ANALISIS *RATIO DATA EMBEDDING* DAN SNR (*SIGNAL TO NOISE RATIO)* PADA AUDIO STEGANOGRAFI DENGAN METODE LSB (*LOW SIGNIFICANT BIT)*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh :

ASTRI VALENTINA DAMAYANTI

NIM. 0310630025-63

KEMENTERIAN PENDIDIKAN NASIONAL

UNIVERSITAS BRAWIJAYA

FAKULTAS TEKNIK

JURUSAN TEKNIK ELEKTRO

MALANG

2010

ANALISIS *RATIO DATA EMBEDDING* DAN SNR (*SIGNAL TO NOISE RATIO)* PADA AUDIO STEGANOGRAFI DENGAN METODE LSB (*LOW SIGNIFICANT BIT)*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh :

ASTRI VALENTINA DAMAYANTI

NIM. 0310630025-63

Telah diperiksa dan disetujui oleh
Dosen Pembimbing :

Ir. Endah Budi P., MT.	**Rusmi Ambarwati, ST., MT.**
NIP. 19621116 198903 2 002	**NIP. 19720204 200003 2 002**

ANALISIS *RATIO DATA EMBEDDING* DAN SNR (*SIGNAL TO NOISE RATIO)* PADA AUDIO STEGANOGRAFI DENGAN METODE LSB (*LOW SIGNIFICANT BIT)*

Disusun oleh :

ASTRI VALENTINA DAMAYANTI

NIM. 0310630025-63

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal 04 Agustus 2010

Majelis Penguji :

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

Dwi Fadila K., ST., MT
NIP. 19720630 200003 1 002

Ali Mustofa, ST., MT
NIP. 19710601 200003

Mengetahui :
Ketua Jurusan Teknik Elektro

Rudy Yuwono, ST., M.Sc
NIP. 19710615 199802 1 003

KATA PENGANTAR

Puji Syukur kehadirat Allah SWT karena dengan berkat rahmat dan karunia serta ridho-Nya penyusunan skripsi ini dengan judul "Analisis *Ratio Data Embedding dan* SNR (*Signal to Noise Ratio)* pada Audio Steganografi dengan Metode LSB (*Low Significant Bit)*" dapat diselesaikan. Penulis menyadari bahwa kajian ini tidak akan mencapai titik akhir penyelesaian tanpa bantuan berbagai pihak, karenanya penulis mengucapkan terima kasih kepada:

1. Keluarga besar Bapak Pitono untuk seluruh do'a dan dukungannya yang telah diberikan kepada Ananda selama studi hingga terselesaikannya skripsi ini.
2. Bapak Rudy Yuwono, ST, MSc. selaku Ketua Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya.
3. Bapak M. Aziz Muslim, ST., MT., Ph.D selaku Sekretaris Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya.
4. Ibu Ir. Endah Budi P., MT selaku dosen pembimbing pada penyusunan skripsi ini.
5. Ibu Rusmi Ambarwati, ST., MT selaku dosen pembimbing pada penyusunan skripsi ini.
6. Bapak dan Ibu dosen serta karyawan Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya.
7. Teman-teman Teknik Elektro Universitas Brawijaya, khususnya Angkatan 2003 yang telah memberi kenangan yang tak terlupakan dan dukungannya.
8. Serta semua pihak yang tak dapat disebutkan satu persatu yang telah turut membantu baik secara langsung maupun tidak langsung dalam penyelesaian skripsi ini.

Semoga Allah SWT memberikan balasan kebahagiaan dan kesuksesan atas segala budi baik yang telah diberikan kepada penulis.

Tiada yang sempurna di dunia ini, tersadar bahwa skripsi ini sangat jauh dari kesempurnaan. Karenanya, segala kritik dan saran yang sifatnya membangun dari pembaca tentang isi skripsi ini akan diterima dengan senang hati. Akhir kata, penulis berharap, semoga skripsi ini dapat bermanfaat bagi semua pihak yang membutuhkan.

Malang, Juli 2010

Penulis

DAFTAR ISI

KATA PENGANTAR ..... **i**
DAFTAR ISI ..... **ii**
DAFTAR GAMBAR ..... **iv**
DAFTAR TABEL ..... **v**
DAFTAR ISTILAH ..... **vi**
ABSTRAK ..... **vii**
BAB I PENDAHULUAN **1**
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Batasan Masalah 2
1.4 Tujuan 2
1.5 Manfaat 2
1.6 Sistematika Penulisan 3
BAB II DASAR TEORI **4**
2.1 Pengertian Kriptografi 4
2.1.1 Aspek-Aspek Keamanan Kriptografi 4
2.2 Pengertian Steganografi 5
2.2.1 Sejarah Steganografi 7
2.2.2 Kriteria Steganografi yang Bagus 7
2.2.3 Penyembunyian Informasi dalam *File* Audio 8
2.2.4 Metode Steganografi 9
2.2.4.1 Metode Steganografi pada Teks 9
2.2.4.2 Metode Steganografi pada Suara/ Audio 11
2.2.4.3 Metode Steganografi pada Gambar 13
2.3 MPEG-1 layer 3 atau MP3 16
2.3.1 Kualitas Audio 18
2.3.2 *Sampling rate* 19

2.4 *File* data. TXT.....22
2.5 SNR (*Signal to Noise Ratio*)…….....…23
BAB III METODE PENELITIAN.....24
3.1 Jenis Data dan Cara Mendapatkan Data.....24
3.2 Variabel dan Cara Analisis Data 24
3.3 Rangka Solusi Permasalahan.....26
BAB IV PERENCANAAN DAN PERANCANGAN PERANGKAT LUNAK 27
4.1 Metode Perencanaan dan Perancangan Steganografi 29
4.1.1 Prinsip Kerja LSB…...…….29
4.1.2 Model Perancangan perangkat Lunak.....30
BAB V PENGUJIAN DAN ANALISA DATA 38
5.1 Hasil Simulasi.....38
5.2 Hasil Pengujian dan Analisa.....39
5.2.1 Data Hasil Pengujian Untuk *Ratio Data Embedding*.....40
5.2.2 Data Hasil Pengujian Untuk Signal to Noise Ratio.....42
5.2.3 Pengujian Tingkat Keberhasilan Steganografi.....45
BAB VI PENUTUP 46
6.1 Kesimpulan.....46
6.2 Saran 46
DAFTAR PUSTAKA.....47
LAMPIRAN…….....48

DAFTAR GAMBAR

Halaman

Gambar 2.1 Kriptografi/steganografi konvensional 5
Gambar 2.2 Klasifikasi dari teknik steganografi 6
Gambar 2.3 Sherlock holmes dancing men semagrams 7
Gambar 2.4 Metode LSB 11
Gambar 2.5 Bit Rate 17
Gambar 2.6 *Frame header* MP3 18
Gambar 2.7 Sinyal analog 19
Gambar 2.8 Sinyal analog hasil sampling 20
Gambar 3.1 Diagram alir perhitungan jumlah bit yang disisipkan dan jumlah total bit *file* pembawa 26
Gambar 3.2 Diagram alir perhitungan *Ratio Data Embedding* 27
Gambar 3.3 Diagram alir perhitungan nilai fungsi transfer kabel tembaga 27
Gambar 3.4 Diagram alir perhitungan SNR 28
Gambar 4.1 Diagram alir LSB 29
Gambar 4.2 Blok diagram stegano 30
Gambar 4.3 Blok diagram proses *encoding* 31
Gambar 4.4 Blok diagram proses *decoding* 31
Gambar 4.5 Diagram alir proses *encoding* 32
Gambar 4.6 Diagram alir proses *decoding* 33
Gambar 4.7 Pengambilan data *carrier* atau *file* MP3 35
Gambar 4.8 Pengambilan data rahasia atau *file* teks yang akan disisipkan 36
Gambar 4.9 Penyimpanan data MP3 yang telah disisipi atau MP3Stego 36
Gambar 4.10 Pengambilan data MP3 yang akan di *decode* 37
Gambar 4.11 Penyimpanan data rahasia atau data yang telah disisipkan 37
Gambar 4.12 Memasukkan kata kunci atau *key/password* 37
Gambar 5.1 Tampilan hasil proses *encoding* yang menunjukkan banyaknya frame 38
Gambar 5.2 Tampilan hasil proses *encoding* yang menunjukkan bit data yang disisipkan 39
Gambar 5.3 Tampilan hasil proses *encoding* untuk data 4(496kb) 40

Gambar 5.4 Grafik *Ratio Data Embedding*.....41
Gambar 5.5 Grafik *Signal to Noise Ratio* 45

DAFTAR TABEL

Halaman

Tabel 2.1 Tabel ASCII 12
Tabel 2.2 Data biner sebelum disisipi 12
Tabel 2.3 Data biner setelah disisipi data 12
Tabel 2.4 *Sampling rate* dan penggunaannya 20
Tabel 5.1 Ratio Data Embedding 41
Tabel 5.2 Signal to Noise Ratio 44

DAFTAR ISTILAH

Kriptografi	Kriptografi adalah seni dan ilmu untuk menjaga keamanan dan kerahasiaan pesan/informasi.[wikipedia.org].
Plaintext	*Plaintext* adalah pesan yang dapat dibaca.
Ciphertext	*Ciphertext* adalah pesan yang tidak dapat dibaca.
Steganografi	Steganografi adalah teknik menyembunyikan data rahasia dalam wadah (media) digital sehingga keberadaan data rahasia tersebut tidak diketahui oleh orang.
Enkripsi	Enkripsi adalah proses pengubahan pesan asal menjadi karakter yang tidak dapat dibaca.
Deskripsi	Deskripsi adalah proses pengubahan karakter yang tidak dapat dibaca menjadi pesan asal.
LSB	*Low Significant Bit* adalah metode audio steganografi yang menyisipkan data rahasia dengan mengganti bit terakhir data penampungnya.
Phase Coding	Adalah metode yang digunakan dengan merekayasa fasa dari sinyal masukan.
Spread Spectrum	Adalah metode penyebaran spektrum. Pesan akan dikodekan dan disebar ke setiap spektrum frekuensi yang memungkinkan.
Echo Hiding	Adalah teknik menyamarkan pesan kedalam sinyal yang membentuk *echo*.
Bit Rate	Adalah banyaknya bit per second

ABSTRAK

ASTRI VALENTINA DAMAYANTI, Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya, Juli 2010, "Analisis *Ratio Data Embedding* dan SNR (*Signal to Noise Ratio*) pada Audio Steganografi dengan metode LSB (*Low Significant Bit*)", Dosen Pembimbing : Ir. Endah Budi P, MT dan Rusmi Ambarwati, ST., MT.

Perkembangan teknologi di bidang teknologi komunikasi dan informasi telah berkembang secara cepat dan luas. Sehingga memungkinkan kita untuk menyimpan dan menyebarkan informasi secara bebas dan luas. Salah satu contohnya adalah dengan adanya internet yang membawa perubahan yang sangat besar dalam distribusi media digital. Setiap teknologi memiliki nilai positif dan negatif, salah satunya adalah dengan adanya internet kita dapat menyebarkan dan mendapatkan informasi dengan mudah. Hal inilah yang menuntut adanya pengamanan data tersebut sehingga tidak sampai tersadap oleh pihak ketiga. Audio steganografi adalah teknik menyembunyikan data rahasia dalam wadah (media) digital dalam hal ini adalah *file* audio (MP3) sehingga keberadaan data rahasia tersebut tidak diketahui oleh orang lain. *Ratio Data Embedding* adalah *ratio* untuk menghitung kemungkinan disisipkannya 1 bit data rahasia pada *file* audio. SNR merupakan *ratio* dari sinyal *power* terhadap *noise power*. Pada perancangan audio steganografi ini menggunakan metode LSB (*Low Significant Bit)*. Dengan menggunakan *file* teks format txt.

Tingkat keberhasilan perangkat lunak yang dibuat pada saat melakukan proses *encoding* dan *decoding* mencapai 100 %. Untuk *file* MP3 2.28 MB dengan *file* data rahasia yang berbeda yaitu data1 (1kb), data2 (10kb), data3 (20kb) didapatkan nilai RDE yang semakin menurun yaitu 1/2280 bit, 1/228 bit, 1/114 bit. Sedangkan untuk nilai SNR data1, data2, dan data3 adalah sama yaitu 29.84 dB.

Kata kunci : Audio steganografi, RDE, SNR, LSB.

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi di bidang teknologi komunikasi dan informasi telah berkembang secara cepat dan luas. Sehingga memungkinkan kita untuk menyimpan dan menyebarkan informasi secara bebas dan luas. Salah satu contohnya adalah dengan adanya internet yang membawa perubahan yang sangat besar dalam distribusi media digital. Media digital dapat diklasifikasikan dalam bentuk teks, citra (gambar), audio dan juga video yang dapat didistribusikan lewat jaringan internet. [Rinaldi Munir, 2004].

Setiap teknologi memiliki nilai positif dan negatif, salah satunya adalah dengan adanya internet kita dapat menyebarkan dan mendapatkan informasi dengan mudah. Selain itu perkembangan dunia digital saat ini membuat lalu lintas pengiriman pesan atau data semakin pesat. Data yang dipertukarkan pun bervariasi baik dari jenisnya maupun tingkat kerahasiaannya. Mulai dari data pribadi, data organisasi sampai data negara yang sangat rahasia. Hal inilah yang menuntut adanya pengamanan data tersebut sehingga tidak sampai tersadap oleh pihak ketiga. Oleh karena itu, steganografi dibutuhkan untuk mengamankan informasi yang dikirimkan. Dimana steganografi membutuhkan dua properti yaitu wadah penampung dan data rahasia yang akan disembunyikan. Diharapkan pengguna steganografi dapat mengamankan data informasi yang akan dikirim dan dalam hal perlindungan hak cipta (*copyright*).

Saat ini kepopuleran dari MP3 sudah sangat meluas dan mendunia. Format kompresi audio MP3 saat ini menjadi yang terpopuler walaupun sudah terdapat jenis kompresi audio yang jauh lebih baik (dalam kapasitas) dan memiliki kualitas jauh lebih baik. Kepopuleran dari MP3 ini tidaklah heran jika digunakan untuk aplikasi keamanan informasi. Teknik steganografi menyamarkan keberadaan dari pesan yang ingin disampaikan, berbeda dengan teknik kriptografi yang dengan mudah dideteksi keberadaannya (walaupun sulit untuk dimengerti).

Salah satu teknik yang digunakan adalah menggunakan *file* dalam format audio yang dapat disisipi pesan yang ingin disampaikan. Teknik ini mungkin untuk dilakukan karena sifat dari *file* audio yang berlebihan (*redundant*) sehingga dengan teknik

pengkompresian menggunakan MP3 dapat menghilangkan informasi yang tidak signifikan bila dihilangkan. Sehingga dengan teknik ini pesan atau data dapat disisipkan pada *file* ini dengan mengganti informasi yang tidak dibutuhkan pada kompresi dengan data tersebut.

Metode *Low Significant Bit* (LSB) adalah metode steganografi yang digunakan untuk media audio dengan cara mengganti bit terendahnya dengan bit data yang akan disisipkan.

1.2 Rumusan Masalah

Berdasarkan pada permasalahan yang telah dijelaskan pada bagian latar belakang, maka rumusan masalah yang diberikan adalah:

- Bagaimana pengaruh penyisipan data rahasia terhadap kualitas data *file* yang dikirimkan dengan cara menghitung SNR (*Signal to Noise Ratio*) dan *Ratio Data Embedding*?

1.3 Batasan Masalah

Batasan permasalahan yang diajukan dalam penyusunan skripsi ini adalah:

- *File carrier*/ *file* pembawa yang digunakan adalah *file* dengan format MP3
- *File* data yang digunakan dalam bentuk *txt
- Analisa hanya difokuskan pada kualitas dan besarnya *file output*/ keluaran
- Menggunakan bahasa pemrograman java J2SDK

1.4 Tujuan

Tujuan dalam penelitian skripsi ini adalah untuk mengetahui dan memahami tentang audio steganografi dan melakukan perancangan program aplikasi audio steganografi dengan menggunakan bahasa pemrograman java serta dapat menerapkan metode *Low Significant Bit* (LSB).

1.5 Manfaat

1. Bagi penulis

- Penulis dapat merancang suatu aplikasi perangkat lunak audio steganografi menggunakan metode LSB.
- Dapat menerapkan ilmu yang diperoleh dari perkuliahan

2. Bagi pembaca

Dapat mengetahui dan memahami tentang perancangan audio steganografi menggunakan metode *Low Significant Bit* (LSB)

1.6 Sistematika Penulisan

Sistematika penulisan dan gambaran yang terdapat dalam laporan skripsi ini terdiri atas 6 bab yaitu Pendahuluan pada bab pertama yang berisi tentang latar belakang, rumusan masalah, ruang lingkup, tujuan penulisan dan sistematika penulisan. Bab kedua adalah Dasar Teori membahas mengenai teori-teori yang mendukung dalam perancangan dan pembuatan audio steganografi. Bab ketiga yaitu Metodologi Penelitian membahas tentang metode yang digunakan dalam menyelesaikan permasalahan yang dibahas dalam skripsi ini. Bab keempat berisi tentang perencanaan dan perancangan perangkat lunak untuk menyisipkan data teks kedalam media *file* pembawa. Bab kelima membahas tentang pengujian terhadap perangkat lunak serta analisis dari data yang telah dihasilkan. Dan bab keenam membahas kesimpulan dan saran-saran untuk pengembangan aplikasi berikutnya.

BAB II
DASAR TEORI

2.1 Pengertian Kriptografi

Kriptografi adalah seni dan ilmu untuk menjaga keamanan dan kerahasiaan pesan/informasi.[wikipedia.org]. Pesan atau informasi yang dapat dibaca disebut sebagai *plaintext* atau *chipertext.* Teknik untuk membuat pesan tidak dapat dibaca disebut sebagai enkripsi. Pesan yang tidak dapat dibaca disebut dengan *chipertext.* Proses yang merupakan kebalikan dari enkripsi disebut dengan deskripsi. Jadi deskripsi akan membuat *chipertext* menjadi *plaintext.*[Ir. Yusuf Kurniawan,1]

Plaintext dinyatakan dengan M (*Message*) atau P(*Plaintext).* Pesan dapat berupa aliran bit, *file* teks, *bitmap*, aliran suara yang digitasi, gambar video digital dan sebagainya. Namun apabila semua pesan tadi berbentuk digital, maka dapat dianggap sebagai data biner. Dan data biner juga dapat diperlakukan sebagai sistem bilangan biner. Karena itulah diperlukan matematika untuk menganalisanya. *Plaintext* dapat disimpan maupun dikirim ke jaringan. Dalam sembarang kasus. M merupakan pesan yang akan dienkrip.[Ir. Yusuf Kurniawan,2]

2.1.1 Aspek –Aspek Keamanan Kriptografi

Kriptografi tidak hanya memberikan kerahasiaan dalam telekomunikasi, namun juga memberikan komponen-komponen berikut ini:

1. *Authentication* **:**

   Penerima pesan dapat memastikan keaslian pengirimnya. Penyerang tidak dapat berpura-pura sebagai orang lain

2. *Integrity* **:**

   Penerima harus dapat memeriksa apakah pesan telah dimodifikasi di tengah jalan atau tidak. Seorang penyusup seharusnya tidak dapat memasukkan tambahan kedalam pesan, mengurangi atau mengubah pesan selama data berada di perjalanan.

3. *Nonrepudiation* **:**

   Pengirim seharusnya tidak dapat mengelak bahwa dialah pengirim pesan yang sesungguhnya. Tanpa kriptografi, seseorang dapat mengelak bahwa dialah pengirim email yang sesungguhnya.

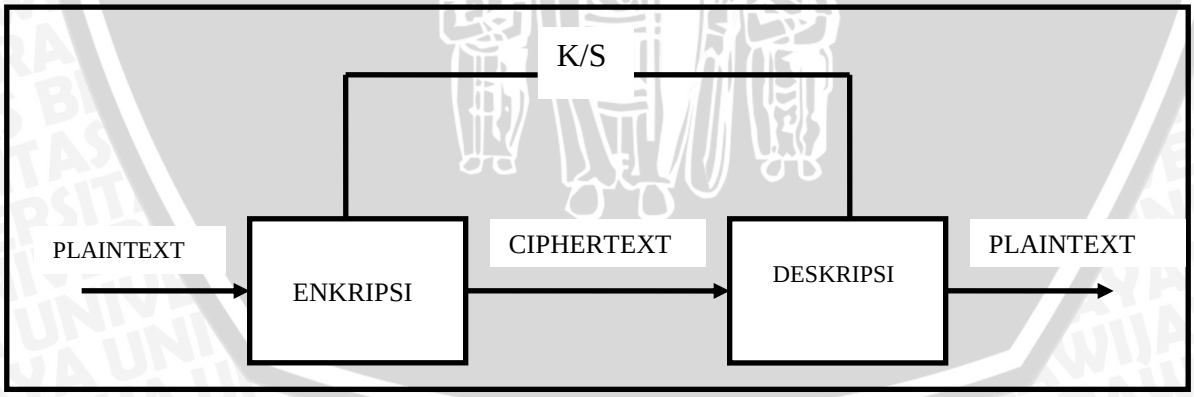
4. *Authority* **:**

Informasi yang berada dalam sistem jaringan seharusnya hanya dapat dimodifikasi oleh pihak yang berwenang. Modifikasi yang tidak diinginkan, dapat berupa penulisan tambahan pesan, pengubahan isi, pengubahan status, penghapusan, pembuatan pesan baru (pemalsuan), atau menyalin pesan untuk digunakan kemudian oleh penyerang.

2.2 Pengertian Steganografi

Steganography (steganografi*)* adalah teknik menyembunyikan data rahasia di dalam wadah (media) digital sehingga keberadaan data rahasia tersebut tidak diketahui oleh orang. Steganografi membutuhkan dua properti yaitu adalah penampung dan data rahasia yang akan disembunyikan. Steganografi digital menggunakan media digital sebagai wadah penampung, misalnya citra, suara (audio), teks, dan video. Data yang disembunyikan juga dapat berupa citra, suara (audio), teks, atau video [Rinaldi Munir, 2004].

Penggunaan steganografi antara lain bertujuan untuk menyamarkan eksistensi (keberadaan) data rahasia sehingga sulit dideteksi, dan melindungi hak cipta suatu produk. Steganografi dapat dipandang sebagai kelanjutan dari kriptografi. Jika pada kriptografi, data yang telah disandikan (*ciphertext*) tetap tersedia, maka dengan steganografi *chipertext* dapat disembunyikan sehingga pihak ketiga tidak mengetahui keberadaannya. Data rahasia yang disembunyikan dapat diekstrasi kembali persis sama seperti keadaan aslinya.



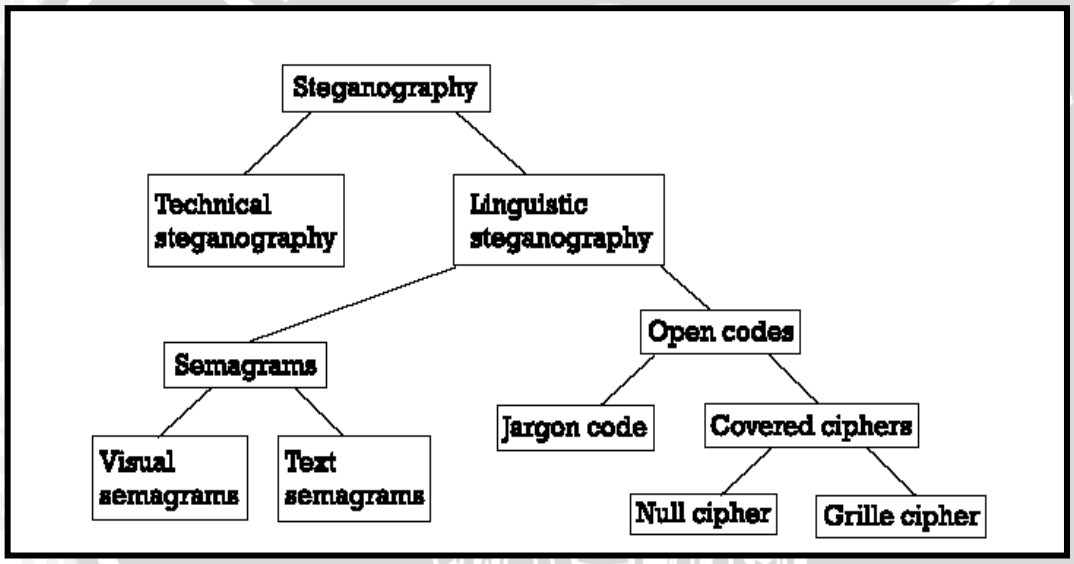
Gambar 2.1 Kriptografi/ steganografi konvensional

Sumber : (Yusuf, 2004: 7)

Pada gambar diatas pesan asli yang belum dikodekan disebut sebagai *plaintext* . Plainteks tidak harus berupa teks, namun dapat berupa *file* gambar (gif, jpeg), *file* biner

(exe,com,ocx) *file* suara (wav, MP3) dan sebagainya. *File* yang telah disandikan disebut *ciphertext.* Enkripsi adalah proses pengubahan pesan asal menjadi karakter yang tidak dapat dibaca. Sedangkan deskripsi adalah proses pengubahan karakter yang tidak dapat dibaca menjadi pesan asal. *Ciphertext* inilah yang biasanya dikirimkan melalui saluran internet yang rawan penyadapan.

Pengertian tentang steganografi pertama kali dicetuskan oleh Johannes Trithemius pada tahun 1499 dengan nama *steganographia*. Namun belum diketahui perbedaan yang jelas antara kriptografi dan steganografi. Pada steganografi klasik dibagi 2 kawasan area yaitu *technical* steganografi dan *lingustic* steganografi yang digambarkan seperti di bawah ini :



Gambar 2.2 Klasifikasi dari teknik steganografi

Sumber : (Michael, 2003: 17)

1. *Technical* steganografi melibatkan pengertian teknik untuk mengetahui keberadaan dari pesan dengan menggunakan pengertian fisika atau kimia. Salah satu contohnya adalah *invisible ink.* Bangsa Romawi kuno mengenal steganografi dengan menggunakan tinta tak tampak (*invisible ink)* untuk menuliskan pesan. Tinta tersebut dibuat dari campuran sari buah, susu, dan cuka. Jika tinta digunakan untuk menulis maka tulisannya tidak tampak. Tulisan diatas kertas dapat dibaca dengan cara memanaskan kertas tersebut [Rinaldi, 2004: 210].

2. *Linguistic* Steganografi sendiri juga dapat dibagi dalam 2 grup yaitu *open codes* dan *semagrams.* Contoh dari *semagrams* termasuk adalah penempatan gambar pada *chessboard* dan lukisan dari *dancing men* pada Doyle's "*The adventure of the Dancing Men*". Seperti yang ditunjukkan pada gambar di bawah ini :

Gambar 2.3 ***Sherlock holmes dancing men semagrams***

Sumber : (Michael, 2003: 18)

2.2.1 Sejarah Steganografi

Steganografi sudah dikenal sejak jaman dahulu, seperti bangsa Yunani dan Bangsa romawi. Contohnya adalah sebagai berikut :

1. Penguasa Yunani dalam mengirim pesan rahasia menggunakan kepala budak atau prajurit sebagai media. Dalam hal ini, rambut budak dibotaki, lalu pesan rahasia ditulis pada kulit kepala budak. Ketika rambut budak tumbuh, budak tersebut diutus untuk membawa pesan rahasia di kepalanya.
2. Metode lain yang digunakan oleh bangsa Yunani kuno adalah dengan menggunakan lilin. Pesan ditulis dalam selembar kertas lalu ditutup dengan lilin. Jika pihak penerima ingin membaca isi pesan tersebut maka dia harus menghilangkan lapisan lilin terlebih dahulu.
3. Kemudian selama Perang Dunia II, militer Jerman memiliki teknologi *microfiche* untuk membuat *microdots*. *Microdots* terdiri atas gambar-gambar dan pesan-pesan teks yang ukurannya mengkerut terhadap suatu periode dan digunakan di dalam teks dari sebuah tulisan biasa atau memorandum. Banyak dari pesan-pesan tersebut yang lolos dari deteksi pasukan sekutu. Steganografi mengalami kemajuan yang sangat pesat sejak tahun 1990-an ketika pemerintah, industri, warga negara biasa bahkan organisasi teroris mulai menggunakan aplikasi perangkat lunak steganografi untuk menyembunyikan pesan-pesan atau foto kedalam beberapa tipe media.

2.2.2 Kriteria Steganografi yang Bagus

Kriteria yang harus diperhatikan dalam penyembunyian data adalah :

1. *Fidelity* : Mutu citra penampung tidak jauh berubah. Setelah penambahan data rahasia, citra hasil steganografi masih terlihat dengan baik. Pengamat tidak mengetahui kalau di dalam citra tersebut terdapat data rahasia.
2. *Robustness* : Data yang disembunyikan harus tahan (*robust*) terhadap berbagai operasi manipulasi yang dilakukan pada citra penampung, seperti pengubahan kontras, penajaman, pemampatan, rotasi, perbesaran gambar, pemotongan (*cropping*), enkripsi, dan sebagainya. Bila pada citra penampung dilakukan operasi-operasi pengolahan citra tersebut, maka data yang disembunyikan seharusnya tidak rusak (tetap valid jika diekstraksi kembali)
3. *Recovery*: Data yang disembunyikan harus dapat diungkapkan kembali (*reveal*). Karena tujuan steganografi adalah data hiding, maka sewaktu-waktu data rahasia di dalam citra penampung harus dapat diambil kembali untuk digunakan lanjut.

2.2.3 Penyembunyian Informasi dalam *File* Audio

Bender et al mengungkapkan penyembunyian informasi dalam sinyal audio sebagai tantangan khusus. Hal ini disebabkan karena fakta bahwa *Human Auditory System* (HAS) berjalan melebihi jangkauan dinamis yang luas. Tetapi sebagaimana yang didiskusikan dalam paper masih terdapat beberapa kemungkinan untuk mengungkap sebagian "*holes*" yang tersedia. Terlebih dahulu untuk menyembunyikan informasi dalam
data audio tidak hanya cukup untuk mengingat dari sensitivitas HAS, namun juga fakta bahwa sinyal audio berjalan diantara *encoding* dan *decoding*. Sinyal audio tersebut juga
dapat berjalan melalui sebuah media penyimpan atau ditransmisikan melalui suatu media. Ketika data audio direpresentasikan secara digital, metode kuantisasi *sample* dan penilaian *sampling temporal* akan menjadi faktor penting. Beberapa teknik dipresentasikan oleh Bender et al untuk menyembunyikan informasi melalui data audio
meliputi *low bit encoding* dan *Phase coding*. Dalam *low bit encoding* informasi disimpan dengan mengubah *Least Siginificant Bit* (LSB) dari masing-masing *sampling point* menggunakan sebuah kode *string* biner. Hasil ini dalam jumlah informasi yang besar dapat diencode dalam sebuah *single* data audio. Sebagai contoh apabila *channel* tanpa suara idealnya berkapasitas I Kbps maka nilai bit yang diberikan sebesar 8 Kbps untuk 8Khz *sequence sample*. Sementara sebagai cara termudah untuk menyembunyikan informasi dalam data audio, skema *low bit encoding* dapat dihancurkan dengan *noise channel* dan *re-*

sampling. *Phase coding* telah terbukti sebagai teknik *coding* yang paling efektif dalam kasus sinyal ke *ratio noise*. Dalam metode ini fasa dari sinyal audio asli akan diubah dengan referensi fasa dari informasi yang akan disembunyikan. Bender et al menemukan bahwa sebuah kapasitas *channel* sekitar 8 bps dapat dicapai dengan mengalokasikan 128 slot frekuensi per bit dengan *background noise* yang minimum. Bender et al mendiskusikan metode untuk memperbaiki sinyal audio *encoding* dibawah *channel* komunikasi yang berbeda.

2.2.4 Metode Steganografi

Terdapat banyak metode yang digunakan dalam melakukan penyembunyian data ke dalam data lainnya. Berikut adalah penjelasan mengenai beberapa metode yang banyak digunakan dalam steganografi.

2.2.4.1 **Metode Steganografi pada Teks :**

1. Metode Spasi Terbuka

Terdapat beberapa cara untuk memanfaatkan spasi terbuka dalam data teks guna menyembunyikan informasi. Metode ini dapat berhasil karena buku bacaan pada umumnya menambahkan satu spasi tambahan pada akhir baris atau diantara dua kata sehingga tidak terbaca aneh. Bagaimanapun, metode spasi terbuka hanya dapat digunakan dengan memakai ASCII (*American Standard Character Interchange*) format. Bender et al memberikan tiga metode untuk mengungkap *white space* dalam proses penyembunyian. Spasi terbuka antar kalimat akan menghasilkan nilai "0" apabila hanya terdapat sebuah spasi yang ditambahkan diantara kalimat tersebut. Dengan menambahkan dua spasi akan menghasilkan nilai "1". Metode ini dapat berhasil, tetapi membutuhkan data dalam jumlah besar untuk menyembunyikan sebuah informasi kecil. Dan juga terdapat banyak *software word processing* yang akan secara otomatis membetulkan spasi antara kalimat, sehingga metode ini seringkali gagal. Metode spasi *end of line* (EOL) mengutarakan *white space* pada akhir dari masing-masing baris. Data disembunyikan menggunakan jumlah spasi yang telah ditentukan sebelumnya dari akhir untuk masing- masing kalimat. Sebagai contoh dua spasi akan menyembunyikan satu bit, empat spasi akan menyembunyikan dua bit dan delapan spasi akan menghasilkan tiga bit dan seterusnya. Teknik ini lebih baik dibandingkan metode spasi terbuka antar kalimat, karena dengan meningkatkan jumlah spasi akan dapat menyembunyikan lebih banyak data. Salah satu kekurangan dari teknik

ini adalah dapat hilangnya informasi tersembunyi jika *hardcopy* data yang diberikan Pada akhirnya, pemerataan kanan dari teks dapat digunakan pula untuk menyembunyikan informasi rahasia pada data teks. Penghitungan dan pengontrolan spasi diantara kata dapat menyembunyikan informasi dalam data teks yang terlihat tidak penting. Sebuah spasi antara kata akan menghasilkan nilai "0" dan dua buah spasi akan menghasilkan nilai "1". Bagaimanapun, pendekatan ini akan mempersulit

untuk mengeluarkan informasi penting dari media data teks tersebut karena akan semakin tidak mungkin untuk membedakan sebuah spasi biasa dengan spasi yang berfungsi untuk penyembunyian data. Untuk mewujudkan hal ini, Bender et al menggunakan *Manchester coding* untuk mengelompokkan bit-bit. Sehingga "01" diinterpretasikan sebagai "1" dan "10" diinterpretasikan sebagai "0". Dimana "00" dan "11" akan dianggap sebagai *null bit string*.

2. Metode *Syntactic*

Sebagaimana yang telah di sarankan oleh Bender et al, mengutarakan penggunaan *punktuasi* dan struktur teks untuk menyembunyikan informasi tanpa secara signifikan mengubah arti dari pesan pembawa. Sebagai contoh terdapat dua frase "*bread, butter, and milk*" dan "*bread, butter and milk*" secara gramatikal benar tetapi berbeda dalam penggunaan koma. Salah satu dapat digunakan secara alternatif dalam pesan teks guna mengintepretasikan nilai "1" apabila salah satu metode dipakai dan nilai "0" untuk metode lain yang dipakai

3. Metode *Semantic*

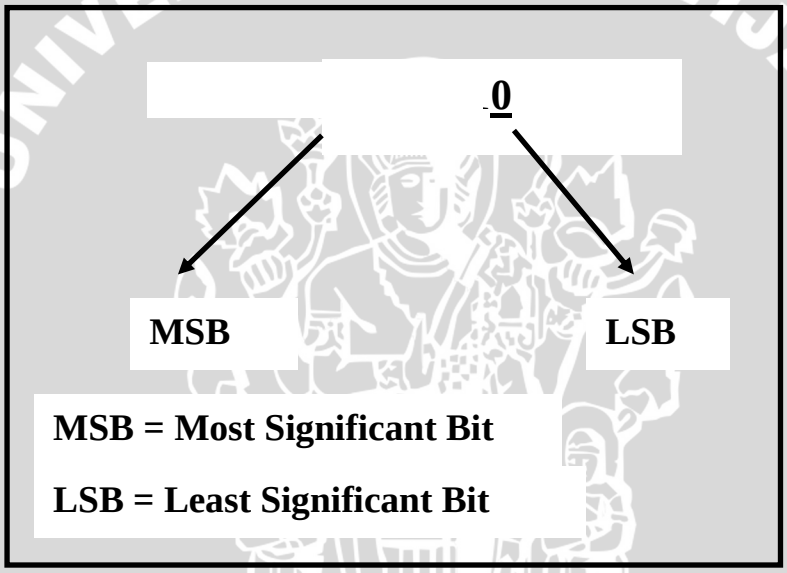
Menggunakan dua sinonim sebagai nilai primer atau sekunder. Nilai tersebut akan diterjemahkan kedalam biner "1" atau "0". Bender et al menggunakan sebuah contoh dimana kata "*big*" berfungsi sebagai primer dan "*large*" berfungsi sebagai sekunder. Oleh karena itu, dalam menguraikan isi sebuah pesan akan menterjemahkan atas penggunaan primer sebagai "1" dan sekunder sebagai "0". Bender et al menyebutkan masalah yang dapat muncul dengan penggunaan metode ini adalah ketika sinonim tidak dapat digantikan karena dapat mengubah arti dari struktur kalimat. Sebagai contoh dalam memanggil seseorang dalam bahasa Inggris dengan "*cool*" mempunyai arti berbeda dibandingkan dengan memanggilnya "*chilly*".

2.2.4.2 Metode Steganografi pada Suara/ Audio

Cara untuk mengaplikasikan steganografi pada *file* audio terdiri dari beberapa cara yang lazim digunakan dan prinsip kerja atau algoritma yang digunakan sama seperti pada metode steganografi pada gambar. Berikut adalah beberapa teknik yang digunakan:

1. *Low Bit coding*

Cara ini lazim digunakan dalam teknik digital steganografi yaitu mengganti LSB *input* setiap *sampling* dengan data yang dikodekan. Dengan metode ini keuntungan yang didapatkan adalah ukuran pesan yang disisipkan *relative* besar, namun berdampak pada hasil audio yang berkualitas kurang dengan banyaknya *noise*.



Gambar 2.4 Metode LSB

Sumber : (Andreas, 2008: *http://andreastjong.wordpress.com/***)**

Pada tabel 2.1 menunjukkan data biner untuk karakter yang akan disisipkan pada data biner *file* penampungnya/ *file carrier*.

Tabel 2.1 Tabel ASCII

Character	ASCII value (decimal)	Hexadecimal	Binary
S	**115**	**73**	**01110011**
E	**101**	**65**	**01100101**
C	**99**	**63**	**01100011**
R	**114**	**72**	**01110010**
E	**99**	**63**	**01100011**
T	**116**	**74**	**01110100**

Sumber : (Andreas, 2008: *http://andreastjong.wordpress.com/*)

Berikut ini adalah data biner dari *file* pembawa/ *file carrier* sebelum disisipi dapat dilihat pada tabel 2.2.

Tabel 2.2 Data biner sebelum disisipi

00000000	00000000	00000001	00000001	00000001	00000001	00000001	00000001
00000000	00000000	00000001	00000001	00000001	00000001	00000001	00000001
00000000	00000000	00000001	00000001	00000001	00000001	00000001	00000001
00000001	00000001	00000010	00000010	00000010	00000011	00000011	00000011
00000001	00000001	00000010	00000010	00000010	00000011	00000011	00000011
00000001	00000001	00000010	00000010	00000010	00000011	00000011	00000011

Sumber : (Andreas, 2008: *http://andreastjong.wordpress.com/*)

Setelah data dari *file carrier* disisipi dengan data rahasia maka hasilnya dapat dilihat pada tabel 2.3 berikut ini.

Tabel 2.3 Data biner setelah disisipi data

00000000	00000001	00000001	00000001	00000000	00000000	00000001	00000001
00000000	00000001	00000001	00000000	00000000	00000001	00000000	00000001
00000000	00000001	00000001	00000000	00000000	00000000	00000001	00000001
00000000	00000001	00000011	00000011	00000010	00000010	00000011	00000010
00000000	00000001	00000011	00000010	00000010	00000010	00000011	00000011
00000000	00000001	00000011	00000011	00000010	00000011	00000010	00000010

Sumber : (Andreas, 2008: *http://andreastjong.wordpress.com/*)

Dari tabel 2.3 dapat dilihat bahwa untuk data biner terakhir akan diganti dengan data biner dari *file* data rahasia yang disisipkan.

2. *Phase coding*

Metode kedua yang digunakan ini adalah merekayasa fasa dari sinyal masukan. Teori yang digunakan adalah dengan mensubstitusi awal fasa dari tiap awal segmen dengan fasa yang telah dibuat sedemikian rupa dan merepresentasikan pesan yang disembunyikan. Fasa dari tiap awal segmen ini dibuat sedemikian rupa sehingga setiap segmen masih memiliki hubungan yang berujung pada kualitas suara yang tetap terjaga. Teknik ini menghasilkan keluaran yang jauh lebih baik daripada metode pertama namun dikompensasikan dengan kerumitan dalam realisasinya.

3. *Spread Spectrum*

Metode yang ketiga adalah penyebaran spektrum. Dengan metode ini pesan dikodekan dan disebar ke setiap spektrum frekuensi yang memungkinkan. Maka dari itu akan sangat sulit bagi yang akan mencoba memecahkannya kecuali ia memiliki akses terhadap data tersebut atau dapat merekonstruksi sinyal acak yang digunakan untuk menyebarkan pesan pada *range* frekuensi.

4. *Echo Hiding*

Metode terakhir yang sering digunakan adalah menyembunyikan pesan melalui teknik *echo*. Teknik menyamarkan pesan kedalam sinyal yang membentuk *echo*. Kemudian pesan disembunyikan dengan menvariasikan tiga parameter dalam *echo* yaitu besar *amplitude* awal, tingkat penurunan atenuasi, dan *offset*. Dengan adanya *offset* dari *echo* dan sinyal asli maka *echo* akan tercampur dengan sinyal aslinya, karena sistem pendengaran manusia yang tidak memisahkan antara *echo* dan sinyal asli. Keempat metode di atas memiliki kesamaan yaitu menggunakan kelemahan dari sistem pendengaran manusia. Maka dari itu teknik steganografi dalam MP3 (MPEG-1 layer 3) juga akan menggunakan kelemahan ini untuk menyembunyikan pesan.

2.2.4.3 Metode Steganografi pada Gambar

Sudah banyak metode yang digunakan untuk menyembunyikan pesan di dalam sebuah *image* tanpa mengubah tampilan *image*, sehingga pesan yang disembunyikan tidak akan terlihat. Berikut akan dibahas beberapa metode umum yang digunakan pada *image* steganografi.

1. Penyisipan *Least Significant Bit*

Cara paling umum untuk menyembunyikan pesan adalah dengan memanfaatkan *Least-Significant Bit* (LSB). Walaupun banyak kekurangan pada metode ini, tetapi kemudahan implementasinya membuat metode ini tetap digunakan sampai sekarang. Metode ini membutuhkan syarat, yaitu jika dilakukan kompresi pada stego, harus digunakan format *lossless compression*, karena metode ini menggunakan bit-bit pada setiap *pixel* pada *image*. Jika digunakan format *lossy compression*, pesan rahasia yang disembunyikan dapat hilang. Jika digunakan *image* 24 bit *colour* sebagai *cover*, sebuah bit dari masing-masing komponen *Red, Green, dan Blue*, dapat digunakan sehingga 3 bit dapat disimpan pada setiap *pixel*. Sebuah *image* 800 x 600 *pixel* dapat digunakan untuk menyembunyikan 1.440.000 bit (180.000 *bytes*) data rahasia. Misalnya, di bawah ini terdapat 3 *pixel* dari *image* 24 bit *colour* :

(00100111 11101001 11001000)
(00100111 11001000 11101001)
(11001000 00100111 11101001)

jika diinginkan untuk menyembunyikan karakter A (10000001b) dihasilkan :

(00100111 11101000 11001000)
(00100110 11001000 11101000)
(11001000 00100111 11101001)

dapat dilihat bahwa hanya 3 bit saja yang perlu diubah untuk menyembunyikan karakter A ini. Perubahan pada LSB ini akan terlalu kecil untuk terdeteksi oleh mata manusia sehingga pesan dapat disembunyikan secara efektif. Jika digunakan *image* 8 bit *colour* sebagai *cover*, hanya 1 bit saja dari setiap *pixel* warna yang dapat dimodifikasi sehingga pemilihan *image* harus dilakukan dengan sangat hati-hati, karena perubahan LSB dapat menyebabkan terjadinya perubahan warna yang ditampilkan pada citra. Akan lebih baik jika *image* berupa *image grayscale* karena perubahan warnanya akan lebih sulit dideteksi oleh mata manusia. Proses ekstraksi pesan dapat dengan mudah dilakukan dengan mengekstrak LSB dari masing-masing *pixel* pada stego secara berurutan dan menuliskan nya ke *output file* yang akan berisi pesan tersebut. Kekurangan dari metode modifikasi LSB ini adalah bahwa metode ini membutuhkan "tempat penyimpanan" yang relatif besar.

Kekurangan lain adalah bahwa stego yang dihasilkan tidak dapat dikompress dengan format *lossy compression*.

2. *Masking dan Filtering*

Teknik *masking* dan *filtering* ini biasanya dibatasi pada *image* 24 bit *colour* atau *image grayscale*. Metode ini mirip dengan *watermark*, dimana suatu *image* diberi tanda (*marking*) untuk menyembunyikan pesan rahasia. Hal ini dapat dilakukan, misalnya dengan memodifikasi *luminance* beberapa bagian dari *image*. Walaupun metode ini akan mengubah tampilan dari *image*, dimungkinkan untuk melakukannya dengan cara tertentu sehingga mata manusia tidak melihat perbedaannya. Karena metode ini menggunakan aspek *image* yang memang terlihat langsung, metode ini akan lebih "*robust*" terhadap kompresi (terutama *lossy compression*), *cropping*, dan beberapa *image processing* lain, bila dibandingkan dengan metode modifikasi LSB.

3. *Transformation*

Metode yang lebih kompleks untuk menyembunyikan pesan pada *image* ini dilakukan dengan memanfaatkan *Discrete Cosine Transformation* (DCT) dan *Wavelet Compression*. DCT digunakan, terutama pada kompresi JPEG, untuk metransformasikan blok 8x8 *pixel* yang berurutan dari *image* menjadi 64 koefisien DCT. Setiap koefisien DCT F(u,v) dari blok 8x8 *pixel image* f(x,y) dihitung sebagai berikut:

$$F(u,v) = \frac{1}{4}C(u)C(v)\left[\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y) * cos\frac{(2x+1)u\pi}{16} cos\frac{(2y+1)u\pi}{16}\right] \quad (2.1)$$

Dimana $C(x) = \frac{1}{\sqrt{2}}$ saat x sama dengan 0 dan C(x) = 1 saat x sama dengan 1. Setelah koefisien-koefisien diperoleh, dilakukan proses kuantisasi sebagai berikut

$$F^2(u,v) = \left[\frac{F(u,v)}{Q(u,v)}\right] \quad (2.2)$$

dengan Q(u,v) adalah 64-elemen dari tabel kuantisasi. Walaupun *image* yang dikompresi dengan *lossy compression* akan menimbulkan kecurigaan karena perubahan LSB akan terlihat jelas, pada metode ini hal ini tidak akan terjadi karena metode ini terjadi di domain frekuensi di dalam *image*, bukan pada *domain spasial*, sehingga tidak akan ada perubahan yang terlihat pada *cover image*. *Wavelet compression* adalah salah satu cara kompresi data yang cocok digunakan untuk kompresi *image*, audio, dan video. Tujuannya adalah untuk menyimpan data dalam "ruang" yang sekecil mungkin dalam sebuah *file*, oleh sebab itu hilangnya informasi tertentu memang sudah diharapkan akan terjadi, kompresi ini

merupakan contoh lossy compression. Sama seperti DCT, *wavelet compression* juga berbasis pada *domain* frekuensi. Keuntungannya, *wavelet compression* lebih baik dalam merepresentasikan daerah transien, contohnya *image* bintang pada langit malam. Artinya, elemen dari data yang transien akan direpresentasikan dalam jumlah informasi yang lebih kecil daripada yang terjadi pada transformasi lain, seperti pada DCT. Kerugiannya, wavelet compression kurang baik digunakan pada data yang bersifat periodik dan *smooth*. Metode yang dilakukan pada *wavelet compression* akan dijelaskan sebagai berikut. Pertama-tama, dilakukan *wavelet transform* yang akan menghasilkan koefisien sesuai dengan jumlah *pixel* pada *image* sebagai berikut

$$[W\varphi f](a,b) = \frac{1}{\sqrt{[a]}} \sum_{-\infty}^{\infty} \varphi \left(\frac{x-a}{a} f(x) dx \right) \quad (2.3)$$

koefisien *wavelet* dapat dicari dengan rumus berikut ini

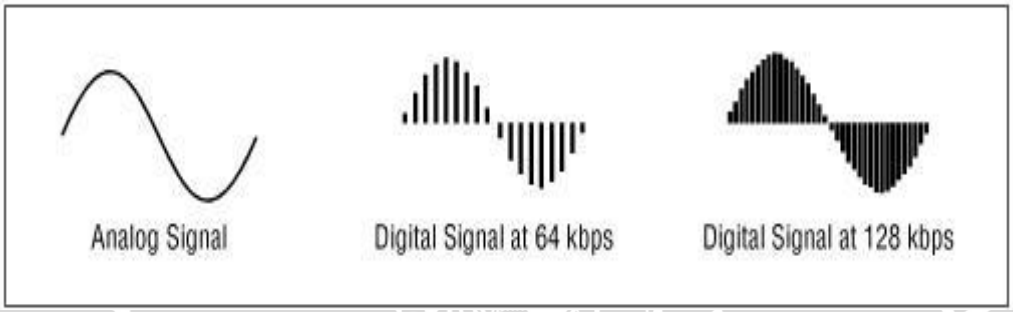
$$C_{jk} = [W\varphi f]\left(2^j . k2^{-j}\right) \quad (2.4)$$

dimana $a = 2^{-j}$ disebut *binary dilation* atau *dyadic dilation*, dan $b = k2^{-j}$ disebut *binary position* atau *dyadic position*. Setelah koefien *wavelet* diperoleh, koefisien ini dapat dikompresi dengan mudah karena informasi terkonsentrasi secara statistik pada beberapa koefisien tertentu saja. Prinsip ini disebut dengan *transform coding*. Setelah itu, koefisien-koefisien tadi dikuantisasi, baru kemudian di-*encode* dengan *entropy encoding* dan/atau *run length encoding*. Proses ekstraksi pesan dengan menggunakan metode transformasi ini dilakukan dengan melakukan transformasi pada stego untuk memperoleh koefisien transformasi *image*. Pilih koefisien yang nilainya lebih kecil dari nilai *treshold*. Ekstrak bit data yang sesuai dengan koefisien ini dan tulis ke *output file* yang akan berisi pesan tersebut.

2.3 MPEG-1 layer 3 atau MP3

Kepopuleran dari MP3 yang sampai saat ini belum tersaingi disebabkan oleh beberapa hal. Pertama MP3 dapat didistribusikan dengan mudah dan hampir tanpa biaya., walaupun sebenarnya hak paten dari MP3 telah dimiliki dan penyebaran MP3 seharusnya dikenai biaya. Walaupun begitu, pemilik hak paten dari MP3 telah memberikan pernyataan bahwa penggunaan MP3 untuk keperluan perorangan tidak dikenai biaya. Keuntungan lainnya adalah kemudahan akses MP3, dimana banyak *software* yang dapat menghasilkan *file* MP3

dari CD dan keberadaan *file* MP3 yang bersifat *ubiquitos* (kosmopolit). Pada perbandingan kualitas suara antara beberapa format kompresi audio hasil yang dihasilkan bervariasi pada *bit rate* yang berbeda, perbandingan berdasarkan *codec* yang digunakan. Pada 128 kbit/s, LAME MP3 unggul sedikit dibandingkan dengan Ogg Vorbis, AAC, MPC and WMA Pro. Kemudian pada 64 kbit/s,AAC-HE dan MP3pro menjadi yang teratas diantara *codec* lainnya. Dan untuk diatas 128 kbit/s tidak terdengar perbedaan yang signifikan.*Bit rate* adalah banyaknya bits per second.[Scot Hacker,2000]



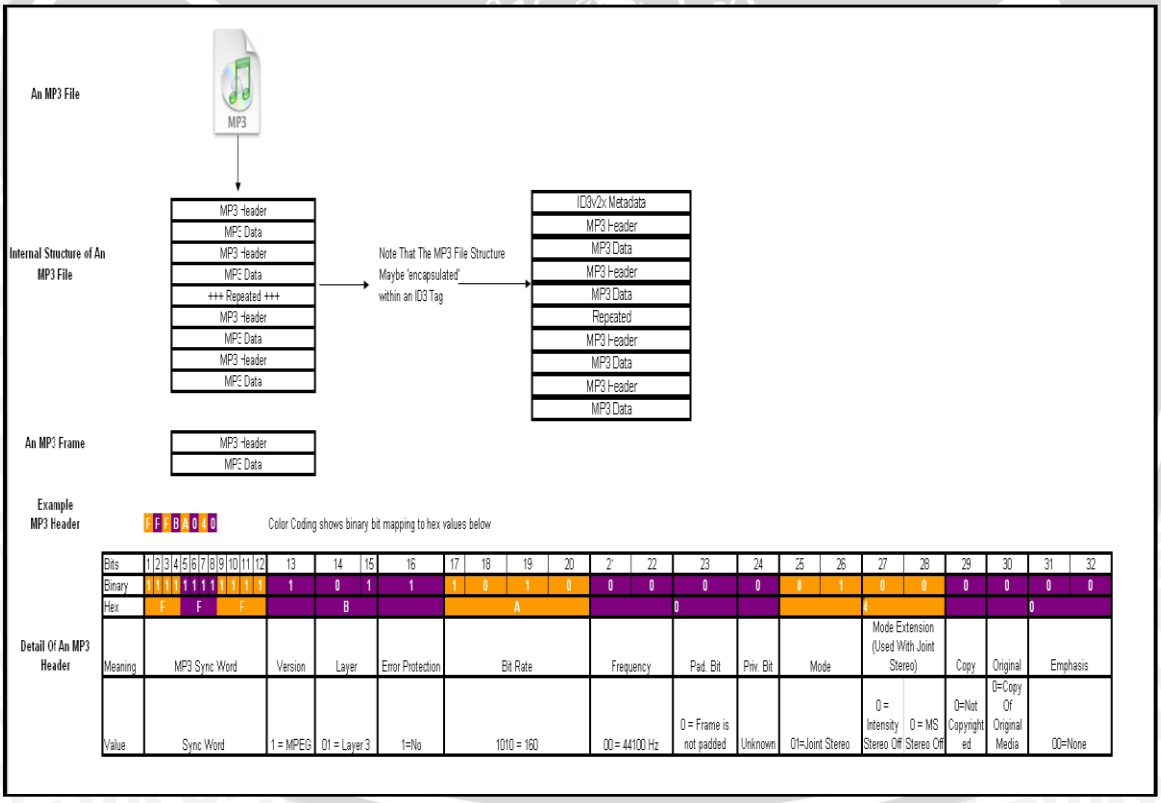
Gambar 2.5 *Bit rate*
Sumber : (Scot Hacker,2000)

Pada umumnya format MP3 sekarang menggunakan 128 kbit/s dan 192 kbit/s sehingga hasil yang dihasilkan cukup baik. Penggunaan MP3 sekarang menjadi sangat tinggi karena pada dasarnya seorang manusia akan lebih suka melakukan hal yang bisa menghibur lebih dibandingkan hal yang statik dan tidak menghibur. Perbedaan ini membuat MP3 lebih dipilih untuk digunakan dibandingkan menggunakan *file* gambar. Dalam melakukan penyembunyian pesan, akan lebih dipilih format yang lebih lumrah digunakan sehingga tidak menimbulkan kecurigaan yang terlalu berlebih Maka dari itu penggunaan MP3 sebagai salah satu media steganografi merupakan langkah yang baik. Lalu lintas pertukaran MP3 di internet merupakan hal biasa sehingga steganografi menggunakan MP3 adalah teknik yang baik untuk mengamankan pesan rahasia melalui media internet. Selain itu jika kita tidak bicara dalam konteks internet, steganografi juga menjadi media yang paling digemari karena paling sering digunakan sebagai sarana hiburan. Semakin sering *file* itu atau semakin terlihat *file* itu maka akan semakin kecil kecurigaan bahwa terdapat pesan tersembunyi dalam *file* tersebut.

2.3.1 Kualitas Audio

Selain merepresentasikan *lossy* audio *encoding* seperti membuat *file* MP3, adalah pertukaran antara sejumlah ruang yang digunakan dan kualitas suara sebagai hasilnya. Biasanya, pencipta menciptakan dengan diikuti mensetting *bit rate*, seperti berapa banyak kilobits dari *file* yang mungkin digunakan tiap detik dari suara. Dengan menggunakan *bit rate* yang rendah akan menghasilkan kualitas audio yang *relative* rendah pula dan diproduksi dalam ukuran *file* yang kecil pula. Begitu halnya dengan jika menggunakan *bit rate* yang tinggi maka akan menghasilkan kualitas audio yang *relative* tinggi dan hasil *file* juga besar.

Disamping *bit rate* dari bagian yang di *encoding* dari audio, kualitas *file* MP3 juga bergantung pada kualitas *encoder* itu sendiri, dan dari tingkat kesulitan sinyalnya saat di *encoding*.

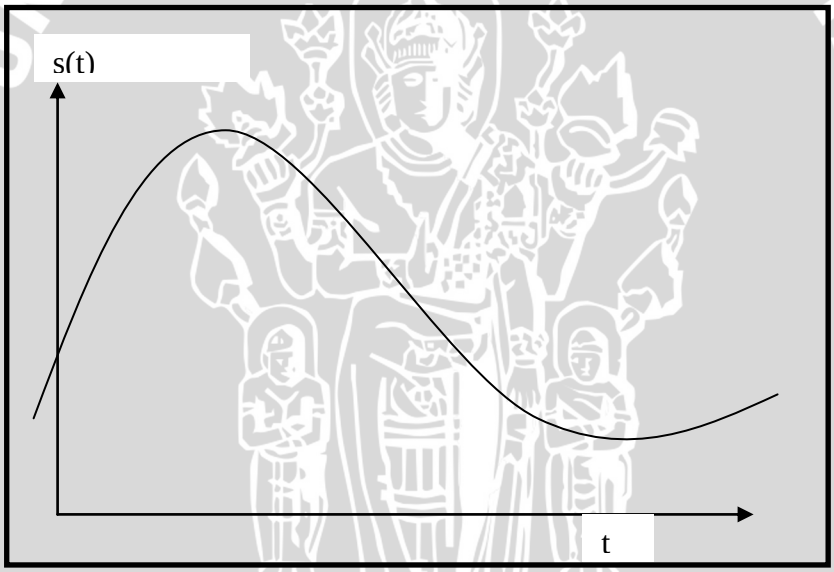


Gambar 2.6 *Frame header* MP3

Sumber : (*http://wikipedia.com/MP3/*)

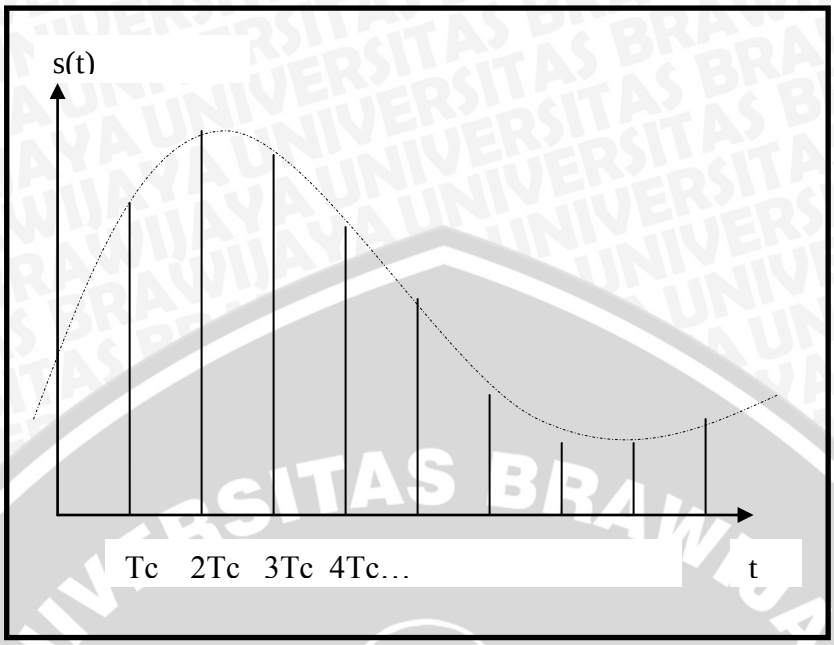
2.3.2 *Sampling Rate*

Sampling rate atau sampling frekuensi adalah jumlah *sample* per detik yang diambil dari sinyal kontinyu untuk dijadikan sinyal diskrit. Kebalikan dari *sampling rate* adalah periode sampling atau interval sampling yaitu waktu diantara tiap *sample*. Konsep dari *sampling rate* hanya dapat digunakan untuk sinyal periodik. Menurut teorema Nyquist-Shannon dikatakan bahwa rekonstruksi sempurna dari sinyal dapat dilakukan ketika sampling rate/frekuensinya lebih besar dua kali lipat dari frekuensi maksimum dari sinyal ketika disampling. Berikut ini adalah gambar dari sinyal analog :



Gambar 2.7 Sinyal analog

Sumber : (*http://wikipedia.com/sampling rate/*)



Gambar 2.8 Sinyal analog hasil sampling

Sumber : (*http://wikipedia.com/sampling rate/*)

Untuk sampling rate dibedakan dalam beberapa rate dan umumnya digunakan pada digital audio yang beragam. Pada tabel 2.4 sampling rate diklasifikasikan dalam beberapa tipe dan penggunaan sebagai berikut :

Tabel 2.4 Sampling rate dan penggunaanya

Sampling rate	Penggunaan
8,000 Hz	Telepon dan enkripsi *walkie talkie*, *wireless intercom* dan tansmisi *wireless microphone*
11,025 Hz	Seperempat kali dari sampling rate CD audio; digunakan untuk PCM kualitas rendah MPEG audio dan untuk analisa audio dari *subwoofer bandpasses*
22,050 Hz	Satu setengah kali dari sampling rate CD audio; digunakan untuk PCM kualitas rendah MPEG audio dan untuk analisa audio dari *low frequency energy*
32,000 Hz	MiniDV *digital video camcorder, video tape* dengan *channel*

	ekstra dari audio (DVCAM dengan 4 *channel* audio), DAT (LP mode), Germany's *Digitales Satellitenradio* **(German)**, NICAM digital audio, digunakan untuk suara televise analog di beberapa negara. *High-quality digital wireless microphones.*
44,056 Hz	PCM adaptor menggunakan NTSC video tape (245 saluran dengan 3 sample dengan 59.94 frame per detik), kadang digunakan untuk *play back audio streams* yang disampling pada 44,100 Hz (dan vice versa)
44,100 Hz	audio CD, juga digunakan dengan MPEG-1 audio (VCD, SVCD, MP3), diadopsi dari PCM *adaptor* menggunakan PAL video tape (588 saluran dengan 3 sample dengan 25 frame per detik). Kebanyakan pro audio gear menggunakan (biasanya menyediakan) sampling 44.1 kHz , termasuk *mixers, EQs, compressors, reverb, crossovers, recording devices* dan CD-*quality encrypted wireless microphones*
47,250 Hz	PCM *sound recorder* komersial pertama di dunia milik Nippon Columbia (Denon)
48,000 Hz	digital sound digunakan untuk miniDV, digital TV, DVD, dan film. Kebanyakan pro audio gear menggunakan (atau menyediakan) sampling 48 kHz, termasuk *mixers, EQs, compressors, reverb, crossovers* dan perangkat rekaman seperti DAT
50,000 Hz	*Digital audio recorder* komersial pertama dari 70s dari 3M dan *Soundstream*
50,400 Hz	*Sampling rate* yang digunakan oleh Mitsubishi X-80 *digital audio recorder*
88,200 Hz	*Sampling rate* yang digunakan oleh *professional recording equipment* ketika tujuannya adalah CD (kelipatan dari 44,100 Hz). Beberapa pro audio gear menggunakan (atau menyediakan) sampling 88.2 kHz, termasuk *mixers, EQs, compressors, reverb, crossovers dan recording devices.*

96,000 Hz	DVD-Audio, beberapa track LPCM DVD, BD-ROM (Blu-ray Disc) audio tracks, dan HD DVD (High-Definition DVD) audio tracks. Beberapa pro audio gear menggunakan (atau menyediakan) sampling 96 kHz, *including mixers, EQs, compressors, reverb, crossovers* dan *recording devices.*
176,400 Hz	sampling rate yang digunakan oleh professional recording equipment ketika tujuannya adalah CD (kelipatan dari 44,100 Hz)
192,000 Hz	DVD-Audio, beberapa track LPCM DVD , BD-ROM (Blu-ray Disc) *audio tracks,* dan HD DVD (*High-Definition* DVD) *audio tracks, High-Definition audio recording devices Iaudio editing software*
2,822,400 Hz	SACD, proses 1-bit sigma-delta modulation yang dikenal dengan *Direct Stream Digital*, dikembangkan oleh Sony dan Philips

2.4 *File* data .TXT

TXT adalah sebuah *filename extension* untuk sebuah *file* yang terdiri dari teks yang biasanyanya terdiri dari format yang sederhana (contohnya tidak menggunakan cetak tebal/*bolding* dan cetak miring/*italics*). Definisi yang tepat dari *file* .txt tidak spesifik, tetapi biasanya sesuai dengan dengan format yang diterima oleh terminal sistem atau teks *editor* yang sederhana. *File* dengan .txt *extensions* dapat dibaca dengan mudah atau dibuka oleh program lain yang membaca teks dan digunakan pada *independent platform*.

Seting dari karakter ASCII adalah umumnya untuk format *file* teks dalam bahasa Inggris, dan biasanya diasumsikan kedalam format *file* yang dapat diubah dalam banyak situasi atau keadaan. Untuk dititik beratkan dan untuk karakter-karakter non-ASCII, biasanya menggunakan karakter *encoding*. Di banyak sistem, hal ini dipilih berdasarkan seting lokal dari komputer pada saat dibaca. Karakter *encoding* yang umum digunakan termasuk ISO 8859-1 untuk kebanyakan bahasa Eropa[*http://wikipedia.com/plaintext//*)

2.5 SNR (*Signal to Noise Ratio)*

Signal to noise ratio (umumnya disingkat **SNR** atau **S/N**) adalah pengukuran yang umumnya digunakan dalam ilmu pengetahuan dan teknik untuk menghitung berapa banyak sinyal yang dikorupsi oleh *noise*. hal ini didefinisikan sebagai *ratio* dari sinyal *power* dengan *noise power* yang mengkorupsi sinyal *power*. Jika *ratio* menunjukkan lebih besar dari 1:1 hal ini mengindikasikan bahwa sinyal lebih besar daripada *noise*. Walaupun SNR biasanya digunakan hanya untuk sinyal elektrik, tetapi juga dapat diaplikasikan untuk bentuk-bentuk dari sinyal yang lain(seperti *level isotop* pada inti es atau pensinyalan *biochemical* antar sel).

SNR adalah salah satu faktor yang mempengaruhi kualitas sistem transmisi digital. Semakin besar nilai S/N (SNR) menunjukkan bahwa kualitas sistem transmisi digital semakin baik. Berikut ini adalah persamaan SNR :

$$\left(\frac{S}{N}\right) = \frac{S(f)|H(f)|^2}{W} \qquad (2.5)$$

dengan : $\left(\frac{S}{N}\right)$ = Signal to Noise Ratio (dB)

S(f) = PSD *Transmitter* (mW/Hz)

H(f) = Fungsi transfer dari kabel tembaga

W = PSD Receiver (mW/Hz)

BAB III
METODOLOGI PENELITIAN

Kajian dalam penelitian ini adalah kajian bersifat aplikatif yaitu Analisis *Ratio Data Embedding* dan SNR (*Signal to Noise Ratio*) pada Audio Steganografi dengan Metode LSB (*Low Significant Bit*). Tahapan yang dilakukan dalam penelitian ini adalah penentuan jenis data dan cara mendapatkan data, variabel dan cara analisis data, dan rangka solusi permasalahan yang disajikan dalam bentuk *flowchart*.

3.1 Jenis Data dan Cara Mendapatkan Data

Jenis data dan cara mendapatkan data dalam pengerjaan skripsi ini merupakan bagian yang harus dicatat secara rinci, terutama dalam tahapan pengerjaan skripsi yang berdasarkan kepustakaan. Jenis data yang dibutuhkan dalam proses pengerjaan skripsi ini adalah data primer dan data sekunder. Data primer adalah data yang didapat dari proses simulasi skripsi ini yaitu penerapan audio steganografi dengan metode LSB (*Low Significant Bit*) menggunakan bahasa pemrograman java J2SDK. Sedangkan data sekunder adalah data yang diperoleh dari berbagai buku, jurnal-jurnal dan dari internet yang berhubungan dengan teori dasar audio steganografi dan metode LSB (*Low Significant Bit*).Mempelajari teori audio steganografi dan metodenya, serta parameter-parameter yang digunakan dalam pengukuran kualitas data *file* mengggunakan metode *Low Significant Bit* yang meliputi *Ratio Data Embedding* dan *Signal to Noise Ratio*.

3.2 Variabel dan Cara Analisis Data

Kualitas data *file* yang akan dianalisis meliputi *Ratio Data Embedding* (RDE) dan *Signal to Noise Ratio* (SNR). Semua parameter tersebut dihitung dan dianalisis pada audio steganografi dengan menggunakan data-data primer dan sekunder yang telah didapatkan dan selanjutnya disimulasikan menggunakan Netbean 6.8. Berikut ini adalah analisa data untuk RDE dan SNR :

- *Ratio Data Embedding*

Ratio Data Embedding merupakan perhitungan dari besarnya kemungkinan disisipkannya 1 bit data rahasia kedalam *file* MP3. RDE dipengaruhi oleh besarnya jumlah bit data rahasia dan jumlah total bit *file* pembawanya (MP3).

$$ratio\ data\ embedding = \left(\frac{jumlah\ bit\ yang\ disisipkan}{jumlah\ bit\ file\ pembawa}\right) bit$$

Jumlah bit yang disisipkan = jumlah bit dari data rahasia yang akan disisipkan

Jumlah bit *file* pembawa = jumlah bit dari *file* MP3

- SNR

SNR digunakan untuk mengetahui pengaruh penyisipan data rahasia kedalam *file* MP3.

$$\left(\frac{S}{N}\right) = \frac{S(f)|H(f)|^2}{W}$$

$\left(\frac{S}{N}\right)$ = Signal to Noise Ratio (dB)

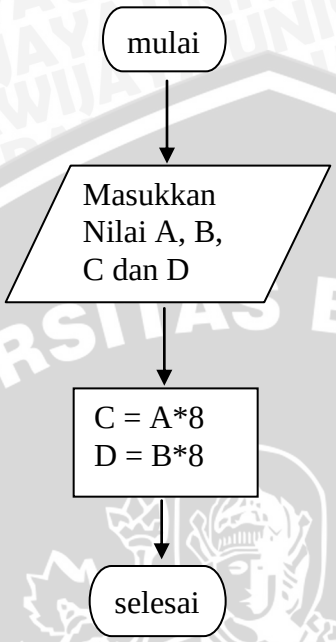
S(f) = PSD *Transmitter* (mW/Hz)

H(f) = Fungsi transfer dari kabel tembaga

W = PSD Receiver (mW/Hz)

3.3 Rangka Solusi Permasalahan

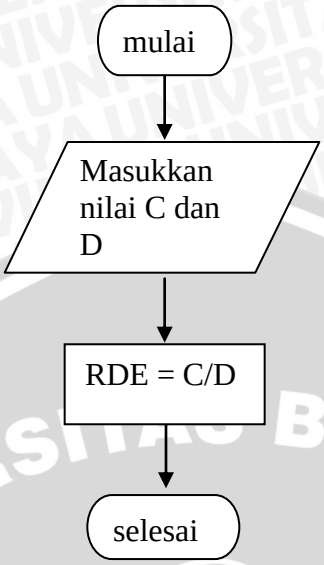
- Jumlah bit yang disisipkan dan jumlah total bit *file* pembawa



Gambar 3.1 Diagram alir perhitungan jumlah bit yang disisipkan dan jumlah total bit *file* pembawa

Data yang dibutuhkan untuk menghitung jumlah bit yang disisipkan (C) adalah besarnya kapasitas dari data rahasia (A). Sedangkan untuk jumlah total bit *file* pembawa (D) yang dibutuhkan adalah besarnya kapasitas dari *file* MP3 (B).

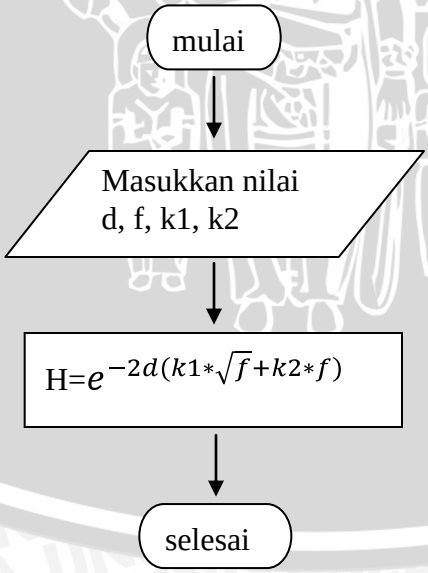
- Perhitungan *Ratio Data Embedding*



Gambar 3.2 Diagram alir perhitungan *Ratio Data Embedding*

Data yang dibutuhkan untuk mendapatkan nilai *Ratio Data Embedding* adalah jumlah bit yang disisipkan (C) dan jumlah total bit *file* pembawa (D).

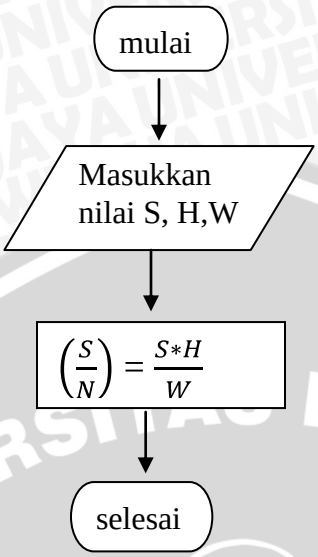
- Perhitungan nilai fungsi transfer kabel tembaga (H(f)



Gambar 3.3 Diagram alir perhitungan nilai fungsi transfer kabel tembaga

Perhitungan nilai fungsi transfer kabel tembaga didapatkan dari data sekunder berupa jenis kabel dan konstanta kabel tembaga 24 AWG.

- Perhitungan SNR



Gambar 3.4 Diagram alir perhitungan SNR

Data yang dibutuhkan untuk mendapatkan nilai SNR adalah *Power Spectral Density Transmitter* (S(f)), *Power Spectral Density Receiver* (W), dan nilai fungsi transfer kabel tembaga (H(f)).

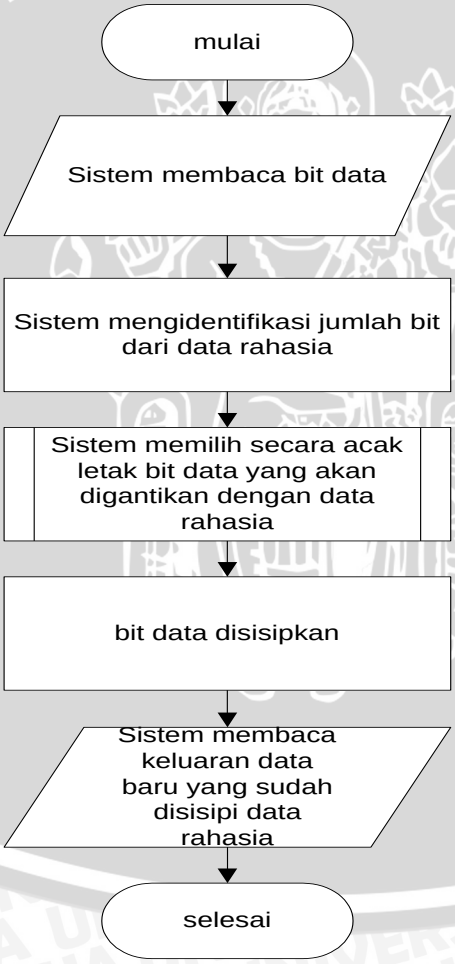
BAB IV
PERENCANAAN DAN PERANCANGAN PERANGKAT LUNAK

4.1 Metode Perencanaan dan Perancangan Steganografi

Pada bab ini dibahas tentang metode perancangan steganografi dengan menggunakan metode LSB secara lengkap.

4.1.1 Prinsip Kerja LSB

Dalam skripsi ini metode LSB dimasukkan dalam program pembentukan steganografi yang akan digunakan pada saat proses *encoding* dan *decoding*. Dimana tiap *input sampling* dari data *carrier* akan diganti dengan data yang dikodekan.

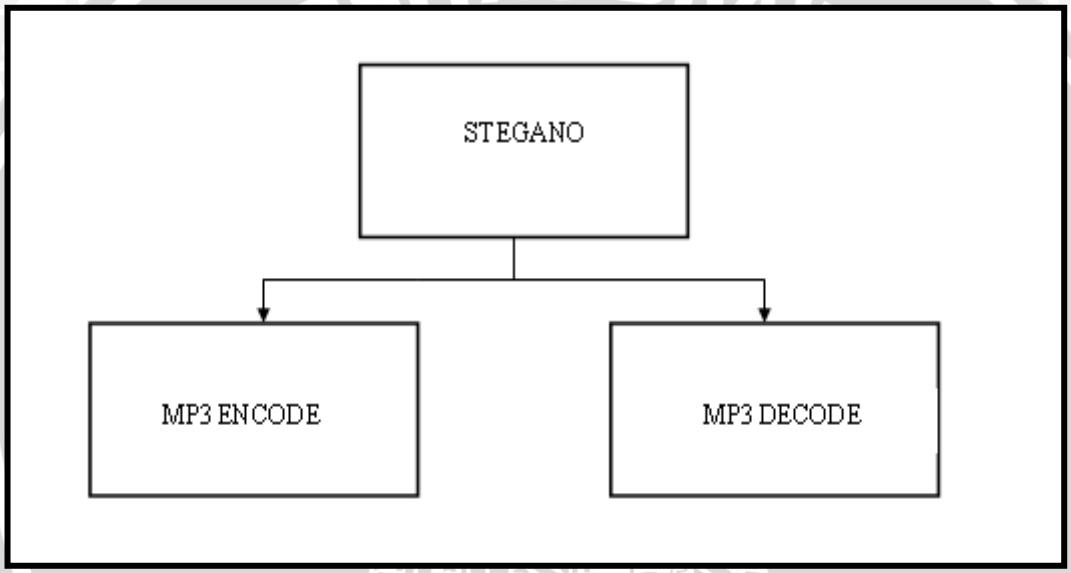


Gambar 4.1 Diagram alir LSB

Sumber : Perancangan

4.1.2 Model Perancangan Perangkat Lunak

Dalam bagian ini akan disimulasikan proses *encoding* dan *decoding* dalam steganografi. Yang akan dianalisa pada skripsi ini adalah bagaimana cara menyisipkan data pada *file* audio dan bagaimana pengaruhnya terhadap kualitas data *file* yang dikirimkan dengan penghitungan SNR (*Signal to Noise Ratio*) *dan Ratio data Embedding*. Berikut ini adalah blok diagram dari steganoLSB :



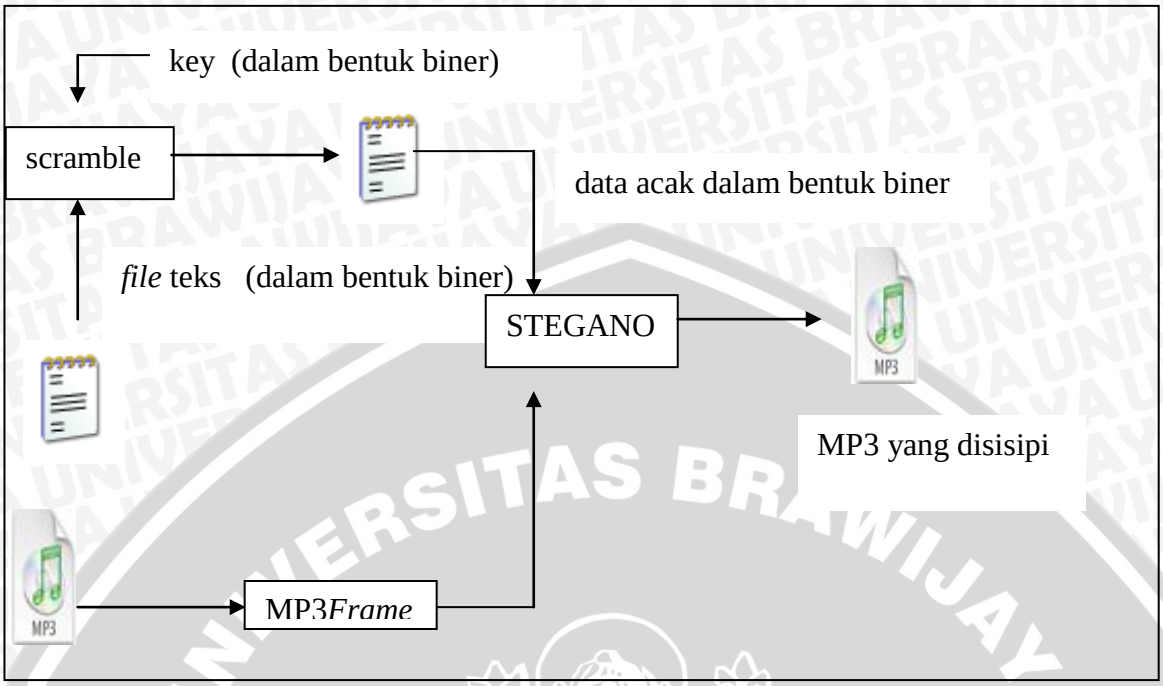
Gambar 4.2 Blok diagram stegano

sumber: Perancangan

Dalam blok stegano terdapat 2 model perancangan yaitu:

a. Proses *encoding* atau penyisipan : yaitu proses untuk menyisipkan data yang sudah dikodekan
b. Proses *decoding* : yaitu proses untuk mengambil kembali pesan yang telah disisipkan.

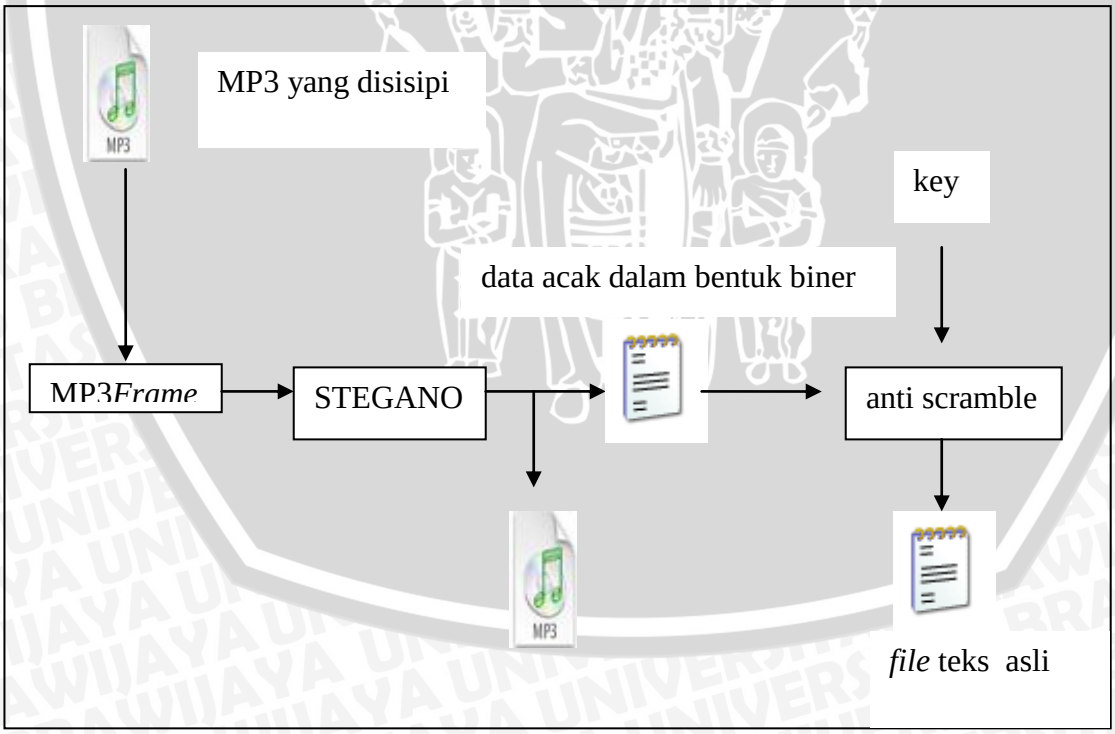
Berikut ini adalah blok diagram untuk proses *encoding*:



Gambar 4.3 Blok diagram proses e*ncoding*

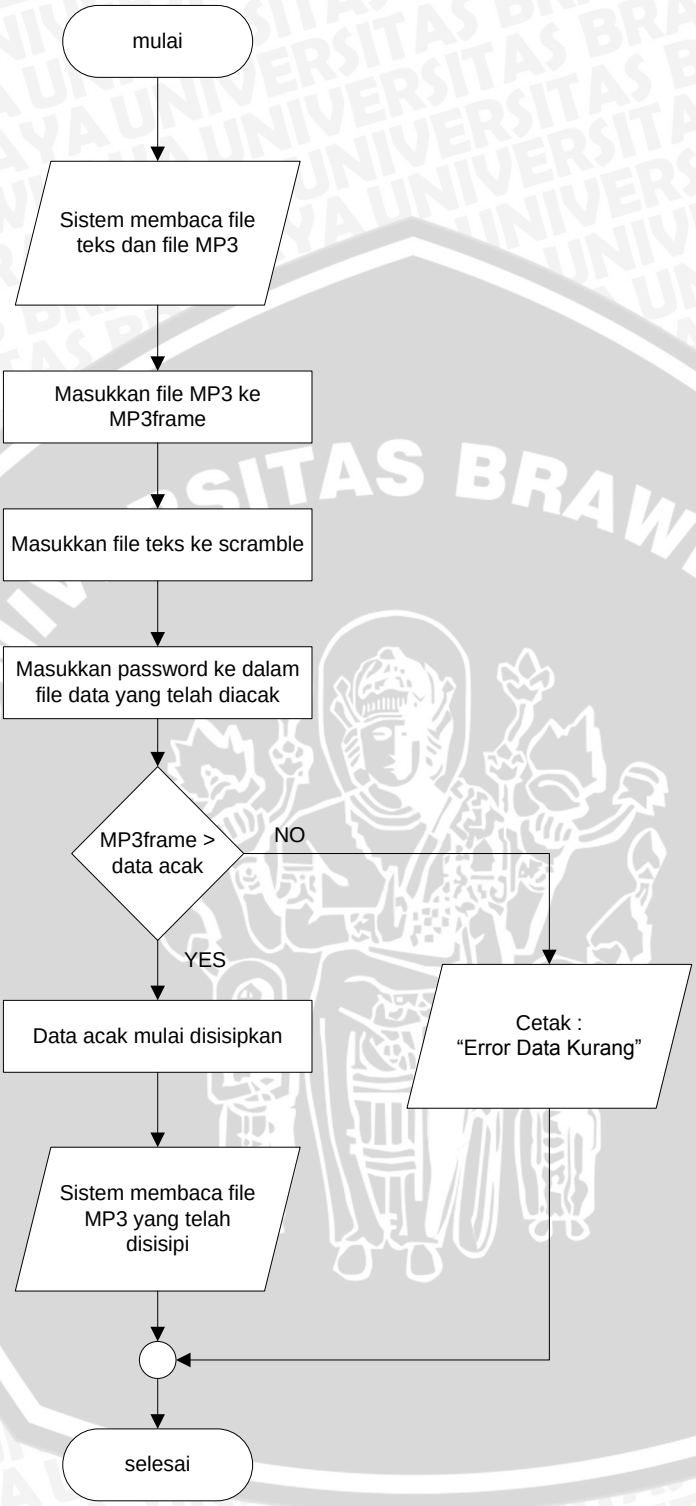
Sumber : Perancangan

Berikut ini adalah blok diagram dari proses *decoding*:



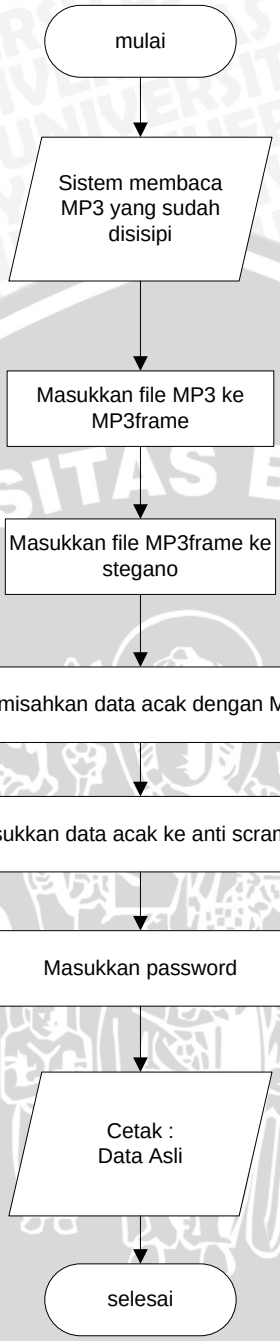
Gambar 4.4 Blok diagram proses *decoding*

Sumber : Perancangan



Gambar 4.5 Diagram alir proses *encoding*

Sumber : Perancangan



Gambar 4.6 Diagram alir proses *decoding*

Sumber :Perancangan

Dalam blok diagram proses *encoding* dan *decoding* terdapat beberapa sub sistem pembentuk steganografi. Berikut ini adalah penjelasannya :

1. *File* teks

Merupakan data atau pesan rahasia yang akan disisipkan dalam *file* MP3 dalam bentuk *file* (.txt). Dalam pengujian yang digunakan adalah *file* teks yang mempunyai ukuran *file* yang berbeda yaitu 1kbps, 10 kbps, 20 kbps, 496 kbps. Untuk ukuran data 496 kbps digunakan untuk menguji kemampuan maksimum dari *file* MP3 untuk menampung *file* yang ukurannya lebih besar dari *file* yang disediakan oleh *file* MP3 .

2. MP3

Merupakan *file carrier* atau sebagai media pembawa pesan rahasia. *File* yang digunakan mempunyai ukuran *file* dan *bi trate* yang berbeda yaitu 2.28 MB(56kbps), 4.86 MB(128kbps), 5.68 MB(192kbps), 6.71 MB(256kbps). Hal ini dilakukan untuk melihat seberapa besar pengaruh *noise* terhadap perubahan *bit rate* yang digunakan.

3. *Scramble/ anti*

Merupakan program *random* java yang digunakan untuk mengacak/mengembalikan urutan data dari *file* teks sebelum dilakukan proses *encoding*/*decoding*. Pada program *scramble* data akan disusun dengan acak sesuai dengan *list* yang digunakan sebagai *root* untuk menjalankan program. Berikut ini adalah *list* acak dari program yang nantinya akan digunakan pada program *anti scramble* sebagai acuan untuk mengembalikan data acak menjadi data asli. *List* acak =(5,1,2,3,4,9,6,7,8,0,10,11,12,13,15,14,16,17,18,19,20,21,22,23,24,25,26,27,31,29,30,28).

4. MP3 *Frame*

Mengubah *file* data MP3 kedalam bentuk *Arraylist*. Disini *frame* data dari *file* MP3 akan dipisah dalam bentuk *frame-frame* yang akan disusun berurutan untuk mempermudah pada penyisipan data acak kedalam *file* MP3. Ukuran tiap *frame* dari *frame* data MP3 berbeda-beda, maka besar data yang akan disisipkan dalam tiap *frame* juga berbeda.

5. *Key*/ kata kunci

Kata kunci atau *password* yang digunakan dalam bentuk *hexadecimal*. Karena *password* akan diubah dalam bentuk *file* biner dan dilakukan operasi XOR.

5. Stegano

Berfungsi untuk menyisipkan dan atau mengambil data *file* teks/ pesan rahasia ke atau dari *file* MP3. Disini untuk menyisipkan data rahasia kedalam *file* MP3 menggunakan metode LSB yaitu menyisipkan data pada bit terakhir. Dimana bit-bit dalam *file* MP3 dipotong tiap kelipatan 8 bit dan mengganti bit ke 8 dengan 1 bit dari data rahasia.

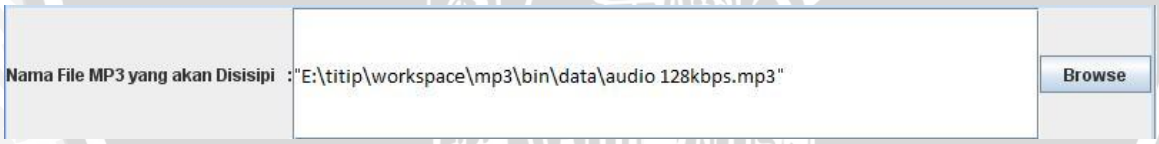
Berikut ini adalah tampilan dari program yang telah dijalankan :

1. Tampilan untuk proses *encoding*

Pada tampilan proses *encoding* terdapat tiga bagian tampilan yang terdiri dari :

a. Pengambilan data *carrier* atau *file* MP3

Pada tahap ini *file carrier* atau *file* MP3 yang akan disisipi dengan data rahasia akan diambil dengan menekan tombol *browse* pada bagian ini. Maka sistem akan menampilkan data secara lengkap dimana tempat kita menyimpan *file* MP3 yang akan kita gunakan sebagai data *carrier*. Kita dapat dengan mudah untuk memilih *file* yang akan kita gunakan. Contohnya adalah kita mengambil data MP3 audio 128kbps.MP3 dari *folder* "E:\titip\workspace\MP3bin\audio 128kbps.MP3\MP3". Maka sistem akan mengambil *file carrier* dari *folder* yang telah ditunjuk.



Gambar 4.7 Pengambilan data *carrier* atau *file* MP3

Sumber : Perancangan

b. Pengambilan data rahasia atau *file* teks yang akan disisipkan

Pada tahap ini kita akan mengambil *file* teks rahasia yang akan kita sisipkan dalam *file* MP3. Dengan menekan tombol *Browse* kita dapat mencari dan menentukan data mana yang akan kita pilih. *File* yang kita gunakan adalah *file* yang tersimpan dalam *file* *.txt. Contohnya adalah kita akan mengambil data 4.txt, dengan cara menekan *browse* kita

mencari dan menentukan *file* yang akan disisipkan. Maka pada layar akan muncul lokasi dari *file* tersebut. Seperti contoh dibawah ini.

Gambar 4.8 Pengambilan data rahasia atau *file* teks yang akan disisipkan

Sumber : Perancangan

c. Penyimpanan data MP3 yang telah disisipi atau MP3Stego

Pada tahap ini kita akan menyimpan data MP3 yang telah disisipi dengan data pesan rahasia. Dimana *file* tersebut kita beri nama atau label baru dan otomatis akan disimpan dalam direktori yang telah kita tunjuk dengan menekan tombol *Browse*. "E:\titipsek\workspace\MP3bin\dataaudio128kbps.MP3hasilsteganoadu_4_MP3" merupakan salah satu contoh dalam pemberian nama atau label baru pada *file* MP3 yang tela disisipi. Kemudian untuk memulai proses *encoding* kita tekan tombol *encode* pada tampilan untuk memulai proses *encoding* data. Setelah proses selesai maka kita tekan tombol *exit* yang terdapat pada tampilan tersebut.

Gambar 4.9 Penyimpanan data MP3 yang telah disisipi atau MP3Stego

Sumber : Perancangan

2. Tampilan untuk proses *decoding*

Sama halnya pada proses *encoding*, pada tahap *decoding* juga terdiri dari tiga tahap atau bagian yaitu :

a. Tahap pengambilan data MP3 yang akan di *decode*

Pada tahap ini data MP3 yang telah diencode diambil untuk dilakukan proses *decode*. Dengan menekan tombol *Browse* maka sistem akan mencari dari direktori yang ditunjuk

dimana *file* MP3 yang telah di *encode*. Contohnya kita ambil dari lokasi berikut ini "E:\titip\workspace\MP3bin\dataaudio128kbps.MP3hasilstegano_4_mp3", maka program akan mencari data dari lokasi tersebut.

Gambar 4.10 Pengambilan data MP3 yang akan di *decode*

Sumber : Perancangan

b. Penyimpanan data rahasia atau data yang telah disisipkan

Pada tahap ini kita akan memberi nama baru untuk *file* teks yang telah di *decode* dan akan disimpan dalam direktori yang telah kita tentukan dengan menekan tombol *Browse*. Maka akan disimpan "E:\titip\workspace\MP3bin\hasil*decode*128_4.txt"

Gambar 4.11 Penyimpanan data rahasia atau data yang telah disisipkan

Sumber : Perancangan

c. Memasukkan kata kunci atau key/password

Pada tahap akhir ini kita diminta untuk memasukkan password sebagai kunci untuk melakukan proses *decoding*. Setelah itu baru kita jalankan proses *decoding* dengan menekan tombol *decode* pada tampilan dan setalah selesai kita tekan tombol *exit*.

Gambar 4.12 Memasukkan kata kunci atau key/password

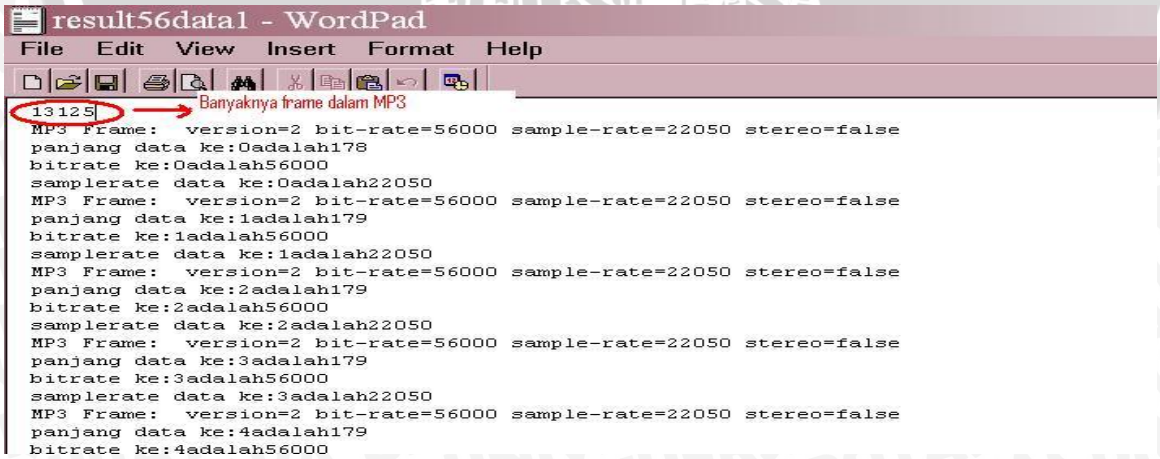
Sumber : Perancangan

BAB V
PENGUJIAN DAN ANALISA DATA

5.1 Hasil Simulasi

Dalam proses simulasi ini, pengambilan data dilakukan dengan cara menghitung besarnya data dari *file* teks yang berubah-ubah *size* atau ukurannya untuk masing-masing *file* MP3. Dalam pengujian *file* MP3 yang digunakan juga memiliki *bit rate*s yang berbeda-beda, yaitu dari *bit rate* 56kbps, 128kbps, 192kbps, 256kbps, dan 320kbps. Hal ini dilakukan agar proses simulasi dapat menguji tingkat keberhasilan dari program steganografi. Hasil yang diperoleh adalah besarnya *frame* dan panjang data yang disisipkan dalam tiap *frame*.Untuk penghitungan *Ratio Data Embedding* yang diperlukan adalah jumlah bit yang disisipkan dalam hal ini adalah besarnya *file* teks atau pesan rahasia yang akan disisipkan dan jumlah total bit *file* pembawa atau *file* MP3nya. Sedangkan untuk penghitungan SNR (*Signal to Noise Ratio*) yang diperlukan adalah jumlah total bit *file* MP3 dan jumlah bit terubah dalam hal ini yang dimaksud adalah jumlah bit dari *file* MP3 yang terubah oleh *file* teks/ pesan rahasia saat disisipkan. Untuk *file* MP3 yang digunakan besarnya adalah 2.28MB, 4.86MB, 5.68MB, 6.71MB, 9.04MB.

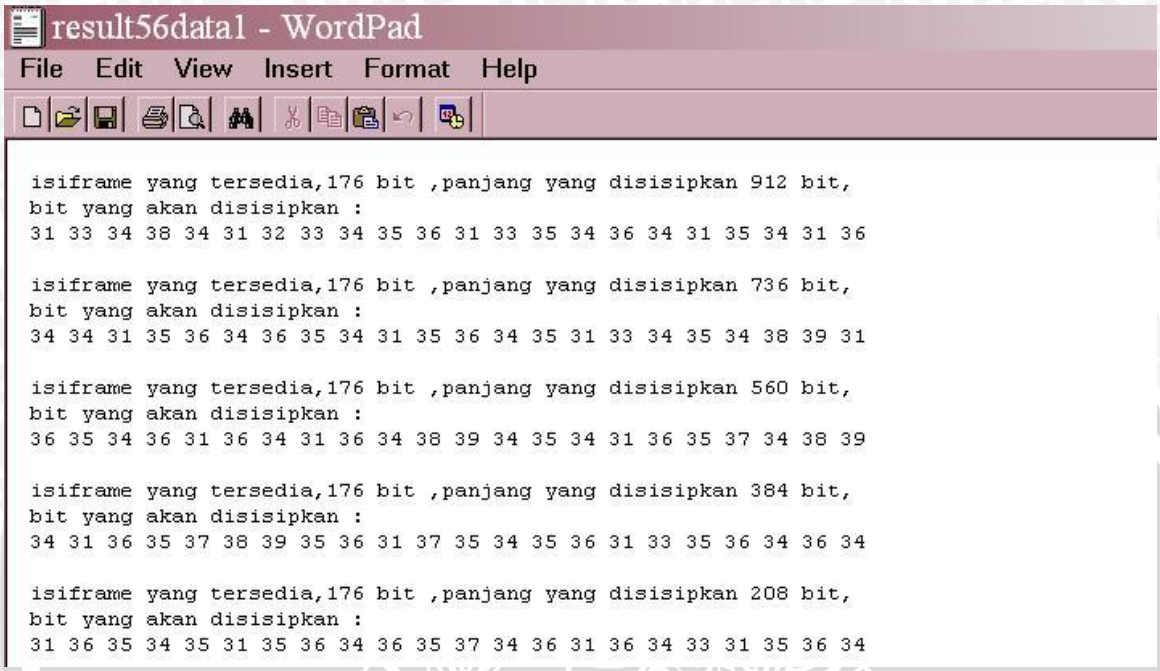
Berikut ini adalah hasil simulasi dari data 1 untuk *file* data 1kb dengan *file* MP3 ukuran 2.28MB dan *bit rate*nya 56kbps:



Gambar 5.1 Tampilan hasil proses *encoding* yang menunjukkan banyaknya frame

Sumber : Perancangan

Dari data diatas dapat diketahui banyaknya *frame* dari *file* MP3, panjang tiap *frame*, *sample rate*, *bit rate*, jenis layer yang digunakan dan panjang data yang disisipkan.

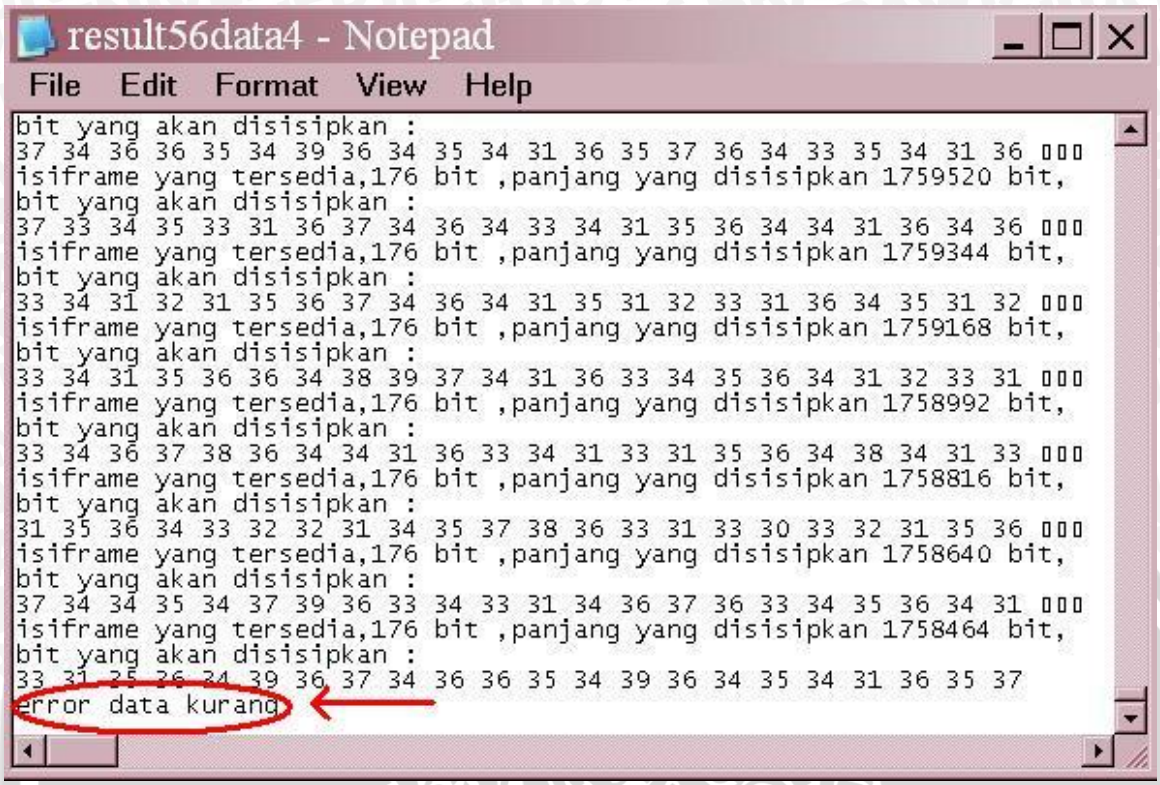


Gambar 5.2 Tampilan hasil proses *encoding* yang menunjukkan bit data yang disisipkan

Sumber : Perancangan

Dari gambar 5.2 diatas menampilkan isi *frame* yang tersedia untuk data pesan rahasia dan panjang bit dari data pesan rahasia yang akan disisipkan serta bit-bit yang akan disisipkan. Hasil dari *encoding* menunjukkan bahwa ukuran *file* MP3 tidak berubah. Untuk ukuran *file* MP3 sebelum disisipi adalah 2.28 MB, setelah disisipi pesan *file* data 1 kb ukuran file MP3 tetap 2.28 MB.

Jika isi *frame* yang tersedia tidak mencukupi untuk data *file* yang akan disisipkan maka akan muncul pesan yang mengatakan “error, data kurang” seperti contoh dibawah ini.



Gambar 5.3 Tampilan hasil proses *encoding* untuk data 4 (496kb)

Sumber : Perancangan

5.2 Hasil Pengujian dan Analisa

5.2.1 Data Hasil Pengujian Untuk *Ratio Data Embedding*

$$ratio\ data\ embedding = \left(\frac{jumlah\ bit\ yang\ disisipkan}{jumlah\ bit\ file\ pembawa}\right) bit$$

Ratio data embedding = 8000/18240000= 1/2280 bit

Dari data diatas dapat dilihat untuk jumlah bit yang disisipkan sebanyak 8000 bit dengan jumlah total bit *file* pembawa sebesar 18240000 bit akan didapatkan perbandingan 1 : 2280. Artinya *file carrier* atau *file* pembawanya mampu menyediakan 2280 bit untuk digantikan datanya dengan 1 bit dari data pesan rahasia. Hasil perhitungan *Ratio Data Embedding* dapat dilihat pada table 5.5 berikut ini.

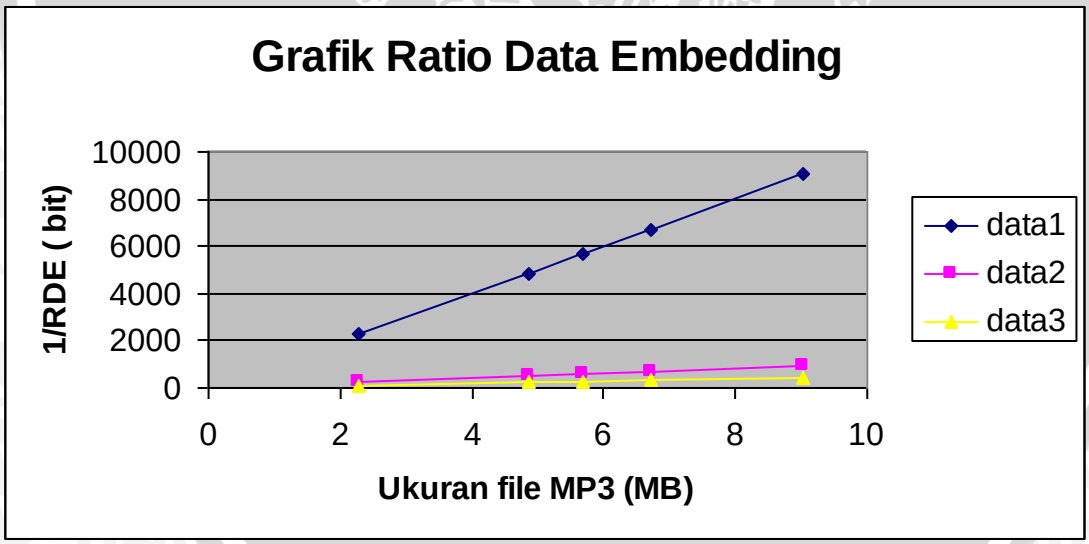
Tabel 5.1 ***Ratio Data Embedding***

Ukuran *File* MP3 (dalam bit)	Data 1 (1kb) (bit)	Data 2 (10kb) (bit)	Data 3 (20kb) (bit)
18240000	1/2280	1/228	1/114
38880000	1/4860	1/486	1/243
45440000	1/5680	1/568	1/284
53680000	1/6710	1/671	1/335
72320000	1/9040	1/904	1/452

Sumber : Pengujian

Dari table 5.1 diatas dapat dilihat bahwa semakin besar kapasitas yang disediakan oleh *file carrier* maka akan semakin besar pula kemungkinan disisipkannya 1 bit data pesan rahasia didalamnya.

Grafik *Ratio Data Embedding* :



Gambar 5.4 Grafik *Ratio Data Embedding*

Sumber : Pengujian

Grafik diatas menunjukkan untuk data pesan rahasia yang berukuran kecil pada data 1 grafiknya menunjukkan kenaikan yang signifikan. Yaitu untuk kapasitas *file carrier* yang semakin besar maka kemungkinan untuk menyisipkan 1 bit data pesan rahasia juga

semakin besar. Sedangkan untuk data 2 dan data 3 memiliki kemungkinan yang hampir sama dengan data 1, tetapi tidak sebesar kemungkinan pada data 1. Karena ukuran dari data rahasia untuk data 2 dan data 3 lebih besar dari data 1.

5.2.2 Data Hasil Pengujian Untuk *Signal to Noise Ratio*

- **Analisa perhitungan Tahanan Isolasi**

Spesifikasi untuk kabel tembaga berdiameter 0.5 mm (24 AWG) adalah sebagai berikut :

- Jari-jari dalam (r1) = 0.25 mm
- Jari-jari luar (r2) = 0.51 mm
- Tahanan jenis (ρ) = 10^{13}Ωm (untuk PE)

Maka nilai tahanan isolasi untuk panjang saluran transmisi 4.886 km adalah sebagai berikut:

$$R_{isolasi} = \frac{2.3\rho}{2\pi l} log \frac{r2}{r1} = \frac{2.3*10^{13}}{2\pi 4.886} log \frac{0.51}{0.25} = 231.6185\ M\Omega$$

- **Analisis Perhitungan Resistansi (R)**

d = 0.51 mm ρ = 0.01754 Ωmm^2 /m ε_0= 8.85 nF/km ε_r= 2.26 nF/km

μ = 4π10^{-7}H/m s=2d=1.02 mm tgθ = 0.0002

$$R_0 = \frac{2\,\rho}{0{,}25\pi d^2} = \frac{2.0{,}01754\Omega m m^2\ /m}{0{,}25.\pi.0.2601\Omega m m^2} = 171.810\ \Omega\ /km$$

Sedangkan pada frekuensi tinggi, nilai R dipengaruhi oleh adanya efek kulit (*skin effect)*. Efek kulit untuk frekuensi 1kHz adalah sebagai berikut :

$$\delta = \sqrt{\frac{\rho}{\pi.f.\mu}} = \sqrt{\frac{0.01754}{\pi.1.10^3 4\pi 10^{-7}}} = 2.108\ mm$$

Sehingga dengan besar resistansi untuk frekueni 1 kHz adalah sebagai berikut :

$$R = R_0 \left[\frac{(d+\delta}{4\delta}\right] = 171.810\left[\frac{(0.51+2{,}108}{4.2{,}108}\right] = 53.34\ \Omega\ /km$$

- **Analisis Perhitungan Kapasitansi (C)**

Nilai kapasitansinya adalah sebagai berikut :

$$C = \frac{\pi.\varepsilon}{ln\left(\frac{s}{d} + \sqrt{\frac{s^2}{d^2} - 1}\right)} = \frac{\pi.8{,}85.2{,}26}{ln\left(\frac{1{,}02}{0{,}51} + \sqrt{\frac{(1{,}02)^2}{(0{,}51)^2} - 1}\right)} = 47{,}712\ nF/km$$

- **Analisis Perhitungan Konduktansi (G)**

Nilai konduktansi untuk frekuensi 1 kHz adalah sebagai berikut :

G = 2 π.f. tgθ.C

= 2 π x 1.10^{3} x 0,0002 x $47{,}712.10^{-9}$= 0,05997 Mho/km

- **Analisis Perhitungan Induktansi (L)**

Induktansi total suatu kabel merupakan hasil penjumlahan dari induktansi dalam (Lin) dengan induktansi luar (Lex) kabel tersebut. Besar induktansi total untuk frekuensi 1 kHz dapat dihitung sebagai berikut :

Lin = $\frac{\mu}{4\pi}\left[\frac{2\delta}{d}\right]$ (H/km)

= $\frac{4\pi.10^{-7}\ x\ 2\ x\ 2{,}108}{4\pi\ x\ 0{,}51} = 0{,}826.10^{-6} H/m$

Lex = $\frac{\mu}{\pi} ln\left(\frac{s}{d} + \sqrt{\frac{s^2}{d^2}} - 1\right)$

= $\frac{4\pi.10^{-7}}{\pi} ln\left(\frac{1{,}02}{0{,}51} + \sqrt{\frac{(1{,}02)^2}{(0{,}51)^2}}\right) = 0{,}5268.10^{-6}\ H/m$

L = Lin + Lex (H/km)

= $0{,}826.10^{-6}$ + $0{,}5268.10^{-6} = 1{,}353$ μH/m = 1353 μH/km

- **Redaman Saluran**

Redaman saluran transmisi kabel tembaga bergantung pada besar frekuensi dan nilai R, L, G, C. Maka nilai redaman saluran untuk frekuensi 1 kHz adalah :

α = $\frac{R}{2}\sqrt{\frac{C}{L}} + \frac{G}{2}\sqrt{\frac{L}{C}}$

= $\frac{53{,}34}{2}\sqrt{\frac{47{,}712.10^{-9}}{1353.10^{-6}}} + \frac{0{,}05997.10^{-6}}{2}\sqrt{\frac{1353.10^{-6}}{47{,}712.10^{-9}}}$ = 0,1584 Np/km

LdB = 8,686 x d x α

= 8,686 x 1 x 0,1584 = 1,3757 dB/km

- **Analisis Perhitungan S/N (Signal to Noise Ratio)**

Untuk perhitungan S/N (SNR) yang dilakukan dengan menggunakan kabel tembaga 24 AWG, menggunakan ketentuan sebagai berikut :

Konstanta kabel k1 = $3{,}8.10^{-3}$, k2 = $-0{,}541.10^{-8}$

Power Spectral Density transmitter (S(f)) = 1.10^{-4} mW/Hz

Power Spectral Density receiver (W) = $3{,}162277.10^{-13}$ mW/Hz

Frekuensi = 300 kHz

Jarak (d) = 4,886 km (3,05375 mil) maka akan didapat S/N (SNR) sebagai berikut :

$$\left(\frac{S}{N}\right) = \frac{S(f)|H(f)|^2}{W}$$

$$= \frac{S(f)x\ e^{-2d(k1\ x\sqrt{f}+k2\ x\ f)}}{W}$$

$$= \frac{1.10^{-4}\ x\ e^{-2\ x\ 3{,}05375(3{,}8.10^{-3}\ x\sqrt{300.10^{3}}-0{,}541.10^{-8}\ x\ 300.10^{3}}}{3{,}162277.10^{-13}}$$

= 963,011 = 29,84 dB

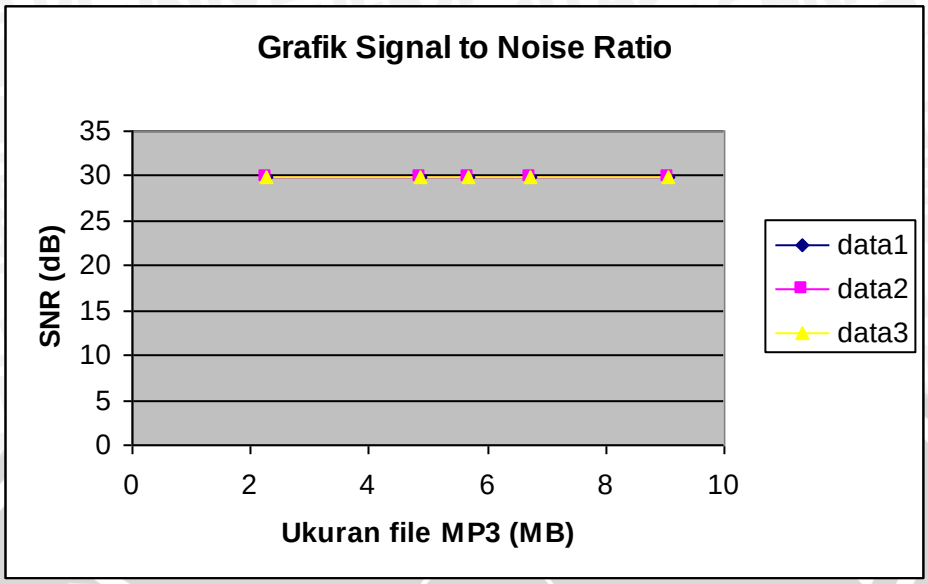
Tabel 5.2 ***Signal to Noise Ratio***

Ukuran *File* MP3 (dalam bit)	**Data 1 (1kb) (dB)**	**Data 2 (10kb) (dB)**	**Data 3 (20kb) (dB)**
18240000	**29,84**	**29,84**	**29,84**
38880000	**29,84**	**29,84**	**29,84**
45440000	**29,84**	**29,84**	**29,84**
53680000	**29,84**	**29,84**	**29,84**
72320000	**29,84**	**29,84**	**29,84**

Sumber : Hasil Pengujian

Dari tabel 5.2 diatas dapat dilihat pada data 1, data 2, dan data 3 untuk kapasitas *file carrier* atau *file* pembawanya yang semakin besar didapatkan nilai SNR yang sama yaitu 29.84 dB. Hal ini mengindikasikan bahwa SNR tidak dipengaruhi oleh besar data yang dikirimkan.

Grafik *Signal to Noise Ratio* :



Gambar 5.5 Grafik *Signal to Noise Ratio*

Sumber : Pengujian

Dari grafik diatas dapat dilihat bahwa untuk data 1, data 2, dan data 3 memiliki grafik yang sama untuk setiap kenaikan kapasitas dari *file carrier* atau *file* pembawanya. Hal ini menunjukkan bahwa nilai SNR tidak dipengaruhi oleh perubahan kenaikan ukuran *file* pembawa (MP3) dan perubahan besarnya kapasitas data yang disisipkan.

5.2.3 Pengujian Tingkat Keberhasilan Steganografi

Dari tabel SNR diatas dapat kita hitung tingkat keberhasilan dari steganografi yang telah kita uji coba yaitu sebesar :

$$= \left(\frac{jumlah\ percobaan\ yang\ dilakukan - jumlah\ percobaan\ gagal}{jumlah\ percobaan\ yang\ dilakukan}\right) * 100\%$$

$$= \frac{15-0}{15} * 100\ \% = 100\ \%$$

Dari hasil pengujian untuk proses *encoding* dan *decoding* tingkat keberhasilan yang dicapai adalah 100 %.

BAB VI
PENUTUP

6.1 Kesimpulan

Berdasarkan hasil pengujian tiap bagian sistem dan pengujian keseluruhan sistem yang telah dilakukan, dapat ditarik kesimpulan sebagai berikut :

1. Perangkat lunak yang telah dibuat memiliki tingkat keberhasilan untuk proses *encoding*/ proses penyisipan data dan proses *decoding*/ proses pengambilan kembali data yang telah disisipkan mencapai 100 %.
2. Penyisipan data dengan menggunakan metode LSB (*Low Significant Bit*) tidak mempengaruhi ukuran *file carrier*/ *file* MP3. Baik sebelum disisipi maupun setelah disisipi ukuran *file* tidak berubah. Untuk penyisipan data 1kb, 10kb, 20 kb pada *file* MP3 2.28 MB, ukuran *file* MP3 setelah disisipi data tetap 2.28 MB.
3. Semakin besar data yang disisipkan, maka semakin kecil kemungkinan penyisipan 1 bit data kedalam *file* MP3, hal ini terbukti dengan semakin kecil dan menurunnya nilai *Ratio Data Embedding* seperti terlihat pada tabel 5.1 dan grafik 5.4 yang nilainya semakin menurun. Untuk file MP3 2.28 MB nilai RED untuk data1 (1kb) adalah 1/2280 bit, data2 (10kb) adalah 1/228 bit dan data3 (20kb) adalah 1/114 bit.
4. Untuk tabel 5.2 dan grafik 5.5 menunjukkan nilai SNR yang tetap. Untuk file MP3 2.28 MB nilai SNR untuk data1 (1kb), data2 (10kb) dan data3 (20kb) adalah sama yaitu 29.84 dB. Hal ini menunjukkan bahwa nilai SNR tidak dipengaruhi oleh perubahan besarnya ukuran data yang disisipkan.

6.2 Saran

Perangkat lunak yang dibuat pada skripsi ini hanya menggunakan *file* audio (MP3) dan *file* teksnya menggunakan format *.txt dengan menggunakan pemrograman java J2SDK yang digunakan untuk aplikasi *dekstop* . Untuk itu perlu dikembangkan dikemudian hari menggunakan media *file* audio yang lain seperti wav dll. Untuk *file* data yang disisipkan tidak hanya dalam bentuk dokumen tetapi juga dapat berupa *file* gambar seperti jpeg, bitmap, png, dll. Diharapkan dengan menggunakan bahasa pemrograman java dapat memberikan kedinamisan dalam penggunaan jenis media *file* yang bervariasi.

DAFTAR PUSTAKA

Bender,W. Gruhl, D. M. N. L. (August 1998).” A.: Techniques for Data Hiding. PhD thesis.”.

Bender, D. Gruhl, N. A. L. (Februari 1996). “Techniques for data hiding. IBM System, 35(3-4).”

Munir, Rinaldi. (Agustus 2004). “Pengolahan Citra Digital dengan Pendekatan Algoritmik”.Bandung: Informatika Bandung

Kurniawan, Yusuf (April 2004). “Kriptografi: Keamanan Jaringan dan Jaringan Telekomunikasi”. Bandung: Informatika Bandung

Arnold, Michael, dkk. 2003.”Techniques and Applications of Digital Watermarking and content Protection”.

Wayner, Peter. 2002.” Disappearing Cryptography”USA: Elsevier Science

Tjong, Andreas. 2008,”Metode Low Bit *Coding*”. *http://andreastjong.wordpress.com/*

Anonymous..Kriptografi.*http://id.wikipedia.org/wiki/kriptografi.* Diakses tanggal 11 Agustus 2008

Anonymous..*Plaintext*.*http://id.wikipedia.org/wiki/plaintext.* Diakses April 2010

LAMPIRAN 1

LISTING PROGRAM MP3FRAME

1. MP3Frame :

```
import java.io.*;

/**
 * An MP3 sound data frame.
 */
public class MP3Frame
{
   public static final int MPEG_Version_2_5 = 0;
   public static final int MPEG_Version_2   = 2;
   public static final int MPEG_Version_1   = 3;

   public static final int MPEG_Layer_3 = 1;
   public static final int MPEG_Layer_2 = 2;
   public static final int MPEG_Layer_1 = 3;

   public static final int CHANNEL_MODE_STEREO      = 0;
   public static final int CHANNEL_MODE_JOINT_STEREO = 1;
   public static final int CHANNEL_MODE_DUAL_CHANNEL = 2;
   public static final int CHANNEL_MODE_MONO        = 3;

   public static final int EMPHASIS_NONE     = 0;
   public static final int EMPHASIS_50_15_MS = 1;
   public static final int EMPHASIS_RESERVED = 2;
   public static final int EMPHASIS_CCIT_J17 = 3;

   protected static final int[] MPEG1BitRates = { 0, 32000, 40000, 48000, 56000, 64000, 80000,
96000, 112000, 128000, 160000, 192000, 224000, 256000, 320000, 0 };
   protected static final int[] MPEG2BitRates = { 0,  8000, 16000, 24000, 32000, 40000, 48000,
56000,  64000,  80000,  96000, 112000, 128000, 144000, 160000, 0 };
```

```
    protected static final int[] MPEG10SampleRates = { 44100, 48000, 32000, 0 };
    protected static final int[] MPEG20SampleRates = { 22050, 24000, 16000, 0 };
    protected static final int[] MPEG25SampleRates = { 11025, 12000,  8000, 0 };

    private static int FRAME_SAMPLES_MPEG_1 = 1152;
    private static int FRAME_SAMPLES_MPEG_2 = 576;

    protected int     mpegVersion;
    protected int     mpegLayer;
    protected boolean isProtected;
    protected int     bitRate;
    protected int     sampleRate;
    protected boolean padded;
    protected int     channelMode;
    protected int     modeExtension;
    protected boolean copyrighted;
    protected boolean original;
    protected int     emphasis;
    protected byte[]  data;

    protected int bit_rate;
    protected int sample_rate;

    public int getBitRate()    { return bitRate; }
    public int getSampleRate() { return sampleRate; }
    public boolean isStereo()  { return channelMode != MP3Frame.CHANNEL_MODE_MONO;
}
    public int getDataLength() { return data.length; }

    public int getSamplesPerFrame()
    {
```

```
    if( mpegVersion == MP3Frame.MPEG_Version_1 ) return
MP3Frame.FRAME_SAMPLES_MPEG_1;
    return MP3Frame.FRAME_SAMPLES_MPEG_2;
  }

  /**
   * Read the next MP3 frame from the stream - return null if no more
   */
  public static MP3Frame readFrame( InputStream in ) throws IOException
  {
    MP3Frame frame = new MP3Frame();

    while( true )
    {
      int b = in.read();

      if( b < 0 ) return null;
      if( b == 0xFF )
      {
        b = in.read();

        if( b < 0 ) return null;
        if( (b & 0xE0) != 0xE0 ) continue;

        frame.mpegVersion = ( b & 0x18 ) >> 3;
        frame.mpegLayer   = ( b & 0x06 ) >> 1;
        frame.isProtected = ( b & 1 ) == 0;

        if( frame.mpegLayer != MPEG_Layer_3 ) continue;
        break;
      }
```

```
    }

    if( frame.isProtected )
    {
      in.read();
      in.read();
    }

    int b = in.read();
    if( b < 0 ) return null;

    frame.bit_rate = ( b & 0xF0 ) >> 4;

    if( frame.mpegVersion == MPEG_Version_1 ) frame.bitRate =
MPEG1BitRates[frame.bit_rate];
    else frame.bitRate = MPEG2BitRates[frame.bit_rate];

    frame.sample_rate = ( b & 0x0C ) >> 2;
    if ( frame.mpegVersion == MPEG_Version_1 ) frame.sampleRate =
MPEG10SampleRates[frame.sample_rate];
    else if( frame.mpegVersion == MPEG_Version_2 ) frame.sampleRate =
MPEG20SampleRates[frame.sample_rate];
    else frame.sampleRate =
MPEG25SampleRates[frame.sample_rate];

    frame.padded = ( b & 2 ) != 0;

    b = in.read();
    if( b < 0 ) return null;

    frame.channelMode  = ( b & 0xC0 ) >> 6;
```

```
    frame.modeExtension = ( b & 0x30 ) >> 4;

    frame.copyrighted = (( b & 0x80 ) >> 3) != 0;
    frame.original    = (( b & 0x40 ) >> 2) != 0;

    frame.emphasis = b & 0x02;

    int size = (((frame.mpegVersion == MPEG_Version_1 ? 144 : 72)
            * frame.bitRate ) / frame.sampleRate ) +
             (frame.padded ? 1 : 0) - 4;

    byte[] data = new byte[size];
    int read = 0;
    int r;

    while( (r = in.read( data, read, size - read )) >= 0 && read < size )
    {
       read += r;
    }

    if( read != size ) throw new IOException( "Unexpected end of MP3 data" );

    frame.data = data;

    //System.out.print( "." );
    return frame;
}

public MP3Frame( ) {}

public void write( OutputStream out ) throws IOException
```

```
{
    out.write( 0xff );

    int b = 0xE1;
    b |= this.mpegVersion << 3;
    b |= this.mpegLayer   << 1;
    out.write( b );

    b  = bit_rate << 4;
    b |= sample_rate << 2;
    if( padded ) b |= 2;
    out.write( b );

    b  = channelMode   << 6;
    b |= modeExtension << 4;
    b |= emphasis;
    out.write( b );

    out.write( data );
}

public String toString()
{
    String version = null;
    if     ( mpegVersion == MP3Frame.MPEG_Version_1   ) version = "1";
    else if( mpegVersion == MP3Frame.MPEG_Version_2   ) version = "2";
    else if( mpegVersion == MP3Frame.MPEG_Version_2_5 ) version = "2.5";

    return "MP3 Frame: " +
           " version=" + version +
           " bit-rate=" + bitRate +
```

```
            " sample-rate=" + sampleRate +
            " stereo=" +  (channelMode != MP3Frame.CHANNEL_MODE_MONO);
    }
    public void dumpData()
    {
            for (int i=0;i<this.data.length;i++){
                System.out.print(Integer.toHexString(data[i]&0xff)+" ");

            }
    }
}
```