JURNAL DASI ISSN: 1411-3201
Vol. 10 No. 1 Maret 2009

PENDEKATAN METODE *LEAST BIT MODIFICATION* UNTUK MERANCANG APLIKASI *STEGANOGRAPHY* PADA FILE AUDIO DIGITAL TIDAK TERKOMPRESI

Ema Utami
STMIK AMIKOM Yogyakarta

Abstraksi

Pengamanan terhadap informasi yang didistribusikan sangat penting untuk menjaga kerahasiaan, keutuhan, dan keaslian informasi. Proses pengamanan dapat dilakukan dengan menyembunyikan informasi tersebut pada media digital lain agar tidak terlihat keberadaannya. Teknik ini disebut Steganography, yaitu seni penyembunyian data ke dalam media digital dengan metode tertentu sehingga orang lain tidak menyadari ada sesuatu di dalam media digital tersebut.

Dalam tulisan ini akan dilakukan penelitian untuk menyembunyikan sebuah informasi ke dalam file audio digital tidak terkompresi (.wav) sebagai file carrier dengan menggunakan metode Least Significant Bit Modification. Least Significant Bit Modification merupakan metode penyembunyian informasi dengan memodifikasi bit terakhir dari file carrier dengan bit-bit informasi dan hanya menyebabkan perubahan nilai bit satu lebih tinggi atau satu lebih rendah. Sistem dirancang dengan dua buah proses utama yaitu tahap Embedding dan tahap Extracting. Sistem diimplemetasikan pada perangkat lunak Microsoft Visual Studio .NET dengan bahasa pemrograman berorientasi objek C#.

Aplikasi Steganografi ini bisa dijadikan salah satu alternatif pengamanan informasi rahasia tanpa perlu membuat pihak lain merasa curiga, karena proses steganografi dengan metode Least Significant Bit Modification tidak merubah ukuran file carrier dan tidak mengakibatkan penurunan kualitas suara yang signifikan sehingga kemungkinan terdeteksi juga sangat kecil. File pembawa yang sudah disisipkan informasi (stego file) tidak dapat dideteksi

tanpa melakukan perbandingan bit-bit LSB terhadap file pembawa yang asli.

Kata Kunci : *Steganografi, Least Significant Bit Modification, Wideband Angular Vibration Experiment (WAVE) Format.*

Pendahuluan

Banyak metode yang dapat digunakan untuk aplikasi steganografi. Salah satu metode yang digunakan yaitu modifikasi LSB *(Least Significant Bit Modification*). *Least Significant Bit Modification* merupakan metode penyembunyian informasi rahasia dengan memodifikasi LSB file *carrier*. Modifikasi LSB dilakukan dengan memodifikasi bit terakhir dalam satu byte data dengan bit-bit informasi dan hanya menyebabkan perubahan nilai bit satu lebih tinggi atau satu lebih rendah. Berdasarkan uraian yang telah diutarakan sebelumnya, maka dapat dirumuskan masalah yaitu:

1. Bagaimana membuat aplikasi steganografi dengan metode *Least Significant Bit Modification (LSB)* pada file audio digital tidak terkompresi (.wav).
2. Bagaimana kinerja file audio digital yang telah disisipkan informasi dengan melihat kualitas suara dan kehandalannya terhadap kemungkinan terjadinya pendeteksian.

Agar permasalahan tidak terlalu meluas, maka diperlukan adanya batasan masalah. Adapun permasalahan yang akan diteliti adalah sebagai berikut :

1. Menggunakan metode modifikasi LSB *(Least Significant Bit Modification)* untuk menyembunyikan informasi ke dalam file audio digital tidak terkompresi (.wav).
2. Pembuatan perangkat lunak untuk aplikasi steganografi dengan menggunakan bahasa pemrograman C# .Net.
3. Analisis pengaruh besar-kecilnya informasi yang disembunyikan terhadap kualitas suara yang dihasilkan oleh file audio.

4. Melakukan uji keberhasilan terhadap kemungkinan terjadinya pendeteksian.

Berdasarkan permasalahan yang telah dibahas sebelumnya, maka tujuan yang akan dicapai dalam penelitian ini adalah sebagai berikut :

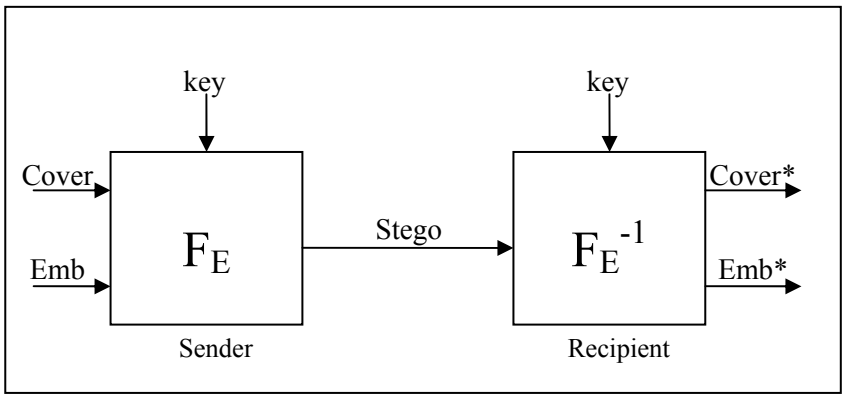
1. Mengaplikasikan teknologi steganografi dengan menggunakan teknik modifikasi LSB file *carrier* untuk menyembunyikan informasi.
2. Menganalisa pengaruh besar-kecilnya informasi yang disembunyikan terhadap kualitas suara yang dihasilkan oleh file audio.

Manfaat yang diharapkan bisa dicapai dalam penelitian ini adalah membantu *user* dalam proses pengamanan informasi untuk menjaga kerahasiaan *(privacy)* informasi, keutuhan informasi *(integrity)*, ukuran file dan keaslian informasi *(authentication)* sehingga informasi yang didistribusikan tidak mengalami perubahan oleh pihak yang tidak berwenang

Pembahasan

1. Steganografi

Steganografi berbeda dengan Kriptografi, letak perbedaannya adalah hasil keluarannya. Hasil dari Kriptografi biasanya berupa data yang berbeda dari bentuk aslinya dan biasanya datanya seolah-olah berantakan dan dapat dikembalikan ke bentuk semula. Sedangkan Steganografi ini memiliki bentuk persepsi yang sama dengan bentuk aslinya, tentunya persepsi disini oleh indera manusia, tetapi tidak oleh komputer atau perangkat pengolah digital lainnya. Gambar 1 dibawah ini menunjukkan metode dasar bagaimana cara kerja Steganografi (Bender, 1996) :



Gambar 1: Cara kerja Steganografi secara umum

F_E : *Embedding* (Penggabungan berkas *cover* dengan berkas pesan)
F_E^{-1} : *Extracting* (Pengambilan berkas pesan dari berkas *cover*)
Cover : Berkas data yang akan disisipkan informasi *(carrier)*
Key : Kunci yang digunakan
Emb : Pesan yang akan disisipkan
Stego : Berkas *cover* yang sudah berisi pesan

Dalam proses Steganografi terdapat beberapa kriteria yang harus dipenuhi, kriterianya adalah sebagai berikut (Vembrina, 2006) :

1. ***Impercepbility***. Keberadaan pesan tidak dapat dipersepsi oleh indra manusia, baik indra pendengaran maupun indra penglihatan.
2. ***Fidelity***. Mutu media pembawa tidak berubah banyak akibat proses penyisipan.
3. ***Recovery***. Pesan yang disembunyikan harus dapat diungkap kembali. Tujuan Steganografi adalah menyembunyikan informasi, maka sewaktu-waktu informasi yang disembunyikan ini harus dapat diambil kembali untuk dapat digunakan lebih lanjut sesuai kebutuhan.

Steganografi menggunakan sebuah berkas yang disebut dengan *cover* atau biasa disebut dengan *carrier*, tujuannya sebagai pembawa dari pesan yang dirahasiakan. Banyak format *carrier* yang dapat dijadikan media untuk menyembunyikan pesan, diantaranya (Johnson, 1995) :

1. Format image (Format gambar) : Bitmap (.bmp), Graphics Interchange Format (.gif), Paintbrush Bitmap Graphic (.pcx), Joint Photographic Expert Group (.jpeg), dll.
2. Format audio (Format suara) : Wideband Angular Vibration Experiment (.wav), Motion Picture Expert Group Audio Stream Layer III (.mp3), Musical Instrument Digital Interface (.midi), dll.
3. Format lain : teks file, HyperText Markup Language (.html), Portable Document Format (.pdf), video dll.

Pada dasarnya setiap media digital dapat digunakan sebagai media pembawa pada proses Steganografi. Penerapan Steganografi pada media digital menggunakan metode tertentu dan tergantung dari media yang dipilih sebagai *carrier*-nya.

2. Least Significant Bit Modification

Metode Steganografi yang paling umum pada format suara adalah *Least Significant Bit Modification*. Metode ini banyak digunakan karena komputasinya tidak terlalu kompleks dan pesan yang disembunyikan cukup aman. Metode ini memodifikasi nilai yang paling kurang signifikan dari jumlah bit dalam 1 byte file *carrier*. Bit yang memiliki signifikansi paling tinggi adalah numerik yang memiliki nilai tertinggi (misal, $2^7 = 128$), artinya bila terjadi perubahan pada bit ini akan menghasilkan perubahan yang sangat signifikan. Bit yang memiliki signifikansi paling rendah adalah numerik yang memiliki nilai terendah (misal, $2^0 = 1$), artinya bila terjadi perubahan pada bit ini akan menghasilkan perubahan yang tidak terlalu signifikan. Sebagai contoh, akan dilakukan proses penyembunyian karakter ‘G’ (ASCII 71) pada berkas *carrier* yang

berukuran 8 byte. *Least Significant Bit* dari file *carrier* ditandai dengan garis bawah.

Berkas *carrier* dalam biner dengan ukuran 8 byte :

‘ 10010101 00001101 11001001 10010110
00001111 11001011 10011111 00010000 ’

Karakter ‘G’ dalam biner dengan ukuran 1 byte :

‘ 01000111 ’

Kedelapan bit ini nantinya akan dimasukan kedalam *Least Significant Bit* dari tiap-tiap byte pada file *carrier* seperti berikut ini :

Berkas *carrier* dalam biner dengan ukuran 8 byte :

‘ 10010101 00001101 11001001 10010110
00001111 11001011 10011111 00010000 ’

Karakter ‘G’ dalam biner dengan ukuran 1 byte :

‘ 01000111 ’

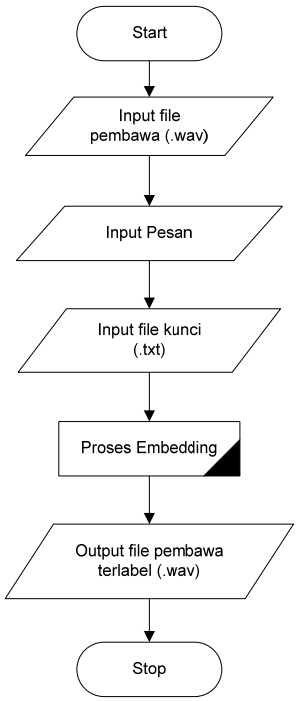
Proses *Least Significant Bit Modification* :

‘ 10010100 00001101 11001000 10010110
00001110 11001011 10011111 00010001 ’

Pada contoh diatas, hanya sebagian dari *Least Significant Bit* file *carrier* yang berubah (ditunjukkan dengan karakter miring). Berdasarkan teori yang didapat adalah bahwa kemungkinan terjadinya perubahan bit adalah sekitar 50 %, karena peluangnya perubahannya adalah antara 0 atau 1 dan dengan mengubah *Least Significant Bit* maka ukuran dari file pembawa tidak akan berubah sehingga akan sulit untuk terdeteksi (Bender, 1996).

3. Perancangan Proses *Embedding* dan Proses *Extracting* pada Perangkat Lunak

Secara umum bagan alir logika program atau *program logic flowchart* tersebut dapat dilihat pada gambar 2 dan 3 berikut :

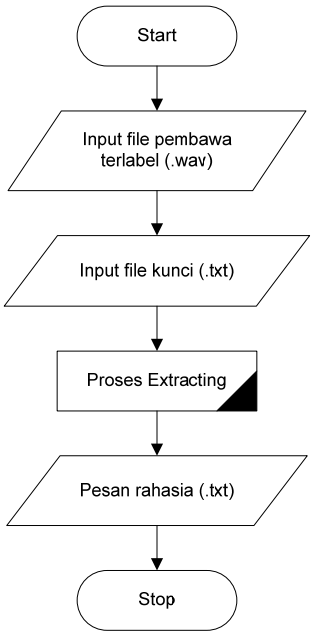


Gambar 2. Gambar Diagram Alir *Embedding Information*

Proses penyembunyian informasi ke dalam file pembawa (.wav) secara umum seperti pada gambar 1 adalah sebagai berikut :

1. Input file pembawa (.wav).
2. Input pesan yang hendak disembunyikan yang dibaca sebagai pesan baru atau pesan berbentuk file (.txt) yang sudah ada sebelumnya.
3. Input file kunci (.txt) sebagai password.

4. Proses *Embedding* informasi ke dalam file pembawa dilakukan dengan mengganti bit-bit LSB pada file pembawa dengan bit-bit informasi.
5. Ouptut berupa file pembawa terlabel (.wav) atau *stego file*.



Gambar 3. Gambar Diagram Alir *Extracting Information*

Proses pengambilan informasi dari dalam file pembawa terlabel (.wav) secara umum seperti pada gambar 3 adalah sebagai berikut :

1. Input file pembawa terlabel (.wav) atau *stego file*.

2. Input file kunci (.txt) sebagai password. File kunci ini merupakan *symmetrical key* atau kunci simetris dimana kunci ini digunakan pada proses *embedding* dan *extracting*.
3. Porses *Extracting* informasi dari file pembawa terlabel.
4. Output berupa file pesan (.txt).

Dalam steganografi terdapat dua proses utama yaitu proses *embedding* atau penyisipan informasi pada file pembawa dan proses *extracting* atau pengambilan informasi dari file pembawa. Berdasarkan diagram alir sebelumnya bahwa proses *embedding information* membutuhkan tiga inputan dan menghasilkan satu output. Inputan berupa file pembawa (.wav), pesan, dan file kunci (.txt). Sedangkan output berupa file pembawa terlabel (.wav). Proses *extracting information* membutuhkan dua inputan dan menghasilkan satu output. Inputan berupa file pembawa terlabel (.wav) dan file kunci (.txt). Sedangkan output berupa pesan yang dapat disimpan kedalam bentuk (.txt). Dalam bagian ini akan dijelaskan tahapan yang dipakai dalam proses penyembunyian informasi dan pengambilan informasi mulai dari inputan yang dibutuhkan, proses, sampai menghasilkan output.

4. Perancangan Antarmuka

Antar muka merupakan media interaksi antara program (komputer) dengan *user*. Melalui tampilan antarmuka, *user* dapat menjalankan program aplikasi dengan mudah tanpa harus mengetahui bagaimana program bekerja dan menghasilkan output yang diminta. Perancangan antar muka aplikasi steganografi pada tugas akhir ini diimplementasikan menggunakan bahasa pemrograman berorientasi objek dengan bahasa pemrograman C#. Adapun perancangan antarmukanya adalah sebagai berikut :

Gambar 4. Rancangan Form Login

Gambar 4 menunjukkan rancangan form Login pada aplikasi steganografi. Keterangan dari rancangan form tersebut adalah sebagai berikut :

1. Button “Ok” adalah tombol untuk login ke menu utama, dimana pengguna harus menginputkan *user name* dan *password* sesuai dengan ketentuan progam aplikasi steganografi ini.
2. Button “Close” adalah tombol untuk keluar aplikasi.

Gambar 5. Rancangan Form Main Menu

Gambar 5 menunjukkan rancangan form Main Menu pada aplikasi steganografi. Keterangan dari rancangan form tersebut adalah sebagai berikut :

1. Pada menu "Steganographic" terdapat tiga submenu yaitu submenu "Hide Message", submenu "Extract Message", dan submenu "Exit".
   a. Submenu "Hide Message" digunakan untuk menyembunyikan informasi ke dalam file pembawa.
   b. Submenu "Extract Message" digunakan untuk mengekstraksi informasi dari file stego.
   c. Submenu "Exit" digunakan untuk keluar dari aplikasi.
2. Pada menu "Help" terdapat dua submenu yaitu submenu "Hide Message Help" dan submenu "Extract Message Help".
   a. Submenu "Hide Message Help" berfungsi sebagai form petunjuk untuk proses penyembunyian informasi.
   b. Sedangkan submenu "Extract Message Help" berfungsi sebagai form petunjuk untuk proses pengambilan informasi dari file pembawanya.
3. Pada menu "About" akan ditampilkan informasi tentang program aplikasi steganografi ini.
4. Label "About Steganographic" berfungsi sebagai tombol dimana akan menampilkan sekilas tentang abstraksi dari program aplikasi ini.

Gambar 6: Rancangan Form Hide Message

Gambar 6 menunjukkan rancangan form Hide Message pada aplikasi steganografi. Keterangan dari rancangan form tersebut adalah sebagai berikut :

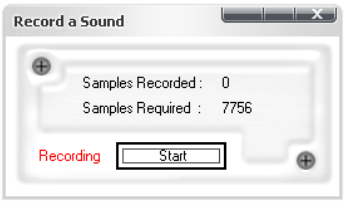
1. Button “Help” pada form Hide Message adalah tombol untuk menampilkan form Help yang mana digunakan sebagai petunjuk untuk proses *embedding*.
2. RadioButton “File (.wav)” digunakan untuk memilih *source* file pembawa yang sudah tersedia sebelumnya.
3. Button “Browse” pada GroupBox “Browse Carrier File” adalah tombol untuk mengambil file pembawa (.wav) yang nantinya digunakan sebagai media yang akan disipkan informasi.
4. RadioButton “Record New File (.wav)” digunakan untuk membuat file pembawa (.wav) yang baru.

5. Button “Borwse” pada GroupBox “Browse Key File” adalah tombol untuk mengambil file kunci (.txt) yang sudah tersedia sebelumnya.
6. RadioButton “File (.txt)” digunakan untuk memilih *source* file pesan yang sudah tersedia sebelumnya.
7. Button “Borwse” pada GroupBox “Browse Embedded File” adalah tombol untuk mengambil file pesan (.txt) yang nantinya akan disispkan dengan file pembawa.
8. RadioButton “New File (.txt)” digunakan untuk menuliskan pesan secara langsung pada program.
9. Button “Borwse” pada GroupBox “Process” adalah tombol untuk memilih letak dan memberi nama file stego (.wav) dalam media penyimpanan.
10. Button “Hide Message” pada GroupBox “Process” adalah tombol yang digunakan untuk proses penyembunyian informasi.
11. Button “Reset” pada GroupBox “Process” adalah tombol yang digunakan untuk membatalkan atau mengeset ulang inputan program ke kondisi awal.
12. Button “Close” pada GroupBox “Process” adalah tombol yang digunakan untuk kembali ke form Main Menu.

Berdasarkan gambar 6, langkah-langkah dalam menjalankan proses *embedding* adalah sebagai berikut :

1. Pilih file (.wav) yang akan dijadikan file pembawa, apakah mengambil file (.wav) yang telah tersedia atau merekam file pembawa (.wav) yang baru pada saat proses dijalankan.
2. Masukkan file kunci dimana digunakan sebagai password.
3. Tentukan pilihan apakah akan menyembunyikan file pesan (.txt) yang sudah tersedia sebelumnya atau membuat file pesan yang baru dengan menuliskannya pada *textBox* yang tersedia.
4. Pilih tempat atau *path* untuk menyimpan file pembawa terlabel (.wav) sebagai outputnya.
5. Tekan tombol *Hide Message* untuk menjalankan proses.

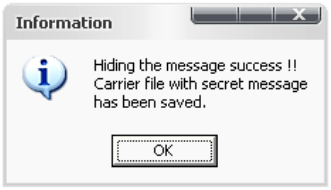
6. Jika user sebelumnya memilih *Record New File (.wav)* pada *groupBox Browse Carrier File*, maka akan muncul kotak dialog *Record a Sound* seperti pada gambar 7.



Gambar 7: Form Record a Sound

7. Tekan tombol *Start* untuk memulai proses perekaman, dan tekan tombol *Stop* untuk menghentikan proses perekaman. Tombol *Stop* ini akan aktif setelah jumlah Sample Recorded ≥ jumlah Sample Required. Setelah selesai maka secara otomatis file pembawa terlabel (.wav) sudah tersimpan.

Jika proses penyembunyian informasi telah berhasil dilakukan maka akan muncul kotak dialog informasi bahwa proses *embedding* telah berhasil dilakukan, seperti pada gambar 8.



Gambar 8: Kotak dialog informasi jika proses *embedding* berhasil

```
//public class WaveStream
private void ReadHeader(){
        BinaryReader Reader = new BinaryReader(m_Stream);
        if (ReadChunk(Reader) != "RIFF")
            throw new Exception("Invalid file format");
            Reader.ReadInt32();
        if (ReadChunk(Reader) != "WAVE")
            throw new Exception("Invalid file format");
        if (ReadChunk(Reader) != "fmt ")
            throw new Exception("Invalid file format");
        int len = Reader.ReadInt32();
        if (len < 16)
            throw new Exception("Invalid file format");

        m_Format = new WaveFormat(22050, 16, 2);
        m_Format.wFormatTag = Reader.ReadInt16();
        m_Format.nChannels = Reader.ReadInt16();
        m_Format.nSamplesPerSec = Reader.ReadInt32();
        m_Format.nAvgBytesPerSec = Reader.ReadInt32();
        m_Format.nBlockAlign = Reader.ReadInt16();
        m_Format.wBitsPerSample = Reader.ReadInt16();
}
//public class Form_Hide
private void btnHide_Click(object sender, System.EventArgs e)
{
        audioStream = new
        WaveStream(sourceStream,destinationStream);
        if (audioStream.Length <= 0){
        throw new Exception("Invalid WAV file");}
}
```

Gambar 9: Kode Program Read WAVE File

```
//public class Form_Hide
private void btnHide_Click(object sender, System.EventArgs e)
{
        if(countSamplesRequired > Int32.MaxValue){
           throw new Exception("Message too long, or bad key! This message/key
           combination requires "+countSamplesRequired+" samples, only
           "+Int32.MaxValue+" samples are allowed.");}

        if(countSamplesRequired > audioStream.CountSamples){
           String errorReport = "Carrier File too small !!\r\n"+
           "Samples available : " + audioStream.CountSamples + "\r\n"
           + "Samples needed : " + countSamplesRequired;
           btnSrcFile.Select();
           txtSrcFile.Text = "";
           throw new Exception(errorReport);}
        catch(Exception ex){
        MessageBox.Show(ex.Message);}
}

private Stream GetMessageStream()
{
           BinaryWriter messageWriter = new BinaryWriter(new
    MemoryStream());
           if(rdoMsgFile.Checked)
           {
                   FileStream fs = new FileStream(txtMsgFile.Text,
        FileMode.Open);
                   int fileLength = (int)fs.Length;
                   messageWriter.Write(fileLength);
                   byte[] buffer = new byte[fs.Length];
                   fs.Read(buffer, 0, fileLength);
                   messageWriter.Write(buffer);
                   fs.Close();
           }
           else
           {
                   messageWriter.Write(txtMessage.Text.Length);
           messageWriter.Write(Encoding.ASCII.GetBytes
        (txtMessage.Text));
           }
           messageWriter.Seek(0, SeekOrigin.Begin);
           return messageWriter.BaseStream;
```

```
}
//public class WaveUtility
private static byte GetKeyValue(Stream keyStream){
      int keyValue;
      if( (keyValue=keyStream.ReadByte()) < 0){
            keyStream.Seek(0, SeekOrigin.Begin);
            keyValue=keyStream.ReadByte();
            if(keyValue == 0){ keyValue=1; }
            }
      return (byte)keyValue;
}

public static long CheckKeyForMessage(Stream keyStream, long
messageLength){
      long requiredLength = 0, messageLengthBits = messageLength*8;
      for(int n=0; n<messageLengthBits; n++)
      {
            requiredLength += GetKeyValue(keyStream);
      }
      keyStream.Seek(0, SeekOrigin.Begin);
      return requiredLength;
}
public void Hide(Stream messageStream, Stream keyStream)
{
      byte[] waveBuffer = new byte[bytesPerSample];
      byte message, bit, waveByte;
      int messageBuffer;
      int keyByte;

      while( (messageBuffer=messageStream.ReadByte()) >= 0 )
      {
            message = (byte)messageBuffer;
            for(int bitIndex=0; bitIndex<8; bitIndex++)
            {
                  keyByte = GetKeyValue(keyStream);
                  for(int n=0; n<keyByte-1; n++)
                  {
                        sourceStream.Copy(waveBuffer, 0, waveBuffer.Length,
                        destinationStream);
                  }
                  sourceStream.Read(waveBuffer, 0, waveBuffer.Length);
                  waveByte = waveBuffer[bytesPerSample-1];
```

```
                bit=(byte)(((message & (byte)(1 << bitIndex))>0)?1:0);
                if((bit == 1) && ((waveByte % 2) == 0)){
                    waveByte += 1;}
                else if((bit == 0) && ((waveByte % 2) == 1)){
                    waveByte -= 1;}
            }
        }
}
```

Gambar 10: Kode Program Hide Message

Gambar 11: Rancangan Form Extract Message

Gambar 11 menunjukkan rancangan form Extract Message pada aplikasi steganografi. Keterangan dari rancangan perangkat lunak tersebut adalah sebagai berikut :

1. Button “Help” pada form Extract Message adalah tombol untuk menampilkan form Help yang mana digunakan sebagai petunjuk untuk proses *extracting*.
2. Button “Browse” pada GroupBox “Browse Stego File” adalah tombol untuk mengambil file *stego* (.wav) yang sudah tersedia sebelumnya.
3. Button “Browse” pada GroupBox “Browse Key File” adalah tombol untuk mengambil file kunci (.txt) yang sudah tersedia sebelumnya.
4. Button “Extract Message” pada GroupBox “Process” adalah tombol yang digunakan untuk proses pengambilan pesan dari file stego.
5. Button “Save Message” pada GroupBox “Process” adalah tombol yang digunakan untuk menyimpan pesan yang sudah diekstrak.
6. Button “Reset” pada GroupBox “Process” adalah tombol yang digunakan untuk membatalkan atau mengeset ulang inputan program ke kondisi awal.
7. Button “Close” pada GroupBox “Process” adalah tombol yang digunakan untuk kembali ke form Main Menu.

```
//public class WaveUtility
private static byte GetKeyValue(Stream keyStream){
      int keyValue;
      if((keyValue=keyStream.ReadByte()) < 0){
            keyStream.Seek(0, SeekOrigin.Begin);
            keyValue=keyStream.ReadByte();
            if(keyValue == 0){ keyValue=1;}
      }
      return (byte)keyValue;
}

//public class Form_Extract
private void btnExtract_Click(object sender, System.EventArgs e)
{
       if(txtSrcFile.Text.Length == 0)
       {
            MessageBox.Show("Please input file (.wav) \nas a stego
```

```
            file !!","
            Warning",MessageBoxButtons.OK,MessageBoxIcon.Warning);
            btnSrcFile.Select();}
        else if(txtKeyFile.Text.Length == 0)
        {
            MessageBox.Show("Please input key file \nas a password
            !!","Warning",MessageBoxButtons.OK,MessageBoxIcon.Warning
            );
        }
        else
        {
            WaveUtility utility = new WaveUtility(audioStream);
            utility.Extract(messageStream, keyStream);

            txtExtractedMessage.Text = new
            StreamReader(messageStream).ReadToEnd();
        }

        if (txtExtractedMessage.Text.Length == 0){
            throw new System.Exception();}
        else{
            MessageBox.Show("Extracting message success !!","
            Infomation",MessageBoxButtons.OK,MessageBoxIcon.Informati
            on);}

        catch(Exception){
            this.Cursor = Cursors.Default;
            MessageBox.Show("Extracting message failed !! \nFile
            Carrier hasn't a secret message \nor wrong key file","
            Errorextracting",MessageBoxButtons.OK,MessageBoxIcon
            .Error);}


//public class WaveUtility
public void Extract(Stream messageStream, Stream keyStream)
{
        byte[] waveBuffer = new byte[bytesPerSample];
        byte message, bit, waveByte;
        int messageLength = 0;
        int keyByte;

        while((messageLength==0||messageStream.Length
```

```
        <messageLength)){

            message = 0;
            for(int bitIndex=0; bitIndex<8; bitIndex++)
            {
                keyByte = GetKeyValue(keyStream);
                for(int n=0; n<keyByte; n++)
                {
                    sourceStream.Read(waveBuffer, 0,
                    waveBuffer.Length);
                }

                waveByte = waveBuffer[bytesPerSample-1];
                bit = (byte)(((waveByte % 2) == 0) ? 0 : 1);
                message += (byte)(bit << bitIndex);
            }
            messageStream.WriteByte(message);
        }
}
```

Gambar 12: Kode Program Extract Message

5. Analisis dan Pembahasan Sistem

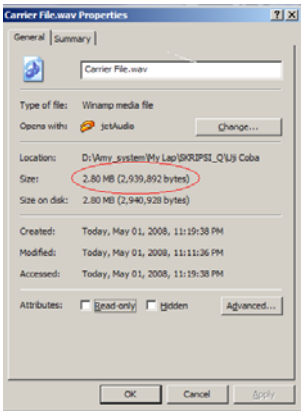
Pada bagian ini, akan dibandingkan antara file pembawa yang asli *(carrier file)* dengan file pembawa terlabel *(stego file)* yaitu dengan melihat perbandingan ukuran kedua file tersebut yang terjadi akibat proses Steganografi. Selain membandingkan ukuran file, akan dilakukan perbandingan terhadap perubahan bit-bit LSB dan perbandingan kualitas suara dari kedua file tersebut. Setelah melakukan perbandingan terhadap beberapa kriteria, akan dilakukan pengujian ketahanan terhadap file pembawa terlabel *(stego file)* untuk tidak terdeteksi oleh *software* yang dapat mendeteksi ada atau tidaknya informasi yang disembunyikan pada suatu file *(Steganalysis Software)*.

6. Percobaan Perbandingan Ukuran File

Percobaan perbandingan dilakukan terhadap file pembawa yang asli (Carrier File.wav) dengan beberapa file pembawa terlabel

yang telah disisipkan dengan beberapa pesan yang berbeda-beda ukurannya. Perbandingan ukuran file ini dilakukan untuk membuktikan bahwa dengan metode modifikasi LSB tidak akan merubah ukuran file walaupun ukuran pesan yang disisipkan berbeda-beda.

Percobaan perbandingan ukuran file dilakukan dengan membuat file (.wav) yang baru. Dengan menggunakan Perangkat Lunak Audio Editing *Sound Forge 6.0,* file dengan format WAVE dapat dibentuk. Pada percobaan kali ini *sample* untuk file pembawa diambil dari sebuah file (.mp3) kemudian dirubah formatnya (Save As...) menjadi format WAVE dengan ekstensi (.wav) dengan nama Carrier File.wav.



Gambar 13: Properties Carrier File.wav

Gambar 13 adalah keterangan *properties* dari Carrier File.wav sebagai file pembawa asli (sebelum mengalami proses Steganografi). Properties Carrier File.wav ini yaitu : Bit Rate 1411 kbps, Audio sample size 16 bit, Channels 2 (stereo), Audio sample rate 44 kHz, Audio Format PCM (Pulse Code Modulation). File pembawa yang sudah terbentuk (Carrier File.wav) akan dibandingkan dengan beberapa file pembawa terlabel yang telah disisipkan dengan beberapa

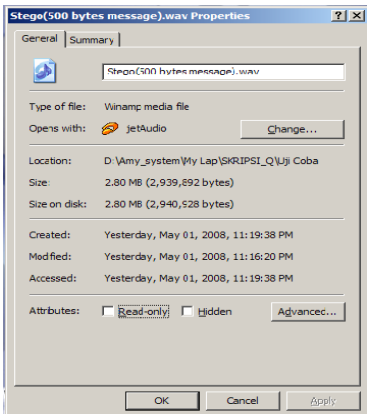
pesan yang berbeda-beda ukurannya. Hasil dari perbandingan terhadap ukuran file ini dapat dilihat dalam tabel 1 berikut :

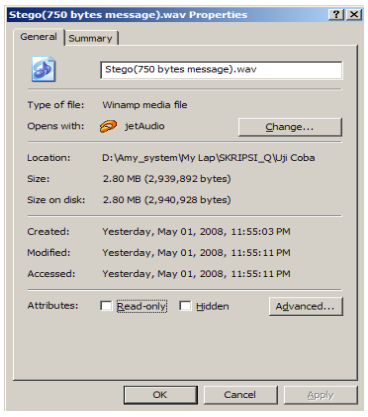
Tabel 1: Perbandingan ukuran file pembawa dan file pembawa terlabel

File Pembawa (.wav)	File Pesan (.txt)	File Pembawa Terlabel (.wav)
Carrier File (2.80 MB)	500 bytes message (500 bytes)	Stego (500 bytes message) (2.80 MB)
Carrier File (2.80 MB)	750 bytes message (750 bytes)	Stego (750 bytes message) (2.80 MB)
Carrier File (2.80 MB)	1000 bytes message (1000 bytes)	Stego (1000 bytes message) (2.80 MB)
Carrier File (2.80MB)	1500 bytes message (1500 bytes)	Stego (1500 bytes message) (2.80 MB)

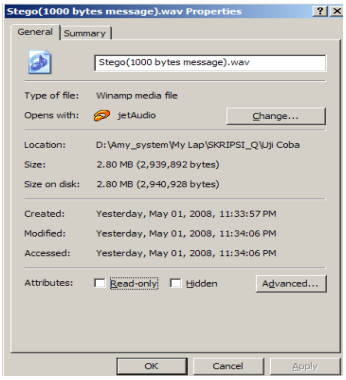
Tabel tersebut menjelaskan perbandingan ukuran file antara file pembawa dengan file pembawa terlabel. Berdasarkan tabel di atas bahwa ukuran file pembawa tidak mengalami perubahan setelah mengalami proses Steganografi dengan ukuran file pesan yang berbeda. Pembuktian ini bisa dilihat pada gambar 14 (a), 14 (b), 14 (c), 14 (d) berikut ini :

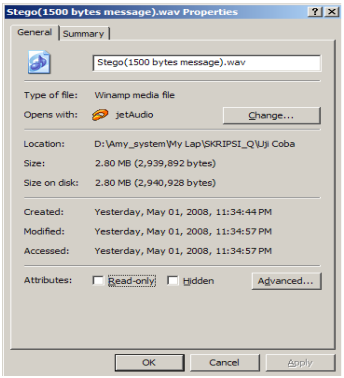
 ISSN: 1411-3201





Gambar 14 (a) Properties Stego (500 bytes message).wav
Gambar 14 (b)Properties Stego (750 bytes message).wav





Gambar 14 (c) Properties Stego (1000 bytes message).wav
Gambar 14 (d) Properties Stego (1500 bytes message).wav

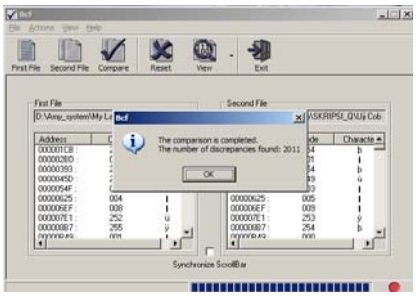
Pada gambar 13 adalah keterangan *properties* dari file pembawa asli (sebelum mengalami proses Steganografi), dan gambar 14 (a), 14 (b), 14 (c), 14 (d) adalah keterangan *properties* dari file pembawa yang telah mengalami proses Steganografi. Terbukti dari hasil pengamatan terhadap *properties* dari masing-masing file bahwa ukuran file pembawa tidak mengalami perubahan meskipun telah mengalami proses Steganografi dengan beragam ukuran file pesan yang tersembunyi di dalamnya. Berdasarkan keterangan di atas dengan tidak berubahnya ukuran file pembawa maka kemungkinan besar orang lain tidak akan mengetahui bahwa ada suatu informasi yang disispkan di dalam file tersebut.

7. Percobaan Perbandingan Bit-Bit LSB

Percobaan perbandingan bit-bit LSB ini bertujuan untuk melihat jumlah dari bit-bit yang mengalami perubahan akibat proses Steganografi terhadap file audio digital yang tidak terkompresi. Semakin banyak bit yang mengalami perubahan maka kualitas suara yang dihasilkan juga semakin buruk sehingga akan mudah dideteksi oleh telinga manusia. Dalam percobaan ini dibutuhkan suatu perangkat lunak bantu yang dapat membandingkan dua file dimana akan dibandingkan antara file pembawa asli dan file pembawa terlabel dengan tujuan untuk mengetahui seberapa besar jumlah perubahan bit-bit yang terjadi. Perangkat lunak yang digunakan adalah aplikasi dari *AX System* yaitu *Binary Comparison of File 2.0*. Hasil percobaannya adalah sebagai berikut :

- **Pesan dengan ukuran 500 Bytes**

Ukuran file pesan yang disembunyikan di dalam file pembawa adalah 500 bytes atau 4000 bit. Untuk inputan First File diisi dengan Carrier File.wav dan untuk inputan Second File diisi dengan Stego (500 bytes message).wav. Tampilan hasil uji coba dengan perangkat lunak *Binary Comparison of File 2.0* adalah sebagai berikut :

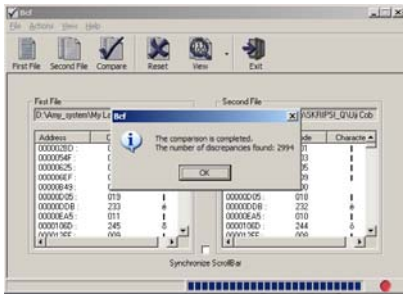


Gambar 15: Hasil perbandingan dengan 500 bytes pesan didalamnya

Dari hasil uji-coba terlihat bahwa bit LSB mengalami perubahan dan terlihat dari hasil analisa dengan perangkat lunak *Binary Comparison of File 2.0* jumlah perubahan yang terjadi adalah sebesar 2011 bit berubah.

- **Pesan dengan ukuran 750 Bytes**

Ukuran file pesan yang disembunyikan di dalam file pembawa adalah 750 bytes atau 6000 bit. Untuk inputan First File diisi dengan Carrier File.wav dan untuk inputan Second File diisi dengan Stego (750 bytes message).wav. Tampilan hasil uji coba dengan perangkat lunak *Binary Comparison of File 2.0* adalah sebagai berikut :

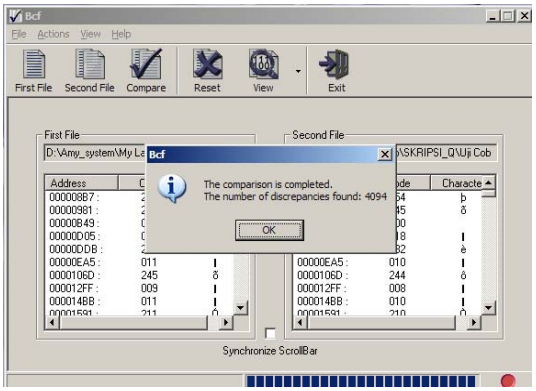


Gambar 16: Hasil perbandingan dengan 750 bytes pesan didalamnya

Dari hasil uji-coba terlihat bahwa bit LSB mengalami perubahan dan terlihat dari hasil analisa dengan perangkat lunak *Binary Comparison of File 2.0* jumlah perubahan yang terjadi adalah sebesar 2994 bit berubah.

- **Pesan dengan ukuran 1000 Bytes**

Ukuran file pesan yang disembunyikan di dalam file pembawa adalah 1000 bytes atau 8000 bit. Untuk inputan First File diisi dengan Carrier File.wav dan untuk inputan Second File diisi dengan Stego (1000 bytes message).wav. Tampilan hasil uji coba dengan perangkat lunak *Binary Comparison of File 2.0* adalah sebagai berikut :

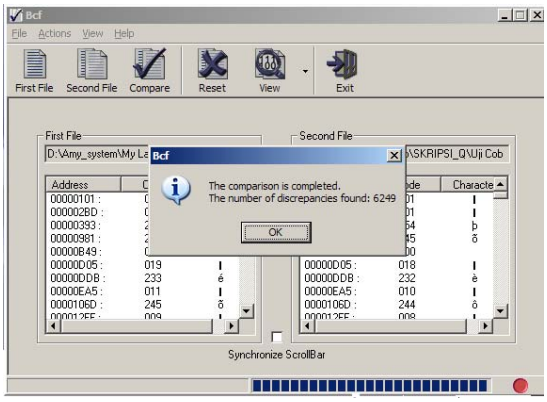


Gambar 17: Hasil perbandingan dengan 1000 bytes pesan didalamnya

Dari hasil uji-coba terlihat bahwa bit LSB mengalami perubahan dan terlihat dari hasil analisa dengan perangkat lunak *Binary Comparison of File 2.0* jumlah perubahan yang terjadi adalah sebesar 4094 bit berubah.

- **Pesan dengan ukuran 1500 Bytes**

Ukuran file pesan yang disembunyikan di dalam file pembawa adalah 1500 bytes atau 12000 bit. Untuk inputan First File diisi dengan Carrier File.wav dan untuk inputan Second File diisi dengan Stego (1500 bytes message).wav. Tampilan hasil uji coba dengan perangkat lunak *Binary Comparison of File 2.0* adalah sebagai berikut :



Gambar 18: Hasil perbandingan dengan 1500 bytes pesan didalamnya

Dari hasil uji-coba terlihat bahwa bit LSB mengalami perubahan dan terlihat dari hasil analisa dengan perangkat lunak *Binary Comparison of File 2.0* jumlah perubahan yang terjadi adalah sebesar 6249 bit berubah. Berdasarkan percobaan perbandingan bit-bit lsb di atas maka kerahasiaan file pembawa yang asli perlu dijaga, agar pihak lain tidak dapat membandingkannya dengan file pembawa terlabel.

Penutup

Keunggulan dari proses Steganografi dengan teknik modifikasi LSB yang digambarkan dalam beberapa ujicoba terhadap output dari perangkat lunak yang dirancang untuk menyembunyikan informasi di dalam sebuah file pembawa adalah sebagai berikut :

1. Ukuran file audio yang sudah disisipkan informasi tidak mengalami perubahan. Dengan tidak berubahnya ukuran file tersebut maka kemungkinan besar orang lain tidak akan mengetahui bahwa ada informasi rahasia di dalam file tersebut.
2. File *stego* tidak dapat dideteksi secara kasat telinga karena penurunan kualitas suara yang terjadi sangat kecil.
3. Output dari program ini dapat tidak terdeteksi oleh *software* yang dapat mendeteksi file *stego*.

Adapun kelemahan dari proses Steganografi dengan teknik modifikasi LSB yang digambarkan dalam beberapa ujicoba terhadap output dari perangkat lunak yang dirancang untuk menyembunyikan informasi di dalam sebuah file pembawa adalah sebagai berikut :

1. Pemilihan file pembawa (.wav) sebagai file *carrier* sangat penting untuk diperhatikan karena *noise* akan lebih terdengar pada alunan suara yang pelan atau lembut (dengan catatan ukuran informasi yang disisipkan sama).
2. Bisa dideteksi jika pihak lain membandingkan file *stego* dengan file *carrier* yang asli.

Terdapat beberapa saran untuk pengembangan lebih lanjut implementasi sistem ini diantaranya :

1. Perlunya pengembangan sistem, tidak hanya file text (.txt) yang disembunyikan melainkan semua file(*.All files).
2. Perlunya pengembangan sistem, tidak hanya satu buah file saja yang dapat disembunyikan melainkan beberapa file.

Daftar Pustaka

Bender, dkk., 1996, *Techniques For Data Hiding,* IBM Systems Journal.

Johnson, Neil F., 1995, *Steganography: Introduction, Purpose, and Structure,* Center Of Secure Information Systems George Mason University.