Audio Steganography Method Using Least Significant Bit (LSB) Encoding Technique

Alaa Abdulsalm Alarood[1], Ahmed Mohammed Alghamdi[2], Ahmed Omar Alzahrani[3], Abdulrahman Alzahrani[4] and Eesa Alsolami[5]

[1] College of Computer Science and Engineering, University of Jeddah, Jeddah 21959, Saudi Arabia;
alaa.alarood@gmail.com, aasoleman@uj.edu.sa

[2]Department of Software Engineering, College of Computer Science and Engineering, University of Jeddah, Jeddah 21493, Saudi Arabia

[3]College of Computer Science and Engineering, University of Jeddah, 21959, Jeddah, Saudi Arabia: aoalzahrani@uj.edu.sa.

[4]College of Computer Science and Engineering, University of Jeddah, Jeddah 21493, Saudi Arabia

[5] Department of Cyber Security, College of Computer Science and Engineering, University of Jeddah, 21959 Jeddah, Saudi Arabia

*Correspondence: Alaa Alarood alaa.alarood@gmail.com, aasoleman@uj.edu.sa.

Summary
MP3 is one of the most widely used file formats for encoding and representing audio data. One of the reasons for this popularity is their significant ability to reduce audio file sizes in comparison to other encoding techniques. Additionally, other reasons also include ease of implementation, its availability and good technical support. Steganography is the art of shielding the communication between two parties from the eyes of attackers. In steganography, a secret message in the form of a copyright mark, concealed communication, or serial number can be embedded in an innocuous file (e.g., computer code, video film, or audio recording), making it impossible for the wrong party to access the hidden message during the exchange of data. This paper describes a new steganography algorithm for encoding secret messages in MP3 audio files using an improved least significant bit (LSB) technique with high embedding capacity. Test results obtained shows that the efficiency of this technique is higher compared to other LSB techniques.

Keywords:
Steganography; Least Significant Bit (LSB); MP3.

1. Introduction

The introduction should broadly describe the study, while also highlighting its significant worth. Also, the introduction should identify the purpose and significance of the study. A well thought-out review of the present research state should be presented, along with citations of main key publications. The controversial and diverging hypotheses should also be presented as needed. The research aim should be mentioned in brief, while the main conclusions are stated. It is important that the introduction is presented in a manner that is intelligible to readers from different research domain. As for references, they must be numbered in order of appearance and noted by a numeral or numerals in square brackets—e.g., [1] or [2, 3], or [4–6]. Refer to the end of the document for more details.

Safety is a crucial element because it assures confidentiality of the transferred information. Owing to this safety concern, a number of methods have been established for the purpose of ensuring message confidentiality. However, preserving the secrecy of message contents may no longer be sufficient as keeping the very existence of the message secret may be required. This necessity has led to the use of steganography. Steganography is a blend of two words in Greek language namely “stéganos” which carries the meaning of covered or secret and “graphy” which carries the meaning of writing or drawing. As such, literally, steganography carries the meaning of “covered writing.” Steganalysis is the process of detecting steganographic content. In other words, the goal of steganalysis is to detect and/or estimate the eventual hidden data. The art of steganalysis makes a major contribution to the selection of features or characteristics that might be shown by Stego-objects. Moreover, the science may provide assistance in consistently testing the features chosen for the existence of hidden information [1].

The general aim of steganography, as indicated by Kim et al. (2014), is to shield the communication that takes place between two parties from the eyes of attackers. In steganography application, a secret message in the form of a copyright mark, concealed communication, or serial number can be embedded in an innocuous file (e.g. computer code, video film, or audio recording), impeding the wrong party from accessing the concealed message during the exchange of data. Kim et al. (2014) described a cover message incorporating a secret image as a stego-object. Following the exchange of data, both the receiver and the sender should destroy the cover message to prevent

Manuscript received July 5, 2022
Manuscript revised July 20, 2022
https://doi.org/10.22937/IJCSNS.2022.22.7.53

accidental reuse. Figure 1 presents the fundamental model of a steganographic system [2].

From this section, input the body of your manuscript according to the constitution that you had. For detailed information for authors, please refer to [1].

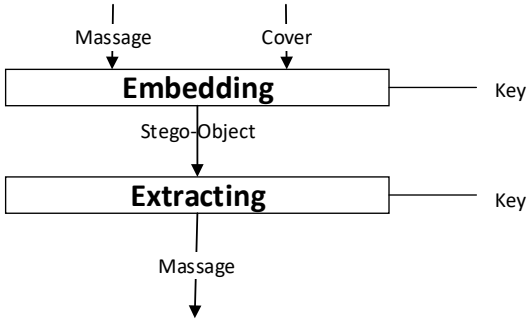

Figure 1 Basic model of steganography.

Hiding data requires an embedding algorithm and an extracting algorithm, whereby embedding algorithm conceals secret messages within a cover message. Here, a key word is used to protect the process of embedding. This ensures that the hidden message would be accessible to only those with the secret key word. Meanwhile, extracting algorithm is applied on a feasibly modified carrier and brings back the concealed secret message [2].
In audio data encoding and representation, MP3 is among the most commonly used file formats [3]. Such popularity of MP3, which is an acronym for Moving Picture Experts Group MPEG-1 Audio Layer 3 [4], has been factored by their significant ability in decreasing audio file sizes as opposed to other techniques of encoding.

The main strengths of MP3 format include its efficiency and effectiveness in reducing the size of audio file while maintaining quality. This has many benefits, such as reducing the disk space needed to store audio files, which has a great impact in reducing the amount of time needed to share and transfer such files. However, MP3 loses some data during the compression process. In fact, it was reported by a number of experts, who listened to MP3 sample files, that there is a slight difference between the coded and original audio tracks [5]. There have been many attempts to solve this problem, and one suggestion was to use MPEG algorithms that can reduce data loss during compression, thus moving towards lossless compression.
Away from the fact that MP3 compression loses data, there are many reasons to consider MP3 as one of the most popular audio compression technologies [6]. These include, but are not limited to, the following:

Ease of implementation: As no single company owns the MP3 is open to all (open standard) [6].
Availability: Many professionals prefer MP3 because of the wide range of MP3 encoders and decoders available in the market to meet their demands [3].
Support: Developments in computer technologies in general (processing power), specifically in sound cards, the spread of hardware such as CD-ROMs and CD-audio writers, and the rising popularity of the internet have all contributed to the increased distribution of audio files in MP3 format. In other words, MP3 was introduced at just the right time [7].

2. Related Work

Encoding refers to the process of compressing the WAV file by reducing the size of the original digital sound file so that it takes up less space. An algorithm that optimizes audio perception is used to maintain quality, and data that do not contribute to this perception are lost.
Different MP3 encoders use one of the following bit rates: CBR (Constant Bit Rate), VBR (Variable Bit Rate), and ABR (Average Bit Rate) [8].

Basic encoders use CBR, whereby every frame uses the exact bit rate in the audio data stream. This means that there will be a fixed bit rate in the whole MP3 file, resulting in variations in quality. The advantage of this mechanism is the possibility to predict the size of the encoded file by multiplying the song length by the bit rate [8].
When using the VBR technique, it is possible to maintain quality while encoding, but the file size cannot be predicted. ABR works by adding extra bits to parts of the audio file that require an increase in quality; this approach enhances quality significantly, while keeping the average file size within predictable ranges. The next sections define and discuss the MP3 file format and frame header

2.1. MP3 File Format

The encoding method determines the content of the MP3 file. In general, any MP3 file consists of three components: tags, padding bytes, and frames (see Figure 2).

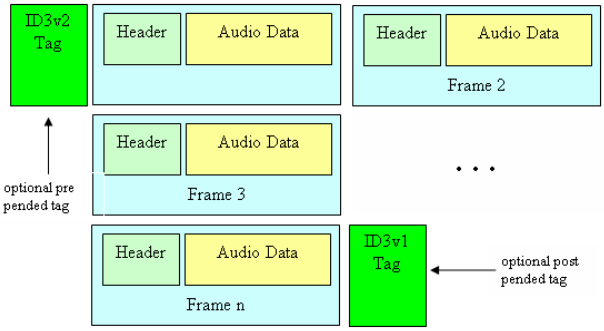

Figure 2. MP3 file structure.

Tags have two different formats, ID3v1 and ID3v2. Between the two, ID3v1 is an old format that post-pends 128-bits at the end of the audio file in the form of seven fields (genre, artist name, album, song title, and so on). However, this format suffers from two main drawbacks: lack of flexibility and static size. Therefore, ID3v2 has been used as replacement as it is more flexible and has advanced format [9]. ID3v2 allows the tag to be pre-pended to the file. The ID3v2 frames could store data of various types, such as the artist name, song title, encoding process, and much more. This type of tag has two main advantages: an unlimited setting size and the ability to provide hints for the encoder [10].Additional data appended to the frame for filling purposes in the encoding process are called padding bytes. These bytes are only used in CBR to assure frames of identical size [11]. Frames are made up of two main parts namely audio data and a file header. These two parts are discussed in more detail in the next section.

2.2. MP3 Frame Headers

In MP3 files, a series of bits are representative of the header. These headers either commence with 0 or 1, where 1 means block synchronization (see Table 1) [12]. A frame consists of a 12-bit stream correspondingly for 1s. It should be noted that there is no unique frame for any specific header. This means that a frame can be found in any longer data block. In general, to recognize a 4-byte data block as a header, certain conditions must be satisfied as follows [13]:

The Layer field cannot be 00
The Frequency field cannot be 11
The Bit-rate field cannot be 0000 or 1111

Table 1 MP3 frame header

Frame Name	length (bits)	Position(bits)	Description
SYNC	12	(1-12)	Frame sync (all bits must be set)
MPEG version	2	(13,14)	MPEG Audio version ID
MPEG Layer	2	(15,16)	Layer description
Protection	1	(17)	Protection bit 0 - Protected by CRC (16bit CRC follows header) 1 - Not protected
Bitrate index	3	(18,20)	<table><tr><th>bitrate</th><th>single channel</th><th>stereo</th><th>intensity stereo</th><th>dual channel</th></tr><tr><td>96</td><td>yes</td><td>Yes</td><td>yes</td><td>yes</td></tr><tr><td>128</td><td>yes</td><td>Yes</td><td>yes</td><td>yes</td></tr><tr><td>192</td><td>yes</td><td>Yes</td><td>yes</td><td>yes</td></tr><tr><td>256</td><td>no</td><td>Yes</td><td>yes</td><td>yes</td></tr><tr><td>320</td><td>no</td><td>Yes</td><td>yes</td><td>yes</td></tr></table>
Sampling rate	2	(21,22)	Sampling rate frequency index, bits=00, MPEG1=44100 Hz
Padding bit	1	(23)	0 - frame is not padded 1 - frame is padded with one extra slot
Padding bit	1	(24)	This one is only informative.
Channel Mode	2	(25,26)	00 - Stereo 01 - Joint stereo (Stereo) 10 - Dual channel (2 mono channels) 11 - Single channel (Mono)
Mode extension	2	(27,28)	Only used in Joint stereo
Copyright	1	(29)	0 - Audio is not copyrighted 1 - Audio is copyrighted
Original	1	(30)	0 - Copy of original media 1 - Original media
Emphasis	2	(31,32)	Indication is here to tell the decoder that the file must be de-emphasized.

The frame size in the 4-byte block that begins with the Sync and is in compliance with the above stipulations is not always clear [10], and so it is better to determine the two ends of the frame. This task should be easy, as all headers have similar contents and structures. The equation below can be used to find the frame size [13] [14].

$$\text{Frame Size (FS)} = \frac{144 \times \text{Bitrate}}{\text{Sample Rate} + \text{Padding}} \quad (1)$$

Bit Rate: measured in bits per second.
Sample Rate: denotes the rate of sample of the original data.
Padding: denotes the additional data appended to the frame to fully fill it during the encoding process [12].
In many cases, there will be some unemployed bits (fields). Additionally, padding stuffing is available as well which can help steganography designers in secret data embedding process.

2.3. Steganography Categories

Steganography comes in three main types: pure steganography, secret key steganography, and public key steganography. Each is explained in the following sections.

2.3.1 Pure Steganography

Pure steganography has no requirement for the preceding exchange of certain secret information, for instance, a stego-key. The embedding process is describable by the mapping $E: C \times M \rightarrow C$. Meanwhile, the extraction process which includes secret message extraction from a cover message is illustratable by the mapping $D: C \rightarrow M$. Here, C denotes the set of probable covers, while M denotes the set of probable messages namely $|C| \geq |M|$.

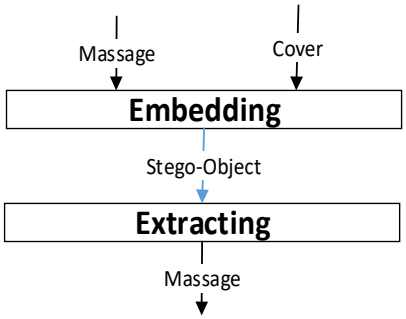

Figure 3 Pure steganography.#

In pure steganography, only the sender and receiver are allowed access to the employed algorithms during the embedding and extraction processes [15]. In other words, the public have no access. However, considering that the sender and receiver depend only on the supposition that this secret message is not known by other parties, it becomes a drawback of this method; it lacks security. Pure steganography is illustrated in Figure 3.

2.3.2 Secret Key Steganography

Secret key steganography involves the use of a secret key (stego-key) to be exchanged before communication. Employing this stego-key; secret key steganography comprises the embedding of the secret message within a cover message. Parties that have access to the secret key can read the message.

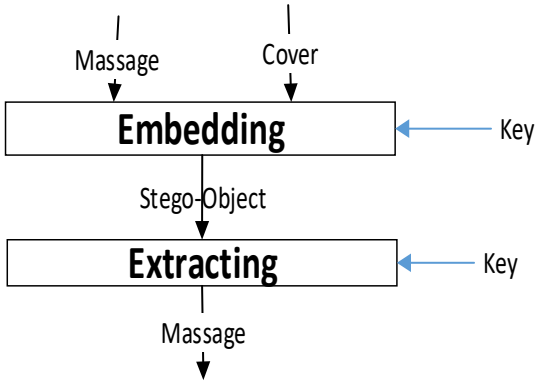

Figure 4 Secret key steganography.

Secret key steganography involves a stego-key exchange; this is different from pure steganography, which contains a perceived invisible communication channel (the reason why pure steganography is more prone to interception). In secret key steganography, even when the cover message is intercepted, only those parties with access to the secret key are allowed access to the secret message [16]. This becomes an advantage of secret key steganography. Figure 4 accordingly illustrates the process of secret key steganography.

2.3.3 Public Key Steganography

Public key steganography is underpinned by the notion of public key cryptography. This type of steganography (public key steganography) involves the use of both public key and private key in assuring that parties are in secure communication. This method entails the use of public key by sender during encoding process. To decipher the secret message, the sender uses only a private key with a direct mathematical linkage to the public key. Public key steganography is more robust because it employs a technology with greater level of robustness and that is well-researched in the field of public key cryptography. Public key steganography is also layered with multiple levels of security. Therefore, the secret message is difficult to access; many attempts have to be made to crack the employed algorithm in the public key system, and only then, the secret message can be intercepted [16]. Public key steganography is illustrated in Figure 5.

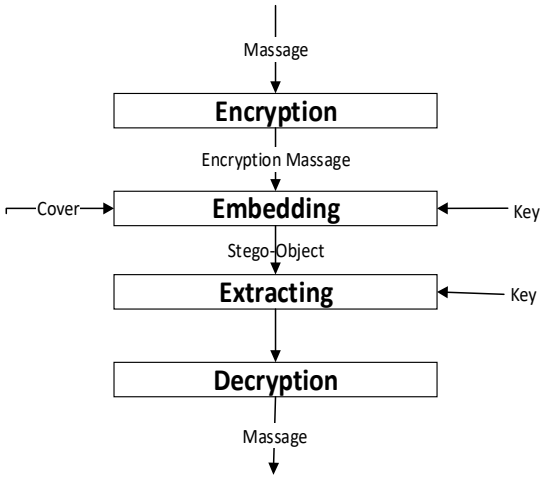

Figure 5 Public key steganography.

However, the use of public key steganography increases the complexity of the steganographic technique while decreasing the size of the secret message to be embedded. This is because the encryption algorithms cause the size of the message to expand more than double its original size.
A new model for steganography will be constructed. This model will have the capacity to embed text messages in MP3 audio cover messages using a technique known as the least significant bit (LSB).

2.4 Least Significant Bit (LSB)

Being the first and most straightforward technique, the LSB method is utilized in the embedding of secret messages into audio files or other file types. Using this method, the

audio file and the secret message are first transformed into a bit stream, and then the secret message is embedded into the audio file. The embedding is performed by swapping the LSB of the audio with one (or more) bit of the secret message before sending it to the receiver [17]. The secret message is extracted by the receiver by accessing the sequence of sample indices that are utilized during the process of embedding. The amount of information that can be embedded in the audio file is 1 kbps per kHz, which is considered low. However, in steganography, the capacity and robustness are crucial components. Wakiyama et al. (2010) has in fact reported a significant improvement in both capacity and robustness.

In order to increase the audio steganography capacity, Kekre et al. (2010) proposed two algorithms that use LSB techniques. In these algorithms, the message bits are embedded into multiple and variable LSBs. The aforementioned method comprises the use of up to seven LSBs for data embedding. The first proposed algorithm will examine the first two Most Significant Bits (MSBs) of the cover samples. Here, the two MSB values in 4–7 LSBs are used in the secret message embedding. This results in the capacity increasing to 5.563 bits per sample from 4 bits per sample. Moreover, the second algorithm extends the first method by checking the MSB of the cover samples, whereby, if it is 0, six LSBs were used, whereas, if it is 1, seven LSBs were used for secret message embedding. These algorithms increase the capacity to 6.574 bits per sample [18].

The secret message should be encrypted prior to its embedding into audio signals [19]. In the work of Ozighor and Izegbu (2020), the higher bit indices replaced the traditional LSB. This resolved the issue of robustness by concealing the data from hackers and safely transmitting them to the specified destination. This proposed method does not alter the file size and is suitable for all audio file formats. Moreover, the sound quality is dictated by the user-selected audio quality and the message length. Some trade-offs between the robustness, capacity, and imperceptibility were highlighted by [22]. In relation to this matter, Devaraj et al. (2009) mentioned that the trade-off between noise acceptance and capacity is impacted by higher bit indices, and this is necessary for a reasonably imperceptible embedding [20].

The lifting scheme was proposed by Shahreza and Shalmani in (2007) in the creation of perfect reconstruction filter banks, namely, Int2lnt. This scheme can also adaptively conceal data in the LSB of the details coefficient. This reduces the error rate in wavelet domain steganography. As reported by [21], this method generates error-free outcomes with a capacity of less than 100 kbps and up to 200 kbps.

2.5. Dataset

This research uses MP3 files as the carrier files. We took 72 MP3 files with various ratios of compression, sizes, genres, and sampling frequencies (320 kbps, 256 kbps, 196 kbps, 128 kbps, and 96 kbps) as the standard dataset [3].

3. The Proposed Method

The embedding process in steganography works by moving bits from one place to another with the intention of inserting additional bits into the carrier. For this study, the carrier bits formed MP3 files, while the embedded bits were from text files. During the embedding, different file formats have different bit insertion methods, as different MP3 file compression ratios were used in this study. The LSB technique is the main embedding procedure. The embedding process in this study is described below, and the steps for pre-processing and embedding are as follows.

3.1. Acquisition of File Properties

The first step in this process is to read the MP3 file. This function reads the MP3 file and takes its name and extension as input arguments. Then, it reads the MP3 file format and returns an output argument as the analogue value of the audio samples only. This function also returns the properties of the MP3 file, frequency of sampling, and number of bits. The header and time frame are removed to simplify the data structure and supply only meaningful data. The next step is to read the text file to be embedded inside the MP3 in order to generate the stego MP3 file.

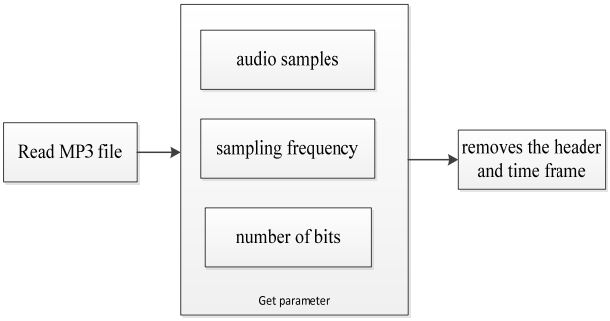

Figure 6. Obtaining the parameters of an MP3 file.

The MP3 file parameters are measured and estimated. This is to ascertain the data size and encoding necessary for the embedding position. The embedding is initiated from a random location inside the MP3 file. To obtain a random location, the following steps are implemented:

Step 1: Select MP3 file for a hidden message to be embedded.

Step 2: Load the MP3 and gather file information such as audio samples, MP3 properties, frequency of sampling, number of bits, and sample rate.
Step 3: Calculate Irand, which specifies the random location inside the MP3 file at which the embedding starts, by the equation below:

Irand = ceil(rand * fix(Espace/2))+200

Step 4: Espace is computed using the following equation, where "r" denotes the number of samples in the MP3 file, "rb*cb" denotes the size of the text message file, and "deg" denotes the number of insertion bits. The equation is as below:

Espace = r - rb*cb/deg – 200

Step 5: The "rand" function generates a random number in the range 0–1.
Step 6: The result is a random starting location for the embedding that ensures the end of insertion will be placed within the MP3 file.

3.2. File Conversion

The digital handling of the text file involves the conversion of all text data into digital format. In the process, first, the text file is converted to ASCII. Here, the text data string becomes the input argument and the ASCII code for each character is returned. As an example:

```
>> double("Aoun")
ans =   65   111   117   110
```

This function generates result in decimal format, rather than hexadecimal. Notably, the result does not have to be converted into hexadecimal format, but should be converted directly into binary format. For each character, it is discretely converted to binary, and the binary conversion results in a two-dimensional matrix as exemplified below: Consider the string "Aoun", where the row denotes the character, while the column represents the binary code for a specified character.

```
>> dec2bin(ans')
ans =
01000001
01101111
01110101
01101110
```

This process uses the following function to convert decimal numbers to binary:

$$\mathbf{Bdi} = \mathbf{rem}(\frac{\boldsymbol{Dd_{i+1}}}{\mathbf{2}}\mathbf{2}) \quad (2)$$

Where: Bdi denotes the binary digit index, Ddi+1 denotes the result of decimal digit division, and "rem" denotes the division remainder. The result is a two-dimensional matrix of size R×8, where 8 denotes the number of bits for ASCII character conversion to binary, while R denotes the number of characters within the text file. R includes alphanumeric characters as well as ASCII symbols such as space and carriage return, as shown in Figure 6. The following steps describe the overall process:
Step 1: Select the text files to be tested.
Step 2: Load text files that will be converted to digital.
Step 3: Convert to ASCII format using the following function:
Double ('text')
Step 4: The single characters (decimal number) are discretely converted to binary by the following:

Bdi = rem (Ddi+1 /2)

Where: Bdi signifies the binary digit index, Ddi+1 signifies the decimal digit division result, while "rem" signifies the division remainder.
Step 5: Save the results in a two-dimensional array of size R×8, where 8 is the number of bits in the binary, while R represents the number of characters within the text file.

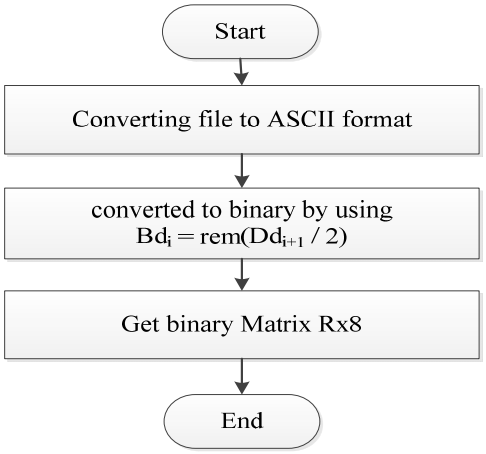

Figure 7. Converting a text file to ASCII.

The raw data for this research must be normalized to prepare them for the final analysis. In some studies, researchers have argued that working with raw data is more suitable than working with normalized data, although other researchers claimed that working with normalized data enhances the accuracy of the model being tested. This shows that there is no consensus over whether to use normalized or raw data. However, in this research situation, raw data in their original forms are analogue MP3 and text coded files, and it is compulsory to transform them into binary format and scale them down to a uniform format suitable for processing by analytical tools. The normalization procedure for this research is illustrated in Figure 8. The MP3 files are read and expressed as analogue values, and then each MP3 sample has a floating-point value in the interval [−1, +1]. In theory, floating point numbers are a rough calculation in digital systems. For this reason, any processing of them will include an accrued error. In order to manage this analogue value with a minimal error

which nears the value of zero, we normalize to a higher value as follows:

$$A_{iN} = (A_i + 1) * 106 \quad (3)$$

where: AiN signifies the ith normalized sample of the audio array A. Thereafter, the normalized samples are converted into their corresponding binary format. In this regard, the conversion function is identical to that carried out for text conversion, and the steps are as follows:

Step 1: Select all MP3 files to be embedded.

Step 2: Load the MP3 files to be converted to digital.

Step 3: Convert each MP3 sample to an analogue value in the interval [−1, +1] with floating point format using the following function: Sprintf(each MP3 sample)

Step 4: Normalize the analogue value to a higher value by multiplying it by 1×106 to minimize the error. Fsample = mp3original(i,1) * 1000000.

Step 5: Convert the bit to an absolute value using the following function:

```
If (Fsample < 0)
signA = 1;
Fsample = -1 * Fsample;
 else
signA = 0.
 end
```

Step 6: Convert the floating point numbers in the digital system by the above process.

Step7: Convert the sample to the corresponding binary format by the following algorithm:

Btemp = dec2bin (Fsample).

Step 8: Save the results in a one-dimensional array of size R, where R represents the number of samples within the MP3 file.

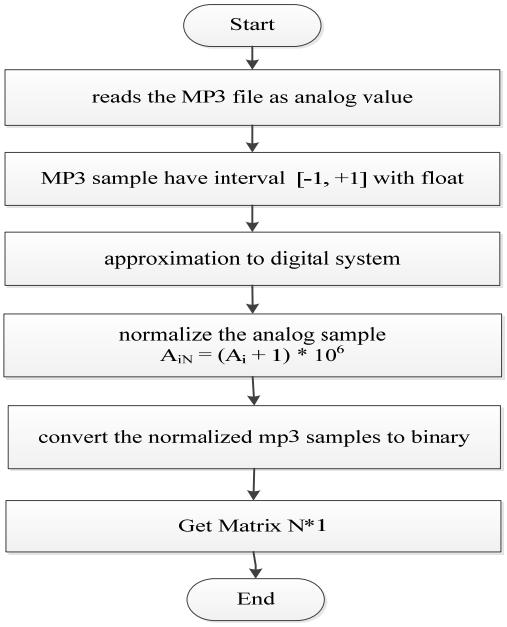

Figure 8. Converting an MP3 file to a bit stream.

3.3 Build Stego Bit Stream

Building a stego-object directly means undergoing a steganographic process that hides a secret message in a carrier file. This study continues from the previous step (normalization). Once the audio data have been normalized and converted to binary, whereas the text file is transformed into ASCII and then binary, the data are set for the stego-file formation. The procedure starts by embedding the binary from the text into the binary representing the audio. This technique of embedding is initiated from a random location within the MP3 file. In this regard, it is necessary that the embedded message is confined within its start location and end location, whereby the former follows the start signature or key, while the latter is just prior to the end signature or key. There is also an avenue for multi-bit insertion; therefore, the key should carry information pertaining to the number of insertion bits. Table 2 summarizes the four signatures to be embedded at the start and end of each and every message.

Table 2. Key or signatures

LSB Insertions	key, signature
Single-Bit Insertion	10101010101010, 10101010101010
Two-Bit Insertion	01010101010101, 01010101010101
Three-Bit Insertion	10101010101010, 01010101010101
Four-Bit Insertion	01010101010101, 10101010101010

The exact signature is applied for the start and end of the embedding, as shown for single-bit insertion. Thus, the message is confined within the exact signature which indicates the start and end of the message, along with the number of insertion bits. Here, the insertion method simply takes a bit or number of bits from the message data and inserts it to the carrier data, as described in Table 2 for four-bit insertion. Figure 9 shows the addition of keys or a signature for single-bit insertion and four-bit insertion of a secret message.

Figure 9. Adding a signature to a text file.

The procedure in Figure 10 demonstrates how one bit and two bits are inserted from the message to the carrier. The process of insertion is carried out involving the least-significant bit. By starting from the least significant bit, the perceptible impact of the real digital value becomes insignificant. Hence, the insertion will not result in any defects and will not impact the original audio data as judged by the individual who listens to the MP3 file.

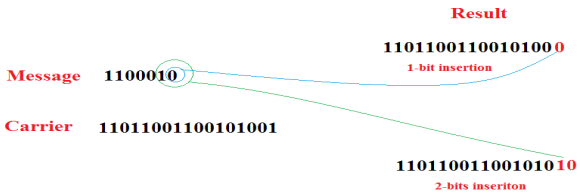

Figure 10. Insertion of one and two bits.

The developed system includes four scenarios of insertion in accordance with the number of bits. In this regard, single-bit insertion, two-bit insertion, three-bit insertion, and four-bit insertion can be implemented according to the input arguments of the developed program. Figure 9 displays the insertion of a single bit and two bits. The scheme is unique, as this research ensures that the insertion procedure continues until all bits of text data have been inserted inside the digital carrier data of the audio MP3 file (see Figure 10). The following steps illustrate the process:

Step 1: Select every MP3 and text file to be embedded.
Step 2: Load the MP3 and text files after converting them into binary bits.
Step 3: Select a random location in the MP3 file to start embedding the text file (see section 4.3.1).
Step 4: Add a signature or key at the beginning and end of the text file, as shown in Table 2.
Step 5: Add the number of insertion bits (1, 2, or 4), as shown in Figure 7.
Step 6: Examine the sizes of the MP3 and text files to start the process of embedding.
Step 7: Start to replace the binary bits from the text in the MP3 file.
Step 8: Upon completion of the embedding process, obtain the bit steam.

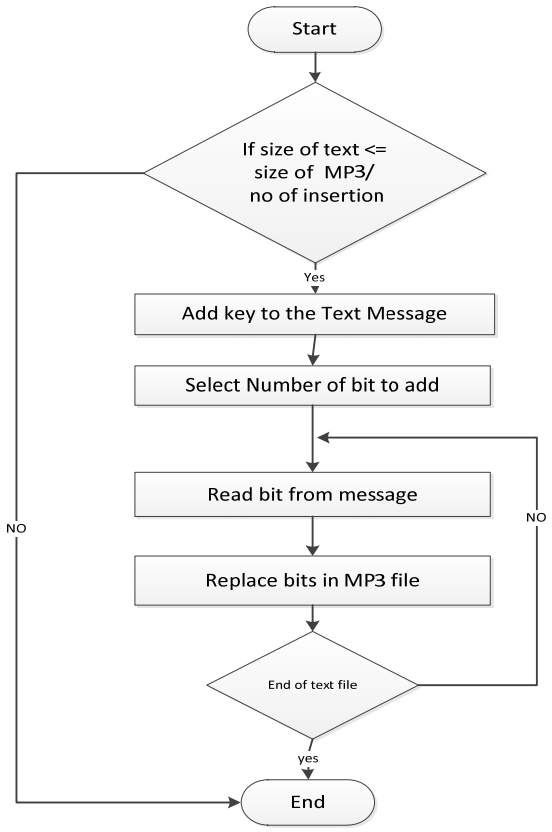

Figure 10. Replacing bits in the carrier file.

3.4. Converting a Bit Stream to an MP3 Stego-Object

This process begins when all of the binary samples of text have been inserted into the carrier file; this is actually the MP3 stego-object, which is now converted to an MP3 format file. To perform the inverse process in the post-processing phases, the digital selection representing the MP3 stego-object is first converted to decimal. The entire inverse process is carried out according to:

$$Dd = \sum b^i * 2^i \quad (4)$$

where: Dd signifies the decimal digit given by conversion, bi signifies the ith binary bit value, and i signifies the index of the binary bit. The index i takes values from 0–22. As such, the maximum normalized decimal number is 2×106. Therefore, the maximum value of i is 22.

The resulting decimal data are normalized following the aforementioned process of normalization. For this reason, de-normalization needs to be carried out to obtain the

original analogue audio format. This is carried out according to:

$$A_i = (A_{iN} * 10^{-6}) - 1 \quad (5)$$

where: Ai is the de-normalized analogue audio sample and AiN is the normalized analogue sample, which now becomes a stego sample. The following steps illustrate the process:

Step 1: For each MP3, convert to a bit stream and insert text.
Step 2: Convert bits to decimal using the following equation:

Dtemp = bin2dec(Btemp)

Step 3: De-normalize bits by the following equation:

Dtemp = bin2dec(Btemp) / 1000000;

Step 4: Convert bits to an absolute value by the following procedure:

If (signA == 1)
Dtemp = -1 * Dtemp;
Else
Dtemp = Dtemp;
End

Step 5: Convert floating point numbers in the digital system by the above process.
Step 6: Recover the original analogue MP3 format.
Step 7: Save the MP3 stego-object under a new file.

Figure 10 explains the process of the LSB embedding technique used to build MP3 stego-objects, including the transformation of MP3 and text files, normalization of MP3 samples, and embedding of text bits into MP3 file. The inverse process is described by the following steps:
Step 1: Select all MP3 files to be embedded.
Step 2: Read the MP3 files.
Step 3: Read the text message files.
Step 4: Check the size of the MP3 and text message; if it could be embedded, go to Step 5: otherwise, stop and output: Audio file size is less than that required for the text file.
Step 5: Select random start location in the MP3 file.
Step 6: Select the character message.
Step 7: Convert ASCII text into binary.
Step 8: Select a sound sample.
Step 9: Normalize the sound sample.
Step 10: Convert the MP3 sample into binary.
Step 11: Insert character message into the MP3 sample.
Step 12: Convert binary MP3 result into decimal.
Step 13: If (charCount = max), end the process; otherwise, go to Step 6.
Step 14: Save the MP3 file.

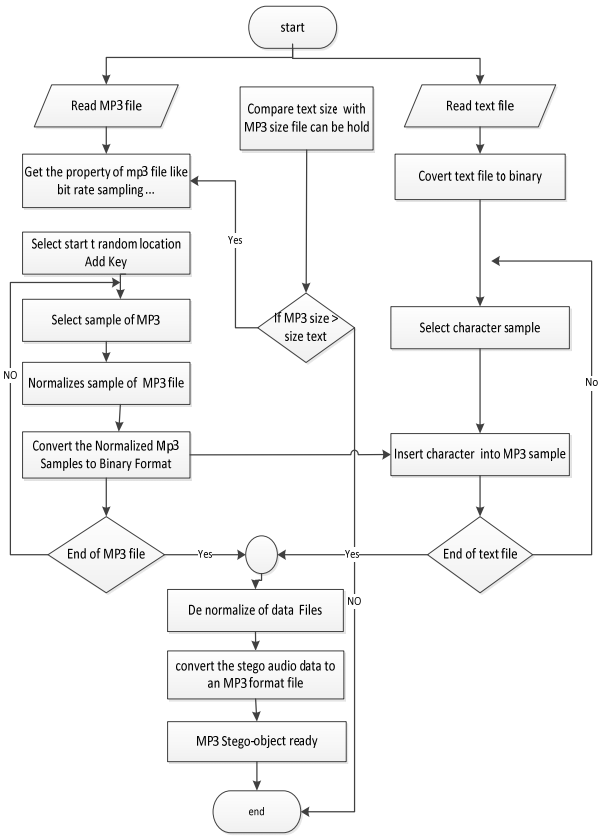

Figure 11 Embedding a text file in an MP3 file

4. Result and Discussion

The Peak Signal-to-Noise Ratio (PSNR) encompasses the ratio between a signal's maximum power and the power of the signal's noise. The application of PSNR has been common among engineers in their measurement of the quality of compressed reconstructed signals. Considering that signals can have an extensive dynamic range, PSNR is generally expressed in decibels as show comparisons between embedded different bits. In statistics, the difference between values inferred by an estimator and the true values of the quantity being estimated can be measured using the Mean Squared Error (MSE) of an estimator. In specific, MSE entails a risk function that corresponds to the anticipated value of the squared error loss or quadratic loss. It measures the average of the squares of the "errors," whereby an error entails the amount by which the value inferred by the estimator is distinct from the quantity to be appraised, as show Comparisons between embedded different bits, the secret message is text data using the secret messages: Data-1 is equal 1 KB and the datasets have different sizes and bitrates.

Table 3 presents the PSNR results for Data-1 with the application of the LSB technique for embedding MP3 files.

Here, the secret message is embedded in 1, 2, and 4 LSBs for various genres including Blues, Classical, Country, Dance, Hip-Hop, Jazz, Metal, Pop, R&B, Rap, Reggae, Rock, under a 320-kbps rate of compression with the cover MP3 file sizes and time. The results demonstrate superior imperceptibility of 1-LSB as opposed to 2-LSB and 4-LSB for all various types of genre. On the other hand, 4-LSB shows imperceptibility that is worse when compared to 1-LSB and 2-LSB. Accordingly, the average values for PSNR are correspondingly 79.14779, 73.12698, and 64.0964 for embedding secret messages in 1, 2, and 4 LSBs. The highest values for 1-LSB (80.8612), 2-LSB (76.8849), and 4-LSB (65.8695) occur in the Metal genre. The lowest values for 1-LSB (77.5252), 2-LSB (73.5076), and 4-LSB (62.3538) occur in the Classical genre. The Rap and Reggae content has the same file size (9.14 MB) and time (3:59 min), but the values for 1-LSB, 2-LSB, and 4-LSB are different. Rap achieves better values for 1-LSB (77.5252), 2-LSB (73.5076), and 4-LSB (62.3538) than Reggae.

Table 1 PSNR results for Data-1 at 320 kbps compression rate

Genre	**Time (min)**	**Size (MB)**	**1-LSB 1st position**	**2-LSB 1st and 2nd**	**4-LSB 1st , 2nd and 4th**
Blues	4:41	10.7	79.56	75.59	64.48
Classical	2:54	6.67	77.52	73.50	62.35
Country	3:42	8.48	78.43	74.55	63.48
Dance	6:12	14.2	80.71	76.72	65.65
Hip-hop	5:27	12.4	80.15	76.28	65.14
Jazz	3:12	7.34	77.96	73.93	62.79
Metal	6:28	14.8	80.86	76.88	65.86
Pop	4:00	9.16	78.87	74.85	63.78
R&B	3:51	8.81	78.65	74.63	63.60
Rap	3:59	9.14	78.87	74.83	63.77
Reggae	3:59	9.14	78.79	74.8	63.79
Rock	4:33	10.4	79.35	75.40	64.40

Table 4 presents the PSNR results for Data-1 using the LSB technique to embed MP3 files, where the secret message is embedded in 1, 2, and 4 LSBs for different genres (Blues, Classical, Country, Dance, Hip-Hop, Jazz, Metal, Pop, R&B, Rap, Reggae, Rock) under a 256-kbps compression rate with the cover MP3 file sizes and time. The results show that the imperceptibility of 1-LSB is better than that of 2-LSB and 4-LSB for all genre types. The imperceptibility in 4-LSB is again worse than that of 1-LSB and 2-LSB. The average PSNR values are 78.164, 73.1735, and 63.13678 for embedding the secret message in 1, 2, and 4 LSBs, respectively. The highest values for 1-LSB (80.0109), 2-LSB (74.8849), and 4-LSB (64) occur in the Metal genre. The lowest values for 1-LSB (76.4867), 2-LSB (71.5076), and 4-LSB (61.4076) occur in the Classical genre. The Rap and Reggae content has the same file size (9.14 MB) and time (3:59 min), but the values for 1-LSB, 2-LSB, and 4-LSB are different. Rap achieves better values for 1-LSB (77.8309), 2-LSB (72.8357), and 4-LSB (62.8812) than Reggae.

Table 2 PSNR results for Data-1 at 256 kbps compression rate

Genre	**Time (min)**	**Size (MB)**	**1-LSB 1st position**	**2-LSB 1st and 2nd**	**4-LSB 1st , 2nd and 4th**
Blues	4:41	6.44	78.58	73.59	63.49
Classical	2:54	4	76.48	71.50	61.40
Country	3:42	5.08	77.45	72.55	62.53

Dance	6:12	8.53	79.78	74.72	64.79
Hip-Hop	5:27	7.48	79.22	74.28	64.15
Jazz	3:12	4.4	76.90	71.92	61.82
Metal	6:28	8.88	80.01	74.88	64.92
Pop	4:00	5.49	77.87	72.85	62.79
R&B	3:51	5.29	77.57	72.66	62.65
Rap	3:59	5.48	77.83	72.83	62.88
Reggae	3:59	5.48	77.82	72.87	62.79
Rock	4:33	6.26	78.39	73.40	63.38

Table 5 presents the PSNR results for Data-1 with the use of the LSB technique for MP3 files embedding. Here, the secret message is embedded in 1, 2, and 4 LSBs for a multitude of genres (Blues, Classical, Country, Dance, Hip-Hop, Jazz, Metal, Pop, R&B, Rap, Reggae, and Rock) at a 192-kbps rate of compression with the cover MP3 file sizes and time. The results demonstrate better degree of imperceptibility of 1-LSB in comparison to 2-LSB and 4-LSB for every type of genre. 4-LSB shows imperceptibility that is worse than that of 1-LSB and 2-LSB. The average PSNR values are correspondingly 76.906, 71.8177, and 60.8678 for embedding the secret message in 1, 2, and 4 LSBs. The highest values for 1-LSB (78.6795), 2-LSB (73.6608), and 4-LSB (62.6441) can be observed in the Metal genre. The lowest values for 1-LSB (75.2853), 2-LSB (70.1953), and 4-LSB (59.1751) occur in the Classical genre. The Rap and Reggae content has the same file size (9.14 MB) and time (3:59 min), but the values for 1-LSB, 2-LSB, and 4-LSB are different. Rap achieves better values for 1-LSB (76.5595), 2-LSB (71.5365), and 4-LSB (60.4908) than Reggae.

Table 3 PSNR results for Data-1 at 192 kbps compression rate

Genre	**Time (min)**	**Size (MB)**	**1-LSB 1st position**	**2-LSB 1st and 2nd**	**4-LSB 1st , 2nd and 4th**
Blues	4:41	6.44	77.27	72.30	61.24
Classical	2:54	4	75.28	70.19	59.17
Country	3:42	5.08	76.21	71.29	60.26
Dance	6:12	8.53	78.45	73.51	62.44
Hip-Hop	5:27	7.48	77.93	72.97	61.93
Jazz	3:12	4.4	75.68	70.65	59.58
Metal	6:28	8.88	78.67	73.66	62.64
Pop	4:00	5.49	76.63	71.54	60.54
R&B	3:51	5.29	76.47	71.37	60.43
Rap	3:59	5.48	76.55	71.53	60.49
Reggae	3:59	5.48	76.57	70.56	60.54
Rock	4:33	6.26	77.09	72.18	61.12

Table 6 presents the PSNR results for Data-1 utilizing the LSB technique for MP3 files embedding. Here, the secret message is embedded in 1, 2, and 4 LSBs for various genres (Blues, Classical, Country, Dance, Hip-Hop, Jazz, Metal, Pop, R&B, Rap, Reggae, and Rock) under a 128-kbps rate of compression with the cover MP3 file sizes and time. The results show superior imperceptibility of 1-LSB to that of 2-LSB and 4-LSB for all types of genre. Meanwhile, 4-LSB

shows imperceptibility that is worse than that of 1-LSB and 2-LSB. The average PSNR values are correspondingly 75.1036, 70.1388, and 59.0868 for embedding the secret message in 1, 2, and 4 LSBs. The highest values for 1-LSB (76.8599), 2-LSB (71.9001), and 4-LSB (60.827) occur in the Metal genre. The lowest values for 1-LSB (73.4161), 2-LSB (68.481), and 4-LSB (57.3223) occur in the Classical genre. The Rap and Reggae content has the same file size (9.14 MB) and time (3:59 min), but the values for 1-LSB, 2-LSB, and 4-LSB are different. Rap achieves better values for 1-LSB (74.878), 2-LSB (69.87), and 4-LSB (58.8803) than Reggae.

Table 4. PSNR results for Data-1 at 128 kbps compression rate

Genre	**Time (min)**	**Size (MB)**	**1-LSB 1st position**	**2-LSB 1st and 2nd**	**4-LSB 1st, 2nd and 4th**
Blues	4:41	4.29	75.48	70.42	59.40
Classical	2:54	2.66	73.41	68.48	57.32
Country	3:42	3.39	74.38	69.568	58.54
Dance	6:12	5.68	76.75	71.70	60.62
Hip-Hop	5:27	4.99	76.08	71.17	60.09
Jazz	3:12	2.93	73.81	68.82	57.79
Metal	6:28	5.92	76.85	71.90	60.87
Pop	4:00	3.66	74.77	69.87	58.78
R&B	3:51	3.52	74.66	69.63	58.58
Rap	3:59	3.65	74.88	69.87	58.88
Reggae	3:59	3.65	74.7	69.78	58.79
Rock	4:33	4.17	75.36	70.41	59.37

Table 7 presents the PSNR results for Data-1 with the usage of the LSB technique for MP3 files embedding, whereby the secret message is embedded in 1, 2, and 4 LSBs for a variety of genres (Blues, Classical, Country, Dance, Hip-Hop, Jazz, Metal, Pop, R&B, Rap, Reggae, Rock) under a 96-kbps rate of compression with the cover MP3 file sizes and time. The results demonstrate superior imperceptibility of 1-LSB when compared with 2-LSB and 4-LSB for all kinds of genre. Further, 4-LSB has imperceptibility that is worse when compared to 1-LSB and 2-LSB. The average PSNR values are correspondingly 73.8674, 68.8632, and 57.8575 for embedding the secret message in 1, 2, and 4 LSBs. The highest values for 1-LSB (75.6018), 2-LSB (70.6062), and 4-LSB (59.5504) occur in the Metal genre. The lowest values for 1-LSB (72.2153), 2-LSB (67.1305), and 4-LSB (56.2055) occur in the Classical genre. The Rap and Reggae content has the same file size (9.14 MB) and time (3:59 min), but the values for 1-LSB, 2-LSB, and 4-LSB are different. Rap achieves better values for 1-LSB (73.5747), 2-LSB (68.5646), and 4-LSB (57.4667) than Reggae.

Table 5 PSNR results for Data-1 at 96 kbps compression rate

Genre	**Time (min)**	**Size (MB)**	**1-LSB 1st position**	**2-LSB 1st and 2nd**	**4-LSB 1st, 2nd and 4th**
Blues	4:41	3.22	74.29	69.29	58.20
Classical	2:54	2	72.21	67.13	56.20
Country	3:42	2.54	73.15	68.30	57.27
Dance	6:12	4.26	75.56	70.41	59.43
Hip-Hop	5:27	3.74	74.90	69.89	58.85

Jazz	3:12	2.2	72.55	67.55	56.58
Metal	6:28	4.4	75.60	70.60	59.55
Pop	4:00	2.75	73.58	68.53	57.60
R&B	3:51	2.64	73.36	68.34	57.39
Rap	3:59	2.74	73.57	68.56	57.46
Reggae	3:59	2.74	73.48	68.57	57.59
Rock	4:33	3.13	74.11	69.13	58.12

Figure 12 compares the results of PSNR for Data-1 with the application of the LSB technique for MP3 files embedding that involves the embedding of secret message in 1-LSB for various genres (Blues, Classical, Country, Dance, Hip-Hop, Jazz, Metal, Pop, R&B, Rap, Reggae, Rock) under different compression rates (96 kbps, 128 kbps, 192 kbps, 256 kbps, and 320 kbps). The 320-kbps compression rate gives the highest PSNR values. For all compression rates, the Metal genre achieves the highest score, because there is less noise in this channel and the file size is large, followed by Dance. Classical scores the lowest because of the high noise level in this channel and small file size; Jazz scores the second lowest.

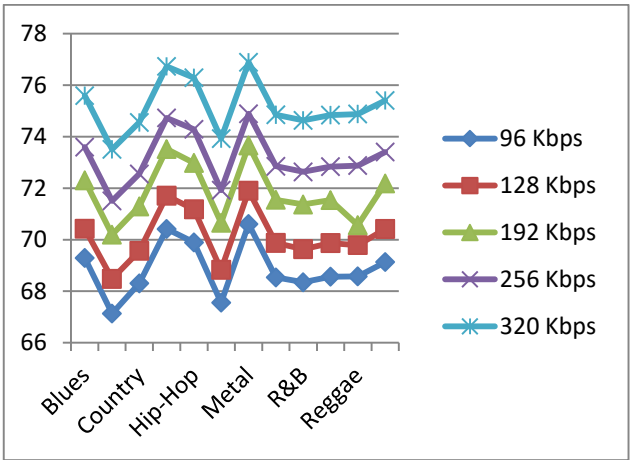

Figure 12. Compression results for PSNR with 1-LSB.

Figure 13 compares the results of PSNR for Data-1 with the utilization of the LSB technique for MP3 files embedding that involves the embedding of secret message in 2-LSB for a number of genres (Blues, Classical, Country, Dance, Hip-Hop, Jazz, Metal, Pop, R&B, Rap, Reggae, Rock) under different compression rates (96 kbps, 128 kbps, 192 kbps, 256 kbps and 320 kbps). The 320-kbps compression rate gives the highest PSNR values. For all compression rates, the Metal genre scores highest, because there is less noise in this channel and the file size is large, followed by Dance. Classical scores lowest because of the high noise level in this channel and small file size, followed by Jazz.

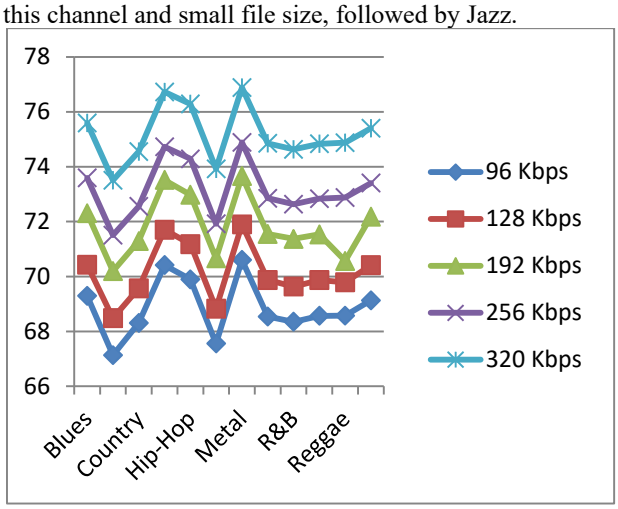

Figure 13. PSNR for 2-LSB with different genres under different compression rates.

Figure 14. compares the results of PSNR for Data-1 with the utilization of the LSB technique for MP3 files embedding that involves the embedding of secret message in 4-LSB for several genres (Blues, Classical, Country, Dance, Hip-Hop, Jazz, Metal, Pop, R&B, Rap, Reggae, Rock) under different compression rates (96 kbps, 128 kbps, 192 kbps, 256 kbps, and 320 kbps). The 320-kbps compression rate gives the highest PSNR values. For all compression rates, the Metal genre scores highest, because there is less noise in this channel and the file size is big, followed by Dance. Classical scores the lowest, because of the high noise level in this channel and small file size, followed by Jazz.

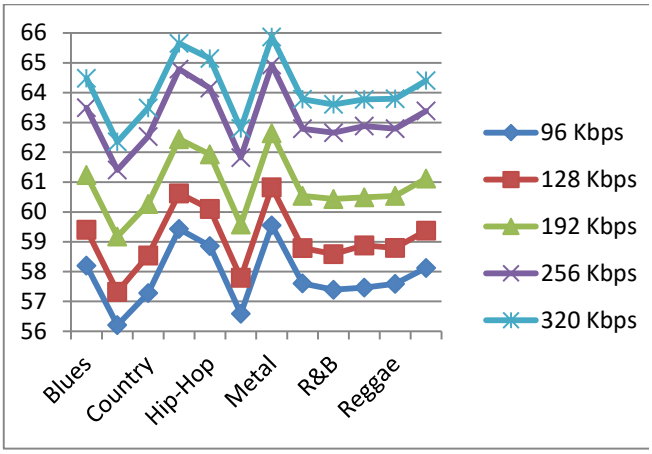

Figure 14. PSNR for 4-LSB with different genres under different compression rates.

As can be observed in Figure 15, for all genre types, 1-LSB shows better imperceptibility when compared to that shown by 2-LSB and 4-LSB. Meanwhile, imperceptibility in 4-LSB appears to be inferior when compared to that of 1-LSB and 2-LSB. Accordingly, the average PSNR values are correspondingly 64.2147, 61.4541, and 57.2901 for embedding in 1, 2, and 4 LSBs. Also, the table shows that the Jazz genre scored the highest value for 1-LSB (64.3768). For 2-LSB, the highest value was scored by the genre of Folk/country (61.8968), while the highest value for 4-LSB (57.8767) can be observed in the Blues genre. On the other hand, the lowest value for 1-LSB (64.1456) can be observed in the Alternative genre. For 2-LSB, the value was scored by the Blues genre was the lowest (61.1305), and for 4-LSB, the Rap/Hip-Hop genre scored the lowest value (57.1153).

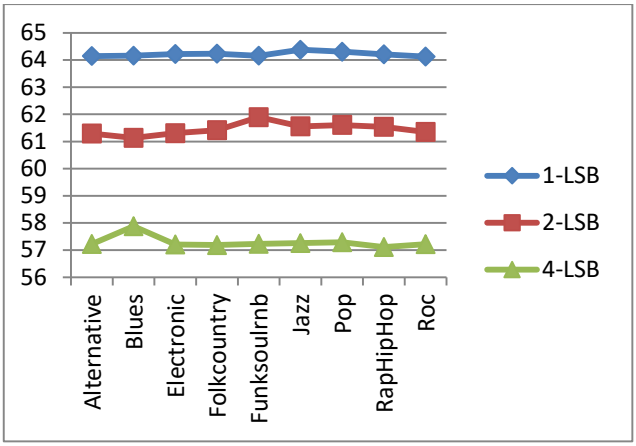

Figure 15. Different compression rates with different LSBs.

5. Conclusions

This paper highlighted the subject of MP3 audio steganography, with the focus on MP3 files post compression. In time domain, LSB has been formulated to use randomly position from cover file for the concealment of the secret message with the application of 1, 2 and 4 bits. A new model was proposed in this study; it fulfils the three most crucial requirements of audio steganography namely imperceptibility, capacity, and robustness. It is crucial that a technique with the purpose of improving the capacity or robustness would also maintain imperceptibility. The new proposed method was able to increase the capacity and robustness, while improving imperceptibility. The model established in this paper could effectively conceal data in Audio file while preserving the high accuracy of the audio. The secret message could still be unveiled but message extraction was challenging to execute.

Acknowledgments

This work was funded by the University of Jeddah, Jeddah, Saudi Arabia, under grant No. (UJ-22-DR-63). The authors, therefore, acknowledge with thanks the University of Jeddah technical and financial support.

References

[1] Fridrich, J. & Goljan, M.(2002). Practical steganalysis of digital images: State of the art. Electronic Imaging 2002, International Society for Optics and Photonics, 1-13.

[2] Kim, D.-S., Lee, G.-J. & Yoo, K.-Y.(2014). A reversible data hiding scheme based on histogram shifting using edge direction predictor. Proceedings of the 2014 Conference on Research in Adaptive and Convergent Systems, ACM, 126-131.

[3] Atoum, M. S.(2015) New MP3 Steganography Data Set. IT Convergence and Security (ICITCS), 2015 5th International Conference on, 2015b. IEEE, 1-7.

[4] Quackenbush, S. (2012). MPEG Audio Compression Advances. The MPEG Representation of Digital Media. Springer.

[5] Sterne, J. (2012). Mp3: The meaning of a format, Duke University Press.

[6] Sayood, K. (2012). Introduction to data compression, Newnes.

[7] Brandenburg, K.(1999). MP3 and AAC explained. Audio Engineering Society Conference: 17th International Conference: High-Quality Audio Coding, Audio Engineering Society.

[8] Sinder, D. J., Varga, I., Krishnan, V., Rajendran, V. & Villette, S. (2015). Recent speech coding technologies and standards. Speech and Audio Processing for Coding, Enhancement and Recognition. Springer.

[9] Supurovic, P. (1998). MPEG audio frame header. Available In Internet.
[10] Nilsson, M. (2000). ID3 tag version 2.4. 0-Main Structure. http//www. id3. org/id3v2.
[11] Salih, M. M. (2015). A New Audio Steganography Method Using Bi-LSB Embedding and Secret Message Integrity Validation. Middle East University.
[12] Jhaveri, N. V., Vaughan, G. B., Anderson, I. W., Gardner, J. J., & Tao, P. T. (2019). U.S. Patent Application No. 16/357,128.
[13] Di Angelo, M., & Salzer, G. (2019, July). Mayflies, breeders, and busy bees in Ethereum: smart contracts over time. In Proceedings of the Third ACM Workshop on Blockchains, Cryptocurrencies and Contracts (pp. 1-10).
[14] Castelan, Y. & Khodja, B. (2015) MP3 Steganography Techniques. Proceedings of the 4th Annual ACM Conference on Research in Information Technology, ACM, 51-54.
[15] Zebari, D. A., Zeebaree, D. Q., Saeed, J. N., Zebari, N. A., & Adel, A. Z. (2020). Image Steganography Based on Swarm Intelligence Algorithms: A Survey. people, 7(8), 9.
[16] Fridrich, J. (2009). Steganography in digital media: principles, algorithms, and applications. Cambridge University Press.
[17] Chhikara, S. & Singh, P. (2013b.) SBHCS: Spike based Histogram Comparison Steganalysis Technique. International Journal of Computer Applications, 75.
[18] Kekre, H. B., Athawale, A., Rao, B. S. & Athawale, U. (2010). Increasing the capacity of the cover audio signal by using multiple LSBs for information hiding. Emerging Trends in Engineering and Technology (ICETET), 2010 3rd International Conference on, 2010. IEEE, 196-201.
[19] Ozighor, E. R., & Izegbu, I. (2020). INFORMATION PROTECTION AGAINST SECURITY THREATS IN AN INSECURE ENVIRONMENT USING CRYPTOGRAPHY AND STEGANOGRAPHY. GSJ, 8(5).
[20] Devaraj, S., Singh, U. & Jialal, I. (2009). The evolving role of C-reactive protein in atherothrombosis. Clinical Chemistry, 55, 229-238.
[21] Shirali-Shahreza, S., Manzuri-Shalmani, M. & Shirali-Shahreza, M. H. (2007). A Skew resistant method for persian text segmentation. Computational Intelligence in Image and Signal Processing, 2007. CIISP 2007. IEEE Symposium on, 2007. IEEE, 115-120.
[22] Gopalan, K. & Shi, Q. (2010). Audio Steganography Using Bit Modification-A Tradeoff on Perceptibility and Data Robustness for Large Payload Audio Embedding. ICCCN, 2010.
[23] Fan, X., Cao, J.: *A Survey of Mobile Cloud Computing*. ZTE Communications 9(1), 4–8 (2011)
[24] Huerta-Canepa, G., Lee, D.: A virtual cloud computing provider for mobile devices. In: Proc. of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond (2010)
[25] Giurgiu, I., Riva, O., Juric, D., Krivulev, I., Alonso, G.: *Calling the Cloud: Enabling Mobile Phones as Interfaces to Cloud Applications*. In: Bacon, J.M., Cooper, B.F. (eds.) Middleware 2009. LNCS, vol. 5896, pp. 83–102. Springer, Heidelberg (2009)

Alaa Abdulslam Alarood received the B.E. and M.E. degrees, from Yarmouk Univ. in 2001 and 2005, respectively. He received the Ph.D. degree from University of Technology Malaysia (UTM). in 2017. After working as, a research assistant and lecturer (from 2006), an assistant professor (from 2017) in the Dept. of Computer Science in College of Computer Science and Engineering, University of Jeddah, Saudi Arabia (from 2018), His research interest includes Security, Multimedia Security, Machine Learning, Internet of Things (IoT).

AHMED MOHAMMED ALGHAMDI is an assistant professor at the Software Engineering Department, College of Computer Science and Engineering, University of Jeddah, Saudi Arabia. He got his Ph.D. in Computer Science from King Abdulaziz University, Jeddah, Saudi Arabia. He received his B.Sc. degree in Computer Science from King Abdulaziz University, Jeddah, Saudi Arabia, in 2005 and the first M.Sc. degree in Business Administration from King Abdulaziz University, Jeddah, Saudi Arabia, in 2010. He received the second master's degree in Internet Computing and Network Security from Loughborough University, UK, in 2013. Dr. Ahmed also has over 11 years of working experience before attending the academic carrier. His research interests include high-performance computing, big data, distributed systems, programming models, software engineering, BYOD, and software testing.

Ahmed Omar Alzahrani is currently an Assistant Professor at Faculty of Computer Science and Engineering, University of Jeddah. He holds a Ph.D. degree in Information Systems and Technology from Claremont Graduate University. He received a Master's of Computer Science from California Lutheran University and a Master's degree in Information Systems and Technology from Claremont Graduate University. His research interest is centered primarily around Geographic Information Systems (GIS), Energy Informatics, Remote Senescing, and Machine Learning,

Abdulrahman Alzahrani is an assistance professor in the College of Computer Science and Engineering at university of Jeddah, Kingdom of Saudi Arabia. Alzahrani's research is focused on the field of information technology and innovations in the area of Data Science, Machine Learning, and Health Informatics. He published his dissertation titled "Exposome Factors: Exploratory Study Approach and the Role of Persuasive Technology to Raise Awareness About the Exposome Concept" in 2020. Abdulrahman teaches several courses to bachelor and master students. He teaches E Commerce, Data Visualization classes. He is currently holding the head of academic and exam affair unit within the college.

Eesa Alsolami is Associate professor of computer science and engineering at university of Jeddah Currently, he is a dean of Admission and registration of university of Jeddah. He gets his PhD in 2012 from Queensland University of technology from Australia. His research projects involve feature selection techniques for continuous biometric authentication. Eesa graduated in computer science in 2002 from King Abdulaziz University, and then in 2008 he received MSc in IT from Queensland university of technology.