ISSN: 2278 - 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
Vol. 1, Issue 6, December 2012

Efficient Method to Increase Robustness in Audio Steganography

Deepak D[1], Karthik M L[2], Manjunath A E[3]

Student, Department of Computer Science and Engineering, R.V College of Engineering, Bangalore-560059, India [1,2]

Assistant Professor, Department of Computer Science and Engineering, R.V College of Engineering, Bangalore-560059, India [3]

***Abstract*- Extensive use of internet applications has created a need for the secure transmission of data. To ensure the privacy of the communication between two parties, various new methods are being developed, Cryptography being the mother to all those projects. Steganography is a technique of hiding information in digital media that only intended recipient is aware of. In this paper we propose a modification to the existing LSB algorithm used in audio steganography that increases its robustness.**

Keywords: Audio Steganography, LSB method, human auditory system

I. INTRODUCTION

One of the reasons that the intruders are successful is the information are in the human readable form or as a cipher text that the intruders try to break. Intruders may leak the information to others, manipulate it to misinterpret as well as to misrepresent an individual or organization. Steganography is one of the solutions to overcome this problem by making intruder believe that there is no useful information. Steganography is a technique of hiding information in digital media. The digital media can be image, audio or a video. There are numerous methods to hide information in digital media, one of them being LSB(Least Significant Bit). If a steganography method causes someone to suspect there is a secret information in a carrier medium, then the method has failed[5]. The success of any method depends on how robust it is and how much data it can hide without giving up its original characteristics significantly.

The rest of the paper is organized as follows. We discuss existing system in section II and our proposed system in section III. Section IV reports the results of our experimental study. Finally section V gives the concluding remarks and section VI presents future enhancements followed by acknowledgement and references.

II. EXISTING SYSTEM

LSB algorithm is one of the most popular methods in audio steganography. By modifying the least significant of several bytes of an audio file, only minor changes occur in the original sound, most of which cannot be distinguished by the human auditory system. We make use wav files to hide the message since it can be edited and manipulated with ease relatively.

Wav files make use of either 8 or 16 bits to store sound information. 8 bit files allow values of sound in the range between 0 and 255 and the 16 bit files will have values from 0 to 65535. By changing the values of bytes slightly, we can store our secret data.

If for example, we have 8 byte sample of wav audio:

200 234 157 141 128 178 62 39

These values would be represented in binary as:

11001000 11101010 10011101 10001101

10000000 10110010 0111110 00100111

Suppose we want to hide the binary file 11101010 (234) inside this sequence. We replace the least significant bit in each byte of wav sample(the least significant bit because it will cause the least amount of change in the value) by bits of the binary form that makes up 234. The sequence of binary after modifying wav by stuffing 234 is shown below:

11001001 11101011 10011101 10001100

10000001 10110010 0111111 00100110

The new binary values deviates from the values of original audio only a little.. These discrepancies are negligible since human auditory system cannot differentiate between the two at such small levels.

ISSN: 2278 - 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
Vol. 1, Issue 6, December 2012

This system has replaced the LSB of every byte whatever its value may be. There won't be much variation from original audio file if the byte value whose LSB being replaced is large. But if the value is very less say 1 or 2, the change in LSB causes change in the value of the byte by around 100% . This large deviation may make one think of LSB being used and one may try to crack it. This could be overcome by modifying least significant bit only when its value is large or by using some pattern to stuff bits in various positions of the byte in all channels of WAV file which is our proposed system.

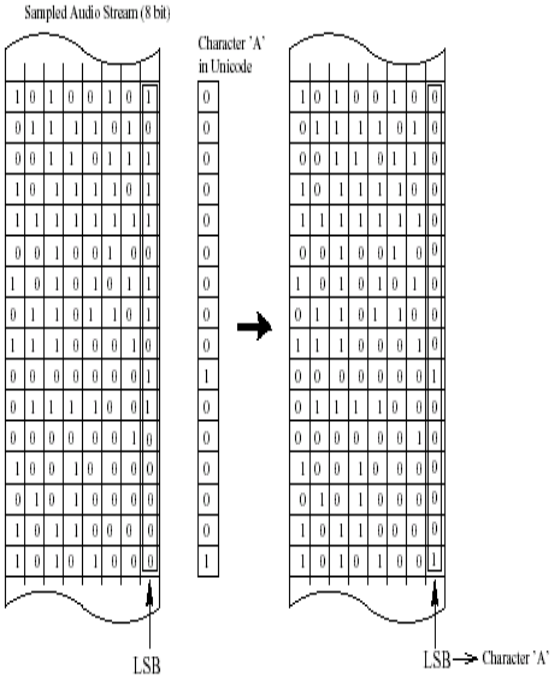

Fig 1: LSB encoding

III. PROPOSED SYSTEM

A. *Encoding*

Wav file consists of number of channels. In the modified LSB algorithm proposed here, instead of stuffing bit of the message only in the least significant bit in the consecutive bytes of wav file, a pattern is used to stuff bits. The same pattern can be made use to decode the file to get back the hidden message. Since it is quite easy to encode and decode if we make use of the same pattern to stuff message bits in different positions of byte in all channels, we stuff the bits in same pattern in all the channels. For example if we use the pattern 3142, then the 1st bit of message is stored at 3rd bit position , 2nd bit of message is stuffed in 1st position, 3rd bit in 4th position, 4th bit in 2nd position, 5th bit in 3rd position and so on.

In this scheme we stuff the entire byte in 1 channel, next byte in next channel and so on using the same pattern. Instead of stuffing bits in consecutive sequential bytes as in conventional LSB, we stuff entire byte in one channel, next byte in next channel and so on. This gives the shield against possible attack by trying to read the wav file sequentially.

For example if we want to stuff the message "abcdef" in a wav file consisting of 3 channels using the pattern 3142, character 'a' will be stuffed in 1st channel, 'b' in 2nd channel, 'c' in 3rd channel, 'd' in 1st channel and so on. While storing a character in a channel, pattern 3142 is used to stuff bits. ASCII value of 'a' is 97 i.e 01100001. It requires 8 bytes of channel 1 to stuff 'a', 1st bit 1 is stored in 1st byte of channel 1 at position 3, 2nd bit 0 is stored in 2nd byte of channel 1 at position 1 and so on according to the pattern chosen. This gives additional security and robustness for encoding scheme.

Pseudocode for encoding is as follows;

Consider a two dimensional array (buffer) which contains the values of frames. row represents the channels in the .wav file and column represents the values of frame.

Encoding one character in a channel: i represents the channel and j represents the value, ascii contains the ascii value of the character.

```
Encode(int buffer[][],int i, int j, int ascii)
begin
for k<-0 to 8 k<-k+4
  temp<-asci&1     //assign temp to ascii&1
  ascii<-ascii>>1   //right shift ascii 1 time
  temp<-temp<<2  //left shift ascii by 2 times
  buffer(i,j+k)<-(buffer(i,j+k)&0xfffb)|temp//3rd bit
  temp<-asci&1
  asci<-asci>>1       //right shift ascii 1 time
  buffer(i,j+k+1)<-(buffer(i,j+k+1)&0xfffe)|temp //1st bit
  temp<-asci&1
  asci<-asci>>1       //right shift ascii 1 time
  temp<-temp<<3    //left shift ascii by 3 times
  buffer(i,j+k+2)=(buffer(i,j+k+2)&0xfff7)|temp //4th bit
  temp<-asci&1
  asci<-asci>>1       //right shift ascii 1 time
  temp<-temp<<1    //left shift ascii by 1 time
  buffer(i,j+k+3)<-( buffer(i,j+k+3)&0xfffd)|temp //2nd bit
end for
end
```

B. Decoding

The same pattern which is used to encode is used to decode the modified wav file to get back the original message. Each byte of secret message stuffed in wav file is retrieved from consecutive channels using the same pattern used for encoding Pseudocode for decoding is as follows;

Decoding one character in a channel: i represent the channel and j represent the value, ch is used to store the ascii value of the character and temp is a local variable.

```
Decode(int buffer[][],int I,int j)
begin
for k<-0 to 8  k<-k+4
          temp<- buffer(i,j+k)&0x4//3rd bit
    if(temp>0)
        ch<-ch+power(2, k)
//power is a function which computes 2k
    temp<- buffer(i,j+k+1)&0x1//1st bit
    if(temp>0)
        ch<-ch power (2, k+1)
    temp<- buffer(i,j+k+2)&0x8//4th bit
    if(temp>0)
        ch<-ch+ power (2, k+2)
    temp<- buffer(i,j+k+3)&0x2//2nd bit
    if(temp>0)
         ch<-ch+ power (2, k+3)
end for
end
```

C. Advantages

- Simple and easy to encrypt and decrypt
- Provides shield to possible guessing of steganography used
- Intruder cannot guess the message by sequential reading of wav files
- Robust and secure than conventional algorithm

ISSN: 2278 - 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
Vol. 1, Issue 6, December 2012

IV. RESULTS AND DISCUSSIONS

The proposed technique has been implemented and tested. A number of cases has been tested to check the robustness of the algorithm and we are able to successfully hide text in wav files without much deviation from the original sound that human auditory system can detect. The deviation from the original wav file increases with the increase in message size to be encoded. The results are explained with test cases for large text size and relatively smaller one with the graphs representing frequency v/s signal strength.

Test Case A. Text Size is Small

Graph of Frequency v/s Signal strength of original wav file with 2 channels and size 570kB and wav file modified by stuffing message of small size of 2kB is as shown

In this test case, we have stuffed smaller data of 2kB i.e around 2000 characters in a Wav file of size 570kB with 2 channels. From the graphs shown , it is clear that the deviation is so less that a human auditory system can never distinguish between the two even though it is not exactly the same as original.

Original audio:

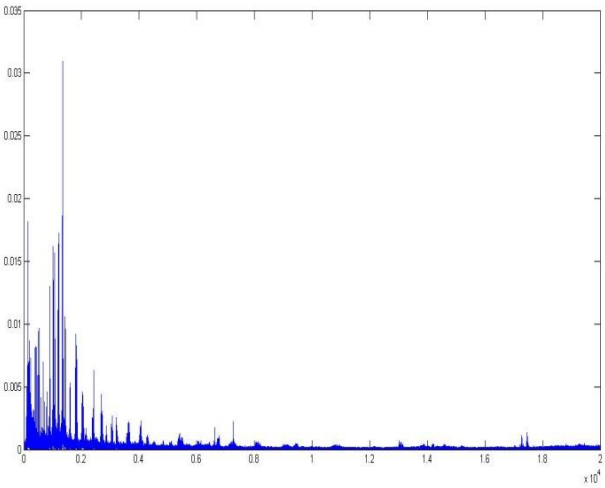

Fig. 2 : Frequency v/s Signal strength graph for original wav audio of 2 channels and size 570kB

Audio after stuffing text of small size :

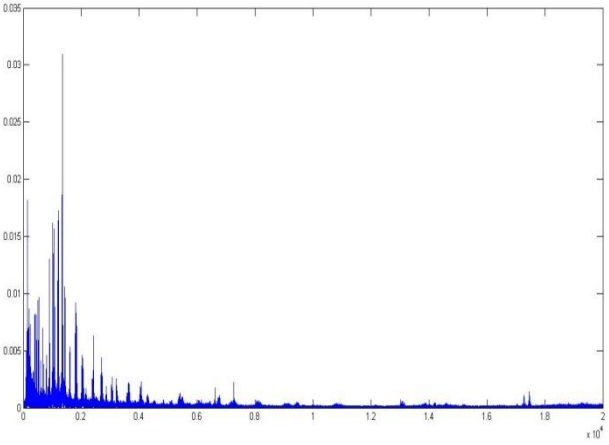

Fig 3: Frequency v/s Signal strength graph for audio after stuffing text of smaller size 2kB

ISSN: 2278 - 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
Vol. 1, Issue 6, December 2012

Test Case B. Text Size is Relatively Large

In this test case, we have stuffed smaller data of 64kB i.e around 64000 characters in a Wav file of size 570kB. We can clearly infer from the graph that though there is no much deviation from original graph, there are spikes in the region of frequencies around 10kHz and 15kHz. A noise is also heard compared to the original audio. Since the 4th bit is manipulated at the probability of 0.25, the change in a byte value will be maximum of 8 and hence it is acceptable.

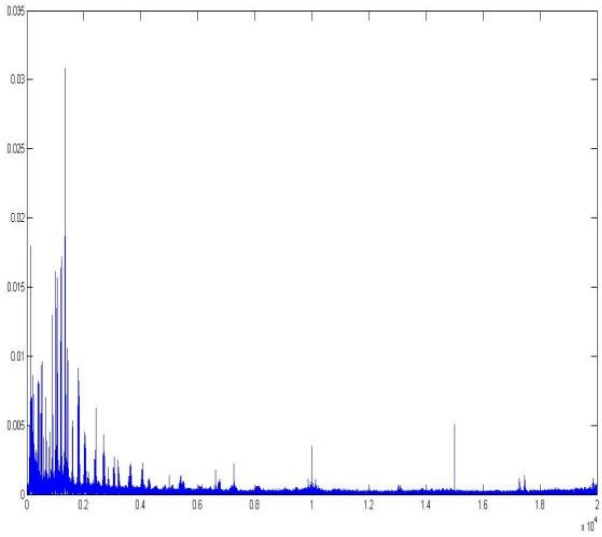

Fig 3: Frequency v/s Signal strength graph for audio after stuffing text of relatively larger size of 64kB

Various other cases were tested by taking different wav files with different number of channels and considering different sizes of wav files as well as text files. Text files of same size but different content were also taken and tested.

The inference from these test cases is that the change that human auditory system can distinguish depends even on the text that is stuffed whose bit pattern decides the extent of change in the byte values of original audio. The bit pattern of text decides the distribution of change in signal strength.

V. CONCLUSION

This proposed system provides a better, efficient method to hide the secret data from intruders and send to the destination in a safe and secure manner. In the scheme proposed the size of the file will not change after stuffing the secret message. Provides robustness and capacity to hold more data without giving up its original content significantly, the desired characteristics required for data hiding.

VI. FUTURE ENHANCEMENTS

In the proposed system , we made use of pattern while stuffing secret message in a channel of a wav file. In a similar fashion, we can even use a pattern to select channels of wav file to stuff the secret data and make it harder to be cracked. We can even make use of encryption and decryption algorithms and stuff cipher text instead of plain text and make it more robust.

ACKNOWLEDGEMENT

We gratefully acknowledge the support for this paper from the Department of Computer Science and Engineering, R.V College of Engineering, Bangalore, India. Our sincere thanks to project guide and other professors for their valuable suggestions and support.

REFERENCES

[1] Neil F.Johnson, Z.Duric and S.Jajodia. "Information Hiding: Steganography and Watermarking—Attacks and Countermeasures", J. Electron. Imaging. 10(3), 825-826 ,Jul 01, 2001

[2] Nedeljko Cvejic, Tapio Seppben , "Increasing the capacity of LSB-based audio steganography " FIN-90014 University of Oulu, Finland ,pp. 336-338, Dec 2002.

[3] David, K., "The History of Steganography", Proc. of First Int. Workshop on Information Hiding, Cambridge, UK , Lecture notes in Computer Science, Vol.1174,p 1-5, May30-June1, 1996

[4] Bender, W.; Gruhl, D.; Morimoto, N.; Lu, A., "Techniques for data hiding", IBM Systems Journal, vol. 35, Issues 3&4, pp. 313-336, 1996.

[5] Jayaram P, Ranganatha H R, Anupama H S," Information hiding using audio steganography- A survey", The International Journal of Multimedia and Its Applications,vol.3, No.3,pp. 86-96, August 2011

[6] Basu, P. N., Bhowmik T., "On Embedding of Text in Audio– A case of Steganography", International Conference on Recent Trends in Information, Telecommunication and Computing, pp.203-206, Mar. 2010.

Biography

Deepak D is currently doing his Engineering degree in R V College of Engineering Bangalore. His areas of interest are Networking, Operating Systems, Data Structures, Java and Cryptography.

Karthik M L is currently doing his Engineering degree in R V College of Engineering Bangalore. His areas of interest are RDBMS, Data Structures, Computer Architecture and Operating Systems.

Manjunath A E is working as an Assistant Professor in R V College of Engineering Bangalore. He has published the paper "Optimized Navigation System using Handheld Devices". His areas of interest are Device Drivers, Context Aware computing, Adhoc Networks, Mobile Computing, Java / J2EE and Java native interface (JNI).