Journal of Computing and Security

April 2015, Volume 2, Number 2 (pp. 119–127)

http://www.jcomsec.org

A High-Payload Audio Watermarking Scheme for Real-Time Applications

Ramin Almasi [a], Hooman Nikmehr [a,*]

[a] *Department of Computer Architecture, Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran.*

ARTICLE INFO.

Article history:
Received: 23 January 2015
Revised: 12 January 2016
Accepted: 21 February 2016
Published Online: 08 May 2016

Keywords:
Real-Time Audio Watermarking, Payload, Robustness, Transparency, Signal Processing Attacks

ABSTRACT

Digital audio watermarking is mainly used for ownership protection of digital audio. In this research, an audio watermarking method is presented which can be adopted in real-time situations due to its high speed in both embedding and detecting stages. This approach has a higher payload in comparison with similar recent approaches. In the proposed algorithm, which uses a time-domain synchronization, to embed the watermark, the audio signal is first split into a number of blocks. Then the watermark is embedded into the FFT components of audio signal with greater magnitudes. Unlike common audio watermarking methods that use the 1-bit embedding approach, in the proposed scheme, a 2-bit embedding approach is presented which doubles the embedding payload compared to a number of similar works. The evaluation results on different audio signals demonstrate that the new scheme is faster and more transparent compared to its counterparts. The presented watermarking is found robust enough against the common attacks like additive noise, MP3 compression, low pass filtering, re-sampling and re-quantization. Using FPGA implementation/evaluation, the time complexities of the major parts of the proposed algorithm are reported.

© 2015 JComSec. All rights reserved.

1 Introduction

Nowadays, broadband Internet connections and nearly error-free transmission of data facilitate the distribution of multimedia files. Since 1992, the number of published articles on digital watermarking has drastically increased. Although these algorithms primarily were developed for still images and video streams, interest and research on audio watermarking commenced later.

In a sense, audio watermarking methods can be categorized into two classes; "time domain and "frequency domain. In both classes, the main issues are transparency, payload, reliability and robustness. However, recent real-time applications of audio watermarking have forced the designers to take the (hardware/software) implementation complexity into consideration as well.

Using digital instruments in live concerts and Desktop Music (DTM) has become a common practice today. Unfortunately these artistic productions are distributed illegally through the Internet and played via digital devices. This illegal duplication and distribution causes great financial losses due to infringing the intellectual property rights of the producers and performers. This is why real-time audio watermarking has recently become an attractive technology promis-

* Corresponding author.
Email addresses: raminalmasi@eng.ui.ac.ir (R. Almasi), nikmehr@eng.ui.ac.ir (H. Nikmehr)
ISSN: 2322-4460 © 2015 JComSec. All rights reserved.

ing to solve this problem [1]. In other real-time applications such as IP telephony, where security is a critical issue, audio watermarking can help to perform secure identification/authentication and authorization, and make sure that the confidentiality, integrity and non-denial/non-repudiation of the dialog are met [2]. Application of real-time audio watermarking can be easily extended to other aspects of the daily life such as telephone banking and health care systems where multimedia integrity and authenticity is very important to the success of the services. In order to use audio watermarking in a wider range of applications, the payload should be increased and at the same time the other aspects of the algorithm should be kept intact as much as possible. According to IFPI [3], any digital watermarking algorithm needs to have more than 20 bps data payload for watermark. Obviously, watermarking techniques with higher payloads potentially lead to more complicated watermarks that are very unlikely to be removed or altered by various attacks.

In [4] a robust and high quality audio watermarking is proposed in time domain that modifies the amplitude in low frequencies. It first divides the audio signal into a group of samples. Then, using differential average of absolute amplitude in a group of samples it can embed the watermark into the host signal. This method periodically recalculates the thresholds based on Human Auditory System (HAS) and the audio signal's properties. This dynamic threshold adjustment increases the algorithm's computing complexity, hence making it inappropriate for real-time applications. Researchers in [5] propose a real-time audio watermarking technique using wavetables. In this method, Pulse Code Modulation (PCM) waveforms are held in wavetables. The results presented in the paper indicate that this algorithm is faster and more transparent compared to other algorithms. In addition, the algorithm is reported to be reasonably robust against attacks such as additive noise, MP3 compression and low-pass filtering. However, the authors do not include the payload of their audio watermarking scheme in the manuscript. In [6] a blind audio watermarking method is proposed using self-synchronization approach, where the synchronization bits are embedded into the time domain by modifying the audio signal's time samples. This technique embeds the watermark bits into the signal's Discrete Cosine Transform (DCT) components. The simulation results shown in the paper reveal that the proposed method cannot be considered robust enough against the conventional attacks. The method presented in [7] embeds the synchronization and watermark bits into the low frequency components of the wavelet transform. Although this method is more robust against the common attacks, its Signal-to-Noise Ratio (SNR) is not high enough producing audible noises in the watermarked output signal. The authors in [8] propose a fast audio watermarking scheme where the synchronization and the watermark bits are embedded into the time and the FFT domains, respectively. This method seems to be robust against the attacks and the signal processing operations. Since the synchronization embedding process is carried out in the time domain, the proposed watermarking is suitable for real-time applications; however it has the payload of 30bps that is not enough in comparison with similar audio watermarking methods. In [9], the synchronization bits are embedded into the low frequency components of the signal's Discrete Wavelet Transform (DWT). To embed the watermark bits, first, DWT transform is applied to the audio signal and then DCT transform is applied to the low frequency components resulted from the preceding transform. Then the watermark bits are embedded into the DCT components. Since synchronization bits are embedded into the DWT domain, this algorithm look robust enough against the common attacks. However, detecting the synchronization bits in this algorithm is a slow process making it inappropriate for real-time applications. In a recent watermarking approach [10], the authors present a blind and robust audio watermarking against common signal processing attacks. Although the payload and the transparency reported in the article is impressive, using the DCT and the SVD transforms makes the design inappropriate for real-time applications. Another new watermarking technique is developed by Xiang et al in [11]. The algorithm looks robust with improved perceptual quality; however, the authors use the outdated measure SNR instead of Objective Difference Grade (ODG) to evaluate the transparency. The article also lacks a quantitative evaluation of the payload of the proposed watermarking. One of the latest high-payload audio watermarking algorithms [12] which is reported robust against both general signal processing and desynchronization attacks, is too slow (due to using complex TLW and SVD transforms) for real-time environments.

In this article, a time domain synchronization algorithm is adopted which is suitable for real-time applications. To embed the watermark, the audio signal is first split into several blocks. Then the watermark bits are embedded into the FFT components with the largest magnitudes. This way, without significant attenuation in SNR, ODG and Bit Error Rate (BER), the payload of the watermark embedded into the host signal increases twofold compared to similar counterparts.

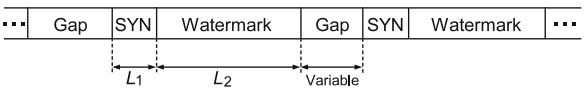

Figure 1. Embedding Segments [8]

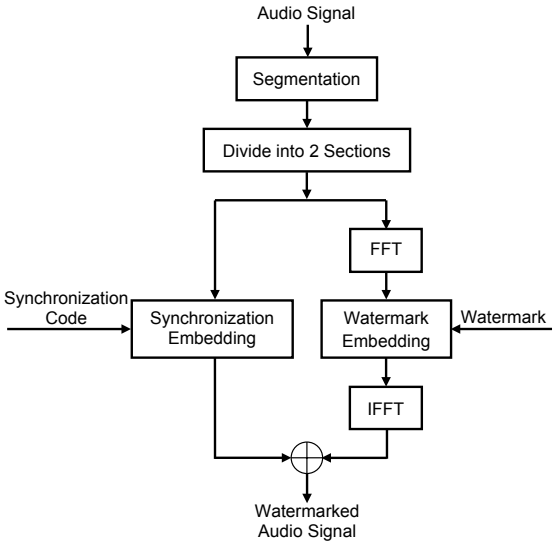

Figure 2. Embedding Scheme

2 Proposed Audio Watermarking

The presented audio watermarking algorithm is based on the scheme introduced in [8]. Since an audio signal contains great amount of data itself, if any watermarking approach wants to embed the watermark bits into the audio signal all at once, a vast (and mostly impractical) volume of processing memory and functional units are required. That is why in the proposed watermarking algorithm, the audio signal is divided into small blocks and then the watermark bits are inserted into these blocks. As shown in Figure 1, this approach embeds the synchronization bits at the beginning of the watermark bit pattern in order to make sure that the position of the watermark can be detected. In the proposed algorithm, as shown in Figure 2, the synchronization bits are embedded into the time domain samples and the watermark bits into the FFT components of the host signal.

2.1 Embedding Synchronization Bits

In the proposed audio watermarking approach, the audio samples are assumed to be normalized in interval $[-1, 1]$. If each sample is represented using n bits, then the signal can be normalized in the interval by dividing each sample by 2^{n-1}. As illustrated in Figure 1, the synchronization code is imbedded into L_1 samples of the host audio signal. If n_{syn} samples are used for embedding each synchronization bit, in order to embed an l-bit synchronization code, $L_1 = l \cdot n_{\text{syn}}$ samples are required. The algorithm

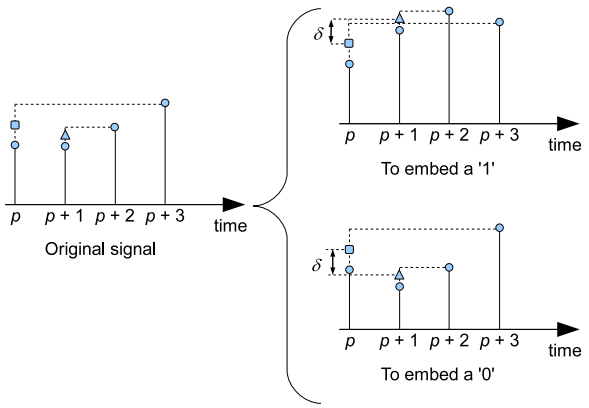

Figure 3. Embedding sichronization bits [8]

for embedding the synchronization bits is as follows. To embed synchronization bit "1" into n_{syn} samples of the host signal, the inner samples are increased so that their average becomes greater than that of the extreme points (the first and the last samples). In order to embed bit "0", the inner samples are decreased so that their average becomes smaller than that of the extreme points (Figure 3). It is assumed that the synchronization bits are intended to be embedded into the time samples s_p, s_{p+1}, $\cdots$, $s_{p+n_{\text{syn}}-1}$ of the host audio signal. a number of notations are listed here to make the readability of the paper easier:

s	signal sampled in time domain
s_p	p-th sample of host signal
$\overline{s}_{\text{int}}$	average of inner samples
$\overline{s}_{\text{ext}}$	average of extreme samples
δ_{min}	minimum distance from $\overline{s}_{\text{ext}}$
φ	distortion introduced with respect to $\overline{s}_{\text{ext}}$
n_{syn}	number of consecutive samples for embedding a synchronization bit

Now, the proposed embedding method can be expressed as Algorithm 1 [8]

In this method, the 16-bit famous Barker code "1111100110101110 is used as the synchronization code and the embedding parameters are selected as $\delta_{min} = 10^{-3}$, $\varphi = 0.05$ and $n_{\text{syn}} = 4$ [8] .

2.2 Embedding Watermark

Unlike most conventional audio watermarking methods which use the 1-bit embedding approach, in the proposed watermark embedding scheme, a novel 2-bit embedding algorithm is proposed that embeds 2-bit string "00, "01, "10 or "11 instead of bit "0 or "1. If the watermark consists of L_2 samples, as shown in Figure 1, and if each watermark bit is embedded in L_3 samples of the host audio signal, then the watermark

Algorithm 1 Embedding Synchronization Bits

$$\overline{s}_{\text{int}} := \frac{1}{n_{\text{syn}}-2} \sum_{j=p+1}^{p+n_{\text{syn}}-2} s_j$$

$$\overline{s}_{\text{ext}} := \frac{s_p + s_{p+n_{\text{syn}}-1}}{2}$$

For $j := p$ **to** $p + n_{\text{syn}} - 1$ **do**

$s'_j := s_j$

EndFor

- To embed a "1":

  If $\overline{s}_{\text{int}} < \overline{s}_{\text{ext}} + \delta$ **then**

  $d := \overline{s}_{\text{ext}} + \delta - \overline{s}_{\text{int}}$

  For $j := p + 1$ **to** $p + n_{\text{syn}} - 2$ **do**

  $s'_j := s_j + d$

  EndFor

  EndIf

- To embed a "0":

  If $\overline{s}_{\text{ext}} < \overline{s}_{\text{int}} + \delta$ **then**

  $d := \overline{s}_{\text{int}} + \delta - \overline{s}_{\text{ext}}$

  For $j := p + 1$ **to** $p + n_{\text{syn}} - 2$ **do**

  $s'_j := s_j - d$

  EndFor

  EndIf

length L_2 contains $|w| \cdot L_3$ samples where $|w|$ is the watermark length in bits and $L_3 = 512$ [8] . The host signal is assumed a mono audio signal; however, this algorithm works for stereo audio signals as well. In this method to embed each watermark bit into the FFT coefficients, a block of L_3 coefficient samples is required. To obtain these, the FFT transform is applied on blocks of L_3 samples each. In this paper, s and S represent the audio signal sampled in the time and in the FFT domain respectively. Now, a set of suitable frequencies of S should be chosen for embedding two bits of the watermark. Since the FFT function is symmetric, $S_{L_3-k} = S_k$ for $k = 0, 1, 2, \cdots, \frac{L_3}{2} - 1$. This means that the second half of the FFT coefficients sequence (S_{L_3-k}) equals the complex-conjugate of the first half of the sequence $(S^*_{L_3-k})$; hence the second half of S does not need to be calculated and is discarded by the watermark embedding process. In addition, S_0 is not suitable for embedding the watermark bits and therefore is ignored [8]. To continue the process, the elements of sequence $S_1, S_2, \cdots, S_{\frac{L_3}{2}-1}$ are sorted in ascending order to construct the sequence $S'_1, S'_2, \cdots, S'_{\frac{L_3}{2}-1}$ such that the new elements satisfy

$$|S'_1| \le |S'_2| \le |S'_3| \le \cdots \le |S'_{\frac{L_3}{2}-1}|$$

where $|S'_i|$ is the magnitude of S'_i. In order to embed two watermark bits, five consecutive elements $S'_m, S'_{m+1}, S'_{m+2}, S'_{m+3}, S'_{m+4}$ with $1 < m < \frac{L_3}{2} - 4$ are chosen. Symbol m represents the starting position where the watermark bits are embedded in the FFT coefficients. In the proposed approach, value of the embedding position m provides the tradeoff between transparency and robustness of the audio watermarking algorithm such that larger (smaller) m leads to less (more) transparency and more (less) robustness. In the presented experiments $m = 250$ is selected. Having represented $|S'_k|$ by symbol M_k (for simplicity), to establish the embedding criteria, thresholds $A = M_{m+4} - M_{m+3}$, $B = M_{m+3} - M_{m+2}$, $C = M_{m+2} - M_{m+1}$ and $D = M_{m+1} - M_m$ are defined and in order to embed bit strings "00", "01", "10" and "11", the following criteria are followed:

If $(B \ge \alpha A$ **AND** $D \ge \beta C)$ **then embed** "00"

If $(A \ge \alpha B$ **AND** $D \ge \beta C)$ **then embed** "01"

If $(B \ge \alpha A$ **AND** $C \ge \beta D)$ **then embed** "10"

If $(A \ge \alpha B$ **AND** $C \ge \beta D)$ **then embed** "11"

where α and β are coefficients defined to avoid possible overlaps between the embedding criteria. In case any of the above conditions is not satisfied, spectrum S' needs to be modified as follows

- To embed "00",

  $S'_{m+3}[\text{new}] = \frac{M_{m+2}+\alpha M_{m+4}}{(\alpha+1)M_{m+3}} S'_{m+3}$

  $S'_{m+1}[\text{new}] = \frac{M_{m}+\beta M_{m+2}}{(\beta+1)M_{m+1}} S'_{m+1}$

- To embed "01",

  $S'_{m+3}[\text{new}] = \frac{M_{m+4}+\alpha M_{m+2}}{(\alpha+1)M_{m+3}} S'_{m+3}$

  $S'_{m+1}[\text{new}] = \frac{M_{m}+\beta M_{m+2}}{(\beta+1)M_{m+1}} S'_{m+1}$

- To embed "10",

  $S'_{m+3}[\text{new}] = \frac{M_{m+2}+\alpha M_{m+4}}{(\alpha+1)M_{m+3}} S'_{m+3}$

  $S'_{m+1}[\text{new}] = \frac{M_{m+2}+\beta M_{m}}{(\beta+1)M_{m+1}} S'_{m+1}$

- To embed "11",

  $S'_{m+3}[\text{new}] = \frac{M_{m+4}+\alpha M_{m+2}}{(\alpha+1)M_{m+3}} S'_{m+3}$

  $S'_{m+1}[\text{new}] = \frac{M_{m+2}+\beta M_{m}}{(\beta+1)M_{m+1}} S'_{m+1}$

In each case, these changes should be applied to the corresponding S_k and its complex-conjugate. However, it can be easily shown mathematically that unlike the algorithm presented in [8], new S'_{m+1} and S'_{m+3} still conform to condition $|S'_m| \le |S'_{m+1}| \le |S'_{m+2}| \le |S'_{m+3}| \le |S'_{m+4}|$. As a result, in terms of the computation load, the proposed algorithm shows a significant advantage over the design presented in [8].

2.3 Synchronization Detection

Like most audio watermarking algorithms, in the proposed scheme, existence of a secret key guarantees the security of the method. The secret key includes information such as the lengths of the watermark and the synchronization parts in bit, number of the required samples for the watermark and the synchronization bit patterns, and m (watermark bits embedding position). In the proposed algorithm, to extract the watermark bits, first the exact location of the synchronization bits are found in the watermarked signals and then the watermark bits are extracted from L_2 samples after the synchronization bit pattern. If the watermark length is not fixed, then in the embedding process, bit pattern "01111110 is attached to the beginning and the end of the watermark to boost the confidence of the watermark detection and the extraction process [8].

To detect each bit (either "0" or "1") of the synchronization pattern with l bits, the following procedure needs to be followed. In this procedure, for every bit, n_{syn} samples of the watermarked signal are processed. To perform this, a sample window with the size of $L_1 = l \cdot n_{\text{syn}}$ is chosen and from the beginning of the watermarked signal in the time domain, the samples inside the window are divided into n_{syn} consecutive non-overlapped sets. Having defined $t_p, t_{p+1}, t_{p+2}, \cdots, t_{p+n_{\text{syn}}-1}$ to represent n_{syn} consecutive samples of the watermarked signal in the time domain, the synchronization detection process for each bit continues as Algorithm 2 [8]

Algorithm 2 Synchronization Detection

$$\bar{t}_{\text{int}} := \frac{1}{n_{\text{syn}}-2} \sum_{j=p+1}^{p+n_{\text{syn}}-2} t_j$$

$$\bar{t}_{\text{ext}} := \frac{t_p + t_{p+n_{\text{syn}}-1}}{2}$$

If $\bar{t}_{\text{int}} > \bar{t}_{\text{ext}}$ **then** "1" is detected
else "0" is detected
EndIf

In this procedure, $\bar{t}_{\text{int}}$ represents the average of the inner samples, whereas $\bar{t}_{\text{ext}}$ shows the average of the extreme points. The process reveals that if the average of the inner points is more than the average of the extreme points, the bit is detected as "1", otherwise as "0". This operation needs to repeat l times to generate an l-bit synchronization candidate for the sample window, which is then compared to the synchronization bit string in the secret-key. If they match, it means that the detection procedure has found the exact location of the synchronization pattern; otherwise the algorithm shifts the window one sample and repeats until the end of the signal. Since every two consecutive 16-bit windows share 15 synchronization bits, the synchronization detection rate can be considerably accelerated through a recursive technique as follows. First, n_{syn} binary strings are defined and indexed from 1 to n_{syn}. Then for each window shift, from the 1st to the (n_{syn})-th, the generated synchronization bit pattern is stored in the corresponding binary string. Now for any shift with the number beyond (n_{syn})-th, the algorithm assigns the bit pattern found in the string with the index equal to the remainder of dividing the shift number by n_{syn}. Finally, the bit pattern is shifted one bit to the left and the bit detected from n_{syn} samples is used as the string's least significant bit (LSB).

2.4 Watermark Extraction

Once the synchronization bit string is detected, the watermark extraction process is carried out on $L_2 = |w|L_3$ samples of the watermarked signal starting right after the samples used for synchronization extraction. For this purpose, first the FFT is applied to these L_2 samples while the second half of the coefficients generated is ignored (see Section 2.2). Then the remaining coefficients are sorted in ascending order of magnitude, and a series of 2-bit strings are extracted as follows. Having defined M'_k with $k = 1, 2, \cdots, \frac{L_3}{2} - 1$ as the magnitudes of the first half of the FFT coefficients, thresholds $A' = M'_{m+4} - M'_{m+3}$, $B' = M'_{m+3} - M'_{m+2}$, $C' = M'_{m+2} - M'_{m+1}$ and $D' = M'_{m+1} - M'_m$ are defined and the following watermark extraction scheme is used:

If $(A' \leq B'$ **AND** $C' \leq D')$ **then extract** "00"
If $(A' > B'$ **AND** $C' \leq D')$ **then extract** "01"
If $(A' \leq B'$ **AND** $C' > D')$ **then extract** "10"
If $(A' > B'$ **AND** $C' > D')$ **then extract** "11".

3 Performance of Proposed Algorithm

In this article, the newly proposed real-time audio watermarking method is evaluated for payload capacity in terms of bit number (the maximum number of watermark bits embedded into one second of the host signal), robustness against the conventional attacks in terms of BER, transparency in terms of SNR and ODG and speed in terms of samples per second.

As mentioned in Section 2.2, to embed the watermark at the beginning and at the end of the frame, delimiters need to be attached to the watermark bit pattern. Next, a Reed-Solomon ECC has to be applied to the watermark. If delimiters and l (length of syn) are both 16 bits long and n_{syn} is set to 4 samples, sam-

Table 1. BER of different schemes after MP3 (128kbps) attack for a variety of audio files

Audio file	Audio watermarking scheme				
	[8]	Proposed $\alpha=4$ $\beta=30$ $m=249$	Proposed $\alpha=3$ $\beta=7$ $m=250$	Proposed $\alpha=3$ $\beta=10$ $m=250$	Proposed $\alpha=4$ $\beta=10$ $m=250$
Floodplain	11.94	11.38	10.1	9.81	9.63
Stop Payment	11.94	15.62	12.77	12.89	11.44
Rust	14.03	18.87	17.13	15.04	16.60
The Firm	36.07	31.99	28.68	28.33	27
Sonati	23.54	30.37	27.52	28.54	29.38

pling rate is chosen 44100 samples per second, and the watermark length after the Reed-Solomon $(255, 249)$ encoding equals 104, then

$$payload = \frac{56-16}{(16\times4+52\times512)\div44100} \cong 66.09$$

As expected, due to embedding two watermark bits in $L_3 = 512$ samples of the audio signal, the payload of the proposed algorithm is double the one reported in [8], where only one watermark bit is embedded into the same number of samples.

In this article, in order to assess the robustness of the proposed method, a number of attacks common in real-time environments such as 128 kbps MP3 (using LAME encoder tool), additive noise (using the StirMark benchmark), RC-low pass filtering (using the StirMark benchmark with the filter cutoff frequency of 10 kHz) and re-sampling (using MATLAB sampled and resampled at 22050 and 44100 samples per second respectively) are applied to the watermarked audio signal. Tables 1, 2, 3, 4 and 5 show BER after MP3, additive noise, low pass filtering, re-sampling and re-quantization attacks respectively with no ECC and for both the proposed method and the algorithm presented in [8]. The tables clearly demonstrate that while the payload of the new scheme is twice that of its counterpart, its robustness is still within the acceptable range for an audio watermarking algorithm.

In the current research, measures SNR and ODG are used to show to what extent the proposed audio watermarking algorithm is transparent. SNR can be calculated as

$$\mathrm{SNR}(A,\tilde{A}) = 10\log_{10}\frac{\sum_{n=1}^{N} a^2(n)}{\sum_{n=1}^{N}(a(n)-\tilde{a}(n))^2} \qquad (1)$$

where A represents N samples of the original audio signal while $\tilde{A}$ is the watermarked version of the same signal. Table 6 which compares SNR of the proposed method (with various α, β and m) with that of [8]

Table 2. BER of different schemes after additive noise attack for a variety of audio files

Audio file	Audio watermarking scheme				
	[8]	Proposed $\alpha=4$ $\beta=30$ $m=249$	Proposed $\alpha=3$ $\beta=7$ $m=250$	Proposed $\alpha=3$ $\beta=10$ $m=250$	Proposed $\alpha=4$ $\beta=10$ $m=250$
Floodplain	1.39	3.49	1.97	2.26	1.8
Stop Payment	8.81	11.9	9.34	8.42	9.11
Rust	14.84	17.07	14.57	13.29	11.73
The Firm	20.99	21.19	19.39	19.22	19.68
Sonati	2.55	5.16	4.00	2.90	3.19

Table 3. BER of different schemes after re-sampling attack for a variety of audio files

Audio file	Audio watermarking scheme				
	[8]	Proposed $\alpha=4$ $\beta=30$ $m=249$	Proposed $\alpha=3$ $\beta=7$ $m=250$	Proposed $\alpha=3$ $\beta=10$ $m=250$	Proposed $\alpha=4$ $\beta=10$ $m=250$
Floodplain	0	0	0	0	0
Stop Payment	0	0	0	0	0
Rust	2.43	5.11	2.72	2.72	2.55
The Firm	3.59	5.34	4.41	4.41	4.29
Sonati	0.11	0.05	0.05	0.05	0.05

Table 4. BER of different schemes after RC-Low pass filter for a variety of audio files

Audio file	Audio watermarking scheme				
	[8]	Proposed $\alpha=4$ $\beta=30$ $m=249$	Proposed $\alpha=3$ $\beta=7$ $m=250$	Proposed $\alpha=3$ $\beta=10$ $m=250$	Proposed $\alpha=4$ $\beta=10$ $m=250$
Floodplain	1.27	4.23	2.20	1.91	1.97
Stop Payment	3.71	7.54	5.11	4.52	4.23
Rust	6.14	11.9	8.3	8.13	7.72
The Firm	6.84	10.97	10.22	9.93	10.10
Sonati	2.08	6.79	5.69	5.63	4.76

indicates that in spite of twice the payload capacity, new SNR is still within the acceptable range (greater than 20 dB [7]) for different input audio signals. The audio files are selected from different music genres such as Rock, Pop and Classic all 10 seconds long.

To extend the evaluation space, SNR and payload

Table 5. BER of different schemes after Re-quantization attack for a variety of audio files

Audio file	Audio watermarking scheme				
	[8]	Proposed $\alpha = 4$ $\beta = 30$ $m = 249$	Proposed $\alpha = 3$ $\beta = 7$ $m = 250$	Proposed $\alpha = 3$ $\beta = 10$ $m = 250$	Proposed $\alpha = 4$ $\beta = 10$ $m = 250$
Floodplain	0.003	0.007	0.006	0.005	0.005
Stop Payment	0.019	0.035	0.029	0.026	0.029
Rust	0.081	0.081	0.081	0.081	0.081
The Firm	0.13	0.16	0.14	0.14	0.14
Sonati	0.001	0.007	0.003	0.002	0.002

Table 6. SNR of different schemes for varieties of parameters and audio files

Audio file	Audio watermarking scheme				
	[8]	Proposed $\alpha = 4$ $\beta = 30$ $m = 249$	Proposed $\alpha = 3$ $\beta = 7$ $m = 250$	Proposed $\alpha = 3$ $\beta = 10$ $m = 250$	Proposed $\alpha = 4$ $\beta = 10$ $m = 250$
Floodplain	24.75	24.63	21.45	21.33	20.80
Stop Payment	22.88	22.53	21.21	21.09	20.72
Rust	26.55	25.15	22.42	22.32	21.81
The Firm	26.12	25.73	24.65	24.52	23.95
Sonati	21.84	23.39	23.13	23.11	23.05

Table 7. SNR and capacity of different schemes

AW scheme	Content	Sync	SNR (dB)	Capacity (bit)
[13]	Short Clip	Yes	43.1	NA
[14]	Song	No	25	43
[8]	Song+Quartet+SQAM	Yes	25.7	33.09
[10]	Song+Quartet+SQAM	Yes	30	43
Proposed $\alpha = 4$ $\beta = 30$ $m = 249$	Song+Quartet+SQAM	Yes	24.08	66.09

capacity of the proposed scheme along with those of four different methods previously published in the literature are listed in Table 7. The table again shows that the payload capacity is nearly doubled while new SNR is still large enough (above 20 dB).

In this study, ODG is computed using Perceptual Evaluation of Audio Quality (PEAQ) algorithm specified in ITU BS.1387-1 [15]. ODG can take value 0, -1, -2, -3 and -4 that represent imperceptible, perceptible but not annoying, mildly annoying, annoying and very annoying respectively [16]. ODG of the proposed algorithm is compared with that of [8] and [10] in Table 8. The results indicate while improved in the other aspects, the proposed method is imperceptible in different cases as well.

Table 8. ODG of different schemes for variety of audio files

Audio file	Time (sec)	Watermark Length (bit)	[10]	[8]	Proposed $\alpha = 4$ $\beta = 30$ $m = 249$	Proposed $\alpha = 3$ $\beta = 10$ $m = 250$
Rust	2:33	117	-0.79	-0.1	-0.025	-0.131
Floodplain	3:13	146	-0.77	0.0	-0.027	-0.03
Stop Payment	2:09	99	-0.52	0.0	-0.022	-0.116
The Firm	0:10	8	-0.28	0.0	0.0	0.0

Table 9. Synchronization detection complexity of different schemes

AW scheme	Detection speed (samples per second)	Time spend per sample (second)
[7]	107.13	0.0093
[8]	1010.1	0.00099
[9]	285.7	0.0035
Proposed $\alpha = 4$ $\beta = 30$ $m = 249$	4040.4	0.00025

As mentioned earlier, only high speed audio watermarking algorithms are applicable in real-time situations. The time complexity (speed) of the synchronization detection process, which is the most time-consuming step in any audio watermarking method, is shown in Table 9 in samples per second for the proposed and a number of recent audio watermarking schemes. The speed evaluation is performed using MATLAB 2009 on a desktop computer with MS-Windows operating system, 2.53 GHz processor and 3 GB RAM. The results reveal that the software implementation of the synchronization detection unit of the proposed algorithm (hence roughly the whole audio watermarking process) is at least four times faster than those of the similar algorithms.

For more clarification, the time complexities of the new algorithm and the method presented in [8] are also compared in Table 10 under the same hard-

Table 10. Watermark embedding complexity of different schemes

AW scheme	Embedding speed (bits per second)	Time spend per bit (second)
[8]	338.03	0.003
Proposed $\alpha = 4$ $\beta = 30$ $m = 249$	1171.43	0.00085

Table 11. Speed of synchronization detection unit of the proposed algorithm on different FPGAs

FPGA family	Package	Device	Time spent per sample (nano second)
Spartan 3	PQ208	XC3S50	11.65
Spartan 6	TQG144	XC6SLX4	7.541
Virtex 4	SF363	XC4VFX12	7.07
Virtex7	FFG1925	XC7V2000T	4.37

ware/software condition. It can be easily found from the table that the proposed watermark embedding algorithm is almost three times faster than that of in [8].

To have a more realistic timing estimation, the synchronization code detection unit of the proposed audio watermarking algorithm is described using VHDL and implemented on a variety of FPGA-based hardware platforms and the speeds are presented in Table 11. The reader can easily infer from the delays reported in the table that the proposed algorithm is well-designed for real-time and fast applications.

4 Conclusion

A high payload method for real-time audio watermarking is presented here. The payload capacity of the proposed approach is greater than that of a number of recent similar works in this area; more specifically is double the payload of the closest counterpart introduced in [8]. The computation load is reduced in the presented method compared to what is reported in [8]. Estimations using the available evaluation tools indicate that the proposed algorithm shows a proper robustness against the most common signal processing attacks and in some cases it is more robust compared to similar methods recently published. The scheme presented in this paper has proper SNR and ODG; hence it does not generate audible noise when listening to the host audio signal. Major modules of the proposed audio watermarking algorithm are implemented in hardware and software and evaluated for speed. The result proves that the new approach is fast enough to be appropriate in practical and real-time applications.

References

[1] K Yamamoto and M Iwakiri. Real-time audio watermarking based on characteristics of pcm in digital instrument. *Journal of Information Hiding and Multimedia Signal Processing*, 1(2): 59–71, 2010.

[2] S Yuan and S A Huss. Audio watermarking algorithm for real-time speech integrity and authentication. In *Proceedings of the 2004 workshop on Multimedia and Security*, pages 220–226. ACM, 2004.

[3] S V Dhavale, R S Deodhar, and L M Patnaik. High capacity lossless semi-fragile audio watermarking in the time domain. In *Advances in Computer Science, Engineering & Applications*, pages 843–852. Springer, 2012.

[4] W-N Lie and L-C Chang. Robust and high-quality time-domain audio watermarking based on low-frequency amplitude modification. *Multimedia, IEEE Transactions on*, 8(1):46–59, 2006.

[5] K Yamamoto and M Iwakiri. Real-time audio watermarking with wavetable alternation in digital instrument. In *Availability, Reliability and Security, 2009. ARES'09. International Conference on*, pages 786–791. IEEE, 2009.

[6] J Huang, Y Wang, and Y Q Shi. A blind audio watermarking algorithm with self-synchronization. In *Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on*, volume 3, pages 627–630. IEEE, 2002.

[7] Sh Wu, J Huang, D Huang, and Y Q Shi. Efficiently self-synchronized audio watermarking for assured audio data transmission. *Broadcasting, IEEE Transactions on*, 51(1):69–76, 2005.

[8] D Megías, J Serra-Ruiz, and M Fallahpour. Efficient self-synchronised blind audio watermarking system based on time domain and fft amplitude modification. *Signal Processing*, 90(12): 3078–3092, 2010.

[9] H Nikmehr and S T Hashemy. A new approach to audio watermarking using discrete wavelet and cosine transforms. In *1st International Conference Communications Engineering*, pages 1–10, 2010.

[10] B Y Lei, Y Soon, and Zh Li. Blind and robust audio watermarking scheme based on svd–dct. *Signal Processing*, 91(8):1973–1984, 2011.

[11] Y Xiang, I Natgunanathan, D Peng, W Zhou, and Sh Yu. A dual-channel time-spread echo method for audio watermarking. *Information*

Forensics and Security, IEEE Transactions on, 7 (2):383–392, 2012.

[12] B Lei, Y Soon, F Zhou, Zh Li, and H Lei. A robust audio watermarking scheme based on lifting wavelet transform and singular value decomposition. *Signal Processing*, 92(9):1985–2001, 2012.

[13] X-Y Wang and H Zhao. A novel synchronization invariant audio watermarking scheme based on dwt and dct. *Signal Processing, IEEE Transactions on*, 54(12):4835–4840, 2006.

[14] K Hyunho, K Yamaguchi, B Kurkoski, K Yamaguchi, and K Kobayashi. Full-index-embedding patchwork algorithm for audio watermarking. *IEICE Transactions on Information and Systems*, 91(11):2731–2734, 2008.

[15] J B Kanade and B S Kumar. Comparison of different wavelet decomposition techniques for peaq model to assess the quality of audio codecs. In *Electronics and Communication Systems (ICECS), 2015 2nd International Conference on*, pages 252–258. IEEE, 2015.

[16] ITU-R Recommendation BS.1387. Method for Objective Measurements of Perceived Audio Quality (PEAQ), 1998.

Ramin Almasi was born in Bandarabbas, Hormozgan, Iran in 1986. He received the BSc degree in Computer (Hardware) Engineering from Isfahan University of Technology and the MSc degree in Computer Architecture Engineering from University of Isfahan in 2008 and 2012 respectively. He is currently working toward the PhD degree with University of Isfahan. From 2012 to 2014, he worked as a network advisor at Hormozgan Power Plant.

Hooman Nikmehr received his BSc in Electronic Engineering and MSc in Computer Architecture Engineering both from University of Tehran, Tehran, Iran, in 1992 and 1997, respectively, and PhD degree in Computer Engineering from the University of Adelaide, Adelaide, Australia, in 2005. He is an Assistant Professor with the Department of Computer Architecture, University of Isfahan, Isfahan, Iran. His current research interests include VLSI, digital arithmetic, computer architecture, reconfigurable hardware design and low-power design.