*IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)*
*e-ISSN: 2278-2834,p- ISSN: 2278-8735.Volume 12, Issue 1, Ver. III (Jan.-Feb. 2017), PP 43-51*
*www.iosrjournals.org*

# Drubbing an Audio Messages inside a Digital Image Using (ELSB) Method

P. Vanchi
*Assistant Professor E.C.E Department A.R Engineering College*

***Abstract:*** *It is mainly focused today to transfer the messages secretly between two communication parties. The message from the sender to receiver should be kept secret so that the information should not known by anyone. Secret is the important thing today. The technique that is used for secure communication is called as steganography and it means that to hide secret information into innocent data. Digital images are ideal for hiding secret information. An image containing a secret message is called a cover image. In this paper will discuss about secret transformation of audio messages. The audio messages are hidden inside a cover image so no one can hack the audio but the audio should be encrypted before hidden inside the image.*
***Keywords:*** *Steganography/ Cryptography/ Audio message/ image hiding/ least-significant bit (LSB) method*

## I. Introduction

A major issue for computer networks is to prevent important information from being disclosed to illegal users. For this reason, encryption techniques were introduced. Most encryption techniques have an easy implementation and are widely used in the field of information security.

A main problem in this computer network is to prevent information from being disclosed to illegal users. This is the main reason, encryption techniques were introduced Cryptography and Steganography [1-4] are well known and widely used techniques that manipulate information (messages) in order to cipher or hide their existence. These techniques have many applications in computer science and other related fields: they are used to protect military messages, E-mails, credit card information, corporate data, personal files, etc.

Cryptography is, traditionally, the study of means of converting information from its normal, comprehensible form into an incomprehensible format, rendering it unreadable without secret knowledge, the art of encryption. Steganography is the art and science of writing hidden messages in such a way that no one, apart from the sender and intended recipient, suspects the existence of the message, a form of security through obscurity. Different applications have different requirements of the Steganography technique used. For example, some applications may require absolute invisibility of the secret information, while others require a larger secret message to be hidden.

The main goal of steganography is to communicate securely in a completely undetectable manner [2] and to avoid drawing suspicion to the transmission of a hidden data [5]. It is not to keep others from knowing the hidden information, but it is to keep others from thinking that the information even exists. If a Steganography method causes someone to suspect the carrier medium, then the method has failed. On other hand, steganalysis is a way of detecting possible secret communication using against steganography. That is, steganalysis attempts to defeat steganography techniques. It relies on the fact that hiding information in digital media alters the carriers and introduces unusual signatures or some form of degradation that could be exploited. Thus, it is crucial that a steganography system to ascertain that the hidden messages are not detectable [6]. Before reviewing some of the basic Steganography techniques, let us mention some terminologies used in this field [7]:

***Payload:*** The information which is to be concealed.

***Carrier File:*** The media where payload has to be hidden.

***Stego-Medium:*** The medium in which the information is hidden.

***Redundant Bits:*** Pieces of information inside a file which can be overwritten or altered without damaging the file.

***Steganalysis:*** The process of detecting hidden information inside of a file.

In this context, the following formula provides a very generic description of the pieces of the steganographic process.

*Cover image + hidden-data + stego_key = stego-image*

The four basic techniques used for Steganography are:

***LSB method:*** The LSB of carrier medium is directly inserted with the message bit. So LSB of the carrier medium contains the payload.

***Injection:*** Hiding data in sections of a file that are ignored by the processing applications. Therefore avoid modifying those file bits that are relevant to an end end-user leaving the cover file perfectly usable.

***Substitution:*** Replacement of the least significant bits of information that determine the meaningful content of the original file with new data in a way that causes the least amount o distortion.

***Generation:*** Unlike injection and substitution, this does not require an existing cover file but generates a cover file for the sole purpose of hiding the message.

Basically, the model for steganography is as shown in Fig.1. The *cover-object* is a carrier or medium to embed a message. There are several suitable medium that can be used as *cover-objects* such as network protocols, audio, file and disk, a text file and an image file. *Message* is the data that the sender wishes to keep confidential and will be embedded into the *cover-object* by using a *stegosystem encoder*. It can be a plain text, a cipher text, an image, or anything that can be embedded in a bit stream such as a copyright mark or a serial number. A *stego-key* is a password, which ensures that only the recipient who knows the corresponding decoding key will be able to extract the message from a *cover-image.* The output of the stegosystem encoder is known as the *stego-object.*

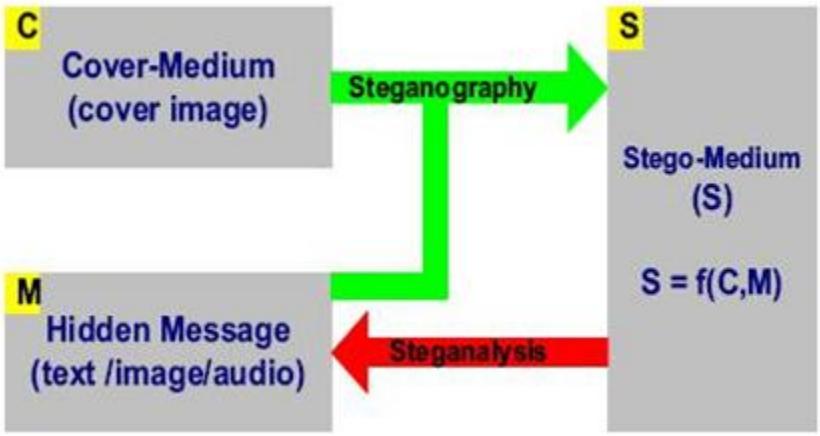



**Fig.1.** Steganography and Steganalysis Model

Steganalysis [8-10] tools can also easily detect this method; increased success can be achieved by removing some of the randomness introduced by the bit changes, e.g. a change of every LSB by one would probably not be detected as there is no random element present. The bits would be assumed to form part of the original image. An increasingly complex method of image Steganography is known as the patchwork algorithm. This algorithm randomly selects pairs of pixels is made brighter, and the darker one darker. This change is so subtle that it is undetectable to the human eye; even at high zoom levels the changes simply are not sufficient to make the image appear altered. The contrast change between these two pixels now forms part of the bit pattern for the hidden file. In order to go undetected by a filtering attack (see section 4) a limit to a few hundred changes can take place.

## II. Proposed Approach

The present work studies the possibility of hiding short audio messages within digital images. There are a number of different types of Audio files. The most common are Wave files (wav) and MPEG Layer-3 files (mp3). There are, however, many other audio file types. The type is usually determined by the file extension (what comes after the "." in the file name). For example, ".wav", ".mp3" or ".dct". Different audio file format concerning the size of each file format. It has been found that the audio WAV files are probably the simplest of the common formats for storing audio samples. Unlike MPEG and other compressed formats, WAVs store samples "in the raw" where no pre-processing is required other that formatting of the data. The description of

WAV file format is listed in Table-1. It is clear that the first 44 bytes describes the header of any wav file. The length of audio data is stored in bytes 40-43 and audio samples itself occupies the remainder of the file starting from byte 44.

Regardless of the selected audio file format, the steganography technique will simply embed the audio message into the *cover-image* without supplying any password or *stego-key.* At this stage, I decided to do so just to understand the ways of LSB insert the message bit into the image and extract the message from the *stego-image* produced. The advantages of LSB are its simplicity to embed the bits of the message directly into the LSB plane of *cover-image* and many techniques use these methods [12]. Modulating the LSB does not result in a human-perceptible difference because the amplitude of the change is trivial. Therefore, to the human eye, the resulting *stego-image* will look identical to the *cover-image.* This allows high perceptual transparency of LSB. However, there are few weaknesses of using LSB. It is very sensitive to any kind of filtering or manipulation of the *stego-image.* Scaling, rotation, cropping, addition of noise, or lossy compression to the *stego-image* will destroy the message.

**Table – 1:** Wav file format description

| Byte Number | Description |
|---|---|
| 0 – 3 | "RIFF" (ASCI I Characters) |
| 4 – 7 | Total Length Of Package To Follow |
| 8 – 11 | "WAVE" (ASCII Characters) |
| 12 – 15 | "fmt_" (ASCII Characters) |
| 16 – 19 | Length Of FORMAT Chunk (Binary, always 0x10) |
| 20 – 21 | Always 0x01 |
| 22 – 23 | Channel Numbers (Always 0x01=Mono, 0x02=Stereo) |
| 24 – 27 | Sample Rate (Binary, in Hz) |
| 28 – 31 | Bytes Per Second |
| 32 – 33 | Bytes Per Sample: 1=8 bit Mono, 2=8 bit Stereo or 16bit Mono, 4=16 bit Stereo |
| 34 – 35 | Bits Per Sample |
| 36 – 39 | "data" (ASCII Characters) |
| 40 – 43 | Length Of Data To Follow |
| 44 - end | Data (Samples) |

Digital images typically use either 8-bit or 24-bit color. When using 8-bit color, there is a definition of up to 256 colors forming a palette for this image, each color denoted by an 8-bit value. A 24-bit color scheme, as the term suggests, uses 24 bits per pixel and provides a much better set of colors. In this case, each pixel is represented by three bytes, each byte representing the intensity of the three primary colors red, green, and blue (RGB), respectively. On the other hand, for the hiding capacity, the size of information to be hidden relatively depends to the size of the *cover-image*. The message size must be smaller than the image. A large capacity allows the use of the smaller *cover-image* for the message of fixed size, and thus decreases the bandwidth required to transmit the *stego-image.*

**2.1 Audio File Embedding**

Audio message embedding is performed by: the contents of the header of WAV audio file are retrieved as separate fields; RIFF, total length of package, WAVE, fmt, length of format chunk, and length of data to follow (bytes 40-43). Each field is converted to a Bit Array and then embedded bit by bit into the least significant bits of the blue channel of the cover image. The selection of the blue channel for embracing the header fields is not mandatory and both the red and green channels can be used as well. The sound samples are then retrieved from the audio file as stream. This stream is converted to a Bit Array and embedded bit by bit into the least significant bits of blue and the remainder of the blue channel of the cover image. The location of embedding each header field of the audio file within the blue channel is considered as a secret key. Moreover, it is not necessary to store the audio samples sequentially in the mentioned channels. These samples can be stored in reverse order and alternatively between the odd and even pixels of the cover image.

The relation between size of the audio file in bytes (*A*) and that of the cover image in pixels (*P*) can be determined as:

8 x A = 3 x W x H,

Where *A* is the size of the audio file (bytes), *H* and *W* are the height and width of the cover image respectively,

8 x *A* = size of the audio file in bits,

3 *x W x H* = number of least significant bits in the three channels of the cover image.

Accordingly the maximum size of the audio file is:

= ,

For the WAV audio files, the header occupies the first 44 bytes and the remaining bytes are dedicated for the sound samples, then:

= + ,

Where M is the size of the actual message in bytes,

= ,

Assuming the time of this message = $t_m$ (seconds), then:

= = – ,

Where BPS is the number of bytes of the recorded message per second,

=

This means that the time of the recorded message (in seconds) must be less than

The algorithm of embedding the audio file within the image is shown in Fig.2.

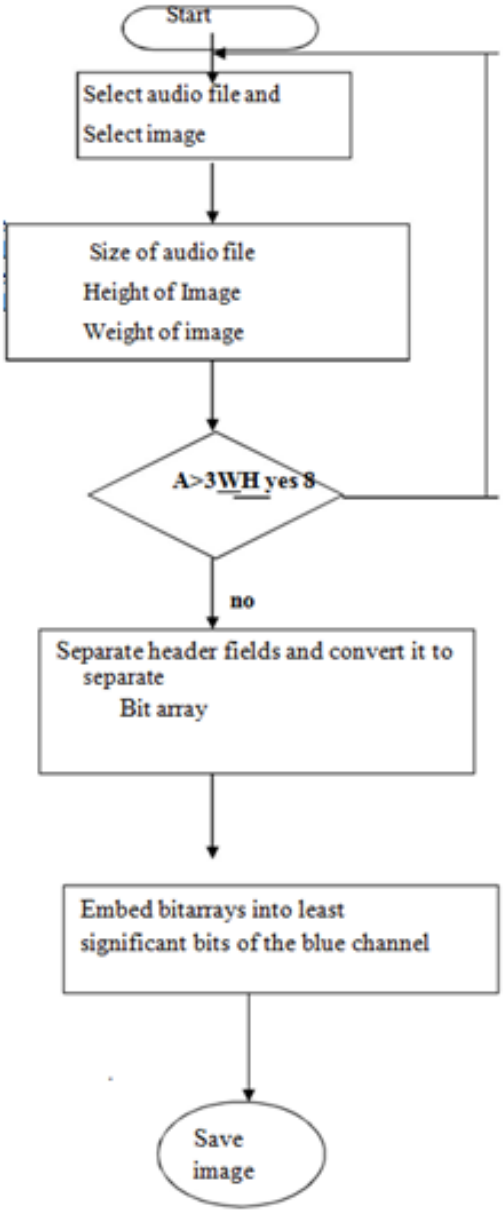



**Fig.2:** The flowchart of audio encryption and embedding algorithm

**2.2 Audio File Extracting**

The retrieving of the audio file from the image is straight forward. The header fields are extracted from the secret positions of the green channel of the image to be stored in corresponding BitArrays. Each BitArray is converted to the suitable data type according to Table.1 and appended to the new audio file. The sound samples are retrieved from the suitable bits of image channels and stored in a BitArray. The number of these samples is determined according to "Length of Data To Follow" field previously retrieved in the first BitArray. The contents of this BitArray are converted to bytes or integer numbers and appended to the audio file. The process of extracting the audio file does not need the original image; it needs only to know the secret position of each header field inside the image. The flowchart of the extraction process is shown in Fig.3.

**2.3 Changing LSB Bits Of The Cover Image Using Encrypted Audio Message**

The audio data can be encrypted before embedding it inside the LSB (bit number 7) of the image channels. Assume *m* represents a bit of the audio message BitArray and $b_5$ and $b_6$ represent the fifth and sixth bits of a certain color of the present pixel respectively. Then the value to be generated for the hidden bit (*h*) will be computed as:

$h = (b_5 \quad b_6) \quad m$ (1)

**Table-2:** Truth table of $b_5$, $b_6$, $m$ and $h$

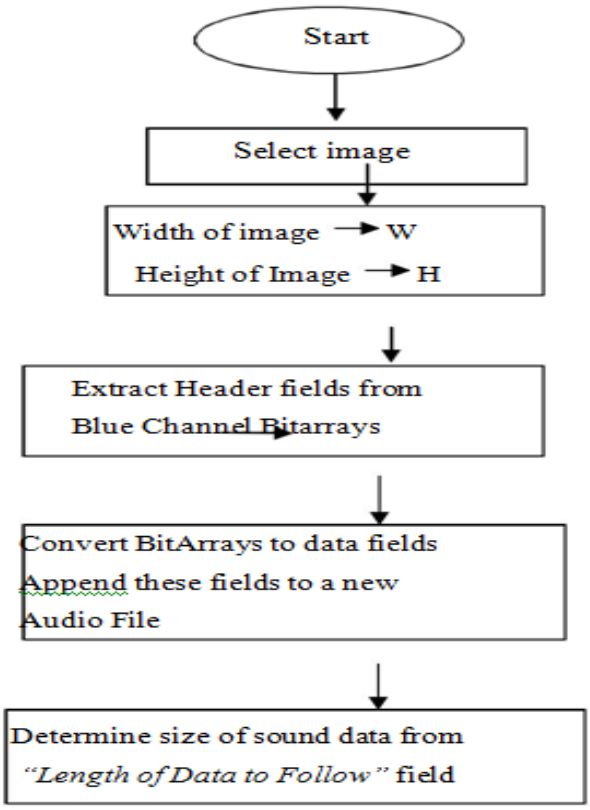
| ***b5*** | ***b6*** | ***m*** | ***($b_5$ $b_6$)*** | ***h*** |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 |

Then bit $h$ can be used to replace $b_7$ of the current pixel in the cover image to produce the stego-image. On the other side, the stego-image is used to extract the original audio message as explained in the following section.

### 2.4 Extracting And Decrypting of the Original Audio Message From The Stego-Image

Now, a pixel has been retrieved from the image and $h$ represents the hidden bit. Then $b_5$ and $b_6$ of this pixel will be used in addition to $h$ to generate the original message's bit $m$. The following truth table shows that:

$$m = (b_5 \quad b_6) \quad h \tag{2}$$



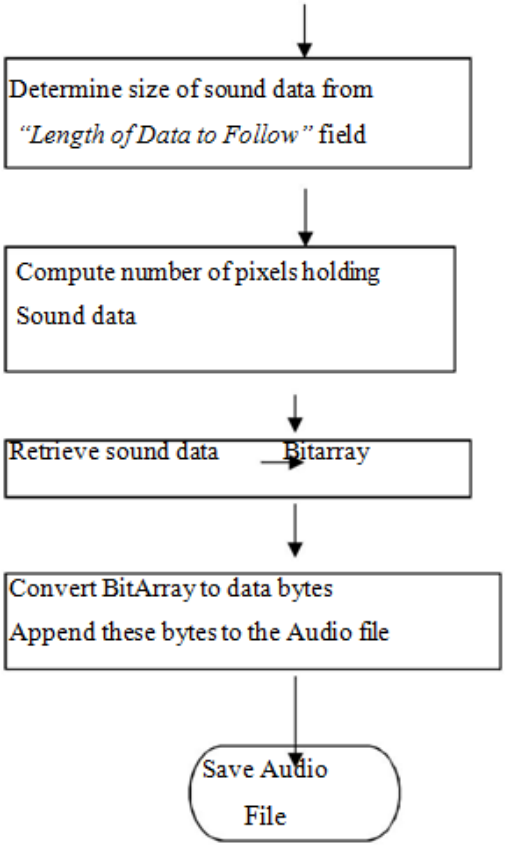



**Fig.3:** The flowchart of audio extraction and decryption algorithm

**Table-3:** Truth table of $b_5$, $b_6$, $h$ and $m$

| ***b5*** | ***b6*** | ***h*** | ***($b_5$ $b_6$)*** | ***m*** |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 |

Comparing Table-2 and Table-3 proves that Eq.2 is valid and can be used to retrieve the original bit *m.* Figure 4 and 5 explain bits encryption and decryption methods.

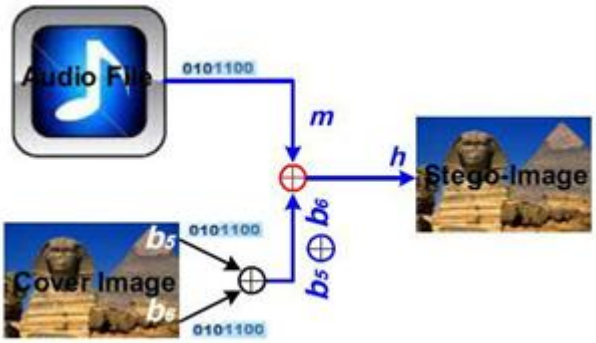



**Fig.4** Bit encryption block diagram

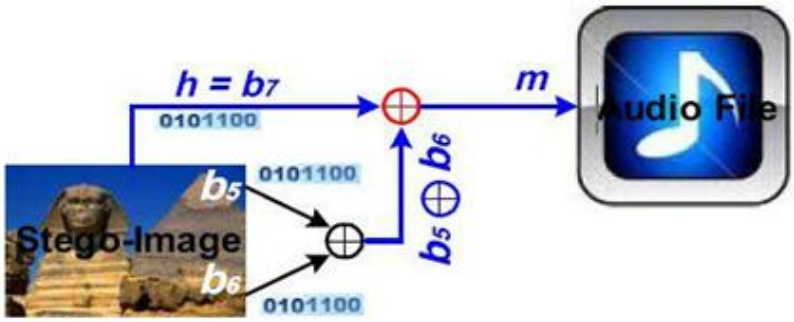
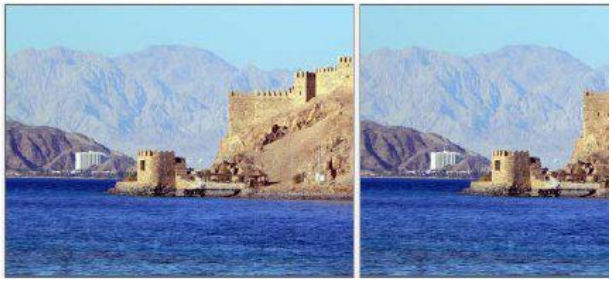



**Fig.5** Bit decryption block diagram

## III. Implementation & Results

The above proposed work has been implemented using Microsoft Visual Studio, Matlab. A group of test cover images are shown in Fig.6 before and after hiding short audio files using the previously mentioned method. The least significant bits of the image pixels were encrypted using the bit streams obtained from the audio files. Moreover, the short audio messages have been extracted from the stego-images using the introduced decryption technique and saved to new wav files. It has been verified that modulating the cover image with the short audio message does not result in a human-perceptible difference because the amplitude of the change is trivial. Therefore, to the human eye, the resulting *stego-image* will look identical to the *cover-image.* The extracted audio messages are compared to the original audio files and were identical with them.

4. A 69 KB wav file has been hidden inside the 800x533 left images yielding to the right one.



5. A 74 KB wav file has been hidden inside the 800x600 left images yielding to the right one.

6. A 48 KB wav file has been hidden inside the 432x346 left images yielding to the right one.

7. A 138 KB wav file has been hidden inside the 1600x1200 left images yielding to the right one.

**Fig.6** Cover (left) and stego (right) images

## IV. Conclusion

This paper discussed the possibility of hiding short audio messages inside digital images. The embedding process creates a *stego medium* by replacing these redundant bits with data from the hidden audio message. The proposed method provided a higher similarity between the cover and the obtained stego pictures. The new approach provides a secured means of secret communication between two parties. The future work could be to extend to embed audio messages within video files. Moreover, this method can be more developed to embed a secret audio channel within any broadcasting medium.

## References

[1]. William Stallings, Cryptography and Network Security, Principles and Practice, Third edition, Pearson Education, Singapore, 2003.
[2]. R.J. Anderson and F. A. P. Petitcolas (2001) On the limits of the Stegnography, IEEE Journal Selected Areas in Communications, 16(4), pp.474-481.
[3]. Johnson, Neil F., and SushilJajodia. "Exploring Steganography: Seeing the Unseen". IEEE Computer Feb. 1998: 26-34.
[4]. A. Ker, "Improved detection of LSB steganography in grayscale images," *in Proc. Information Hiding Workshop, vol. 3200,* Springer LNCS, pp. 97-115, 2004.
[5]. C. C. Lin, and W. H. Tsai, "Secret Image Sharing with Steganography and Authentication," Journal of Systems and Software, 73(3): 405-414, December 2004.
[6]. KafaRabah. Steganography – The Art of Hiding Data. Information technology Journal 3– 2004.
[7]. A. Westfeld, "F5-A Steganographic Algorithm: High Capacity Despite Better Steganalysis", LNCS, Vol. 2137, pp.289-302, April 2001.
[8]. Krenn, R., "Steganography and Steganalysis", http://www.Krenn.nl/univ/cry/steg/article.pdf