Embedding watermark images in transcoded audio streams for content protection and quality of service monitoring

Shervin Shirmohammadi*
Distributed Collaborative Virtual Environment Research Lab (DISCOVER)
School of Information Technology and Engineering
University of Ottawa, Ottawa, Ontario, Canada
E-mail: shervin@discover.uottawa.ca
*Corresponding author

Hani Jabbour
Distributed Collaborative Virtual Environment Research Lab (DISCOVER)
School of Information Technology and Engineering
University of Ottawa, Ottawa, Ontario, Canada
E-mail: hjabbour@discover.uottawa.ca

Jiying Zhao
Multimedia Communications Research Lab (MCR Lab)
School of Information Technology and Engineering
University of Ottawa, Ottawa, Ontario, Canada
E-mail: jyzhao@site.uottawa.ca

Abstract: Audio streaming has become a popular application on the Internet over the past few years, and is expected to grow with the advent of high-speed Internet access. Content adaptability, copyright protection and Quality of Service (QoS) monitoring of these streams are some of the major issues encountered in audio streaming. While many studies have tackled the issue of audio watermarking for copyright protection, few have considered using watermarks for QoS monitoring and even fewer considered involving content adaptation at the same time. In this article, we present a universal media access compliant (UMA) transcoding module along with a watermarking-based module that provides copyright protection, data integrity and authenticity, and QoS monitoring, all of which in an optionally private manner depending on the intended usage.

Keywords: audio watermarking, audio transcoding, audio copyright protection, audio quality of service.

Biographical notes: S. Shirmohammadi, Ph.D., P.Eng., SMIEEE, is an Assistant Professor at the School of Information Technology and Engineering, University of Ottawa, Canada. He received his B.A.Sc. in Electrical Engineering from the University of Ottawa in 1995, graduating with the highest standing among all engineering students, and continued to receive his M.A.Sc. and Ph.D. in Electrical Engineering from the same University in 1997 and 2000, respectively. His current research interests include audio/video streaming and transcoding, telecollaboration systems, multimedia and P2P networking protocols, and tele-haptics. Dr. Shirmohammadi is also an industry consultant with over a dozen technology transfers to the private sector. He is a licensed Professional Engineer in Ontario, a Senior Member of the IEEE, and a Professional Member of the ACM.

H. Jabbour is currently pursuing a Ph.D. degree in Electrical Engineering at the University of Ottawa, while working as an engineer at Bridgewater Systems in Ottawa, Canada. He received his B.A.Sc. in Computer Engineering and M.A.Sc. in Electrical Engineering from the University of Ottawa in 2005 and 2006, respectively. His research interests include audio transcoding, audio watermarking, content protection, and quality of service monitoring.

J. Zhao, Ph.D., P.Eng., received his B.Eng. and M.Eng. in Computer Engineering, and Ph.D. degree in Electrical Engineering from North China Electric Power University (NCEPU), China, in 1983, 1988, and 1992, respectively, and his second Ph.D. degree in Computer Engineering from Keio University, Japan, in 1998. He worked for NCEPU as an assistant lecturer, lecturer, and associate professor from 1983 to 1994, and as an instructor and associate professor for Department of Computing Science at University College of the Cariboo (now Thompson Rivers University), Canada, from 1998 to 2001. He is currently an Associate Professor at the School of Information Technology and Engineering, University of Ottawa , Canada. His current research interests are on image/video processing and multimedia communications. He is a licensed Professional Engineer in Ontario and a Member of the IEEE.

1 INTRODUCTION

With the proliferation of the internet and the abundance of large bandwidth, audio streaming is becoming very popular and is used throughout the net. The market is delivering very interesting products, most of them offering astonishing multimedia capabilities with countless options and gadgets, yet these products are still far from providing fundamental needs that users and music producers require, such as content adaptability, copyright issues, authenticity, integrity, and quality of service issues. There are a number of areas in audio streaming that need further research.

The emergence of the wide variety of hand held computers and PDA(s) is giving a big push to pervasive computing. In fact, standards or guidelines for multimedia delivery would suggest a layering aspect of the media where the consumer, or receiver can have a direct or ubiquitous control over choosing which layers to use, and thus decide on the quality/cost factor. The universal multimedia access (UMA) paradigm suggests different embedded levels of content quality [1]. The user or receiver would be able to choose the quality of the content to retrieve depending on the device capabilities or network conditions.

At the same time, illegal downloads and the fight between music producers and organizations that are distributing the same music online for free is still ongoing, making copyright protection as well as authenticity of the music and its integrity a vital issue for many companies and users around the globe.

In addition, users do not have any means to know if the quality of the audio has changed or not during the transmission due to bad network conditions or malicious attacks. The normal way to check this is to actually compare the transmitted audio to the original audio, but in case of live streaming comparison with the original audio is not possible in real-time since the audio is created on the go. There must be a capability for users to know the quality of the service they are getting compared to the service that they ought to get. Ideally, this capability should be transparent, meaning that it should not slow down the streaming, affect the service in any adverse way, or change the bandwidth needs. In other words, no additional information should be sent with the stream, but everything should be incorporated within the stream without changing its overall size. At the same time it might be necessary in some cases to leave this quality of service monitoring available only to specific audience, either for security reasons or for business reasons by making this service only available for paid members for example. A very important issue that also tags along is security - users should be able to trust the outcome of the quality of service monitoring. Users should know that the indications they receive about the quality of the audio is not tampered with (intrusion protection), or at least be able to tell whether the values were tampered with or not (intrusion detection).

In this paper, we present a real time 3-level watermarking algorithm, together with a homogeneous transcoding system, that can be used for private copyright embedding, content protection, and QoS monitoring. Let us begin by examining related work.

2 RELATED WORK

Although a lot of research was done on media transcoding, audio has been a second-class citizen in this area. Big companies like IBM have tackled the issue [2] but mostly looking at video transcoding and image transcoding, while other research offered full automatic transcoding [3] which included any format to any other format; however they do not offer authenticity, integrity, and ownership solutions or any form of quality of service detection [5]. Audio transcoding sometimes means moving from one specification to another, the transformation can go from G 7.11 to G 7.22 to G 7.28 etc… [3][4][5] without changing the actual sampling frequency or sample size of the audio. This can be thought of as heterogeneous transcoding, which is opposite to the transcoding that we present here by adapting the sampling frequency and the sample size of the audio and not merely on the encoding specification change. In terms of quality of service or watermarking, much research has been performed also, some from a network perspective – for quality of service - and others from an application level perspective – both watermarking and quality of service. Some have considered delivering audio with a given quality or service [1], but they were relying on a new queuing theory at the networking level and the work was not based on monitoring the quality of service of the already existing audio stream. Some work with watermarking techniques alone [2], while others deal with watermarking from an application-layer perspective, mostly for copyright issues [3][4]. Many watermarking papers present robust techniques that can withstand MP3 compression [6], or noise and resampling [7] or even malicious attacks like copysample, zerocross, or flipsample

[8]. But they only present techniques and not applications. The research about real-time audio watermarking turns out quite interesting in [9] and [10], but they deal with the watermarking as a service by itself in complement to other protocols. Interesting papers like the paper by Yan and Huss [9] deal with speech watermarking using GSM and relied heavily on the SRTP protocol. Tachibana's method [10] on the other hand did deal with live audio but not live streaming; the approach was to broadcast the watermark in an analog manner in parallel to the played music in a way for the watermark to be embedded in the recording devices trying to record the music, but the integrity and authenticity of the audio cannot be guaranteed. Another interesting technique of watermarking embedded a binary picture in audio after it transformed this audio into Cepstrum domain [11]. This allowed the watermark to be quite robust but the paper's focus was on the watermarking technique itself, without addressing the above-mentioned issues. In fact, not much work has been done on dealing with watermarking as a method for both quality of service monitoring and copyright protection at the same time. None of the papers encountered took advantage of the existence of encryption algorithms and used them in parallel with watermarking in order to achieve data integrity and data authenticity. An interesting paradigm for using watermarking for QoS purposes is the work by Wang et al [6] where quality changes in images are tracked using a watermarking technique and used in order to detect quality changes in a video. However it only deals with image and video, not with audio; moreover it does not offer any security or privacy for the quality detection itself. New research using Turbo Codes [14] and some other novel techniques have improved watermark embedding and watermark detection to high levels [12] [13] and have produced very robust watermarking. These can all be exploited in a paradigm offering larger scale applications in terms of privacy, integrity and quality of service but they fail to step behind the point of simply defining watermarking techniques due to lack of privacy, integrity and quality of service assessment. In general, research that deals with quality of service does not look at the issues that come along with the quality tracking. Here, we present a system that not only uses watermarking to track audio QoS, but also provides security against an attacker trying to change the transmitted data in a way that conveys a different quality of service. Finally, our system offers privacy, meaning that only the intended receivers will be able to view the watermark and check its quality of service. Let us begin by presenting the theory and design behind our system.

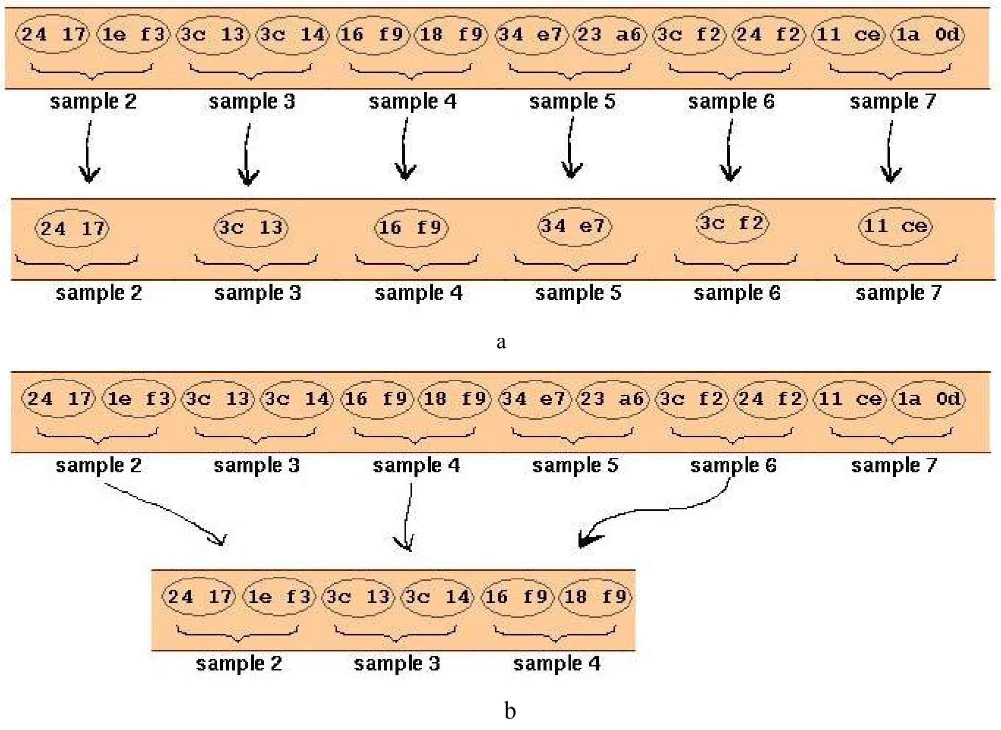
3 THEORY AND SYSTEM DESIGN

In the scope of this paper we are dealing with audio transcoding, audio quality of service and watermarking. The audio used is ITU-T G7.11 coded audio, which is the mandatory codec required in all ITU-T compliant system implementations. In order to achieve our goals, the proposed system is designed with two separate modules, the first module deals with on the fly transcoding, the second module uses a 3-level watermarking scheme, each level having its own purpose. The first level tries to answer the most basic need in our goal which is to provide authenticity, the second level is done to protect the watermark from being changed and detect any changes in the audio data itself, and the third level tracks the quality of the received audio stream. We start the discussion by the first module: the transcoding module.

3.1 Transcoding Module

Transcoding here refers to adapting a live audio stream that comes from a microphone from a given frequency and bit rate to another frequency and bit rate chosen by the end user, while keeping the same standard audio representation. ITU-T G 7.11, which is PCM based, is the audio data representation that is adopted in this paper. PCM leaves the door open for future improvements with other codecs. The user has a list of frequencies and bit rates to choose from. The transcoded streams are obtained by resampling and by downsizing on the fly the samples according to the choice of parameters while the stream is being sent at the server side. This is shown in Figure 1. Resampling means choosing samples out of the original samples at a certain frequency that would generate the new needed audio frequency (figure 1a), while downsizing the samples is done simply by dropping the lowest bits of each sample (figure 1b). For example, if we resample a 48 kHz audio at a sample rate of 1 sample out of 2, we will end up with a 24 kHz audio. On the other hand if we have a 16 bit sample size audio and we drop the lowest byte of every sample we will end up with the same audio with 8 bit sample size.

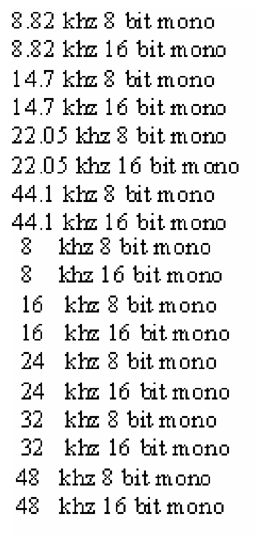
Table 1 shows the various audio specs possible in our transcoding system. This allows the receiving application to adapt the audio according to its context and capabilities in real-time and on the fly without forcing the server to store or capture different streams, for the same audio, in all of the possible specs. This is very useful with live transmission because it is very cumbersome to capture the audio with all possible frequencies and sample size combinations at the same time while waiting for listeners to choose what they need. It is also useful for non-live transmission since transcoding will prevent the need to store all possible combinations of frequencies and sample size on the server while waiting for the user to still make a choice. The main issue would become the ability to make the transcoding in real time because the whole purpose of transcoding would be lost if the process was not able to keep up with the streaming and play back. The idea is to read the stream into buffers and transcoded the buffers one by one before re-streaming them. If the buffer is too big the transcoding process would take too long, and if the buffer is too small the reading from the input stream and sending to the output stream would be very frequent and would also take too long. The solution would be in finding a suitable buffer size that would be big enough not to have too much reading and writing and big enough not to have too big of a chuck to transcode. Obviously two things are true: The first one is that the size of the buffer would then depend on the CPU power available at the server since it is related to how fast the processing (reading, writing, and transcoding) can be done. The second issue is that the buffer size would be a range and not an exact size. Now that we have a better understanding of the transcoding system, let us tackle the copyright, integrity, authenticity, and quality of service problem. These are solved by the watermarking scheme shown next.



24 17 1e f3 3c 13 3c 14 16 f9 18 f9 34 e7 23 a6 3c f2 24 f2 11 ce 1a 0d
sample 2
sample 3
sample 4
sample 5
sample 6
sample 7
24 17 3c 13 16 f9 34 e7 3c f2 11 ce
sample 2
sample 3
sample 4
sample 5
sample 6
sample 7
a
24 17 1e f3 3c 13 3c 14 16 f9 18 f9 34 e7 23 a6 3c f2 24 f2 11 ce 1a 0d
sample 2
sample 3
sample 4
sample 5
sample 6
sample 7
24 17 1e f3 3c 13 3c 14 16 f9 18 f9
sample 2
sample 3
sample 4
b

Figure 1 *a) AudioDownsizing , b) Audio Resampling*

Table 1 *List of Possible Frequencies and Sample Sizes*



8.82 khz 8 bit mono
8.82 khz 16 bit mono
14.7 khz 8 bit mono
14.7 khz 16 bit mono
22.05 khz 8 bit mono
22.05 khz 16 bit mono
44.1 khz 8 bit mono
44.1 khz 16 bit mono
8 khz 8 bit mono
8 khz 16 bit mono
16 khz 8 bit mono
16 khz 16 bit mono
24 khz 8 bit mono
24 khz 16 bit mono
32 khz 8 bit mono
32 khz 16 bit mono
48 khz 8 bit mono
48 khz 16 bit mono

3.2 Watermarking Module

In order to achieve our goals, the system is designed with three levels of watermarking, each level having its own purpose. The first level is discussed next.

3.2.1 First level watermarking

The choice of the watermarking technique depends on a lot of factors. The watermarking and detection technique has to be fast enough not to alter the real time transcoding. The technique should also be independent of the sampling rate and sample size of the digital audio data since the audio used as cover work – or data in which the watermark is embedded – has different frequencies and sample sizes. At the same time, the watermarking technique needs to provide the least audio alteration possible, within a fragile setting that is capable of detecting any alteration of the original work. This last requirement is crucial in order to protect its integrity of the audio. The technique should also be able to some extent to preserve the copyright of the audio even after a malicious attack or during bad network conditions. The receiver is assumed to know about the existence of the watermark, and only about its existence; it is a blind detection mechanism. Moreover any unaware users should be able to use the audio generically just like any non watermarked audio. Finally, the watermarking should not change the size or the specifications of the audio, in other words it should be transparent. The most suitable technique would be the LSB (least significant bit) replacement. It consists of replacing the least significant bit of each audio sample by a watermark bit - a bit taken from the watermarking image. The watermark would thus be embedded bit by bit in the audio. This is simple, reliable and fast. It fulfils the needed specifications in terms of watermark fragility. At the same time, the audio to watermark is an audio stream, and the stream is transcoded using buffers as mentioned earlier, therefore we are watermarking every buffer by the watermark and that gives a high level watermark robustness: In fact, in an attack not all the watermark bits will be lost; some of them will remain and can be extracted to form a distorted, maybe meaningful picture. Moreover, even if the watermark was not extractable at all in a meaningful way, the probability that at least one of the buffers have a meaningful extractable watermark is high considering that a continuous stream would contain many buffers. At the same time the 1-bit variation is the least possible damage that one can do to an audio sample. Some of the watermark bits might end up being the same as the cover work's bits, which means that theoretically some samples will stay intact. But since anyone can extract the least significant bit from the audio data the watermark is publicly accessible. To simply solve this problem in our system, the user will choose a private key to insert the watermark (Figure 2). The position of the watermark bits in the audio would depend upon this key. The binary representation of the private key would decide on the bits to replace in the audio stream. This way the extraction of the watermark can only be possible if the receiver has this private key. The embedding party can always make the key public if they need the watermark to be publicly available. A remaining problem is that it is possible for a malicious attacker to know that the watermark is embedded in the least significant bit and erase the watermarking by replacing all the LSBs by '0's without the need for the private key. There is no way for the receiver to detect this attack. This is the main reason for the second-level watermark discussed next.

3.2.2 Second level watermarking

The adopted solution was to embed another watermark that contains information about the watermarked data in order to catch any alteration attempts. The idea is to choose a hashing function and a private key. The input to the hash function would be all the bits contained within a buffer in addition to the private key excluding the bits that would eventually be replaced by the output of the hash value. In fact, the output would be a string that will be embedded the same way the first watermark was embedded but at the second least significant bit, which would not be fed into the hash function. This watermarking technique is an informative technique, since it relies on the original stream itself. This way if the attacker changes any of the streams' bits either watermark data or audio data, he will need to reproduce a new hash value using the changed stream and the private key to replace with it the second watermark. Of course this would not be possible if s/he doesn't have the private key. Failing to do so will permit the receiver, who has the private key, to check that the hash value embedded does not match the hash value calculated using the extracted watermark, the audio data (without the second LSB) and the private key. A flag will be raised indicating the attempt of alteration of the watermark and/or the audio stream.

This solution has a drawback: theoretically, the audio quality is lowered for another notch. But practically for the high sample sizes the quality change is still imperceptible. For the lower sample sizes, the audio degradation generated by the transcoding itself is much greater than the one added by this bit change. The degradation is therefore masked and still imperceptible. This is verified later in our experimental evaluation and results.

Up till now, any user that has access to the private key is able to extract a watermark from the audio, being sure - without knowing a priori how the watermark looks like - that the extracted image is indeed the correct watermark embedded by the sender. This is achieved by verifying the value of the hash value, in other words, this is achieved by the second level watermark.

The remaining question is how to measure the Quality of Service of the overall data streaming. The idea is to compare a set of original data before the communication starts to the same set after the communication finishes and check for any lost or erroneous bits.

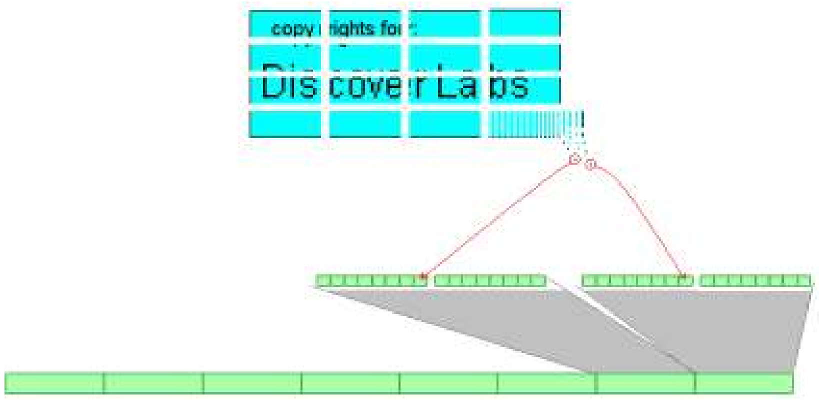


Figure 2 *LSB Watermarking of 16 bit Audio with Little Endian Representation Using a Private Key Equal to "1"*

The live audio or the embedded private watermark cannot be used for this matter since the live stream is generated on spot and the user does not have an original copy of it to compare it to. The private watermark is also private and the user still is not supposed to have a copy of it. The solution can exist only if all the clients have a known piece of information that they know it's going to be sent over the network and that they can check for any alterations. This piece of information can be a picture that is distributed publicly to all the clients and is embedded as another watermark in the original image watermark: the third level of watermarking.

3.2.3 Third level watermarking

The public watermark should be available to every user, or at least be easily creatable in case of unavailability. The watermarking technique must be extremely fragile in order to reflect any change or alteration done to the cover work, thus to the audio. A good suggestion would also be to actually use a binary picture as the watermark because any alteration in the watermark's bit would be visually detectable. An example of a watermark that could be adequate for the purpose of this section is a 20x20 black square bitmap image (figure 3). This square can be available to all the users. Users who don't have it can easily create it.

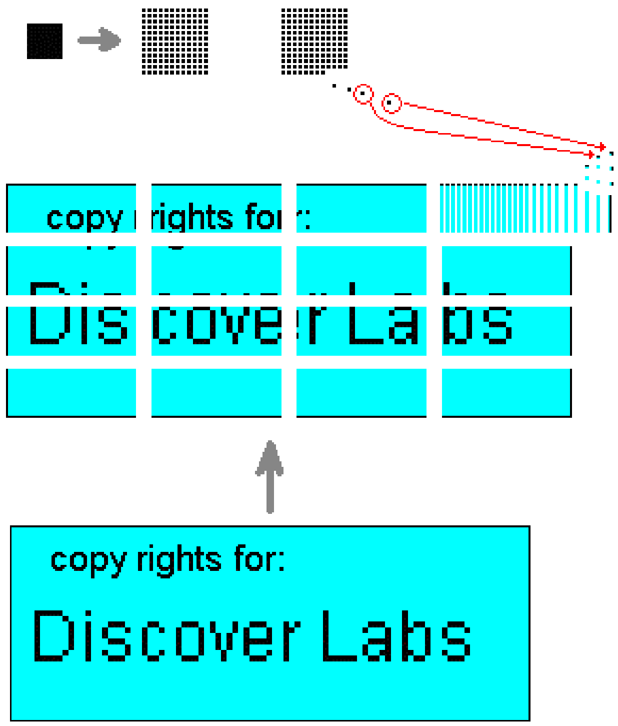


Figure 3 *Embedding of the public watermark in the private watermark*

The fragility of the watermarking technique is essential in this situation since the power to catch any alteration done to the cover work will reflect the efficiency of the quality of service detection. It is also very important to preserve the main watermark (the cover work) in order not to defeat the purpose of the first private watermarking. A possible watermarking technique is to make the sample of the cover work even if the corresponding bit of the watermark is zero and odd if it is one, which basically means that we change the least significant bit according to the bit read of the public watermark. The advantage of this technique in contrast to simply replacing the LSB is that the embedder will only be looking to change all the odd numbers to even numbers – since we know that the watermark is mostly made of a repetition of one same bit (since it is only black) – which makes the embedding much more efficient. If the public watermark was random then this method would not offer any advantages and a simple LSB replacement can also be used. The receiver detects the private watermark for each buffer then detects the public watermark and extracts it, he then can compare bit by bit the extracted watermark to the public watermark that he has in his system, and depending on the match can make an assumption about the quality of service in percentage for the given received buffer. Every buffer would then have its quality of service and looking at all the values, the client can make a general assessment of the quality of service of the whole stream.
However there are three issues to resolve: first, the size of the cover work - the private watermark - is not known and is variable depending on the sender therefore the public watermark cannot be expected to watermark each sample, the watermark is then spread evenly through the private cover work in order to capture the best statistical quality representation. It goes without saying that the closer the size of the private watermarking to the size of the public watermarks the better the representation. The second issue is that watermarking the private watermark would change its bit values and that would change the hash value of the second level watermark, therefore the third level watermarking (the public watermarking) must be done first, this way the black square would be embedded in an image, and this image with its watermark will be then embedded privately in the audio, and the hash value would then automatically take into consideration both private and public watermarks. The first level and second level watermarking modules do not need to know about the presence of the third level watermarking done before them. The third issue deals with the specifications of the private watermark since watermarking a 4 bit per sample image is different then watermarking a 32 bits per sample image. The solution to this problem was to have a default assumption of the type of watermarks to be used and to indicate another specification if a different type of images is used.

4 IMPLEMENTATION AND FEATURES

In order to evaluate the feasibility of the system and to evaluate the real time transcoding aspect of the framework as well as the feasibility of the three-level watermarking technique, we developed a fully functional implementation of the system and tested it thoroughly. The implementation was done in java, in fact, since it is slower than C or C++ at run time: if a real-time implementation can be done in java, then a real time C/C++ counterpart is trivially possible. Using the framework, a client-server architecture was implemented using multithreading programming to allow one server to handle multiple independent clients with different requests and needs. In the next two sections, we examine the features of our system by presenting the inputs and outputs of the system.

4.1 System Inputs

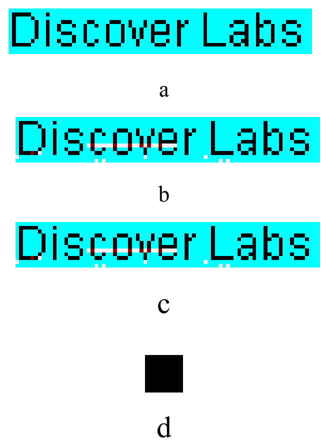


Figure 4 *a) Private watermark b) Watermarked private watermark c) Extracted watermark d) Public watermark, 20 x 20 black image*

Server side:
One obvious input to the server is an audio stream that is captured in real time from the microphone. Another input is the watermark. It is chosen to be a generic image (figure 4a) to be embedded as the "private" image in the audio. This image is supposed to be an image that represents the audio provider. Depending on the need of the provider this image can be a copyright notice or a label or just the name of the provider. It could be anything the provider chooses to embed in the audio. The sample size of private watermark image is also given as an input; this is used internally by the server to embed correctly the public watermark.

Another input is the public watermark used for the QoS assessment. A black 20x20 bmp image is used to watermark the private watermark (figure 4d). The input is the file name of the square. We can see in Figure 4b the watermarked private watermark; this is the final image that will watermark the audio. The last input is the private key: a random number of a random size that the audio provider chooses to secure and privatize the watermarking. This is used at the first and second level watermarking.

Client side:
An obvious input here is the adaptation configuration made by the user: every client can independently choose an audio frequency and a sample size to be played back at its terminal. This can be done either manually, or through an intelligent algorithm that adjusts the transcoding parameters based on the user's context. After this configuration, a simple protocol allows the server to know the chosen format and use it as a target for the audio to be transcoded. The server begins sending the transcoded audio stream that will be received as an input by the client. A private key, matching that of the server side, should also be entered as an input at the client side. This key is used for both watermark extraction and the integrity check of the audio at the client side. Finally, the public watermark used for the QoS assessment is also specified as an input at the client side, the same black 20x20 bmp image used to watermark the private watermark. This will be used to be compared with the extracted 20x20 black square image and make a conjecture about the quality of service of the audio streaming.

4.2 System Outputs

The system has different outputs depending on the presence or lack of attacks.

No attacks:
The obvious output is an audio stream adapted to the client's choice. The stream is played back directly as it arrives at the client and as it is treated for watermark extraction and integrity and quality of service checking. The second output is an extracted private watermark from the specific buffer that the user chose (figure 4c). A quality of service assessment based on the percentage of similarity between the extracted black square image and the image available at the client side also appears as an output. This gives us an idea in terms of how much of the received signal has changed compared to the original, hence indicating the quality. In addition, there are messages that are displayed at the client side as warnings, or flags, in case of various attacks.

Attacks:
The first possible attack is an unauthorized user trying to extract the watermark with the wrong private key. In this situation no valid watermark can be extracted. The output would be a random collection of bits; no meaningful quality of service assessment can be done, the assessment is just random values of percentages. The second type of attack is an attack on the watermark itself by replacing the least significant bit in every sample of the audio in order to override the watermark. In this case, for each buffer, the authorized client (who has a correct key) would receive a message indicating, "The content of the audio has been altered". Any attack on any portion of the audio data itself results in the same output; however, if the watermark is attacked, the QoS module shows a decrease of the quality of service, otherwise the quality of service would still indicate 100%. This way the user can tell if the attack was done on the audio data alone or on the watermark and the audio data. Knowing the aim of the attack we can tell if the attack is a malicious attack or just bad network conditions because a

malicious attack can be selective to only one part of the data, but a non malicious attack cannot be selective; it would not differentiate between the audio data and the watermark data.

5 EVALUATION AND EXPERIMENTAL RESULTS

The system was deployed on a 100Megabit Ethernet LAN with 15 Pentium4 HT 3.4GHz, 1GB RAM machines all running Windows XP. One of the machines was running a server and a client, the others were just running clients, and each client chose a random audio format for adaptation. Subjectively, with 15 simultaneous clients, no client experienced performance degradations from a human hearing point of view. The quality of the audio clearly decreases with transcoding to low levels, but this fact is due to transcoding itself, which drops large amounts of audio data making it lesser quality. The decrease of quality in the audio due to the transcoding and to the "three level watermarking" is very predictable and strongly bounded. In fact, knowing that the dynamic range DR is such that DR(dB) = 20 log10 (2n) = 6.02n where "n" is the sample size, the three level watermarking changes the two least significant bits of the audio resulting in a maximum reduction of the quality of roughly 12 dB for 8 bit samples of a 48 dB range and a minimum reduction of 0 dB if all the replaced bits are replaced by similar bits. Practically, because of the use of a private key, not all the least significant bits are changed, at the same time the probability of match of a replacing bit by a replaced bit turns out to be quite high statistically, it is in fact a 50% chance given that we are replacing 1 or 0 by either 1 or 0. An SNR calculation for 100 watermarked images using the three level watermarking by randomly generated watermarks shows that the decrease of quality for 8 bit sample audio ranges between 2 and 8 dB with a mean of 4 dB which is around 8% audio alteration on a 48 dB range. We should however keep in mind that this quality decrease is much less perceived with higher sample size. We have tested the worse case scenario, as for 16 bit samples the quality decrease would be of a maximum of 12 dB for a 96 dB range with a uniform distribution of mean as low as 4 dB because of the uniform distribution of the occurrence of watermarking bits and because of the uniform probability of having a similar watermarking and watermarked bits, making a mean audio quality decrease of around only 4%.

In order to verify the expectations, direct measurements of the performance of the system were performed. This was achieved by adding timers around the processing blocks in the program and measuring the time needed to perform various tasks done by the system. The performance of the system is dependent on several factors that were all timed:

- The total speed of the transcoding and the effects of the transcoding choices.
- The total speed of the watermarking and the effects of the use of encryption (SHA for second-level watermarking).
- The speed of the extraction and the effects of the use of the QoS detection option.
- The size of the buffer used during the audio transmission.

Other major issues like the network delay are important factors in real-time performance, but they are issues outside of the scope of our system. We are only interested in measuring the performance of the system itself. In the case of this research, the system is a soft real-time system, where the real-time performance is achieved when the data processing of the system is equal or greater to the audio data rate transmission. The purpose is for the users not to sense any change in the live audio transmission. Using this definition, we can say that our system is performing in real time when, for example, the system is able to process more than 32 KB per second when streaming 32 KHz 8-bit audio. Table 2 shows the expected performance for a real-time system.

Table 2 *List of frequencies and sample sizes with the corresponding data rate and needed performance for real time limits*

Audio specifications	**Data rate**	**Real time performance limit for a 24 KB buffer.**
48 Khz 8 bits	48 KB/s	500 ms
48 Khz 16 bits	96 KB/s	250 ms
44.1 Khz 8 bits	44.1 KB/s	544 ms
44.1 Khz 16 bits	88.2 KB/s	272 ms
32 Khz 8 bits	32 KB/s	750 ms
32 Khz 16 bits	64 KB/s	375 ms
24 Khz 8 bits	24 KB/s	1000 ms
24 Khz 16 bits	48 KB/s	500 ms
22.05 Khz 8 bits	22.05 KB/s	1088 ms
22.05 Khz 16 bits	44.1 KB/s	544 ms
16 Khz 8 bits	16 KB/s	1500 ms
16 Khz 16 bits	32 KB/s	750 ms
14.7 Khz 8 bits	14.7 KB/s	1633 ms
14.7 Khz 16 bits	29.4 KB/s	816 ms
8.82 Khz 8 bits	8.82 KB/s	2721 ms
8.82 Khz 16 bits	17.64 KB/s	1361 ms
8 Khz 8 bits	8 KB/s	3000 ms
8 Khz 16 bits	16 KB/s	1500 ms

With this in mind, we present our performance measurements and demonstrate the real-time characteristic of our system.

5.1 Experimental result

The following graphs show measurements of transcoding time for various frequencies and bitrates of the audio. It should be noted that each curve represents the values of 500 consecutive measurements, sorted from longest to shortest.

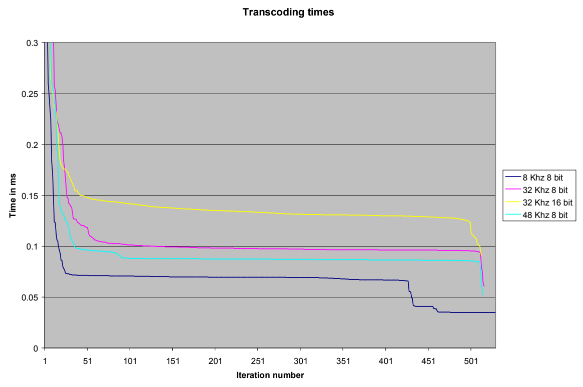


Figure 5 *Total Transcoding Times for Different Frequencies and Bit Rates*

We can see from the results in Figure 5 that the performance of the transcoder seems to depend on the quality requested, since requesting 48 KHz 8-bit is more efficient than requesting 32 KHz 16-bit audio, but less efficient then 8 KHz 8-bit audio. But the dependence is not confirmed because although 32 KHz 16-bit audio has a higher data rate, 48 KHz 8-bit can be considered as higher quality. The performance is, however, due to the fact that the audio is being transcoded from a 48 KHz 16-bit or 44.1 KHz 16-bit source, and the performance depends on how much transformation the source audio needs to have. However, one should not fail to notice that the transcoding stage is a high-performance module and does not exceed (at its worst peak) an average of 0.15 ms. The watermark embedding stage takes more time to perform and practically masks the transcoding stage as the following section will show. The third-level watermarking is done at the start-up time of the server by loading the public watermark into the private watermark, and from then on, the private watermarked watermark is used. Therefore, it is a one-time start-up operation and does not affect in any way the performance of the system.

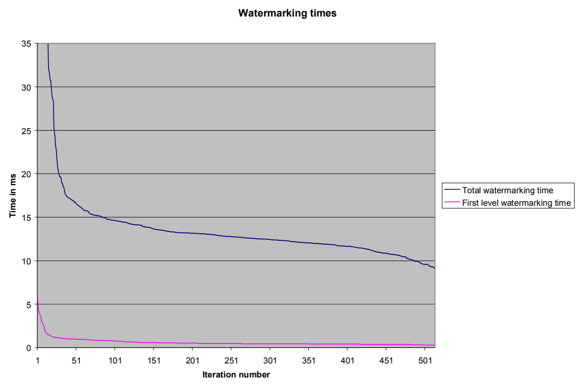


Figure 6 *First Level and Total Watermarking Times*

Figure 6 shows that the first-level watermarking, before the creation of the message digest, has a very high performance and is achieving in the order of 0.5 ms. The results are for a 500 times watermarking repetition for a 32 KHz 16-bit audio, but all the other frequencies and bit rates are in the same order of magnitude. However, the same Figure clearly shows that the second-level watermarking, which encapsulates the call to Java encryption libraries in order to create message digests, is much less efficient and is in fact in the order of 12 ms. This level's magnitude masks the other levels and makes the server's performance dependant on it alone. The performance of the watermark extraction can be seen in Figure 7.

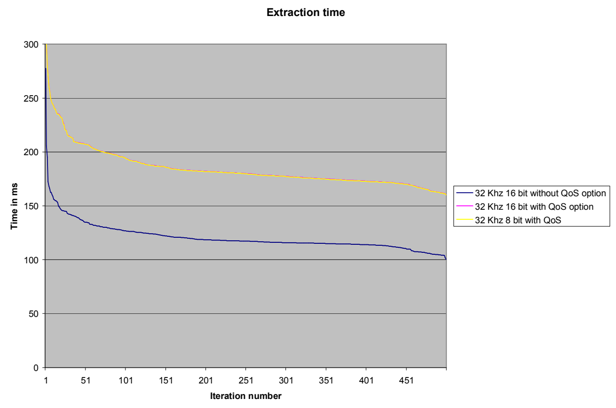


Figure 7 *Extraction Times under Different Options*

The first measurement recorded the watermarking extraction time alone; the second one measured the extraction and the QoS. The measurements were done from the same 500 iterations in order to be able to compare more accurately the effects of the QoS since other measurements showed no difference. However, the figures still show no effect of the QoS assessment. The two graphs seem identical to the naked eye. A closer look at the data showed light differences in the order of 100 nanosecond, which was very negligible when compared to the magnitude of the overall measurements (170 ms).

Our system is capable of behaving as a soft real-time system on an average user computer. As the number of connections increase, the client/server connection tends to be less and less efficient. The server is a bottleneck by concept. One of the possible performance improvements that can be made is the use of the Java Hot Spot technology™ that is aimed for high-performance Java applications and that can increase the performance of the system dramatically.

7 CONCLUSION AND FUTURE WORK

This work proposed an architecture for embedding images, used as watermarks, inside audio bitstreams to provide content protection and quality of service, all while being able to transcode between various audio configurations. The designed system is a transparent system that works ubiquitously on generically captured audio and delivers generically playable audio. Performance results demonstrated the feasibility and real-time characteristics of the system.

The system presented here is a first prototype and more work can also be done. Extending the system to work with other compressed audio formats like MP3 is one example. In that later case the proposed framework should remain the same and the transcoding module would not change by much. The challenge in the future work consists on integrating an MP3 decoder/encoder without changing the real time qualities of the process. The watermarking module however will have to be adapted to MP3 audio. The three level watermarking framework proposed can still be used but different watermarking techniques would be adopted. Any future work will be focusing on finding a proper MP3 audio watermarking technique that could be applied without having to decode the MP3 – it would be too costly time wise – and that could be fragile and transparent enough to be used in the proposed three level watermarking method.

ACKNOWLEDGEMENT

The authors acknowledge the financial contribution of the Natural Sciences and Engineering Research Council's Learning Object Repository Network (LORNET).

REFERENCES

[1] MPEG-7 Content Description for Universal Multimedia Access, ISO/IEC JTC1/SC29/WG11/M4749, MPEG99, Vancouver, BC July 99.

[2] Transcoding solution and service "Extending the reach and exploiting the value of data" Special report, IBM, USA May 1999.

[3] Frost & Sullivan,"empowering unified conferencing with advanced Transcoding", Communication and its solutions, Frost and Sullivan 2004.

[4] VCON Combining Interactive and Streaming Video to Extend the Reach of Multipoint Videoconferencing, VCON white paper, February 2005.

[5] Kevin Curran, Stephen Annesley, "Transcoding media for bandwidth constrained mobile devices", International journal of network management, John Wiley & Sons, inc, March 2005.

[6] M. Arnold, "Audio watermarking: Features, applications and algorithms" IEEE Int. Conf. Multimedia and Expo 2000, vol. 2, 2000, pp. 1013–1016.

[7] Changsheng Xu Jiankang Wu Qibin Sun, "A Robust digital audio watermarking technique", Fifth International Symposium on Signal Processing and its Applications, ISSPA '99, Brisbane, Australia, 22-25 August, 1999

[8] Hamza Özer, Bülent Sankur, Nasir Memon, "An SVD-Based Audio Watermarking Technique", Proceedings of the 7th workshop on Multimedia and security MM&Sec, ACM Press, August 2005

[9] Song Yuan and Sorin A. Huss, "Audio Watermarking Algorithm for Real-time Speech Integrity and Authentication", Proceedings of the 2004 workshop on Multimedia and security MM&Sec, ACM Press, Septemeber 2004

[10] R. Tachibana, "Audio watermarking for live performance," in Security and Watermarking of Multimedia Contents V, vol. 5020 of Proceedings of SPIE, pp. 32–43, Santa Clara, Calif, USA, January 2003.

[11] Lili Cui; Shu-xun Wang; Tanfeng Sun, "The application of binary image in digital audio watermarking". Proceedings of the 2003 International Conference on Neural Networks and Signal Processing, 2003 Volume 2, 14-17 Page(s):1497 - 1500 Vol.2 Dec. 2003.

[12] Xiaomei Quan, Hongbin Zhang, "Statistical Audio Watermarking Algorithm Based on Perceptual Analysis", Proceedings of the 5th ACM workshop on Digital rights management DRM, November 2005.

[13] Hafiz Malik, Ashfaq Khokhar, Rashid Ansari, "Improved Watermark Detection for Spread-Spectrum Based Watermarking Using Independent Component Analysis", Proceedings of the 5th ACM workshop on Digital rights management DRM, November 2005.

[14] Nedeljko Cvejic, Djordje Tujkovic, Tapio Seppanen "Increasing Robustness of an Audio Watermark using Turbo Codes," Conference on Multimedia and Expo 2003 ICME, July 2003.