TUGAS AKHIR - CI 1599

IMPLEMENTASI ALGORITMA GENETIKA UNTUK STEGANOGRAFI PADA CITRA JPEG

ARIF HIDAYAT
NRP 5103 100 026

Dosen Pembimbing I
Rully Soelaiman, S.Kom., M.Kom.

Dosen Pembimbing II
Mediana Aryuni, S.Kom., M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2008

FINAL PROJECT - CI 1599

IMPELEMENTATION OF GENETIC ALGORITHM FOR STEGANOGRAPHY IN JPEG IMAGE

ARIF HIDAYAT
NRP 5103 100 026

First Supervisor
Rully Soelaiman, S.Kom., M.Kom.

Second Supervisor
Mediana Aryuni, S.Kom., M.Kom.

DEPARTMENT OF INFORMATICS
Faculty of Information Technology
Sepuluh Nopember Institute of Technology
Surabaya 2008

IMPLEMENTASI ALGORITMA GENETIKA UNTUK STEGANOGRAFI PADA CITRA JPEG

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Sistem Bisnis Cerdas
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
ARIF HIDAYAT
Nrp. 5103 100 026

Disetujui oleh Tim Pembimbing Tugas Akhir:

1. Rully Soelaiman, S.Kom., M.Kom. …….. (Pembimbing I)

2. Mediana Aryuni, S.Kom., M.Kom. ……. (Pembimbing II)

SURABAYA
JANUARI, 2008

ABSTRAK

Steganografi adalah metode untuk menyimpan suatu informasi ke dalam citra sedemikian hingga informasi tersebut bisa tersembunyi dengan aman dan isi informasi hanya bisa diketahui oleh penerima yang diinginkan. Standar yang diinginkan dalam steganografi adalah mampu untuk menyimpan informasi (payload) sebanyak mungkin dalam citra pembawa (carrier) tanpa mengurangi kualitas citra dan dengan cara hingga citra hasil steganografi (package) tidak bisa dideteksi dengan mata manusia atau yang disebut dengan steganalysis.

Untuk menjaga kerahasiaan pesan digunakan pesudo-random number generator (PRNG) untuk mencari dan memilih secara pseudo-random nilai discrete cosine transform (DCT) yang akan digantikan dengan pesan yang akan disembunyikan. Karena pemilihan ini bersifat pseudo-random, informasi yang tersembunyi tidak bisa ditemukan tanpa mengetahui kunci rahasia yang digunakan sebagai seed pada PRNG. Algoritma Genetika akan digunakan untuk melakukan optimasi dengan yang menghasilkan citra steganografi yang memiliki tingkat perbedaan paling rendah bila dibandingkan dengan citra asal.

Dari implementasi Algoritma Genetika untuk steganografi ini dihasilkan citra dengan kualitas visual dan distorsi yang lebih kecil bila dibandingkan dengan citra yang dihasilkan dengan steganografi tanpa menggunakan Algoritma Genetika.

Kata kunci: Steganografi, discrete cosine transform, JPEG, Algoritma Genetika

ABSTRACT

A Steganography is a method to hide an information inside an image so that the information is hidden securely and its content is only known to intended recepient. Desired standard for steganography are capability to hide information (payload) as many as possible inside medium image (carrier) without reducing image quality and in a way that result image (package) cannot be detected by human visual system, or a method called steganalysis.

In order to preserve the secret information, we use pesudo-random number generator (PRNG) to find and select pseuo-randomly value of discrete cosine transform (DCT) which will be embedded with the hidden information. Because the selection is pseudo-random, the hidden information cannot be found without knowing secret key used for seeding PRNG. Genetic Algorithm will be used to improve result image's quality so that it has minimum differences when compared with the original images.

From this implementation of Genetic Algorithm for steganography, resulted in an image which has better visual quality and less distortion than image that is produced from steganography without Genetic Algorithm.

Keywords: Steganography, Discrete Cosine Transform, JPEG, Genetic Algorithm

KATA PENGANTAR

Segala puji bagi Allah SWT yang atas kehendak-Nya penyusunan tugas akhir berjudul "***Implementasi Algoritma Genetika untuk Steganografi pada Citra JPEG"*** ini bisa terlaksana.

Pada kesempatan ini, penulis mengucapkan rasa terima kasih yang sebesar-sebesarnya kepada:

1. Bapak Rully Soelaiman, S.Kom.,M.Kom. dan Ibu Mediana Aryuni, S.Kom.,M.Kom atas inspirasi dan arahan beliau selama penyusunan tugas akhir ini sebagai pembimbing.
2. Bapak, Ibu serta segenap keluarga penulis di Bojonegoro atas bantuan doa, dorongan semangat dimanapun dan kapanpun.
3. Seluruh teman di Teknik Informatika baik angkatan 2003, junior dan senior yang telah membantu hinggga suksesnya Tugas Akhir ini.
4. Segenap dosen dan staf TU yang telah memberikan ilmu, fasilitas, bantuan dan kemudahan kepada penulis selama menjalankan kuliah di Informatika ITS.
5. Para Administrator Laboratorium Pemrograman (LP), Grid Computing Laboratory (GCL), Intelligence Bussiness Sytem (IBS) dan Lab Rekayasa Perangkat Lunak (RPL) yang telah membantu saya, terutama menyediakan fasilitas, dalam proses pengerjaan TA ini.
6. Para kru ITS Online atas semangat dan bantuanya selama pengerjaan Tugas Akhir ini.
7. Serta semua pihak yang tidak dapat disebutkan satu persatu yang telah banyak memberikan berbagai macam bantuan.

Tugas akhir ini jauh dari kesempurnaan, oleh karena itu sangat diharapkan kritik dan saran yang membangun untuk perbaikan kedepan.

Surabaya, Januari 2008

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1

PENDAHULUAN

1.1 Pendahuluan

Steganografi adalah metode untuk menyimpan suatu informasi ke dalam citra sedemikian hingga informasi tersebut bisa tersembunyi dengan aman dan isi informasi hanya bisa diketahui oleh penerima yang diinginkan.

Standar yang diinginkan dalam steganografi adalah mampu untuk menyimpan informasi (*payload*) sebanyak mungkin dalam citra pembawa (*carrier*) tanpa mengurangi kualitas citra dan dengan cara sedemikian hingga citra hasil steganografi (*package*) tidak bisa dideteksi baik dengan mata manusia ataupun dengan analisa statistik, atau yang disebut dengan steganalysis..

Untuk menjaga kerahasiaan pesan digunakan *pseudo-random number generator* (PRNG) untuk mencari dan memilih secara pseudo-random nilai *discrete cosine transform* (DCT) yang akan digantikan dengan pesan yang akan disembunyikan. Karena pemilihan ini bersifat *pseudo-random*, informasi yang tersembunyi tidak bisa ditemukan tanpa mengetahui kunci rahasia yang digunakan sebagai *seed* pada PRNG.

Untuk meningkatkan kualitas citra, digunakan Algoritma Genetika untuk melakukan optimasi dengan yang menghasilkan citra steganografi yang setelah disisipi dengan bit dari pesan yang disembunyikan akan memiliki tingkat perbedaan paling rendah bila dibandingkan dengan citra asal. Dengan kualitas citra yang lebih baik maka proses steganografi akan lebih aman.

Beberapa variabel yang terlibat dalam steganografi akan diamati untuk menentukan konfigurasi optimal yang dapat menghasilkan citra steganografi dengan kualitas visual yang terbaik.

1.2 Permasalahan

Permasalahan yang akan dibahas dalam tugas akhir ini adalah :

1. Bagaimana menyembunyikan informasi ke dalam citra secara aman. Informasi yang disimpan di dalam citra harus hanya bisa dambil (di-*retrieve*) oleh penerima (*recepient*) yang diinginkan.
2. Bagaimana agar citra yang dihasilkan tidak bisa dideteksi dengan serangan visual. Penglihatan manusia harus tidak bisa membedakan antara citra asal dan citra citra yang di dalamnya sudah disisipi informasi.

1.3 Batasan Tugas Akhir

Sejumlah permasalahan yang dibahas dalam tugas akhir ini akan dibatasi ruang lingkup pembahasannya, antara lain:

1. Tugas akhir ini menggunakan Algoritma Genetika untuk melakukan optimasi proses steganografi yang berdasarkan pada algoritma OutGuess yang menggunakan *pseudo-random number generator* untuk melakukan seleksi konstanta DCT
2. Citra yang digunakan sebagai medium stegaografi (*cover image*) adalah citra dengan format JPEG. Citra JPEG dipilih karena memiliki karakteristik kualitas yang baik namun ukuran yang relatif kecil. Citra JPEG juga adalah salah satu format citra yang paling banyak digunakan sehingga cocok untuk digunakan menyimpan informasi tersembunyi.
3. Pengujian citra dilakukan dengan mengetes kualitas citra secara visual dengan menghitung *Peak Signal to Noise Ratio* (PSNR) untuk menghitung distorsi yang terjadi pada citra hasil steganografi bila dibandingkan terhadap citra asal.
4. Pengembangan aplikasi menggunakan sistem operasi Windows dan bahasa pemrograman Java.

1.4 Tujuan Pembuatan Tugas Akhir

Tujuan dari pembuatan Tugas Akhir ini adalah mengimplementasikan Algoritma Genetika untuk meningkatkan kualitas citra steganografi sehingga lebih tahan terhadap steganalisis baik secara visual.

1.5 Metodologi

Metodologi yang digunakan dalam Tugas Akhir ini adalah sebagai berikut:

1. Pemahaman Permasalahan dan Studi Literatur

Mempelajari dan memahami makalah yang dijadikan acuan penyusunan Tugas Akhir ini dan beberapa sumber lain yang terhubung dengan steganografi yang dijadikan rujukan pada makalah utama. Tahap ini melingkupi kegiatan pencarian literatur baik melalui media internet maupun dengan melakukan studi pustaka pada jurnal-jurnal ilmiah cetak. Setelah mendapatkan literatur yang dibutuhkan maka dilakukan penelaahan yang menyeluruh pada literatur tersebut.

Proses studi literatur ini meliputi tahap-tahap belajar sebagai berikut:

a. Pemahaman tentang format citra *Joint Photographic Experts Group* (JPEG).
b. Pemahaman tentang konsep domain frekuensi dan *Discrete Cosine Transform* (DCT).
c. Pemahaman terhadap *stream cipher* RC4.
d. Pemahaman terhadap konsep steganografi
e. Pemahaman terhadap konsep Algoritma Genetika
f. Pemahaman terhadap uji kualitas citra dengan *Peak Signal to Noise Ratio* (PSNR).

2. Perancangan Aplikasi

Tahap ini merupakan tahapan analisis dan desain aplikasi yang akan dikembangkan dengan mengacu pada literatur-literatur yang telah dikumpulkan dan dipelajari. Tahap ini meliputi perancangan mesin utama aplikasi, perancangan data baik untuk proses pelatihan dan pengujian serta perancangan skenario latih dan uji. Petimbangan optimasi akan sangat diperhatikan pada tahap ini mengingat

optimasi adalah tujuan utama dalam pengusulan pemakaian Algoritma Genetika untuk steganografi.

3. Pembuatan aplikasi

Pada tahap ini akan dilakukan proses implementasi perancangan aplikasi yang akan dikembangkan menggunakan bahasa pemrograman Java sesuai dengan hasil perancangan pada tahap sebelumnya.

4. Uji coba dan evaluasi

Melakukan ujicoba aplikasi untuk melihat permasalahan-permasalahan yang timbul selama proses uji, mengevaluasinya dan mengadakan perbaikan jika diperlukan.

5. Analisis hasil uji coba dan evaluasi

Pada tahap ini dilakukan pengkajian dan analisis keluaran dari perangkat lunak yang telah dibuat sebelumnya. Tahap analisis meliputi penentuan batas-batas optimal citra masukan maupun parameter-parameter lain yang dibutuhkan oleh aplikasi serta pengambilan simpulan-simpulan seputar implementasi algoritma pada aplikasi.

6. Pembuatan dokumentasi tugas akhir

Pada tahap terakhir ini disusun buku sebagai dokumentasi dari pelaksanaan Tugas Akhir. Dokumentasi selama pelaksanaan tugas akhir ini dirasa perlu agar tugas akhir ini bisa menjadi rujukan untuk pengembangan-pengembangan proses steganografi pada citra lain.

1.6 Sistematika Penulisan

Uraian sistematika penulisan pada tugas akhir ini bertujuan agar perancangan dan pelaksanaan implementasi Algoritma Genetika untuk optimalisasi steganografi pada citra JPEG yang dibahas menjadi mudah dipahami, jelas dan sistematis untuk tiap-tiap bab atau sub bahasan.

Bab 1 berisi gambaran umum yang membahas latar belakang, permasalahan yang dihadapi, batasan-batasan dan juga tujuan yang akan dicapai pada akhir pengerjaan Tugas Akhir ini.

Bab 2 merupakan tinjauan pustaka dari teori-teori yang mendasari Tugas Akhir ini meliputi pengantar proses pemilihan komponen utama, konsep DCT, proses steganografi hingga konsep Algoritma Genetika..

Bab 3 mendeskripsi sistem secara umum perancangan data, pemodelan proses perhitungan DCT dan diagram alir algoritma steganografi.

Bab 4 membahas proses implemetasi steganografi dan optiomasi dengan algoritma gentika serta analisis hasilnya.

Bab 5 merupakan simpulan dan saran untuk pengembangan selanjutnya.

(halaman ini sengaja dikosongkan)

BAB 2

TINJAUAN PUSTAKA

Pada bab ini akan diuraikan mengenai dasar-dasar teori yang digunakan dalam pengerjaan tugas akhir ini. Teori yang dibahas meliputi format citra JPEG, kosep domain frekuensi DCT, konsep steganografi, konsep *stream cipher* RC4 dan algoritma genetika.

2.1 Format Citra JPEG

Joint Photographic Experts Group (JPEG) adalah format citra yang banyak digunakan untuk menyimpan citra. JPEG menyediakan kompresi citra secara *lossy* (dengan menghilangkan data citra kurang penting) ataupun *lossless* (tanpa menghilangkan data).

Tahapan *encoding* JPEG bisa melibatkan proses:

- *Color space transformation*, merubah citra dari color sample RGB menjadi YCbCr,
- *Chroma subsampling*, menurunkan tingkat Cr dan Cb pada citra terhadap Y. Citra JPEG dapat menggunakan rasio YCbCr 4:4:4 (tanpa *Subsampling*), 4:2:0 atau 4:2:2.
- *Block splitting*, memecah citra menjadi blok dengan dimensi 8x8. Bila dimensi citra tidak genap kelipatan 8 maka blok harus diisi dengan data *dummy*.
- *Discrete cosine transformation.* Proses ini merubah citra JPEG dari domain spasial menjadi domain frekuensi dengan DCT.
- *Quantization*. kuantisasi dari nilai DCT tiap blok untuk mengurangi jumlah informasi yang disimpan.
- *Entropy coding*, dilakukan secara zig-zag pada konstanta hasil kuantisasi DCT citra.

Langkah-langkah perhitungan pada format citra JPEG adalah sebagai berikut:

A. Melakukan konversi ruang warna dari RGB ke YCbCr**.**

Pertama citra perlu di konversi ke ruang warna YCbCr dari RGB. Ini karena YCbCr menyimpan data warna berdasarkan *luminance* dan *chroma*. Sistem penglihatan mata manuasia (*Human Visual System*, HVS) sangat peka terhadap perbedaan *luminance* tapi kurang peka terhadap perbedaan *chrominance*. Sehingga informasi kroma yang disimpan dalam citra dapat dikurangi jika ingin melakukan kompresi. Dengan demikian dapat dilakukan kompresi dengan melakukan *chroma subsampling* terhadap data *chrominance*.

Rumus yang digunakan dalam perhitungan konversi RGB ke YCbCr ini adalah:

$$\begin{aligned} y &= 16 + (\frac{64{,}481 \times R}{255}) + (\frac{128{,}553 \times G}{255}) + (\frac{24{,}966 \times B}{255}) \\ cb &= 128 - (\frac{37{,}797 \times R}{255}) + (\frac{74{,}203 \times G}{255}) + (\frac{112 \times B}{255}) \\ cr &= 128 - (\frac{112 \times R}{255}) + (\frac{93{,}768 \times G}{255}) + (\frac{18{,}214 \times B}{255}) \end{aligned} \quad (2.1)$$

Ruang warna YCbCr ini akan dirubah kembali menjadi ruang warna RGB pada saat decoding citra JPEG. Perubahan ruang warna YCbCr menjadi RGB ini mengikuti rumus:

$$\begin{aligned} R &= 255 \times ((0{,}004566 \times y) + (0{,}00625893 \times cr)) \\ G &= 255 \times ((0{,}04566 \times y) - (0{,}0015363 \times cb) \\ &\quad - (0{,}003881 \times cr)) \\ B &= 255 \times ((0{,}00456621 \times y) + (0{,}00791071 \times cr)) \end{aligned} \quad (2.2)$$

Dengan konversi ke ruang warna YCbCr ini informasi yang semula tersebar merata di dalam ruang warna RGB akan terkonsentasi ke nilai *luminance*. Nilai *chrominance* (Cb dan Cr) relatif lebih sulit dibedakan oleh mata manusia. Berikut adalah contoh dari konversi dari ruang warna RGB menjadi ruang warna YCbCr:

Gambar 2-1 Citra pada ruang warna RGB

Gambar 2-2 Citra pada ruang warna YCbCr – Layer Y

Gambar 2-3 Citra pada ruang warna YCbCr – Layer Cb

Gambar 2-4 Citra pada ruang warna YCbCr – Layer Cr

Tabel 2-1 Contoh konversi ruang warna RGB ke YCbCr

R	G	B		Y	Cb	Cr
12	255	5		148,12	54,21	39,13
14	254	12		148,82	57,28	39,87
16	253	19		149,51	60,35	40,62
18	252	26		150,21	63,42	41,37
20	251	33		150,90	66,49	42,11
22	250	40		151,60	69,56	42,86
24	249	47		152,29	72,63	43,60
26	248	54		152,99	75,70	44,35
28	247	61		153,68	78,77	45,10
30	246	68		154,38	81,84	45,84

B. *Chorma Subsampling*.

Chroma subsampling adalah tahapan dalam ecoding citra atau video dimana resolusi *chrominace* dikecilkan terhadap resolusi *luminance*. *Chorma subsampling* digunakan dalam banyak *encoding* terutama pada JPEG dan video.

Karena batasan penyimpanan dan transmisi terkadang diperlukan pengurangan terhadap informasi citra yang disimpan. Salah satu cara untuk melakukan ini adalah dengan melakukan *chroma subsampling*. Ini karena sistem penglihatan manusia lebih peka terhadap variasi gelap terang (*luminance*) dari pada terhadap variasi warna (*chroma*). Dengan demikian informasi warna dalam ruang warna YCbCr bisa dikurangi untuk mengurangi waktu transmisi dan ukuran penyimpanan yang diperlukan tanpa secara visual mengurangi kualitas citra secara berarti.

Sampling dari YCbCr in dituliskan dalam notasi logical. notasi tersebut memiliki aturan:

- angka pertama menunjukkan tingkat *luminance*
- notasi kedua menunjukkan faktor Cb dam Cr terhadap Y secara horisontal
- notasi ketiga menunjukkan faktor Cb dan Cr relatif terhadap Y secara vertikal

- notasi keempat, bila ada menunjukkan komponen alpha atau transparansi citra

Subsampling yang banyak dipakai adalah 4:4:4, 4:2:2 dan 4:2:0. Perbedaan dari *subsampling* berikut adalah:

- 4:4:4 YCbCr

  Pada tiap-tiap ketiga komponen YCbCr, terdapat sample rate yang sama. Skema *sampling* ini biasanya digunakan dalam citra atau video berkualitas tinggi. Dengan skema ini tidak ada informasi kroma yang dihilangkan

- 4:2:2 YCbCr

  Pada skema ini untuk kroma biru dan merah (Cb dan Cr) disampling pada tingkat separuh secara horisontal dari *sampling* dari *luminansi* (Y). Ini mengurangi besar *bandwith* hingga separuh dari yang dibutuhkan bila tidak di-*subsampling*. Skema ini banyak digunakan dalam citra JPEG dan video Digital Betacam, DVCPRO50 dan DVCPRO HD, Digital-S, CCIR 601 / Serial Digital Interface / D1, dan ProRes 422.

- 4:2:0 YCbCr

  Pada skema ini Cb dan Cr masing-masing disampling dengan tingkat separuh dari *luminance* pada horisontal dan vertikal. Ada tiga variasi penempatan Cb dan Cr hasil *subsampling* ini

  - Pada MPEG-2, Cb dan Cr diletakkan secara horisontal.
  - Pada JPEG/JFIF, H.261, dan MPEG-1, Cb dan Cr diletakkan secara *interstitially* (separuh jalan diantara *sample luminance*).
  - Pada 4:2:0 DV, Cb dan Cr diletakkan bergantian tiap baris.

C. Menghitung *Discrete Cosine Transform*

Discrete cosine transform (DCT) adalah suatu representasi citra dalam domain frekuensi sebagai nilai penjumlahan sinususoid dengan frekuensi dan magnitudo yang bervariasi. DCT mirip dengan *discrete Fourier transform* (DFT) tapi hanya mengggunakan bilangan real. DCT memiliki delapan standar variasi

tipe DCT tapi yang paling banyak digunakan adalah tipe kedua. Pada satu dimensi perhitungan DCT tipe kedua ini adalah:

$$X_n = \sum_{n=0}^{n-1} x_n \cos\frac{\pi}{N}\left(n+\frac{1}{2}\right)k,\ 0 \le k \le N-1 \qquad (2.3)$$

DCT memiliki sifat, untuk sebagian besar citra, informasi dikumpulkan pada beberapa koefisien DCT saja yang terletak diujung kiri atas. Ini memungkinkan DCT digunakan untuk melakukan kompresi citra. DCT ini menjadi bagian utama kompresi yang digunakan dalam format citra JPEG.

Perhitungan DCT dapat dilakukan dengan DCT multidimensi yang dilakukan pada kolom dan baris sekaligus. DCT dua dimensi pada matriks dengan dimensi MxN diberikan oleh persamaan berikut ini:

$$B_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} A_{m,b} \cos\frac{\pi(2m+1)p}{2M} \cos\frac{\pi(2n+1)q}{2N}, \begin{matrix} 0 \le p \le M-1 \\ 0 \le q \le N-1 \end{matrix} \qquad (2.4)$$

$$\alpha_p = \begin{cases} 1/\sqrt{M} & , p = 0 \\ \sqrt{2/M} & , 0 < p \le M-1 \end{cases} \qquad \alpha_q = \begin{cases} 1/\sqrt{N} & , q = 0 \\ \sqrt{2/N} & , 0 < q \le N-1 \end{cases}$$

Dalam perhitungan yang dipakai untuk kompresi JPEG ini, matriks berbentuk bujur sangkat dengan ukuran M dan N sama.

Nilai DCT terkonsentrasi pada variabel di paling kiri atas. Nilai ini adalah koefisien DCT Secangkan 63 nilai sisanya disebut koefisien AC. Perhitungan DCT secara sementara meningkatkan jumlah memori yang diperlukan, karena koeffisien DC dapat membutuhkan bit data yang lebih besar dari bit sebelumnya yang digunakan untuk menyimpan data piksel. Sifat DCT yang mengakumulasi informasi dibagian kiri atas ini sangat menguntungkan dalam proses selanjutnya. Sisa bit yang berupa deretan angka nol bisa dipotong dengan penanda EOB (*End of Byte*). Ini bisa digunakan untuk kompresi.

Kompresi juga bisa dilakukan pada saat perhitungan DCT ini. Dengan cara mengatur seberapa besar koefisien DCT yang disimpan dalam perhitungan selanjutnya.

Berikut adalah contoh hasil dari perhitungan DCT. Misalkan berikut adalah blok citra yang akan dilakukan perhitungan DCT.

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 114 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 122 & 154 & 106 & 70 & 69 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

Gambar 2-5 Contoh blok citra dengan ukuran 8x8

Maka sebelum di lakukan perhitungan DCT, pada citra JPEG dilakukan terlebih dahulu tranformasi nilai piksel citra yang berada pada kisaran postif 0-255 ini menjadi terpusat pada nilai 0. Ini dilakukan dengan cara mengurangkan dengan nilai tengah dari *grayscale* citra yaitu 128.

Dengan pengurangan nilai ini akan didapatkan nilai piksel antara -128 sampai 127. Perhitungan pada matriks diatas akan didapatkan nilai :

$$\begin{bmatrix} -76 & -73 & -67 & -62 & -58 & -67 & -64 & -55 \\ -65 & -69 & -73 & -38 & -19 & -43 & -59 & -56 \\ -66 & -69 & -60 & -15 & 16 & -24 & -62 & -55 \\ -65 & -70 & -57 & -6 & 26 & -22 & -58 & -59 \\ -61 & -67 & -60 & -24 & -2 & -40 & -60 & -58 \\ -49 & -63 & -68 & -58 & -51 & -60 & -70 & -53 \\ -41 & -49 & -59 & -60 & -63 & -52 & -50 & -54 \end{bmatrix}$$

Gambar 2-6 Citra setelah dipusatkan nilainya pada 0

Nilai ini yang akan dirubah menjadi domain frekuensi dengan DCT. Pada JPEG digunakan DCT tipe 2.

$$G_{u,v} = \alpha(u)\alpha(v)\sum_{x=0}^{7}\sum_{y=0}^{7} g_{x,y}\cos\left[\frac{\pi}{8}(x+\frac{1}{2})u\right]\cos\left[\frac{\pi}{8}(y+\frac{1}{2})v\right] \qquad (2.5)$$

dimana:

- u adalah *horizontal spacial frequency*
- v adalah *vertical spacial frequency*
- $\alpha_p(n)$ adalah konstanta dengan nilai :
- $g_{x,y}$ adalah nilai piksel pada domain spasial dengan koordinat *x,y*
- $G_{u,v}$ adalah nilai koefisien DCT pada koordinat *u,v*

Hasil perhitungan pada matriks diatas adalah:

$$\begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix}$$

Gambar 2-7 Hasil perhitungan DCT blok citra

Untuk mengembalikan citra dari domain frekuensi ke domain spasial digunakan perhitungan *Inverse Discrete Cosine Transform* (IDCT). Adapun perhitungan IDCT diberikan dengan rumus seperti berikut ini

$$A_{mn} = \sum_{p=0}^{M-1}\sum_{q=0}^{N-1}\alpha_p\alpha_q B_{p,q}\cos\frac{\pi(2m+1)p}{2M}\cos\frac{\pi(2n+1)q}{2N}, \begin{matrix} 0 \le m \le M-1 \\ 0 \le n \le N-1 \end{matrix} \qquad (2.6)$$

$$\alpha_p = \begin{cases} 1/\sqrt{M} & , p = 0 \\ \sqrt{2/M} & , 0 < p \le M-1 \end{cases} \qquad \alpha_p = \begin{cases} 1/\sqrt{M} & , p = 0 \\ \sqrt{2/M} & , 0 < p \le M-1 \end{cases}$$

D. Menghitung kuantisasi DCT

DCT selanjutnya akan mengalami proses kuantisasi. Kuantisasi dalam pengolahan citra adalah kompresi bersifat *lossy* yang dilakukan dengan mengecilkan jangkauan nilai pada suatu nilai. Dengan mengurangi simbol diskrit pada suatu citra maka, akan menjadi lebih mudah untuk dikompresi.

Kuantisasi dilakukan dengan membagi nilai di blok DCT matriks dengan nilai pada koordinat yang bersesuaian di matrik kuantisasi. Nilai pada matriks kuantisasi ini ikut menentukan seberapa besar kompresi yang akan dilakukan.

Adapun matriks kuantisasi yang digunakan dalam program ini adalah:

$$\begin{bmatrix} 8 & 10 & 10 & 2 & 2 & 2 & 2 & 3 \\ 10 & 10 & 2 & 2 & 2 & 2 & 3 & 3 \\ 10 & 2 & 2 & 2 & 2 & 3 & 3 & 3 \\ 2 & 2 & 2 & 2 & 2 & 3 & 3 & 4 \\ 2 & 2 & 2 & 2 & 3 & 3 & 4 & 4 \\ 2 & 2 & 2 & 3 & 3 & 4 & 4 & 5 \\ 2 & 2 & 2 & 3 & 3 & 4 & 5 & 6 \\ 2 & 2 & 3 & 3 & 4 & 5 & 6 & 8 \end{bmatrix}$$

Gambar 2-8 Matriks kuantisasi yang digunakan pada proses kuantisasi

Untuk perhitungan kuantisasi digunakan pembagian per elemen matriks dengan rumus:

$$B_{j,k} = round\left(\frac{A_{j,k}}{Q_{j,k}}\right) \qquad (2.7)$$

dimana $A_{j,k}$ adalah konstanta DCT pada koordinat (*j*,*k*) dan $Q_{j,k}$ adalah elemen matrik kuantisasi pada koordinat (*j*,*k*).

Untuk mendapatkan kembali nilai DCT dilakukan dekuantisasi, dengan cara melakukan perkalian *entry-for-entry product* dimana tiap elemen dari matriks hasil kuatisasi (QDCT) dikalikan dengan nilai elemen bersesuaian matriks kuantisasi.

$$D_{j,k} = B_{j,k} \times Q_{j,k} \qquad (2.8)$$

Misalkan bila terdapat blok DCT dengan nilai berikut:

$$\begin{bmatrix} -217 & -31 & -9 & 0 & 0 & 0 & 0 & 0 \\ 0 & -20 & 6 & 9 & -1 & -1 & 0 & 1 \\ -6 & 6 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Gambar 2-9 Contoh blok DCT dengan ukuran 8x8

Maka dengan matriks kuantisasi di atas bila dilakukan kuantisasi akan menghasilkan nilai Quantized DCT:

$$\begin{bmatrix} -27 & -3 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -2 & 3 & 5 & -1 & -1 & 0 & 0 \\ -1 & 3 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Gambar 2-10 Hasil kuantisasi blok DCT

Dasar dari proses kuantisasi ini adalah, mata manusia bagus dalam membedakan perbedaan kecil dalam brightness pada ruang yang relatif luas. Sehingga bisa dikurangi atau dihilangkan detail informasi pada frekuensi tinggi. Tahap kuantisasi untuk menghilangkan detail pada frekuensi tinggi ini merupakan bagian utama dari kompresi citra dengan metode *lossy*. Kuantisasi juga mengurangi jumlah bit yang diperlukan untuk menyimpan data DCT.

2.2 Steganografi

Steganografi adalah seni dan ilmu menyembunyikan informasi. Sistem steganografi menyembunyikan informasi pada media yang tidak mencurigakan. Pada masa lalu orang menyembunyikan pesan pada tato di kepala yang kemudian pesan tersebut ditutupi dengan rambut sehingga tersembunyi. Di masa modern ini teknologi komputer dan jaringan menyediakan media yang menarik untuk steganografi.

Inti dari proses penyembunyian informasi adalah mengidentifikasi *redundant bit* dari media (*cover medium*), yang bisa dimodifikasi tanpa merusak integritas medium. Proses penyisipan (*embeding)* membuat hasil steganografi (*stego medium*) dengan menggantikan *redundant bi*t tersebut dengan bit dari pesan yang akan disembunyikan.

Tujuan dari steganografi adalah membuat kehadiran informasi yang tersembunyi ini tidak terdeteksi. Namun karena sifat steganografi yang invasif, proses ini akan meninggalkan jejak di medium. Perubahan statistik yang diakibatkan oleh steganografi ini bisa digunakan untuk melacak adanya informasi yang tersembunyi. Hal ini disebut dengan *statistical steganalisis*.

Salah satu media yang baik dipakai untuk steganografi adalah citra JPEG. Ini karena citra JPEG memiliki kualitas citra yang baik dan ukuran yang relatif kecil sehingga banyak digunakan. Selain itu steganografi pada citra JPEG dilakukan pada domain frekuensi, bukan pada domain spasial sehingga lebih tahan pada serangan visual. Ini berbeda dengan citra GIF yang berbasis palet, yang bila bitnya dimodifikasi dengan steganografi menimbulkan jejak yang sangat terlihat.

Steganografi pada citra JPEG dapat menyembunyikan bit informasi pada bit DCT. Algoritma dasar yang digunakan untuk steganografi ini dapat dilihat pada gambar 2.10:

```
Input: message, cover image
Output: stego image
while data left to embed do
  get next DCT coefficient from cover image
```

```
      if DCT ≠ 0 and DCT ≠ 1 then
            get next LSB from message
            replace DCT LSB with message LSB
      end if
            insert DCT into stego image
    end while
```

Gambar 2-11 Hasil kuantisasi blok DCT

Algoritma steganografi JSTEG dari Derek Ulham pada gambar 2.10 adalah algoritma pertama untuk menyembunyikan pesan ke dalam citra JPEG pada domain frekuensi. Algoritma ini menyembunyikan informasi dengan menyisipkan bit informasi pada bit redundan, yang pada JPEG adalah pada konstanta DCT. Algoritma diatas tidak membutuhkan *shared secret* antara pengirim dan penerima pesan, sehingga sangat rentan terhadap serangan steganalisis. Setiap orang yang mengetahui metode ini dapat dengan mudah mengetahui isi pesan yang tersembunyi.

Selain itu Westfeld dan Pfitzman menunjukkan bahwa proses steganografi yang memodifikasi *least significant bit* (LSB) menyebabkan distorsi yag bisa diketahui melalui steganalisis. Westfeld dan Pfitzman menggunakan *Chi-square test* untuk menentukan apakah suatu citra mengalami distorsi. Dengan test ini. Meskipun DCT dari citra asal tidak diketahui diasumsikan jumlah dari koefisien DCT yang bersesuaian bersifat invarian. Dengan demikian bisa dihitung ekspektasi dari distribusi *y** dari citra hasil steganografi. Bila n_1 adalah histogram dari koeffisien DCT maka bisa dihitung nilai *y** dengan persamaan

$$y_i^* = \frac{n_{21} + n_{2i+1}}{2} \qquad (2.9)$$

Berikuut perhitungan ekspektasi distribusi dan membandingkanya dengan distribusi yang diamati

$$y_i = n_{2i} \qquad (2.10)$$

sehingga nilai χ^2 untuk kedua distribusi adalah

$$\chi^2 = \sum_{i=1}^{v+1} \frac{(y_1 - y_i*)}{y_i*} \qquad (2.11)$$

dimana v adalah derajat kebebasan, yaitu satu kurang dari jumlah kriteria konstanta DCT.

$$p = 1 - \int_0^{\chi^2} \frac{t^{(v-2)/2} e^{-t/2}}{2^{v/2}\Gamma(v/2)} \qquad (2.12)$$

dimana Γ adalah fungsi Gamma Euler. Kemungkinan terdeteksi dari adanya suatu informasi dalam citra adalah *p*.

Fungsi Gamma Euler untuk bilangan bulat positif adalah:

$$\Gamma(z) = (z-1)! \qquad (2.13)$$

Dalam tugas akhir ini akan diberikan beberapa metode yang dapat meningkatkan kemanan dan kulitas dari steganografi pada citra JPEG ini.

2.3 Stream Cipher RC4

Secara umum, proses penyembunyian informasi dimulai dengan mengetahui adanya bit yang redundan dalam citra yang digunakan sebagai cover. *Redundant bit* adalah bit yang bisa dimodifikasi tanpa merusak kesatuan dari medium cover. Proses *embedding* lalu memilih subset dari bit yang redundan ini untuk menyimpan data dari pesan. Medium stego dibuat dengan mengganti bit yang redundan ini dengan bit pesan.

Salah satu cara penyempurnaan steganografi ini adalah dengan menggunakan *pseudo random number generator* (PRNG) untuk melakukan pilihan konstanta DCT. Pilihan ini dihasilkan secara berdeda dari *seed* PRNG yang berupa *shared secret* atau *password* yang berbeda. Karena hasli PRNG tergantung dari *shared secret* yang digunakan sebagai *seed*, maka seleksi DCT dan isi pesan tidak dapat diketahui tanpa mengetahui *shared secret tersebut.*

Untuk algoritma steganografi yang diusulkan oleh Niels Provost, PRNG yang digunakan adalah RC4. Untuk proses ini, user diminta memasukkan suatu String yang kemudian akan dibaca sebagai *array of bytes*. Array ini akan

digunakan sebagai *seed* untuk membuat bilangan random dengan menggunakan algoritma RC4.

RC4 adalah *stream cipher* yang menggunakan permutasi dari *secret key* yang dimasukkan pengguna untuk membuat keystream. *Keystream* ini adalah yang digunakan untuk enkripsi dan bisa dianggap pula sebagai aliran dari nilai angka random.

Karena PRNG ini dikunci dengan menggunakan kata sandi atau *shared secret* tersebut, isi pesan yang disisipkan tidak bisa diketahui dengan tanpa mengetahui kata kuncinya. Penerima dari citra steganografi ini harus menjalankan PRNG dengan kata kunci yang dia tahu untuk memulai proses seleksi dan mengambil isi pesan yang disisipkan dalam citra.

RC4 terdiri atas dua bagian yaitu *key-scheduling algorithm* (KSA) dan *pseudo-random number generation algorithm* (PRNG).

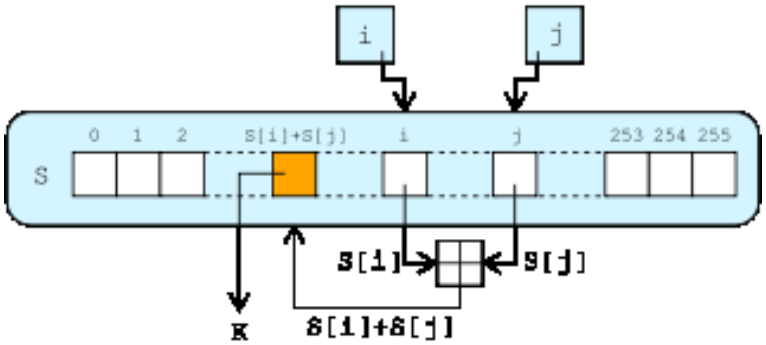
- *Key-Scheduling algorithm*

Key scheduling algorithm ini digunakan untuk melakukan inisialisasi permutasi pada array S. *Keylength* adalah jumlah dari byte pada key, yang bisa pada range $1 \leq keylength \leq 256$. Pertama array S diinisialisasi dengan menggunakan *identity permutation*. S array ini kemudian diproses sebanyak 256 iterasi.

Pseudocode untuk KSA dapat dilihat pada gambar 2.11

```
for i from 0 to 255
    S[i] := i
endfor
j := 0
for i from 0 to 255
    j := (j+S[i]+key[i mod keylength])
         mod 256
    swap(S[i],S[j])
Endfor
```

Gambar 2-12 Pseudocode untuk Key-Scheduling algorithm



Gambar 2-13 Ilustrasi algoritma RC4

Byte hasil dari RC4 dibuat dengan melihat nilai S[i] dan S[j], menjumlahkan keduanya kemudiaan dicari nilai modulo 256 dan mencari nllainya di tabel S. Nilai byte S(S(i) + S(j)) digunakan sebagai byte dari *keystream* K yang digunakan untuk membuat byte output dengan operasi bitwise XOR

- *Pseudo-random Number Generation Algorithm*

Sebanyak iterasi yang diperlukan (sebanyak jumlah byte dari pesan yang dimasukkan), PRNG memodifikasi byte dari *keystream*. Hasilnya adalah *keystream* yang memiliki panjang sama dengan input. *Keystream* ini yang akan di-XOR-kan per byte dengan input untuk menghasilkan output.

Keystream ini juga yang akan digunakan untuk menginisialisasi pemilihan koefisien DCT yang akan disisipi pada prosses *embedding* pada steganografi.

Pseudocode untuk PRNG ini bisa dilihat pada gambar 2.13.

```
i := 0
j := 0
while GeneratingOutput:
    i := (i + 1) mod 256
    j := (j + S[i]) mod 256
    swap(S[i],S[j])
    output S[(S[i] + S[j]) mod 256]
Endwhile
```

Gambar 2-14 Pseudocode untuk PRNG

2.4 Algoritma Genetika (GA)

Algoritma Genetika (*genetic algorithm*, GA) adalah teknik pencarian pada komputasi untuk menemukan solusi tepat atau pendekatan terhadap suatu masalah. GA dikategorikan kedalam *global search heuristics* dan *evolutionary algorithms* yang menggunakan prinsip biologi seperti *inheritance*, *mutation*, *selection*, dan *crossover* (disebut juga *recombination*).

GA umumnya memerlukan dua hal untuk didefiniskan:

- representasi genetik dari domain solusi,
- suatu fungsi *fitness* untuk mengevaluasi domain solusi.

Pada alam ini, informasi genetik dari sebuah individu disimpan dalam *chromosome*, yang terdiri dari sekumpulan gen. Karakteristik dari setiap individu dikendalikan oleh gen-gen tersebut, yang kemudian akan diwariskan kepada keturunan-keturunannya ketika individu tersebut berkembang biak. Selain faktor perkembangbiakan, suatu ketika juga terjadi peristiwa yang disebut mutasi, yang menyebabkan terjadinya perubahan informasi pada *chromosome*. Berdasarkan teori Darwin tersebut, nilai rata-rata karakteristik dari populasi akan meningkat setiap generasi, seiring dengan bertambahnya individu yang mempunyai kriteria yang bagus dan punahnya individu yang mempunyai kriteria yang buruk.

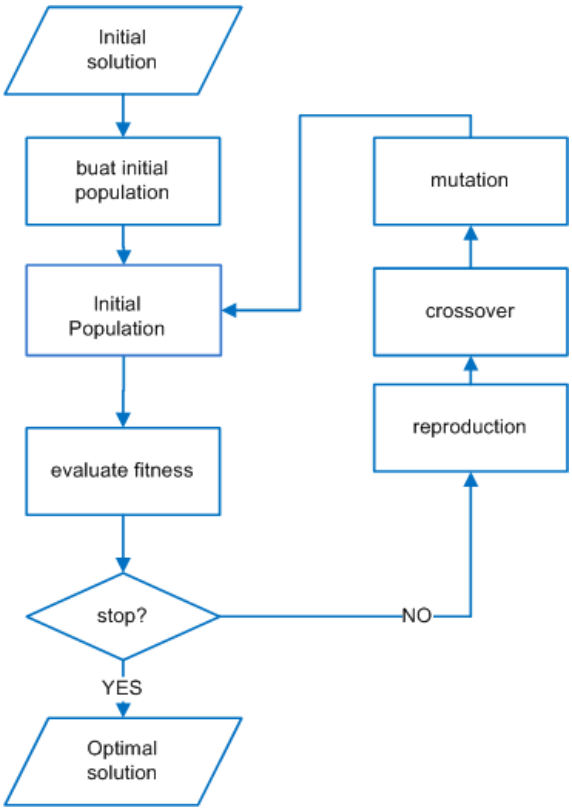
Terinspirasi dari teori Darwin tersebut, pada tahun 1975 John Holland dan timnya menciptakan teori GA. Ide utama dibalik GA ini adalah memodelkan proses evolusi alami menggunakan warisan. Meskipun diperkenalkan oleh John Holland, penggunaan GA untuk memecahkan persoalan yang kompleks didemonstrasikan kemudian (pada tahun yang sama) oleh De Jong, dan kemudian oleh Goldberg pada tahun 1989.

Menyesuaikan dengan teori Darwin, dalam Algoritma Genetika digunakan istilah-istilah yang mewakili elemen-elemen dalam teori Darwin tersebut, yaitu:

- *Population*: Merupakan sekumpulan solusi dari permasalahan yang akan diselesaikan menggunakan Genetic Algorithm. Population terdiri dari sekumpulan c*hromosome*.

- *Chromosome*: Mewakili sebuah solusi yang mungkin (*feasible solution*) untuk permasalahan yang ingin diselesaikan. Sebuah kromosom terdiri dari sekumpulan gen.
- *Gen*: Mewakili elemen-elemen yang ada dalam sebuah solusi.
- *Parent*: Merupakan chromosome yang akan dikenai operasi genetik (*crossover*).
- *Offspring*: Adalah chromosome yang merupakan hasil dari operasi genetik (*crossover*).
- *Crossover*: Merupakan operasi genetik yang mewakili proses perkembangbiakan antar individu. Proses *crossover* ini memerlukan dua buah parent dan menghasilkan satu atau lebih *offspring* (keturunan). Detail dari proses *crossover* ini akan dibahas kemudian.
- *Mutation*: Merupakan operasi genetik yang mewakili proses mutasi dalam perjalanan hidup individu. Mutasi berperan menghasilkan perubahan acak dalam populasi, yang berguna untuk menambah variasi dari *chromosome* dalam sebuah populasi.
- *Selection Procedure*: Merupakan proses yang mewakili proses seleksi alam (*natural selection*) dari teori Darwin. Proses ini dilakukan untuk menentukan parent dari operasi genetik (*crossover*) yang akan dilakukan untuk menghasilkan keturunan.
- *Fitness Value*: Merupakan penilaian yang menentukan bagus tidaknya sebuah kromosom. Kromosom yang mempunyai *Fitness Value* yang rendah pada akhirnya akan tersingkir oleh kromosom yang mempunyai *Fitness Value* yang lebih baik.
- *Evaluation Function*: Merupakan fungsi yang digunakan untuk menentukan nilai dari *Fitness Value*. *Evaluation Function* ini merupakan sekumpulan kriteria-kriteria tertentu dari permasalahan yang ingin diselesaikan.
- *Generation*: Merupakan satuan dari populasi setelah mengalami operasi genetika, berkembang biak, dan menghasilkan keturunan. Pada akhir setiap *generation*, agar jumlah kromosom dalam populasi tetap konstan,

kromosom yang mempunya *Fitness Value* yang rendah akan dihapus dari populasi.



Gambar 2-15 Alur Algoritma Genetika

GA merupakan metode pembelajaran *heuristic* yang adaptif, efektif, sederhana dan relatif mudah untuk diimplementasikan. GA memiliki keunggulan-keunggulan dibandingkan dengan metode-metode heuristic yang lain, yaitu:

- GA menyelesaikan masalah dengan mengkodekan permasalah menjadi kromosom, bukan dengan menyelesaikan permasalahan itu sendiri. Karena itu diperlukan pemodelan yang baik dan efektif yang dapat mewakili solusi dari permasalahan yang dihadapi.
- GA memulai prosesnya dengan sekumpulan *initial solutions*, berbeda dengan *metaheuristic* lain seperti *Simulated Annealing* dan *Tabu Search* yang memulai proses dengan sebuah solusi tunggal, dan berlanjut ke solusi lainnya melalui suatu transisi. Karena itu GA melakukan pencarian *multi-*

directional dalam *solution space,* yang memperkecil kemungkinan berhentinya pencarian pada kondisi *local optimum.*

- Hanya diperlukan sebuah fungsi evaluasi tunggal yang berbeda untuk tiap permasalahan.
- GA merupakan algoritma yang ‘buta’, karena GA tidak mengetahui kapan dirinya telah mencapai solusi optimal.

Secara umum GA harus memenuhi kriteria-kriteria dibawah ini untuk menghasilkan solusi yang optimal:

- Sebuah representasi yang tepat dari sebuah solusi permasalahan, dalam bentuk kromosom.
- Pembangkit populasi awal. Umumnya populasi awal dibangkitkan secara acak, namun untuk beberapa kasus juga bisa dibangkitkan melalui metode-metode tertentu. Penggabungan pembangkitan populasi awal secara acak dan menggunakan metode-metode tertentu, disebut *seeding*. Populasi awal yang dihasilkan sebaiknya bersifat heterogen, karena jika terlalu homogen, GA akan kehilangan kemampuannya untuk mencari dalam solution space, sampai populasi mempunyai variasi kromosom yang beragam melalui operasi genetik lain (*mutation*).
- Sebuah *evaluation function* untuk menentukan *fitness* value dari tiap solusi.
- GA, mensimulasikan proses reproduksi dan mutasi.
- Parameter-parameter lain, seperti kapasitas populasi, probabilitas dari operasi genetik, dan semacamnya.

Kapasitas populasi sangat mempengaruhi kemampuan GA dalam mencari solusi. Kapasitas populasi yang terlalu kecil menyebabkan kurangnya variasi kromosom yang muncul, sehingga dapat menyebabkan hasil akhir yang buruk. Kapasitas populasi yang besar biasanya memberikan hasil yang lebih baik, dan mencegah terjadinya konvergensi yang prematur.

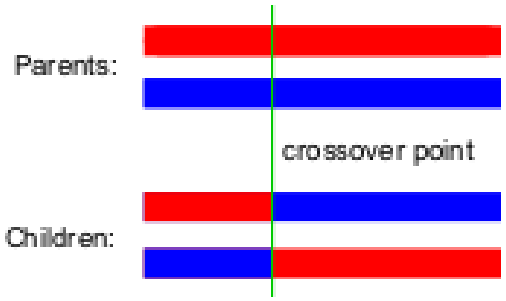
Pada GA salah satu bagian penting adalah *crossover*. *Crossover* dilakukan untuk memberikan suatu variansi pada kromosom didalam satu populasi secara random. Pada Algoritma Genetika terdapat tiga macam *crossover* yang utama:

- *Uniform Crossover*

Pada *Uniform Crossover*, kedua kromosom induk akan dikombinasikan untuk menghasilkan kromosom anak. Setiap bit individual pada kedua kromosom dibandingkan. Bit-bit ini akan ditukar dengan kemungkinan tetap. Misalnya dengan kemungkinan 0,5. Persilangan jenis ini akan menghasilkan dua buah kromosom anak dari dua buah kromosom induk.

- *One Point Crossover*

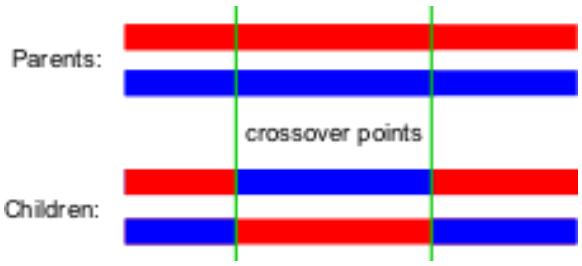
Pada crossover tipe ini satu titik (satu *gene*) pada tiap kromosom dipilih kemudian data setelah batas *gene* tersebut akan ditukar ke kedua kromosom. Kromosom hasil dari persilangan ini adalah kromosom anak.



Gambar 2-16 Crossover dengan metode One Point

- Two Point Crossover

Pada *Two-point crossover* dipilih dua titik (dua *gene*) di kedua kromosom yang akan disiilangkan. Maka *gene* yang terletak diantara kedua titik tersebut akan ditukarkan. Persilangan jenis ini akan menghasilkan dua buah kromosom anak dari dua buah kromosom induk.



Gambar 2-17 Crossover dengan metode Two Point

2.5 Peak Signal-to-Noise Ratio (PSNR)

PSNR (*Peak Signal-to-Noise Ratio*) adalah istilah yang digunakan untuk menyatakan perbandingan antara kuat sinyal maksimum dengan *noise* yang mepengaruhi kualitas dari sinya tersebut. Karena sinyal sering memiliki jangkauan yang lebar maka PSNR dinyatakan dalam skala logaritmik (beda satu dalam skala ini berarti beda nilai sebesar sepuluh kali).

PSNR ini biasa digunakan dalam menghitung kualitas kompresi pada citra. PSNR bisa dihitung dari *mean squared error* (MSE) antara dua citra dengan dimensi MxN.

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{i=0}^{n-1}\left\|I(i,j) - K(i,j)\right\|^2 \tag{2.14}$$

PSNR didefinisikan sebagai:

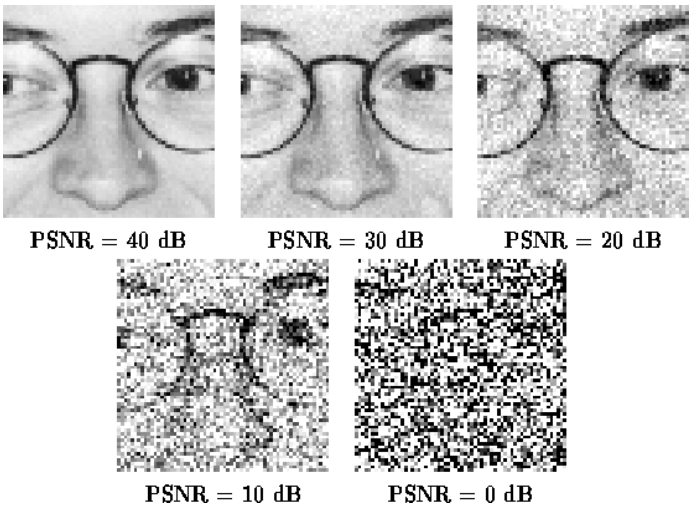
$$PSNR = 10\log_{10}\left(\frac{MAX_I^2}{MSE}\right) = 20\log_{10}\left(\frac{MAX_I}{MSE}\right) \tag{2.15}$$

Di sini MAX_i adalah nilai maksimal dari suatu pixel di dalam citra, yaitu 255. Perhitungan diatas adalah untuk citra dengan satu nilai (*grayscale*), sedangkan untuk menghitung PSNR pada citra RGB sama saja, hanya saja MSE dihitung dengan menjumlahkan semua kuadrat selisih pada ketiga nilai R, G dan B dan kemudia dibagi tiga kali panjang kali lebar citra.

$$\begin{aligned} MSE_{RGB} = \frac{1}{3mn}\sum_{i=0}^{m-1}\sum_{i=0}^{n-1} & \left\|I_R(i,j) - K_R(i,j)\right\|^2 \\ & + \left\|I_G(i,j) - K_G(i,j)\right\|^2 \\ & + \left\|I_B(i,j) - K_B(i,j)\right\|^2 \end{aligned} \tag{2.16}$$

$$PSNR_{RGB} = 10\log_{10}\left(\frac{MAX_I^2}{MSE_{RGB}}\right) = 20\log_{10}\left(\frac{MAX_I}{MSE_{RGB}}\right) \tag{2.17}$$

Biasanya nilai PSNR berada pada nilai 30 sampai 50 dB dengan nilai semakin besar semakin bagus, sebaliknya semakin rendah nilai PSNR maka kualitas citra semakin rendah. Biasanya Nilai PSNR dituliskan hingga nilai dua angka dibelakang koma. Nilai PSNR ini biasanya penting bila digunakan untuk membandingkan kualitas antara dua citra. Misalnya komite MPEG menggunakan batas perbandingan PSNR 0.5 untuk menentukan apakah kompresi dapat diterima dan menghasilkan kualitas yang baik.



Gambar 2-18 Illustrasi ukuran PSNR dan kualitas Citra

Perbandingan nilai PSNR ini dapat digunakan untuk mengukur kualitas citra secara objektif. Bila diketahui dua citra, dapat dibandingkan secara subjektif dengan memberikan citra tersebut kepada pengamat untuk diamati secara visual perbedaanya. Namun bila tingkat perbedaan kecil, secara subjektif tidak akan bisa dibedakan. Karena itu dibutuhkan ukuran objektif untuk menentukan kualitas dari citra.

BAB 3

METODOLOGI

Pada bab ini akan dijelaskan mengenai analisis sistem, perancangan sistem, dan pembuatan sistem beserta implementasi dari steganografi dan Algoritma Genetika untuk optimasi dari steganografi ini.

3.1 Analisis Sistem

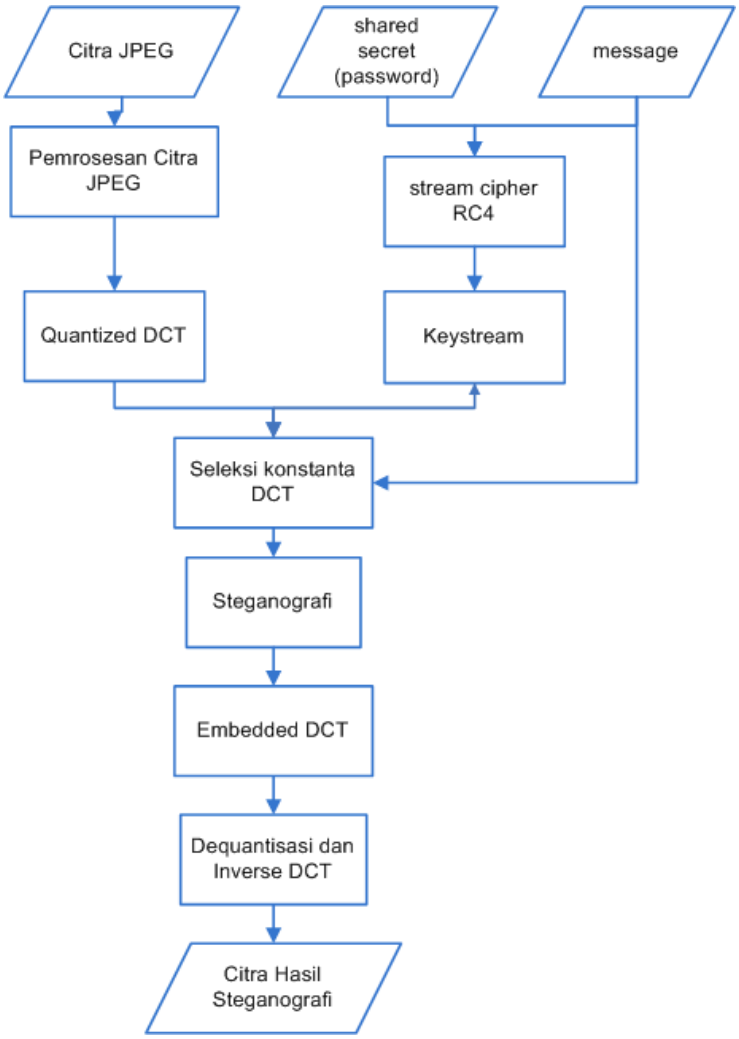
Pada bagian ini, akan dijelaskan proses-proses utama yang merupakan inti dari implementasi dari steganografi pada citra JPEG. Proses-proses tersebut mulai dari pembacaan citra, konersi ke ruang warna YCbCr, perhitungan DCT, dan kuantisasi DCT.

Tahapan berikutnya adalah stream cipher RC4. RC4 digunakan untuk dua hal. Pertama adalah enkripsi pesan yang akan disisipkan. Ini akan memberikan tambahan keamanan dalam proses steganografi. RC4 digunakan sebagai *random number generator* untuk melakukan seleksi konstanta DCT yang akan disisipi oleh bit dari pesan.

Setelah mendapatkan *keystream* dari RC4 ini, dilakukan seleksi terhadap konstanta hasil kuantisasi DCT. Steganografi dilakukan dengan mengganti *Least-Significant Bit* (LSB) dari konstanta terpilih dengan bit dari pesan. Sementara konstanta DCT yang tidak terpilih (di-*skip*) akan dibiarkan tetap.

Untuk proses steganografi dengan algoritma genetika, konstanta hasil kuantisasi DCT dijadikan kromosom dan dihitug fungsi *fitness*-nya mana yang setelah dilakukan penggantian LSB konstanta DCT dengan bit dari pesan menghasilkan citra yang paling sama bila dibandingkan dengan citra asal.

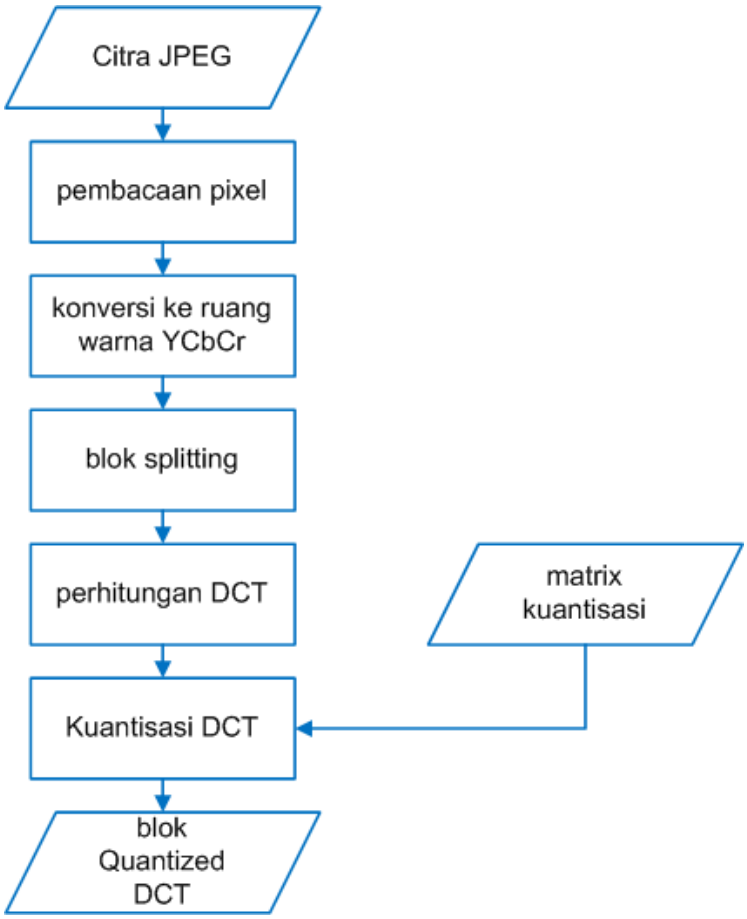
Untuk pengujian kualitas citra di lakukan perhitungan *Peak Signal-to-Noise Ratio* (PSNR) untuk analisa kualitas visual. Alur dari sistem diilustrasikan dalam gambar 3.1.



Gambar 3-1 Diagram alir garis besar sistem

3.1.1.1 Pemrosesan Citra JPEG

Pemrosesan citra JPEG dilakukan mulai dari membaca nilai piksel citra (dalam domain spasial), melakukan konversi ke ruang warna YCbCr, merubah ke dalam domain frekuensi dengan DCT dan kuantisasi DCT. Konstanta hasil kuantisasi ini akan menjadi tempat penyisipan pesan pada proses steganografi. Sedangkan pada steganografi dengan algoritma genetika konstanta hasil kuantisasi DCT ini akan menjadi kromosom yang akan dioptimasi. Alur proses ini diilustrasikan dengan gambar 3.2.



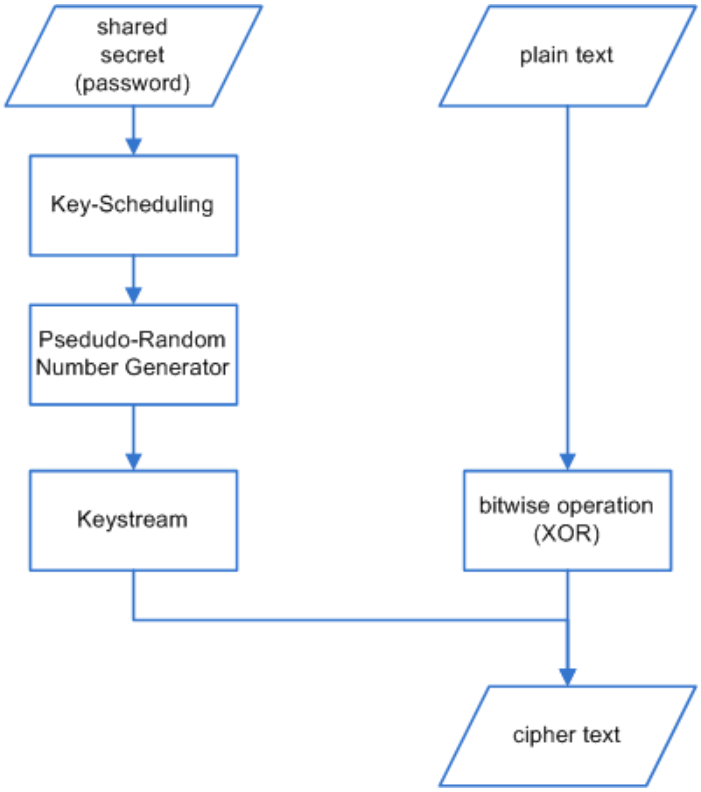
Gambar 3-2 Diagram alir pemrosesan citra JPEG

Masukan dari proses pemrosesan JPEG ini adalah Citra dengan format JPEG. Hasil akhir dari tahapan ini adalah adalah matriks hasil kuantisasi DCT dari tiap layer dari ruang warna YCbCr. Pada tahapan steganografi nantinya, pesan akan disembunyikan ke dalam citra dengan mengganti *Least Significant Bit* (LSB) dari citra dengan bit dari pesan.

Steganografi akan dilakukan pada domain frekuensi agar hasil steganografi lebnih tahan terhadap serangan visual. Sebab karena perubahannya dilakukan di domain frekuensi, perubahan informasi akibat perubahan bit DCT pada citra akan lebih tidak kentara dibandingkan bila pada domain spasial.

Hasil dari steganografi akan mengalami distorsi visual apabila dibandingkan dengan citra asal. Bila perbedaan ini teralu besar maka adanya pesaan tersembunyi bisa dideteksi oleh pihak yang tidak diinginkan.

3.1.1.2 Stream Cipher RC4



Gambar 3-3 Diagram alir proses RC4

Untuk algoritma steganografi yang diusulkan oleh Niels Provost, PRNG yang digunakan adalah RC4. Untuk proses ini, user diminta memasukkan suatu *String* yang kemudian akan dibaca sebagai *array of bytes*. Array ini akan digunakan sebagai *seed* untuk membuat bilangan acak dengan menggunakan algoritma RC4.

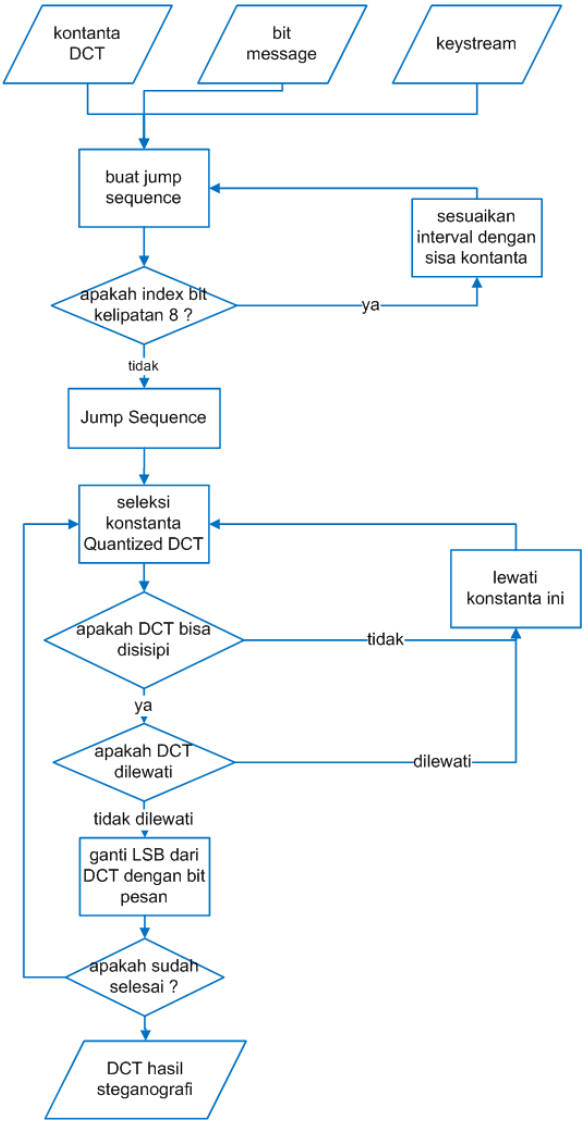
RC4 adalah *stream cipher* yang menggunakan permutasi dari *secret key* yang dimasukkan pengguna untuk membuat keystream. *Keystream* ini adalah yang digunakan untuk enkripsi dan bisa dianggap pula sebagai aliran dari nilai angka random.

Karena PRNG ini dikunci dengan menggunakan kata sandi, isi pesan yang disisipkan tidak bisa diketahui dengan tanpa mengetahui kata kuncinya. Penerima dari citra steganografi ini harus menjalankan PRNG dengan kata kunci yang dia tahu untuk memulai proses seleksi dan mengambil isi pesan yang disisipkan dalam citra.

Hasil akhir dari tahapan ini adalah *cipher text* dari pesan yang akan disisipkan dalam bentuk *array of bytes* dan keystream dalam bentuk *array of bytes* hasil RC4 yang akan digunkan untuk melakukan seleksi konstanta DCT yang akan dipakai pada proses steganografi.

3.1.1.3 Steganografi

Pada proses steganografi *keystream* yang didapat digunakan untuk memilih konstanta DCT. Akibatnya akan ada konstanta DCT yang dilewati. Sehingga ada konstanta DCT yang tidak dipakai.



Gambar 3-4 Diagram alir proses Steganografi

Ini memiliki keuntungan. Kualitas citra akan meningkat karena perubahan dari DCT tersebar pada konstanta yang ada di seluruh citra. Kedua, karena seleksi ini menggunakan *pseudo—random number generator* yang di-*seed* dengan *shared secret*, maka pihak yang tidak mengetahui shared secred ini tidak bisa mengetahui seleksi dan isi dari citra yang disembunyikan.

Keystream yang dihasilkan dengan RC4 akan digunakan menentukan bit mana yang akan disisipi pesan :

$$b_0 = 0$$
$$bi = b_{i-1} + R_i(x) \quad for\ \mathrm{i} = 1,...,n \qquad (3.1)$$

dimana b adalah posisi bit terseleksi yang ke i, dan Ri (x) adalah random *offset* dalam interval (1,x).

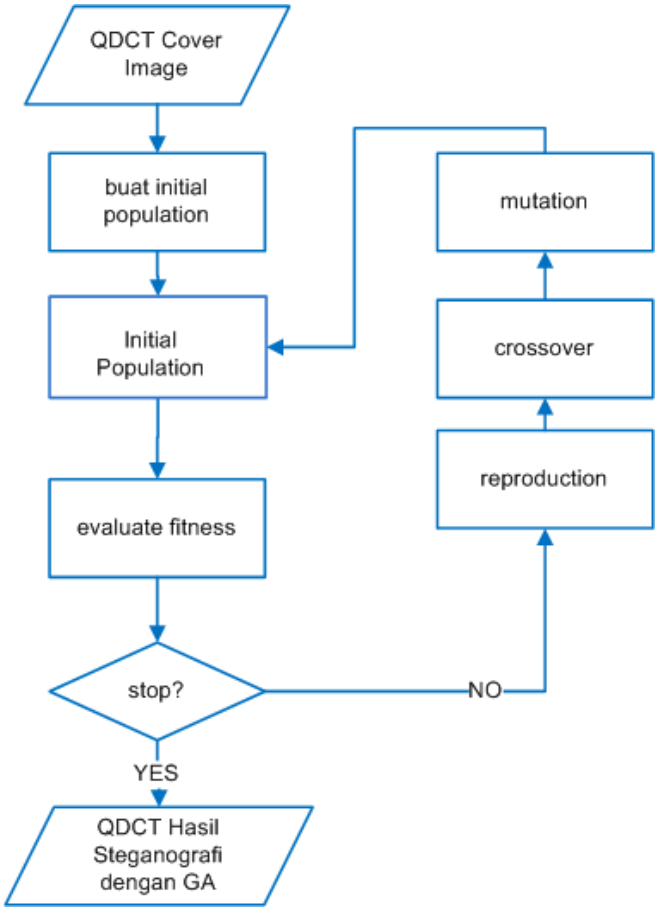
Proses ini akan memasukkan bit pesan ke dalam citra secara *pseudo-random*. Sehingga pesan tidak dapat dibaca tanpa mengetahui *key/password* yang digunakan untuk men-*generate pseudo-random number* tersebut.

Untuk mengetahui apakah proses teganografi telah berhasil, pesan yang tersembunyi dalam citra akan dicoba untuk di-*retrieve* dengan menggunakan shared secret untuk mengetahui lokasi konstanta QDCT yang digunakan untuk menyimpan pesan.

Bila dimasukkan *seed* yang salah untuk PRNG maka seleksi yang dihasilkan akan salah, sebaliknya bila *seed* yang dimasukkan sama dengan yang digunakan untuk menyisipkan pesan seleksi yang dilakukan akan tepat dan pesan yang disembunyikan dapat diketahui.

Pesaan yang disembunyikan sebelumya telah dienkripsi terlebih dahulu dengan menggunakan RC4. Karena itu setelah bit pesan didapatkan dari citra, pesan harus didekripsi terlebih dahulu dengan menggunakan RC4 untuk mengetahui pesan yang sebenarnya.

3.1.1.4 Algoritma Genetika



Gambar 3-5 Alur GA untuk Optimasi Steganografi

Algoritma Genetika (*genetic algorithm*, GA) adalah teknik pencarian pada komputasi untuk menemukan solusi tepat atau pendekatan terhadap suatu masalah. GA dikategorikan kedalam *global search heuristics* dan evolutionary algorithms yang menggunakan prinsip biologi seperti *inheritance*, *mutation*, *selection*, dan *crossover* (disebut juga *recombination*).

Suatu algoritma genetika umumnya memerlukan dua hal untuk didefiniskan:

1. representasi genetik dari domain solusi,
2. suatu fungsi *fitness* untuk mengevaluasi domain solusi.

Untuk tugas akhir ini GA akan diimplementasikan dengan menggunakan library **GALib** (C++ Library of Genetic Algorithm Components) oleh Matthew Wall dari MIT, yang diport untuk bahasa pemrograman Java oleh Jeff Smith.

1. Representasi Domain Solusi

Dalam sistem steganografi dengan GA ini satu blok Quantized DCT dengan ukuran 8x8 akan menjadi kromosom. Kromosom dibuat sebagai array satu dimensi berisi 64 konstanta DCT dengan tipe data integer. Satu blok akan dicari solusi optimalnya dengan GA selama sejumlah generasi tertentu.

Kromosom untuk GA dalam sistem ini akan dirancang sebagai array satu dimensi dari 64 konstanta dari suatu blok Quantized DCT.

Misalnya bila terdapat blok Quantized DCT dari hasil pengolahan citra dengan nilai seperti gambar 3-6.

$$\begin{bmatrix} 27 & -3 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -2 & 3 & 5 & -1 & -1 & 0 & 0 \\ -1 & 3 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Gambar 3-6 Blok QDCT citra asal

Maka pada awal proses GA akan dilakukan inisialisasi populasi dengan nilai QDCT diatas. Kromosom pertama dari generasi pertama akan diset sama persis dengan nilai blok QDCT.

$$[27\ -3\ -1\ 0\ 0\ \ldots.\ 0\ -2\ 3\ 5\ -1\ -1\ \ldots.\ 0\ 0\ 0]$$

Gambar 3-7 Kromososm pertama dari generasi awal (initial population)

Sedangkan kromosom lainya adalah modifikasi dengan nilai random pada interval tertentu dari nilai QDCT.

$$[27\ -3\ 0\ 0\ 1\ \ldots.\ 0\ -2\ 3\ 4\ -1\ 1\ \ldots.\ 0\ 0\ 0]$$

$$[26\ -3\ -1\ 0\ 0\ \ldots.\ 0\ -1\ 2\ 5\ -1\ -1\ \ldots.\ 0\ 0\ 0]$$

Gambar 3-8 Kromososm lain dari generasi awal (initial population)

GA akan berusaha mencari kromosom yang bila disisipi pesan akan menghasilkan QDCT, yang bila dikembalikan ke domain spasial menghasilkan citra dengan kualitas terbaik (termirip citra asal). Pencarian ini dilakukan secara random dengan melakukan perubahan pada kromosom dengan mutasi dan persilangan kromosom hingga mendapatkan kromosom dengan fitness terbaik.

Kromosom dengan nilai *fitness* terbaik dan sudah disipi pesan akan diambil dan dimasukkan ke dalam citra hasil steganografi. Misalkan hasil perhitungan dengan GA menghasilkan blok QDCT dengan nilai:

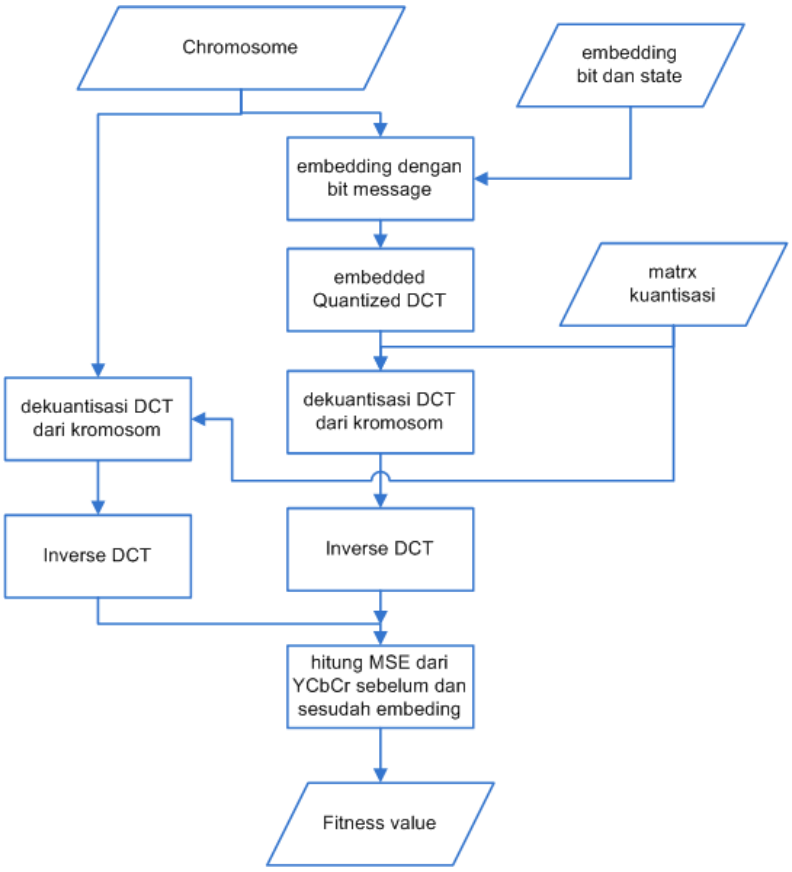
$$[27\ -3\ -1\ \ 0\ \ 0\ \ ....\ \ 0\ -2\ \ 2\ \ 5\ -1\ -1\ ....\ 0\ \ 0\ \ 0]$$

Gambar 3-9 Kromososm dengan nilai fitness tebaik

Dengan demikian GA akan mencari nilai QDCT yang menhasilkan citra yang semirip mungkin dengan citra asal agar proses steganografi lebih sulit dideteksi secara visual.

2. Fungsi Fitness

Setelah GA men-*generate* kromosom maka GA akan mengecek manakah diantara kromosom ini yang terbaik. Setelah dibuat kromosom untuk suatu generasi, kromosom tersebut masing-masing disipi dengan bit dari pesan yang akan disembunyikan, kemudian kromosom yang sudah diisi pesan ini akan dicek *fitness*-nya.



Gambar 3-10 Alur Perhitungan Fitness

Sebagai fungsi *fitness* dihitung *Mean Absolute Difference* (MAD) antara blok citra sebelum dan sesudah steganografi:

$$\left|\bar{\delta}\right| = \frac{1}{64}\sum_{0}^{x=7}\sum_{0}^{y=7}\left|I'(x,y) - I(x,y)\right| \qquad (3.2)$$

dimana *I'* dan *I* masing-masing adalah intensitas grayscale dari blok citra dengan ukuran 8x8 sesudah dan sebelum dilakukan Steganografi. Kromosom dengan nilai *fitness* terkecil adalah kromosom yang terbaik. Sehingga untuk untuk menghitung *fitness* harus diubah data citra dari domain frekuensi kembali ke domain spasial.

3.2 Perancangan Sistem

Perancangan sistem pada bab ini meliputi garis besar aktifitas sistem, perancangan data yang digunakan dalam proses, perancangan operasi atau prosedur yang dipakai dan penjelasan algoritma-algoritma yang dipakai secara terstruktur menggunakan diagram alir.

3.2.1 Garis Besar Sistem

Implementasi dari tugas akhir ini dibagi menjadi tiga tahapan, yaitu : tahap analisis, tahap pembandingan dan tahap uji coba atau tahap pemakaian aplikasi.

3.2.1.1 Tahap analisis (pengujian)

Yang dimaksud dengan tahap analisis adalah tahap pengujian semua variabel yang mempengaruhi sistem stegaografi. Variabel-variabel ini diuji untuk mendapatkan optimasi sistem. Pada implementasi tugas akhir ini variabel yang diuji adalah:

1. Ukuran *cover image* dan jumlah konstanta QDCT yang terdapat didalamnya
2. Panjang bit pesan yang akan disembunyian (*hidden message*) pada citra
3. Jenis *crossover* yang digunakan pada GA
4. Jumlah generasi maksimal yang digunakan pada GA

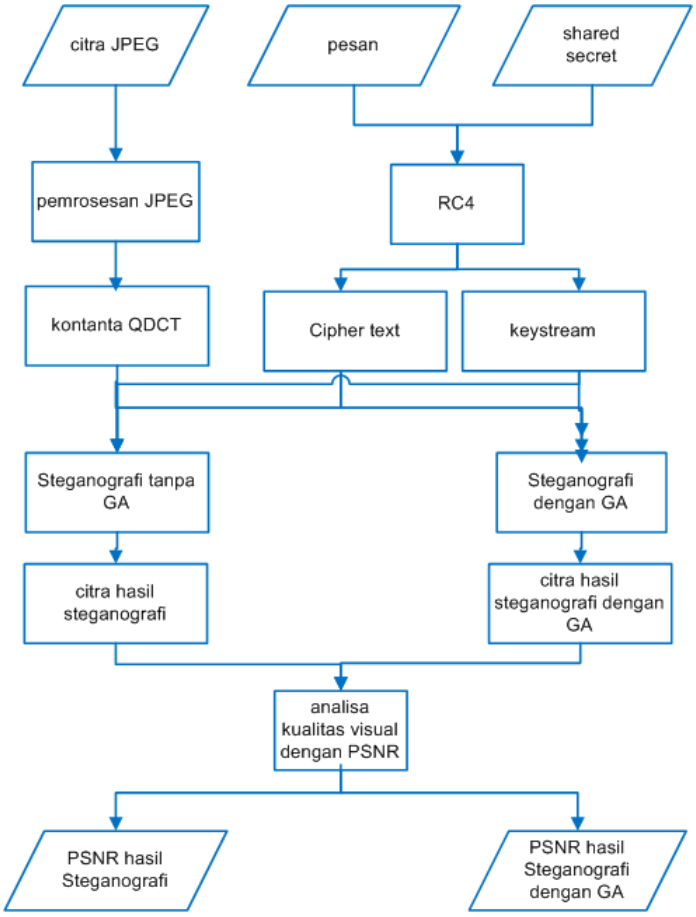
Dari pengujian variabel-variabel di atas diharapkan ditemukan konfigurasi optimal sistem untuk digunakan pada tahap berikutnya. Ukuran optimalisasi sistem dimaksud adalah:

1. Meminimalisasi distorsi visual citra akibat proses *embedding*
2. Tingkat distorsi citra akibat steganografi dengan GA harus lebih kecil dari steganografi tanpa GA.
3. Pesan yang disembunyikan ke dalam citra dapat di-*retrieve* kembali dengan menggunakan *pseudo-random number generator* yang di-*seed* dengan *shared secret* yang sama dengan yang digunakan pada saat *embedding*

Tahap analisis dilakukan dengan menguji semua basis data yang akan diuji menggunakan prosedur dan dilakukan dengan prosedur yang sama. Diharapkan

dengan menjaga kesamaan prosedur dan lingkungan pengujian akan didapatkan hasil klasifikasi yang lebih mewakili konfigurasi yang diujikan.

Tahap ini dilakukan mengikuti prosedur pada gambar 3-10.



Gambar 3-11 Diagram Alur tahap Analisis

Penjelasan dari diagram di atas adalah sebagai berikut:

3.2.1.2 Tahap Pembandingan

Setelah menentukan konfigurasi teroptimal, sesuai dengan tujuan utama penyusunan tugas akhir ini, secara khusus akan dilakukan pengamatan atas perbandingan penggunaan algoritma steganografi tanpa algoritma genetika dan dengan menggunakan GA.

Tahap ini dilakukan dengan menghitung tingkat *Peak to Signal Noise Ratio* (PSNR) dan waktu eksekusi kedua algoritma menggunakan data yang didapatkan

dari tahap analisis. Kemudian membuat analisis perbandingan antara kedua algoritma.

Sebagian besar prosedur pembandingan sama dengan prosedur pengujian pada tahap sebelumnya. Hanya saja, pada tahap ini konfigurasi sistem telah ditetapkan dan tidak memerlukan perulangan untuk konfigurasi-konfigurasi lain.

3.2.1.3 Tahap uji coba (pemakaian aplikasi)

Tahap demonstrasi adalah tahap pengaplikasian sistem. Setelah mendapatkan konfigurasi sistem yang optimal dari tahap-tahap sebelumnya. Pada tahap ini, konfigurasi optimal tersebut digunakan untuk melakukan proses steganografi pada citra.

3.2.2 Analisis Data

Perancangan data dibutuhkan sebagai batasan bagi aplikasi untuk melakukan proses. Perancangan data masukan akan diuraikan pada sub bab data masukan sedangkan hasil dari proses Steganografi akan diuraikan dalam perancangan data keluaran. Data proses adalah data yang dipakai dan atau dihasilkan selama proses.

3.2.2.1 Masukan

Secara umum, data yang digunakan sebagai masukan dalam proses klasifikasi ini merupakan data citra dalam format JPEG, pesan yang akan disipkan dan suatu *shared secret* (*password*).

Citra dalam format JPEG akan dibaca nilai pixelnya kemudian dirubah menjadi ruang warna YCbCr. Nilai YCbCr ini akan dirubah menjadi domain frekuensi dengan DCT. Kemudian nilai ini akan dikuantisasi dengan matriks kuantisasi untuk menghasilkan

Untuk memudahkan perhitungan, citra yang digunakan sebagai inputan harus berukuran panjang dan lebar masing-masing dengan nilai kelipatan 8. Ini karena perhitungan DCT, dilakukan terhadap blok citra dengan ukuran tetap sebesar 8x8.

Untuk tahapan uji coba citra yang dipakai berukuran antara 80x80 hingga 200x200 pixel. Ini karena dalam proses GA, setiap blok akan diolah secara sendiri dan bergantian. Sehingga semakin banyak jumlah pixel di dalam citra akan semakin lama waktu yang diperlukan untuk melakukan steganografi dengan algoritma genetika ini.

Pesan yang akan disembunyikan berupa file. Inputan ini akan dibaca menjadi *array of bytes* kemudian dienkripsi dengan menggunakan RC4 yang di-*seed* dengan *shared secret* yang diberikan.

Shared secret ini berupa string dengan panjang $0 < key < 255$ byte yang akan dibaca sebagai *array of bytes* dan digunakan untuk *seed pseudo-random number generator* dalam RC4.

Ukuran bit pesan yang akan disispkan ke dalam citra dibatasi oleh jumlah konstanta Quantized DCT (QDCT) yang tersedia sebagai medium penyembunyian pesan. Semakin besar ukuran citra maka jumlah QDCT yang tersedia, dan panjang pesaan yang bisa disisipkan semakin besar. Namun semakin banyak pula waktu yang diperlukan untuk melakukan proses steganografi.

3.2.2.2 Data Proses

Data yang digunakan selama proses bisa dilihat pada tabel berikut.

Tabel 3-1 Data Proses

No	Nama Data	Jenis Data	Keterangan
1.	*YCbCr Image*	Matriks nilai piksel dalam ruang warna YCbCr	Digunakan untuk menghitung nilai DCT dari citra dan untuk menghitung nilai fitness pada algoritma genetika
2.	*DCT Image*	Matriks nilai konstanta DCT dari citra	Digunakan untuk menghitug Quantized DCT. Terdapat tiga matriks untuk masing-masing *layer* Y,Cb dan Cr

3	*QDCT Image*	Matriks nilai Quantized DCT	Hasil pembagian dari matriks DCT dengan matrrix kuantisasi. Digunakan sebagai *embedding medium* pesan yang akan disembunyikan dan sebagai inisiator populasi awal pada GA
4	*Input Byte*	Data pesan yang akan dimasukkan dalam bentuk *array of bytes*	Hasil pembacaan file inputan pesan yang akan dienkripsi terlebih dahulu dalam RC4
5	*Keystream*	*Keystream* hasil RC4	Digunakan untuk enkripsi pada RC4 dan membuat *Jumping Sequence* untuk seleksi konstanta QDCT untuk setagnografi
6	*Cipher Byte*	Data pesan yang sudah dienkripsi dengan RC4	Akan disisipkan dengan steganografi
7	*Jumping Sequence*	*Array of integer* untuk seleksi QDCT	Digunakan untuk untuk seleksi konstanta QDCT . Nilai pada elemen indeks tertentu menunjukkan jumlah konstanta QDCT yang di-*skip* oleh steganografi sebelum bit pesan dengan indeks tersebut disisipkan ke QDCT citra.

3.2.2.3 Keluaran

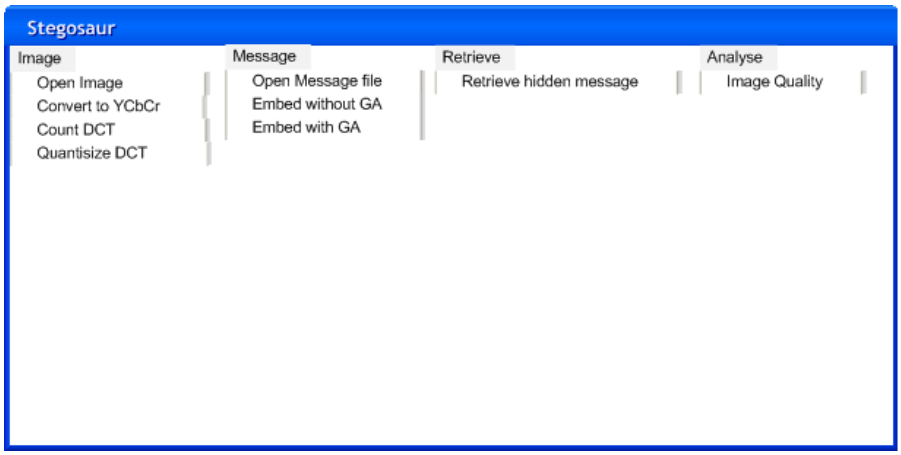
Data keluaran dari pengerjaan tugas akhir ini terbagi sesuai dengan tahapan-tahapan yang dilaksanakan dalam pengerjaan tugas akhir ini.

Keluaran akhir berupa citra dengan hasil steganografi yang nemiliki informasi tersembunyi di dalamnya. Citra hasil ini akan dibandingkan terhadap citra asal untuk mengetahui tingkat kualitas Citra hasil steganografi disebut *stego image*. Dalam citra ini sudah tersembunyi pesan rahasia dalam konstanta QDCT.

Dengan menggunakan *shared secret* yang sama yang digunakan untuk memilih konstanta QDCT maka akan dapat diambil dan diketahui kembali pesan yang disembunyikan. Namun bila pada tahap *retrieval* ini *key* yang dimasukkan sebagai seed tidak sama dengan yang digunakan untuk penyisipan, maka hasil seleksi akan salah dan pesan tidak dapat diambil dengan tepat.

3.2.3 Perancangan Antarmuka

Perancangan antar muka dimaksudkan untuk memudahkan pengguna dalam menjalankan perangkat lunak. Antar muka yang digunakan akan menggunakan paradigma *Multiple Document Interface* (MDI) dimana *window-window* bagian dari aplikasi akan ditampilkan dalam satu *window* utama. Modul dalam aplikasi akan bisa diakses dengan *menu item* yang disusun ke dalam menu bar di bagian atas dari *window* utama. Window modul akan muncul dan diletakkkan di *window* utama ini.



Gambar 3-12 Rancangan Antar Muka

Antar muka pada perangkat lunak ini terdiri dari modul:

1. **Modul Image**: Antar muka ini digunakan untuk pemrosesan citra JPEG sebelum dilakukan proses steganografi. Dalam hal ini adalah membuka citra input, konversi ruang warna RGB ke YCbCr, perhitungan DCT dan kuantisasi DCT.

   Gambar diatas menunjukkan bagian dari modul Image ini. Menu modul ini dibuat dalam bentuk Menu Bar, dengan tiap tahapannya menjadi menu item dari menu utama.

   Menu item pertama adalah *Open Image*. Menu ini digunakan oleh pengguna untuk memilih citra JPEG yang akan digunakan untuk cover image dalam proses steganografi.

   Menu item kedua adalah *Convert to YCbCr*. Menu item ini digunakan untuk melakukan digunakan untuk merubah ruang warna di citra input dari ruang warna RGB menjadi ruang warna YCbCr.

   Menu item ketiga adalah *Count DCT*. Menu item ini digunakan untuk melakukan perhitungan DCT untuk merubah citra dari domain spasial menjadi domain frekuensi.

   Menu item keempat adalah *Quantisize DCT*. Menu item ini digunakan untuk melakukan kuantisasi terhadap konstanta DCT hasil perhitungan sebelumnya dengan matriz kuantisasi yang sudah ditentukan sebelumnya.

2. **Modul Message:** Antar muka ini digunakan untuk mengolah pesan yang akan disembunyikan dan melakukan steganografi dengan metode yang dipilih apakah menggunakan algoritma genetika atau tidak. Menu modul ini dibuat dalam bentuk Menu Bar, dengan tiap tahapannya menjadi menu item dari menu utama.

   Menu modul ini dibuat dalam bentuk Menu Bar, dengan tiap tahapannya menjadi menu item dari menu utama.

Menu item pertama adalah *Open Image*. Menu item ini digunakan untuk memilih pesan yang akan disipkan dalam proses steganografi kedalam citra.

Menu item kedua adalah *Embed without GA*. Menu item ini digunakan utnuk menyisipkan pesan ke dalam citra dengan menggunakan *pseudo-random number generator* yang di-seed dengan menggunakan *shared secred (password)* yang akan dimasukkan user pada form yang akan ditampilkan.

Menu item ketiga adalah *Embed with GA*. Menu item ini digunakan utnuk menyisipkan pesan ke dalam citra dengan menggunakan proses yang sama, hanya saja dengan menggunakan GA untuk melakukan optimasi kualitas citra.

3. **Modul Retrieve:** Antar muka ini digunakan untuk mengecek apakah pesan sudah dimasukkan ke dalam citra dengan benar dan dapat dibaca ulang jika dan hanya jika menggunakan *shared secred* yang tepat.

   Menu item pada modul ini adalah Retrieve hidden message. Menu item ini akan meminta user memasukkan suatu shared secret, yang kemudian digunakan untuk men-*generate pseudo-random number generator* yang dicoba digunakan untuk membaca pesan terdsembunyi dari citra

   4. **Modul Analyse:** Antar muka ini digunakan untuk mengecek kualitas citra setelah terjadi proses steganografi Menu item pada modul ini adalah *Analyse visual quality*. Menu item ini akan menghitung kualitas visual citra sebelum., dekuantisasi, dan sesudah disisipi pesan.

3.3 Pembuatan Sistem

Setelah melalui proses perancangan, dilakukan pembuatan sistem. Tahap-tahap pengimplementasian mencakup lingkungan implementasi, antarmuka dan program dalam *script* Java, yang akan dijelaskan dalam subbab berikut.

3.3.1 Lingkungan Implementasi

Perangkat lunak diimplementasikan pada lingkungan sebagai berikut:

- Perangkat keras

  Perangkat lunak ini diimplementasikan pada sebuah komputer desktop dengan spesifikasi prosesor Intel Pentium IV 2.0 GHz dan memori RAM 1 GB.

- Perangkat lunak

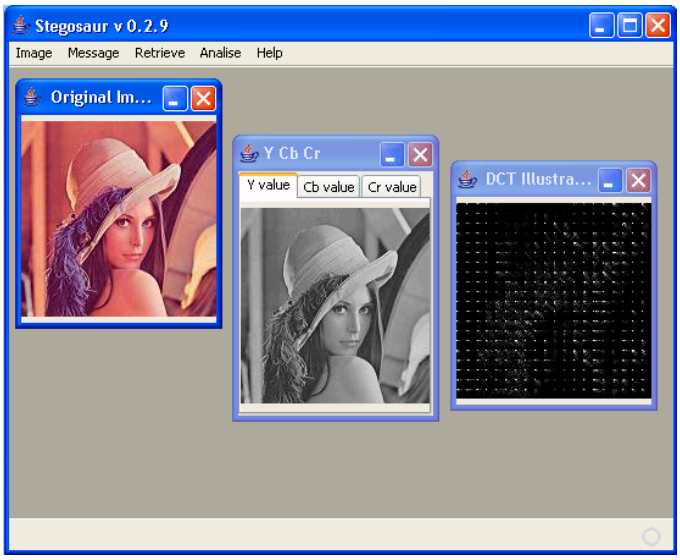
  Perangkat lunak ini dikembangkan pada sistem operasi Microsoft Windows XP Service Pack 2 dengan menggunakan Netbeans 6 Beta 2.

3.3.2 Antarmuka dan Program

Dalam bagian ini akan dijelaskan beberapa implementasi dari desain antarmuka pada sub bab 3.2.3 dan potongan program dalam kode matlab yang akan menjalankan proses-proses seperti dijelaskan pada sub bab 3.1.

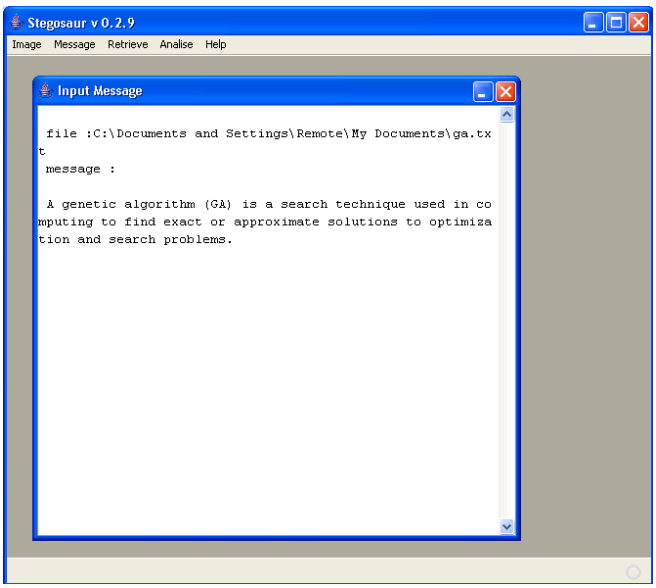
3.3.2.1 Antarmuka aplikasi

Seperti disebutkan pada sub bab 3.2.1 aplikasi yang akan dibangun ini menggunakan *Multiple Document Interface* (MDI) dimana modul akan berupa *window* bagian yang ditampilkan dalam suatu *window* utama.



Gambar 3-13 Implementasi modul Image

Gambar 3-10 adalah gambar modul analisis. Terlihat pada gambar diatas masukan yang bisa dimasukkan oleh pengguna adalah bab 2.4.



Gambar 3-14 Implementasi modul Message

Gambar 3-10 adalah gambar modul analisis. Terlihat pada gambar diatas masukan yang bisa dimasukkan oleh pengguna adalah bab 2.4.

3.3.2.2 Snippet kode program

Pada bagian berikut akan ditampilkan potongan kode Java yang mengimplementasikan proses yang dibahas sebelumnya.

1. Implementasi pemrosesan cover citra JPEG

   A. Membuka *cover image*

   Citra yang akan digunakan sebagai *cover image* adalah citra dengan format JPEG. Citra ini akan dibaca nilai per pikselnya untuk tiap layer dalam ruang warna RGB.

```
public Image makeRGB (String filename) {

myRgbImage =
Toolkit.getDefaultToolkit().getImage(filename);

PixelGrabber grabber = new PixelGrabber
                  (myRgbImage, 0,0, imageWidth,
                  imageHeight,true);
      try
        {
        grabber.grabPixels();
        }
        catch (Exception e)
        {System.out.println("PixelGrabber
exception"); }

// mengalokasikan data per pixel
totalPixel = imageWidth * imageHeight;
redPixels   = new int[imageWidth][imageHeight];
greenPixels = new int[imageWidth][imageHeight];
bluePixels  = new int[imageWidth][imageHeight];
int[] tempPixels = (int [])grabber.getPixels();

rgbPixels = (int [])grabber.getPixels();
int index = 0;
int width = 0;
int height = 0;
```

```
for (height = 0; height < imageHeight ; height++)
    {
    for (width = 0; width < imageWidth ; width++)
      {
    Color cpix = new Color(rgbPixels[index]);
    redPixels[width][height]   = cpix.getRed();
    greenPixels[width][height] = cpix.getGreen();
    bluePixels[width][height]  = cpix.getBlue();

    tempPixels[index] = (new
            Color(redPixels[width][height],
            greenPixels[width][height],
            bluePixels[width][height])).getRGB();
           index++;
       }
      }
          myRgbImage = rebuiltImage();
          return myRgbImage;
}
```

Gambar 3-15 Implementasi modul membuka cover image

B. Konversi ruang warna RGB ke YCbCr

Baris di bawah ini mengimplementasikan konversi ruang warna RGB ke ruang warna YCbCr dengan menggunakan rumus 2.1

```
  public void makeYCbCr(StegoRgb imageRgb)
      {
  // jumlah blok yang ada dalam satu baris Y
  blokWidth = (imageHeight/8);
  // jumlah keseluruhan blok di citra
  int jumlahBlok = blokWidth * blokWidth ;
  // matrik 2d untuk menyimpan blok DCT
  blokDctY =  new int[jumlahBlok][64];
  blokDctCb =  new int[jumlahBlok][64];
  blokDctCr =  new int[jumlahBlok][64];
```

```
          /* formula
          Y = 0.299 * R + 0.587 * G + 0.114 * B
          U = 0.436 * (B - Y) / (1 - 0.114)
          V = 0.615 * (R - Y) / (1 - 0.299)
           */

for (int coX = 0; coX < imageWidth; coX++)
  {
   for (int coY = 0; coY < imageHeight; coY++)
      {
// rubah ke double untuk presisi hitungan
double R = imageRgb.getRedPixel(coX,coY);
double G = imageRgb.getGreenPixel(coX,coY);
double B = imageRgb.getBluePixel(coX,coY);

// menghitung nilai Ycbcr dengan rumus Matlab // -
-> tools/images/rgb2ycbcr.m

// 1/255 --> trasform RGB[0 255] ke RGB[0 1]
double c =  0.0039215686;

double y  = 16 + (65.481 * R * c) + (128.553 * G *
c) + (24.966 * B * c);

double cb = 128 - (37.797 * R * c) - (74.203 * G *
c) + (112 * B * c);

double cr = 128 + (112 * R * c) - (93.786 * G * c)
- (18.214 * B * c);

yPixels[coX][coY] = round(y);
cbPixels[coX][coY] = round(cb);
crPixels[coX][coY] = round(cr);
            }
        }
    }
```

Gambar 3-16 Kode modul konversi RGB ke YCbCr

C. Perhitungan DCT

Baris di bawah ini mengimplementasikan rumus perhitungan DCT untuk blok citra dengan ukuran 8x8.

```
// membuat blok dct 2 dimensi
public int[] makeBlokDct2(int indexBlok)
{
 tempY =  new double[8][8];
 tempCb = new double[8][8];
 tempCr = new double[8][8];
 for(int index = 0;index < 64; index++)
 {
  // substraksi Citra
  // dan memasukkan data ke blok pixel
  int coX = ((indexBlok % blokWidth) * 8)
+ (index%8);
  int coY = ((indexBlok / blokWidth) * 8)
+ (index/8);

tempY[index%8][index/8] = yPixels[coX][coY] -128;
tempCb[index%8][index/8]  =  cbPixels[coX][coY]  -
128;
tempCr[index%8][index/8]  =  crPixels[coX][coY]  -
128;
 }
  double resultY, resultCb, resultCr;
// coX dan coY adalah u dan v di paper
// batasnya sampai semua image
  for (int coX = 0; coX < 8; coX++)
   {
   for (int coY = 0; coY < 8; coY++)
     {
resultY = constant(coX) * constant(coY)
          * sigmaCos(tempY, coX,coY);
resultCb = constant(coX) * constant(coY)
         * sigmaCos(tempCb, coX,coY);

resultCr = constant(coX) * constant(coY)
```

```
         * sigmaCos(tempCr, coX,coY);

 blokDctY[indexBlok][coX+(coY*8)]=round(resultY);
 blokDctCb[indexBlok][coX+(coY*8)]=round(resultCb);
 blokDctCr[indexBlok][coX+(coY*8)]=round(resultCr);
     }
   }
   return  blokDctY[indexBlok];
}
/** menghitung nilai double sigma untuk DCT 2d */
public double sigmaCos
(double[][] temp, int coX,int coY)
 {
 double result = 0;
 for(int indexX = 0; indexX < 8; indexX++)
   {
   for(int indexY = 0; indexY < 8; indexY++)
       {
        result += temp[indexX][indexY]
 * Math.cos(((2*indexX) +1)
 * coX * Math.PI *0.0625000)
 * Math.cos(((2*indexY) +1) * coY
 * Math.PI * 0.0625000);
       }
     }
     return result;
}
public double constant(int input)
{
 double result;
 if(input == 0)
      result =  Math.sqrt(0.125);
 else
      result = Math.sqrt(0.25);
      return result;
}
```

Gambar 3-17 Kode Perhitungan DCT

D. Kuantisasi konstanta DCT

Baris di bawah ini mengimplementasikan perhitungan kuantisasi DCT pada citra dengan matriks kuantisasi yang sudah ditentukan dari persamaan 2.7.

Inputan dari tahap ini adalah suatu *array* satu dimensi dari 64 konstanta nilai DCT dan output juga berupa array satu dimensi kontanta QDCT

```
/** matriks quantisasi */
double [][] quantMatriks =
             {{ 8, 10, 10, 2, 2, 2, 2, 3},
               {10, 10, 2, 2, 2, 2, 3, 3},
               {10, 2, 2, 2, 2, 3, 3, 3},
               {2, 2, 2, 2, 2, 3, 3, 4},
               {2, 2, 2, 2, 3, 3, 4, 4},
               {2, 2, 2, 3, 3, 4, 4, 5},
               {2, 2, 2, 3, 3, 4, 5, 6},
               {2, 2, 3, 3, 4, 5, 6, 8}};
/** fungsi untuk kuantisasi blok DCT */
public int[] quantDct(int[] blokInput)
{
 int result[] = new int[64];

 for(int indexX = 0; indexX < 8; indexX++)
  {
    for(int indexY = 0; indexY < 8; indexY++)
       {
int value=round(blokInput[indexX+(indexY*8)]
           / quantMatriks[indexY][indexX] );

result[indexX +(indexY*8)] = value;
        }
  }
  return result;
 }
```

Gambar 3-18 Kode Perhitungan Kuantisasi DCT

E. Dekuantisasi konstanta DCT

Baris di bawah ini mengimplementasikan persamaan 2.8 untuk melakukan dekuantisasi pada konstanta QDCT untuk menjadikannya sebagai DCT bisa kembali. Proses ini adalah kebalikan dari proses 1.D.

Inputan dari tahap ini adalah suatu array satu dimensi dari 64 konstanta nilai QDCT dan output juga berupa array satu dimensi kontanta DCT

```
  /** fungsi untuk dequntisasi DCT */
public int[] dequantDct(int[] blokInput)
{
int result[] = new int[64];

for(int indexX = 0; indexX < 8; indexX++)
 {
 for(int indexY = 0; indexY < 8; indexY++)
  {
   // dihitung dengan Hadamard product
   result[indexX + (indexY*8)] =
         round( blokInput[indexX + (indexY*8)]
         * quantMatriks[indexX][indexY] );
   }
 }
   return result;
}
```

Gambar 3-19 Kode PerhitunganDekuantisasi DCT

F. Perhitungan IDCT

Baris di bawah ini adalah impolementasi dari persamaan persamaan 2.8. Proses ini merubah citra dari domain frekuensi menjadi domain spasial. Proses ini adalah kebalikan dari proses 1.C.

Inputan dari proses ini adalah array satu dimensi yang berisi 64 konstanta DCT dan outputnya berupa array dua dimensi dari nilai pixel citra dalam ruang warna YCbCr dengan ukuran 8x8.

```
  /** inverse DCT 2D */
  public int[][] inverseBlokDct2
  (int inputBlok[])
      {
  // inverse dari fungsi makeBlokDct2
  // inputan adalah index blok DCt,
  // lalu blok tersebut dihitung inversenya
  // dan disimpan dalam matriks 2d blokInverse

     tempY =  new double[8][8];
     int[][] result = new int[8][8];

  for(int index = 0;index < 64; index++)
     tempY[index%8][index/8]= nputBlok[index];

  // coX dan coY adalah u dan v di paper,
  // batasnya sampai semua image

  for (int coX = 0; coX < 8; coX++)
  {
     for (int coY = 0; coY < 8; coY++)
     {
    double value = sigmaInverse(coX,coY);
    result[coX][coY] = (round(value)) + 128;
     }
  }

   return result;
}

public double sigmaInverse(int coX, int coY)
{
    double result = 0;
    for(int indexX = 0; indexX < 8; indexX++)
    {
      for(int indexY = 0; indexY < 8; indexY++)
      {
```

```
      // 1/16 = 0.0625
  result += constant(indexX)
  * constant(indexY) * tempY[indexX][indexY]
  * Math.cos(((2*coX) +1) * indexX
  * Math.PI * 0.0625000) * Math.cos(((2*coY)+1)
  * indexY * Math.PI * 0.0625000);
      }
    }
    return result;
}
```

Gambar 3-20 Kode Perhitungan IDCT

2. Implementasi Stream Cipher RC4

   A. Inisialisasi *Key-scheduling algorithm*

   Baris dibawah ini merupakan implementasi dari proses stream cipher RC4. Inputan dari proses ini adalah atraay of bit dari *shared secred* (*password*) yang digunakan untuk enkripsi pesan dan membuat *keystream* untuk seleksi konstanta DCT pada saat steganografi.

   Array of bytes ini akan digunakan untuk melakukan *seeding pseudo-random number* pada fungsi `rc4Init`. Proses initialisasi ini emenggunakan teori permutasi bilangan untuk membuat keystream yang panjangnya nati bergantung dan sama dengan panjang dari citra inputan.

```
 private static final int
 ARRAY_SIZE = 256,
 BLOCK_SIZE = 1;

 private byte[] sArray = new byte[ARRAY_SIZE]; //
 State array atau "sbox"

 // keystream dan output
 public byte[] keystream, output;
 public String keyString, outputString;
```

```
 private int x,y; // i, j

 // insialisasi user
 public void engineInit(byte[] key)
     throws InvalidKeyException
 {
  if (key == null )
    throw new InvalidKeyException("Null key");

 if (key.length < 1 || key.length > 256)
   throw new InvalidKeyException
 ("Invalid   key   length   (req.   1-256   bytes
 "+key.length+")");
  rc4Init(key);
}
/** Inisialisasi cipher */
public void rc4Init(byte[] key)
{
// mengisi state array
for (int i = 0; i < ARRAY_SIZE; i++)

sArray[i] = (byte)i;
x = 0;
byte t; // tmp
for (int i = 0; i < ARRAY_SIZE; i++)
 {
 x = ((key[i % key.length] & 0xff)
      + sArray[i] + x) & 0xff;
 // swap
 t = sArray[i];
 sArray[i] = sArray[x];
 sArray[x] = t;
 }
 x = y = 0;
}
```

Gambar 3-21 Kode Perhitungan RC4

B. Pseudorandom number generator

Proses ini adalah tahapan kedua dari implementasi dari *stream cipher* RC4. Pada proses ini inputan adalah pesan yang akan dienkripsi dalam bentuk *array of bytes*. Dengan opreasi bitwise XOR, pesan ini akan dienkripsi dengan *keystream* yang telah dihasilkan pada proses sebelumnya.

Keluaran dari proses ini adalah cihper text hasil enkripsi dalam bentuk *array of bytes.* Cipher text inilah yang bitnya akan disisipkan pada proses steganografi.

```
/** Enkripsi dengan Algoritma RC4 */
public byte[] engineCrypt(byte[] input)
{

// inisialisasi variabel
int outIndex = 0;
output = new byte[input.length];
keystream = new byte[input.length];
outputString = "";
keyString = "";

byte t; // tmp
for (int i = 0; i < input.length; i++)
    {
    x = ((x + 1) & 0xff) % 256;
    y = ((sArray[x] + y) & 0xff) % 256;

    // swap
    t = sArray[x];
   sArray[x] = sArray[y];
   sArray[y] = t;

   // xor input dengan keystream
  byte xor =  sArray[(sArray[x]
              + sArray[y]) & 0xff];
```

```
  keystream[outIndex] = xor;
  keyString += " "+  byteToHex(xor);

  output[outIndex] = (byte)(input[i] ^ xor );
  outputString += " "+
              byteToHex(output[outIndex]);

if((outIndex % 15) +1  == 0)
   outputString += "\n";

   outIndex++;
     }
  return output;
}
```

Gambar 3-22 Kode Perhitungan PRNG

3. Implementasi proses steganografi

   A. Inisialisasi *jump sequence* dari *keystream pseudo-random number*

   Baris dibawah ini adalah implementasi dari tahapan steganografi. Pada proses ini dibuat *jump sequence* untuk menseleksi konstanta DCT secara pseudo-random.Inputan dari proses ini adalah keystream yang dihasilkan oleh RC4 pada proses 2.B.

   Keystream dan pada akhirnya *jump sequence* ini bersifat *pseudo-random*, yang tidak dapat dibuat atau diketahui tanpa menggunakan *shared secret* yang sama untuk seeding pada *pseudo-random number generator*.

   Dengan demikian seleksi DCt dan lokasi penyembunyian pesan tidak akan dapat diketahui kecuali oleh target penerima yang sudah mengetahui *shared secred.*

```
/** membuat jumping sequence dr keystream */
public int[] makeJump(int[] keystream)
{
int[] result =  new int[keystream.length+16];
```

```
System.out.print(" \n Jump sequence :");
for(int i = 0; i < keystream.length; i++)
  {
// mengeset interval agar embedding merata
if(i < 16)
result[i] = 0;
else if(i >= 16 && i%8 == 7)
 {
    int val1 = 2 * (keystream.length - 32);
    int val2 = 1 + keystream.length - i;
    result[i] =  (val1 / val2 );
   }
else
    result[i] = keystream[i-16];

    System.out.print(" "+result[i]);
 }

  System.out.print(" \n");
  return result;
}
```

Gambar 3-23 Kode pembuatan jump sequence dari RC4

B. Seleksi konstanta DCT dengan *jump sequence*

Baris dibawah ini adalah implementasi dari proses seleksi konstanta QDCT yang ada di *cover image* dengan menggunakan *jump sequence* yang dihasilkan pada proses 3.B.

Pada proses steganografi ini, konstanta QDCT yang ada di *cover image* akan dibaca satu persatu. Bila nilai QDCT adalah 0 atau 1 maka akan dilewati (di-*skip*). Ini karena data pada 0 atau 1 dalam sistem bilangan *binary* hanya direpresentasikan dalam satu bit saja. Sehingga tidak boleh dirubah LSB-nya.

Sedangkan bilai nilai QDCT bukan 0 atau 1 akan dicek apakah konstanta tersebut akan dilewati atau akan disipi dengan bit dari

pesan. Ini ditentukan oleh nilai dari *jump sequence* untuk bit dari pesan tersebut.

Bila seluruh bit dari pesan sudah disisipkan ke dalam QDCT *cover image* maka seluruh dari sisa QDCT citra hasil steganografi (*stego image*) akan diset sama dengan QDCT pada citra asal, tanpa ada perubahan.

```
/** embedding satu layer Y, Cb atau Cr */
 public int embedQdct
 (String mode, int im, int[] embedded)
 {
  int totalBlok = (oDct.getHeight()
                  * oDct.getHeight()) /64;
  StegoMessage mess = new StegoMessage();

  // melakukan embeddeng untuk tiap blok
  for(int ib = 0; ib < totalBlok; ib++)
  {
     for(int id = 0; id < 64; id++)
     {
   int dv = oDct.getQdctValue(mode,ib, id);

  // jika nilai DCT itu 0 atau 1, skip
    if(dv == 0 || dv == 1)
          rDct.setQdctValue(mode,ib , id, dv);
// jika bukan 0 dan 1
// dan masih ada pesan yang bisa disisipkan
    else if(im<embedded.length && jump[im]==0)
    {
// cek apakah perlu melewati ini
    int v = embedded[im];

rDct.setQdctValue(mode,ib,id,mess.setLsb(dv,v));
im++;
    }
  // jika bukan 0 dan 1, tapi skip
  else if(im < embedded.length && jump[im] > 0) {
```

```
   jump[im]--;
   rDct.setQdctValue(mode, ib , id, dv);
   }
  // kalau pesan yang disipkan dah habis
    else
      rDct.setQdctValue(mode, ib , id, dv);
  } // end for one blok
 } // end for  one layer
        return im;
}
```

Gambar 3-24 Kode Penyisipan Pesan ke dalam QDCT

C. Embedding bit pesan pada QDCT

Baris di bawah ini adalah implementasi dari penyisipan bit pesan ke LSB dari konstanta QDCT yang sudah dipilih dalam proses 3.B. Penyisipan ini dilakukan operasi Bitwise XOR. Bila LSB konstanta QDCT sudah sama dengan bit dari pesan yang akan disisipkan maka tidak ada perubahan pada konstanta tersebut. Namun bila berkebalikan maka nilai konstanta QDCT akan di XOR dengan byte 1, sehingga di-*toggle* atau berubah dari nilai bit LSB 0 emnjadi bit 1 dan sebaliknya.

```
/** set LSB dengan nilai 0 atau 1 */
public int setLsb(int input, int value)
{
// input dalam bentuk int
byte dctByte = (byte) ( input & 0x00FF);

// hasil dari fungsi ini,
int result = input;
byte resultByte;

 // byte untk men-toggle LSB dengan XOR
 byte xor = 0x01;

 // ambil Least Significant Bit
```

```
  int lsb = dctByte & 0x0001;

  // jika sudah sama antara LSB
  // dan nilai yang akan diset itu bagus
  if( lsb == value )
              return result;
  // tapi jika tidak,, ubah LSB
  else
   {
   resultByte = (byte) ((byte) dctByte ^ xor);
   result = (int) resultByte;
   return result;
   }
}
```

Gambar 3-25 Kode Modifikasi LSB

4. Implementasi Steganografi dengan GA

   A. Inisialisasi algoritma genetika

   Baris di bawah ini adalah implementasi dari GA untuk optimalisasi proses steganografi. Tahapan ini menginisialisasi kelas GA dengan parameter-parameter yang telah ditetapkan.

   Adapaun parameter tersebut adalah:

   a. Jumlah gen di satu kromosom
   b. Jumlah populasi dalam satu generasi
   c. Kemungkinan terjadi *crossover* (antara 0 hingga 1)
   d. Jumlah peluang seleksi acak dalam persen.
   e. Jumlah preliminary run untuk membuat populasi awal yang bagus
   f. Jumlah maximal *prelimnary run*. Dalam sistem ini *preliminary run* tidak diperlukan karena kromosom diinisialisasi dengan nilai QDCT dari blok yang akan dioptimasi.
   g. Peluang terjadinya mutasi (antara 0 sampai 1)

h. Metode *crossover* yang digunakan Dalam GA ini disediakan tiga macam jenis *crossover* yaitu:
    a. *One Point Crossover*
    b. *Two Point Crossover*
    c. *Uniform Crossover*
i. Presisi dari gen dalam kromosom. Karena sistem ini menggunakan bilangan integer makajumlah presisi yang diperlukan adalah 0 angka di belakang koma.
j. Nilai *boolean* apakah gen dalam kromosom bernilai positif saja. Dalam sistem ini gene adalah representasi dari QDCT. Karena QDCT dapat bernilai positif ataupun negatif, maka nilai ini diset *false*.
k. Nilai *boolean* apakah menghitung statistik dari generasi. Nilai statistik yang dihitung adalah rata-rata (*mean*, μ) dan variansi (σ)
l. Nilai awal dari konstanta QDCT dalam bentuk array of integer satu dimensi yang terdiri atas 64 elemen.
m. Pesan yang akan disisipkan ke dalam konstanta QDCT. Dalam proses perhitungan fitness, akan disimulasikan penyisipan bit pesan dalam QDCT dan hasilnya (*stego image*) akan dievaluasi kualitasnya terhadap citra asal untuk mendapatkan nilai *fitness*
n. *Jump sequence* yang digunakan untuk seleksi konstanta QDCT pada proses peyisipan bit pesan.
o. *Starting point* atau titik awal mulai dari index bit pesan ke berapa proses penyisipan akan dimulai.
p. *Target* nilai pixel citra dalam ruang warna YCbCr yang diinginkan untuk hasil dari proses GA ini. Semakin mendekati nilai dari target ini maka kualitas citra hasil semakin bagus.

```
public StegoGenetic(int[] inputGene,
       int[] embeddedMessage, int[] sequence,
       int startingPoint, int[][] target
       ) throws GAException
{ super(64, // jumlah gene di satu kromosom
       10, //jumlah populasi di satu generasi
       0.9, // kemungkinan terjadi crossover
       6, // jumlah peluang seleksi acak
       15, // jumlah total generasi
        0, // jumlah prelim runs
          // untuk membentuk populasi bagus,
      // tidak perlu, populasi diinit dari DCT
      0, // max prelim generations
      0.01, // kemungkinan mutasi kromosom
      Crossover.ctTwoPoint, // tipe crossover
      0, // decimal presisi, integer maka 0
      false, // hanya positif bila true
      true,  // hitung statistik atau tidak
      inputGene, // nilai DCT awal
      embeddedMessage,
      sequence, startingPoint,
      target // Y,Cb atau Cr blok tersebut ,
      );
}
```

Gambar 3-26 Kode Inisialisasi GA

B. Penentuan *Fitness function*

Baris dibawah ini adalah implementasi dari perhitungan fungsi *fitness* untuk algoritma gentika dalam sistem ini. Untuk perhitngan ini, kromosom akan dijadikan sebagi medium untuk steganografi dengan

```
protected double getFitness(int iChromIndex)
{
  double result = 0;
  int[] qdctGene = new int[chromosomeDim];
```

```
  for(int i = 0; i< chromosomeDim; i ++)
   {
   qdctGene[i] = (int)
   this.getChromosome(iChromIndex).getGene(i);
   }

  this.doEmbed(qdctGene);

  int[][] resultGray = new int[8][8];
  resultGray = this.inverseBlokDct2(
           this.dequantDct(this.resultDct));

  result = this.countMAD(
           this.targetGray, resultGray);

return result;
}

public double countMAD(int[] input,
                       int[] output)
    {
        double result = 0;
        for(int i = 0; i < 64; i++)
     {
     result += Math.abs(input[i] - output[i]);
     }
     return result;
}
```

Gambar 3-27 Kode Perhitungan fitness function

5. Implementasi analisa kualitas citra

   Baris di bawah ini mengimplementasikan perhitungan kualitas citra dengan PSNR sebagai ukuran kualitas.

```
  /** semua data YcbCr di StegoDct dah diisi
   tinggal membuat tampilan hasil RGB
```

```
    sekaligus menghitung kualitas citra
    dalam PSNR */
public Image makeResultRgb()
  {
  rRgb = new StegoRgb();

    // menghitung PSNR
  /**  nilai Mean Square Error */
  double MSE = 0;

  rRgb.setDimension(oRgb.getWidth(),oRgb.getHeight
  ());

for(int w = 0; w < rRgb.getWidth(); w++)
{
   for(int h = 0; h < rRgb.getHeight(); h++)
  {
   int[] resRgb = rRgb.makeRgb(rDct, w, h );
   MSE += addMse(resRgb, w, h);
      }
   }
  // MSE
  double size = 3 * rRgb.getWidth() *
               rRgb.getHeight();

  System.out.print(" \n ERR : "+MSE);
  MSE = MSE / ( size);

  // PSNR
  double MAX = 255;
  double PSNR = 20 * Math.log10(MAX /
                     Math.sqrt(MSE));
  System.out.print(" \n MSE = "+MSE+" \n
                    PSNR = "+PSNR+" \n \n");
  return rRgb.rebuiltImage();
}
public int addMse(int [] res, int w, int h)
```

```
{
 double result = 0;

  result = Math.pow(
  oRgb.getRedPixel(w, h) - res[0], 2)
+ Math.pow(oRgb.getGreenPixel(w, h)-res[1], 2)
+ Math.pow(oRgb.getBluePixel(w, h)-res[2], 2);

  return (int) result;
}
```

Gambar 3-28 Kode Perhitungan Analisa Kualitas Citra

BAB 4

ANALISIS HASIL DAN PEMBAHASAN

Dalam bab ini dibahas mengenai hasil uji coba sistem yang telah dirancang dan dibuat. Uji coba dilakukan untuk mengetahui apakah sistem dapat berjalan sebagaimana mestinya dengan lingkungan uji coba yang telah ditentukan serta dilakukan sesuai dengan skenario uji coba.

Pada skenario uji coba yang akan dilakukan akan dilakukan pembandingan hasil penggunaan GA untuk proses steganografi dengan hasil steganografi yang tidak menggunakan GA.

4.1 Uji Coba 1 (Analisis Konfigurasi)

Pada uji coba satu ini, akan dilakukan pengujian terhadap semua variabel yang diharapkan akan mempengaruhi hasil akurasi dari sistem. Variabel-variabel tersebut adalah:

1. Besar dan nilai *cover image* steganografi

   Besarnya ukuran citra mempengaruhi banyaknya konstanta QDCT yang bisa digunakan untuk steganografi. Kontsanta yang bisa digunakan adalah konstanta dengan nilai yang lebih besar dari 1 atau lebih kecil dari 0. Konstanta QDCT yang terpilih akan diganti LSB-nya dengan bit dari pesan yang disembunyikan.

   Citra dengan ukuran yang sama namun nilainya berbeda juga akan memiliki jumlah konstanta QDCT berbeda dan nilai dari kontanta ini akan berbeda pula. Sehingga bila dua citra yang berbeda disisipi pesan yang sama akan mengalami distorsi visual yang berbeda pula.

2. Panjang pesan yang disisipkan proses steganografi

   Panjang citra menentukan besarnya jumlah bita yang harus disisipkan dengan steganografi ke dalam citra. Sehingga jumlah konstanta QDCT yang harus dirubah juga akan bertambah. Kontanta QDCT yang berubah akan menimbulkan distorsi citra.

3. Metode *crossover* yang digunakan dalam algoritma genetika

   Pada GA, *crossover* digunakan untuk melakukan rekombinasi kromosom dan menghasilkan kromosom anak pada populasi berikutnya dengan nilai yang baru. Uji coba akan membandingkan kualitas dari hasil steganografi dengan GA yang menggunakan metode *crossover* berbeda diatas.

4. Jumlah generasi pada GA

   Pada GA, jumlah generasi menentukan soluasi akhir. Jumlah generasi harus cukup banyak agar yerjadi konvergensi *fitness* dimana nilai fintess akan terus stabil pada nilai optimal.

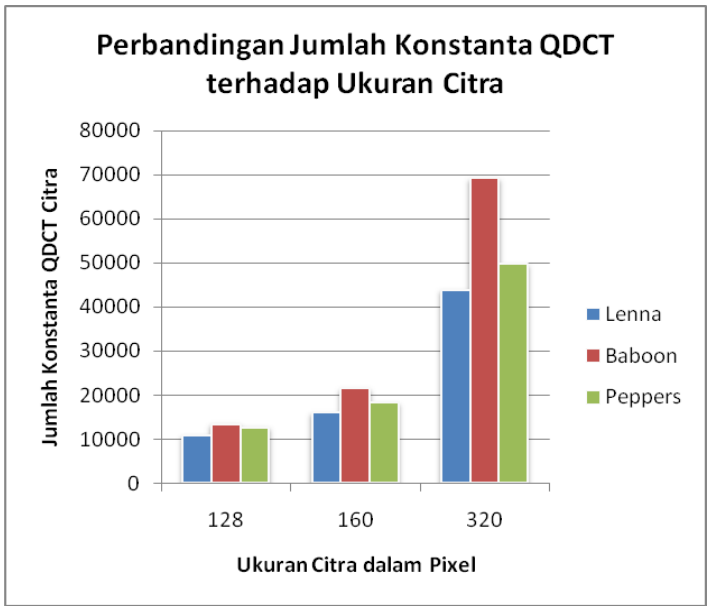
Berikut hasil pengujian dan analisis hubungan antar variabel terhadap kualitas citra hasil steganografi..

4.1.1 Ukuran *cover image* dan jumlah Konstanta QDCT

Untuk uji ini akan digunakan citra JPEG dengan ukuran dan nilai yang berbeda yang digunakan untuk sebagai cover image yang digunakan untuk menyembunyikan pesan dengan metode steganografi.

Tabel 4-1 Ukuran Citra dan jumlah Konstanta QDCT

Nama Citra	Ukuran Citra	Jumlah konstanta QDCT tersedia
lenna-128	128x128	10871
lenna-160	160x160	15980
lenna-320	320x320	43857
baboon-128	128x128	13409
baboon-160	160x160	21451
baboon-320	320x320	69157
peppers-128	128x128	12652
peppers-160	160x160	18216
peppers-320	320x320	49736



Gambar 4-1 Ukuran Citra dan jumlah Konstanta QDCT yang tersedia

Gambar 4-2 Citra Uji Baboon, Lenna, dan Peppers

Dari tabel 4.1 dan gambar 4.1 terlihat jumlah konstanta QDCT yang ada di dalam citra degan ukuran besar lebih banyak dibandingkan dengan pada citra yang lebih kecil.

Semakin besar citra *cover image* yang digunakan maka sebakin banyak konstanta QDCT citra medium yang tersedia untuk bisa digunakan sebagai medium steganografi. Peningkatan ini berbeda-beda tergantung dari nilai piksel dari citra. Semakin banyak kontanta QDCT ini memungkinkan *embedding* (penyisipan) pesan yang semakin banyak ke dalam citra. Selain itu distorsi visual yang terjadi pada citra juga tergantung pada ukuran citra.

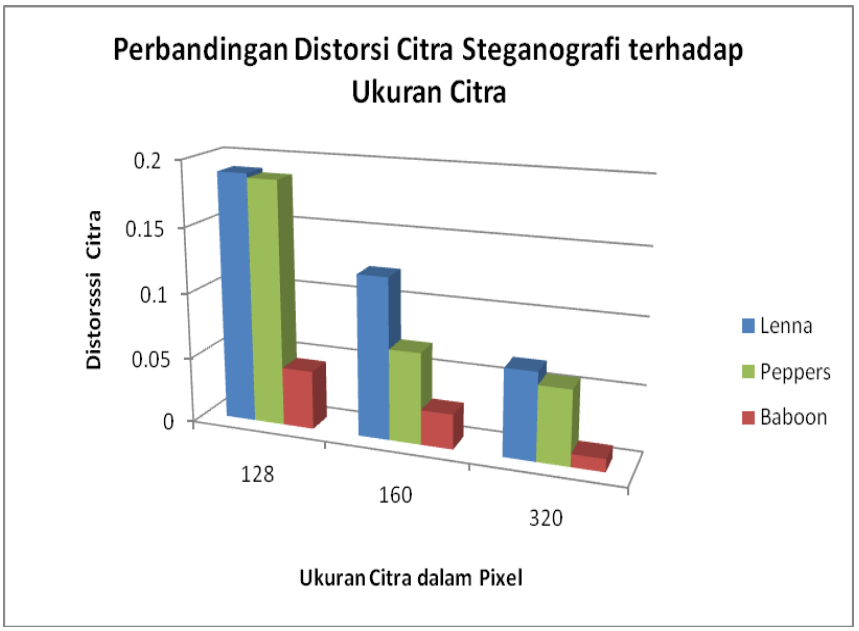
Pada bagian ini akan diamati perubahan kualitas citra dengan asumsi citra dengan ukuran yang besar jika dijadikan *cover image* untuk steganografi akan menghasilkan *stego image* dengan tingkat distorsi relatif lebih kecil bila dibandingkan citra yang lebih kecil, bila pesan yang disisipkan adalah sama.

Hipotesis 1: Semakin besar ukuran citra *cover image* dan banyak jumlah konstanta QDCT, relatif lebih sedikit distorsi akibat steganografi yang terjadi pada citra hasil.

Pada percobaan ini sembilan citra yang berbeda akan disisipi hidden message yang sama dan menggunakan *shared secre*t yang sama, sehingga yang berubah hanya ukuran citra .

Tabel 4-2 Ukuran Citra dan tingkat kualitas citra hasil steganografi dalam PSNR

Nama Citra	Ukuran Citra	Jumlah konstanta QDCT tersedia	PSNR Dekuantisasi	PSNR Steganografi	Distorsi
lenna-128	128x128	10871	38.083429	37.893873	0.189557
lenna-160	160x160	15980	38.233751	38.111451	0.122300
lenna-320	320x320	43857	41.149331	41.083393	0.065937
baboon-128	128x128	13409	35.055746	35.011231	0.044515
baboon-160	160x160	21451	33.961708	33.935346	0.026362
baboon-320	320x320	69157	36.028901	36.019230	0.009671
peppers-128	128x128	12652	39.776468	39.590176	0.186292
peppers-160	160x160	18216	40.057885	39.988753	0.069132
peppers-320	320x320	49736	41.389976	41.333577	0.056399



Gambar 4-3 Tingkat distorsi citra hasil steganografi dalam PSNR

Tampak dari tabel 4.2 dan gambar 4.3, distorsi terbanyak muncul pada citra yang memiliki ukuran kecil dan jumlah konstanta QDCT kecil. Sedangkan pada citra yang berukuran besar dan memiliki lebih banyak konstanta QDCT, bila disisipkan pesan yang sama perubahan ini akan relatif lebih sedikit.

4.1.2 Panjang pesan disisipkan (*hidden message*)

Proses steganografi menyisipkan pesan ke dalam citra dengan merubah nilai LSB dari konstanta QDCT *cover image*. Perubahan ini menyebabkan terjadinya distorsi pada citra. Pada bagian ini akan diamati perubahan visual citra dengan asumsi perubahan akan terjadi semakin banyak seiring dengan panjang besarnya pesan yang disisipkan

Hipotesis 2: Semakin panjang *hidden message* yang disipkan dalam *cover image* dengan steganografi akan semakin besar tingkat distorsi visual citra hasil.

Dalam bagian ini suatu citra akan disisipi dengan pesan-pesan yang memiliki panjang berbeda untuk mengetahui pengaruh panjang pesan yang disembunyikan terhadap hasil dari proses steganografi.

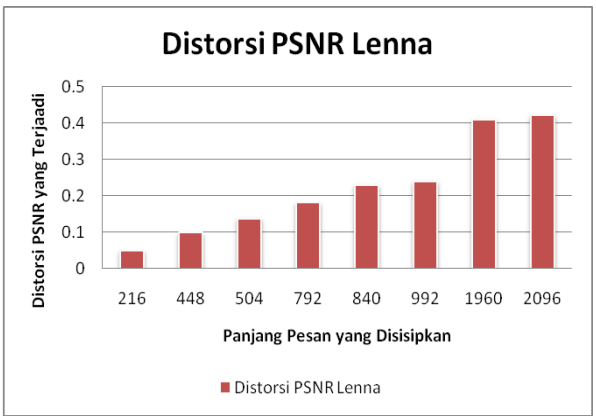
Percobaan 1:

Citra: Lenna.jpg Ukuran citra : 320x320 px

Tabel 4-3 Ukuran Citra dan kualitas citra steganografi dalam PSNR

	panjang pesan (bit)	PSNR Hasil Steganografi	distorsi
1	216	41.09912456	0.050206158
2	448	41.04992407	0.099406644
3	504	41.01026693	0.139063785
4	792	40.96708791	0.182242807
5	840	40.91782421	0.231506509
6	992	40.90841013	0.240920585
7	1960	40.73896254	0.410368171
8	2096	40.72636454	0.422966172

Gambar 4-4 Citra asal dan hasil steganografi dengan GA



Gambar 4-5 Tingkat distorsi kualitas citra hasil steganografi dalam PSNR pada panjang pesan (hidden message) yang berbeda

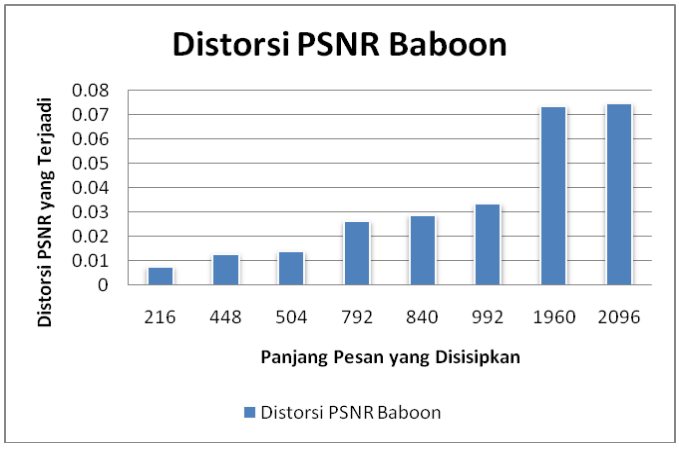
Percobaan 2:

Citra: Baboon.jpg Ukuran citra : 320x320 px

Tabel 4-4 Distorsi citra steganografi pada panjang pesan berbeda

	panjang pesan (bit)	PSNR Hasil Steganografi	distorsi
1	216	36.0212148	0.007685993
2	448	36.01580993	0.013090856
3	504	36.01488314	0.014017647
4	792	36.00237736	0.026523429
5	840	35.99990908	0.028991708
6	992	35.99539518	0.033505609
7	1960	35.95514771	0.073753076
8	2096	35.95416954	0.074731247

Gambar 4-6 Citra asal dan hasil steganografi dengan GA



Gambar 4-7 Tingkat distorsi kualitas citra hasil steganografi dalam PSNR pada panjang pesan tersembunyi (hidden message) yang berbeda

Kualitas citra yang lebih baik bisa dicapai dengan menyisipkan pesan yang lebih kecil. Namun yang dikehendaki dalam steganografi adalah citra yang memiliki kualitas baik dengan jumlah pesan yang disisipkan sebanyak mungkin. Karena itu diperlukan metode lain untuk meningkatkan kualitas citra hasil steganografi.

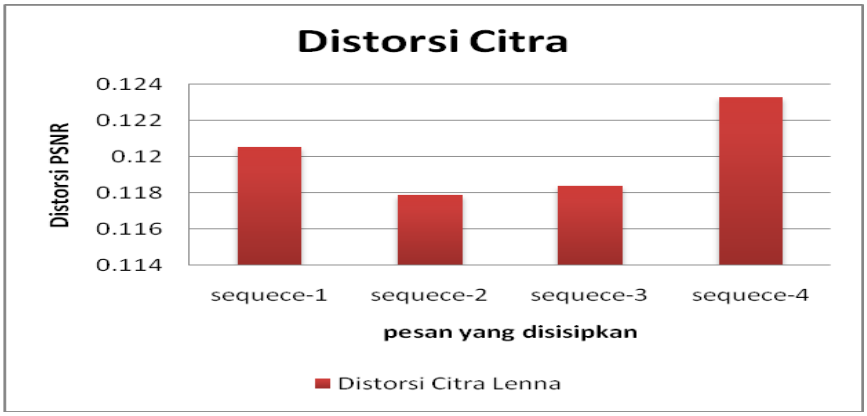
Dengan panjang pesan yang sama namun dengan isi pesan yang berbeda, steganografi dapat menghasilkan citra dengan distori yang berbeda pula. Pada percobaan berikut ini citra uji Lenna dengan ukuran 320x320 piksel akan disisipi dengan empat pesan yang memiliki panjang sama (480 bit) tetapi memiliki isi yang berbeda.

Percobaan 3:

Citra: Lenna.jpg Ukuran citra : 320x320 px

Tabel 4-5 Distorsi PSNR Citra dengan Pesan tersisip sama panjangnya namun beda isinya

citra	pesan	panjang pesan	PSNR Stego	distorsi
lenna 320x320	sequece-1	480 bit	41.02878677	0.120543947
PSNR asal	sequece-2	480 bit	41.03145192	0.1178788
41.14933072	sequece-3	480 bit	41.03093915	0.118391561
	sequece-4	480 bit	41.02605166	0.123279051



Gambar 4-8 Distorsi PSNR Citra dengan Pesan tersisip sama panjangnya namun beda isinya

4.1.3 Metode *crossover* dalam GA

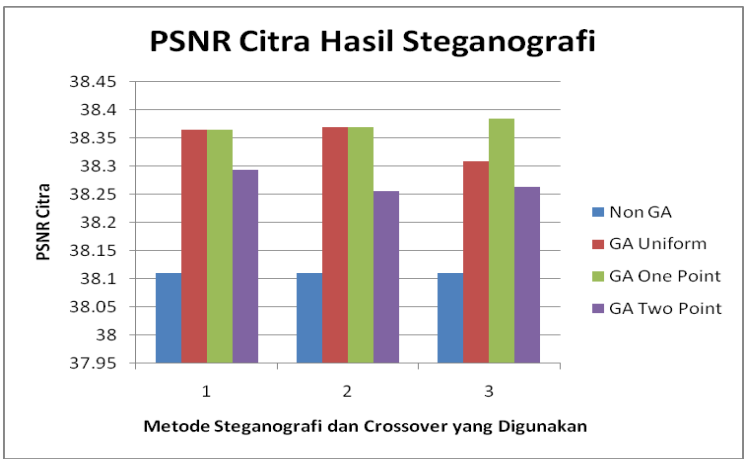
Dalam GA, solusi dicari secara random dengan menggunakan kromosom yang dibuat dari inisial populasi yang dimodifikasi. Modifikasi kromosom ini bisa dilakukan dengan mutasi ataupun dengan persilangan antar kromosom atau yang disebut dengan *crossover*.

Penggunaan GA dengan *crossover* yang berbeda dapat memungkinkan terjadinya solusi yang berbeda, karena kromosom yang digunakan pada perhitungan juga dihasilkan dengan cara yang berbeda. Namun pengaruhnya tidak terlalu besar.

Pada bagian ini akan digunakan tiga macam metode *crossover* yaitu *Uniform Crossover*, *One Point Crossover dan Two Point Crossover* untuk melihat pengaruhnya terhadap solusi yang dihasilkan oleh GA

Tabel 4-6 PSNR Citra dengan metode Crossover berbeda

No Ga	GA Uniform	GA One-Point	GA Two-Point
38.109975829	38.365882868	38.291676075	38.293290151
38.109975829	38.370135165	38.278545146	38.256154376
38.109975829	38.309086886	38.264287365	38.264287365



Gambar 4-9 Kualitas citra dalam PSNR antara steganografi tanpa GA dan dengan GA, dengan metode Crossover yang berbeda

Dari tabel 4.6 dan gambar 4.9 bisa dilihat bahwa solusi yang dihasilkan oleh GA dengan metode *crossover* yang akan berbeda. Namun solusi yang dihasilkan masih layak karena hasil steganografi masih lebih baik dari proses steganografi tanpa GA.

4.1.4 Jumlah generasi pada GA

Jumlah generasi mempengaruhi hasil dari solusi GA. Untuk mendapatkan hasil GA yang baik jumah generasi harus cukup agar nilai *fitness* dari GA bisa mencapai konvergensi. Pada bagian ini akan ditunjukkan hasil *fitness* dari GA pada tiap populasi apakah sudah mencapai konvergensi pada nilai jumlah generasi tertentu.

Nama citra: Lenna.jpg

Dimensi citra : 25600 (160 x 160) piksel

Konstanta DCT tersedia : 15980 DCT (400 blok)

Panjang pesan : 2096 bit

Metode Crossover : *One-Point crossover*

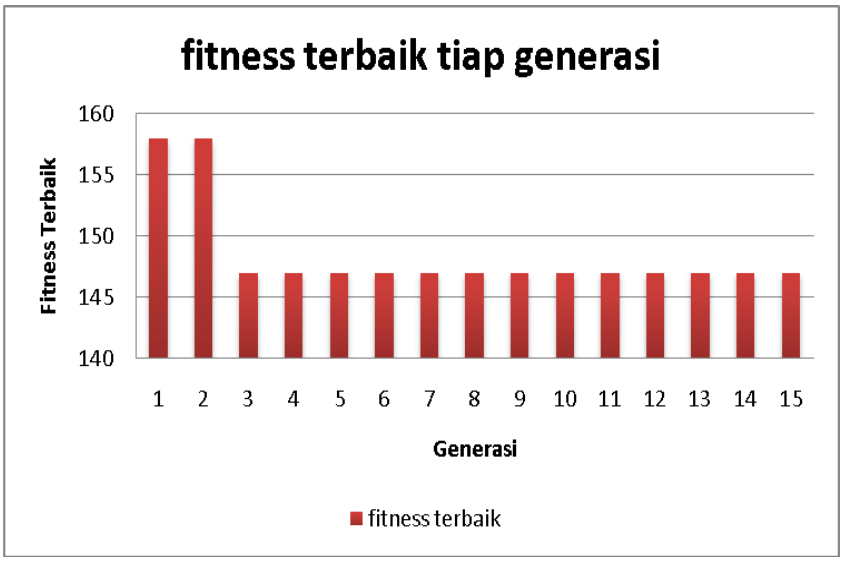
Jumlah generasi maksimal = 15

Jumlah populasi = 10

I. Blok I

Tabel 4-7 Perubahan nilai fitness terhadap generasi pada blok I

Gene-rasi	Populasi										fitness terbaik
	1	2	3	4	5	6	7	8	9	10	
1	158	204	281	269	249	205	195	249	280	246	158
2	158	158	221	195	280	269	227	230	193	267	158
3	147	147	238	147	147	158	147	147	209	158	147
4	147	158	238	147	158	147	147	158	199	199	147
5	147	158	147	147	147	243	147	180	147	147	147
6	147	147	180	147	147	147	199	209	147	147	147
7	147	147	148	147	147	147	147	147	147	147	147
8	147	147	147	261	147	147	147	147	147	147	147
9	147	147	147	147	147	147	147	147	147	147	147
10	147	147	183	147	147	147	147	147	147	147	147

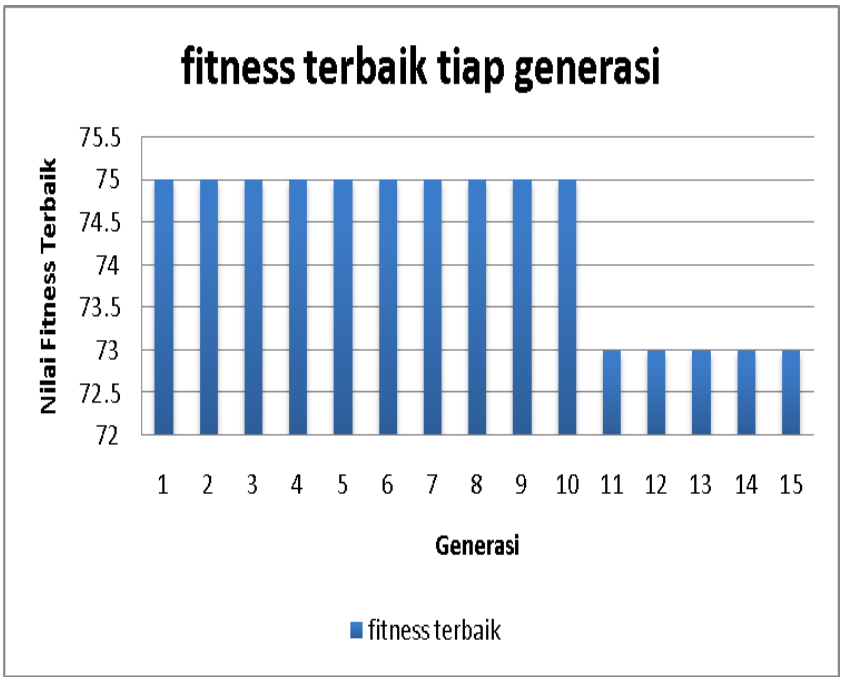


Gambar 4-10 Perubahan nilai fitness terhadap generasi pada blok I

II. Blok II

Tabel 4-8 Perubahan nilai fitness terhadap generasi pada blok II

Generasi	Populasi										fitness terbaik
	1	2	3	4	5	6	7	8	9	10	
1	75	166	157	168	170	169	185	155	158	207	75
2	75	158	221	168	170	269	168	166	75	157	75
3	75	157	185	166	147	158	147	147	209	158	75
4	75	158	238	147	158	147	87	87	75	775	75
5	75	75	75	147	75	75	75	75	75	75	75
6	75	75	75	87	151	147	199	75	75	75	75
7	75	75	75	75	75	75	87	75	75	75	75
8	75	75	75	75	75	75	75	75	75	75	75
9	75	75	75	75	75	75	75	75	75	75	75
10	75	75	75	75	75	75	75	75	75	75	75
11	75	75	75	75	73	75	73	73	75	78	73
12	73	73	73	73	73	73	73	73	73	73	73
13	73	73	73	73	73	73	73	73	73	73	73
14	73	73	73	73	73	73	73	73	73	73	73
15	73	73	73	73	73	73	73	73	73	73	73



Gambar 4-11 Perubahan nilai fitness terhadap generasi pada blok II

Dari percobaan yang dilakukan terlihat nilai *fitness* sudah konvergen pada jumlah generasi maksimal 15. Jumlah generasi maksimal lebih dari ini tidak diperlukan karena nilai *fitness* akan tetap sama (konvergen) dengan nilai generasi yang lebih kecil.

4.2 Uji Coba 2 (Pembandingan Steganografi dengan GA dan tanpa GA)

Dalam bagian ini akan dibandingkan pengaruh penggunaan GA untuk optimasi steganografi dengan hasil dari proses steganografi yang tidak menggunakan GA.

Hasil yang diharapkan dari penggunaan GA pada steganografi ini adalah citra yang dihasilkan memiliki kualitas visual yang lebih baik bila dibandingkan dengan citra hasil steganografi tanpa GA.

Dalam bagian ini akan dibandingkan hasil steganografi dengan menggunakan GA dan tanpa GA pada citra. Percobaan berikutnya dilakukan dengan menggunakan citra dan pesan tersembunyi yang sama namun menggunakan metode *crossover* yang berbeda.

Dalam percobaan akan digunakan tiga citra uji (Lenna, Baboon, dan Peppers) yang terdapat pada gambar 4.2 sebagai *cover image* untuk tempat penyisipan pesan.

Variabel yang akan diubah dan diamati dalam percobaan berikut ini adalah *cover image*, pesan yang disisipkan, dan metode *crossover* dalam GA yang digunakan. Pesan yang disisipkan bervariasi panjangnya (448, 504, 792, 840 dan 992 bit). Dalam tiap kombinasi citra dan pesan akan dilakukan steganografi dengan GA dengan menggunakan metode *crossover One Point, Two Point* dan *Uniform*.

Karena solusi pada GA dihasilkan secara random maka dilakukan lebih dari satu kali percobaan untuk setiap metode *crossover* yang digunakan. Masing-masing percobaan yang menggunakan GA akan diulang sebanyak lima kali.

Nilai rata-rata dari PSNR hasil steganografi dengan GA akan dibandingkan dengan nilai PSNR hasil steganografi tanpa GA untuk mengamati tingkat peningkatan kualitas citra yang dihasilkan dari optimasi dengan GA .

Pada akhir ujicoba, nilai peningkatan PSNR ini akan dievalusi dengan uji hipotesis untuk menentukan apakah peningkatan ini telah secara signifikan membuktikan bahwa penggunaan GA untuk optimasi steganografi telah berhasil menghasilkan citra steganografi dengan kualitas visual yang lebih baik.

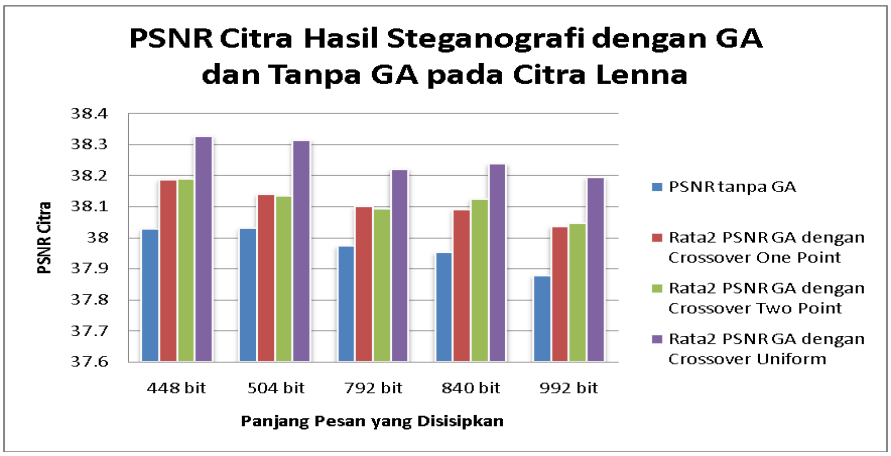
Nama citra: Lenna.jpg

Dimensi citra : 25600 (160 x 160) piksel

Konstanta DCT tersedia : 15980 DCT

Tabel 4-9 Kualitas steganografi tanpa dan dengan GA citra Lenna

Citra	Metode Cross over	panjang pesan	PSNR Stego tanpa GA	PSNR Stego dengan GA	
				rata2 dari lima kali percobaan	rata2 peningkatan PSNR
lenna 160x160	One Point	448 bit	38.0269923	38.18594465	0.1589523
		504 bit	38.0293064	38.13732535	0.108019
		792 bit	37.9714567	38.09853346	0.1270767
		840 bit	37.951459	38.08924678	0.1377878
		992 bit	37.8761661	38.03447609	0.1583099
	Two Point	448 bit	38.0269923	38.18702273	0.1600304
		504 bit	38.0293064	38.13224922	0.1029428
		792 bit	37.9714567	38.09040183	0.1189451
		840 bit	37.951459	38.12219997	0.170741
		992 bit	37.8761661	38.04390454	0.1677384
	Uniform	448 bit	38.0269923	38.32537002	0.2983777
		504 bit	38.0293064	38.3129884	0.283682
		792 bit	37.9714567	38.21971516	0.2482584
		840 bit	37.951459	38.23654145	0.2850825
		992 bit	37.8761661	38.19283031	0.3166642



Gambar 4-12 Kualitas steganografi tanpa dan dengan GA citra Lenna

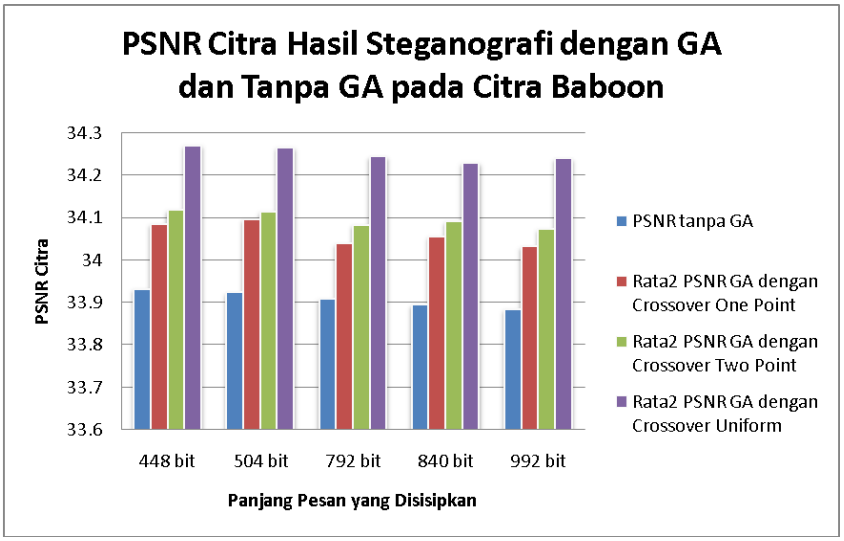
Nama citra: Baboon.jpg

Dimensi citra : 25600 (160 x 160) piksel

Konstanta DCT tersedia : 21451 DCT

Tabel 4-10 Kualitas steganografi tanpa dan dengan GA citra Baboon

Citra	Metode Cross over	panjang pesan	PSNR Stego tanpa GA	PSNR Stego dengan GA	
				rata2 dari lima kali percobaan	rata2 peningkatan PSNR
baboon 160x160	One Point	448 bit	33.9311838	34.08370494	0.1525212
		504 bit	33.9227528	34.09514221	0.1723894
		792 bit	33.9072417	34.03744503	0.1302033
		840 bit	33.8938084	34.05428951	0.1604811
		992 bit	33.8832885	34.03074057	0.1474521
	Two Point	448 bit	33.9311838	34.1164351	0.1852513
		504 bit	33.9227528	34.11265423	0.1899015
		792 bit	33.9072417	34.08177474	0.174533
		840 bit	33.8938084	34.08969988	0.1958915
		992 bit	33.8832885	34.0714134	0.1881249
	Uniform	448 bit	33.9311838	34.26905969	0.3378759
		504 bit	33.9227528	34.26422557	0.3414728
		792 bit	33.9072417	34.24249177	0.33525
		840 bit	33.8938084	34.22800746	0.3341991
		992 bit	33.8832885	34.23805521	0.3547667



Gambar 4-13 Kualitas steganografi tanpa dan dengan GA citra Baboon

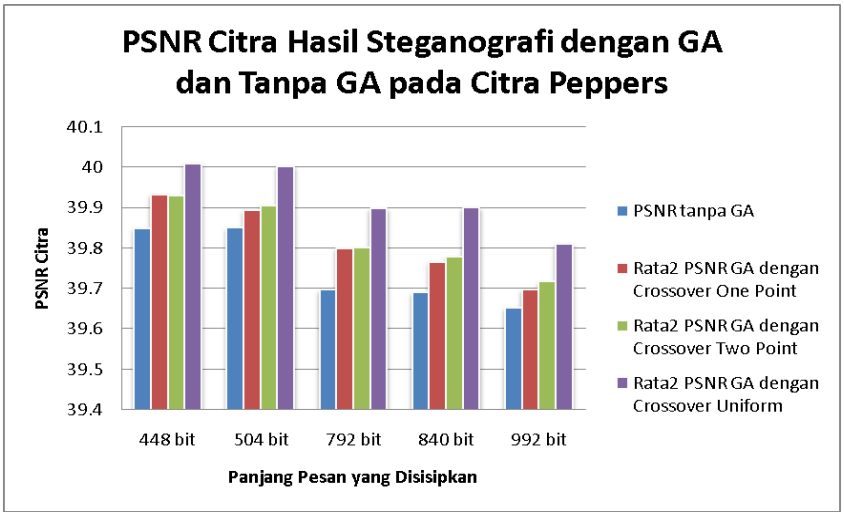
Nama citra: Peppers.jpg

Dimensi citra : 18216 (160 x 160) piksel

Konstanta DCT tersedia : 21451 DCT

Tabel 4-11 Kualitas steganografi tanpa dan dengan GA citra Peppers

Citra	Metode Cross over	panjang pesan	PSNR Stego tanpa GA	PSNR Stego dengan GA	
				rata2 dari lima kali percobaan	rata2 peningkatan PSNR
peppers 160x160	One Point	448 bit	39.8471673	39.93135142	0.0841841
		504 bit	39.84862	39.8925931	0.0439731
		792 bit	39.6955236	39.79874872	0.1032251
		840 bit	39.6889533	39.76411117	0.0751579
		992 bit	39.6512873	39.6963925	0.0451052
	Two Point	448 bit	39.8471673	39.92788978	0.0807224
		504 bit	39.84862	39.904653	0.056033
		792 bit	39.6955236	39.79926084	0.1037372
		840 bit	39.6889533	39.77703007	0.0880768
		992 bit	39.6512873	39.71545336	0.064166
	Uniform	448 bit	39.8471673	40.00836573	0.1611984
		504 bit	39.84862	40.00108833	0.1524683
		792 bit	39.6955236	39.89625282	0.2007292
		840 bit	39.6889533	39.89846752	0.2095142
		992 bit	39.6512873	39.81010397	0.1588166



Gambar 4-14 Kualitas steganografi tanpa dan dengan GA citra Peppers

Dari percobaan yang telah dilakukan terlihat peningkatan kualitas visual dari citra yang bisa dilihat pada gambar 4.4. Gain ini dihitung dari peningkatan PSNR citra hasil steganografi yag tidak menggunakan GA dan citra yang menggunakan GA. Terlihat dalam 18 percobaan nilai PSNR hasil steganografi dengan GA lebih baik dari hasil steganografi tanpa GA.

Tabel 4-12 Peningkatan kualitas citra dalam PSNR

	Peningkatan PSNR
Two Point	0.119220957
	0.101342293
	0.150938632
	0.098568346
	0.075367816
	0.102924495
Uniform	0.305516335
	0.235579932
	0.252511489
	0.263270047
	0.230421352
	0.282080844
One Point	0.160623416
	0.155121559
	0.184132803
	0.156993625
	0.181503927
	0.155978761

Untuk mengetahui apakah penggunaan GA dapat meningkatkan kualaitas citra digunakan uji statistik dengan menggunakan sampel data dari percobaan yang telah dilakukan.

Hipotesis 4: Penggunaan GA dapat meningkatkan kualitas visual citra hasil steganografi.

I. Tentukan statemen H_0 dan H_1.

Ini adalah uji hipotesis dari *sample* pasangan (perlakukan pada sample yang sama sebelum dan sesudah). Maka ynag dipakai bukanlah nilai dari tiap sample tetapi dari selisihnya (D). Yang ditanyakan adalah apakah berbeda sesudah dan sebelum penyimpanan maka hipotesisnya:

$$H0: \mu_D \leq 0$$

$$H1: \mu_D > 0$$

II. Ambil sample (n) dan parameternya.

n = 18,

$$\overline{D} = \frac{1}{n}\sum_{i=1}^{n} Di = \frac{1}{n}\sum_{i=1}^{n}(x_{ai} - x_{bi}) = 3.21209663$$

$$S_D = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(D_i - \overline{D})^2} = 0.078269765$$

III. Tentukan Level Signifikan (α)

$$\alpha = 0{,}05$$

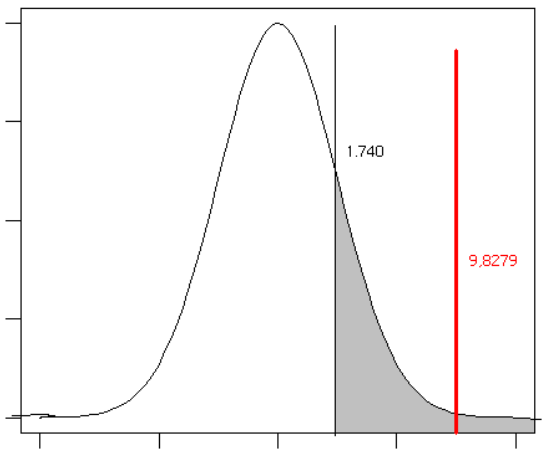
IV. Lakukan test yang sesuai

Ini adalah uji hipotesis dua sampel yang berpasangan. Untuk mengetes hipotesis berpasangan ini digunakan t-test.

$$t = \frac{\overline{D} - \mu_D}{\frac{S_D}{\sqrt{n}}} = \frac{0.178449813 - 0}{\frac{0.078269765}{\sqrt{18}}} = 9.67293614$$

degree of freedom = n - 1 = 17

V. Tentukan *critical section*



Gambar 4-15 Hasil Uji Hipotesis

Pertama ditentukan area yang akan di-*reject*. Karena tanda H_0 adalah lebih kecil sama dengan, maka daerah yang di-reject (H_1) adalah yang lebih besar, yaitu di ujung (*tail*) kanan distribusi.

Cara mencari batas nilai di tabel *t-Distribution* ini dilihat di barisnya nilai *degree of freedom* = 17 dan di kolom nilai dari α = 0,05. Terdapat nilai 1,740. Nilai t hasil perhitungan adalah 9,8729 yang berada di *critical section*, maka harus menolak H_0.

VI. Tentukan yang berada di *critical section* H_0 atau H_1.

Nilai t berada di *critical section* maka yang memenuhi adalah H_1.

VII. Buat kesimpulan apakah menerima atau menolak H_0.

Kesimpulan: Tolak H_0 dan terima H_1

Dari uji hipotesis terbukti penggunaan GA dapat meningkatkan kualitas citra hasil steganografi.

BAB 5

SIMPULAN DAN SARAN

Simpulan yang dapat diambil dari tugas akhir ini adalah sebagai berikut:

1. Proses steganografi menyebabkan terjadi distorsi terhadap citra yang menyebabkan rawan terhadap deteksi secara visual.
2. Penggunaan GA dapat meningkatkan kualitas visual citra yang dihasilkan dari proses steganografi.
3. Kualitas citra steganografi dipengaruhi oleh:
    a. ukuran *cover image* yang digunakan sebagai medium steganografi. Semakin besar cover image maka semakin banyak pesan yang dapat dimasukkan ke dalam citra dan semakin relatif sedikit distori yang terjadi pada proses steganografi
    b. panjang pesan yang disisipkan (*hidden message*). Semakin panjang pesan yang disisipkan maka semakin besar ditorsi visual yang terjadi pada citra hasil steganografi.
4. Pada proses steganografi yang menggunakan Algoritma Genetika kualitas citra juga dipengaruhi parameter GA yaitu metode *crossover* dan jumlah generasi

Dari kesimpulan-kesimpulan diatas, disarankan untuk pengembangan metode ini selanjutnya:

1. Melakukan percobaan steganografi dengan GA dengan nilai parameter GA lain yang diubah untuk mengetahui pengaruh perubahan parameter tersebut terhadap kualitas citra.
2. Mencoba menggunakan metode optimasi selain GA untuk meningkatkan kualitas visual citra hasil steganografi.

DAFTAR PUSTAKA

[1] Fard , A M. Akbarzadeh, M. and Varasteh, F., *"A New Genetic Algorithm Approach for Secure JPEG Steganography" , 2006 IEEE International Conference on Engineering of Intelligent Systems*, 22-23 April 2006 Page(s):1-6

[2] Johnson N. and Jajodia S,, *"Exploring steganography: Seeing the Unseen"*. Computer, 31, no 2:26-34, February 1998

[3] Provos N., Honeyman P., *"Hide and Seek: An Introduction to Steganography"*, IEEE Security & Privacy, May/June 2003

[4] Provos N. and Honeyman P., *"Detecting Steganographic Content on the Internet"*, Center for Information Technology Intergration, University of Michigan. Technical Report 01 November, 2001

[5] Sellars D, *"An Introduction to Steganography"*. cs.uct.ac.za/courses/CS400W/NIS/papers99/ dsellars/stego.html

[6] Westfeld A,, and Pfitzman P,, *"Attacks on Steganographic System"*. Dresden University of Technology, Department of Computer Science, Dresden, Germany

BIODATA PENULIS

Penulis dilahirkan di Bojonegoro, 13 Desember 1985, merupakan anak pertama dari empat bersaudara. Penulis menempuh pendidikan formal di MI Muhammadiyah Sumberrejo Bojonegoro, MTs Negeri Babat Lamongan, dan kemudian di SMA Negeri I Bojonegoro.

Kemudian melanjutkan pendidikan S1-nya di Jurusan Teknik Informatika ITS Surabaya Jawa Timur pada tahun 2003, terdaftar dengan NRP. 5103100026. Dengan pilihan bidang minat *Sistem Bisnis Cerdas*.

Penulis pernah bekerja pada ITS Online sebagai jurnalis dan redaktur pada periode tahun 2004-2008. Penulis juga pernah menjadi asisten mata kuliah Pengolahan Citra Digital dan Teknologi Multimedia pada semester ganjil tahun ajar 2007/2008.

Selama masa kuliahnya, penulis lebih sering menyibukkan dirinya pada beberapa organisasi kampus dengan ketertarikan yang cukup besar pada jurnalisme dan komunikasi publik.