Studi dan Implementasi Steganografi pada Berkas MIDI

Agus Susanto
Departemen Teknik Informatika
Institut Teknologi Bandung
Jalan Ganesha 10 Bandung 40132

E-mail : agus99010@yahoo.com

Abstrak

Steganografi merupakan ilmu dan seni yang mempelajari cara penyembunyian informasi pada suatu media sedemikian rupa sehingga keberadaannya tidak terdeteksi oleh pihak lain yang tidak berhak atas informasi tersebut. Pengguna pertama (pengirim pesan) dapat mengirim media yang telah disisipi informasi rahasia tersebut melalui jalur komunikasi publik, hingga dapat diterima oleh pengguna kedua (penerima pesan). Penerima pesan dapat mengekstraksi informasi rahasia yang ada di dalamnya.

Pada Tugas Akhir ini dikembangkan sebuah perangkat lunak steganografi pada berkas MIDI yang diberi nama MIDSteM (*MIDI's Steganography Machine*). Algoritma steganografi MIDSteM yang dikembangkan menggunakan metode *spread spectrum*. MIDSteM diimplemantasikan dalam lingkungan sistem operasi Windows 32-bit, menggunakan bahasa pemrograman C#. Karakteristik file MIDI yang relatif berukuran kecil dibandingkan format berkas audio lainnya menjadi sebuah tantangan untuk memanfaatkannya sebagai *carrier* dari pesan rahasia yang akan disisipkan. Kunci dimanfaatkan untuk menjaga aspek keamanan dari pesan rahasia yang telah disisipkan.

Setelah dilakukan pengujian, akan dapat dilihat apakah MIDSteM dapat menyembunyikan pesan rahasia yang relatif singkat ke dalam berkas MIDI dan dapat mengekstrak kembali pesan rahasia tersebut. Selain itu juga akan dapat dilihat pengaruhnya terhadap kualitas berkas MIDI yang telah disisipi data. Kualitas berkas MIDI ditentukan berdasarkan perbedaan amplitudo tiap kunci not pada berkas MIDI sebelum dan setelah disisipi data. Setelah didapatkan nilainya, maka dapat ditentukan apakah kualitasnya jauh berkurang atau tidak.

Kata kunci : Teknik penyembunyian data, data hiding, Steganografi, MIDI, spread spectrum.

1. Pendahuluan

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.

Salah satu hal yang dapat dilakukan untuk mengatasi situasi di atas 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 bisa 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.

Kebanyakan aplikasi steganografi tidak menggunakan prinsip Kerckhoffs, yang menyatakan bahwa metode steganografi sebaiknya diketahui publik, sedangkan untuk keamanannya dapat

digunakan kunci untuk mengekstraksi pesan yang telah disisipkan [MOE03]. Oleh karena algoritma steganografi bisa digunakan secara publik, maka steganografi perlu juga memberikan keamanan terhadap data yang telah disembunyikan dengan menggunakan kunci yang digunakan untuk mengacak susunan data pada media. Kunci yang dipakai adalah kunci simetris, artinya kunci pada pengirim sama persis dengan kunci pada penerima.

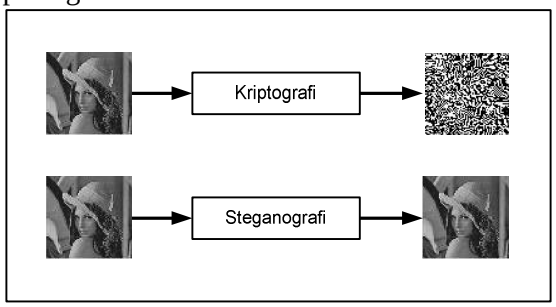
Salah satu jenis multimedia yang banyak digunakan adalah berkas media MIDI (*Musical Instrumental Device Interface*), yang umum digunakan sebagai polyponic ringtone pada pesawat telepon selular. Media MIDI juga telah lama digunakan sebagai latar musik sederhana dari berbagi perangkat lunak. Banyaknya penggunaan media MIDI ini, memungkinkan sekali apabila format MIDI ini dijadikan sebagai container dari data yang akan disembunyikan pada proses steganografi. Format MIDI berisi data musik yang berbasis event bukan sampel dari data suara yang akan dimainkan. Hal ini memungkinkan penyembunyian data juga berbasis event.

Rancangan aplikasi steganografi yang akan dikembangkan memiliki kriteria mampu melakukan proses penyembunyian data, dapat melakukan proses *recovery* terhadap data yang telah disembunyikan, serta mampu mengukur tingkat kualitas berkas MIDI sebelum dan sesudah proses steganografi, yang dihitung berdasarkan perbandingan amplitudo dari tiap kunci not dalam berkas MIDI.

2. Definisi Steganografi

Steganografi (*steganography*) adalah ilmu dan seni menyembunyikan pesan rahasia (*hiding message*) sedemikian sehingga keberadaan (eksistensi) pesan tidak terdeteksi oleh indera manusia [MUN04]. 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.



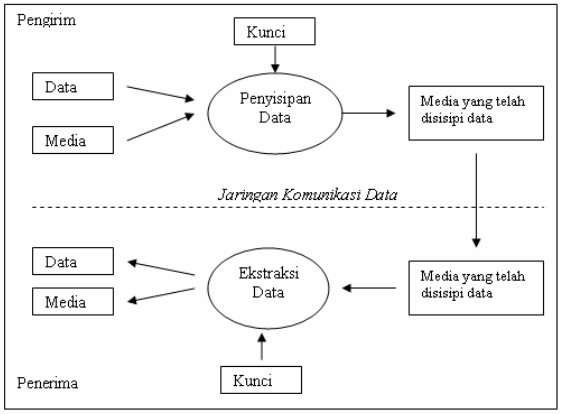
Gambar 2-1. Ilustrasi kriptografi dan steganografi pada citra digital.

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 [MUN04]:

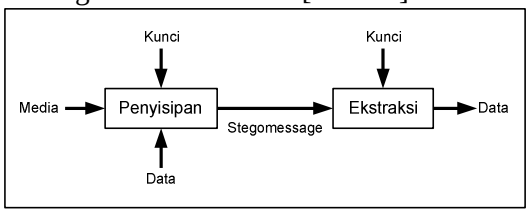
1) *Fidelity*. Mutu citra penampung tidak jauh berubah. Setelah penambahan data rahasia, citra hasil steganografi masih terlihat dengan baik. Pengamat tidak mengetahui kalau di dalam citra tersebut terdapat data rahasia.

2) *Recovery*. Data yang disembunyikan harus dapat diungkapkan kembali (*recovery*). Karena tujuan steganografi adalah *data hiding*, maka sewaktu-waktu data rahasia di dalam citra penampung harus dapat diambil kembali untuk digunakan lebih lanjut.



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 [SUH04].



Gambar 2-3 Proses penyisipan dan ekstraksi dalam steganografi

3. 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.

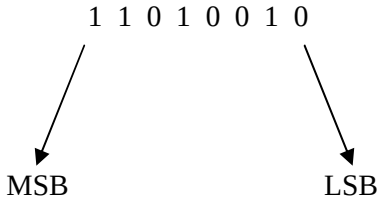
Bangsa Romawi mengenal steganografi dengan menggunakan tinta tak-tampak (*invisible ink*) untuk menuliskan pesan. Tinta tersebut dibuat dari campuran sari buah, susu, dan cuka. Jika tinta digunakan untuk menulis maka tulisannya tidak tampak. Tulisan di atas kertas dapat dibaca dengan cara memanaskan 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*) [MUN04].

4. Metode Modifikasi LSB

Sistem Steganografi akan menyembunyikan sejumlah informasi dalam suatu berkas dan akan mengembalikan informasi tersebut kepada pengguna yang berhak. Terdapat dua langkah dalam sistem Steganografi yaitu proses penyembunyian dan *recovery* data dari berkas penampung.

Penyembunyian data dilakukan dengan mengganti bit-bit data di dalam segmen citra dengan bit-bit data rahasia. Metode yang paling sederhana adalah metode modifikasi LSB (*Least Significant Bit Modification*). Pada susunan bit di dalam sebuah byte (1 byte = 8 bit), ada bit yang paling berarti (*most significant bit* atau MSB) dan bit yang paling kurang berarti (*least significant bit* atau LSB).. Contoh :



Bit yang cocok untuk diganti adalah bit LSB, sebab perubahan tersebut hanya mengubah nilai byte satu lebih tinggi atau satu lebih rendah dari nilai sebelumnya. Misalkan byte tersebut menyatakan warna merah, maka perubahan satu bit LSB tidak mengubah warna merah tersebut secara berarti. Lagi pula, mata manusia tidak dapat membedakan perubahan yang kecil.

Misalkan segmen data citra sebelum perubahan:

```
0 0 1 1 0 0 1 1   1 0 1 0 0 0 1 0
1 1 1 0 0 0 1 0   0 1 1 0 1 1 1 1
```

Segmen data citra setelah ‘0 1 1 1‘ disembunyikan:

```
0 0 1 1 0 0 1 0   1 0 1 0 0 0 1 1
1 1 1 0 0 0 1 1   0 1 1 0 1 1 1 1
```

Untuk memperkuat teknik penyembunyian data, bit-bit data rahasia tidak digunakan mengganti byte-byte yang berurutan, namun dipilih susunan byte secara acak. Misalnya jika terdapat 50 byte dan 6 bit data yang akan disembunyikan, maka byte yang diganti bit LSB-nya dipilih secara acak, misalkan byte nomor 36, 5, 21, 10, 18, 49.

Bilangan acak dapat dibangkitkan dengan algoritma *pseudorandom number generator*. *Pseudorandom number generator* menggunakan kunci rahasia untuk membangkitkan posisi pixel yang akan digunakan untuk menyembunyikan bit-bit. *Pseudorandom number generator* dibangun dalam sejumlah cara, salah satunya dengan menggunakan algoritma kriptografi berbasis blok (*block cipher*). Tujuan dari enkripsi adalah menghasilkan sekumpulan bilangan acak yang sama untuk setiap kunci enkripsi yang sama. Bilangan acak dihasilkan dengan cara memilih bit-bit dari sebuah blok data hasil enkripsi.

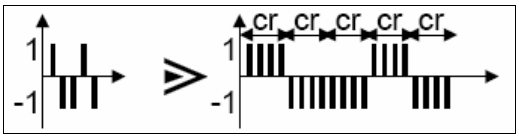
Ukuran data yang akan disembunyikan bergantung pada ukuran citra penampung. Pada citra 24-bit yang berukuran 256 ´ 256 pixel terdapat 65536 pixel, setiap pixel berukuran 3 byte (komponen RGB), berarti seluruhnya ada 65536 ´ 3 = 196608 byte. Karena setiap byte hanya bisa menyembunyikan satu bit di LSB-nya, maka ukuran data yang akan disembunyikan di dalam citra maksimum.

196608/8 = 24576 byte

Ukuran data ini harus dikurangi dengan panjang nama berkas, karena penyembunyian data rahasia tidak hanya menyembunyikan isi data tersebut, tetapi juga nama berkasnya. Semakin besar data disembunyikan di dalam citra, semakin besar pula kemungkinan data tersebut rusak akibat manipulasi pada citra penampung.

5. Metode *Spread Spectrum*

Metode *spread spectrum* dalam steganografi diilhami dari skema komunikasi *spread spectrum*, yang mentransmisikan sebuah sinyal pita sempit ke dalam sebuah kanal pita lebar dengan penyebaran frekuensi [FLI97]. Kegunaan dari penyebaran ini adalah untuk menambah redundansi dan bit-bit data sehingga diharapkan memiliki tingkat *robustness* yang tinggi. Adapun besaran redundansi ditentukan oleh faktor pengali yang dapat ditentukan oleh pengguna. Contoh dari penyebaran bit-bit informasi dapat dilihat pada Gambar 5-1. Faktor pengali dilambangkan dengan *cr* yang bernilai skalar. Panjang bit-bit hasil penyebaran ini menjadi *cr* kali panjang bit-bit awal.

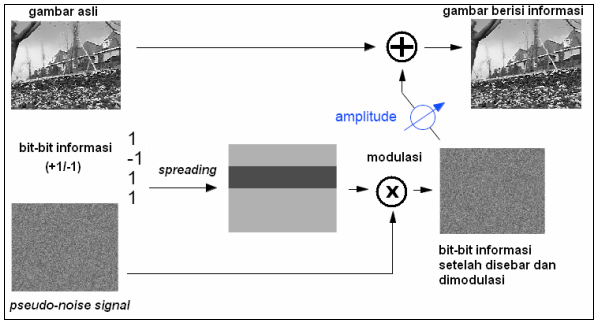


Gambar 5-1 Penyebaran bit-bit informasi

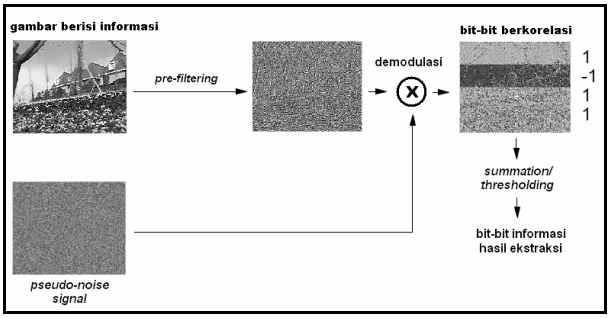
Pada proses penyembunyian data, bit-bit informasi yang telah mengalami proses spreading ini kemudian akan dimodulasi dengan *pseudo-noise signal* yang dibangkitkan secara acak berdasarkan kunci penyembunyian. Hasil dari proses modulasi ini kemudian digabungkan sebagai *noise* ke dalam sebuah berkas media dengan panjang bit penyisipan (*amplitude*) tertentu. Hasil dari penyisipan ini adalam sebuah media yang berisi informasi (*stegomessage*). Contoh dari proses penyembunyian informasi menggunakan metode *spread spectrum* dapat dilihat pada Gambar 5-2.

Untuk proses ekstraksi data, media yang telah berisi informasi rahasia tersebut disaring terlebih daluhu dengan proses pre-filtering untuk mendapatkan noise. Noise yang dihasilkan kemudian di-demodulasi dengan menggunakan *pseudo-noise signal* untuk mendapatkan bit-bit yang berkorelasi. Bit-bit yang berkorelasi dianalisa dengan perhitungan tertentu untuk menghasikan bit-bit informasi yang sesungguhnya. Analisa terhadap bit-bit yang berkorelasi yang masih dalam bertuk tersebar ini, diperlukan untuk menentukan

bit-bit informasi. Caranya dengan menghitung rata-rata setiap segmen berukuran sesuai dengan faktor pengali (*summation*) dan menentukan bit sebenarnya dari suatu segmen yang tersebar (*thresholding*). Contoh proses ekstraksi dapat dilihat pada Gambar 5-3.



Gambar 5-2. Penyembunyian informasi dengan metode spread spectrum [FLI97].



Gambar 5-3. Ekstraksi informasi dengan metode spread spectrum [FLI97].

6. Pembangkitan Bilangan *Pseudorandom*

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.

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 :

$$Z_i = (aZ_{i-1} + c) \pmod m$$

Dalam hal ini :

Z_i : bilangan bulat ke-i
a : bilangan pengali
c : bilangan penambah
m : modulus
Z_0 : nilai awal berupa bilangan bulat tak negatif

Dengan demikian nilai Z_i terdefinisi pada :

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

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.

7. Berkas 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.

Format ini didesain secara generik 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.

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.

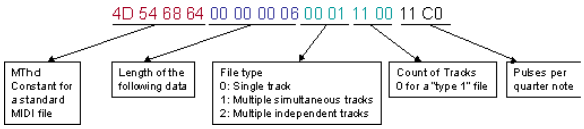
Semua *chunk header* dari sebuah file MIDI mempunyai ID “MThd” dengan ukuran *chunk* 6. Apabila dituliskan secara heksadesimal, sebuah *header* berkas MIDI menjadi sebagai berikut :

4D 54 68 64 00 00 00 06

Adapun 6 byte data *header* ini terdiri atas :

- 2 byte berisi tipe format. Ada 3 tipe format, yaitu :
  - tipe 0, berkas terdiri atas satu track tunggal berisi data MIDI yang mungkin pada semua (16) *channel* MIDI.
  - tipe 1, berkas terdiri atas satu atau lebih *track* secara simultan.
  - tipe 2, berkas terdiri atas satu atau lebih *track* secara independen dan sekuensial.
- 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.
- 2 byte terakhir mengindikasikan jumlah resolusi *Pulse per Quarter Note* (PPQN) atau dalam istilah musik berarti menentukan jumlah birama/ketukannya (*Division*)

Contoh dari sebuah *chunk* MThd dijelaskan pada gambar 7-1.

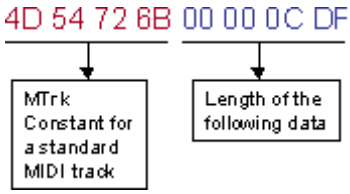


Gambar 7-1. Format sebuah chunk Mthd pada berkas MIDI

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 7-2.



Gambar 7-2. Format sebuah chunk Mtrk pada berkas MIDI.

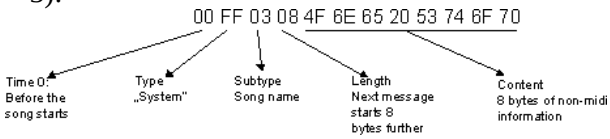
Panjang *track* pada *header* MTrk menandakan jumlah byte dalam *track* hingga dijumpai *track* selanjutnya.

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 7-1 Beberapa tipe event MIDI yang umum terdapat pada berkas MIDI

Tipe	Nama event	Data	Keterangan
80	Note Off	2 Bytes (*Note, Velocity*)	Stop memainkan suatu kunci not
90	Note On	2 Bytes (*Note, Velocity*)	Start memainkan suatu kunci not
A0	After Touch	2 Bytes (*Note, Pressure*)	Perubahan intensitas penekanan terhadap sebuah not.
B0	Control Change	2 Bytes (*Control, Value*)	Perubahan spesifik terhadap setting suatu device.
C0	Program change	1 Byte (*Program Number*)	Pemilihan Program MIDI (instrument musik)
D0	Channel Pressure	1 Byte (*Pressure*)	*After Touch* untuk seluruh channel
E0	Pitch Wheel	2 Bytes (*combined to a 14-bit value*)	Setting *pitch wheel*.
F0	System Exclusive	*all Bytes 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 7-3):

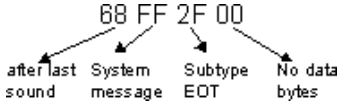


Gambar 7-3. Format event System Exclusive

Adapun contoh untuk event Control Change, Program Change, serta Note On dijelaskan pada gambar 7-4.

Gambar 7-4. Format event Control Change, Program Change, dan Note On.

Akhir dari suatu track ditandai dengan event End of Track (gambar 7-5).



Gambar 7-5. Format event End of Track.

8. Perbandingan 2 Berkas Audio

Perbandingan antara dua buah berkas audio dibutuhkan untuk mengetahui tingkat perbedaan antara berkas audio asli dangan berkas audio yang telah mengalami suatu proses perubahan (kompresi atau modifikasi). Untuk itu diperlukan suatu perhitungan untuk menghitung tingkat perbedaan yang ditimbulkan antara sebelum dan setelah proses rekonstruksi.

Ada beberapa metode untuk menghitung metrik perbandingan antara dua buah sinyal, diantaranya adalah *peak signal-to-noise ratio* (PSNR). PSNR sangat umum digunakan sebagai ukuran kualitas dalam kompresi citra, suara, dan video digital. PSNR sangat mudah didefinisikan dengan cara menghitung *mean squared error* (MSE) terlebih dahulu, dimana untuk n amplitudo (untuk berkas suara) dua buah sinyal suara I dan K adalah sebagai berikut :

$$MSE = \frac{1}{n}\sum_{i=0}^{n-1} \|I(i) - K(i)\|^2$$

Dengan demikian maka PSNR didefinisikan sebagai :

$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) = 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right)$$

MAX_i merupakan nilai maksimal dari amplitudo yang diijinkan dalam berkas audio. Pada berkas audio MIDI, parameter amplitudo ini dinyatakan dalam atribut *velocity* setiap event *note on*. Nilai atribut *velocity* ini berkisar antara 0 hingga 127 (7 bit). Dengan demikian, maka batas maksimal MAX_i adalah 127.

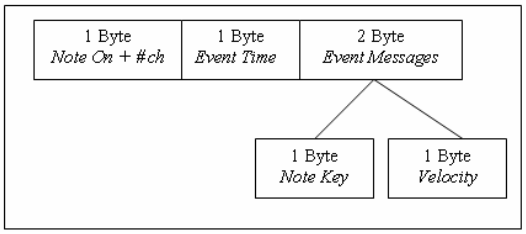
9. Usulan Metode Steganografi

Teknik yang akan digunakan untuk implementasi steganografi pada berkas MIDI ini adalah teknik *spread spectrum* terhadap pesan yang akan disembunyikan. Adapun metode yang digunakan adalah DSSS(*Direct-Sequence Spread Spectrum*). Metode ini menyebarkan sinyal dengan melipatgandakannya, untuk selanjutnya dimodulasi dengan *chip* dari kunci.

Adapun besaran skalar faktor pengali pelipatgandaan ini ditentukan secara tetap oleh sebuah konstanta. *Pseudo-noise signal* digunakan untuk

mengacak pesan. *Pseudo-noise signal* ini dihasilkan dari algoritma pembangkitan bilangan *pseudorandom* LCG dengan bibit pembangkitan dihasilkan berdasarkan kunci yang diberikan pengguna.

Adapun metode penyisipan pesan ke dalam berkas MIDI mengadopsi metode modifikasi LSB terhadap event MIDI. 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 9-1.



Gambar 9-1 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. Meskipun blok untuk *velocity* berkapasitas 1 byte (8 bit), namun implementasi untuk *velocity* ini hanya 7 bit saja. Nilainya bervariasi pada selang 0 hingga 127.

Metode modifikasi LSB untuk lokasi penyisipan pesan yang akan diusulkan adalah pada 1 hingga 7 bit terakhir atribut velocity dari event *note on* ini. Asumsinya adalah *velocity* dari sebuah event ini identik dengan amplitudo pada sampel berkas suara biasa. Atribut *velocity* melambangkan kecepatan pukulan terhadap suatu kunci not. Contohnya apabila suatu kunci not piano ditekan dengan kecepatan tekanan tertentu. Semakin besar nilainya, maka semakin keras pula amplitudo suara yang akan dihasilkan. Dengan sedikit modifikasi pada nilai dari *velocity* ini, maka diharapkan tidak terlalu mengubah besarnya volume dari suatu kunci not, sehingga indra pendengaran manusia biasa tidak dapat membedakan perubahan volume suara kunci not yang telah terjadi.

Metode steganografi ini membutuhkan 3 data input untuk memulai proses, yaitu berkas MIDI, pesan rahasia, serta kunci. Ketiga data input ini dibutuhkan untuk input pada metode pembangkitan *pseudonoise*, metode *spread spectrum* untuk metode penyisipan serta metode ekstraksi.

10. Deskripsi Metode Pembangkitan *Pseudonoise*

Noise dibutuhkan untuk mengacak pesan rahasia yang akan disembunyikan dalam berkas media. *Noise* yang sama dibutuhkan untuk mengembalikan pesan rahasia semula yang telah disisipkan dalam media. Untuk itu dibutuhkan pembangkitan *noise* dengan bibit pembangkitan yang ditentukan berdasarkan kunci masukan pengguna. *Noise* yang dihasilkan tidak betul-betul bilangan acak, karena ada nilai awal pembangkitannya. Oleh karenanya *noise* yang dihasilkan dari proses pembangkitan ini disebut *pseudonoise*. Adapun algoritma yang digunakan dalam pembangkitan *pseudonoise* ini adalah algoritma LCG.

Bibit awal pembangkitan ditentukan berdasarkan perhitungan dari kunci masukan dari pengguna. Kunci yang dimasukan oleh pengguna akan ditransformasi dengan fungsi tertentu ke dalam format data kunci yang berukuran tetap. Adapun fungsi yang digunakan adalah fungsi pelipatan data. Contoh, bila kunci masukan pengguna adalah sebagai berikut :

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

Bila ukuran bibit pembangkitan yang ditetapkan adalah 16 bit, maka perhitungannya menggunakan fungsi XOR (*Exclusive OR*) adalah sebagai berikut :

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

Apabila ukuran kunci lebih panjang lagi, maka dilakukan kembali fungsi yang sama terhadap sisa dari panjang kunci yang belum mengalami operasi dengan hasil dari 2 buah blok 16 bit sebelumnya.

Hasil dari transformasi kunci masukan pengguna kemudian dijadikan

sebagai bibit pembangkitan *pseudonoise* (Z_0) dalam algoritma LCG (dapat dilihat pada persamaan 2.1). Contoh apabila parameternya adalah sebagai berikut :

Z_0 = ’1 0 1 0 1 0 0 0 0 0 0 1 0 1 1 1’
a = ’0 1 1 1 0 1 0 1 0 0 0 1 1 0 0 0’
c = ’1 1 0 0 1 0 1 0 1 1 1 0 0 0 1 1’
m = ’1 0 0 0 0 0 1 1 0 1 1 1 0 1 0 1’

Maka Z_1 yang dihasilkan adalah :

Z_1 = ’0 1 1 1 0 1 0 0 0 0 0 0 1 0 0 1’

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

Z_2 = ’0 0 0 0 1 1 0 1 1 1 1 1 1 0 0 0’

Z_3 = ’0 1 0 1 1 0 1 1 0 1 1 1 1 1 1 0’

Z_4 = ’0 1 0 1 0 0 0 1 1 1 1 0 1 1 0 1’

Z_4 = .. .

Pseudonoise signal yang dihasilkan adalah penggabungan dari *pseudonoise* yang dihasilkan (Z_1, Z_2 ,Z_3 ,Z_4 , ... dst) , yaitu sebagai berikut :

’0 1 1 1 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0
0 1 1 0 1 1 1 1 1 0 0 0 0 0 1 0 1 1 0 1
1 0 1 1 1 1 1 0 0 0 1 0 1 0 0 0 1 1 1 1
0 1 1 0 1 ...’

Panjang *pseudonoise* bersifat dinamis disesuaikan dengan panjang pesan yang akan dimodulasi. Artinya jika seluruh bit Z_1 telah digunakan untuk modulasi pesan dan masih ada sisa panjang pesan yang belum dimodulasi, maka akan dibangkitkan Z_2 dan seterusnya.

11. Deskripsi Metode Penyisipan dengan *Spread Spectrum*

Proses pertama terhadap pesan rahasia dalam metode *spread spectrum* adalah dengan melakukan proses *spreading*. Misalkan suatu segmen pesan rahasia adalah ’1 0 1 1 0 0 1 0 ’, maka setelah proses *spreading* dengan besaran skalar pengalinya 4, akan menghasilkan segmen baru yaitu :

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

Terlihat bahwa setiap bit dalam segmen pesan akan mengalami penggandaan bit sebanyak 4 kali dan ukuran segmen pesan menjadi lebih panjang 4 kali ukuran segman semula.

Proses selanjutnya terhadap pesan ini adalah dengan proses modulasi, yaitu mengacaknya dengan suatu *pseudonoise signal* yang dibangkitkan menggunakan algoritma LCG. Pembangkitan *pseudonoise signal* ini menggunakan bilangan *pseudonumber* dari algoritma LCG dengan bibit pembangkitan yang diambil dari variabel input kunci. Misalkan dari hasil pembangkitan, diperoleh *pseud-noise signal* sebagai berikut :

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

Selanjutnya segmen pesan akan dimodulasi dengan *pseudo-noise signal* menggunakan fungsi XOR (*Exclusive OR*).

Segmen pesan :
’ 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0
0 0 0 0 0 1 1 1 1 0 0 0 0 ’

Pseudo-noise :
’ 0 1 1 1 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0
0 1 1 0 1 1 1 1 1 0 0 0 0 ’

Hasil :
’ 1 0 0 0 0 1 0 0 1 1 1 1 0 1 1 0 0 0 0
0 1 1 0 1 0 0 0 0 0 0 0 0 ’

Hasil dari proses modulasi inilah yang kemudian akan disisipkan ke dalam bit-bit LSB dari atribut *velocity* event *note-on* pada berkas MIDI.

Metode penyisipan yang dipilih adalah modifikasi LSB terhadap *atribut velocity* event-event *note-on* pada berkas MIDI. Proses selanjutnya adalah dengan mem-*parsing* event-event *note-on* yang terdapat dalam berkas MIDI, dan diurutkan berdasarkan posisi kemunculannya dalam berkas MIDI tersebut (penempatan event-event dalam berkas MIDI tidak selalu terurut berdasarkan fungsi waktu, tetapi bisa ditempatkan dalam urutan acak).

Event-event *note-on* yang telah didapatkan kemudian akan dipilih hanya atribut *velocity*-nya saja yang akan dimodifikasi bit-bit LSB-nya. Jumlah event *note-on* yang dibutuhkan untuk menampung pesan adalah sejumlah bit-bit dalam pesan tersebut. Misalkan untuk menyisipkan suatu segmen pesan hasil dari modulasi sebesar 4 byte dengan

modifikasi 1 bit LSB, maka dibutuhkan 32 event *note-on* untuk menampungnya.

Contoh penyisipan 4 bit pertama dari segmen pesan ’ 1 0 0 0 ’ dengan 4 buah event *note-on* dengan urutan atribut *velocity* sebagai berikut :

0 1 1 0 1 1 1 0 0 0 1 0 0 0 0 1 1
0 1 0 0 0 0 1 0 0 1 1 0 1 1 0 1

Maka dengan operasi penggantian bit terakhir dengan 4 bit segmen pesan secara berurutan menjadi sebagai berikut :

Velocity :

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

Pesan :

1 0 0 0

Hasil :

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

Hasil dari perhitungan ini kemudian ditulis ke dalam berkas MIDI yang baru. Dengan sedikit modifikasi ini, maka efek dari perubahan *velocity* yang terjadi tidak terlalu berpengaruh terhadap amplitudo dari suara yang dihasilkan *sequence*r berkas MIDI.

12. Deskripsi Metode Ekstraksi dengan *Spread Spectrum*

Proses dalam metode ekstraksi adalah kebalikan dari proses dalam metode penyisipan. Pertama-tama berkas MIDI yang berisi pesan, akan di-*parsing* untuk menghasilkan event-event *note-on* untuk kemudian disaring atribut *velocity*-nya. Contoh bila terdapat 4 buah event *note-on* dengan urutan atribut *velocity* sebagai berikut :

0 1 1 0 1 1 1 1 0 0 1 0 0 0 1 0
0 1 0 0 0 0 1 0 0 1 1 0 1 1 0 0

maka hasil penyaringan bit-bit terakhirnya adalah ’ 1 0 0 0 ’. Setiap bit terakhir atribut *velocity* ini akan di-demodulasi dengan *pseudo-noise signal* yang dihasilkan dengan algoritma LCG yang sama dengan metode penyembunyian. Proses demodulasi terhadap bit-bit LSB atribut *velocity* dengan *pseudo-noise* ini menggunakan fungsi XOR. Hasilnya adalah pesan dalam bentuk yang tersebar dengan faktor pengali tertentu. Contoh bila hasil penyaringan bit-bit terakhir dari nilai *velocity* event-event *note on* seterusnya adalah :

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

maka proses demodulasi menggunakan *pseudo-noise* yang sama dengan proses penyisipan adalah sebagai berikut :

Hasil penyaringan :

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

Pseudo-noise :

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

Hasil demodulasi :

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

Untuk dapat menghasilkan pesan yang sesungguhnya, maka dibutuhkan proses perhitungan terhadap besaran faktor pengali ini. Faktor pengali yang digunakan sama dengan faktor pengali pada pada proses *spreading*. Selanjutnya dilakukan proses *de-spreading* dengan mennyusutkan segmen-segmen bit yang sama menjadi bit-bit yang lebih sederhana. Hasil dari proses *de-spreading* inilah yang dianggap sebagai pesan rahasia sesungguhnya. Contoh bila suatu segmen pesan yang masih tersebar adalah sebagai berikut :

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

Maka terlihat ada suatu pola pengulangan dari bit-bit tersebut, yaitu 4 kali bit 1, 4 kali bit 0, 8 kali bit 1, 8 kali bit 0, 4 kali bit 1, dan 4 kali bit 0. Dengan demikian faktor pengalinya dapat ditentukan besarnya adalah 4 dan proses penyusutan bit (*despreading*) segmen tersebut menjadi sebagai berikut :

Sebelum :

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

Sesudah :

’ 1 0 1 1 0 0 1 0 ’

Sehingga diperoleh hasil akhir berupa segmen pesan rahasia, yaitu : ’ 1 0 1 1 0 0 1 0 ’. Hasil ekstraksi ini sama dengan segmen pesan rahasia yang disembunyikan pada metode penyisipan sebelumnya.

13. Deskripsi Umum Perangkat Lunak

Aplikasi yang dibangun berkaitan dengan steganografi pada berkas MIDI ini, memiliki spesifikasi sebagai berikut :

a) Menerima masukan berkas MIDI asli maupun berkas MIDI yang telah disisipi data.
b) Mampu menyisipkan data ke dalam berkas MIDI.
c) Menyimpan berkas MIDI yang sudah disisipi data.
d) Mampu mengekstraksi berkas MIDI yang telah disisipi data untuk mendapatkan berkas data yang valid.
e) Mampu memutar berkas MIDI, agar pengguna dapat mengamati perubahan yang terjadi sebelum dan setelah berkas MIDI disisipi data.

Perangkat lunak ini diberi nama MIDSteM (*MIDI's Steganography Machine*). Secara garis besar, perangkat lunak MIDSteM 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 file 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 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.

Perangkat lunak MIDSteM dikembangkan pada komputer dengan sistem operasi Microsoft Windows XP Professional. Bahasa yang digunakan dalam implementasi perangkat lunak MIDSteM adalah bahasa C# dan kompilator yang digunakan adalah Microsoft Visual Studio .net 2003 dalam *framework* Microsoft .net versi 1.1.

14. Implementasi Perangkat Lunak

Rancangan layar antarmuka perangkat lunak MidSteM diimplementasikan dengan menggunakan *form* pada lingkungan implementasi Microsoft Visual Studio.net 2003. Terdapat 2 antarmuka utama, yaitu antarmuka proses *hiding message* (Gambar 14-1) dan antarmuka proses *extract message* (Gambar 14-2).

Pada antarmuka proses *hiding*, pengguna dapat memasukkan nama berkas MIDI dengan menekan tombol "Browse..." yang sesuai dengan baris Berkas MIDI. Pengguna dapat mendengarkan berkas MIDI ini dengan menekan tombol "Play MIDI". Selanjutnya pengguna diminta untuk memasukkan kunci pada baris kunci secara langsung atau dapat pula dengan membuka berkas yang berisi kunci. Kunci yang dimasukkan tidak akan ditampilkan, tetapi ditampilkan dalam bentuk karakter bintang/asteriks (*). Pada baris pesan, pengguna dapat memasukkan pesan rahasia baik secara langsung, maupun dengan me-*load* nama berkas yang akan dijadikan pesan rahasia dengan menekan tombol "Load". Bila ketiga jenis masukan ini telah terisi, maka pengguna dapat menekan tombol "Hide Message".

Gambar 14-1 Antarmuka proses Hiding

Pada antarmuka proses ekstraksi, pengguna dapat memasukkan nama berkas MIDI dengan menekan tombol "Browse..." yang sesuai dengan baris Berkas MIDI. Pengguna dapat mendengarkan berkas MIDI ini dengan

menekan tombol ”Play MIDI”. Selanjutnya pengguna diminta untuk memasukkan kunci pada baris kunci secara langsung atau dapat pula dengan membuka berkas yang berisi kunci dengan menekan tombol ”Browse....” Kunci yang dimasukkan tidak akan ditampilkan, tetapi ditampilkan dalam bentuk karakter bintang/asteriks (*). Selanjutnya pengguna dapat menekan tombol ”Extract Message” untuk memulai proses ekstraksi pesan dalam berkas MIDI. Bila terdapat pesan rahasia, maka hasilnya akan ditampilkan dalam *Text Box* pada baris Pesan.

Gambar 14-2 Antarmuka proses Extract

Gambar 14-3 Antarmuka Fitur Bantuan

Gambar 14-4 Antarmuka Fitur Perihal

Untuk implementasi tambahan lainnya, tersedia juga fitur bantuan (*help*) serta fitur perihal (*about*). Fitur bantuan berfungsi untuk memberikan petunjuk penggunaan aplikasi MIDSteM. Implementasinya dapat dilihat pada Gambar 14-3. Sedangkan uintuk fitur perihal berfungsi untuk menampilkan perihal mengenai pengembang aplikasi MIDSteM ini. Fitur ini dapat dilihat pada Gambar 14-4.

15. Pengujian Perangkat Lunak

Pengujian perangkat lunak MIDSteM dilakukan pada lingkungan sebagai berikut :

1) Prosesor : Intel Pentium IV 2.4 GHz
2) Memori : DDR 1 GB PC 2700
3) Kartu Suara : Creative Vibra 128
4) Hard disk : Maxtor 80 GB

Adapun aspek yang dijadikan sebagai bahan pengujian adalah :

a) Kesesuaian data, yaitu persentase jumlah bit-bit kode yang berhasil diekstraksi dengan benar dari sinyal suara berkode terhadap jumlah bit-bit kode yang disisipkan ke dalam sinyal suara.
b) Nilai PSNR dari hasil penyisipan data, yaitu pengujian nilai PSNR antara berkas MIDI yang belum dilakukan proses penyisipan data, dengan berkas MIDI yang telah dilakukan penyisipan data.

Parameter nilai PSNR ini diambil dari nilai atribut *velocity* pada event *note on* yang identik terhadap amplitudo setiap kunci not yang dimainkan.

Pengujian perangkat lunak terdiri atas 2 bagian, yaitu pengujian kesesuaian data dan pengujian kualitas berkas media. Hasil dari pengujian kesesuaian data dapat dilihat pada tabel 15-1 dan 15-2, dan hasil pengujian kualitas berkas media dapat dilihat pada tabel 15-3 dan 15-4.

Proses penyembunyian data dapat dikatakan berhasil apabila ukuran data yang akan disembunyikan lebih kecil atau sama dengan kapasitas data. Apabila ukuran data lebih besar dari kapasitas data yang dapat ditampung, maka aplikasi tidak akan melanjutkan proses penyisipan.

Kapasitas data dipengaruhi oleh jumlah event *note on*, nilai *cr*, serta panjang LSB yang dijadikan sebagai bit *carrier*. Semakin besar jumlah event *note on*, maka semakin besar pula kapasitas yang akan dihasilkan. Semakin besar nilai *cr*, semakin kecil kapasitas data yang dapat ditampung. Semakin besar panjang LSB yang digunakan, maka semakin besar pula kapasitas yang akan dihasilkan. Jadi kapasitas maksimum data yang dapat ditampung berbanding lurus dengan jumlah event *note on* dan panjang LSB, serta berbanding terbalik dengan nilai *cr*.

Tabel 15-1 Pengujian penyisipan data

No.	Panjang berkas MIDI (byte)	Panjang berkas pesan (byte)	cr	LSB	Status Penyisipan
1	80091	58	6	1	Berhasil
2	80091	99	6	1	Berhasil
3	80091	173	6	1	Berhasil
4	80091	237	6	1	Gagal
5	80091	237	4	1	Berhasil
6	80091	237	6	2	Berhasil
7	81687	58	6	1	Berhasil
8	81687	99	6	1	Berhasil
9	81687	173	6	1	Berhasil
10	81687	237	6	1	Gagal
11	81687	237	6	2	Berhasil
12	81687	237	5	1	Berhasil
13	81687	403	6	2	Berhasil
14	81687	982	6	2	Gagal
15	81687	982	5	3	Gagal
16	81687	982	4	4	Berhasil
17	60594	99	6	1	Berhasil
18	80369	99	6	1	Berhasil
19	101872	99	6	1	Berhasil
20	60369	99	6	1	Berhasil

Dari data pengujian hasil proses penyembunyian, terlihat bahwa dari 20 kali pengujian, ada 4 pengujian yang tidak berhasil, yaitu pengujian nomor 4,10,14,15. Sedangkan sisanya berhasil. Hal ini dipengaruhi oleh besarnya ukuran pesan yang akan disembunyikan dibandingkan dengan kapasitas data yang dapat ditampung. Agar dapat tertampung, maka dapat dilakukan perubahan terhadap nilai parameter cr dan panjang LSB. Seperti untuk kasus nomor 4, dengan menurunkan nilai cr dari 6 menjadi 4 (pengujian nomor 5), data dapat ditampung ke dalam berkas. Untuk kasus nomor 10, dengan menaikkan jumlah bit LSB yang digunakan, maka data dapat tertampung. Hal yang sama juga berlaku untuk kasus nomor 14 dan 15.

Hasil dari proses penyembunyian ini (berkas MIDI yang berisi data), belum dapat dikatakan sesuai dengan spesifikasi apabila :

1) Berkas hasil tidak dapat dimainkan.
2) Data yang telah disembunyikan tidak berhasil diekstraksi.

Oleh karenanya, dibutuhkan pengujian terhadap proses ekstraksi dan pengujian terhadap berkas MIDI yang telah berisi data.

Dari tabel 15-2 terbukti bahwa semua berkas hasil penyembunyian data, dapat dimainkan. Untuk menentukan apakah perangkat lunak ini telah memenuhi spesifikasi, maka harus dapat dibuktikan bahwa data dapat diekstaksi. Agar data dapat diekstraksi, maka syaratnya adalah kunci yang digunakan harus sesuai dengan kunci yang digunakan saat proses penyembunyian data.

Berdasarkan tabel hasil penelitian tersebut, terlihat bahwa semua data dapat diekstraksi apabila kunci yang digunakan untuk ekstraksi sama dengan kunci

penyisipan. Kegagalan ekstraksi data pada penelitian ini terjadi karena kunci yang salah (kasus no. 13-20). Data yang dihasilkan dari proses ekstraksi tidak ada sama sekali, atau menghasilkan karakter-karakter aneh yang tidak dapat dibaca.

Tabel 15-2 Pengujian ekstraksi data

No.	Panjang berkas MIDI (byte)	Panjang berkas pesan (byte)	Dapat dimain-kan	Status Ekstraksi	Kese-suaian data
1	80091	58	ya	Berhasil	ya
2	80091	99	ya	Berhasil	ya
3	80091	173	ya	Berhasil	ya
4	*Gagal*	*Gagal*	*Gagal*	*Gagal*	*Gagal*
5	80091	237	ya	Berhasil	ya
6	80091	237	ya	Berhasil	ya
7	81687	58	ya	Berhasil	ya
8	81687	99	ya	Berhasil	ya
9	81687	173	ya	Berhasil	ya
10	*Gagal*	*Gagal*	*Gagal*	*Gagal*	*Gagal*
11	81687	237	ya	Berhasil	ya
12	81687	237	ya	Berhasil	ya
13	81687	403	ya	Berhasil	ya
14	*Gagal*	*Gagal*	*Gagal*	*Gagal*	*Gagal*
15	*Gagal*	*Gagal*	*Gagal*	*Gagal*	*Gagal*
16	81687	982	ya	Berhasil	ya
17	60594	99	ya	Berhasil	ya
18	80369	99	ya	Berhasil	ya
19	101872	99	ya	Berhasil	ya
20	60369	99	ya	Berhasil	ya

Berdasarkan hasil pengujian kesesuaian data, dapat disimpulkan bahwa perangkat lunak yang diimplementasikan telah sesuai dengan kebutuhan pada bagian analisis dan perancangan. Hal ini dibuktikan dengan keberhasilan MIDSteM untuk menyembunyikan pesan ke dalam berkas MIDI dan melakukan ekstraksi pesan terhadap berkas yang telah disisipi data.

Semua skenario pengujian yang benar, berhasil dilakukan sesuai dengan spesifikasi perangkat lunak. Sedangkan semua skenario pengujian menggunakan masukan kunci yang salah, tidak memberikan data yang sesuai dengan harapan.

Tabel 15-3 Pengaruh panjang LSB terhadap nilai PSNR.

No. Pengujian	Panjang LSB	PSNR (dB)
1	1	48.03
2	2	42.7
3	3	41.93
4	4	35.81
5	5	28.63
6	6	26.71
7	7	19.47
8	1	45.32
9	2	40.82
10	3	36.69
11	4	31.36
12	5	26.41
13	6	20.97
14	7	14.69

Tabel 15-4 Pengaruh panjang data terhadap nilai PSNR.

No. Pengujian	Panjang data (bytes)	PSNR (dB)
15	58	48.03
16	99	42.7
17	173	41.93
18	237	35.81
19	403	28.63
20	982	26.71
21	58	45.32
22	99	40.82
23	173	36.69
24	237	31.36
25	403	26.41
26	982	20.97

Metrik pengukuran kualitas media yang digunakan dalam pengujian ini adalah PSNR. PSNR dihitung berdasarkan perbedaan nilai atribut velocity yang berkorelasi antara berkas MIDI hasil penyisipan dan berkas MIDI aslinya.

Pengujian kualitas beras media dikatakan berhasil apabila kualitas berkas yang dihasilkan tidak mengalami distorsi yang besar dibandingkan dengan berkas aslinya. Dengan kata lain, suara dari berkas yang dihasilkan sulit untuk dibedakan dengan dengan suara dari berkas aslinya.

Berdasarkan data hasil pengujian pada tabel 15-3, terlihat bahwa kualitas berkas yang dihasilkan sangat dipengaruhi oleh panjang LSB yang digunakan sebagai bit *carrier*, serta panjang dari pesan yang

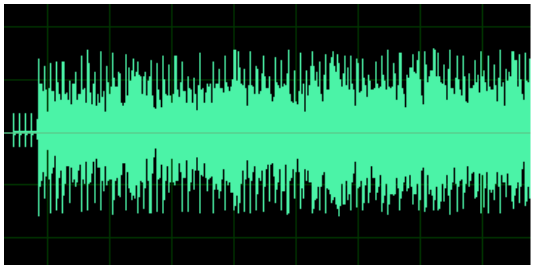
akan disisipkan. Hal ini dibuktikan dengan pengujian pada nomor 1-7 terhadap suatu berkas MIDI, serta nomor 8 – 14 terhadap berkas MIDI yang berbeda, dimana dilakukan penambahan terhadap panjang LSB. Terbukti bahwa semakin besar panjang LSB yang digunakan, maka kualitas suara yang dihasilkan akan semakin buruk. Hal ini juga ditandai dengan semakin mengecilnya nilai PSNR apabila panjang LSB semakin diperbesar. Nilainya turun secara bertahap, diikuti penurunan terhadap kualitas suara yang dihasilkan.

Panjang pesan yang disisipkan juga dapat mempengaruhi kualitas berkas hasil, apabila panjang LSB yang digunakan relatif besar. Hal ini dibuktikan pada hasil pengujian pada tabel 15-4, nomor 15 – 20 (menggunakan suatu berkas MIDI tertentu) serta nomor 21 – 26 (menggunakan suatu berkas MIDI yang berbeda). Dengan semakin besar ukuran pesan yang disisipkan, maka semakin buruk nilai PSNR-nya serta semakin buruk kualitas berkas yang dihasilkan. Berdasarkan hasil pengamatan, pengujian dengan nilai LSB yang besar menggunakan ukuran pesan yang kecil akan berpengaruh terhadap kualitas dari suatu *track* yang berisi not dari suatu instumen. Semakin besar ukuran pesan yang disisipkan, semakin banyak *track* dan semakin luas cakupan not instrumen yang mengalami distorsi.

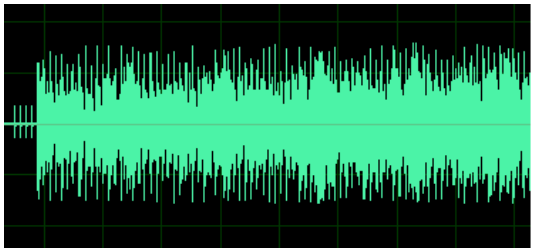
Berdasarkan hasil pengujian kualitas berkas media yang telah disisipi data, dapat disimpulkan bahwa kualitas media sangat dipengaruhi oleh panjang bit *velocity* yang digunakan sebagai *carrier*. Semakin panjang bit yang digunakan, semakin buruk kualitas media yang akan dihasilkan.

Panjang bit yang ideal untuk digunakan berkisar antara 1 hingga 4 bit. Lebih dari 4 bit, maka kemungkinan kualitas suara yang dihasilkan akan sangat berkurang. Penambahan bit LSB sebaiknya dilakukan hanya apabila kapasitas data yang dapat ditampung kurang mencukupi. Konsekuensinya tentu semakin berkurangnya kualitas media hasil penyisipan.

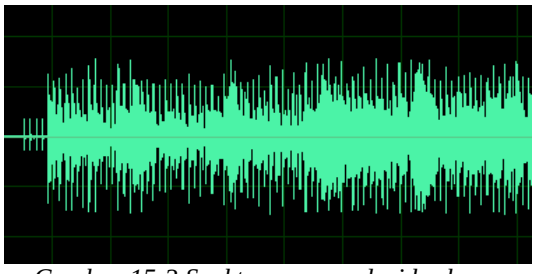
Efek dari berkurangnya kualitas berkas media diantaranya adalah volume yang melemah, volume suara yang tidak teratur untuk beberapa instrumen hingga bunyi seluruh instrumen terdengar sangat kacau. Sehingga dapat disimpulkan bahwa panjang bit LSB yang layak digunakan seharusnya tidak lebih dari 4 bit, agar hasilnya masih layak untuk didengarkan (tidak terlalu banyak distorsi). Contoh spektrum suara yang dihasilkan dari berkas MIDI sebelum proses penyembunyian data dapat dilihat pada Gambar 15-1. dan setelah proses penyembunyian data dengan panjang LSB yang bervariasi dapat dilihat pada Gambar 15-2 dan 15-3.



Gambar 15-1 Spektrum suara dari berkas yang belum disisipi data



Gambar 15-2 Spektrum suara dari berkas yang telah disisipi data (LSB =4)



Gambar 15-3 Spektrum suara dari berkas yang telah disisipi data (LSB=7)

Dari hasil pengamatan terlihat bahwa berkas yang dihasilkan dari proses penyembunyian data dengan LSB yang besar (Gambar 15-3) mengakibatkan spektrum suara yang dihasilkan mengalami penurunan yang cukup signifikan dibandingkan dengan spektrum

suara sebelumnya (Gambar 15-1). Sedangkan dengan parameter panjang LSB sebesar 4 (Gambar 15-2)., spektrum suara yang dihasilkan tidak berbeda jauh dengan spektrum suara sebelumnya (Gambar 15-1)

16. Kesimpulan dan Saran

Berdasarkan hasil analisis dan kajian yang telah dilakukan, ada beberapa kesimpulan yang dapat diambil dari pembangunan aplikasi ini, yaitu:

1) Steganografi dapat dilakukan pada berkas MIDI, khususnya dengan memanfaatkan nilai dari atribut *velocity* pada event *note on*. Nilai dari atribut ini terbukti berpengaruh terhadap besarnya amplitudo dari suara yang dikeluarkan saat berkas MIDI diputar.
2) Kapasitas data yang dapat ditampung dalam berkas MIDI, dipengaruhi oleh jumlah event *note on* dalam berkas, besarnya faktor pengali, serta panjang LSB yang dimodifikasi.
3) Kualitas berkas MIDI hasil steganografi sangat dipengaruhi oleh panjang LSB yang dimodifikasi, semakin panjang LSB yang digunakan untuk penyisipan, semakin buruk kualitas yang dihasilkan.

Selain kesimpulan di atas, ada beberapa saran yang perlu dipertimbangkan lebih lanjut, diantaranya :

1) MIDSteM yang dikembangkan hanya memanfaatkan event note on dari berkas MIDI. Oleh karena itu perlu untuk dilakukan pengembangan dengan memanfaatkan event-event lain, sehingga diharapkan dapat menambah kapasitas data yang dapat disembunyikan.
2) MIDSteM memanfaatkan event note on yang ditulis secara sekuensial dalam berkas, yang mungkin tidak sesuai dengan urutan waktu event. Oleh karena itu perlu pengembangan agar proses penyisipan ini urutannya disesuaikan dengan urutan waktu dari setiap event.
3) MIDSteM masih dikembangkan untuk perangkat keras PC. Akan lebih praktis apabila juga dikembangkan lebih lanjut untuk dapat digunakan dalam lingkungan perangkat keras mobile seperti telepon genggam.

17. Daftar Pustaka

[AKO01] Akoff Sound Labs (2001). *Music Recognition - Wav to Midi Conversion*. http://www.akoff.com/.

[FLI97] Flikkema, Paul G. (1997). *Spread Spectrum Techniques for Wireless Communications*. IEEE Signal Processing, 14(3):26-36.

[JUA02] Juanda. (2002). *Aplikasi Watermarking untuk Data Video Digital*. Program Pascasarjana Teknik Sistem Komputer, Jurusan Teknik Elektro, Institut Teknologi Bandung.

[MOE03] Moerland, T. (2003). *Steganography and Steganalysis.*

[MUN04] Munir, Rinaldi (2004). *Diktat Kuliah IF5054 Kriptografi: Steganografi dan Watermarking.* Institut Teknologi Bandung.

[SUH04] Suharto, Edy. (2004). *Pendam : Aplikasi Penyembunyian Data di Dalam Berkas MPEG/Audio Layer III (MP3) dengan Metode Low Bit Coding*. Tugas Akhir, Program Studi Informatika, Institut Teknologi Bandung.

[SUP00] Supangkat, Suhono H., Kuspriyanto, Juanda. (2000). *Watermarking sebagai Teknik Penyembunyian Label Hak Cipta pada Data Digital*. Departemen Teknik Elektro, Institut Teknologi Bandung.