**BAB I**

**PENDAHULUAN**

1. **Latar Belakang**

Teknologi internet saat ini sudah sangat mempengaruhi setiap sisi kehidupan. Internet memberikan kemudahan bagi masyarakat dalam mendapatkan informasi-informasi yang belum tentu berhak mereka terima. Informasi-informasi ini yang dapat disalah gunakan oleh para pelaku kejahatan untuk melakukan kejahatan. Karena itulah diperlukan sebuah mekanisme perlindungan informasi dan data pada internet agar data-data dapat terlindungi dari orang-orang yang tidak memiliki hak untuk mengakses data tersebut.

Terdapat berbagai metode untuk melindungi data-data yang ada agar tidak dapat disalah gunakan, salah satu metode tersebut yaitu steganografi. Steganografi adalah suatu teknik untuk menyembunyikan informasi yang bersifat pribadi dengan sesuatu yang hasilnya akan tampak seperti informasi normal lainnya. Secara umum teknik steganografi menggunakan *redundant* *bits* sebagai tempat menyembunyikan pesan dan kemudian menggunakan kelemahan indera manusia yang tidak *sensitive* sehingga pesan tersebut tidak terlihat. Teknik *End of File* dan *First of File* merupakan salah satu teknik yang digunakan dalam steganografi. Teknik ini digunakan dengan cara menambahkan data atau pesan rahasia seperti gambar maupun teks pada akhir atau awal *file*. Perhitungan ukuran *file* yang telah disisipkan data sama dengan ukuran *file* sebelum disisipkan data ditambah ukuran data rahasia yang telah diubah menjadi encoding *file* [1].

Penelitian tentang steganografi metode *End of File* menyebutkan efek samping penggunaan metode *End of File* atau *First of File* adalah ukuran berkas menjadi lebih besar dari ukuran semula. Ukuran berkas yang terlalu besar dari yang seharusnya, tentu akan menimbulkan kecurigaan bagi yang mengetahuinya [2], oleh karenanya diperlukan sebuah proses kompresi terhadap pesan agar dapat mengurangi ukuran data pesan. Salah satu metode untuk melakukan kompresi adalah *Run-Length-Encoding*, Penelitian tentang *Run-Length-Encoding* dan steganografi menyebutkan *Run-Length-Encoding* menyembunyikan data hampir setengah dari ukuran *file* dan pesan terkompresi dapat didekompresi dengan baik [3].

Terdapat kekurangan yang dimiliki teknik kompresi *Run-Length-Encoding* jika diterapkan pada data pixel warna pada gambar 8 bit maupun dalam bentuk ASCII 8 bit yaitu terdapat kemungkinan hasil kompresi menjadi lebih besar dari ukuran *file* asli. Ini disebabkan oleh proses perhitungan panjang data juga dilakukan pada data-data yang tidak berulang. Oleh karena permasalahan tersebut pada penelitian ini akan dilakukan modifikasi terhadap teknik kompresi *Run-Length-Encoding* agar dapat menghilangkan kemungkinan adanya hasil kompresi yang lebih besar dari ukuran asli sehingga dapat diterapkan dengan maksimal pada steganografi *First of File* dan *End of File.*

1. **Rumusan Masalah**

Berdasarkan pada latar belakang yang telah diuraikan, perumusan masalah yang didapat pada tugas akhir ini yaitu sebagai berikut:

1. Bagaimana kemampuan kompresi *Run-Length-Encoding* dan *Run-Length-Encoding* yang dimodifikasi?
2. Bagaimana membuat sebuah aplikasi yang dapat mengkompresi gambar dan menyisipkannya pada cover?
3. Bagaimana kemampuan steganografi dengan menggunakan metode *First of File*, *End of File,* dan kombinasinya jika diterapkan pada berbagai jenis *file*?
4. **Batasan masalah**

Dari permasalahan yang telah diuraikan di atas terdapat batasan-batasan masalah dalam pembangunan aplikasi ini yaitu:

1. Pesan yang disisipkan berupa *file* gambar
2. Ukuran *file* pesan dan *file* cover kurang dari 10 *Megabytes.*
3. *File* cover yang digunakan adalah docx, pptx, xlsx, pdf, gambar JPEG, gambar PNG dan BMP.
4. **Tujuan**

Adapun tujuan yang akan dicapai dari tugas akhir ini yaitu:

1. Mengetahui kemampuan kompresi *Run-Length-Encoding* dan *Run-Length-Encoding* yang dimodifikasi jika diterapkan pada steganografi.
2. Membuat sebuah aplikasi yang dapat mengkompresi gambar dan menyisipkannya ke dalam cover.
3. Mengetahui jenis-jenis file yang dapat diterapkan metode steganografi *First of File*, *End of File* dan kombinasinya.
   1. **Manfaat**

Manfaat yang diperoleh dari tugas akhir ini adalah:

1. Bagi penyusun
2. Menerapkan ilmu yang telah diperoleh selama proses perkuliahan di Teknik Informatika Universitas Mataram.
3. Menambah wawasan dalam bidang steganografi dan kompresi data.
4. Pembelajaran untuk mengembangkan algoritma yang bertujuan menyelesaikan suatu permasalahan.
5. Bagi pembaca
6. Menambah ilmu pengetahuan terutama mengenai teknologi penyisipan pesan dalam steganografi.
7. Menerapkan hasil dari skripsi ini pada permasalahan yang berhubungan dengan penyisipan pesan dan pengamanan informasi digital.
   1. **Sistematika Penulisan**

Sistematika penulisan dalam penyusunan tugas akhir ini adalah sebagai berikut:

1. Bab I Pendahuluan

Bab ini menjelaskan dasar-dasar dari penulisan laporan tugas akhir, yang terdiri dari latar belakang, rumusan masalah, batasan masalah, tujuan, serta sistematika penulisan laporan tugas akhir.

1. Bab II Landasan Teori

Bab ini membahas teori-teori yang berhubungan dengan topik penelitian, meliputi Java, Steganografi, *End of File* dan *First of File* dan format citra.

1. Bab III Metodologi Penelitian

Bab ini membahas tentang metodologi yang digunakan dalam penelitian dan pengembangan perangkat lunak.

1. Bab IV Analisis Dan Perancangan

Pada bab ini merupakan pembahasan tentang analisis perangkat lunak, meliputi analisis, analisis masalah, analisis metode, analisis kebutuhan sistem, serta perancangan. Perancangan sistem yang terdiri dari perancangan diagram alir (*flowchart*).

1. Bab V Implementasi Dan Pengujian Metode

Bab ini membahas implementasi yang dilakukan terhadap steganografi pada citra di *desktop* menggunakan bahasa pemrograman Java dengan metode Kombinasi EOF dan FOF*.*

**BAB II**

**TINJAUAN PUSTAKA DAN DASAR TEORI**

**2.1 Tinjauan Pustaka**

Penelitian tentang implementasi *Run Length Encoding* pada kompresi citra [4] menunjukkan citra yang mempunyai sebaran nilai piksel tidak merata memiliki rasio kompresi yang relatif besar sedangkan citra dengan nilai piksel yang merata memiliki rasio kompresi yang lebih kecil. Pada penelitian ini kompresi dilakukan pada citra grayscale dengan mengelompokkan pixel-pixel yang seragam, teknik kompresi masih dilakukan dengan menulis jumlah dari setiap pixel meskipun pixel tidak berulang. Perbedaannya dengan penelitian yang akan dilakukan adalah kompresi akan dilakukan pada *byte* data citra dengan menggunakan kompresi *Run Length Encoding* yang akan dimodifikasi sehingga kompresi hanya akan dilakukan pada data yang berulang lebih dari 5 kali.

Penelitian tentang digital watermarking dengan menggunakan metode *End of File* telah berhasil menyisipkan watermark pada akhir suatu *file* cover [1]. Pada penelitian ini dikatakan beberapa *file* seperti exe dan zip, penempatan flag di awal *file* asli tidak akan menjadi masalah, namun untuk jenis *file* lain semisal JPG, BMP dan DOC, penempatan flag di awal *file* akan merusak *file* asli karena mengganggu isi *file* asli dan merusak CRC (*Cyclic Redundancy Check*) *file* tersebut. Persamaan sekaligus menjadi perbedaanya dengan penelitian yang akan dilakukan terletak pada metode steganografi yaitu metode *End of File* yang pada penelitian ini akan dimodifikasi dan digabungkan dengan metode *First of File*.

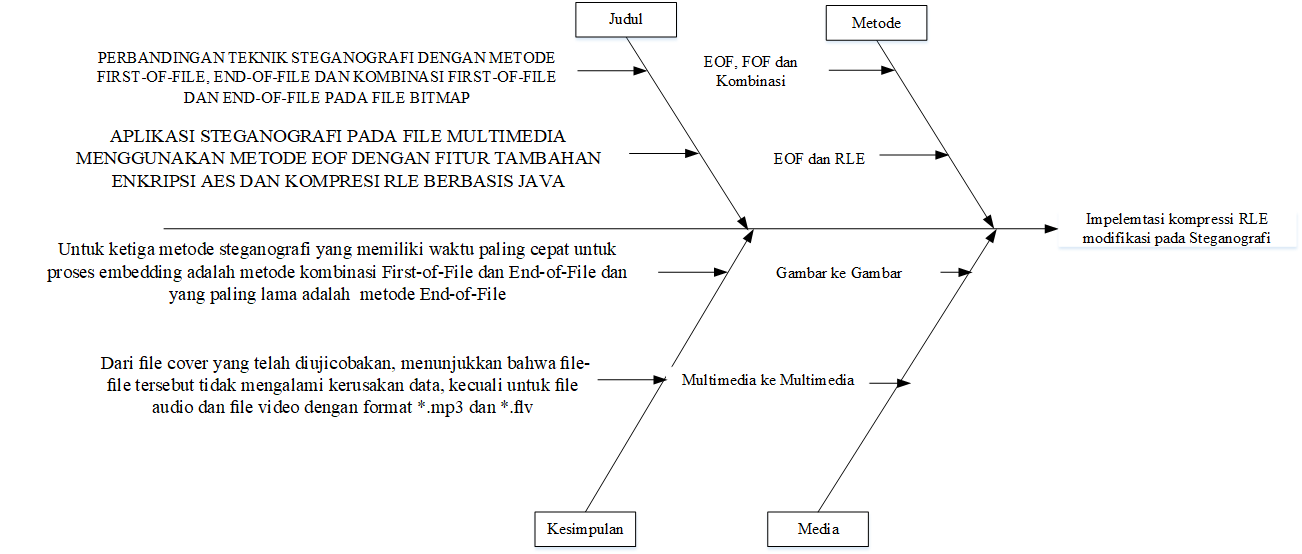
Penelitian tentang teknik steganografi *End of File*, *First of File* dan kombinasinya berhasil diimplementasikan pada penyisipan citra ke citra [5]. Pada penelitian ini pesan disisipkan ke dalam pixel citra, penyisipan yang dilakukan ke dalam pixel citra membuat perubahan yang tampak secara kasat mata pada cover citra sehingga menimbulkan kecurigaan pada citra hasil penyisipan. Sehingga pada penelitian ini dilakukan penyisipan dengan metode yang sama tetapi penyisipan dilakukan pada *byte* data cover dan dilakukan kompresi terlebih dahulu pada pesan.

Penelitian tentang kompresi *Run-Length-Encoding* pada citra digital yang telah dilakukan menunjukkan kompresi *Run-Length-Encoding* tanpa modifikasi yaitu dengan menuliskan jumlah setiap data walaupun data tersebut tidak berulang akan menyebabkan hasil kompresi menjadi lebih besar dari ukuran *file* asli. Terlihat pada hasil pengujian terdapat rasio yang -0,08% yang artinya hasil kompresi lebih besar 0,08 % dari ukuran asli. Melihat hasil tersebut pada penelitian ini akan diterapkan *Run-Length-Encoding* yang akan dimodifikasi sehingga hanya data-data yang berulang lebih dari 5 data yang akan dikompresi [6].

Penelitian tentang kompresi *Run Length Encoding* pada citra dan diimplementasikan pada penyisipan citra ke dalam citra [3] yang dilakukan menghasilkan algoritma *Run Length Encoding* menyembunyikan data hampir setengah dari ukuran *cover* dalam ukuran dan menjamin pemulihan pesan secara efisien dengan kesehatan penuh. Kualitas citra stego tidak terpengaruh secara signifikan dan PSNR dari gambar stego yang dihasilkan tetap berada di atas ambang batas 30dB untuk kapasitas penyembunyian data sebesar 50%.

Penelitian tentang implementasi *Run Length Encoding* pada kompresi citra dengan digabungkan dengan kompresi Arithmetic Coding [7]. Hasil penelitian menunjukkan pada perhitungan dengan menggunakan algoritma kompresi *Run Length Encoding* rasio kompresi dapat mencapai -33% yang artinya hasil kompresi menjadi lebih besar 33% dari ukuran asli data. Kelemahan yang ditunjukkan pada penelitian ini diakibatkan oleh semua data pada pixel dicoba untuk dikompresi meskipun data tersebut tidak berulang, hal ini yang menyebabkan adanya kemungkinan hasil kompresi menjadi lebih besar. Pada penelitian yang akan dilakukan kompresi *Run Length Encoding* akan dimodifikasi sehingga hanya data yang berulang lebih dari 5 yang akan dikompresi.

Penelitian tentang implementasi steganografi *End of File* dan kompresi *Huffman* menunjukkan bahwa *End of File* berhasil menyisipkan data dengan ukuran besar sehingga memerlukan kompresi data [8]. Hasil penelitiannya menunjukkan kompresi *Huffman* bekerja sangat baik pada teks dengan rasio kompresi 55,067% dan bekerja kurang efektif pada *file* multimedia seperti gambar, audio, dan video. Pada penelitian yang akan dilakukan metode steganografi yang akan digunakan adalah menggabungkan metode *First of File* dan *End of File* serta menggunakan kompresi *Run Length Encoding*.



Gambar 2.1 Gambar diagram *fishbone*

Gambar 2.1 adalah diagram yang menggambarkan penelitian-penelitian sebelumnya yang berkaitan tentang penelitian yang akan dilakukan.

**2.2** **Dasar Teori**

* + 1. **Algoritma Run Length Encoding**

Algoritma RLE menggunakan pendekatan ruang. Algoritma ini cocok digunakan untuk memampatkan citra yang memiliki kelompok-kelompok piksel berderjat keabu-abuan yang sama. Metode ini dilakukan dengan menyatakan seluruh baris citra menjadi sebuah baris *run*, lalu menghitung *run – length* untuk setiap derajat keabu-abuan yang berurutan [9].

Contoh, diketahui sebuah citra memiliki *byte* data sebagai berikut dalam vektor:

Hasil kompresi:

= (1,2)(2,1) (1,1)(4,1) (6,2)(3,1) (1,1)(4,8) (6,1)(2,1)(7,2) (2,2)(3,1)(4,1)(1,2)

= (1 2 2 1 1 1 4 1 6 2 3 1 1 1 4 8 6 1 2 1 7 2 3 1 4 1 1 2)

Ukuran sebelum dikompresi = 25 \* 1 *byte* = 25 *byte*

Ukuran sesudah dikompresi = 28 \* 1 *byte* = 28 *byte*

Menghitung rasio kompresi dengan rumus:

Rasio = 100 % - 28/25 x 100% = -12%

Terlihat pada contoh, hasil kompresi menjadi lebih besar dari ukuran *file* asli sebanyak 12%.

Pada penelitian ini dilakukan modifikasi yaitu dengan mengelompokkan data *byte* yang sama dari citra dan penulisan jumlah data hanya dilakukan pada *byte* yang berulang lebih dari 5 kali. Jika menerapkan teknik modifikasi ini maka diperlukan *byte* penanda yang digunakan untuk mengetahui bahwa *byte* data tersebut merupakan *byte* yang terkompresi. Penentuan penanda dilakukan dengan mencari *byte* yang memiliki frekuensi kemunculan terkecil dari *byte* pesan, *byte* ini yang akan dijadikan sebagai penanda.

Menggunakan contoh yang sama dengan menerapkan teknik modifikasi pada kompresi menghasilkan:

Sebelum dikompresi = 25 \* 1 *byte* = 25 *byte*

Sesudah dikompresi = 20 \* 1 *byte* = 20 *byte*

Menghitung rasio kompresi sesuai dengan persamaan 2-1:

100 % - 20/25 x 100% = 20%

Terlihat hasil kompresi menjadi lebih baik dengan rasio kompresi 20% dengan tidak mengkompresi data yang memiliki panjang data berulang <5. *Byte* data (4 0 8) adalah merupakan hasil kompresi, dengan *byte* 0 sebagai penanda, penanda didapatkan dari mencari frekuensi kemunculan setiap *byte*. Seperti yang terlihat pada Tabel 2.1 *byte* 0 memiliki frekuensi 0.

Tabel 2.1 Frekuensi *byte*

| *Byte* | Frekuensi |
| --- | --- |
| 0 | 0 |
| 1 | 6 |
| 2 | 4 |
| 3 | 2 |
| 4 | 10 |
| 5 | 0 |
| 6 | 2 |
| 7 | 1 |

Pemilihan panjang data berulang minimal 5 dilakukan untuk menghindari hasil kompresi yang lebih besar dari ukuran asli, pada modifikasi ini akan terdapat 3 kasus pencarian frekuensi yaitu:

* 1. Terdapat *byte* data yang berfrekuensi 0 sehingga membutuhkan 1 penanda saja.
  2. Penanda hasil kompresi dengan 2 tanda, jika tidak terdapat *byte* berfrekuensi 0 maka akan dilakukan pencarian penanda baru dengan 2 tanda tetapi tetap memperhatikan *byte* data yang memiliki frekuensi terkecil.
  3. Penanda hasil kompresi dengan 3 tanda, jika tidak terdapat *byte* kombinasi yang sesuai dengan 2 tanda maka akan dilakukan pencarian penanda baru dengan 3 tanda dan tetap memperhatikan *byte* data yang memiliki frekuensi terkecil.

Dengan melihat kasus ketiga yaitu dengan 3 penanda, maka ada kemungkinan jika mengkompresi data yang memiliki data berulang <5 akan terjadi penambahan jumlah *byte* yang tidak perlu.

* + 1. **Steganografi**

Kata steganografi berasal dari bahasa Yunani, yaitu dari kata Stegos yang berarti tersembunyi (*covered*) dan Graphein yang berarti tulisan (*writing*). Steganografi di dunia modern biasanya mengacu pada informasi atau suatu arsip yang telah disembunyikan ke dalam suatu arsip citra digital, audio, atau video. Teknik steganografi ini telah banyak digunakan dalam strategi peperangan dan pengiriman sandi rahasia sejak jaman dahulu kala.

Semakin pentingnya nilai dari sebuah informasi, maka semakin berkembang pula metode-metode yang dapat digunakan untuk melakukan penyisipan informasi yang didukung pula dengan semakin berkembangnya media elektronik. Berbagai macam media elektronik kini telah dapat digunakan untuk melakukan berbagai fungsi steganografi dengan berbagai macam tujuan dan fungsi yang diharapkan oleh penggunanya. Sebagai fungsi yang umum, steganografi digunakan untuk memberikan cap khusus dalam sebuah karya yang dibuat dalam format media elektronik sebagai identifikasi.

Dua teknik lain yang sangat erat kaitannya dengan steganografi adalah watermarking dan *fingerprinting*. Kedua teknik ini berfokus pada perlindungan hak cipta dengan menyisipkan informasi hak cipta pada media lain dan memberikan ijin kepada pihak ketiga untuk mengetahui keberadaan informasi yang disisipkan tersebut. Hal ini berbeda dengan steganografi yang menjaga informasi yang disisipkan pada media lain agar tidak terlihat oleh pihak ketiga. Jika ada pihak ketiga yang ingin menghack isi informasi tersembunyi tersebut, maka tujuan mereka adalah berbeda [10].

* + 1. **Kriteria Steganografi**

Penilaian sebuah metode atau algoritma steganografi yang baik dapat dinilai dari beberapa faktor berikut ini [2]:

1. Tidak dapat dipersepsi (*Impercepbility*) Keberadaan data rahasia tidak dapat dipersepsi oleh indera manusia. Jika pesan disisipkan ke dalam sebuah citra, citra yang telah disisipi pesan harus tidak dapat dibedakan dengan citra asli saat dilihat dengan mata. Begitu pula dengan suara, telinga haruslah mendapati tidak ada perbedaan antara suara asli dan suara yang telah disisipi.

2. Ketepatan (*Fidelity*) Kualitas citra penampung tidak jauh berubah setelah penyisipan data rahasia. Pengamat tidak mengetahui kalau di dalam citra tersebut terdapat data rahasia.

3. Kapasitas (*Capacity*) Berhubungan dengan jumlah informasi yang dapat disisipkan ke dalam media penampung.

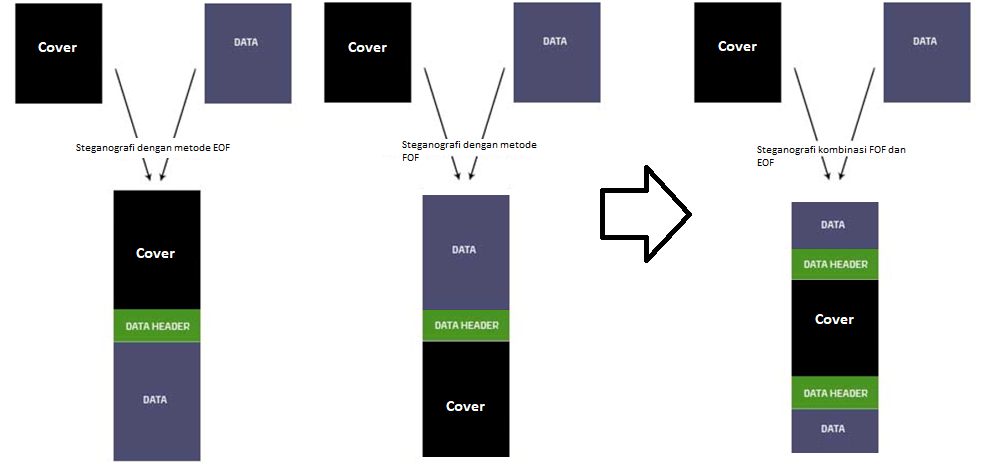
5. Tidak terdeteksi (*Undetecability*) Kemampuan untuk menghindari deteksi oleh indera manusia maupun analisis statistik.

6. Pemulihan (*Recovery*) Data yang disembunyikan harus dapat diungkapkan kembali (*reveal*).

* + 1. **Metode *First of File* dan metode *End of File***

Metode ini merupakan metode pengembangan LSB. Dalam metode ini pesan disisipkan diakhir berkas. Teknik E*OF* atau *End of File* merupakan salah satu teknik yang digunakan dalam steganografi. Teknik ini menggunakan cara dengan menyisipkan data pada akhir *file*. Teknik ini dapat digunakan untuk menyisipkan data yang ukurannya sesuai dengan kebutuhan [11]. Ukuran *file* yang telah disisipkan data sama dengan ukuran *file* sebelum disisipkan data ditambah dengan ukuran data yang disisipkan ke dalam *file* tersebut.

Dengan metode E*OF*, secara umum media steganografi (*file* yang akan disisipi data) memiliki struktur seperti Gambar 2.2:

****

Gambar 2.2 Struktur File Steganografi dengan Metode EOf, FOF dan Kombinasi EOF dan FOF [1]

Pada penelitian ini diterapkan steganografi *First of File*, *End of File* serta kombinasi *First of File* dan *End of File*. Pada *First of File* data akan disisipkan seluruhnya di awal dari *cover*. Pada metode *End of File* data akan disisipkan seluruhnya di akhir *file*. Pada metode kombinasi *First of File* dan *End of File* data akan disisipkan sebanyak 1000 data genap di awal dari *file* serta menyisipkan 1000 data ganjil dan sisa data lainnya di akhir *file.*

* + 1. **Perhitungan MSE dan PSNR**

Kualitas media penampung setelah ditambahkan pesan rahasia tidak jauh berbeda dengan kualitas media penampung sebelum ditambahkan pesan. Setelah penambahan pesan rahasia, kualitas citra penampung tidak jauh berubah, masih terlihat dengan baik. Sehingga pengamat tidak mengetahui kalau di dalam citra tersebut terdapat pesan rahasia. Untuk mengukur kualitas citra steganografi diperlukan suatu pengujian secara obyektif. Pengujian secara objektif adalah dilakukan dengan menghitung nilai PSNR. *Peak Signal to Noise Ratio* (PSNR) adalah perbandingan antara nilai maksimum dari sinyal yang diukur dengan besarnya derau yang berpengaruh pada sinyal tersebut. PSNR diukur dalam satuan desibel. Pada penelitian ini, PSNR digunakan untuk mengetahui perbandingan kualitas citra *cover* sebelum dan sesudah disisipkan pesan. Untuk menentukan PSNR, terlebih dahulu harus ditentukan MSE (*Mean Square Error*). MSE adalah nilai eror kuadrat rata-rata antara citra *cover* dengan citra tersteganografi, secara matematis dapat dirumuskan sebagai berikut:

Di mana:

* MSE = Nilai *Mean Square Error* citra steganografi
* M = Panjang citra stego (dalam *pixel*)
* N = Lebar citra stego (dalam *pixel*)
* I (x,y) = nilai *pixel* dari citra *cover*
* I’(x,y) = nilai *pixel* pada citra stego

Setelah diperoleh nilai MSE maka nilai PSNR dapat dihitung dari kuadrat nilai maksimum dibagi dengan MSE. Secara matematis, nilai PSNR dirumuskan sebagai berikut:

Di mana:

MSE = nilai perhitungan MSE

MAXi = nilai maksimum dari *pixel* citra yang digunakan

Semakin rendah nilai MSE maka akan semakin baik, dan semakin besar nilai PSNR maka semakin baik kualitas citra steganografi [12].

Contoh perhitungan PSNR dan MSE:

Terdapat sebuah citra *grayscale* 3 x 3 dengan nilai pixel yaitu:

I =

Setelah mendapatkan suatu operasi pengolahan citra didapatkan pixel citra baru yaitu:

I’ =

(2-2)

(2-3)

Berdasarkan pada penelitian Khan dkk pada tahun 2015 ambang batas nilai PSNR adalah 30 dB, artinya hasil perhitungan di atas menunjukkan kualitas citra sebelum dan sesudah dilakukan pengolahan citra tidak jauh berbeda.

**BAB III**

**METODE PENELITIAN**

1. **Bahan dan Alat Penelitian**

Bahan dan alat yang digunakan dalam penelitian ini yaitu:

1. Bahan penelitian

Bahan yang digunakan dalam proses penelitian ini bersumber dari artikel ilmiah, situs resmi, buku-buku yang terkait dengan steganografi dan kompresi data sebagai pendukung.

1. Alat Penelitian

Alat yang digunakan dalam proses penelitian ini terbagi menjadi dua bagian yaitu:

1. Perangkat keras

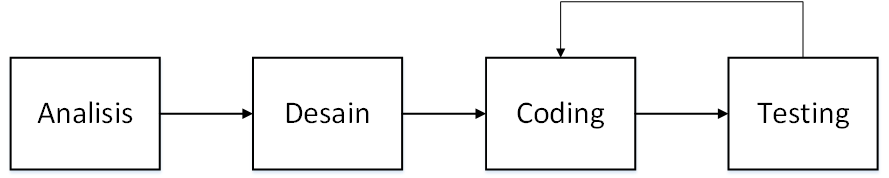
Perangkat keras yang digunakan dalam penelitian ini adalah *notebook* dengan spesifikasi sebagai berikut:

1. *Processor* Intel® Core™ i7-4720HQ 3.60GHz
2. Memori RAM 4 GB
3. Perangkat lunak

Perangkat lunak yang digunakan dalam penelitian ini, yaitu:

1. Sistem operasi Windows
2. Java
3. NetBeans IDE 8.0.2
4. **Jalan Penelitian**

Metode yang akan digunakan dalam penelitian ini adalah metode *End of File, First of File* dengan melakukan kompresi data terhadap pesan yang disisipkan. Pada steganografi terdapat dua proses utama yaitu *embedding* (penanaman pesan rahasia) dan ekstraksi. Pada penelitian ini akan dibangun sebuah perangkat lunak steganografi dengan menggunakan model *waterfall* pada Gambar 3.1, yang merupakan sebuah metode pengembangan perangkat lunak yang bersifat sekuensial.



Gambar 3.1Model *Waterfall.*

1. Analisis kebutuhan (*analysis*)

Analisis kebutuhan merupakan tahapan untuk menganalisa kebutuhan-kebutuhan yang diperlukan untuk melakukan pengembangan perangkat lunak yang akan dibuat, sehingga perangkat lunak yang dikembangkan sesuai dengan tujuan pembuatan perangkat lunak.

1. Perancangan sistem (*design*)

Proses desain sistem merupakan tahapan untuk mendesain atau merancang system, seperti merancang *interface* *system*, merancang alur kerja dan merancang algoritma.

1. Pengkodean (*coding*)

Tahap ini adalah tahap implementasi dari desain yang telah dibuat. Menerapkan modul-modul yang telah dirancang ke dalam bahasa pemrograman agar desain perangkat lunak dapat menjadi sebuah sistem yang utuh.

1. Pengujian program (*test*)

Pada tahapan ini program yang telah dibuat akan diuji coba dan dianalisa kembali untuk memeriksa kesesuaian pangkat lunak dengan persyaratan yang ada. Jika pada pengujian aplikasi belum memenuhi *requirements* yang sudah ditentukan pada analisa dan perancangan maka pengerjaan akan kembali ketahapan coding untuk menyesuaikan dengan persyaratan yang ada hingga aplikasi memenuhi semua requirements yang ditentukan.

1. **Kebutuhan Sistem**

Dalam penelitian ini analisis kebutuhan sistem dibagi menjadi tiga, yaitu analisis pengguna, analisis perangkat keras, dan analisis perangkat lunak.

1. **Analisis pengguna**

Pengguna yang akan menggunakan aplikasi steganografi ini dibagi menjadi dua bagian, yaitu pengirim dan penerima dengan kebutuhan spesifikasi seperti pada Tabel 3.1.

Tabel 3.1 Kebutuhan spesifikasi pengguna**.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tipe Pengguna** | **Hak Akses** | **Tingkat Keterampilan** | **Jenis Pelatihan** |
| Pengirim | Dapat melakukan penyisipan pesan ke dalam *file* cover. | Dapat mengoperasikan komputer dan memahami konsep steganografi. | Tidak diperlukan pelatihan khusus. |
| Penerima | Dapat melakukan ekstraksi pesan dari *file* cover. | Dapat mengoperasikan komputer dan memahami konsep steganografi. | Tidak diperlukan pelatihan khusus. |

1. **Analisis perangkat keras**

Perangkat keras merupakan salah satu kebutuhan penting dalam pembuatan steganografi. Perangkat keras yang digunakan akan mempengaruhi kinerja dari pembuatan steganografi, semakin tinggi spesifikasi pangkat keras yang digunakan maka akan semakin cepat pula pembuatan steganografi dilakukan. Pada penelitian ini perangkat keras yang digunakan dalam pembangunan dan menguji aplikasi steganografi, yaitu seperti pada Tabel 3.2.

Tabel 3.2 Kebutuhan perangkat keras untuk membangun dan menguji aplikasi**.**

|  |  |  |
| --- | --- | --- |
| No | Nama Perangkat | Spesifikasi |
| 1 | Prosessor | Intel® Core™ i7-4720HQ 3.6 GHz |
| 2 | Memori | RAM 4 GB DDR3 |

1. **Analisis perangkat lunak**

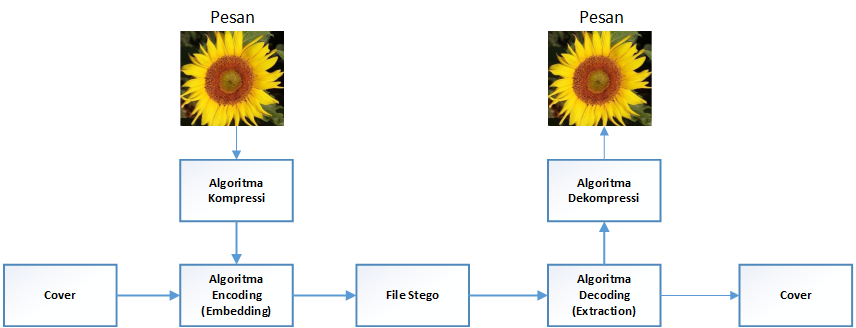
Perangkat lunak yang akan digunakan dalam membangun dan menguji aplikasi steganografi ini yaitu seperti pada Tabel 3.3.

Tabel 3.3 Kebutuhan perangkat lunak untuk membangun dan menguji aplikasi**.**

|  |  |  |
| --- | --- | --- |
| No | Nama Perangkat | Spesifikasi |
| 1 | Sistem Operasi | Windows 10 |
| 2 | JDK (*Java Development Kit*) | JDK v. 8 *update* 60 (build 1.8.0) |
| 3 | IDE | NetBeans IDE 8.0.2 |
| 4 | Microsoft Office | Office 2013 |
| 5 | Photo viewer | Windows Phot Viewer |
| 6 | PDF Reader | Nitro Pro 10 |

1. **Rancangan Algoritma**

Aplikasi ini secara garis besar memiliki dua proses utama yaitu penyisipan (*embedding*) pesan dan ekstraksi (*extraction*) pesan yang tersembunyi. Hubungan antara kedua proses tersebut dapat dilihat pada Gambar 3.2.



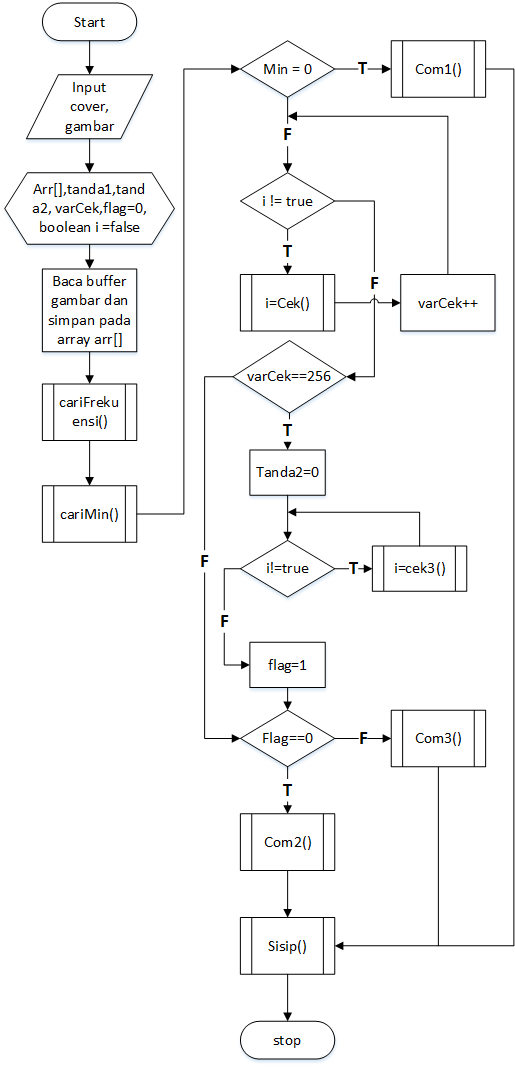
Gambar 3.2 Proses penyisipan dan ekstraksi pesan.

Proses penyisipan pesan dimulai dengan memilih *file* gambar yang akan menjadi pesan rahasia dan dilakukan kompresi untuk mengurangi ukuran dari gambar. Selanjutnya memilih *file* yang menjadi *cover* sebagai tempat penyisipan pesan. Langkah selanjutnya yaitu penyisipan pesan rahasia pada *file cover* (*Embedding*). Berikutnya yaitu ekstraksi pesan rahasia yang dimulai dengan memilih *file* stego. Kemudian mendapatkan kembali informasi pesan rahasia dengan melakukan ekstraksi sesuai dengan algoritma*.* Data hasil ekstraksi merupakan data yang masih terkompresi dan harus dilakukan dekompresi lagi untuk mendapatkan pesan asli.

Berikut algoritma beserta dengan diagram alir (*Flow Chart*) yang akan menjelaskan proses penyisipan pesan rahasia sampai menghasilkan *file stego* dan proses ekstraksi pesan rahasia yang telah disisipkan pada *file stego* sampai mendapatkan kembali pesan rahasia*.*

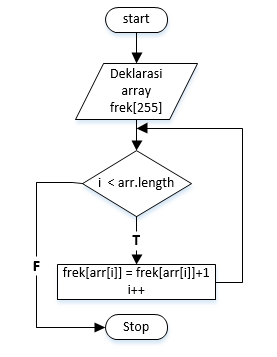
* 1. **Proses penyisipan pesan**

Berikut merupakan *flowchart* dalam tahap untuk menyisipkan pesan rahasia pada gambar, dapat dilihat pada Gambar 3.3.



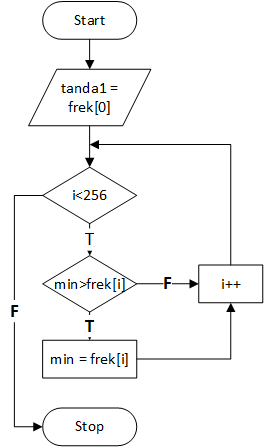
Gambar 3.3 *Flowchart Embedding* pesan.

*Flowchart* sub-proses mencari frekuensi digunakan untuk mencari frekuensi kemunculan *byte* 0-255 yang ada pada *file* gambar, ini digunakan untuk mencari penanda dari kompresi, pemilihan penanda dengan mengambil *byte* yang memiliki frekuensi terkecil pada pesan gambar, *flowchart* dapat dilihat pada Gambar 3.4.

****

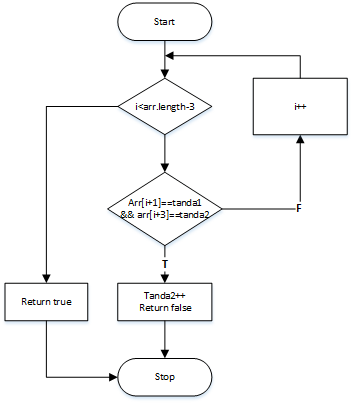
Gambar 3.4 *Flowchart* sub proses cari Frekuensi.

*Flowchart* sub-proses cari minimum ini adalah proses mencari nilai frekuensi terkecil dari *byte*-*byte* data yang akan digunakan sebagai penanda kompresi maupun flag pada penyisipan pesan seperti yang diperlihatkan pada Gambar 3.5.

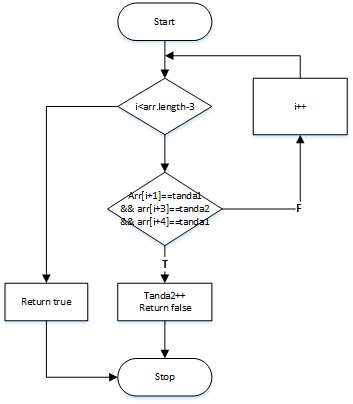


Gambar 3.5 *Flowchart* sub proses cari Min.

Sub-proses cek dan cek3 digunakan untuk melakukan pengecekan kombinasi tanda 1 dan tanda 2 ada pada *byte*-*byte* pesan atau tidak, jika subproses mendapatkan suatu kombinasi yang tidak ada pada *file* gambar maka akan diberikan pengembalian bernilai *true,* seperti yang terlihat pada flowchart Gambar 3.6.

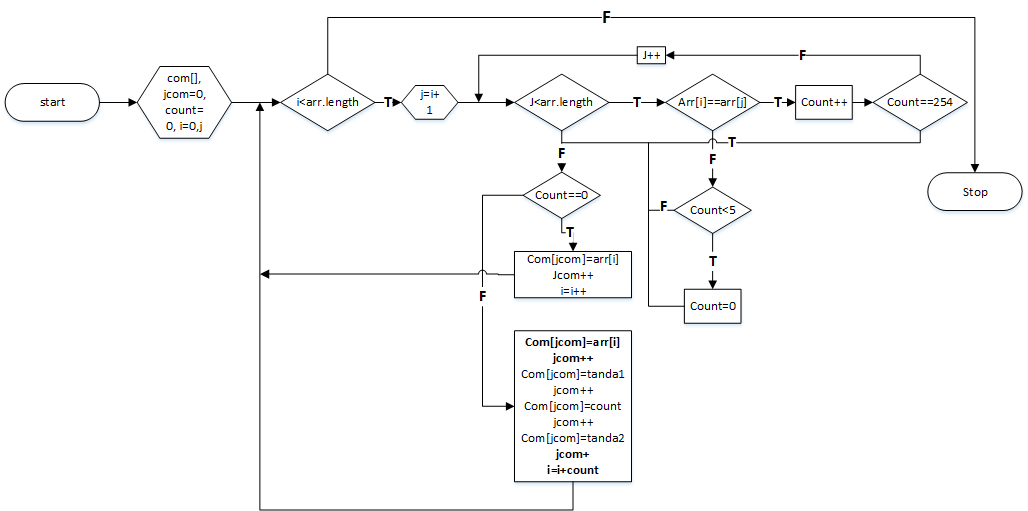


Gambar 3.6 *Flowchart* sub proses cek.

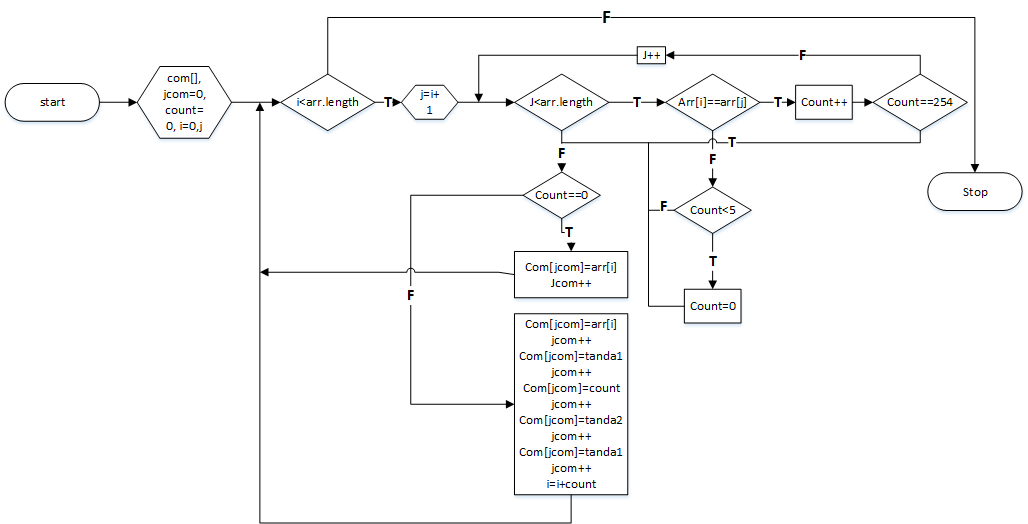


Gambar 3.7 *Flowchart* sub proses cari cek3.

Sub-proses com1, com2 dan com3 adalah subproses untuk melakukan kompresi data pesan dengan menggunakan algoritma RLE, seperti yang terlihat pada *flowhcart* Gambar 3.8 dan Gambar 3.9.

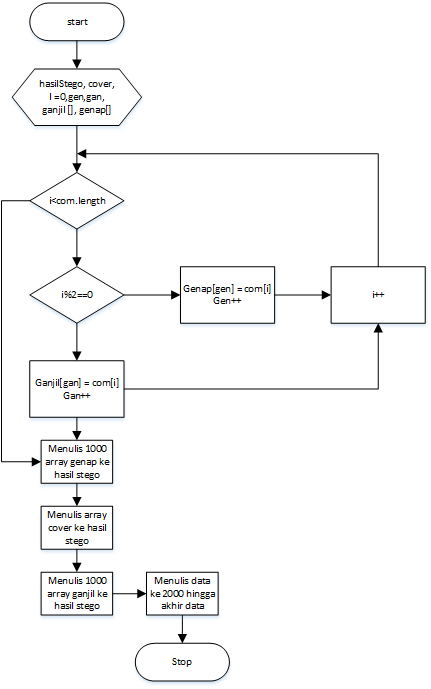


Gambar 3.8 *Flowchart* sub proses com2.



Gambar 3.9 *Flowchart* sub proses com3.

Sub-proses sisip digunakan untuk melakukan penyisipin *byte* data hasil kompresi kedalam *file* cover menggunakan kombinasi *End of File* dan *First of File*, seperti yang terlihat pada flowchart Gambar 3.10.



Gambar 3.10 *Flowchart* sub proses sisip.

Berikut algoritma mengkompresi dan menyisipkan *file* gambar kedalam suatu *file*:

* + - 1. Langkah awal yang dilakukan yaitu input *file* gambar yang menjadi pesan rahasia dan *file cover*.
      2. Baca *byte* *file* gambar, lalu hitung frekuensi kemunculan setiap *byte* dari 0 – 255.

Misalkan diketahui sebuah *file* gambar berformat PNG memiliki *byte* sebagai berikut:

[ 2 2 5 6 9 2 6 8 8 8 8 8 8 8 8 8 99 7 7 55 9 7 6 231 55 56 7 100 123 45 22 7 46 2 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 8 9]

* + - 1. Dapatkan frekuensi terkecil yang akan digunakan sebagai penanda hasil kompresi.

Berdasarkan *byte*-*byte* di atas didapatkan sebuah tabel frekuensi pada Tabel 3.4.

Tabel 3.4 Frekuensi *byte*

|  |  |
| --- | --- |
| *Byte* | Frekuensi |
| 0 | 21 |
| 1 | 3 |
| 2 | 4 |
| 5 | 1 |
| 6 | 3 |

Tabel 3.4 Lanjutan

|  |  |
| --- | --- |
| *Byte* | Frekuensi |
| 7 | 5 |
| 8 | 10 |
| 9 | 3 |
| 22 | 1 |
| 45 | 1 |
| 46 | 1 |
| 55 | 2 |
| 56 | 1 |
| 99 | 1 |
| 100 | 1 |
| 123 | 1 |
| 231 | 1 |
| Total | 60 |

* + - 1. Kemudian mencari kombinasi yang tidak ada pada *byte* data gambar antara *byte* yang memiliki frekuensi terkecil dengan *byte* 0 – 255.

Contoh pada poin 2 terdapat *byte* yang berfrekuensi 0 yang artinya tidak ada pada *file* pesan oleh karenanya *byte* berfrekuensi 0 dapat dijadikan penanda hasil kompresi. Pada contoh pada poin 2 penanda yang digunakan adalah 3.

* + - 1. Setelah itu lakukan kompresi RLE dengan mencari *byte*-*byte* data yang memiliki keseragaman lebih dari 5 akan dimampatkan seperti contoh dibawah ini:

[ 2 2 5 6 9 5 6 8 8 8 8 8 8 8 8 8 99 7 7 55 9 7 6 231 55 56 7 100 123 45 22 7 46 5 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 8 9]

Hasil kompresi:

[ 2 2 5 6 9 5 6 8 3 9 99 7 7 55 9 7 6 231 55 56 7 100 123 45 22 7 46 5 1 1 1 0 3 21 8 9]

* + - 1. Setelah data hasil kompresi didapatkan maka dilakukan penulisan pada *file* cover *byte*-*byte* hasil kompresi, pada kasus ini penyisipan dilakukan menggunakan kombinasi *First of File* dan *End of File* dengan aturan semua index genap akan disisipkan pada *First of File* cover dan index ganjil akan disisipkan pada *End of File* cover. Misalkan *byte*-*byte* cover sebagai berikut:

[ 5 6 2 3 7 8 9 2 4 5 7 1 3 3 7 8 3 4 8 9 0 0 2 3 5 1 7 8 9 5 6 8 0 9 7 6 3 2 3 1 2 3 5 7 8 1 1 3 4 2 7 8 0 9 7 8 2 3]

Hasil penyisipan:

[ 3 0 2 5 9 6 3 99 7 9 6 55 7 123 22 46 1 1 3 8 112 110 103 3 3 3 5 6 2 3 7 8 9 2 4 5 7 1 3 3 7 8 3 4 8 9 0 0 2 3 5 1 7 8 9 5 6 8 0 9 7 6 3 2 3 1 2 3 5 7 8 1 1 3 4 2 7 8 0 9 7 8 2 3 3 3 3 2 6 5 8 9 7 55 7 231 56 100 45 7 5 1 0 21 9 0 1]

Angka berwarna hijau diawal *byte* adalah penanda hasil kompresi, angka berwarna biru adalah penanda atau flag batas dari pesan dan cover dan angka hijau sebelum [3 3 3] adalah type *file* pesan yang disisipkan dalam ASCII [112 110 103] yang artinya PNG. Angka [1] pada akhir *file* adalah penanda pesan disisipkan dengan menggunakan kombinasi *First of File* dan *End of File.* Angka [0] pada length-2 adalah penanda pesan dikompresi dengan 1 penanda.

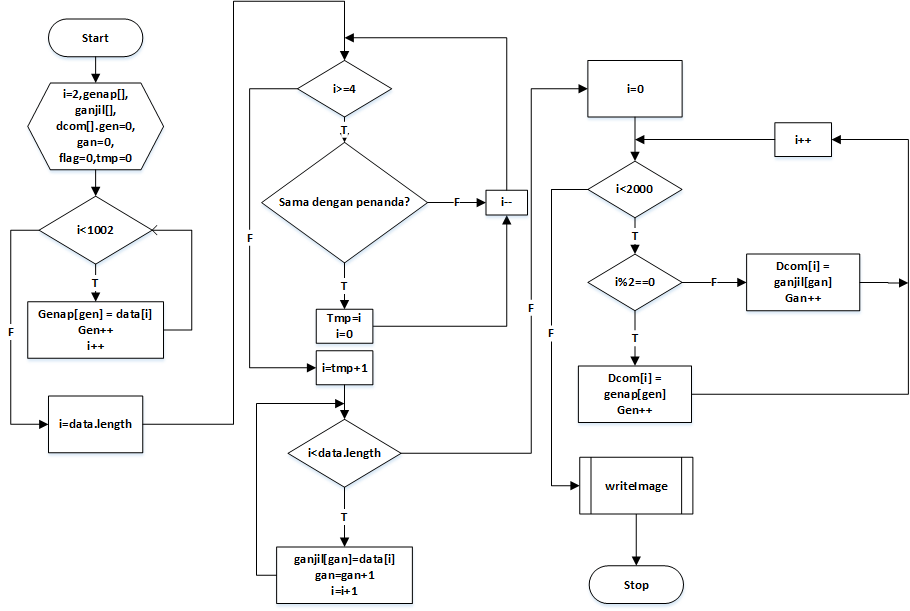
* + - 1. Selesai *file* gambar berhasil disisipkan pada *file* cover.
  1. **Proses ekstraksi pesan**

Flowchart proses ekstraksi pesan yang telah disisipkan pada *file* dapat dilihat pada Gambar 3.11.



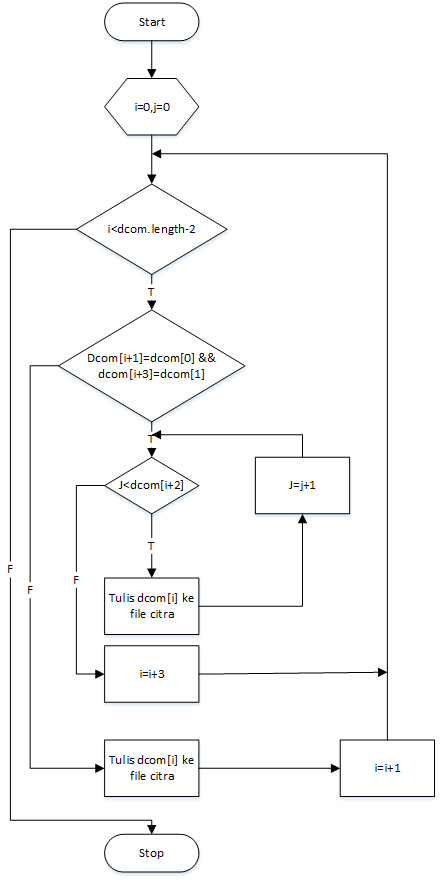
Gambar 3.11 *Flowchart* ekstraksi pesan gambar.

Sub-proses dekom2, dekom3 dan dekom1 digunakan untuk melakukan ekstraksi pesan dari *file* stego. Pada sub-proses ini data-data yang sudah disisipkan diawal dan di akhir *file* diambil kemudian dilakukan dekompresi dan ditulis ke *file* gambar hasil sehingga didapatkan pesan gambar yang sudah disisipkan, seperti yang terlihat pada *flowchart* Gambar 3.12.



Gambar 3.12 *Flowchart* sub proses dekom2, dekom3 dan dekom1.

Sub-proses *writeimage* adalah sub-proses yang berguna untuk menyatukan kembali *byte*-*byte* data ganjil dan genap pada satu *file* gambar sehingga didapatkan satu *file* pesan secara utuh, proses dapat dilihat pada *flowchart* Gambar 3.13.



Gambar 3.13 *Flowchart* sub proses cretaeimage.

Berikut algoritma mendekompresi dan mendapatkan pesan gambar:

* + - 1. Langkah awal yang dilakukan yaitu input *file stego* yang sudah disisipkan pesan
      2. Baca *byte* *file* stego.

Misalkan diketahui sebuah *file* stego memiliki *byte* sebagai berikut:

[ 3 0 2 5 9 6 3 99 7 9 6 55 7 123 22 46 1 1 3 8 112 110 103 3 3 3 5 6 2 3 7 8 9 2 4 5 7 1 3 3 7 8 3 4 8 9 0 0 2 3 5 1 7 8 9 5 6 8 0 9 7 6 3 2 3 1 2 3 5 7 8 1 1 3 4 2 7 8 0 9 7 8 2 3 3 3 3 2 6 5 8 9 7 55 7 231 56 100 45 7 5 1 0 21 9 0 1]

* + - 1. *Byte* terakhir data adalah penanda pesan disisipkan dengan metode *First of File*, *End of File* atau kombinasi keduanya, pada permasalahan *byte* bernilai [1] artinya pesan disisipkan dengan metode kombinasi *First of File* dan *End of File*
      2. *Byte* ke-0 dan ke-1 adalah merupakan penanda dan bukan merupakan bagian dari *byte* pesan. *Byte*-*byte* awal dimasukan kedalam array *byte* genap dimulai dari data ke-2 hingga 3 *byte* terakhir sebelum ditemukan data dengan kombinasi [3 3 3] secara berurutan.

[ 3 0 2 5 9 6 3 99 7 9 6 55 7 123 22 46 1 1 3 8 112 110 103 3 3 3 5 6 2 3 7 8 9 2 4 5 7 1 3 3 7 8 3 4 8 9 0 0 2 3 5 1 7 8 9 5 6 8 0 9 7 6 3 2 3 1 2 3 5 7 8 1 1 3 4 2 7 8 0 9 7 8 2 3 3 3 3 2 6 5 8 9 7 55 7 231 56 100 45 7 5 1 0 21 9 0 1]

Maka didapatkan *byte* genap adalah:

* + - 1. Melanjutkan mencari data [3 3 3] untuk mendapatkan *byte* ganjil dari pesan, pencarian dilakukan dimulai dari belakang, jika ditemukan maka *byte* setelahnya hingga *byte* akhir merupakan *byte* ganjil.

[ 3 0 2 5 9 6 3 99 7 9 6 55 7 123 22 46 1 1 3 8 112 110 103 3 3 3 5 6 2 3 7 8 9 2 4 5 7 1 3 3 7 8 3 4 8 9 0 0 2 3 5 1 7 8 9 5 6 8 0 9 7 6 3 2 3 1 2 3 5 7 8 1 1 3 4 2 7 8 0 9 7 8 2 3 3 3 3 2 6 5 8 9 7 55 7 231 56 100 45 7 5 1 0 21 9 0 1]

Maka didapatkan *byte* ganjil adalah:

* + - 1. *Byte* genap dan *byte* ganjil ditulis disimpan pada satu array secara bergantian dimulai dari *byte* genap hingga akhir. Pada tahap ini sudah didapatkan *byte* *file* pesan gambar yang masih terkompresi.

[ 2 2 5 6 9 5 6 8 3 9 99 7 7 55 9 7 6 231 55 56 7 100 123 45 22 7 46 5 1 1 1 0 3 21 8 9]

* + - 1. Maka dilakukan dekompresi *byte*-*byte* yang tersimpan di array dengan mencari penanda 3.
      2. Pada *byte* length-2 adalah bernilai 0 artinya pesan terkompresi dengan menggunakan 1 penanda. Maka dilakukan pencarian *byte* terkompress dengan memperhitungkan 1 penanda data yaitu data ke-0 [3] menjadi:

[ 2 2 5 6 9 5 6 8 3 9 99 7 7 55 9 7 6 231 55 56 7 100 123 45 22 7 46 5 1 1 1 0 3 21 8 9]

* + - 1. Dekompresi dilakukan dengan menulis kembali *byte*-*byte* sebelum penanda sebanyak angka yang ada diantara tepat setelah angka 3, misalkan pada contoh [8 3 9] maka hasil dekompresi adalah [8 8 8 8 8 8 8 8 8] 🡪 8 ditulis sebanyak 9 kali. Langkah 7 berlaku pada *byte*-*byte* selanjutnya hingga akhir data.

[ 2 2 5 6 9 5 6 8 8 8 8 8 8 8 8 8 99 7 7 55 9 7 6 231 55 56 7 100 123 45 22 7 46 5 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 8 9]

* + - 1. *Byte*-*byte* hasil dekompresi ditulis pada sebuah *file* dan disimpan sesuai dengan type *file* yang sudah didapatkan dari *file* stego. Pada contoh type *file* adalah PNG dengan nilai ASCII 112 110 103.
      2. Selesai *file* pesan berhasil didapatkan

1. **Metode Pengujian**

Proses pengujian *file* stego dari aplikasi steganografi pada penelitian ini menggunakan kriteria steganografi secara umum sebagai acuan dasar pengujian. Kriteria yang digunakan antara lain:

1. *Impercepbility,* adalah kriteria yang mengukur kemampuan *file* stego untuk menyembunyikan pesan rahasia tanpa dapat dirasakan secara indrawi. Pengujian terhadap kriteria ini dapat dilakukan dengan pemeriksaan secara kasatmata menggunakan aplikasi pengolah kata ataupun hasil cetak dari *file* stego. Pengguna dapat melakukan pemeriksaan terhadap *file* stego dan mencari seberapa banyak perubahan yang dapat mengindikasikan keberadaan pesan rahasia. Pengujian ini akan dilakukan dengan membagikan kuisioner untuk dijawab responden setelah melihat *file* stego yang sudah disisipkan pesan.
2. *Fidelity,* adalah kriteria yang mengukur mutu dari media penampung yang digunakan setelah disisipkan pesan. Pengujian kriteria ini dapat dilakukan dengan cara memeriksa seberapa besar perubahan yang dialami ketika *file* *cover* disisipkan pesan. Pengguna dapat membandingkan *file* stego dengan *file* *cover* sebelum disisipkan pesan atau dengan memeriksa perubahan yang terjadi pada *properties* *file* stego seperti membandingkan ukuran *file* stego dengan *file* asli.
3. *Recovery,* adalah kriteria yang mengukur kemampuan algoritma steganografi dalam mengungkap kembali pesan yang disembunyikan. Tujuan steganografi adalah menyembunyikan informasi, maka sewaktu-waktu informasi yang disembunyikan ini harus dapat diambil kembali untuk dapat digunakan lebih lanjut sesuai keperluan. Pengujian kriteria ini juga dapat dilakukan dengan menggunakan metode perbandingan nilai PSNR *(Peak Signal to Noise Ratio)* dari *file* pesan sebelum disisipkan dan setelah diekstraksi dari *file* cover.
4. Rasio kompresi, adalah kriteria yang mengukur besaran data yang terkompresi dari ukuran asli data. Pengujian dilakukan dengan menghitung rasio pada *file* yang menggunakan kompresi *Run Length Encoding* dan *file* yang menggunakan kompresi *Run Length Encoding* termodifikasi, kemudian membandingkan hasil rasio keduanya pada beberapa *file*. Pada pengujian ini akan dilakukan penyisipan 11 gambar berbeda pada *file* cover dengan menerapkan algoritma kompresi RLE dan RLE modifikasi, kemudian dilihat perbedaan rasio pada 11 gambar.

* Terdapat beberapa skenario dalam pengujian aplikasi yaitu :
  + - 1. Gambar akan disisipkan pada *file* dokumen bertipe pdf, docx, pptx dan xlsx.
      2. Gambar akan disisipkan pada *file* gambar bertipe JPG, PNG dan BMP.

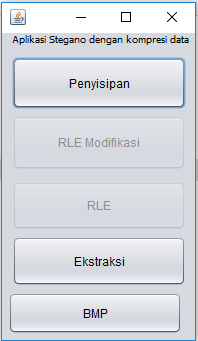
Setelah melakukan penyisipan, *file* cover akan dievaluasi berdasarkan kriteria pengujian steganografi dan kompresi yaitu *Impercepbility, Fidelity, Recovery* dan Rasio kompresi*.*

**BAB IV**

**HASIL DAN PEMBAHASAN**

1. **Aplikasi Penyisipan dan Ekstraksi Pesan**

Gambar 4.1 merupakan tampilan aplikasi yang dikembangkan untuk melakukan proses penyisipan dan ekstraksi pesan.

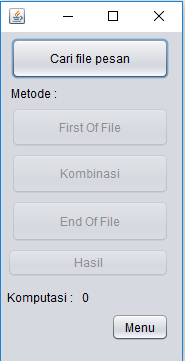


Gambar 4.1Tampilan menu utama.

Gambar 4.1 merupakan menu utama dari aplikasi. Terdapat 3 tombol aktif yaitu Penyisipan, Ekstraksi dan BMP. Tombol penyisipan digunakan untuk melakukan penyisipan pada cover dengan terlebih dahulu memilih metode kompresi yang ingin digunakan yaitu RLE atau RLE dengan modifikasi. Tombol ekstraksi digunakan untuk mendapatkan kembali pesan yang sudah disisipkan pada sebuah *file* stego. Tombol BMP digunakan untuk melakukan steganografi dengan pesan berformat BMP.

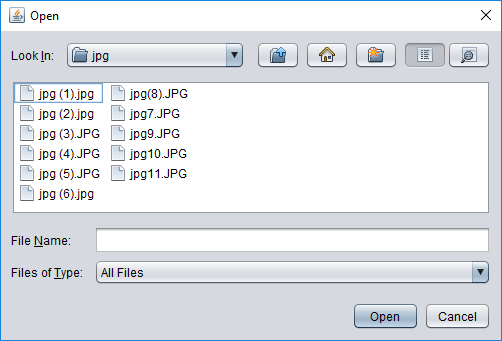
1. **Penyisipan pesan**

Sebelum melakukan penyisipan pesan pengguna terlebih dahulu harus menyiapkan *file* *cover* yang bisa berupa *file* DOCX, PPTX, XLSX, PDF, JPG atau PNG dan menyiapkan pesan gambar yang akan disisipkan. Gambar 4.2 – Gambar 4.5 merupakan langkah-langkah proses penyisipan pesan.

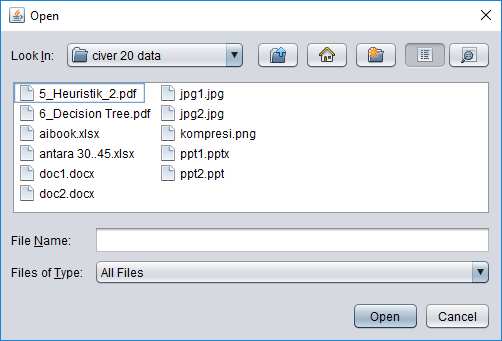
****

Gambar 4.2 Tampilan proses penyisipan pesan.

Gambar 4.2 merupakan tampilan yang akan muncul ketika pengguna sudah memilih jenis kompresi data yang akan digunakan. Tombol yang aktif pertama kali adalah tombol cari pesan yang digunakan untuk mencari pesan yang akan disisipkan pada *file* cover seperti yang terlihat pada Gambar 4.3. Setelah memilih pesan kemudian memilih metode penyisipan pesan kedalam *file* cover, terdapat 3 metode yaitu *First of File*, *End of File* atau kombinasi kedua metode. Setelah memilih metode kemudian akan diarahkan pada window untuk memilih *file* cover yang akan disisipkan pesan seperti yang terlihat pada Gambar 4.4.

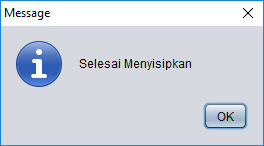


Gambar 4.3 Tampilan proses pemilihan *file* *pesan*.



Gambar 4.4 Tampilan proses pemilihan *file* *cover*.

Setelah pengguna memilih cover yang akan disisipkan pesan, aplikasi akan melakukan komputasi untuk mengkompres pesan sesuai dengan metode kompresi yang sudah dipilih pada menu awal Gambar 4.1 dan melakukan penyisipan sesuai dengan metode yang dipilih pada menu penyisipan Gambar 4.2. Setelah proses selesai maka akan muncul pesan *dialog* seperti Gambar 4.5

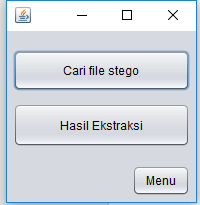


Gambar 4.5 Tampilan *message dialog* setelah berhasil menyisipkan pesan.

Hasil *file* stego yang sudah disisipkan pesan akan disimpan pada /stego/. Penyimpanan pesan akan disesuaikan dengan jenis cover yang digunakan.

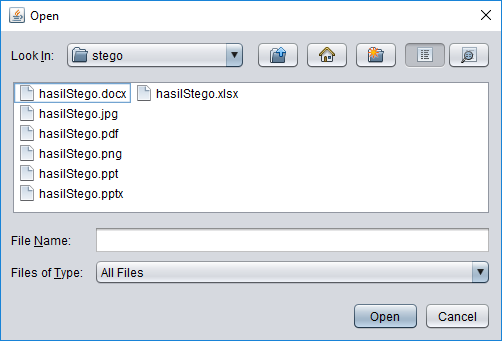
1. **Ekstraksi pesan**

Setelah melakukan proses penyisipan pesan, pengguna dapat melakukan proses ekstraksi pesan dengan menekan tombol “EKSTRAKSI”. Gambar 4.6 merupakan tampilan aplikasi setelah pengguna menekan tombol “EKSTRAKSI”.



Gambar 4.6 Tampilan Ekstraksi pesan.

Terdapat dua tombol aktif yaitu Cari *file* stego yang digunakan untuk melakukan pencarian *file* yang sudah disisipkan pesan sebelumnya seperti terlihat pada Gambar 4.7.



Gambar 4.7 Tampilan proses pemilihan *file* stego.

Setelah pengguna memilih *file* stego yang sudah disisipkan pesan aplikasi akan melakukan ekstraksi terhadap *file* tersebut sesuai dengan metode penyisipan dan metode kompresi yang digunakan pada *file* tersebut, hasil ekstraksi akan disimpan pada /pesan/.

1. **Pengujian**

Pengujian untuk membandingkan dua metode kompresi pada percobaan ini dilakukan dengan mengukur rasio kompresi yang dihasilkan pada berbagai jenis gambar pesan dengan beragam ukuran. Serta pengujian kualitas metode steganografi pada percobaan ini dilakukan dengan beberapa cara yaitu, menghitung nilai MSE dan PSNR *file* stego gambar dan *file* pesan setelah *recovery*.

Bahan yang digunakan untuk melakukan pengujian tersebut menggunakan 33 gambar pesan terdiri dari 11 bertipe JPG,11 bertipe PNG dan 11 bertipa BMP. Terdapat 11 jenis *file* cover yang digunakan bertipe DOCX, PPTX, XLSX, PDF, JPG dan PNG. Adapun daftar pesan gambar yang digunakan dalam percobaan ini dapat dilihat pada Tabel 4.1.

Tabel 4.1 Daftar pesan gambar yang digunakan dalam proses pengujian.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| Id: J1 | Id: J2 | Id: J3 | Id: J4 | Id: J5 | Id: J6 |
| Type: JPG | Type: JPG | Type: JPG | Type: JPG | Type: JPG | Type: JPG |
| Ukuran: 51 | Ukuran: 113 | Ukuran: 60 | Ukuran: 235 | Ukuran: 445 | Ukuran:710 |
|  |  |  |  |  |  |
| Id: J7 | Id: J8 | Id: J9 | Id: J10 | Id: J11 | Id: P1 |
| Type: JPG | Type: JPG | Type: JPG | Type: JPG | Type: JPG | Type: PNG |
| Ukuran: 1287 | Ukuran: 2100 | Ukuran: 2171 | Ukuran: 2486 | Ukuran: 5028 | Ukuran: 330 |
|  |  |  |  |  |  |
| Id: P2 | Id: P3 | Id: P4 | Id: P5 | Id: P6 | Id: P7 |
| Type: PNG | Type: PNG | Type: PNG | Type: PNG | Type: PNG | Type: PNG |
| Ukuran: 621 | Ukuran: 729 | Ukuran: 307 | Ukuran: 1246 | Ukuran: 1990 | Ukuran: 2066 |
|  |  |  |  |  |  |
| Id: P8 | Id: P9 | Id: P10 | Id: P11 | Id: B1 | Id: B2 |
| Type: PNG | Type: PNG | Type: PNG | Type: PNG | Type: BMP | Type: BMP |
| Ukuran: 2368 | Ukuran: 3059 | Ukuran: 3506 | Ukuran: 3925 | Ukuran: 676 | Ukuran: 801 |
|  |  |  |  |  |  |
| Id: B3 | Id: B4 | Id: B5 | Id: B6 | Id: B7 | Id: B8 |
| Type: BMP | Type: BMP | Type: BMP | Type: BMP | Type: BMP | Type: BMP |
| Ukuran: 1025 | Ukuran: 149 | Ukuran: 302 | Ukuran: 384 | Ukuran: 907 | Ukuran: 1250 |
|  |  |  |
| Id: B9 | Id: B10 | Id: B11 |
| Type: BMP | Type: BMP | Type: BMP |
| Ukuran: 2402 | Ukuran: 3597 | Ukuran: 4006 |

\*Ukuran dalam Kilobyte

Selain pesan gambar bahan lainnya yang dibutuhkan pada proses pengujian ini adalah *file* *cover.* Informasi mengenai *file* *cover* yang digunakan pada percobaan ini dapat dilihat pada Tabel 4.2.

Tabel 4.2 Daftar *file* *cover* dan ukuran yang digunakan dalam proses pengujian.

|  |  |  |  |
| --- | --- | --- | --- |
| No. | *File* *Cover* | Jenis *File* | Ukuran *File* (KB) |
| 1 | ppt1 | PPTX | 242 |
| 2 | ppt2 | PPTX | 3444 |
| 3 | jpg1 | JPG | 445 |
| 4 | jpg2 | JPG | 654 |
| 5 | doc1 | DOCX | 576 |
| 6 | doc2 | DOCX | 622 |
| 7 | xl1 | XLSX | 55 |
| 8 | xl2 | XLSX | 31 |
| 9 | pdf1 | PDF | 1221 |
| 10 | pdf2 | PDF | 1321 |
| 11 | png1 | PNG | 1636 |

1. **Pengujian Rasio Kompresi**

Pengujian Rasio kompresi bertujuan untuk mengetahui dan membandingkan metode kompresi RLE tanpa modifikasi dan RLE dengan modifikasi. Besarnya rasio kompresi merupakan salah satu indikator baik atau tidaknya sebuah algoritma kompresi yang digunakan.

Pengujian dilakukan dengan menyisipkan 33 pesan gambar pada 11 cover sesuai dengan Tabel 4.1 dan Tabel 4.2. Sebelum disisipkan gambar akan terlebih dahulu dikompresi dengan dua metode yaitu RLE modifikasi dan RLE tanpa modifikasi. Hasil pengujian dapat dilihat pada Tabel 4.3

Tabel 4.3 Hasil pengujian rasio kompresi

| No. | Pesan | Rasio RLE (%) | Rasio RLE Modifikasi (%) |
| --- | --- | --- | --- |
| 1 | B4 | 75,81 | 83,7300 |
| 2 | B9 | 43,85 | 63,0000 |
| 3 | B11 | -16,85 | 19,6300 |
| 4 | B10 | -3,57 | 16,4000 |
| 5 | B7 | -30,29 | 13,9900 |
| 6 | B8 | -44,71 | 3,5200 |
| 7 | B6 | -77,06 | 3,2500 |
| 8 | J7 | -95,00 | 1,3900 |
| 9 | B5 | -80,80 | 1,2100 |
| 10 | B1 | -75,11 | 1,0900 |
| 11 | J6 | -96,87 | 0,8860 |
| 12 | B2 | -87,41 | 0,8600 |
| 13 | B3 | -85,41 | 0,4600 |
| 14 | J5 | -97,99 | 0,4330 |
| 15 | J8 | -98,38 | 0,2850 |
| 16 | J9 | -98,38 | 0,2770 |
| 17 | J10 | -98,43 | 0,2440 |
| 18 | J11 | -98,55 | 0,1610 |
| 19 | J1 | -98,59 | 0,1300 |
| 20 | J2 | -98,92 | 0,0500 |
| 21 | J3 | -98,62 | 0,0400 |
| 22 | J4 | -99,07 | 0,0270 |
| 23 | P5 | -98,41 | 0,0053 |
| 24 | P8 | -98,70 | 0,0009 |
| 25 | P7 | -98,59 | 0,0008 |
| 26 | P10 | -98,95 | 0,0006 |
| 27 | P3 | -99,29 | 0,0006 |
| 28 | P11 | -97,67 | 0,0003 |
| 29 | P6 | -98,80 | 0,0002 |
| 30 | P9 | -98,48 | 0,0001 |
| 31 | P4 | -98,46 | 0,0001 |
| 32 | P1 | -98,01 | 0,0000 |
| 33 | P2 | -97,01 | 0,0000 |
| Rata-rata | | -71,03 | 6,39 |

Gambar 4.8 Grafik rasio kompresi JPG dan PNG

Gambar 4.9 Grafik rasio kompresi BMP

Berdasarkan data pada Tabel 4.3 rasio kompresi dengan menggunakan metode RLE modifikasi menunjukkan hasil yang lebih baik dari metode RLE tanpa modifikasi, ditunjukkan dengan sebanyak 22 data JPG dan PNG menghasilkan rasio negatif setelah dilakukan kompresi dengan RLE tanpa modifikasi yang artinya data hasil kompresi menjadi lebih besar dari ukuran data sebelum terkompresi. Sebaliknya dengan data yang sama dilakukan kompresi menggunakan RLE modifikasi menunjukkan 22 data memiliki rasio > 0 artinya data hasil kompresi lebih kecil dari data sebelum dilakukan kompresi.

Pengujian algoritma kompresi pada gambar berformat BMP menunjukkan hasil yang sangat baik, ini ditunjukkan dengan nilai rasio kompresi dengan menggunakan RLE modifikasi meningkat sangat tinggi pada gambar yang memiliki warna yang seragam begitu juga dengan kompresi dengan RLE tanpa modifikasi juga memiliki rasio yang tinggi, tetapi pada beberapa gambar yang memiliki kombinasi warna yang beragam seperti data uji dengan id B1, B3, B5, B6, dan B7, rasio kompresi RLE modifikasi terlihat lebih baik jika dibandingkan dengan RLE tanpa modifikasi pada gambar yang sama. Ini ditunjukkan dengan dengan semua data yang diuji menghasilkan rasio > 0, sebaliknya pada RLE tanpa modifikasi terdapat beberapa data yang menghasilkan rasio negatif yang artinya data hasil kompresi menjadi lebih besar dari ukuran data sebelum terkompresi.

Kompresi RLE dengan modifikasi menjadi lebih baik dari RLE tanpa modifikasi disebabkan oleh kompresi dengan modifikasi tidak ikut memperhitungkan data yang tidak berulang atau sama. Sedangkan RLE tanpa modifikasi ikut memperhitungkan data yang tidak berulang untuk dikompresi sehingga sangat memungkinkan jika terdapat sebuah data yang memiliki tingkat keragaman *byte/pixel* data sangat tinggi hasil kompresi yang didapatkan dengan RLE tanpa modifikasi menjadi lebih besar dari data sebelum dilakukan kompresi.

1. ***Fidelity***

Pengujian nilai PSNR (*Peak Signal to Noise Ratio*) dilakukan untuk mengetahui kualitas *file* stego. Proses pengujian ini akan menghasilkan nilai MSE (*Mean Square Error*) dan nilai PSNR dari sebuah *file* stego yang dihitung sesuai dengan rumus pada persamaan 2-2 dan 2-3. Semakin kecil nilai MSE yang dihasilkan maka semakin baik kualitas *file* stego. Pengujian nilai PSNR pada percobaan ini dilakukan terhadap *file* stego gambar dan *file* pesan hasil proses ekstraksi. Tabel 4.4 merupakan hasil pengujian nilai MSE dan PSNR dalam percobaan ini.

Tabel 4.4 Hasil pengujian nilai MSE dan PSNR

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No. | Pesan | Metode | Cover | MSE Pesan Recovery | PSNR Pesan Recovery | MSE Cover | | PSNR cover |
| 1 | J5 | *End of File* | JPG | 0 | Infinity | 0 | | Infinity |
| 2 | J6 | PNG | 0 | Infinity | 0 | | Infinity |
| 3 | J11 | JPG | 0 | Infinity | 0 | | Infinity |
| 4 | P5 | JPG | 0 | Infinity | 0 | | Infinity |
| 5 | P6 | PNG | 0 | Infinity | 0 | | Infinity |
| 6 | P11 | JPG | 0 | Infinity | | 0 | Infinity |
| 7 | B5 | JPG | 0 | Infinity | | 0 | Infinity |
| 8 | B6 | JPG | 0 | Infinity | | 0 | Infinity |
| 9 | B7 | PNG | 0 | Infinity | | 0 | Infinity |
| 10 | J1 | docx | 0 | Infinity | | NA | NA |
| 11 | J2 | pptx | 0 | Infinity | | NA | NA |
| 12 | J3 | xlxs | 0 | Infinity | | NA | NA |
| 13 | J7 | docx | 0 | Infinity | | NA | NA |
| 14 | J8 | pptx | 0 | Infinity | | NA | NA |
| 15 | J9 | xlxs | 0 | Infinity | | NA | NA |
| 16 | P1 | docx | 0 | Infinity | | NA | NA |
| 17 | P2 | pptx | 0 | Infinity | | NA | NA |
| 18 | P3 | xlxs | 0 | Infinity | | NA | NA |
| 19 | P7 | docx | 0 | Infinity | | NA | NA |
| 20 | P8 | pptx | 0 | Infinity | | NA | NA |
| 21 | P9 | xlxs | 0 | Infinity | | NA | NA |
| 22 | B1 | PDF | 0 | Infinity | | NA | NA |
| 23 | B2 | PDF | 0 | Infinity | | NA | NA |
| 24 | B3 | Docx | 0 | Infinity | | NA | NA |
| 25 | B4 | Docx | 0 | Infinity | | NA | NA |
| 26 | B8 | Pptx | 0 | Infinity | | NA | NA |
| 27 | B9 | pptx | 0 | Infinity | | NA | NA |
| 28 | B10 | xlsx | 0 | Infinity | | NA | NA |
| 29 | B11 | xlsx | 0 | Infinity | | NA | NA |
| 30 | J4 | Kombinasi | PDF | 0 | Infinity | | NA | NA |
| 31 | P4 | PDF | 0 | Infinity | | NA | NA |
| 32 | J10 | *First of File* | PDF | 0 | Infinity | | NA | NA |
| 33 | P10 | PDF | 0 | Infinity | | NA | NA |

Tabel 4.4 merupakan tabel hasil perhitungan nilai MSE dan PSNR pada *file* stego gambar dan *file* pesan hasil dari proses ekstraksi yang sudah melalui proses dekompresi. Pengujian pada *file* cover ini bertujuan untuk memeriksa kesamaan *file* cover sebelum dan setelah digunakan dalam metode steganografi ini. Selain itu pengujian ini juga digunakan untuk mengukur tingkat keberhasilan algoritma ekstraksi dan dekompresi yang digunakan. Jika hasil pengujian MSE semakin mendekati nol maka akan semakin baik. Terdapat beberapa data yang berlabel NA atau *Not* *Available* artinya pengujian MSE dan PSNR tidak dapat dilakukan karena bukan merupakan sebuah *file* gambar.

Berdasarkan hasil pengujian *file* stego gambar pada Tabel 4.4 dapat dilihat bahwa seluruh nilai MSE yang dihasilkan adalah 0, MSE dengan nilai 0 menyebabkan hasil perhitungan PSNR Infinity artinya data PSNR dari *file* tersebut tak hingga karena MSE seusai dengan persaman 2-3 menjadi pembagi pada proses perhitungan PSNR. MSE dengan nilai 0 menunjukkan bahwa *file* cover gambar sebelum dan sesudah disisipkan adalah identik. Pada *file* stego PDF, XLSX, DOCX, PPTX, JPG, dan PNG dengan menggunakan metode steganografi *First of File* memiliki kelemahan hanya dapat dibuka dengan beberapa reader. Kelemahan metode *First of File* ini disebabkan oleh penempatan data di awal dari suatu *file* dapat mengganggu isi *file* dan menyebabkan CRC (*Cyclic Redundancy Check*) *file* rusak. *Cyclic Redundancy Check* (CRC) adalah salah satu fungsi hash yang dikembangkan untuk mendeteksi kerusakan data dalam proses transmisi ataupun penyimpanan. CRC menghasilkan suatu checksum yaitu suatu nilai dihasilkan dari fungsi hash-nya, dimana nilai inilah yang nantinya digunakan untuk mendeteksi error pada transmisi ataupun penyimpanan data. Ketika CRC dari suatu *file* rusak maka CRC tidak dapat lagi menggambarkan isi dari suatu *file*.

Berdasarkan Tabel 4.4 nilai MSE pada pesan setelah diekstraksi dibandingkan dengan pesan sebelum disisipkan dan dikompresi seluruhnya menghasilkan nilai 0. Hal tersebut menandakan bahwa seluruh *file* pesan hasil proses ekstraksi dapat kembali ke bentuk sebelumnya seperti *file* pesan sebelum proses penyisipan.

* + 1. ***Imperceptibility***

Kriteria ini merupakan kriteria yang diukur dengan cara hanya melihat secara kasat mata, untuk melihat kemampuan steganografi dalam penelitian ini digunakan kuesioner yang dibagikan kepada beberapa orang yang melihat perbedaan cover sebelum dan sesudah dilakukan penyisipan pesan serta pesan sebelum disisipkan dan setelah dilakukan ekstraksi. Untuk melihat dan menguji *imperceptibility* maka digunakan kuesioner yang dibagikan kepada 30 responden.

Pertanyaan-pertanyaan yang pada kuesioner adalah sebagai berikut:

1. Terdapat perbedaan antara *file* cover dengan *file* stego?
2. Terdapat perubahan makna kata atau informasi dan dimensi pada *file* stego jika dibandingkan dengan cover?
3. Terdapat perubahan terhadap pesan gambar sebelum dan sesudah diekstrak?

Setelah kuesioner diisi oleh responden maka didapatkan hasil pada Tabel 4.5.

Tabel 4.5 Hasil kuesioner

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No. | Pertanyaan | Score | | | | |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Terdapat perbedaan antara *file* cover dengan *file* stego? | 33,3% | 56,6% | 9,9% | 0% | 0% |
| 2 | Terdapat perubahan makna kata atau informasi dan dimensi pada *file* stego jika dibandingkan dengan cover? | 46,6% | 46,6% | 6,6% | 0% | 0% |
| 3 | Terdapat perubahan terhadap pesan gambar sebelum dan sesudah diekstrak? | 49,9% | 36,6% | 13,3% | 0% | 0% |

**Keterangan :**

1 = Sangat Tidak Setuju 3 = Cukup 5 = Sangat Setuju

2 = Tidak Setuju 4 = Setuju

Berdasarkan Tabel 4.5 dapat disimpulkan sebagai berikut:

1. Perbedaan antara *file* cover dengan *file* stego sangat kecil hampir tidak terlihat perbedaannya, ini di tunjukkan dengan hasil kuesioner yaitu 89,9% responden menjawab tidak dan 9,9% responden menjawab cukup. Tidak ada responden yang setuju jika terdapat perubahan pada *file* cover setelah disisipkan pesan.
2. Tidak terdapat perubahan makna kata, informasi atau dimensi pada *file* stego, ini ditunjukkan dengan hasi kuesioner yaitu 93,2% responden menjawab tidak terdapat dan 6,6% responden menjawab cukup. Tidak ada responden yang setuju jika terdapat perubahan makna, informasoi serta dimensi pada *file* cover setelah disisipkan pesan.
3. Tidak terdapat perubahan yang signifikan terhadap pesan sebelum disisipkan dengan pesan setelah disisipkan, ini ditunjukkan dengan hasil kuesioner 86,5% responden menjawab tidak dan 13,3% responden menjawab cukup. Tidak ada responden yang setuju jika terdapat perubahan pada *file* pesan setelah diekstraksi dari *file* stego.

Hasil kuesioner pertanyaan satu dan dua menunjukkan pesan berhasil disembunyikan dengan baik pada *file* cover sehingga ketika dilihat dengan kasat mata orang tidak dapat mengetahui jika *file* sudah disisipkan pesan gambar. Hasil kuesioner pertanyaan 3 menunjukkan pesan berhasil diekstraksi dan didekompresi dengan baik sehingga dapat dibaca dan memiliki tingkat kesamaan yang tinggi antara pesan sebelum dan sesudah disisipkan.

* + 1. ***Robustness***

Pengujian *Robustness* atau ketahanan dilakukan untuk mengetahui kemampuan dari *file* stego ketika *file* stego mengalami perubahan apakah data pesan rahasia mengalami gangguan atau bahkan tidak dapat diekstraksi. Tabel 4.6 merupakan tabel hasil pengujian pada *file*-file *cover* yang diuji pada penelitian, setelah disisipkan pesan kemudian *file*-*file* tersebut dilakukan pengubahan terhadap isi dari *file* dan dilakukan ekstraksi terhadap *file* tersebut.

Tabel 4.6 Hasil pengujian *robustness*

|  |  |  |
| --- | --- | --- |
| No. | File Cover | Berhasil diekstraksi |
| 1. | PDF | Tidak |
| 2. | JPG | Tidak |
| 3. | PNG | Tidak |
| 4. | DOCX | Tidak |
| 5. | PPTX | Tidak |
| 6. | XLSX | Tidak |

Berdasarkan Tabel 4.6 dapat disimpulkan *file* hasil penyisipan atau stego objek dari metode *First of File, End of File* atau kombinasinya akan mengalami kerusakan ketika isi dalam *file* mengalami perubahan sekecil apapun, ini dapat disebabkan oleh perubahan *byte* data pada *file* stego sehingga informasi terkait pesan mengalami perubahan yang dapat menyebabkan pesan tidak dapat diekstraksi kembali.

**BAB V**

**KESIMPULAN DAN SARAN**

1. **Kesimpulan**

Dari penelitian yang telah dilakukan kesimpulan yang telah didapatkan yaitu sebagai berikut:

1. Kompresi RLE dengan modifikasi berhasil memperbaiki kekurangan RLE konvensional yaitu tidak terdapat hasil kompresi yang menjadi lebih besar dari ukuran sebelum dikompresi. Ini ditunjukkan dengan rata-rata rasio kompresi dengan RLE pada JPG, PNG dan BMP sebesar -71,03% sedangkan RLE dengan modifikasi sebesar 6,4%.
2. Kompresi dengan menggunakan RLE Modifikasi atau RLE tanpa modifikasi lebih baik jika diterapkan pada gambar BMP dibandingkan dengan gambar JPG dan PNG, ini ditunjukkan oleh data dimana rata-rata hasil kompresi pada *file* berformat BMP lebih besar yaitu 18,8% jika dibandingkan JPG dengan PNG yang memiliki rata-rata 0,17% menggunakan RLE modifikasi dan memiliki rasio -34,7% jika dibandingkan JPG dengan PNG yang memiliki rata-rata -98,2% menggunakan RLE tanpa modifikasi.
3. Metode *First of File* tidak dapat diterapkan pada *file* bertipe JPG, PNG, DOCX, XLSX dan PPTX, ini disebabkan karena penempatan data di awal *file* akan merusak *file* asli karena mengganggu isi *file* asli dan merusak CRC (*Cyclic Redundancy Check*) *file*.
4. Metode steganografi dengan *First* *of* *File*, *End* *of* *File* dan kombinasinya memiliki kelemahan terkait ketahanannya terhadap perubahan data, pengujian yang dilakukan pada berbagai format stego setelah disisipkan pesan menunjukkan setelah dilakukan pengubahan pada *file* stego, pesan tidak dapat diekstraksi dari *file*.
   1. **Saran**

Saran untuk penelitian selanjutnya penggunaan metode steganografi *First of File*, *End of File* dan kombinasi dapat dimodifikasi agar dapat diterapkan pada berbagai *file* cover. Pada algoritma kompresi RLE dapat dikembangkan lagi konsep penanda pada hasil kompresi sehingga tidak perlu melakukan komputasi yang tinggi dalam melakukan pencarian penanda.

Lampiran I.

KUESIONER

Nama :

NIM :

Silahkan memberi tanda centang (ѵ) pada pilihan yang saudara/I pilih.

**Keterangan :**

1 = Sangat Tidak Setuju 3 = Cukup 5 = Sangat Setuju

2 = Tidak Setuju 4 = Setuju

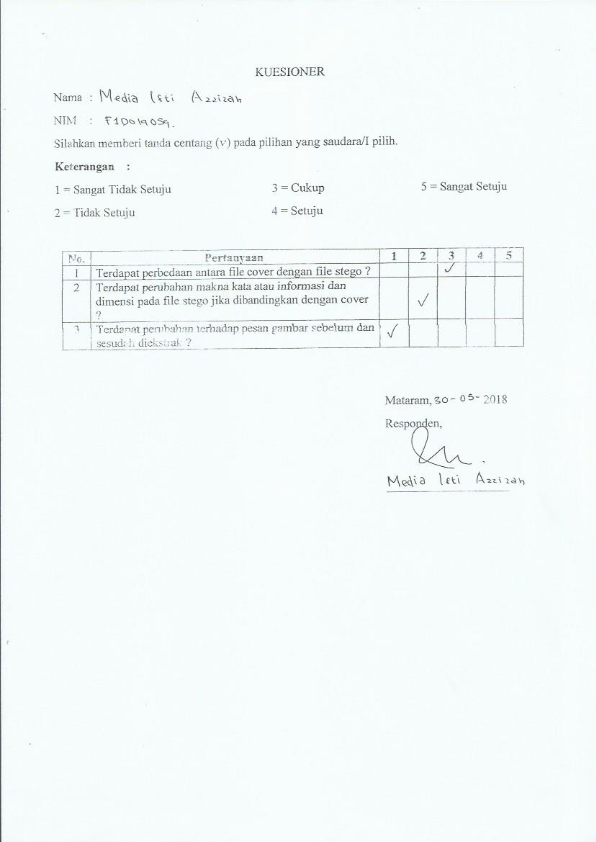
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Pertanyaan** | **1** | **2** | **3** | **4** | **5** |
| 1 | Terdapat perbedaan antara file cover dengan file stego ? |  |  |  |  |  |
| 2 | Terdapat perubahan makna kata atau informasi dan dimensi pada file stego jika dibandingkan dengan cover ? |  |  |  |  |  |
| 3 | Terdapat perubahan terhadap pesan gambar sebelum dan sesudah diekstrak ? |  |  |  |  |  |

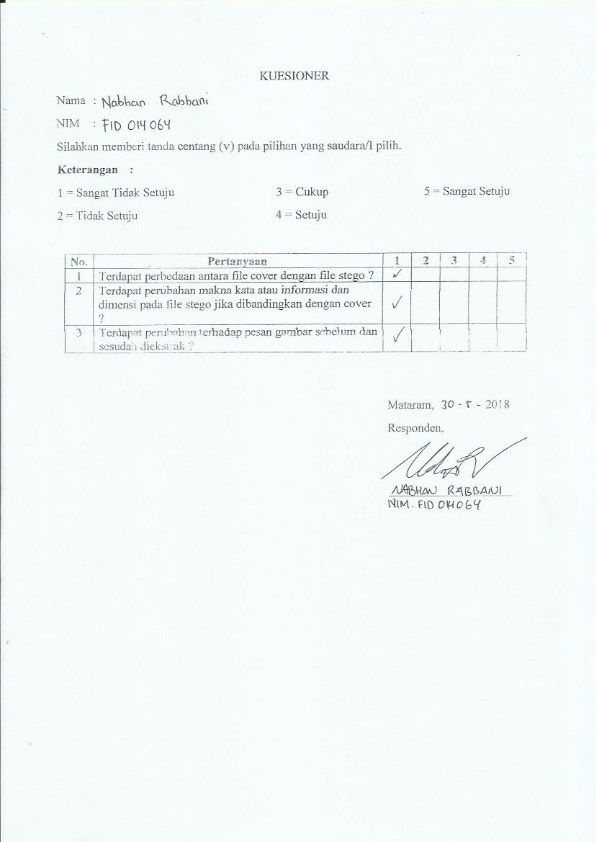
Mataram, 2018

Responden,









Lampiran II.

**Struktur File**

* + - * 1. JPEG, Joint Photographic Experts Group (JPEG) merupakan skema kompresi file bitmap. Awalnya, *file* yang menyimpan hasil foto digital memiliki ukuran yang besar sehingga tidak praktis. Dengan format baru ini, hasil foto yang semula berukuran besar berhasil dikompresi (dimampatkan) sehingga ukurannya kecil.

Tabel 1. Struktur *file* JPEG.

|  |
| --- |
|  |
| **Short name** | ***Byte*s** | **Payload** | **Name** | **Comments** |
| **SOI** | 0xFF, 0xD8 | *none* | Start *Of* Image |  |
| **S*OF*0** | 0xFF, 0xC0 | *variable size* | Start *Of* Frame (baseline [DCT](https://en.wikipedia.org/wiki/Discrete_cosine_transform)) | Indicates that this is a baseline DCT-based JPEG, and specifies the width, height, number *of* components, and component subsampling (e.g., 4:2:0). |
| **S*OF*2** | 0xFF, 0xC2 | *variable size* | Start *Of* Frame (progressive DCT) | Indicates that this is a progressive DCT-based JPEG, and specifies the width, height, number *of* components, and component subsampling (e.g., 4:2:0). |
| **DHT** | 0xFF, 0xC4 | *variable size* | Define Huffman Table(s) | Specifies one or more Huffman tables. |
| **DQT** | 0xFF, 0xDB | *variable size* | Define Quantization Table(s) | Specifies one or more quantization tables. |
| **DRI** | 0xFF, 0xDD | 4 *byte*s | Define Restart Interval | Specifies the interval between RST*n* markers, in Minimum Coded Units (MCUs). This marker is followed by two *byte*s indicating the fixed size so it can be treated like any other variable size segment. |
| **SOS** | 0xFF, 0xDA | *variable size* | Start *Of* Scan | Begins a top-to-bottom scan *of* the image. In baseline DCT JPEG images, there is generally a single scan. Progressive DCT JPEG images usually contain multiple scans. This marker specifies which slice *of* data it will contain, and is immediately followed by entropy-coded data. |
| **RST*n*** | 0xFF, 0xD*n*(*n*=0..7) | *none* | Restart | Inserted every *r* macroblocks, where *r* is the restart interval set by a DRI marker. Not used if there was no DRI marker. The low three bits *of* the marker code cycle in value from 0 to 7. |
| **APP*n*** | 0xFF, 0xE*n* | *variable size* | Application-specific | For example, an [Exif](https://en.wikipedia.org/wiki/Exif) JPEG file uses an APP1 marker to store metadata, laid out in a structure based closely on [TIFF](https://en.wikipedia.org/wiki/TIFF). |
| **COM** | 0xFF, 0xFE | *variable size* | Comment | Contains a text comment. |
| **EOI** | 0xFF, 0xD9 | *none* | End *Of* Image |  |

Pada penelitian ini data atau pesan rahasia berupa gambar akan disisipkan sebelum SOI(*Start of Image*) untuk metode *First of File.* Pada metode *End of File* data atau pesan rahasia akan disisipkan setelah EOI (*End of Image*).

* + - * 1. PNG, PNG (Portable Network Graphics) adalah salah satu format penyimpanan citra yang menggunakan metode pemadatan yang tidak menghilangkan bagian dari citra tersebut (lossless compression).

Berikut merupakan struktur *file* PNG:

* + - 1. PNG file signature, delapan *byte* pertama dari *file* PNG selalu mengandung nilai-nilai berikut (desimal):

137 80 78 71 13 10 26 10

Tanda ini menunjukkan bahwa sisa *file* berisi gambar PNG tunggal, yang terdiri dari serangkaian potongan yang diawali dengan potongan IHDR dan diakhiri dengan potongan IEND.

* + - 1. *Chunk* *layout*, terdiri dari *length, chunk type, chunk* datadan CRC.
      2. *Chunk Naming Conventions,* Kode jenis Chunk ditetapkan sehingga decoder dapat menentukan beberapa properti potongan bahkan ketika tidak mengenali kode jenis. Aturan-aturan ini dimaksudkan untuk memungkinkan perluasan format PNG yang aman dan fleksibel, dengan memungkinkan decoder untuk memutuskan apa yang harus dilakukan ketika menemukan potongan yang tidak diketahui.
      3. CRC Algoritma, Chunk CRC dihitung menggunakan metode CRC standar dengan pengkondisian pra dan pasca, sebagaimana didefinisikan oleh ISO 3309 [ISO-3309] atau ITU-T V.42 [ITU-V42].

Pada penelitian ini data atau pesan rahasia berupa gambar akan disisipkan sebelum PNG *file signature* untuk metode *First of File.* Pada metode *End of File* data atau pesan rahasia akan disisipkan setelah CRC algoritma.

1. PDF, PDF (*Portable* *Document* *Format*) adalah sebuah format berkas yang dibuat oleh Adobe *Systems* pada tahun 1993 untuk keperluan pertukaran dokumen digital. Format PDF digunakan untuk merepresentasikan dokumen dua dimensi yang meliputi teks, huruf, citra dan grafik vektor dua dimensi.

Berikut merupakan struktur *file* PDF:

* + - 1. Header, yang memberikan nomor versi PDF.
      2. Badan, yang berisi halaman, konten grafis, dan banyak informasi tambahan, semuanya dikodekan sebagai serangkaian objek.
      3. Tabel referensi silang, yang berisi daftar posisi setiap objek dalam *file*, untuk memfasilitasi akses acak.
      4. Trailer termasuk kamus trailer, yang membantu untuk menemukan setiap bagian dari *file* dan daftar berbagai potongan metadata yang dapat dibaca tanpa memproses seluruh *file*.

Pada penelitian ini data atau pesan rahasia berupa gambar akan disisipkan sebelum *header file* pdf untuk metode *First of File.* Pada metode *End of File* data atau pesan rahasia akan disisipkan setelah trailer.