Jurnal Rekursif, Vol. 5 No. 1 Maret 2017, ISSN 2303-0755
http://ejournal.unib.ac.id/index.php/rekursif/

APLIKASI PENYEMBUNYIAN MULTIMEDIA MENGGUNAKAN METODE *END OF FILE* (EOF) DAN *HUFFMAN CODING*

Yetti Rahayu Nasution[1], Asahar Johar[2], Funny Farady Coastera[3]

[1,2,3]Program Studi Teknik Informatika, Fakultas Teknik, Universitas Bengkulu
Jl. WR. Supratman Kandang Limun Bengkulu 38371A INDONESIA
(telp: 0736-341022; fax: 0736-341022)

[1]yettirahayunst@gmail.com,
[2]asahar.johar@unib.ac.id,
[3]ffaradyc@unib.ac.id

Abstrak : Semakin berkembangnya teknologi internet, maka semakin berkembang pula penyalahgunaan informasi. Pertukaran informasi harus dilakukan dengan teknik yang aman. Salah satu teknik keamanan informasi adalah steganografi. Pada penelitian ini, konsep steganografi diterapkan pada multimedia dengan metode *End of File* dan *Huffman Coding.* Metode penyembunyian data rahasia dengan *End of File* dapat menyisipkan data berukuran besar kedalam media penampung yang ukurannya lebih kecil. Namun metode ini memiliki kekurangan, yakni ukuran media penampung yang sudah disisipi data rahasia akan bertambah besar. Maka dari itu, penelitian ini juga menggunakan teknik kompresi untuk memampatkan data yang akan disisipkan agar data yang berulang pada data rahasia tersebut dapat tereduksi, yaitu dengan metode *Huffman Coding*. Hasil dari penelitian ini menunjukkan bahwa metode *End of File* efektif dalam menyisipkan data dengan ukuran besar, serta dengan diterapkannya metode *Huffman Coding,* maka ukuran data yang disisipkan berkurang. Hanya saja metode *Huffman Coding* kurang efektif jika digunakan sebagai algoritma kompresi untuk media gambar, *audio* dan *video*. Rasio kompresi yang paling tinggi yaitu pada media teks dengan rasio kompresi 55,076%, rasio kompresi pada media gambar 6,0656%, media *audio* 4,788%, serta media *video* sebesar 4,04%. Sistem ini dibangun dengan menggunakan bahasa pemrograman *java* dengan IDE Netbeans 7.4.

Kata kunci : Keamanan Data, Steganografi, Kompresi, *End of file* (EOF), *Huffman Coding*

***Abstract* : The continued development of internet technology, it is also growing a crime of misuse of the information. The exchange of information should be done with secure technique. One of the techniques of information security is steganography. In this study, the concept of steganography applied to multimedia with End of File, and compressin by Huffman Coding methodes. Method of data hiding by the End of File can insert large data size into a smaller size of media container. But this method has its drawbacks, namely the size of the media**

container that has been inserted secret data will increase. Therefore, this study also uses compression techniques to compress the data that to be inserted, so that repetitive data on secret data can be reduced, the method that used is Huffman Coding. The results of this study indicate that the method End of File is effective in inserting data with a large size, and also with the implementation of Huffman Coding method, the size of data that is inserted is reduced. But Huffman coding method is less effective when used as compression algorithms for image, audio and video. The system is built using java language program with Netbeans IDE 7.4.

I. PENDAHULUAN

Pada era informasi digital sekarang ini, manusia semakin dipermudah untuk bertukar informasi melalui *internet* tanpa harus bertatap muka secara langsung. Banyak perusahaan, instansi biasa, bahkan instansi pemerintahan melakukan pertukaran informasi melalui internet. Dimana informasi yang dipertukarkan tentu saja merupakan informasi penting yang banyak diminati oleh berbagai pihak yang memiliki kepentingan didalamnya. Tetapi pertukaran informasi tanpa adanya penyembunyian pesan atau informasi sangat tidak aman. Apalagi untuk pengiriman pesan yang bersifat rahasia seperti nomor kode kartu kredit, dokumen rahasia perusahaan maupun instansi yang dikirim melalui perusahaan. Bisa saja informasi rahasia ini diambil oleh orang yang tidak berhak. Untuk itu keamanan data rahasia sudah menjadi sangat penting pada zaman sekarang ini. Salah satu jenis pengamanan informasi adalah steganografi, yaitu dengan cara menyembunyikan informasi atau pesan ke media lain, seperti gambar, *audio* atau *video* sehingga tidak menimbulkan kecurigaan orang lain.

Namun seiring dengan perkembangan teknologi yang semakin pesat, penyembunyian pesan seperti steganografi ini juga sudah tidak aman, karena sudah adanya metode penyerangan terhadap *steganografi* dengan memanfaatkan kelemahannya. Metode tersebut yaitu *Visual Attacks* dan *Statistical Attacks*. Serangan visual (*visual attacks*), untuk menjelaskan perbedaan antara *noise* dan *visual patterns*, sedangkan serangan statistik (*statistical attacks*) untuk mendeteksi metode steganografi yang digunakan [9]. Untuk mengatasi masalah ini, maka dibutuhkan aplikasi yang bisa melindungi *stegofile* dengan *password* sehingga tidak bisa diakses oleh pihak yang tidak berwenang.

Kapasitas *file* menampung pada *steganografi* menjadi masalah selanjutnya yang sering muncul karena keterbatasan *file* penampung untuk disisipi *file* rahasia. Karena besarnya pesan yang akan disisipi bergantung pada ukuran piksel pada citra. Maka dari itu, solusi yang diperlukan untuk mengatasi masalah ini adalah dengan menggunakan algoritma yang bisa mengatasi masalah kapasitas *file* penampung, yaitu dengan metode *End of File.* Dengan metode *End of File,* ukuran pesan yang akan disisipi bisa lebih besar daripada *file* penampung, tetapi akan mengubah ukuran *stegofile*-nya (*file* yang sudah disteganografi). Perubahan ukuran *stegofile* ini juga akan menimbulkan masalah baru pada

steganografi, karena akan memunculkan kecurigaan dikarenakan ukuran *file* yang begitu besar. Untuk itu perlu dilakukan proses kompresi. Dengan proses kompresi ini, maka diharapkan ukuran *stegofile* bisa menjadi lebih kecil. Metode yang digunakan untuk kompresi adalah *Huffman Coding*.

Berdasarkan masalah-masalah yang telah dipaparkan diatas, maka dibutuhkan aplikasi pengamanan data yang dapat menyisipkan *file* rahasia dengan ukuran yang lebih besar kedalam *file* penampung dengan ukuran yang lebih kecil, dapat memampatkan data, serta dapat menjaga keamanan *stegofile* dengan menggunakan *password.*

II. LANDASAN TEORI

A. *Steganografi*

1) Pengertian *Steganografi*

Kata *steganorafi* berasal dari Bahasa Yunani yang berarti "tulisan tersembunyi" (*covered writing*). *Steganografi* merupakan suatu ilmu dan seni menyembunyikan pesan rahasia sedemikian rupa, sehingga keberadaan pesan tidak terdeteksi oleh indera manusia. *Steganografi* membutuhkan dua properti, yaitu wadah penampung dan data rahasia yang akan disembunyikan [4].

Perbedaan steganografi dan kriptografi terletak pada bagaimana proses penyembunyian data, serta bagaimana hasil akhir dari proses tersebut. Kriptografi melakukan pengacakan data asli, sehingga menghasilkan data terenkripsi yang benar-benar acak dan berbeda dengan aslinya. Sedangkan steganografi menyembunyikan pesan rahasia dalam media lain tanpa mengubah media yang ditumpanginya [2].

2) Sejarah *Steganografi*

Steganografi pertama kali dipakai pada masa pemerintahan Yunani kuno dan Persia, Caesar menulis pesan dengan media papan. Setelah pesan ditulis, kemudian papan dilapisi dengan lilin sehingga pesan tidak bisa dibaca. Papan yang berisi pesan tersebut kemudian dikirim dengan menggunakan jasa kurir. Lama-lama metode menyembunyikan pesan ini diketahui oleh pihak lawan, sehingga Caesar mengganti metode ini dengan metode *Caesar Cipher*. Steganografi juga digunakan di Cina dengan menggunakan teknik *histaleus,* yaitu menulis pesan yang akan dikirim diatas kepala kurir yang telah digunduli. Tapi metode ini membutuhkan waktu yang lama, karena pesan yang telah diukir bisa dikirim setelah rambut kurirnya tumbuh [2].

3) Kriteria *Steganografii* yang Bagus

Penyembunyian data rahasia ke dalam citra digital akan mengubah kualitas citra tersebut. Kriteria yang harus diperhatikan dalam penyembunyian data adalah [4]:

a. *Fidelity*, yaitu kualitas citra penampung tidak jauh berubah setelah data rahasia disisipkan, serta pengamat tidak mengetahui kalau di dalam citra tersebut terdapat data rahasia.
b. *Robustness*, yaitu data yang disembunyikan harus tahan terhadap manipulasi yang dilakukan pada citra penampung. Bila pada citra

dilakukan operasi pengolahan citra, maka data yang disembunyikan tidak rusak.

c. *Recovery*, yaitu 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.

B. Kompresi

Kompresi merupakan proses memampatkan ukuran suatu data untuk menghasilkan representasi digital yang padat atau mampat (*compact)*, namun tetap dapat mewakili kuantitas informasi yang terkandung pada data tersebut. Tujuan daripada kompresi data tiada lain adalah untuk mengurangi data berlebihan, sehingga ukuran data menjadi lebih kecil dan lebih ringan dalam proses transmisi [5].

Kompresi data penting dilakukan karena dapat memperkecil kebutuhan penyimpanan data, mempercepat pengiriman data, serta memperkecil kebutuhan *bandwidth*. Teknik kompresi bisa dilakukan terhadap data teks/biner, gambar, *audio*, dan *video*. Kompresi data memiliki banyak jenis, seperti *dialog mode, retrieval mode, lossy compression, loseless compression* [3].

Rasio kompresi data adalah ukuran presentase data yang telah berhasil dimampatkan. Secara matematis, rasio pemampatan data ditulis sebagai berikut [1]:

$$\text{Rasio kompresi data} = \frac{(\text{Ukuran data asli} - \text{ukuran data terkompresi})}{\text{Ukuran data asli}} \times 100\% \quad \text{Persamaan (2.1)}$$

Metode penempatan data atau kompresi data dapat dikelompokkan kedalam dua kelompok besar, yaitu:

a) *Losslesss Compression*

Pada teknik ini tidak ada kehilangan informasi. Jika data dimampatkan secara *lossless*, data asli dapat direkonstruksi kembali sama persis seperti data yang telah dimampatkan. Dengan kata lain, data asli tetap sama sebelum dan sesudah pemampatan. Secara umum teknik *lossless* digunakan untuk penerapan yang tidak bisa mentoleransi setiap perbedaan antara asli dan data yang telah direkonstruksi.

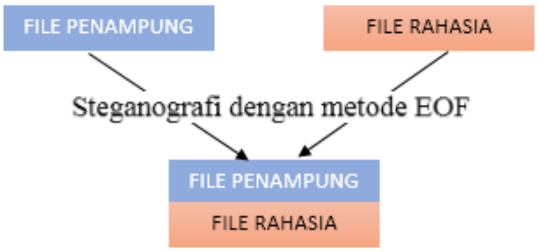
b) *Lossy Compression*

Pada teknik ini akan terjadi kehilangan sebagian informasi. Data yang telah dimampatkan dengan teknik ini secara umum tidak bisa direkonstruksi sama persis dari data aslinya. Biasanya teknik ini membuang bagian-bagian data yang sebenarnya tidak begitu berguna, tidak begitu dirasakan, tidak beranggapan bahwa data tersebut masih bisa digunakan walaupun sudah dikompresi.

C. Metode *End of File*

Metode *End of File* (EOF) merupakan salah satu teknik yang menyisipkan data pada akhir *file.* Teknik ini dapat digunakan untuk menyisipkan data yang ukurannya sama dengan ukuran *file* sebelum disisipkan data ditambah dengan ukuran data yang

disisipkan kedalam *file* tersebut. Metode EOF merupakan sebuah metode yang di adaptasi dari metode penanda akhir *file* (*End of File*) yang digunakan oleh sistem operasi *windows.* Prinsip kerja EOF menggunakan karakter/simbol khusus yang diberikan pada setiap akhir *file*. Pada metode EOF ukuran pesan yang akan disisipi bisa lebih besar dari ukuran citranya. Kualitas citra setelah disisipi pesan tidak berubah, tetapi akan mengubah ukuran citranya [7]. Dengan metode EOF, secara umum media steganografi (*file* yang akan disisipi data) memiliki struktur seperti Gambar 1 berikut:



Gambar 1 Konsep EOF pada *Steganografi*

Gambar 1 merupakan konsep EOF pada steganografi, dimana ukuran *stegofile* sama dengan ukuran *file* penampung ditambah dengan ukuran *file* rahasia.

D. Metode *Huffman Coding*

Istilah kode Huffman diambil dari nama penemunya, yaitu David A. Huffman Pada tahun 1951. Kode Huffman mirip dengan kode Morse, yaitu memanfaatkan kekerapan kemunculan simbol yang akan dimampatkan, sehingga simbol-simbol yang sering muncul dapat direpresentasikan dalam kode-kode yang lebih pendek. Algoritma Huffman menggunakan prinsip pengkodean dimana tiap karakter *(symbol)* dikodekan hanya dengan rangkaian beberapa bit, dimana karakter yang sering muncul dikodekan dengan rangkaian bit yang pendek, dan karakter yang jarang muncul dikodekan.dengan rangkaian bit yang lebih panjang. Skema kode Huffman dilakukan dengan membangun sebuah pohon *biner* dengan jalur berbobot 0 atau 1 di mana elemen-elemen berupa simbol-simbol yang akan dikodekan diurutkan berdasarkan kekerapan kemunculannya [8].

III. METODE PENELITIAN

A. Jenis Penelitian

Jenis penelitian yang dilakukan adalah penelitian terapan. Penelitian terapan adalah penyelidikan yang hati-hati, sistematik dan terus menerus terhadap suatu masalah dengan tujuan untuk digunakan dengan segera untuk keperluan tertentu. Hasil dari penelitian terapan tidak memerlukan suatu penemuan baru, tapi merupakan aplikasi baru dari penelitian yang telah ada. Penelitian ini berusaha menerapkan teori atau metode yang telah dikembangkan baik dalam cakupan peneltian murni maupun penelitian terapan seperti sistem basis data, bahasa pemrograman, dan lain-lain.

Pada penelitian ini akan dibangun suatu aplikasi yang dapat menyembunyikan *file* kedalam suatu media. Metode yang digunakan dalam penelitian ini adalah *End of File* untuk menyembunyikan *file*, serta metode *Huffman Coding* untuk kompresi *file*.

B. Teknik Pengumpulan Data

1) Studi Pustaka

Penulis melakukan pengumpulan data

dengan metode studi pustaka, yaitu mengumpulkan berbagai referensi dari buku, *internet*, artikel maupun sumber-sumber lainnya sebagai bahan yang digunakan untuk membantu membangun Aplikasi Penyembunyian Multimedia Menggunakan Metode *End of File* (EOF) dan *Huffman Coding*.

2) Studi Literatur

Selain studi pustakan, penulis juga menggunakan referensi lain berupa tulisan dari skripsi atau penelitian yang objeknya hampir sama.

3) Wawancara

Kuesioner atau pertanyaan ini dilakukan penulis dengan memberikan pertanyaan yang secara logis berhubungan dengan masalah penelitian. Dimana kuisioner disini digunakan penulis pada tahap penelitian.

C. Metode Pengembangan Sistem

Penulis menggunakan metode pengembangan sistem *Waterfall Model* dalam perancangan aplikasi ini. Model *waterfall* sering juga disebut model sekuensial linier atau alur hidup klasik. Berikut adalah penjelasan dari tahap-tahap *Waterfall Model* [6]*:*

1) Analisis Kebutuhan Perangkat Lunak

Proses pengumpulan kebutuhan dilakukan secara intensif untuk menspesifikasikan kebutuhan perangkat lunak agar dapat dipahami perangkat lunak seperti apa yang dibutuhkan oleh *user*. Spesifikasi kebutuhan perangkat lunak pada tahap ini perlu didokumentasikan.

2) Desain

Desain perangkat lunak adalah proses aneka langkah yang fokus pada desain pembuatan program perangkat lunak termasuk struktur data, arsitektur perangkat lunak, representasi antarmuka, dan prosedur pengodean. Tahap ini mentranslasi kebutuhan perangkat lunak dari tahap analisis kebutuhan ke representasi desain agar dapat diimplementasikan menjadi program. Pada tahap selanjutnya. Desain perangkat lunak yang dihasilkan pada tahap ini juga perlu didokumentasikan.

3) Pembuatan Kode Program

Desain harus ditranslasikan kedalam program perangkat lunak. Hasil dari tahap ini adalah program komputer sesuai dengan desain yang telah dibuat pada tahap desain.

4) Pengujian

Pengujian fokus pada perangkat lunak secara dari segi lojik dan fungsional dan memastikan bahwa semua bagian sudah diuji. Hal ini dilakukan untuk meminimalisir kesalahan *(error)* dan memastikan keluaran yang dihasilkan sesuai dengan yang diinginkan.

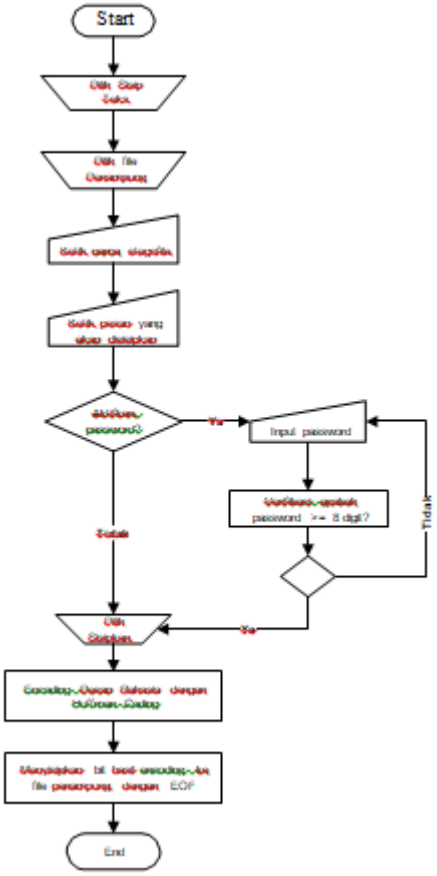
5) Pendukung *(Support)* atau Pemeliharaan *(Maintenance)*

Tahap pendukung atau pemeliharaan akan mengulangi proses pengembangan mulai dari analisis spesifikasi untuk perubahan perangkat lunak yang sudah ada, tapi tidak untuk membuat perangkat lunak baru.

IV. ANALISIS DAN PERANCANGAN

A. *Flowchart*

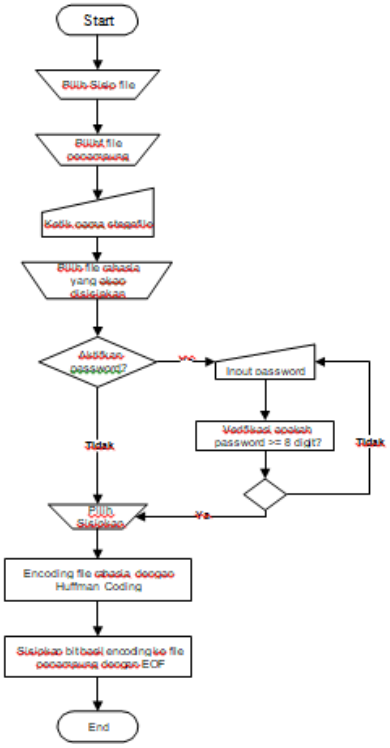
1) *Flowchart* Penyisipan Data Teks



Gambar 4.1. *Flowchart* Penyisipan Data Teks

Berdasarkan *flowchart* penyisipan data teks pada Gambar 4. 1, langkah-langkahnya adalah setelah aplikasi dimulai, maka *user* memilih menu Sisip Teks, kemudian *user* memilih *file* penampung, lalu mengetikkan nama *stegofile.* Setelah itu akan muncul layar *embed message*, lalu *user* mengetikkan teks yang akan disisipkan ke *file* penampung. Jika pengamanan *stegofile* tidak diperlukan maka *user* mengklik tombol Sisipkan. Sistem akan menyisipkan pesan rahasia ke *file* penampung menggunakan algoritma *End of File.* Jika *user* ingin melindungi *stegofile*, maka *user* mengaktifkan *password*. lalu mengetikkan *password* minimal 8 digit. Sistem akan memvalidasi apakah *password* sudah benar. Jika *password* yang diinputkan *user* kurang dari 8 digit, maka sistem akan mengarahkan *user* untuk kembali memasukkan *password*. Setelah *password* benar, lalu *user* mengklik tombol Sisipkan. Sistem akan mengompresi lalu menyisipkan pesan rahasia ke *file* penampung.

2) *Flowchart* Penyisipan *File*

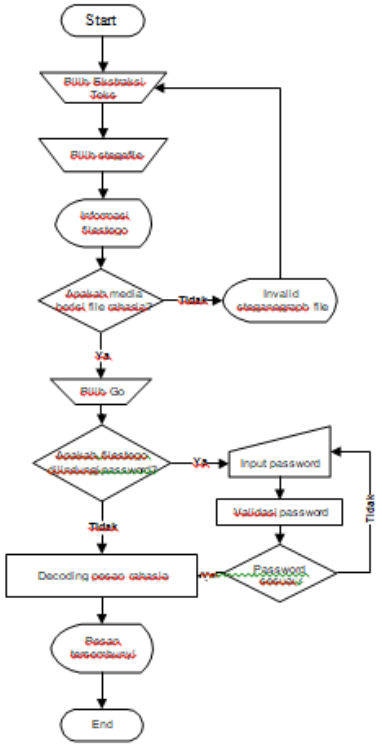


Gambar 4.2. *Flowchart* Penyisipan *file*

Berdasarkan *flowchart* penyisipan *file* pada Gambar 4. 2 langkah-langkahnya adalah setelah aplikasi dimulai, maka *user* memilih menu Sisip *File*, kemudian *user* memilih *file* penampung, lalu mengetikkan nama *stegofile.* Setelah itu *user* memilih *file* yang akan disisipkan ke media penampung. Jika pengamanan *stegofile* tidak diperlukan maka

user mengklik tombol Sisipkan. Sistem akan mengkompresi *file* rahasia dengan metode *Huffman Coding*, dan menyisipkan *file* rahasia ke *file* penampung menggunakan algoritma *End of File*. Jika *user* ingin melindungi *stegofile*, maka *user* mengaktifkan *password*. Lalu mengetikkan *password* minimal 8 digit. Sistem akan memvalidasi apakah *password* sudah benar. Jika *password* yang di *input*-kan *user* kurang dari 8 digit, maka sistem akan mengarahkan *user* untuk kembali memasukkan *password*. Setelah *password* benar, *user* akan mengklik tombol Sisipkan. Maka sistem akan mengkompresi *file* rahasia dengan metode *Huffman Coding*, dan menyisipkan *file* rahasia terkompresi ke *file* penampung menggunakan algoritma *End of File.*

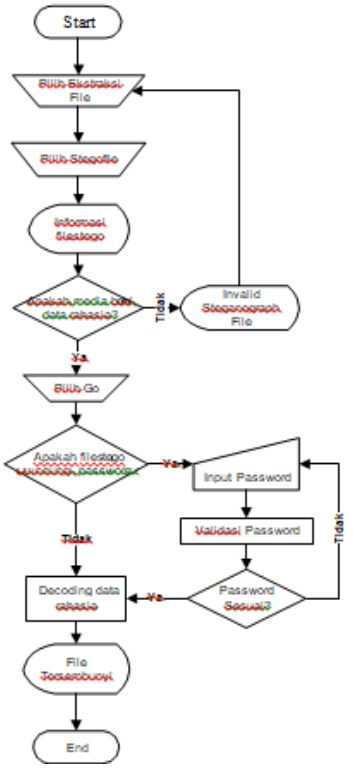
3) *Flowchart* Ekstraksi Teks



Gambar 4.3. *Flowchart* Ekstraksi teks

Berdasarkan *flowchart* ekstraksi pesan pada Gambar 4. 3, setelah aplikasi dimulai, maka *user* memilih menu Ekstraksi Teks. Setelah itu *user* memilih *stegofile*, maka sistem akan menampilkan informasi *stegofile*. Jika media merupakan *stegofile* atau berisi pesan teks yang disembunyikan, maka *user* mengklik tombol *Go*. Apabila *stegofile* tidak dilindungi dengan *password*, maka sistem akan langsung menampilkan pesan yang sudah disisipkan sebelumnya ke layar. Tetapi jika *stegofile* dilindungi dengan *password*, sistem akan meminta pengguna untuk menginputkan *password*. Jika *password* benar maka sistem akan menampilkan *embeded message* ke layar.

4) *Flowchart* Ekstraksi *file*

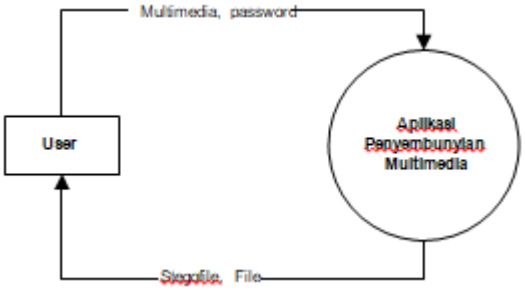


Gambar 4.4. *Flowchart* Ekstraksi *file*

Berdasarkan *flowchart* ekstraksi *file* pada Gambar 4. 4, setelah aplikasi dimulai, maka *user* memilih menu Ekstraksi *File*. Setelah itu *user* memilih *stegofile*, maka sistem akan menampilkan informasi *stegofile*. Jika media merupakan *stegofile* atau berisi *embeded file*, maka *user* mengklik tombol *Go*. Apabila *stegofile* tidak dilindungi dengan *password*, maka sistem akan langsung menampilkan *embeded file* ke layar. Tapi jika *stegofile* dilindungi dengan *password*, sistem akan meminta pengguna untuk menginputkan *password*. Jika *password* benar maka sistem akan menampilkan *file* rahasia ke layar.

B. *Data Flow Diagram (DFD)*

1) Diagram Konteks *(Context Diagram)*

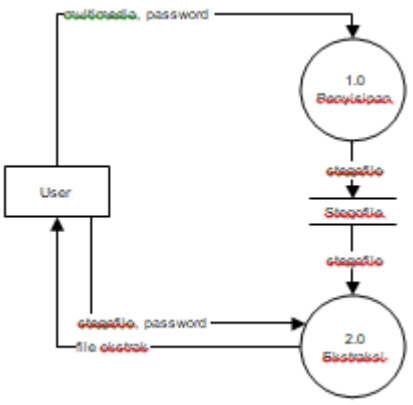


Gambar 4.5. Diagram Konteks

Diagram konteks seperti ditunjukkan pada Gambar 4.5. menggambarkan tahap utama sistem.

2) DFD Level 1 Proses 0

Diagram Konteks pada Gambar 4.5. dapat diperinci lagi menjadi DFD level 1 yang ditunjukkan pada Gambar 4. 6.

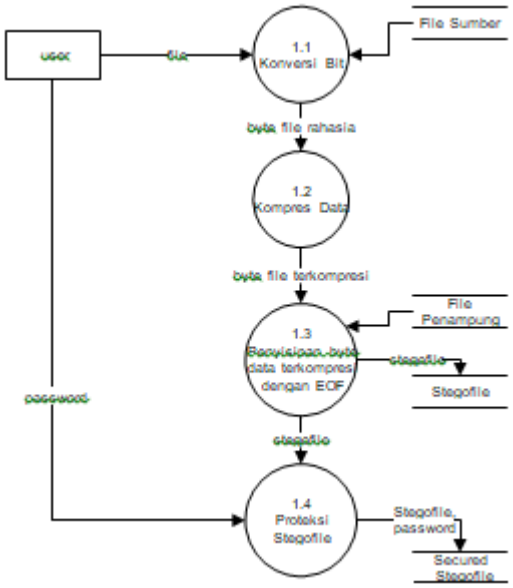


Gambar 4.6. DFD Level 1 Proses 0

DFD Level 1 Proses 0 yang ditunjukkan pada Gambar 4.6. menggambarkan proses yang terjadi antara *user* dengan sistem. Proses yang dapat dilakukan oleh *user* adalah menyisipkan dan ekstraksi *file*. Untuk menyisipkan media, *user* harus memasukkan media yang akan disisipkan, media penampung serta mengetikkan *password* jika diperlukan. Setelah *user* memasukkan media yang akan disisipkan serta *file* penampung, maka sistem akan melakukan proses penyisipan yang nantinya akan menghasilkan *output* berupa *stegofile*.

Untuk mengekstrasksi *file*, maka *user* harus mamsukkan *stegofile* kedalam sistem serta menginputkan *password*. Setelah itu sistem akan menampilkan hasil ekstraksi ke *user*.

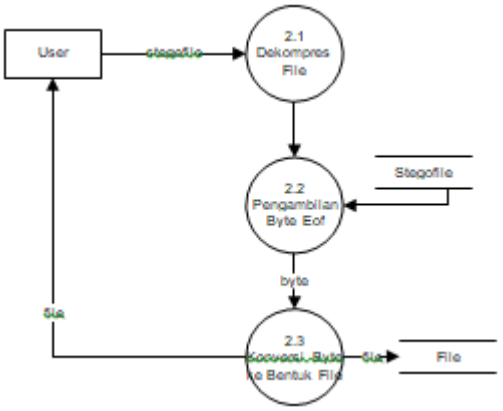
3) DFD Level 1 Proses 1



Gambar 4.7. DFD Level 1 Proses 1

DFD Level 1 Proses 1 yang ditunjukkan pada Gambar 4.7. terdapat 3 proses. Pada proses "Konversi Bit" (proses 1.1), akan dilakukan konversi file kedalam bentuk *byte file*. Proses yang kedua yaitu "Kompres Data" (Proses 1.2), dalam proses ini file rahasia akan akan dikompresi menggunakan metode *Huffman Coding*. Setelah dikompresi, maka akan dilakukan proses 1.3, yaitu "Penyisipan *Byte* Data Terkompresi dengan EOF". Dalam proses ini, *byte file* rahasia yang sudah dikompresi tadi akan disisipkan di akhir berkas dari *file* penampung. Setelah itu sistem akan melakukan proses "Proteksi *Stegofile"* (proses 1.4).

4) DFD Level 1 Proses 2



Gambar 4.8. DFD Level 1 Proses 2

DFD Level 1 Proses 2 yang ditunjukkan pada Gambar 4. 4 terdapat 3 proses. Pada proses "Dekompres *File"* (proses 2.1) akan dilakukan proses *uncode*, yaitu menyusun kembali data dari pesan rahasia terkompresi. Pada proses kedua, yaitu proses "Pengambilan *Byte* EOF", akan dilakukan pengambilan *byte* pada akhir *file*, sesuai dengan konsep metode *End of File* yang menyisipkan pesan rahasia di akhir *file*. Proses selanjutnya yaitu proses "Konversi *Byte* ke Bentuk *File*" (Proses 2,3). Pada proses ini *byte* pesan yang sudah dikumpulkan dikonversi kebentuk *file*. *Output* yang dihasilkan adalah sebuah *file* rahasia yang disisipkan.

V. HASIL DAN PEMBAHASAN

A. Implementasi Antarmuka

1) Halaman Utama

Interface halaman utama adalah tampilan yang akan muncul pertama kali setelah program dijalankan. Pilihan menu pada program aplikasi penyembunyi multimedia dapat dlihat pada Gambar 5.1.

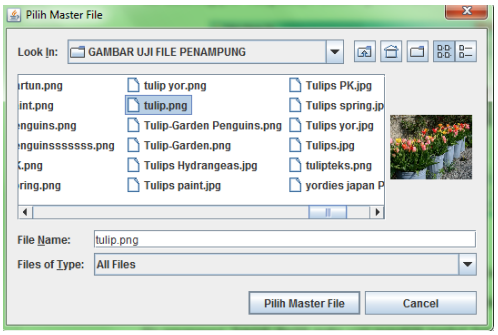
Gambar 5.1. *Interface* Halaman Utama

Dapat dilihat pada gambar 5.1. merupakan tampilan halaman utama pada sistem penyembunyian multimedia ini. Adapun menu pada halaman utama yaitu :

a. Menu Sisip Teks berfungsi untuk menyisipkan pesan teks kedalam suatu media, media yang dimaksud adalah media gambar, media *audio*, atau media *video*.
b. Menu Ekstraksi Teks berfungsi untuk mengekstrak pesan tersembunyi dalam *stegofile*.
c. Menu Sisip *File* berfungsi untuk menyisipkan *file* kedalam suatu media. Media yang dimaksud adalah media gambar, media *audio*, atau media *video*.
d. Menu Ekstraksi *File* berfungsi untuk mengekstrak *file* tersembunyi dalam *stegofile*.
e. Menubar *File*, menubar *file* ini berisi submenu Sisip Teks, Ekstraksi Teks, Sisip *File*, Ekstrakasi *File*, dan Keluar.
f. Menubar Lainnya, menubar lainnya berisi submenu Bantuan.
g. Tombol Bantuan berfungsi untuk menampilkan bantuan yang berisi petunjuk penggunaan aplikasi.

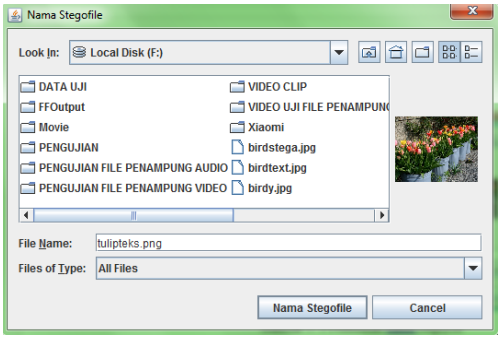
2) *Interface* Penyisipan Pesan Teks

Jika *user* memilih menu Sisip Teks, maka akan tampil pemilihan *Master File*, dan nama *stegofile*. Tampilannya adalah seperti berikut:



Gambar 5.2. Pemilihan *Master File*

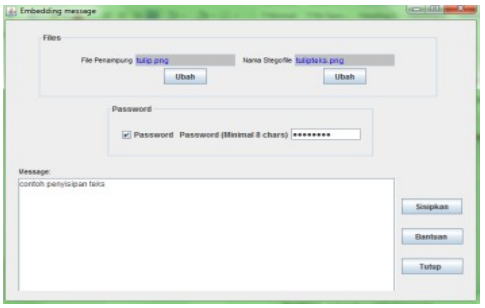
Pada pemilihan *Master File* dengan *interface* seperti pada Gambar 5. 2, *user* dapat memilih media seperti media gambar, media *audio*, atau media *video* sebagai *file* penampung. Setelah dipilih maka *user* mengklik tombol Pilih *Master File*. Maka akan muncul tampilan seperti berikut:



Gambar 5.3. Pemilihan *nama Stegofile*

Pada pemilihan nama *stegofile* seperti pada Gambar 5. 3, user mengetik nama *stegofile* sesuai keinginannya. Lalu *user*

mengklik tombol Nama *Stegofile*. Maka akan muncul layar *embed message* seperti pada Gambar 5. 4 berikut:

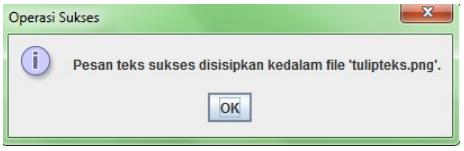


Gambar 5.4. *Interface* Penyisipan Teks

Pada *interface* penyisipan teks seperti pada Gambar 5.4, *user* dapat mengganti *file* penampung dan nama *stegofile* dengan mengklik tombol Ubah. *User* juga dapat melindungi *stegofile* dengan mengaktifkan *checkbox Password* dan mengetik *password* minimal 8 *chars*. Jika *password* yang *user input* kurang dari 8 *chars*, maka akan muncul tampilan seperti pada Gambar 5. 5.

Gambar 5.5. *Interface Invalid Password*

User kemudian mengetik pesan atau meng-*copy paste* pesan dari sumber lain ke teks *area*. Jika *user* mengklik tombol Sisipkan maka sistem akan menyisipkan pesan ke *master file*. Jika penyisipan pesan sukses maka akan muncul tampilan seperti Gambar 5. 6 berikut:

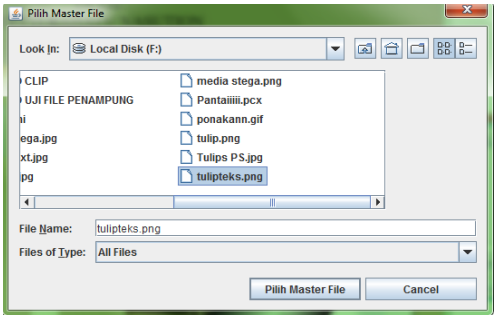


Gambar 5.6. Laporan suskes penyisipan teks

Jika user mengklik tombol Bantuan maka akan muncul layar bantuan yang berisi petunjuk penggunaan aplikasi. Jika *user* mengklik tombol tutup maka layar *embed message* akan ditutup. user mengklik tombol Bantuan maka akan muncul layar bantuan yang berisi petunjuk penggunaan aplikasi. Jika *user* mengklik tombol tutup maka layar *embed message* akan ditutup.

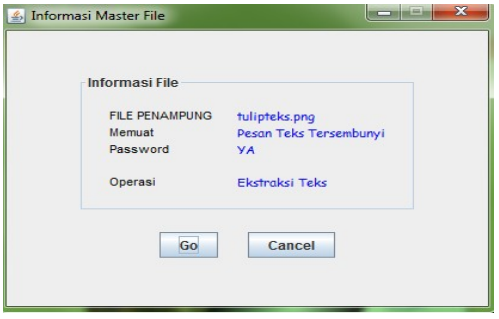
3) *Interface* Ekstraksi Teks

Jika *user* mengklik tombol Ekstraksi Teks, maka sistem akan langsung mengarahkan pada *input filestego*. Tampilannya adalah seperti berikut:



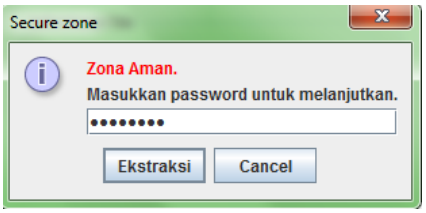
Gambar 5.7. Pemilihan *Stegofile*

Gambar 5.7. merupakan *interface* untuk pemilihan *stegofile*. Jika *user* sudah memilih *stegofile*, maka diklik tombol Pilih *Master File*. Maka akan muncul tampilan informasi tentang *stegofile* seperti pada Gambar 5.8.



Gambar 5.8. Informasi *filestego*

Gambar 5.8. merupakan tampilan dari informasi *stegofile*. Jika *user* ingin menampilkan pesan tersembunyi dalam gambar, *user* mengklik tombol *Go*. Jika *stegofile* tidak dilindungi *password*, maka teks tersembunyi akan dimunculkan ke layar oleh sistem. Tapi karena pada contoh *stegofile* merupakan *file* yang dilindungi *password*, maka sistem akan menampilkan tampilan seperti pada Gambar 5.9. berikut :

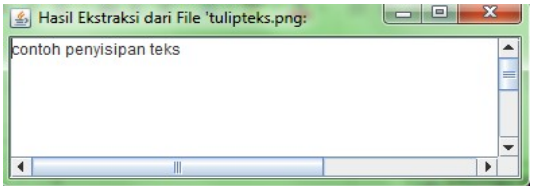


Gambar 5.9. *Interface Input Password*

Jika *password* yang *user* masukkan salah, maka akan muncul tampilan seperti pada Gambar 5. 10.

Gambar 5.10. *Feedback* salah *password*

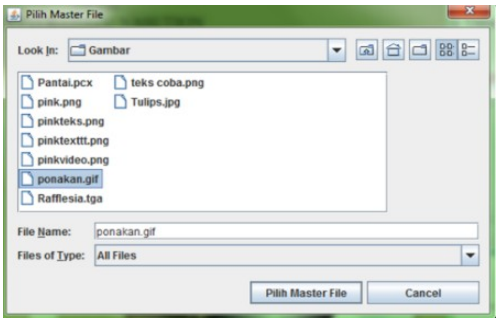
Jika *password* yang *user* masukkan benar, maka pesan tersembunyi akan langsung ditampilkan ke layar.



Gambar 5.11. *Pesan Terekstraksi*

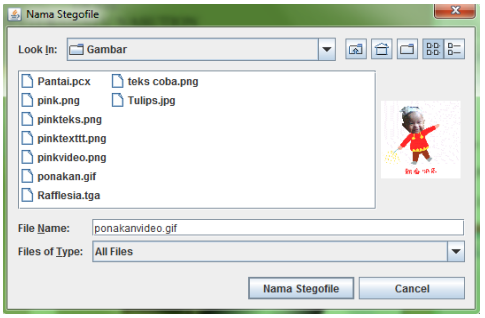
Gambar5 .11.merupakan pesan tersembunyi yang sudah diekstraksi. Jika proses ekstraksi berhasil, maka pesan akan langsung dimunculkan sistem ke layar.

4) *Interface* Sisip *File*



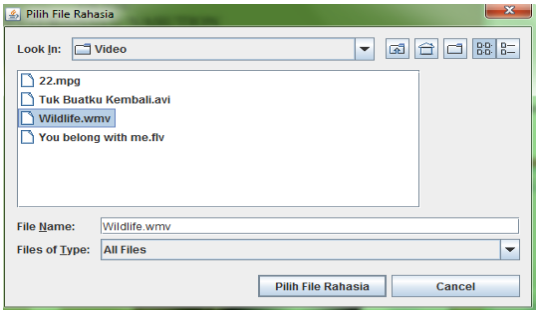
Gambar 5.12. Pemilihan *File* Penampung

Gambar 5.12. merupakan tampilan untuk pemilihan *Master File*. Jika *user* sudah menentukan *master file*, maka diklik tombol Pilih *Master File*. Maka akan muncul tampilan pemilihan nama *stegofile* seperti pada Gambar 5. 13 berikut:



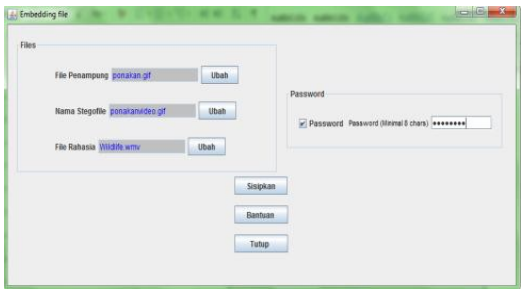
Gambar 5.13. *Input* nama *Stegofile*

Setelah nama *stegofile* di *input*, maka *user* mengklik tombol Nama *Stegofile*. Maka akan muncul tampilan pemilihan *file* rahasia. Misalnya kita ingin memasukkan *video* dengan nama "*Wildlife*.wmv" ke file penampung bernama ponakan.gif.



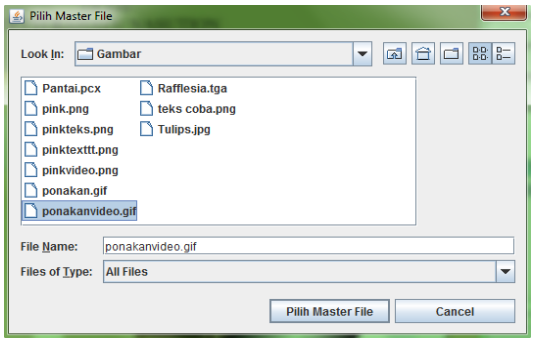
Gambar 5.14. Pemilihan Data *File*

Gambar 5. 14 merupakan pemilihan untuk *file* rahasia. Setelah *user* memilih *file* rahasia, maka diklik tombol Pilih *File* Rahasia. Maka akan muncul tampilan *embed file* seperti Gambar 5. 15.



Gambar 5.15. *Interface Embed File*

Untuk menyisipkan *file* seperti pada Gambar 5. 15, *user* dapat mengubah *file Stegofile* dan *file* rahasia. Untuk lebih jelasnya dapat dilihat pada Gambar 5. 17.



Gambar 5.17. Pemilihan *Stegofile*

penampung, nama *stegofile*, dan *file* rahasia dengan menekan tombol Ubah. *User* juga dapat melindungi *stegofile* dengan mengaktifkan *checkbox password* dan menginputkan *password minimal* 8 *chars*. Untuk menyisipkan *file*, maka *user* mengklik tombol Sisipkan.

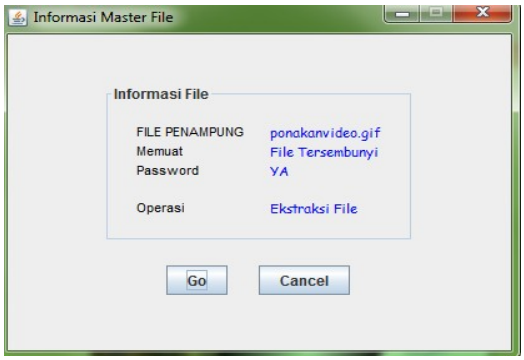
Gambar 5.16. Laporan Sukses *Embed File*

Jika proses penyisipan berhasil, maka akan muncul laporan sukses seperti pada Gambar 5. 16.

5) Ekstraksi *File*

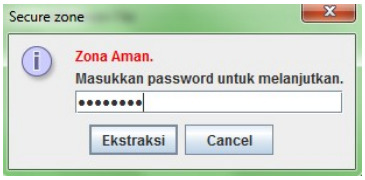
Menu ekstraksi *file* berfungsi untuk mengekstrak *file* tersembunyi pada *stegofile*. Dalam aplikasi pemyembunyian multimedia ini, jika *user* memilih menu *retrieve file*, maka sistem akan langsung mengarahkan *user* untuk memilih *file* penampung, nama

Gambar 5. 17 merupakan *interface* untuk pemilihan *stegofile*. Jika *user* sudah memilih *stegofile*, maka diklik tombol Pilih *Master File*. Maka akan muncul tampilan informasi tentang *stegofile* seperti pada Gambar 5. 18 berikut:



Gambar 5.18. Informasi *Stegofile*

Gambar 5. 18 merupakan tampilan dari informasi *stegofile*. Jika *user* ingin menampilkan pesan tersembunyi dalam gambar, *user* mengklik tombol *Go*. Jika *stegofile* tidak dilindungi *password*, maka *file* tersembunyi akan dimunculkan ke layar oleh sistem. Tapi karena pada contoh *stegofile* merupakan *file* yang dilindungi *password*, maka sistem akan menampilkan tampilan seperti pada Gambar 5. 19.



Gambar 5.19. *Interface Input Password*

Jika *password* yang *user* masukkan salah, maka akan muncul tampilan seperti pada Gambar 5. 20.

Gambar 5.20. *Feedback* jika salah *Password*

Jika *password* yang *user* masukkan benar, maka *file* tersembunyi akan langsung ditampilkan ke layar.

Gambar 5.21. *File* terekstraksi

Gambar 5.21.merupakan *file* tersembunyi yang sudah diekstraksi. Jika proses ekstraksi berhasil, maka pesan akan langsung dimunculkan sistem ke layar.

6) *Interface Form* Bantuan

Tampilan *Form* Bantuan adalah sebagai berikut:

Gambar 5.22. *Interface Form Help*

Form Bantuan seperti yang ditunjukkan pada Gambar 5. 21 berisi tentang petunjuk penggunaan aplikasi.

B. Pengujian Sistem

Pada tahap ini dilakukan pengujian sistem. Pengujian sistem aplikasi ini meliputi pengujian penyisipan dan ekstraksi data uji. Dimana akan dihitung juga rasio kompresi untuk mengetahui seberapa besar pengurangan yang terjadi pada *file* hasil

kompresi jika dibandingkan dengan data aslinya. Semakin rasio kompresi. Berarti semakin banyak pengurangan ukuran pada *file* hasil kompresinya.

1) Pengujian dengan Data Uji Teks

Tabel 5. 1 Pengujian Dengan Data Uji Teks

Ukuran Pesan Rahasia (Kb)	Ukuran File Penampung (Kb)	Hasil Ekstraksi (Kb)	Ukuran (Kb)		Rasio Kompresi (%)
			Filestego	File Terkompresi	
3	747	3	749	2	66,7
12		12	756	9	25
14		14	758	11	21,43
18		18	750	3	83.3
38		38	755	8	78,95
Rata-rata rasio kompresi					55,076

Berdasarkan Tabel 5.1. dapat diketahui bahwa ukuran *file* yang beragam tidak mempengaruhi besarnya rasio kompresi. Hal tersebut terjadi karena metode yang dipakai untuk algoritma kompresinya adalah algoritma *Huffman Coding*. Dimana metode ini memanfaatkan kekerapan kemunculan *string* yang berulang. Dengan kata lain, algoritma *Huffman Coding* mengompresi data melalui besarnya ukuran. Pada Tabel 5.1. dapat dilihat bahwa rata-rata rasio kompresi yang didapat dari 5 pesan rahasia dengan ukuran yang bervariasi yaitu 55,076%.

2) Pengujian dengan Data Uji Gambar

Gambar yang diujikan adalah citra dengan *format* *.tif, *.jpg, *.png, *.bmp, *.gif. Pengujian dengan data uji gambar dapat dilihat pada Tabel 5. 2.

Tabel 5.2. Pengujian Dengan Data Uji Gambar

Format File Gambar	Nama File (Kb))	File Penampung	Ukuran (Kb)	Hasil (Kb)	Perhitungan untuk Kompresi		
					File Asli	File Terkompresi	Rasio (%)
TIF	Me (1.436)	Rose flowers.jpg (540 Kb)	1.941	1.436	1.436	1.401	2,44
	poland (2.170)		2.640	2.170	2.170	2.100	3,23
	gnome (1.399)		1.896	1.399	1.399	1.356	3,07
	Night (14.148)		14.383	14.148	14.148	13.843	2,16
	Shanghai (15.391)		15.594	15.391	15.391	15.054	2,19
Rata-rata Rasio Kompresi							2,618
JPG	152H (1.231)	Rafflesia.jpg (14.763 Kb)	15.852	1.231	1.231	1.089	11,54
	144H (1.408)		16.153	1.408	1.408	1.390	1,28
	224H (1.850)		16.975	1.850	1.850	1.816	1,84
	230H (2.197)		18.304	2.197	2.197	1.947	11.34
	279H (3.604)		16.710	3.604	3.604	3.541	1,75
Rata-rata Rasio Kompresi							5.55
PNG	Angsa (199)	rumah adat.png (747 Kb)	935	199	199	188	5,5
	rose (260)		1.006	260	260	239	8
	kids (441)		1.178	441	441	431	2,3
	(594)		1.322	594	594	575	3,2
	(455)		1180	455	455	455	4,8
Rata-rata Rasio kompresi							4,76
BMP	Barbara (66)	pooh.gif (103 Kb)	160	66	66	57	13,6
	(66)		164	66	66	61	7,6
	Lena2 (66)		161	66	66	58	12,1
	(258)		322	258	258	219	15,1
	Baboon (258)		550	258	258	233	9,7
Rata-rata Rasio kompresi							11,62
	Ponaka n (103)		14.235	103	103	100	2,9
	Swim (158)		14.265	158	158	130	17,7
GIF	Throne (315)	pcx (14.135 Kb)	14.436	315	315	301	4,4
	Dinosaurus (434)		14.559	434	434	424	2,3
	Tiger (1000)		15.119	1000	1000	984	1,6
Rata-rata Rasio kompresi							5,78

Berdasarkan Tabel 5.2. rata-rata rasio kompresi dari data uji gambar adalah sebagai berikut:

Tabel 5.3. Rata-rata Rasio Kompresi Gambar

	Rasio Kompresi (%)
Rata-rata *file* .tif	2,618
Rata-rata *file* .jpg	5,55
Rata-rata *file* .png	4,76
Rata-rata *file* .bmp	11,62
Rata-rata *file* .gif	5,78
Total	30,328
Rata-rata	**6,0656**

Berdasarkan Tabel 5.3. rata-rata rasio kompresi paling besar adalah pada gambar dengan format .bmp, yaitu sebesar 11,62%.

Rasio kompresi 11,62% artinya telah terjadi pengurangan sebanyak 11,62% dari data asli. Kemudian diikuti oleh gambar dengan *format* .gif sebesar 5,78%, *format* .jpg sebesar 5,55%, *format* .png sebesar 4,76%. Rasio kompresi terkecil yaitu pada gambar dengan *format* .tif, yaitu 2,618%. Rata-rata rasio kompresi pada *file* gambar adalah 6,0656%.

3) Pengujian dengan Data Uji *Audio*

Audio yang diuji adalah *audio* dengan *format* *.mp3, *.m4a, *.ogg, *.wav, *.wma dengan ukuran 2MB<audio<3MB

Tabel 5.4. Data Uji Audio

Format	Nama File (Kb)	File	Ukuran (Kb)	Hasil	Perhitungan untuk Kompresi		
					File asli	File Perlindungan	Rasio (%)
MP3	Endeng (2.035)	Gorontalo Tempo dulu (5.998 Kb)	7.984	2.035	2.035	1.986	2,4
	DDAD (2.045)		7.900	2.045	2.045	1.902	6,9
	Ambon (2.182)		8.155	2.182	2.182	2.137	2,1
	Pue-huto (2.482)		8.565	2.482	2.482	2.367	4,6
	Heathens (2.992)		8.962	2.992	2.992	2.864	4,3
Rata-rata Rasio kompresi							4,06
OGG	Don't let me down (2.431)	Suda taleuta (5.567 Kb)	7.917	2.431	2.431	2.350	3,3
	Cake by the ocean (2.604)		8.126	2.604	2.604	2.559	1,7
	Waterloo (2.637)		8.141	2.637	2.637	2.574	2,4
	Just a friend to you (2.759)		8.279	2.759	2.759	2.712	1,7
	Merry (2.800)		8.338	2.800	2.800	2.771	1
Rata-rata Rasio kompresi							2,02
	You don't own me (2.106)	Dulo Samosir (5.460 Kb)	7.504	2.106	2.106	2.044	2,9
	I want to be free (2.314)		7.552	2.314	2.314	2.092	9,6
	Come into my dream (2.542)		7.892	2.542	2.542	2.432	4,3
	Just another night (2.757)		8.207	2.757	2.757	2.647	3,9
	So big (2.782)		8.235	2.782	2.782	2.675	3,9
Rata-rata Rasio kompresi							4,92
WAV	My girl (2.159)	Tano Mandai	2.193	2.159	2.159	1.859	13,9
	Bad blood (2.372)		2.493	2.372	2.372	2.129	10,3
	Crash (2.382)		2.679	2.582	2.582	2.545	1,6
	Don't kick the chair (2.779)		2.822	2.779	2.779	2.488	10,5
	Seven nation army (2.930)		2.842	2.930	2.930	2.508	14,4
Rata-rata Rasio kompresi							10,14
WMA	Tears (2.217)	Bonu pangasonan ape (27.031 Kb)	29.191	2.217	2.217	2.160	2,6
	Someone like you (2.291)		29.264	2.291	2.291	2.233	2,5
	Lay me down (2.667)		29.610	2.667	2.667	2.579	3,3
	Thinking out loud (2.684)		29.659	2.684	2.684	2.628	2,1
	Hasbi Rabbi Jalallah (2.690)		29.627	2.690	2.690	2.596	3,5
Rata-rata Rasio kompresi							2,8

Berdasarkan Tabel 5.4. pada pengujian data dengan format .wav, dapat dilihat bahwa data berhasil disisipkan meskipun ukuran *file* penampung lebih kecil daripada *file* rahasia. Pada Tabel 5.4. juga dapat dilihat bahwa besarnya ukuran data tidak mempengaruhi besarnya rasio kompresi. Rata-rata rasio kompresi dari data uji *audio* dengan ukuran 2MB<audio<3MB dapat dilihat pada Tabel 5.5. berikut :'

Tabel 5.5. Rata-rata rasio kompresi *Audio*

	Rasio Kompresi (%)
Rata-rata *file* .mp3	4,06
Rata-rata *file* .m4a	2,02
Rata-rata *file* .ogg	4,92
Rata-rata *file* .wav	10,14
Rata-rata *file* .wma	2,8
Total	22,98
Rata-rata	**4,788**

Berdasarkan Tabel 5. 5, rasio kompresi yang paling besar adalah pada data *audio* dengan *format* .wav dengan rasio kompresi sebesar 10,14%, diikuti oleh *audio* dengan *format* .ogg sebesar 4,92%, *format*.mp3 sebesar 4,06%, *format* .wrna

sebesar 2,8% serta *audio* dengan *format* .m4a sebesar 2,02%. Rata-rata rasio kompresi pada *file audio* yaitu 4,788%.

4) Pengujian dengan Data Uji Video

Data yang diuji adalah *video* dengan ukuran 1MB<*video*<5MB dengan *format* *.3gp, *.mp4, *.mpg, *.mov, *.avi.

Tabel 5.6. Pengujian dengan Data Uji Video

Format File Rahasia	Nama File Asli (Ukuran (Kb))	File Penampung (Kb)	Ukuran Steofil c (Kb)	Hasil Ekstraksi (Kb)	Perhitungan untuk Kompresi: Ukuran (Kb) File Asli	Ukuran (Kb) File Terkompresi	Rasio (%)
3GP	(1.488)	22 mpg (41.683 Kb)	43.121	1.488	1.488	1.438	3,4
	Red shoes & 7 dwarfs (2.444)		44.084	2.444	2.444	2.401	1,8
	Twilie (2.483)		44.140	2.483	2.483	2.457	1,1
	[illegible] (3.327)		44.908	3.327	3.327	3.285	1,3
	Moana (4.284)		45.850	4.284	4.284	4.147	3,2
Rata-rata Rasio kompresi							2,16
MP4	Vid1 (2.027)	You belong with me.flv (34.983 Kb)	36.979	2.027	2.027	1996	1,5
	Vid2 (3.193)		38.117	3.193	3.193	3.134	1,9
	Vid3 (4.148)		39.072	4.148	4.148	4.089	1,4
	Boss baby (6.414)		41.316	6.414	6.414	6.333	1,3
	Lagu anak2 (3.875)		38.784	3.875	3.875	3.803	1,9
Rata-rata Rasio kompresi							1,6
MPG	Barbie (1.201)	Tuk buatku Jambi.avi (30.118 Kb)	31.176	1.201	1.201	1.058	11,9
	Madagascar (2.160)		31.902	2.160	2.160	1.784	17,4
	Romantico (2.463)		32.509	2.463	2.463	2.391	2,9
	Movie (3.106)		33.105	3.106	3.106	3.047	1,9
	[illegible] (4.314)		34.059	4.314	4.314	3.941	8,7
Rata-rata Rasio kompresi							8,56
MOV	Turma da monica (1.389)	Wildlife.wmv (25.631 Kb	26.959	1.389	1.389	1.328	4,4
	Trechos de filmes (1.390)		26.979	1.390	1.390	1.348	3
	Bocos de madagascar (1.939)		27.519	1.939	1.939	1.888	2,6
	Sheik amor amo (1.970)		27.575	1.970	1.970	1.944	1,3
	Rarbie (4.631)		30.179	4.631	4.631	4.548	1,8
Rata-rata Rasio kompresi							2,62
AVI	Penguin (2.152)	22 mpg (41.683 Kb)	43.790	2.152	2.152	2.107	2,1
	Stoodre (2.221)		43.792	2.221	2.221	2.109	5
	Karla (3.383)		44.033	3.383	3.383	2.950	12,8
	Dami waktu (3.914)		45.551	3.914	3.914	3.868	1,2
	Kavla (4.694)		46.153	4.694	4.694	4.450	5,2
Rata-rata Rasio kompresi							5,26

Pengujian data *video* berdasarkan Tabel 5.6dengan ukuran 1MB<*video*<5MB dapat dilihat bahwa ukuran data tidak mempengaruhi besarnya rasio kompresi. Rasio kompresi pada pengujian *video* ini termasuk rendah. Hal tersebut dikarenakan algoritma kompresi yang dipakai, yaitu *Huffman Coding* lebih cocok mengompresi data dalam bentuk teks.

Rata-rata rasio kompresi video dengan ukuran 1MB<*video*<5MB dapat dilihat pada Tabel 5.7. berikut:

Tabel 5.7. Rata-rata rasio kompresi Video

	Rasio Kompresi (%)
Rata-rata *file* .3gp	2,16
Rata-rata *file* .mp4	1,6
Rata-rata *file* .mpg	8,56
Rata-rata *file* .mov	2,62
Rata-rata *file* .avi	5,26
Total	20,2
Rata-rata	**4,04**

Berdasarkan Tabel 5.7, rasio kompresi yang paling besar adalah pada *video* dengan *format* .mpg, yaitu sebesar 8,56% diikuti oleh *video* dengan *format* .avi dengan rasio 5,26%, *format* .mov 2,62% , *format* .3gp 2,16% serta *format* .mp4 1,6%. Rata-rata rasio kompresi untuk seluruh data uji *video* yaitu 4,04%.

C. Analisa Hasil Pengujian

1) Analisa Hasil Pengujian Berdasarkan Metode *End of File*

Hasil pengujian dapat dilihat pada Tabel 5.1. sampai dengan Tabel 5.7. Metode *End of File* merupakan algoritma

penyisipan yang menyisipkan data akhir di *file*, sehingga ukuran *stegofile* (*file* yang sudah disisipkan data) merupakan ukuran *file* sebelum disisipkan data ditambah dengan ukuran data rahasia atau data yang disisipkan kedalam *file* tersebut. Pada Tabel hasil pengujian, dapat dilihat bahwa ukuran *stegofile* mengalami penambahan ukuran, yaitu ukuran data asli ditambah ukuran data rahasia terkompresi.

Dengan menerapkan metode *End of File* pada sistem, maka data dengan ukuran yang besar dapat disisipkan kedalam *file* penampung dengan ukuran yang lebih kecil. Hasil pengujian ini dapat dilihat pada Tabel 5. 4 pada pengujian *audio* dengan *format* .wav.

Kelebihan metode *End of File* adalah *file* penampung yang telah disisipi pesan tidak berubah. Berikut adalah tampilan *file* asli serta *file* yang sudah disisipi rahasia:

Tabel 5.8. Perbedaan *File* Asli dengan *Stego File*

File penampung	*ile* yang Disisipkan	*Stegofile*
 Rose flowers.jpg (540 Kb)	 gnome.tif (1.399 Kb)	 Rose flowers.jpg (1.896 Kb)
 Rafflesia.tga (14.763 Kb)	PPAP.mp3 (2.045 Kb)	 Rafflesia.tga (16.665 Kb)
 Pink.png (327 Kb)	Moana.3gp (4.284 Kb)	 Pink.png (4.474 Kb)

Perbedaan antara *file* asli dengan *stegofile* dapat dilihat pada Tabel 5. 8. Jika dilihat kualitas *file* asli dengan *stegofile* secara visual, tidak tampak perbedaan diantara keduanya. Hanya saja, ukuran dari *stegofile* bertambah, yaitu ukuran *file* asli ditambah ukuran *file* rahasia terkompresi.

2) Analisa Hasil Pengujian Berdasarkan Metode *Huffman Coding*

Algoritma kompresi data diterapkan dalam aplikasi penyembunyian multimedia ini untuk menutupi kekurangan metode *End of File* yang membuat ukuran *stegofile* menjadi besar, sehingga diharapkan dapat mengurangi kecurigaan bagi pihak yang melihatnya. Tetapi algoritma kompresi yang digunakan, yaitu metode *Huffman Coding*, kurang cocok jika digunakan untuk mengompresi media gambar, *audio*, serta *video*. Metode *Huffman Coding* lebih cocok digunakan untuk media teks. Hal ini dapat dibuktikan dari hasil pengujian pada Tabel 5. 1 sampai dengan Tabel 5. 7. Dari tabel-tabel tersebut, dapat disimpulkan bahwa rasio kompresi media pada masing-masing data uji adalah sebagai berikut:

Tabel 5.9. Analisa Hasil Pengujian

Media	Rata-rata Rasio Kompresi (%)
Teks	55,076
Gambar	6,0656
Audio	4,788
Video	4,04

Berdasarkan Tabel 5. 9, rata-rata rasio kompresi yang paling besar adalah pada media teks dengan rata-rata rasio kompresi sebesar 55,076%. Hal ini sesuai metode kompresi yang dipakai, yaitu media *Huffman Coding* yang memang

lebih cocok untuk media teks. Kemudian diikuti oleh media gambar dengan rata-rata rasio kompresi sebesar 6,0656%, media *audio* 4,788% serta *video* dengan rata- rata rasio terkecil, yaitu 4,04%.

Dari pengujian yang sudah dilakukan juga dapat dilihat bahwa, besarnya ukuran data tidak mempengaruhi rasio kompresi. Hal ini karena metode *Hufman Coding* tidak mengompresi *file* berdasarkan ukuran *file*, melainkan menggunakan prinsip pengkodean dimana tiap karakter (simbol) yang sering muncul dikodekan dengan rangkaian bit yang pendek, dan karakter yang jarang muncul dikodekan.dengan rangkaian bit yang lebih panjang.

VI. KESIMPULAN

Berdasarkan hasil penelitian dan pembahasan yang telah dilakukan, kesimpulan yang dapat diambil adalah:

1. Aplikasi penyembunyian multimedia dengan menggunakan metode *End of File* dan *Huffman Coding* ini berhasil dibangun.
2. Pengujian penyisipan data rahasia seperti teks, *file* citra (*.tif, *.jpg, *.png, *.bmp, *.gif), *file audio* (*.mp3, *.m4a, *.ogg, *.wav, *.wma) dan *file video* (*.3gp, *.mp4, *.mpg, *.mov, *.avi) berhasil disisipkan kedalam *file* penampung, meskipun ukuran data rahasia yang disisipkan lebih besar daripada *file* penampung. Hal ini membuktikan bahwa metode *End of File* berhasil diterapkan.
3. Metode *End of File* efektif digunakan sebagai algoritma penyisipan untuk data dengan ukuran yang besar.
4. Dalam analisa rasio kompresi didapatkan bahwa, besarnya ukuran data tidak mempengaruhi besarnya rasio kompresi. Rata- rata rasio kompresi secara berturut-turut dari yang tertinggi ke yang terendah adalah pada media teks, diikuti media gambar (*.bmp, *.gif, *.jpg, *.png, *.tif), *audio* (*.wav, *.ogg, *.mp3, *.wma, *.m4a) serta *video* (*.mpg, *.avi, *.mov, *.3gp, *.mp4).
5. Pengujian ekstraksi *file* berhasil, yaitu *file* yang disisipkan sebelumnya berhasil diekstraksi tanpa ada perubahan dari ukuran, isi dan kualitas pada *file* tersebut.
6. Algoritma *Huffman Coding* kurang efektif digunakan untuk mengompresi media gambar, *audio*, dan *video*.

VII. SARAN

Berdasarkan hasil penelitian, pengujian serta pembahasan yang dibahas, maka saran untuk pengembangan penelitian dimasa yang akan datang adalah sebagai berikut:

1. Algoritma *Huffman Coding* dapat diganti dengan algoritma *lossless* lain yang dapat memampatkan multimedia dengan lebih baik.
2. Meningkatkan keamanan *stegofile*, dengan cara mengenkripsi *file* rahasia terlebih dahulu sebelum disisipkan.

REFERENSI

[1] Amri, Y. F. (2012). Analisis kinerja kompresi *file audio* menggunakan algoritma *Arithmetic Coding* dengan metode bilangan integer. Skripsi Fakultas Teknik UNS. Surakarta.

[2] Ariyus, Dony. (2006). Kriptografi. Penerbit Graha Ilmu.Yogyakarta.

[3] Muchbarak, A., Harvianto, F.D. (2014). Pendekatan metode *Least Significant Bit* untuk merancang aplikasi *steganography* pada *file* PNG dengan metode *Huffman Code* dalam kompresi pesan. Skripsi Fakultas Teknik Universitas Budi Luhur. Jakarta.

[4] Munir, R. (2004). *Steganography* dan *Watermarking*. Bandung: Institut Teknologi Bandung.

[5] Putra,D.(2010). Pengolahan Citra Digital. Yogyakarta: C.V Andi Offset..

[6] Rosa A.S. & M. Shalahuddin. (2013). Rekayasa Perangkat Lunak. Bandung: Informatika.

[7] Sejati, A. (2007), Studi dan perbandingan steganografi metode EOF *(End of File)* dengan *DCS (Dynamic Cell Spreadin*g). J.Kalbiscentia. 2(1): 77-88.

[8] Tri, Arya (2008). Kode *Huffman*. Skripsi Fakultas Teknik ITB. Bandung.

[9] Westfeld, A., & Pfitzmann, A. (1999, September). *Attacks on steganographic systems.* J. Computer Science. 1768: 61-76.