

**STEGANOGRAFI MENGGUNAKAN METODE *DISCRETE*  
*FOURIER TRANSFORM* (DFT)**

**SKRIPSI**

Digunakan Sebagai Syarat Maju Ujian Diploma IV  
Politeknik Negeri Malang

**Oleh:**

**Bias Kristian Cahyaning Paris      Nim. 1341180127**



**PROGRAM STUDI TEKNIK INFORMATIKA  
JURUSAN TEKNOLOGI INFORMASI  
POLITEKNIK NEGERI MALANG**

**2017**



**STEGANOGRAFI MENGGUNAKAN METODE *DISCRETE*  
*FOURIER TRANSFORM* (DFT)**

**SKRIPSI**

Digunakan Sebagai Syarat Maju Ujian Diploma IV  
Politeknik Negeri Malang

**Oleh:**

**Bias Kristian Cayhaning Paris      Nim. 1341180127**



**PROGRAM STUDI TEKNIK INFORMATIKA  
JURUSAN TEKNOLOGI INFORMASI  
POLITEKNIK NEGERI MALANG  
2017**

## HALAMAN PENGESAHAN

### STEGANOGRAFI MENGGUNAKAN *METODE DISCRETE* *FOURIER TRANSFORM (DFT)*

Disusun oleh:

**BIAS KRISTIAN CAHYANING PARIS NIM. 1341180127**

Skripsi ini telah diuji pada tanggal

Disetujui oleh:

- |                  |   |   |
|------------------|---|---|
| 1. Penguji I     | : | <u>DR.ENG. Cahya Rahmad, ST., M.KOM</u><br>NIP. 19720202 200502 1 002 |
| 2. Penguji II    | : | <u>Ely Setyo Astuti, ST., MT</u><br>NIP. 19760515 200912 2 001        |
| 3. Pembimbing I  | : | <u>Yuri Ariyanto S.Kom.,M.Kom</u><br>NIP. 19800716 201012 1 002       |
| 4. Pembimbing II | : | <u>Rizky Ardiansyah S.Kom.,MT</u>                                     |



Mengetahui,

Ketua Jurusan  
Teknologi Informasi



Rudy Ariyanto, S.T., M.Cs.  
NIP. 19711110 199903 1 002

Ketua Program Studi  
Teknik Informatika

Ir. Deddy Kusbianto P., M.MKom.  
NIP. 19621128 198811 1 001

## **PERNYATAAN**

Dengan ini saya menyatakan bahwa Skripsi ini tidak terdapat karya yang pernah diajukan untuk memperoleh gelar kesarjanaan di suatu Perguruan Tinggi, dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

Malang,

Bias Kristian Cahyaning Paris  
NIM. 1341180127

## ABSTRAK

**Paris, Bias Kristian Cahyaning**, “Steganografi Menggunakan Metode *Discrete Fourier Transform* (DFT)” Pembimbing (1) **Yuri Ariyanto S.Kom., M.Kom** (2) **Rizky Ardiansyah S.Kom., MT**

**Skripsi Program Studi Teknik Informatika, Jurusan Teknologi Informasi, Politeknik Negeri Malang, 2017.**

Kemajuan teknologi telah merubah kebiasaan aktifitas manusia. Salah satunya kini manusia dapat saling bertukar dan mengirim citra dengan sangat cepat dan mudah tanpa mengenal jarak dan waktu. Namun dengan seiring berkembangnya teknologi, serangan-serangan juga terjadi pada industri *photography* di mana banyak penyalahgunaan foto yang memiliki hak cipta tanpa seijin pemilik foto tersebut. Karena itulah dibuat sebuah aplikasi yang berfungsi untuk menyisipkan *watermark* dengan menggunakan metode DFT (*Discrete Fourier Transform*). Metode tersebut sebuah metode matematika yang sering digunakan dalam bidang elektronika dan komputer. Metode ini secara khusus digunakan untuk menyelesaikan masalah yang berhubungan dengan frekuensi, sehingga metode ini dapat digunakan dalam bidang *photography*. Metode ini diterapkan untuk melakukan penyisipan dan ekstraksi *watermark* pada citra penampung.

Pada bidang *photography*, metode ini dapat dimanfaatkan untuk mencegah penyalahgunaan hak cipta pada produk *photography*. *Watermark* tersebut disisipkan kedalam frekuensi domain pada gambar dan akan menghasilkan *output* citra ber-*watermark* atau *embedded image*. Hal ini adalah untuk mencegah penyalahgunaan hak cipta, namun *watermark* tersebut tidak nampak secara fisik. Hal ini dilakukan selain memberikan jaminan keamanan terhadap gambar, tapi juga tidak mengurangi estetika pada gambar tersebut.

Analisa yang dilakukan adalah tingkat keberhasilan proses *insertion* dan *extraction*, serangan pada citra, uji kemiripan dengan pengujian NPCR (*Number of Pixel of Change Rate*), UACI (*Unified Averaged Changed Intensity*), dan PSNR (*Peak Signal-to-Noise Ratio*) pada proses *insertion* dan *extraction*. Metode DFT disimpulkan aman terhadap serangan berupa *cropping*, *resize*, dan *editing*. Selain itu, dihasilkan nilai presentase perubahan yang rendah pada pengujian NPCR & UACI dan nilai yang tinggi pada pengujian PSNR.

**Kata kunci:** *Discrete Fourier Transform*, *Watermark*, *embedded image*, *photography*, frekuensi

## **ABSTRACT**

**Paris, Bias Kristian Cahyaning**, “Steganography Using The Discrete Fourier Transform (DFT) Method” **Advisors: (1) Yuri Ariyanto S. Kom., M.Kom (2) Rizky Ardiansyah S.Kom., MT**

*Thesis, Informatics Engineering Study Program, Department of Information Technology, State Polytechnic of Malang, 2017.*

*The advancement of technology has changed the habits of human activities. One of them is nowadays we can exchange and send the images very quickly and easily without having to know the distance and time. However, with the development of technology, along with attacks also occurred in photography industries, there are a lot of misuse of photographs without permission of copyright owner of the photo. Therefore, an application that serves to insert watermarks using DFT (Discrete Fourier Transform) is developed. The method is a mathematical method often used in the field of electronics and computers. Moreover, the method is specifically used to resolve issues related to frequency that can be used in the field of photography. This method is then applied to do insertion and extraction of the watermark on the image container.*

*In the field of photography, this method can prevent abuse of copyright on the photography products. The watermark is inserted into the frequency domain on the picture and it generates the image output with watermark or embedded image. This is to prevent abuse of the copyright, but the watermark does not appear visually. This is done in addition to provide security guarantees against the image, but it also does not reduce the aesthetics on the picture.*

*Analysis conducted is the success rate of insertion and extraction process, the attack on the test image, the picture's resemblance to the NPCR (Number of Pixels of Change Rate), UACI (Unified Average Changed Intensity) and PSNR (Peak Signal-to-Noise Ratio) on the process of insertion and extraction. From the analysis, it can be concluded that the methods of DFT is secure against attacks in the form of cropping, resizing, and editing. In addition, the result value of the percentage changes in testing NPCR & UACI is low and high on PSNR testing.*

**Keywords:** *Discrete Fourier Transform, watermarks, embeded image, photography, frequency*

## KATA PENGANTAR

Puji Syukur penulis panjatkan kepada Tuhan Yesus Kristus atas segala hikmat -Nya penulis dapat menyelesaikan skripsi dengan judul “Steganografi menggunakan metode *discrete fourier transform* (DFT)”. Skripsi ini penulis susun sebagai persyaratan untuk menyelesaikan studi program Diploma IV Program Studi Teknik Informatika, Jurusan Teknologi Informasi, Politeknik Negeri Malang.

Penulis menyadari tanpa adanya dukungan dan kerja sama dari berbagai pihak, penulisan skripsi ini tidak akan dapat berjalan baik. Untuk itu, penulis ingin menyampaikan rasa terima kasih kepada:

1. Bapak Rudi Ariyanto, ST., MCs selaku Ketua Jurusan Teknologi Informasi.
2. Bapak Ir. Deddy Kusbianto P., M.MKom selaku Ketua Program Studi Teknik Informatika.
3. Bapak Yuri Ariyanto S.kom.,M.Kom dan Bapak Rizky Ardiansyah S.Kom.,MT selaku Dosen Pembimbing Politeknik Negeri Malang Prodi Teknik Informatika.
4. Orang tua dan keluarga yang telah memberikan dukungan berupa doa dan dorongan semangat dalam proses penyusunan skripsi ini.
5. Teman-teman satu angkatan Teknik Informatika 2013 yang telah banyak memberikan dukungan dan bantuan untuk terselesaikannya skripsi ini.

Penulis menyadari bahwa dalam penyusunan skripsi ini, masih banyak terdapat kekurangan dan kelemahan yang dimiliki penulis baik itu sistematika penulisan maupun penggunaan bahasa. Untuk itu penulis mengharapkan saran dan kritik dari berbagai pihak yang bersifat membangun demi penyempurnaan laporan ini. Semoga laporan ini berguna bagi pembaca secara umum dan penulis secara khusus. Akhir kata, penulis ucapkan banyak terima kasih.

Malang, 8 Juni 2017



Penulis

## DAFTAR ISI

	Halaman
COVER.....	i
HALAMAN PENGESAHAN .....	ii
PERNYATAAN .....	iii
ABSTRAK.....	iv
<i>ABSTRACT</i> .....	v
KATA PENGANTAR .....	vi
DAFTAR ISI .....	viii
DAFTAR GAMBAR.....	x
DAFTAR TABEL .....	x
DAFTAR LAMPIRAN .....	xii
BAB I. PENDAHULUAN.....	1
1.1 Latar Belakang .....	1
1.2 Rumusan Masalah .....	2
1.3 Batasan Masalah.....	2
1.4 Tujuan.....	2
1.5 Manfaat Penelitian.....	3
1.6 Sistematika Penulisan.....	3
BAB II. LANDASAN TEORI.....	5
2.1 Steganografi .....	5
2.2 Citra Digital.....	6
2.3 <i>Discrete Fourier Transform (DFT)</i> .....	7
2.4 Pengujian <i>Dan</i> Analisa.....	9
BAB III. METODOLOGI PENELITIAN .....	12
3.1 Metode Pengembangan .....	12
3.1.1 Study Literatur .....	12
3.1.2 Analisa Kebutuhan.....	12
3.1.3 Implementasi.....	14
3.2 Subjek dan Objek Penelitian .....	15
3.3 Bahan Penelitian .....	15
3.3.1 Bahan Penelitian .....	15
BAB IV. ANALISIS DAN PERANCANGAN.....	17
4.1 Analisa Sistem.....	17
4.2 Deskripsi Umum Sistem .....	17
4.3 Analisis Kebutuhan .....	17
4.4 Perhitungan Manual Metode <i>Discrete Fourier Transform</i> .....	18
4.5 Desain Antar Muka .....	23
4.5.1 <i>Insertion</i> .....	23
4.5.2 <i>Extraction</i> .....	24
BAB V. IMPLEMENTASI .....	25

5.1 Implementasi Sistem .....	25
5.1.1 Pembagian citra menjadi subblock .....	25
5.1.2 Perhitungan DFT .....	26
5.1.3 Penyisipan <i>Watermark</i> .....	27
5.1.3 Perhitungan IDFT .....	27
5.1.4 Proses Penggabungan Citra .....	28
5.1.5 Proses Membaca <i>Watermark</i> .....	29
5.2 Implementasi Design.....	29
5.2.1 Halaman <i>Insertion</i> .....	29
5.2.2 Halaman <i>Extraction</i> .....	31
BAB VI. PENGUJIAN DAN PEMBAHASAN.....	33
6.1 Pengujian Sistem.....	33
6.2 Analisis.....	35
6.2.1 Analisis Perbandingan Citra .....	35
6.2.2 Analisis Perbandingan Waktu <i>Insertion</i> dan <i>Extraction</i> .....	35
6.2.3 Analisis Hasil Penyisipan .....	37
6.2.4 Analisis Ketahanan Citra Terhadap Serangan .....	38
BAB. VII PENUTUP.....	44
7.1 Kesimpulan.....	44
7.2 Saran.....	44
DAFTAR PUSTAKA .....	45
LAMPIRAN-LAMPIRAN .....	46

## DAFTAR GAMBAR

	Halaman
Gambar 4. 1 Contoh nilai pada citra .....	18
Gambar 4. 2 Nilai DFT pada <i>real</i> (kiri) dan imajiner (kanan) .....	20
Gambar 4. 3 Nilai IDFT .....	20
Gambar 4. 4 Penyisipan.....	21
Gambar 4. 5 Gambar setelah penyisipan .....	22
Gambar 4. 6 Gambar DFT hasil penyisipan .....	23
Gambar 4. 7 Tampilan tab <i>insertion</i> .....	23
Gambar 4. 8 Tab <i>Extraction</i> .....	24
Gambar 5. 1 Halaman <i>insertion</i> .....	30
Gambar 5. 2 Halaman <i>window open image</i> .....	30
Gambar 5. 3 Hasil <i>insertion</i> .....	31
Gambar 5. 4 Halaman <i>extraction</i> .....	31
Gambar 5. 5 Halaman <i>open image</i> .....	32
Gambar 5. 6 Hasil <i>extraction</i> pada gambar ber- <i>watermark</i> .....	32

## DAFTAR TABEL

	Halaman
Tabel 3. 1 <i>Flowchart</i> sistem secara garis besar .....	13
Tabel 3. 2 Contoh foto dan <i>watermark</i> yang akan digunakan.....	14
Tabel 3. 3 Contoh foto yang digunakan.....	15
Tabel 3. 4 Contoh <i>watermark</i> yang digunakan.....	16
Tabel 4. 1 Perhitungan manual DFT.....	18
Tabel 4. 2 Perhitungan IDFT .....	20
Tabel 4. 3 Perhitungan IDFT setelah penyisipan.....	21
Tabel 4. 4 Perhitungan DFT dari gambar penyisipan.....	22
Tabel 6. 1 Pengujian <i>Blackbox</i> .....	33
Tabel 6. 2 Perbandingan UACI dan NPCR .....	35
Tabel 6. 3 Perbandingan Waktu <i>Insertion</i> .....	36
Tabel 6. 4 Perbandingan Waktu Ekstraksi.....	37
Tabel 6. 5 Perbandingan Hasil Ekstraksi .....	38
Tabel 6. 6 Hasil <i>Extraction</i> setelah serangan <i>cropping</i> .....	38
Tabel 6. 7 Hasil <i>Extraction</i> setelah di <i>filtering</i> .....	40
Tabel 6. 8 Hasil <i>Extraction</i> setelah di <i>resize</i> (pembesaran).....	41
Tabel 6. 9 Hasil <i>Extraction</i> setelah di <i>resize</i> (pengecilan) .....	42

## DAFTAR LAMPIRAN

	Halaman
Lampiran 1. Pembagian subblock.....	47
Lampiran 2. Perhitungan DFT .....	47
Lampiran 3. Perhitungan DFT .....	48
Lampiran 4. Perhitungan IDFT .....	48
Lampiran 5. Penggabungan subblock citra.....	49
Lampiran 6. Membaca watermark.....	49

# BAB I. PENDAHULUAN

## 1.1 Latar Belakang

Kemajuan teknologi saat ini telah merubah banyak sekali kehidupan dan kebiasaan aktifitas manusia. Salah satunya kini manusia dapat saling bertukar dan mengirim informasi dengan sangat cepat dan mudah tanpa mengenal jarak dan waktu. Sehingga banyak hal-hal baru yang muncul baik jual beli, bersosial, trend, bisnis, dan banyak lagi. Seiring dengan hal tersebut maka kebutuhan akan keamanan komputer semakin dibutuhkan, untuk melakukan pecegahan terhadap serangan-serangan dari peretas.

Maka muncul kriptografi sebagai salah satu fitur untuk mengamankan pengguna dari berbagai serangan. Namun walaupun telah ada kriptografi, bukan berarti data yang di simpan menjadi 100% aman, karena peretas juga bisa menemukan celah untuk membuka atau memecahkan enkripsi yang telah dibuat untuk mendapatkan informasi.

Namun seiring dengan perkembangan, lahirlah metode steganografi sebagai perkembangan dari kriptografi. Didalam steganografi, kita bisa menyembunyikan sebuah pesan, gambar, atau file kedalam sebuah gambar, video, atau audio. Didalam steganografi, terdapat proses penyisipan bit-bit pesan kedalam bit-bit penampung pada tiap-tiap pixel citra. Sehingga penampung/citra yang telah di sisipi pesan bisa di kirimkan tanpa menimbulkan kecurigaan, Karena dalam metode ini mata dan telinga manusia tidak dapat melihat dan mendengar suatu perubahan yang sangat kecil akibat penyisipan pesan yang dilakukan lakukan melalui steganografi.

Salah satu metode yang dapt digunakan dalam steganografi adalah *Discrete Fourier Transfom* (DFT). DFT adalah metode yang digunakan untuk mengambil gelombang pada citra digital. Gelombang tersebut dapat dimanfaatkan untuk meyisipkan *watermark*, yang sangat populer digunakan di dunia steganografi.[1]

Steganografi juga berkembang menjadi digital watermarking, dan sangat berguna dalam bidang *photography*. Fotografer seringkali mengalami tindakan-tindakan yang tidak diinginkan yang berkaitan pencurian hak cipta, sehingga

timbulah perkembangan dari steganografi yang diterapkan kedalam bidang *photography*. Dalam *study* kasus ini "*Inod Photography*" sebagai salah satu komunitas fotografi yang ingin menerapkan steganografi kedalam komunitas tersebut. harapan komunitas tersebut, bahwa hasil foto-foto komunitas tidak mudah untuk di langgar hak ciptanya oleh pihak lain.

## 1.2 Rumusan Masalah

Berdasarkan latar belakang di atas maka bisa disimpulkan beberapa masalah berikut :

1. Apakah penerapan metode DFT tepat untuk menyelesaikan permasalahan di *inod photography*?
2. Apakah penggunaan metode DFT tahan terhadap serangan cropping, rotasi, dan filtering pada *embedded image*?

## 1.3 Batasan Masalah

Batasan masalah yang diangkat dalam skripsi ini dapat dipaparkan sebagai berikut :

1. Aplikasi yang dibangun menggunakan algoritma DFT untuk melakukan *insertion* dan *extraction*.
2. Format file citra penampung dan *watermark* yang digunakan adalah jpg.
3. Citra penampung yang digunakan adalah citra berwarna (RGB), sedangkan citra *watermark* berwarna *monochrome*.
4. Citra *watermark* yang digunakan adalah citra berukuran 50x50 dan tidak dapat diubah
5. Ukuran citra penampung minimal 100x100 pixel

## 1.4 Tujuan

Tujuan yang ingin dicapai dalam penelitian ini adalah:

1. Menyelesaikan masalah di *inod photograpy*
2. Membuat aplikasi steganografi dengan menggunakan metode DFT



## 1.5 Manfaat Penelitian

Adapun Manfaat dari penulisan dan pembuatan Skripsi ini adalah:

1. Diharapkan bisa menjadi referensi bagi yang ingin mengetahui proses pengolahan informasi untuk disisipkan kedalam citra digital.
2. Dapat menjadi acuan untuk lebih lanjut bagi yang tertarik untuk mengembangkannya.

## 1.6 Sistematika Penulisan

Uraian dalam laporan skripsi penulis menyusun dengan sistematika penulisan sebagai berikut :

### **BAB I PENDAHULUAN**

Pada bab ini menjelaskan tentang latar belakang diadakannya penelitian ini dan yang menjadi dasar permasalahan, yang meliputi latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi, sistematika penulisan, dan penjadwalan kegiatan penelitian.

### **BAB II LANDASAN TEORI**

Bagian ini menjelaskan mengenai sumber dan referensi yang dijadikan acuan dalam pelaksanaan penelitian. Teori-teori tersebut diantaranya mengenai pengenalan dari steganografi, Metode *Discrete Fourier Transform*, serta implementasi program.

### **BAB III METODOLOGI PENELITIAN**

Bagian ini menjelaskan mengenai sumber dan referensi yang dijadikan acuan dalam pelaksanaan penelitian. Teori-teori tersebut diantaranya mengenai pengenalan dari steganografi, Metode *Discrete Fourier Transform*, serta implementasi program.

### **BAB IV ANALISIS DAN PERANCANGAN**

Bab ini menjelaskan tentang perencanaan dan pembuatan sistem secara keseluruhan dan analisa terhadap hasil dari data yang sudah didapat.

### **BAB V IMPLEMENTASI**

Bab ini menjelaskan tentang bagaimana aplikasi dibuat dan berjalan berdasarkan analisa dan perancangan yang dilakukan sebelumnya. Dimana aplikasi diharapkan

dapat melakukan penyisipan *watermark* dengan menggunakan Metode *Discrete Fourier Transform* dengan baik.

## **BAB VI PENGUJIAN DAN PEMBAHASAN**

Bab ini berisikan tentang tampilan yang diusulkan seperti form – form penginputan dan output dalam aplikasi yang mengimplementasikan Steganografi Pada Citra Digital Menggunakan Metode *Discrete Fourier Transform*. Selain itu dilakukan juga pembahasan tentang analisa hasil yang diperoleh dari aplikasi yang dibuat.

## **BAB VII PENUTUP**

Bab ini dibagi menjadi dua sub bab, kesimpulan yang menjawab permasalahan yang dihadapi dan saran yang berisikan solusi alternatif untuk permasalahan yang terjadi pada laporan akhir ini.

## BAB II. LANDASAN TEORI

### 2.1 Steganografi

Steganografi (*steganography*) adalah ilmu dan seni menyembunyikan pesan rahasia (hiding message) sedemikian sehingga keberadaan (*eksistensi*) pesan tidak terdeteksi oleh indera manusia. Kata steganografi berasal dari Bahasa Yunani yang berarti “tulisan tersembunyi” (*covered writing*). Steganografi membutuhkan dua properti: wadah penampung dan data rahasia yang akan disembunyikan.

Steganografi digital menggunakan media digital sebagai wadah penampung, misalnya citra, suara, teks, dan video. Data rahasia yang disembunyikan juga dapat berupa citra, suara, teks, atau video. Steganografi dapat dipandang sebagai kelanjutan kriptografi. Jika pada kriptografi, data yang telah disandikan (*ciphertext*) tetap tersedia, maka dengan steganografi ciphertexts dapat disembunyikan sehingga pihak ketiga tidak mengetahui keberadaannya.

Di negara-negara yang melakukan penyensoran informasi, steganografi sering digunakan untuk menyembunyikan pesan-pesan melalui gambar (*images*), video, atau suara (*audio*). Penyembunyian data rahasia ke dalam citra digital akan mengubah kualitas citra tersebut. Kriteria yang harus diperhatikan dalam penyembunyian data adalah:

a) *Fidelity*

Mutu citra penampung tidak jauh berubah. Setelah penambahan data rahasia, citra hasil steganografi masih terlihat dengan baik. Pengamat tidak mengetahui kalau di dalam citra tersebut terdapat data rahasia.

b) *Robustness*

Data yang disembunyikan harus tahan terhadap manipulasi yang dilakukan pada citra penampung (seperti perubahan kontras, penajaman, pemampatan, rotasi, perbesaran gambar, pemotongan (*cropping*), enkripsi, dan sebagainya). Bila pada citra dilakukan operasi pengolahan citra, maka data yang disembunyikan tidak rusak.

c) *Recovery*

Data yang disembunyikan harus dapat diungkapkan kembali (*recovery*). Karena tujuan steganografi adalah data hiding, maka sewaktu-waktu data rahasia

di dalam citra penampung harus dapat diambil kembali untuk digunakan lebih lanjut [2].

## 2.2 Citra Digital

Secara fisik atau visual, sebuah citra adalah representasi dari informasi yang berkembang di dalamnya sehingga mata manusia dapat menganalisis dan menginterpretasikan informasi tersebut sesuai dengan tujuan yang diharapkan. Kandungan informasi citra dapat dibagi menjadi dua bagian yaitu informasi dasar dan informasi yang bersifat abstrak.

- a. Informasi dasar adalah informasi yang dapat diolah secara langsung tanpa membutuhkan bantuan tambahan pengetahuan khusus. Informasi dasar ini adalah berupa warna (*color*), bentuk (*shape*), dan tekstur (*texture*). Analisis terhadap informasi dasar citra dikenal dengan sebutan *low level image analysis*.
- b. Informasi abstrak adalah informasi yang tidak secara langsung dapat diolah kecuali dengan bantuan tambahan pengetahuan khusus. Contoh informasi yang bersifat abstrak adalah ekspresi wajah di dalam sebuah citra dapat menggambarkan perasaan seseorang.

Kedua informasi ini tidak dapat dianalisis dan dikenali oleh komputer kecuali menggabungkan informasi dasar dengan tambahan pengetahuan khusus.

Secara matematis, sebuah citra dapat didefinisikan sebagai fungsi dua dimensi  $f(x,y)$  di mana  $x$  dan  $y$  adalah koordinat spasial (*plane*) dan  $f$  adalah nilai intensitas warna pada koordinat  $x$  dan  $y$ . Nilai  $x$ ,  $y$ , dan  $f$  semuanya adalah nilai berhingga. Bila nilai-nilai ini bersifat kontinu maka citranya disebut **citra analog**, seperti yang ditampilkan pada layer monitor TV, komputer atau foto cetak. Bila nilai-nilai ini bersifat diskret maka citranya disebut **citra digital**, seperti yang tersimpan dalam memori komputer dan CD-ROM.

Citra Digital umumnya dua dimensi yang dinyatakan dalam bentuk matriks dengan jumlah elemen berhingga. Setiap elemen matriks citra memiliki posisi koordinat  $x$  dan  $y$  tertentu dan juga memiliki nilai. Secara umum, citra digital merupakan representasi piksel-piksel dalam ruang dua dimensi yang dinyatakan dalam matriks berukuran  $N$  baris dan  $M$  kolom seperti pada persamaan di bawah.

$$f(x, y) \approx \begin{bmatrix} f(0, 0) & f(0, 1) & \dots & f(0, M-1) \\ f(1, 0) & f(1, 1) & \dots & f(1, M-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(N-1, 0) & f(N-1, 1) & \dots & f(N-1, M-1) \end{bmatrix} \quad (2.1)$$

Setiap elemen matriks citra disebut **piksel**. Nilai setiap piksel  $f$  pada posisi koordinat  $x$  dan  $y$  merepresentasikan intensitas warna dan dapat dikodekan dalam 24 bit untuk citra berwarna (yang memiliki tiga komponen yaitu **RGB**, R = merah, G = hijau, dan B = biru), 8 bit untuk citra gray level atau 1 bit untuk citra biner [3].

### 2.3 Discrete Fourier Transform (DFT)

DFT merupakan prosedur matematika yang digunakan untuk menentukan harmonik atau frekuensi yang merupakan isi dari urutan sinyal diskrit. Urutan sinyal diskrit adalah urutan nilai yang diperoleh dari sampling periodik sinyal kontinu dalam domain waktu [4]. DFT berasal dari fungsi Transformasi Fourier  $X(f)$  yang didefinisikan:

$$X(f) = \int_{-\infty}^{\infty} x[t] \cdot e^{-f2\pi ft} dt \quad (2.2)$$

Dimana:

$N$  = jumlah sampel input

$X(m)$  = urutan ke- $m$  komponen output DFT( $X(0), X(1), \dots, X(N-1)$ )

$m$  = indeks output DFT dalam domain frekwensi ( $0, 1, \dots, N-1$ )

$x(n)$  = urutan ke- $n$  sampel input ( $x(0), x(1), \dots, x(N-1)$ )

$n$  = indeks sampel input dalam domain waktu ( $0, 1, \dots, N-1$ )

$j$  = bilangan imajiner ( $\sqrt{-1}$ )

$\pi$  = derajat ( $180^\circ$ )

$e$  = basis logaritma natural (2.718281828459)

Dalam bidang pemrosesan sinyal kontinu, Persamaan 1 digunakan untuk mengubah fungsi domain waktu kontinu  $x(t)$  menjadi fungsi domain frekuensi kontinu  $X(f)$ . Fungsi  $X(f)$  memungkinkan untuk menentukan kandungan isi frekuensi dari beberapa sinyal dan menjadikan beragam analisis sinyal dan

pengolahan yang dipakai di bidang teknik dan fisika. Dengan munculnya komputer digital, ilmuwan di bidang pengolahan digital berhasil mendefinisikan DFT sebagai urutan sinyal diskrit domain frekuensi  $X(m)$ ,

$$f(x) = \sum_{n=0}^{N-1} x(n) \cdot e^{-\frac{j2\pi nm}{N}} \quad (2.3)$$

Dimana:

$N$  = jumlah sampel input

$X(m)$  = urutan ke- $m$  komponen output DFT( $X(0), X(1), \dots, X(N-1)$ )

$m$  = indeks output DFT dalam domain frekwensi ( $0, 1, \dots, N-1$ )

$x(n)$  = urutan ke- $n$  sampel input ( $x(0), x(1), \dots, x(N-1)$ )

$n$  = indeks sampel input dalam domain waktu ( $0, 1, \dots, N-1$ )

$j$  = bilangan imajiner ( $\sqrt{-1}$ )

$\pi$  = derajat ( $180^\circ$ )

Kemudian hubungkan dengan rumus Euler  $e^{-j\theta} = \cos(\theta) - j \sin(\theta)$  sehingga setara dengan:

$$X(m) = \sum_{n=0}^{N-1} x(n) \cdot \left[ \cos\left(\frac{2\pi nm}{N}\right) - j \sin\left(\frac{2\pi nm}{N}\right) \right] \quad (2.4)$$

Dimana:

$N$  = jumlah sampel input

$X(m)$  = urutan ke- $m$  komponen output DFT( $X(0), X(1), \dots, X(N-1)$ )

$m$  = indeks output DFT dalam domain frekwensi ( $0, 1, \dots, N-1$ )

$x(n)$  = urutan ke- $n$  sampel input ( $x(0), x(1), \dots, x(N-1)$ )

$n$  = indeks sampel input dalam domain waktu ( $0, 1, \dots, N-1$ )

$j$  = bilangan imajiner ( $\sqrt{-1}$ )

$\pi$  = derajat ( $180^\circ$ )

Meski lebih rumit daripada Persamaan 2, Persamaan 3 lebih mudah untuk dipahami. Konstanta  $j = \sqrt{-1}$  hanya membantu membandingkan hubungan fase di dalam berbagai komponen sinusoidal dari sinyal. Nilai  $N$  merupakan

parameter penting karena menentukan berapa banyak sampel masukan yang diperlukan, hasil domain frekuensi dan jumlah waktu proses yang diperlukan untuk menghitung N-titik DFT. Diperlukan N-perkalian kompleks dan N-1 sebagai tambahan. Kemudian, setiap perkalian membutuhkan N-perkalian riil, sehingga untuk menghitung seluruh nilai N ( $X(0), X(1), \dots, X(N-1)$ ) memerlukan  $N^2$  perkalian. Hal ini menyebabkan perhitungan DFT memakan waktu yang lama jika jumlah sampel yang akan diproses dalam jumlah besar [4].

Transformasi Fourier Diskrit (DFT) 2 Dimensi adalah transformasi Fourier diskrit yang dikenakan pada fungsi 2D (fungsi dengan dua variabel bebas), yang didefinisikan sebagai berikut :

$$F(u, v) = \frac{1}{MN} \sum_{y=0}^{M-1} \sum_{x=0}^{N-1} f(y, x) \left( \cos \left( 2\pi \left( \frac{ux}{N} + \frac{vy}{M} \right) \right) - j \sin \left( 2\pi \left( \frac{ux}{N} + \frac{vy}{M} \right) \right) \right) \quad (2.5)$$

DFT 2D ini banyak digunakan dalam pengolahan citra digital, karena data citra dinyatakan sebagai fungsi 2D.

## 2.4 Pengujian Dan Analisa

Pengujian dilakukan untuk menjamin dan memastikan bahwa sistem yang dirancang dapat berjalan seperti yang diharapkan. Strategi pengujian perangkat lunak yang digunakan yaitu

### a. NPCR dan UACI

NPCR (*Number of Pixel of Change Rate*) dan UACI (*Unified Averaged Changed Intensity*) adalah salah satu perhitungan yang banyak digunakan untuk melakukan evaluasi kekuatan enkripsi citra digital dengan pendekatan perbedaan citra. Pada enkripsi citra, analisa digunakan untuk melakukan kalkulasi antara matriks piksel penyusun citra *input* dan dengan matriks piksel penyusun citra *output*. Perhitungan ini didesain untuk melakukan tes terhadap perubahan piksel dan rata-rata perubahan intensitas antara citra yang dibandingkan.

Perhitungan NPCR berfokus pada nilai absolut yang terjadi di setiap perubahan citra asli dan citra pembandingan. Rumus dari perhitungan NPCR dijabarkan dengan persamaan 2.6:

$$NPCR = \frac{\sum_{i=1}^M \sum_{j=1}^N D(i,j)}{M \times N} \times 100\% \quad (2.6)$$

With :  $D(i,j) = 0$  if  $I_o(i,j) = I_{enc}(i,j)$ , 1

Perhitungan UACI berfokus pada rata-rata perbedaan antara 2 citra yang dibandingkan, yaitu citra asli dan citra pembanding. Rumus dari perhitungan UACI dijabarkan dengan persamaan 2.7:

$$CI = \left[ \sum_{i=1}^M \sum_{j=1}^N \frac{|I_o(i,j) - I_{enc}(i,j)|}{255} \right] \times \frac{100\%}{M \times N} \quad (2.7)$$

Dengan demikian, bisa dilihat hasil perhitungan rata-rata perubahan antara citra asli dengan citra pembanding.

Nilai NPCR memiliki satuan prosentase nilai. Nilai ini mempresentasikan tingkat perbedaan pixel citra asli dan citra pembanding. Pada nilai 0%, maka dipastikan citra asli dan citra pembanding adalah identic, namun jika memiliki nilai, maka mulai ada perbedaan. Semakin besar nilai NPCR, maka semakin besar perbedaan citra asli dan citra pembanding.

Nilai UACI berfokus kepada prosentase perubahan citra asli dan citra pembanding. Penilaian sama dengan UACI, jika nilai adalah 0% maka dipastikan citra asli dan citra pembanding adalah identic. Jika memiliki nilai diatas 0% maka pada citra terdapat perbedaan. Semakin besar nilai, maka semakin besar pula perbedaan antara citra tersebut.

#### b. MSE dan PSNR

MSE (*Mean Square Error*) dan PSNR (*Peak Signal-to-Noise Ratio*) adalah perhitungan untuk menghitung error dari 2 matriks yang dibandingkan. Hal ini bertujuan untuk membandingkan kualitas dari citra yang dibandingkan. MSE merepresentasikan nilai kumulatif dari error antara citra pertama dan citra kedua. Nilai dari MSE diharapkan adalah serendah mungkin. Rumus dari perhitungan MSE dijabarkan dengan persamaan dibawah:

$$MSE = \frac{1}{m \times n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2 \quad (2.8)$$

Setelah nilai MSE diketahui, maka selanjutnya bisa dilakukan perhitungan nilai psnr. Nilai psnr dihitngan dalam ukuran dB(decibel). Rumus dari perhitungan PSNR dijabarkan dengan persamaan dibawah:



$$PSNR = 10 \cdot \log_{10} \left( \frac{MAX_I^2}{MSE} \right) \quad (2.9)$$

$$PSNR = 20 \cdot \log_{10} \left( \frac{MAX_I}{\sqrt{MSE}} \right)$$

$$PSNR = 20 \cdot \log_{10}(MAX_I) - 10 \cdot \log_{10}(MSE)$$

Dengan demikian bisa dilakukan analisa terhadap nilai pembandingan dan rasio perbedaan diantara keduanya. Jika citra pembandingan memiliki nilai yang identic, maka MSE akan memberikan nilai 0 dan PSNR adalah tak terbatas (*infinite* atau *undefinied*). Jika citra pasli dan citra pembandingan tidak sama, maka akan muncul nilai pada MSE. Semakin besar nilai MSE, maka akan semakin banyak perbedaan yang terhitung. Nilai MSE tersebut kemudian dikalkulasi untuk menghitung banyak derau yang ada. Derau tersebut merupakan nilai kerusakan yang ada citra pembandingan. Semakin tinggi nilai PSNR, maka semakin rendah noise pada citra pembandingan. Jika menimbulkan nilai tak terbatas, maka citra asli dan citra pembandingan dinyatakan identic[5].

### c. Pengujian Serangan Terhadap Citra

Perbuatan dari pihak ketiga tidak hanya untuk melihat dan mencuri informasi kada sebuah citra, namun juga bisa untuk membuat penerima citra tidak mengerti apa maksud dari citra yang di dekripsi. Terdapat berbagai serangan yang bisa dilakukan pada citra terenkripsi seperti melakukan proses *filtering*, dan melakukan *cropping* pada citra.

## BAB III. METODOLOGI PENELITIAN

### 3.1. Metode Pengembangan

Bab ini menjelaskan langkah – langkah yang dilakukan untuk membuat Aplikasi yang mengimplementasikan steganografi dengan menggunakan metode *Discrete Fourier Transform (DFT)*. Langkah-langkah yang diperlukan antara lain:

#### 3.1.1 Study Literatur

Pada tahap ini penelitian dilakukan dengan mempelajari berbagai literature melalui pengumpulan dokumen-dokumen, referesi buku, sumber-sumber dari internet, atau sumber-sumber lain yang diperlukan untuk merancang dan mengimplementasikan system yang berkaitan dengan penulisan skripsi yang dilakukan.

#### 3.1.2 Analisa Kebutuhan

Tujuan menganalisa antara lain menganalisa kebutuhan dan keperluan dasar yang akan digunakan dalam pembuatan aplikasi yang diinginkan. Hasil perancangan yang diperoleh adalah pembuatan aplikasi yang dapat melakukan steganografi dengan menggunakan metode *Discrete Fourier Transform (DFT)*.

##### a. Algoritma penyisipan

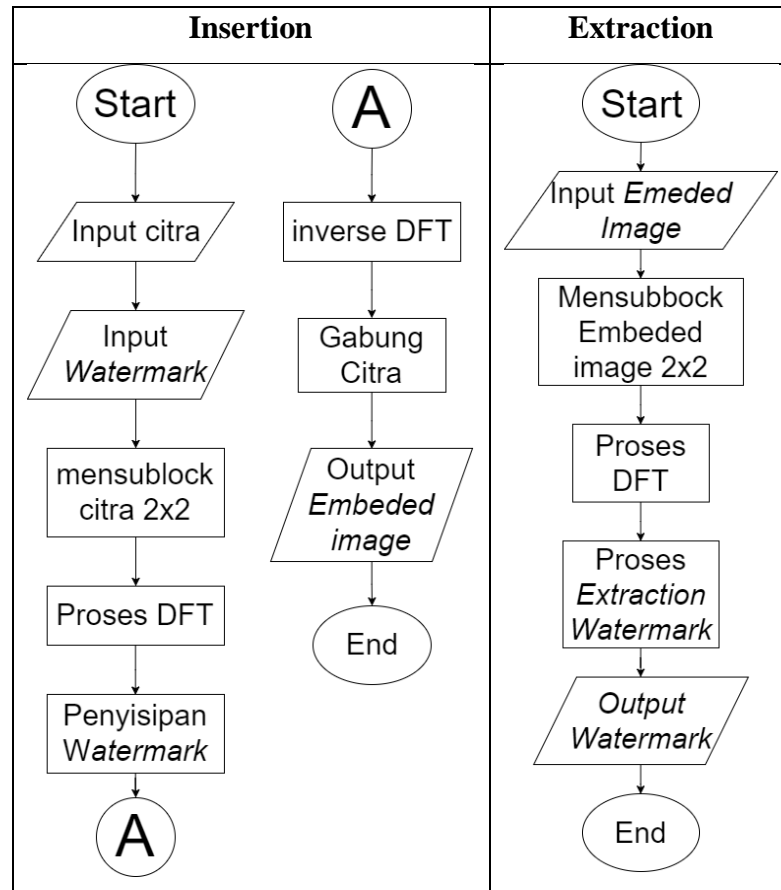
Dalam metode ini semua penyisipan dilakukan pada *frequency domain* dan diterapkan pada *source image*. Pecah *source image* menjadi berukuran 2x2 pixel. Terapkan rumus DFT pada tiap-tiap sublock, kemudian ambil *channel* merah pada koordinat 1x1. Ubah *channel* merah menjadi biner, dan ubah angka pertama dengan angka 1 jika sublock *watermark* berwarna putih, dan masukan angka 0 jika sublock *watermark* berwarna hitam. Kemudian lakukan proses inversi DFT pada masing-masing sublock. Lakukan proses penggabungan sublock menjadi 1 citra utuh kembali [6].

##### b. Algoritma Ekstraksi

Pecah *embedded image* menjadi 2x2 tiap sublocknya. Lakukan proses DFT, dan ambil *channel* merah pada 1 *pixel* citra dengan koordinat 1x1. Ubah *channel* yang telah diambil menjadi bilangan biner. Ambil 1 bilangan biner, jika angka menunjukkan angka 0 maka adalah hitam, jika menunjukkan angka 1 maka

berwarna putih. Kemudian bentuk kembali menjadi 1 watermark utuh berukuran 50x50 [6]. Algoritma *insertion* dan *extraction* dapat dilihat pada tabel dibawah:

Tabel 3. 1 *Flowchart* sistem secara garis besar



Mengacu pada proses insertion diatas, langkah - langkah yang dilakukan adalah sebagai berikut:

- Pemecahan gambar/citra penampung menjadi sublock yang berukuran 2x2 pixel
- Hitung setiap block pixel dengan metode DFT
- Lakukan pemecahan terhadap *watermark* menjadi 1x1 pixel
- Ambil 1 pixel citra pada koordinat 1x1 pada setiap sublock citra penampung pada *channel* merah.
- ubah 1 bit awal pada biner dari channel merah yang telah diambil dengan nilai dari watermark. Jika sublock watermark berwarna putih maka sisipkan 1 pada awal biner sublock citra penampung. jika watermark berwarna hitam, masukan angka 0 pada awal bilangan biner.

- f. Inversikan setiap sublock agar menjadi bentuk semula
- g. Satukan kembali sublock menjadi 1 citra utuh seperti semula.

Mengacu pada proses insertion diatas, langkah - langkah yang dilakukan adalah sebagai berikut:

- a. Pemecahan gambar/citra penampung menjadi sublock yang berukuran 2x2 pixel
- b. Hitung setiap block pixel dengan metode DFT
- c. Ambil 1 pixel pada koordinat 1x1 lalu ambil channel merah pada citra tersebut
- d. Ubah citra merah tersebut menjadi biner, dan ambil 1 angka biner pada awal bilangan. Jika angka 1 berarti putih, dan jika angka 0 berarti warna hitam.
- e. Susun kembali watermark dengan ukuran 50x50

### 3.1.3 Implementasi

Implementasi steganografi dengan menggunakan metode *Discrete Fourier Transform (DFT)* mengacu kepada perancangan sistem. Langkah – langkah pada tahap implementasi dengan :

- a. Pembuatan script berdasarkan perancangan yang telah dibuat.
- b. Menggunakan bahasa pemrograman C#.
- c. Data yang digunakan ada 2 macam, pertama adalah foto yang akan didistribusikan olah “*Inod Photography*”, dan yang kedua adalah watermark inod photography yang akan disisipkan.

Tabel 3. 2 Contoh foto dan *watermark* yang akan digunakan

Contoh foto yang akan di gunakan	Contoh watermark
	

### 3.2. Subjek dan Objek Penelitian

Aplikasi Steganografi ini dirancang dengan menggunakan metode *Discrete Fourier Transform* dengan Bahasa pemrograman C#. Aplikasi ini dikhususkan untuk digunakan pada bidang photography secara khusus digunakan dalam komunitas photography bernama inod photography. Namun juga tidak menutup kemungkinan untuk digunakan oleh komunitas photography lainnya yang membutuhkan.

### 3.3. Bahan Penelitian




Pada penerapan penelitian mengenai judul ini, terdapat beberapa objek penelitian yang dibutuhkan. Ada pun objek yang dibutuhkan adalah beberapa foto dan juga gambar *watermark* dari *inod photography*. Berikut merupakan data-data atau objek penelitian yang digunakan:

#### 3.3.1 Bahan Penelitian

Bahan yang digunakan dalam penelitian adalah beberapa sample foto *full resolution* dari “*inod Photography*”, dan beberapa *watermark* dari “*inod Photography*”. Gambar/foto dan watermark yang digunakan semua dengan format file jpg atau jpeg.


Tabel 3. 3 Contoh foto yang digunakan

Contoh foto yang akan di digunakan	Basic info
	Width : 1184 px Height : 789 px Type file : JPG

Contoh foto yang akan di digunakan	Basic info
	Width : 1184 px Height : 789 px Type file : JPG
	Width : 512 px Height : 512 px Type file : JPG
	Width : 128 px Height : 128 px Type file : JPG

Ketiga gambar diatas adalah contoh foto yang akan disisipi *watermark* yang telah disiapkan. Gambar tersebut adalah gambar dari camera DSLR dan telah di downscale resolusinya. Terdapat dua foto yang dihasilkan, foto horsontal dan vertical.

Tabel 3. 4 Contoh *watermark* yang digunakan

Contoh <i>watermark</i> yang di digunakan	Basic info
	Width : 50 px Height : 50 px Type file : JPG

Watermark yang digunakan adalah watermark berukuran 50x50 pixel berwarna *monochrome* atau hitam dan putih. *Watermark* ini lah yang akan disisipkan kedalam foto yang telah di siapkan.

## BAB IV. ANALISIS DAN PERANCANGAN

### 4.1. Analisa Sistem

Tujuan dari menganalisa antara lain menganalisa kebutuhan dan keperluan dasar yang akan digunakan dalam pembuatan aplikasi yang dibuat. Hasil perancangan yang diperoleh adalah pembuatan aplikasi yang dapat melakukan steganografi.

### 4.2. Deskripsi Umum Sistem

Pada aplikasi ini terdapat dua proses yaitu proses *insertion* dan *extraction* data berupa gambar *watermark*. Citra penampung atau *source image* berupa gambar berwarna (RGB) dengan format .JPG. untuk proses penyisipan digunakan metode *Discrete Fourier Transform* (DFT).

Implementasi Metode *Discrete Fourier Transform* akan digunakan untuk merubah citra digital dari *domain* spasial ke domain frekuensi. Pada metode ini sebelum dilakukan penyisipan *watermark*, citra RGB dibagi menjadi subblok berukuran 2x2. Kemudian dilakukan proses DFT pada setiap subbloknnya, setelah itu *watermark* disisipkan pada koordinat 1x1 pada subblock. Kemudian dilakukan proses inverse DFT agar citra dapat kembali kedalam bentuk citra RGB atau citra asli[7].

### 4.3. Analisis Kebutuhan

Pada tahap ini semua kebutuhan aplikasi didefinisikan sesuai dengan sasaran yang ingin dicapai. Analisis tersebut menyangkut tentang masukan (input) dan keluaran (ouput) dari aplikasi yang akan dibuat.

Adapun data-data yang menjadi masukan bagi aplikasi ini merupakan citra RGB dengan format .JPG berukuran 128x128 *pixel*, 512x512 *pixel*, dan 1200x800 *pixel* sebagai citra penampung. Sedangkan untuk *watermark* yang disisipkan berupa citra *monochrome* berukuran 50x50 *pixel*. Hasil yang diharapkan sebagai *output* pada proses *insertion* merupakan file citra RGB yang

berekstensi .JPG yang telah disisipi watermark dengan menggunakan metode *Discrete Fourier Transform*.

#### 4.4. Perhitungan Manual Metode *Discrete Fourier Transform*

Transformasi Fourier Diskrit (DFT) 2 Dimensi adalah transformasi fourier diskrit yang dikenakan pada fungsi 2D (fungsi dengan dua variabel bebas), yang didefinisikan sebagai berikut :

$$F(u, v) = \frac{1}{MN} \sum_{y=0}^{M-1} \sum_{x=0}^{N-1} f(y, x) \left( \cos \left( 2\pi \left( \frac{ux}{N} + \frac{vy}{M} \right) \right) - j \sin \left( 2\pi \left( \frac{ux}{N} + \frac{vy}{M} \right) \right) \right)$$

DFT 2D ini banyak digunakan dalam pengolahan citra digital, karena data citra dinyatakan sebagai fungsi 2D.

##### Contoh :

Diketahui sebuah gambar dengan nilai pixel sebagai berikut :

<b>34</b>	<b>33</b>
<b>22</b>	<b>12</b>

Gambar 4. 1 Contoh nilai pada citra

DFT dari gambar di atas adalah :

Tabel 4. 1 Perhitungan manual DFT

$  \begin{aligned}  f(0,0) &= 1/2 * 2 (34(\cos(2*3.14(0*0/2+0*0/2))) - j * \sin(2*3.14(0*0/2+0*0/2))) + \\  &\quad 33(\cos(2*3.14(0*0/2+0*1/2))) - j * \sin(2*3.14(0*0/2+0*1/2))) + \\  &\quad 22(\cos(2*3.14(0*1/2+0*0/2))) - j * \sin(2*3.14(0*1/2+0*0/2))) + \\  &\quad 12(\cos(2*3.14(0*1/2+0*1/2))) - j * \sin(2*3.14(0*1/2+0*1/2))) \\  &= 1/4(34+33+22+12) \\  &= 1/4(101) \\  f(0,0) &= 25.25  \end{aligned}  $
$  \begin{aligned}  f(0,1) &= 1/2 * 2 (34(\cos(2*3.14(0*0/2+1*0/2))) - j * \sin(2*3.14(0*0/2+1*0/2))) + \\  &\quad 33(\cos(2*3.14(0*0/2+1*1/2))) - j * \sin(2*3.14(0*0/2+1*1/2))) + \\  &\quad 22(\cos(2*3.14(0*1/2+1*0/2))) - j * \sin(2*3.14(0*1/2+1*0/2))) + \\  &\quad 12(\cos(2*3.14(0*1/2+1*1/2))) - j * \sin(2*3.14(0*1/2+1*1/2))) \\  &= 1/4(34*(1-0) + 33*(-0.99999873173-0.001592652916 j) + 22(1-0) + 12(- \\  &\quad 0.99999873173-0.0031853017 j))  \end{aligned}  $



$$=1/4(34-32.99999581471-0.05255754623j+22-11.9999847808-0.0382236204 j)$$

$$=1/4(11 -0.0908 j)$$

$$=2.75 -0.0227 j$$

$$\begin{aligned} f(1,0) &= 1/2 * 2(34(\cos(2*3.14(1*0/2+0*0/2)) - j*\sin(2*3.14(1*0/2+0*0/2))) + \\ &\quad 33(\cos(2*3.14(1*0/2+0*1/2)) - j*\sin(2*3.14(1*0/2+0*1/2))) + \\ &\quad 22(\cos(2*3.14(1*1/2+0*0/2)) - j*\sin(2*3.14(1*1/2+0*0/2))) + \\ &\quad 12(\cos(2*3.14(1*1/2+0*1/2)) - j*\sin(2*3.14(1*1/2+0*1/2))) \\ &= 1/4(34(1-0)) + (33(1-0)) + (22(-0.99999873173-0.00159652916 j)) + 12(-0.99999873173-0.00159652916 j) \end{aligned}$$

$$=1/4(34+33+(-21.9999720981)+(-0.03512364152j)+(-11.9999847808)+(-0.01915834992 j))$$

$$=1/4(33.0000431211 -0.05428199144 j)$$

$$=7.92 -0.01357 j$$

$$\begin{aligned} f(1,1) &= 1/2 * 2(34(\cos(2*3.14(1*0/2+1*0/2)) - j*\sin(2*3.14(1*0/2+1*0/2))) + \\ &\quad 33(\cos(2*3.14(1*0/2+1*1/2)) - j*\sin(2*3.14(1*0/2+1*1/2))) + \\ &\quad 22(\cos(2*3.14(1*1/2+1*0/2)) - j*\sin(2*3.14(1*1/2+1*0/2))) + \\ &\quad 12(\cos(2*3.14(1*1/2+1*1/2)) - j*\sin(2*3.14(1*1/2+1*1/2))) \\ &= 1/4(34(0.87758256189-0) + 33(0.87758256189-0.479425538604 j) + 22(-0.99999873173-0.001592652916j) + 12(-0.87834500736-0.47802724614 j)) \end{aligned}$$

$$=1/4(29.83780710426+28.96022454237+(-21.9999720981j)+(-10.5401400883-0.03503836415 j)+(-10.5401400883-5.73632695368 j))$$

$$=1/4(37.71775147003 -27.7713374159 j)$$

$$=9.43 -6.9429 j$$

Dari perhitungan diatas menghasilkan nilai DFT yang digambarkan seperti dibawah:

25.25	2.750014
8.250011	-2.25

0	-0.01792 j
-0.01354 j	-0.01234 j

Gambar 4. 2 nilai DFT pada *real* (kiri) dan *imaginer* (kanan)

Lalu harus dilakukan perhitungan IDFT untuk mengembalikan gambar seperti semula dengan perhitungan seperti pada table dibawah:

Tabel 4. 2 Perhitungan IDFT

$f(0,0)=(9.43(\cos(2*3.14(0*0/2+0*0/2)))+(6.9429)*\sin(2*3.14(0*0/2+0*0/2))+$ $7.92(\cos(2*3.14(0*0/2+1*0/2)))+(-0.01357)*\sin(2*3.14(0*0/2+1*0/2))+$ $2.75(\cos(2*3.14(1*0/2+0*0/2)))+(-0.0227)*\sin(2*3.14(1*0/2+0*0/2))+$ $25.25(\cos(2*3.14(1*0/2+1*0/2)))+(0)*\sin(2*3.14(1*0/2+1*0/2)))$ $f(0,0)=34.00003$
$f(0,1)=(9.43(\cos(2*3.14(0*0/2+0*1/2)))+(-6.9429)*\sin(2*3.14(0*0/2+0*1/2))+$ $7.92(\cos(2*3.14(0*0/2+1*1/2)))+(-0.01357)*\sin(2*3.14(0*0/2+1*1/2))+$ $2.75(\cos(2*3.14(1*0/2+0*1/2)))+(-0.0227)*\sin(2*3.14(1*0/2+0*1/2))+$ $25.25(\cos(2*3.14(1*0/2+1*1/2)))+(0)*\sin(2*3.14(1*0/2+1*1/2)))$ $f(0,1)=32.99999$
$f(1,0)=(9.43(\cos(2*3.14(0*1/2+0*0/2)))+(-6.9429)*\sin(2*3.14(0*1/2+0*0/2))+$ $7.92(\cos(2*3.14(0*1/2+1*0/2)))+(-0.01357)*\sin(2*3.14(0*1/2+1*0/2))+$ $2.75(\cos(2*3.14(1*1/2+0*0/2)))+(-0.0227)*\sin(2*3.14(1*1/2+0*0/2))+$ $25.25(\cos(2*3.14(1*1/2+1*0/2)))+(0)*\sin(2*3.14(1*1/2+1*0/2)))$ $f(1,0)=22.00001$
$f(1,1)=(9.43(\cos(2*3.14(0*1/2+0*1/2)))+(-6.9429)*\sin(2*3.14(0*1/2+0*1/2))+$ $7.92(\cos(2*3.14(0*1/2+1*1/2)))+(-0.01357)*\sin(2*3.14(0*1/2+1*1/2))+$ $2.75(\cos(2*3.14(1*1/2+0*1/2)))+(-0.0227)*\sin(2*3.14(1*1/2+0*1/2))+$ $25.25(\cos(2*3.14(1*1/2+1*1/2)))+(0)*\sin(2*3.14(1*1/2+1*1/2)))$ $f(1,1)=12$

Dari perhitungan IDFT diatas menghasilkan nilai dengan nilai seperti tabel dibawah:

34.00003	32.99999
22.00001	12

Gambar 4. 3 Nilai IDFT

Dari hasil perhitungan IDFT meghasilkan gambar yang mirip dengan gambar yang belum melalui proses perhitungan DFT dan IDFT.

Pada table perhitungan DFT sebelumnya akan dilakukan penyisipan watermark pada koorinat 1x1. Penyisipan dilakukan dengan mengganti nilai pada DFT dengan angka 0 atau 255 . Dan pada ontoh dibawah dilakukan penyisipan angka 0, dan menghasilkan gambar seperti diawah:

25.25	2.750014
8.250011	0

Gambar 4. 4 Penyisipan

Dengan nilai gambar di atas akan dlakukan perhitungan IDFT yang ditunjukkan dengan pehitungan pada table dbawah:

Tabel 4. 3 Perhitungan IDFT setelah penyisipan

$f(0,0)=(25.25(\cos(2*3.14(0*0/2+0*0/2)))+(6.9429)*\sin(2*3.14(0*0/2+0*0/2)))+$ $2.75(\cos(2*3.14(0*0/2+1*0/2)))+(-0.01357)*\sin(2*3.14(0*0/2+1*0/2))+$ $8.25(\cos(2*3.14(1*0/2+0*0/2)))+(-0.0227)*\sin(2*3.14(1*0/2+0*0/2))+$ $0(\cos(2*3.14(1*0/2+1*0/2)))+(0)*\sin(2*3.14(1*0/2+1*0/2)))$ $f(0,0)=36.25003$
$f(0,1)=(25.25(\cos(2*3.14(0*0/2+0*1/2)))+(6.9429)*\sin(2*3.14(0*0/2+0*1/2)))+$ $2.75(\cos(2*3.14(0*0/2+1*1/2)))+(-0.01357)*\sin(2*3.14(0*0/2+1*1/2))+$ $8.25(\cos(2*3.14(1*0/2+0*1/2)))+(-0.0227)*\sin(2*3.14(1*0/2+0*1/2))+$ $0(\cos(2*3.14(1*0/2+1*1/2)))+(0)*\sin(2*3.14(1*0/2+1*1/2)))$ $f(0,1)=30.75$
$f(1,0)=(25.25(\cos(2*3.14(0*1/2+0*0/2)))+(6.9429)*\sin(2*3.14(0*1/2+0*0/2)))+$ $2.75(\cos(2*3.14(0*1/2+1*0/2)))+(-0.01357)*\sin(2*3.14(0*1/2+1*0/2))+$ $8.25(\cos(2*3.14(1*1/2+0*0/2)))+(-0.0227)*\sin(2*3.14(1*1/2+0*0/2))+$ $0(\cos(2*3.14(1*1/2+1*0/2)))+(0)*\sin(2*3.14(1*1/2+1*0/2)))$ $f(1,0)=19.75001$
$f(1,1)=(25.25(\cos(2*3.14(0*1/2+0*1/2)))+(6.9429)*\sin(2*3.14(0*1/2+0*1/2)))+$ $2.75(\cos(2*3.14(0*1/2+1*1/2)))+(-0.01357)*\sin(2*3.14(0*1/2+1*1/2))+$ $8.25(\cos(2*3.14(1*1/2+0*1/2)))+(-0.0227)*\sin(2*3.14(1*1/2+0*1/2))+$ $0(\cos(2*3.14(1*1/2+1*1/2)))+(0)*\sin(2*3.14(1*1/2+1*1/2)))$

$f(1,1)=14.24999$
-------------------

Dengan perhitungan diatas menghasilkan gambar dengan nilai seperti dibawah:

36.25003	30.75
19.75001	14.24999

Gambar 4. 5 Gambar setelah penyisipan

Pada gambar diatas adalah gambar dari hasil perhitungan penyisipan IDFT. Dari data gambar diatas dapat dilakukan ekstraksi watermark pada koordnat 1x1. Untuk melakukan ekstraksi, lakukan proses perhitungang DFT dari data diatas. Dan perhitungan ditunjukan pada perhitungan pada table dibawah:

Tabel 4. 4 Perhitungan DFT dari gambar penyisipan

$f(0,0)=1/2*2(36.25(\cos(2*3.14(0*0/2+0*0/2))j*\sin(2*3.14(0*0/2+0*0/2))+$ $30.75(\cos(2*3.14(0*0/2+0*1/2))-j*\sin(2*3.14(0*0/2+0*1/2))+$ $19.75(\cos(2*3.14(0*1/2+0*0/2))-j*\sin(2*3.14(0*1/2+0*0/2))+$ $14.24(\cos(2*3.14(0*1/2+0*1/2))-j*\sin(2*3.14(0*1/2+0*1/2)))$ $f(0,0)=25.25000718$
$f(0,1)=1/2*2(36.25(\cos(2*3.14(0*0/2+1*0/2))-j*\sin(2*3.14(0*0/2+1*0/2))+$ $30.75(\cos(2*3.14(0*0/2+1*1/2))-j*\sin(2*3.14(0*0/2+1*1/2))+$ $19.75(\cos(2*3.14(0*1/2+1*0/2))-j*\sin(2*3.14(0*1/2+1*0/2))+$ $14.24(\cos(2*3.14(0*1/2+1*1/2))-j*\sin(2*3.14(0*1/2+1*1/2)))$ $f(0,1)=2.750026729 -0.01792 j$
$f(1,0)=1/2*2(36.25(\cos(2*3.14(1*0/2+0*0/2))-j*\sin(2*3.14(1*0/2+0*0/2))+$ $30.75(\cos(2*3.14(1*0/2+0*1/2))-j*\sin(2*3.14(1*0/2+0*1/2))+$ $19.75(\cos(2*3.14(1*1/2+0*0/2))-j*\sin(2*3.14(1*1/2+0*0/2))+$ $14.24(\cos(2*3.14(1*1/2+0*1/2))-j*\sin(2*3.14(1*1/2+0*1/2)))$ $f(1,0)=8.250016186 -0.01354 j$
$f(1,1)=1/2*2(36.25(\cos(2*3.14(1*0/2+1*0/2))-j*\sin(2*3.14(1*0/2+1*0/2))+$ $30.75(\cos(2*3.14(1*0/2+1*1/2))-j*\sin(2*3.14(1*0/2+1*1/2))+$ $19.75(\cos(2*3.14(1*1/2+1*0/2))-j*\sin(2*3.14(1*1/2+1*0/2))+$ $14.24(\cos(2*3.14(1*1/2+1*1/2))-j*\sin(2*3.14(1*1/2+1*1/2)))$ $f(1,1)=-2.06091 -0.01234 j$

Dari perhitungan diatas menghasilkan gambar DFT, dan untuk diambil nilainya pada koordinat 1x1.

25.25000718	2.750026729
8.250016186	2.06091

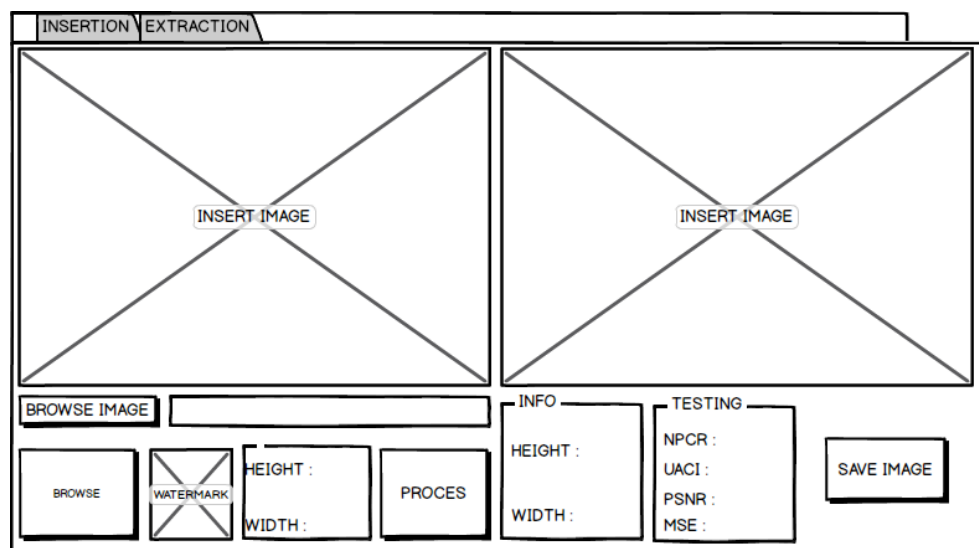
Gambar 4. 6 Gambar DFT hasil penyisipan

Dari hasil perhitungan DFT setelah proses penyisipan. Mengasilkan table seperti diatas. Pada koordinat 1x1 menghasilkan nilai 2.06091 sehingga program secara otomatis akan merubah angka tersebut menjadi 0 atau hitam. Proses ini dilakukan terus menerus hingga subblok terakhir. Dari hasil perhitungan subblock pertama hingga subblock ke 2500 sajalah yang akan diambil nilainya dari kordinat 1x1. Sehingga akan membentuk suatu watermark berukuran 50x50 *pixel*.

#### 4.5.Desain Antar Muka

Design aplikasi dibuat dengan 2 tab utama, tab *insertion* dan *extraction*. Tab *insertion* digunakan untuk menyisipkan *watermark* kedalam foto. Dan tab *extraction* digunakan untuk mengekstrak watermark yang telah disisipkan sebelumnya.

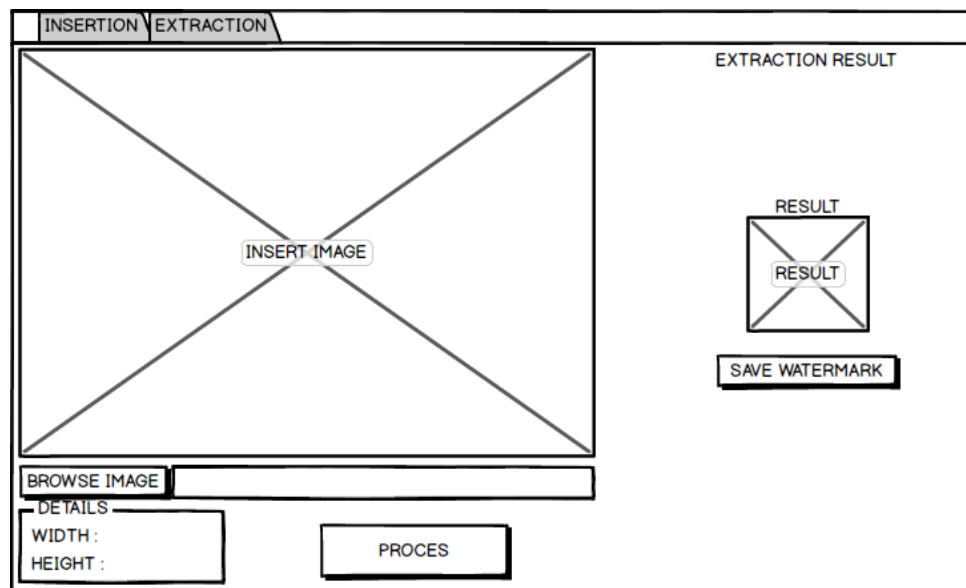
##### 4.5.1.Insertion



Gambar 4. 7 Tampilan tab *insertion*

Pada tab *insertion* sisi kiri terdapat *picture box* yang menampilkan gambar atau foto yang akan disisipkan watermark. *Picture box* watermark yang digunakan untuk menampilkan watermark yang akan disisipkan kedalam foto. Pada *details* akan menampilkan detail berupa lebar, panjang, dan ukuran foto. Tombol proses digunakan untuk mengeksekusi perintah penyisipan, dan setelah gambar disisipkan, gambar akan ditampilkan di *picture box embeded image*, dan kemudian gambar/foto dapat di save.

#### 4.5.2. Extraction



Gambar 4. 8 Tab *Extraction*

Tab *extraction* digunakan untuk mengekstraksi *watermark* dari gambar yang telah disisipkan *watermark* atau *embeded image*. tombol *browse image* digunakan untuk mencari *embeded image* pada *explorer* kemudian ditampilkan pada *picture box*. Kemudian tekan tombol *process* untuk mengeksekusi ekstraksi watermark yang terdapat pada *embeded image*.

## BAB V. IMPLEMENTASI

Pada bab ini akan membahas implementasi aplikasi yang telah dibuat dengan menggunakan bahasa pemrograman C# menggunakan Visual Vb.Net. Di bawah ini merupakan langkah-langkah penggunaan aplikasi steganografi yang telah dibuat.

### 5.1. Implementasi Sistem

Bab implementasi adalah melakukan penulisan kode sesuai dengan apa yang direncanakan. Aplikasi memiliki 6 tahap pada proses penyisipan atau *insertion* yaitu pembagian citra menjadi *subblock*, perhitungan DFT, penyisipan *watermark*, perhitungan IDFT, dan proses penggabungan subblock citra menjadi 1 citra yang utuh kembali, sedangkan pada tahap *extraction* atau ekstraksi membutuhkan 3 proses, yang pertama adalah proses pembagian subblock, melakukan proses perhitungan DFT pada setiap subblock citra, 2 proses ini adalah proses yang sama yang dilakukan pada proses *insertion*, dan terakhir, membaca watermark pada red channel koordinat sub block 1x1. Untuk tahapan dari implementasi penulisan kode adalah sebagai berikut:

#### 5.1.1. Pembagian citra menjadi subblock

Proses pertama yang dilakukan aplikasi adalah membagi citra penampung menjadi subblock berukuran 2x2 *pixel*. yang kemudian subblock tersebut kan di gunakan untuk menampung pesan pada koordinat 1x1 di tiap-tiap subblock. Proses ini juga merupakan proses yang sama yang digunakn pada proses *extraction* .Berikut merupakan potongan kode dari pembagian citra menjadi subblock:

```

public List<Bitmap> subBlokCitraAwal(Image hostImage, int
blockSize)
{
    List<Bitmap> res = new List<Bitmap>();
    int jumlahBaris = hostImage.Height / blockSize;
    int jumlahKolom = hostImage.Width / blockSize;
    for (int i = 0; i <= jumlahBaris - 1; i++)
    {
        for (int j = 0; j <= jumlahKolom - 1; j++)
        {
            Bitmap subBlokCitra = new Bitmap(blockSize,
blockSize);
            Graphics subBlock =
Graphics.FromImage(subBlokCitra);
            subBlock.DrawImage(hostImage, 0, 0, new Rectangle(j
* blockSize, i * blockSize, blockSize, blockSize),
GraphicsUnit.Pixel);
            res.Add(subBlokCitra);
        }
    }
    return res;
}

```

### 5.1.2. Perhitungan DFT

Setelah citra penampung terbagi-bagi menjadi subblock yang berukuran 2x2. Maka bisa dilakuka perhitungan DFT pada masing-masing subblok citra tersebut. Proses ini dilakukan untuk membuka dimensi gelombang pada setiap subblock citra penampung tersebut. Proses in juga digunakan pada proses *extraction*. Berikut merupakan kode dari proses perhitungan DFT:

```

public void hitungDFT()
{
    if (jumlahData > 0)
    {
        for (int k = 0; k <= jumlahData - 1; k++)
        {
            komponenFourier[k].real = 0;
            komponenFourier[k].imajiner = 0;
            for (int l = 0; l <= (jumlahData - 1) - 1; l++)
            {
                komponenFourier[k] =
hitungPenjumlahanKompleks(komponenFourier[k],
hitungPerkalianKompleks(nilaiKompleks[(k * l) % jumlahData],
data[l]));
            }
            komponenFourier[k].real = komponenFourier[k].real /
jumlahData * 2;
            komponenFourier[k].imajiner = -komponenFourier[k].imajiner
/ jumlahData * 2;
        }
        komponenFourier[0].real = komponenFourier[0].real / 2;
        komponenFourier[0].imajiner = komponenFourier[0].imajiner / 2;
    }
}

```



### 5.1.3 Penyisipan *Watermark*

Setelah setiap sublock citra dihitung menggunakan metode *Discrete Fourier Transform*, langkah selanjutnya adalah menyisipkan *watermark* pada setiap sublock citra penampung. *Watermark* akan disisipkan langsung pada *channel* merah citra penampung pada koordinat  $f(1,1)$ . Terdapat beberapa proses pada tahapan ini, pertama merubah bilangan decimal menjadi biner, membaca warna hitam & putih pada *watermark*, dan proses penyisipan *watermark* kedalam *channel* merah. Dan hanya mengubah angka pertama pada bilangan biner citra penampung. Berikut adalah potongan kode dari proses penyisipan *watermark*:

```
if (count <= watermark.Length - 1)
{
    int w = (listsublock.ElementAt(s).Width) - 1;
    int h = (listsublock.ElementAt(s).Height) - 1;
    Bitmap img = new Bitmap(pbImgWatermark.Image);
    if (watermark[count] == 255) //putih
    {
        Bitmap img2 = new Bitmap(listsublock.ElementAt(s));
        int r = img2.GetPixel(w, h).R;
        int g = img2.GetPixel(w, h).G;
        int b = img2.GetPixel(w, h).B;
        int newValueR = 255;
        listsublock.ElementAt(s).SetPixel(w, h,
        Color.FromArgb(newValueR, g, b));
    }
    else
    {
        Bitmap img2 = new Bitmap(listsublock.ElementAt(s));
        int r = img2.GetPixel(w, h).R;
        int g = img2.GetPixel(w, h).G;
        int b = img2.GetPixel(w, h).B;
        int newValueR = 50;
        listsublock.ElementAt(s).SetPixel(w, h,
        Color.FromArgb(newValueR, g, b));
    }
}
```

### 5.1.3. Perhitungan IDFT

Tahap setelah proses *insertion* adalah tahap perhitungan IDFT, tahap ini dilakukan untuk melakukan pengabalian citra menjadi bentuk aslinya dari bentuk gelombang. Proses ini dilakukan pada setiap sublock citra yang telah di proses sebelumnya. Berikut adalah potongan kode dari proses perhitungan IDFT:

```
public void hitungIDFT(int jumlahKomponen)
{
    for (int k = 0; k <= jumlahData - 1; k++)
    {
```

```

        output[k] = 0;
        for (int l = 0; l <= (jumlahKomponen - 1); l++)
        {
            output[k] = output[k] + komponenFourier[l].real *
Math.Cos(2.0 * Math.PI * Convert.ToDouble(l * k) /
Convert.ToDouble(jumlahData)) + komponenFourier[l].imajiner *
Math.Sin(2.0 * Math.PI * Convert.ToDouble(l * k) /
Convert.ToDouble(jumlahData));
        }
    }
}

```

#### 5.1.4. Proses Penggabungan Citra

Pada proses ini adalah proses terakhir dari proses *insertion*, proses ini secara garis besar melakukan proses penggabungan sublock cira penampung yang berukuran 2x2 menjadi 1 citra utuh kembali. Berikut merupakan potongan program dari proses penggabungan citra:

```

public Bitmap gabungCitra(List<Bitmap> citraIDFT, Bitmap
citraAsal)
{
    int lebar = citraAsal.Width;
    int panjang = citraAsal.Height;
    int jumlahBaris = panjang / 2;
    int jumlahKolom = lebar / 2;
    Bitmap subBlok = null;
    int indexBlok = 0;
    int x = 0;
    int y = 0;
    int tinggi = 0;
    Bitmap citraAkhir = new Bitmap(lebar, panjang);
    Graphics grp = Graphics.FromImage(citraAkhir);

    for (int i = 0; i <= jumlahBaris - 1; i++)
    {
        x = 0;
        for (int j = 0; j <= jumlahKolom - 1; j++)
        {
            subBlok = citraIDFT[indexBlok];
            indexBlok += 1;
            grp.DrawImage(subBlok, new Rectangle(x, y, subBlok.Width,
subBlok.Height));
            x += subBlok.Width;
            tinggi = subBlok.Height;
        }
        y += tinggi;
    }
    return citraAkhir;
}

```

#### 5.1.5. Proses Membaca *Watermark*

Proses ini dilakukan paling akhir, setelah proses pembagian subblock dan proses perhitungan DFT dilakukan. Proses pembagian subblock dan perhitungan DFT yang digunakan adalah proses perhitungan yang sama pada proses *insertion*. Berikut adalah potongan program dari proses membaca *watermark*:

```
if (s <= imgW.Length - 1)
{
    Bitmap img = new Bitmap(listsubblock.ElementAt(s));
    int red = img.GetPixel(img.Width - 1, img.Height - 1).R;
    if (red >= 1)
    {
        imgW[s] = 255;
    }
    else
    {
        imgW[s] = 0;
    }
}
```

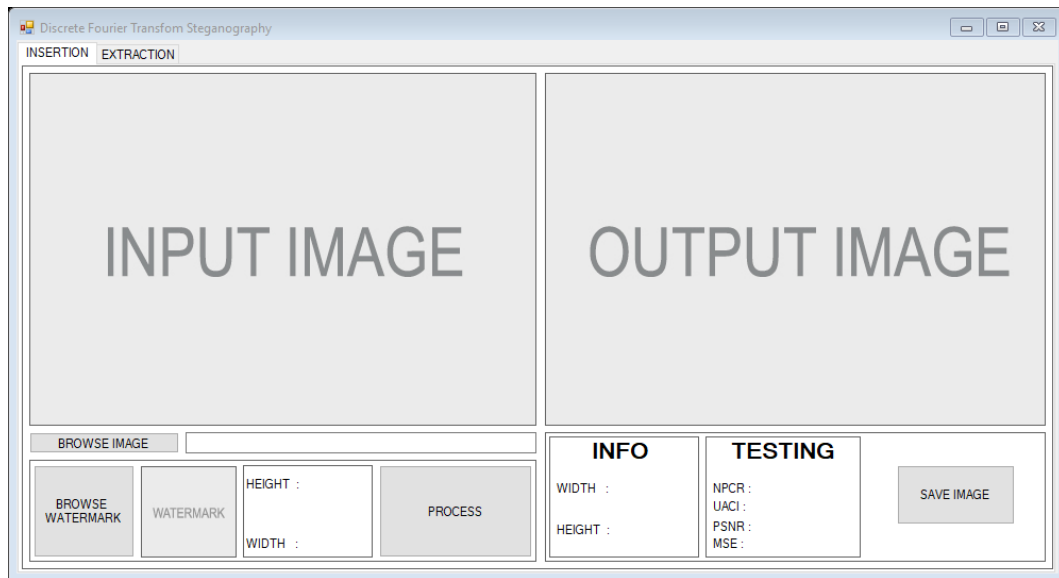
## 5.2. Implementasi Design

pada sub bab ini akan membahas implementasi dari design system yang telah dibuat. Aplikasi tersebut dibuat dengan 2 halaman utama yang berupa tab. Tab pertama adalah halaman/tab *insertion* yang menangani proses *insertion watermark* yang dilakukan. Kedua adalah halaman/tab *extraction* yang digunakan dalam proses *extraction* atau ekstraksi *watermark* pada *embedded image*. Berikut adalah implementasi design dari system yang telah dibuat:

### 5.2.1. Halaman *Insertion*

Halaman *insertion* merupakan halaman yang digunakan sebagai halaman penyisipan watermark ke dalam citra penampung. Pada halaman ini pengguna dapat memilih citra penampung yang diinginkan dan melakukan proses *insertion watermark* kedalam citra penampung. Pada halaman ini, pengguna dapat juga menyimpan file atau gambar hasil dari *insertion* kedalam komputer. Dan dapat mengekstrak *watermark* yang telah disisipkan pada citra penampung .

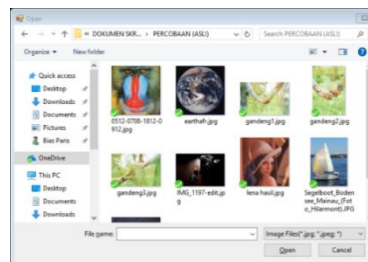
Berikut gambar dibawah ini menunjukkan halaman *insertion*.



Gambar 5. 1 Halaman *insertion*

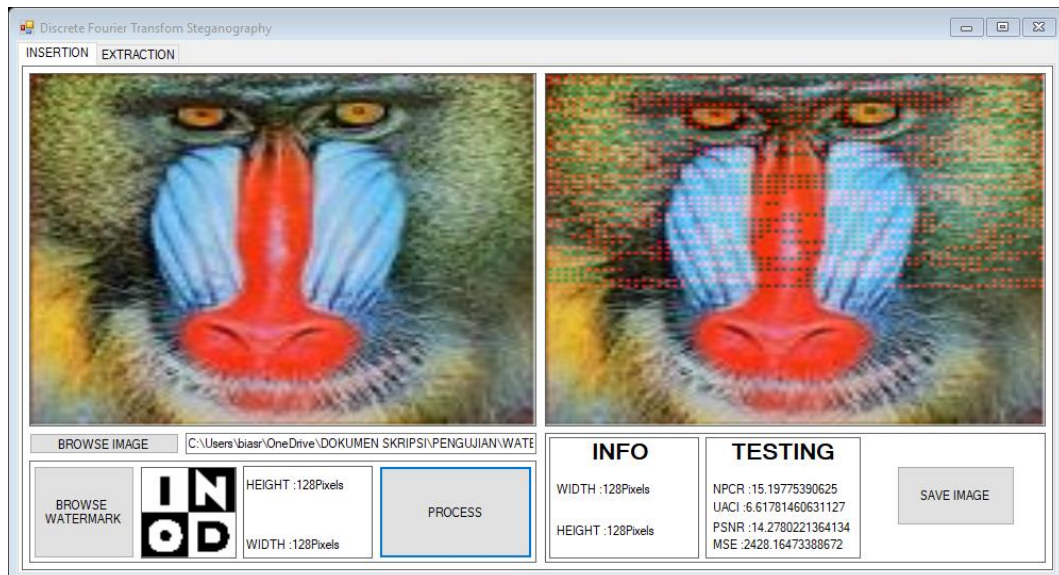
Tahapan untuk melakukan proses *insertion* adalah sebagai berikut:

1. Klik tobol “BROWSE IMAGE” untuk membuka dan memilih citra yang akan disisipi oleh *watermark*.



Gambar 5. 2 Halaman window open image

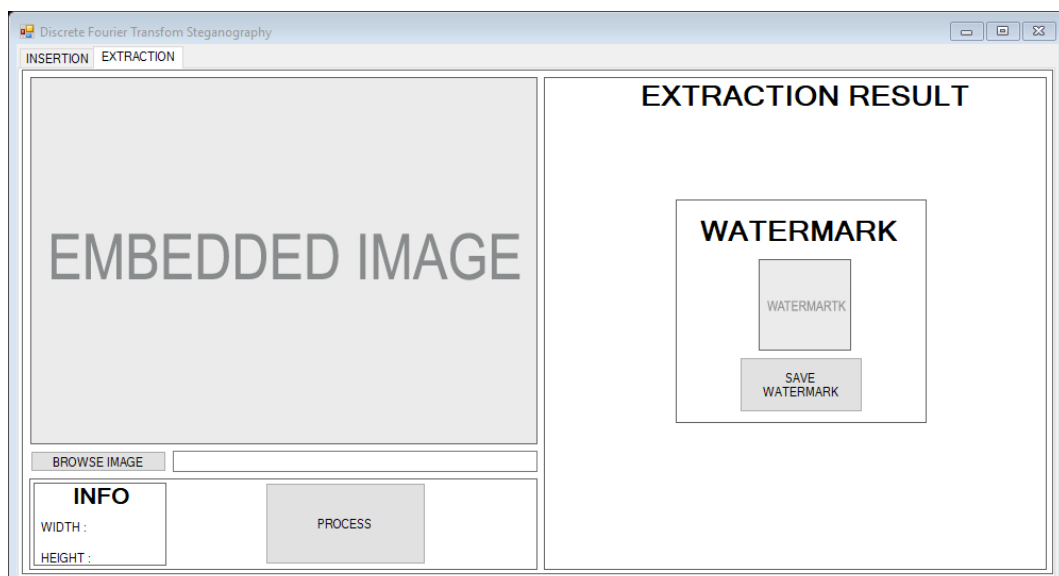
2. Klik tombol “PROCESS” untuk melakukan proses *insertion watermark*, kedalam citra penampung.

Gambar 5. 3 Hasil *insertion*

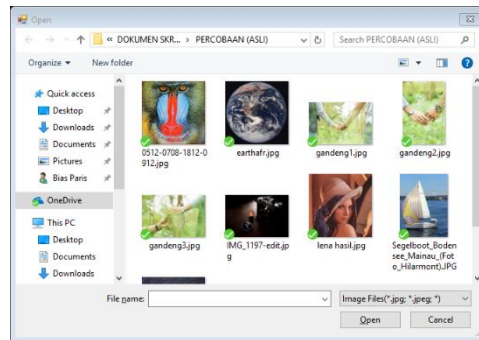
3. Kemudian klik “SAVE IMAGE” untuk menyimpan gambar yang telah di proses.

#### 5.2.2. Halaman *Extraction*

Halaman *extraction* adalah halaman yang digunakan untuk mengekstraksi *watermark* yang telah disisipkan dalam citra penampung. Berikut adalah implementasi design pada halaman *extraction*.

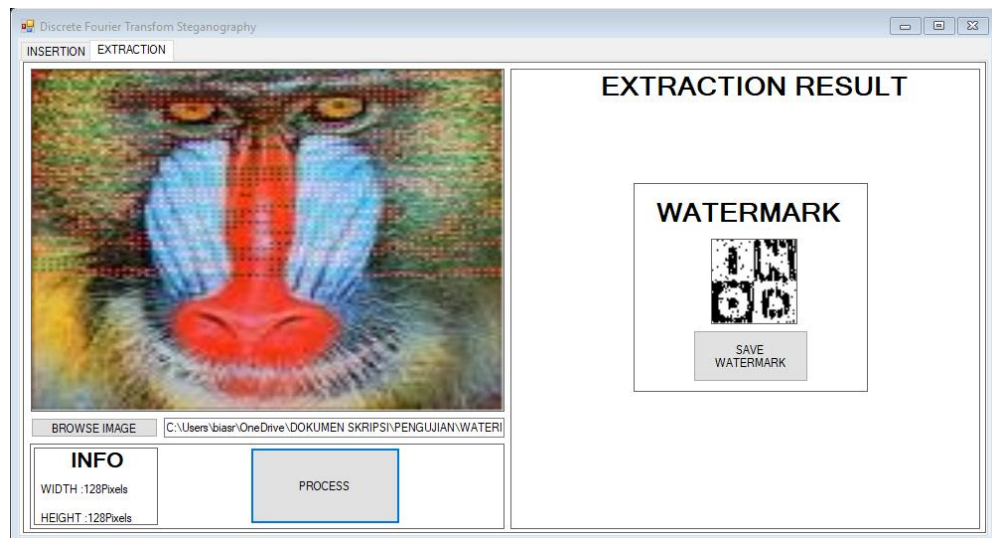
Gambar 5. 4 Halaman *extraction*

1. Klik tombol “BROWSE IMAGE” untuk membuka dan memilih citra yang telah disisipi oleh *watermark*.



Gambar 5. 5 Halaman *open image*

2. Klik tombol “PROCESS” untuk melakukan proses *extraction watermark* dari citra penampung.



Gambar 5. 6 Hasil *extraction* pada gambar ber-*watermark*

3. Klik tombol “SAVE WATERMARK” untuk menyimpan *watermark* yang telah berhasil di ekstraksi dari *embeded image* atau citra penampung.

## BAB VI. PENGUJIAN DAN PEMBAHASAN

Setelah melakukan proses penyisipan pesan dan ekstraksi pesan, maka untuk melihat apakah hasil dari *insertion watermark* dan *extraction watermark* telah berhasil akan dilakukan suatu pengujian.

Pengujian dilakukan berdasarkan spesifikasi sistem dan pengujian ketahanan data. Pengujian spesifikasi sistem yang dilakukan meliputi pengujian kesesuaian proses, pengujian kesesuaian data, dan pengujian kualitas citra. Pengujian berdasarkan spesifikasi sistem dan ketahanan data diuraikan menjadi 2 faktor pengujian sebagai berikut:

- a. Kesesuaian proses, yaitu aplikasi dapat melakukan proses penyisipan pesan dan ekstraksi pesan.
- b. Kesesuaian data, yaitu pengujian kesesuaian antara data yang berhasil diekstraksi dengan data yang disisipkan.

### 6.1. Pengujian Sistem

Untuk tahap pengujian sistem menggunakan metode blackbox. Metode ini memungkinkan adanya pengembangan untuk melatih seluruh fungsi pada sistem. Metode ini digunakan untuk mendemonstrasikan jalannya aplikasi dan menemukan kesalahan saat aplikasi dijalankan. Dengan menggunakan metode ini dapat dinilai apakah input yang diterima dan output yang dihasilkan sudah tepat atau belum. Berikut blackbox dari pengujian sistem:

Tabel 6. 1 Pengujian *Blackbox*

No	Skenario Pengujian	Hasil yang diharapkan	Hasil pengujian	Kesimpulan
1	Pada tab <i>INSERTION</i> klik tombol “ <i>open image</i> ” untuk memilih citra penampung	pbInputInsertion menampilkan citra penampung	Sesuai Harapan	Berhasil

	Pada tab <i>INSERTION</i> klik tombol “ <i>process</i> ” untuk menyisipkan <i>watermark</i> ke dalam citra penampung	imgOutputInsertion menampilkan hasil citra yang telah tersisipi oleh <i>watermark</i>	Sesuai harapan	Berhasil
3	Pada tab <i>INSERTION</i> klik tombol “ <i>save image</i> ” untuk menyimpan citra berwatermark yang telah di proses	Muncul window untuk menyimpan gambar	Sesuai harapan	Berhasil
4	Pada tab <i>Extraction</i> klik tombol “ <i>Browse image</i> ” untuk memilih <i>embeded image</i> atau citra yang telah di disisipi <i>watermark</i>	Muncul window untuk memilih citra. Dan citra yang di pilih akan muncul pada imgEmbeddedImage	Sesuai harapan	Berhasil
5	Pada tab <i>Extraction</i> klik tombol “ <i>process</i> ” untuk mengekstraksi <i>watermark</i> pada <i>embeded image</i>	Hasil watermark yang diekstraksi akan muncul pada imgEmbeddedWatermark	Sesuai harapan	Berhasil
6	Pada <i>Extraction</i> klik “ <i>save watermark</i> ” untuk menyimpan <i>watermark</i>	Muncul window untuk menyimpan <i>watermark</i>	Sesuai harapan	Berhasil

Pada Tabel dapat dilihat hasil pengujian sistem menggunakan blackbox. Berdasarkan dari hasil pengujian sistem menggunakan blackbox, maka dapat ditarik kesimpulan bahwa aplikasi penyisipan watermark menggunakan metode DFT sudah berjalan sesuai dengan harapan.



## 6.2. Analisis

Untuk melakukan analisis terhadap citra, dilakukan beberapa cara untuk menganalisis hasil penyisipan dan hasil ekstraksi. Langkah ini dilakukan untuk menguji ketahanan, keakurasian, kecepatan program yang telah dibuat.

### 6.2.1 Analisis Perbandingan Citra

Pada sub bab ini dilakukan proses membandingkan citra. Perbandingan citra dilakukan dengan membandingkan antara citra asli (*source image*), dengan citra berwatermark (*embedded image*). Tiap-tiap citra akan disisipi *watermark* pada koordinat subblock 1x1, pada table ditunjukkan hasil pengujian yang diukur dengan UACI dan NPCR:

Tabel 6. 2 Perbandingan UACI dan NPCR


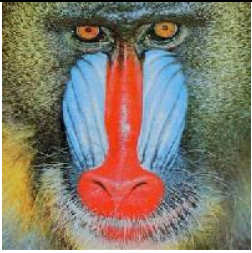


No	Citra	UACI	NPCR	PSNR
1	Picture 1	0.378099329331342	0.9490966796875	27.9304298620428
2	Picture 2	6.50596469056373	15.234375	14.2548977779448
3	Picture 3	0.363214052287582	28.316875	32.6551739489079
4	Picture 4	1.53128513071895	34.9057291666667	25.6782653216038

Dari hasil pengujian diatas *Picture 2* memiliki skor terrendah pada kedua pengujian yangtelah dilakukan. Dan *picture 1* memiliki skor terbaik dari keempat file yang diuji.

### 6.2.2 Analisis Perbandingan Waktu *Insertion* dan *Extraction*

Pada sub bab ini dilakukan analisis waktu *insertion* dan *extraction* pada pesan yang akan disisipkan pada citra. Terdapat 4 masukan citra penampung dengan ukuran dimensi citra yang berbeda, yang mempengaruhi kecepatan penyisipan *watermark* kedalam citra penampung:


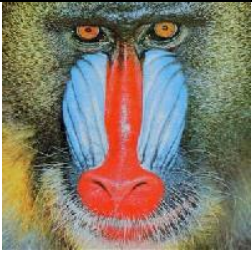


Tabel 6. 3 Perbandingan Waktu *Insertion*

No	Citra penampung	Dimensi Citra	Waktu
1		512x512	05,67 detik
2		128x128	0,88 detik
3		1200x800	19,84 detik
4		1200x800	20,66 detik

Dari table *insertion* diatas dapat disimpulkan bahwa durasi waktu *insertion* dipengaruhi oleh dimensi atau ukuran *source image*. Semakin besar *source image*, semakin lama waktu yang dibutuhkan dalam proses *insertion*. Demikian sebaliknya, semakin kecil ukuran *source image* semakin cepat waktu yang dibutuhkan untuk melakukan proses *insertion*.

Sedangkan untuk hasil perbandingan waktu ekstraksi dijelaskan pada tabel berikut:

Tabel 6. 4 Perbandingan Waktu Ekstraksi






No	Citra penampung	Dimensi Citra	Waktu
1		512x512	2,68 detik
2		128x128	0,48 detik
3		1200x800	9,87 detik
4		1200x800	9,90 detik

Dari table ekstraksi diatas dapat disimpulkan bahwa durasi waktu ekstraksi dipengaruhi oleh dimensi atau ukuran dari *embeded image*, sama seperti proses penyisipan. Semakin besar *embeded image*, semakin lama waktu yang dibutuhkan dalam proses ekstraksi. Demikian sebaliknya, semakin kecil ukuran *embeded image* semakin cepat waktu yang dibutuhkan untuk melakukan proses ekstraksi.

### 6.2.3 Analisis Hasil Penyisipan

Selanjutnya dilakukan proses pengujian kesesuaian antara *watermark* asli, dengan hasil watermark hasil *extraction*.

Tabel 6. 5 Perbandingan Hasil Extraksi

No	Gambar	Watermark Asli	Hasil <i>Extraction</i>
1	Picture 1.jpg		
2	Picture 2.jpg		
3	Picture 3.jpg		
4	Picture 4.jpg		







Dari hasil *extraction* yang dilakukan terhadap 4 *source image*, terlihat pada Picture 1 & Picture 2 *watermark* Nampak sangat jelas bila dibandingkan dengan *watermark* hasil ekstraksi dari Picture 3 & Picture 4.





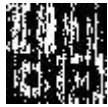

#### 6.2.4 Analisis Ketahanan Citra Terhadap Serangan

Analisis terhadap serangan dilakukan dengan melakukan *cropping*, *fltering*, dan *resize* terhadap *embedded image*. Berikut merupakan hasil pengujian *cropping* pada *source image*:

##### a. Serangan *cropping*

Tabel 6. 6 Hasil Extraction setelah serangan cropping

No	Gambar Setelah Serangan	Hasil <i>Watermark</i> Asli	Hasil Setelah Serangan
1			
2			













No	Gambar Setelah Serangan	Hasil <i>Watermark</i> Asli	Hasil Setelah Serangan
3			
4			

Dari hasil pengujian *cropping* terhadap *embeded image* di atas, hanya *watermark* hasil ekstraksi pada Picture 3 yang masih memiliki kemiripan dengan hasil ekstraksi sebelum *serangan*. Dan untuk citra yang lainnya, semua tidak tahan terhap serangan *cropping*, terlihat bahwa semua *watermark* hasil ekstraksi tidak memiliki kemiripan. dan dapat disimpulkan bahwa serangan *cropping* terhadap *embeded image* menghasilkan kualitas *watermak* yang tidak sesuai.

b. Serangan filtering/editing

Pengujian kedua adalah pengujian serangan *filtering* atau memberikan efek atau filter terhadap *embeded image*. Serangan ini seringkali dilakukan untuk merubah penampilan dari gambar. Perubahan yang biasa dilakukan adalah memberikan efek filter untuk mendapatkan gambar sesuai dengan keinginan. berikut adalah table perbandingan terhadap serangan *filtering* terhadap *embeded image*:

Tabel 6. 7 Hasil *Extraction* setelah di *filtering*

No	Gambar Setelah Serangan	Hasil <i>Watermark</i> Asli	Hasil Setelah Serangan
1			
2			
3			
4			











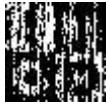

Dari hasil pengujian *filtering* terhadap *embeded image* di atas, keseluruhan gambar telah melalui skenario *editing* atau pemberian *filter*. Namun serangan tersebut tidak berpengaruh terhadap *watermark* yang telah tersisipi didalamnya. Pada Picture 3 dan Picture 4 terlihat *watermak* justru semakin terlihat setelah dilakukan serangan. Dan dapat disimpulkan bahwa serangan *editing* dan *filtering* tidak berpengaruh besar terhadap *watermark* yang telah tersisipi kedalam *embeded image*.



c. Serangan *resize*

Dan pengujian serangan yang terakhir adaah serangan *resize* terhadap *embeded image*. Pengujian ini dilakukan dengan cara mengecilkan & membesarkan ukuran *embeded image*. Skema serangan dlakukan dengan menambahkan dan mengurangi ukuran *embeded image* seesar 10 *pixel*. berikut merupakan table perbandingan serangan dengan membesarkan *size* gambar:











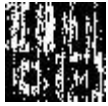

Tabel 6. 8 Hasil *Extraction* setelah di *resize* (pembesaran)

No	Gambar Setelah Serangan	Hasil Watermark Asli	Hasil Setelah Serangan
1	 532x532		
2	 138x138		
3	 1210x806		
4	 1210x806		

Dari hasil serangan *resize* dengan membesarkan gambar sebesar 10 *pixel*, Picture 1 dan Picture 3 terlihat masih menghasilkan *watermark* yang masih sedikit terbaca. Sedangkan pada Picture 2 dan Picture 4, menghasilkan *watermark* yang tak terbaca.

Serangan yang terakhir adalah *resize* terhadap *embedded image*, dengan mengecilkan ukuran *embedded image* sebesar 10 *pixel*. berikut adalah table hasil perbandingan serangan *reize* yang ke 2:

Tabel 6. 9 Hasil *Extraction* setelah di *resize* (pengecilan)

No	Gambar Setelah Serangan	Hasil <i>Watermark</i> Asli	Hasil Setelah Serangan
1	 502x502		
2	 118x118		
3	 1190x793		
4	 1190x793		



dari skenario serangan *resize* dengan mengecilkan ukuran *embeded image* diatas. Hanya Picture 3 saja yang memiliki sedikit kemiripan dengan hasil ekstraksi *embeded image* sebelum serangan. Dan dapat disimpulkan bahwa mengecilkan/*resize* terhadap *embeded image* dapat merusak *watermark* yang telah tersisipi kedalam citra penampung.

## BAB. VII PENUTUP

### 7.1 Kesimpulan

Penyisipan *watermark* menggunakan metode *Discrete Fourier Transform* telah dilakukan dengan menggunakan bahasa pemrograman C#. Berdasarkan hasil pengujian yang telah dilakukan, dapat ditarik kesimpulan:

- a. *Embedded image* dengan ukuran dimensi citra diatas 512x512 *pixel* memiliki ketahanan terhadap serangan *editing* atau *filtering* dibandingkan citra dengan dimensi citra dengan dimensi dibawah 512x512 *pixel*.
- b. *Embedded image* tidak tahan terhadap serangan *resize* pembesaran dan pengecilan diatas 10 *pixel*.
- c. *Watermark* akan tidak terbaca jika mendapat serangan *cropping* sebesar 10 *pixel* pada bagian atas *embedded image*
- d. Citra dengan ukuran diatas 512x512 *pixel*, memiliki nilai UACI dan NPCR yang kecil dibandingkan dengan citra dengna ukuran dibawah 512x512 *pixel*
- e. Pada pengujian PSNR, citra dengan ukuran 512x512 *pixel* memiliki tingkat kemiripan yang lebih banyak jika dibandingkan dengan citra dengan ukuran dibawah 512x512 *pixel*
- f. Ukuran citra berpengaruh terhadap kualitas citra penapung.
- g. Ukuran dimensi citra berpengaruh terhadap durasi waktu *insertion & extraction*.

### 7.2 Saran

Berdasarkan penelitian yang diperoleh, ada beberapa saran untuk pengembangan sistem lebih lanjut, sebagai berikut:

- a. Pengguna dapat memilih *watermark* sesuai dengan keinginan pengguna.

## DAFTAR PUSTAKA

- [1] Dhian Sweetania, ST., MMSI. 2015. “*Metode Endkripsi Dekripsi*”. [Online]  
Tersedia:  
[http://dhian\\_sweetania.staff.gunadarma.ac.id/Downloads/files/35348/Enkripsi-I.pdf](http://dhian_sweetania.staff.gunadarma.ac.id/Downloads/files/35348/Enkripsi-I.pdf) [26 Desember 2016]
- [2] *Pengertian citra digital* . 2013 [Online]  
Tersedia:  
<http://www.temukanpengertian.com/2013/08/pengertian-citra-digital.html>  
[26 Desember 2016]
- [3] *Pengertian citra digital*, 2012 [Online]  
Tersedia :  
<http://repository.usu.ac.id/bitstream/123456789/31325/4/Chapter%20II.pdf>  
[26 Desember 2016]
- [4] Pengertian Transformasi Fourier Diskrit [Online]  
Tersedia :  
<http://www.landasanteori.com/2015/10/pengertian-transformasi-fourier-diskrit.html> [29 Desember 2016]
- [5] Image Authentication Technique in Frequency Domain based on Discrete Fourier Transformation (IATFDDFT). [Online]  
Tersedia:  
<https://arxiv.org/abs/1212.3371> [29 Desember 2016]
- [6] DFT Based Image Enhancement and Steganography [Online]  
Tersedia:  
[static.ijcsce.org/wp-content/uploads/2013/03/IJCSCE020213.pdf](http://static.ijcsce.org/wp-content/uploads/2013/03/IJCSCE020213.pdf)  
[1 Januari 2017]
- [7] Rahul, Lokesh, Salony, 2013, Image Steganography With LSB, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), Volume 2, Issue 1, January. [Online]  
Tersedia:  
<http://www.ijarcet.org/index.php/ijarcet/article/download/675/pdf>  
[5 Maret 2017]

## **LAMPIRAN-LAMPIRAN**

## Lampiran 1. Pembagian subblock

```
public List<Bitmap> subBlokCitraAwal(Image hostImage, int
blockSize)
{
    List<Bitmap> res = new List<Bitmap>();
    int jumlahBaris = hostImage.Height / blockSize;
    int jumlahKolom = hostImage.Width / blockSize;
    for (int i = 0; i <= jumlahBaris - 1; i++)
    {
        for (int j = 0; j <= jumlahKolom - 1; j++)
        {
            Bitmap subBlokCitra = new Bitmap(blockSize,
blockSize);
            Graphics subBlock =
Graphics.FromImage(subBlokCitra);
            subBlock.DrawImage(hostImage, 0, 0, new Rectangle(j
* blockSize, i * blockSize, blockSize, blockSize),
GraphicsUnit.Pixel);
            res.Add(subBlokCitra);
        }
    }
    return res;
}
```

## Lampiran 2. Perhitungan DFT

```
public void hitungDFT()
{
    if (jumlahData > 0)
    {
        for (int k = 0; k <= jumlahData - 1; k++)
        {
            komponenFourier[k].real = 0;
            komponenFourier[k].imajiner = 0;
            for (int l = 0; l <= (jumlahData - 1) - 1; l++)
            {
                komponenFourier[k] =
hitungPenjumlahanKompleks(komponenFourier[k],
hitungPerkalianKompleks(nilaiKompleks[(k * l) % jumlahData],
data[l]));
            }
            komponenFourier[k].real = komponenFourier[k].real /
jumlahData * 2;
            komponenFourier[k].imajiner = -komponenFourier[k].imajiner
/ jumlahData * 2;
        }
        komponenFourier[0].real = komponenFourier[0].real / 2;
        komponenFourier[0].imajiner = komponenFourier[0].imajiner / 2;
    }
}
```

### Lampiran 3. Perhitungan DFT

```
if (count <= watermark.Length - 1)
{
    int w = (listsubblock.ElementAt(s).Width) - 1;
    int h = (listsubblock.ElementAt(s).Height) - 1;
    Bitmap img = new Bitmap(pbImgWatermark.Image);
    if (watermark[count] == 255) //putih
    {
        Bitmap img2 = new Bitmap(listsubblock.ElementAt(s));
        int r = img2.GetPixel(w, h).R;
        int g = img2.GetPixel(w, h).G;
        int b = img2.GetPixel(w, h).B;
        int newValueR = 255;
        listsubblock.ElementAt(s).SetPixel(w, h,
        Color.FromArgb(newValueR, g, b));
    }
    else
    {
        Bitmap img2 = new Bitmap(listsubblock.ElementAt(s));
        int r = img2.GetPixel(w, h).R;
        int g = img2.GetPixel(w, h).G;
        int b = img2.GetPixel(w, h).B;
        int newValueR = 50;
        listsubblock.ElementAt(s).SetPixel(w, h,
        Color.FromArgb(newValueR, g, b));
    }
}
```

### Lampiran 4. Perhitungan IDFT

```
public void hitungIDFT(int jumlahKomponen)
{
    for (int k = 0; k <= jumlahData - 1; k++)
    {
        output[k] = 0;
        for (int l = 0; l <= (jumlahKomponen - 1); l++)
        {
            output[k] = output[k] + komponenFourier[l].real *
            Math.Cos(2.0 * Math.PI * Convert.ToDouble(l * k) /
            Convert.ToDouble(jumlahData)) + komponenFourier[l].imaginer *
            Math.Sin(2.0 * Math.PI * Convert.ToDouble(l * k) /
            Convert.ToDouble(jumlahData));
        }
    }
}
```

## Lampiran 5. Penggabungan subblock citra

```
public Bitmap gabungCitra(List<Bitmap> citraIDFT, Bitmap citraAsal)
{
    int lebar = citraAsal.Width;
    int panjang = citraAsal.Height;
    int jumlahBaris = panjang / 2;
    int jumlahKolom = lebar / 2;
    Bitmap subBlok = null;
    int indexBlok = 0;
    int x = 0;
    int y = 0;
    int tinggi = 0;
    Bitmap citraAkhir = new Bitmap(lebar, panjang);
    Graphics grp = Graphics.FromImage(citraAkhir);

    for (int i = 0; i <= jumlahBaris - 1; i++)
    {
        x = 0;
        for (int j = 0; j <= jumlahKolom - 1; j++)
        {
            subBlok = citraIDFT[indexBlok];
            indexBlok += 1;
            grp.DrawImage(subBlok, new Rectangle(x, y, subBlok.Width, subBlok.Height));
            x += subBlok.Width;
            tinggi = subBlok.Height;
        }
        y += tinggi;
    }
    return citraAkhir;
}
```

## Lampiran 6. Membaca watermark

```
if (s <= imgW.Length - 1)
{
    Bitmap img = new Bitmap(listsubblock.ElementAt(s));
    int red = img.GetPixel(img.Width - 1, img.Height - 1).R;
    if (red >= 1)
    {
        imgW[s] = 255;
    }
    else
    {
        imgW[s] = 0;
    }
}
```