Pengamanan Pesan Menggunakan Metode MLSB PRNG dan Kompresi File dengan Algoritma RLE pada File Audio

Rian Rivaldo, Handrizal*, Herriyance

Program Studi S-1 Ilmu Komputer, Fakultas Ilmu Komputer dan Teknologi Informasi,
Universitas Sumatera Utara, Jl. Universitas No. 9-A, Kampus USU, Medan 20155, Indonesia

Naskah Diterima : 27 Agustus 2020; Diterima Publikasi : 30 Desember 2020

DOI: 10.21456/vol11iss1pp1-8

Abstract

The development of sending messages from one place to another can be done regardless of distance and time. However, the delivery of these messages is hampered by problems of confidentiality and message security. Especially if the data contains important and confidential information that not just anyone is allowed to read and find out about it. In overcoming this problem, steganography techniques can be used with the Modified Least Significant Bit algorithm, where the determination of the embedding index is based on random numbers generated by the Pseudo-Random Number Generator with the Multiply with Carry algorithm. In addition to security, data size is also an important factor in data transmission. The larger the size the more time it will take to transmit the data. Therefore, the Run Length Encoding algorithm is needed to compress the data size, which will shorten the time to transmit the data. In the message extraction process, a stego key is needed to generate random numbers. Based on the testing of the extraction process with an arbitrary key, it is obtained that the message tested is not the original message that has been embedded previously. In the results of the embedding and extraction process, it is obtained that the average value of PSNR is 63.61498 dB, which means the quality of the stego object produced is quite good. Whereas the measurement of file compression performance results with an average value of Compression Ratio at 1.00113, Space Savings at 0.1133%, and Bitrate at 584025.33 bits/sample. These results indicate that RLE algorithm compression is not efficient to compress file sizes.

Keywords: Steganography; Modified Least Significant Bit; Pseudo-Random Number Generator; Multiply with Carry; Compression; Run Length Encoding

Abstrak

Perkembangan pengiriman pesan dari suatu tempat ke tempat lain dapat dilakukan tanpa memandang jarak dan waktu. Akan tetapi, pengiriman pesan tersebut banyak terkendala dengan permasalahan kerahasiaan dan keamanan pesan. Terlebih jika data tersebut mengandung informasi penting dan rahasia yang tidak sembarang orang boleh membaca dan mengetahuinya. Dalam mengatasi permasalahan tersebut dapat digunakan teknik steganografi dengan algoritma *Modified Least Significant Bit*, dimana penentuan indeks penyisipannya didasarkan pada bilangan acak yang dihasilkan *Pseudo Random Number Generator* dengan algoritma *Multiply with Carry*. Selain keamanan data, ukuran data juga menjadi faktor penting dalam pengiriman data. Semakin besar ukuran data maka semakin banyak juga waktu yang diperlukan dalam mentransmisikan data tersebut. Oleh karena itu, dibutuhkan algoritma kompresi *Run Length Encoding* untuk dapat memampatkan ukuran data sehingga mempersingkat waktu dalam mentransmisikan data. Pada proses pengekstrakan pesan diperlukan sebuah *stego key* yang berfungsi untuk membangkitkan bilangan acak. Berdasarkan hasil pengujian proses *extraction* dengan kunci sembarang, diperoleh bahwa pesan yang dihasilkan bukan merupakan pesan asli yang telah disisipkan sebelumnya. Pada hasil pengujian proses *embedding* dan *extraction* diperoleh bahwa nilai rata – rata PSNR sebesar 63.61498 dB yang berarti kualitas *stego object* yang dihasilkan cukup baik. Sedangkan pada pengukuran hasil kompresi *file*, diperoleh nilai rata – rata *Compression Ratio* sebesar 1.00113, *Space Savings* sebesar 0.1133 %, dan *Bitrate* sebesar 584025.33 *bits/sample* yang berarti kompresi algoritma RLE tidak efisien dalam memampatkan ukuran *file* bila dibandingkan penelitian sebelumnya.

Keywords: Steganografi; Modified Least Significant Bit; Pseudo-Random Number Generator; Multiply with Carry; Kompresi; Run Length Encoding.

1. Pendahuluan

Pengiriman pesan banyak terkendala dengan permasalahan kerahasiaan dan keamanan pesan. Terlebih jika data tersebut mengandung informasi penting dan rahasia yang tidak sembarang orang boleh mengetahuinya. Dalam literatur, terdapat beberapa cara yang dapat dilakukan untuk menyembunyikan dan merahasiakan pesan yang hendak dikirim kepada pihak tertentu. Salah satu

*) Penulis korespondensi: Handrizal@usu.ac.id

teknik yang dapat dilakukan adalah dengan menyisipkan pesan yang akan dikirimkan ke dalam media lain, sehingga pesan tersebut akan tersembunyi dan yang terlihat oleh orang lain hanya media yang telah digunakan untuk menyisipkan pesan rahasia tersebut (Ricky, 2018).

Steganografi merupakan salah satu teknik untuk menyembunyikan data yang dirahasiakan dengan menyisipkannya ke dalam media digital, seperti citra, audio maupun video. Pada steganografi pesan disisipkan dalam bentuk yang relatif aman sehingga sulit untuk diekstraksi menjadi bentuk yang semula (Krisnawati, 2008). Dalam steganografi terdapat beberapa metode yang dapat digunakan dalam menyisipkan data ke dalam sebuah audio digital, salah satunya adalah metode Least Significant Bit (LSB). Metode ini banyak digunakan karena proses komputasi yang tidak terlalu kompleks dan pesan yang disembunyikan relatif aman. Prinsip dasar metode ini adalah dengan mengganti bit terakhir dari data sampel audio dengan bit-bit pesan yang akan disisipkan (Purba, 2012).

Metode LSB pada umumnya sudah banyak diketahui oleh steganalisis tentang penerapan dan teknik ekstraksi pesan. Oleh karena itu, dalam penelitian ini penulis menggunakan metode *Modified Least Significant Bit* (MLSB) dimana penentuan bit yang akan digantikan didasarkan oleh bilangan acak yang dihasilkan *Pseudo Random Number Generator* (PRNG) dengan algoritma *Multiply With Carry* (MWC) untuk meningkatkan keamanan data yang akan dirahasiakan. Selain keamanan data, pengiriman file dengan waktu yang singkat sudah menjadi prioritas utama dalam meningkatkan efisiensi waktu pengiriman. Kecepatan pengiriman ini sangatlah bergantung pada kecepatan transmisi dan ukuran file yang akan dikirim. Salah satu solusi untuk masalah tersebut adalah dengan melakukan pemampatan data (kompresi) sebelum dilakukan pengiriman kepada penerima, lalu penerima akan mengembalikan data yang telah diterima menjadi data aslinya atau yang dikenal dengan proses dekompresi (Rahandi, 2012).

Berdasarkan latar belakang yang telah dijabarkan diatas, maka penulis ingin mengimplementasikan metode MLSB dimana indeks penyisipan didasarkan pada bilangan acak yang dihasilkan algoritma MWC, berbeda pada penelitian Purba (2012) dengan judul implementasi steganografi pesan *Text* kedalam *File Sound* (.Wav) dengan modifikasi jarak Byte pada *Algoritma Least Significant Bit* (LSB) yang indeks penyisipannya didasarkan pada *PosLeap*. Selain itu, penulis juga mengkombinasikan algoritma MLSB dengan algoritma RLE untuk pengamanan pesan teks dan kompresi file audio, dimana implementasi algoritma RLE didasarkan pada penelitian yang berjudul analisis dan implementasi kompresi *File* audio dengan menggunakan *Algoritma Run Length Encoding* (RLE) (Rahandi, 2012).

2. Kerangka Teori

2.1. Audio Digital

Audio digital merupakan bentuk suara analog yang telah melalui proses digitalisasi sebelumnya. Suara sendiri merupakan gelombang yang merambat akibat tekanan udara melalui medium (benda padat, cair, atau gas) dan terdiri atas frekuensi yang dapat didengar dalam jangkauan tingkat pendengaran. Proses digitalisasi audio analog ini dimulai dengan mengubah sinyal suara menjadi bilangan biner menggunakan sebuah alat yang bernama *Analogue to Digital Converter* (ADC). Kemudian untuk dapat memperdengarkan audio yang sudah digitalisasi, audio digital diubah kembali menjadi audio analog terlebih dahulu menggunakan *Digital to Analogue Converter* (DAC).

Audio digital merupakan penyajian dari suara asli atau dengan kata lain sampel suara yang disimpan sebagai informasi digital dalam bit atau *byte*. Kualitas perekaman digital bergantung pada tingkat *sampling rate* (frekuensi), yaitu seberapa sering sampel yang diambil per detik dan berapa banyak angka yang digunakan untuk menampilkan nilai dari tiap sampel (*bit depth*, ukuran sampel, resolusi, jarak dinamis). Semakin tinggi *sampling rate* gelombang suara yang diambil mengakibatkan data yang disimpan mengenai sampel semakin banyak, artinya kualitas suara yang dihasilkan menjadi lebih baik. Akan tetapi, ukuran dari *file* suara tersebut menjadi lebih besar seiring dengan meningkatnya resolusi dan kualitas suara yang dihasilkan (Arifin, 2015).

2.2. Steganography

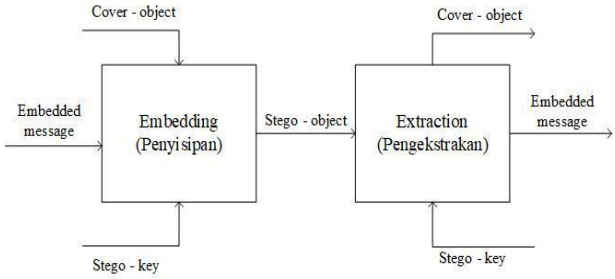
Dalam teknik steganografi terdapat dua properti yang dibutuhkan, yaitu wadah penampung sebagai tempat penyisipan pesan dan data rahasia yang akan disisipkan ke dalam wadah penampung tersebut. Media penampung dan informasi yang akan disisipkan dalam teknik steganografi dapat berupa media digital, seperti audio, citra, teks, serta video. Informasi yang nantinya disisipkan ke dalam media penampung tidak akan terlihat oleh indra manusia, melainkan hanya media yang dipakai untuk menyisipkan informasi tersebut yang akan terlihat dikarenakan informasi yang akan dikirim atau diberikan kepada pihak tertentu sudah disembunyikan ke dalam media penampung. Sehingga tidak menimbulkan kecurigaan akan adanya informasi di dalam media penampung (Lubis, 2012).

Pada steganografi terdapat beberapa faktor yang perlu diperhatikan, yaitu (Ricky, 2018):

1. *Imperceptibility* yaitu keberadaan informasi rahasia tidak dapat dipersepsi oleh pihak lain selain penerima pesan.
2. *Fidelity* yaitu kualitas media penampung informasi tidak mengalami perubahan yang signifikan setelah proses steganografi. Perubahan

kualitas tersebut tidak dapat dipersepsi oleh pihak lain.

3. *Recovery* yaitu pengekstrakan kembali pesan yang telah disisipkan ke dalam media penampung. Hal ini bertujuan untuk sewaktu-waktu pesan rahasia yang telah disisipkan dapat diambil kembali serta digunakan lebih lanjut.
4. *Robustness* yaitu keberadaan informasi tidak mengalami kerusakan jika terjadi berbagai operasi manipulasi.
5. Dalam steganografi terdapat dua proses yang harus dilakukan, yaitu penyisipan pesan ke dalam media penampung yang disebut proses *embedding* dan pengekstrakan kembali pesan yang telah disisipkan yang disebut proses *extraction*. Proses *embedding* bertujuan untuk menyisipkan pesan atau informasi yang masih dalam bentuk aslinya, berupa pesan teks biasa, *cipher text*, atau bentuk apa pun yang dapat disisipkan ke dalam *bitstream*. Proses *extraction* adalah proses yang bertujuan untuk menguraikan pesan yang telah disisipkan ke dalam *cover object* (Emam, 2016). Gambaran dari proses steganografi dapat dilihat pada Gambar 1.



Gambar 1. Proses Steganografi

2.3. Algoritma Least Significant Bit (LSB)

Teknik steganografi yang umum digunakan dalam menyisipkan sebuah informasi ke dalam sebuah audio digital dapat menggunakan metode *Least Significant Bit* (LSB). Secara sederhana metode ini bekerja dengan mengubah terlebih dahulu *cover object* dan juga pesan rahasia ke dalam bentuk biner (0 dan 1). Kemudian mengganti bit terakhir (LSB) pada wadah penampung dengan bit pesan yang akan disisipkan. Metode LSB dapat dilakukan dengan Langkah-langkah sebagai berikut (Rachmawati, 2017) :

1. Jika nilai bit pesan yang akan disisipkan bernilai 1 dan hasil nilai sampel audio modulus 2 adalah 0, maka tingkatkan nilai sampel audio sebesar 1.
2. Jika nilai bit pesan yang akan disisipkan bernilai 0 dan hasil nilai sampel audio modulus 2 adalah 1, maka turunkan nilai sampel audio sebesar 1.
3. Jika nilai bit pesan yang akan disisipkan sama dengan nilai sampel audio modulus 2, maka lanjutkan ke proses penyisipin bit pesan berikutnya.

2.4. Algoritma Multiply With Carry (MWC)

Algoritma *Multiply with Carry* (MWC) merupakan salah satu algoritma pembangkit bilangan acak atau *Pseudo Random Number Generator* (PRNG) yang merupakan pembangkit bilangan acak yang sulit untuk diprediksi urutannya oleh pihak ketiga. Metode ini bertindak dengan mengambil sebuah bilangan *(seed)*, lalu kemudian memproduksi sebuah urutan bilangan (Emam, 2016). Untuk memperoleh bilangan acak yang dibangkitkan dengan algoritma ini didapat dari persamaan (Marsaglia, 2003):

$$X_n = (aX_{n-1} + C_{n-1}) \bmod b$$
$$C_n = \left\lfloor \frac{aX_{n-1} + C_{n-1}}{b} \right\rfloor$$

dimana,

X_n = bilangan acak ke – *(n) dari* barisan bilangan acak

X_{n-1} = bilangan acak ke – *(n-1)*

a = konstanta pengali

C_n = konstanta kenaikan ke – *(n)*

C_{n-1} = konstanta kenaikan ke – *(n-1)*

b = konstanta modulus dan pembagi

2.5. Algoritma Modified Least Significant Bit (MLSB)

Algoritma MLSB merupakan sebuah pengembangan dari algoritma LSB. Tujuan pengembangan metode ini adalah untuk menghasilkan metode yang lebih baik dan lebih aman dari metode LSB. Algoritma MLSB secara garis besar mempunyai implementasi yang sama dengan metode LSB, hanya saja penentuan indeks sampel audio yang akan disisipkan pesan teks didasarkan pada bilangan acak yang dihasilkan oleh algoritma MWC. Algoritma MWC akan menghasilkan deret bilangan acak sebanyak jumlah bit pesan teks yang akan disisipkan.

2.6. Kompresi

Kompresi merupakan sebuah proses pemampatan ukuran data yang digunakan untuk merepresentasikan sebuah berkas menjadi lebih kecil dari ukuran awal tanpa mengurangi kualitas data secara signifikan. Tujuan dari kompresi ini diperlukan untuk mengurangi jumlah data yang diperlukan dalam mewakili sampel audio digital. Oleh karena itu juga, nilai yang diperlukan untuk penyimpanan dan pengiriman audio digital menjadi lebih kecil (Rachmawati, 2017). Hasil dari proses kompresi tidak dapat ditampilkan secara langsung. Sehingga dibutuhkan sebuah proses untuk merekonstruksi data tersebut menjadi data semula yang disebut proses dekompresi (Darwis, 2016).

Metode kompresi dapat dibagi menjadi dua jenis, yaitu *lossless* dan *lossy*. Setelah proses dekompresi, algoritma *lossless* menghasilkan berkas yang identik dengan berkas sebelum terkompresi tanpa hilangnya beberapa data. Pada algoritma jenis *lossy* berkas yang dihasilkan mirip tetapi tidak identik dengan

sebelumnya, karena dalam proses kompresi terdapat beberapa data mengenai audio telah hilang (Rachmawati, 2017).

2.7. Algoritma Run Length Encoding (RLE)

Algoritma *Run Length Encoding* (RLE) merupakan algoritma kompresi berjenis *lossless* yang sering digunakan. Teknik kompresi audio jenis *lossless* terdiri atas tiga langkah. Pertama, suara asli didigitalisasi terlebih dahulu. Kedua, sampel audio tersebut dikonversi yang dalam prosesnya menggunakan prediksi linier ke beberapa bilangan yang disebut residu. Residu kemudian diganti dengan *variable-length codes.* Langkah terakhir adalah menghasilkan kompresi itu sendiri (Salomon, 2007).

Pada algoritma RLE pengkodean dilakukan dengan mengidentifikasi minimal tiga deret simbol yang identik, lalu mengganti setiap simbol dengan sebuah penanda dan panjang dari deret tersebut. Berikut adalah Langkah-langkah proses kompresi menggunakan algoritma RLE, yaitu :

1. Nilai ASCII karakter pertama akan dibandingkan dengan karakter kedua. Jika nilai keduanya sama, selanjutnya akan dihitung jumlah karakter berulang lalu membandingkan dengan karakter berikutnya sampai menghasilkan nilai ASCII yang berbeda dengan karakter pertama.
2. Jika jumlah karakter berulang tersebut bernilai lebih besar atau sama dengan tiga, maka selanjutnya merubah karakter berulang tersebut menjadi karakter penanda, jumlah perulangan karakter dan karakter berulang itu sendiri.
3. Lakukan langkah 1 dan 2 sampai akhir karakter pada pesan teks.

2.8. Mean Squared Error

Mean Squared Error (MSE) merupakan tingkat keakuratan suatu model prediksi, nilainya merepresentasikan rata-rata kesalahan antara hasil prediksi dengan nilai sebenarnya dengan memperhitungkan kuadrat bias. Sehingga MSE berguna dalam membandingkan audio asli dan audio hasil penyisipan dengan memeriksa selisih nilai keduanya yang dapat dirumuskan pada persamaan sebagai berikut.

$$MSE = \frac{1}{n}\sum_{i=0}^{n-1}[M' - M]^2$$

2.9. Peak Signal to Noise Ratio

Peak Signal to Noise Ratio merupakan nilai perbandingan antara nilai maksimum pada audio hasil pengolahan dengan kuantitas gangguan atau disebut juga *noise*, yang dinyatakan dalam satuan desibel (dB). Secara matematis PSNR dapat dirumuskan seperti pada persamaan dibawah ini:

$$PSNR = 10\log_{10}\left(\frac{M^2}{MSE}\right)$$

2.10. Parameter Pengukuran Kompresi

Parameter – parameter yang diperlukan dalam pengukuran kualitas hasil kompresi adalah sebagai berikut (Budiman & Rachmawati, 2017):

1. *Compression Ratio*

Compression Ratio (CR) secara matematis dapat dirumuskan sebagai berikut:

$$CR = \frac{Ukuran\ data\ sebelum\ proses\ kompresi}{Ukuran\ data\ setelah\ proses\ kompresi}$$

2. *Space Savings*

Space Savings (SS) secara matematis dapat dirumuskan sebagai berikut:

$$SS = \left(1 - \frac{Ukuran\ data\ setelah\ proses\ kompresi}{Ukuran\ data\ sebelum\ proses\ kompresi}\right) \times 100\%$$

3. Bitrate

Bitrate secara matematis dapat dirumuskan sebagai berikut:

$$Bitrate = \frac{Ukuran\ data\ setelah\ proses\ kompresi}{Jumlah\ simbol}$$

3. Metode

Kemudahan dalam transmisi data yang berkembang saat ini membuat data dapat ditransmisikan oleh siapa, kapan, dan di mana saja. Hal tersebut banyak terkendala dengan keamanan data yang mengandung informasi rahasia dan kecepatan transmisi data yang disebabkan oleh besarnya ukuran data yang akan dikirimkan. Oleh karena itu, pada penelitian ini, masalah yang akan dibahas adalah bagaimana mengamankan suatu file dengan menyisipkan data tersebut ke dalam audio digital dan menghasilkan ukuran file yang lebih kecil dengan melakukan pemampatan pada file tersebut.

Langkah – langkah dalam proses penyisipan pesan dengan algoritma *Modified Least Significant Bit* adalah sebagai berikut:

1. Memasukkan pesan, *cover object*, dan *seed* PRNG
2. Membangkitkan bilangan acak dengan algoritma MWC.
3. Penyisipan pesan dengan algoritma MLSB, menghitung nilai MSE, PSNR, dan *Running Time*.
4. Simpan hasil *stego object* dan tampilkan nilai MSE, PSNR, dan *Running Time*.

Langkah – langkah dalam proses kompresi *stego object* dengan algoritma *Run Length Encoding* adalah sebagai berikut:

1. Memasukkan *stego object* dan melakukan pembacaan nilai *file*.
2. Kompresi nilai sampel jika ditemukan data nilai sampel yang sama secara berurutan lebih dari dua dengan cara mengganti deret nilai sampel berulang dengan bit penanda (#), jumlah perulangan nilai sampel, dan nilai sampel.

3. Menghitung nilai CR, SS, *Bitrate*, dan *Running Time*.
4. Simpan *stego object* terkompresi, dan tampilkan nilai CR, SS, *Bitrate*, dan *Running Time*.

Langkah-langkah dalam proses dekompresi *stego object* terkompresi dengan algoritma *Run Length Encoding* adalah sebagai berikut:

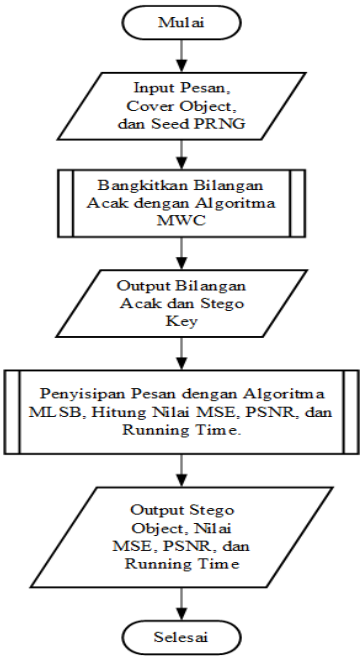
1. Memasukkan *stego object* terkompresi dan lakukan pembacaan nilai *file*.
2. Jika terdapat nilai sampel yang identik dengan bit penanda (#), maka akan dibaca jumlah nilai sampel berulang, dan nilai sampelnya. Selanjutnya tulis nilai sampel sebanyak jumlah perulangan pada *file* hasil dekompresi.
3. Jika nilai sampel tidak identik dengan bit penanda maka nilai tersebut akan langsung dituliskan pada *file* hasil dekompresi.

Langkah – Langkah dalam proses pengekstrakan pesan dengan algoritma *Modified Least Significant Bit* adalah sebagai berikut:

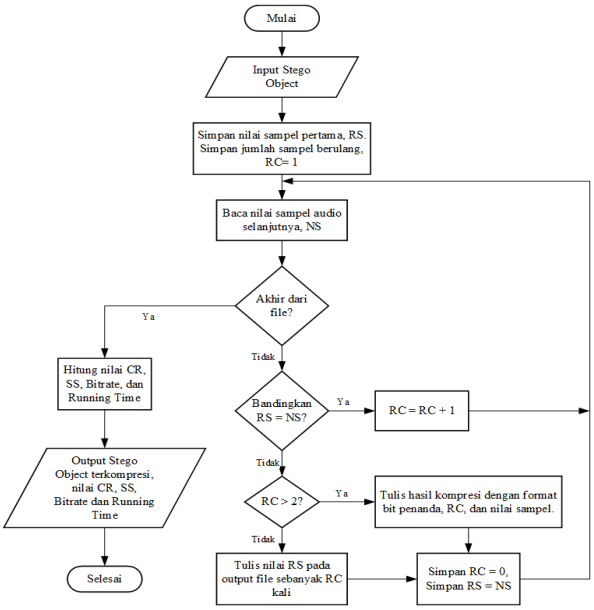
1. Memasukkan *stego object* dan *stego key* yang dihasilkan pada proses penyisipan sebelumnya.
2. Membangkitkan bilangan acak dengan algoritma MWC.
3. Proses pengekstrakan pesan dari *stego object*.

3.1. Flowchart

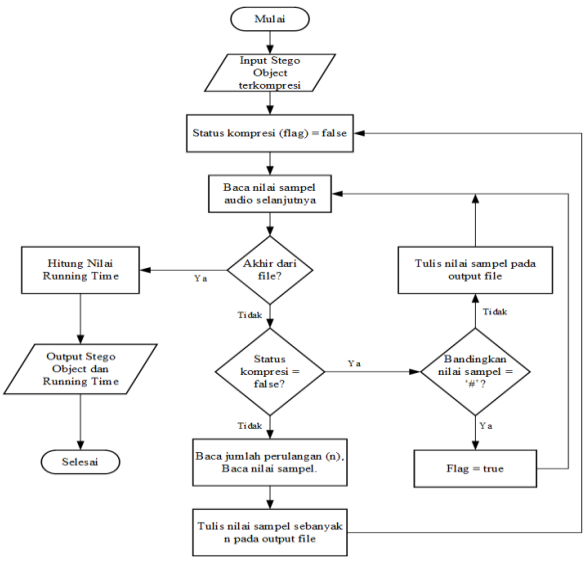
Flowchart atau diagram alir merupakan langkah-langkah dalam menyelesaikan masalah dengan sistematis yang divisualisasikan dengan bentuk diagram. Dalam penelitian ini terdapat beberapa *flowchart*, yaitu *flowchart embedding* (Gambar 2), *flowchart* kompresi (Gambar 3), *flowchart* dekompresi (Gambar 4), dan *flowchart extraction* (Gambar 5).



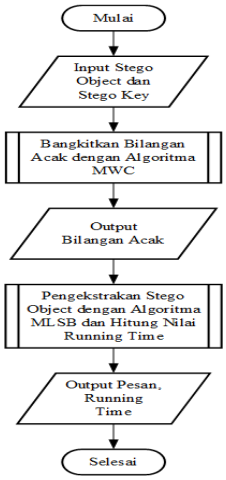
Gambar 2. *Flowchart Embedding*



Gambar 3. *Flowchart* Kompresi



Gambar 4. *Flowchart* Dekompresi



Gambar 5. *Flowchart Extraction*

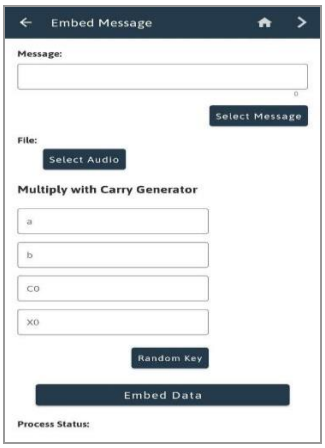
4. Hasil dan Pembahasan

Implementasi sistem dibangun pada sistem operasi android versi 7.0 ke atas yang dibangun menggunakan IDE *Android Studio* versi 3.6.3 dengan bahasa pemrograman *Java*.

Fragment home (Gambar 6) merupakan halaman pertama yang tampil saat sistem dijalankan. Pada *fragment* ini terdapat lima tombol untuk menuju lima *fragment* lainnya yang terdapat pada sistem, yakni tombol menuju *fragment embedding*, *fragment compression*, *fragment decompression*, *fragment extraction*, dan *fragment about*.

Gambar 6. *Fragment Home*

Fragment embedding (Gambar 7) bertujuan untuk melakukan proses penyisipan pesan kedalam *cover object*. Pada *fragment* ini terdapat tombol untuk pengguna memasukkan *file* pesan yang akan disisipkan, tombol untuk memasukkan *cover object* sebagai media penampungnya, empat buah *EditText* untuk memasukkan *seed* PRNG, tombol untuk membangkitkan bilangan *seed* PRNG secara acak, tombol untuk membangkitkan bilangan acak yang dihasilkan PRNG berdasarkan *seed* yang sudah dimasukkan, dan tombol untuk memulai proses penyisipan pesan dengan algoritma *Modified Least Significant Bit*. Selain itu, terdapat juga status dari proses yang sedang dijalankan.



Gambar 7. *Fragment Embedding*

Fragment compression (Gambar 8) bertujuan untuk melakukan proses kompresi pada *stego object*. Pada *fragment* ini terdapat tombol untuk memilih *stego object* yang akan dikompresi dengan algoritma *Run Length Encoding* dan tombol untuk memulai proses kompresi. Selain itu, terdapat juga status dari proses yang sedang dijalankan.

Gambar 8. *Fragment Compression*

Fragment decompression (Gambar 9) bertujuan untuk melakukan proses dekompresi pada *stego object* yang sudah dikompresi. Pada *fragment* ini terdapat tombol untuk memilih *stego object* terkompresi yang akan didekompresi dengan algoritma *Run Length Encoding* dan tombol untuk memulai proses dekompresi. Selain itu, terdapat juga status dari proses yang sedang dijalankan.

Gambar 9. *Fragment Decompression*

Fragment extraction (Gambar 10) bertujuan untuk melakukan proses pengekstrakan pesan dari *stego object*. Pada *fragment* ini terdapat tombol untuk pengguna memasukkan *stego object* yang akan dilakukan pengekstrakan, tombol untuk memasukkan *stego key*, empat buah *EditText* untuk menampilkan *stego key*, tombol untuk memulai proses pengekstrakan *stego object* dengan algoritma *Modified Least Significant Bit*, dan *EditText* untuk menampilkan pesan yang disisipkan. Selain itu, terdapat juga status dari proses yang sedang dijalankan.

Gambar 10. *Fragment Extraction*

4.1. Hasil Pengujian Algoritma Modified Least Significant Bit (MLSB)

Pada penelitian ini dilakukan pengujian dengan mengukur performa steganografi dengan menghitung nilai *Mean Squared Error* (MSE) dan nilai *Peak Signal to Noise Ratio* (PSNR). Pada pengujian ini digunakan lima file pesan teks berekstensi *.txt yang masing – masing berukuran 10kb, 15kb, 20kb, 25kb, dan 50kb. Sedangkan, cover object yang digunakan untuk pengujian ini terdapat lima file berekstensi *.wav yang masing-masing berukuran 0.882Mb, 1.80Mb, 5.76Mb, 9.60Mb, dan 28.80Mb. Hasil pengujian algoritma MLSB dapat dilihat pada Tabel 1 berikut ini:

Tabel 1. Hasil Pengujian Algoritma MLSB

Cover Object (Mb)	Pesan (kb)	MSE	PSNR (dB)
0.882	10	0.04680	61.42778
0.882	15	0.06935	59.71982
0.882	20	0.09228	58.47967
0.882	25	0.11467	57.53626
0.882	30	0.13768	56.74205
1.80	10	0.02225	64.65750
1.80	15	0.03292	62.95528
1.80	20	0.04361	61.73427
1.80	25	0.05479	60.74310
1.80	30	0.06557	59.96356
5.76	10	0.00714	69.58988
5.76	15	0.01070	67.83411
5.76	20	0.01416	66.61895
5.76	25	0.01766	65.65954
5.76	30	0.02112	64.88215
9.60	10	0.00424	71.85300
9.60	15	0.00634	70.10743
9.60	20	0.00838	68.89498
9.60	25	0.01049	67.92043
9.60	30	0.01253	67.15023
28.80	10	0.00143	76.55462
28.80	15	0.00212	74.84984
28.80	20	0.00283	73.60336
28.80	25	0.00353	72.65191
28.80	30	0.00423	71.85967
Rata – rata		0.03103	63.61498

Berdasarkan Tabel 1 dapat dilihat bahwa semakin besar ukuran *file* pesan teks maka akan membuat nilai MSE semakin besar dan nilai PSNR menjadi lebih rendah.

4.2. Hasil Pengujian Algoritma Run Length Encoding

Pada penelitian ini dilakukan pengujian kompresi dan dekompresi algoritma RLE untuk mengetahui kualitas hasil kompresi. Parameter yang digunakan dalam menilai kualitas hasil kompresi, yaitu Compression Ratio, Space Savings, dan Bitrate. Pada pengujian ini digunakan lima file cover object yang masing-masing berukuran 0.882Mb, 1.80Mb, 5.76Mb, 9.60Mb, dan 28.80Mb. Hasil pengujian algoritma RLE yang dapat dilihat pada Tabel 2.

Berdasarkan Tabel 2 dapat dilihat bahwa efisiensi hasil kompresi menggunakan algoritma RLE menghasilkan *Space Savings* yang kecil sehingga dapat dikatakan algoritma *Run Length Encoding* tidak berpengaruh secara signifikan dalam memampatkan ukuran *file stego object*. Hasil pengujian ini berbeda dengan hasil pengujian yang dilakukan pada penelitian sebelumnya, dimana ukuran pemampatan rata – rata dari hasil pengujian yang dilakukan oleh peneliti sebelumnya sebesar 13.83% (Rahandi, 2012).

Tabel 2. Hasil Pengujian Efisiensi Kompresi

File	Ukuran file (byte)		Efisiensi Kompresi		
	Sebelum	Setelah	CR	SS (%)	Bitrate
sample1.wav	882044	881594	1.00051	0.05101	55099.62
GI60_15s.wav	1800044	1799983	1.00003	0.00338	112498.93
LI192_15s.wav	5760044	5754027	1.00104	0.10446	359626.68
GI16.wav	9600044	9598176	1.00019	0.01945	599886
GR48.wav	28800044	28688247	1.00389	0.38818	1793015.43
Rata – rata			1.00113	0.1133	584025.33

5. Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan dalam tahap analisis, implementasi, dan pengujian sistem pengamanan pesan teks menggunakan algoritma *Modified Least Significant Bit* dan algoritma kompresi *Run Length Encoding*, maka dapat disimpulkan bahwa kombinasi dari algoritma *Modified Least Significant Bit* dalam penyisipan pesan teks dengan algoritma *Run Length Encoding* dapat dibangun untuk melakukan pengamanan pesan teks dan dapat dilakukan kompresi pada *stego object* yang digunakan.

Berdasarkan hasil pengujian proses *embedding* dan *extraction*, diperoleh bahwa algoritma MLSB menghasilkan kualitas file yang cukup baik dalam melakukan penyisipan. Berdasarkan hasil pengujian proses kompresi file menggunakan parameter pengukuran kualitas kompresi, diperoleh bahwa efisiensi kompresi algoritma RLE tidak efisien dalam memampatkan ukuran file. Faktor yang mempengaruhi *running time* secara signifikan pada proses *embedding* dan proses *extraction* adalah ukuran dari pesan teks yang digunakan.

Berdasarkan hasil pengujian proses *extraction* dengan menggunakan kunci sembarang yang bukan merupakan kunci asli, diperoleh bahwa pesan yang dihasilkan bukan merupakan pesan asli pada saat proses *embedding*.

Pengembangan penelitian selanjutnya dapat dilakukan pada proses pembangkitan bilangan acak menggunakan algoritma PRNG selain algoritma *Multiply with Carry* yang lebih efisien dan pada proses kompresi diharapkan menggunakan algoritma lain yang lebih efisien daripada algoritma *Run Length Encoding*. Selain itu, dapat juga menggunakan algoritma kompresi dengan kategori *Variable Length Encoding* dan berjenis *lossless* untuk menghindari data yang hilang setelah proses kompresi.

Daftar Pustaka

Arifin Y., Ricky, M.Y. dan Yesmaya, V., 2015. *Digital Multimedia*. Jakarta: Bina Nusantara, 2015.

Budiman, M.A., & Rachmawati, D., 2017. On Using Goldbach G0 Codes and Even-Rodeh Codes for Text Compression on Using Goldbach G0 Codes and Even-Rodeh Codes for Text Compression. IOP Conference Series: Materials Science and Engineering, 180(1).

Darwis, D., 2016. Implementasi Teknik Steganografi Least Significant Bit (LSB) dan Kompresi untuk pengamanan data pengiriman surat elektronik. Jurnal Teknoinfo, 10(2), 32.

Emam, M.M, Aly, A.A. and Omara, F.A., 2016. An Improved Image. Steganography Method Based on LSB Technique with Random Pixel Selection, Int. J. Adv. Comput. Sci. Appl., vol. 7, no. 3, pp. 361–366, 2016.

Krisnawati, 2008. Metode Least Significant Bit (LSB) dan End of File (EOF), *Semin. Nas. Inform.*, pp. 39–44, 2008.

Lubis, A.R., Lidya, M.S. and Budiman, M.A., 2012. Perancangan perangkat lunak Steganografi Audio MP3 Menggunakan Metode Least Significant Bit (LSB) dengan Visual Basic 6.0, *J. Dunia Tekno. Inf.*, vol. 1, no. 1, pp. 63–68, 2012.

Marsaglia, G., 2003. Random number generators. Journal of Modern Applied Statistical Methods, 2(1), 2–13.

Purba, J.V., Situmorang, M. dan Arisandi, D., 2012. Implementasi Steganografi Pesan Text ke dalam File Sound (.Wav) dengan Modifikasi Jarak Byte pada Algoritma Least Significant Bit (LSB)," *J. Dunia Teknologi Inf.*, vol. 1, no. 1, pp. 50–55, 2012.

Rachmawati, D., Amalia, A., & Surya, J., 2017. Combination of Huffman Coding Compression Algorithm and Least Significant Bit Method for Image Hiding. Journal of Physics: Conference Series, 801(1), 012059.

Rahandi A, Rachmawati, D. and Sembiring, S.. 2012. Analisis dan implementasi kompresi file audio dengan menggunakan Algoritma Run Length Encoding (RLE)," *J. Alkhawarizmi.*, 2012.

Ricky M., Setyaningsih, F.A., Diponegoro, M., 2018. Analisis kompresi steganography pada citra digital dengan menggunakan Metode Least Significant Bit Berbasis Mobile, *J. Coding*, vol. 06, no. 03, pp. 75–86, 2018.

Salomon, D., 2007. Variable-length Codes for Data Compression. Springer London.