Audio Embedding with an Error Correction Algorithm

Husnu Narman

Introduction

Multimedia data hiding is the one of the most important topics which has numerous application areas. Such applications include, but not limited to, security, media, covered communication, and annotation. [4],[5]. Therefore, it takes considerable attention by researchers. Many hiding algorithms have been proposed to develop a technique to hide data in a secure way.

There are two important requirements in audio embedding. Two of them, which are perceptual transparency, and high data rate of hidden data mentioned by Nedeljko and Seppänen [6]

Steganography is an art of language which takes advantages of the weakness of the human auditory and visual systems to communicate. Audio steganography is challenging to human auditory system [HAS]. The human ear can detect changes in audio files as low as one part in 10 million. [3],[7]. Therefore, throughout this paper it is assumed that small changes in an audio file cannot be detected by HAS. However, due to the audio file specifications, we can encounter errors while decoding hidden data from embedded audio. It is important to protect the ratio between transparency, decoding without errors, and hidden data rate. To manage this goal, numerous techniques to hide data behind an audio file have been developed. Modifying lower tones, changing frames, translating peak to integer, and several others can be found in the literature.

While hiding data inside a waveform audio file (WAV), the audio file should be modified by considering the two requirements mentioned above. The best way to accomplish this task is to make small changes in the peaks of WAV file which have values between [-1, 1]. For example, the audio values are transformed to integers between *0-255* by using;

$$fix\left(\frac{255}{2}\times(audio\ peak\ value+1\right)$$

where *fix* is a MATLAB function that truncates the decimal part of a rational number. To make small modifications, *0-1000* range can be used. To illustrate;

$$fix\left(\frac{1000}{2}\times(audio\ peak\ value+1\right)$$

Detailed explanation about how to make use of these modifications are discussed in the next section.

Due the latter modification, the stego cannot be decoded correctly. Hence, it is necessary to implement error correction in the code to protect stego. In this paper, Hamming Code [9] is used for error correction.

Proposed Algorithm

In this section, two audio steganography algorithms are explained. The first one is to use any functional transform to the specified integer range within threshold to hide the message. Second algorithm is audio embedding using error correction method

based on functional integer transformation. Integer transformation is also used by Agaian et al. [3]. The method described by Agaian is improved to obtain higher SNR and higher rate of hidden data.

The aim of the second algorithm is to reduce the error of hidden message after decoding even if the hidden message can be a large file or image. The difference of this method from the first one is implementation of Hamming Code.

First Algorithm - Encoding:

Step1: Translate secret file to binary and read audio file using *wavread* function.
Step2: Find the peaks to hide based on the user specified threshold value.
Step3: Use a transformation function to obtain integer numbers for the peak values (by using the range between 0 and 255, the hidden data recovered with negligibly small errors. However, when the ranged is increased, the hidden data is recovered with noticeable errors.).
Step4: Modify peak not to exceed range [-1, 1] and the [-threshold, threshold] to hide stego file data bits.
Step5: Write the modified audio file using *wavwrite* function.

First Algorithm - Decoding:

Step1: Read the audio file using *wavread* function
Step2: Apply the integer transformation to the peaks which are outside of [-threshold, threshold] interval.
Step3: Evaluate mod(2) of transformed integers and add to buffer.
Step4: Invert the buffer in binary form to secret data.
Step5: Display the result.

Second Algorithm - Encoding:

Step1: Translate secret file to binary and read audio file using *wavread* function.
Step2: Split the binary stego file into eight bits (Depending on which error correction is selected).
Step3: Send to error handler to create 12 bit blocks with error correction bits.
Step4: Find the peaks to hide based on the user specified threshold value.
Step5: Use a transformation function to obtain integer numbers for the peak values.
Step6: Modify peak not to exceed range [-1, 1] and the [-threshold, threshold] to hide error correction bits using 0-255 interval integer transformation.
Step7: Modify peak not to exceed range [-1, 1] and the [-threshold, threshold] to hide stego file bits using 0-1000 interval integer transformation or any other suitable ranges

Second Algorithm - Decoding:

Step1: Read the audio file using *wavread* function.
Step2: Split into 12 bit blocks.
Step3: Apply integer transformation for both error correction bits (range 0-255) and stego file data bits (0-1000 or any other suitable ranges).
Step4: Evaluate mod (2) of transformed integers.
Step5: Check the blocks for error using Hamming Code.
Step6: Add the corrected 8 bit blocks to buffer.
Step7: Invert the buffer in binary form to secret data.

Experimental Result

Signal to Noise Ratio (SNR) and Mean Square Error (MSE) measures are calculated for the raw audio and the embedded audio files for comparison. In addition, another measure,

Percentage Stego Error (PSE), is presented to compare the decoded stego file with the original stego file.

$$MSE(\varphi) = \sum E((\varphi - \varphi^*)^2)$$

$$SNR(\varphi) = 10 \cdot \log_{10}\left[\frac{\sum \varphi^2}{MSE(\varphi)}\right]$$

$$PSE = \frac{\sum Found\ error\ bits}{Stego\ size} \times 100$$

Table 1 and Table 2 show the analysis of five audio files. (Same threshold, 0.08, is used for all experiments)

As it can be observed from tables, by looking at PSE values, 0-1000 integer range transformation with error correction ([0-1000]*) is able to recover the hidden data more correctly than 0-1000 range transformation without error correction. The tables also show that [0-1000]* method always has higher SNR than 0-255 range transformation.

Conclusion

Two algorithms for digital audio steganography are presented. Experimental results show that the changes in audio section to hide data are inaudible while the integer range is large enough. We also successfully developed the audio embedding using an error correction method based on functional integer transformation algorithm to decrease the amount of error while decoding. Hamming Code is selected as the error correction algorithm because of its easy implementation.

The error correction provides an additional security wall since the error correction bits are only known by the programmer. Future work will include the study of other error correction algorithms.

Table 1: MSE, SNR, and PSE for two audio files processed with different integer intervals for relatively large stego files

	Method	Time	Data	MSE	SNR	PSE
File 1	[0-255]	29	48kb	0.174	33.608	2.54E-04
	[0-1000]	29	48kb	0.0446	45.42	0.351
	[0-1000]*	29	48kb	0.1307	36.093	0.0725
	[0-255]	29	3.16kb	0.0446	45.424	0.0039
	[0-1000]	29	3.16kb	0.0117	57.048	0.528
	[0-1000]*	29	3.16kb	0.0335	47.904	0.0424
File 2	[0-255]	29	48kb	0.183	33.875	2.54E-04
	[0-1000]	29	48kb	0.047	45.683	0.3537
	[0-1000]*	29	48kb	0.1378	36.34	0.0643
	[0-255]	29	3.16kb	0.0467	45.729	0.0039
	[0-1000]	29	3.16kb	0.0121	57.449	0.4124
	[0-1000]*	29	3.16kb	0.0353	48.163	0.0732

* Implies method includes error correction.

Table 2: MSE, SNR, and PSE for two audio files processed with different integer intervals for relatively small stego files

	Method	Time	Data	MSE	SNR	PSE
File 3	[0-255]	32	3.16kb	0.0424	39.368	0.0039
	[0-1000]	32	3.16kb	0.0111	51.045	0.0809
	[0-1000]*	32	3.16kb	0.0319	41.834	0.0308
File 4	[0-255]	9	3.16kb	0.0467	48.809	0.3507
	[0-1000]	9	3.16kb	0.0122	60.471	0.3468
	[0-1000]*	9	3.16kb	0.0354	51.205	0.0925
File 5	[0-255]	4	3.16kb	0.0445	34.118	0.0039
	[0-1000]	4	3.16kb	0.0116	45.79	0.4856
	[0-1000]*	4	3.16kb	0.0337	36.522	0.0848

* Implies method includes error correction.

References

[1] K. Gopalan and S. Wenndt, "*Audio Steganography for Covert Data Transmission by Imperceptible Tone Insertion",* Communication Systems and Applications - 2004

[2] S. Agaian, D. Akopian, O. Caglayan, and S. A. D'Souza, (2005) "*Lossless Adaptive Digital Audio Steganography"* In Proc. IEEE Int. Conf. Signals, Systems and Computers, pp. 903- 906.

[3] K. Gopalan,(2003) *"Audio Steganography Using Bit Modification*", Proc. of the IEEE

2003 International Conference on Multimedia and Exposition (ICME 2003).
[4] N. Cvejic "*Reduced Distortion Bit-Modification for LSB Audio Steganography*", Journal of Universal Computer Science, 11(1): 56-65.
[5] R. J Anderson and F. A. P. Peticolas , (2001) "*On the Limits of the Steganography*", IEEE Journal Selected Areas in Communications, 16(4), pp. 474-481
[6] N. Cvejic and T. Seppänen, (2004*) "Increasing Robustness of High Bit Rate LSB Audio Watermarking Using a Novel LSB Embedding Method*." Proc. International Conference on Information Technology, Las Vegas, NV, 533-537.
[7] W. Bender, D. Gruhl, N. Morimoto, and A. Lu,(1996) *"Techniques for Data Hiding"*, IBM System Journal, Vol. 35, Nos. 3&4, pp. 313-336
[8] S. Agaian, D. Akopian, and S. A. D'Souza, "*FreQuency Domain Based Secure Digital Audio Steganography Algorithms*", Proc of IEEE SP/CAS, 2005 International Workshop on Spectral Methods and Multirate Signal Processing, SMMS, June 20-22, 2005, Riga, Latvia.
[9] R.W. Hamming, "*Digital Filters*", Section 5.8.