MULTI-CHANNEL AUDIO INFORMATION HIDING

J M Blackledge, [*]

Audio Research Group
Dublin Institute of Technology
jonathan.blackledge@dit.ie

A I Al-Rawi,[†]

Audio Research Group
Dublin Institute of Technology
abdulrahman.alrawi@gmail.com

R Hickson,[‡]

Audio Research Group
Dublin Institute of Technology
notruairi@gmail.com

ABSTRACT

We consider a method of hiding many audio channels in one host signal. The purpose of this is to provide a 'mix' that incorporates information on all the channels used to produce it, thereby allowing all, or, at least some channels to be stored in the mix for later use (e.g. for re-mixing and/or archiving). After providing an overview of some recently published audio water marking schemes in the time and transform domains, we present a method that is based on using a four least significant bits scheme to embed five MP3 files into a single 16-bit host WAV file without incurring any perceptual audio distortions in either the host data or embedded files. The host WAV file is taken to be the final mix associated with the original multi-channel data before applying minimal MP3 compression (WAV to MP3 conversion), or, alternatively, an arbitrary host WAV file into which other multi-channel data in an MP3 format is hidden. The embedded information can be encrypted and/or the embedding locations randomized on a channel-by-channel basis depending on the security protocol desired by the user. The method is illustrated by providing example m-code for interested readers to reproduce the results obtained to date and as a basis for further development.

1. INTRODUCTION

The approach to information hiding discussed in this paper relates to an investigation into the design of 'intelligent' coding algorithms for audio post-production based on research undertaken jointly between the Audio Research Group [1] and the Digital Media Centre [2] at Dublin Institute of Technology and Tamborine Productions Limited, London [3]. This includes the use of watermarking methods for audio signal authentication and multiple audio signal embedding techniques using computational solutions based on image information hiding and authentication techniques, e.g. [4] and [5].

While there are a number of different approaches for hiding a signal in another 'host signal', inclusive of encryption, or otherwise, there are relatively few published methods that consider a multi-signal to host signal approach. In this paper we consider the problem of embedding many audio signals provided in MP3 formats into a single wave of WAV file under the following conditions: (i) the embedded MP3 audio data can be recovered without any loss of information; (ii) the host signal has no perceptual change in its audio fidelity. An algorithm is presented based on a time domain method in which five MP3 files are hidden in a single WAV file using a four least significant bit scheme which satisfies these conditions. The MP3 files may be encrypted prior to application of the algorithm with the option of encrypting the host data as required.

The method considered has applications in covert information transmission with applications in areas such as communications security, signal authentication and DRM (Digital Rights Management), for example. However, in the context of this paper, the method focuses on an audio post-production product that allows a stereo 'mix' to incorporate information on all or some of the channels used to produce the same mix, thereby allowing all or selective channels to be embedded in the mix for re-mixing at a later date and/or for data storage. Current scenarios usually involves a post-production centre having to maintain records of all the data used to generate a mix with the view that this same data may be required at a later date, thereby necessitating the need for a data base management infrastructure which may require significant overheads.

After providing a detailed overview of recently published works that focus on the area of audio information hiding in the time and transform domains (Section 2), we present a time domain solution to the problem as given in Section 3 which is based on a four least significant bits embedding scheme. We include prototype m-code for interested readers to test the algorithms developed to-date and as a basis for developing and extending the method further.

2. AUDIO INFORMATION HIDING

Audio information hiding or *Audio Steganography* is based on two specific and distinct approaches which consider the development of algorithms using time domain data or transform domain data such as the cosine transform, for example. In order to appreciate the context of the algorithms reported in this paper, we provide a brief overview of some of the more recently published material in this area, publications that have been studied by the author's in the context of developing a multi-channel information hiding scheme.

[*] Stokes Professor, Science Foundation Ireland

[†] Research Assistant, School of Electrical Engineering Systems

[‡] Research Assistant, School of Electrical Engineering Systems

2.1. Audio Hiding in the Time Domain

Time domain audio hiding has the advantage of being relative simple to implement and computationally efficient. However, the approach does not yield algorithms that are as diverse and robust as those developed for the transform domain. This is because of the relatively limited number of ways in which data can be manipulated in the time domain, subject to the generation of an output that is perceptually compatible with the original audio signal. In the literature survey that follows, it is clear that all the methods conceived are based on some form of data manipulation by modifying the binary representation of the audio signal, i.e. variations on the basic theme of modifying the Least Significant Bits of a data stream. For example, in [6], the authors propose a novel audio embedding method using amplitude differencing. It involves embedding a covert message of any format into two cover audio files which are of a similar size. The difference in amplitude values between the two signals is compared to the maximum range index of the range (ranging from 0 to 255 where all audio values are represented as bytes). The covert message is divided into a series of 4-bits and new amplitude difference values calculated for the two cover audio files which contain the message. This process hides 2-bits per audio file with a total capacity of 4-bits in both files. The data extraction process follows the same embedding procedure (in the inverse sense). The main advantage of using two cover audio files is to distribute the payload equally among more than one file which prevents the introduction of noise into the cover files.

In [7] the authors propose two novel approaches of Least Significant Bit (LSB) substitution to improve the capacity of the audio hiding methods. The first approach increases the number of bits that can be used for hiding from 4 LSBs to 7 LSBs by using the first and second Most Significant Bits (MSB), respectively, as follow: (i) if the MSB-1 and MSB-2 values are 00 then 4 LSBs are used for data embedding; (ii) if the values are 01 then 5 LSBs are used; (iii) if the values are 10 then 6 LSBs are used; (iv) if the values are 11 then 7 LSBs are used for data embedding. The second approach shifts the maximum limit by only considering the first MSB where 6 LSBs are used for data hiding if its value is 0 and 7 LSBs are used if its value is 1. The data extraction procedure uses the same scenario for retrieving the embedded data (in an inverse sense).

Audio watermarking schemes based on amplitude modification are presented in [8] using a similar approach to that reported in [9] but with increased hiding capacity. The proposed method embeds the watermark data by modifying the AOAA (Average Of Absolute Amplitude) differences calculated from three sections in a GOS (Group Of Samples). In the watermark embedding process, the original audio signal is divided into consecutive lengths of GOSs where each GOS contains three non-overlapping sections. The AOAAs of these sections are calculated as detailed in [8], the values being sorted in descending order and the differences between them computed. Bit stream embedding is undertaken using a dynamic procedure that depends upon a fixed threshold and the extraction procedure is predicated on knowing the GOSs and using the same principal processing steps coupled with a comparative bit-by-bit analysis as given in [8]. This method is highly robust to attacks including MP3 compression and low-pass filtering.

An encrypted watermarking scheme is discussed in [10]. The 'secrete message' (watermark) is encrypted using a bit exchange encryption method and the encrypted data embedded into the LSB and LSB+3 bits of the cover file. The embedding process starts by dividing each byte of the secret message into groups of two bits, the first 2-bits are embedded into the LSB and LSB+3 bits of the cover file leaving one byte intact. 2-bits are then again embedded into the LSB and LSB+3 bits leaving one byte intact. The same process is repeated for all bits of the secret message yielding an algorithm that embeds one byte of the secret message into 8 bytes of the cover file.

In [11] the author's present two methods to improve the conventional LSB modification method for Audio Steganography. The first method is based on randomizing the LSB bit number of the host file, the bit selection being based on the 1^{st} and 2^{nd} MSBs, respectively. If the MSB-1 and MSB-2 values are 00 then the 3^{rd} LSB is used for embedding a secret bit; if the values are 01 then the 2^{nd} LSB is used, and if the values are 10 or 11 then the 1^{st} LSB is used. The second approach randomizes the sample numbers containing the next secret bit of the data, the decision criterion relying on the first, second and third MSB: If the MSB-1, MSB-2 and MSB-3 values are 000 then the sample containing the next secret bit is i+1; if their values are 001 then the sample containing the next secret bit is i+2; if their values are 010 then the sample containing the next secret bit is i+3, and so on. In both cases (i.e. the methods considered) data extraction is performed by comparing the MSB values to find the LSBs with the hidden data.

Different low-bit coding methods for Audio Steganography are considered in [12]. Conventional low-bit coding methods embed the bits of the watermark into the LSBs of the host audio data. This can be improved by using variable low-bit coding via two approaches. The first approach defines two thresholds T_1 and T_2, say, based on the standard level about a mid-range of 128 where the samples are taken to be bytes with a maximum value 255. If the audio amplitude value is greater than T_2, two bits are used for embedding. If the amplitude value is between T_1 and T_2, one bit is used to embed, no embedding being undertaken if the watermark data has a value less than T_1. The second approach calculates the average amplitude data of the surrounding audio signal to provide a threshold. If the amplitude level is greater than the average value, then 2-bits are embedded in the host signal, else, no embedding is undertaken, the amplitude level being considered to be too low.

In [13] an audio hiding method is proposed that is based on the escape sequences of ACC (Advanced Audio Coding) audio files. The secret information is transformed into a bit stream and XORed with a pseudo random generator sequence to obtain an encrypted data sequence. The AAC file is unpacked and the escape sequences located and used as a 'carrier' for data hiding. The embedding process is performed based on the matrix encoding described in [13]. The LSBs of each escape sequence are extracted as carrier data, the LSBs of every 15 escape sequences constituting a 15 dimensional carrier vector **a**. Every 4-bits of the secret data are taken to form a 4-dimensional vector **b** and a 4-dimensional column vector $\mathbf{c}^T$. **b** is XORed with **c** to generate the vector **d**. A new vector **e** is then calculated and tested as follows: If **e=0**, then no bits are modified, else, the d^{th} bit of the carrier vector is modified so that the LSB of the d^{th} escape sequence is modified. This means that 4-bits of the secret data is embedded (i.e. 4-bits of the secret information is embedded into 15 bits of the carrier). The process is repeated until the secret information is completely embedded or the end of the host audio is reached.

2.2. Audio Hiding in the Transform Domain

Using the transform domain provides the potential to generate a wider class of audio information hiding methods than are available using time domain techniques. However, any transform domain technique comes at the expense of the computational costs involved in computing the transform and its inverse. The transform concerned include the Fourier transform computed using a Fast Fourier Transform (FFT) and transforms such as the Discrete Wavelet Transform and the Discrete Cosine Transform. For example, in [14], an audio watermarking algorithm is considered based on a RSVD (Reduced Singular Value Decomposition) of the FFT of the audio signal. The method relies on manipulating the coefficients of one of the resulting unitary matrices for watermark embedding. The audio signal is split into frames of length *L*, a FFT applied to compute the magnitude spectrum in each frame, and, finally, the frequency components of each frame organized into a RSVD input matrix (RSVD being applied to a matrix rather than a vector). RSVD decomposes each input matrix into three matrices U, S and V [15]. The coefficients of *U* are changed to embed the watermark bits by creating a local peak in $u_{1..p,2}$. If the watermark bit to embed is of type 1, the magnitude of the element $u_{c,2}$ is increased (where c is a user defined key value that represents the peak value) and the surrounding elements $u_{c-1,2}$ and $u_{c+1,2}$ reduced to insignificant amplitude values ϕ. If the watermark bit to be embedded is of type 0, then the magnitude of the element $u_{c+1,2}$ (which represents the peak value) and the surrounding elements $u_{c,2}$ and $u_{c+1,2}$ are reduced to the same insignificant amplitude value ϕ. To extract the hidden watermark, the watermarked signal is split into frames, the FFT applied to each frame and the frequency components organized into a matrix *A*. RSVD is then applied to this matrix and the following process applied: if $u_{c,2}$-$u_{c+1,2}$ > 0, then the watermark is taken to be of bit-type 1, otherwise it is taken to be of type 0.

An audio watermarking algorithm based on the Fast Fourier Transform and a Quadratic Matrix is considered in [16]. A non-singular quadratic form is chosen to obtain a corresponding quadratic matrix *Q* and the secret data converted into a matrix *M* which is taken to be equivalent to *Q*. These matrices are multiplied to generate an encrypted data matrix *E*. The cover audio data is transformed into the frequency domain using a FFT and the secret information *E* embedded into the frequency domain of the cover signal. Retrieval is performed by recovering the hidden data *E* to regenerate the audio cover frequency domain and decryption performed by multiplying *E* by the inverse matrix Q^{-1} to recover the original watermark.

A speech signal watermarking method that uses sinusoidal modeling and QIM (Quantization Index Modulation) is presented in [17]. The principal idea is to hide the watermark data in the phase information of an appropriate sine wave because the human auditory system is insensitive to the absolute phase. The QIM watermarking method [18] for embedding the watermark data divides the speech signal into voiced and unvoiced parts. The voiced part *S* is segmented into *L* overlapped frames and a STFT (Short Time Fourier Transform) applied to each frame. The amplitude and frequency of each component is estimated by locating the peaks of the STFT and the phases computed using the corresponding real and imaginary parts. The appropriate sine wave components are selected and the watermark embedded by changing the phase value of selected components as detailed in [17]. The watermarked signal is synthesized based on the amplitude, frequency and quantized phase value and the watermark data recovered by comparing the phase values with the quantization levels.

Application of multiple scrambling and amplitude modulation is considered in [19]. The watermark data is scrambled with a coded-image instead of a chaotic or pseudo-random sequence since the extracted coded-image can be post-processed to enhance the quality of the recovered watermark. The embedding procedure is initiated by applying a pre-selection process on the host audio file to select only those regions whose power exceeds a certain threshold, thereby preventing the selection of silent potions of the audio signal. The selected segments are then divided into adjacent frames with a 50% overlap and a Short Time Fourier Transform (STFT) applied to them to obtain their frequency spectrum. These segments are further subdivided into *N* two-dimensional blocks, each block being used to embed one sub-watermark *B* where *B* is a part of the watermark *W*) as detailed in [19]. Each block *N* is segmented into different levels of granularities, unit, slot, tile and bin and the watermark bits embedded through the process of amplitude modulation as discussed in [20] and [21]. Finally, the watermarked signal for each frame is constructed in the frequency domain using the magnitude, phase, and the sign of the signal spectrum which is then added to the host frame after its transformation to the time domain. To increase the security of the method, another scrambling operation can be applied by randomly selecting *N'* blocks from *N* blocks and then randomizing their order for encoding. The watermark bits are detected by examining whether there is an increase or decrease in the magnitude of the corresponding tiles of the received signal [21], [22].

A blind audio watermarking algorithm using the wavelet transform and chaotic encryption is presented in [23]. Chaotic encryption is used for encrypting the watermark data and randomizing its (hiding) location. The watermark bit stream is encrypted and the original audio signal divided into non-overlapping frames. Randomization of the order of the frames is undertaken using chaotic sequences and each frame processed with an *L* grade wavelet transform. Finally, the encrypted watermark bits are embedded into the *L* grade detail value by choosing *M* coefficients from the absolute values located between the detail values. The coefficients are then modified according to the watermark bit value as given in [23]. Watermark extraction is performed by applying the inverse process, segmenting the watermarked signal into non-overlapped frames and applying the *L* grade wavelet transform to the frames selected by the chaotic sequence used, the hidden bits being extracted as detailed in [23] followed by decryption to recover the original watermark.

A further example of the application of the wavelet transform is considered in [24] based on an adaptive wavelet packed modification. The embedding method consists of four components: (i) Wavelet packet decomposition is applied to decompose the audio signal using a three level wavelet packet, where the 'HLH band' is used for data embedding; (ii) Binary mapping is applied where the coefficients in the selected band are sampled to evenly distribute the hidden data over the entire host audio data for improving audio imperceptibility. The coefficients are converted from a one-dimensional vector to a two-dimensional matrix and each matrix further subdivided into 2 × 2 blocks *B*. A pseudorandom sequence is applied to select the *B* block to be used for secret data embedding and the selected blocks mapped to binary form using trend detection. If the block values follow an increasing trend, then they are mapped into a bit of type 1 and if the values follow a decreasing trend they are mapped into a bit of type 0, each binary block being referred as a pattern matrix which is classified accordingly [25]; (iii) Adaptive wavelet packet modification is used for embedding. To embed the secret bits, the pattern matrix *B* is modified to obtain *B'* making sure that *B* and *B'* belong to the same class whose range maintains the modified bits with regard to their neighbours; (iv) The inverse wavelet packet is applied to the modified packets to generate a 'stego-audio' signal [24].

Application of an integer wavelet transform is considered in [25]. The scheme is based on two essential ideas: The first is to use a pseudorandom sequence to scramble the secret signal bits and randomize the wavelet coefficients used for data embedding. The second idea is to calculate the 'hearing threshold' to be used as an embedding threshold. The estimation of the hearing threshold is very important because low values lead to low payloads while high values cause auditable distortion. The embedding process begins by performing a Haar DWT (Discrete Wavelet Transform) on the cover audio signal and then converting the DWT coefficients into integer form. The encrypted data is then embedded into the integer DWT coefficients (the number of bits embedded in each coefficient being determined by the hearing threshold). Finally, the modified coefficients are converted into the wavelet domain and the resulting coefficients transformed back into the time domain to generate the 'stego-signal'. The data extraction steps are generally similar to the embedding process.

Application of the Discrete Cosine Transform (DCT) for audio watermarking is considered in [26]. A Hamming error correcting code and a Neural Network (NN) are applied. The cover audio signal is divided into non-overlapped frames of size 512 samples and the DCT applied to each frame. The watermark is divided into sets of length 8 and each set is encoded with a Hamming error correcting code [27], the encoded sets being processed using a pseudorandom sequence for increased security. The bands of the middle frequencies are then located for insertion of the watermark bits. Each band is trained using a BPNN (Back-Propagation Neural Network). The located bands and the NN training result are compared prior to watermark insertion in order to maintain the relationship between the sample to be modified and its neighborhood samples. Following watermark bit insertion an inverse DCT is applied to generate the watermarked audio signal. The extraction process is performed by adopting the equivalent inverse embedding process with knowledge of the the NN weights, pseudorandom sequence and the band positions. The extracted watermark bits are corrected using Hamming decoding to increase the accuracy of the recovered watermark.

Application of the wavelet transform coupled with higher-order statistics are considered in [28]. The two dimensional watermark data (image or audio samples organized into a matrix) is initially scrambled using an Arnold transform and converted into a one dimensional sequence of zeros and ones. A wavelet de-noising algorithm is applied to the host audio data to remove as much noise as possible and recover its original characteristics. The de-noised audio signals are divided into segments, each segment A being further sub-divided into two parts A_1 and A_2 of length L_1 and L_2 elements, respectively. The embedding process starts by embedding a synchronization code into the average value of the audio samples of A_1 as described in [28] which is required to survive de-synchronization. The watermark is embedded into A_2 as follows: sub-divide A_2 into audio sub-segments of length $L2/(M\times N$ samples (where L_2 is the length of A_2 and $M\times N$ is the watermark size), perform an H-level DWT on each sub-segment, embed the watermark into the DWT coefficients based on the higher-order statistics computed using the Hausdorff distance and finally apply an inverse DWT to obtain the watermarked audio segment. The synchronization code and watermark bits are embedded into each audio segment to increase the robustness of the method. The watermark detection is performed by de-noising the watermarked audio signal, detecting the synchronization code and using the detected codes to locate the watermark into audio sub-segments based on higher-order statistics.

An approach using SVD-DCT (Singular Value Decomposition-Discrete Cosine Transform) and synchronization codes is considered in [29]. The host audio signal is partitioned into two parts, the first for synchronization code insertion and the second for watermark embedding. A logistic chaotic sequence with an initial value in the interval $[0, 1]$ is used to generate the synchronization code. To embed the watermark, the host audio signal is organized into a 2D matrix which is then segmented into non-overlapping blocks. An 8×8 block based SVD is used and the first SVD coefficients arranged to form a new matrix where most of the signal energy is concentrated in the largest singular values. This matrix is partitioned into 4×4 sub-blocks and the DCT applied to each block to generate the SVD-DCT coefficient blocks, each block being 'zigzagged' [29]. The watermark embedding starts by selecting the coefficient pairs to be modified (the selection is based on a pseudo-random sequence) from the potential location of each SVD-DCT block. The frequency mask is computed to weight the watermark amplitude (the frequency mask is used to determine the level of tolerance against distortion caused by embedding the watermark), and, finally, the watermark is embedding by modifying the magnitude difference of the selected pairs.

3. MULTI-CHANNEL INFORMATION EMBEDDING

None of the audio watermarking methods discussed in the previous section and other recent publications studied by the authors consider the problem of embedding multiple channels into a single channel. Compared with single channel information hiding, multi-channel watermarking has a number of potential advantages. The most obvious of these is that a number of entirely independent and uncorrelated sources of (audio) information can be embedded into a single audio host signal either as *Plaintext* or *Ciphertext* data streams (depending upon whether data encryption of the hidden information is required).

We consider the 5:1 (the limit for maintaining audio imperceptibility with the current algorithm) multi-channel audio embedding and extraction process illustrated in Figure 1 and Figure 2, respectively, The scheme is based on embedding five MP3 files into a single WAV file using 4 LSB coding. It focuses on the use of five MP3 files in terms of the optimum number of channels that can be embedded subject to the following conditions:

- minimal perceptual distortion of the host WAV file according to the Perceptual Evaluation of Audio Quality (PEAQ ITU-R recommendation BS.1387) discussed in [31];
- high integrity on the fidelity of all channels after extraction from the host signal based on 4-bit LSB coding.

The use of 4-bit LSB is the minimum required to satisfy the second of these conditions and is predicated on experimental results associated with the first condition. This approach follows on from the method reported in [32] which uses a Frequency Modulation method called 'Chirp-Coding', originally designed for the

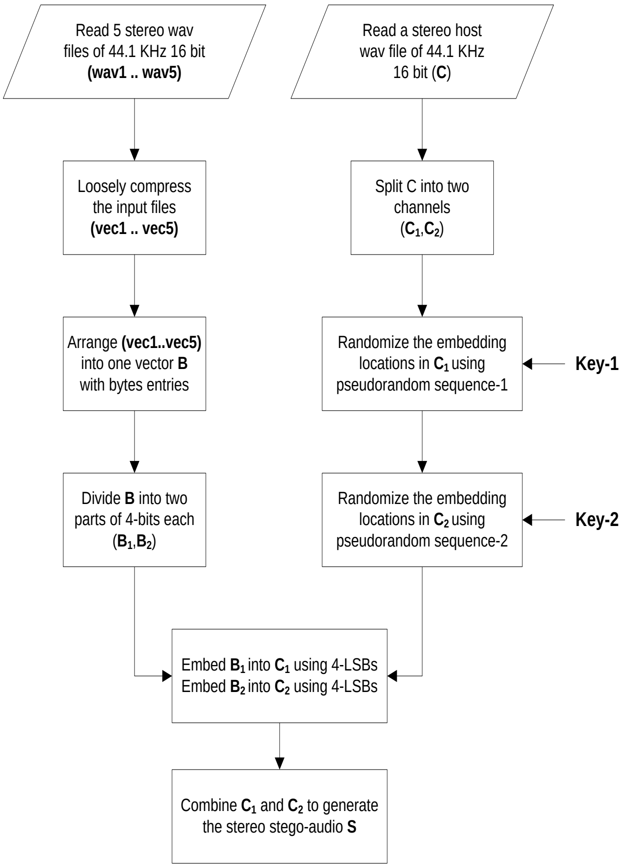
Read 5 stereo wav files of 44.1 KHz 16 bit **(wav1 .. wav5)**
Loosely compress the input files **(vec1 .. vec5)**
Arrange **(vec1..vec5)** into one vector **B** with bytes entries
Divide **B** into two parts of 4-bits each **(B$_1$,B$_2$)**
Read a stereo host wav file of 44.1 KHz 16 bit (**C**)
Split C into two channels (**C$_1$,C$_2$**)
Randomize the embedding locations in **C$_1$** using pseudorandom sequence-1
Key-1
Randomize the embedding locations in **C$_2$** using pseudorandom sequence-2
Key-2
Embed **B$_1$** into **C$_1$** using 4-LSBs
Embed **B$_2$** into **C$_2$** using 4-LSBs
Combine **C$_1$** and **C$_2$** to generate the stereo stego-audio **S**

Figure 1: High level data flow diagram for the 5:1 multi-channel audio embedding process.

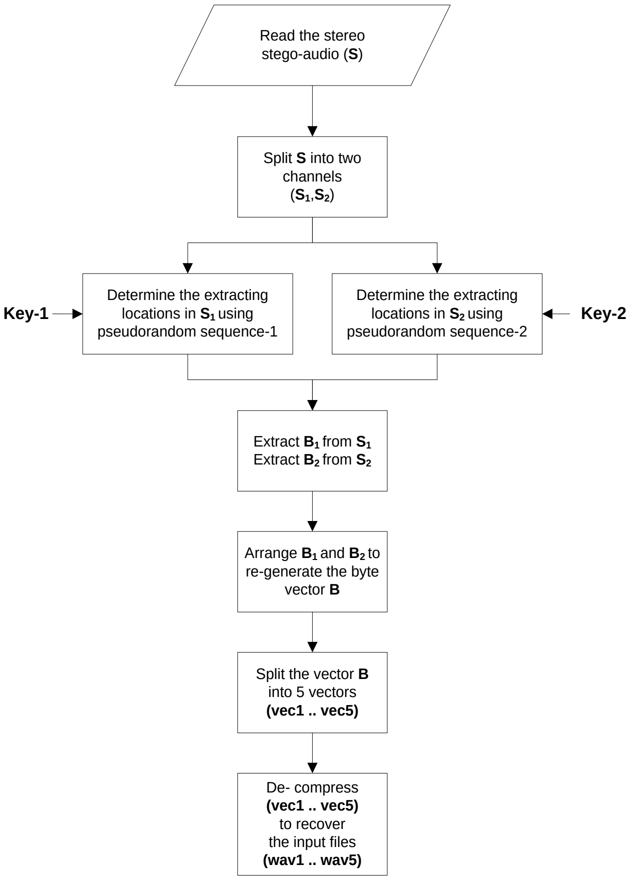
Read the stereo stego-audio (**S**)
Split **S** into two channels (**S$_1$,S$_2$**)
Key-1
Determine the extracting locations in **S$_1$** using pseudorandom sequence-1
Determine the extracting locations in **S$_2$** using pseudorandom sequence-2
Key-2
Extract **B$_1$** from **S$_1$**
Extract **B$_2$** from **S$_2$**
Arrange **B$_1$** and **B$_2$** to re-generate the byte vector **B**
Split the vector **B** into 5 vectors **(vec1 .. vec5)**
De- compress **(vec1 .. vec5)** to recover the input files **(wav1 .. wav5)**

Figure 2: High level data flow diagram for the 5:1 multi-channel audio extraction process.

purpose of self-authenticating audio signals [33] modified to embed up to four ‘information packets’ subject to minimal perceptual distortion. The information capacity of these packets is low compared to the method proposed in this paper. However, unlike the technique reported here, the method given in [32] is robust to various attacks subject to the carrier frequency of the chirp where a low frequency sweep provides greater robustness when compared to a high frequency sweep. In this case, the watermark sequence is derived from sub-band energies which are unique and signal dependent. Due to the different processes associated with information extraction, an additional advantage of self-authentication is achieved, thereby making this multi-level watermarking scheme simultaneously robust and fragile (to an attack). By comparison, the method proposed in this paper, is not taken to be robust but relatively fragile. In other words, the watermarked WAV file can not be subjected to any form of lossy compression or other processes which involves data degradation although lossless encryption is possible if required. The method reported in [32] is designed for signal authentication using limited information embedding and focuses on issues associated with Digital Right Management and Copyright Protection. The approach considered in this paper focuses on attempting to pack many audio channels into a single audio signal with minimal loss of information while sustaining audio fidelity of the host signal.

Referring to Figure 1, the input audio channels are taken to be MP3 files which may by loosely compressed WAV files. The MP3 data vectors are concatenated to form a single vector expressed in bytes. This vector is split into two component vectors composed of 4-bit entries. The host WAV file is decomposed into two component vectors representing the left and right channels of the stereo data stream. An additional step can then be considered whereby each channel is randomized in terms of the embedding locations for the input data as illustrated in Figure 1. A further security feature is possible by encrypting the input data vector before embedding is undertaken. This extra step can be executed by using a pseudo-random or pseudo-chaotic cipher which is either converted to binary form or output in binary form based on the methods discussed in [34], for example.

The information hiding process is accomplished in the time domain by embedding the 2-part input vector into the 2-component host vector using 4-bit LSB in both cases. The stereo stego-audio signal is then constructed by combining the two component output into a single WAV file.

The algorithm for reconstruction of the embedded data is illustrated in Figure 2. After reading the stereo stego-audio WAV file, the data is split into two channels and, as required, the keys used to obtain the position of the embedded data in the channels. Reconstruction of the embedded data is then undertaken by extracting the 4-bit LSBs and arranging the data to regenerate the original (byte) vector. This vector is then split into the five original channels and decompressed as required, thereby recovering the original WAV files.

Prototype m-code for executing these processes is given in Appendix A and Appendix B which provide software solutions for the embedding and extraction of data, respectively. In this example, embedding position randomization and encryption of the input data is not included. It is assumed that users have access to WAV-MP3 conversion software (e.g. [35]) and an encryption application that encrypts MP3 files to a file of the same type, i.e. MP3 extension.

4. SUMMARY

Multi-channel information hiding has a range of applications but the method presented in this paper is focused on problems associated with audio post-production, in particular, the problem of embedding the audio signals used to produced a stereo mix in that same mix. Clearly, the current solution is limited in that it only provides a 5:1 embedding scenario and relies on the input WAV files being loosely MP3 compressed. However, by double, triple and quadruple sampling the host file (by ‘sinc interpolation’, for example), it is clearly possible to embed 10, 15 and 20 channels into the mix using the proposed approach by concatenation on a 5-by-5 multi-channel basis.

From an information theoretic point of view, it is clearly not possible to embed many channels into one channel without some loss of information in terms of both the embedded data and the host. However, provided the result does not lead to distortions that are perceptible on an audio basis, such distortions can be made acceptable. In the context of the algorithms presented in this paper, tests have been undertaken using the Perceptual Evaluation of Audio Quality (PEAQ ITU-R recommendation BS.1387) discussed in [31] following those reported in [32].

Appendix A: m-Code for Multi-Channel Information Hiding in a WAV file

The following code has been written to accommodate a two-column format and consequently uses the MATLAB continuation syntax ‘...’. The code is somewhat condensed to minimize space. For simplicity, the data is not encrypted and the embedding positions are not randomized.

```
% Program to watermark one host WAV file
% with 5 MP3 files. The basic steps are:
% (i) Read 5 mp3 files; (ii) combine the
% files into a single vector; (iii) embed
% the vector into a host WAV file.
clear; clc;
fprintf(’Combining 5 MP3 files...’);
% Combine 5 MP3 files into 1 binary file
% Read 5 name MP3 files as binary files
% of type uint8
filename1 = ’filename_1.mp3’;
filename2 = ’filename_2.mp3’;
filename3 = ’filename_3.mp3’;
filename4 = ’filename_4.mp3’;
filename5 = ’filename_5.mp3’;
fid1 = fopen(filename1,’r’);
Signal_1 = fread(fid1,’uint8=>uint8’);
fclose(fid1); fid2 = fopen(filename2,’r’);
Signal_2 = fread(fid2,’uint8=>uint8’);
fclose(fid2); fid3 = fopen(filename3,’r’);
Signal_3 = fread(fid3,’uint8=>uint8’);
fclose(fid3); fid4 = fopen(filename4,’r’);
Signal_4 = fread(fid4,’uint8=>uint8’);
fclose(fid4); fid5 = fopen(filename5,’r’);
Signal_5 = fread(fid5,’uint8=>uint8’);
fclose(fid5);
% Combine 5 MP3 files into a single
% vector: the array "Signal_5MP3"
Len_1 = length(Signal_1);
Len_2 = length(Signal_2);
Len_3 = length(Signal_3);
Len_4 = length(Signal_4);
Len_5 = length(Signal_5);
Signal_5MP3 = uint8(zeros(Len_1+Len_2+...
Len_3+Len_4+Len_5,1));
Signal_5MP3(1 : Len_1,1) = Signal_1;
Signal_5MP3(Len_1+1 : Len_1 + Len_2,1) =...
Signal_2;
Signal_5MP3(Len_1+Len_2 + 1 : Len_1+Len_2...
            + Len_3,1) = Signal_3;
Signal_5MP3(Len_1+Len_2+Len_3 + 1 : Len_1+...
           Len_2+Len_3 + Len_4,1) = Signal_4;
Signal_5MP3(Len_1+Len_2+Len_3+Len_4 + 1 :...
Len_1+Len_2+Len_3+Len_4 + Len_5,1) = Signal_5;
% Save "Signal_5MP3" as a binary file
fid = fopen(’Signal_5MP3_file.bin’,’w’);
fwrite(fid,Signal_5MP3,’uint8’); fclose(fid);
fprintf(’(DONE)\n’);
fprintf(’Embedding the 5 MP3 files...’);
% Embed 5 MP3 files into 1 host WAV file
[Signal_host,SamplingRate_Host,...
   NumofBits_Host]= wavread(’host.wav’);
fid = fopen(’Signal_5MP3_file.bin’ , ’r’);
Signal_Secret = fread(fid,’uint8=>uint8’);
fclose(fid); Signal_embed = Signal_Secret;
RemSamples = length(Signal_host)...
           -length(Signal_Secret);
if ( RemSamples >= 0)
    Len_5mp3 = length(Signal_Secret);
% Begin the Embedding Process
Signal_stego = Signal_host;
Signal_stego = ((Signal_stego+1)./2).*...
65535; % quantize to [0,65535]
% Embedding the signal
for i = 1:length(Signal_embed)
bit4_1 = bitand( uint8(Signal_embed(i)),...
         uint8(15) );
bit4_2 = bitand( bitshift( uint8(...
       Signal_embed(i)),-4 ) , uint8(15) );
% Embed the first 4-bit signal
Signal_stego(i,1) = bitand(uint16(...
         Signal_stego(i,1)),uint16(65520));
Signal_stego(i,1) = bitor(uint16(...
        Signal_stego(i,1)),uint16(bit4_1));
```

```
% Embed the second 4-bit signal
Signal_stego(i,2) = bitand(uint16(...
         Signal_stego(i,2)),uint16(65520));
Signal_stego(i,2) = bitor(uint16(...
        Signal_stego(i,2)),uint16(bit4_2));
end
Signal_stego = ((Signal_stego./65535) .*2)...
    - 1; % quantize to [-1,1]
% Write the stego signal into a WAV file
wavwrite(Signal_stego,44100,16,...
         'Stego_5MP3.wav');
else
fprintf('Process failed...\n');
fprintf('Secret Signal length exceeds the\n');
fprintf('Host Signal length\n');
end % related to the statement
    % "if ( RemSamples >= 0)" given above
fprintf('(DONE)\n');
fprintf('Embedding Process Completed.\n\n');
% Write length of MP3 files to TXT file.
Length_Data=[Len_1,Len_2,Len_3,Len_4,Len_5];
fid = fopen('MP3_File_Lengths.txt','w');
fwrite(fid,Length_Data,'long'); fclose(fid);
```

Appendix B: m-Code for Multi-Channel Extraction from a Stego-Audio WAV file

```
% Program to extract 5 MP3 from host WAV file
% The basic steps are as follows:
% 1. Extract a vector from the host WAV file.
% 2. Separate the 5 MP3 files from the vector.
clear; clc;
fprintf('Extracting 5 MP3 files...');
% Extract 5 MP3 files from the host WAV file.
[Signal_stego] = wavread('Stego_5MP3.wav');
Signal_stego = ((Signal_stego+1)./2).*65535;
% Quantize host signal to range [0,65535].
Len_5MP3 = length(Signal_stego);
Signal_extract = zeros(Len_5MP3,1);
% Extract the first signal.
for i = 1:Len_5MP3
    % Extract the first 4-bit signal
    s4_1_ext = bitand(uint16(...
               Signal_stego(i,1)),uint16(15));
    % Extract the second 4-bit signal
    s4_2_ext = bitand(uint16(...
               Signal_stego(i,2)),uint16(15));
    % Combine the two 4-bit signals to
    % generate the extracted 8-bit signal.
    Signal_extract(i,1) = bitor(...
                 uint8(0) , uint8(s4_2_ext) );
    Signal_extract(i,1) = bitor(...
                 bitshift(uint8(...
                 Signal_extract(i,1)),4),...
                 uint8(s4_1_ext) );
end
fprintf('(DONE)\n');
fprintf('Extraction Process Completed...\n');
fprintf('Seperating the 5 MP3 files...');
% Seperate 5 MP3 files from the extracted
% vector "Signal_extract".
% Read MP3 original file length data
fid = fopen('MP3_File_Lengths.txt','r');
Length_Data=fread(fid,'long'); fclose(fid);
Len_1=Length_Data(1); Len_2=Length_Data(2);
Len_3=Length_Data(3); Len_4=Length_Data(4);
Len_5=Length_Data(5);
Signal_1_sep = Signal_extract(1 : Len_1);
Signal_2_sep = Signal_extract(Len_1...
                    + 1 : Len_1 + Len_2);
Signal_3_sep = Signal_extract(Len_1+Len_2...
                 + 1 : Len_1+Len_2+Len_3);
Signal_4_sep = Signal_extract(Len_1+Len_2+...
      Len_3 + 1 : Len_1+Len_2+Len_3 + Len_4);
Signal_5_sep = Signal_extract(Len_1+Len_2+...
    Len_3 + Len_4 + 1 : Len_1+Len_2+Len_3+...
      Len_4 + Len_5);
fprintf('(DONE)\n'); % Write MP3 data to file.
fid1=fopen('extracted_filename_1.mp3','w');
fwrite(fid1,Signal_1_sep,'uint8');fclose(fid1);
fid2=fopen('extracted_filename_2.mp3','w');
fwrite(fid2,Signal_2_sep,'uint8');fclose(fid2);
fid3=fopen('extracted_filename_3.mp3','w');
fwrite(fid3,Signal_3_sep,'uint8');fclose(fid3);
fid4=fopen('extracted_filename_4.mp3','w');
fwrite(fid4,Signal_4_sep,'uint8');fclose(fid4);
fid5=fopen('extracted_filename_5.mp3','w');
fwrite(fid5,Signal_5_sep,'uint8');fclose(fid5);
```

5. REFERENCES

[1] Audio Research Group, Dublin Institute of Technology http://www.audioresearchgroup.com/

[2] Digital Media Centre, Dublin Institute of Technology http://www.dmc.dit.ie/

[3] Tamborine Productions Limited, London http://www.tamborine.co.uk/

[4] J M Blackledge and A Al-Rawi, *Application of Stochastic Diffusion for Hiding High Fidelity Encrypted Images*, ISAST Transaction on Computing and Intelligent Systems, Vol. 3, No. 1, 24-33, 2011.

[5] J M Blackledge and A Al-Rawi, A, *Steganography using Stochastic Diffusion for the Covert Communication of Digital Images*, IANEG International Journal of Applied Mathematics, Vol. 41, Issue: 4, 270 - 298, 2011.

[6] K. Shafi, A. Sankaranarayanan, G. Prashanth and A. Mohan, *A Novel Audio Steganography Scheme using Amplitude Differencing*, Trends in Information Sciences and Computing (TISC), 163-167, 17-19 Dec. 2010.

[7] H. B. Kekre, A. Athawale, B. S. Rao and U. Athawale, *Increasing the Capacity of the Cover Audio Signal by Using Multiple LSBs for Information Hiding*, Emerging Trends in Engineering and Technology (ICETET), 196-201, 19-21 Nov. 2010.

[8] A. Ogihara, H. Murata, M. Iwata, and A. Shiozaki, *Multi-Layer Audio Watermarking Based on Amplitude Modification*, Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, IIH-MSP '09, 68-71, 12-14 Sept. 2009.

[9] W. N. Lie and L. C. Chang, *Robust and High-Quality Time-Domain Audio Watermarking Based on Low-Frequency Amplitude Modification*, IEEE Transactions on Multimedia, Vol. 8, No.1, 46- 59, Feb. 2006.

[10] A. Dutta, A. K. Sen, S. Das, S. Agarwal and A. Nath, *New Data Hiding Algorithm in MATLAB Using Encrypted Secret Messages*, International Conference on Communication Systems and Network Technologies (CSNT), 262-267, 3-5 June 2011.

[11] M. Asad, J. Gilani and A. Khalid, *An Enhanced Least Significant Bit Modification Technique for Audio Steganography*, International Conference on Computer Networks and Information Technology (ICCNIT), 143-147, 11-13 July 2011.

[12] M. Wakiyama, Y. Hidaka and K. Nozaki, *An Audio Steganography by a Low-Bit Coding Method with Wave Files*, Sixth International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 530-533, 15-17 Oct. 2010.

[13] Y. Wang, L. Guo, Y. Wei and C. Wang, *A Steganography Method for AAC Audio Based on Escape Sequences*, International Conference on Multimedia Information Networking and Security (MINES), 841-845, 4-6 Nov. 2010.

[14] J. Wang, R. Healy and J. Timoney, *A Novel Audio Watermarking Algorithm Based on Reduced Singular Value Decomposition*, Sixth International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 143-146, 15-17 Oct. 2010.

[15] L. Trefethen and D. Bau, *Numerical Linear Algebra*, SIAM: Society for Industrial and Applied Mathematics, PA, USA, 1997.

[16] A. C. Sekhar, C. Suneetha, G. NagaLakshmi and B. RaviKumar, *Fast Fourier Transforms and Quadratic Forms for Digital Audio Watermarking*, International Conference on Advances in Recent Technologies in Communication and Computing, ARTCom '09., 449-452, 27-28 Oct. 2009.

[17] M. Narimannejad, S. M. Ahadi, *Watermarking of speech Signals Through Phase Quantization of Sinusoidal Models*, 19th Iranian Conference on Electrical Engineering (ICEE), 1-4, 17-19 May 2011.

[18] B. Chen, G. W. Wornell, *Quantization Index Modulation: A Class of Provably Good Methods for Digital Watermarking and Information Embedding*, IEEE Transactions on Information Theory, Vol.47, No.4, 1423-1443, May 2001.

[19] Y. Lin and W. H. Abdulla, *A Secure and Robust Audio Watermarking Scheme using Multiple Scrambling and Adaptive Synchronization*, 6th International Conference on Information, Communications and Signal Processing, 1-5, 10-13 Dec. 2007.

[20] R. Tachibana, S. Shimizu, and S. Kobayashi, *An Audio Watermarking Method using a Two-dimensional Pseudo-random Array*, Signal Processing, Vol. 82, No. 10, 1455-1469, 2002.

[21] Y. Q. Lin and W. H. Abdulla, *Robust Audio Watermarking Technique Based on a Gamma-tone Filter Bank and Coded Image*, International Symposium on Signal Processing and Its Application (ISSPA'07), 2007.

[22] Y.Q. Lin and W.H. Abdulla, *Robust Audio Watermarking for Copyright Protection*, Technical Report (No. 650), Dept. of Electrical and Computer Engineering, The University of Auckland, 2006.

[23] X. S. Chen, Y. T. Yang, H. Zhang and X. J. Lu, *An Audio Blind Watermarking Algorithm in the Wavelet Domain Based on Chaotic Encryption*, International Conference on Wavelet Analysis and Pattern Recognition, ICWAPR '07. 470-473, 2-4 Nov. 2007.

[24] P. Shah, P. Choudhari and S. Sivaraman, *Adaptive Wavelet Packet Based Audio Steganography using Data History*, IEEE Region 10 and the Third international Conference on Industrial and Information Systems, ICIIS 2008, 1-5, 8-10 Dec. 2008.

[25] A. Delforouzi and M. Pooyan, *Adaptive Digital Audio Steganography Based on the Integer Wavelet Transform*, Third International Conference on Intelligent Information Hiding and Multimedia Signal Processing, IIHMSP 2007, Vol. 2, 283-286, 26-28 Nov. 2007

[26] C. Maha, E. Maher, K Mohamed and B. A. Chokri, *A DCT Based Blind Audio Watermarking Scheme*, Proceedings of the 2010 International Conference on Signal Processing and Multimedia Applications (SIGMAP), 139-144, 26-28 July 2010.

[27] W. Hamming, *Error Detecting and Error Correcting Codes*, Bell Systems Technical Journal 26(2), 137-160, 1950.

[28] H. Y. Yang, X. Y. Wang and T. X. Ma, *Robust Digital Audio Watermarking using Higher-order Statistics*, AEU - International Journal of Electronics and Communications, Vol. 65, Issue 6, 560-568, June 2011.

[29] B. Y. Lei, I. Y. Soon and Z. Li, *Blind and Robust Audio Watermarking Schemes Based on SVD-DCT*, Original Research Article: Signal Processing, Vol. 91, Issue 8, 1973-1984, August 2011.

[30] F. Han, X. Yu and S. Han, *Improved Baker Map for Image Encryption*, Proceedings of the First International Symposium on Systems and Control in Aerospace and Astronautics, ISSCAA 2006, 1273-1276, 2006.

[31] P. Kabal, *An Examination and Interpretation of ITU-R BS.1387: Perceptual Evaluation of Audio Quality*, Technical Report, McGill University, Version 2, 2003.

[32] J. M. Blackledge and O. Farooq, *Audio Data Verification and Authentication using Frequency Modulation based Watermarking*, International Society for Advanced Science and Technology, Journal of Electronics and Signal Processing, Vol. 3, No 2, (ISSN 1797-2329), 51 - 63, 2008.

[33] J. M. Blackledge and E. Coyle, *Self-Authentication of Audio Signals by Chirp Coding*, Proceedings of the 12th International Conference on Digital Audio Effects (DAFx-09), Como Italy, 2009.

[34] J. M. Blackledge, *Cryptography and Steganography: New Algorithms and Applications*, (Ed. Stanislaw Janeczko), Centre for Advanced Studies, Textbook Series 1, Warsaw University of Technology, Poland ISBN: 978-83-61993-05-6, 2012. http://eleceng.dit.ie/papers/195.pdf

[35] WAV MP3 Converter. http://www.wav-mp3.com