Proc. of the Intl. Conf. on Advances in Computer Science and Electronics Engineering — CSEE 2013

ISBN: 978-981-07-5461-7

A Modified Echo Hiding Technique for Hexadecimal Data

Vivek Sampat Kapil Dave Nikita Sakhare Pragati Pejlekar

***Abstract*—Various techniques for information hiding in audio files are available and are widely used for secure communication and digital identification. Commonly used methods of steganography add data into a cover file in the form of noise. However, retaining the information hidden in a particular signal after compression or other processing performed on the latter is inevitable. Echo hiding introduces change in the cover audio, rather than adding noise to it, which makes the hidden information immune to any kind of processing performed on the signal. We provide an enhancement to the conventional technique for echo hiding and the method of finding suitable parts in the file fit for hiding data without loss.**

Keywords- Echo hiding, audio steganography

I. INTRODUCTION

Numerous computer applications require a particular piece of information to be embedded into a digital file. The embedded information must not be perceivable to unintended individuals. With the extensive amount of research in the field of data hiding, a number of techniques for hiding data in digital files have evolved, emerged, and have found practical applications like broadcast monitoring, ownership identification, proof of ownership, authentication, transactional watermarks (fingerprint), copy control and covert communication [2].

A variety of techniques for audio steganography have been suggested by Bender et al. [3]. One amongst them is the technique of echo hiding where the echo are used to hide data into audio files. The echo hiding system hides data in the form of echoes in a cover audio file. This techniques exploits the holes present in the wide perceptive range of HAS (human auditory system) to determine where data can be hidden int the audio.

Vivek Sampat, Nikita Sakhare, Pragati Pejlekar
Saraswari College of Engineering
India

Kapil Dave
Padmabhushan Vasantdada Patil College of Engineering
India

EH scheme was first introduced by Bender et al. [3] and Gruhl et al. [1]. The echo kernel is expressed as

$$h(n)=\delta(n)+\alpha\delta(n-d) \text{ [1]} \quad (1)$$

If the cover signal produces one echo, the data being transferred is one bit. In order to hide more than one bit, the cover audio is divided into smaller sections or chunks. Each of this chunk will contain one bit of information. [1]

The benefits of the echo hiding technique have been described in [1] and [3]. Firstly, it is stated that the original scheme of echo hiding adds data into the cover file with minimum degradation, that is, the average human cannot hear any objectionable distortion in the stego audio. In echo hiding, data is hidden by introducing changes to the cover audio that are characteristics of environmental conditions rather than random noise. Hence, robustness with regards to filtering, re-sampling, block editing, or lossy data compression is also seen.

Various modification to the echo hiding scheme have been suggested. These modifications include use of multi-echo kernel [1], a pre-echo kernel [1], backward and forward kernel [4], mirrored kernel [5]. These modifications are made in order to improve the recovery of data, or protect against malicious attacks [1] [6].

Though the above discussed techniques fairs well in terms of robustness the amount of data being transferred is very less as traditionally echo hiding deals with binary data. The modifications proposed in this paper hide four bits of data in the chunks of the audio file, using hexadecimal data, improving the amount of data transferred in one file. The proposed system also uses an additional technique for finding chunks that are fit for hiding data. This technique of finding 'good chunks' in the audio file will prevent any loss of data while embedding.

II. RELATED WORK

A novel echo-hiding scheme with backward and forward kernels [4]

The paper presents a novel echo-hiding method for audio watermarking. The method is quite different from previous echo-hiding methods since it presents a new echo kernel which introduces a forward kernel as well as a backward kernel. The new kernel, a combination of the backward and forward kernels, can enhance considerably the watermark detection

Proc. of the Intl. Conf. on Advances in Computer Science and Electronics Engineering — CSEE 2013

ISBN: 978-981-07-5461-7

rate. Thus, it is possible to reduce echo amplitude. The paper proves mathematically that the combination of kernels improves watermark performance. Experimental results show that the proposed watermarking scheme is much better than previous echo-hiding schemes in terms of detection rate and imperceptibility.

Highly robust, secure, and perceptual-quality echo hiding scheme [7]

This paper presents the echo hiding scheme in which the analysis-by-synthesis approach, interlaced kernels, and frequency hopping are adopted to achieve high robustness, security, and perceptual quality. The amplitudes of the embedded echoes are adequately adapted during the embedding process by considering not only the characteristics of the host signals, but also cases in which the watermarked audio signals have suffered various attacks. Additionally, the interlaced kernels are introduced such that the echo positions of the interlaced kernels for embedding "zero" and "one" are interchanged alternately to minimize the influence of host signals and various attacks on the watermarked data. Frequency hopping is employed to increase the robustness and security of the proposed echo hiding scheme in which each audio segment for watermarking is established by combining the fractions selected from all frequency bands based on a pseudo noise sequence as a secret key. Experimental results indicate that the proposed analysis-by-synthesis echo hiding scheme is superior to the conventional schemes in terms of robustness, security, and perceptual quality.

III. PROPOSED SYSTEM

The proposed system will introduce echoes which can have four separate values of offset and intensity each. hence making a possible 16 sets with different offset and intensity. Each of the hexadecimal value will be assigned one of these 16 sets, the value of which will be used while embedding the echo for the particular value. The detailed algorithm including each of these steps is explained below:

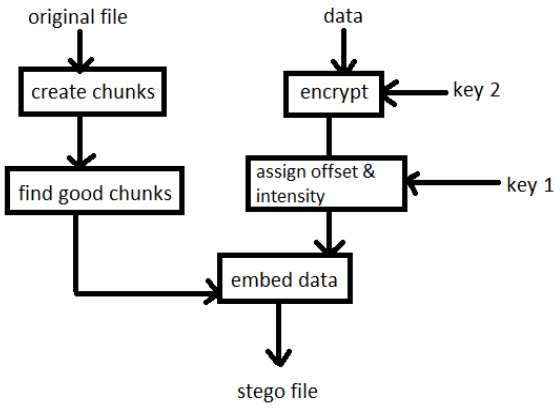

Figure 1: Encoding Process

A. *ENCODING:*

Step 1: Pre processing audio

1. *Divide into chunks:* The audio file provided by the user is checked for the sample rate. Then the file is divided into chunks where each chunk consist of x*fs samples, where x will be the duration of the chunk in seconds. Hence, every chunk will consist of x second's.
2. *Check for good chunks:* For embedding of data, an analysis of each of the chunks is done to find whether the chunk is good for embedding the data. The analysis checks for the following:

a) There should enough data in the part to be echoed so that its echo is recognizable by the decoder. Suppose there is a silence in part to be echoed then its resulting echo will have a negligible intensity which will be difficult to recognize by the decoder causing loss of data. This is seen in fig 2(a) and 2(b).

Proc. of the Intl. Conf. on Advances in Computer Science and Electronics Engineering — CSEE 2013

ISBN: 978-981-07-5461-7

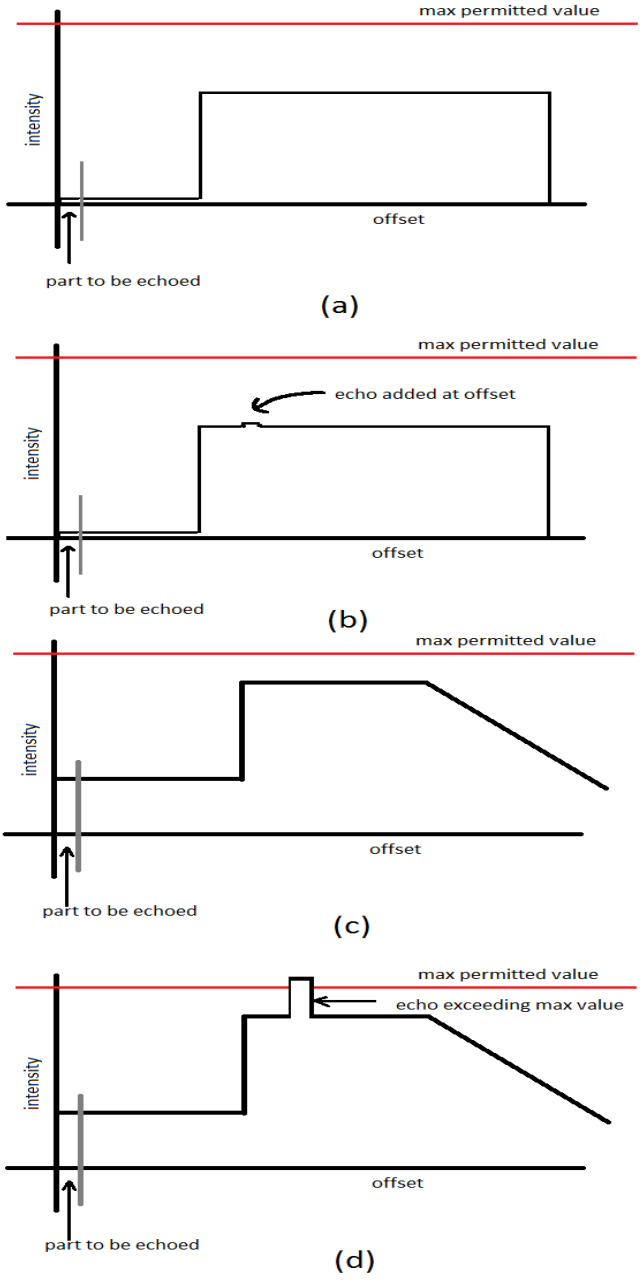

Figure 2: Example of Bad Chunks

b) The summation of the part to be echoed and the part at the offset should not exceed the maximum permitted value of the file format. If the intensity of the part to be echoed and the part at which the echo is inserted is large then their addition may result in a value which exceeds the maximum value that can be represented in the audio and will result into truncation while writing the file. This will result to loss of the data (as truncation will change the intensity). This is seen in fig 2(c) and 2(d).

The chunks satisfying the criteria are marked as good chunks.

Step 2: Pre process message

1. *Encrypt:* The message is first encrypted using the any encryption algorithm. The encryption algorithm is according to the choice of the system designer. 'Key1', as shown in the diagram, is the key used to encrypt the message.
2. *Convert to hexadecimal:* The encrypted message is converted into the hexadecimal format. This is done as the embedding process accepts hexadecimal values to hide in the chunk.

Step 3: Pre-embed

1. *Check capacity:* Each hexadecimal value of the message will be hidden in one chunk. So,

$$Nc >= Nm \quad (2)$$

where,
Nc is the no. of good chunks
Nm is no. of hexa values to be hidden

2. *Generate offset and intensity:* Any pattern or algorithm can be used to generate the offset and intensity values to be assigned to a given hexadecimal value.

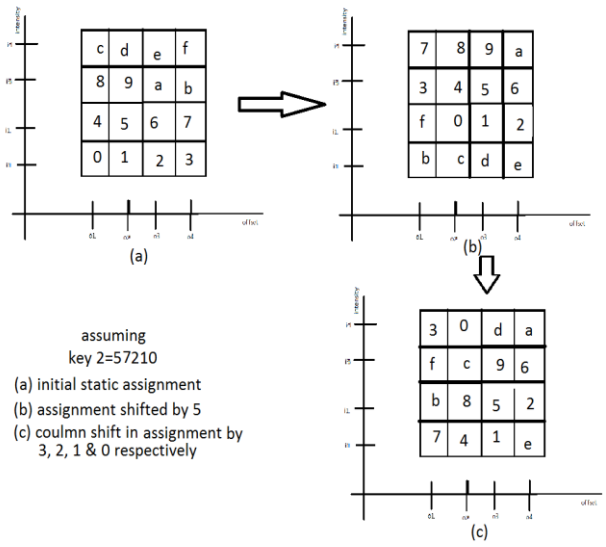

Figure 3: Assignment of Offset and Intensity

Example: Suppose a is the initial static assignment taken is shown in the figure (a).Now we take assume 'key2' as 57210.We first shift the whole assignment by 5 the first digit in the key to obtain figure (b). For the next four digits their mod 4 values are considered. Eg:7 mod 4=3.Then every column is shifted by the resulting values in order respectively to get fig 3.

This generated assignment can be used throughout the file, or various such assignments can be generated using different methods, keys or different initial static assignment. The multiple assignments thus generated can be changed after every 'n' echoes or in a sequentially circular manner, where an assignment is repeated every 'm'th time, where 'm' is the

Proc. of the Intl. Conf. on Advances in Computer Science and Electronics Engineering — CSEE 2013

ISBN: 978-981-07-5461-7

cycle length. The flexibility and the complexity of this algorithm is left to the choice of system designer.

Step 4: Embedding

An echo is embedded in every good chunk for each hexadecimal value in the message. We have taken this echo to be of the first 0.05sec of the 1 second chunk. The intensity and the offset of this echo is in accordance with the assignment that is used so one of the 4 values for the intensity and offset will be chosen.

After all embedding is done the chunks are concatenated original sequence; the good chunks will replace their original counterparts and the bad chunks remain as it is hence completing the process of generating a stego file.

B. *DECODING:*

We know security is obtained by something a user has and something a user knows. The user will require the original file along with the keys used for encryption and assignment of offset and intensity. The steps are explained below

Step 1: Subtraction

The original file is subtracted from the stego file. So the remainder is only the echoes that were embedded.

Step 2: Get encrypted Data

1. *Divide into chunks:* This is similar to the process of dividing into chunks done during embedding.
2. *Generate the assignments for offset and intensity:* This is the same procedure which is while embedding.
3. *Get data within each chunk:* The offsets of the echoes in the chunk can be determined by the occurrence of signal if any. Their intensities can be determined based on the occurred echo, along with data from the original file.

The offsets and intensities obtained are checked with the assignment belonging to the particular chunk and the data is obtained.

Step 3: Decrypt

The obtained data is in the encrypted form and can be decrypted using key 1. the decryption will lead to obtaining the hidden data.

IV. EXPERIMENTAL RESULTS

The system was implemented in MATLAB R2011a, using mono .wav files, with varying sample rate. The chunks were taken of 1 second duration, with the first 0.05 seconds of the chunk echoed at offsets 0.3, 0.4, 0.5 or 0.6 and intensities to be 20%, 30%, 40% or 50% of the first 0.05 seconds.

The technique was tested on clips of various genres and speeches. Bollywood songs, blues and soft rock had more good chunks compared to hard rock and metal.

The echoes added were generally not perceivable in most audio clips. However, there were a certain glitches when data was hidden in speeches and other audio files containing silent parts where the echoes were not masked by the sound in the audio.

V. CONCLUSION

The introduced enhancement in the conventional echo hiding system provides better capacity for hiding data using hexadecimal format. The encoding and decoding processes are efficient and provide better results, as the technique for selecting good chunks guarantees no loss of data. Better security is provided by the flexibility obtained in using various methods for assignment of offset and intensities. Thus the proposed system is better efficient, more usable and secure.

VI. FUTURE WORK

Multiple echoes can be introduced to further increase the hiding capacity. The techniques to find good chunks can be more dynamic and efficient by focusing more on the good areas and then creating chunks instead of creating the chunks first and then finding good ones. Also the technique of forward backward kernels can be incorporated for better decoding of data.

ACKNOWLEDGEMENT

The Authors would like to thank the institutes: Saraswati College of Engineering and Padmabhushan Vasantdada Patil Pratihans College of Engineering for providing its valuable resources and assistance to the authors in their research work.

VII. REFERENCES

[1] Gruhl, Daniel, Anthony Lu, and Walter Bender. "Echo hiding." Information Hiding. Springer Berlin/Heidelberg, 1996.

[2] Cox, Ingemar J., Matt L. Miller, and Jeffrey A. Bloom. "Watermarking applications and their properties." Information Technology: Coding and Computing, 2000. Proceedings. International Conference on. IEEE, 2000.

[3] Bender, Walter, et al. "Techniques for data hiding." IBM systems journal 35.3.4 (1996): 313-336.

[4] Kim, Hyoung Joong, and Yong Hee Choi. "A novel echo-hiding scheme with backward and forward kernels." Circuits and Systems for Video Technology, IEEE Transactions on 13.8 (2003): 885-889.

[5] Wu, Wen-chih, and OT-C. Chen. "Analysis-by-synthesis echo hiding scheme using mirrored kernels." Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on. Vol. 2. IEEE, 2006.

[6] Chou, S. A., and Shih-Fu Hsieh. "An echo-hiding watermarking technique based on bilateral symmetric time spread kernel." Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on. Vol. 3. IEEE, 2006.

[7] Chen, OT-C., and Wen-Chih Wu. "Highly robust, secure, and perceptual-quality echo hiding scheme." *Audio, Speech, and Language Processing, IEEE Transactions on* 16.3 (2008): 629-638.

[8] Wang, Huiqin, et al. "Fuzzy self-adaptive digital audio watermarking based on time-spread echo hiding." *Applied Acoustics* 69.10 (2008): 868-874.

[9] Huang, Dong-Yan, and Theng Yeo. "Robust and inaudible multi-echo audio watermarking." *Advances in Multimedia Information Processing—PCM 2002*(2002): 45-75.

ISBN: 978-981-07-5461-7

[10] Oh, Hyen O., et al. "New echo embedding technique for robust and imperceptible audio watermarking." *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*. Vol. 3. IEEE, 2001.

[11] Li, Li, and Ya-Qi Song. "Experimental research on parameter selection of echo hiding in voice." *Machine Learning and Cybernetics, 2009 International Conference on*. Vol. 4. IEEE, 2009.

[12] Katzenbeisser, Stephan, and Fabien Petitolas. "Information Hiding Techniques for Steganography and Digital Watermaking." (2000): 1-2.

[13] Foo, Say Wei, Theng Hee Yeo, and Dong Yan Huang. "An adaptive audio watermarking system." *Electrical and Electronic Technology, 2001. TENCON. Proceedings of IEEE Region 10 International Conference on*. Vol. 2. IEEE, 2001.

[14] Ko, Byeong-Seob, Ryouichi Nishimura, and Yôiti Suzuki. "Time-spread echo method for digital audio watermarking." *Multimedia, IEEE Transactions on* 7.2 (2005): 212-221.