BAB V

PENGUJIAN

Pada tugas akhir ini, dilakukan pengujian kebenaran perangkat lunak, mencakup proses penyisipan pesan dan ekstraksi pesan, serta pengujian kinerja perangkat lunak, yaitu dengan melakukan perbandingan terhadap kualitas berkas audio hasil disisipi pesan dan berkas audio sebelum disisipi pesan. Berikut ini akan dijelaskan mengenai lingkungan pengujian, kasus uji, hasil pengujian dan kesimpulan hasil pengujian.

5.1 Lingkungan Pengujian

Lingkungan pengujian perangkat lunak ini, memiliki spesifikasi yang sama dengan lingkungan implementasi perangkat lunak yang telah dijelaskan pada sub bab 4.4 .

5.2 Tujuan Pengujian

Terdapat beberapa hal yang merupakan tujuan dari pengujian perangkat lunak dalam tugas akhir ini, yaitu :

1. Menguji kebenaran proses penyisipan dan ekstraksi berkas audio.
2. Mengukur kinerja dan kualitas pada berkas audio setelah disisipi pesan.

5.3 Perancangan Kasus Uji

Rancangan kasus uji dibuat dibuat agar pengujian dapat terstruktur sehingga tujuan pengujian dapat tercapai.

5.3.1 Kasus Uji Kebenaran Perangkat Lunak

Kasus uji ini dibuat untuk membuktikan kebenaran dan kesesuaian antara perangkat lunak yang dibangun dengan spesifikasi kebutuhannya secara

fungsional. Rancangan pengujian uji kebenaran perangkat lunak ini adalah sebagai berikut :

1. Penyisipan berkas data ke dalam berkas audio .WAV .
2. Ekstraksi berkas audio .WAV yang telah disisipi data untuk mendapatkan kembali data yang valid.
3. Pemutaran berkas audio sebelum disisipi dan sesudah disisipi data.

5.3.2 Kasus Uji Kinerja Perangkat Lunak

Pengujian kinerja perangkat lunak dilakukan dengan variasi ukuran dari berkas data yang akan disisipkan pada berkas audio .WAV yang sama, dan juga sebaliknya yaitu dengan variasi ukuran dari berkas audio yang akan disisipi dengan berkas data yang sama. Dalam pengujian ini variasi ukuran berkas data dibatasi oleh kapasitas dari kemampuan *stego-object* dalam menampung berkas data tersebut. Pengujian ini dilakukan untuk mengetahui kinerja perangkat lunak melalui lama waktu proses penyisipan dan ekstraksi pesan. Dan juga melalui kompleksitas algoritma perangkat lunak.

5.3.3 Kasus Uji Kualitas Audio Hasil Penyisipan

Pengujian kualitas audio hasil penyisipan dilakukan dengan melakukan penilaian secara subjektif yaitu dengan cara mendengarkan suara hasil pemutaran berkas audio .WAV. dan pengujian objektif dengan melakukan perhitungan nilai PSNR nya. Pengujian ini dilakukan untuk mengetahui kualitas berkas audio .WAV yang telah disisipi berkas data.

5.4 Data Pengujian

Pada Tabel 5.1 menunjukn berkas audio yang digunakan dalam perangkat lunak ini. Sedangkan pada Tabel 5.2 menunjukan berkas data yang digunakan dalam pengujian perangkat lunak ini.

Tabel 5.1 Berkas Audio

No	Nama Audio .WAV	Kapasitas	Durasi
1	Citra Scholastika - Pasti Bisa	10.1 MB	00:01:00
2	Akuistik	4.45 MB	00:00:26
3	Reason__Instrumental	15.4 MB	00:01:32
4	Aku yang terindah	20 MB	00:01:59

Tabel 5.2 Berkas Data

No	Nama Berkas Data .txt	Isi Pesan yang Disisipkan	Kapasitas (bytes)
1	tes 1	Sebuah kapal FERY BAHARI tenggelam 46 januari	45
2	tes 2	ABCDEFGHIJKL.,;[] 1234567890/=ABC35	35
3	tes 3	ABCDEFGHIJKL 1234567890/=ABC30	30
4	tes 4	ABCDE 7890/=AfC.7q20	20
5	tes 5	123ABCDEFG	10

5.5 Pelaksanaan dan Hasil Pengujian

Pada sub bab ini dijelaskan mengenai pelaksanaan pengujian dan hasil pengujian setiap kasus uji yang telah dijelaskan pada sub bab 5.3 . Dan data pengujian dapat dilihat pada sub bab 5.4 .

5.5.1 Pengujian Kebenaran Perangkat Lunak

Pengujian kebenaran perangkat lunak dilakukan secara berurutan , dimulai dengan menjalankan aplikasi. Pengujian yang dilakukan adalah sebagai berikut :

1. Pemutaran berkas audio .WAV sebelum disisipi berkas data. Langkah pengujiannya adalah sebagai berikut :
    a. Masukan berkas audio .WAV yang akan disisipi berkas data.
    b. Putar berkas audio .WAV .

2. Penyisipan sebuah berkas data ke dalam sebuah berkas nerkas audio .WAV . langkah pengujiannya adalah sebagai berikut :
    a. Masukkan berkas data yang akan disisipkan.
    b. Masukkan kunci
    c. Lakukan proses penyisipan.
3. Pemutaran berkas audio .WAV setelah disisipi berkas data. Langkah pengujiannya adalah sebagai berikut :
    a. Masukkan berkas berkas audio .WAV yang telah disispi berkas data.
    b. Putar berkas audio .WAV .
4. Ekstraksi berkas data dari berkas audio .WAV yang telah disisipi. Langkah pengujiannya adalah sebgai berikut :
    a. Masukkan kunci yang sama dengan kunci saat melakukan penyisipan.
    b. Lakukan proses ekstraksi.

Hasil pengujian kebenaran perangkat lunak dapat dilihat pada Tabel 5.3 . Keberhasilan proses pemutaran berkas audio .WAV dinilai dari dapat tidaknya berkas audio tersebut diputar. Keberhasilan proses ekstraksi dinilai dari kesesuaian antara berkas data sebelum disisipkan dengan berkas data hasil ekstraksi.

Tabel 5.3 Proses Penyisipan Variasi Berkas Audio

No	Nama Berkas Data .txt	Isi Pesan yang Disisipkan	Nama Audio .WAV	Penyisipan Pesan
1	tes 1	Sebuah kapal FERY BAHARI tenggelam 46 januari	Akuistik	berhasil
2	tes 1		Citra Scholastika - Pasti Bisa	berhasil
3	tes 1		Reason__Instrumental	berhasil
4	tes 1		Aku yang terindah	berhasil

Tabel 5.4 Proses Penyisipan Variasi Berkas Data

No	Nama Berkas Data .txt	Isi Pesan yang Disisipkan	Nama Audio .WAV	Penyisipan Pesan
1	tes 1	Sebuah kapal FERY BAHARI tenggelam 46 januari	Citra Scholastika - Pasti Bisa	berhasil
2	tes 2	ABCDEFGHIJKL.,;[] 1234567890/=ABC35		berhasil
3	tes 3	ABCDEFGHIJKL 1234567890/=ABC30		berhasil
4	tes 4	ABCDE 7890/=AfC.7q20		berhasil

Tabel 5.5 Proses Ekstraksi Variasi Berkas Audio

No	Nama Audio .WAV	Isi Pesan yang Disisipkan	Isi Pesan Hasil Ekstraksi	Ekstraksi Pesan
1	Akuistik-hasil	Sebuah kapal FERY BAHARI tenggelam 46 januari	Sebuah kapal FERY BAHARI tenggelam 46 januari	berhasil
2	Citra Scholastika - Pasti Bisa-hasil		Sebuah kapal FERY BAHARI tenggelam 46 januari	berhasil
3	Reason__Instrumental-hasil		Sebuah kapal FERY BAHARI tenggelam 46 januari	berhasil
4	Aku yang terindah-hasil		Sebuah kapal FERY BAHARI tenggelam 46 januari	berhasil

Tabel 5.6 Proses Ekstraksi Variasi Berkas Data

No	Isi Pesan yang Disisipkan	Nama Audio .WAV	Hasil Ekstrasksi Pesan	Keterangan
1	Sebuah kapal FERY BAHARI tenggelam 46 januari	Citra Scholastika - Pasti Bisa	Sebuah kapal FERY BAHARI tenggelam 46 januari	berhasil
2	ABCDEFGHIJKL.,;[] 1234567890/=ABC35		ABCDEFGHIJKL.,;[] 1234567890/=ABC35	berhasil
3	ABCDEFGHIJKL 1234567890/=ABC30		ABCDEFGHIJKL 523<567890/=ABC30	Dua angka berubah
4	ABCDE 7890/=AfC.7q20		ABCDE 7890/=AfC.7q20	berhasil

Tabel 5.7 Hasil Pemutaran Berkas Audio

No	Nama Audio .WAV Sebelum disisipi	Nama Audio .WAV Setelah Disisipi	Di mainkan
1	Akuistik	Akuistik-hasil	bisa
2	Citra Scholastika - Pasti Bisa	Citra Scholastika - Pasti Bisa-hasil	bisa
3	Reason__Instrumental	Reason__Instrumental-hasil	bisa
4	Aku yang terindah	Aku yang terindah-hasil	bisa

5.5.2 Pengujian Kinerja Perangkat Lunak

Pengujian kinerja menggunakan data pengujian yang telah dijelaskan pada sub bab 5.4 . pengujian kinerja perangkat lunak diukur dari :

1. Waktu penyisipan berkas data.
2. Waktu ekstraksi berkas data.
3. Kompleksitas algoritma perangkat lunak.

Hasil dari pengujian dapat dilihat pada tabel-tabel berikut.

Tabel 5.8 Waktu Proses Penyisipan dengan Variasi Berkas Audio

No	Kapasitas	Nama Audio .WAV	Pesan	Waktu Proses (detik)
1	4.45 MB	Akuistik	Tes 2	27:02
2	10.1 MB	Citra Scholastika - Pasti Bisa	Tes 2	32:06
3	15.4 MB	Reason__Instrumental	Tes 2	37:04
4	20 MB	Aku yang terindah	Tes 2	42:06

Tabel 5.9 Waktu Proses Penyisipan dengan Variasi Berkas Data

No	Kapasitas (bytes)	Nama Berkas Data .txt	Nama Audio	Waktu Proses (detik)
1	20	tes 1	Citra Scholastika - Pasti Bisa	25:02
2	30	tes 2		32:06
3	35	tes 3		36:00
4	46	tes 4		43:07

Tabel 5.10 Waktu Proses Ekstraksi dengan Variasi Berkas Audio

No	Nama Berkas Data .txt	Nama Audio .WAV	Waktu Proses (detik)
1	tes 2	Akuistik-hasil	12:05
2	tes 2	Citra Scholastika - Pasti Bisa-hasil	17:16
3	tes 2	Reason__Instrumental-hasil	21:09
4	tes 2	Aku yang terindah-hasil	25:05

Tabel 5.11 Waktu Proses Ekstraksi dengan Variasi Berkas Data

No	Nama Berkas Data .txt	Nama Audio .WAV	Waktu Proses (detik)
1	tes 1	Citra Scholastika - Pasti Bisa-hasil	17:06
2	tes 2	Citra Scholastika - Pasti Bisa-hasil	19:00
3	tes 3	Citra Scholastika - Pasti Bisa-hasil	21:05
4	tes 4	Citra Scholastika - Pasti Bisa-hasil	15:00

5.5.2.1 Pengujian Kompleksitas Algoritma

a. Algoritma Proses FFT

```
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;
```

Kompleksitas algoritma = O (N log N)

b. Algoritma IFFT

```
      public static Complex[] ifft(Complex[] x) {
      int N = x.Length;
      Complex[] y = new Complex[N];


      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;
```

Kompleksitas algoritma = O (N^3)

c. Algoritma *Random Generator* (LCG)

```
            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_;
        }
```

Kompleksitas algoritma = O (N)

d. Algoritma Penyisipan Pesan

```
                double tresholdAmp = 0;

                for (int i = 0; i < freqMag.GetLength(0); i++)
                {
                    if (   Math.Abs(freqMag[i,1]) > tresholdAmp)
                    {
                        tresholdAmp = freqMag[i,1];
                    }

                }
                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);
                }
```

Kompleksitas algoritma = O (N) + (N^3)

e. Algoritma Ekstraksi Pesan

```
                double tresholdAmp = 0;
                for (int i = 0; i < freqMag.GetLength(0); i++)
                {
                    if (Math.Abs(freqMag[i, 1]) > tresholdAmp)
                    {
                        tresholdAmp = freqMag[i, 1];
                    }
                }
                Boolean isRest = false;
                if (tresholdAmp < .01)
                {
                    isRest = true;
                }

                double ampToTest = 0;
                if (!isRest)
                {
                    for (int i = 0; i < freqMag.GetLength(0); i++)
                    {
                        if (Math.Abs(Math.Abs(freqMag[i, 0]) - 20000) < 5)
                        {
                            ampToTest = freqMag[i, 1];
                        }
                    }
                }

                if (!isRest)
```

```
                {

                    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;                          }
                        currentCharIndex++;
                    }
```

Kompleksitas algoritma = O (N) + (N^3)

Sehingga jumlah kompleksitas algoritma adalah :

O (N log N) + O (N^3) + (O (N) + (O (N) + (N^3)) + (O (N) + (N^3)) =

N log N + 3(N) + 3(N^3)

5.5.3 Pengujian Kualitas Audio Hasil Penyisipan

Pengujian kualitas audio hasil penyisipan menggunakan data pengujian yang telah dijelaskan pada sub bab 5.4 . pengujian ini dilakukan dengan cara melakukan penilaian. Penilaian kualitas berkas audio .WAV tersebut dilakukan melalui dua hal, yaitu :

1. Penilaian subjektif dengan cara mendengarkan suara hasil pemutaran berkas audio .WAV .
2. Menghitung nilai PSNR *(Peak Signak to Noise Ratio)* . nilai PSNR dalam satuan desibel(dB) dihitung dengan rumus :

$$PSNR = 10 \log_{10} \left(\frac{P_1^2}{P_1^2 + P_0^2 - 2P_1P_0} \right) \quad \textbf{5.1}$$

Dengan P0 menyatakan kekuatan sinyal awal dan P1 menyatakan kekuatan sinyal setelah disisipi data. P0 dan P1 diukur dalam satuan deisbel(dB).

Tabel 5.12 Nilai PSNR

No	*Size audio* (MB)	Durasi (menit)	*Size* pesan (*bytes*)	PSNR (dB)
1	10.1	01:00	46	44.13
2			35	44.13
3			30	44.13
4			20	50.18
5			10	50.18

5.6 Analisis Hasil Pengujian

Berikut ini adalah analisis hasil uji dari seluruh hasil pengujian yang telah dilakukan.

5.6.1 Analisis Hasil Uji Kebenaran Perangkat Lunak

Perangkat lunak yang diimplementasikan telah sesuai dengan spesifikasi kebutuhan perangkat lunak yang telah dijelaskan sebelumnya. Hal ini ditunjukan dengan keberhasilan dalam pengujian kebenaran perangkat lunak dalam melakukan penyisipan dan ekstraksi pesan, serta pemutaran berkas audio .WAV saat sebelum dan sesudah penyisipan. Dimana dari hasil pengujian yang dilakukan pada empat berkas audio dengan kapasitas dan jenis lagu yang berbeda disisipi pesan yang sama, pada tabel 5.3 sedangkan pada tabel 5.4 pengujian dilakukan dengan variasi berkas data yang berbeda, menunjukan hasil bahwa proses penyisipan pesan berhasil dilakukan.

Pada tabel 5.5 pengujian proses ekstraksi pesan dilakukan dengan variasi audio dengan pesan yang disisipkan sam menunjukan hasil ekstraksi yang sama sebagaimana pesan aslinya. Sedangkan pada tabel 5.6 pengujian dilakukan dengan variasi berkas data menunjukan hampir keseluruhan hasil ekstraksi pesan sama sebagaumana pesan aslinya, hanya saja pada berkas data nomor tiga terjadi sedikit perubahan pada dua angka yang diberi tanda merah.

Setelah dilakukan beberapa pengujian dan analisa, kesalahan tersebut dapat terjadi karena faktor dari posisi pesan, yaitu tempat dimana disisipkannya pesan pada berkas audio (*stego-object*). Dimana pada posisi tertentu tersebut ketika dilakukan proses penyisipan, pesan terletak pada berkas audio dengan frekuensi yang rendah, sehingga ketika dilakukan proses ekstraksi dari berkas audio tersebut, pesan tidak ter-ekstraksi dengan benar sehingga menyebabkan nilai dari pesan berubah dari pesan aslinya. Hal ini bisa terjadi dikarenakan pada saat dilakukan proses *invers* (IFFT) pada pesan yang tersisip pada frekuensi rendah tersebut, proses IFFT tidak berjalan dengan sempurna sehingga menyebabkan adanya *flooting point* yang menyebabkan berubahnya isi pesan.

Selanjutnya dilakukan pengujian pemutaran berkas audio saat sebelum penyisipan maupun sesudah penyisipan, dari hasilpengujian tersebut pemutaran berkas audio berhasil dilakukan dengan baik dan benar secara keseluruhan pada ke empat jenis berkas audio yang berbeda tersebut.

5.6.2 Analisis Hasil Uji Kinerja Perangkat Lunak

Hasil pengujian kinerja perangkat lunak menunjukkan hasil yang cukup baik. Dimana dalam hal ini kecepatan penyisipan dan ekstraksi data bergantung pada ukuran berkas data dan berkas audio. Pada tabel 5.8 pengujian dilakukan dengan variasi berkas audio, terlihat bahwa lama waktu proses penyisipan dipengaruhi oleh besarnya berkas audio yang digunakan sebagai media penyisipan, dimana semakin besar kapasitas berkas audio maka semakin lama waktu yang dibutuhkan untuk proses penyisipan. Sedangkan pada tabel 5.9 pengujian dilakukan dengan variasi berkas data, terlihat bahwa semakin besar kapasitas berkas data yang akan disisipkan maka waktu proses penyisipan pun membutuhkan waktu yang lebih lama.

Sedangkan pengujian proses ekstraksi pada tabel 5.10 dan 5.11 juga menunjukan kesimpulan yang sama, bahwa lama waktu proses ekstraksi pesan dipengaruhi oleh besarnya kapasitas berkas audio maupun berkas data. Namun rata-rata waktu yang dibutuhkan untuk ekstraksi pesan hanya separuh dari waktu penyisipan pesan. Sehingga dari hasil pengujian tersebut lama waktu proses penyisipan dan ekstraksi masih praktis dan layak untuk digunakan.

5.6.3 Analisis Hasil Uji Kualitas Audio Hasil Penyisipan

Hasil pengujian tersebut menunjukan bahwa kualitas berkas audio .WAV sangat bergantung pada ukuran berkas data yang disisipkan, sehingga bisa disimpulkan bahwa semakin besar data yang disisipkan maka semakin buruk kualitas berkas audio .WAV . Tabel 5.12 menunjukan hasil pengujian kualitas berkas audio .WAV yang telah disisipi. Dari tabel tersebut dapat di ambil kesimpulan bahwa menurunnya nilai PSNR seiring bertambahnya ukuran data yang disispkan , akan tetapi tidak terlalu terjadi penurunan signifikan karena ukuran data yang disispkan tidak terlampau besar. Dari pengujian tersebut terlihat nilai PSNR masih dalam batas yang wajar bagi indera manusia. Yang mana indera manusia hanya akan mendengar *noise* pada audio namun tidak sampai merusak dari suara audio tersebut.