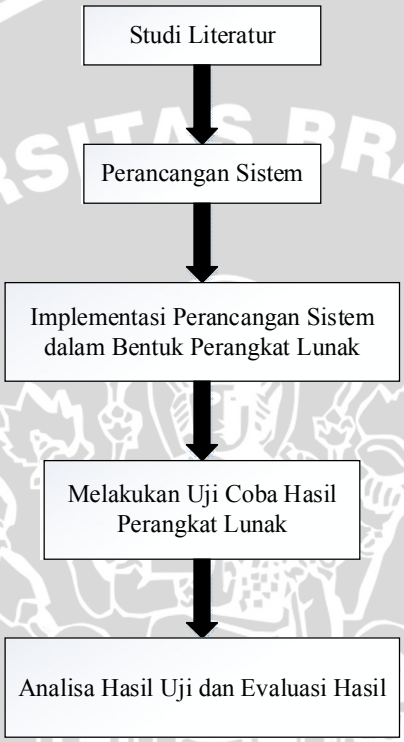
BAB III
METODOLOGI DAN PERANCANGAN SISTEM

Pada bab metodologi dan perancangan sistem ini berisi penjelasan metode, rancangan sistem, dan langkah-langkah yang dilakukan dalam penelitian ini. Langkah-langkah yang dilakukan ditunjukkan pada gambar 3.1



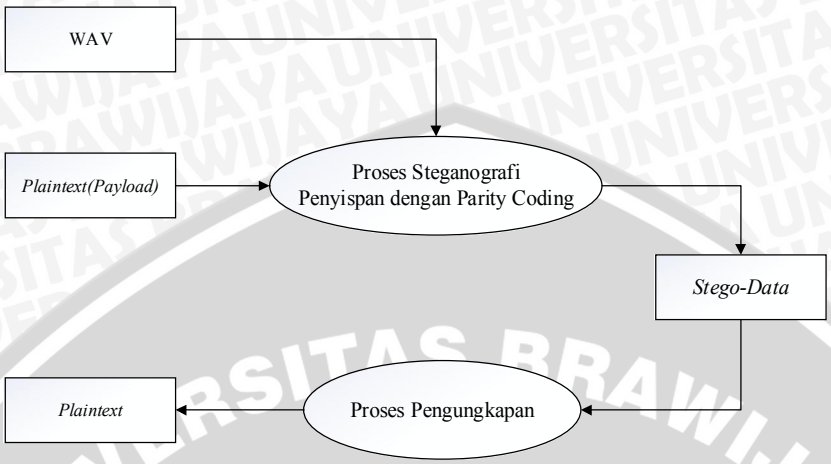
Gambar 3.1 Langkah – langkah penelitian

Berdasarkan gambar 3.1, langkah-langkah yang dilakukan dalam penelitian ini adalah sebagai berikut :

1. Melakukan studi literatur mengenai penyisipan pesan ke dalam berkas WAV melalui teknik *parity coding* dan karakteristik berkas WAV.
2. Melakukan perancangan sistem dengan membuat diagram alir sebagai pedoman bagaimana sistem berjalan serta melakukan simulasi secara sederhana bagaimana metode *parity coding* bekerja.
3. Mengimplementasikan perancangan sistem yang telah dibuat ke dalam bentuk perangkat lunak yang dapat menyisipkan pesan ke dalam berkas WAV.
4. Melakukan uji coba hasil perangkat lunak yang telah dibuat.
5. Menganalisa hasil uji coba dan mengevaluasi hasil.

3.1 Skema Umum Steganografi

Skema umum proses steganografi ditunjukkan pada gambar 3.2



Gambar 3.2 Skema umum proses steganografi

Dari gambar 3.2 diperlihatkan bahwa terdapat 2 komponen penting untuk dapat melakukan proses steganografi, yaitu *payload* dan berkas WAV. Kedua komponen tersebut dimasukkan ke dalam sistem untuk menyisipkan *payload* ke dalam berkas audio WAV yang akan menghasilkan berkas *stego-data*. Untuk mendapatkan pesan yang telah disisipkan ke dalam audio, *stego-data* dimasukkan ke dalam sistem dan dilakukan proses pengungkapan. Proses pengungkapan ini menghasilkan pesan yang telah disisipkan sebelumnya.

3.2 Deskripsi Umum Sistem

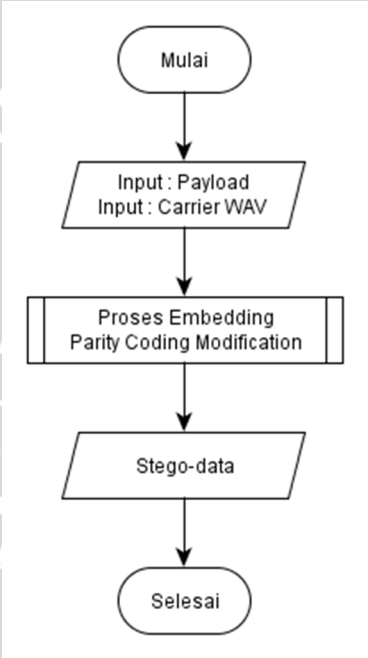
Perangkat lunak yang akan dibuat merupakan perangkat lunak yang dapat menyisipkan pesan berupa teks ke dalam berkas audio dengan format WAV. Perangkat lunak akan mengimplementasikan metode steganografi *parity coding* untuk menyisipkan *payload* ke dalam berkas audio WAV. Metode penyisipan *parity coding* ini memanfaatkan *parity* bit dari setiap *region* yang terbentuk dalam berkas WAV. Pengungkapan kembali pesan dari berkas WAV dilakukan dengan menyusun kembali nilai *parity* bit dari setiap *region*. *Input*-an pada perangkat lunak ini berupa berkas teks dan berkas audio WAV. Hasil *output* perangkat lunak ini berupa berkas audio WAV (*stego-data*) yang telah disisipkan pesan di dalamnya.

3.3 Perancangan Sistem

Terdapat dua proses utama dalam penelitian ini yaitu proses penyisipan pesan dan proses pengungkapan pesan.

3.3.1 Penyisipan Pesan

Diagram alir penyisipan *payload* ke dalam berkas audio WAV ditunjukkan pada gambar 3.3



Gambar 3.3 Diagram alir penyisipan pesan pada berkas audio WAV

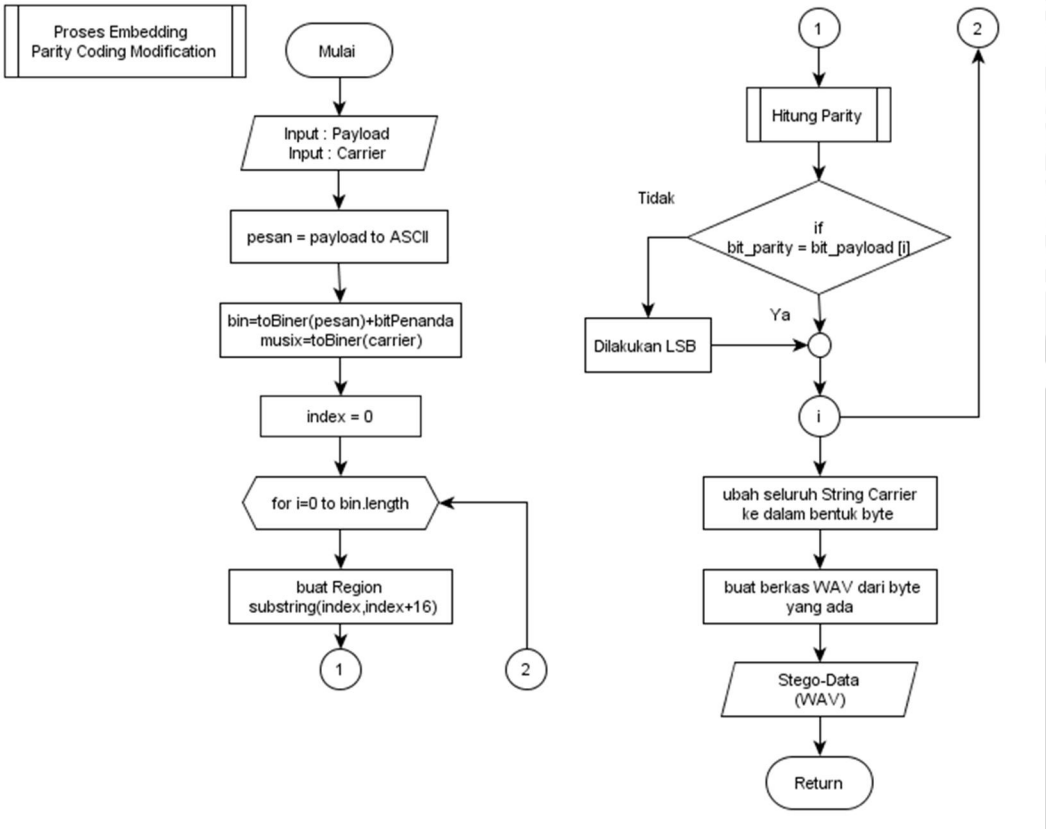
Berikut ini adalah penjelasan proses penyisipan *payload* ke dalam berkas audio

1. *Payload* berupa teks dan *carrier* yang merupakan berkas audio WAV dimasukkan ke dalam sistem.
2. Sistem akan melakukan proses *embedding parity coding modification* pada *payload* dan *CarrierWAV*.
3. Sistem akan menghasilkan berkas berkas audio (*stego-data*) yang sudah terdapat pesan di dalamnya.

3.3.1.1 Proses *Embedding* dengan *Parity Coding Modification*

Proses *embedding* merupakan proses yang sangat penting dalam penyisipan *payload* ke dalam berkas audio WAV. Proses penyisipan ini

menggunakan metode *parity coding*, yaitu metode yang membandingkan antara bit pesan dengan bit *parity* dari region yang terbentuk dari bit-bit berkas audio. Diagram alir proses *embedding* ditunjukkan pada gambar 3.4



Gambar 3.4 Diagram alir proses *embedding*

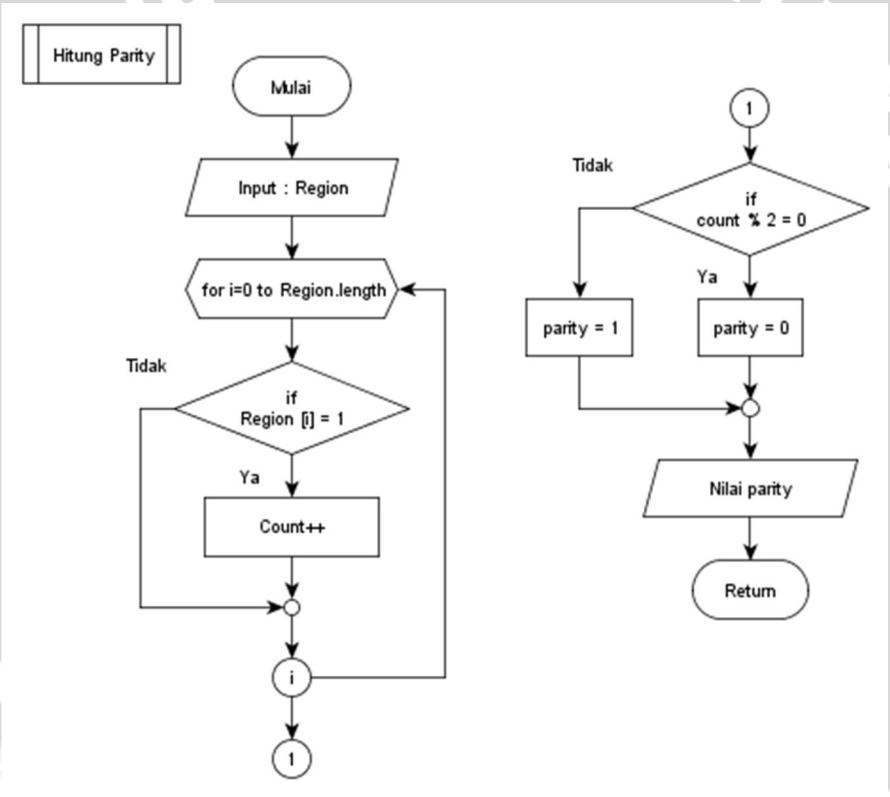
Berikut ini adalah penjelasan diagram alir proses *embedding*

1. *Payload* dan *carrier* yang merupakan berkas audio WAV dimasukkan ke dalam sistem.
2. Setiap karakter *payload* dan *bitPenanda* diubah ke dalam bentuk ASCII.
3. ASCII dari *payload* dan diubah ke dalam bentuk biner.
4. Dilakukan perulangan sampai dengan panjang seluruh *bitPayload*. Kemudian, sistem akan membuat region sebanyak 16 bit dari *bitCarrier*. Region yang telah dibuat akan dicari nilai *parity*-nya dan nilai *parity*-nya akan dibandingkan dengan *bitPayload*, apabila *bitPayload* tidak sama dengan nilai *parity* dari region tersebut maka akan dilakukan perubahan pada bit terakhir (bit LSB) region. Sedangkan apabila nilai *parity*-nya sama dengan *bitPayload*, maka tidak terjadi perubahan.

5. Langkah keempat dilakukan sampai dengan panjang *bitPayload*.
6. *Carrier* yang dimodifikasi akan diubah kembali ke dalam bentuk *byte*. Selanjutnya, *byte-byte* tersebut diubah kembali menjadi berkas audio.
7. Hasil yang diperoleh adalah sebuah *stego-data* berekstensi WAV yang menyimpan pesan di dalamnya.

3.3.1.2 Proses Hitung Bit *Parity*

Proses ini merupakan subproses dari proses *embedding*. Proses ini digunakan untuk mencari nilai *parity* dari suatu region yang telah dibagi sebelumnya. Diagram alir proses hitung bit_parity ditunjukkan pada gambar 3.5



Gambar 3.5 Diagram alir proses hitung bit_parity

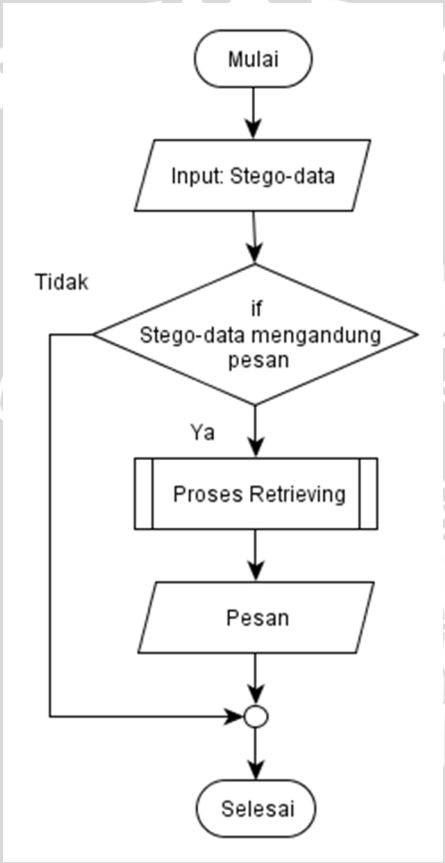
Berikut ini adalah penjelasan diagram alir proses hitung bit_parity

1. Region dimasukkan ke dalam sistem.
2. Dilakukan perulangan sampai dengan panjang bit suatu region.
3. Apabila bit-bit region terdapat nilai "1" maka akan ditambahkan secara *increment*, sehingga didapatkan jumlah total nilai "1" pada region tersebut.

4. Jika jumlah total nilai "1" pada region tersebut berjumlah genap, maka nilai *parity*-nya adalah 0, sedangkan apabila jumlah nilai "1" tersebut berjumlah ganjil, maka nilai *parity*-nya adalah 1.
5. Kembalikan nilai *parity* untuk diproses lebih lanjut.

3.3.2 Pengungkapan Pesan

Diagram alir pengungkapan pesan dari berkas audio WAV (*stego-data*) ditunjukkan pada gambar 3.6



Gambar 3.6 Diagram alir pengungkapan pesan dari berkas audio WAV

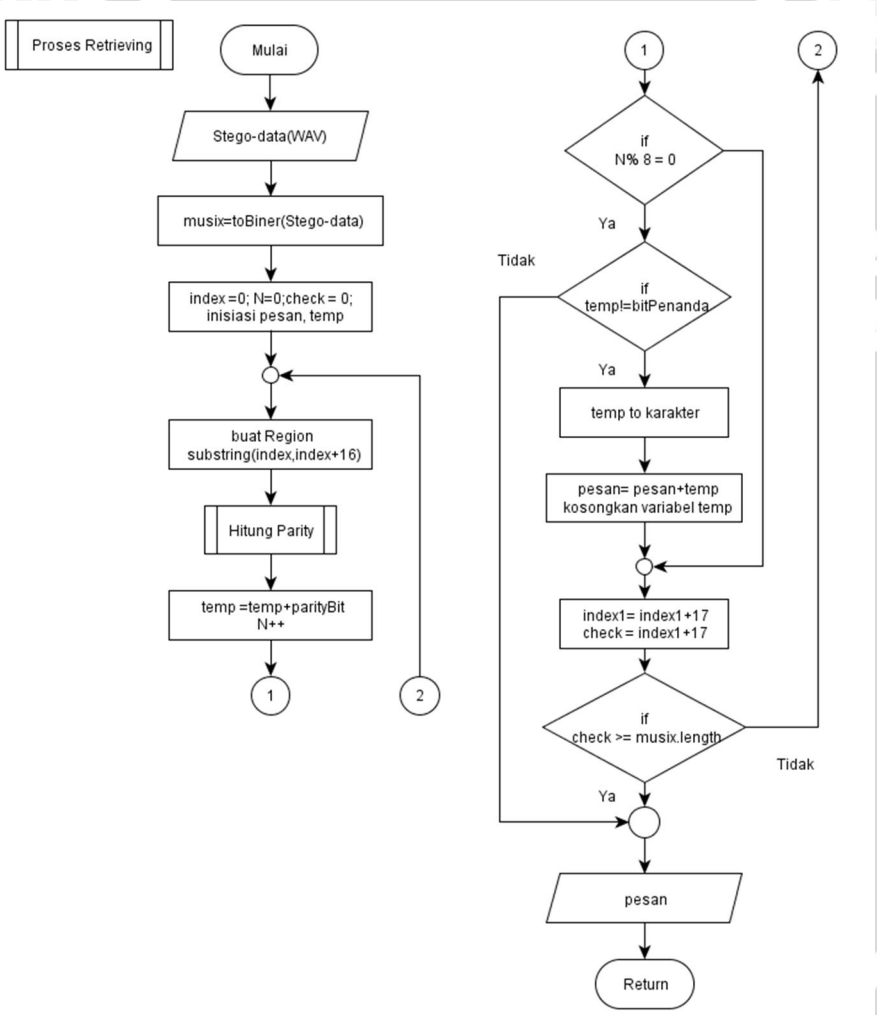
Berikut ini adalah penjelasan proses pengungkapan pesan dari berkas audio

1. Berkas audio WAV (*stego-data*) dimasukkan ke dalam sistem.
2. Dilakukan pengecekan apakah berkas audio yang dimasukkan terdapat pesan di dalamnya.

3. Jika terdapat pesan di dalam berkas audio, maka selanjutnya akan dilakukan proses *retrieving* yang mana akan mendapatkan kembali pesan yang disembunyikan.

3.3.2.1 Proses *Retrieving*

Retrieving adalah proses utama dalam pengungkapan kembali pesan. Diagram alir proses *retrieving* ditunjukkan pada gambar 3.7



Gambar 3.7 Diagram alir proses retrieving

Berikut ini adalah penjelasan proses *retrieving*

1. *Stego-data* berupa berkas audio WAV yang terdapat pesan di dalamnya, dimasukkan ke dalam sistem.
2. *Stego-data* diubah ke dalam bentuk biner.

3. Dilakukan perulangan sampai dengan panjang *bitStego-data*.
4. Sistem akan membuat region sebesar 16 bit dari *bitStego-data*. Kemudian nilai *parity* region dicari dan variabel *N* di *increment*.
5. Jika variabel *N* belum berjumlah 8, maka proses keempat dilakukan kembali sampai variabel *N* berjumlah 8 dan sebagai tanda bahwa *parity bit* dari region yang terbentuk sudah berjumlah 8.
6. Selanjutnya akan dilakukan pengecekan dari 8 bit yang terkumpul, apakah bit-bit tersebut merupakan bit dari *bitPenanda*. Jika bukan *bitPenanda* maka bit pesan tersebut ditampung di variabel pesan dan proses kembali ke langkah 4 dan langkah 5. Jika merupakan *bitPenanda* maka proses perulangan akan dihentikan dan pesan yang disisipkan didapatkan kembali.

Dalam proses *retrieving* terdapat subproses hitung bit_parity yang mempunyai alur yang sama dengan ketika proses *embedding* dan dapat dilihat pada gambar 3.5.

3.4 Perhitungan Manual

3.4.1 Penyisipan Pesan

Proses pertama pada penyisipan *payload* ke dalam berkas audio WAV adalah memasukkan *payload* berupa teks ke dalam sistem. *Payload* selanjutnya akan diubah ke dalam bentuk biner. Sebagai contoh *payload* yang akan disisipkan adalah sebuah kata yaitu "Juara". Berikut adalah representasi kata "Juara" yang diubah dalam bentuk biner ditunjukkan pada tabel 3.1

Tabel 3.1 Representasi *payload* dalam bentuk biner

Huruf	ASCII	Biner
J	74	0100 1010
u	117	0111 0101
a	97	0110 0001
r	114	0111 0010
a	97	0110 0001

Setelah *payload* diubah ke dalam bentuk biner, maka proses selanjutnya adalah memasukkan berkas audio WAV (*carrier*) ke dalam sistem dan mengubahnya ke

dalam bentuk biner. Dimulai pada *bytes* ke-44 pada WAV, dibentuk region-region sebanyak *bitPayload*. Berikut ini adalah contoh representasi berkas WAV *carrier* yang telah diubah menjadi biner dan dibagi dalam bentuk region-region :

1010111110001110 1110110110001110 0110110110001110
0001001011000001 0001001001100001 0010000100010010
0101100100000000 0111110101010101 0000000011000101
1101001101010101 0101000000000000 1100011110101010
0000111110001101 0110010110101010 0000100000001111
1110011010101010 0000111111111010 1110101011111111
1100111100111101 0000101011111111 0010000111001111
1111010011111111 0010100011001111 1100010001000111
0100011101101100 0100011101010110 0011111101100101
0101111010101010 1000000100111111 1010101001011001
0011111101110111 0111101110101010 1001001110011100
0111001010011100 1001101010011100 1111111110011111
1001111111111110 1001111111101110 1010111010001000
0010000010001000

Selanjutnya dari setiap region yang terbentuk akan dilakukan pencarian nilai *parity*-nya dengan cara *even parity*. Dikatakan *even parity* ketika jumlah bit '1' berjumlah genap maka nilai *parity*-nya "0". Sedangkan apabila jumlah bit '1' berjumlah ganjil maka nilai *parity*-nya "1". Kemudian, nilai *parity* akan dibandingkan dengan *bitPayload*. Apabila nilai *parity* berbeda dengan *bitPayload* maka akan dilakukan modifikasi pada LSB dari sampel region, sehingga nilai *parity* dari region tersebut akan sama dengan *bitPayload*. Apabila nilai parity sama dengan bitPayload, maka tidak perlu dilakukan perubahan. Penyisipan *payload* secara *even parity* ke dalam berkas audio yang akan ditunjukkan pada tabel 3.2

Tabel 3.2 Penyisipan *payload* ke dalam berkas audio

Bit Asal (Region)	P1	BP	Bit Akhir (Region)	P2	Berbeda
1010111110001110	0	0	1010111110001110	0	Tidak
1110110110001110	0	1	1110110110001111	1	Ya
0110110110001110	1	0	0110110110001111	0	Ya

Bit Asal (Region)	P1	BP	Bit Akhir (Region)	P2	Berbeda
0001001011000001	1	0	0001001011000000	0	Ya
0001001001100001	1	1	0001001001100001	1	Tidak
0010000100010010	0	0	0010000100010010	0	Tidak
0101100100000000	0	1	0101100100000001	1	Ya
0111110101010101	0	0	0111110101010101	0	Tidak
0000000011000101	0	0	0000000011000101	0	Tidak
1101001101010101	1	1	1101001101010101	1	Tidak
0101000000000000	0	1	0101000000000001	1	Ya
1100011110101010	1	1	1100011110101010	1	Tidak
0000111110001101	0	0	0000111110001101	0	Tidak
0110010110101010	0	1	0110010110101011	1	Ya
0000100000001111	1	0	0000100000001110	0	Ya
1110011010101010	1	1	1110011010101010	1	Tidak
0000111111111010	0	0	0000111111111010	0	Tidak
1110101011111111	1	1	1110101011111111	1	Tidak
1100111100111101	1	1	1100111100111101	1	Tidak
0000101011111111	0	0	0000101011111111	0	Tidak
0010000111001111	0	0	0010000111001111	0	Tidak
1111010011111111	1	0	1111010011111110	0	Ya
0010100011001111	0	0	0010100011001111	0	Tidak
1100010001000111	1	1	1100010001000111	1	Tidak
0100011101101100	0	0	0100011101101100	0	Tidak
0100011101010110	0	1	0100011101010111	1	Ya
0011111101100101	0	1	0011111101100100	1	Ya
0101111010101010	1	1	0101111010101010	1	Tidak
1000000100111111	0	0	1000000100111111	0	Tidak
1010101001011001	0	0	1010101001011001	0	Tidak
0011111101110111	0	1	0011111101110110	1	Ya

Bit Asal (Region)	P1	BP	Bit Akhir (Region)	P2	Berbeda
0111101110101010	0	0	0111101110101010	0	Tidak
1001001110011100	0	0	1001001110011100	0	Tidak
0111001010011100	0	1	0111001010011101	1	Ya
1001101010011100	0	1	1001101010011101	1	Ya
1111111110011111	0	0	1111111110011111	0	Tidak
1001111111111110	1	0	1001111111111111	0	Ya
1001111111101110	0	0	1001111111101110	0	Tidak
1010111010001000	1	0	1010111010001001	0	Ya
0010000010001000	1	1	0010000010001000	1	Tidak

Keterangan :

BP : *bitPayload* dari karakter inputan

P1 : nilai *parity* dari region awal yang terbentuk.

P2 : nilai *parity* baru (setelah nilai BP dibandingkan dengan P1).

3.4.2 Pengungkapan Pesan

Untuk mendapatkan kembali pesan yang telah disisipkan ke dalam berkas audio, yaitu dengan cara mencari nilai *parity* dari representasi biner berkas *stego-data* (WAV) yang telah disisipkan pesan di dalamnya karena *bitPayload* dikodekan dalam bentuk *parity*. Dengan menyusun kembali nilai *parity* maka akan didapatkan pesan yang telah disisipkan sebelumnya. Proses pengungkapan pesan akan ditunjukkan pada tabel 3.3

Tabel 3.3 Pengungkapan kembali pesan dari berkas audio

No.	Representasi Biner	*Parity*	No.	Representasi Biner	*Parity*
1	1010111110001110	0	21	0010000111001111	0
2	1110110110001111	1	22	1111010011111110	0
3	0110110110001111	0	23	0010100011001111	0
4	0001001011000000	0	24	1100010001000111	1
5	0001001001100001	1	25	0100011101101100	0

No.	Representasi Biner	*Parity*	No.	Representasi Biner	*Parity*
6	0010000100010010	0	26	0100011101010111	1
7	0101100100000001	1	27	0011111101100100	1
8	0111110101010101	0	28	0101111010101010	1
9	0000000011000101	0	29	1000000100111111	0
10	1101001101010101	1	30	1010101001011001	0
11	0101000000000001	1	31	0011111101110110	1
12	1100011110101010	1	32	0111101110101010	0
13	0000111110001101	0	33	1001001110011100	0
14	0110010110101011	1	34	0111001010011101	1
15	0000100000001110	0	35	1001101010011101	1
16	1110011010101010	1	36	1111111110011111	0
17	0000111111111010	0	37	1001111111111111	0
18	1110101011111111	1	38	1001111111101110	0
19	1100111100111101	1	39	1010111010001001	0
20	0000101011111111	0	40	0010000010001000	1

Dari tabel 3.3 akan di dapatkan hasil :

0100 1010	0111 0101	0110 0001	0111 0010	0110 0001

Dari hasil di atas akan didapatkan nilai ASCII yaitu 74, 117, 97, 114, 97. Apabila nilai ASCII tersebut diubah dalam bentuk karakter menjadi 74 = J, 117 = u, 97 = a, 114 = r, dan 97 = a.

3.4.3 *Signal to Noise Ratio* (SNR)

Kualitas audio setelah dilakukan penyisipan akan mengalami perubahan. Untuk mengetahui perubahan tersebut secara matematis dapat dilakukan dengan menghitung nilai *signal to noise ratio*. Terdapat langkah-langkah untuk mendapatkan nilai *signal to noise ratio* dari berkas audio yang telah mengalami perubahan setelah melalui proses *parity coding modification*, yaitu :

1. Menghitung total bit dari berkas audio *carrier* (lihat tabel 3.2). Sebagai contoh pada kasus ini jumlah total bit adalah

$$Total_{bit} = Jumlah_{region} * jumlahBit_{TiapRegion}$$

$$Total_{bit} = 40 * 16 = 640$$

2. Menghitung jumlah total bit terubah. Jumlah ini bisa didapatkan dari tabel 3.2. Jadi jumlah total bit terubah adalah 15
3. Setelah didapat informasi dari dua langkah sebelumnya, maka dapat dihitung nilai SNR (%) dengan persamaan 2.1

$$SNR = \left(\frac{jumlahTotdBitFileWAV - jumlahBitTerubah}{jumlahTotdBitFileWAV}\right) * 100\ \%$$

$$SNR = \left(\frac{640 - 15}{640}\right) * 100\% = 97{,}66\ \%$$

Dari hasil perhitungan SNR dapat ditarik kesimpulan bahwa tingkat kemiripan berkas audio WAV sesudah disisipkan pesan di dalamnya dengan berkas audio WAV sebelum disisipkan pesan pesan adalah sebesar 97,66 %.

3.5 Perancangan Uji Coba

Perancangan uji coba pada sistem ini berfungsi sebagai pengukuran berjalannya sistem sesuai dengan yang diharapkan. Pengujian pada penelitian ini untuk mengetahui apakah metode *parity coding* memenuhi aspek keamanan steganografi atau tidak. Aspek keamanan steganografi yang diuji adalah *fidelity, recovery,* dan *robustness*.

3.5.1 Pengujian *Fidelity*

Pengujian *fidelity* digunakan untuk mengetahui perubahan kualitas audio WAV setelah disisipkan pesan di dalamnya. Perubahan yang terjadi dapat diketahui dengan menghitung nilai *signal to noise ratio* (SNR). Hasil perhitungan berupa nilai persentase yang semakin besar persentase nilai SNR, maka kualitas audio setelah disisipkan pesan tidak banyak berubah. Pengukuran nilai SNR ditunjukkan pada tabel 3.4

Tabel 3.4 Pengukuran nilai SNR

Nama dan Ukuran WAV *Carrier*	Nama dan Ukuran Pesan	Rasio Ukuran Pesan	Nama dan Ukuran *stego-data*	SNR (%)

3.5.2 Pengujian *Recovery* dan *Robustness*

Pengujian *recovery* digunakan untuk mengetahui keberhasilan sistem dalam pengembalian kembali pesan yang telah disisipkan sebelumnya pada berkas audio. Pengujian *recovery* ini dikombinasikan dengan *robustness*, agar mengetahui ketahanan berkas *strego-data* setelah dilakukan manipulasi. Manipulasi data yang dilakukan yaitu *cropping* dan perubahan format *stego-data*. Pengujian ketahanan *stego-data* terhadap perubahan format ditunjukkan pada tabel 3.5 dan pengujian ketahanan *stego-data* sesudah dilakukan *robustness* berupa teknik *cropping* ditunjukkan pada tabel 3.6

Tabel 3.5 Pengujian ketahanan *stego-data* terhadap perubahan format

Berkas *Stego-data*	Pesan Asli	Perlakuan	Hasil Esktraksi	Kesimpulan
Nama berkas *stego-data* yang diuji	Isi pesan asli	Perlakuan yang dilakukan	Hasil ekstraksi setelah dilakukan manipulasi	Sukses/Gagal

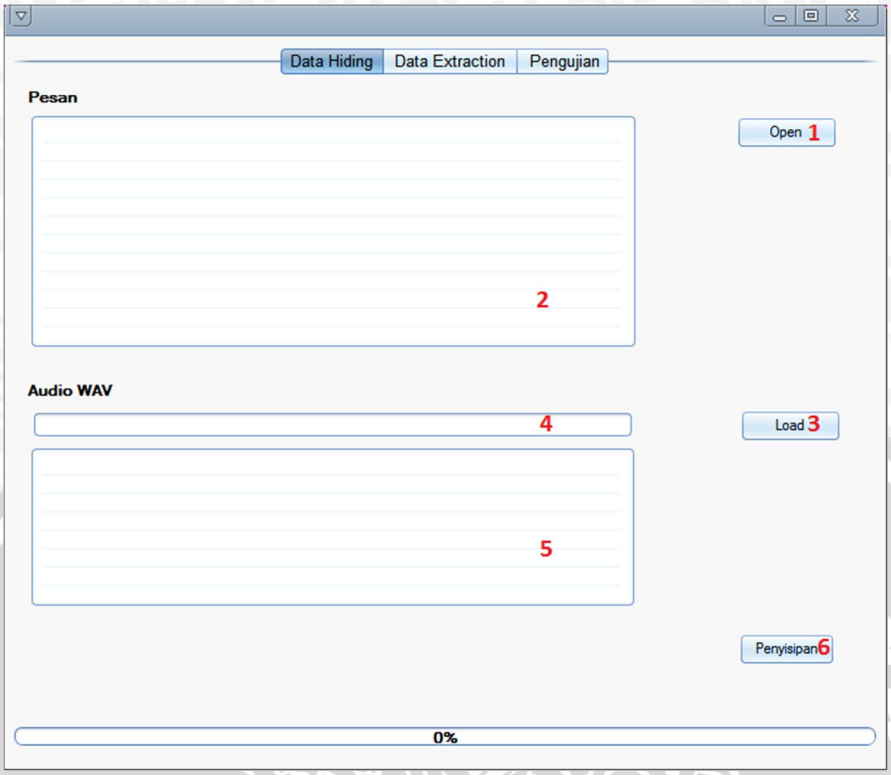
Tabel 3.6 Pengujian ketahanan *stego-data* sesudah dilakukan *robustness* berupa teknik *cropping*

Berkas *Stego-data*	Pesan Asli	Perlakuan	Hasil Esktraksi	Kesimpulan
Nama berkas *stego-data* yang diuji	Isi pesan asli	Perlakuan yang dilakukan(*cropping* bagian depan/belakang)	Hasil ekstraksi setelah dilakukan manipulasi	Sukses/ Gagal

Penilaian sukses atau gagal dalam kolom kesimpulan pada tabel uji berdasarkan pada sukses atau tidaknya program dalam mengungkapkan kembali pesan yang disisipkan. Nilai "berhasil" didapatkan ketika program dapat mengungkapkan kembali dengan sempurna pesan yang sudah disisipkan setelah dilakukan manipulasi data, sedangkan nilai "gagal" ketika program tidak sempurna atau tidak dapat mengungkapkan kembali pesan yang sudah disisipkan setelah dilakukan manipulasi data.

3.6 Perancangan Interface

Sistem ini memiliki tiga antarmuka (*interface*) utama yang akan disajikan oleh jendela menu saat pertama kali sistem ini dijalankan. Jendela utama yang terdapat pada sistem ini adalah *Data Hiding*, *Data Extraction*, dan Pengujian. Untuk bagian antarmuka *data hiding* akan ditunjukkan pada gambar 3.8

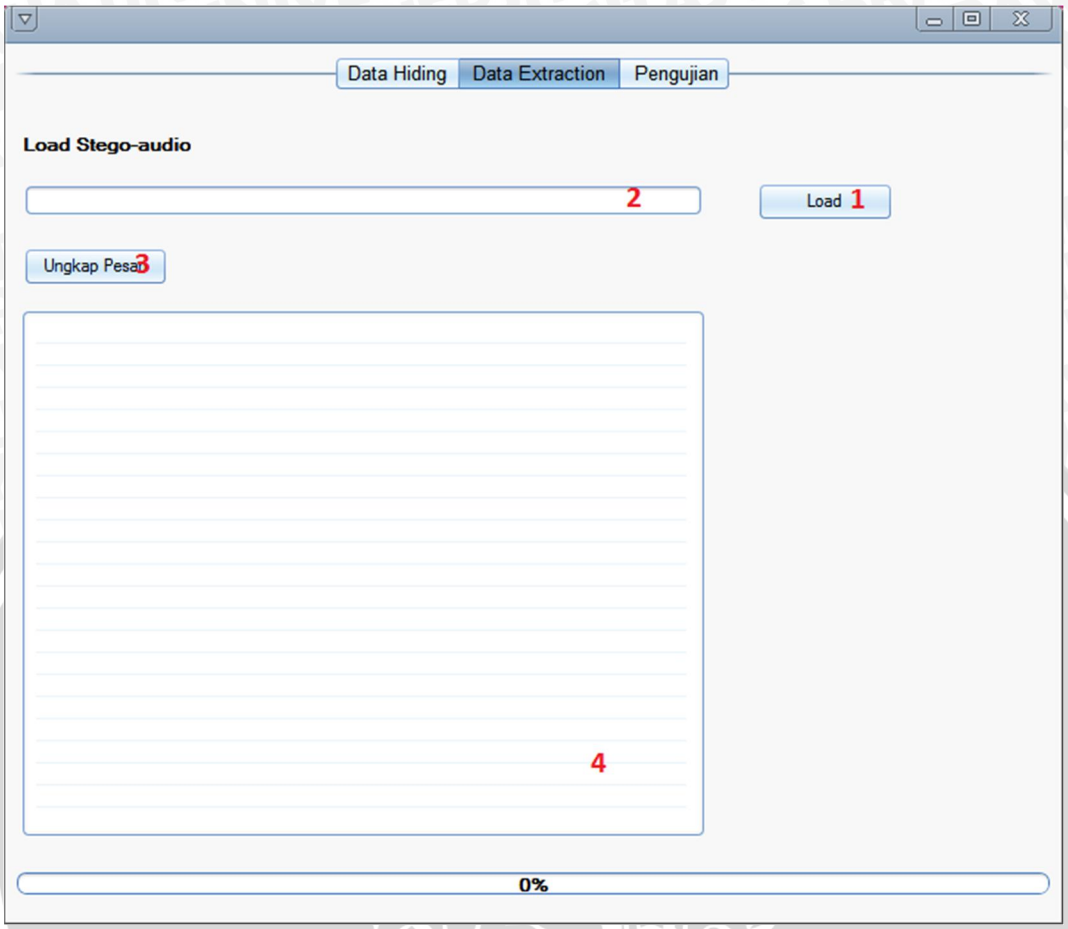


Gambar 3.8 Antarmuka *data hiding*

Antarmuka *data hiding* yang ditunjukkan pada gambar 3.8 terdiri dari :

1. Tombol *open* berfungsi untuk mencari berkas teks (*payload*) yang akan disisipkan ke dalam berkas audio.
2. *Text area* berfungsi untuk menampilkan isi *payload* yang akan disisipkan ke dalam audio.
3. Tombol *load* berfungsi untuk memilih berkas audio (*carrier*) yang akan disisipi oleh pesan.
4. *Textfield* berfungsi untuk menampilkan direktori berkas audio (*carrier*) yang akan disisipi *payload*.
5. *Text area* berfungsi untuk menampilkan informasi tentang atribut *carrier* yang akan disisipi *payload*.
6. Tombol penyisipan berfungsi untuk melakukan proses penyisipan *payload* ke dalam berkas audio terpilih.

Bagian antarmuka *data extraction* akan ditunjukkan pada gambar 3.9

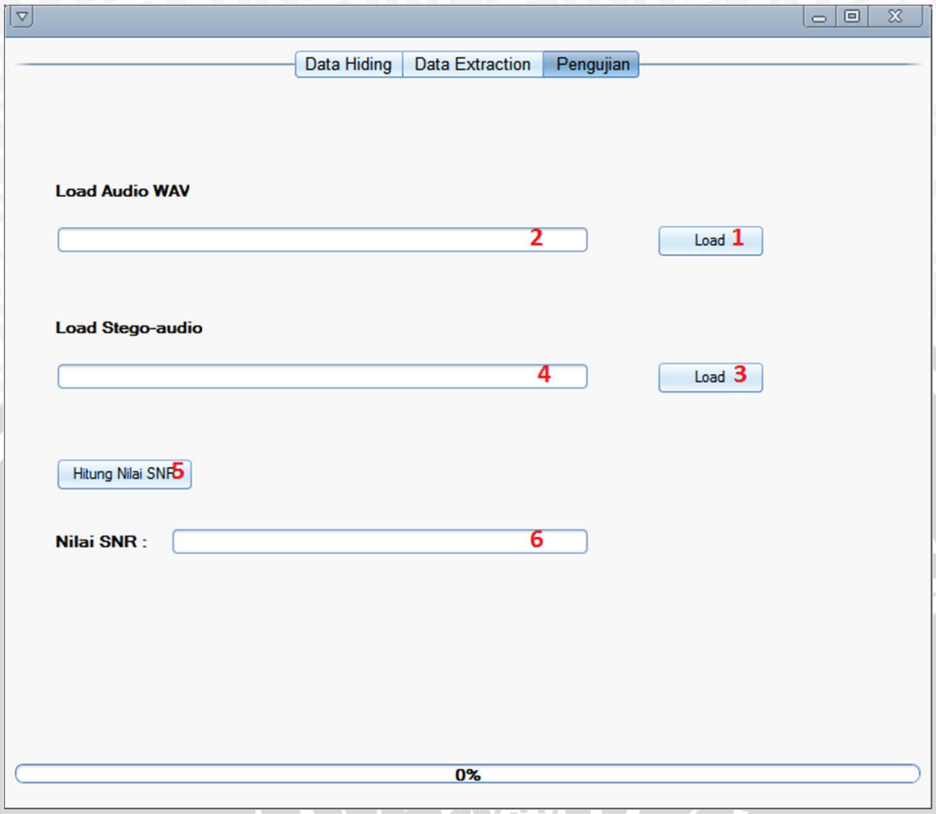


Gambar 3.9 Antarmuka *data extraction*

Antarmuka *data extraction* yang ditunjukkan pada gambar 3.9 terdiri dari :

1. Tombol *load* berfungsi untuk mencari / memilih berkas *stego-data* yang akan diungkap pesan yang terdapat di dalamnya.
2. *Textfield* berfungsi untuk manampilkan direktori *stego-data* yang akan diungkap pesan yang terdapat di dalamnya.
3. Tombol ungkap pesan berfungsi untuk melakukan proses pengungkapan pesan dari *stego-data*.
4. *Text area* berfungsi untuk menampilkan isi pesan yang diungkap dari dalam *stego-data*.

Bagian antarmuka pengujian ditunjukkan pada gambar 3.10



Gambar 3.10 Antarmuka pengujian

Antarmuka pengujian yang ditunjukkan pada gambar 3.10 terdiri dari :

1. Tombol *load* berfungsi untuk memilih berkas audio yang akan dihitung nilai *signal to noise ratio*.
2. *Textfield* berfungsi untuk menampilkan direktori berkas berkas audio yang akan dihitung nilai *signal to noise ratio*.
3. Tombol *load* berfungsi untuk memilih berkas *stego-data* yang akan dihitung nilai *signal to noise ratio*.
4. *Textfield* berfungsi untuk menampilkan direktori berkas berkas *stego-data* yang akan dihitung nilai *signal to noise ratio*.
5. Tombol hitung SNR berfungsi untuk melakukan proses perhitungan nilai *signal to noise ratio*.
6. *Textfield* berfungsi untuk manampilkan besar nilai *signal to noise ratio.*