AUDIO STEGANOGRAPHY METHOD USING LEAST SIGNIFICANT BIT (LSB) ENCODING TECHNIQUE

EESA ABDULLAH ALSOLAMI1

[1] College of Computer Science and Engineering, Department of Cypher security, university of Jeddah,

Jeddah 21493, Saudi Arabia.

E-mail: [1] eaalsulami@uj.edu.sa

ABSTRACT

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. The aims to add effectiveness and performance for hide message in MP3 file and explains all detail in steganography process in MP3 files, and The new proposed method was able to increase the capacity and robustness, while improving imperceptibility.

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

1. INTRODUCTION

This guide provides details to assist authors in preparing a paper for publication in JATIT so that there is a consistency among papers. These instructions give guidance on layout, style, illustrations and references and serve as a model for authors to emulate. Please follow these specifications closely as papers which do not meet the standards laid down, will not be published.

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

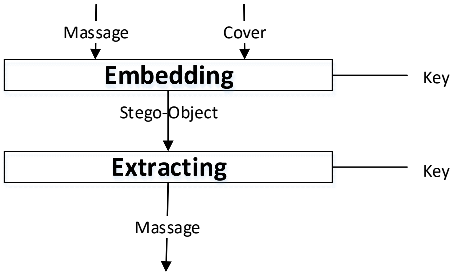

Figure 1 Basic Model Of Steganography.

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

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.

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].

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

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].

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

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].

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.

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 × M → C. Meanwhile, the extraction process which includes secret message extraction from a cover message is illustratable by the mapping D: C → M. Here, C denotes the set of probable covers, while M denotes the set of probable messages namely $|C| \geq |M|$.

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.

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. This becomes an advantage of secret key steganography. Figure 4 accordingly illustrates the process of secret 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].

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 2 presents the process.

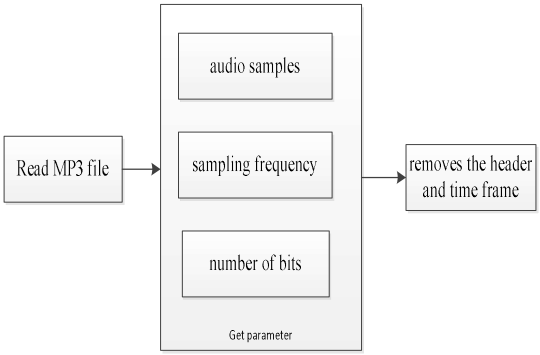

Figure 2 Obtaining The Parameters Of An MP3 File.

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:

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

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

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 3. The following steps describe the overall process:

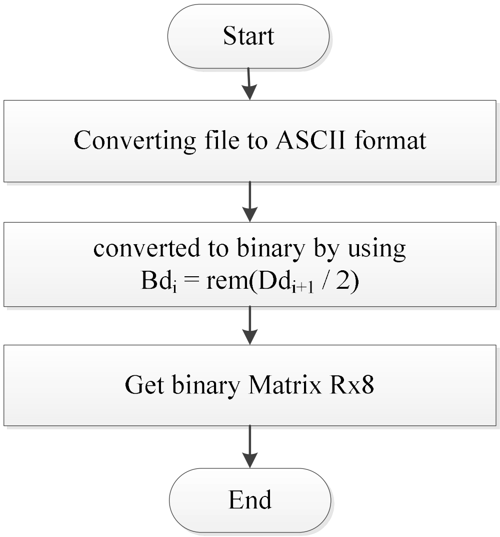
Start
Converting file to ASCII format
converted to binary by using
Bd$_i$ = rem(Dd$_{i+1}$ / 2)
Get binary Matrix Rx8
End

Figure 3 Converting A Text File To Ascii.

3.3 Normalization of Files

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 4.

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:

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

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:

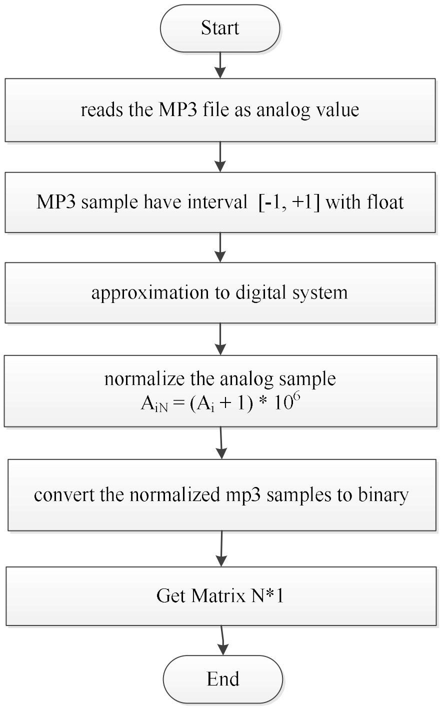
Start
reads the MP3 file as analog value
MP3 sample have interval [-1, +1] with float
approximation to digital system
normalize the analog sample
$A_{iN} = (A_i + 1) * 10^6$
convert the normalized mp3 samples to binary
Get Matrix N*1
End

Figure 4. Converting An Mp3 File To A Bit Stream.

3.4 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 1 summarizes the four signatures to be embedded at the start and end of Each And Every Message [23].

Table 1 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 5 shows the addition of keys or a signature for single-bit insertion and four-bit insertion of a secret message.

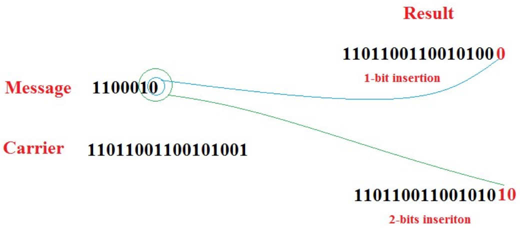

Figure 5. Insertion Of One And Two Bits.

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 5). The following steps illustrate the process:

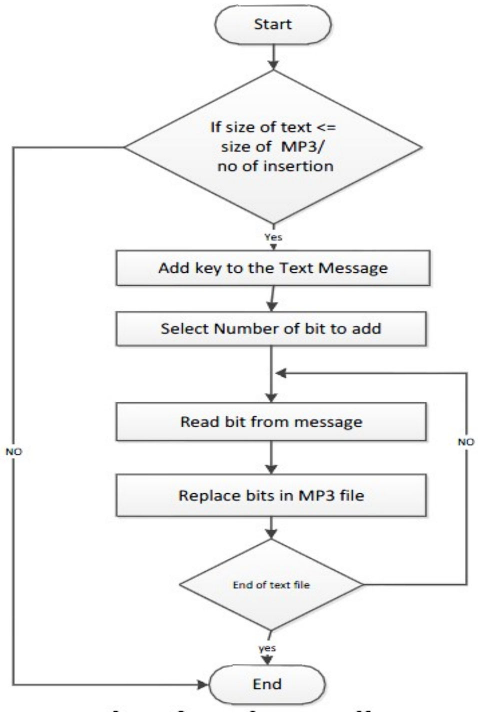
Start
If size of text <= size of MP3/ no of insertion
Yes
Add key to the Text Message
Select Number of bit to add
Read bit from message
Replace bits in MP3 file
End of text file
yes
NO
NO
End

Figure 6. Replacing Bits In The Carrier File.

3.5 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:

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

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:

$$\mathbf{A_i = (A_{iN} \times 10^{-6}) - 1} \qquad \mathbf{(5)}$$

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:

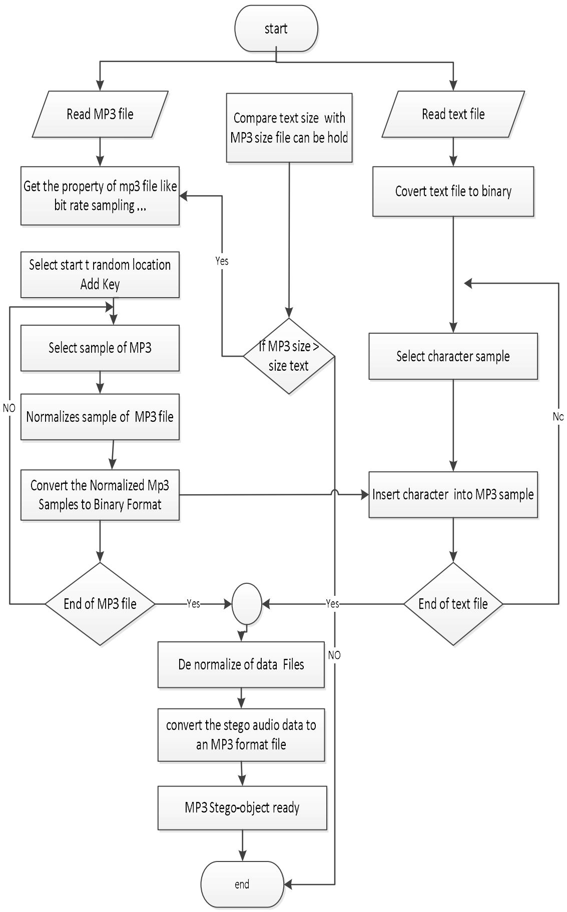
start
Read MP3 file
Compare text size with MP3 size file can be hold
Read text file
Get the property of mp3 file like bit rate sampling ...
Covert text file to binary
Yes
Select start t random location Add Key
If MP3 size > size text
Select sample of MP3
Select character sample
NO
Nc
Normalizes sample of MP3 file
Convert the Normalized Mp3 Samples to Binary Format
Insert character into MP3 sample
End of MP3 file
Yes
Yes
End of text file
De normalize of data Files
NO
convert the stego audio data to an MP3 format file
MP3 Stego-object ready
end

Figure 7 Embedding A Text File In An MP3 File.

3. *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(1,2,3) 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 3.1 PSNR Results For Data-1 At 320 Kbps Compression Rate

Genre	Time (min)	Size (MB)	1-LSB 1st position
Blues	4:41	10.7	79.5696
Classical	2:54	6.67	77.5252
Country	3:42	8.48	78.4335
Dance	6:12	14.2	80.7125
Hip-hop	5:27	12.4	80.1509
Jazz	3:12	7.34	77.9665
Metal	6:28	14.8	80.8612
Pop	4:00	9.16	78.8722
R&B	3:51	8.81	78.6551
Rap	3:59	9.14	78.8766
Reggae	3:59	9.14	78.7926
Rock	4:33	10.4	79.3576

Table 3.2 PSNR Results For Data-1 At 320 Kbps Compression Rate

Genre	Time (min)	Size (MB)	2-LSB 1st and 2nd
Blues	4:41	10.7	75.5938
Classical	2:54	6.67	73.5076
Country	3:42	8.48	74.5546
Dance	6:12	14.2	76.728
Hip-hop	5:27	12.4	76.2863
Jazz	3:12	7.34	73.923
Metal	6:28	14.8	76.8849
Pop	4:00	9.16	74.8513
R&B	3:51	8.81	74.6336
Rap	3:59	9.14	74.8357
Reggae	3:59	9.14	74.878
Rock	4:33	10.4	75.4048

Table 3.3 PSNR Results For Data-1 At 320 Kbps Compression Rate

Genre	Time	Size	4-LSB
	(min)	(MB)	1st , 2nd and 4th
Blues	4:41	10.7	64.4854
Classical	2:54	6.67	62.3538
Country	3:42	8.48	63.4877
Dance	6:12	14.2	65.6537
Hip-hop	5:27	12.4	65.1445
Jazz	3:12	7.34	62.7977
Metal	6:28	14.8	65.8695
Pop	4:00	9.16	63.7806
R&B	3:51	8.81	63.6058
Rap	3:59	9.14	63.7763
Reggae	3:59	9.14	63.7976
Rock	4:33	10.4	64.4034

4. 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.

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 AudioProcessing 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] Alarood, A. A. S. (2017). *Improved Steganalysis Technique Based on Least Significant BIT Using Artificial Neural Network for Mp3 Files* (Doctoral dissertation, Universiti Teknologi Malaysia).