pascal.	0%
7.	Perubahan *format* ke jpg dan kembali ke bmp		nama m ziki elfirman ilmu komputer brawijaya 2004. program kriptografi steganografi yang digunakan menggunakan bahasa pascal.	0%

Dari tabel 4.6 pengujian ketahanan citra steganografi dapat dijelaskan bahwa manipulasi citra yang umum dilakukan akan menyebabkan kegagalan dalam pendekripsian pesan. Sedangkan perubahan *format* citra dari bmp ke *format* citra lain kemudian kembali ke *format* bmp lagi tidak menyebabkan kegagalan dalam pendekripsian pesan.

4.6 Pembahasan

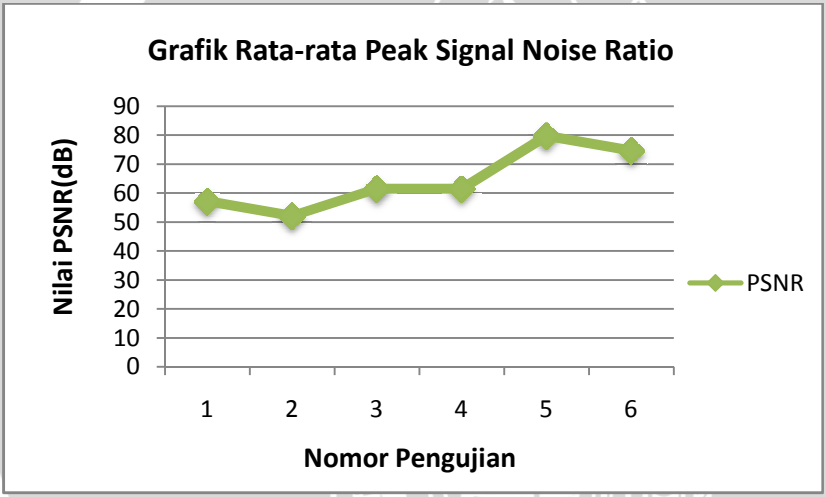
Hasil uji yang didapatkan akan di analisis lebih lanjut untuk melihat apakah hasilnya sesuai dengan maksud dari pengujian yang bersesuaian. Selain itu akan diambil kesimpulan terhadap hasil uji yang didapat.

4.6.1 Analisis Hasil Uji Fungsionalitas Perangkat Lunak

Hasil uji menunjukkan bahwa perangkat lunak pengamanan data kriptografi dan steganografi yang telah dibuat pada penelitian ini telah memenuhi kebutuhan perangkat lunak yang telah dipaparkan pada bab 3. Hal ini dibuktian dengan keberhasilan perangkat lunak dalam melakukan proses enkripsi dan dekripsi pesan, serta proses penyisipan pesan dan penguraian pesan dari citra tanpa mengalami pesan kegagalan ataupun *error* pada saat eksekusi perangkat lunak. Walaupun pada kasus tertentu beberapa karakter pada teks hasil dekripsi berbeda dengan teks asli.

4.6.2 Analisis Hasil Uji Kinerja Perangkat Lunak

Dalam analisis hasil uji kinerja perangkat lunak dapat disimpulkan bahwa ekstraksi pesan dari citra steganografi yang dihasilkan oleh perangkat lunak cukup baik, hal ini dibuktikan dengan nilai PSNR yang cukup tinggi dimana semakin tinggi nilai PSNR maka semakin mirip citra steganografi dengan citra aslinya. Tingginya nilai PSNR juga dipengaruhi oleh jumlah kapasitas yang dapat digunakan untuk menampung pesan pada citra. Semakin besar jumlah bit yang tidak terpakai untuk menampung pesan, maka semakin tinggi nilai PSNR. Pada gambar 4.5 ditunjukkan terjadinya peningkatan nilai PSNR seiring dengan peningkatan kapasitas citra untuk menampung pesan.



Gambar 4.5 Grafik rata – rata uji kinerja perangkat lunak

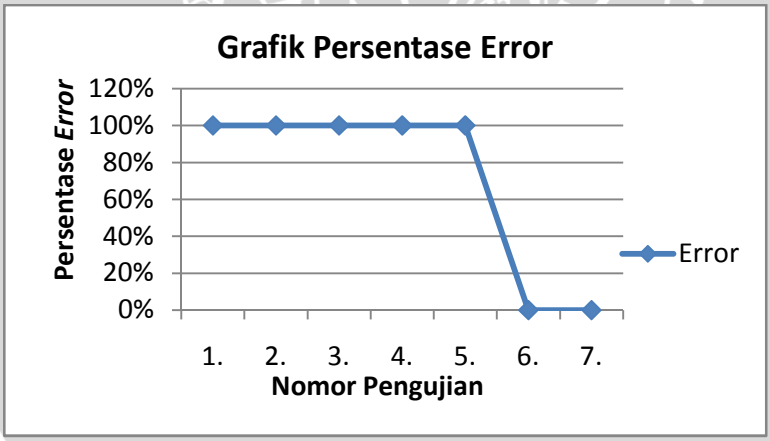
Grafik rata-rata *Peak Signal Noise Ratio* (PSNR) pada gambar 4.5 menunjukkan bahwa tingkat noise tertinggi terdapat pada nomor pengujian kedua yaitu pada citra bunga.bmp dengan pesan file2.txt. Hal ini dikarenakan hampir semua lsb pada *file* citra digunakan sehingga terjadi perubahan nilai pada sebagian besar lsb citra.

Komposisi warna pada citra tidak berpengaruh terhadap nilai PSNR karena pesan akan dirubah ke dalam bentuk biner dan

melewati proses penyandian bit – bit tersebut akan teracak. Apabila terdapat kasus bit – bit *ciphertext* identik dengan bit – bit lsb yang bersesuaian maka dalam kasus ini citra steganografi akan identik dengan citra asli.

4.6.3 Analisis Hasil Uji Ketahanan Citra Steganografi

Dari hasil uji ketahanan dapat disimpulkan bahwa, citra steganografi hanya memiliki ketahanan terhadap perubahan *format* sementara ke *format* lain, setelah *format* dirubah dari bmp ke *format* lain kemudian akan di kembalikan ke *format* bmp lagi. Hal ini dapat dilihat dari kesamaan pesan hasil penguraian dengan pesan asli. Sedangkan untuk manipulasi citra yang umum dilakukan yaitu *cropping*, penambahan dan pengurangan resolusi citra, pemberian efek sephia dan rotasi, perangkat lunak tidak dapat mengembalikan pesan secara utuh dan benar. Hal ini dikarenakan berubahnya sususan dan urutan dari bit – bit *ciphertext* yang disisipkan pada bit terakhir citra steganografi. Grafik rata – rata mengenai pengujian ketahanan citra steganografi dapat dilihat pada gambar 4.6.



Gambar 4.6 Grafik Rata – rata uji ketahanan citra steganografi

4.6.4 Analisis Umum Hasil Uji

Perangkat lunak dinilai telah bekerja relatif baik. Kinerja yang yang dihasilkan perangkat lunak cukup memuaskan meskipun

pada kasus tertentu terjadi perbedaan beberapa karakter pada pesan hasil penguraian dengan pesan yang asli dan citra steganografi tidak tahan terhadap manipulasi citra pada umumnya.

Tingginya nilai PSNR dipengaruhi oleh jumlah lsb yang tidak digunakan dan bit – bit *ciphertext* yang dihasilkan. Semakin banyak jumlah lsb yang tidak digunakan dan semakin banyak kesamaan bit *ciphertext* dengan bit lsb maka nilai PSNR akan semakin besar.

BAB V
KESIMPULAN DAN SARAN

Bagian ini berisi kesimpulan dari seluruh pengerjaan skripsi yang telah dilakukan dan juga saran untuk pengembangan lebih lanjut.

5.1 Kesimpulan

Setelah melakukan skripsi ini maka disimpulkan :

1. Teknik pengamanan data kriptografi dan steganografi dapat diimplementasikan pada sebuah perangkat lunak secara bersamaan. Teknik kriptografi algoritma 3DES diproses terlebih dahulu sehingga menghasilkan *ciphertext*, *ciphertext* kemudian disisipkan ke dalam citra penampung dengan teknik steganografi LSB-*insertion.*
2. Kualitas citra yang dihasilkan relatif sama dengan citra aslinya secara kasat mata, hal ini ditunjukkan oleh grafik rata – rata PSNR yang memiliki nilai yang relatif tinggi. Faktor yang mempengaruhi kualitas citra yang dihasilkan adalah persentase jumlah bit yang berubah, semakin sedikit bit yang dirubah maka semakin mirip dengan citra aslinya.
3. Manipulasi citra yang umum dilakukan yaitu *cropping,* penambahan dan pengurangan resolusi citra, pemberian efek sephia dan rotasi akan membuat pesan tidak dapat diuraikan, sedangkan perubahan format citra ke format lain kemudian kembali ke format bmp tidak mempengaruhi pesan pada citra.

5.2 Saran

Untuk pengembangan lanjut perangkat lunak maka ada beberapa saran yang dapat diberikan :

1. Untuk meningkatkan keamanan data proses pembuatan tiga kunci internal dapat digunakan metode enkripsi seperti MD5, AES atau metode lainnya yang mempunyai keamanan yang tinggi.
2. Sebelum disisipkan ke dalam citra, kunci untuk 3DES dienkripsi terlebih dahulu.

3. Untuk pengembangan selanjutnya dapat digunakan variasi metode steganografi agar tahan terhadap manipulasi citra.

DAFTAR PUSTAKA

Barker, William C. 2004. *Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher*. NIST, Gaithersburg.

Grabbe, J. Orlin. *The DES Algorithm Illustrated*.
url : http://orlingrabbe.com/des.htm
tanggal akses : 04 Mei 2010

Hakim, M. 2007. Skripsi : *Studi dan Implementasi Steganografi Metode LSB dengan Preprocessing Kompresi data dan Ekspansi Wadah*. Institut Teknologi Bandung, Bandung.

Hidayat, A. 2008. Skripsi : *Enkripsi dan Dekripsi Data Dengan Algoritma 3DES (Triple Data Encryption Standard)*. Universitas Padjajaran, Jatinangor.

Laksono, S. 2009. Skripsi : *Kompresi Citra Digital Menggunakan Fuzzy Learning Vector Quantization*. Universitas Brawijaya, Malang.

Menezes, A. J., Paul Van Oorschot, dan Scott A. Vanstone. 1996. *Handbook of Applied Cryptography*. CRC Press, Ontario.

Mohanty, S. P. 1999. *Digital Watermarking : A Tutorial Review*. University of South Florida, Florida.

Morkel, T., JHP. Eloff, dan MS. Olivier. 2005. *An Overview Of Image Steganography*. Pretoria : Information and Computer Security Architecture (ICSA) Research Group, Department of Computer Science, University of Pretoria.

Munir, R. 2004. Kuliah IF5054 Kriptografi : *Pengantar Kriptografi*. Institut Teknologi Bandung. Bandung.

Munir, R. 2004. Kuliah IF5054 Kriptografi : *Steganografi dan Watermarking*. Institut Teknologi Bandung. Bandung.

Schneier, B. 1996. *Applied Cryptography, Second Edition: Protocols, Algorthms, and Source Code in C*. John Wiley & Sons, Inc.