IMPLEMENTASI STEGANOGRAFI DENGAN METODE PENYISIPAN LSB (*LEAST SIGNIFICANT BIT*) DAN *IMAGE SHUFFLING*

SKRIPSI

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:
TEGUH PIDEKSO ADHINOTO
0610630106-63

KEMENTERIAN PENDIDIKAN NASIONAL
UNIVERSITAS BRAWIJAYA
FAKULTAS TEKNIK
MALANG
2011

LEMBAR PERSETUJUAN

IMPLEMENTASI STEGANOGRAFI DENGAN METODE PENYISIPAN LSB (*LEAST SIGNIFICANT BIT*) DAN *IMAGE SHUFFLING*

SKRIPSI

JURUSAN TEKNIK ELEKTRO

Diajukan Untuk Memenuhi Sebagian Persyaratan

Memperoleh Gelar Sarjana Teknik

Oleh:

TEGUH PIDEKSO ADHINOTO

0610630106-63

Telah mengetahui dan disetujui oleh:

Dosen Pembimbing I	**Dosen Pembimbing II**
Waru Djuriatno, ST., MT.	**Ir. Muhammad Aswin, MT.**
NIP. 19690725 199702 1 001	**NIP. 19650626 199002 1 001**

LEMBAR PENGESAHAN

IMPLEMENTASI STEGANOGRAFI DENGAN METODE PENYISIPAN LSB (*LEAST SIGNIFICANT BIT*) DAN *IMAGE SHUFFLING*

SKRIPSI
JURUSAN TEKNIK ELEKTRO

Diajukan Untuk Memenuhi Persyaratan
Memperoleh Gelar Sarjana Teknik

Disusun oleh:
TEGUH PIDEKSO ADHINOTO
NIM. 0610630106 - 63

Skripsi ini telah diuji dan dinyatakan lulus pada
Tanggal 23 Desember 2011

DOSEN PENGUJI

R. Arief Setyawan, ST., MT.
NIP.19750819 199903 1 001

Ali Mustofa, ST., MT.
NIP.19710601 200003 1 001

Adharul Muttaqin, ST ., MT.
NIP. 19760121 200501 1 001

Mengetahui
Ketua Jurusan Teknik Elektro

Dr. Ir. Sholeh Hadi Pramono, MS.
NIP. 19580728 198701 1 001

KATA PENGANTAR

Puji syukur kepada Allah Tuhan Yang Maha Esa yang telah memberikan segala nikmat, rahmat, serta hidayah-Nya sehingga penyusunan skripsi ini dapat diselesaikan. Karena hanya dengan pertolongan-Nya semata penulis mampu melewati segala kendala yang ditemui selama penyusunan skripsi ini.

Sholawat serta salam tetap tercurah kepada Nabi Muhammad Rasulullah SAW atas segenap perjuangan dalam memperjuangkan agama Islam yang benar ini sehingga menjumpai umat di akhir zaman ini.

Skripsi berjudul "*Implementasi Steganografi Dengan Metode Penyisipan LSB* (*Least Significant Bit*) *Dan Image Shuffling*" ini disusun sebagai salah satu syarat untuk mendapatkan gelar Sarjana Teknik di Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya. Terselesaikannya skripsi ini tentunya tidak lepas juga dari bantuan berbagai pihak. Oleh sebab itu, dengan segala kerendahan hati penulis menyampaikan terima kasih kepada:

1. Kedua orangtua tersayang, Ayahanda H. Nurhadi MH, ST., dan Ibunda Dra. Hj. Aryanti Kardi, atas segenap usaha, cinta, do'a dan segalanya selama ini, esok, dan seterusnya.
2. Adik-adikku tercinta, Gigih Yudhanto Purbonoto, AMd., (alm.) Luluk Prasasti Yudhaningtyas, Tuhu Uboyo Wicaksono, dan Luhur Pambudi Wijayanto, atas segala do'a dan dukungannya.
3. Seluruh keluarga penulis yang senantiasa turut memberikan do'a, nasihat dan motivasi yang tulus dalam menyelesaikan skripsi ini.
4. Bapak Dr. Ir. Sholeh Hadi Pramono, MS., selaku Ketua Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya.
5. Bapak M. Azis Muslim, ST., MT., Ph.D., selaku Sekretaris Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya.
6. Bapak Mochammad Rif'an, ST., MT., selaku Plt. Ketua Program Studi Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya.
7. Bapak Waru Djuriatno, ST., MT., selaku KKDK Rekayasa Komputer Teknik Elektro sekaligus sebagai Dosen Pembimbing I yang telah memberikan arahan dalam pengambilan judul serta pengerjaan skripsi ini.

8. Bapak Ir. Muhammad Aswin, MT., selaku Dosen Pembimbing II atas bantuan serta bimbingannya selama ini.
9. Bapak Ir. Hari Santoso, MT., selaku dosen pembimbing akademik atas nasehatnya selama menempuh masa perkuliahan.
10. Bapak dan Ibu Dosen serta karyawan jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya atas kesediannya berbagi ilmu.
11. Seluruh pengurus Griya Mahasiswa Nusantara, Kelompok Klaseman, Desa Klaseman, Daerah Malang Tengah, Malang Raya, Pembina GARIS, Pembina PPM Malang Raya, dan Pembina HIMALAYA atas bimbingan dan peramutan kepada saya selama ini.
12. Mas Akhmad Zainuri, ST., dan Kuncoro Adi Pradono, ST., sebagai guru, kakak, dan sahabat yang telah mengajari banyak hal.
13. Mas Samsul Anam, STP., dan Marissa Rizqil Haque, atas dukungannya, terutama di sabil dan saat-saat akhir perkuliahan.
14. Yuniar Adi Setiawan, teman sekamar yang telah rela berbagi banyak hal.
15. Seluruh saudara-saudaraku di Griya Mahasiswa Nusantara, Kelompok Klaseman, Desa Klaseman, Daerah Malang Tengah, Malang Raya, FP, GARIS, PPM Malang Raya, dan HIMALAYA atas makna yang diberikan dalam kehidupan di Kota Malang ini.
16. Teman-teman Asisten Laboratorium Informatika dan Komputer Teknik Elektro Universitas Brawijaya yang setia menemani.
17. Teman-teman Ge-Force 2006
18. Semua pihak yang tidak dapat saya sebutkan satu persatu, atas segenap dukungan dan bantuannya.

Penulis menyadari sepenuhnya bahwa skripsi ini masih banyak kekurangan. Segala kritik dan saran yang membangun akan penulis terima demi kesempurnaan skripsi ini. Harapan penulis semoga skripsi ini bermanfaat bagi seluruh pembaca.

Malang, Desember 2011

Penyusun

IMPLEMENTASI STEGANOGRAFI DENGAN METODE PENYISIPAN LSB (LEAST SIGNIFICANT BIT) DAN IMAGE SHUFFLING

ABSTRAK

Steganografi merupakan seni penyembunyian pesan ke dalam pesan lainnya. Teknik ini meliputi banyak sekali metode komunikasi untuk menyembunyikan pesan rahasia. Steganografi pada media digital berupa file citra digunakan untuk mengeksploitasi keterbatasan sistem penglihatan manusia dengan cara menurunkan kualitas warna pada file citra yang belum disisipi pesan rahasia. Sehingga dengan keterbatasan tersebut manusia sulit menemukan gradasi penurunan kualitas warna pada file citra yang telah disisipi pesan rahasia.

Pada tugas ahir ini dikembangkan sebuah aplikasi menggunakan algoritma penyisipan LSB dan metode image shuffling untuk melakukan pengacakan terhadap file citra terlebih dahulu sebelum penyisipan pesan ke dalam file citra dilakukan.

Pada proses pengujian diketahui bahwa perangkat lunak yang mengimplementasikan steganografi dengan metode penyisipan LSB dan image shuffling telah berhasil dibangun. Kemudian metode penyisipan LSB yang dikombinasikan dengan image shuffling terbukti lebih tahan terhadap steganalisis. Sedangkan ukuran file pesan rahasia dan jumlah LSB yang digunakan berbanding terbalik dengan mutu dan kesamaran citra pembawa. Berdasarkan pengujian perangkat lunak yang dilakukan, diketahui pula bahwa berkas pesan rahasia tidak mengalami perubahan kualitas setelah melalui proses penyisipan dan ekstraksi.

Kata Kunci: steganografi, citra, S-Box, image shuffling

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1. Latar Belakang

Internet saat ini telah menjadi bagian yang sangat penting bagi infrastruktur komunikasi di dunia. Pertukaran informasi melalui internet memiliki banyak kelebihan dibandingkan dengan media komunikasi lainnya, terutama dari segi kecepatan. Namun informasi yang dikirimkan melalui internet tidak dapat dijamin keamanannya. Penyadapan terhadap informasi rahasia sering terjadi pada media komunikasi ini. Saluran yang digunakan internet pada umumnya bukan merupakan saluran yang aman. Saluran yang aman sebenarnya tersedia, namun kecepatan koneksi menggunakan saluran yang aman ini cenderung lambat.

Salah satu usaha untuk menangani masalah keamanan data rahasia yang dikirimkan melalui internet adalah menggunakan teknik kriptografi dan steganografi. Steganografi adalah teknik menyembunyikan data rahasia ke dalam data lainnya sehingga lawan tidak menyadari keberadaan data rahasia tersebut. Dalam implementasinya, steganografi menggunakan suatu media sebagai pembawa. Salah satu media yang kerap digunakan adalah citra digital.

Terdapat beberapa parameter yang menjadi tolak ukur kualitas steganografi, yaitu *fidelity* (mutu media pembawa tidak berubah secara signifikan), *recovery* (data dapat diungkap kembali), dan *imperceptibility* (keberadaan pesan tidak terdeteksi). Khusus untuk parameter terakhir, pada tugas akhir kali disamping menggunakan metode penyisipan LSB (*Least Significant Bit*), digunakan pula *image shuffling* untuk memperkuat *imperceptibility* serta sebagai bentuk pertahanan dari *steganalysis*.

1.2. Rumusan Masalah

Dari latar belakang yang telah diutarakan sebelumnya, rumusan masalah yang akan diselesaikan pada tugas akhir ini adalah sebagai berikut:

1. Bagaimana cara melindungi data dengan keamanan yang baik
2. Bagaimana cara melakukan penyisipan data tanpa menimbulkan perubahan pada citra pembawa yang terdeteksi indra manusia
3. Bagaimana cara mengungkapkan kembali data yang disisipkan tanpa kesalahan yang berarti

1.3. Tujuan

Tujuan utama dari tugas akhir ini adalah untuk merancang dan mengembangkan sebuah aplikasi metode steganografi yang memiliki tingkat keamanan tinggi sekaligus daya *recovery* yang baik untuk mencegah tindakan pencurian data oleh pihak-pihak yang tidak berkepentingan dalam proses pengiriman data.

1.4. Batasan Masalah

Dalam upaya mengatasi permasalahan yang terdapat pada bagian rumusan masalah, perlu batasan yang digunakan adalah sebagai berikut:

1. Metode yang digunakan adalah steganografi LSB (*Least Significant Bit*) untuk penyisipan data.
2. Menggunakan citra digital dengan format bitmap.
3. Bahasa pemrograman yang digunakan adalah Borland Delphi 7.
4. Diasumsikan tidak ada perubahan data sekecil apapun selama proses transmisi.

1.5. Manfaat

Manfaat yang dapat diperoleh oleh penyusun melalui pengerjaan tugas akhir ini adalah:

1. Penyusun dapat mengaplikasikan ilmu yang telah didapatkan selama menempuh perkuliahan di Jurusan Teknik Elektro Universitas Brawijaya.
2. Penyusun memahami metode beserta algoritma steganografi citra digital.

3. Penyusun mampu mengimplementasikan metode LSB (*Least Significant Bit*) dan *Image Shuffling* untuk meningkatkan keamanan data.

1.6. SISTEMATIKA PENULISAN

Sistematika penulisan dalam laporan tugas akhir ini adalah sebagai berikut:

BAB I Pendahuluan

Memuat latar belakang, rumusan masalah, tujuan, batasan masalah, manfaat dan sistematika penulisan.

BAB II Dasar Teori

Membahas teori dasar dan teori penunjang yang berkaitan dengan algoritma yang digunakan untuk proses penyembunyian data pada citra digital.

BAB III Metodologi Penenelitian

Membahas metode yang digunakan dalam penulisan yang terdiri dari studi literatur, perancangan perangkat lunak, implementasi perangkat lunak, pengujian perangkat lunak, serta pengambilan kesimpulan dan saran.

BAB IV Perancangan dan Implementasi

Berisi perencanaan pembuatan keseluruhan sistem dan segmen-segmen program yang ada dalam aplikasi yang dibuat.

BAB V Pengujian dan Analisis

Menjelaskan langkah-langkah pengujian dan pembahasan hasil pengujian terhadap aplikasi yang dibuat.

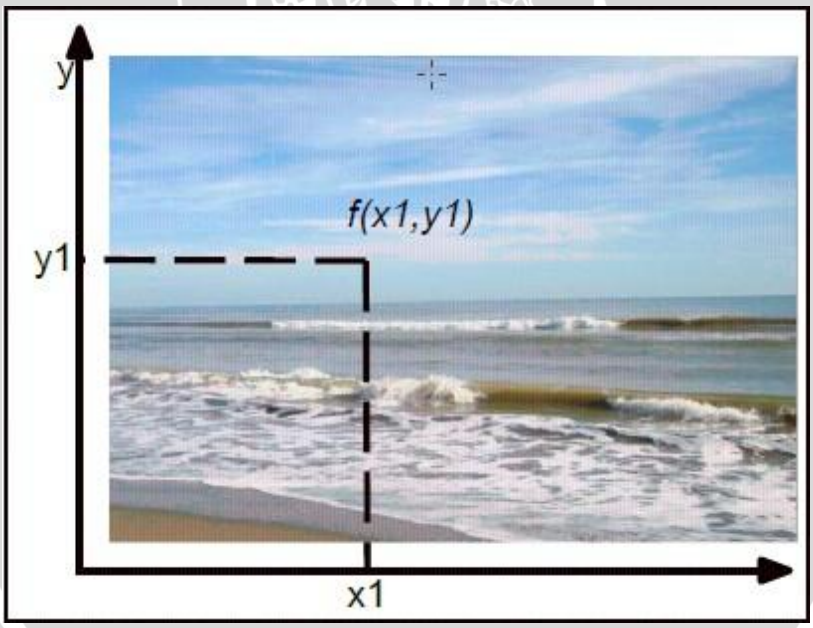
BAB VI Penutup

Memuat kesimpulan yang diperoleh dari pembuatan dan pengujian perangkat lunak, serta saran–saran untuk pengembangan lebih lanjut.

BAB II
DASAR TEORI

2.1. *Digital Image Processing*

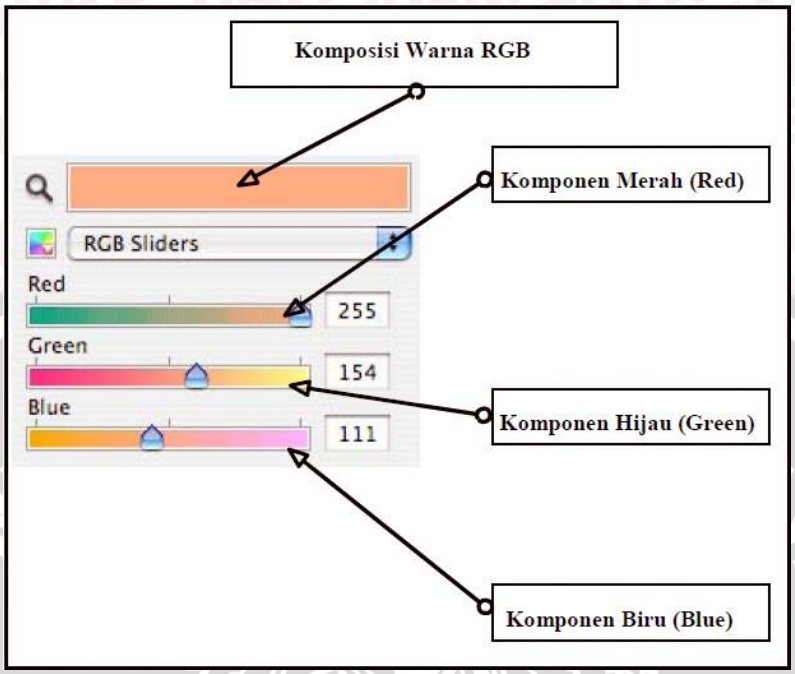
Citra digital dapat didefinisikan sebagai fungsi dua variabel, *f(x,y)*, dimana *x* dan *y* adalah koordinat spasial dan nilai *f(x,y)* adalah intensitas citra pada koordinat tersebut, hal tersebut diilustrasikan pada Gambar 2.1. Teknologi dasar untuk menciptakan dan menampilkan warna pada citra digital berdasarkan pada penelitian bahwa sebuah warna merupakan kombinasi dari tiga warna dasar, yaitu merah, hijau dan biru (*Red, Green, Blue* - RGB). Komposisi warna RGB tersebut dapat dijelaskan pada Gambar 2.1.



Gambar 2.1. Citra Digital

Sebuah citra diubah ke bentuk digital agar dapat disimpan dalam memori komputer atau media lain. Proses mengubah citra ke bentuk digital bisa dilakukan dengan beberapa perangkat, misalnya *scanner*, kamera digital dan *handycam*. Ketika sebuah citra sudah diubah ke dalam bentuk digital

(selanjutnya disebut citra digital), bermacam-macam proses pengolahan citra dapat diperlakukan terhadap citra tersebut.



Gambar 2.2. Komposisi Warna RGB

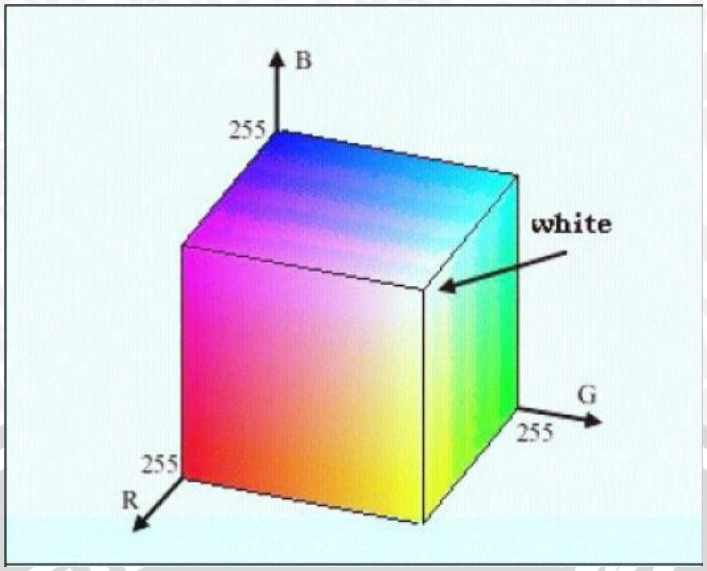
2.2. Berkas Citra Bitmap

Kriteria yang paling penting dari citra ini adalah kedalaman warna yaitu berapa banyak bit per pixel yang didefinisikan dari sebuah warna (Rinaldi Munir, 2005).

Bitmap dengan mengikuti kriteria tadi maka dapat dilihat:

a. 8 bit = 256 warna (256 gray scales).

b. 24 bit = 16.777.216 warna



Gambar 2.3. Warna Bitmap

Secara umum dapat dikatakan semakin banyaknya warna, maka akan diperlukan keamanan yang tinggi dikarenakan bitmap memiliki area yang sangat luas dalam sebuah warna yang seharusnya dihindari. Dilihat dari kedalaman atau kejelasan dari sebuah warna, bitmap dapat mengambil sejumlah data tersembunyi dengan perbandingan sebagai berikut (ukuran ratio dari bitmap dalam byte = ukuran dari data yang disembunyikan) :

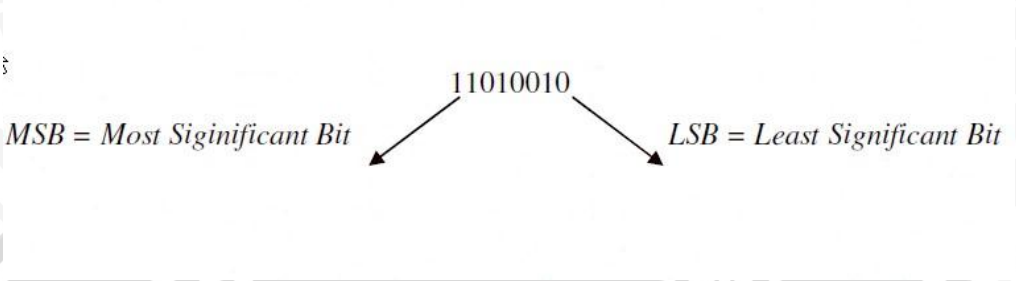
1. 8 bit = 256 warna : 8 : 1
2. 24 bit = 16.777.216 warna : 8 : 1

Perbandingan tersebut diperoleh dari penentuan LSB dalam suatu byte, untuk citra 8 bit letak LSB adalah pada bit terakhir sedangkan untuk citra 24 bit letak LSB adalah pada bit ke-8, bit ke-16 dan bit ke 24 dimana masing-masing byte mewakili warna merah (*red*), warna hijau (*green*) dan warna biru (*blue*).

Manipulasi pada bitmap tidak dapat diubah ke dalam bentuk format grafik yang lain karena data yang tersembunyi dalam file tersebut akan hilang. Mengurangi ukuran dari *carrier file* sangatlah penting untuk melakukan transmisi online, yaitu dengan menggunakan utilitas kompresi (seperti : ARZ, LZH, PKZIP, WinZip), dikarenakan kerja mereka tidak terlalu berat.

2.3. Penyisipan LSB (*Least Significant Bit Insertion*)

Pada susunan bit di dalam sebuah *byte* (1 *byte* = 8 bit), ada bit yang paling berarti (*most significant bit* atau *MSB*) dan bit yang paling kurang berarti (*least significant bit* atau *LSB*). Berikut contoh sebuah susunan bit pada sebuah *byte*:



Gambar 2.4. MSB-LSB

Bit yang cocok untuk diganti adalah bit *LSB*, sebab perubahan tersebut hanya mengubah nilai *byte* satu lebih tinggi atau satu lebih rendah dari nilai sebelumnya. Misalkan *byte* tersebut menyatakan warna merah, maka perubahan satu bit LSB tidak mengubah warna merah tersebut secara berarti. Lagi pula, mata manusia tidak dapat membedakan perubahan yang kecil.

Misalkan segmen data citra sebelum perubahan:

00110011	10100010	11100010	10101011	00100110
10010110	11001001	11111001	10001000	10100011

Segmen data citra setelah pesan ‘1110010111‘ disembunyikan:

00110011	10100011	11100011	10101010	00100110
10010111	11001000	11111001	10001001	10100011

Untuk memperkuat teknik penyembunyian data, bit-bit data rahasia tidak digunakan mengganti *byte-byte* yang berurutan, namun dipilih susunan *byte* secara acak. Misalnya jika terdapat 50 *byte* dan 6 bit data yang akan disembunyikan, maka maka *byte* yang diganti bit *LSB*-nya dipilih secara acak, misalkan *byte* nomor 36, 5, 21, 10, 18, 49.

41	42	43	44	45	46	47	48	49	50
31	32	33	34	35	36	37	38	39	40
21	22	23	24	25	26	27	28	29	30
11	12	13	14	15	16	17	18	19	20
1	2	3	4	5	6	7	8	9	10

Gambar 2.5. Proses Penempatan Bit Pesan

Untuk membangkitkan bilangan acak maka digunakan algoritma pembangkit bilangan acak semu (*pseudo-random number generator*).

$$X_{n+1} = (aX + c) \bmod p$$

Dimana:

Xn+1 : bilangan acak yang dihasilkan

p : jumlah pixel dikali 3 (tiga), yaitu red, green dan blue

a : pengali *(multiplier)*

c : penambah *(increment)*

X_0 : nilai awal *(seed or start value)*

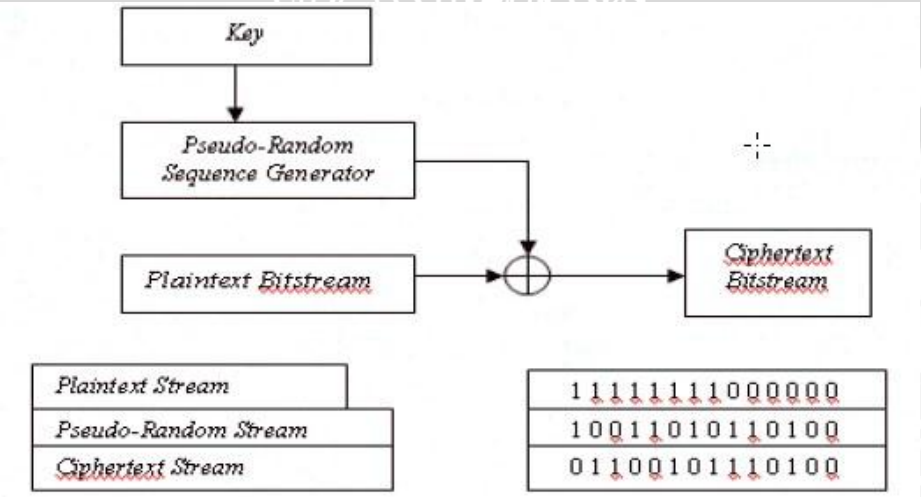
Ukuran data yang akan disembunyikan bergantung pada ukuran citra penampung. Pada citra 24-bit yang berukuran 256 x 256 *pixel* terdapat 65536 *pixel*, setiap *pixel* berukuran 3 *byte* (komponen *RGB*), berarti seluruhnya ada 65536 x 3 = 196608 *byte*. Karena setiap *byte* hanya bisa menyembunyikan satu bit di *LSB*-nya, maka ukuran data yang akan disembunyikan di dalam citra maksimum 196608/8 = 24576 *byte* Ukuran data ini harus dikurangi dengan panjang nama berkas, karena penyembunyian data rahasia tidak hanya menyembunyikan isi data tersebut, tetapi juga nama berkasnya.

Untuk memperkuat keamanan, data yang akan disembunyikan dapat dienkripsi terlebih dahulu. Sedangkan untuk memperkecil ukuran data, data dimampatkan sebelum disembunyikan. Bahkan, pemampatan dan enkripsi dapat juga dikombinasikan sebelum melakukan penyembunyian data.

2.4. *S-Box* pada Algoritma Enkripsi RC4 (*Ron's Chiper #4*)

RC4 merupakan salah satu jenis *stream cipher*, yaitu memproses unit atau input data pada satu saat. Unit atau data pada umumnya sebuah byte atau bahkan kadang-kadang bit (byte dalam hal RC4). Dengan cara ini enkripsi atau dekripsi dapat dilaksanakan pada panjang yang variabel. Algoritma ini tidak harus menunggu sejumlah input data tertentu sebelum diproses, atau menambahkan byte tambahan untuk mengenkrip. Contoh stream cipher adalah RC4, Seal, A5, Oryx dan lain-lain. Tipe lainnya adalah block cipher yang memproses sekaligus sejumlah tertentu data (biasanya 64 bit atau 128 bit blok).

RC4 merupakan *enkripsi stream simetrik proprietary* yang dibuat oleh RSA Data Security, Inc (sekarang *RSA Security*) pada 1987. Penyebarannya diawali dari sebuah source code yang diyakini sebagai RC4 dan dipublikasikan secara anonim pada tahun 1994. Algoritma yang dipublikasikan ini sangat identik dengan implementasi RC4 pada produk resmi. RC4 digunakan secara luas pada beberapa aplikasi dan umumnya dinyatakan sangat aman. Sampai saat ini diketahui tidak ada yang dapat memecahkan/membongkarnya, hanya saja versi ekspor 40 bitnya dapat dibongkar dengan cara *brute force* (mencoba semua kunci yang mungkin). RC4 tidak dipatenkan oleh RSADSI, hanya saja tidak diperdagangkan secara bebas (*trade secret*).

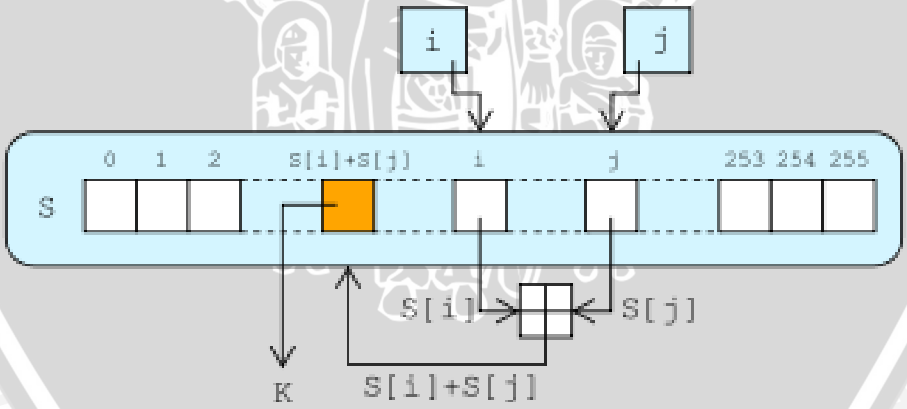


Gambar 2.6. Blok Diagram Algoritma RC4

Algoritma RC4 cukup mudah untuk dijelaskan. RC4 mempunyai sebuah SBox, S0,S1,...,S255, yang berisi permutasi dari bilangan 0 sampai 255, dan permutasi merupakan fungsi dari kunci dengan panjang yang variabel. Terdapat dua indeks yaitu i dan j, yang diinisialisasi dengan bilangan nol. Untuk menghasilkan random byte langkahnya adalah sebagai berikut :

i = (i + 1) mod 256
j = (j + Si) mod 256
swap Si dan Sj
t = (Si + Sj) mod 256
K = St

Byte K di XOR dengan *plainteks* untuk menghasilkan *cipherteks* atau di XOR dengan *cipherteks* untuk menghasilkan *plainteks*. Enkripsi sangat cepat kurang lebih 10 kali lebih cepat dari DES.



Gambar 2.7. Proses Pembangkitan Bilangan Acak pada RC4

Inisialisasi S-Box juga sangat mudah. Pertama isi S0 = 0, S1 = 1,...,S255 = 255. Kemudian isi array 256 byte lainnya dengan kunci yang diulangi sampai seluruh array K0, K1,...,K255 terisi seluruhnya. Set indeks j dengan nol, Kemudian lakukan langkah berikut :

for i = 0 to 255

j = (j + Si + Ki) mod 256

swap Si dan Sj

Salah satu kelemahan dari RC4 adalah terlalu tingginya kemungkinan terjadi tabel S-box yang sama, hal ini terjadi karena kunci *user* diulang-ulang untuk mengisi 256 bytes, sehingga 'aaa' dan 'aaaaa' akan menghasilkan permutasi yang sama. Untuk mengatasi ini maka pada implementasi dapat digunakan hasil hash MD5 dari *password* yang dimasukkan untuk mencegah hal ini terjadi. (Budi Sukmawan: 1998)

2.5. *Image Shuffling*

Image shuffling (pengacakan gambar) dilakukan untuk meningkatkan keamanan pada metode steganografi dengan mencegah tindakan *brute force attack* pada citra digital yang telah diisi data. Dengan menggunakan suatu perhitungan pada *secret key* maka akan terbentuk suatu *S-Box* yang akan digunakan sebagai dasar untuk melakukan pengacakan gambar dan menyusun kembali gambar yang telah teracak.

Gambar 2.8. Pengacakan Gambar Menggunakan *S-Box*

2.6. Steganografi

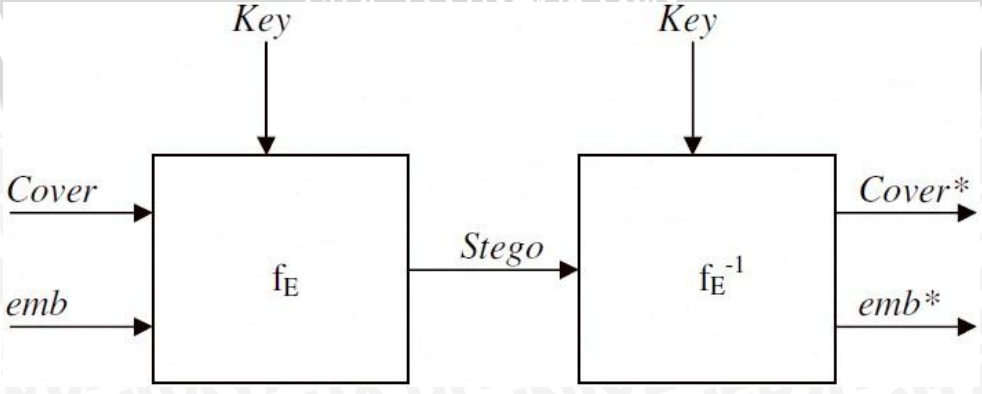
Steganografi merupakan seni penyembunyian pesan ke dalam pesan lainnya sedemikian rupa sehingga orang lain tidak menyadari ada sesuatu di dalam pesan tersebut. Kata steganografi (*steganography*) berasal dari bahasa Yunani yaitu steganos yang artinya tersembunyi atau terselubung dan graphein, yang artinya menulis, sehingga kurang lebih artinya adalah

"menulis tulisan yang tersembunyi atau terselubung" (Johnson, 1998). Teknik ini meliputi banyak sekali metoda komunikasi untuk menyembunyikan pesan rahasia. Metoda ini termasuk tinta yang tidak tampak, microdots, pengaturan kata, tanda tangan digital, jalur tersembunyi dan komunikasi spektrum lebar.

Pada komputer, gambar yang tampil di layar monitor merupakan kumpulan array yang merepresentasikan intensitas cahaya yang bervariasi pada pixel. Pixel adalah titik di layar monitor yang dapat diatur untuk menampilkan warna tertentu. Pixel disusun di layar monitor dalam susunan baris dan kolom. Susunan pixel dalam baris dan kolom ini yang dinamakan resolusi monitor.

Resolusi monitor yang sering dijumpai adalah 640x480, 800x600, 1024x768. Resolusi 640x480 akan menampilkan pixel sejumlah 640 baris dan 480 kolom. Melalui pixel inilah suatu gambar dapat dimanipulasi untuk menyimpan informasi yang akan digunakan sebagai salah satu pengimplementasian steganografi.

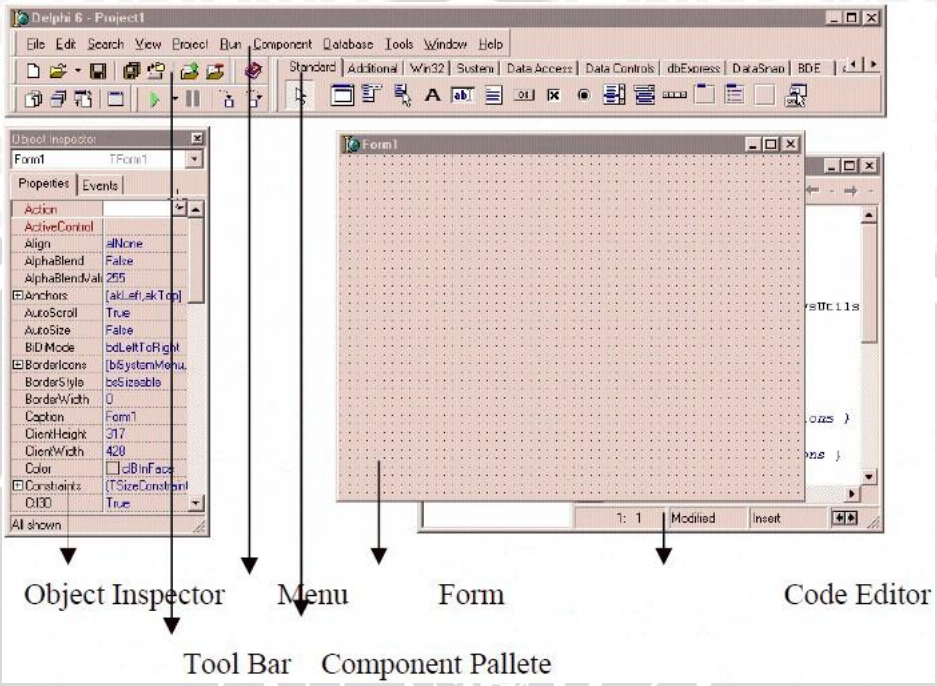
Steganografi pada media digital file gambar digunakan untuk mengeksploitasi keterbatasan kekuatan sistem penglihatan manusia dengan cara menurunkan kualitas warna pada file gambar yang belum disisipi pesan rahasia. Sehingga dengan keterbatasan tersebut manusia sulit menemukan gradasi penurunan kualitas warna pada file gambar yang telah disisipi pesan.



Gambar 2.9. Proses Steganografi

2.7. Borland Delphi 7

Borland Delphi merupakan program aplikasi database yang berbasis object Pascal dari Borland. Selain itu, Borland Delphi juga memberikan fasilitas pembuatan aplikasi visual.



Gambar 2.10. Tampilan IDE Borland Delphi

Borland Delphi memiliki komponen-komponen visual maupun non visual berintegrasi yang akan menghemat penulisan program. Terutama dalam hal perancangan antarmuka grafis (*Graphical User Interface*), kemampuan Borland Delphi untuk menggunakan Windows API (*Application Programming Interface*) ke dalam komponen-komponen visual menyebabkan pemrograman Borland Delphi yang bekerja dalam lingkungan Windows menjadi lebih mudah. Kelebihan Borland Delphi dalam hal kompilasi program juga menjadi faktor yang mempengaruhi pemilihan bahasa pemrograman yang digunakan. Karena program dikembangkan berdasarkan bahasa Pascal yang telah dikenal luas, maka untuk pengembangan program akan lebih mudah. Borland Delphi juga mempunyai kemampuan bekerja untuk pengolahan gambar dengan tersedianya unit GRAPHICS.

Warna pixel pada Borland Delphi dijabarkan dalam 4 byte hexadecimal, dengan 3 byte terendahnya adalah nilai intensitas RGB, dengan susunan sebagai berikut: $00BBGGRR, sebagai contoh nilai dari $00FF0000 berarti intensitas biru murni, $0000FF00 adalah hijau murni dan $000000FF adalah merah murni. $0000000 adalah hitam dan $00FFFFFF adalah putih. Untuk mengambil nilai intensitas RGB suatu pixel dapat menggunakan perintah GetRvalue, GetGvalue, GetBvalue.

BAB III
METODOLOGI PENELITIAN

3.1. Studi Literatur

Studi literatur menjelaskan dasar teori yang digunakan untuk menunjang penulisan skripsi. Teori-teori pendukung tersebut meliputi:

a. Membaca dan mempelajari buku – buku yang berhubungan dengan *image processing* dan kriptografi.
b. Mempelajari metode penyisipan data pada citra.
c. Mempelajari teknik – teknik dasar pemrograman dengan menggunakan Borland Delphi 7.

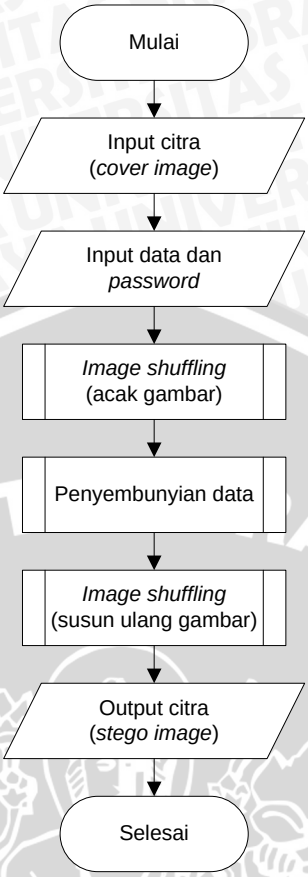
3.2. Analisa Kebutuhan

Analisa kebutuhan bertujuan untuk mendapatkan semua kebutuhan yang diperlukan dari sistem yang akan dibangun.

a. Perangkat Keras :
   - *Pentium IV CPU 2,26 GHz*
   - *RAM 512 MB*
   - *VGA Card NVIDIA GeForce2 MX/MX 400*
b. Perangkat Lunak:
   - Sistem Operasi *Windows XP Service Pack 2*
   - Aplikasi *Borland Delphi 7*

3.3. Diagram Alir

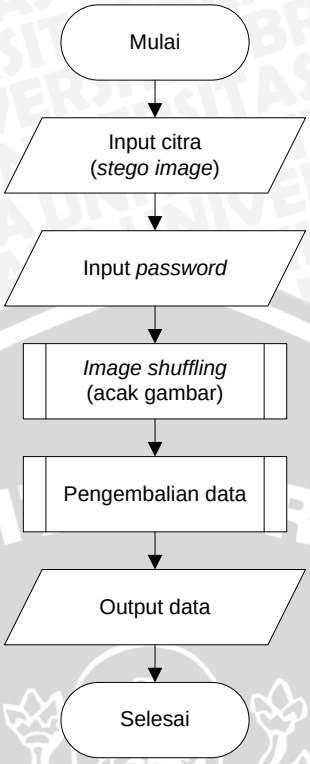
Untuk membantu memahami sistem, berikut ini merupakan blok diagram dari aplikasi yang akan dibuat. Dalam blok diagram tersebut terdapat proses penyembunyian pada diagram pertama dan pengembalian data pada diagram kedua. Berikut adalah blok diagramnya.



Gambar 3.1. Diagram Alir Proses Penyembunyian Data

Keterangan:

- Input citra adalah proses untuk melakukan pemilihan citra yang akan digunakan sebagai *cover image*. Setelah dilakukan pemilihan, citra akan ditampilkan.
- Input data merupakan proses pemasukan data yang akan disembunyikan beserta *password*nya.
- Pengacakan citra dilakukan dengan menggunakan *S-Box* yang dihasilkan melalui perhitungan pada *password*.
- Penyembunyian citra dilakukan dengan metode penyisipan LSB.



Gambar 3.2. Diagram Alir Proses Pengembalian Data

Keterangan:

- Input citra pembanding (*cover image*) adalah melakukan pemilihan citra yang akan difungsikan sebagai *cover image*. Setelah dilakukan pemilihan, citra akan ditampilkan.
- Input citra pembawa (*stego image*) adalah melakukan citra yang di dalamnya telah terdapat data yang pada proses akan dibandingkan dengan citra pembanding (*cover image*). Setelah dilakukan pemilihan, citra akan ditampilkan.
- Pengacakan citra dilakukan dengan menggunakan *S-Box* yang dihasilkan melalui perhitungan pada *password*.
- Pengembalian data dilakukan dengan metode kebalikan dari penyisipan LSB.

3.4. Perancangan dan Implementasi

Penyembunyian data dalam citra digital dengan metode steganografi terdiri dari banyak proses yang meliputi proses pengolahan citra, proses

pembacaan data, proses enkripsi dan proses dekripsi, oleh karena itu, selain perangkat keras yang memadai desain aplikasi yang baik dan *user friendly* juga dibutuhkan demi kemudahan menggunakan perangkat lunak ini.

a. Perancangan antarmuka aplikasi yaitu mendesain aplikasi agar sesuai dengan platform dan bahasa pemrograman yang dipakai.
b. Perancangan algoritma *image shuffling* berupa penyusunan *S-box* dari hasil pengolahan kunci.
c. Perancangan implementasi algoritma penyisipan LSB untuk diterapkan pada aplikasi.
d. Perancangan perangkat lunak agar dapat melakukan proses membaca, menyembunyikan dan mengembalikan data sesuai kebutuhan aplikasi.

3.5. Pengujian

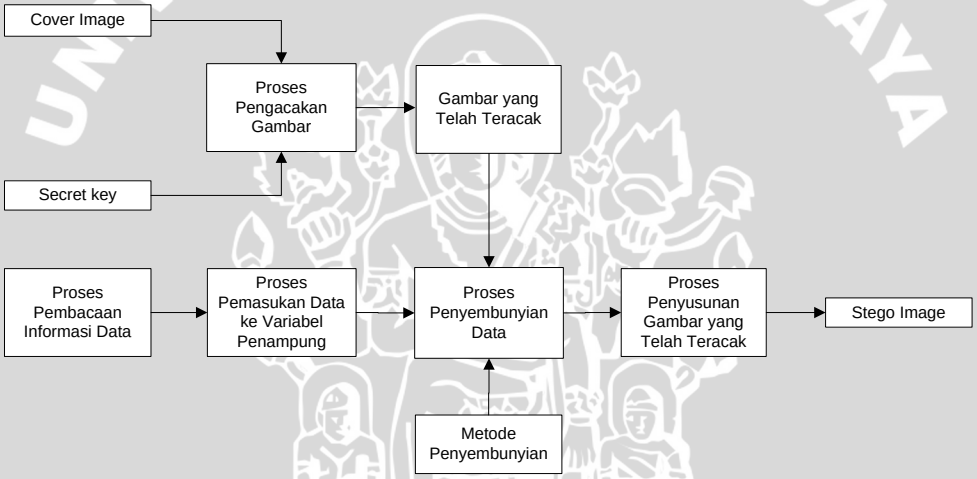
Pengujian dilakukan untuk menjamin dan memastikan bahwa sistem yang telah dirancang memiliki tingkat kesalahan yang kecil. Untuk mengetahui apakah sistem bekerja dengan baik dan sesuai dengan perancangan, maka diperlukan serangkain pengujian. Pengujian yang dilakukan adalah sebagai berikut:

a. Pengujian ketahanan metode yang digunakan terhadap *steganographic attack*
b. Pengujian mutu (*fidelity*) pada citra pembawa (*stego image*)
c. Perbandingan data hasil ekstraksi dari citra dengan data sebenarnya

BAB IV
PERANCANGAN DAN IMPLEMENTASI

4.1. Perancangan Secara Umum

Perangkat lunak ini didesain untuk melakukan penyembunyian data pada citra dengan menggunakan metode steganografi serta algoritma enkripsi RC4 sebagai pendukungnya. Secara garis besar algoritma terdiri dari 2 bagian besar, yaitu algoritma yang digunakan untuk menyembunyikan data pada citra digital dan algoritma untuk mengembalikan data dari citra digital.

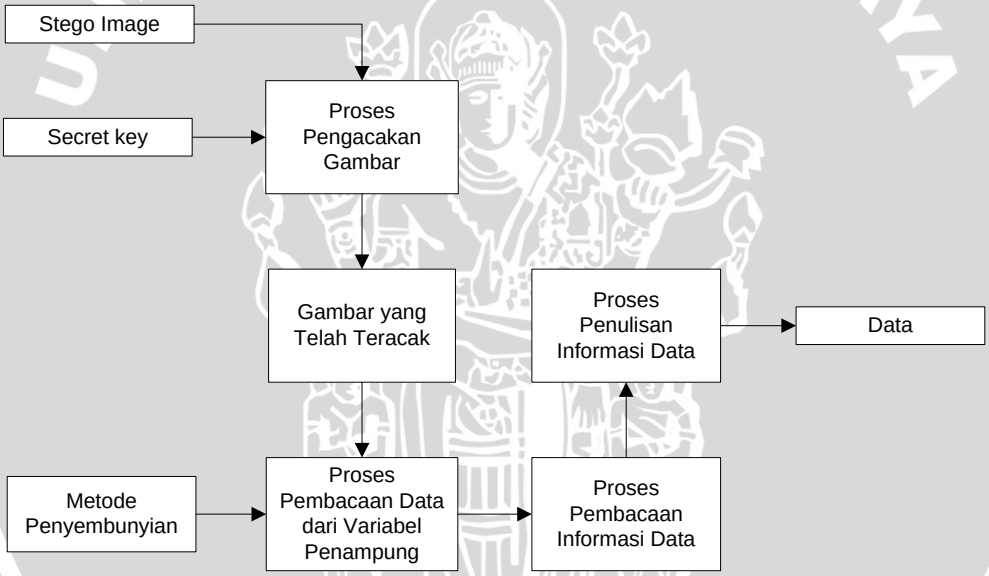


Gambar 4.1. Proses Penyembunyian Data

Diagram blok diatas dapat dijelaskan menjadi langkah-langkah teknis sebagai berikut:

1. *Cover image* diperoleh dengan me*load* citra yang ada di dalam *hardisk* menggunakan komponen `OpenPictureDialog` pada Delphi.
2. Kata kunci dimasukkan oleh *user* yang akan digunakan sebagai *secret key* yang akan digunakan pada proses penyembunyian data.
3. Proses pengacakan gambar menggunakan metode *image shuffling* dan *secret key* yang telah dimasukkan sebelumnya.

4. Data merupakan berkas dengan jenis apapun yang berada di dalam *hardisk* , dan dapat *diload* menggunakan komponen `OpenDialog` pada Delphi.
5. Proses penyembunyian data melibatkan gambar yang telah teracak (*shuffled image*) dan data menggunakan penyisipan LSB (*Least Significant Bit*) sebagai metode penyembunyian data.
6. Setelah citra selesai disisipi dengan data, maka citra disusun ulang menggunakan metode *image shuffling* dan *secret key* yang telah digunakan pada saat pengacakan citra.
7. *Stego image* yang diperoleh melalui proses ini telah siap ditransmisikan ke tempat tujuan.



Gambar 4.2. Proses Pengembalian Data

Diagram blok diatas dapat dijelaskan menjadi langkah-langkah teknis sebagai berikut:

1. *Stego image* diperoleh dengan me*load* citra hasil proses penyembunyian data yang ada di dalam *hardisk* menggunakan komponen `OpenPictureDialog` pada Delphi.

2. Kata kunci yang dimasukkan oleh *user* sebagai *secret key* harus sama dengan kata kunci yang digunakan pada proses penyembunyian data.
3. Proses pengacakan gambar menggunakan metode *image shuffling* dan *secret key* yang telah dimasukkan sebelumnya.
4. Proses pembacaan data melibatkan gambar yang telah teracak (*shuffled image*) menggunakan metode pembacaan LSB (*Least Significant Bit*) yang merupakan kebalikan dari metode penyisipan LSB.
5. Setelah data selesai dibaca, maka data ditulis ulang menjadi sebuah berkas sebagaimana berkas yang asli.

4.2. Perancangan Perangkat Lunak

Pada perancangan perangkat lunak terdapat sebuah modul program yaitu modul program steganografi. Perangkat lunak untuk modul program ini dibangun dengan bahasa pemrograman Borland Delphi 7. Saat program dijalankan, user dapat memilih pesan rahasia yang akan disisipkan beserta citra asli yang akan menjadi citra pembawanya. User juga diminta untuk memasukan kunci sebagai *key* untuk kemudian digunakan dalam proses *image shuffling*. Kunci yang sama kembali diminta ketika user akan mengekstrak pesan rahasia yang telah disisipkan.

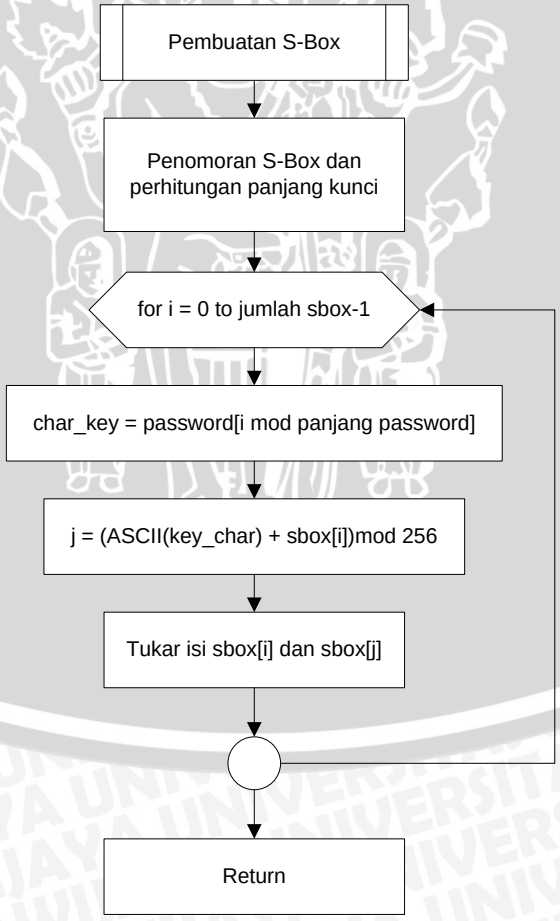
4.3. Perancangan Modul Program Steganografi

Perancangan modul steganografi meliputi pengolahan kunci dimana kunci yang dimasukkan oleh user akan digunakan untuk menjalankan proses *image shuffling,* baik dalam proses penyembunyian maupun pembacaan data atau pesan rahasia. Proses penyembunyian data berupa proses penyisipan bit-bit data pesan rahasia ke dalam citra asli yang telah diacak dengan metode *image shuffling* yang kemudian disusun kembali dengan proses yang sama untuk menjadi sebuah citra pembawa. Sedangkan proses pembacaan data akan mengekstrak bit-bit data yang berada di dalam citra pembawa menjadi

data pesan rahasia. Dalam setiap proses ini akan dijelaskan tahapan-tahapan yang membentuk semua proses sehingga menjadi program yang utuh.

4.3.1. Modul Program *Image Shuffling*

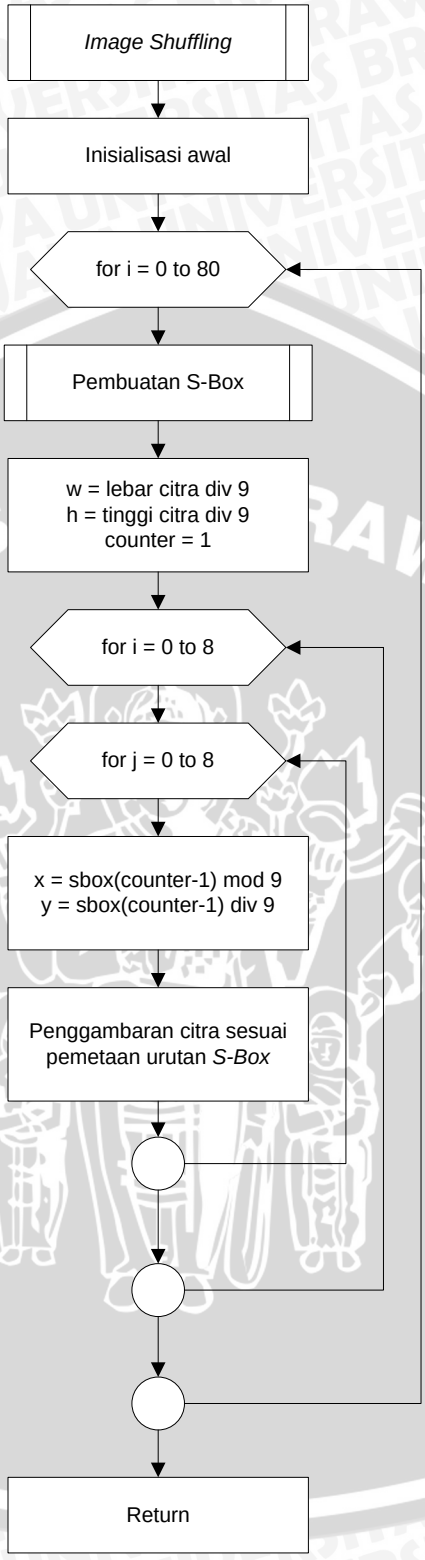
Image shuffling merupakan proses untuk mengacak citra digital berdasarkan urutan *S-Box* yang dihasilkan melalui perhitungan pada *password*. Penggunaan *S-Box* dalam pengacakan citra digital bertujuan agar citra yang telah diacak dapat dikembalikan ke bentuk semula dengan sempurna. Hal ini berarti dalam proses pengembalian data data yang telah disembunyikan dalam bagian-bagian citra digital yang telah diacak dapat dikembalikan ke susunan semula dan tidak terjadi kerusakan pada data. Proses pembuatan *S-Box* dijelaskan pada gambar berikut.



Gambar 4.3. Diagram Alir Proses Pembuatan *S-Box*

Algoritma pembuatan *S-Box* ini diadopsi dari pembuatan *S-Box* pada algoritma enkripsi RC4 (*Ron's Cipher #4*). Diagram alir diatas dapat dijelaskan menjadi langkah-langkah teknis sebagai berikut:

1. Penomoran pada sel-sel *S-Box* serta perhitungan panjang *secret key* atau kata kunci yang digunakan.
2. Melakukan proses perulangan sejumlah sel yang terdapat pada *S-Box*
3. Dalam setiap proses perulangan dilakukan perhitungan nilai ASCII dari setiap karakter pada *secret key*
4. Hasil perhitungan nilai ASCII kemudian ditambahkan dengan nilai pada sel *S-Box*
5. Hasil dari perhitungan tersebut digunakan untuk melakukan pertukaran sel pada *S-Box*.



Gambar 4.4. Diagram Alir Proses *Image Shuffling*

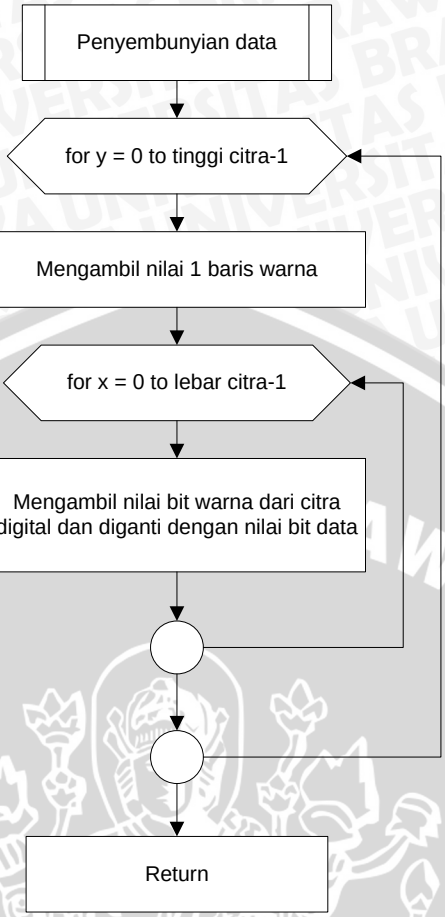
Diagram alir diatas dapat dijelaskan menjadi langkah-langkah teknis sebagai berikut:

1. Proses pengacakan citra dilakukan dengan membagi citra menjadi beberapa bagian kecil sesuai dengan jumlah sel pada *S-Box* yang telah ditentukan
2. Ukuran array *i* adalah nilai kuadrat dari jumlah pembagian citra yang dikehendaki. Apabila panjang dan lebar citra dikehendaki dibagi menjadi lima, maka ukuran array *i* adalah 5x5 = 25.
3. *S-Box* diinisialisasi, lalu dilakukan pembagian citra menjadi bagian-bagian kecil.
4. Bagian-bagian citra tersebut lalu disusun sesuai dengan urutan pada *S-Box*.

Pada proses pengembalian data langkah yang dilakukan sama kecuali pada peletakan bagian-bagian citra. Ini dikarenakan RC4 yang merupakan dasar dari *S-Box* yang digunakan adalah algoritma enkripsi simetris, yaitu algoritma yang menggunakan kunci yang sama untuk melakukan proses enkripsi maupun proses dekripsi.

4.3.2. Perancangan Program Penyembunyian Data

Proses penyembunyian data pada citra menggunakan metode penyisipan LSB (*Least Significant Bit*). Metode ini dilakukan dengan mengganti bit-bit akhir warna pada citra dengan bit-bit data. Proses penyembunyian data dengan metode ini dijelaskan pada gambar berikut.



Gambar 4.5. Diagram Alir Metode Penyembunyian Data

Diagram alir diatas dapat dijelaskan menjadi langkah-langkah teknis sebagai berikut:

1. Proses diawali dengan melakukan perulangan sebanyak tinggi dari citra digital .
2. Kemudian dilakukan pembacaan baris data warna yang dilanjutkan oleh perulangan sebanyak lebar citra digital.
3. Proses utama dalam diagram alir ini terdapat pada pengambilan nilai bit-bit warna dari citra digital untuk diproses kemudian digabungkan dengan bit-bit data. Proses ini dilakukan pada setiap warna yang ada (merah, hijau, biru) dengan melakukan operasi AND dengan nilai bilangan 248 yang memiliki nilai biner 11111000 untuk membuang 3 bit paling belakang dari warna pada citra digital.

4. Proses dilanjutkan dengan operasi OR dengan bit data yang dimasukkan ke dalam variabel penampung. Hasil dari operasi ini merupakan nilai *byte* warna yang telah disisipi data pada 3 bit paling belakang dari total 8 bit untuk masing-masing warna pada citra digital.
5. Perulangan terus dilakukan hingga seluruh titik warna (*pixel*) pada citra selesai diproses.

4.3.3. Perancangan Program Pengembalian Data

Proses pengembalian data memiliki prinsip yang sama dengan proses penyembunyian data, kecuali pada urutan proses. Pada proses pengembalian data, proses dilakukan dengan urutan terbalik, proses yang dilakukan pertama kali pada pengembalian data adalah proses terakhir yang dilakukan pada penyembunyian data.

Dalam proses pengembalian data dilakukan pengambilan nilai data yang telah disembunyikan pada metode penyisipan LSB. Langkah yang pertama kali dilakukan adalah pembacaan *byte* warna pada citra digital, kemudian dilakukan proses menggunakan operasi AND dengan bilangan 7 yang memiliki nilai biner 00000111 untuk mengambil nilai 3 bit paling belakang dari *byte* warna yang merupakan data yang disembunyikan pada proses *snap from image*, lalu data yang telah terbaca dimasukkan ke dalam variabel penampung yang akan dikembalikan ke bentuk data semula.

4.4. Implementasi Sistem

Setelah tahap perancangan sistem, tahap selanjutnya adalah tahap implementasi. Tujuan dari tahap implementasi ini merupakan proses transformasi hasil perancangan perangkat lunak. Pembahasan terdiri dari lingkungan implementasi (spesifikasi perangkat lunak dan perangkat keras), implementasi antarmuka aplikasi dengan sintaks dari bahasa pemrograman yang digunakan.

4.4.1. Lingkungan Implementasi

Sistem dibuat dengan menggunakan Borland Delphi 7 dengan sistem diimplementasikan dengan menggunakan spesifikasi sebagai berikut:

1. Perangkat keras :
   1.1. Komputer
   Spesifikasi :
   - Processor : Pentium IV CPU 2,26 GHz
   - Memory : 512 MB RAM
   - OS : Windows XP XP SP 2 2002
   - VGA : NVIDIA GeForce2 MX/MX 400
2. Perangkat Lunak :
   2.1. Sistem operasi : Microsoft Windows XP SP 2 2002
   2.2. Bahasa pemrograman : Borland Delphi 7

4.4.2. Implementasi Penyembunyian Data

Implementasi pada antarmuka penyembuyian data. Operasi pemanggilan berkas, baik citra asli menggunakan `OpenPictureDialog` maupun data menggunakan `OpenDialog` dilakukan pada bagian ini. Berikut bagian dari *script* Delphi 7.

```
Encrypt(OpenDialog1.FileName,
OpenPictureDialog1.FileName, SavePictureDialog1.FileName,
UpDown1.Position, ProgressBar1, MaskEdit1.Text,
MaskEdit1.GetTextLen, Image1, Image2,Image3);
```

Prosedur `Encrypt` yang ditampilkan pada segmen program utama ini merupakan prosedur yang melakukan seluruh proses penyembunyian data.

4.4.3. Implementasi Proses Penyisipan Data

Implementasi pada proses penyisipan data menggunakan penyisipan LSB (*Least Significant Bit*). Pada bagian ini, bit-bit citra asli

(*cover image*) ditukar dengan bit-bit data. Berikut bagian dari *script* Delphi 7.

```
procedure TForm1.ProcessEncrypt(Bitmap: TBitmap; Source:
TFileStream; Destination : string; BPC: LongInt;
ProgressBar: TProgressBar);
```

Prosedur `ProcessEncrypt` yang ditampilkan pada segmen program ini merupakan bagian dimana proses penyisipan LSB (*Least Significant Bit*) dilakukan. Bagian prosedur `ProcessEncrypt` adalah prosedur `CheckNextPixel`, yang berfungsi untuk mengecek piksel pada yang tersedia sebelum dilakukan proses penyisipan data. Prosedur `CheckNextPixel` ditunjukkan dalam potongan program berikut.

```
    procedure CheckNextPixel;
    begin
      if (RGBIndex <= 3) and (PixelBitIndex + 1 < BPC)
then // Pindah ke bit berikutnya
        Inc(PixelBitIndex)
      else if RGBIndex < 3 then // Tukar ke channel RGB
selanjutnya
        begin
        Inc(RGBIndex);
        PixelBitIndex:= 0;
        end
      else if RGBIndex = 3 then // Load pixel berikutnya
        begin
        PixelBitIndex:= 0;
        RGBIndex:= 1;
        if PixelsRowIndex = PixelsRowMax then // Seluruh
pixel pada baris ini telah digunakan
          begin
          Inc(CurrentRow);
          PixelsRow:= Bitmap.ScanLine[CurrentRow];
          PixelsRowIndex:= 1;
          end
        else  // Masih ada pixel tersisa pada baris ini
```

```
            begin
            Inc(PixelsRowIndex);
            Inc(PixelsRow); // Menambah nilai pointer
agarmenunjuk pada pixel berikutnya
            end;
          end;
      end;
```

Bagian dari prosedur `ProcessEncrypt`. Pada bagian ini dilakukan penyimpanan bit-bit data ke dalam jumlah LSB (*Least Significant Bit*) yang digunakan, mulai 1 (minimal) hingga 8 (maksimal).

```
begin
      PixelsRow:= Bitmap.ScanLine[0];

      // Menggunakan 2 bit samapi 4 bit pada channel B
agar dapat menyimpan nilai 8 pada Bit per Channel
      SetBitAt(PixelsRow^[1], 0, GetBitAt(BPC, 0));
      SetBitAt(PixelsRow^[1], 1, GetBitAt(BPC, 1));
      SetBitAt(PixelsRow^[2], 0, GetBitAt(BPC, 2));
      SetBitAt(PixelsRow^[3], 0, GetBitAt(BPC, 3));
```

Bagian dari prosedur `ProcessEncrypt`. Pada bagian ini dilakukan penyimpanan panjang data yang sebenarnya ke dalam variabel penampung sementara.

```
        SourceSize:= Source.Size;
      for BitIndex:= 0 to SizeOf(SourceSize) * 8 - 1 do
        begin
        SetBitAt(PixelsRow^[RGBIndex], PixelBitIndex,
GetBitAt(SourceSize, BitIndex));
        CheckNextPixel;
        end;
```

Bagian dari prosedur `ProcessEncrypt`. Pada bagian inilah dilakukan penyimpanan bit-bit data ke dalam piksel-piksel citra.

```
      // Menyalin bit dari tiap byte pada source stream
pada bit-bit pada pixel
      Source.Seek(0, soFromBeginning);
      for SourceIndex:= 0 to SourceSize - 1 do
        begin
          if (SourceIndex + 1) mod 10 = 0 then
          begin
            ProgressBar.StepIt;
            Application.ProcessMessages;
          end;
```

Implementasi Prosedur *SetBitAt* pada proses penyisipan LSB (*Least Significant Bit*), merupakan prosedur untuk menempatkan posisi pointer.

```
procedure SetBitAt(var Variable: LongInt; Position: Byte;
Value: Boolean);
begin
  if Value then
    Variable:= Variable or (1 shl Position)
  else
    Variable:= Variable and ((1 shl Position) xor
$FFFFFFFF);
end;

procedure SetBitAt(var Variable: Byte; Position: Byte;
Value: Boolean);
begin
  if Value then
    Variable:= Variable or (1 shl Position)
  else
    Variable:= Variable and ((1 shl Position) xor $FF);
end;
```

Implementasi Prosedur *GetBitAt* pada proses penyisipan LSB (*Least Significant Bit*), merupakan fungsi untuk mengambil nilai bit pada data.

```
function GetBitAt(Variable: LongInt; Position: Byte):
Boolean;
begin
  if Variable and (1 shl Position) <> 0 then
    Result:= True
  else
    Result:= False;
end;
```

4.4.4. Implementasi Penyisipan Data

Implementasi pada penyembuyian data mencakup algoritma penyisipan LSB (*Least Significant Bit*), pembuatan *S-Box*, dan *Image Shuffling* dan *Image Ordering*. Berikut bagian dari *script* Delphi 7.

```
procedure TForm1.Encrypt(const SourceFile, SourceBitmap,
Destination: string; BitsPerChannel: LongInt; ProgressBar:
TProgressBar; key : string;keyL : integer; CoverImage :
TImage; ShuffledImage : TImage;OrderedImage : TImage);
var Bitmap: TBitmap;
    sSource: TFileStream;
    KeyLength, x, y, j, i, blokh, blokw, count, temp :
integer;
      KeyChar : char;
    sbox : Array[1..25] of integer;
```

Prosedur `Encrypt` yang ditampilkan pada segmen program ini merupakan bagian dimana pemanggilan prosedur `ProcessEncrypt` penyisipan LSB (*Least Significant Bit*) dan metode *Image Shuffling* dan *Image Ordering* dilakukan. Pada bagian deklarasi, terlihat *S-Box* yang akan digunakan pada proses *Image Shuffling* dan *Image Ordering* diinisialisasi.

Pada bagian program berikut, ditampilkan proses pengecekan citra asli (*cover image*) yang kan digunakan sebagai media pembawa data. Program akan mengecek apakah citra memiliki kedalaman 24 bit, memiliki ukuran lebih dari nol byte dan lebih besar daripada ukuran data

yang kan disisipkan. Jika syarat-syarat tersebut tidak dipenuhi, maka citra asli (*cover image*) tersebut tidaka dapat dijadikan media pembawa bagi data rahasia.

```
try
    Bitmap.LoadFromFile(SourceBitmap);
    if Bitmap.PixelFormat <> pf24bit then
      raise Exception.Create('The image must have a 24-bit
depth.');

    sSource:= TFileStream.Create(SourceFile, fmOpenRead);

    if sSource.Size = 0 then
      raise Exception.Create('The source file is 0 bytes.
There''s nothing to hide.');

    if sSource.Size * 8 + SizeOf(LongInt) * 8  + 1 >
Bitmap.Width * Bitmap.Height * 3 * BitsPerChannel then
      raise Exception.Create('The image is not big enough
to accommodate the file.');
    // + 1: Nilai BitsPerChannel disimpan pada piksel
pertama
    // SizeOf(LongInt) * 8: jumlah data asli yang
terenkripsi akan disimpan pada piksel pertama setelah
BitsPerChannel
```

4.4.5. Implementasi *Image Shuffling*

Implementasi pada proses *image shuffling* atau pengacakan citra. Terdiri dari pembuatan *S-Box* dari algoritma enkripsi RC4, pembagian citra menjadi beberapa bagian, dan pengacakan bagian-bagian citra sesuai dengan *S-Box*. Berikut bagian dari *script* Delphi 7.

Langkah pertama adalah dengan menginisialisasi *S-Box*.

```
for i:=0 to 24 do
        sbox[i] := i;

KeyLength := KeyL;
```

```
//pembuatan Sbox
  for i:=0 to 24 do
  begin
  KeyChar := Key[i mod KeyLength];
        count := (ord(KeyChar) + sbox[i]) mod 25;
        temp := sbox[i];
          sbox[i] := sbox[count];
        sbox[count] := temp;
  end;
```

Langkah berikutnya adalah membagi tinggi dan lebar citra menjadi jumlah yang sama.

```
  blokw := ShuffledImage.Picture.width div 5;
  blokh := ShuffledImage.Picture.height div 5;
```

Kemudian bagian-bagian potongan citra diacak sesuai dengan urutan *S-Box* yang telah ditentukan sebelumnya.

```
count := 1;
  //proses mengacak citra digital
  for i:=0 to 24 do
        for j:=0 to 4 do
        begin
            x := sbox[count-1] mod 5;
            y := sbox[count-1] div 5;

    ShuffledImage.Canvas.CopyRect(rect(j*blokw,
i*blokh,((j+1)*blokw),((i+1)*blokh)),
    CoverImage.Picture.Bitmap.Canvas, rect(x*blokw,
y*blokh,((x+1)*blokw),((y+1)*blokh)));

    count := count + 1;
```

4.4.6. Implementasi *Image Ordering*

Implementasi pada proses *image ordering* atau penyusunan citra. Terdiri dari langkah-langkah yang sama dengan proses *image shuffling* namun memiliki fungsi kebalikan dari proses tersebut. Seperti dijelaskan sebelumnya, hal ini disebabkan karena RC4 yang merupakan dasar algoritma dari *S-Box* yang digunakan adalah algoritma enkripsi simetris, yaitu algoritma yang menggunakan kunci yang sama untuk melakukan proses enkripsi maupun proses dekripsi. Berikut bagian dari *script* Delphi 7.

Langkah pertama adalah dengan menginisialisasi *S-Box*.

```
for i:=0 to 24 do
          sbox[i] := i;

KeyLength := KeyL;

  //pembuatan Sbox
    for i:=0 to 24 do
    begin
          KeyChar := key[i mod KeyLength];
          count := (ord(KeyChar) + sbox[i]) mod 25;
          temp := sbox[i];
          sbox[i] := sbox[count];
    sbox[count] := temp;
    end;
```

Langkah berikutnya adalah membagi tinggi dan lebar citra menjadi jumlah yang sama.

```
    blokw := CoverImage.Picture.width div 5;
    blokh := CoverImage.Picture.height div 5;
```

Kemudian bagian-bagian potongan citra disusun sesuai dengan urutan *S-Box* yang telah ditentukan sebelumnya.

```
count := 1;
    //proses menyusun citra digital
    for i:=0 to 24 do
        for j:=0 to 4 do
        begin
            x     := sbox[count-1] mod 5;
            y     := sbox[count-1] div 5;


        OrderedImage.canvas.CopyRect(rect(x*blokw,
y*blokh, ((x+1)*blokw), ((y+1)*blokh)),
        ShuffledImage.Picture.bitmap.canvas,
rect(j*blokw, i*blokh,((j+1)*blokw),((i+1)*blokh)));

        count := count + 1;
```

4.4.7. Implementasi Pengembalian Data

Implementasi pada antarmuka pengembalian data. Operasi pemanggilan berkas citra pembawa menggunakan `OpenPictureDialog` dilakukan pada bagian ini. Berikut bagian dari *script* Delphi 7.

```
Decrypt(OpenPictureDialog1.FileName,
SaveDialog1.FileName, ProgressBar1,MaskEdit1.Text,
MaskEdit1.GetTextLen, Image1, Image2);
```

Prosedur `Decrypt` yang ditampilkan pada segmen program utama ini merupakan prosedur yang melakukan seluruh proses pengembalian data.

4.4.8. Implementasi Proses Pembacaan Data

Implementasi pada proses pembacaan data menggunakan kebalikan proses penyisipan LSB (*Least Significant Bit*) sebagai metode pembacaan data. Pada bagian ini, bit-bit data diambil dari LSB bit-bit citra yang digunakan sebagai tempat penyimpanan data. Berikut bagian dari *script* Delphi 7.

```
procedure TForm1.ProcessDecrypt(Bitmap: TBitmap;
Destination: TFileStream; ProgressBar: TProgressBar);
```

Prosedur `ProcessDecrypt` yang ditampilkan pada segmen program ini merupakan bagian dimana proses pembacaan LSB (*Least Significant Bit*) dilakukan.Pprosedur `CheckNextPixel`, yang berfungsi untuk mengecek piksel pada yang tersedia sebelum dilakukan proses penyisipan data, kembali digunakan di sini. Prosedur `CheckNextPixel` ditunjukkan dalam potongan program berikut.

```
procedure CheckNextPixel;
begin
  if (RGBIndex <= 3) and (PixelBitIndex  + 1 < BPC) then
    Inc(PixelBitIndex)
  else if RGBIndex < 3 then
    begin
    Inc(RGBIndex);
    PixelBitIndex:= 0;
    end
  else if RGBIndex = 3 then
    begin
    PixelBitIndex:= 0;
    RGBIndex:= 1;
    if PixelsRowIndex = PixelsRowMax then
      begin
      Inc(CurrentRow);
      if CurrentRow > MaxRows then
        raise Exception.Create('The end of the image was
reached while trying to read the hidden information.' +
#13#10+ 'This is probably caused by an image that doesn''t
contain any hidden data.');

      PixelsRow:= Bitmap.ScanLine[CurrentRow];
      PixelsRowIndex:= 1;
      end
    else
      begin
      Inc(PixelsRowIndex);
```

```
      Inc(PixelsRow);
      end;
    end;
end;
```

Exception pada potongan program di atas berfungsi untuk memunculkan kotak pesan aapabila tidak ditemukan data rahasia setelah pengecekan piksel dilakukan.

Bagian dari prosedur `ProcessDecrypt`. Pada bagian ini dilakukan pembacaan bit-bit data dari LSB (*Least Significant Bit*) yang digunakan.

```
begin
  PixelsRow:= Bitmap.ScanLine[0];
  BPC:= 0;
  SetBitAt(BPC, 0, GetBitAt(PixelsRow^[1], 0));
  SetBitAt(BPC, 1, GetBitAt(PixelsRow^[1], 1));
  SetBitAt(BPC, 2, GetBitAt(PixelsRow^[2], 0));
  SetBitAt(BPC, 3, GetBitAt(PixelsRow^[3], 0));
```

Bagian dari prosedur `ProcessDecrypt`. Pada bagian ini dilakukan pembacaan panjang data yang sebenarnya ke dalam variabel penampung sementara. Padabagian ini juga dilakukan pengecekan ukuran citra pembawa, apabila lebih kecil sama dengan nol byte, maka diasumsikan tidak ada yang tersimpan pada citra tersebut.

```
for BitIndex:= 0 to SizeOf(DataSize) * 8 - 1 do
    begin
    SetBitAt(DataSize, BitIndex,
GetBitAt(PixelsRow^[RGBIndex], PixelBitIndex));
    CheckNextPixel;
    end;

  if DataSize <= 0 then
    raise Exception.Create('The stored size of the hidden
data is not correct.' + #13#10 + 'This is probably caused
by an image that doesn''t contain any hidden data.');
```

Bagian dari prosedur `ProcessDecrypt`. Pada bagian inilah dilakukan ekstraksi bit-bit data dari piksel-piksel citra.

```
for DataIndex:= 1 to DataSize do
  begin

  if DataIndex mod 10 = 0 then
    begin
    ProgressBar.StepIt;
    Application.ProcessMessages;
    end;

  for BitIndex:= 0 to 7 do
    begin
    SetBitAt(Data, BitIndex,
GetBitAt(PixelsRow^[RGBIndex], PixelBitIndex));
    CheckNextPixel;
    end;

  Destination.Write(Data, 1);

  end; //untuk DataIndex
   end;
```

4.4.9. Implementasi Pembacaan Data

Implementasi pada pembacaan data mencakup algoritma pembacaan LSB (*Least Significant Bit*), pembuatan *S-Box*, dan *Image Shuffling*. Berikut bagian dari *script* Delphi 7.

```
procedure TForm1.Decrypt(const SourceFile, DestFile:
string; ProgressBar: TProgressBar; key : string; keyL :
integer; CoverImage : TImage; ShuffledImage : TImage);
var Bitmap: TBitmap;
    Destination: TFileStream;
    KeyLength, x, y, j, i, blokh, blokw, count, temp :
integer;
      KeyChar : char;
     sbox : Array[1..25] of integer;
```

Prosedur `Decrypt` yang ditampilkan pada segmen program ini merupakan bagian dimana pemanggilan prosedur `ProcessDecrypt` pembacaan LSB (*Least Significant Bit*) dan metode *Image Shuffling* dan dilakukan. Segneb program ini serupa dengan prosedur `Encrypt` pada proses penyembunyian data.

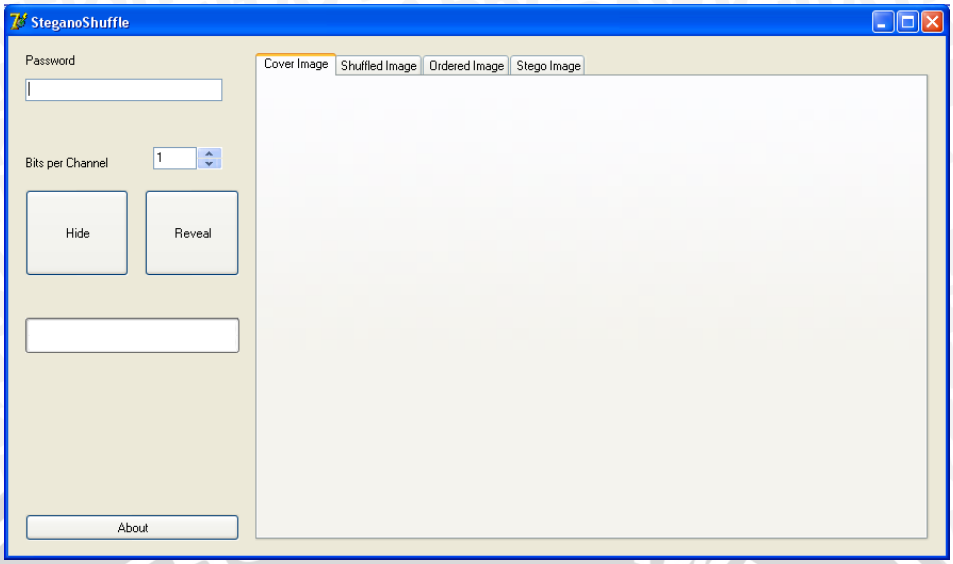
```
try
    try
      if FileExists(DestFile) then
DeleteFile(PAnsiChar(DestFile));

      Destination:= TFileStream.Create(DestFile, fmCreate);
      Bitmap.LoadFromFile(SourceFile);

      if Bitmap.PixelFormat <> pf24bit then
        raise Exception.Create('The image doesn''t have a
24-bit depth. It surely hasn''t been created by this
program.');
```

4.4.10. Cara Instalasi

Program ini tidak memerlukan cara instalasi yang khusus, dengan alasan bahwa semua *file* yang dibutuhkan oleh aplikasi ini dapat dikompilasi menjadi satu *file executeable*. Instalasi program ini cukup dengan meng-*copy file executeable*nya ke dalam lokasi *folder* yang dipilih pada *harddisk*.

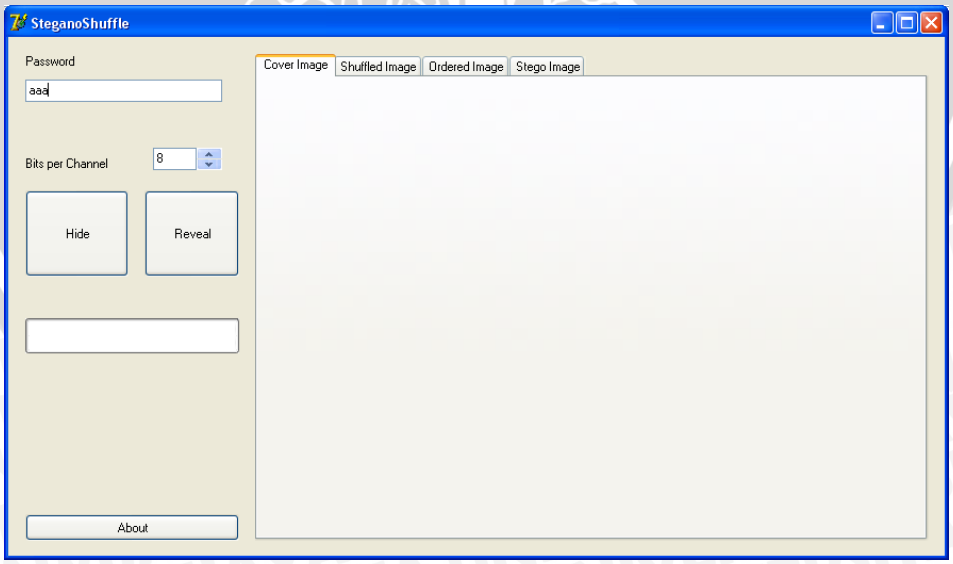
4.4.11. Implementasi Antarmuka

Untuk menjalankan program ini dapat dilakukan dengan mengklik pada ikon program. Setelah program di-*load* maka akan terlihat bentuk tampilan seperti gambar di bawah ini. Bentuk antarmuka program ini hanya terdiri dari satu *form*, sehingga mudah untuk dioperasikan. Terdapat tiga *button* perintah, “Hide” untuk melakukan penyembunyian data, “Reveal” untuk melakukan pembacaan data, serta “About” yang menampilkan informasi penyusun program.



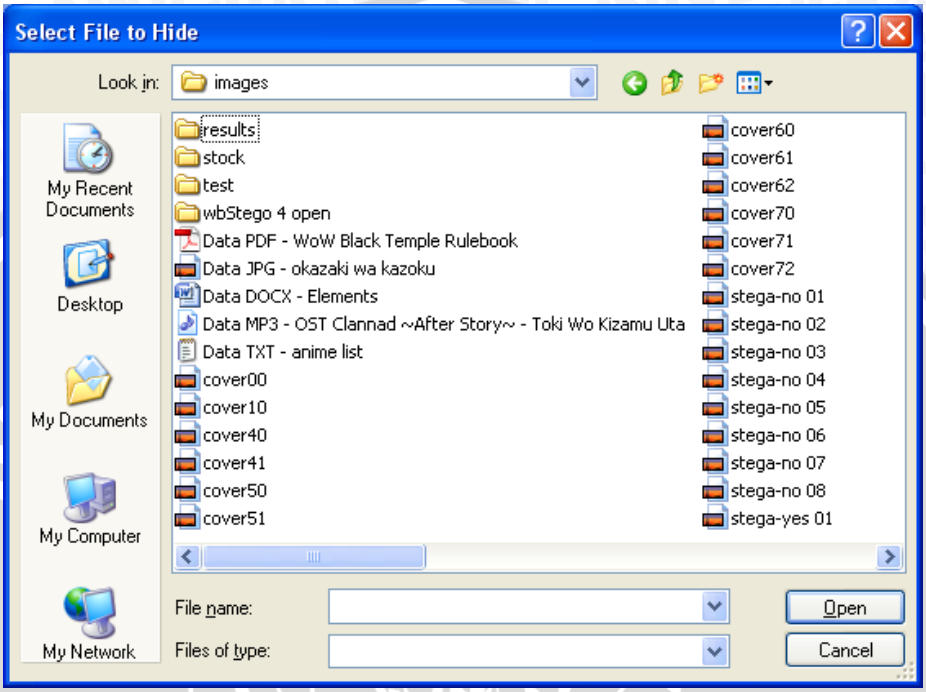
Gambar 4.6. Tampilan Utama

User dapat memilih untuk melakukan penyembunyian ke atau pembacaan data dari suatu citra pembawa. Untuk melakukan penyembunyian data, dapat dilakukan dengan mengklik pada tombol "Hide". Namun sebelum itu, *user* diminta untuk memasukkan *password* dan memilih jumlah LSB pada citra asli yang kan digunakan untuk menampung data pesan rahasia.



Gambar 4.7. Tampilan Pengisian *Password* dan Pemilihan Jumlah LSB

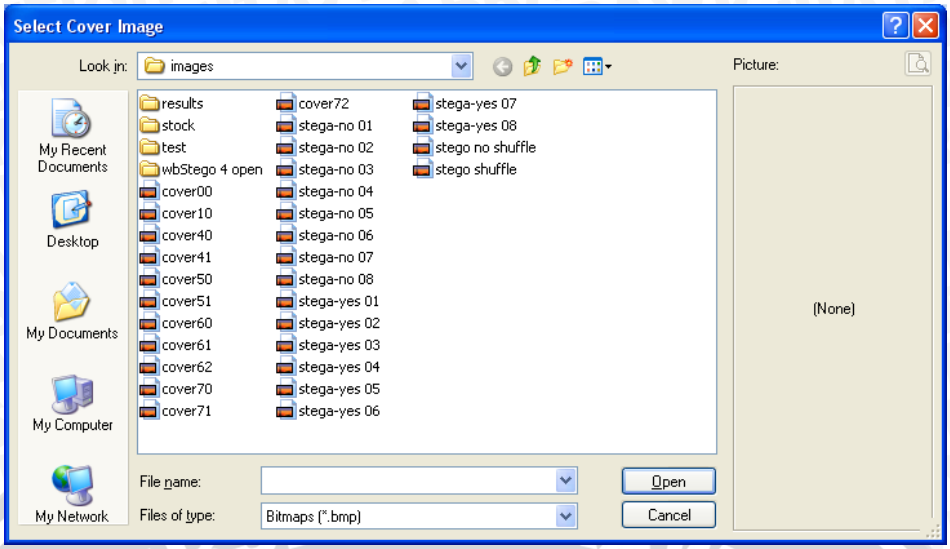
User kemudian dapat mengklik pada tombol "Hide" untuk memilih sebuah *file* tunggal yang akan disisipkan/disembunyikan dengan memunculkan sebuah kotak dialog untuk memilih satu buah *file* berjenis apapun.



Gambar 4.8. Tampilan Pemilihan File *Hidden Message* Melalui Kotak Dialog

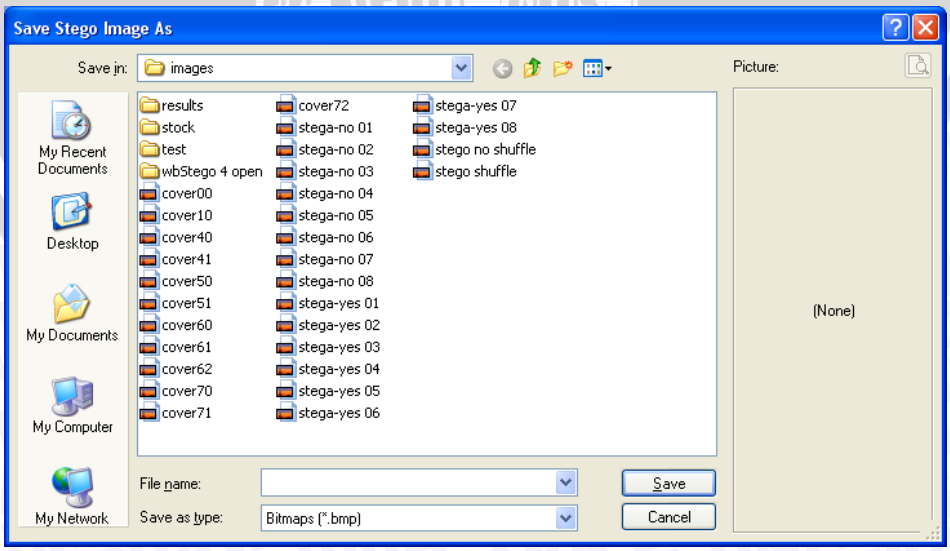
Selanjutnya, *user* dapat memilih *file* citra asli (*cover image*) yang akan dijadikan citra pembawa (*stego image*). *File* yang dimunculkan dalam daftar pada kotak dialog hanya file citra dalam format *bitmap*.

Perlu diperhatikan di sini, bahwa *file* citra asli (*cover image*) yang dipilih harus lebih besar dari *file* data pesan rahasia.



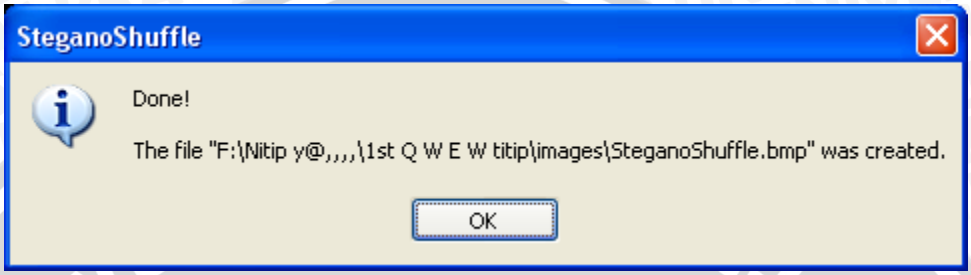
Gambar 4.9. Tampilan Pemilihan File *Cover Image* Melalui Kotak Dialog

User kemudian memilih *path* output sebagai *path* keluar *file* yang diproses. Bila *user* menggunakan *path* yang sama dengan *path file* sumber maka *file* hasil akan menimpa *file* sumber tanpa memberikan konfirmasi kepada *user* terlebih dahulu. Maka sebaiknya *path stego image* dibuat berlainan dengan path *file cover image*. *Stego image* akan disimpan dalam format *bitmap* secara otomatis. Ketika memilih *path stego image* maka akan dimunculkan gambar seperti berikut ini.



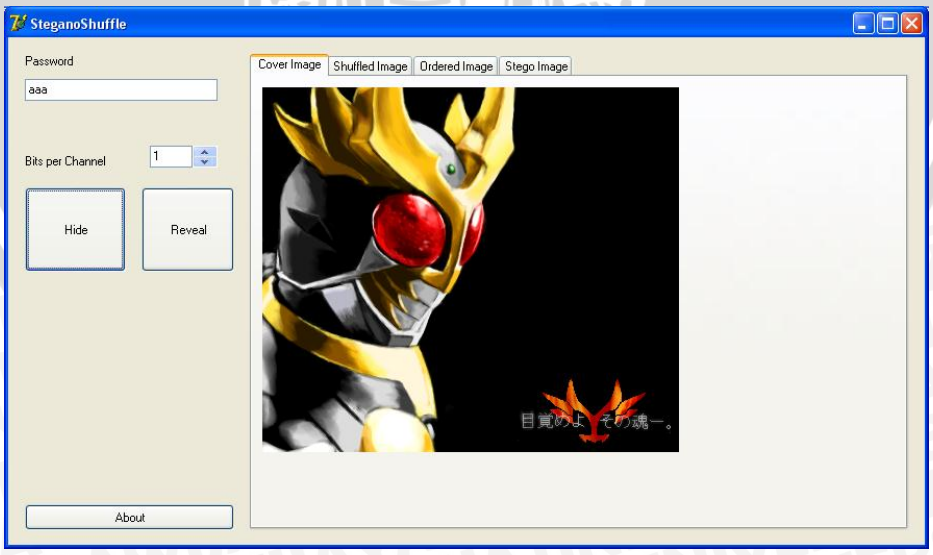
Gambar 4.10. Tampilan Pemilihan Lokasi *Stego Image* Melalui Kotak Dialog

Jika proses penyembunyian data berhasil, maka akan muncul pesan sebagai berikut.

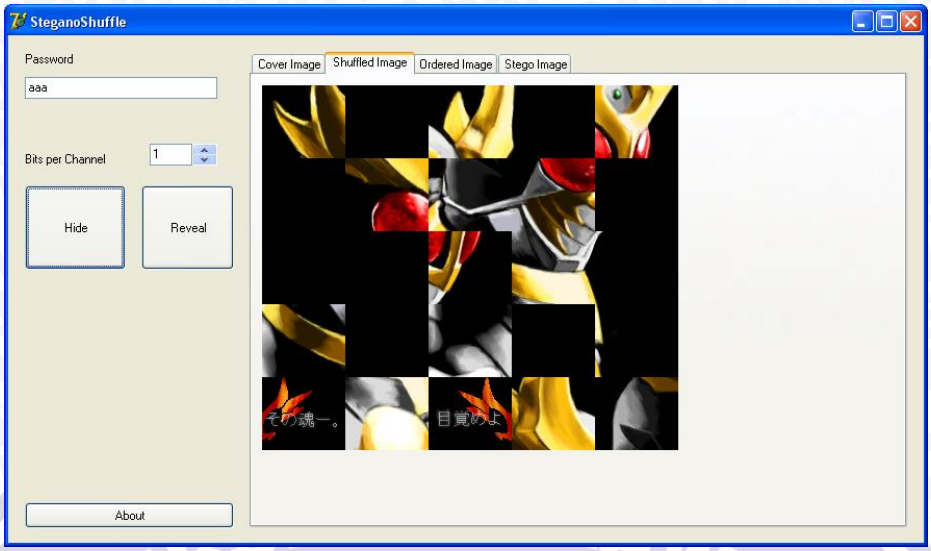


Gambar 4.11. Tampilan Pesan Keberhasilan Proses Penyembunyian Data

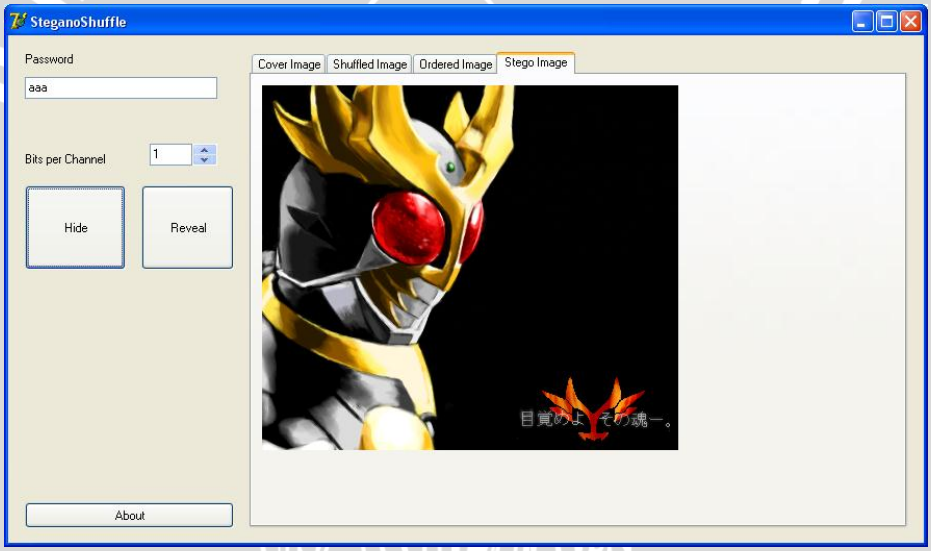
Setelah proses penyembunyian data berhasil, maka *user* dapat melihat tampilan proses pengolahan citra asli (*cover image*) menjadi citra pembawa (*stego image*), mulai dari citra asli, citra hasil *image shuffling*, maupun citra pembawa (*stego image*) yang merupakan hasil dari proses penyisipan data dan *image ordering*.



Gambar 4.12. Tampilan *Cover Image*

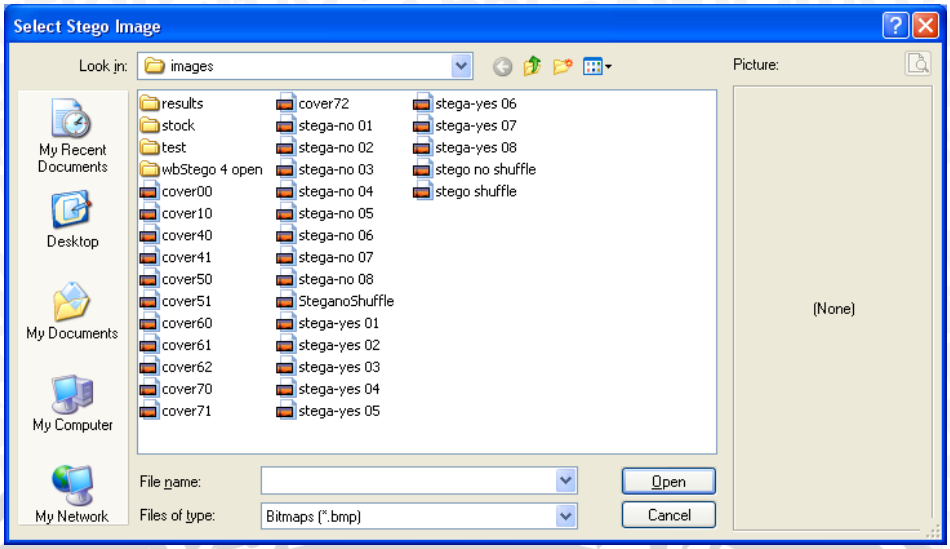


Gambar 4.13. Tampilan *Shuffled Image*



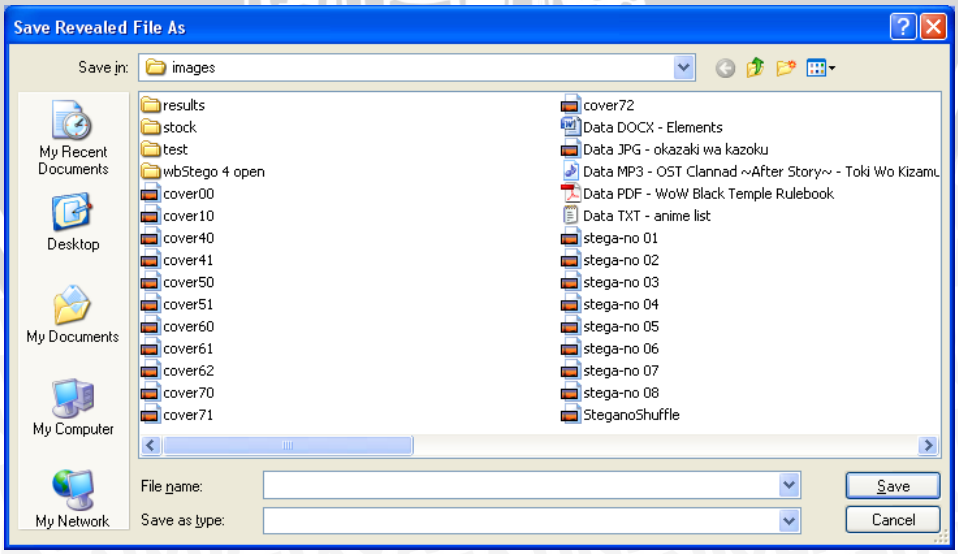
Gambar 4.14. Tampilan *Stego Image*

Sedangkan untuk melakukan pembacaan data, dapat dilakukan dengan mengklik pada tombol "Reveal" untuk memilih sebuah *file stego image*. *File* yang dimunculkan dalam daftar pada kotak dialog hanya file citra dalam format *bitmap.*



Gambar 4.15. Tampilan Pemilihan File *Stego Image* Melalui Kotak Dialog

User kemudian memilih *path* output sebagai *path* keluar *file* yang diproses. Data hasil ekstraksi akan berupa *file* tanpa ekstensi, sehingga tidak dimungkinkan menimpa *file* lainnya meskipun diberi nama dan *path* yang sama dengan *file* lain. Sebagai konsekuensinya, *user* perlu mengganti nama *file* sesuai dengan ekstensi data asli untuk dapat dieksekusi.

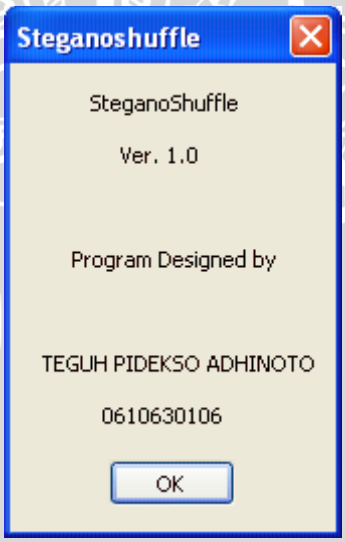


Gambar 4.16. Tampilan Pemilihan Lokasi Data Hasil Ekstraksi

Jika proses pembacaan data berhasil, maka akan muncul pesan sebagai berikut.

Gambar 4.17. Pesan Keberhasilan Proses Pengembalian Data

Untuk menampilkan sebuah form Author *user* dapat mengklik *button* "About". Tampilan bentuk *form* ini diperlihatkan pada gambar berikut ini.



Gambar 4.18. Tampilan *About*

BAB V
PENGUJIAN DAN ANALISIS

Pengujian dilakukan untuk menjamin dan memastikan bahwa sistem yang telah dirancang memiliki tingkat kesalahan yang kecil. Untuk mengetahui apakah sistem bekerja dengan baik dan sesuai dengan perancangan, maka diperlukan serangkaian pengujian.

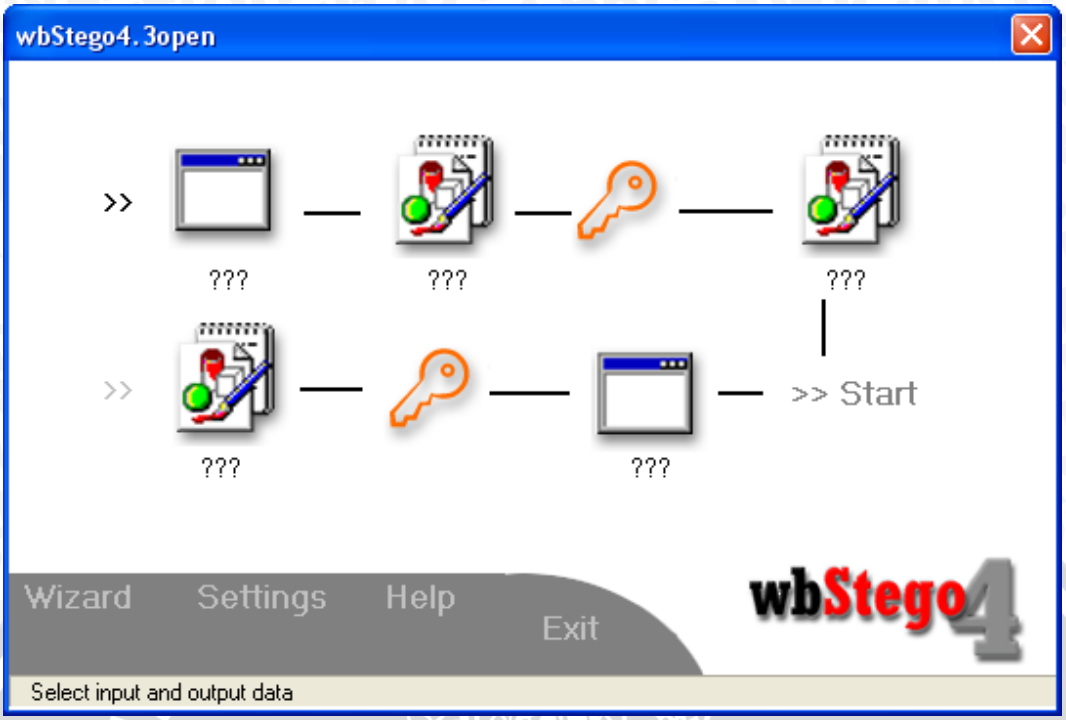
5.1. Pengujian Ketahanan Terhadap *Steganographic Attack*

Pengujian ini dilakukan untuk mengetahui ketahanan metode penyisipan LSB yang dikombinasikan dengan *image shuffling*, dibanding metode penyisipan LSB saja, terhadap steganalisis (teknik pengungkapan keberadaan pesan rahasia pada suatu media pembawa), yang merupakan salah satu bentuk *steganographic attack*.

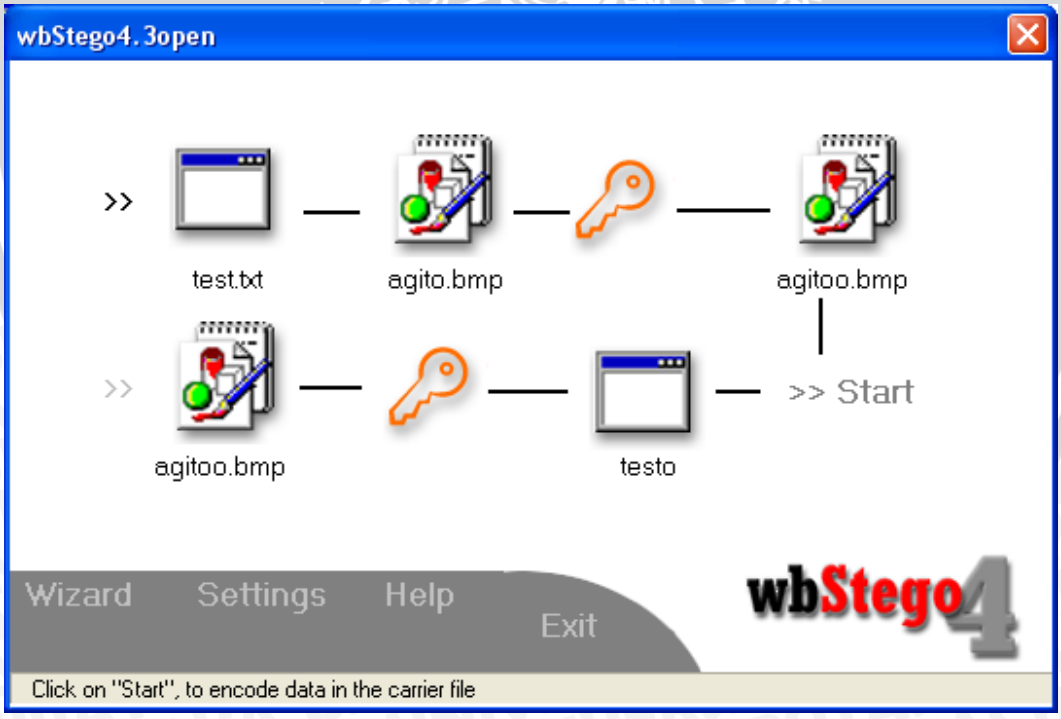
Pengujian ini dilakukan dengan cara menjalankan program steganalisis pada citra pembawa yang hanya menggunakan metode penyisipan LSB dan pada citra pembawa yang menggunakan metode penyisipan LSB dan *image shuffilng*, lalu membandingkan hasil dari kedua citra pembawa tersebut.

Program steganalisis yang digunakan adalah *wbStego4.3.open*, suatu *freeware* yang dapat diunduh di situs wbstego.wbailer.com. Dalam pengujian, citra pembawa hasil penyisipan data program *wbStego4.3.open* dan citra pembawa hasil penyisipan data dengan metode penyisipan LSB dan *image shuffling* didekripsi dengan fitur dekripsi pada program *wbStego4.3.open*.

Gambar 5.1. *wbStego4.3.open*



Gambar 5.2. Antarmuka Program *wbStego4.3.open*



Gambar 5.3. Antarmuka Proses Steganalisis Pada Program *wbStego4.3.open*

Tabel 5.1.

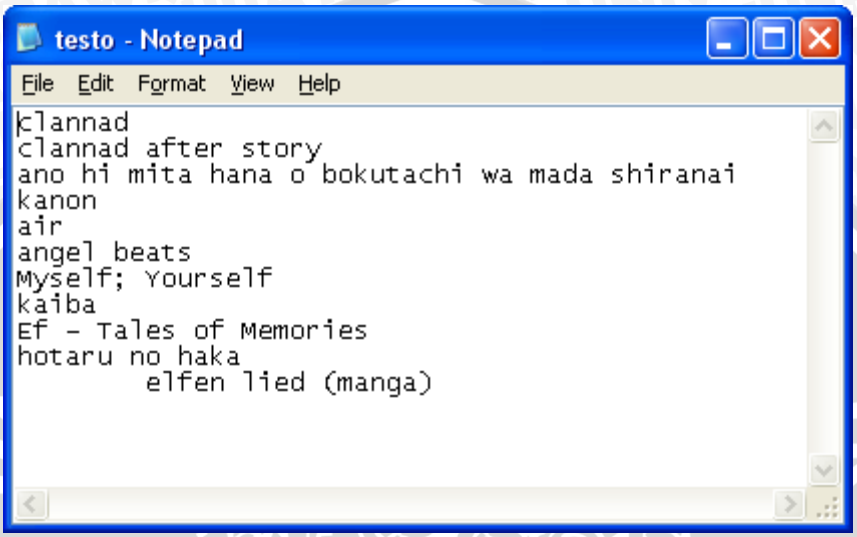
Properti berkas pesan rahasia yang digunakan pada pengujian 5.1., .2.2. Dan 5.2.4.

No.	Properti	Nilai
1	Nama	elements.docx
2	Tipe	MS Word Document
3	Ukuran	25,7 KB

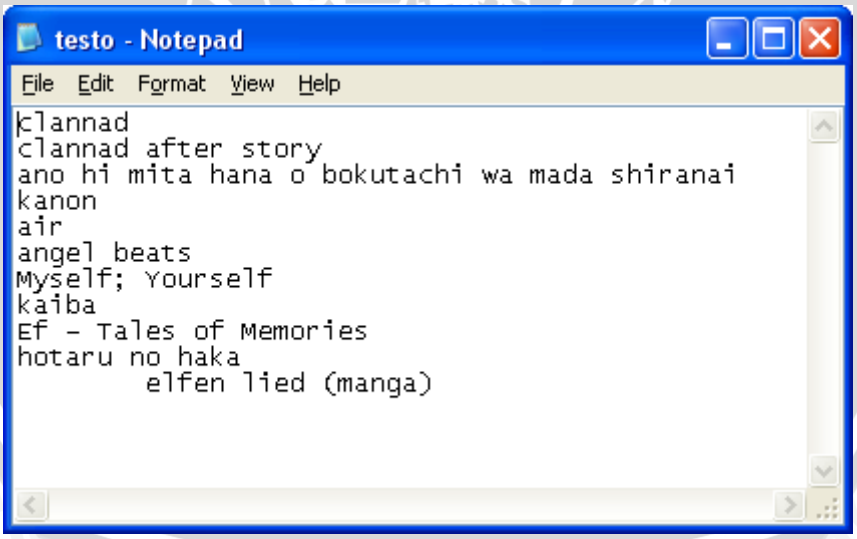
Tabel 5.2. Daftar Citra Pembawa (*Stego Image*)

No.	File Citra	Ukuran (KB)	Dimensi (pixels)
1	 agitoO.bmp	411	400 x 350
2	 agitoX.bmp	411	400 x 350
3	 catduckO.bmp	1.407	800 x 600
4	 catduckX.bmp	1.407	800 x 600
5	 flowerO.bmp	2.305	1024 x 768

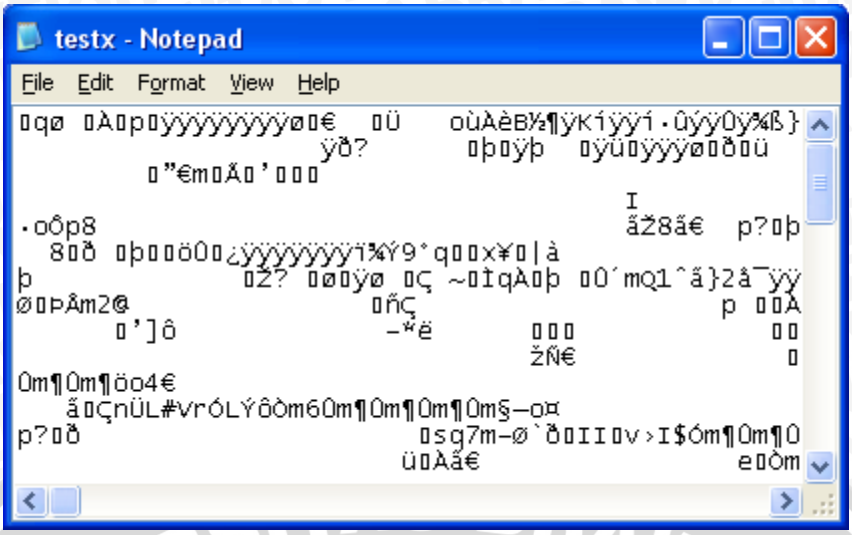
6	 flowerX.bmp	2.305	1024 x 768



Gambar 5.4. Contoh Data Pesan Rahasia



Gambar 5.5. Contoh Hasil Pengujian Steganalisis Pada *File* Yang Menggunakan Metode Penyisipan LSB



Gambar 5.6. Contoh Hasil Pengujian Steganalisis Pada *File* yang Menggunakan Metode Penyisipan LSB dan *Image Shuffling*

Pengujian steganalisis pada *file* yang pesan rahasianya disembunyikan hanya dengan menggunakan metode penyisipan LSB berhasil didekripsi oleh program *wbStego4.3.open*, sedangkan *file* yang pesan rahasianya disembunyikan dengan menggunakan metode penyisipan LSB dan *image shuffling* tidak berhasil didekripsi.

Tabel 5.3. *Test Case* Untuk Pengujian Steganalisis

No.	File Citra	Metode	Keberadaan Pesan Rahasia
1	agitoO.bmp	LSB	Terdeteksi
2	agitoX.bmp	LSB + IS	Tidak terdeteksi
3	catduckO.bmp	LSB	Terdeteksi
4	catduckX.bmp	LSB + IS	Tidak terdeteksi
5	flowerO.bmp	LSB	Terdeteksi
6	flowerX.bmp	LSB + IS	Tidak terdeteksi

Sumber: Pengujian

Berdasarkan hasil pengujian steganalisis yang telah dilakukan dapat dilakukan analisis dengan cara membandingkan hasil yang telah didapatkan antara citra yang hanya menggunakan metode LSB dan citra yang

menggunakan metode LSB dan *image shuffling*. Dari hasil perbandingan dapat ditarik kesimpulan bahwa citra yang menggunakan metode LSB dan *image shuffling* lebih tahan menghadapi steganalisis dibandingkan citra yang hanya menggunakan metode LSB.

5.2. Pengujian Mutu Pada Citra Pembawa (*Stego Image Fidelity*)

Pengujian ini dilakukan untuk mengetahui seberapa banyak mutu citra pembawa turun dibanding mutu citra asli setelah disisipi dengan pesan rahasia.

Pengujian ini terbagi menjadi dua bagian, menggunakan PSNR (*peak signal to noise ratio*) dan menggunakan indera penglihatan manusia, dimana dalam tiap bagian menggunakan dua parameter pengujian, ukuran pesan rahasia dan jumlah penggunaan LSB, sehingga menjadi empat bagian pengujian.

5.2.1. Pengujian Nilai PSNR Dengan Penyisipan Berkas Pesan Rahasia Dengan Ukuran Berbeda

Pada pengujian ini, dilakukan pengujian nilai PSNR (*peak signal to noise ratio*) terhadap berbagai ukuran berkas pesan rahasia yang berbeda.

PSNR sendiri adalah terminologi untuk perbandingan antara kekuatan sinyal maksimum dan kekuatan *noise* yang merusak yang mempengaruhi mutu dari representasinya. Dikarenakan banyak sinyal yang memiliki cakupan dinamis yang luas, umumnya PSNR ditunjukkan dalam skala desibel logaritmik.

PSNR didefiniskan sebagai

$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right)$$
$$= 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \quad \text{.....(1)}$$

MAX_I adalah nilai piksel maksimum suatu citra. Ketika piksel direpresentasikan menggunakan 8 bit per sampel, nilainya adalah 255. Secara umum, ketika sampel direpresentasikan menggunakan PCM linear dengan *B* bit per sampel, nilai MAX_I adalah MAX_I is 2^B-1. Untuk citra berwarna dengan tiga nilai RGB per piksel, definisi PSNR sama kecuali nilai MSE adalah jumlah total selisih nilai pangkat dibagi dengan ukuran citra dikali tiga.

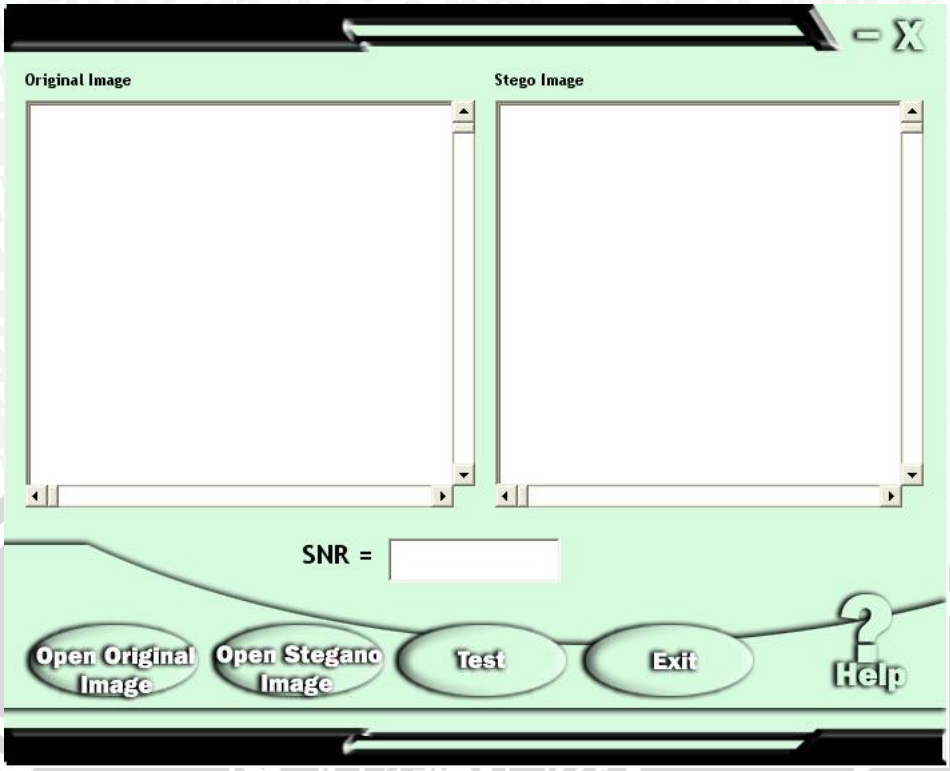
Sedangkan MSE didefiniskan sebagai

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

Nilai PSNR yang kerap ditemui pada kompresi citra dan video yang *lossy* berada di rentang 30 dan 50 dB, dimana semakin besar nilainya maka semakin baik pula mutunya. Ketika dua citra identik, nilai MSE akan sama dengan nol, sehingga nilai PSNRnya tak terdefinisi.

Pengujian ini ditujukan untuk mengetahui pengaruh ukuran berkas pesan rahasia terhadap nilai PSNR yang merupakan representasi digital dari mutu citra pembawa (*stego image fidelity*).

Program penghitung nilai PSNR yang digunakan adalah *Stegano File.* Dalam pengujian, citra pembawa hasil penyisipan data program dibuka menggunakan program *Stegano File* untuk kemudian dihitung nilai PSNR.



Gambar 5.7. Antarmuka Program *Stegano File*

Gambar 5.8.

Berkas Citra Asli (*Cover Image*) Pada Pengujian 5.2.1. Dan 5.2.3.

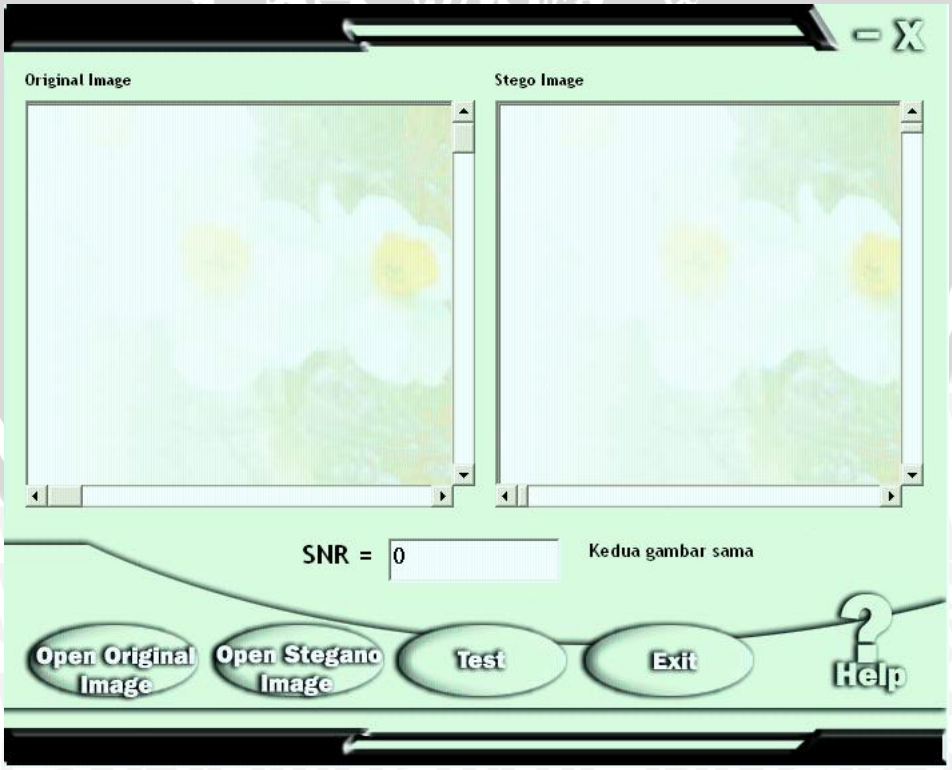
Tabel 5.4.

Properti Berkas Citra Asli Pada Pengujian 5.2.1. Dan 5.2.3.

No.	Properti	Nilai
1	Nama	flower.bmp
2	Tipe	Bitmap image
3	Ukuran	2,3 MB
4	Dimensi	1024 x 768 pixels
5	Resolusi	96 dpi
6	Kedalaman Bit	24

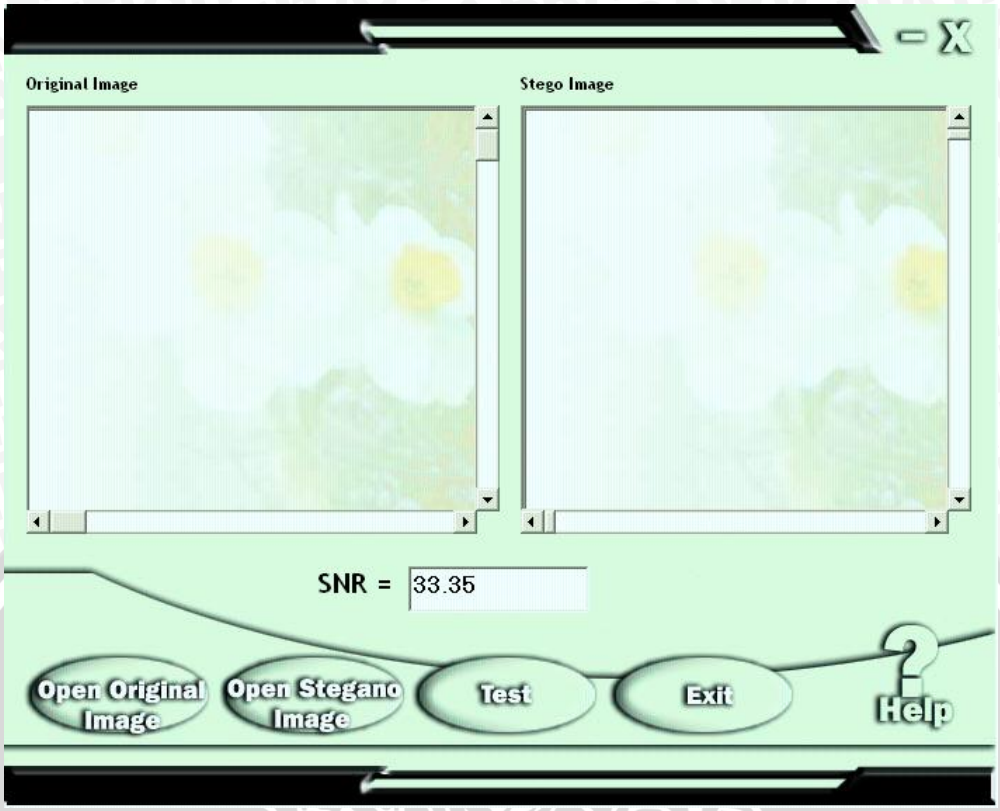
Tabel 5.5. Daftar Pesan Rahasia

No.	Nama Berkas	Tipe	Ukuran
1	elements.docx	Dokumen	26 KB
2	clannad OP.mp3	Audio	703 KB
3	clannad.jpg	Citra	1.111 KB



Gambar 5.9. Contoh Hasil Pengujian PSNR

Pada Dua *File* Citra yang Sama



Gambar 5.10. Contoh Hasil Pengujian PSNR Pada Dua *File* Citra yang Berbeda

Tabel 5.6. *Test Case* Untuk Pengujian Mutu Citra Pembawa Dengan Berbagai Pesan Rahasia

No.	Nama Pesan	Ukuran	Nilai PSNR
1	elements.docx	26 KB	33,35
2	clannad OP.mp3	703 KB	31,77
3	clannad.jpg	1.111 KB	28,43

Sumber: Pengujian

Berdasarkan hasil pengujian nilai PSNR, dapat dilakukan analisis dengan membandingkan nilai PSNR tiap ukuran berkas pesan rahasia.

Dari hasil analisis, dapat ditarik kesimpulan bahwa semakin besar ukuran berkas pesan rahasia maka semakin rendah nilai PSNR, yang berarti mutu citra pembawa juga semakin rendah.

5.2.2. Pengujian Nilai PSNR Dengan Variasi Penggunaan Jumlah LSB Pada Penyisipan Data

Pada pengujian ini, dilakukan pengujian nilai PSNR (*peak signal to noise ratio*) terhadap variasi penggunaan jumlah LSB pada penyisipan data.

Pengujian ini dimaksudkan untuk mengetahui pengaruh jumlah LSB yang digunakan terhadap nilai PSNR yang merupakan representasi digital dari mutu citra pembawa (*stego image fidelity*).

Gambar 5.11.
Berkas Citra Asli (*Cover Image*) Pada Pengujian 5.2.2. Dan 5.2.4.

Tabel 5.7. Properti Berkas Citra Asli Pada Pengujian 5.2.2. Dan 5.2.4.

No.	Properti	Nilai
1	Nama	agito.bmp
2	Tipe	Bitmap image
3	Ukuran	411 KB
4	Dimensi	400 x 350 pixels
5	Resolusi	96 dpi
6	Kedalaman Bit	24

Tabel 5.8. *Test Case* Untuk Pengujian Mutu Citra Pembawa Dengan Variasi Jumlah LSB Yang Digunakan

No.	Nama Pesan	LSB	Nilai PSNR
1	 stego01.bmp	1	45,67
2	 stego02.bmp	2	40,71
3	 stego03.bmp	3	35,63
4	 stego04.bmp	4	30,45
5	 stego05.bmp	5	25,18
6	 stego06.bmp	6	19,87

7	 stego07.bmp	7	14,47
8	 stego08.bmp	8	9,15

Sumber: Pengujian

Serupa dengan hasil pada pengujian nilai PSNR pada ukuran berkas yang berbeda, pada pengiujian ini dapat dilakukan analisis dengan membandingkan nilai PSNR tiap variasi jumlah LSB dari citra pembawa yang digunakan untuk menyisipkan pesan rahasia.

Dari hasil analisis, dapat disimpulkan bahwa semakin banyak jumlah LSB yang digunakan maka semakin rendah nilai PSNR, yang berbanding lurus dengan mutu citra pembawa.

5.2.3. Pengujian Subyektif Terhadap Penyisipan Berkas Pesan Rahasia Dengan Ukuran Berbeda

Pada pengujian ini, dilakukan pengujian subyektif (pengamatan langsung dengan mata telanjang/kacamata) terhadap berbagai citra pembawa dengan ukuran berkas pesan rahasia yang berbeda.

Pengujian ini bertujuan untuk mengetahui pengaruh ukuran berkas pesan rahasia terhadap kesamaran citra pembawa (*stego image imperceptibility*), yaitu kemampuan citra pembawa untuk menyembunyikan keberadaan pesan rahasia dari pengamatan indera manusia.

Responden dari berbagai usia dilibatkan dalam pengujian ini. Hal ini dimaksudkan juga untuk mengetahui pengaruh kemampuan indera

penglihatan manusia terhadap kemampuan mendeteksi keberadaan pesan rahasia pada citra pembawa.

Tabel 5.9. *Test Case* Untuk Pengujian Mutu Citra Pembawa Dengan Berbagai Pesan Rahasia

No.	Pesan	Keberadaan *Noise*	Jumlah Responden
1	elements.docx	Tidak terdeteksi	10
		Tidak yakin	0
		Terdeteksi	0
2	clannad OP.mp3	Tidak terdeteksi	6
		Tidak yakin	4
		Terdeteksi	0
3	clannad.jpg	Tidak terdeteksi	3
		Tidak yakin	3
		Terdeteksi	4

Sumber: Pengujian

Berdasarkan hasil pengujian subyektif, dapat dilakukan analisis dengan membandingkan data hasil pengamatan tiap responden terhadap tiap berkas pesan rahasia yang berbeda ukuran.

Dari hasil analisis, dapat ditarik kesimpulan bahwa semakin besar ukuran berkas pesan rahasia maka semakin jelas *noise* di mata responden pada citra yang merupakan representasi keberadaan pesan rahasia, yang juga berarti mutu citra pembawa semakin rendah.

5.2.4. Pengujian Subyektif Terhadap Variasi Penggunaan Jumlah LSB Pada Penyisipan Data

Pengujian kali ini dilakukan dengan subyektif (pengamatan langsung dengan mata telanjang/kacamata) responden terhadap berbagai citra pembawa dengan penggunaan jumlah LSB yang berbeda.

Pengujian ini bertujuan untuk mengetahui pengaruh penggunaan jumlah LSB terhadap kesamaran citra pembawa (*stego image imperceptibility*).

Tabel 5.10. *Test Case* Untuk Pengujian Mutu Citra Pembawa Dengan Berbagai Pesan Rahasia

No.	Pesan	Keberadaan *Noise*	Jumlah Responden
1	 stego01.bmp	Tidak terdeteksi	10
		Tidak yakin	0
		Terdeteksi	0
2	 stego02.bmp	Tidak terdeteksi	10
		Tidak yakin	0
		Terdeteksi	0
3	 stego03.bmp	Tidak terdeteksi	10
		Tidak yakin	0
		Terdeteksi	0
4	 stego04.bmp	Tidak terdeteksi	5
		Tidak yakin	3
		Terdeteksi	2
5	 stego05.bmp	Tidak terdeteksi	4
		Tidak yakin	2
		Terdeteksi	4
6	 stego06.bmp	Tidak terdeteksi	0
		Tidak yakin	0
		Terdeteksi	10

7	 stego07.bmp	Tidak terdeteksi	0
		Tidak yakin	0
		Terdeteksi	10
8	 stego08.bmp	Tidak terdeteksi	0
		Tidak yakin	0
		Terdeteksi	10

Sumber: Pengujian

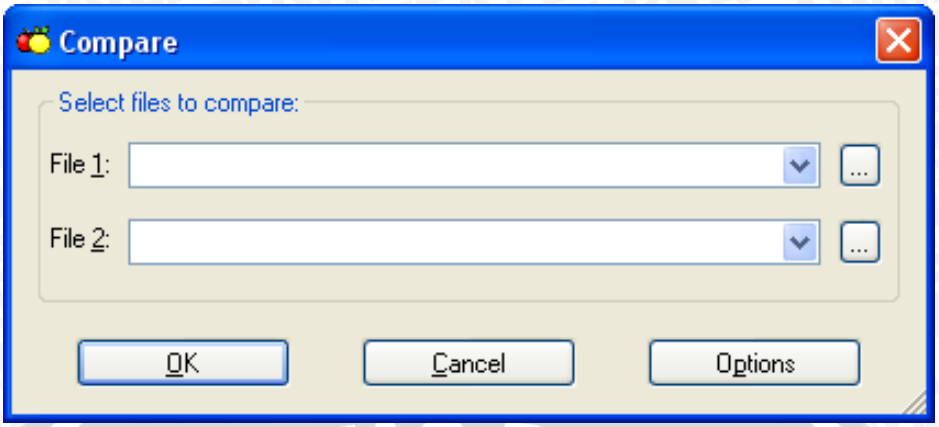
Serupa dengan hasil pada pengujian subyektif terhadap ukuran berkas pesan rahasia yang berbeda, pada pengiujian ini dapat dilakukan analisis dengan membandingkan data hasil pengamatan tiap responden terhadap tiap citra pembawa yang menggunakan jumlah LSB yang bervariasi.

Dari hasil analisis, dapat disimpulkan bahwa semakin banyak jumlah LSB yang digunakan maka keberadaan *noise* –yang merupakan representasi keberadaan pesan rahasia– semakin jelas, yang juga berarti semakin rendahnya mutu citra pembawa.

5.3. Pengujian Perbandingan Data Hasil Ekstraksi Dari Citra Dengan Data Sebenarnya

Pengujian ini dilakukan untuk mengetahui apakah terdapat perbedaan antara data hasil ekstraksi dengan data asli. Perbedaan ini dapat diketahui beberapa parameter, seperti ukuran dan *accesbility* data (data bisa dibuka oleh aplikasi yang sesuai).

Pengujian ini dilakukan dengan cara menjalankan program *file comparator* pada *file* data asli dan *file* data hasil ekstraksi. Program ini membandingkan kedua *file* bit per bit. Program steganalisis yang digunakan adalah *ExamDiff*, suatu *freeware* yang dapat diunduh di situs www.prestosoft.com.



Gambar 5.12. Antarmuka Program *ExamDiff*

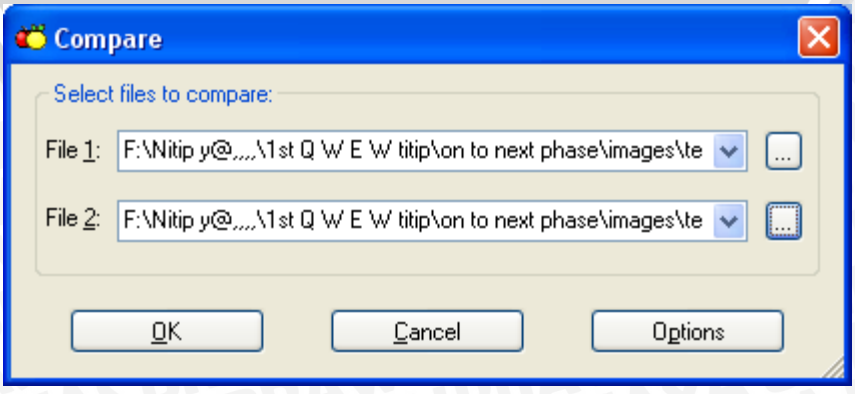
Pengujian ini terbagi menjadi dua bagian, menggunakan ukuran pesan rahasia dan jumlah penggunaan LSB.

5.3.1. Pengujian Perbandingan Dengan Jenis Dan Ukuran Berkas Pesan Rahasia Yang Berbeda

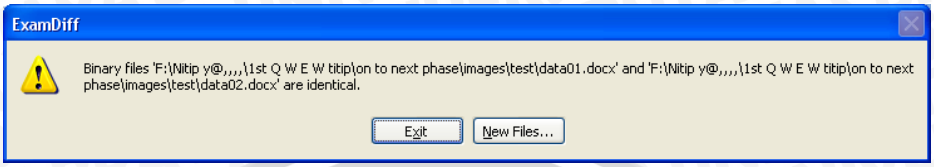
Pengujian ini dilakukan dengan cara membandingkan ukuran dan *accesbility* data hasil ekstraksi dengan data asli.

Tabel 5.11. Daftar Data Hasil Ekstraksi

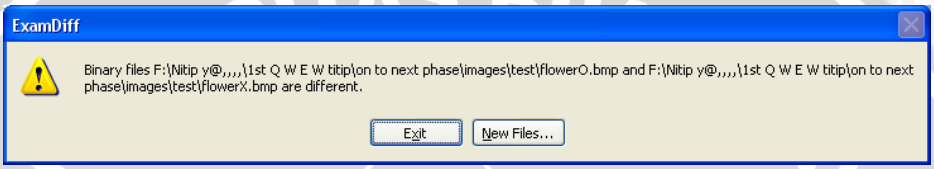
No.	Data Hasil Ekstraksi	Data Asli
1	docx.docx	elements.docx
2	mp3.mp3	clannad OP.mp3
3	jpg.jpg	clannad.jpg



Gambar 5.13. Antarmuka Proses *File Comparation*
Pada Program *ExamDiff*



Gambar 5.14. Contoh Hasil Pengujian
Pada Dua *File* Yang Sama/Identik



Gambar 5.15. Contoh Hasil Pengujian Pada Dua *File* Yang
Berbeda/Tidak Identik

Tabel 5.12.
***Test Case* Perbandingan Data Hasil Ekstraksi Dengan Data Asli**

No.	Data Ekstraksi	Hasil Komparasi
1	docx.docx	Identik
2	mp3.mp3	Identik
3	jpg.jpg	Identik

Sumber: Pengujian

Berdasarkan hasil pengujian, dilakukan analisis dengan membandingkan hasil ekstraksi dari tiap jenis dan ukuran berkas pesan rahasia.

Setelah dianalisis, dapat disimpulkan bahwa jenis dan ukuran berkas pesan rahasia tidak berpengaruh terhadap data hasil ekstraksi, atau dengan kata lain data hasil ekstraksi sama persis dengan data asli.

5.3.2. Pengujian Perbandingan Dengan Variasi Penggunaan LSB

Pengujian ini dilakukan dengan cara membandingkan ukuran dan *accesbility* data hasil ekstraksi dengan data asli.

Tabel 5.13. Daftar Data Hasil Ekstraksi Dengan Lsb Berbeda

No.	Data Hasil Ekstraksi	LSB
1	data01.docx	1
2	data02.docx	2
3	data03.docx	3
4	data04.docx	4
5	data05.docx	5
6	data06.docx	6
7	data07.docx	7
8	data08.docx	8

Tabel 5.14.

***Test Case* Perbandingan Data Hasil Ekstraksi LSB Dengan Data Asli**

No.	Data Ekstraksi	Hasil Komparasi
1	data01.docx	Identik
2	data02.docx	Identik
3	data03.docx	Identik
4	data04.docx	Identik
5	data05.docx	Identik
6	data06.docx	Identik
7	data07.docx	Identik
8	data08.docx	Identik

Sumber: Pengujian

Melalui pengujian, diperoleh data sebagaimana di atas dan dapat dilakukan analisis dengan membandingkan hasil ekstraksi dari tiap citra pembawa dengan variasi jumlah LSB yang digunakan.

Setelah dianalisis, dapat disimpulkan bahwa jumlah LSB yang digunakan pada citra pembawa tidak berpengaruh terhadap data hasil ekstraksi, atau dengan kata lain data hasil ekstraksi sama persis dengan data asli.

BAB VI
PENUTUP

6.1. Kesimpulan

Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan, implementasi dan pengujian aplikasi telah selesai dilakukan. Kesimpulan diambil dari hasil pengujian dan analisis terhadap aplikasi yang dibangun. Tahap terakhir dari penulisan adalah saran yang dimaksudkan untuk memperbaiki kesalahan-kesalahan yang terjadi dan menyempurnakan penulisan serta untuk memberikan pertimbangan atas pengembangan aplikasi selanjutnya.

Berdasarkan hasil perancangan, implementasi, dan pengujian maka dapat diambil kesimpulan sebagai berikut:

a. Metode penyisipan LSB yang dikombinasikan dengan *image shuffling* terbukti lebih tahan terhadap steganalisis, yang merupakan salah satu bentuk *steganographic attack*, dibandingkan dengan metode penyisipan LSB standar.
b. Semakin besar ukuran data atau ukuran berkas pesan rahasia yang disisipkan, maka mutu dan kesamaran citra pembawa (*stego image fidelity and imperceptibility*) semakin rendah.
c. Semakin banyak jumlah LSB pada citra pembawa yang digunakan, maka mutu dan kesamaran citra pembawa (*stego image fidelity and imperceptibility*) semakin rendah.
d. Jenis maupun ukuran data atau berkas pesan rahasia yang disisipkan tidak mempengaruhi kualitas data hasil ekstraksi.
e. Jumlah LSB pada citra pembawa yang digunakan tidak mempengaruhi kualitas data hasil ekstraksi.
f. Data atau berkas pesan rahasia tidak mengalami perubahan kualitas setelah melalui proses penyisipan dan ekstraksi.

6.2. Saran

Dalam perancangan dan pengimplementasian perangkat lunak ini, masih terdapat kekurangan dan kelemahan, sehingga masih diperlukan adanya penyempurnaan dalam rangka pengembangan kedepan. Adapun saran yang dapat diberikan antara lain:

a. Untuk pengembangan lebih lanjut, dapat menggunakan dasar algoritma yang lebih tahan dari serangan kriptografi.
b. Untuk meningkatkan variasi citra pembawa, pengembangan lebih lanjut dapat menggunakan seluruh jenis *file* citra.
c. Untuk meningkatkan kapasitas penyimpanan data pada citra pembawa, pengembangan lebih lanjut dapat menggunakan lebih dari satu *file* citra untuk menampung pesan rahasia yang memiliki rasio ukuran cukup besar dengan citra pembawa.

DAFTAR PUSTAKA

Gonzalez, Rafael C., Woods, Richard C. 2002. *Digital Image Processing.* Prentice-Hall Inc.

Cummins, Jonathan., Diskin, Patrick., Lau, Samuel., Parlett, Robert. 2004. *Steganography and Digital Watermarking.* The University of Birmingham.

Johnson, Neil F., Jajodia, Sushil. 1998. *Exploring Steganography: Seeing the Unseen*. George Mason University.

Masaleno, Andino, 2003, *Pengantar Steganografi*, IlmuKomputer.com, diakses pada 5 Mei 2011.

Hartono, Samuel, 2005, *Pengembangan Metode Steganography Untuk Penyembunyian Data di Dalam Citra Digital dengan Menggunakan Metode LSB dan Cross Dissolve*, Teknik Informatika, Universitas Kristen Petra Surabaya.

Sukmawan Budi, 1998, *RC4 STREAM CIPHER,budiSukmawan.blogspot.com,* diakses pada 5 Mei 2011.

Wikipedia, 2011, *RC4*, diakses 5 Mei 2011.