BAB IV

PERANCANGAN DAN IMPLEMENTASI

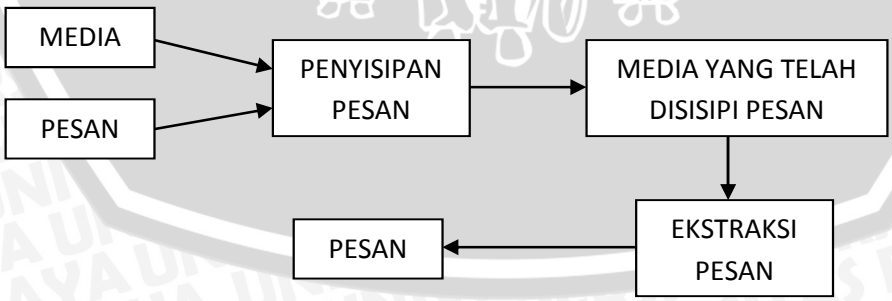
Bab ini akan menjelaskan perancangan dan implementasi aplikasi steganografi yang terdiri dari beberapa tahap. Tahap pertama merupakan tahap preproses dimana proses ini menyiapkan audio yang akan diolah, tahap kedua merupakan proses penyisipan pesan ke dalam berkas audio *digital*, dan tahap ketiga adalah proses ekstraksi pesan.

4.1 Perancangan Secara Umum

Tahap awal yang dibutuhkan adalah membuat perancangan aplikasi secara global dimana tahap awal ini berfungsi sebagai acuan dalam proses pembuatan aplikasi yang akan dibuat. Perancangan ini diawali dengan pendefinisian kegiatan pelaku atau *user* dalam menggunakan program steganografi dengan menggunakan teknik pengolahan berkas audio, serta perangkat yang digunakan meliputi blok sistem diagram dan cara kerja aplikasi.

4.1.1 Blok Diagram Sistem

Pada sistem steganografi terdiri dari beberapa langkah yang dapat digambarkan menjadi blok diagram dengan model seperti Gambar 4.1.



Gambar 4.1 Diagram Blok Sistem Secara Keseluruhan

Fungsi masing-masing bagian dalam diagram blok ini adalah senagai berikut :

1. Pesan dan media penampung pesan digunakan sebagai *input* sistem.
2. Melakukan proses penyisipan pesan kedalam media.
3. Proses penyisipan pesan menghasilkan *stego-object*.
4. Melakukan proses ekstraksi pesan dari *stego-object*.

4.1.2 Cara Kerja Aplikasi

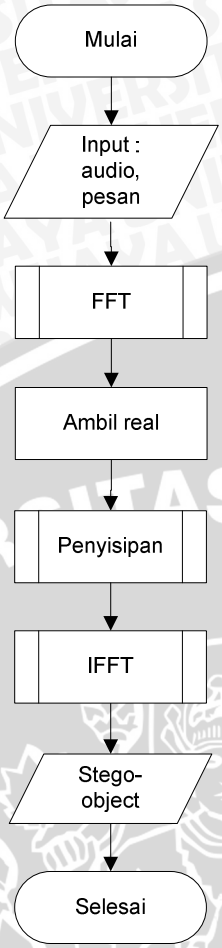
Aplikasi steganografi pada audio digital ini menggunakan metode *Spread spectrum* dimana memiliki cara kerja yang dimulai dari pengambilan berkas audio (*cover-object*) yang akan digunakan sebagai media penyisipan pesan yang sebelumnya telah dilakukan tranformasi menggunakan FFT dan pesan rahasia yang ingin disampaikan, key digunakan sebagai pengacak pesan yang setelah itu kemudian dilakukan *Spread Spectrum*, setelah itu dilakukan penyisipan pesan ke dalam audio sehingga menghasilkan audio yang telah disisipi pesan (*stego-object*), setelah itu, dilakukan proses ekstraksi pesan dari *stego-object* sehingga menghasilkan pesan rahasia yang ingin disampaikan.

4.2 Perancangan Perangkat Lunak

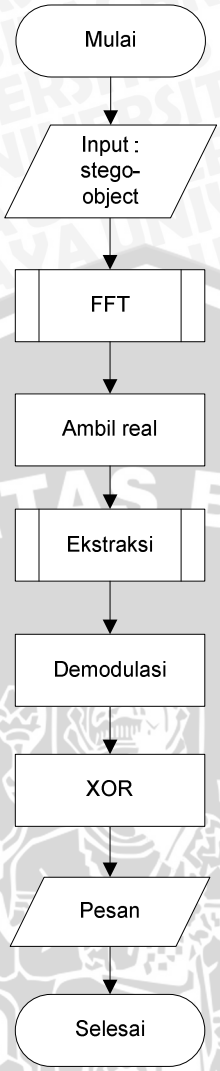
Pada bagian perancangan ini perangkat lunak yang akan dibuat menggunakan bahasa pemrograman *Microsoft Visual Studio C#.NET 2012* dan sistem yang digunakan untuk membangun perangkat lunak ini dirancang dengan spesifikasi mampu melakukan hal-hal berikut :

1. Mengakses data audio yang telah tersimpan di dalam komputer.
2. Mengakses pesan yang akan disisipkan kedalam audio.
3. Melakukan proses FFT.
4. Melakukan proses penyisipan pesan.
5. Malakukan proses IFFT.
6. Melakukan proses ekstraksi pesan.

Sedangkan untuk detail desain aplikasi secara umum akan ditunjukkan pada gambar 4.2 dan 4.3.



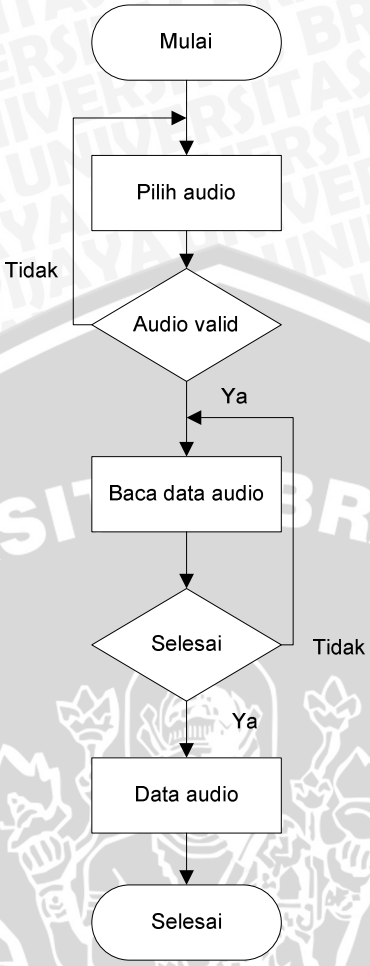
Gambar 4.2 Detail Desain Aplikasi Penyisipan Pesan



Gambar 4.3 Detail Desain Aplikasi Ekstraksi Pesan

4.2.1 Perancangan *Preprocess*

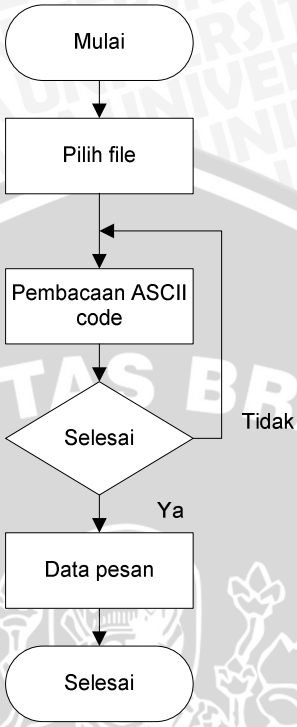
Pada *preprocess* terdapat dua langkah yaitu sub *preprocess input* audio dan *input* pesan. *Preprocess input* audio adalah proses dimana dilakukan pengambilan berkas audio sebagai masukan yang akan digunakan sebagai *cover-object* pada proses steganografi. Sedangkan input pesan adalah proses dimana dilakukannya mengambil masukan berupa pesan teks yang akan disisipkan pada *cover-object*. *Flowchart* dari perancangan *preprocess* tersebut dapat dilihat pada Gambar 4.4 dan 4.5.



Gambar 4.4 *Flowchart Input* Audio

Penjelasan *flowchart* sebagai berikut :

1. Pilih berkas audio sebagai masukan.
2. Validasi berkas audio jika sesuai dengan format yang diinginkan jalankan proses pembacaan data, jika tidak kembali ke pemilihan berkas audio.
3. Membaca data dari berkas audio sampai data pada berkas audio terbaca secara keseluruhan.
4. Jika data telah selesai dibaca maka tampilkan info data berkas audio jika tidak jalankan proses pembacaan data kembali.
5. Menampilkan info data berkas audio yang sudah dipilih.



Gambar 4.5 *Flowchart Input* Pesan

Penjelasan *flowchart* sebagai berikut :

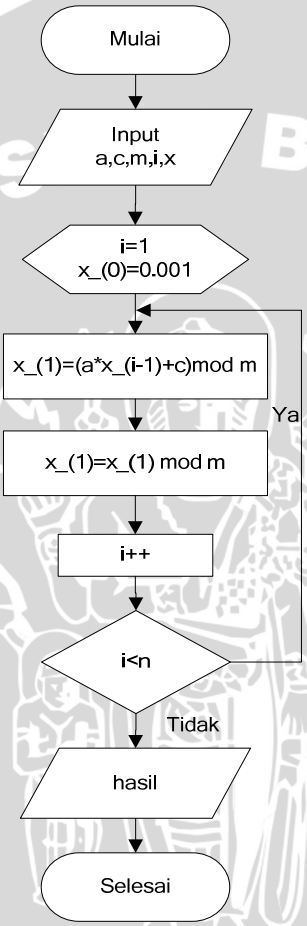
1. Memilih pesan sebagai masukan.
2. Lakukan proses pembacaan kode ASCII pada pesan yang dipilih.
3. Jika semua proses diatas selesai maka akan ditampilkan data pesan jika tidak lakukan proses pembacaan data pesan kembali.

4.2.2 Perancangan Proses

Setelah didapat berkas audio dan pesan maka dilanjutkan dengan pemrosesan audio untuk melakukan steganografi pada audio. Proses yang dilakukan pada tahap ini antara lain pembangkitan *pseudonoise*, *spread spectrum*, transformasi audio ke ranah frekuensi, penyisipan pesan, dan transformasi kembali audio ke ranah waktu.

4.2.2.1 Perancangan Proses Pembangkitan Bilangan Acak Semu LCG

Linear Congruental Generator (*LCG*) adalah salah satu algoritma pembangkitan bilangan *pseudorandom*. Algoritma ini merupakan pembangkit bilangan acak semu yang sederhana yang dalam proses *spread spectrum* nantinya akan di gunakan sebagai noise dalam modulasi dengan pesan yang akan di sisipkan. Berikut adalah *flowchart* dari proses LCG :



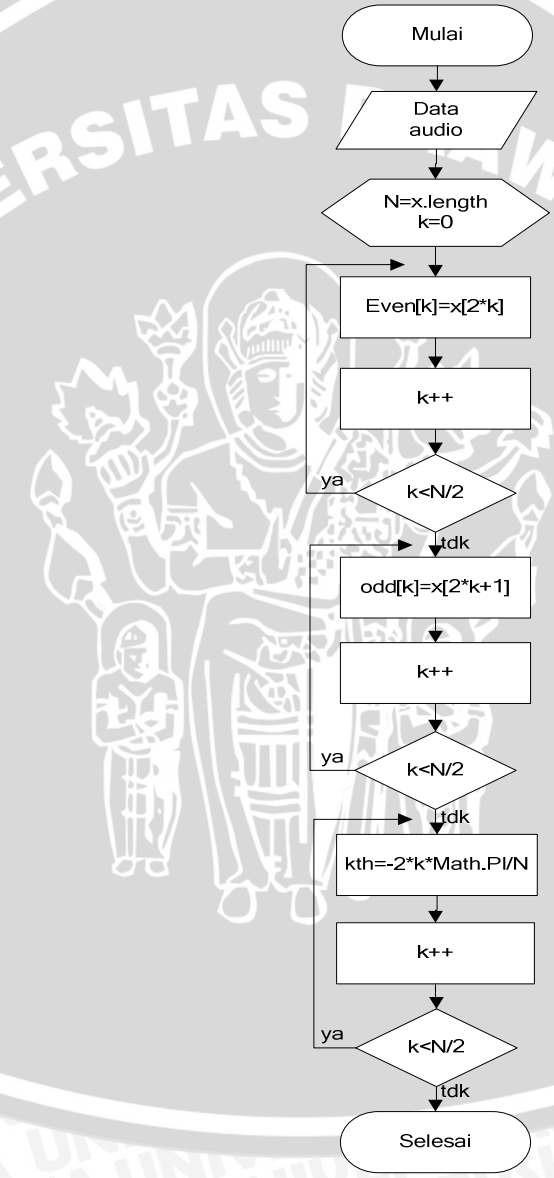
Gambar 4.6 *Flowchart* LCG (*Linear Congruental Generator*)

Dengan algoritma sebagai berikut :

```
double[] X_ = new double[count];
     X_[0] = 0.001;
     for (int i = 1; i < count; i++)
     {
         X_[i] = (a * X_[i - 1] + c) % m;
         X_[i] = X_[i] / m;
     }
     return X_;
```

4.2.2.2 Perancangan Proses Transformasi FFT

Fast Fourier Transform (FFT) adalah proses yang digunakan untuk mentransformasikan berkas audio dari domain waktu ke domain frekuensi dengan waktu yang relatif cepat. transformasi ini diperlukan dalam steganografi dengan teknik *spread spectrum* ini, dikarenakan pesan natinya akan disisikan pada berkas audio setelah berkas audio ditransformasikan ke domain frekuensi. Berikut adalah *flowchart* dari proses FFT tersebut :



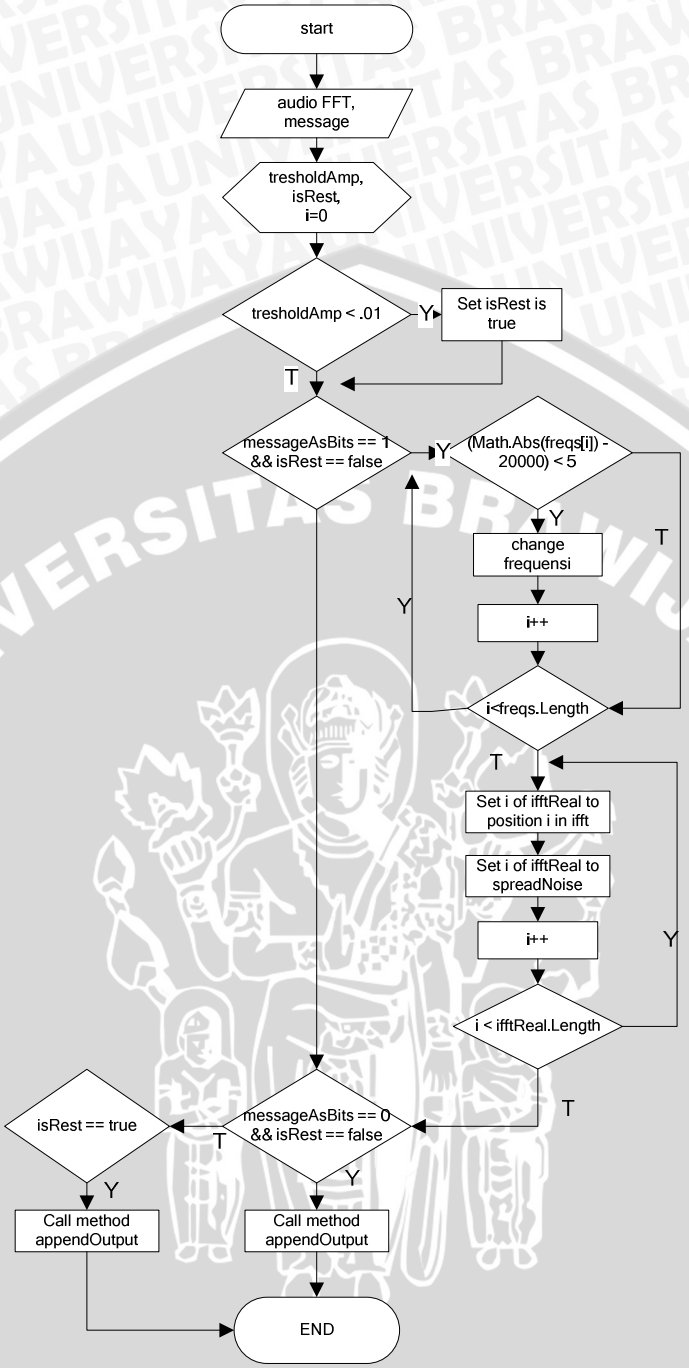
Gambar 4.7 *Flowchart* Transformasi FFT

Dengan algoritma sebagai berikut :

```
int N = x.Length;

if (N == 1) return new Complex[] { x[0] };

if (N % 2 != 0) { throw new ArithmeticException;
}

Complex[] even = new Complex[N/2];
for (int k = 0; k < N/2; k++) {
    even[k] = x[2*k];
}
Complex[] q = fft(even);

Complex[] odd  = even;
for (int k = 0; k < N/2; k++) {
    odd[k] = x[2*k + 1];
}
Complex[] r = fft(odd);

Complex[] y = new Complex[N];
for (int k = 0; k < N/2; k++) {
    double kth = -2 * k * Math.PI / N;
    Complex wk = new Complex(Math.Cos(kth), Math.Sin(kth));
    y[k]       = q[k].plus(wk.times(r[k]));
    y[k + N/2] = q[k].minus(wk.times(r[k]));
}

return y;
```

4.2.2.3 Perancangan Proses Penyisipan Pesan

Pada proses penyisipan pesan, pesan akan disisipkan pada berkas audio dengan cara mengabungkannya dengan frekuensi dari berkas audio tersebut. Penyisipan ini dilakukan pada ranah frekuensi dengan tujuan agar pesan lebih tahan terhadap serangan. Yang setelah disisipkan berkas audio akan dikembalikan ke domain waktu kembali sehingga berkas audio kembali seperti semula dengan pesan yang sudah disisipkan di dalamnya. Berikut adalah *flowchart* dari proses tersebut:



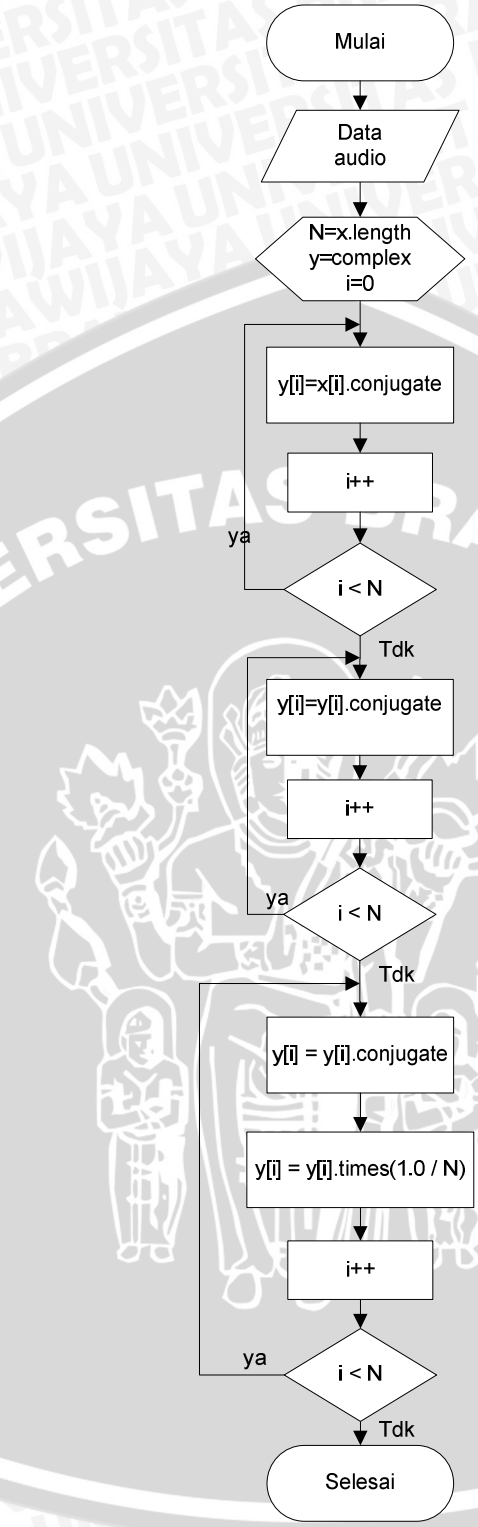
Gambar 4.8 *Flowchart* Penyisipan Pesan

Dengan algoritma sebagai berikut :

```
        Boolean isRest = false;
                if (tresholdAmp < .01)
                {
                    isRest = true;
                }
                if (messageAsBits[currentBit] == 1 && isRest == false)
                {
                    for (int i = 0; i < freqs.Length; i++)
                    {
                        if (Math.Abs(Math.Abs(freqs[i]) - 20000) < 5)
                        {
                            complexMags[i] = new FFT.Complex(Alpha *
channelData.Length, 0);
                        }
                    }
                    FFT.Complex[] ifft = FFT.FFT.ifft(complexMags);
                    double[] ifftReal = new double[ifft.Length];
                    double[] spreadNoise = randGenerator(Alpha, Cr,
totalBytes, ifft.Length); ;
                    for (int i = 0; i < ifftReal.Length; i++)
                    {
                        ifftReal[i] = ifft[i].getReal();
                        ifftReal[i] += spreadNoise[i];
                     }
                    appendOutput(getAllSamples(ifftReal), bytesRead -
bytesToRead, aout);

                    currentBit++;
                }
                else if (messageAsBits[currentBit] == 0 && isRest == false)
                {
                    appendOutput(samples, bytesRead - bytesToRead, aout);
                    currentBit++;
                }
                else if (isRest == true)
                {
                    appendOutput(samples, bytesRead - bytesToRead, aout);
                }
```

4.2.2.4 Perancangan Proses Transformasi IFFT

Proses *invers fast fourier transform* (IFFT) adalah sebuah proses kebalikan dari FFT yang berfungsi untuk mengembalikan berkas audio yang sebelumnya diubah ke domain frekuensi kembali menjadi domain waktu setelah dilakukan proses penyisipan pesan. Yang bertujuan agar berkas audio kembali seperti semula sehingga dapat di putar sebagaimana berkas audio pada umumnya. Berikut adalah *flowchart* dari proses tersebut:



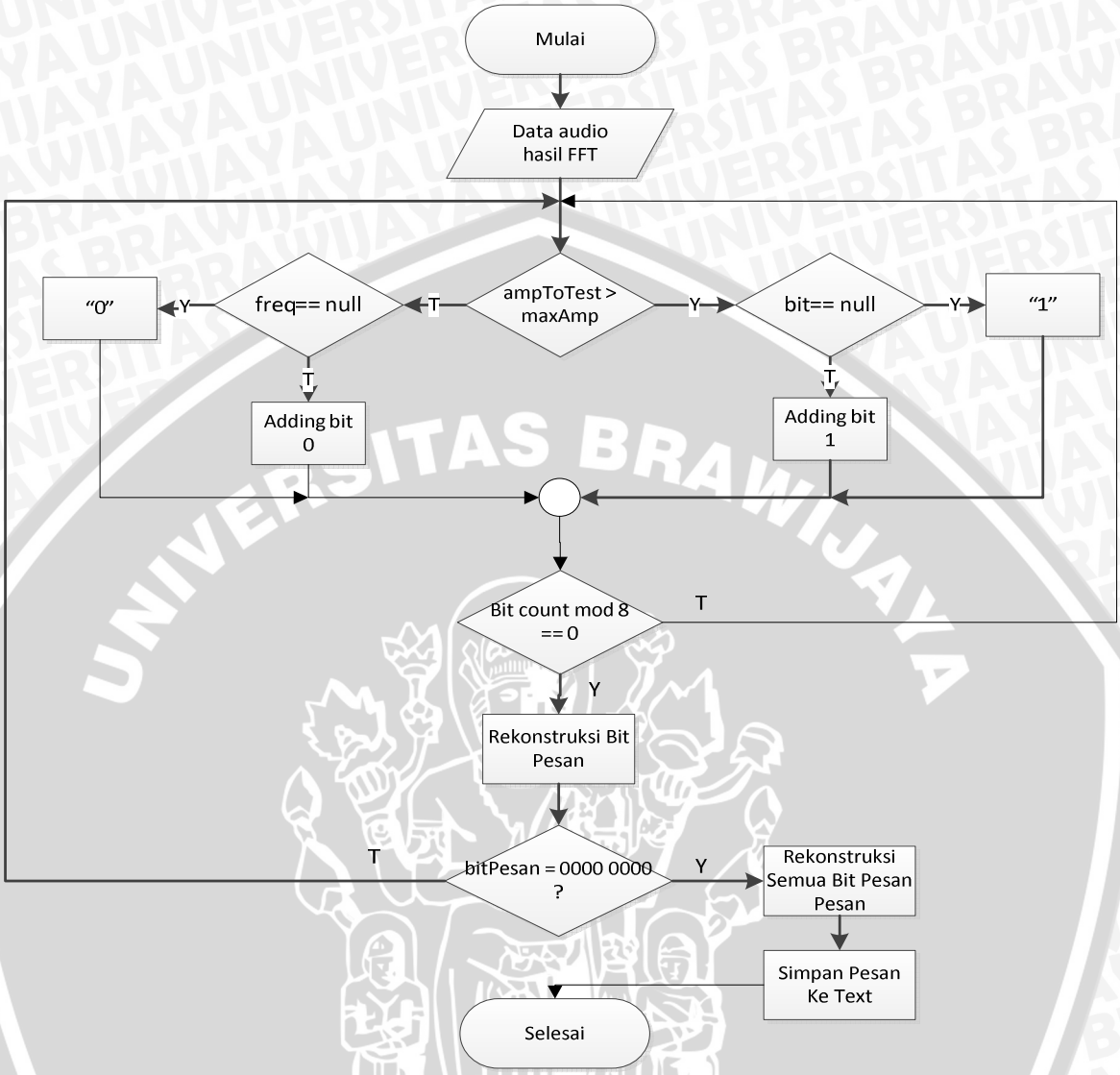
Gambar 4.9 *Flowchart* Transformasi IFFT

Dengan algoritma sebagai berikut :

```
for (int i = 0; i < N; i++) {
                y[i] = x[i].conjugate();
            }
            y = fft(y);
            for (int i = 0; i < N; i++) {
                y[i] = y[i].conjugate();
            }
            for (int i = 0; i < N; i++) {
               y[i] = y[i].conjugate();
                y[i] = y[i].times(1.0 / N);
            }

            return y;
```

4.2.2.5 Perancangan Proses Ekstraksi Pesan

Proses ekstraksi adalah proses dilakukannya pengambilan pesan dari *stego-object*. Yaitu pengambilan pesan yang sebelumnya disisipkan pada berkas audio. Proses ini dilakukan dengan sebelumnya mentransformasikan terlebih dahulu berkas audio ke domain frekuensi kemudian diambillah bagian noise yang merupakan pesan tersebut dan menyusun ulang kembali sehingga pesan dapat terbaca. *Flowchart* dari proses ekstraksi tersebut adalah sebagai berikut:



Gambar 4.9 *Flowchart* Ekstraksi Pesan

Dengan algoritma sebagai berikut :

```
        if (ampToTest > .009)
{
    if (messageAsBytes[currentCharIndex] == null)
    {
        messageAsBytes[currentCharIndex] = "1";
    }
    else
    {
        messageAsBytes[currentCharIndex] = "1" +
messageAsBytes[currentCharIndex];
    }
```

```
                    }
                    else
                    {
                        if (messageAsBytes[currentCharIndex] == null)
                        {
                            messageAsBytes[currentCharIndex] = "0";
                        }
                        else
                        {
                            messageAsBytes[currentCharIndex] = "0" +
messageAsBytes[currentCharIndex];
                        }
                    }
                    Console.WriteLine(messageAsBytes[currentCharIndex]);
                    bitsSaved++;
                    if (bitsSaved % 8 == 0)
                    {
                        if
(messageAsBytes[currentCharIndex].Equals("00000000"))
                        {
                            System.Console.WriteLine
                         break; //is done
                        }
```

4.3 Implementasi Sistem

Spread spectrum adalah teknik pembangkitan sinyal (elektrik, elektromagnetik atau akustik) yang dengan sengaja disebar pada rentang *bandwidth* yang lebih lebar dari yang seharusnya. Sehingga hal ini cocok digunakan untuk steganografi audio karena keamanannya dalam mengacak dan menyembunyikan pesan. Steganografi spread spectrum pada arsip audio ini akan diimplementasikan dengan skema sebagai berikut:

1. Mengubah data audio *cover-object* di *time-domain* ke *frequency-domain*
2. Menambahkan sinyal informasi dengan metode *spread spectrum* ke *cover-object frequency- domain*
3. Mengubah lagi data audio cover-object frequency- domain ke time-domain

4.4 Lingkungan Implementasi

Aplikasi dibuat menggunakan *Microsoft Visual Studio C#*. Sistem diimplementasikan dengan menggunakan spesifikasi sebagai berikut :

1. Perangkat Keras (Laptop)
   Spesifikasi :
   Processor : Intel core i5-3337U 1.8Ghz
   Memory : 4 GB
   VGA : NVDIA GEFORCE 740M

2. Perangkat Lunak
   Sistem Operasi : *Microsoft Windows 8*
   Bahasa Pemrograman : *Microsoft Visual Studio C#.NET 2012*

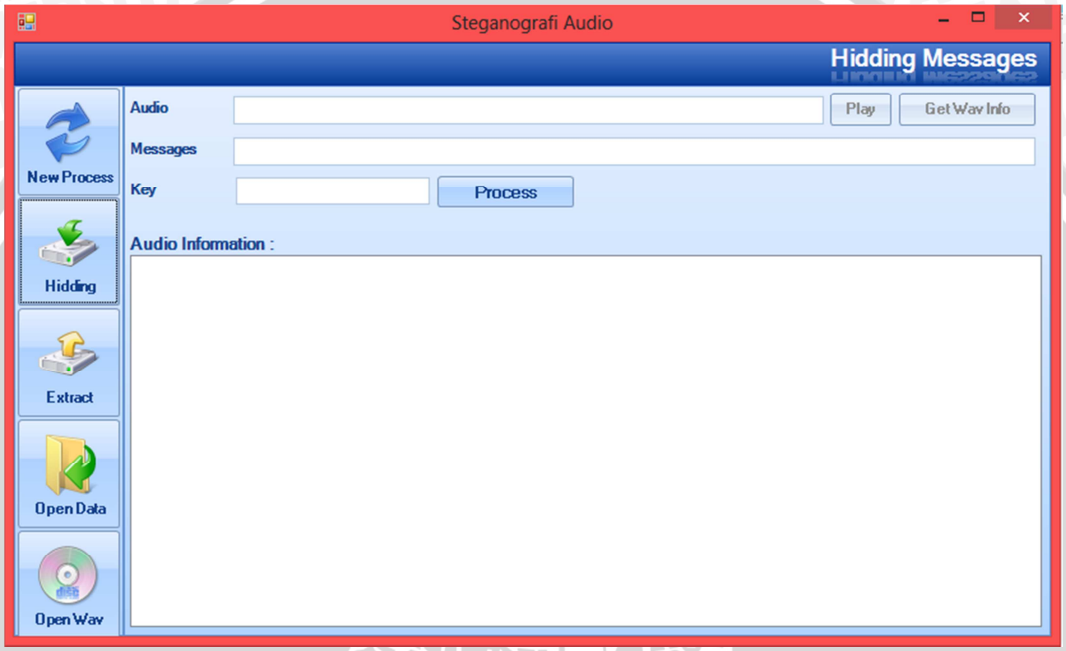
4.5 Implementasi *Interface* (Antarmuka)

Program steganografi pada citra menggunakan metode *Spread Spectrum* ini didesain untuk melakukan proses penyisipan pesan sesuai dengan langkah-langkah yang berurutan. Hal ini bertujuan agar pemakai tidak mengalami kesulitan dalam pengoperasian program ini. Pada tampilan utama program ini memiliki 8 *button* yang memiliki fungsi yang berbeda-beda. Selain itu pada tampilan utama program terdapat tiga *box* yang berguna untuk menampilkan audio masukan, pesan, dan info spesifikasi audio.

4.5.1 Implementasi Antarmuka Tampilan Penyisipan Pesan

Pada saat aplikasi ini dijalankan, maka akan muncul jendela sebagaimana pada Gambar 4.11 . Dengan menekan tombol “Open Wav” maka akan muncul *file browser* untuk memilih masukan berkas audio, demikian pila apabila menekan tombol “Open Data” ,maka akan muncul *file browser* untuk memlih masukan pesan. Setelah kedua masukan ini dimasukan, pengguna dapat memasukan *key* untuk melanjutkan proses penyisipan. Pengguna dapat memasukan *key* pada

teksfield yang sudah tersedia. Dengan menekan *Get Wav Info*, pengguna dapat melihat informasi mengenai spesifikasi berkas audio, serta dapat melihat berapa kapasitas maksimum dari berkas audio untuk dapat menampung pesan rahasia yang akan disisipkan. Apabila semua proses sudah terpenuhi, maka pengguna dapat menyisipkan pesan dengan menekan tombol *process*. Untuk dapat mendengarkan suara dari berkas audio, pengguna dapat menekan tombol *Play* dan untuk menghentikannya pengguna dapat menekan tombol *Stop*.

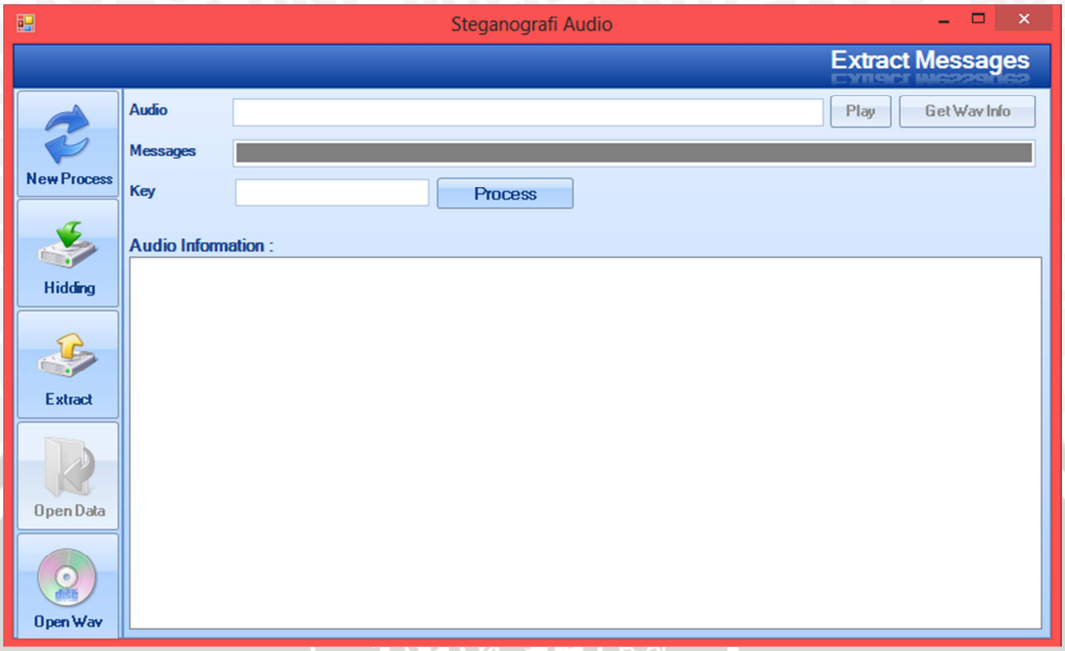


Gambar 4.10 Tampilan Penyisipan Pesan

4.5.2 Implementasi Antarmuka Tampilan Ekstraksi Pesan

Pada antarmuka “Extract” hanya terdapat audio sebagai masukan. Tombol “Open Wav” akan menampilkan *file browser* yang akan digunakan untuk memilah berkas audio sebagai masukan. Setelah berkas audio dimasukan, pengguna dapat memasukan *key* pada *teksfield* yang sudah disediakan. Proses ekstraksi pesan dapat dilakukan oleh pengguna dengan menekan tombol *process*. Dengan menekan tombol *Get Wav Info*, maka pengguna akan mendapat informasi mengenai spesifikasi berkas audio yang dimasukan. Untuk mendengarkan suara dari berkas audio yang telah dimasukan, penguna dapat

menekan tombol *Play* dan untuk menghentikannya pengguna dapat menekan tombol *Stop*. Antarmuka pada menu *extrack* ditunjukan pada Gambar 4.12.



Gambar 4.11 Tampilan Ekstraksi Pesan.