IMPLEMENTASI STEGANOGRAFI UNTUK DATA AUDIO AMR PADA MMS TELEPON SELULER MENGGUNAKAN METODE LSB

I Made Widhi Wirawan
Program Studi Teknik Informatika, Jurusan Ilmu Komputer
Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Udayana
Email : made_widhi@cs.unud.ac.id

Abstrak: Keamanan pada informasi saat dikirim melalui pengiriman pesan multimedia atau Multimedia Message Service menjadi sebuah hal yang penting untuk dilakukan. Metode steganografi menjadi salah satu pilihan dalam pengamanan untuk menghindari jika terjadinya serangan pada Multimedia Message Service Centre yang dapat mengakibatkan pesan diterima oleh orang – orang yang tidak berhak untuk mengetahui informasi. Steganografi membutuhkan dua properti yaitu wadah penampung dan data yang dirahasiakan.

Objek pada penelitian ini adalah sebuah *File* audio Adaptive Multi Rate yang akan disisipkan dengan pesan teks. Penyisipan dilakukan dengan menggunakan metode Least Significant Bit. Pada metode Least Significant Bit, 1 bit akhir pada frame akan diganti dengan 1 bit dari nilai pesan. Setelah diganti, nilai frame tersebut akan menjadi baru pada *File* audio.

Steganografi pada *File* audio Adaptive Multi Rate dapat memberikan keamanan pesan dengan memperkecil kemungkinan terbacanya pesan tersembunyi dalam audio.

Kata Kunci : Steganografi, *Least Significant Bit, Adaptive Multi Rate, Multimedia Message Service*

The Steganografi Implementation For AMR Audro Data On Cellular Phone MMS Using LSB Method

Abstract: *The security when the information sent via Multimedia Messaging, or Multimedia Message Service become important think to do. Steganographic methodes become one option in safeguards to avoid an attack on the Multimedia Message Service Centre which ca lead to a message received by people-people who arenot entitled knowing information. Steganography requires two properties of the container receptacle and confidential data.*

Objects in this study is an Adaptive Multi-Rate Audio files which will be pasted with a text message. Insertion is done by using the method of Least Significant Bit. In the Least Significant Bit method one last bit in the frame will be replaced with one bit of the message value after being replaced the frame will be new audio file.

Steganography on Adaptive Multi Rate Audio files can provide message security by minimizing the possiblility of the hidden messages in audio to be read.

Key Word : *Steganografi, Least Significant Bit, Adaptive Multi Rate, Multimedia Message Service*

I. PENDAHULUAN

Fasilitas yang sangat digemari masyarakat pengguna telepon seluler (ponsel) adalah *Short Message Service* (SMS) dan *Multimedia Message Service* (MMS). MMS merupakan fasilitas telepon seluler yang dapat digunakan untuk transmisi data berupa teks, audio, *image*, dan video sebagai media komunikasi. Dengan demikian MMS adalah *store and forward messaging service* yang memfasilitasi pengguna *mobile device* untuk melakukan pertukaran pesan multimedia. MMS biasa dikatakan sebagai bentuk *evolusi* SMS, dimana pada teknologi pengiriman pesan tersebut terdapat transmisi jenis media tambahan yang meliputi *teks*, *image*, *audio*, animasi, video klip, atau kombinasi media – media tersebut (Asti, 2006).

Proses pengiriman dilakukan pada telepon seluler yang akan melakukan pengiriman MMS dan akan diterima dan disimpan oleh MMSC, selanjutnyamelakukan pengecekan dan memberitahukan kepada telepon seluler yang kedua bahwa ada pengiriman MMS, jika telepon seluler yang dituju mendukung fasilitas MMS dan telah dilakukan penyetingan yang tepat maka MMSC akan mengirimkan pesan dari telepon seluler yang mengirimkan pesan tersebut (Asti, 2006). Celah keamanan terbesar komunikasi MMS adalah pesan yang dikirimkan akan disimpan di MMSC, sehingga apabila terjadi serangan pada MMSC, maka pesan yang dikirimkan dari satu pengguna telepon seluler ke pengguna lainnya dapat terbaca. Salah satu cara untuk menanggulangi celah tersebut adalah dengan melakukan penyisipan terhadap pesan yang dikirimkan. Pesan yang dikirim adalah pesan yang sudah disisipkan pada *File* objek sehingga tidak dapat dibuka dan dibaca, pesan ini hanya bisa dibuka jika penerima memiliki aplikasi khusus yang dirancang dengan menggunakan metode didalamnya.

Penyisipan pesan dilakukan pada *File* objek *audio* dengan *extension File* AMR (*Adaptive Multi Rate*). AMR merupakan *speech codec* yang terdiri dari *encoder* dan *decoder* yang yang memungkinkan penyedia kualitas suara yang superior dengan bitrate rendah (Suhartani & Aulia, 2003). Teknik yang digunakan untuk menyisipkan pesan didalam *File* audio adalah *Low Bit Encoding* yang mirip dengan teknik LSB (*Least Significant Bit*) yaitu dengan menyisipkan bit – bit dari pesan yang akan disembunyikan ke dalam bit media penampungan pesan tersebut (Ria, 2006).

II. METODE PENELITIAN

2.1 MMS (Multimedia Message Service)

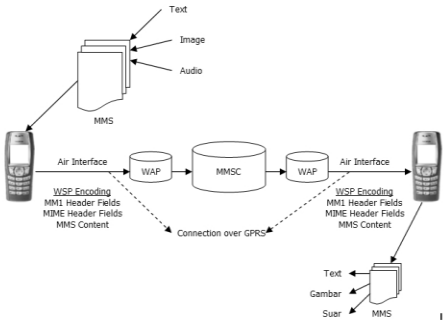
MMS (*Multimedia Message Service*) adalah *store and fordward messaging service* yang memfasilitasi pengguna mobile *device* untuk melakukan pertukaran pesan multimedia. MMS bisa dikatakan sebagai bentuk *evolusi* SMS (*Short Messaging Service*), dimana pada teknologi pengiriman pesan tersebut terdapat transmisi jenis media tambahan yang meliputi teks, *image*, audio, animasi, video, atau kombinasi media – media tersebut. MMS merupakan *open wireless standart* yang dispesifikasikan oleh forum WAP untuk 3GPP. Meskipun teknologi ini merupakan standar 3G, *operator* jaringan saat ini mengembangkan teknologi tersebut di atas jaringan GPRS dengan menggunakan WAP sebagai transport.

Secara konsep MMS dan SMS berbeda karena data-data yang dikirimkan dan proses pengirimannya pun berbeda. Perbedaan mendasar keduanya adalah banyaknya *paging* untuk pesan dan *channel* dimana kedua pesan tersebut berjalan.

SMS bekerja pada SS7 *channel*, yaitu sebuah saluran pembawa data yang bergerak lambat, sedangkan MMS beroperasi via *protocol* IP (sama dengan jalannya web akses pada internet). MMS menggunakan beberapa *platform* dalam MMSE (*Multimedia Messaging Enviroment*), termasuk MMS *relay*, MMS *message relay*, MMS message store, MMS Pengguna database dan platform lainnya. Para *operator* mengintegrasikan infrastruktur MMS *server* dan *relay* kedalam sebuah *platform* yang disebut MMS *Center*.

Sebuah pesan MMS dikirimkan menggunakan kombinasi antara Teknologi SMS dan WAP. Pada saat sebuah ponsel menerima sebuah pesan MMS, pada dasarnya ponsel tersebut menerima sebuah pemberitahuan melalui SMS yang menginformasikan bahwa ponsel tersebut mendapatkan sebuah kiriman berupa data multimedia yang berada dalam sebuah URL tertentu. Pesan tersebut terdiri atas *content header* dan URL yang harus diakses untuk mendapatkan data multimedia yang telah dikirimkan oleh seseorang tadi. Alamat URL ini secara dinamis dihasilkan oleh MMSC (*MMS Center*) disesuaikan dengan data multimedia dan pengguna layanan yang bersangkutan.

2.2 Sistem Dasar Operasi Pengiriman MMS



Gambar Proses Pengiriman MMS
(Sumber: Asti, 2006)

Secara umum, proses pengiriman dan penerimaan sebuah MMS berlangsung sebagai berikut:

1. Ponsel yang mengirimkan data akan mulai menginisialisasi koneksi data yang disediakan melalui jaringan TCP/IP melalui GPRS.

2. Ponsel tersebut akan membuat pengkodean pesan MMS dengan format pembungkusan yang didefinisikan dalam Open Mobile Alliance melalui HTTP POST ke sebuah MMSC. Pesan MMS yang telah terkode tersebut meliputi semua pesan MMS termasuk didalamnya informasi header yang berisi tujuan dari pesan tersebut. Koneksi koneksi tersebut terjadi dengan menggunakan proxy server melalui WAP.

3. MMSC selanjutnya akan menerima permintaan terhadap pesan MMS serta memvalidasi pengirim dari pesan.

4. MMSC menyimpan *content* dari pesan MMS sehingga data tersebut dapat diakses. Caranya adalah dengan menyediakan URL dinamis dalam pesan ke pengguna sebagai link.

5. MMSC selanjutnya membangkitkan sebuah pesan MMS pembitahuan yang akan dikirimkan melalui WAP Push over SMS ke penerima yang dituju. Pesan ini berisi URL yang harus diakses agar data yang dikirimkan dapat diperoleh.

6. Penerima selanjutnya akan menerima pemberitahuan MMS tersebut dan selanjutnya menginisialisasi sebuah koneksi data melalui jaringan TCP/IP menggunakan GPRS.

7. Ponsel penerima selanjutnya mengambil data yang dikirimkan melalui sebuah HTTP (atau WSP) ke MMSC. Meski secara umum tidak berbeda jauh dengan proses pada SMS, namun secara umum, GPRS menjadi faktor dominan yang mempengaruhi keberhasilan transaksi (Asti, 2006).

2.3 LSB (Least Significant Bit)

Metode LSB (*Least Significant Bit*) adalah salah satu teknik yang dipelajari dalam penyembunyian data dan watermarking digital audio. Pada metode LSB Standar, LSB encoder biasanya mengambil subset dari contoh audio utama yang dipilh dengan kunci rahasia. Operasi substitusi pada LSB dilakukan pada subset ini, nilai bit asli diganti dengan bit yang ingin disembunyikan. Proses ekstraksi dengan membaca nilai bits dari audio stego *object*. Maka dari itu, decoder memerlukan semua contoh dari stego audio yang digunakan saat penyisipan. Seiring dengan meningkatnya penggunaan LSB saat pemrograman, modifikasi terhadap kedalaman layer LSB semakin besar, kemungkinan pesan tersembunyi mampu dideteksi juga meningkat dan persepsi yang transparan terhadap stego *object* menurun. Sehingga muncul batasan untuk penggunaan tingkat kedalaman layer LSB pada contoh audio yang bisa digunakan untuk menyembunyikan data.

Maksimum kedalaman layer LSB secara rata-rata yang bisa digunakan tanpa menyebabkan distorsi yang bisa diketahui layer keempat LSB dengan 16 bits per contoh *audio sequences* digunakan. Di sisi lain, tingkat kekokohan dari *stego-object* akan bertambah seiring meningkatnya kedalaman layer LSB yang digunakan.

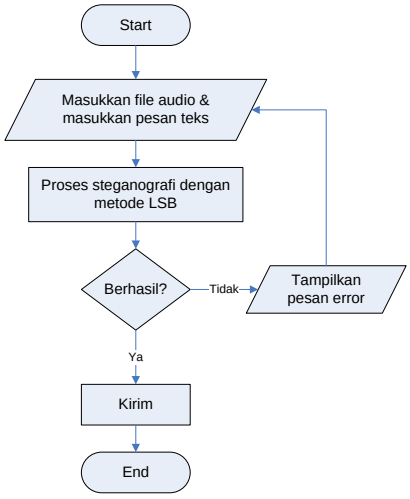
III. HASIL DAN PENELITIAN

Perancangan dan pengimplementasian aplikasi ini menggunakan metode SDLC (*System Development Life Cycle*) dengan tahapan diantaranya sebegai berikut:

3.1 Perancangan Aplikasi

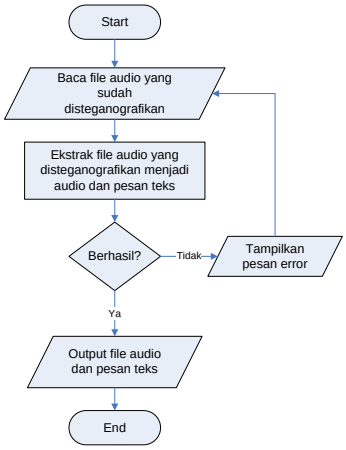
Aplikasi yang akan dirancang adalah dengan menggunakan metode LSB untuk steganografi pada *File* objek audio dengan *extension File* AMR. Sedangkan aplikasi untuk telepon seluler (*handphone*) akan dirancang dengan menggunakan bahasa pemrograman J2ME.

Flowchart untuk steganografi MMS dengan metode LSB pada saat penyisipan pesan pada *File* objek audio:



Gambar 3.1 Flowchart Alur Sistem Steganografi untuk MMS Saat Penyisipan Pesan Teks

Flowchart untuk steganografi MMS pada saat pesan yang disteganografikan diekstrak:



Gambar 3.2 Flowchart Alur Sistem Steganografi Untuk MMS Saat Pesan Diekstrak

3.2 Pengujian

Pengujian untuk aplikasi akan menggunakan metode *black-box* yang berfokus pada hasil akhir dan untuk pengujian sistem steganografi akan menggunakan metode RMS (*Root Mean Square*) untuk mengetahui nilai parameternya, semakin besar nilai yang didapat, maka semakin besar perbedaan kemiripan pada saat sebelum dan setelah pesan itu disisipkan.

3.3 Steganografi Pada MMS Menggunakan Metode LSB

Aplikasi MMS dirancang dengan menerapkan teknik steganografi dengan menggunakan metode LSB yang merupakan teknik menyembunyikan pesan teks kedalam *File*, *File* yang digunakan dalam aplikasi MMS ini adalah audio AMR karena *File* ini merupakan

File audio yang mempunyai ukuran sangat kecil daripada audio lainya. Karena AMR merupakan *speech codec* yang terdiri dari *encoder* dan *decoder* yang yang memungkinkan penyedia kualitas suara yang superior dengan bitrate rendah.

3.4 Proses Penyisipan

Pada proses penyisipan ini dilakukan pada saat pengiriman MMS. Saat pengiriman, pengguna harus memasukkan nomor telepon, membuat pesan teks, dan memasukkan *File* audio AMR. Berikut pseudocode untuk proses penyisipan adalah sebagai berikut:

Nilai Awal :
    by : larik byte dari *File* amr
    dtBy : larik byte dari text pesan
    id : indek bit data pesan diawali dengan indek 0
    ic : index byte *File* amr diawali dengan 5 karena 0 - 4 adalah byte header *File* amr
Algoritma :
Selama *ic* < panjang *by* dan *id* < panjang *dtBy* lakukan :
    ambil frame size *fs* dari *by* untuk yang dimulai dari byte *ic*;
    tambahkan *ic* dengan *fs* terhadap *ic*
    jika *ic* < panjang *by* yg berarti indek *ic* ada pada byte *File* amr maka lakukan
        ubah bit terakhir pada frame amr yaitu bit terakhir dari indek byte *ic* pada *by* sesuai bit pada *dtBy* indek ke *id*
        tambahkan *id* dengan 1

Berikut contoh Hexdump untuk *File* AMR:

Tabel3.1 Hexdump Untuk *File* AMR

```
0000: 23 21 41 4D 52 0A 3C 3E 5E 24 E2 48 98 46 4C EE  #!AMR.<>^$.H.FL.
0010: DF FA 27 66 41 57 C0 00 66 E2 D1 39 14 AC 00 00  ..'fAW..f..9....
0020: 07 79 59 04 78 20 3C 28 7F 6A 2F F9 B5 B3 FE 07  .yY.x <(.j/.....
0030: 29 4F 89 3C 69 C9 82 F1 A1 A5 A8 AC 3E 66 E2 06  )O.<i.......>f..
0040: CF 12 2E 05 AC D0 3C 3A 4A 5D 8F F9 9D 0E 7E 04  ......<:J]....~.
0050: C2 2A 89 2C B7 C1 3B 6A 38 2A 42 D9 E2 75 2B 55  .*.,..;j8*B..u+U
0060: E5 D6 DF 12 BC E0 3C 30 7F 61 3F F9 E2 1F FE 09  ......<0.a?.....
0070: 29 1A BF 8C B7 37 2F 76 E8 C9 E1 D5 7F 65 68 9E  ).....7/v.....eh.
0080: 36 2F 13 8B 5A 60 3C 3E 6D 6D 87 F9 B5 0A 5E 08  6/..Z`<>mm....^.
0090: D5 DA BF FE 8D 14 C3 E1 22 34 FA F5 10 32 54 28  ........"4...2T(
00A0: 19 2A 82 C9 24 60 3C 2C 76 B2 3F 53 4A BF BE 1E  .*..$`<,v.?SJ...
00B0: 7F 2A 77 1F DF C8 D7 69 66 9C 24 F8 29 90 3F A3  .*w....if.$.).?.
00C0: 5B B6 F2 F0 60 90 3C 2E 7B 16 90 00 E6 1F 5E F0  [...`.<.{.....^.
00D0: 1D 0F 28 B5 35 21 9F C3 CA FC 01 77 98 CE 9F 00  ..(.5!.....w....
00E0: 7D 89 8F F0 01 10 3C 2C BA AD F8 00 5C 21 DE 5A  }.....<,....\!.Z
00F0: 01 1B 04 64 2C 3D 48 44 EB DD A6 BE 24 11 9B E7  ...d,=HD....$...
```

Pada tabel Hexdump menunjukkan letak *Header* pada frame pertama yaitu **#!AMR..** Setiap frame terdiri dari sebuah 1-byte header, maka sisa dari frame adalah data audio. 4 bit paling atas header terdiri dari CMR (Codec Mode Request), nilai 0 – 7 yang berlaku untuk AMR.

Pesan teks yang sudah diubah menjadi bilangan biner akan disisipkan pada frame setelah *Header*. Cara kerja metode LSB dengan contoh akan disisipkan dengan karakter a, b, dan c adalah sebagai berikut:

1. Membaca nilai biner dari ASCII misal pada karakter a sebagai berikut :
   a → kode ASCII = 97 nilai biner 01100001
2. Mengambil satu persatu bagian dari setiap bit, yaitu 0, 1, 1, 0, 0, 0, 0, 1. Kemudian setiap bit akan disisipkan pada data dengan metode LSB, misalnya:

- Data 1 : 10111111, disisipi bit 0 menjadi 10111110
- Data 2 : 10101111, disisipi bit 1 menjadi 10101111
- Data 3 : 10101100, disisipi bit 1 menjadi 10101101
- Data 4 : 11001010, disisipi bit 0 menjadi 11001010
- Data 5 : 10011010, disisipi bit 0 menjadi 10011010
- Data 6 : 10100111, disisipi bit 0 menjadi 10100111
- Data 7 : 10011000, disisipi bit 0 menjadi 10011000
- Data 8 : 11001010, disisipi bit 1 menjadi 11001011

3.5 Proses Ekstrak

Pada proses ekstrak ini dilakukan pada saat penerimaan yaitu pada pesan masuk MMS dan juga dapat dilakukan saat *browsing File* pada memory telepon dengan mengambil *File* AMR yang sudah disisipkan dengan pesan teks tersebut. Berikut pseudocode untuk proses penyisipan adalah sebagai berikut:

```
Nilai awal :
        by : larik byte dari File amr terkompresi
        ic : index byte File amr yg terkompresi diawali dengan 5 karena 0 - 4 adalah byte header File amr
        dtBy : larik byte dari teks pesan
        temp : variable byte pesan
        idsub : indek bit pesan yg sekarang diisi 0
        id : index byte pesan yg sekarang diisi 0
Algortima :
Selama ic < jumlah by lakukan
        ambil frame size fs dari by untuk yang dimulai dari byte ic;
        tambahkan ic dengan fs terhadap ic
        jika ic < panjang by yg berarti indek ic ada pada byte File amr maka lakukan
                ambil nilai bit x pada by byte ke ic untuk bit yg paling terakhir
                Masukkan bit x ke temp sesuai indek idsub
                tambahkan idsub dengan nilai 1
        jika idsub = 8 yg berarti pengisian bit temp sudah 1 byte lakukan
                idsub direset ke 0
                temp dimasukkan ke dtBy sesuai indek id
                tambahkan id dengan nilai 1
```

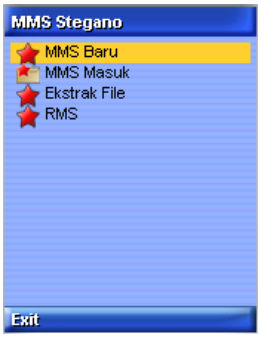
Proses ekstrak dilakukan dengan langkah – langkah sebagai berikut:

1. Mengambil nilai bit terakhir byte pesan ke-*temp* dengan meng-*and*-kan dengan 1.
2. Menyimpan hasil setelah di-*and*-kan dengan 1.
3. Menjumlahkan semua hasil perhitungan untuk *temp*=0 sampai *temp*=7.
4. Menentukan karakter ASCII yang bersesuaian dengan hasil perhitungan.

3.6 Antarmuka MMS

Antarmuka MMS dibangun dengan menggunakan NetBeans 6.7.1 dengan menggunakan bahasa pemrograman J2ME yang merupakan pemrograman untuk *mobile device.*

Menu utama yang diberi nama MMS Stegano dibangun dengan menggunakan *Form* yang didalamnya terdapat beberapa *List* yang mampu menyediakan fungsi untuk memilih elemen kepada pengguna.



Gambar 3.3 Menu Awal Pesan MMS

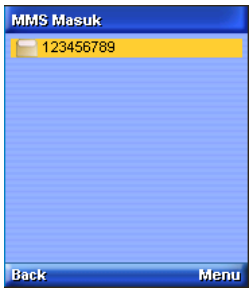
3.7 MMS Baru

List ini berfungsi untuk memulai pengiriman pesan MMS. Pengguna diwajibkan untuk memasukkan nomor tujuan, membuat pesan teks, dan memasukkan File yang dapat diambil dalam memori telepon agar proses pengiriman dapat dilakukan.

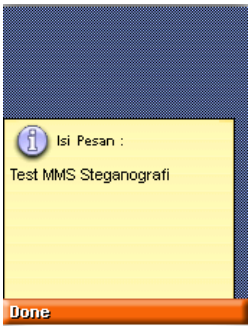
Gambar 3.4 Membuat MMS Baru

3.8 MMS Masuk

List ini berfungsi sebagai tempat pesan MMS yang sudah diterima oleh pengguna telepon seluler. Sebuah pesan teks yang disisipkan dalam *File* tersebut akan dapat dilihat dan dibaca jika melakukan proses ekstrak terlebih dahulu.



Gambar 3.5 Pesan Pada MMS Masuk

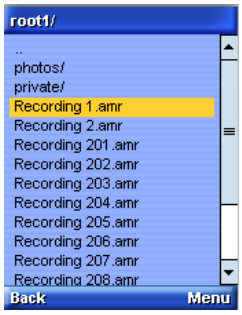


Gambar3.6 Pesan Yang Sudah Diekstrak

Untuk dapat melihat isi pesan yang disembunyikan dalam *File* AMR tersebut, pengguna harus melakukan proses ekstrak.

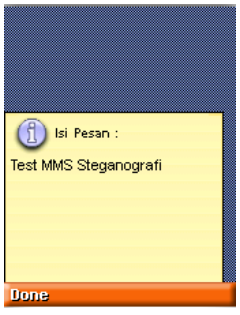
3.9 Ekstrak *File*

Berikut adalah gambar saat *File* yang sudah disisipkan dengan pesan dicari dalam memori telepon.



Gambar 3.7 Proses Pada *File Browser*

Jika *File* tersebut sudah ditemukan, maka proses ekstrak pun bisa dilakukan dan isi pesan teks pada *File* tersebut bisa dibaca.



Gambar 3.8 Pesan Yang Sudah Diekstrak Melalui *File Browser*

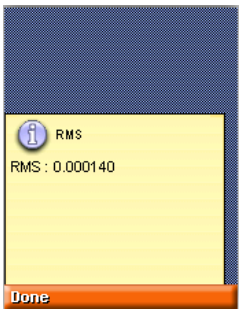
3.10 RMS

List RMS (Root Mean Square) merupakan list yang berfungsi untuk dapat mengetahui nilai parameter dari *File* audio AMR saat sebelum dan setelah disisipkan. Hasil yang

diperoleh dalam perhitungan RMS adalah untuk mengetahui seberapa besar keamanan pesan yag disisipkan dalam *File* AMR. Berikut adalah gambar untuk proses inputan 2 *File* sebelum dan setelah dilakukan penyisipan.

Gambar 3.9 Proses Inputan Untuk Hitung RMS

Dua inputan *File* yang digunakan seperti yang terlihat pada gambar 4.8 adalah *File* yang sebelum dan sesudah disisipkan dengan pesan teks.



Gambar3.10 Hasil Perhitungan RMS

3.11 Uji Coba Kualitas Suara

Uji coba ini dilakukan untuk mengetahui kualitas suara dari *File* AMR saat sebelum dan setelah disisipkan dengan pesan teks. Uji coba dilakukan pada aplikasi desktop yaitu Audacity 1.3.12-Beta. Uji coba kualitas ini dilakukan dengan dua *File* AMR yang berbeda ukurannya yaitu dengan *File* yang berukuran 3.030 *Bytes* dengan banyaknya inputan yaitu 10 karakter dan 12.460 *Bytes* dengan banyaknya inputan yaitu 47 karakter. Dimana inputan banyaknya karakter tersebut merupakan inputan penuh untuk *File* yang sudah ditentukan yaitu 3.030 *Bytes* dan 12.406 *Bytes*.

Kualitas suara yang dihasilkan oleh *File* asli dengan *File* hasil yaitu yang sudah disisipkan dengan pesan teks. Analisis telah dilakukan pada tiap titik frekuensi yang menghasilkan tinggi rendahnya suatu sinyal dengan satuan decibel (dB). Berikut adalah beberapa titik frekuensi yang terdapat pada audio AMR.

Tabel 3.1 Titik Frekuensi dan Desibel Untuk *File* 3.030 *Bytes*

File Audio Asli (AMR)		*File* Audio Hasil (AMR)	
Frequency (Hz)	Level (dB)	*Frequency* (Hz)	Level (dB)
500,000000	-50,252426	500,000000	-50,252426
1000,000000	-64,873795	1000,000000	-64,873795
1500,000000	-60,753960	1500,000000	-60,753960
2000,000000	-66,844803	2000,000000	-66,844803
2500,000000	-70,571045	2500,000000	-70,571045
3000,000000	-72,319199	3000,000000	-72,319199
3500,000000	-82,159119	3500,000000	-82,159119

Tabel3.2 Titik Frekuensi dan Desibel Untuk *File* 12.406 *Bytes*

File Audio Asli (AMR)		*File* Audio Hasil (AMR)	
Frequency (Hz)	Level (dB)	*Frequency* (Hz)	Level (dB)
500,000000	-53,000725	500,000000	-53,000725
1000,000000	-45,980370	1000,000000	-45,980370
1500,000000	-47,768562	1500,000000	-47,768562
2000,000000	-52,023022	2000,000000	-52,023022
2500,000000	-58,130688	2500,000000	-58,130688
3000,000000	-59,296417	3000,000000	-59,296417
3500,000000	-59,387749	3500,000000	-59,387749

3.12 Uji RMS

Untuk mengetahui besarnya perbandingan *File* AMR saat sebelum dan setelah disisipkan dengan pesan teks maka dilakukan uji coba dengan RMS.

Uji RMS dengan menggunakan *File* AMR yang berukuran 3.030 *Bytes*.

Tabel3.3 Uji RMS Dengan Besar *File* 3,030 *Bytes*

Inputan Pesan (Karakter)	RMS (Satu Satuan Byte)
1 (a)	0,000572
2 (ab)	0,000808
3 (abc)	0,001044
4 (abcd)	0,001190
5 (abcde)	0,001361
6 (abcdef)	0,001512
7 (abcdefg)	0,001683
8 (abcdefgh)	0,001777
9 (abcdefghi)	0,001896
10 (abcdefghij)	0,002008

Uji RMS dengan menggunakan *File* AMR yang berukuran 12.406 *Bytes*.

Tabel3.4 Uji RMS Dengan Besar *File* 12.406 *Bytes*

Inputan Pesan (Karakter)	RMS (Satu Satuan Byte)
1 (a)	0,000140
2 (ab)	0,000197
3 (abc)	0,000255
4 (abcd)	0,000291
5 (abcde)	0,000332
6 (abcdef)	0,000369
7 (abcdefg)	0,000411
8 (abcdefgh)	0,000434
9 (abcdefghi)	0,000463
10 (abcdefghij)	0,000490

Berdasarkan nilai yang diperoleh dari tabel, semakin banyak jumlah karakter yang disisipkan pada *File*, maka membuat nilai RMS semakin besar.

IV. KESIMPULAN

Kesimpulan yang dapat diambil dari penelitian yang telah dilakukan adalah sebagai berikut:

1. Steganografi pada *File* audio AMR dapat memberikan keamanan pesan dengan menyisipkan dan mengekstrak pesan dalam *File* menggunakan metode LSB,

sehingga memperkecil kemungkinan terbacanya pesan tersembunyi dalam *File* audio yaitu *File* dengan *extension File* AMR.

2. Apllikasi MMS yang dibangun pada penelitian ini dengan mengimplementasikan steganografi menggunakan metode LSB, dapat menyembunyikan pesan kedalam *file* dengan tahapan menyisipkan bit karakter pesan kedalam bit *file* audio amr dan dapat melihat isi dari pesan tersebut dengan cara ekstrak *file*.
3. Pada penelitian ini perbedaan kualitas suara sebelum dan setelah penyisipan pada *file* yang berukuran 3.030 *bytes* dengan jumlah maksimal 10 karakter pesan dan pada *file* yang berukuran 12.406 *bytes* dengan jumlah maksimal 47 karakter pesan perbedaan kualitas suara yang dihasilkan sangat kecil hal ini disebabkan karena nilai bit awal sampai bit akhir dalam 1 *byte* karakter pesan disisipkan pada nilai bit akhir setiap frame. Dapat diketahui jumlah frame yang terdapat pada file yang berukuran 3.030 *bytes* adalah 80 frame dan pada file yang berukuran 12.406 *bytes* adalah 376 frame. Jumlah frame didapat dengan menghitung jumlah bit dari 10 karakter pesan untuk file berukuran 3.030 *bytes*.

V. DAFTAR PUSTAKA

[1] Asti Dwi Irfanti, M.Kom. 2006. “Implementasi Smart Shopping dengan Menggunakan Short Message Service dan Multimedia Message Service”. Sekolah Tinggi Manajemen Informatika dan Tehnik Komputer Surabaya. Surabaya.

[2] Dewi, Risa Astari. 2009. “Penerapan Audio Steganografi dalam Intrasonics”. Departemen Teknik Informatika. Institut Teknologi Bandung.

[3] www.informatika.org/~rinaldi/Kriptografi/2008-2009/Makalah1/MakalahIF30581-2009-a053.pdf

[4] Kendall, Kenneth E dan Julie E. Kendall. 2006. “Analisis dan Perancangan Sistem”. Edisi Kelima. Indeks. Jakarta.

[5] Muhdin. 2003. ”Penyusunan dan Validasi Fungsi Volume Batang”. Studi Kasus pada Jenis Gmelina arborea Roxb di Areal P.T. Wanakasita Nusantara. Jambi.

[6] Purnama. 2008. “Pemrograman J2ME Tingkat Dasar”. Penerbit: GITAMEDIA PRESS. Surabaya

[7] Pressman, Roger S. 2005. “Software Engineering A Practitioner’s Approch Sixth Edition”. New York: Mc-Graw-Hill.

[8] Ria, Gemita. 2006. "Studi Perbandingan Steganografi pada Audio, Video, dan Gambar”. Bandung: Institut Teknologi Bandung.

[9] Scott B. Guthery & Mary J. Cronin, 2003, Developing MMS Application Multimedia Messaging Services for Wireless Network, The McGraw-Hill Companies, Inc. United State of America.