Reversible Digital Watermarking of Audio Wav Signal Using Additive Interpolation-Error Expansion

Anand Baby Alias[1] Minu Kuriakose[2]
[1]Student, M. Tech [2]Assistant Professor
[1,2]Electronics and Communication Department, FISAT

Abstract-- Digital watermarking is the process of hiding information in a carrier signal. An audio watermarking is a kind digital watermarking over which the carrier signal is audio. Here in this paper, an advanced audio watermarking scheme called additive interpolation-error expansion technique is proposed. The paper proposes a successful method for embedding and extraction of binary bits and text in audio wav file.
Keywords: Multi-Embedding, Input-Flexibility

I. INTRODUCTION

Reverse data hiding is the process of hiding a data which is normally called as a watermark. The extraction process should recover the original cover signal along with the watermark. The watermark may be perceptible or imperceptible in nature. For an audio signal, watermarking is typically done in order to identify ownership of copyright for that audio. The watermark should do in a way that it is difficult to remove. The watermark information is also copied as the audio is being copied.
A practically useful reversible watermark scheme should meet the requirements [1]: (1) Robustness, (2) Imperceptibility, (3) Readily Embedding and Retrieving. Also it has to be (a) blind, (b) Higher embedding capacity. The reversible watermarking techniques can of three classes [2]: (I) Reversible watermarking scheme by applying data compression, (II) Reversible watermarking scheme by using histogram bin shifting, (III) Reversible watermarking schemes by using difference expansion.
Interpolation is done in order to increase the size. Here in this paper, a reversible audio watermarking algorithm using additive interpolation-error expansion is proposed for the embedding and extraction of binary bits and text.

II. ADDITIVE INTERPOLATION-ERROR EXPANSION APPROACH

This section recalls the additive interpolation-error expansion presented in [3].

A. Embedding Algorithm

The audio wav file has to quantize into q bits and hence the sample values must be in the range $[0,2^q - 1]$.
The proposed algorithm was implemented in MATLAB. First, read any audio wav file. By using any interpolator, find the interpolation of the wav file. Let x be the original wav file and x‟ be its interpolation values. Then the interpolation-errors are calculated using:

$e=x-x'$ (1)

Next is to calculate LM and RM using:

$LM= \arg_{e \epsilon E} \max hist(e)$ (2)

$RM= \arg_{e \epsilon E-[LM]} \max hist(e)$ (3)

Where LM and RM denote the corresponding values of the two highest points of interpolation-error histogram, arg denotes the position and E denotes the set of interpolation-errors. The explanation for (2) and (3) is given as follows.
For calculating LM, first find the histogram of (1). The proposed way is to find the unique values of e and then find the position having the maximum histogram value. The value in that position in the unique value array of e gives LM. For finding RM value, first delete all the LM values from (1) and then precede the same procedure for LM to get (3). Any other methods can also be used for calculating the histogram. In most of the cases LM is 0 and RM is equal to 1.
The next step is to split the error array into two different arrays. The splitting is purely based on the LM and RM values. The splitting is as follows.

Left interpolation-errors (LE): e<=LM (4)

Right interpolation-errors (RE): e>=RM (5)

From the newly developed arrays, find two more parameters LN and RN which are defined as

$LN=\arg_{e \epsilon LE} \min hist(e)$ (6)

$RN=\arg_{e \epsilon RE} \min hist(e)$ (7)

The procedure for finding LN and RN can be done by following the same way for finding (2) and (3). Usually LN is a smaller integer that with no interpolation-error satisfying e = LN and RN is a larger integer with no interpolation-error satisfying e = RN.
After finding LM, RM, LN, RN it is possible to define additive interpolation-error expansion.

$$E'' = \begin{cases} e + sign(e) * b, & e = LM \text{ or } RM \\ e + sign(e) * 1, & e \in (LN, LM)\ U\ (RM, RN) \\ e, & otherwise \end{cases} \quad (8)$$

e‟‟ is the expanded interpolation-error, b is the embedded bit and sign(e) is sign function which is given as

$$\text{sign}(e) = \begin{cases} 1, & e \epsilon RE \\ -1, & e \epsilon LE \end{cases} \quad (9)$$

The actual bit embedding process is carried out in this stage. While implementing, (9) should be defined before (8).
The actual watermarked wav sample x” is given as

$x'' = x'' + e''$ (10)

In order to overcome the overflow/underflow issue, the below given algorithm should be used. Overflow/underflow issue arises when samples are changed from 2^q -1 to 2^q or from 0 to -1.
Algorithm 1
Map = [];
For first_sample to last_sample
if x”=0 V x”=2^q-1
Map= [Map; 0];

elseif x”= -1
Map = [Map; 1];
x”= 0;
elseif x” = 2^q
Map = [Map; 1];
x” = 2^q-1
end
end
The map values will only be generated in case of overflow/underflow happens.

B. Extraction Algorithm

Before the actual data extraction steps begins a reverse of the above given algorithm has to be implemented in order to reverse the truncation produced by that algorithm
Algorithm 2
Map_counter=1;
for first_sample to last_sample
if x”=0 V x”=2^q-1
if Map(Map_counter)=0
Map_counter=Map_counter+1;
elseif Map(Map_counter)=1
If x”=0
x”=-1
elseif x”=2^q-1
x”=2^q
end
Map_counter=Map_counter + 1;
end
end
end
Algorithm 1 and Algorithm 2 are given above directly in the MATLAB format.
Now compute the interpolation values of x” by using the same interpolator used in the embedded stage. The newly obtained interpolation values is x‟. Then compute e‟ using:

$$e' = x'' - x' \quad (11)$$

For extracting the embedded bits

$$b = \begin{cases} 0, & e' = LM \text{ or } RM \\ 1, & e' = LM - 1 \text{ or } RM + 1 \end{cases} \quad (12)$$

For get back the original interpolation-errors,

$$e = \begin{cases} e' - sign(e') * b, e' \epsilon\, [LM-1, LM] U\, [RM, RM+1] \\ e' - sign(e') * 1, \quad e' [LN, LM-1) U\, (RM+1, RN] \\ e', \qquad otherwise \end{cases} \quad (13)$$

It should be checked that (13) and (1) be same.
For getting back the original audio samples

$$x = x'' + e \quad (14)$$

It should be noted that the above embedding algorithm is to embed a single binary bit, either 1 or 0. So for embedding multiple binary bits, multi-embedding approach is explored. This can be done by make use of „for loop‟.

III. INPUT FLEXIBILITY APPROACH

The above explained algorithm works only if the number of bits which is to be embedding should be equal to or greater than the number of input wav array. So in order to overcome this limitation, Input Flexibility Approach is used. The idea is that, after embedding the exact input bits, default values-NaN are given to all the other embedding spaces.

IV. EMBEDDING OF TEXT

The approach is that before the text files are being embedding to the actual implementation code, all the characters are converted into its respective binary values and then after extracting, all the binary values are converted back to characters. The above idea can be adapted by using „de2bin‟ and „bi2de‟ command.

V. RESULTS

A null difference between the original and restored wav is obtained. A sample wav is taken and the results obtained are shown below:

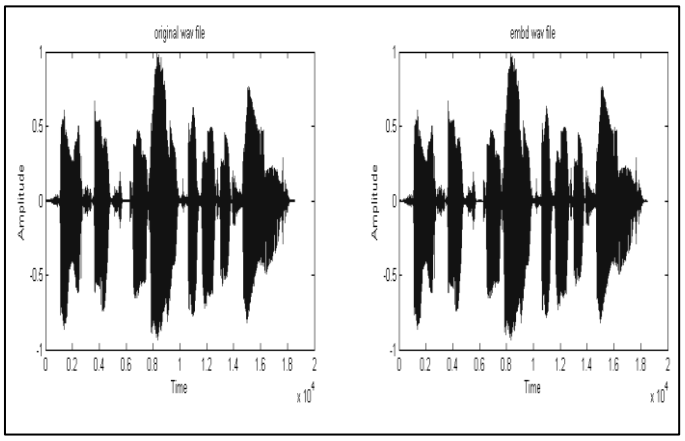

Fig. 1: Original wav and wav after bit embedding

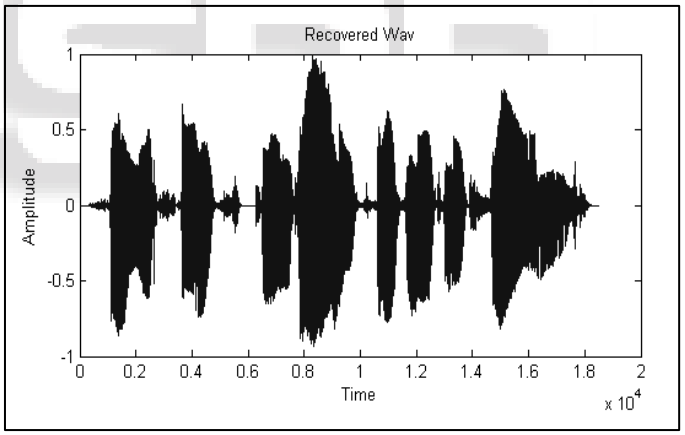

Fig. 2: wav after bit extraction

The null-difference between the embedded wav and the original wav is obtained as:

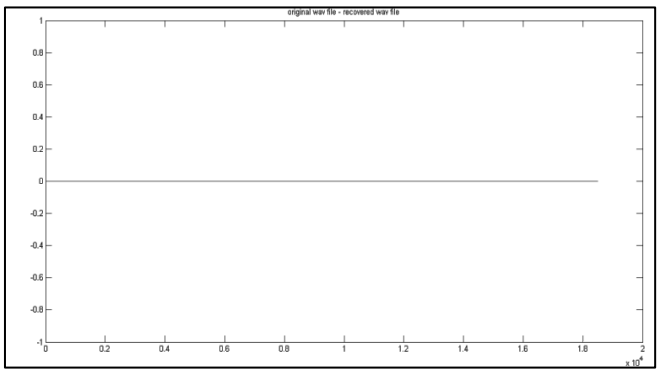

Fig. 3: Original wav – Recovered wav

The null-difference between the bits embedded and extracted bits is obtained as:

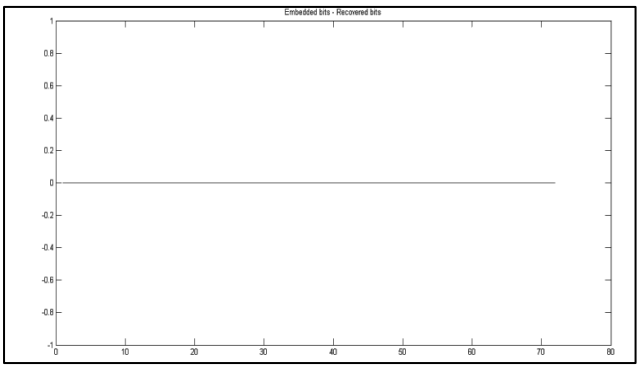

Fig. 4: Embedded bits – Extracted bits

VI. CONCLUSION

In this paper, a reversible audio watermarking scheme by exploring additive interpolation-error expansion is proposed. Method for generation of compact boundary maps is proposed. Multi-embedding alone with Input Flexibility Approach is also explained. The same implementation procedure is then expanded for the embedding of text. The generated MATLAB code was undergone test for around hundreds of input wav file. So it is clear that more research is needed to understand the exact characteristics of the input wav file for multi-embedding approach.

ACKNOWLEDGEMENT

We would like to extend heartful and sincere gratitude to Jose Juan Garcia-Hernandez, the author of "Using additive interpolation-error expansion for reversible digital watermarking in audio signals", 2012 IEEE, for his technical and moral guidance. His paper on "Using additive interpolation-error expansion for reversible digital watermarking in audio signals" was very insightful which acted as the backbone structure for this paper.

VII. REFERENCES

[1]. Jen-Bang Feng, Iuon-Chang Lin, Chwei-Shyong Tsai, and Yen-Ping Chu, "Reversible Watermarking: Current Status and Key Issues", International Journal of Network Security, Vol.2, No.3, PP.161-171, May 2006

[2]. Mikko Loytnoja, Nedeljko Cvejic, Tapio Seppanen, "Audio Protection with Removable Watermarking", 2007 IEEE

[3]. Jose Juan Garcia-Hernandez and Lorenzo Delgado-Guillen, "Using additive interpolation-error expansion for reversible digital watermarking in audio signals", 2012 IEEE

