BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Seiring dengan semakin meluasnya penggunaan jaringan internet, pengiriman informasi semakin rentan terhadap penyadapan yang dapat mengubah autentifikasi dan integritas data. Seringkali seseorang yang hendak mengirim pesan kepada orang lain, tidak ingin isi pesan tersebut diketahui oleh orang lain. Biasanya isi pesan tersebut bersifat sangat rahasia atau pribadi, yang hanya boleh diketahui antara pihak pengirim dan pihak penerima pesan, atau kalangan terbatas saja. Oleh karena itu, biasanya pengirim tersebut mengirim pesan secara sembunyi-sembunyi agar tidak ada pihak lain yang mengetahui.

Untuk mengurangi atau mencegah terjadinya hal diatas adalah mengembangkan suatu aplikasi yang mampu menyamarkan pesan tersebut pada suatu media yang dapat diakses oleh setiap orang. Teknik ini disebut Steganografi, yaitu teknik penyembunyian data pada suatu media. Setiap orang dapat menampilkan atau membuka media tersebut, namun tidak menyadari bahwa media tersebut telah dibubuhkan pesan rahasia oleh pengirim. Steganografi memungkinkan penyembunyian data pada berbagai jenis media digital seperti berkas citra, suara, video, dan teks.

Salah satu jenis multimedia yang banyak digunakan adalah berkas media MIDI (*Musical Instrumental DeviceInterface*), yang umum digunakan sebagai *polyponic ringtone* pada pesawat telepon selular. Media MIDI juga telah lama

digunakan sebagai latar musik sederhana dari berbagi perangkat lunak. Karena Banyaknya penggunaan media MIDI ini dan karakteristik berkas MIDI yang relatif berukuran kecil dibandingkan format berkas *audio* lainnya sehingga tidak menimbulkan kecurigaan yang terlalu berlebih. Maka, memungkinkan sekali apabila format MIDI ini dijadikan sebagai *container* dari data yang akan disembunyikan pada proses steganografi. Serta format MIDI berisi data musik yang berbasis *event* bukan sampel dari data suara yang akan dimainkan. Hal ini memungkinkan penyembunyian data juga berbasis *event.*

Ada beberapa metode steganografi yang banyak digunakan untuk menyisipkan informasi digital kedalam berkas MIDI. Yaitu *low bit encoding, phase coding, spread spectrum* dan *echo data* hiding. Teknik yang akan digunakan untuk implementasi steganografi pada berkas MIDI ini adalah Teknik *Phase coding* terhadap informasi yang akan disembunyikan.

Untuk memenuhi kebutuhan akan jaminan terhadap kerahasiaan data atau pesan, maka akan di bangun sebuah aplikasi yang dapat menyelesaikan permasalahan diatas yaitu ***"Aplikasi steganografi pada berkas midi dengan teknik phase coding ".***

1.2 Identifikasi Masalah

Berdasarkan uraian latar belakang diatas dapat diidentifikasikan masalah adalah Bagaimana membangun aplikasi *steganografi* pada berkas MIDI dengan menggunakan teknik *Phase coding* ?

1.3 Maksud dan Tujuan

Maksud dari penulisan Tugas Akhir ini adalah untuk membangun program aplikasi *steganografi* pada berkas MIDI dengan menggunakan teknik *Phase Coding*.

Sedangkan tujuan dari penulisan Tugas Akhir ini adalah untuk :

1. Untuk menyembunyikan pesan rahasia berupa text (*.txt) kedalam berkas MIDI (*.mid).
2. Untuk melakukan uji kehandalan terhadap deteksi dan *cracking* pada *stego file*.
3. Untuk menganalisis pengaruh ukuran informasi text (*.txt) yang disisipkan terhadap kualitas suara yang dihasilkan oleh MIDI (*.mid) dengan menggunakan metode Phase Coding.

1.4 Batasan Masalah

Adapun batasan masalah dalam penulisan Tugas Akhir ini adalah :

1. Sinyal audio yang akan digunakan sebagai berkas asli (*host file*) adalah berkas MIDI (*.mid).
2. Sebagai informasi yang disisipkan digunakan berkas teks (*.txt).
3. Teknik transformasi yang digunakan adalah teknik *Phase Coding* terhadap informasi yang akan disembunyikan.

4. Kunci digunakan untuk mengacak pesan adalah kunci simetris dengen menggunakan algoritma pembangkit bilangan pseudorandom *Linear Congruental Generator* (LCG) dan bit awal pembangkitan ditentukan berdasarkan perhitungan dari kunci masukan dari pengguna dengan panjang 32 bit.
5. Aplikasi steganografi pada berkas MIDI ini diimplementasikan secara *Stand alone*.

1.5 Metodologi Penelitian

Metodologi yang digunakan dalam penulisan tugas akhir ini adalah sebagai berikut :

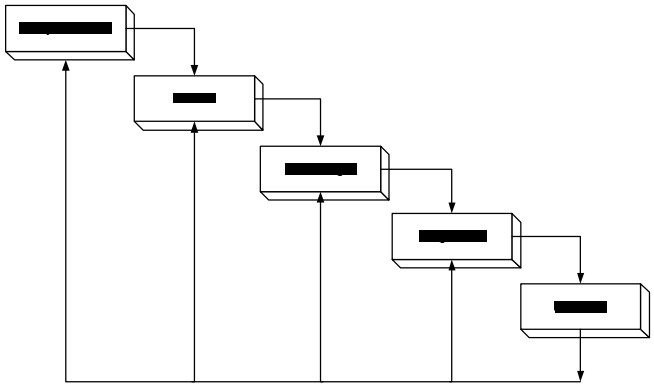
1.5.1. Metode Pengumpulan Data

a. Studi literatur, yaitu dilakukan dengan mengumpulkan dan mempelajari referensi (jurnal-jurnal, karya ilmiah, dan buku-buku yang diperoleh dari pustaka yang ada dan dari internet.) yang berhubungan dan mendukung implementasi *audio* steganografi.

b Eksperimental, yaitu perancangan dan implementasi sistem untuk membuat gambaran yang jelas dari masalah yang dihadapi dengan pembuatan program aplikasi menggunakan bahasa C dan kompilator yang dipakai adalah *Borland C++ Builder*.

1.5.2. Metode Pembuatan Perangkat Lunak

Metode pembuatan perangkat lunak yang digunakan adalah metode *sekuensial linier* yang sering disebut juga *waterfall.* Adapun alur model *waterfall* yang dapat digambarkan dengan diagram yang ada pada gambar 1.1.



Gambar 1.1 Diagram Waterfall

1. Rekayasa Sistem

   Merupakan bagian dari sistem yang terbesar dalam pengerjaan suatu proyek, dimulai dengan menetapkan kebutuhan – kebutuhan dari semua elemen yang diperlukan sistem dan mengalokasikannya kedalam pembentukan perangkat lunak.

2. Analisis

   Merupakan tahap dimana perekayasa menganalisis hal – hal yang diperlukan dalam pelaksanaan proyek pembuatan perangkat lunak.

3. Perancangan

   Tahap penerjemahan dari keperluan atau data yang dianalisis kedalam bentuk yang mudah dimengerti oleh user atau pemakai.

4. Pengkodean

   Tahap penerjemahan data atau pemecahan masalah yang telah dirancang ke dalam bahasa pemrograman yang ditentukan.

5. Pengujian

   Tahap pengujian dari perangkat lunak yang telah dibangun.

1.6 Sistematika Penulisan

Sistematika penulisan proposal penelitian ini disusun untuk memberikan gambaran umum tentang penelitian yang dijalankan. Sistematika penulisan tugas akhir ini adalah sebagai berikut :

BAB I PENDAHULUAN

Menguraikan tentang latar belakang masalah, maksud dan tujuan, identifikasi masalah, batasan masalah/ruang lingkup kajian, metode penelitian, serta sistematika penulisan per-bab yang merupakan gambaran umum

BAB II. LANDASAN TEORI

Membahas berbagai konsep dasar dan teori-teori yang berkaitan dengan teori-teori yang berhubungan dengan *steganografi* secara umum, *audio steganografi* secara khusus.

BAB III. ANALISIS DAN PERANCANGAN SISTEM

Bab ini membahas mengenai analisis dan perancangan sistem perangkat lunak yang akan dibangun, yang meliputi spesifikasi sistem dan fungsional sistem yang akan dibangun.

BAB IV. IMPLEMENTASI DAN PENGUJIAN SISTEM

Bab ini akan membahas mengenai penjabaran dari rancangan yang sudah dibuat dengan menggunakan data hasil analisis yang telah dijelaskan pada bab sebelumnya. Penjabaran dari perancangan sistem yang dimaksud adalah proses implementasi dalam bentuk sebuah aplikasi menggunakan perangkat lunak yang diperlukan. Selanjutnya dilakukan pengujian sistem dan analisis terhadap hasil yang diperoleh.

BAB V. KESIMPULAN DAN SARAN

Bab ini menguraikan kesimpulan dari penyelesaian permasalahan yang dibahas serta saran-saran yang diharapkan bermanfaat untuk pengembangan sistem selanjutnya.

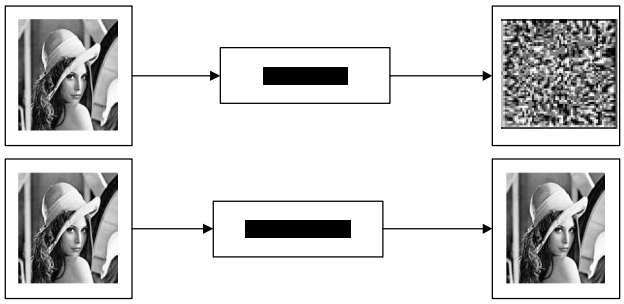
BAB II

LANDASAN TEORI

2.1 Definisi Steganografi

Steganografi (*steganography*) adalah ilmu dan seni menyembunyikan pesan rahasia (*hiding message*) sedemikian sehingga keberadaan (eksistensi) pesan tidak terdeteksi oleh indera manusia [5]. Kata steganorafi berasal dari Bahaya Yunani yang berarti "tulisan tersembunyi" (*covered writing*). Steganografi membutuhkan dua properti: wadah penampung dan data rahasia yang akan disembunyikan. Steganografi digital menggunakan media digital sebagai wadah penampung, misalnya citra, suara, teks, dan video. Data rahasia yang disembunyikan juga dapat berupa citra, suara, teks, atau video.

Steganografi berbeda dengan kriptografi, di mana pihak ketiga dapat mendeteksi adanya data (*chipertext*), karena hasil dari kriptografi berupa data yang berbeda dari bentuk aslinya dan biasanya datanya seolah-olah berantakan, tetapi dapat dikembalikan ke bentuk semula. Ilustrasi mengenai perbedaan kriptografi dan steganografi dapat dilihat pada gambar 2.1.

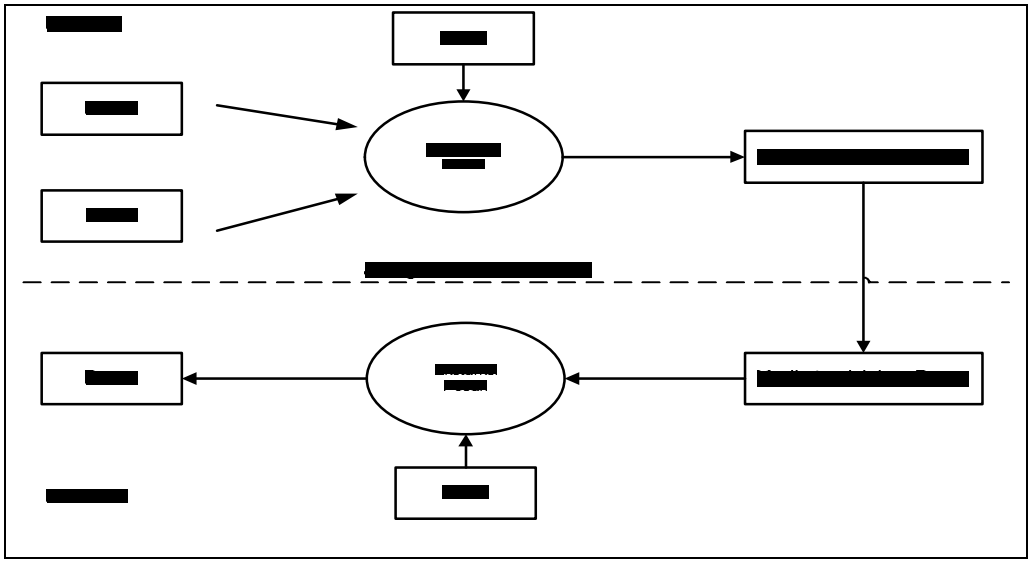


Gambar 2.1. Ilustrasi kriptografi dan steganografi pada citra digital.

Steganografi membahas bagaimana sebuah pesan dapat disisipkan ke dalam sebuah berkas media sehingga pihak ketiga tidak menyadarinya. Steganografi memanfaatkan keterbatasan sistem indera manusia seperti mata dan telinga. Dengan adanya keterbatasan inilah, metoda steganografi ini dapat diterapkan pada berbagai media digital. Hasil keluaran dari steganografi ini memiliki bentuk persepsi yang sama dengan bentuk aslinya, tentunya persepsi di sini sebatas oleh kemampuan indera manusia, tetapi tidak oleh komputer atau perangkat pengolah digital lainnya. Ilustrasi mengenai proses steganografi dapat dilihat pada gambar 2.2.

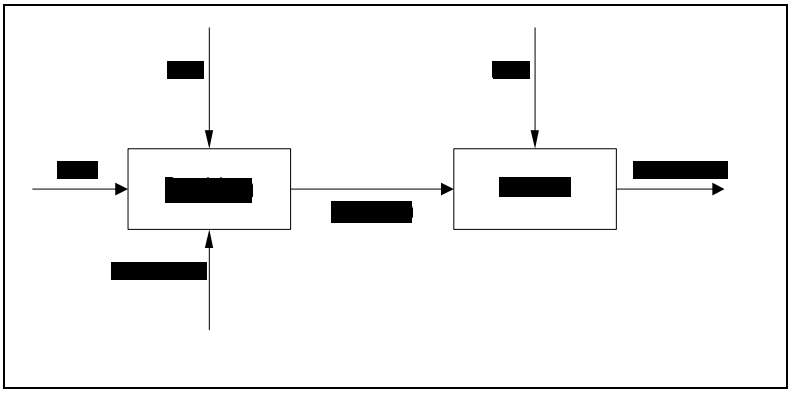
Penyembunyian data rahasia ke dalam media digital mengubah kualitas media tersebut. Kriteria yang harus diperhatikan dalam penyembunyian data diantaranya adalah [5]:

1. *Fidelity*. Mutu media penampung tidak jauh berubah. Setelah penambahan data rahasia pengamat tidak mengetahui kalau di dalam media tersebut terdapat data rahasia.
2. *Recovery*. Data yang disembunyikan harus dapat diungkapkan kembali (*recovery*). karena tujuan steganografi adalah *data hiding*, maka sewaktu-waktu data rahasia didalam media penampung harus dapat diambil kembali untuk digunakan lebih lanjut.



Gambar 2.2 Diagram Sistem Steganografi

Steganografi digital menggunakan media digital sebagai wadah penampung, misalnya citra, suara, teks, dan video. Sedangkan data rahasia yang disembunyikan dapat berupa berkas apapun. Media yang telah disisipi data disebut stegomessage. Proses penyembunyian data ke dalam media disebut penyisipan (*embedding*), sedangkan proses sebaliknya disebut ekstraksi. Proses tersebut dapat dilihat pada gambar 2.3. Penambahan kunci yang bersifat opsional dimaksudkan untuk lebih meningkatkan keamanan [10].



Gambar 2.3 Proses penyisipan dan ekstraksi dalam steganografi

Sejarah Steganografi

Steganografi sudah dikenal oleh bangsa Yunani. Herodatus, penguasa Yunani, mengirim pesan rahasia dengan 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 balik rambutnya.

Bangsa Romawi 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 di atas kertas dapat dibaca dengan cara memanaskan kertas tersebut. Saat ini di negara-negara yang melakukan penyensoran informasi, steganografi sering digunakan untuk menyembunyikan pesan-pesan melalui gambar (*images*), video, atau suara (*audio*).

Aplikasi Steganografi

Steganografi sebagai suatu teknik penyembunyian informasi pada data digital lainnya dapat dimanfaatkan untuk berbagai tujuan seperti :

1. *Tamper-proofing* dimana steganografi digunakan sebagai alat untuk mengidentifikasikan atau alat indikator yang menunjukkan data *host* telah mengalami perubahan dari aslinya.
2. *Feature location* dimana steganografi digunakan sebagai alat untuk mengidentifikasikan isi dari data digital pada lokasi-lokasi tertentu, seperti

contohnya penamaan objek tertentu dari beberapa objek yang lain pada suatu citra digital.

3. *Annotation/caption* dimana steganografi hanya digunakan sebagai keterangan tentang data digital itu sendiri.
4. *Copyright-Labeling* dimana steganografi dapat digunakan sebagai metoda untuk penyembunyian label hak cipta pada data digital sebagai bukti otentik kepemilikan karya digital tersebut.

Media Steganografi

Steganografi menggunakan sebuah berkas yang disebut dengan *cover*, tujuannya sebagai kamuflase dari pesan yang sebenarnya. Banyak format berkas digital yang dapat dijadikan media untuk menyembunyikan pesan. Pada jaman modern seperti saat ini, steganografi biasanya dilakukan dengan melibatkan berkas-berkas seperti teks, audio, dan gambar.

Steganografi pada Teks

Teknik steganografi yang menggunakan teks sebagai *cover* adalah hal yang menantang. Ini dikarenakan berkas teks memiliki ukuran data yang kecil untuk bisa digantikan dengan berkas rahasia. Dan kekurangan lainnya adalah teks yang mengandung teknik steganografi ini dengan mudah dapat diubah oleh pihak yang tidak diinginkan dengan cara mengubah teks itu sendiri maupun mengubah format dari teksnya (misal .TXT menjadi .PDF). Ada beberapa metode yang digunakan pada media teks ini yaitu, *Line-Shift Encoding, Word-shift Encoding*

dan *Feature Coding*. Ketiganya merupakan metode *encoding* yang membutuhkan berkas asli dan juga format aslinya untuk dapat di*decode* atau diekstrak kembali

Steganografi pada Gambar

Steganografi pada gambar adalah metode yang paling banyak digunakan secara luas didunia digital saat ini. Hal ini dikarenakan keterbatasan kemampuan dari visual atau *Human Visual System* (HVS). Format gambar yang biasanya digunakan adalah format bitmap (bmp), gif, pcx, jpeg dan format gambar lainnya.

Hampir semua *plain* teks, *cipher* teks, gambar dan media lainnya dapat di*encode* kedalam aliran bit untuk disembunyikan didalam gambar digital. Perkembangan dari metoda ini sangat pesat, didukung dengan semakin canggihnya komputer grafik yang *powerful*, dan software steganografi yang sekarang sudah banyak tersebar luas di internet. Pendekatan yang paling sering dilakukan pada media jenis ini adalah : *Least Significant Bit Insertion, Masking and Filtering* dan *Algorithm and Transformation*. Sebagai catatan, masih ada banyak lagi teknik-teknik yang digunakan untuk *encoding* pada media gambar ini.

Steganografi pada Audio

Penyembunyian data pada audio merupakan teknik yang paling menantang pada steganografi ini. Hal ini disebabkan *Human Auditory System* (HAS) memiliki jangkauan yang dinamis. HAS memiliki kemampuan mendengar lebih dari satu sampai 1 miliar. Dan jangkauan frekuensi lebih dari satu hingga seribu. *Auditory System* ini juga sangat peka pada gangguan suara (*noise*) yang

halus sekalipun. Sedikit saja terdapat gangguan pada sebuah berkas audio maka dengan mudah akan terdeteksi. Satu-satunya kelemahan yang dimiliki HAS dalam membedakan suara adalah kenyataan bahwa suara keras bisa menenggelamkan suara pelan. Terdapat dua konsep yang harus dipertimbangkan sebelum memilih metoda mana yang akan dipakai. Yaitu format digital audio dan media transmisi dari audio.

Metode-metode Steganografi pada Audio

Terdapat empat *encoding* yang paling terkenal dalam menyembunyikan data pada format ini, yaitu : *low-bit encoding, phase-coding, spread spectrum* dan *echo data hiding*.

Low Bit Encoding

Metode steganografi yang paling umum pada tipe berkas audio dan gambar adalah *least significant bit* atau disebut juga *Low Bit Encoding*. Metode ini berasal dari angka yang paling kurang signifikan dari jumlah bit dalam 1 byte. Bit yang memiliki signifikansi paling tinggi adalah numerik yang memiliki nilai tertinggi (misal, $2^7 = 128$) , artinya yang paling tidak signifikannya adalah yang memiliki nilai terrendah (misal, $2^0 = 1$). Sebagai contoh, kita akan menyisipkan karakter ‘G’ pada 8 byte berkas *carrier* atau bit dari media pembawa. *Low Bit Encoding* ditandai dengan garis bawah.

10010101 00001101 11001001 10010110

00001111 11001011 10011111 00010000

01000111 adalah bentuk huruf G dalam biner. kedelapan bit ini bisa dituliskan kedalam *Low Bit Encoding* dari tiap-tiap byte pada kedelapan *carrier* seperti berikut ini :

1001010*0* **0000110*1*** **1100100*0*** **1001011*0***

0000111*0* **1100101*1*** **1001111*1*** **0001000*1***

Pada contoh diatas, hanya sebagian dari *Low Bit Encoding* yang berubah (ditunjukkan dengan karakter miring). Dengan mengubah *Low Bit Encoding* , nilai dari byte tidak akan berubah banyak, sehingga akan sulit dideteksi oleh telinga manusia.

Phase Coding

Metode *Phase coding* adalah metode yang mensubstitusi bagian inisialisasi dari sebuah berkas audio dengan sebuah *phase* lain yang berisikan data yang akan disembunyikan. Metode ini memanfaatkan kelemahan system pendengaran manusia untuk merasakan fase absolut yang mengabaikan suara yang lebih lemah jika dua suara itu datang bersamaan.

Phase coding adalah salah satu metode dalam audio *watermarking* yang bekerja berdasarkan karakteristik sistem pendengaran manusia *(Human Auditory System),* yang mengabaikan suara yang lebih lemah jika dua suara itu datang bersamaan. Data *watermark* dibuat menjadi derau dengan amplitudo yang lebih lemah dibandingkan amplitudo data audio, kemudian keduanya digabungkan.

Ide dasar dibalik teknik *phase coding* adalah menyembunyikan data dengan cara menukarkan fase asli segmen inisial dari sinyal suara dengan fase absolut dari sinyal *watermark* dengan tetap menjaga fase relatif antara segmen sinyal menggunakan beda fase segmen dari sinyal asli. Ketika beda fase antara sinyal asli dan sinyal yang dimodifikasi besarnya kecil, maka perbedaan suara yang dihasilkan tidak terdeteksi oleh pendengaran manusia.

Phase coding termasuk dalam kelompok teknik audio *steganografi* berbasis domain frekuensi yang bekerja dengan cara mengubah *spectral content* dalam *domain* frekuensi dari sinyal. *Phase coding* merupakan metode paling efektif dari segi perbandingan *noise signal-to-perceived.*

Dalam pembahasan berikutnya akan dijelaskan mengenai parameter apa saja yang mempengaruhi *phase coding*, bagaimana cara menyisipkan *watermark* ke dalam data audio (enkoding), dan cara pengekstraksian *watermark* dari data audio yang telah disisipi (dekoding) dengan teknik *phase coding*.

Spread Spcetrum

Encoding dari *spread spectrum* pada audio ini melibatkan keseluruhan dari spektrum frekuensinya. Kemudian mentransmit audio tersebut melalui frequensi yang berbeda-beda tergantung dari metode *spread spectrum* yang digunakan. *Direct Sequence Spread Spectrum* (DSSS) adalah salah satu metode *spread spectrum* yang digunakan dalam menyebarkan sinyal dengan melipatgandakan sumber sinyal dengan beberapa bagian *pseudorandom* yang dikenal dengan nama chip. *Sampling rate* dari sumber sinyal ini kemudian akan digunakan sebagai *rate*

dari chip dalam proses *coding* nya. Teknik s*pread spectrum* ini adalah teknik paling aman untuk mengirimkan pesan didalam audio, tapi teknik ini juga menimbulkan *noise* secara acak pada audio, dan hal ini berarti ada beberapa data asli dari audio tersebut hilang.

Echo Data Hiding

Teknik e*cho data hiding* dilakukan dengan cara menambahkan data kedalam sumber sinyal dengan cara menggunakan echo. Data disembunyikan dengan tiga parameter echo yang berbeda-beda: *Initial Amplitude, Decay rate dan offset*, atau *delay*. Ketika *offset* antara data audio asli dengan echo nya berkurang, maka dua sinyal akan berbaur. Pada saat-saat tertentu. Pendengaran manusia tidak dapat membedakan antara dua sinyal dan echo biasanya hanya dianggap resonansi saja. Penggunaan metode ini bergantung pada beberapa faktor seperti kualitas dari rekaman aslinya, tipe audio, dan tentu saja pendengarnya.

2.2 Definisi Kriptografi

Kriptografi adalah ilmu sekaligus seni untuk menjaga kerahasiaan pesan (data atau informasi) dengan cara menyamarkannya menjadi bentuk yang tidak dapat dimengerti. Tujuan penyandian adalah agar isi pesan tidak dapat dimengerti oleh orang yang tidak berhak. Kehidupan saat ini dikelilingi oleh kriptografi, diantaranya mulai dari transmisi pada teknolgi nirkabel *bluetooth,* ATM, telepon genggam, jaringan komputer di kantor, *internet*, sampai ke pangkalan militer.

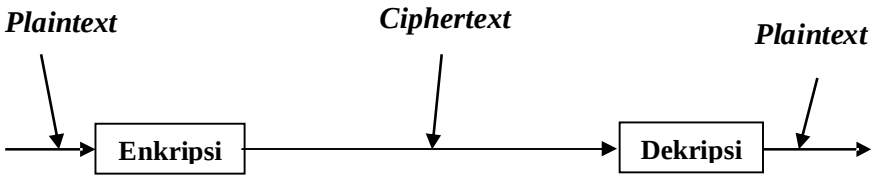
Kriptografi merupakan ilmu dan seni untuk menjaga pesan agar aman. "*Crypto*" berarti "*secret*" (rahasia) dan "*graphy*" berarti "*writing*" (tulisan) [6]. Para pelaku atau praktisi kriptografi disebut *cryptographers*. Sebuah algoritma kriptografik (*cryptographic algorithm*), disebut *cipher*, merupakan persamaan matematik yang digunakan untuk proses enkripsi dan dekripsi. Biasanya kedua persamaan matematik (untuk enkripsi dan dekripsi) tersebut memiliki hubungan matematis yang cukup erat.

2.2.1 Terminologi Dasar Kriptografi

Untuk mengenal kriptografi lebih jauh, ada beberapa terminologi yang harus diketahui terlebih dulu sebagai berikut :

1. Plainteks (*plaintext* atau *cleartext*, artinya teks jelas yang dapat dimengerti), yaitu pesan asli yang akan disandikan.
2. Chiperteks (*ciphertext* atau *cryptogram*, artinya teks tersandi), yaitu pesan hasil penyandian.
3. Enkripsi (*encryption* atau *enchipering*), yaitu proses penyandian dari plainteks ke chiperteks.
4. Dekripsi (*decryption* atau *dechipering*), yaitu proses pembalikan dari chiperteks ke plainteks.

Secara sederhana proses enkripsi/ dekripsi dapat digambarkan seperti pada gambar 2.4.



Gambar 2.4 Proses Enkripsi dan Dekripsi Sederhana

5. Algoritma kriptografi (*cipher*), yaitu aturan untuk *enciphering* dan *deciphering*. Algoritma kriptografi dapat diartikan juga sebagai fungsi matematika yang digunakan untuk enkripsi dan dekripsi.
6. Kunci, adalah parameter yang digunakan untuk transformasi *ciphering*.
7. Sistem Kriptografi (*cryptosystem*), yaitu algoritma kriptografi beserta kuncinya.
8. Penyusup (*intruder* / *interceptor*), adalah orang yang mencoba menangkap pesan selama ditransmisikan. Nama lain dari penyususp ini adalah *enemy*, *eavesdropper*.
9. Kriptografer, yaitu orang yang menggunakan algoritma kriptografi untuk merahasiakan pesan dan mendekripsikannya kembali.
10. Kriptanalisis (*cryptanalysis*), adalah ilmu dan seni untuk memecahkan chiperteks, berupa proses untuk memperoleh *plaintext* dari *ciphertext* tanpa mengetahui kunci yang diberikan. Pelakunya disebut kriptanalis

11. Kriptanalis. Persamaan kriptografer dan kriptanalis adalah bahwa keduanya sama-sama menerjemahkan *ciphertext* menjadi *plaintext*. Perbedaannya, kriptografer bekerja atas legitimasi pengirim atau penerima pesan, sedangkan kriptanalis bekerja atas nama penyusup yang tidak berhak.
12. Kriptologi (*cryptology*), yaitu studi mengenai kriptografi dan kriptanalisis.

2.2.2 Tujuan Kriptografi

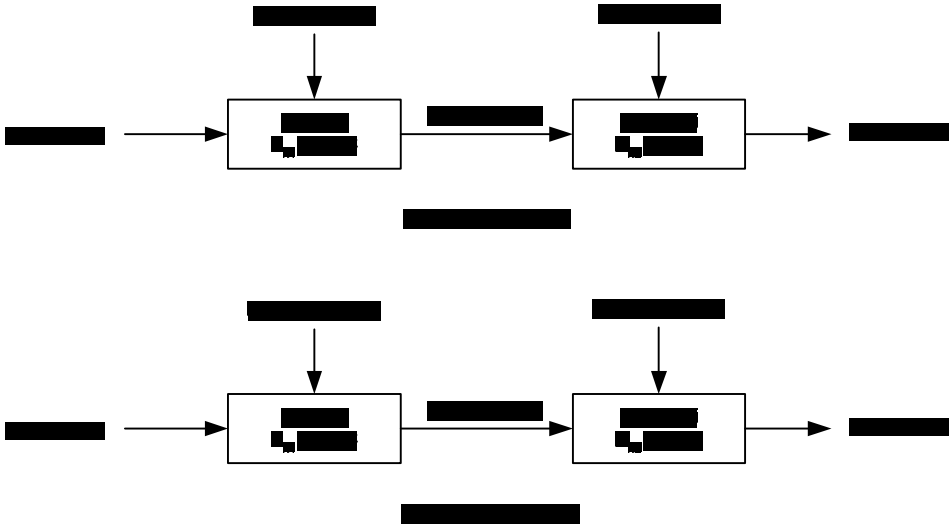
Kriptografi bertujuan untuk memberi layanan keamanan yang juga dinamakan aspek-aspek keamanan. Keamanan komputer jaringan melingkupi empat aspek utama yaitu *privacy, integrity, autentication* dan *availabilty* .

Selain untuk menjaga kerahasiaan (*confidentiality*) pesan, kriptografi juga digunakan untuk menangani masalah keamanan yang mencakup dua hal berikut:

1. Keabsahan pengirim (*user authentication*). Hal ini berkaitan dengan keaslian pengirim. Dengan kata lain, masalah ini dapat diungkapkan sebagai pertanyaan:“Apakah pesan yang diterima benar-benar berasal dari pengirim yang sesungguhnya?”
2. Keaslian pesan (*message authentication*). Hal ini berkaitan dengan keutuhan pesan (*data integrity*). Dengan kata lain, masalah ini dapat diungkapkan sebagai pertanyaan: “Apakah pesan yang diterima tidak mengalami perubahan (modifikasi)?”
3. Anti-penyangkalan (*nonrepudiation*).Pengirim tidak dapat menyangkal (berbohong) bahwadialah yang mengirim pesan.

2.2.3 Kunci Dalam Kriptografi

Berdasarkan kuncinya kriptografi dibagi menjadi dua yaitu kriptografi kunci simetris dan asimetris, sebagai penekanan dapat dilihat pada gambar 2.5.



Gambar 2.5 Kunci dalam Kriptografi

Jika kunci enkripsi sama dengan kunci dekripsi, maka sistem kriptografinya disebut **sistem simetri** atau **sistem konvensional**. Algoritam kriptografinya disebut algoritma simetri atau algoritma konvensional. Contoh algoritma simetri: *DES* (*Data Encyption Standard*).

Beberapa sistem kriptografi menggunakan kunci yang berbeda untuk enkripsi dan dekripsi. Misalkan kunci enkripsi adalah *K*1 dan kunci dekripsi yang adalah *K*2, yang dalam hal ini *K1* ¹ *K2*. Sistem kriptograsi semacam ini dinamakan sistem sistem asimetri atau sistem kunci-publik. Algoritam kriptografinya disebut algoritma asimetri atau algoritma kunci-publik. Contoh algoritma asimetri: *RSA* (*Rivest-Shamir-Adleman*)

2.2.4 Pembangkit Bilangan Pseudorandom

Bilangan acak (*random*) banyak digunakan dalam kriptografi. Pembangkitan bilangan acak pada steganografi, dapat digunakan untuk menentukan kunci penyisipan dan ekstraksi data dari berkas media. Namun, sangat sulit untuk menghasilkan bilangan yang benar-benar acak dengan menggunakan komputer. Komputer hanya mampu menghasilkan bilangan semu acak (*pseudorandom*). Deret bilangan *pseudorandom* adalah deret bilangan yang kelihatan acak dengan kemungkinan pengulangan yang sangat kecil atau periode pengulangan yang sangat besar.

2.2.4.1 *Linear Congretial Generator*

Salah satu algoritma pembangkitan bilangan pseudorandom adalah *Linear Congruental Generator* (*LCG*). Algoritma ini diciptakan oleh D. H. Lehmer pada tahun 1951. Deret bilangan bulat dalam *LCG* diformulasikan sebagai berikut :

$$Zi = (aZi_{-1} + c) \pmod{m} \qquad (2.1)$$

Dalam hal ini :

Zi : bilangan bulat ke-*i*

a : bilangan pengali

c : bilangan penambah

m : modulus

*Z*0 : nilai awal berupa bilangan bulat tak negatif

Dengan demikian nilai *Zi* terdefinisi pada :

$0 = Zi = m\text{-}1$, $i = 1, 2, 3, \ldots$

Untuk memulai bilangan acak ini dibutuhkan sebuah bilangan bulat *Z*0, yang dijadikan sebagai nilai awal (bibit pembangkitan). Bilangan acak pertama yang dihasilkan selanjutnya menjadi bibit pembangkitan bilangan bulat acak selanjutnya. Jumlah bilangan acak yang tidak sama satu sama lain (unik) adalah sebanyak *m*. Semakin besar nilai m, semakin kecil kemungkinan akan dihasilkan nilai yang sama.

Sebagai contoh : $Zn = (7Z_{i-1} + 11) \bmod 17 \; ; \; Z_0 = 0$

Rangkaian bilanagan acak yang dibangkitkan oleh persamaan diatas adalah

n	**Z_n**
0	0
1	11
2	3
3	15
4	14
5	7
6	9
7	6
8	2
9	8
10	16
11	4

12	5
13	12
14	10
15	13
16	0
17	11
18	3
19	15
20	14
21	7
22	9
23	6
24	2

Priode *LCG* diatas adalah 16, sebab setelah 16 kali perhitungan bilanagan acaknya terulang kembali. *LCG* mempunyai priode tidak lebih besar dari *m*, dan pada kebanyakan kasus periode penuh (*m - 1*) jika memenuhi syarat berikut [6]:

1. *c* relatif prima terhadap *m.*
2. *a – 1* dapat dibagi dengan semua faktor prima dari *m.*
3. *a – 1* adalah kelipatan 4 jika *m* adalah kelipatan 4.
4. *m* > maks (*a, b, Z_0*).
5. *a > 0, c > 0.*

Meskipun *LCG* secara teoritis mampu menghasilkan bilangan acak yang cukup, namun *LCG* sangat sensitif terhadap pemilihan nilai- nilai *a, c* dan *m.* Pemilihan nilai –nilai yang buruk dapat mengarah pada implementasi *LCG* yang tidak bagus.

Keunggulan *LCG* terletak pada kecepatannya dan hanya membutuhkan sedikit operasi bit. Akan tetapi, *LCG* tidak dapat digunakan untuk kriptografi karena bilangan acaknya dapat diprediksi urutan kemunculannya. Oleh karena itu *LCG* tidak aman digunakan untuk kriptografi. Namun demikian, *LCG* tetap berguna untuk aplikasi non-kriptografi.

2.3 C++ Builder

C++ Builder adalah sebuah aplikasi yang digunakan untuk pengembangan dengan memanfaatkan keistimewaan konsep – konsep antar muaka grafis dalam Microsoft windows. Aplikasi yang dihasilkan C++ bulder berkaitan erat dengan

windows itu sendiri sehingga dibutuhkan pengetahuan bagaimana cara kerja dari windows. C++ adalah aplikasi pemograman yang menghasilkan program aplikasi yang berjalan di bawah lingkungan windows 9.x maupun yang berjalan dilingkungan windows NT. C++ merupakan bahasa pengembangan dari bahasa C sebelumnya yaitu C++. C++ merupakan merk dagang atau produk dari Borland International Inc, yang telah banyak mengahsilkan bahsa pemoggraman yang sngat handal. Seperti layaknya bahsa pemograman yang berobjek visual lainnya C++ builder ini mempunyai kelebihana antara lain digolongkan kedalam kelompok RAD (rapid Aplication Development), yang merupakan perangkat pemograman yang mampu membuat aplikasi dengan mudah dan cepat. C++ builder juga tergolong dalam kelompok bahasa pmograman yang menggunakan prinsip event driven. Keunggulan lain dari C++ builder ini adalah dalam pengembangan DataBase dalam pemograman databaseini, terutama program yang sangat sederhana sekali,anda sudah dapat menjalankan program tersebut tanpa harus menuliskan sebaris kode. Jika anda sudah pernah menggunakan Borland Delphi , Visual Basic, ataupun bahasa visual lainnya , maka anda akan dapat dengan mudah mempelajari C++ builder ini, perbedaanaya hanya pada struktur pemogramannya saja, yang perlu diingat adalah kelompok pemograman C,C++, dan C++ builder merupakan bahasa pemograman yang menggunakan Case sensitive yaitu program akan mengecek penggunaan huruf besar maupun huruf kecil.

2.4 Bahasa C

2.4.1 Sejarah C

Akar dari bahasa C adalah dari bahasa BCPL ysng dikembangkan oleh Martin Richard pada tahun 1967. Bahasa ini memberikan ide kepada Ken Thompson yang kemudian mengembangkan bahasa yang disebut dengan B pada tahun 1970. Perkembangan selanjutnya dari bahasa B adalah bahsa C oleh Dennis Ricthie sekitar tahun 1970an di Bell TelephoneLaboratories Inc. (sekarang adalah AT&T Bell Laboratories). Bahsa C pertama kali digunakan oleh computer Digital Equipment Corporation PDP-!! Yang menggunakan system operasi UNIX.

C adalah bahasa yang standar, artinya suatu program yamg ditulis dengan versi bahas C tertentu akan dapat dikompilasidengan bahasa C yang lainnya dengan sedikit modifikasi. Standar bahasa C yang asli adalah standar dari UNIX. Patokan standar dari UNIX ini diambil dari buku yang ditulis oleh Brian Kerninghan dan Dennis Ritchie berjudul “the C Programming Language”, diterbitkan oleh Prentice-Hall tahun 1987. Deskripsi C dari kerninghan dan Ritchie ini kemudian dikenal secara umum sebagai “K&R C”.

2.4.2 Struktur Program C

Untuk dapat memahami bagaimana suatu program ditulis, maka struktur dari program harus dimengerti terlebih dahulu. Tiap bahasa computer mempunyai struktur program yang berbeda. Jika struktur dari suatu program tidak diketahui , maka akan sulit bagi pemula untuk memulai menulis suatu program debgab bahasa yang bersangkutan. Struktur dari program memberikan gambaran secara

luas, bagaimana bentuk dari program secara umum. Selanjutnya dengan pedoman struktur program ini, penulis program dapat memulai bagaimana seharusnya program tersebut ditulis.

Struktur program dari C dapat dilihat sebagai kumpulan dari sebuah atau lebih fungsi –fungsi . Fungsi pertama yang harus ada di program C sudah ditentukan namanya, yaitu bernama main(). Suatu fungsi di program C dibuka dengan kurung kurawal ({) dn ditutup dengan kurung kurawal tutup (}). Diantara kurung – kurung kurawal dapat dituliskan denga statement-statement program C. Berikut inin adalah struktur dari program C.

```
Main()
{
        Statement-statement;        Fungsi utama
}
Fungsi_fungsi lain()
{                                   fungsi-fungsi lain
Statement-statement                 yang ditulis oleh
}                                   pemogram computer
```

Bahasa C dikatakan sebagai bahasa pemograman terstruktur karena strukturnya menggunakan fungsi-fungsi sebagai program-program bagian (*subroutine*). Fungsi –fungsi selain fungsi utama merupakan program-program bagian. Fungsi-fungsi dapat ditulis setelah fungsi utama atau diletakan di file pustaka (*library*). Jika fungsi-fungsi diletakan di file pustaka dan akan dipakai di

suatu program, maka suatu file judulnya *(header file*) harus dilibatkan didalam program yang menggunakannya dengan *processor directive* #include.

2.4.3 Operator

Operator atau tanda operasi adalah suatu tanda atau symbol yang digunakan suatu operasi tertentu. Bahasa C merupakan bahasa yang kaya dengan operator. Bahasa C menyediakan banyak sekali tanda-tanda operasi, yaitu operator logika, operator pengerjaan, operator aritmetika, operator tipe, operator hubungan, operator *bitwise*, ternary operator dan operator koma.

2.4.3.1 Operator *Bitwise*

Operator – operator bitwise digunakan untuk memanipulasi bit-bit dari nilai data yang ada di memori. Karena bahasa C memang dimaksudkan untuk dapat beroperasi secara level rendah, sepertin halnya bahasa mesin atau bahasa perakit, maka operator-operator bitwise ini sangat diperlukan keberadaanya. Operator-operator ini dan artinya dapat dilihat pada di tabel 2.6.

Tabel 2.1 Operator-operstor bitwise

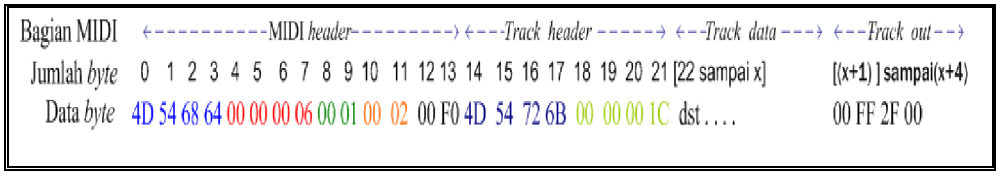
Operator	Arti
<<	Pergeseran bit kekiri
>>	Pergeseran bit kekanan
&	Bitwise AND
^	Bitwise XOR (exclusive OR
|	Bitwise OR

Disamping lima macam operator bitwise ini , sebuah operator lagi dapat digunakan untuk operasi bit , yaitu operator unary ‘~’ (*bitwise* NOT). Operator-operator bitwise hanya dapat digunakan untuk tipe char , int dan long int saja.

2.5 Spesifikasi MIDI

Format MIDI didesain secara generic agar setiap data penting dapat dibaca oleh semua jenis *sequencer*. Data pada fomat berkas MIDI disimpan dalam sebuah *chunk* (kumpulan byte yang dikenali berdasarkan ID dan ukurannya) yang dapat di-*parsing*, di-*load*, atau dilewati. Oleh karenanya, format SMF cukup fleksibel bagi *sequencer* tertentu untuk menyimpan datanya sendiri sedemikian rupa sehingga *sequencer* lainnya tidak pusing untuk membuka file tersebut, serta mengabaikan data yang tidak dibutuhkannya.

Data selalu disimpan dalam *chunk*. Ada berbagai macam *chunk* di dalam sebuah berkas MIDI. Ukuran *chunk* berisi informasi tentang jumlah byte (8-bit) dalam *chunk* tersebut. Data dalam *chunk* biasanya berhubungan dengan suatu maksud tertentu. Contohnya sebuah byte dalam suatu *chunk* berisi data untuk sebuah *track* dari suatu *sequencer*, dan data untuk *track sequencer* lainnya disimpan dalam sebuah *chunk* yang berbeda. singkatnya *chunk* berisi sekumpulan byte data yang saling berhubungan. Sebagai contoh sebuah *chunk* dari berkas MIDI secara garis besar dapat di gambarkan sebagai berikut :



Gambar 2.6. Chunk berkas MIDI

Setiap *chunk* harus diawali dengan 4 karakter ID yang menyatakan tipe dari *chunk* tersebut. 4 byte selanjutnya berisi informasi tentang ukuran *chunk* tersebut (panjang *chunk* sebesar 32 bit). Jadi *header* dari setiap *chunk* berisi dua macam informasi tersebut.

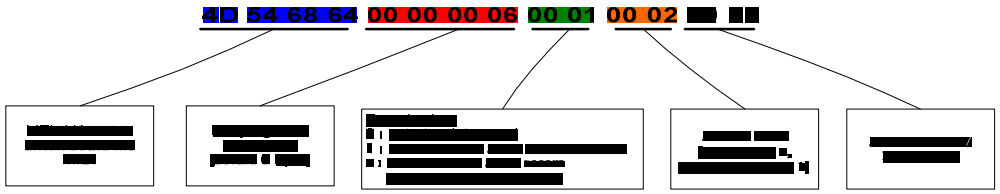
Semua *chunk header* dari sebuah file MIDI mempunyai ID “MThd” dengan ukuran *chunk* 6. Apabila dituliskan secara heksadesimal, sebuah *header* berkas MIDI menjadi sebagai berikut :

4D 54 68 64 00 00 00 06 00 01 00 02 00 F0

Adapun 6 byte data *header* ini terdiri atas :

1. 2 byte berisi tipe format. Ada 3 tipe format, yaitu :
2. tipe 0, berkas terdiri atas satu track tunggal berisi data MIDI yang mungkin pada semua (16) *channel* MIDI.
3. tipe 1, berkas terdiri atas satu atau lebih *track* secara simultan.
4. tipe 2, berkas terdiri atas satu atau lebih *track* secara independen dan sekuensial.
5. 2 byte selanjutnya menyatakan jumlah *track* yang ada dalam berkas tersebut. Tentu saja untuk tipe format 0, jumlahnya hanya 1, sedangkan untuk dua tipe format lainnya bisa lebih dari satu.

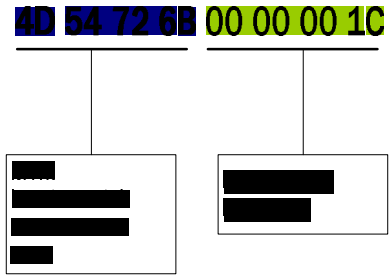
1. 2 byte terakhir mengindikasikan jumlah resolusi *Pulse per Quarter Note* (PPQN) atau dalam istilah musik berarti menentukan jumlah birama/ketukannya (*Division*)
2. Contoh dari sebuah *chunk* MThd dijelaskan pada gambar 2.7.



Gambar 2.7. Format sebuah chunk Mthd pada berkas MIDI

Setelah *chunk* MThd, maka akan ditemui *chunk* MTrk yang merupakan ID *chunk* lainnya yang telah didefinisikan selain *chunk* MThd. Jika ada ID *chunk* selain kedua jenis di atas, maka itu adalah tipe *chunk* yang dihasilkan oleh program lainnya, sehingga dapat diabaikan.

Chunk MTrk berisi seluruh data MIDI, dan data non-MIDI yang bersifat opsional, untuk satu *track*. Adapun jumlah *chunk* MTrk yang akan ditemui dalam sebuah berkas MIDI sesuai dengan jumlah *track* yang telah ditentukan dalam *header* berkas MIDI (NumTracks pada *chunk* MThd). *Header* MTrk diawali dengan ID "MTrk" diikuti dengan panjang data *chunk* tersebut. Adapun contoh dari sebuah *header chunk* MTrk dijelaskan pada gambar 2.8.



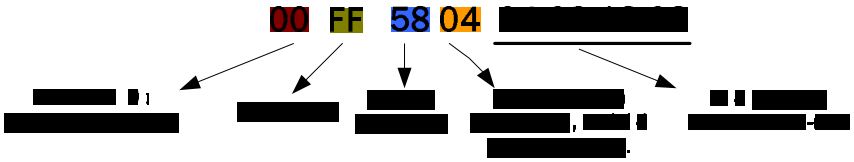
Gambar 2.8. Format sebuah chunk Mtrk pada berkas MIDI

Panjang *track* pada *header* MTrk menandakan jumlah byte dalam *track* hingga dijumpai *track* selanjutnya. *Chunk* MTrk ini berisi data dalam bentuk event, baik berupa event MIDI maupun event sistem. Setiap event memiliki parameter tipe, waktu, dan data dengan ukuran yang spesifik untuk setiap event. Berikut adalah beberapa tipe MIDI event tersebut :

Tabel 2.2 Beberapa tipe event MIDI yang umum terdapat pada berkas MIDI

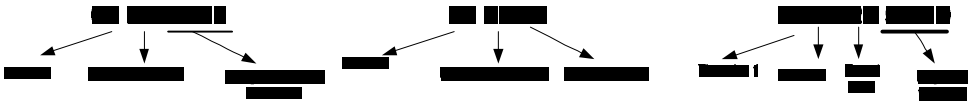
Tipe	**Nama Event**	**Data**	**Keterangan**
80	*Note Off*	2 Byte *(note, velocity)*	*Stop* memainkan suatu kunci not.
90	*Note On*	2 Byte *(note, velocity)*	*Start* memainkan suatu kunci not.
A0	*After Touch*	2 Byte *(note, pressure)*	Perubahan intensitas penekanan terhadap sebuah not
B0	*Control Change*	1 Byte (*control, value*)	Prubahan spesifik terhadap *setting* suatu *device*
C0	*Program Change*	1 Byte (*program number*)	Pemilihan program (*instrument* musik)
D0	*Channel Pressure*	1 Byte (*pressure*)	*After touch* untuk seluruh *channel.*
E0	*Pitch Wheel*	2 Byte (*combined to a 14-bit value*)	*Setting Pitch Wheel*
F0	*System Exlusive*	All Byte to next 0xF7	Pesan khusus suatu *device*

Selain MIDI event, terdapat juga suatu event sistem yang diawali dengan 1 byte tipe FF, 1 byte sub tipe dan panjang data non-MIDI. Contoh sebuah event sistem adalah sebagai berikut (Gambar 2.9):



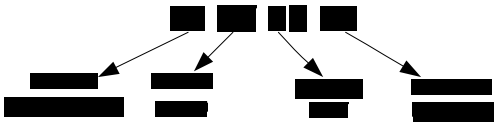
Gambar 2.9. Event Sistem

Adapun contoh untuk event control change, program change serta note on dijelaskan pada gambar 2.10



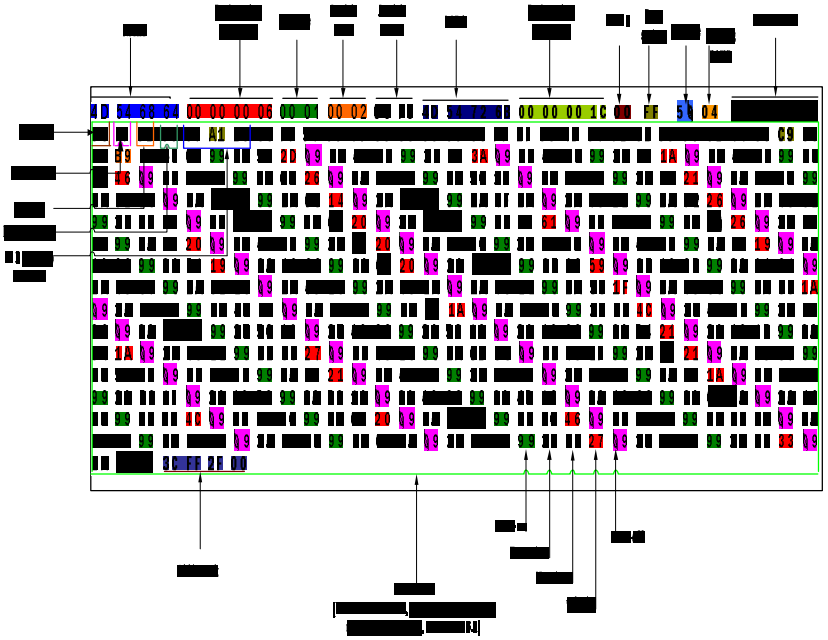
Gambar. 2.10 Format event control change, program change serta note on

Akhir dari suatu track ditandai dengan event End of Track (gambar 2.11).



Gambar. 2.11 Format event end of track

Dan untuk lebih jelasnya, secara lengkap sebuah chunk berkas MIDI dapat direpresentasikan sebagai berikut



Gambar 2.12. Nilai Chunk berkas MIDI

2.6 SNR (*Signal to-Noise Ratio*)

Salah satu kriteria penyembunyian data rahasia ke dalam media digital mengubah kualitas media adalah *Fidelity* dimana mutu media penampung tidak jauh berubah. Dimana setelah penambahan data rahasia pengamat tidak mengetahui kalau di dalam media tersebut terdapat data rahasia[3].

Dalam *audio* steganografi, pemecah kode menggunakan sistem pendengaran manusia sama baiknya dengan menggunakan analisis komputer dalam mendeteksi *device*. Sebagai perbandingan pendekatan, spektogram dari sinyal asli dan yang sudah terubah biasanya diberikan atau dihadirkan dalam bentuk data. Spektogram

adalah plot sederhana yang berisi frekuensi dari konten sinyal audio sebagai fungsi waktu. Dalam beberapa kasus tertentu, kualitas dari *file audio* yang berisi data tersembunyi bisa diperkirakan menggunakan *perceptual audio measure, noise to-mask ratio,* atau *Signal to-Noise Ratio (SNR)*[1].

Biasanya, *Signal to-Noise Ratio (SNR)* dari berkas audio original dibandingkan dengan audio steganografi dapat digunakan sebagai suatu ukuran mutu yang kuantitatif[3].

$$SNR = 10\log_{10}\left\{\frac{\sum_{n=0}^{n-1} x^2(n)}{\sum_{n=0}^{n-1}\left(\tilde{x}(n) - x(n)\right)^2}\right\}$$

Dimana, Xn = sinyal berkas *original,* X̃ = sinyal berkas steganografi dan n adalah jumlah *sample.*

Semakin tinggi *SNR*, maka semakin jelas juga ketersembunyian pesan yang ada[1]. Meski jumlah yang dapat ditolerir dari *noise* bergantung pada aplikasi steganografi dan karakteristik-karakteristik dari unstegano, akan tetapi distorsi *noise* yang jelas memiliki nilai *SNR* =35dB[3].

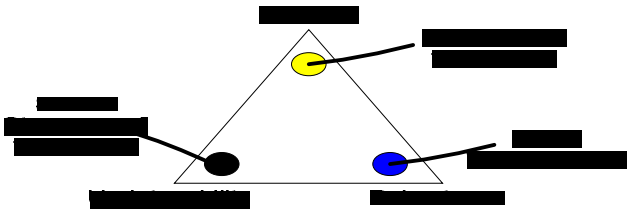
BAB III

ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Masalah

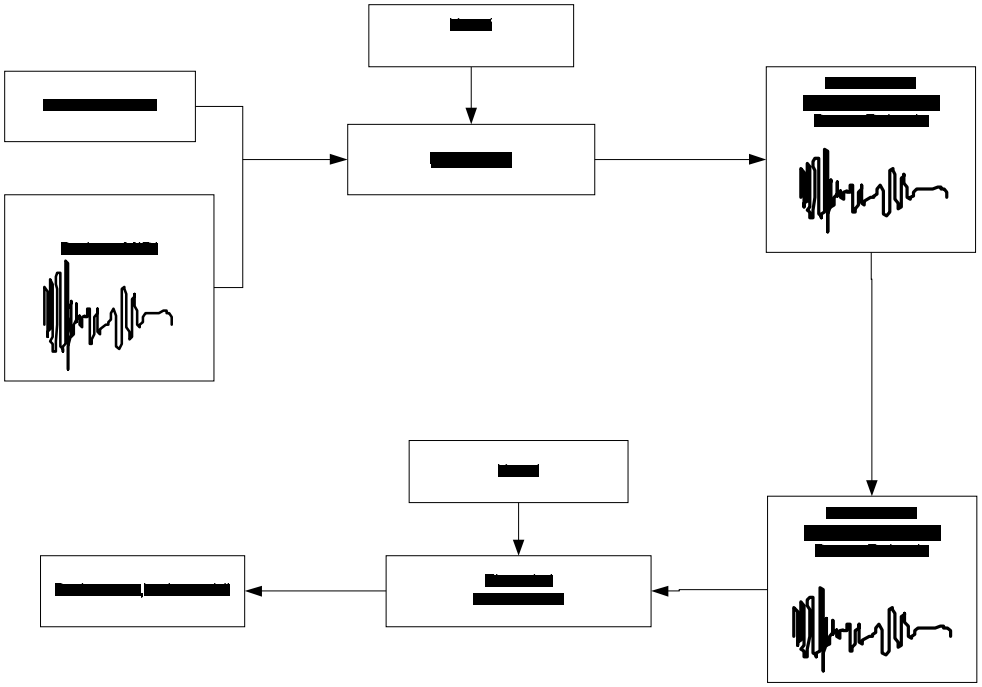
Steganografi merupakan sebuah seni atau teknik untuk menyembunyikan atau menyisipkan pesan / rahasia kedalam sebuah media , sehingga keberadaanya tidak terdektesi oleh pihak lain yang tidak berhak atas informasi tersebut. Setagografi dapat dipandang sebagai kelanjutan kriptografi. Dalam prakteknya pesan di enkripsi didalam media lain sehingga pihak ketiga tidak dapat menyadari keberadaan file. Steganografi membutuhkan 2 properties yaitu pesan dan media penampung. Media penampung pada umumnya dapat berupa teks, suara, gambar, atau video. Sedangkan pesan yang disembunyikan teks, gambar atau pesan lainnya. Keuntungan penggunaan steganografi adalah memungkinkan pengiriman pesan secara rahasia tanpa diketahui bahwa pesan sedang dikirim, sedangkan kelemahan stganografi adalah membutuhkan banyak ruang untuk meyembunyikan beberapa bit pesan.

Dalam menyisipkan informasi ada beberapa faktor yang saling berkompetisi satu sama lainnya. Gambar 3.1 menunjukan tiga faktor yang saling berkompetisi ini, capacity, undetecability dan robustness.



Gambar 3.1 Faktor yang berkompetisi steganografi

Sehubungan dengan hal diatas penulis hanya menekankan pada sistem yang akan dibangun berdasarkan permasalahan yang terjadi pada system, yaitu sistem aplikasi steganografi pada file midi dalam hal menyisipkan teks kedalam file midi dan mengekstrakssinya kembali dengan algoritma, maka akan dibangun sebuah aplikasi yaitu ***aplikasi steganografi pada berkas midi dengan menggunakan teknik phase coding*.** Adapun sistem yang dibangun seperti yang digambarkan pada Gambar 3.2.



Gambar 3.2. Sistem Steganografi pada MIDI

Sistem yang akan dibangun berdasarkan permasalahan yang terjadi pada sistem aplikasi steganografi pada berkas MIDI, dalam hal penyisipan text(*.txt) kedalam berkas MIDI(*.mid) dan ektraksinya dengan menggunakan algoritma *Phase Coding* dengan mengimplementasikanya. Kemudian dilakukan pengujian,

pada proses penyisipan akan dibandingkan ukuran berkas MIDI yang telah disisipkan pesan rahasia terhadap berkas MIDI *original*, kemudian dilakukan perbandingan jumlah perubahan bit(antara berkas MIDI yang telah disisipkan pesan rahasia dengan berkas MIDI *original*), melakukan pengujian terhadap ketahanan untuk tidak terdeteksi oleh *software* yang mendeteksi ada atau tidaknya pesan rahasia yang disembunyikan di dalam suatu berkas serta menganalisis pengaruh informasi (*.txt) yang disembunyikan terhadap kualitas suara yang dihasilkan oleh berkas MIDI (*.mid). Hal ini dilakukan untuk mengetahui keunggulan sekaligus kelemahan hasil *output* dari proses steganografi dengan teknik Phase coding. Diharapkan *output* dari proses steganografi ini memiliki kemampuan dan ketahanan yang maksimal.

3.1.1 Steganografi dengan phase coding

Steganografi dengan phase coding merupakan salah satu teknik steganografi pada audio. Phase coding termaksud dalam kelompok teknik audio steganografi dengan berbasis domain frekuensi yang bekerja dengan cara mengubah spectral content dalam domain frekuensi dari sinyal. Ide dasar dibalik teknik phase coding adalah menyenbunyikan data dengan cara menukarkan fase asli segmen initial dari sinyal suara dengan fase absolute dari sinyal watermark dengan tetap menjaga fase relative antara segmen sinyal menggunakan beda fase segmen dari sinyal asli. Ketika beda fase antara sinyal asli sdan sinyal yang dimodifikasi besarnya kecil, maka perbedaan suara yang dihasilkan tidak terdeteksi oleh pendengaran manusia. Terdapat beberapa parameter yang berpengaruh dalam teknik steganografi dengan

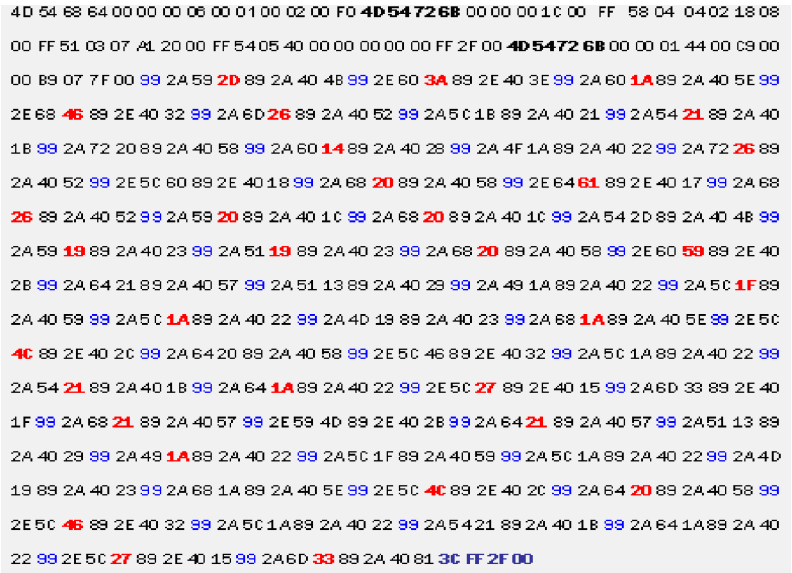
phase coding, parameter tersebut antara lain fase relative dan amplitudo sinyal suara, fase absolute data watermark, serta beda fase relative antara segmen sinyal suara.

3.1.2 Steganografi dengan phase coding terhadap media pemanpung (Midi)

Standard MIDI File (SMF) adalah format berkas yang digunakan untuk menampung data MIDI (*Musical Instrument Digital Interface*) ditambah beberapa jenis data yang biasa dibutuhkan oleh *sequencer* (pemutar berkas MIDI). Format ini menampung standard MIDI *message* ditambah dengan *time-stamp* untuk setiap *message*. Format ini juga membolehkan menyimpan informasi mengenai tempo, waktu dan kunci, *signature*, nama *track* dan *pattern*, dan informasi lainnya yang biasa dibutuhkan oleh sequencer. Satu SMF dapat menyimpan informasi untuk beberapa *pattern* dan *track*, sehingga setiap *sequencer* yang berbeda dapat mendukung struktur ini ketika membuka berkas tersebut.

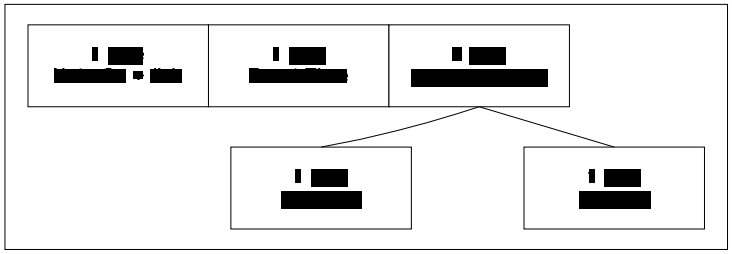
Pada umumnya prinsip kerja sistem aplikasi steganografi dengan phase coding pada berkas *audio* MIDI ini sama seperti halnya aplikasi Steganografi pada berkas *audio* lain yang telah ada. Secara garis besar, aplikasi steganografi pada berkas *audio* MIDI ini memiliki dua komponen utama, yaitu komponen penyisipan data ke dalam berkas MIDI dan komponen ekstraksi berkas MIDI yang telah disisipi data. Masukan untuk komponen penyisipan data ke dalam berkas audio MIDI ini adalah sebuah berkas *audio* digital dengan format SMF (*Standard MIDI File*), data yang akan disisipkan, dan sebuah kunci. Keluaran dari komponen ini adalah sebuah berkas audio digital dengan format MIDI yang telah

disisipi data tersebut. Komponen ekstrasi berkas *audio* digital yang telah disisipi data adalah melakukan proses ekstraksi kembali berkas audio digital yang telah disisipi data untuk mendapatkan berkas data yang valid. Masukan dari komponen ini adalah sebuah berkas *audio* digital dengan format MIDI dan sebuah kunci. Keluaran dari komponen ini adalah sebuah berkas *audio* digital dengan format MIDI dan berkas data yang disisipkan. Dibawah ini merupakan nilai chunk midi dimana 53 byte *event note on* (**99**) dan misalkan byte *velocity* yang diambil (merah) adalah 2D, 3A, 1A, 46, 26, 21, 14, 26, 20, 61, 14, 26, 20, 19, 19, 20, 59, 1F, 1A,1A, 4C, 21, 1A, 27, 21, 21, 1A, 4C, 20, 46, 27, 33.



Gambar 3.3 Nilai Chunk MIDI Original[heksadesimal]

Event MIDI yang dipilih sebagai penampungnya adalah event *note on*,karena jenis event ini diasumsikan sebagai event yang paling banyak muncul. Hierarki format dari sebuah event note on adalah seperti pada gambar 3.8



Gambar 3.4 Hierarki format event note on

Event *note on* terdiri atas 4 byte. Byte pertama terdiri atas kode event *note on* beserta nomor *channel* event tersebut. Byte selanjutnya berisi informasi waktu dari event tersebut. Dua byte selanjutnya berisi *message* dari event tersebut, yang terdiri atas satu byte kunci not serta satu byte *velocity* dari not tersebut.

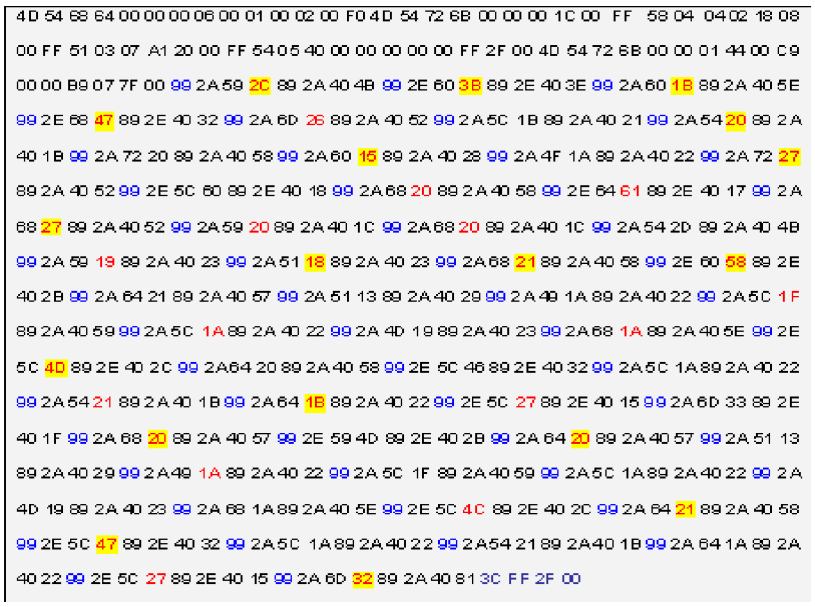
3.1.3 Steganografi dengan phase coding terhadap media penyisip (Txt)

Txt merupakan salah satu ektension nama file dari teks dengan format yang sangat kecil. Txt dalam steganografi dengan phase coding , dimana teks yang akan disisipi terlebih dahulu akan diubah kedalam bit-bit, karena Phase coding bekerja berdasarkan frekuensi atau sinyal suara, maka pesan yang telah diubah menjadi bit-bit akan diubah lagi kedalam domain frekuensi. Pesan txt yang diubah dengan pseudonoise signal akan berubah menjadi bentuk plaintext, sebelum dilakukan FFT.

3.1.4 Proses Steganografi dengan phase coding pada berkas midi

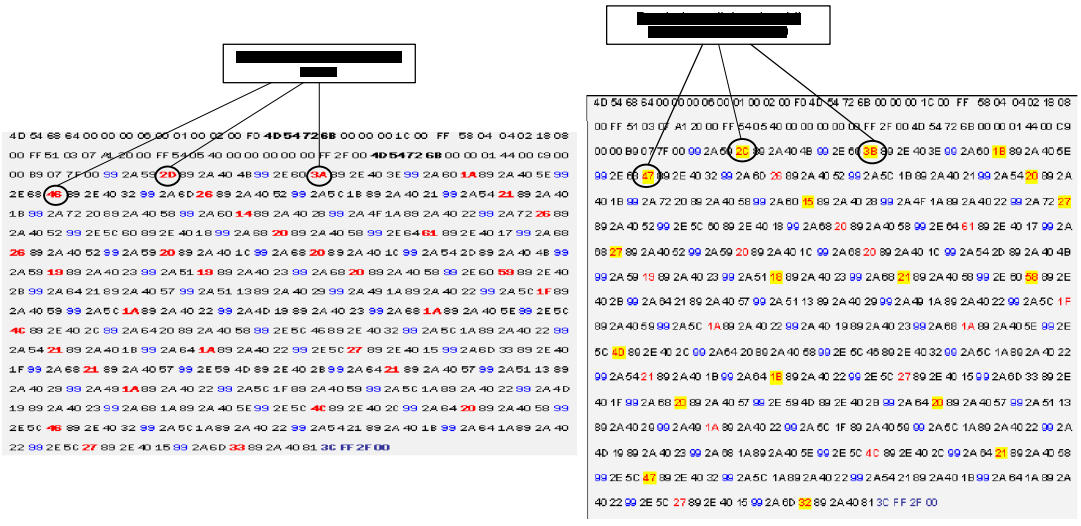
Steganografi dengan phase coding dalam proses penyisipan txt kedalam midi pada prinsipnya tidak jauh berbeda dengan prinsip kerja steganografi audio lainnya. Siapkan berkas midi , data suara berupa informasi mengenai midi dan sampel suara midi akan disimpan di memori komputer, kemudian baca berkas teks

siapkan bit-bit data txt, panjang txt dan data txt disimpan dalam memori komputer. Masukan kunci sebagai kunci proses. Kunci yang dimasukan akan diubah dalam bit-bit dengan metode pembangkit pseudonoise. Hasil dari perubahan kunci akan dijadikan sebagai pseudonoise signal. Proses pertama terhadap pesan suara adalah dengan proses modulasi, yaitu mengacak dengan pseudonise signal yang dibangkitkan menggunakan algoritma LCG, pembangkit pseudonoise signal ini menggunakan bilangan pseudonumber dari algoritma dengan pembangkit yang diambil dari variabel input kunci. Hasil modulasi yang akan disisipi oleh phase coding akan digabung dalam sinyal suara dan dibagi menjadi beberapa segmen , karena prinsip kerja dari phase coding adalah menggunakan frekuensi atau sinyal suara. Setiap segmen sinyal suara akan diubah kedalam domain frekuensi dengan mengunakan Fast Fourier Transform (FFT), dan selanjutnya adalah menghitung amplitudo dan nilai fase setiap segmenya. Beda fase yang terjadi akan dihitung kembali kemudian nilai fase segmen awal diganti dengan nilai fase data watermark. Kemudian lakukan Fast Fourier Transform terhadap seluruh segmen, maka akan didapatkan sinyal suara yang sudah disisipi pesan. Hal tersebut dapat dilihat pada gambar dibawah ini :



Gambar 3.5. Nilai Chunk MIDI Stego(tersisip pesan)

Perbandingan nilai pada chunk midi antara midi yang belum tersisipi pesan dengan midi yang sudah tersisipi pesan, dapat dilihat pada gambar dibawah ini :



Gambar 3.6. Perbandingan nilai chunk midi

3.1.5 Proses Ekstraksi dengan Phase coding

Ekstraksi dibutuhkan untuk mengembalikan data yang telah disisipi kembali ke data asli. Ekstraksi menggunakan phase coding hampir sama dengan proses penyisipannya dimana dilakukan dengan cara melakukan Fast Fourie Transfrom kepada element pertaman sinyal suara karena data watermark hanya disisipkan disegmen pertama sinyal suara, maka akan didapatkan elemen real dan imaginernya. Nilai fasenya akan dikonversi Π/2 merepresentasikan bit 1 dan - Π/2 merepresentasikan bit 0. Hasil ekstraskinya akan menunjukan kesesuaian dengan bit watermark asal.

3.2 Deskripsi Metode Pembangkit Pseudonoise

Noise dibutuhkan untuk mengacak pesan rahasia yang akan disembunyikan dalam berkas media. Noise yang sama dibutukan untuk pesan rahasia semula yang telah disisipi dalam media. Untuk itu dibutuhkan pembangkit yang ditentukan berdasarkan berdasarkan kunci masukan sang pengguna. Noise yang dihasilkan tidak betul-betul bilangan acak, karena ada nilai awal pembangkitnya. Oleh karena itu noise yang dihasilkan dari proses pembangkit itu disebut pseudonoise. Adapun algoritma yang digunakan dalam pembangkit pseudonoise ini adalah algoritma LCG.

Bibit awal pembangkit ditentukan berdasarkan perhitungan dari kunci masukan dari pengguna. Kunci yang dimasukan pengguna akan ditransformasikan dengan fungsi tertentu kedalam format data kunciyang berukuran tetap.

Contoh, bila masukan pengguna = a (65h = 01100101b) adalah sebagai berikut : ’ 0 1 1 0 0 1 0 1 ’

Bila ukuran bibit pembangkit yang ditetapkan adalah 32 bit , maka panjang ditransformasikanmenjadi :

’ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 1 ’

Hasil dari transformasi kunci masukan pengguna kemudian dijadikan sebagai bibit pembangkit pseudonoise (Z0) dan algoritma LCG (dapat dilihat pada persamaan dibaawah ini. Contoh apabila parameternya adalah sebagai berikut :

Z0 = ‘0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 1’

a = ‘0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 0 0 0 1 1 0 0 0 0’

c = ’ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 1 1 1 1 1’

m = ’ 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1’

Maka *Z1* yang dihasilkan adalah :

Z1 = ’ 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 1 1 0 0 1 0 0 1 1 1 1’

Dengan menggunakan parameter *a*, *c*, dan *m* yang sama maka diperoleh pseudonoise selanjutnya, yaitu :

Z2 = ’ 0 1 0 0 0 1 1 0 1 1 1 1 0 1 0 1 1 0 0 1 0 1 1 1 0 0 0 0 1 0 0 1’

Z3 = ’ 0 1 1 1 1 0 1 1 1 0 0 0 0 0 1 0 0 0 1 1 1 0 1 0 0 0 0 0 0 1 0 1’

Z4 = . . .

Pseudonoise signal yang dihasilkan adalah penggabungan dari *pseudonoise* yang dihasilkan (*Z1, Z2 ,Z3* ,, ... dst) , yaitu sebagai berikut :’ 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 0 1 1 0 1 1 1 1 0 1 0 1 1 0 0 1 0 1 1 1 0 0 0 0 1 0 0 1 0 1 1 1 1 0 1 1 1 0 0 0 0 0 1 0 0 0 1 1 1 0 1 0 0 0 0 0 0 1 0 1 ...’

Panjang *pseudonoise* bersifat dinamis disesuaikan dengan panjang pesan yang akan dimodulasi. Artinya jika seluruh bit Z1 telah digunakan untuk modulasi pesan dan masih ada sisa panjang pesan yang belum dimodulasi, maka akan dibangkitkan Z2 dan seterusnya.

3.3 Algoritma *Phase coding*

Phase coding adalah salah satu metode dalam audio *watermarking* yang bekerja berdasarkan karakteristik sistem pendengaran manusia *(Human Auditory System)*, yang mengabaikan suara yang lebih lemah jika dua suara itu datang bersamaan. Data *watermark* dibuat menjadi derau dengan amplitudo yang lebih lemah dibandingkan amplitudo data audio, kemudian keduanya digabungkan.

Ide dasar dibalik teknik *phase coding* adalah menyembunyikan data dengan cara menukarkan fase asli segmen inisial dari sinyal suara dengan fase absolut dari sinyal *watermark* dengan tetap menjaga fase relatif antara segmen sinyal menggunakan beda fase segmen dari sinyal asli. Ketika beda fase antara sinyal asli dan sinyal yang dimodifikasi besarnya kecil, maka perbedaan suara yang dihasilkan tidak terdeteksi oleh pendengaran manusia.

Phase coding termasuk dalam kelompok teknik audio *watermarking* berbasis domain frekuensi yang bekerja dengan cara mengubah *spectral content* dalam *domain* frekuensi dari sinyal. *Phase coding* merupakan metode paling efektif dari segi perbandingan *noise signal-to-perceived*.

Dalam pembahasan berikutnya akan dijelaskan mengenai parameter apa saja yang mempengaruhi *phase coding*, bagaimana cara menyisipkan *watermark* ke

dalam data audio (enkoding), dan cara pengekstraksian *watermark* dari data audio yang telah disisipi (dekoding) dengan teknik *phase coding*.

3.3.1 Parameter

Terdapat beberapa parameter yang berpengaruh dalam teknik steganografi *phase coding*. Parameter tersebut antara lain fase relatif dan amplitudo sinyal suara, fase absolut data *watermark*, serta beda fase relatif antara segmen sinyal suara.

Phase coding didasarkan pada kenyataan bahwa pendengaran manusia lebih peka terhadap beda fase relatif dalam sinyal audio dibandingkan dengan fase absolut yang ditambahkan. Jika hubungan fase antar setiap komponen frekuensi diubah secara dramatis, akan terjadi dispersi fase yang tampak dengan jelas. Akan tetapi, selama modifikasi fase cukup kecil (tergantung pada pengamat) dan dengan menjaga beda fase relatif antara segmen-segmen sinyal suara, maka modifikasi fase yan dilakukan tidak akan terdengar.

Amplitudo sinyal suara digunakan bersama dengan nilai fase sinyal suara untuk mengubah kembali sinyal suara dari domain frekuensi menjadi domain waktu sehingga dapat didengarkan.

3.3.2 Enkoding

Langkah-langkah untuk melakukan penyisipan dengan teknik *phase coding* adalah sebagai berikut :

1. Bagi urutan suara menjadi N segmen, $s[\ i\ ]$, $0 \leq i \leq L\text{-}1$, dimana setiap segmen memiliki panjang yang sama yaitu sebesar L. (Gambar 3.7)



Gambar 3.7. Sinyal asli dan sinyal yang dibagi menjadi beberapa segmen

2. Hitung DFT (dalam hal ini menggunakan FFT) pada masing-masing segmen. Hasil dari perhitungan ini adalah berupa *X(k)* untuk masing-masing segmen dimana $0 \leq k \leq L\text{-}1$.

3. Hitung nilai fase ϕ dan amplitudo *A* untuk tiap-tiap segmen dimana a_k adalah bagian *real* dari nilai FFT dan b_k adalah bagian *imaginer*-nya. Hasil yang diperoleh untuk tiap segmennya akan tampak seperti pada Gambar II-15.

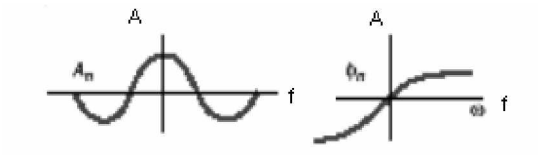
$$|A| = \sqrt{a_k^{\,2} + b_k^{\,2}} \tag{2.6}$$

$$\phi = \tan^{-1}\left(\frac{b_k}{a_k}\right) \tag{2.7}$$

4. Setelah itu hitung beda fase antara fase segmen yang berdekatan.

$$\Delta\phi_{N-1}(\varpi_k) = \phi_{N-1}(\varpi_k) - \phi_N(\varpi_k)$$

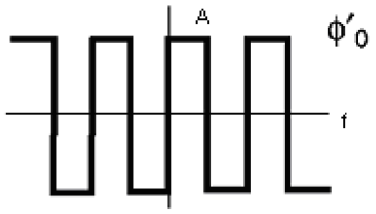
5. Fase absolut dari sinyal data *watermark* ditambahkan ke dalam beda fase yang dihasilkan. Sinyal *watermark* dengan panjang *Lw*, *w*[*j*], $0 \leq j \leq Lw\text{-}1$, disajikan sebagai $\Phi_{data} = \pi/2$ atau $\Phi_{data} = -\pi/2$ yang merepresentasikan bit 1 atau 0. Hasil yang diperoleh akan tampak seperti Gambar 3.8.



Gambar 3.8. Magnitudo dan plot fase dari FFT

6. Subsitusikan fase segmen awal Φ'_0 dengan fase sinyal *watermark* Φ'_{data} (Gambar 3.9)

$\Phi'_0 = \Phi'_{data}$



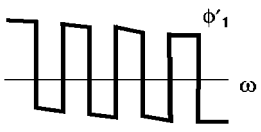
Gambar 3.9 Fase absolut dari sinyal data watermark

7. Buat matriks fase untuk N>0 dengan menggunakan beda fase untuk menjaga relativitas fase antara segmen suara. Hal ini dilakukan untuk menjaga kesinambungan sinyal setelah proses modifikasi fase segmen awal.

$$\begin{bmatrix} (\phi'_1(\varpi_k) = \phi'_0(\varpi_k) + \Delta\phi_1(\varpi_k)) \\ \cdots \\ (\phi'_n(\varpi_k) = \phi'_{n-1}(\varpi_k) + \Delta\phi_n(\varpi_k)) \\ \cdots \\ (\phi'_N(\varpi_k) = \phi'_{N-1}(\varpi_k) + \Delta\phi_N(\varpi_k)) \end{bmatrix}$$

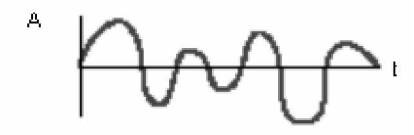
Gambar 3.10. matriks fase

8. Gabungkan segmen-segmen yang telah dimodifikasi fasenya tersebut menjadi satu.



Gambar 3.11 Menambahkan fase absolut watermark ke beda fase sinyal asli

9. Hitung kembali nilai fase yang baru dan nilai amplitudo yang sudah dihitung sebelumnya untuk melakukan *invers*e DFT terhadap masing-masing segmen untuk mengembalikan sinyal ke domain waktu.



Gambar 3.12 Sinyal watermarking yang dihasilkan dengan teknik phase coding

3.3.3 Dekoding

Dekoding dilakukan dengan melakukan sinkronisasi terhadap proses enkoding. Panjang segmen dan panjang *watermark* harus diketahui. Proses dekoding memerlukan sinyal suara asli untuk melakukan pendeteksian. Langkah-langkah pendeteksian *watermark* yang dilakukan adalah sebagai berikut :

1. Ambil *n* bagian pertama dari sinyal suara dimana *n* adalah panjang segmen enkoding yang diketahui, *s*[*0*] ..*s*[*n-1*]. Perhitungan hanya dilakukan terhadap *n* elemen pertama sinyal karena data *watermark* disisipkan hanya di segmen awal sinyal suara asli.
2. Lakukan FFT terhadap *n*-1 sinyal tersebut, kemudian cari nilai fasenya Φ sesuai persamaan 2.7.
3. Konversi nilai fase yang didapatkan, $\pi/2$ menjadi bit 1 dan $-\pi/2$ menjadi bit 0 sebanyak panjang bit *watermark*. Nilai fase dibandingkan dengan nilai fase sinyal suara asli.

4. Didapatkan data bit-bit *watermark* sesuai dengan hasil konversi. Bit-bit *watermark* dibandingkan dengan bit-bit *watermark* asli untuk mengetahui kebenarannya.

3.4 Deskripsi proses steganografi dengan phase coding

1. Baca berkas suara midi.
2. Siapkan data suara. Data suara berupa informasi mengenai suara midi dan sampel suara midi tersebut disimpan di dalam memori komputer.
3. Baca berkas teks.
4. Siapkan bit-bit data teks. Panjang teks dan data teks disimpan di dalam memori computer.
5. Masukan kunci sebagai kunci proses, kunci yang dimasukan akan diubah menjadi bit – bit denga metode pembangkit pdseudonoise. Sebagai contoh : apabila kunci masukan pengguna = A (61_h =01100101). Apabila ukuran bibit pembangkit yang ditetapkan adalah 32 bit, maka panjang kunciyang ditransformasikan menjadi :‘00000000000000000000000001100101’

Hasil dari transformasi kunci masukan pengguna kemudian dijadikan sebagai bibit pembangkit pseudonoise (Z0) dalam algoritma LCG.

Z0 = ‘00000000000000000000000001100101’

a = ‘00000000000000000101111000110000’

c = ‘00000000000000000110011000011111’

m =‘01111111111111111111111111111111’

maka Z1 yang dihasilkan adalah :

Z1= ‘ 00000000001001000001011001001111’

6. Hasil perubahan kunci akan dijadikan sebagai pseudonoise signal..Proses pertama terhadap pesan rahasia adalah dengan proses modulasi, yaitu mengacak dengan suatu psedonoise signal yang dibangkitkan menggunakan algoritma LCG,pembangkit pseudonoise signal ini menggunakan bilangan pseudonumber dari algoritma LCG denga bibit pembangkit yang diambil dari variable input kunci. Hasil pembangkitan , diperoleh pseudonoise signal sebagai berikut : '00000000001001000001011001001111' , selanjutnya segmen pesan dimodulasikan dengan pseudo-noise signal menggunakan fungsi XOR (exclusive OR) .

   Contoh : pesan ('saya' =73617961_h) :

   ' 01110011010000010111100101000001'

   Pseudo-noise : ' 00000000001001000001011001001111 '

   ' 01110011011001010110111100001110'

7. Hasil dari modulasi inilah kemudian akan disisipi kedalam sinyal suara melalui proses encoding , dan proses ekstraksi melalui proses decoding.

3.4.1 Contoh Proses Enkoding dan Dekoding

Diketahui Sinyal suara s = {1, 2, 3, 4, 5, 6, 7, 8}.

Bit-bit yang akan disisipkan = "1010"

Akan dilakukan penyisipan ke sinyal suara s. Langkah-langkah enkodingnya :

1. Sinyal suara s dibagi menjadi N segmen yang sama panjang (dalam hal ini ditentukan bahwa N = 2).

   s_0 = {1, 2, 3, 4} ; s_1 = {5, 6, 7, 8}

2. Setiap segmen diubah ke domain frekuensi dengan FFT, menghasilkan dua elemen *point* yaitu bagian *real* :

   sr_0 = {36, -4, -4, -9.657} ; sr_1 = {-4, -4, -4, -1.657}

   dan bagian *imaginer* :

   si_0 = {1, 2, 3, 4} ; si_1 = {5, 4, 3, 2}

3. Hitung amplitudo *A* (sesuai persamaan 2.6) dan nilai fase ϕ (sesuai persamaan 2.7) untuk tiap segmen

   A_0 = {36.01, 4.47, 5, 10.45} ; A_1 = {6.4, 5.65, 5, 2.6}

   ϕ_0 = {0.02, -0.46, -0.64, -0.39} ; ϕ_1 = {-0.89, -0.78, -0.64, -0.87}

4. Hitung beda fase antara segmen-segmen yang berdekatan sesuai dengan persamaan 2.8.

   $\Delta\phi_0$ = {0.91, 0.32, 0, 0.48}

5. Hitung fase dari sinyal data *watermark* dimana $\pi/2$ merepresentasikan bit 1 dan $-\pi/2$ merepresentasikan bit 0. Kemudian nilai fase segmen awal diganti dengan nilai fase data *watermark*. ($\pi/2$ = 1.57 radian, $-\pi/2$ = -1.57 radian)

   ϕ_0' = ϕ_{data} = {1.57, -1.57, 1.57, -1.57}

6. Untuk segmen selain segmen pertama, hitung kembali nilai fase tiap segmen sesuai dengan persamaan 2.10.

   ϕ_1' = {2.48, -1.25, 1.57, -1.09}

7. Lakukan invers FFT terhadap seluruh segmen (hasil dari langkah 5 dan 6) dengan menggunakan nilai fase ϕ' dan *A* yang telah dihitung sebelumnya. Maka didapatkan sinyal suara yang sudah disisipi *watermark*.

   Sinyal *y* = {1, 2, 3, 4, 5, 6, 7, 8}

Dekoding dilakukan dengan cara melakukan FFT ke *N* elemen pertama dari sinyal suara *y* dimana *N* adalah panjang segmen = 4 karena data *watermark* hanya disisipkan di segmen pertama sinyal suara. Didapatkan elemen *real* dan *imaginer*-nya.

sr = {36, -4, 4, -9.657}

si = {~, ~, ~, ~}

Hitung nilai fase dari segmen tersebut.

ϕ = {1.57, -1.57, 1.57, -1.57}

Konversi nilai fase, $\pi/2$ merepresentasikan bit 1 dan $-\pi/2$ merepresentasikan bit 0. Hasil ekstraksi akan menunujukkan bit '1010' dimana hal ini menujukkan kesesuaian dengan bit *watermark* asal.

3.5 Analisis Kebutuhan Non-Fungsional

Analisis non fungsional bertujuan untuk menghasilkan spesifikasi kebutuhan non fungsional. Spesifikasi kebutuhan non fungsional adalah spesifikasi yang rinci mengenai hal-hal yang akan dilakukan sistem ketika diimplementasikan. Analisis kebutuhan ini diperlukan untuk menentukan keluaran yang akan dihasilkan sistem, masukan yang diperlukan sistem, lingkup proses yang digunakan untuk megolah masukan menjadi keluaran, *volume* data yang akan ditangani sistem, jumlah pemakai dan kategori pemakai, serta kontrol terhadap sistem.

3.5.1 Analisis Kebutuhan User

Analisis kebutuhan *user* dimaksudkan untuk mengetahui siapa saja *user* yang dapat menggunakan aplikasi ini terlibat.

Adapun *user* yang dapat menggunakan aplikasi ini adalah *user* yang memiliki kemampuan untuk mengoperasikan komputer kelas PC dan familiar dengan sistem operasi berbasis Windows.

3.5.2 Analisis Kebutuhan Software

1. Sistem operasi untuk menjalankan aplikasi ini adalah Sistem operasi *Microsoft Windows XP*.
2. Perangkat lunak yang digunakan untuk membangun aplikasi ini adalah C++ Builder.

3.5.3 Analisis Kebutuhan Hardware

Kebutuhan sistem yang ada sekarang untuk menjalankan perangkat lunak ini konfiguransinya adalah sebagai beriut :

a. *Processor* Intel *Pentium IV*
b. *Memory* 512 MB
c. *Videocard* 256 MB
d. Ruang *harddisk* 35 GB
e. Mouse dan keyboard

Sedangkan kebutuhan sistem untuk menjalankan perangkat lunak ini tidak terlalu tinggi, tetapi untuk kerja yang optimum dibutuhkan konfigurasi minimum sebagai berikut :

a. *Processor* minimal *Pentium IV*
b. *Memory* minimal 256 MB
c. *Videocard* minimal 128 MB
d. Ruang *harddisk* minimal 10 GB
e. Mouse dan keyboard

Untuk kerja optimal dari aplikasi tersebut, sebaiknya konfigurasi komputer yang dibutuhkan atau yang akan digunakan sama dengan kebutuhan system yang ada sekarang

3.6 Perancangan Sistem

Perancangan sistem adalah suatu proses yang menggambarkan bagaimana suatu sistem dibangun untuk memenuhi kebutuhan pada tahap analisis. Adapun tahapan yang dilakukan dalam perancangan sistem ini membahas mengenai tujuan perancangan sistem, perancangan funsional dan perancangan antarmuka.

3.6.1 Tujuan Perancangan Sistem

Perancangan sistem merupakan tindak lanjut dari tahap analisa. Perancangan sistem bertujuan untuk memberikan gambaran sistem yang akan dibuat. Dengan kata lain perancanagn sistem didefinisikan sebagai penggambaran atau pembuatan sketsa dari beberapa elemen yang terpisah kedalam satu kesatuan yang utuh dan berfungsi. Selain itu juga perancangan bertujuan untuk lebih mengarahkan sistem

yang terinci, yaitu pembuatan perancangan yang jelas dan lengkap yang nantinya akan digunakan untuk pembuatan simulasi. Aplikasi steganografi ini dibuat dengan sederhana, sehingga diharapkan *user* dapat dengan mudah menggunakan aplikasi steganografi ini.

3.7 Perancangan fungsional

Dalam perancangan ini akan menggunakan diagram konteks , *Data Flow dan Diagram(DFD)* dan *flowchart.*

3.7.1 Diagram konteks

Diagram konteks digunakan untuk menggambarkan hubungan antara entitas-entitas yang terdapat diluar sistem dan masukan serta keluaran dari sistem, diagram konteks pada Gambar 3.16 menggambarkan bagaimana proses yang terjadi pada aplikasi steganografi.



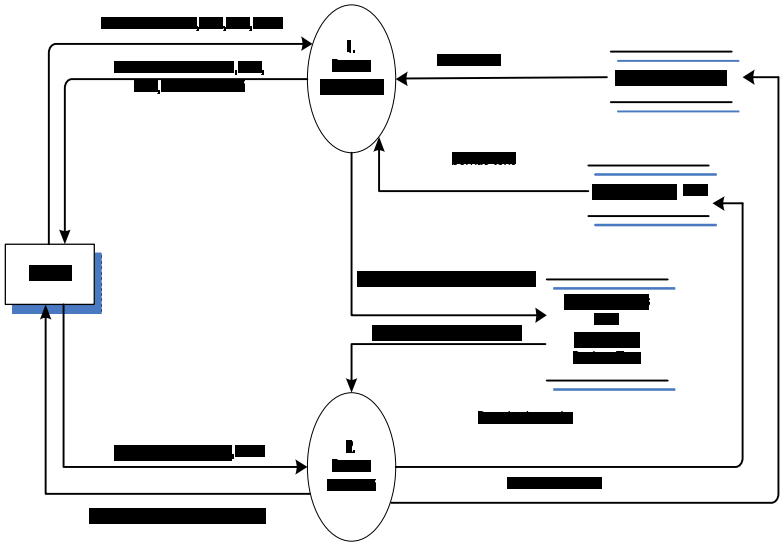
Gambar 3.13. Diagram Konteks aplikasi Steganografi

3.7.2 Data Flow Diagram (DFD)

Data Flow Diagram (DFD) adalah sebuah teknik grafis yang menggambarkan aliran informasi dan transformasi yang diaplikasikan pada saat data bergerak dari input menjadi output. DFD dapat digunakan untuk menyajikan sebuah sistem atau perangkat lunak pada setiap tingkat abstraksi. DFD dapat dipartisis ke dalam tingkat-tingkat yang merepresentasikan aliran informasi yang bertambah dan fungsi ideal. DFD memberikan suatu mekanisme bagi pemodelan fungsional dan pemodelan aliran informasi.

3.7.3 *Data Flow Diagram (DFD) Level 0*

DFD Level 0 Aplikasi Steganografi menggambarkan keseluruhan proses Aplikasi Steganografi. Adapun DFD Level 0 yang digambarkan dalam perancangan sistem adalah sebagai berikut :

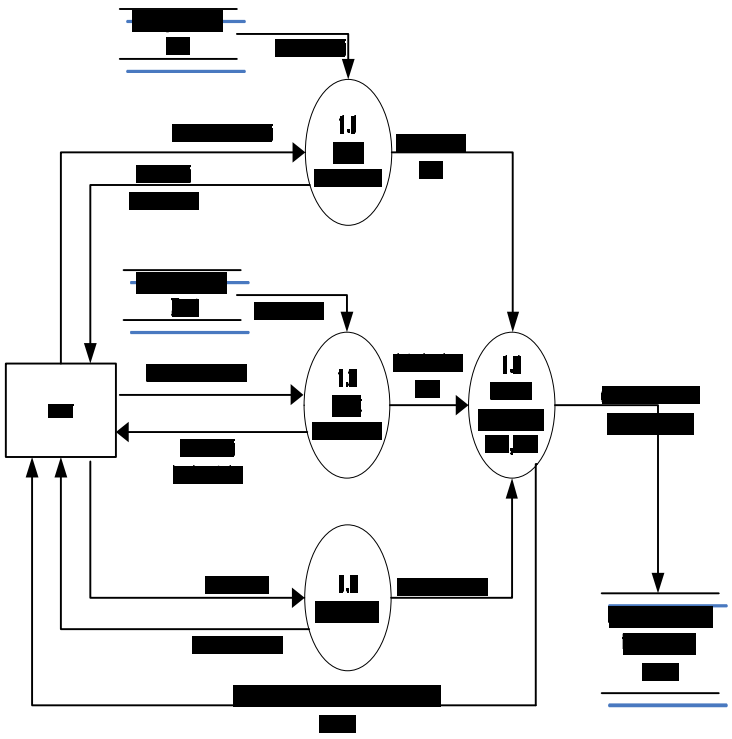


Gambar 3.14 DFD Level 0 Aplikasi Steganografi

Pada proses yang terjadi pada Gambar 3.17, data berkas MIDI, data berkas Text dan data Kunci dimasukkan oleh *User*, kemudian terjadi proses penyisipan yang menghasilkan sebuah berkas *audio digital* dengan format MIDI yang telah disisipi berkas Text (dalam hal ini sebuah pesan rahasia berformat *.txt).

3.7.3.1 *Data Flow Diagram (DFD)* Level 1 proses penyisipan

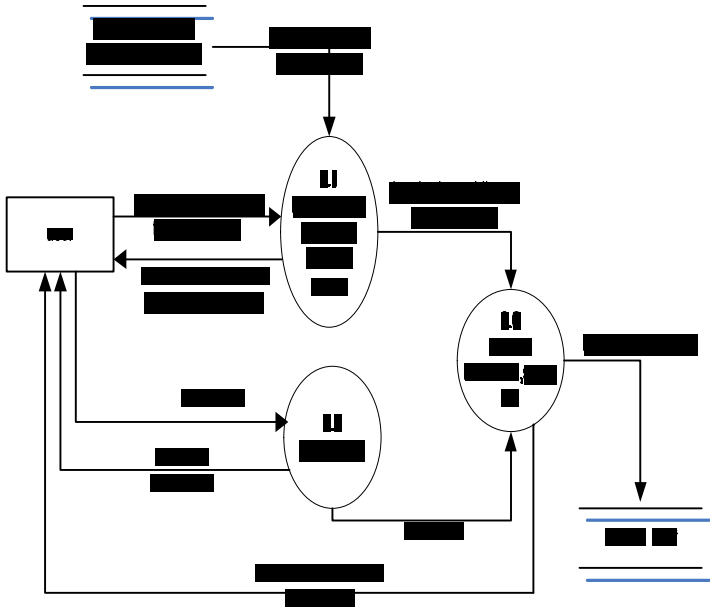
DFD Level 1 proses Aplikasi Steganografi menggambarkan proses steganografi atau penyisipan yang terjadi. Adapun DFD Level 1 proses yang digambarkan dalam perancangan sistem adalah sebagai berikut :



Gambar 3.15 DFD Level 1 Proses penyisipan Aplikasi Steganografi

3.7.3.2 *Data Flow Diagram (DFD)* Level 1 proses ekstraksi

DFD Level 1 proses Aplikasi Steganografi menggambarkan proses ekstraksi yang terjadi. Adapun DFD Level 1 proses yang digambarkan dalam perancangan sistem adalah sebagai berikut :

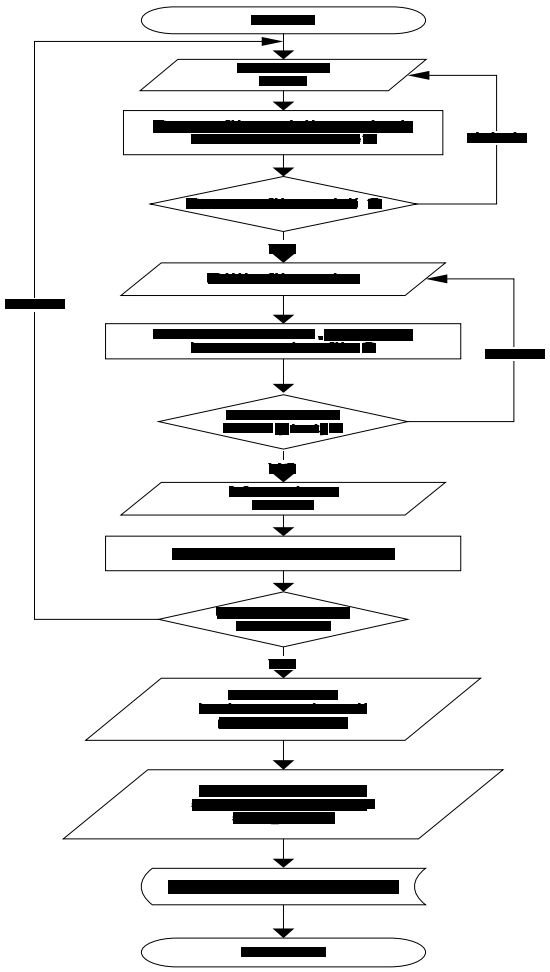


Gambar 3.16. DFD Level Proses ekstraksi Aplikasi Steganografi

3.8 Perancangan Prosedural

Dalam perancangan ini akan menggunakan *flowchart.*

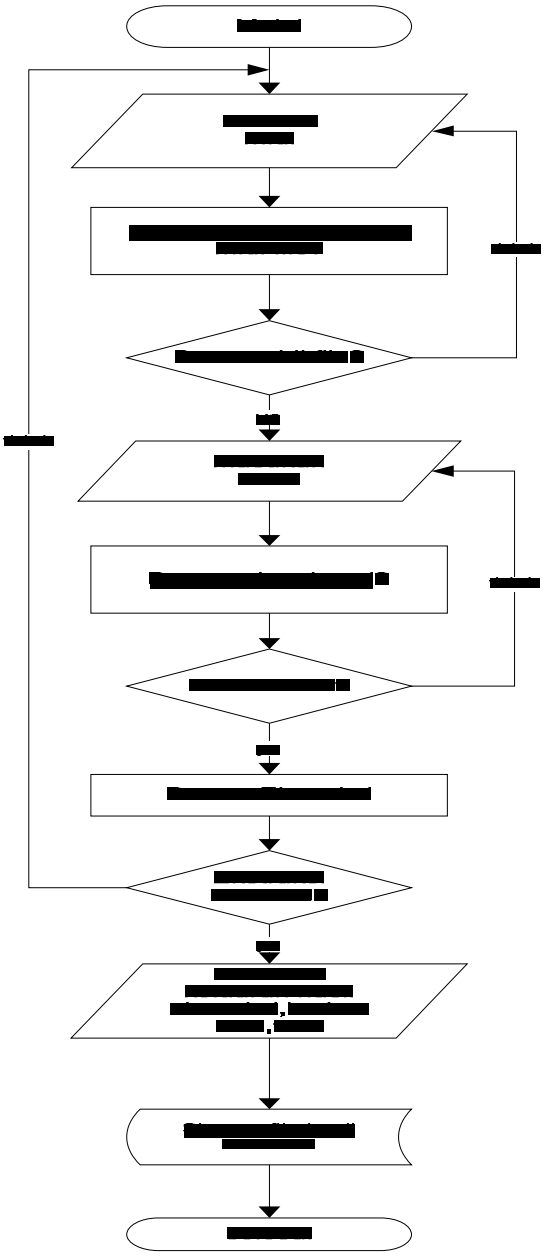
3.8.1 *Flowchart* Proses Penyisipan



Gambar 3.17 Flowchart Proses Penyisipan

Pada proses yang terjadi pada Gambar 3.20, *user* memasukkan/*input* data berkas MIDI, kemudian memasukkan/*input* data berkas Text dan memasukkan/ *input* data Kunci. Setelah ketiga masukkan/*input*an lengkap maka proses selanjutnya adalah melakukan proses encoding dimana urutan suara dibagi menjadi beberapa segmen dimana setiap segemne memiliki panjang yang sama,kemudian hitung dft nya dengan menggunakan rumus fft,hitung nilai fasenya setiap segemn,hitung beda fase antara fase segmen yang berdekatan. Fase absolute dari sinyal data ditambah kedalam beda fase yang dihasilkan.proses encoding dilakukan dengan melakukan proses sinkronisasi tehadap proses encoding. Setelah disisipkan/ditamankan selanjutnya berkas MIDI yang telah tersisipkan Berkas Text disimpan pada *Storage* Berkas MIDI yang telah tersisipkan Berkas Text.

3.8.2 *Flowchart* Proses Ekstraksi

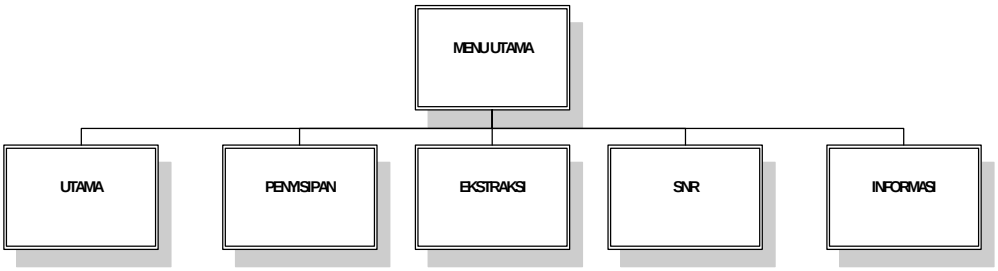


Gambar 3.18 Flowchart proses ekstraksi

Pada proses yang terjadi pada Gambar 3.21, *user* memassukkan/*input* data berkas MIDI yang telah disisipkan Text, kemudian *user* memasukkan/*input* data Kunci sesuai dengan data Kunci pada proses penyisipan. Setelah semua *input*an lengkap proses selanjutnya adalah Bangkit Bilangan *Pseudo-random* dari proses ini menghasilkan proses Hasilkan Lokasi *psudo-random* yang mana menghasilkan informasi lokasi bit-bit yang ditanamkan/disisipkan setelah itu dilakukan proses Saring bit-bit *enkoding* pada Lokasi *Pseudo-random* yang menghasilkan data teks terenkripsi kemudian dilakukan proses dekoding menghasilkan berkas text terdeskripsi selanjutnya disimpan pada *storage* berkas Text.

3.9 Perancangan Struktur menu

Struktur menu merupakan penggambaran fungsi-fungsi atau link yang ada pada sebuah program.Perancangan struktur menu berfungsi untuk mengetahui alur – alur rancangan program. Dalam pembuatan program aplikasi ini menggunakan alat bantu berupa bagan struktur. Perancangan struktur menu adalah sebagai berikut :

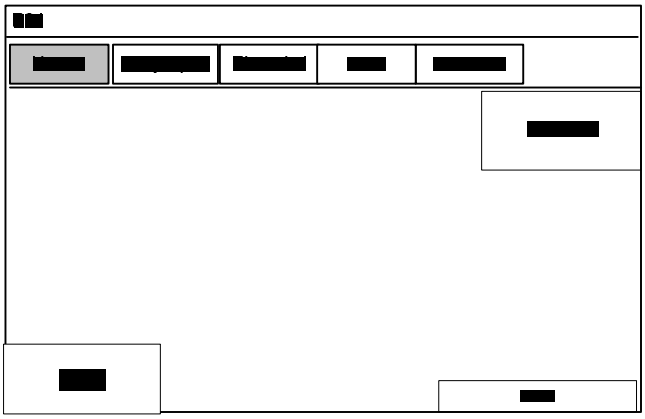


Gambar 3.19 Struktur menu aplikasi steganografi

3.9.1Perancangan Antar Muka Perangkat lunak

Setelah melakukan perancangan sistem, maka tahap selanjutnya adalah perancangan antarmuka perangkat lunak. Perancangan antarmuka perangkat lunak (*user Interface*) dibuat untuk memperkuat interaksi antara sistem dan pengguna. Antarmuka dari perangkat lunak dirancang untuk memberikan kemudahan bagi penggunanya dalam memanfaatkan fitur-fitur yang ada. Rancangan antarmuka dapat digambarkan sebagai berikut:

3.9.1.1 Layar Saji Menu Utama



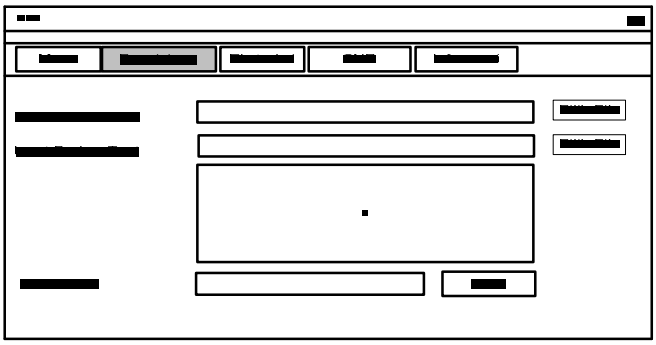
Gambar 3.20 Rancangan antar muka menu utama

Tabel 3.1. Keterangan tampilan menu utama

No Objek	Jenis Objek	Fungsi Objek
1	Main Menu	Untuk menampilkan menu utama
2	Penyisipan	Untuk menunjukan menu penyisipan
3	ekstraksi	Untuk menunjukan menu ekstraksi
4	SNR	Untuk menu SNR
5	Information	Untuk menunjukan menu informasi
6	Nama penulis`	Menampilkan identitas penulis

7	Nama Aplikasi	Menampilkan nama aplikasi
8	Logo	Menampilkan logo dan namauniversitas
9	Email	Alamat email penulis

3.9.1.2 Layar Saji Proses Penyisipan



Gambar 3.21 Rancangan antar muka menu penyisipan

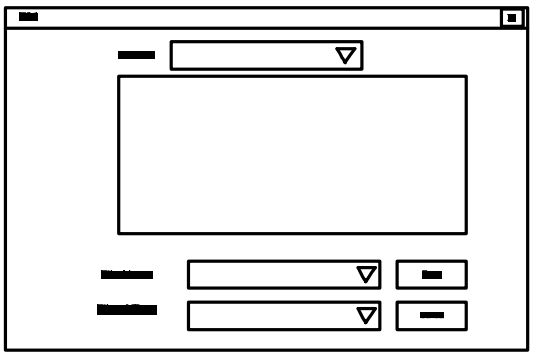
Tabel 3.2. Keterangan tampilan menu penyisipan

No Objek	Jenis Objek	Fungsi Objek
1	Main Menu	Untuk menampilkan menu utaman
2	Penyisipan	Untuk menunjukan menu penyisipan
3	ekstraksi	Untuk menunjukan menu ekstraksi
4	SNR	Untuk menu SNR
5	Information	Untuk menunjukan menu informasi
6	Pilih file	Untuk mengambil file midi yang akan dijadikan sebagai media penyisipan
7	Pilih File	Untuk mengambil file teks sebagai pesan yang akan disembunyikan
8	teksbox	Untuk menampilkan pesan yang akan disisipi

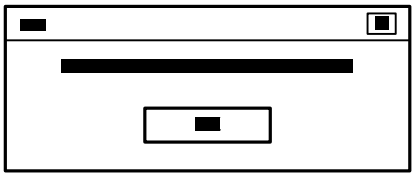
9	input	Untuk memasukan kunci
10	Button sisip	Untuk memulai proses penyisipan
11	Informasi event	Untuk menujukan informasi
12	tanggal	Untuk menunjukan tanggal

Deskripsi antar muka penyisipan pesan rahasia ke dalam berkas MIDI dapat dijabarkan sebagai berikut :

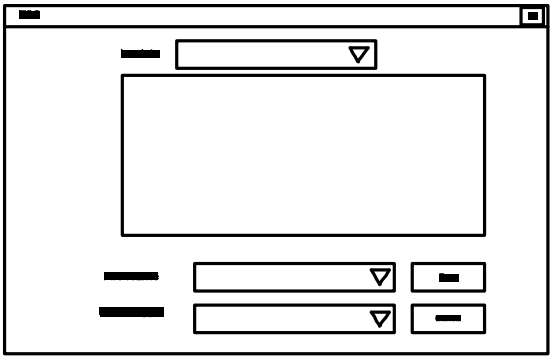
1. Tampilan layer *User* akan memilih dan memasukkan berkas MIDI, kemudian akan sebagai berikut:



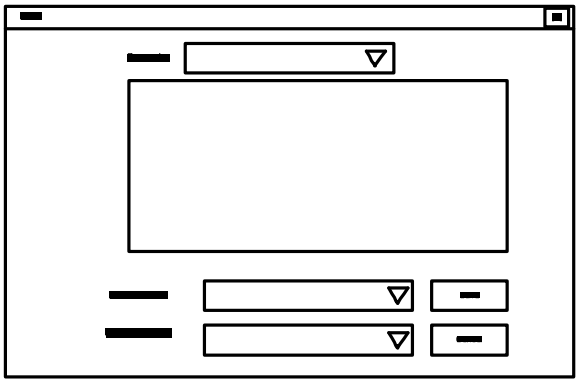
2. Tampilan pesan apabila yang dimasukan bukan berkas Midi sebagai berikut :



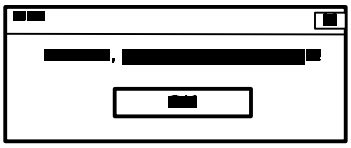
3. Tampilan layer *User* memasukkan berkas teks yang akan disisipkan, sebagai berikut:



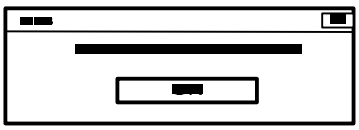
4. Proses dimana *User* memasukkan kunci dari *ketikan keyboard*.
5. Tampilan layer apabila tombol sisip di tekan, dan akan muncul tampilan dimana berkas midi yang telah tersisipkan berkas text disimpan :



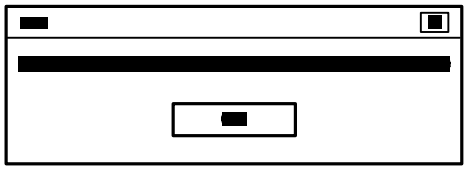
Tampilan form pesan apabila proses berhasil :



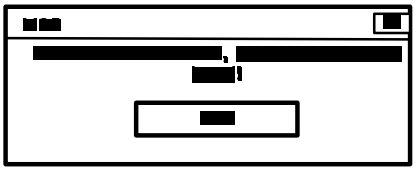
Tampilan form pesan apabila teks dimasukan telalu besar :



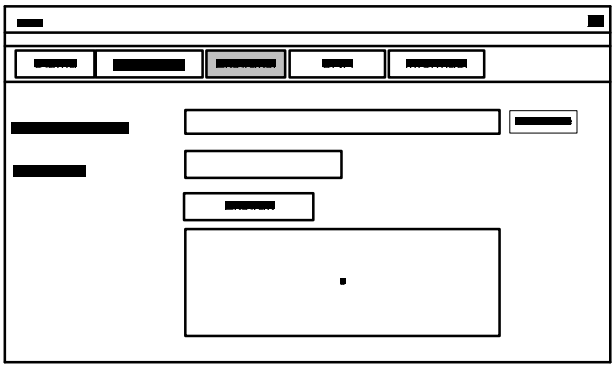
Tampilam form pesan informasi batasan ukuran teks yang bias disisipi :



Tampilan form pesan apabila proses berhenti atau gagal :



3.9.1.3 Layar Saji Proses Ekstraksi



Gambar 3.22 *Rancangan antar muka menu ekstraksi*

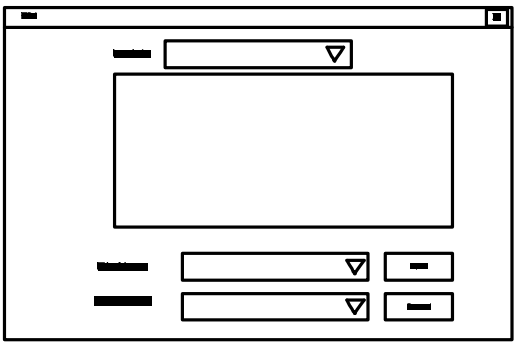
Tabel 3.3.Keterangan tampilan menu ekstraksi

No Objek	Jenis Objek	Fungsi Objek
1	Main Menu	Untuk menampilkan menu utaman
2	Penyisipan	Untuk menunjukan menu penyisipan
3	ekstraksi	Untuk menunjukan menu ekstraksi
4	SNR	Untuk menu SNR
5	Information	Untuk menunjukan menu informasi

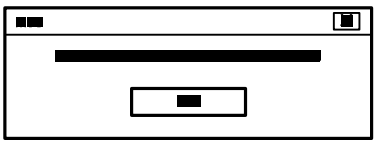
6	Pilih file	Untuk mengambil file midi yang telah disipi pesan dari storage
7	Box edit	Untuk memasukan kunci
8	Ekstraksi	Untuk menunjukan pada proses ekstraksi
9	Teksbox	Untuk menunjukan pesan yang telah terekstraksi
10	Informasi event	Untuk menujukan informasi
11	Tanggal	Untuk menunjukan tanggal

Deskripsi antar muka proses ekstraksi berkas MIDI dapat dijabarkan sebagai berikut :

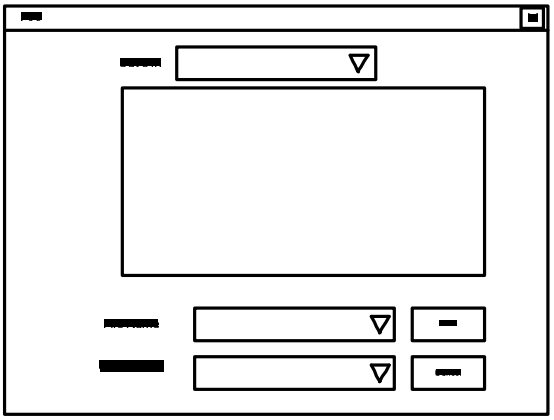
1. Bentuk tampilan apabila *User* akan memasukkan berkas MIDIStego, kemudian akan muncul layer sebagai berikut:



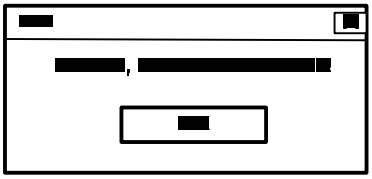
Tampilan form pesan apabila yang dimasukan bukan berkas Midistego sebagai berikut :



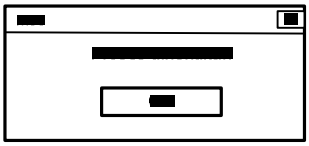
2. Masukan kunci sama seperti proses penyisipan
3. Form tampilan apabila tombol ekstraksi ditekan sebagai berikut :



Tampilan form pesan apabila porses ekstraksi berhasil:



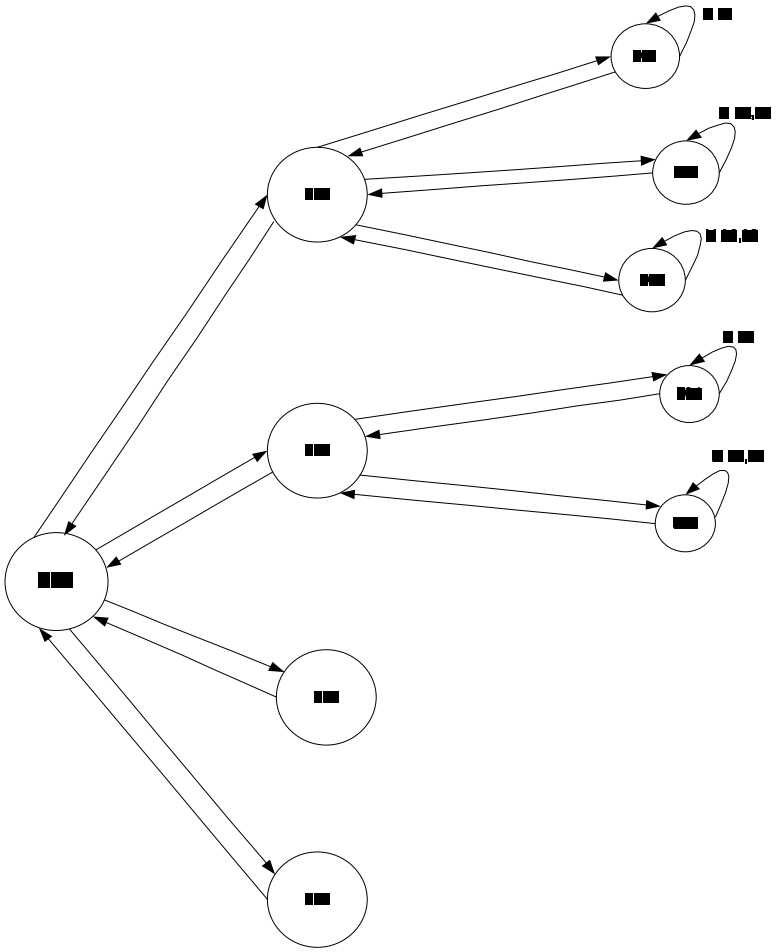
Tampilan form pesan apabila proses gagal atau dihentikan



3.10 Jaringan semantic

Jaringan semantik merupakan gambaran pengetahuan grafis yang menunjukan hubungan antara berbagai objek. Jaringan semantik terdiri dari lingkaran-lingkran yang menunjukan objek dan informasi tentang objek-objek tersebut. Kelebihan dari jaringan semantik adalah bias mewariskan.

3.10.1 Jaringan semantik steganografi pada midi



Gambar 3.23 Jaringan semantik steganografi pada midi

BAB IV

IMPLEMENTASI DAN PENGUJIAN SISTEM

4.1 Implementasi Sistem

Untuk menghasilkan sebuah berkas *audio digital* yang menyimpan informasi rahasia yang terintegrasi dalam perangkat lunak sesuai prosedur yang dibahas pada bab sebelumnya maka *cover file* yang akan dipakai adalah berkas dengan ekstensi *.mid, sedangkan informasi yang ingin disembunyikan bisa berupa berkas text(*.txt) atau pesan.

Penyembunyian informasi pada berkas MIDI dilakukan dengan menggunakan perangkat lunak yang telah dirancang, dimana perancangannya memanfaatkan teknik Phase coding yaitu dengan mensubstitusikannya pada berkas MIDI dengan bit-bit informasi.

4.1.1 Lingkungan Implementasi

Lingkungan implementasi yang dimaksud adalah lingkungan sistem yang digunakan untuk menjalankan aplikasi, yaitu berbasis *Windows*. Sistem operasi yang digunakan adalah sistem operasi *Microsoft Windows XP*.

4.1.2 Implementasi antar muka (*User Interface*)

Implementasi antarmuka dikembangkan dengan pendekataan metode berorientasi objek. Hal ini dilakukan karena adanya kemudahan penempatan dan pengaturan komponen visual pada layar saji dan tersedianya fasilitas tersebut pada

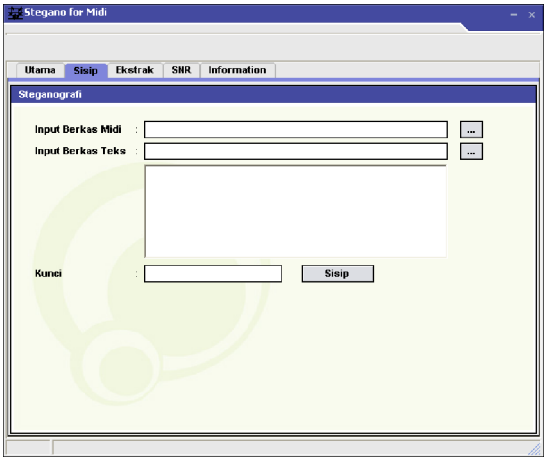
kompilator *Borland C++ Builder*. Antarmuka perangkat lunak diimplementasikan dalam bentuk jendela (*window*). Komponen masukan-keluaran *visual* diimplementasikan dari kelas-kelas visual dasar yang disediakan *Borland C++ Builder*. Terdapat 2 antarmuka utama, yaitu antarmuka proses *Penyisipan Pesan Rahasia ke dalam Berkas MIDI* dan antarmuka proses *Ekstraksi Pesan Rahasia dari Bekas MIDI Stego*.

4.1.2.1 Tampilan menu utama



Gambar 4.1 Tampilan Menu Utama

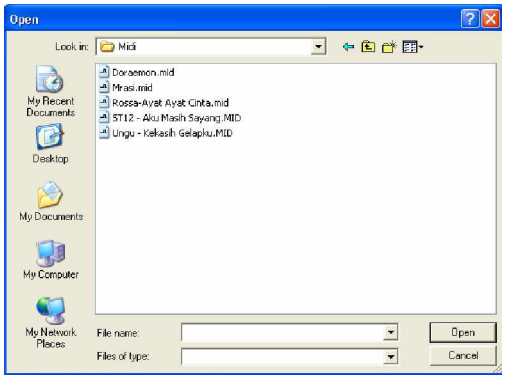
4.1.2.2 Tampilan antarmuka Proses Penyisipan Pesan Rahasia ke dalam Berkas MIDI



Gambar 4.2 Tampilan antar muka proses penyisipan

Pada antarmuka proses sisip, pengguna dapat memasukkan nama berkas MIDI dengan menekan tombol ”...” yang sesuai dengan baris Berkas MIDI. Selanjutnya dapat memasukkan nama berkas Text dengan menekan tombol ”...“ yang sesuai dengan baris Berkas Text maka isi dari pesan akan tampil pada kotak *edit* di bawahnya. Dan Pengguna diminta untuk memasukkan kunci pada baris kunci secara langsung. Kunci yang dimasukkan tidak akan ditampilkan, tetapi ditampilkan dalam bentuk karakter bintang (*). Bila ketiga jenis masukan ini telah terisi, maka pengguna dapat menekan tombol ”sisip”.

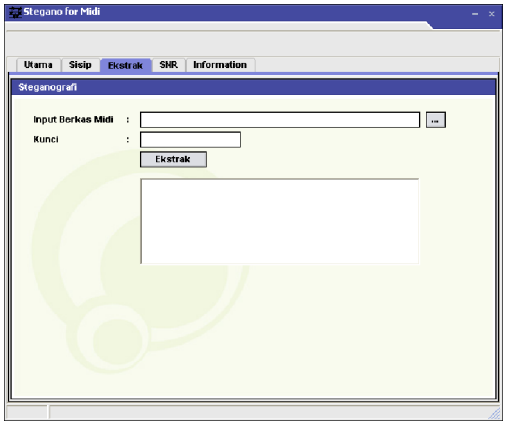
Kemudian simpan dimana lokasi penyimpanan yang diinginkan, setelah ditentukan tekan ‘Save’.



Apabila proses berhasil akan tampil kotak pesan seperti dibawah ini :



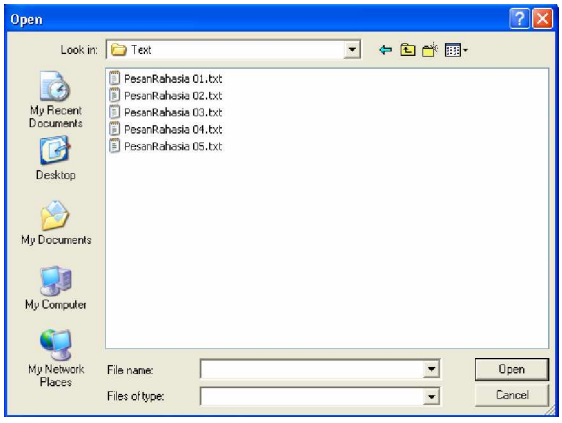
4.1.2.3 Tampilan antarmuka Proses Ekstraksi Pesan Rahasia dari Berkas MIDI Stego



Gambar 4.3 Tampilan antar muka proses ekstraksi

Pada antarmuka proses ekstrak, pengguna dapat memasukkan nama berkas MIDI Stego dengan menekan tombol ” ... ” yang sesuai dengan baris Berkas MIDI. Dan Pengguna diminta untuk memasukkan kunci pada baris kunci secara langsung sesuai dengan masukkan kunci pada proses penyisipan. Kunci yang dimasukkan tidak akan ditampilkan, tetapi ditampilkan dalam bentuk karakter bintang (*). Bila kedua jenis masukan ini telah terisi, maka pengguna dapat menekan tombol ”ekstrak”.

Kemudian simpan dimana lokasi penyimpanan yang diinginkan, setelah ditentukan tekan ‘Save’. Setelah itu hasil dari ekstraksi berupa berkas *txt maka isi dari berkas tersebut dapat dilihat pada kotak *edit* di bawahnya.



Apabila proses berhasil akan tampil kotak pesan seperti dibawah ini :



4.2 Pengujian Sistem

Pengujian perangkat lunak aplikasi ini menggunakan pengujian *blackbox*. Pengujian *blackbox* berfokus pada persyaratan fungsional perangkat lunak yang dibangun/dikembangkan.

4.2.1 Rencana Pengujian

Pengujian perangkat lunak MIDIStego dilaksanakan dengan tujuan agar perangkat lunak / sistem yang dibangun memiliki kualitas dan kinerja yang optimal baik dalam performa perangkat lunak dalam melakukan proses penyisipan dan ekstraksi, kesesuaian data yaitu pengujian kesesuaian antara data yang berhasil diekstrak dengan data yang disisipkan, kualitas suara yaitu pengujian sama tidaknya suara berkas MIDI original dengan suara berkas MIDI yang tersisipkan berkas Teks dan ketahanan untuk tidak terdeteksi oleh *software* yang mendeteksi ada atau tidaknya berkas pesan yang disembunyikan di dalam suatu berkas MIDI.

Tabel 4.1 Rencana Pengujian Aplikasi MIDIStego

Kelas Uji	**Butir Uji**	**Jenis Pengujian**
Sisip	Menyisipan pesan berupa text ke dalam berkas *audio* MIDI, memainkan kembali midi yang telah disisipi pesan.	*Blackbox*
Ekstraksi	Mengekstraksi berkas midi, berkas midi tersisipi pesan text dari berkas *audio* MIDI yang telah disisipkan pesan text.	*Blackbox*

4.2.2 Kasus Pengujian

4.2.2.1 Penyisipan

Berdasarkan rencana hasil pengujian dan yang telah disusun, maka dapat dilakukan pengujian sebagai berikut. Pengujian yang dilakukan meliputi pengujian penyisipan dan ekstraksi. Berikut ini merupakan pengujian-pengujian yang dilakukan : **Penyisipan**

Pengujian penyisipan pada aplikasi steganografi yang diberi nama MIDIStego dilakukan untuk menyisipkan pesan rahasia ke dalam berkas *audio* MIDI.

Tabel 4.2 Pengujian proses Penyisipan data normal

Kasus dan Hasil Uji (Data Normal)	
Data Masukan	Berkas MIDI, Berkas Teks dan Kunci
Yang diharapkan	Masukan berkas MIDI dengan format *.mid, berkas Teks dengan format *.txt dan masukan kunci dari ketikan dari *keyboard* oleh pengguna walaupun 1 karakter, kemudian tekan tombol **Sisip** maka sistem akan melakukan proses penyisipan berkas Teks ke dalam berkas MIDI.
Pengamatan	Proses penyisipan selesai dan berhasil.
Kesimpulan	Diterima

Tabel 4.3 Pengujian proses Penyisipan data salah

Kasus dan Hasil Uji (Data Salah)	
Data Masukan	Tidak lengkap atupun kosong
Yang diharapkan	Kemudian tekan tombol **Sisip** maka sistem akan menampilkan pesan.
Pengamatan	Tampil pesan sebagai berikut : "Data Masukkan Tidak Lengkap"
Kesimpulan	Diterima

4.2.2.2 Ekstraksi

Pengujian ekstraksi pada aplikasi steganografi yang diberi nama MIDIStego dilakukan untuk mengekstraksi pesan rahasia dari dalam berkas *audio* MIDI yang telah tersisipkan berkas teks.

Tabel 4.4 Pengujian proses Ekstraksi data normal

Kasus dan Hasil Uji (Data Normal)	
Data Masukan	Berkas MIDI yang tersisipkan Berkas Teks dan Kunci
Yang diharapkan	Masukan berkas MIDI yang tersisipkan Berkas Teks dengan format *.mid, dan masukan kunci sesuai dengan kunci proses penyisipan kemudian tekan tombol **Ekstrak** maka sistem akan melakukan proses ekstraksi berkas Teks dari berkas MIDI yang tersisipkan berkas Teks..

Pengamatan	Proses Ekstraksi selesai dan berhasil.
Kesimpulan	Diterima

Tabel 4.5 Pengujian proses Ekstraksi data salah pertama

Kasus dan Hasil Uji (Data Salah)	
Data Masukan	Berkas MIDI salah dan kunci
Yang diharapkan	Kemudian tekan tombol **Ekstrak** maka sistem akan menampilkan pesan.
Pengamatan	Tampil pesan sebagai berikut : “Proses dihentikan” dan selesai
Kesimpulan	Diterima

Tabel 4.6 Pengujian proses Ekstraksi data salah Kedua

Kasus dan Hasil Uji (Data Salah)	
Data Masukan	Berkas MIDI yang tersisipkan Berkas Teks dan kunci salah
Yang diharapkan	Kemudian tekan tombol **Ekstrak,** maka sistem akan melakukan proses ekstraksi berkas Teks dari berkas MIDI yang tersisipkan berkas Teks akan tetapi hasil akan tidak sesuai dengan Berkas teks yang tersisipkan.

Pengamatan	Proses Ekstraksi selesai dan berhasil.
Kesimpulan	Diterima

4.2.3 Hasil Pengujian

Dalam proses pengujian berikutnya adalah membuktikan bagaimana hasil dari proses penyisipan dan ekstraksi, yaitu pengujian kesesuaian antara data yang berhasil diekstrak dengan data yang disisipkan, kualitas suara yaitu pengujian sama tidaknya suara berkas MIDI original dengan suara berkas MIDI yang tersisipkan berkas Teks dan ketahanan untuk tidak terdeteksi oleh *software* yang mendeteksi ada atau tidaknya berkas pesan yang disembunyikan di dalam suatu berkas MIDI. Adapun berkas-berkas yang digunakan dalam pengujian yang digunakan adalah sebagai berikut:

1. **Berkas MIDI**, dengan *properties* sebagai berikut :

Tabel 4.7 Properties berkas MIDI

Nama Berkas Midi (byte)	**Ukuran (byte)**
Ungu-Kekasih gelapku.mid	32.192
Rosa-Ayat-ayat Cinta.mid	35.134
Ost Nobita.mid	52.261
ST12-Aku masih sayang.mid	43.161

Tabel 4.8 Properties berkas Teks

Nama Berkas Teks	Ukuran Teks (byte)
[12]PesanRahasia 01.txt	12
[147]PesanRahasia 02.txt	147
[294]PesanRahasia03.txt	294
[592]PesanRahasia 03.txt	592
[1019]PesanRahasia 04.txt	1019
[1658]PesanRahasia 05.txt	1658
[2484]PesanRahasia 06.txt	2484
[3182]PesanRahasia 07.txt	3182

4.2.3.1 Hasil Keluaran Penyisipan dan Ekstraksi

Tabel 4.9 Tabel Hasil Pengujian Proses Penyisipan dan ekstraksi

No.	Panjang Berkas MIDI (Byte)	Panjang Berkas Pesan (Byte)	Status Penyisipan	Dapat dimainkan	Status Ekstraksi	Kesesuaian
Ungu-Kekasih gelapku.mid						
1	32.192	12	Berhasil	Ya	Berhasil	Ya
2	32.192	147	Berhasil	Ya	Berhasil	Ya
3	32.192	294	Berhasil	Ya	Berhasil	Ya
4	32.192	592	Tidak dapat disisipi	-	-	-
5	32.192	1019	Tidak dapat disisipi	-	-	-
6	32.192	1658	Tidak dapat disisipi	-	-	-
7	32.192	2484	Tidak dapat	-	-	-

			disisipi			
8	32.192	3182	Tidak dapat disisipi	-	-	-
Rosa-Ayat-ayat Cinta.mid						
1	35.134	12	Berhasil	Ya	Berhasil	Ya
2	35.134	147	Berhasil	Ya	Berhasil	Ya
3	35.134	294	Berhasil	Ya	Berhasil	Ya
4	35.134	592	Tidak dapat disisipi	-	-	-
5	35.134	1019	Tidak dapat disisipi	-	-	-
6	35.134	1658	Tidak dapat disisipi	-	-	-
7	35.134	2484	Tidak dapat disisipi	-	-	-
8	35.134	3182	Tidak dapat disisipi	-	-	-
Ost Nobita.mid						
1	52.261	12	Berhasil	Ya	Berhasil	Ya
2	52.261	147	Berhasil	Ya	Berhasil	Ya
3	52.261	294	Berhasil	Ya	Berhasil	Ya
4	52.261	592	Berhasil	Ya	Berhasil	Ya
5	52.261	1019	Tidak dapat disisipi	-	-	-
6	52.261	1658	Tidak dapat disisipi	-	-	-
7	52.261	2484	Tidak dapat disisipi	-	-	-
8	52.261	3182	Tidak dapat disisipi	-	-	-
ST12-Aku masih sayang.mid						
1	43.161	12	Berhasil	Ya	Berhasil	Ya
2	43.161	147	Berhasil	Ya	Berhasil	Ya
3	43.161	294	Berhasil	Ya	Berhasil	Ya
4	43.161	592	Berhasil	Ya	Berhasil	Ya
5	43.161	1019	Tidak dapat disisipi	-	-	-
6	43.161	1658	Tidak dapat disisipi	-	-	-
7	43.161	2484	Tidak dapat disisipi	-	-	-
8	43.161	3182	Tidak dapat disisipi	-	-	-

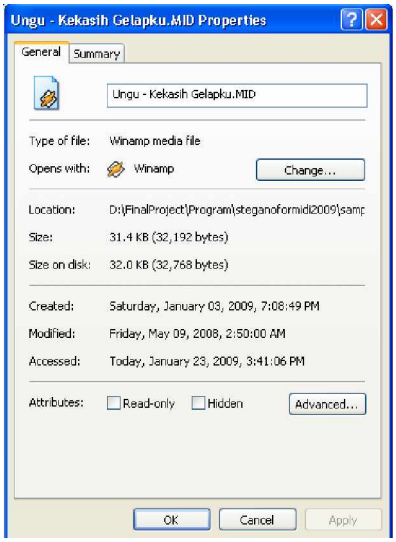
Dari hasil pengujian hasil penyisipan dan ekstraksi dapat disimpulkan bahwa kapasitas teks(*.txt) yang dapat disisipkan bergantung pada ukuran midi sebagai penampung, adapun dari hasil pengujian maksimal ukuran teks(*.txt) yang dapat disisipkan 1byte – 592byte.

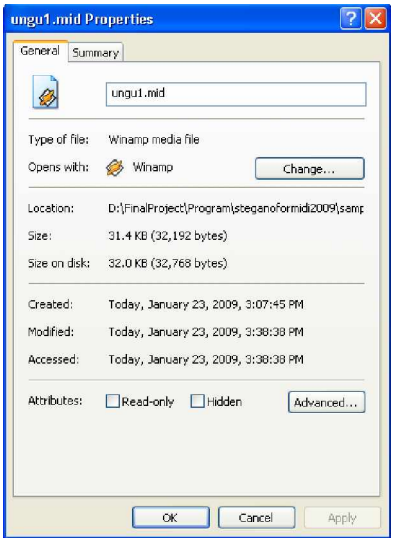
4.2.3.2 Hasil Pengujian Ekstraksi terhadap midi yang belum disisipi

Tabel 4.10 Tabel Hasil Pengujian Ekstraksi terhadap midi yang belum disisipi

No	Nama Berkas Midi	Ukuran	Status Midi	Hasil Ekstraksi
1	Ungu-Kekasih gelapku.mid	32.192	Belum disisipi	Proses dihentikan
2	Rosa-Ayat-ayat Cinta.mid	35.134	Belum disisipi	Proses dihentikan
3	Ost Nobita.mid	52.261	Belum disisipi	Proses dihentikan
4	ST12-Aku masih sayang.mid	43.161	Belum disisipi	Proses dihentikan

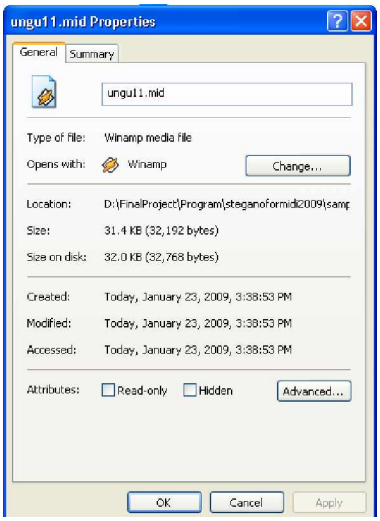
Setelah proses *steganography* dalam hal ini proses penyisipan, dengan melihat *properties* antara berkas MIDI original dengan berkas MIDI yang telah disisipkan pesan/text, diharapkan untuk menghindari kecurigaan bahwa di dalam berkas MIDI yang telah disisipkan pesan/text tersebut ada sesuatu yang lain, berhubung teknik *steganography* yang digunakan dalam penelitian tugas akhir ini adalah dengan algoritma *Phase coding* menggunakan teknik Phase coding , maka secara teori ukuran berkas *output* tidak akan berbeda dengan berkas *original*nya. Pembuktian ini dapat dilihat pada Gambar 4.4 dan 4.5

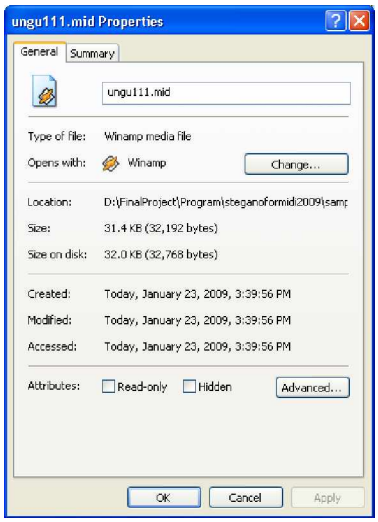




Gambar 4.4 Properties berkas MIDI original ***Properties midi pengujian-1***

Gambar diatas memperlihatkan perbandingan properties antara properties midi ungu-kekasih gelapku.mid dengan properties ungu-kekasih gelapku.mid dimana tidak terjadi perubahan pada propertiesnya





Properties midi pengujian-2 ***Properties midi pengujian-3***

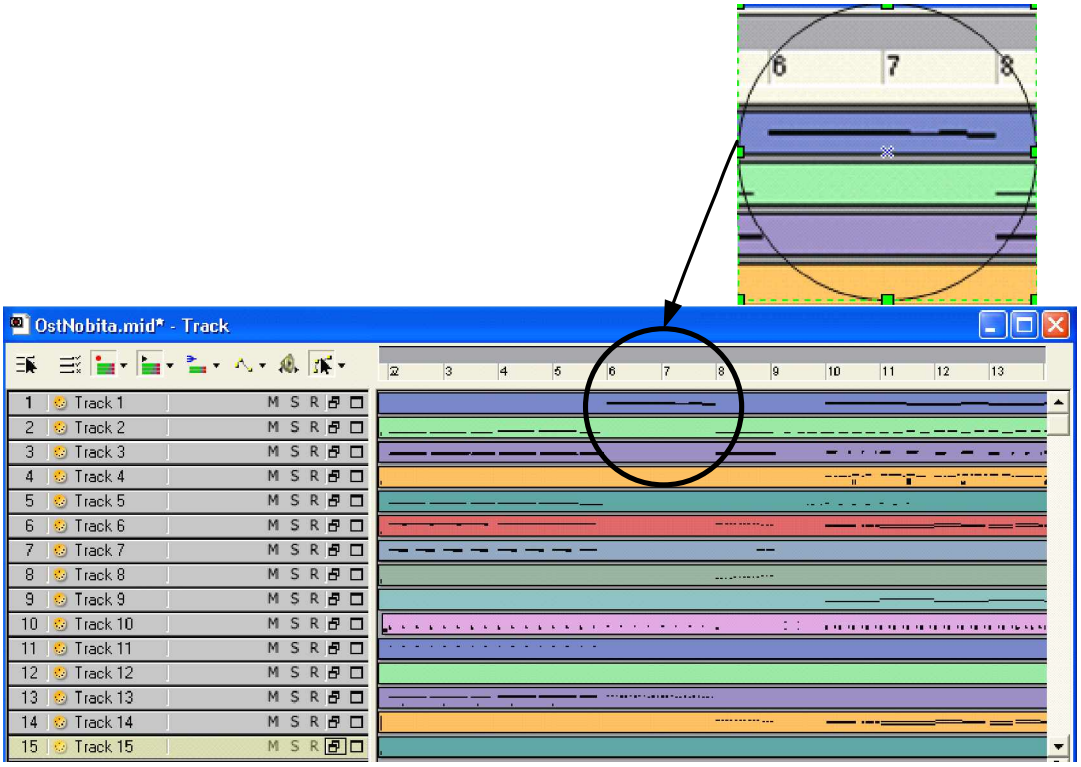
Pada gambar 4.4 dan keterangan *properties* dari berkas MIDI original dengan berkas MIDI yang telah tersisip pesan tidak mengalami perubahan meskipun telah mengalami proses *steganography* dengan tersisipnya berkas text di dalamnya. Jadi dengan teknik phase coding, dalam penelitian tugas akhir ini penulis dapat menyembunyikan suatu berkas text di dalam berkas *audio digital* dalam hal ini adalah Berkas MIDI.

Dengan tidak berubahnya ukuran berkas MIDI yang telah tersisip pesan/text yang *significant* terhadap berkas MIDI originalnya, kemungkinan besar orang lain tidak akan mengetahui bahwa ada suatu informasi di dalam berkas MIDI tersebut.

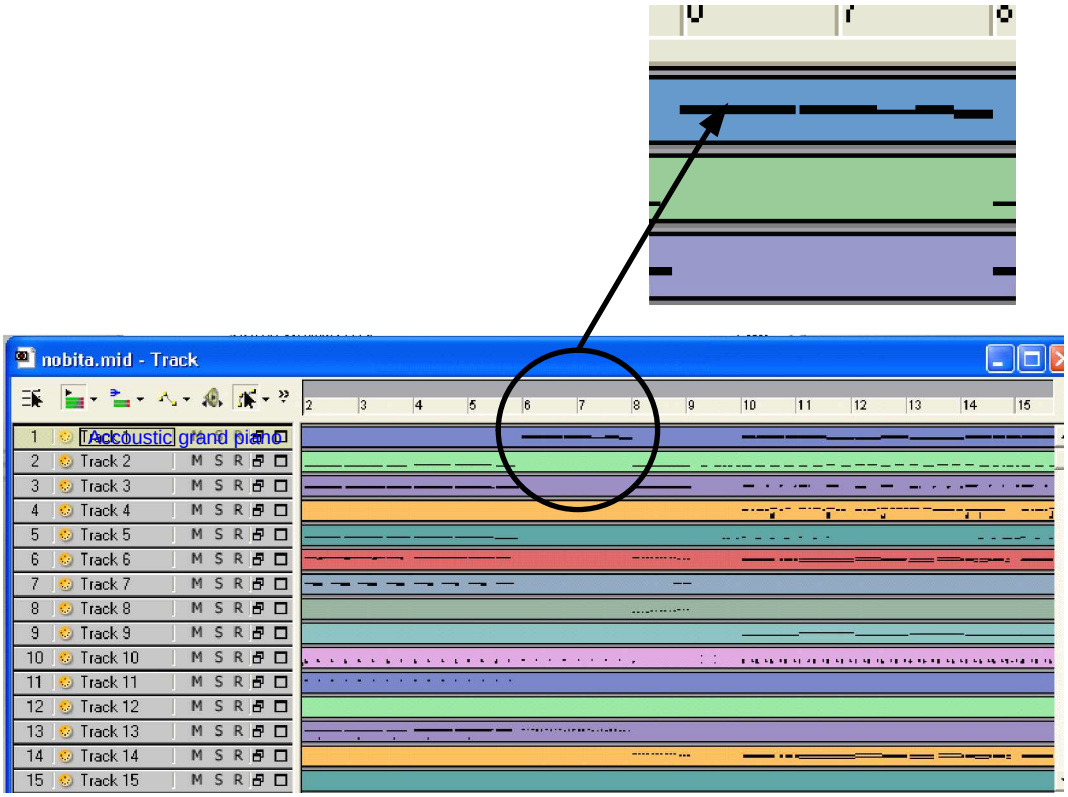
4.2.3.3 Hasil pengujian perbandingan visualisasi gelombang sinyal terhadap midi dengan midi yang sudah tersisipi pesan

Pada bagian ini akan dilakukan pengujian perbandingan visualisasi gelombang sinyal terhadap gelombang sinyal midi dengan gelombang sinyal midi yang sudah tersisipi, hal ini dilakukan untuk melihat perubahan-perubahan yang terjadi pada midi dan midi yang sudah tersisipi.

1. OstNobita.mid



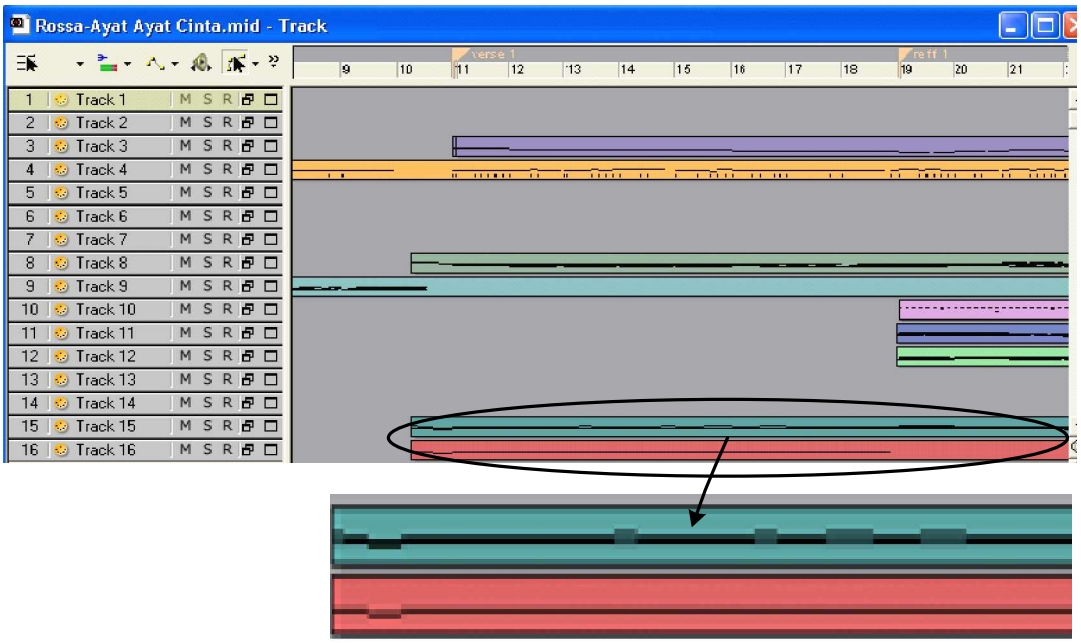
Gambar 4.5. Gelombang sinyal pada midi



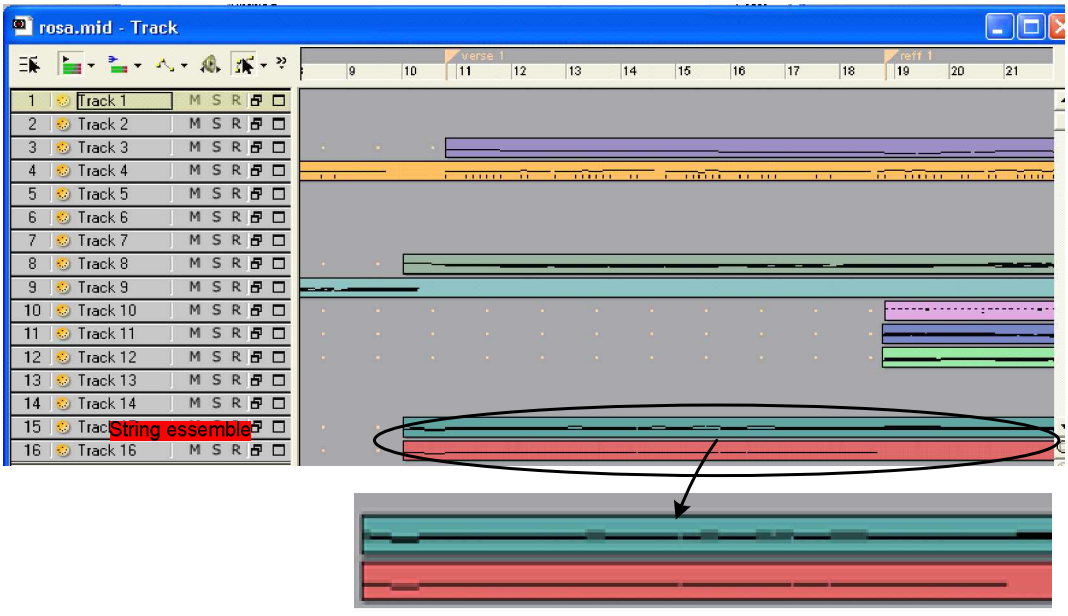
Gambar 4.6 Gelombang sinyal pada midi yang sudah tersisipi pesan

Dari dua perbandingan gambar antara sinyal gelombang midi nobita.mid dan sinyal gelombang midi yang sudah tersisipi pesan .txt dapat dilihat terjadi perubahan pada sinyal gelombang acoustic grand piano pada midi tersisipi pesan .txt. Pada track 1 sinyal gelombang midi tersisipi pesan terlihat ada sinyal gelombang yang terputus, begitu juga pada track 6 terjadi perubahan sinyal gelombang. Hal ini disebabkan terjadi perubahan pada sinyal gelombang yang disebabkan oleh noise yang disebabkan oleh pesan yang tersisipi pada midi.

2. Rossa.mid



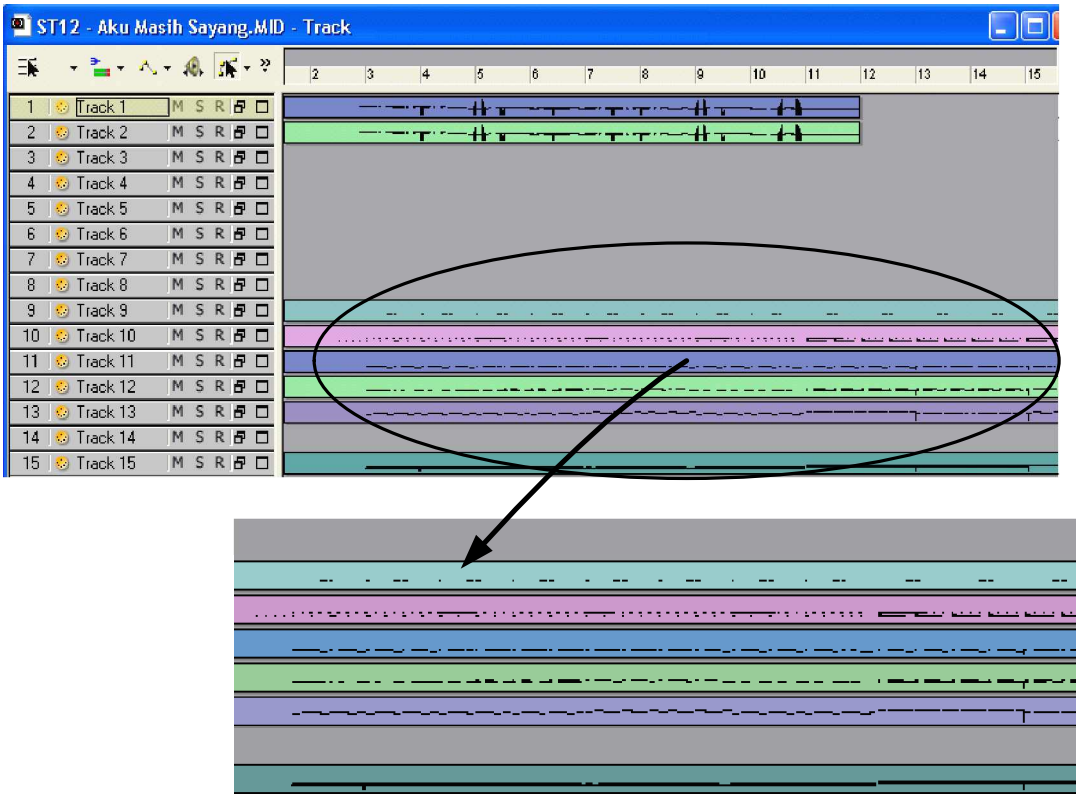
Gambar 4.7. Gelombang sinyal pada midi



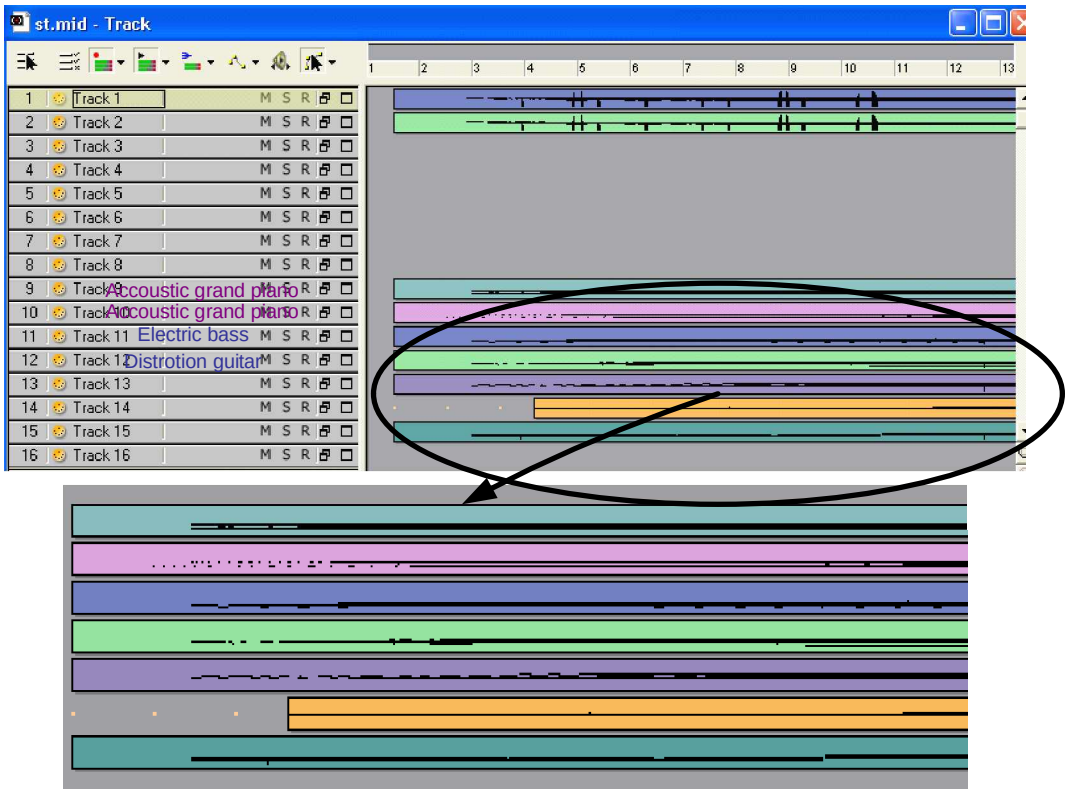
Gambar 4.8. Gelombang sinyal pada midi yang sudah tersisipi pesan

Dari dua perbandingan gambar antara sinyal gelombang midi rosa.mid dan gelombang sinyal midi yang sudah tersisipi pesan .txt, dapat dilihat terjadi perubahan sinyal gelombang string essemble pada midi rosa.mid yang sudah tersisipi pesan .txt. Pada track 15 terlihat sinyal gelombang yang terputus-putus. Hal ini juga disebabkna noise yang ditimbulkan oleh pesan yang tersisipi dalam midi tersebut.

3. ST12.mid



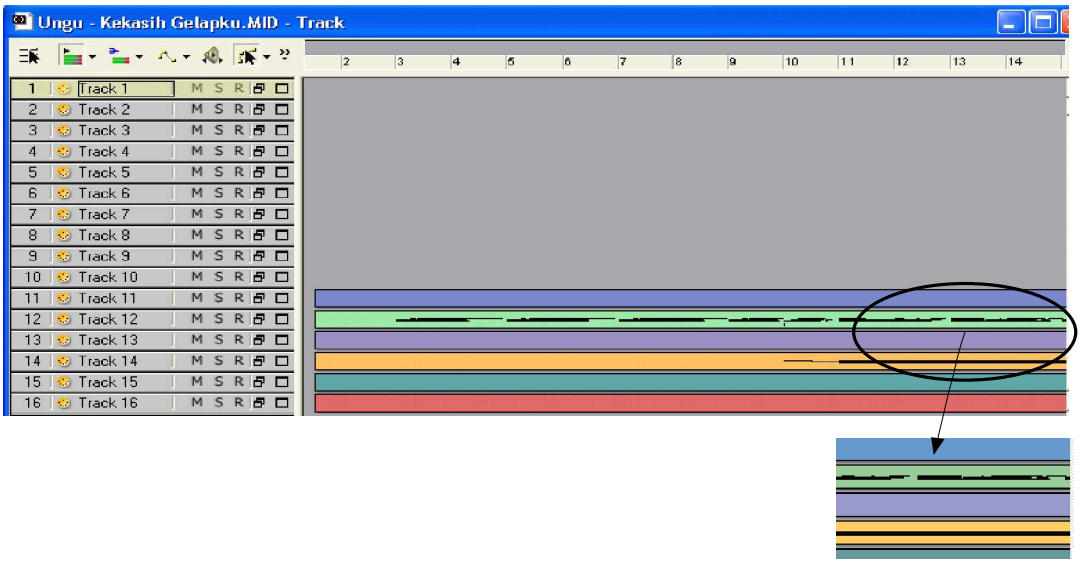
Gambar 4.9 Gelombang sinyal pada midi



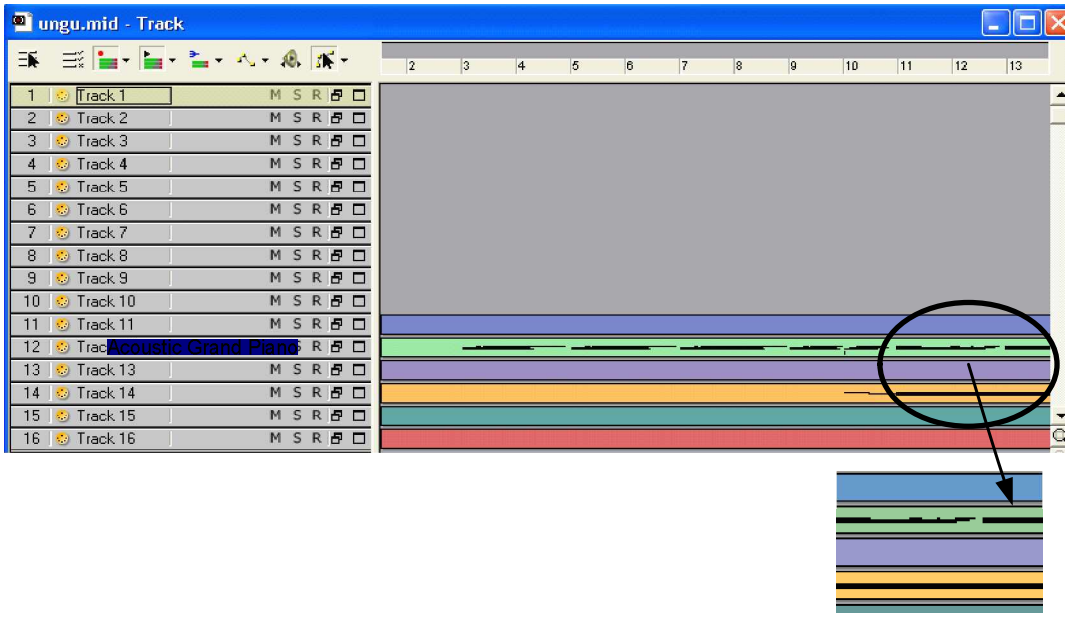
Gambar 4.10. Gelombang sinyal pada midi yang sudah tersisipi pesan

Dari dua perbandingan gambar antara sinyal gelombang midi st12.mid dan sinyal gelombang acoustic grand piano, electric bass,distortion guitar midi yang sudah tersisipi pesan teks.txt dapat dilihat terjadi perubahan pada sinyal gelombang midi tersisipi pesan. Pada track 9,10,11,12,13,14,15, terlihat perubahan yang sangat besar pada sinyal gelombang nidi yang tersisipi pesan. Sebagai contoh pada track 9, sinyal gelombang midi terputus-putus sedangkan sinyal gelombang pada midi yang tersisipi pesan terlihat sinyal gelombang yang sangat solid .Dapat disimpulkan noise yang ditimbulkan oleh pesan yang disisipka pada midi tersebut sangat besar.

4. Ungu.mid



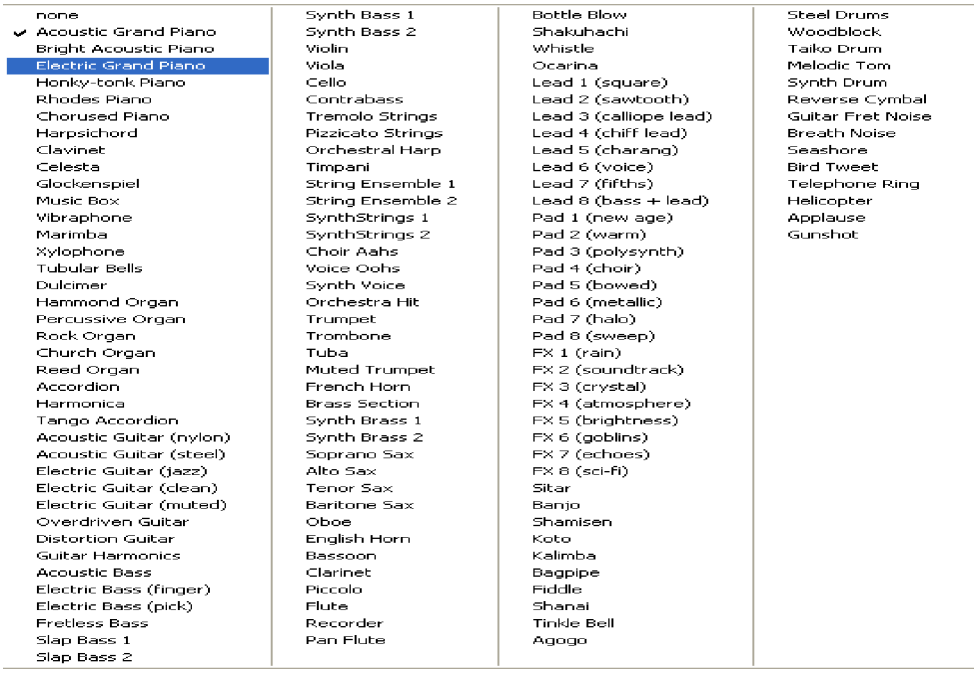
Gambar 4.11 Gelombang sinyal pada midi



Gambar 4.12 Gelombang sinyal pada midi yang sudah tersisipi pesan

Dari dua perbandingan gambar antara sinyal gelombang midi ungu.mid dan gelombang sinyal midi yang sudah tersisipi pesan teks.txt, secara pandangan mata tidak terlihat perubahan yang sangat besar atau mencolok pada sinyal gelombang midi ungu.mid. Tapi apabila dilihat secara teliti pada track 12 dapat dilihat perubahan yang terjadi pada sinyal gelombang midi diatas. Jadi pada midi ungu.mid noise yang ditimbulkan sangat kecil atau hamper tidak ada.

Dibawah ini merupakan instrument-instrumen yang berpengaruh dalam sebuah midi.



Gambar 4.13 Instrument-inturment yang berpengaruh pada sebuah midi

Jadi dari hasil perbandingan sinyal gelombang antara midi dan midi yang tersisipi pesan dapat disimpulkan perubahan sinyal gelombang pasti terjadi terhadadap midi yang tersisipi pesan, tapi seberapa besar perubahan yang akan terjadi pada midi yang tersisipi pesan tersebut tidak dapat diprediksi atau diketahui dengan tepat.

4.2.3.3 Hasil Pengujian Ketahanan Terhadap Deteksi dan Cracking

Dalam bagian pengujian ketahanan ini, penulis akan mengujikan *output* perangkat lunak dengan menggunakan *software* yang telah diakui kemampuannya untuk melakukan deteksi dan *cracking* terhadap *file steganography. software* deteksi yang dimaksud yaitu *Steg Spy v2.1*.

StegSpy dapat mengidentifikasi suatu *file stego* dimana *Steg-spy* ini dapat mendeteksi steganografi yang digunakan untuk menyembunyikan informasi/pesan. Versi yang paling terakhir juga mengidentifikasi lokasi isi pesan yang disembunyikan dimana dengan cara menemukan kunci yang digunakan pada saat proses penyisipan. *StegSpy* sekarang ini mengidentifikasi program-program steganografi yang ada seperti: *Hiderman, JPHideandSeek, Masker, JPegX,* dan program-program aplikasi *data hiding/*penyembunyian data lainnya.

Adapun program aplikasi *Steg-spy* yang dimaksud didapat dilihat pada gambar 4.14. Dimana tombol *Run* berfungsi untuk membuka *file* steganografi sekaligus melakukan deteksi dan *cracking* terhadap *file* steganografi.

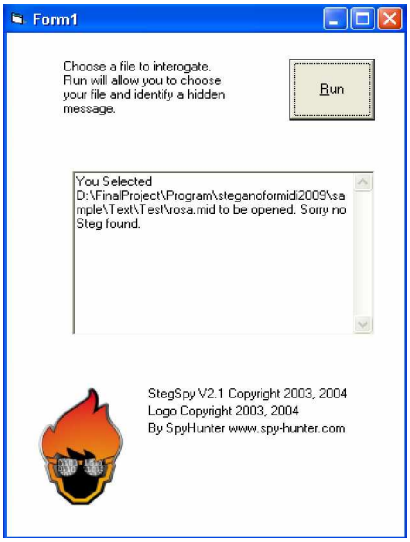


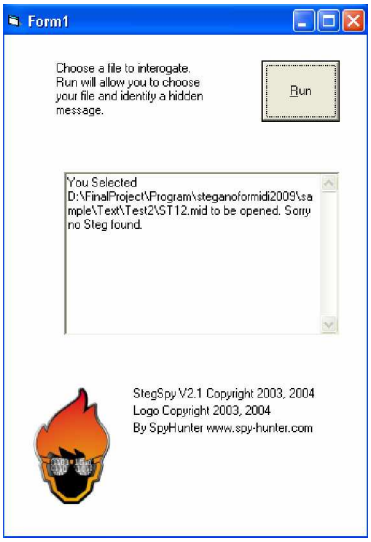
Gambar 4.14 Tampilan Main Window dari Steg Spy v2.1

Contoh apabila *stegspy* ini dapat mengidentifikasi *file stego* yang ada pada berkas media penampung seperti pada gambar 4.15



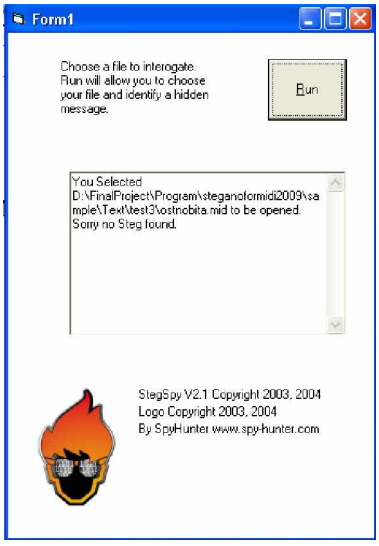
Gambar 4.15 Hasil deteksi file Stego

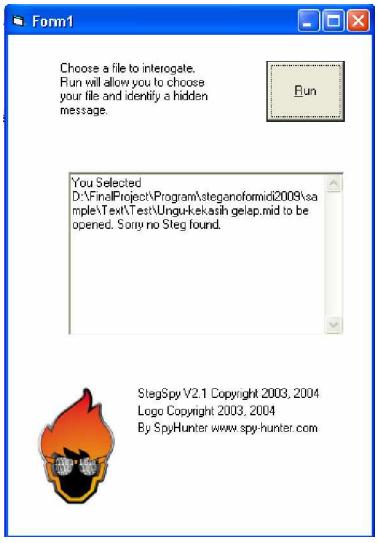




Pengujian Rosa.Mid ***Pengujian ST12.mid***

Pada pengujian menggunakan aplikasi pendukung *StegSpy* untuk mendeteksi keberadaan stego file dalam file midi rosa.mid dan st12.mid, aplikasi tersebut tidak dapat mendeteksi adanya keberadaan pesan yang sudah tersisipi kedalam berkas midi tersebut.





Pengujian Ostnobita.mid ***Pengujian Ungu-Kekasih Gelap.mid***

Dari hasil pengamatan melalui pengujian terhadap berkas MIDI yang telah disisipkan, *Steg Spy* tidak mampu untuk mendeteksi adanya informasi rahasia di dalam berkas MIDI tersebut seperti terlihat pada Gambar 4.15. Jadi dari segi keamanan dan kapasitas proses penyisipan pesan dengan algoritma *Phase coding* memberikan hasil yang memuaskan, dimana secara kasat mata orang lain tidak akan mengetahui bahwa di dalam berkas MIDI terdapat suatu informasi rahasia, karena dari segi ukuran berkas MIDI tidak mengalami perubahan, asalkan orang lain tidak mengetahui berkas MIDI originalnya, sangat sulit untuk mendeteksi adanya informasi rahasia.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil pengujian dan analisis yang dilakukan terhadap sistem yang dibangun maka dapat ditarik kesimpulan bahwa :

1. Steganografi dapat dilakukan pada berkas MIDI, khususnya dengan modifikasi nilai fase sinyal suara.
2. Dari pengujian yang telah dilakukan, sampai saat ini *output* proses Steganografi dengan teknik Phase coding yang dibangun terbukti memiliki ketahanan terhadap deteksi dan *cracking,* sehingga dapat mengurangi atau mencegah terjadinya penyadapan pesan rahasia.
3. Kapasitas pesan text yang dapat ditampung dalam berkas MIDI, dipengaruhi oleh panjang segmen pembagi suaradan modifikasi sinyal suara dalam berkas midi. Ukuran berkas teks yang dapat disisipkan dengan teknik phase coding relatif lebih kecil dibandingkan dengan teknik audio steganografi lainnya.hal itu disebabkan karena ukuran berkas teks yang disisipkan tidak boleh lebih dari panjang segmen pembagian suara dan modifikasi yang semakin banyak.

5.2 Saran

Berdasarkan kesimpulan diatas untuk lebih meningkatkan faktor keamanan data yang dikirimkan maka untuk proses pengembangan sistem selanjutnya, penulis memberikan saran bahwa:

1. Dalam Tugas Akhir ini, data audio yang didukung hanyalah yang memiliki format MIDI. Untuk pengembangan lebih lanjut, perangkat lunak dapat dibuat sehingga mendukung watermarking pada format data audio lain, seperti WAV, MP3, dan lain sebagainya.
2. Format data yang dapat disembunyikan pada MIDIStego ini berkas teks (*.txt), oleh karena itu format masukkan untuk data yang akan disembunyikan dapat menggunakan semua format berkas seperti : word (*.doc/*.docx).
3. MIDIStego yang dibangun masih untuk perangkat keras PC. Akan lebih praktis apabila juga dikembangkan lebih lanjut untuk dapat digunakan dalam lingkungan perangkat keras *mobile* seperti telepon genggam.
4. Masih menggunakan program aplikasi lain untuk menguji perubahan jumlah bit dan kehandalan/ ketahanan terhadap deteksi/*craking.*



Daftar Riwayat Hidup

DATA DIRI		
Nama	:	Dadang Sujana
Tempat Lahir	:	Aimere-Ngada
Tanggal Lahir	:	26 Mei 1984
Jenis Kelamin	:	Laki-Laki
Status	:	Belum Kawin
Kewarganegaraan	:	Indonesia
Alamat	:	Jln.Jalan Raya, Aimere-Ngada Flores-NTT
Hp.	:	081-321277630
Email	:	ddank_05@yahoo.com / d2ng_474@telkom.net

PENDIDIKAN FORMAL		
Sekolah	Jurusan	Tahun
Universitas Komputer Indonesia (UNIKOM) Bandung	Teknik Informatika	2003 - 2009
SMU Negeri 1 Aimere-Ngada	IPA	1999 - 2002
SLTP Negeri 1 Aimere-Ngada	-	1996 - 1999
SDN 1 Aimere-Ngada	-	1990 - 1996