uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Hani Jabbour

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Electrical Engineering)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

A Real-time Audio Transcoder with Watermark Images For Content Protection and Quality of Service Monitoring

TITRE DE LA THÈSE / TITLE OF THESIS

S. Shirmohammadi

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

J. Zhao

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

A. El Saddik

P.X. Lui

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

A Real-time Audio Transcoder with Watermark Images for Content Protection and Quality of Service Monitoring

By

Hani Jabbour

A thesis submitted to the

Faculty of Graduate and Postdoctoral Studies

In partial fulfillment of the requirements for the degree of

Master in Electrical Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering

School of Information Technology and Engineering

University of Ottawa

Your file Votre référence
ISBN: 978-0-494-25789-0
Our file Notre référence
ISBN: 978-0-494-25789-0

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. In this research study, we present a universal media access-compliant (UMA) transcoding module along with a watermarking-based data-hiding module that provides copyright protection as well as data integrity, authenticity assurance, and QoS monitoring. The audio was first adapted to multiple clients' needs on the server side, watermarked, and then sent to the connected clients. The content adaptability was done by down sampling and a sample size reduction of the live microphone-captured audio. The adaptation was done in conformance to the UMA paradigm. The watermarking, on the other hand, was done on different levels in order to provide guaranteed reliability and authenticity of the audio data as well as an efficient lightweight quality of service monitoring. The watermarking scheme was done on three different levels: First, the private watermark was watermarked by a public watermark. Second, the audio was watermarked by the watermarked private watermark using a private key. Third, the watermarked audio was watermarked again by a hashed value or itself using the same private key. The audio contains an image along with encryption and another image protecting it from temperament and allowing users to track the quality of the audio. The client would have the ability to extract the private watermark and the hash value with his private key, check the integrity of the audio using this hash value and the received audio, extract the public watermark from the private watermark, and compare it to its supposed value in order to obtain a broad idea of the quality of the service. The watermarking techniques used throughout this research were least significant bit (LSB) embedding and some variations of this technique. All the audio data that is outputted can be played with legacy audio players. The content adaptation and the watermarking will not change in any way the conformance of the audio to legacy standards.

Table of Contents

List of Tables

List of Figures

List of Acronyms

API Application Programming Interface
dB Decibel
DCT Discrete Cosine Transform
DFT Discrete Cosine Transform
DR Dynamic Range
DTD Document Type Definition
DWT Discrete Wavelet Transform
FFT Fast Fourier Transform
GSM Global System for Mobile
HAS Human Auditory System
HT Hyper Threading
IDWT Inverse Discrete Wavelet Transform
ITU-T International Telecommunication Union – Telecommunication Standardization Sector
JND Just Noticeable Distance
LAN Local Area Network
LSB Least Significant Bit
MD2 Message Digest Algorithm 2
MD5 Message Digest Algorithm 5
MP3 MPEG-1 Audio Layer-3
MPEG Moving Picture Experts Group
MSE Mean Square Error
PCM Pulse Code Modulation
PIN Personal Identification Number
PSNR Peak Signal to Noise Ratio
QoS Quality of Service
RAM Random Access Memory
SHA Secure Hash Algorithm

SNR Signal to Noise Ratio

TCP Transmission Control Protocol

UDP User Datagram Protocol

UMA Universal Multimedia Access

XML Extensible Mark-up Language

Acknowledgements

At the end of this fruitful journey, I would like to express my gratitude to all who gave me the encouragement and power to complete this thesis, and all who gave me the necessary knowledge and practical support to achieve the goals set at the start of this journey.

I am deeply in debt to my supervisor, Dr. Shervin Shirmohammadi, for his continuous help, support, and guidance and for all the time and trust that he invested in me.

Furthermore, I would like to thank my co-supervisor, Dr. Jiying Zhao, for all his help and advice – without which I could not have reached my goals.

Finally, I dedicate this thesis to my parents, Georges and Milade, my brother and sister Samer and Dima, and my girlfriend Ruba, who have provided everything possible to make my life smooth and to help me achieve my objectives.

1. Introduction

1.1 Motivation

Watermarking is the process of inserting a certain mark into a certain product. It is seen all around us – on money, in pictures, on TV. An impression that is made during the papermaking of money bills is a watermark that helps authenticate the bill in use. The visible or semi-visible logos in images or on TV channels are also watermarks to indicate the proprietorship of the video broadcast or image concerned. Hidden information and messages can be passed invisibly with various media as watermarks for many civil and military reasons.

Watermarking is used to achieve many different goals, most of which are enrolled under copyright protection and content authentication. In fact, the insertion of a watermark in a cover work can be used to prove the ownership of the cover work. Private keys can be used along the watermark to prevent others from changing or viewing the watermark, or even detecting its presence. These private keys play a major role in authenticating a cover work.

Multiple users receive many multimedia products blindly over the Internet. The watermark in the media is sufficient proof for the receiver regarding the authenticity of the work. In fact, the sender uses a private key (a PIN, i.e., a sequence of numbers known only to himself and to the receiver) to embed the watermark in the media that he is sending; the key is shared with the intended receiver(s) or audience. Only the receivers with the matching key can view the watermark. This serves as a proof of authenticity for the receivers. The watermark can also be embedded so that it will be destroyed or altered if a third party tries to change the content of the media. The ability to correctly extract the watermark is also proof of the integrity of the media for the receivers.

Although watermarking is used mostly for privacy- and security-related issues such as authentication and integrity check and copyright and ownership matters, it can also be used for different purposes related to the quality of service assessment. A

watermark accompanying some media can share the same alterations that the media would hold due to network disturbances or due to third party interference. Measuring the percentage of alterations done to the watermark would reveal the quality of the media that held this watermark.

On the other hand, an apparently unrelated topic is the topic of content adaptation. Content adaptation is the process of transforming an incoming product to fit the needs of the receiver. The area is very large and the content can vary from digital to analogue and can consist of web pages, text files, or various media. A more specific digital media adaptation (also referred to as transcoding) is the transformation of digital media from a specific format and properties to another specific format and properties that matches the needs of the receiver.

Transcoding is becoming more and more important with the mass proliferation of heterogeneous multimedia-enabled portable devices such as PDAs and cellular phones. Multiple users can be listening to a news audio broadcast with different devices. Each user has a device with different capabilities that limit him in terms of bandwidth needs, processing time needs, or quality playback needs. The live broadcast, however, does not keep different copies of the same audio broadcast, each with specific properties, in order to match the users' needs. It would also not be reasonable to capture the audio live under different properties and broadcast all of them in order to reply to all the needs. Transcoding provides a solution to this situation where a system is capable of transforming the same content from a certain format and specification to multiple formats and specifications at the time of broadcast. The users will all receive the broadcast according to their needs; the broadcaster would not need to handle more than one set of specifications.

To make transcoding more usable and widely portable, general standards for content adaptation were put in place. These standards, if followed, could allow any transcoding module to communicate with any multimedia receiver and possibly any multimedia sender. The UMA standards are one example of a successful standardisation attempt.

While many watermarking algorithms achieved specific needs for various users, they failed to compile into a unified scheme that would allow many needs to be fulfilled at once. They also failed to acknowledge the need to interact with content adaptation, which would lead to a complete system that would ubiquitously manage generically captured media (in terms of watermarking and transcoding) and hand it over to multiple users with multiple needs.

In this thesis, it is believed that a system exists that would use simple watermarking techniques and transcoding techniques to achieve a complete task of watermarking and transcoding in real time, a task that will give the receiver the ability to receive media according to his own terms and to check the authenticity, the integrity, and the quality of the received media. The media of choice for this thesis is audio.

1.2 Research Problem

Currently there are three main needs that the different studies have overlooked since the research on digital multimedia watermarking and transcoding began. The first need is the need for a general watermarking scheme capable of authenticating and checking the integrity and the quality of the media. The superposition of different watermarking algorithms is a resource-consuming and sometimes impossible matter. Different watermarking algorithms and systems are used for different purposes; a copyright notice can be inserted by a system, while another inserts a private watermark in order to protect the integrity or authenticity of the media. A third watermarking scheme would then be applied to the media to track its quality. However, not all these watermarking are not compatible; each watermark will interfere with the other. The quality-tracking watermark would probably lead to an integrity breach for the integrity-checking watermark, and both these watermarks would probably overwrite or alter the copyright notice inserted earlier. Even if it were possible to find a way where these different watermarking schemes would hack each other and not override each other, the resources that would be consumed in order to do that would be much larger then any efficient system would require.

The second need is the need to focus on what the current users with heterogeneous portable devices are looking for. It is the need to provide the right media to the right user along with the authenticity, integrity, and quality check. The literature focuses on watermarking or media content adaptation alone and does not merge them together. Although the algorithmic aspect of content adaptation and watermarking is not the same, the need for adapted content and for authentic content that is checked for integrity and quality should not be separable. The separation of these needs might contribute to the proliferation of the combination of different non-compatible systems to achieve both needs. Transcoding would breach the integrity of the watermarked audio if not handled properly, and the resources to combine two systems working back-to-back would also be poorly managed.

The third need is the need to do all of the above seamlessly in terms of processing time and playback capabilities. The transcoding/watermarking scheme should achieve a real time performance that would not affect the experience of the users. An adapted authentic audio stream that is checked for integrity breaches and quality is only useful if it arrives as quickly and reliably as one that has not been modified, and could be played back the same way. Users should be able to replay the stream without any need to change or modify any of their original settings. The modified stream should be compliant to the original stream's playback specifications.

1.3 Research Objective

The research conducted was motivated by the need to know whether a system that offers a full content adaptation, authentication, and integrity and quality check using watermarking and UMA-compliant transcoding techniques is possible to achieve – and whether this system would be able to perform in real time and without any alterations to the user's experience.

Many researchers have proposed very efficient and different watermarking schemes, each capable of reaching a specific goal. However, a general use watermarking scheme is still to be found.

The objective of this thesis is to make the best use of already existing watermarking and transcoding algorithms in order to come up with a scheme that can be at the basis of a universal real time transcoding and watermarking system used for content adaptation as well as for authentication, integrity, and quality checks. The full design deployment and testing of such a system is the ultimate goal of this research.

1.4 Research Contributions and New Ideas

- Proposal and implementations of a novel audio watermarking scheme to address content protection, privacy, and quality assessment simultaneously.
- Proposal and implementation of a buffering-based UMA compliant real time audio transcoder.
- Design and implementation of a real-time watermarking and transcoding system using the proposed three-level-watermarking scheme and buffered transcoder.
- Peer-reviewed publications:
  - H Jabbour, S. Shirmohammadi, and J. Zhao, "Embedding Watermark Images in Transcoded Audio Streams for Content Protection and Quality of Service Monitoring", International Journal of Advanced Media and Communication (accepted, to appear).
  - H. Jabbour, S. Shirmohammadi, and J. Zhao, "Watermarking of Audio Learning Objects With Quality Of Service", 3rd Annual Scientific Conference 2006 - LORNET Research Network, Montreal, Canada, September 2006 (Submitted).

1.5 Organization of the Thesis

The remaining part of this thesis is organized as follows:

Chapter 2 provides background knowledge of digital watermarking. It gives a more specific overview about audio watermarking as well as the application of watermarking and the criteria for evaluating watermarking systems. It also offers an overview of content adaptation with some of its applications and a more specific overview of audio transcoding. Standards such as UMA are introduced in this chapter.

Chapter 3 presents a literature review for what researchers have achieved recently in close and related subjects.

In **Chapter 4**, the proposed content adaptation and watermarking system is presented in detail. The design of the system (and all of its components) is discussed in detail.

Chapter 5 presents the implemented prototype. The fine implementation details of the transcoder and the three-level watermarking are presented in this chapter with the Java source code needed to perform the full implementation.

Chapter 6 presents the experimental results showing all the system inputs and outputs. It follows the behaviour of the system under normal functioning and under the abnormal situation of third party attacks. A complete detailed performance evaluation is available in this chapter, showing real time behaviour. It also discusses the possible limitations of the system and possible improvements that can be made to the system for better performance.

Chapter 7 summarizes and concludes the thesis with future work recommendations.

2. Digital Watermarking and State of the Art

2.1 Digital watermarking

A watermark is defined as a mark that shows the greatest height to which water has risen[1]. However, the term "watermarking" was used throughout the centuries to indicate the act of hiding information inside art works or letters without visually changing the aspect of the cover work. Margaret Thatcher was known to ask her aides to make invisible changes to the fonts of each paper handed to any minister. Each minister was given a copy that was changed in a different way. In order to control the source of information leaks, a leaked copy would be compared to the original text. By checking the differences between the copy and the original, the leak could be traced to the corresponding minister. This is watermarking; it is the art of inserting random information in a work that can be extracted when needed without changing the quality of the work.

In the new world of multimedia and communication, the art of watermarking has been adapted to the new digital age; it is more focused on the insertion of marks inside video, audio, or images in order to prevent them from being illegally copied or distributed and to preserve their authenticity and integrity.

Not very far from Thatcher's analogue techniques, digital watermarking is based on digitally changing sample values of the media to a watermark without leaving a noticeable trace. These changes should be detectable by the final recipient in order to extract specific messages or watermarks.

A typical watermarking system consists of a module that is capable of embedding a watermark in a cover work and another module capable of extracting the watermark form the cover work.

[1] http://www.answers.com/topic/watermark

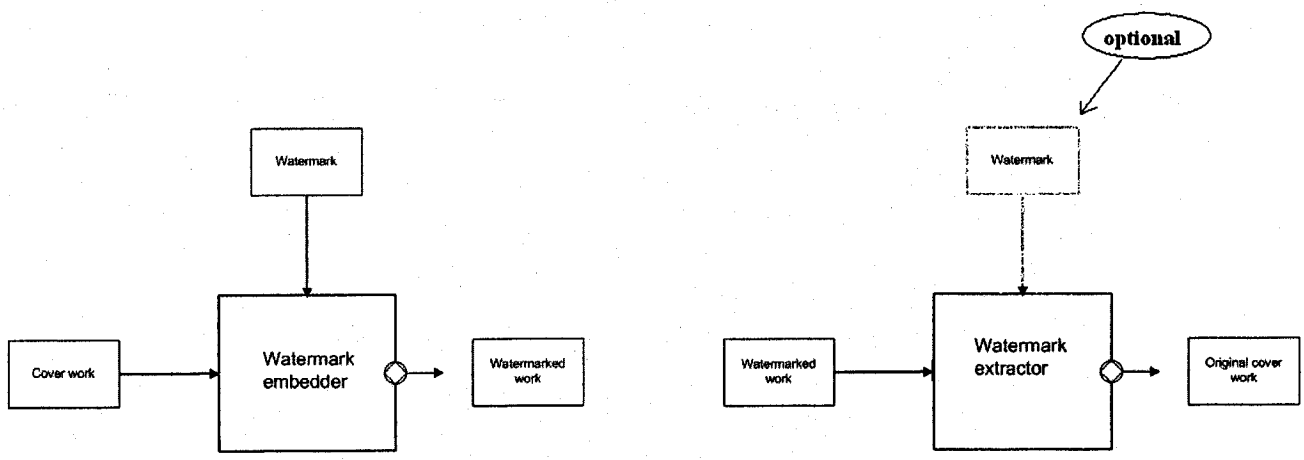

Figure 1: Basic Watermarking System

Figure 1 above shows an example of a system that would be available for anyone to use. Many variations of this system are possible; a variation would be achieved by adding a private key that would allow secure access to the embedding and/or extraction process. Another variation is not to make the watermark available to the extractor; the system would then have a blind detection. Another basic example found in [23] is modeled in Figure 2.

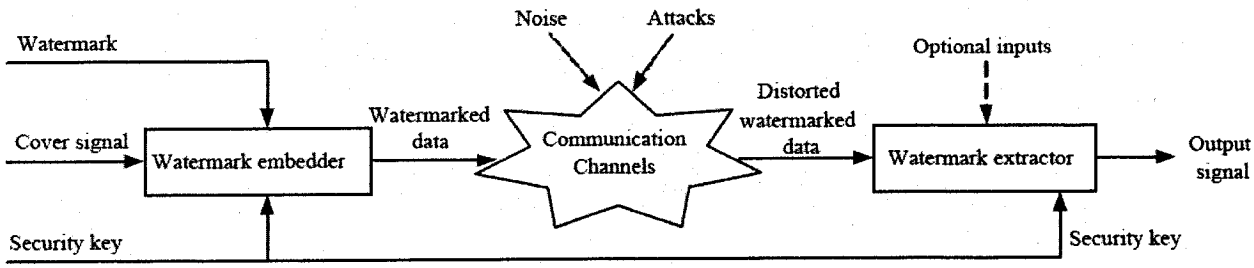

Figure 2: Digital Watermarking System with a Security Key

The ultimate goal of this watermarking system is for the output signal to match the embedded watermark. The security key, optional inputs, or parameters needed for the extraction are all combined with the distorted watermarked data in order to achieve this goal.

Properties of a watermark:

Although a watermark is generally considered to be invisible, some areas of usage require the use of a simple visible watermark. A visible watermark is a watermark that is made so it can be seen or heard when the media is consulted. An invisible watermark is supposed to leave the cover work virtually unchanged. The use of either visible or invisible watermarks depends on the applications' intention. For example, posters' online sellers put extremely visible watermarks on the displayed posters in order to prevent stealth. In Figure 4, IBM embeds a semi-invisible watermark of its logo - shown in Figure 3- to affirm the proprietorship of the picture.

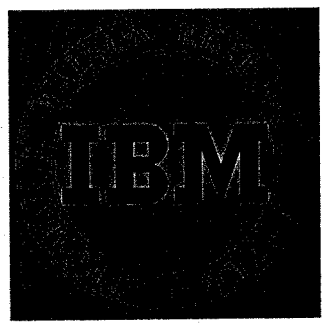

Figure 3: Watermark*

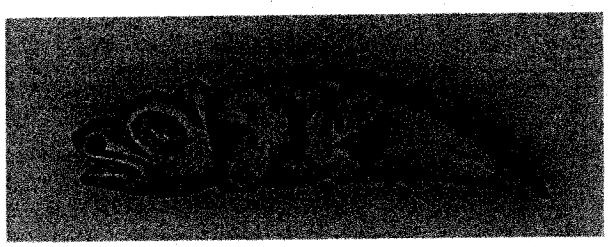

Figure 4: Image with Visible Watermark*

An invisible watermark is harder to extract and might be lost if the watermarked media is subject to transformations. The ability of the watermark to sustain such transformations and the ease of inflicting such transformations or alterations is directly linked to how deep this watermark was embedded in the cover work; this ability is called robustness. There are three types of invisible watermarks:

- A fragile watermark with a very low robustness.

- A semi-fragile watermark with a moderate robustness that allows the watermark to survive some of the possible attacks.
- A robust watermark with a high degree of robustness in order to survive the highest number of attacks possible.

Although this research is focused on audio watermarking, it is easier to explain the different watermarking techniques by using watermarked images. This is mainly due to the visual format of the thesis and because obviously you can't put audio in a printed thesis.

There is a trade-off between visibility and robustness; if the watermark is made very robust, it might become visible. A fragile watermark is one that leaves the cover work visibly unchanged but can be easily removed by a third party or an attacker, or it can easily become undetectable by doing any minor transformations to the watermarked work. A robust watermark is much harder to alter or remove, but will affect the quality of the cover work. The semi-fragile watermark comes halfway between the robust watermark and the fragile one. Figure 5 and Figure 6 show two images one with a fragile watermark and the other with a more robust watermark. Notice the effect of the robustness degree on the final quality of the watermarked image.

Figure 5: Image with a Fragile Watermark

Figure 6: Image With a Very Robust Watermark

Typical equations for a watermarking system are as follows:

Embedding:

$$\begin{aligned} W_m &= \begin{cases} W_r & if\ m = 1 \\ -W_r & if\ m = 0 \end{cases} \\ W_a &= \alpha W_m \\ C_w &= C_0 + W_a \end{aligned}$$

Detection:

$$z_{lc}(c, W_r) = \frac{1}{N} c \cdot W_r = \frac{1}{N} \sum_{x,y} c[x, y] W_r[x, y]$$

Where W_m is the watermark, W_a is the scaled watermark, C_0 is the cover work, and C_w is the watermarked outcome. The α denotes the robustness of the watermark embedding. Figure 5 is an example where the α is small, whereas Figure 6 is an example where the α used is actually a very large number. In this equation m is the watermark message to embed, z_{lc} is the correlation between the absolute value of the watermark (Wr) and the incoming work that is being tested (c). N is the total number of bits. The value of z_{lc} would reveal the detected value of the watermark. If $z_{lc} > 1$ then the message embedded is 1, if $z_{lc} < 1$ then the message embedded is 0.

2.1.1 Applications

Copyright:

A watermark, visible or invisible, can act as a legal notice indicating the ownership of the cover work and its copyright. Many schemes can be used to achieve this goal. The important step is the ability to prove the ownership of the cover work by extracting the necessary information from it or by simply making it displayable when the cover work is consulted. This application is becoming increasingly important these days with the proliferation of uncontrolled digital media distribution.

Integrity:

Instead of a copyright notice, a watermark can hold information about the cover work itself before watermarking; any change to this cover work would be noticeable after comparing it to the information contained in the watermark.

Authenticity:

A private key can be used in order to embed a watermark; the successful extraction of the watermark can indicate the authenticity of the source that embedded the watermark. This scheme would rely on the integrity of the private key itself, but other schemes can incorporate lightweight secure key exchange mechanisms, thus resolving the key integrity problem.

Illegal Copy Protection/Control:

Watermarking can also be used in order to prevent copying of stored media. Although this area has not yet been explored, it is possible to design readers and copiers that would refuse to perform a copy action whenever a watermark indicating a copyright notice is found. The only problem with this application is the necessity of creating a hardware standard to which all the manufacturers should conform and that forces the operating system of the hardware to make the needed checks and perform the needed actions.

Quality Control:

Received media is unknown to the receiver before he receives it; therefore, it is impossible for him to check its quality since he has nothing to compare it to. On the other hand, an embedded watermark can always be compared to what it is supposed to be, thus revealing any quality degradation during the transmission due to transmission noise or lost packets.

Information Hiding:

The most trivial application of watermarking, and the least used, is to use the watermark as hidden information to convey information to select receivers. In fact, watermarking is in concept very similar to encryption where an analogy can be made between encryption/decryption and embedding/extraction. Both military and civil applications exist.

2.2 Audio Watermarking

Sound is digitally captured by taking samples and representing them in a binary configuration. The digital sound is therefore characterized by the sampling frequency, the sample size, and the sample representation in a binary format, or in other words, the modulation. A high sampling frequency will result in a finer representation of the audio, and a larger sample size will also result in a more precise representation of the audio. The modulation will change the quality of the audio, but it is mainly subject to the manner of representation. Pulse code modulation (PCM or G7.11) is the standard modulation for Microsoft WAVE files.

Audio can be later compressed according to different standards such as MP3, G723.1, GSM, and many others. These compressions will affect irreversibly the quality of the audio and its digital content.

In general, audio watermarking is no different from any other media watermarking. The only difference is the cover work: instead of being a video or an image, it is audio. The properties of audio might differ a little bit from other media since

other thresholds apply, such as the hearing thresholds and the echo thresholds, as well as masking thresholds like the temporal masking, the amplitude, and frequency masking. All these properties are used to enhance the inaudibility of the watermark in the audio.

2.2.1 Digital Audio Watermarking

The watermarking can be done digitally in different domains:

Time Domain:

The digital audio is indeed sampled according to a sampling rate, and every sample represents a frequency; it is therefore a time representation domain. An example of watermarking in the time domain would be replacing the values of the samples by other similar values, given the difference in similarities would generate enough information to construct a message: the watermark.

Frequency Domain:

The audio can be viewed from a frequency perspective, the sound waves being a function of their frequencies. In fact, transformations of the time domain representation like the discrete wave transform (DWT) will give away the frequency domain representation. In this domain, watermarking can also be done by replacing the sample with other similar ones – the difference would form the watermark.

Other Domains:

A relatively simple transformation of the time domain data will allow the data to be viewed from a frequency domain perspective; other simple transformations that can be done to time domain data or to frequency domain data also lead to different representations of the data that are not in the time domain, the frequency domain, or in any domain that is relevant to human conception. These domains, however, can have important properties that allow the watermark to be less visible and more robust. An example of these domains is the Cepstrum domain, which is obtained by taking a series

of Fourier transforms, logarithms, and inverse Fourier transforms. This domain has the properties of making the separation of convoluted signals easier.

2.3 Watermarking System Evaluation

The evaluation of the watermarking system is dependent on the application in which the system was used, the application for which the system was initially designed, and the expectations of the users. The robustness of the watermark and its visibility are the most important factors in this evaluation, which also depends on the effectiveness of the watermarking algorithm and the security measures taken to prevent forgery or replacement of the watermark.

Two points need to be clarified. The first is that the watermark's robustness depends on the application: not all the applications require a robust watermark. Some of them require very fragile watermarks; therefore, the evaluation would be a comparison between the needed robustness and the actual one. It is the same situation for the visibility criteria. The second point is that these criteria are indeed contradictory (robustness versus invisibility), which makes the evaluation subjective to the criteria, which is most important in the application at hand.

Visible Watermarking System:

In this case, the watermarking system evaluation would be dependent on how visible the watermark is and whether or not the presence of the watermark altered the original image in a way that prevents it from being used for its purpose. How robust is the watermark? Can it be easily removed by cropping the image or by using simple filters? Did the addition of the watermark change the properties of the cover work in terms of size or dimensions? How efficient are the watermarking embedding and extraction algorithms?

Invisible watermarking system:

The evaluation of this system would start by looking at the alterations that were done by the watermark to the cover work. Are these alterations visible? Is the watermark detectable? Did the watermark change the properties of the cover work in term of size and dimensions? Is the watermark removable? What transformations can the watermark withstand? How efficient are the watermarking embedding and extraction algorithms? What are the characteristics of the watermarking algorithm itself is it blind or is it informed?

2.4 Content adaptation

On the other hand, the growing network of heterogeneous devices ranging from powerful desktop computers to small hand-held devices all trying to access the same media pool pushes for a more pervasive computing and advanced media adaptation. After the introduction of watermarking, content adaptation is a must in the current Internet proliferation paradigm. Various users with various browsing capabilities or various subjective choices all want to have access to the same media, each according to his choice or capability.

2.4.1 UMA

The universal multimedia access (UMA) perspective suggests different embedded levels of content quality. The user or receiver would be able to choose the quality of the content to retrieve depending on his machine's capabilities, network conditions, or personal preference.

2.4.2 Applications

A lot of applications can make use of a content adaptation engine, but they all have the same goal of serving different users at the same time, each according to his need. One obvious application would be an online media broadcasting company to which users connect and access media content. This content can be adaptable to the users' needs. The broadcasting company would not need to save multiple copies of the media (each with a different quality), but would only have one copy that could be adapted on demand to each user.

2.5 Audio Transcoding

Audio transcoding is the act of transforming digital audio from one format to another. The transformation of compressed audio using MP3 compression into another form of compressed audio (e.g., GSM) is transcoding. Transforming a 128Kbps MP3 audio into a 64Kbps MP3 is also transcoding (Figure 7).

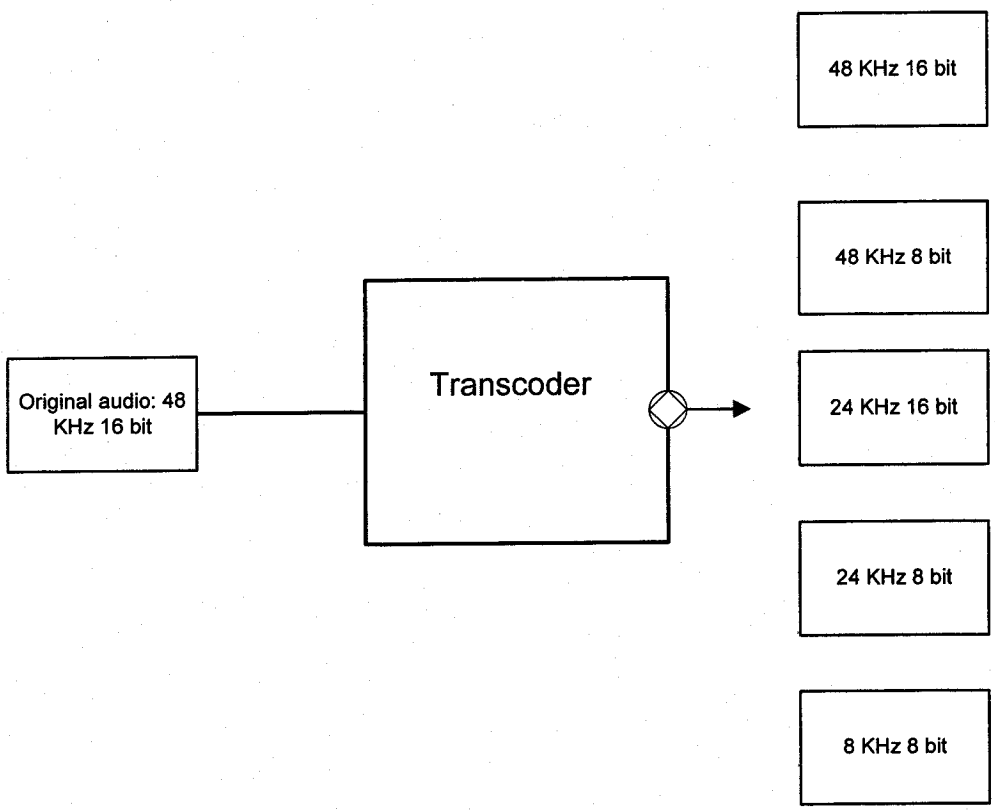

Figure 7: Frequency and Sample Size Transcoder

2.5.1 Evaluation of a transcoder

The quality of a transcoding mechanism is easily verifiable. Transcoded audio should be identical to original audio taken at the transcoded rate and sample size. The quality of the transcoder is the subject to the similarity of the transcoded audio and the expected audio. In other words, if we record voice at a 48 KHz sampling frequency with 16 bits per sample and at a 24 KHz sampling frequency with 8 bits per sample at the same time, the transcoding of the first recording from 48 KHz 16 bits to 24 KHz 8 bits should be identical to the second recording that was directly made at 24 KHz 8 bits.

Another quality metric is the efficiency of the transcoder – in other words, how fast the transcoder is capable of doing the necessary transcoding. The most important advantage of audio transcoding is the ability to adapt the audio into different streams with different specifications, which means that there will be no need to have different copies of these streams, each with different specifications matching the users' needs. But if the transcoding engine takes much more time and resources to generate these different streams, then actually having multiple copies of these streams would lose its advantage.

3. Literature Review

Both audio watermarking and audio content adaptation have been treated extensively in the literature; however, they were never combined or used in any novel combinations to achieve what this research is attempting to achieve. Many watermarking papers offered new watermarking techniques, many of which have been very successful and widely used. Some techniques look for maximum robustness, and others look for minimum distortion.

The problem with these papers is that they all treat the watermarking technique itself and do not go beyond it to look at its usage in a more useful system—i.e., as part of a bigger watermarking system or as part of a larger authentication, integrity, and QoS assessment application with all the needs that such applications need in terms of payload, speed, efficiency, and reliability restraints. Content adaptation papers have the same problem. Very important research has been done for content adaptation, but none have been supplemented with any of the fundamental issues of data transmission like integrity check, authenticity, or QoS. The integration of the research in a real-life application, such as real time audio streaming, also fails to appear in the literature as well as the integration of watermarking as a general solution for the integrity, authenticity, and QoS issues in multimedia streaming and as a specific solution of the same problems in the case of audio transcoding.

Watermarking research in general is all about watermarking algorithms. We are mostly interested in audio watermarking. In [15] and [7] the work was based on delivering an algorithm to find a balance between audibility and robustness by making use of a content adaptive embedding scheme. The watermarking embedding was based on the properties of the human auditory system and on the audio content. The audio was analyzed and important feature measures were extracted from each audio frame constituting the characteristics of the audio. Depending on the characteristics of the audio data, a more optimal embedding scheme was used. The embedding technique in general is based on introducing a delay of the host signal. A time delay was chosen to represent a binary 1 and another was chosen to represent a binary zero. All the delays remained transparent to the human auditory system by respecting a precise threshold. Robustness

was also enhanced by allowing multiple-bit hopping – in other words, more than one bit was embedded in order to represent one binary bit in the watermark. This allows a greater margin for distortion—i.e., if one bit is changed in the embedded sequence, it is still possible to retrieve the original represented bit. The technique of this paper was shown experimentally to be resistant to digital-to-analogue and analogue-to-digital conversions and to white noise addition.

More robust techniques were presented in [8] where the audio was subject to the short-time Fourier transform from which singular value decomposition (SVD) coefficients are drawn. These values were adaptively modified with the watermark bits of a random signal or key. The watermark would be embedded in the singular values of the short-time Fourier transform of the audio. The extraction was done by simply reversing the process. We calculated the inverse short-time Fourier transform and then drew a comparison by means of a correlation to the same random signal used as a key during the embedding of the watermark. The correlation values indicated the presence or absence of the random signal, and thus indicated a logical zero or one. Experimentally, this technique was shown to be extremely robust and capable of withstanding various attacks such as noise addition, filtering, compression, sample additions, sample cutting, sample flipping, resampling, echo addition, and many more.

Other researchers were more focused on the ability to protect the watermark after specific compression techniques like with the MP3, which is becoming very popular in the audio world. In [16] the watermark is inserted as a modification of the magnitude of the relative frequency error between the compressed audio and the uncompressed audio. The modifications are all done in the frequency domain using the Fourier transform. This technique is supposed to be very robust; the watermark should survive a large number of attacks. The watermark extraction is blind and is based on a voting scheme, making it even better to extract.

Other papers such as [17] went further and saw in the MP3 a very important opportunity for steganography due to its file size and its popularity. The focus of this paper was on finding a procedure to identify statistically the particular characteristics of each kind of MP3 encoder. The aim was to determine the encoder of MP3 files on the basis of features that are typical for certain implementations of the encoder; doing so

provide a guide for the steganalysis to run an appropriate detection scheme that is designed specifically to fit the encoded audio better, given the type and implementation of the original encoder. The belief is that this scheme will allow a better reliability of the detection process. Features such as size control in stereo files or the usage of characteristics of the reservoir mechanism, or features related to the psycho-acoustic model used in each implementation are all used to identify encoders such as shine, xing, lame, gogo, plugger, m3ec, mp3enc31, and many more.

In some research studies, the aim becomes specific for producing watermarking schemes that are robust to a very narrow type of attack. In [18] a watermarking scheme was developed to be robust against stereo attacks – in other words, attacks that aim to remove one channel of the stereo audio or to merge two channels in one, thus obtaining mono audio. In order to achieve the goals for this scheme, the fast Fourier transform of the audio was computed to obtain the frequency spectrum, and then the signal was compressed (using MP3 compression) and decompressed back to PCM. The fast Fourier transform is also computed out of the decompressed signal.

A comparison of the frequency spectrum of the original audio and the frequency spectrum of the compressed/uncompressed audio will lead to some relatively unchanged frequency magnitudes; the watermark is inserted as a variation of these magnitudes. The watermark reconstruction is done by applying the Fourier transform to the incoming signal and upon studying carefully the magnitudes of the frequencies. The scheme is not blind since the original signal is needed to reconstruct the watermark; it also assumes that the audio used is all in PCM representation. For the robustness against stereo attacks, the same scheme is used, but on a new signal that represents an addition of the right and left channels, the scheme will result in the unchanged frequencies that should be used to insert the watermark. The magnitude modification is done at the same frequencies found independently for the left and right channels.

Some papers focused more on other attacks. IBM research [19] aimed at creating watermarking that is robust against time and frequency fluctuations. The idea was to obtain the frequency domain representation of the audio by taking its DFT and then modifying the magnitude of the transform in every DFT frame depending on the watermark bits. The detection would be done by detecting the variation of the magnitude

between the original audio and the watermarked audio. The watermarking bits are, however, obtained after optimizing the initial watermark bits. In fact, a pseudo-random array is generated and used to shape the watermark bits, making the watermark more detectable when provided with this array at the detection level. The outcome is also put against a psycho-acoustic model in order to make sure that the changes made to the audio are below a standard hearing threshold, making the watermarking as transparent as possible.

In a move towards the wavelet domain, the research in [12] strives to guarantee the best robustness possible with the best transparency possible. This goal was achieved by applying three steps. The first step estimates the objective-masking thresholds for the wavelet packet domain and the time domain, making the watermark transparent regardless of the host signal. The first step then incorporates an adaptation of the threshold to the subjective host signal by using the MPEG psycho-acoustic model I for layer I and the human auditory system statistical thresholds. This gives the closest range to the critical band for the watermark, making it very robust yet still transparent. The second step tries to enhance the robustness of the watermark by only introducing it in the areas that are less affected by attacks. The introduction of the watermark is done mostly at the tonal component of the audio, which is less susceptible to low bit-rate encoding and time-scaling modification attacks. The third step deals with the watermark itself. The embedded watermark is mapped to different subsets; the combination of subsets gives away the watermark to be embedded instead of re-embedding all the watermark bits every time, thus doubling the embedding speed.

Another paper [11] dealt with a different domain, the Cepstrum domain. This paper's goal was to watermark audio data with a binary watermark, a binary picture. The watermarking scheme proposed was said to be robust and could withstand various common attacks. In this scheme, the embedding was done in the Cepstrum domain, which is a representation that is robust to many manipulations because it holds the acoustic information of the audio. This representation is usually used in voice recognition systems; it is achieved by taking a Fourier transform, a logarithm, and an inverse Fourier transform. Thus the convolution operation is transformed into an addition operation. Obtaining the Cepstrum representation is linear since all we are using are Fourier and

logarithmic functions. Therefore, the original can be forged back using the inverse of these functions. It is mainly robust against time-scaling and pitch-shifting attacks. The technique consists of transforming the audio into the Cepstrum domain, embedding the image, controlling the audio distortions, and then bringing back the audio to the time domain. One bit is embedded in every frame.

Acoustic coefficients are introduced in order to control the distortion effect of the watermark on the original cover work. These coefficients are based on audio masking properties and on the human hearing threshold. The extraction assumes that the audio is watermarked and extracts the embedded bit accordingly. The testing of the developed system included various attacks such as white Gaussian noise, MP3 compression, and filtering.

An interesting paper shows a classification of attacks that can be launched against digital watermarking [20]. In this paper, attacks are classified as removal attacks, de-synchronization attacks, embedding attacks, and detection attacks. Removal attacks can be malicious; filtering the digital signal according to a certain filter, or even compressing raw digital content into an MP3 file and then decompressing it are attacks that can lead to the removal of the watermark depending on which embedding algorithm is being used. Removal attacks can, on the other hand, be very specific if the embedding algorithm is known. They can be tailored to target the weakness of the method used in order to replace or remove the watermark. However, if the watermark is independent of the cover work, an attacker can obtain an estimation of the watermark if he can get a hold of more than one watermarked work.

Preventing the legitimate detection of the watermark using de-synchronization attacks is another type of attack. In these attacks, instead of erasing the watermark, the embedded watermark is misaligned and the corresponding detector would fool the detector into classifying the watermarked audio as not watermarked. These attacks can be performed very simply by inserting periodical time delays in the audio. Other attacks that can be classified as de-synchronization attacks consist of scrambling some of the audio parts before the digital content is provided to the detector. Embedding attacks, on the other hand, are totally different; they do not try to remove the watermark or fool the detector into thinking that it is not there. On the contrary, these types of attacks try to fool

the detector into detecting the presence of a simulated, malicious watermark even when there is no watermark embedded. The simplest form of these types of attacks is the copy attack. Copy attacks consist of using an extracted watermark from one part of the digital content and copying it into the rest of the digital content.

In a shift towards research that is closer to the proposed system, a paper by Libin Cai and Jiying Zhao [21] shows the potential application of semi-fragile watermarking in an audio quality measurement. The scheme is based on digital wavelet transform and quantization. A watermark is embedded in the DWT sub-bands of an audio signal and it is retrieved after audio manipulations. If the audio is distorted, the rate of extraction of the watermark decreases according to the amount of distortion in the watermarked audio. In order to make the scheme more precise, an adaptive method is used to obtain the quantization level since different audio files have different robustness levels with the same distortion, which makes the audio quality measurements inaccurate. This way the quantization level is changed in order to meet a fixed detection rate according to which the audio quality can be accurately measured. The experimental results of the paper show accurate quality measurements, but they rely on adaptive schemes that need repetitive embedding and extractions, and repetitive statistical evaluations for watermark embedding with different quantization levels in order to obtain the desired extraction rates. All this should be done and known prior to the attacks in order to be able to measure the quality of the audio accurately.

Aside from watermarking, audio transmission has another important issue to contend with: content adaptation. In fact, audio stream receivers can be very different; they can be end users with different bandwidth capabilities or even electronic devices with specific input formats. Here we look at audio transcoding as the process of adapting audio to the needs of the subscriber. The theme, however, is always studied alone in the literature and has never been linked to any other audio disruptions that may occur, like audio watermarking. Recently, researchers were looking at the effects of using small hand-held devices to access the Internet and how to adapt web surfing to be enjoyable on small screens.

In [4] the proposed solution in the paper is a way to adapt a web page to fit nicely into small hand-held devices and enhance the user's enjoyment while surfing the net. The

main idea is to provide the user with an index page containing an overview of the web page, allowing them to choose which area to view by clicking on it. It would either be a picture-thumbnail-based table of contents of the webpage or a sequential next/back series of adapted pages.

Large companies such as IBM have worked extensively on multimedia transcoding [2]. They offer video and image transcoding and web page adaptation, which includes other forms of transcoding. XML-to-DTD and text-to-speech transformation were all considered as transcoding and presented in the research. However, in most of the literature, audio is considered a second-class issue, and there is a lack of audio transcoding techniques and solutions. Other researchers made an attempt to develop a fully automatic everything-to-everything transcoder [3]. Their motivation was the arrival of the 3G networks with mobile audio and video conferencing. The paper did not contain any details about the eventual functionalities of the powerful automatic transcoder, but it showed that it should be capable of handling any incoming media and transforming it to any form of needed media. The limitations of this system can be picked up immediately; this includes how low-rate incoming media can be transformed to higher rate media. The paper still did not reflect any interactions of the media with other needs such as security, authenticity, and factors that result from a parallel watermarking system.

The rise of the content adaptation needs created a need to standardize the way content adaptation would function. This need was summed up in the establishment of the UMA standard [1] that started in 1999 (and has been on the rise since) in order to achieve a better performing, more suitable standard – especially through MPEG-7 and MPEG-21 development. The emergence of the wide variety of hand-held computers and PDA(s) is giving a big push to pervasive computing. In fact, the standards or guidelines for multimedia delivery would suggest a layering aspect of the media where the consumer or receiver can have direct or ubiquitous control over choosing which layers to use, thus deciding on the quality/cost factors. The UMA paradigm suggests different embedded levels of content quality. The user or receiver would be able to choose the quality of the content to retrieve depending on the device capabilities or network conditions.

Although watermarking and adaptation have been present in the research for a long time, the trend was to develop new techniques to achieve them or to enhance the old

techniques, but there has been no research that deals with how these two themes would work together, or how a single system would deal with content adaptation as well as content security, authenticity, integrity, and QoS using the simplest techniques of adaptation and watermarking.

4. The Proposed Content Adaptation And Watermarking System

In this section, the architecture of the proposed system is discussed. We start the discussion by looking at Transcoding Module which is the chronologically first module of the system. We then would cover the watermarking module and its three-level architecture

4.1 Transcoding Module

Transcoding here refers to adapting a live audio stream that comes from a microphone with 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 re-sampling and downsizing on-the-fly the samples according to the choice of parameters while the stream is being sent at the server side. This can be seen in Figures 8 and 9.

Re-sampling means choosing samples out of the original samples at a certain frequency that would generate the new needed audio frequency (see Figure 8); downsizing the samples is done simply by dropping the lowest bits of each sample (see Figure 9). 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 an 8-bit sample size.

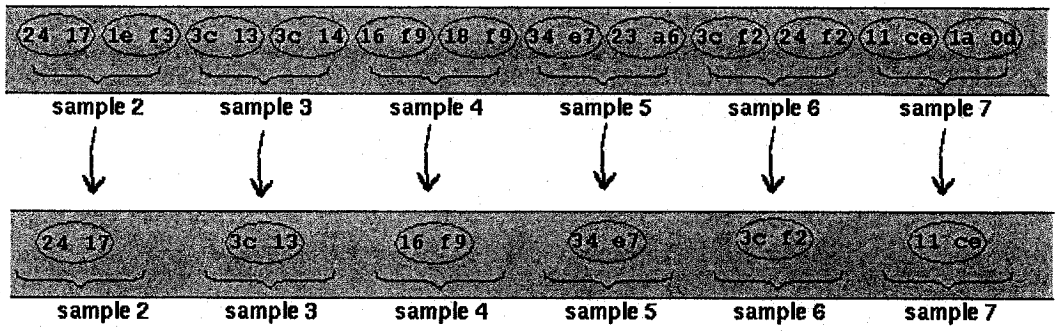

Figure 8: Downsizing

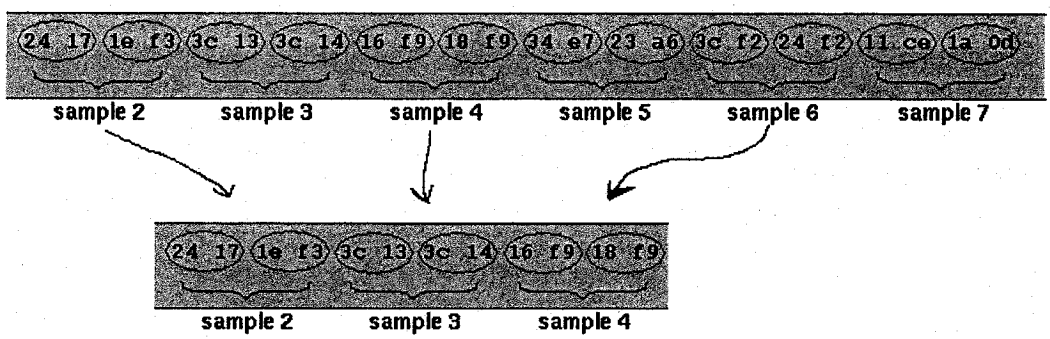

Figure 9: Re-sampling

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 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 were not able to keep up with the streaming and playback. The idea is to read the stream into buffers and transcode the buffers one by one before re-streaming them. If the buffer were too big, the transcoding process would take too long,

and if the buffer were 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 chunk to transcode. Obviously two things are true: The first is that the size of the buffer would then depend on the CPU power available at the server since it is related to how quickly 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.

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

Now that we have a better understanding of the transcoding system, let us tackle the copyright, integrity, authenticity, and QoS problem. These can be solved by the watermarking scheme, shown next.

4.2 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.

4.2.1 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 preserve the copyright of the audio to some extent 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 least significant bit (LSB) replacement. It consists of replacing the LSB 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 fulfills 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 were not extractable at all in any meaningful way, the probability that at least one of the buffers would 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.

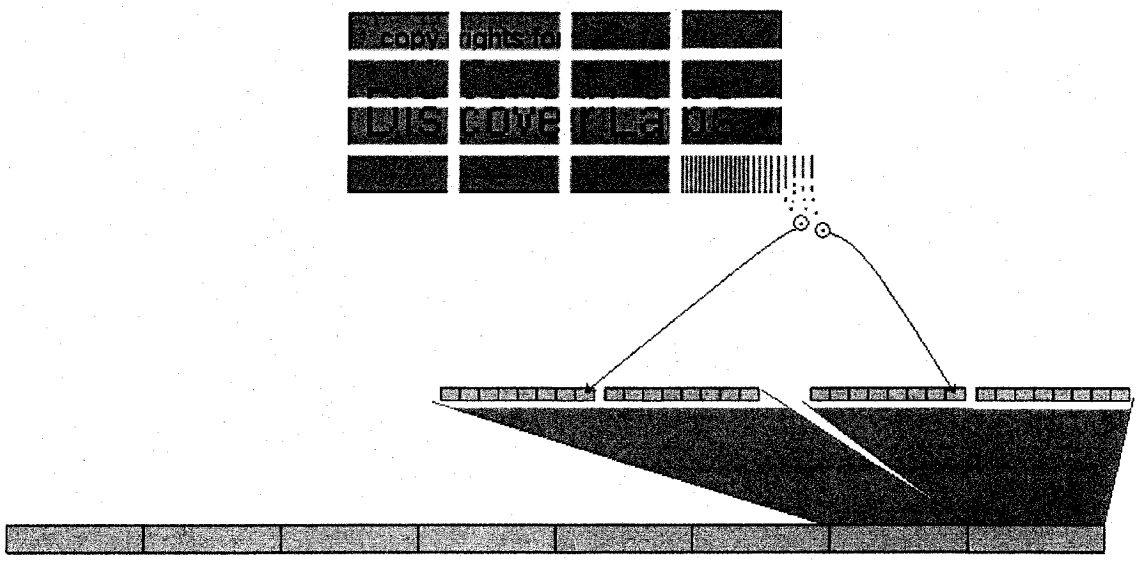

Figure 10: LSB Watermarking of 16 bit Audio with Little Endian Representation Using a Private Key Equal to "1"

Since anyone can extract the LSB from the audio data, the watermark is publicly accessible. To solve this problem in our system, the user will choose a private key to insert the watermark (Figure 10). 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 LSB and erase the watermarking by replacing all the LSBs with '0's without the need for the private key. There is no way for the receiver to detect this attack. In fact, this is the reason why in practice LSB replacement is never used; it is regarded as too simplistic and very easy to defeat. However, this is true if it is used by itself, but the three-level watermarking scheme has been designed especially to make use of the simplicity, speed, and fragility of LSB and to give a solution capable of making it practically impossible to defeat by an attacker due to the second-level watermark. This is the main reason for the second-level watermark discussed next.

4.2.2 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 would be embedded in the same way as the first watermark, but at the second LSB, 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 the second watermark. Of course, this would not be possible if he does not have the private key. Failing to do so will permit the receiver, who has the private key, to check that the embedded hash value does not match the calculated hash value using the extracted watermark, the audio data (without the second LSB), and the private key. A flag will be raised indicating the attempt to alter the watermark and/or the audio stream. In other words, the proposed scheme offers a powerful solution for any alteration attacks such as the common reordering and replacement attacks.

This solution has a drawback: theoretically, the audio quality is lowered 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.

Until now, any user who has access to the private key is able to extract a watermark from the audio, being sure – without knowing a priori what 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 change of the streamed audio and thus know about the QoS 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. The live audio or the embedded private watermark cannot be used for this matter since the live stream is generated on the 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 is 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.

4.2.3 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 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 11). This square can be made available to all users. Users who do not have it can easily create it. 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 QoS 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 LSB 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 – as 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

were random, then this method would not offer any advantages and a simple LSB replacement can also be used.

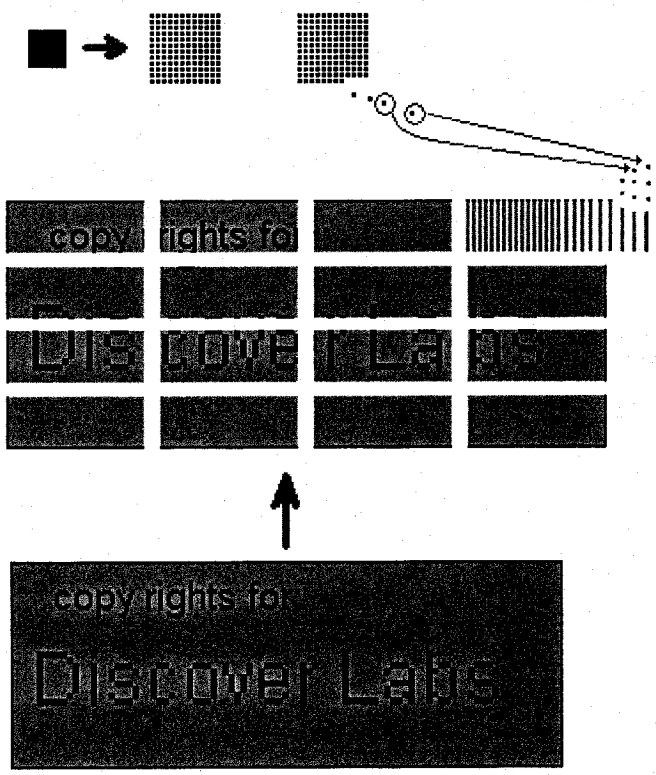

Figure 11: Embedding of the public watermark in the private watermark

The receiver detects the private watermark for each buffer, and 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 QoS in terms of a percentage for the received buffer. Every buffer would then have its QoS, and looking at all the values, the client can make a general assessment of the QoS 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 throughout 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 is 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 (i.e., 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 then be embedded privately in the audio, and the hash value would automatically take into consideration both private and public watermarks. The first- 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 than watermarking a 32-bit 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 image is used.

4.3 Achievement of The System and Combinational Possibilities

So far, the copyright of the audio is protected by the image embedded in the audio, the integrity of the audio data is secured by the second-level watermark, and the authenticity of the data is verified by the use of the private key and by the second-level watermarking as well. The QoS is also tracked with the third-level watermark. All of this is done along with audio transcoding as a soft real-time system. The above-stated goals will be verified in the next section, which contains the experimental results. Nevertheless, the modules and the three watermarking levels are designed in a modular layered architecture and therefore can be used independently to a certain extent, resulting in many possible combinations. The transcoding module can be used alone, the first-level watermarking can be used alone, and the first two levels can be used without the QoS. The third level (QoS) can also be used alone; the public watermarking would then be applied directly to the audio and not to the private watermark. But this would be another research topic.

5. Implementation

The implementation of the system was done completely in Java™. The design of the system was done in an object-oriented paradigm; one static class contained static methods that could be called in order to watermark or transcode various forms of audio input. Multiple implementations of the same methods were done using different inputs. Transcoding and/or watermarking can be done to arrays containing audio data, to audio files, and to audio streams. All the methods have the same basic functionalities; they just work with different parameters. It is very important to know that the audio input is taken in real time from the microphone. The microphone captures the audio according to a certain format in which the programmer specifies the sample size, sampling frequency, and the number of channels of audio capture, as well as the standard adopted to represent the audio in terms of signed representation and which endian to use. In Java™, there is a class ("AudioFormat") in which you specify the parameters you need for audio capture while doing its instantiation:

```
AudioFormat(float sampleRate, int sampleSizeInBits,
int channels, boolean signed, boolean bigEndian)
```

5.1 Transcoding by Sub-Sampling and Re-Sampling.

Audio data can be represented in various forms depending on the characteristics of the audio, on the format in which the system chose to capture it, and on the standard used to represent this audio. Theoretically, there is an infinite number of ways to represent audio data. In this research study, we are looking at PCM audio, which is the ITU-T G7.11 standard. In this standard for an 8 KHz 8-bit sample size audio, every 8 bits (or one byte) of data is a linear representation of 1/8000 sec of audio, and 8000 samples, or 64Kb will account for one second of audio. A 22 KHz 16-bit audio representation would be mean that every 22,000 samples will account for 1 second of audio, and every

sample is linearly represented by 16 bits of data (or two bytes). The higher the frequency, the finer the sampling of the audio, and the larger the sample size, the more precise the representation of the audio gets for the represented time segment. One should not fail to notice that audio that is represented by a16-bit sample takes twice the amount of space as one represented by an 8-bit sample. The doubling of the sampling frequency also doubles the space needed to store the audio data.

For more advanced audio capture and playback (e.g., a stereo), the PCM standard has a specific representation where the right channel is represented by the first sample and the left channel is represented by the sample that follows. Thus, two samples are needed to represent the same audio segment, but each is dedicated to a channel. In this research study, however, we are limited to single-channel audio representation since it is enough for the proof of concept and since moving to a dual channel (i.e., stereo) or more (i.e., 4.1 or 5.1 surround) would not change the basic transcoding techniques used, but would change the parsing of the input.

The transcoding is achieved by adapting directly the exact specifications of raw audio data. During the microphone recording, or audio capture, we chose the original format to be the highest in terms of sample size and sampling frequency among the possible frequencies and sample sizes:

```
format = new AudioFormat (44100, 16, 1, true, false);
```

or

```
format = new AudioFormat(48000, 16, 1, true, false);
```

This way, changing the frequency of the audio is achieved by re-sampling the audio, while changing the sample size of the audio was done by sub-sampling the audio data. We will explain later why we have two choices of format for the base audio capture.

5.1.1 Sub-Sampling

Sub-sampling is used to adapt an audio stream for a 16-bit sample size to an 8-bit sample size without changing the frequency. The representation of the 16 bits, or two bytes, can either be done either in a big endian or in a little endian representation. Little endian means that the least significant byte is stored in the first 8 bits – in the lowest memory address – and the most significant byte is stored at the highest memory address. The big endian representation, on the other hand, reverses the storing places of the bytes – in other words, the most significant byte is stored at the lowest memory address and the least significant one is stored at the highest memory address.

While capturing the audio, one specifies in the capture format if the representation should be done in big endian or in little endian (Figures 12 and 13). The choice between the two is irrelevant for our research purposes; there is no advantage of any format over the other. The capture was done in little endian. The audio stream was stored into an array of bytes; the index of each byte serves as an indicator whether this byte is the most significant or the least significant one. The adaptation from a 16-bit to an 8-bit sample size is therefore done by copying this array to another array half its size while dropping the least significant bytes during the copying.

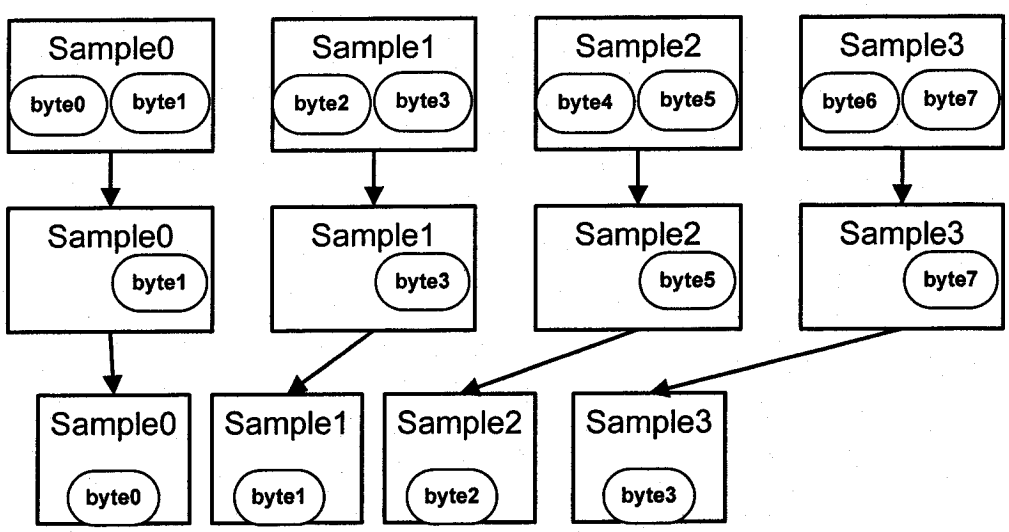

Figure 12: Sub-Sampling of Data with Little Endian Representation

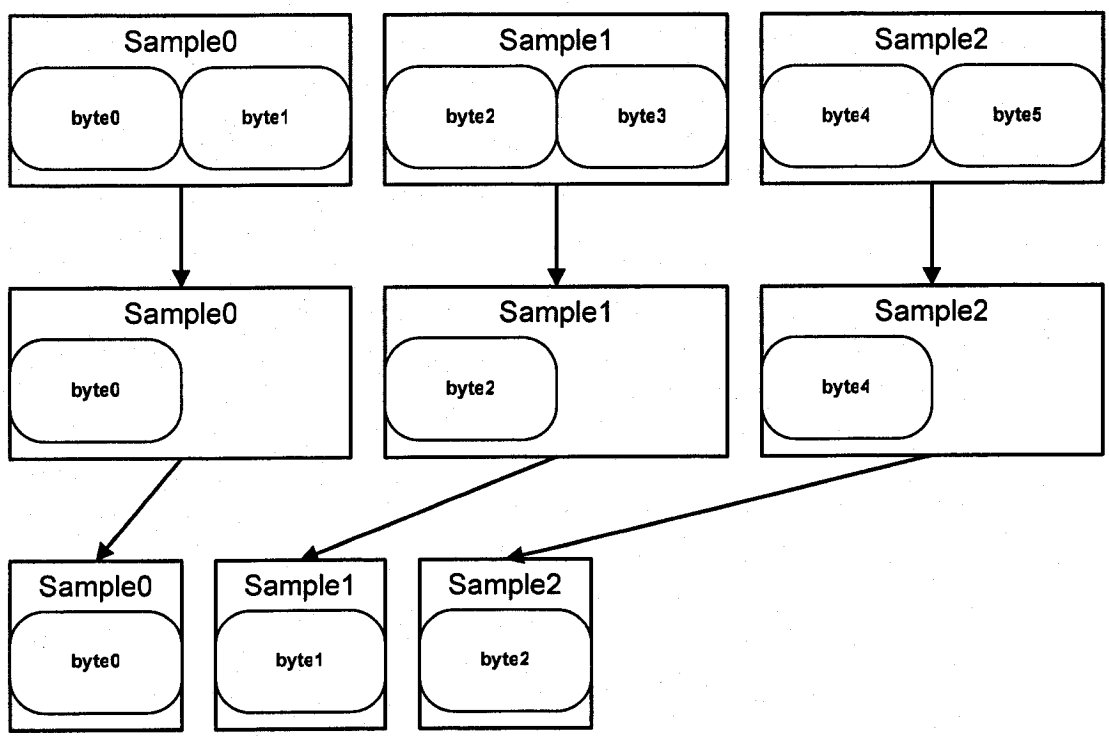

Figure 13: Sub-Sampling of Data with Little Endian Representation

5.1.2 Resampling

Re-sampling, shown in Figure 14, is used to adapt an audio stream from a specific sample rate, or sampling frequency, to another sample rate. An example would consist of moving from 48 KHz to 24 KHz. The idea in this case is to keep a certain number of samples and drop the other samples in order to achieve a sampling rate that would correspond to the same audio as if it had been sampled originally at this later rate. For example, in order to obtain 24 KHz audio from 48 KHz audio, we could keep one of two consecutive samples. To obtain 8 KHz audio from 48 KHz audio we would keep one sample every 5 consecutive samples.

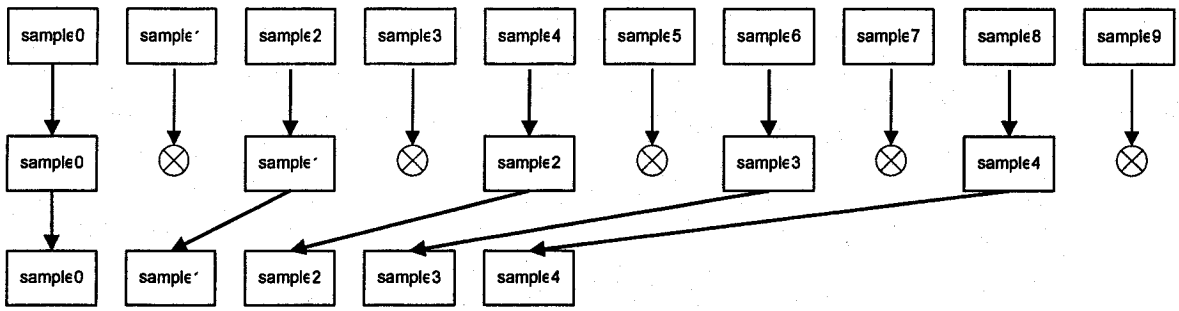

Figure 14: 1 out of 2 Re-Sampling Rate

We should keep in mind that every sample can be either one slot in a byte array, or two consecutive slots in a byte array, depending on the sample size—i.e., whether it is 16 bits or 8 bits.

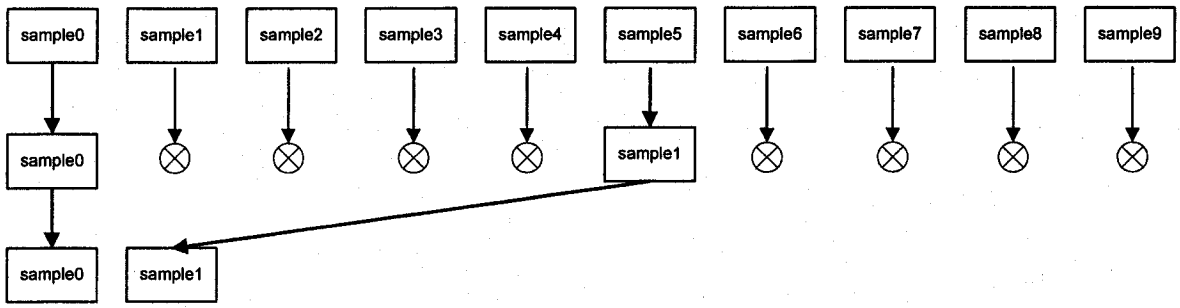

Figure 15: 1 out of 5 Resampling Rate

We notice that it is not possible to obtain 8.82 KHz out of 48 KHz audio since 48 is not a multiple of 8.82; however, we would be able to obtain it from 44.1 KHz. This is the reason why the base audio capture had two options: either capture the audio with 44.1 KHz or with 48 KHz. Depending on the needs of the user, the audio capture can be done accordingly. The table of the implemented transformations is shown on the next page. The audio stream was stored into an array of bytes; however, the stream can have 16-bit samples or 8-bit samples, and the index of each byte will be an indicator as to whether this byte is the most significant byte of a sample or the least significant one, or whether it is a sample by itself. The adaptation is therefore done by copying this array to another array by copying the number of samples needed, as explained earlier. The chosen samples were perfectly spread in the audio stream in order to keep the best audio

representation possible. We should note that if we choose 1 sample out of 4 the size of the array would be reduced 4 times.

Table 2: Possible transformations with corresponding resampling rates

From 48KHz (in KHz)	From 44.1KHz (in KHz)	Resampling rate
8	8.82	1/5
16	14.7	1/3
24	22.05	1/2
32	N/A	2/3
48	44.1	1/1

5.1.3 Adaptation

The complete audio adaptation process is done by combining both re-sampling and sub-sampling in order to achieve the needed, or wanted, audio specification. Going from 48 KHz 16-bit to 24 KHz 8-bit would require choosing one sample from two consecutive samples, and to only keep the least significant byte of the sample at the same time. This combinatory process is shown in Figure 16.

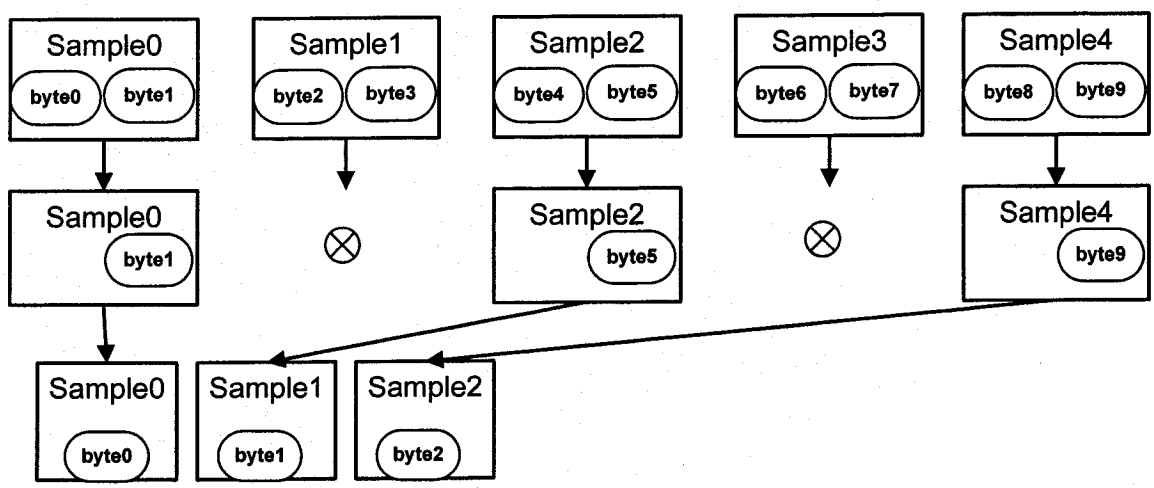

Figure 16: Adaptation from 44.1KHz 16 bit to 22.05KHz 8 bit Audio OR 48 KHz 16 bit to 24 KHz 8 bit Audio

In doing so, the audio data is transformed but the format of the data remains unchanged. In order for the playback to be successful, the format of the transcoded audio should also be changed accordingly, otherwise the playback will attempt to play a 24 KHz sound using a 48 KHz specification; the playback will be too fast. Therefore, after transcoding the arrays of data, a new format will accompany the array in order to ensure a correct playback.

At the same time, the size of the transcoding array should be meticulously followed in order not have empty slots in it. These empty slots will be transformed into zeros and will be written as blank in the output stream or into the playback buffer.

The important part of this section is to know that a continuous audio stream is read consecutively into arrays, or buffers, of sizes specified by the user, and every buffer is transcoded the same way.

```
TargetDataLine.Info info = new
DataLine.Info(TargetDataLine.class,format);
                    TargetDataLine line = (TargetDataLine)
AudioSystem.getLine(info);
                    line.open(format); // Opens the line with
the specified format,
                                                        //causing
the line to acquire any required system resources and
become operational.

int numBytesRead;
is:"+line.getBufferSize());
Begin audio capture.
line.start();


int counter = 0 ;
numBytesRead = 0;
```

```
while (true){

// Read the next chunk of data from the TargetDataLine.
numBytesRead =  line.read(buffer, 0, buffer.length);

...
}
```

The transcoding is done continuously on the buffers, one after the other, in order to keep the data flow going as if the stream were never interrupted. Every buffer is sent for the next third-level watermarking module.

5.2 *Three-Level Watermarking*

The watermarking is done after the transcoding; this way any transcoding would not affect the watermark, but would simply affect the input to the third-level watermarking module introduced earlier, which will handle it accordingly. The third-level watermarking was explained in earlier sections; the first-level being watermarking the audio by an image, the second-level being watermarking the audio by itself, and the third being the watermarking of the image watermark for QoS purposes. However, implementation-wise the third-level watermarking is processed first, then the first-level is done, followed by the second-level watermarking.

5.2.1 Watermarking the Image by a Public Picture

The technique used for the implementation of this level is a slight variation of the LSB replacement method. The choice of this method has slight advantages that were explained in the previous chapter. The watermark at this stage is a black 20x20 square, the cover work, or the data to be watermarked, which is any random picture chosen by the user operating the server side. The watermark, a standard binary image, is loaded bit

by bit into an integer array where each entry is either 0 or 1 depending on the bit of the watermark. In order to read the image bit by bit, it was cast first into an ImageInputStream Object:

```
FileImageInputStream water = new FileImageInputStream(new
File(watermark));
                long watermarkSizeLong = water.length();
                int watermarkSize = (new
Integer(""+watermarkSizeLong)).intValue();
                int [] w = new int[watermarkSize*8];
                for (int x =0; x<w.length; x++){
                     w[x]=water.readBit();
                }
```

The cover work is also loaded into an integer array in a similar way. The watermarking module runs through both arrays simultaneously. When it encounters a 0 in the watermark array, it makes the corresponding sample in the cover work even; if it encounters 1 in the watermark array, it makes the sample odd. Making the sample even or odd is done by replacing the last bit of the sample with either zero or one respectively. However, in order to do that, the sample size of the cover work should be known, as well as the header length of the cover work, in order not to tamper with it while watermarking. Therefore, although the cover work is a random image, and although it is not known by any client, the client should receive minimal information about the cover work in terms of its sample size and its header size.

```
public static void watermarkQOS (int[] in, int[] water, int
sampleSize, int offset){
```

One should not fail to notice two issues: the first is that the two arrays are not necessarily, and are probably not, of the same size. The second issue is that the smaller the sample size of the cover work, the more pronounced the alteration.

The first issue is solved by evenly spreading the watermark within the cover work; this can be easily worked out by looking at the sizes of the arrays and making the necessary calculations:

```
int numberOfBlocks =
(int)((float)inLengthInSamples/(float)waterLength);
```

The second issue is an important issue since we want to inflict the least possible alterations to the private watermark because it is the copyright image of the audio owner, or the image that the audio owner needs to bind to his audio. This issue can only be softened if the audio owner knows about it and refrains from using binary images. The use of 4-bit sample size (i.e., 16 colours) images is enough to embed the public watermark without any noticeable difference. These issues will, however, be revisited in the section about the limitations of the system.

5.2.2 Watermarking the Audio by an Image

At this level, the previously watermarked image is considered as an image to use as a watermark. The fact that it was previously watermarked is irrelevant and is not considered at this level. The image is loaded into an integer array bit by bit where every entry is either zero or one, corresponding to the bits of the image; the transcoded audio is also in an array after being transcoded at the first stage of the processing. The user has specified a private key, and the key consists of an integer. The transcoding method returned the value of the sample size of the transcoded audio, so the sample size is known. The watermarking technique used is an LSB replacement, which is done by transforming the key into its binary representation and going simultaneously through every bit of the key, every entry of the watermark arrays, as well as every audio sample. The audio samples are put in a little endian representation and the least significant byte can be known from the index of the array. While going simultaneously through all three inputs, we check if the key bit is zero, we leave the audio sample as it is, and move on to

the second key bit and second sample, if the key bit is one the last bit of the audio sample is replaced by a bit of the image, and so on till all the bits of the image are embedded in the transcoded audio. We should keep in mind that the audio array is only a buffer that contains audio for a short amount of time, the continuous audio stream is continuously being put into buffers – byte arrays – transcoded and watermarked, and therefore an audio stream would be repeatedly watermarked in each buffer in exactly the same way.

Practically, each bit of the image is considered as a message by itself. An image is a sequence of bits like any other digital data, and each bit will be treated as a watermark on its own. For each bit m of the two dimensional image matrix W:

$m = W_{ij}$

where m would be embedded according to the following equation:

$n <= \text{sample size}$

$C_n = C_n \quad \text{if} \quad n < \text{sample size}$

$C_n = m \quad \text{if} \quad n = \text{sample size}$

where C is the one dimensional matrix representing an audio sample in the case of a big endian representation and n is the variable index in the matrix. The bit from the image would replace a bit from the least significant bit of the audio sample.

In the case of little endian representation, which is the representation used during the implementation of the system throughout the thesis, the equation becomes the following:

$n <= \text{sample size}$

$C_n = C_n \quad \text{if} \quad n < \text{sample size}/2 \text{ or } n > \text{sample size}/2$

$C_n = m \quad \text{if} \quad n = \text{sample size}/2$

However the image bits are not embedded in every sample, but the selection of which sample to contain the image bit depends on the private key entered by the user. The selection would be done according to the following equation:

$B_i = C \quad \text{if} \quad K_j = 1$

Where B_j is the matrix containing the sequence of samples that are selected to be watermarked and K_j is the matrix containing the binary representation of the key entered

by the user, i and k are the indecies of both matrices. The reader should note that if j reaches the maximum value which would correspond to the size of the matrix K, it would be reset to 0 and the iteration through K would start again while j would still be incremented till the stream of audio samples finishes.

At the end of this stage the audio would contain an image while suffering the least possible variation that a digital media can suffer which is the replacement of a least significant bit. Moreover, the replacement is not done to all the sample, and the replacement of a bit might actually be by the same bit value (50% chance), therefore the variation made to the audio stream is inaudible for the users. On the other hand, the end users can only extract and view the image embedded in the audio if the have the key that was used to embed the image in the first place.

The extraction of the watermark would be done according to the following equations:

$n <= \text{sample size}$

$m = C_n \quad \text{if} \quad n = \text{sample size}$

$W_i = m$

where m is a bit extracted from the audio C_n that would represent one bit out of the matrix of bits W_i which would constitute the end of the extraction of the image. This equation is valid when the audio is represented using the big endian format. If it is using the little endian format the equation would then be:

$n <= \text{sample size}$

$m = C_n \quad \text{if} \quad n = \text{sample size}/2$

$W_i = m$

5.2.3 Watermarking the Audio by a Message Digest of Itself

At this level, the audio is watermarked by a message digest of itself, using a private key provided by the user.

5.2.3.1 Message Digest

A message digest is essentially a one-way hash function that takes arbitrary data and outputs a fixed-length string hash value. This essentially means that if you have the output, it is very improbable to be able to reform the original message. At the same time, it is very improbable for two different messages to produce the same message digest. The importance of a good message digest algorithm is that two very similar inputs would reproduce very different message digests as if they were not similar at all.

There are different algorithms to produce message digests, and there are also different standards. In the Java API, many standards are implemented, and we can find the following[2]:

"MD2: The MD2 message digest algorithm as defined in RFC 1319.

MD5: The MD5 message digest algorithm as defined in RFC 1321.

SHA-1: The Secure Hash Algorithm, as defined in Secure Hash Standard, NIST FIPS 180-1.

SHA-256, SHA-384, and SHA-512: New hash algorithms for which the draft Federal Information Processing Standard 180-2, Secure Hash Standard (SHS) is now available. SHA-256 is a 256-bit hash function intended to provide 128 bits of security against collision attacks, while SHA-512 is a 512-bit hash function intended to provide 256 bits of security. A 384-bit hash may be obtained by truncating the SHA-512 output."

[2] http://java.sun.com/j2se/1.4.2/docs/api/java/security/MessageDigest.html

5.2.3.2 Private SHA Message Digest of the Audio

The choice of the algorithm to be used for the message digest can be left to the user, but since it is not of major importance for the system, SHA was chosen. The message digest was produced using all the audio buffer concatenated with the private key that the user produces. This way the message digest cannot be recreated by an attacker who receives the audio because he would need the private key in order to obtain the same message digest.

```
    //disregard the second lsb while computing the hash
value because
    // we are going to replace it later on by the values
obtained.
        byte[] b2 = new byte[b.length + 1];
        for (int j =0; j<b.length; j++){
            b2[j]=new Integer(b[j] & 253).byteValue();
            b2[b.length]= new Long (key).byteValue();
        }

        byte[] bDigest = new byte[20]; //set size for SHA
by default.

        try {

         MessageDigest md =
MessageDigest.getInstance("SHA");
// this performs an automatic last update and then directly
digest it, // the fastest possible method
         bDigest = md.digest(b2);.
```

The message digest obtained is 20 bytes long regardless of the input, and the SHA algorithm used a 128-bit encryption key; it is the best choice in terms of speed/protection.

It can, however, be changed in later implementations to another more secure algorithm, or it can be left to the user to decide, since all you need to change is the parameter of the getInstance() method.

5.2.3.3 Watermarking by the Message Digest

At this stage, we are equipped with the secure message digest that we want to use as a watermark and with the transcoded and watermarked (first-level watermarking) audio that is still in an array. We cast the message digest as an ImageInpustStream object from which we can read bit by bit; we go through the audio array and store each bit read of the message digest at the second LSB of the audio sample. We should point out that the message digest was formed from the audio stream and a user private key without including the second LSB of the audio because it will be replaced during the embedding of the digest as a watermark. At the other hand, the message digest is only 20 bytes; the audio buffer is usually much greater than 20 samples. Therefore, each time we get to the end of the message digest, we go back to the beginning and keep embedding it in the audio stream until all the samples are marked by the message digest.

```
 ByteArrayInputStream bais = new
ByteArrayInputStream(bDigest);
 ImageInputStream ios =
ImageIO.createImageInputStream(bais);
```

```
            int counter = 0;
            byte byt;
            int readbit;

        if (sampleSize == 8){
         for (int i = 0; i<b.length; i=i+freq){
            readbit = ios.readBit();
```

```
                byt = b[i];
// the method markXOR puts the integer readbit which
represents a bit -// zero or one - in the second LSB of the
byte byt
                byt = markXOR(byt, readbit );
                b[i]=byt;
                counter ++;
                if (counter == bDigest.length*8){
                     ios.seek(0);
                     counter = 0;
                }
           }
          }
          if (sampleSize == 16){

                for (int i=0; i<b.length; i=i+freq){

                if (i%2 == 0){

                     byt=b[i];
          // get the value of the XOR of all the bits in
the byte.
                     readbit = ios.readBit();
// put the bit found by the XOR at the second LSB.
                     byt = markXOR(byt,readbit);
                     b[i]=byt;
                     counter ++;
                     if (counter == bDigest.length*8){
                          ios.seek(0);
                          counter = 0;
                     }
                }
```

5.3 Three-Level Extraction

After the third-level watermarking implementation, we need to proceed with the implementation of the extraction schemes of the watermarks and the implementation of the interpretation of the extracted data. The extraction is performed on the audio data that is received by the clients. The extraction sequence starts with extracting the image watermark from the LSB of the audio, then the extraction of the message digest from the second LSB of the audio and comparing it to what we are supposed to obtain, and then the last level of extraction is done on the extracted image by extracting from it the 20x20 black square image and comparing it to what we are suppose to obtain.

5.3.1 First Level of Extraction: Extraction of the Watermark

Given the private key that the user is supposed to provide, we follow the exact reverse process used to embed the image by extracting the correct corresponding bits from the LSBs of the correct samples. The correct samples to extract from are known by following the binary representation of the private key; if the corresponding bit in the private key is 0 then we do not extract anything. We move on to the next sample and the next bit in the key; if it is 1 then we extract the LSB of the sample and save it into a file. When we reach the end of the audio data, the image would be totally extracted and saved to the system. A copy of the extracted bits is kept in an array to be used by the third-level extraction level.

5.3.2 Second Level of Extraction: Message Digest and Integrity Checks

In this section, we need to extract the message digest watermark, as well as check its validity.

5.3.2.1 Computation of the Message Digest

Having the private key (from the user) and the audio, we take the exact same steps in 5.2.3.2 in order to make a 20-byte message digest of the audio – we cannot forget to include the second LSB of the audio.

5.3.2.2 Extraction of the Message Digest from the Audio and Comparison with the Computed one

We extract the second LSB of each audio sample, knowing the sample size (because it depends on the transcoding choice of the client). We then compare the extracted bit to the bit that we are supposed to obtain. The SHA is only 20 bytes, so after 20 * 8 = 160 iterations, we seek again the start of the computed message digest and we keep the comparison (because the message digest was re-embedded in the audio each time it finished) until we reach the end of the audio data.

```
            if (sampleSize == 8){
             for (int i = 0; i<b.length; i=i+freq){

                  secondLSB = (b[i]&2)>>1; // take the second
lsb  and //shift it
            if (secondLSB == ios.readBit()) {
                        temp = 0;
                        }
                  else {
                        temp = 1;
                        }
                  toRet = toRet + temp; // if it is zero then
everything is normal
                  counter ++;
                  if (counter == bDigest.length*8){
```

```
                    ios.seek(0);
                    counter = 0;
               }
          }
         }
         if (sampleSize == 16){



               for (int i=0; i<b.length; i=i+freq){

               if (i%2 == 0){

                    secondLSB = (b[i]&2)>>1; // take the
second lsb //  and shift it
               if (secondLSB == ios.readBit()) {
                         temp = 0;
                         }
                    else {
                         temp = 1;
                                        }
                    toRet = toRet + temp; // if it is zero
then everything is normal
                    counter ++;
                    if (counter == bDigest.length*8){
                         ios.seek(0);
                         counter = 0;
                    }
               }
              }
```

The comparison is done by extracting the two values from each other; a non-zero value would indicate that the embedded message digest does not match the computed one. In other words, the data used to generate the embedded message are different from the data currently found in the audio stream.

5.3.3 Third-Level of Extraction: The Quality of Service

At this stage, we make use of the extracted image obtained at the first extraction level. Given the sample size of the image, the size of the image (which we know since we extracted it), the length of its header, and the size of the black square, we can tell exactly where the watermark bits were inserted and we can extract them to reconstruct what is supposed to be a black 20x20 binary image consisting of a black square.

```
// offset is the header length
// inLength is the size of the image
// sampleSize is the sample size of the image
// numberOfBlocks is the size of the jump made in order to
make the
// watermark spread across the image.
          for (int s=offset; s<inLength; s++){

//give an exist gate if the end of the
// watermark is reached earlier... usually it is the case
since the
// watermark is unlikely to be a multiple of the input (in
terms of
// length).
                if (i==waterLength) {
                     break;}
```

```
            if (s%sampleSize== 0){
                if (sampleSize==16)
                    s=s+8;

            //even => 0
            if (watermarked[s-1]==0){
                watermarkOut[i]=0;
                i++;
            }
            //odd => 1
            else {
                watermarkOut[i]=1;
                i++;
            }
            s=s+numberOfBlocks*sampleSize-1;

        }
        }
```

The extracted bits are compared one by one to the bits of the black square saved on the client's side. A percentage of similarity is returned by this method and is outputted on the screen as an indication of the QoS of the audio streaming. An important thing to note here is that every buffer is watermarked and every buffer is tested, and the quality of service is an indication of each buffer; therefore, during the audio streaming a continuous QoS assessment will be performed.

```
for (int s=0; s<waterLength; s++){
            if (watermarkOut[s]==publicWatermark[s])
                toRet++;
        }
```

The arrays holding the bits of the watermark image and the black square image were obtained by transforming the files into ImageInputStreams and reading them bit by bit into integer arrays. Every input of that array is either zero or one, indicating the corresponding bit.

```
File pwFile = new File(publicWatermark);
FileImageInputStream fis = new
FileImageInputStream(pwFile);
publicWatermarkBits = new int[new
Integer(""+pwFile.length()*8).intValue()];
for (int x=0; x<publicWatermarkBits.length; x++)
    publicWatermarkBits[x]=fis.readBit();
```

5.3.4 The Performance Analysis

For the measurement of the time needed to perform different operations in the system, the system method System.nanoTime() was used in order to obtain the current time in nanoseconds. Making this call before the action and then after the action and substituting the values provides the duration of the action in nanoseconds. For example:

```
start=System.nanoTime();
AudioFile.watermarkBitWise(b,watermark,bitRate,new
Long (key.getText()).longValue());
finish=System.nanoTime();
```

This method is the same for all the performance measurements.

6. Performance Evaluation and Results

6.1 Achievement of the System and Combinational Possibilities

So far, the copyright of the audio is protected by the image embedded in the audio, the integrity of the audio data is secured by the second-level watermark, and the authenticity of the data is verified by the use of the private key and by the second-level watermarking. The QoS is also tracked with the third-level watermark. All of this is done along with audio transcoding as a soft real-time system. The above goals will be verified in the next section, which contains the experimental results. Nevertheless, the modules and the three watermarking levels are designed in a modular layered architecture and therefore can be used independently to a certain extent, resulting in many possible combinations. The transcoding module can be used alone, the first-level watermarking can be used alone, and the first two levels can be used without the QoS. The third-level (QoS) can also be used alone; the public watermarking would then be applied directly to the audio and not to the private watermark. But this would be another research topic.

6.2 Experimental Results

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 came up with 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 and 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 multi-threading programming to allow one server to handle multiple independent clients with different requests and needs. The framework was first tested using TCP protocol for

proof of concept establishment, then was ported into UDP since UDP is much more suitable for multimedia streaming.

Figure 17: Private Watermark

Figure 18: Watermarked Private Watermark

Figure 19: Extracted Watermark

System inputs

Server Side

- The most important 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 (see Figure 17) 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, 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 the public watermark correctly.
- The public watermark used for the QoS assessment is also specified as an input. A black 20x20 bmp image is used to watermark the private watermark (see Figure 20). The input is the file name of the square. We can see in Figure 18 the watermarked private watermark; this is the final image that will watermark the audio. The reader can notice that the quality of the image changed slightly with the appearance of a white dash in the background. The quality change is dependent on the initial sample size of the private watermark. The bigger the sample size the less it would be affected

by watermarking

- One additional 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.

Figure 20: Public watermark, 20 x 20 black image

Client Side

- One input is the adaptations choice made by the user; every client can independently choose an audio frequency and a sample size to be played back at its terminal.
- After the adaptation is chosen, 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 should also be entered as an input on the client side. This key should match the key entered at the server side; without it, the watermark extraction and all its applications will not be feasible. It is also used part of the integrity check of the audio on the client side.
- The client specifies as an input the file name to be used for the extracted private watermark.
- The client also specifies one buffer from which the extraction of the watermark should be done. Since all the buffers are watermarked with the same watermark, it would be a great waste of resources to extract the watermark from each buffer.
- An input is also used to specify the sample size of the private watermark. It is used internally by the client to extract correctly the public black square watermark from the private watermark.
- The public watermark used for the QoS assessment is also specified as an input on the client side; the same black 20x20 bmp image used to watermark the private watermark. The input is the file name of the square. This will be compared with the extracted 20x20 black square image, and it will be used to make a conjecture about

the QoS of the audio streaming.

Most of these inputs will eventually be done automatically by the system and not by the user, putting in place a more pervasive and context-aware framework.

System Outputs (Client Side)

Under Normal Functioning and No Attacks

- The essential output is an audio stream adapted to the client's choice. The stream is played back directly as it arrives on the client side and as it is treated for watermark extraction, integrity, and QoS checking.
- The second output is an extracted private watermark from the specific buffer that the user chose (see Figure 19).
- A QoS assessment based on the percentage of similarity between the extracted black square image and the image available on 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.
- There are messages that can be outputted on the client side as warnings, or flags, in case of various attacks.

Attacks

The system was subject to some basic attacks in order to prove its abilities. The first attack was to consider an unauthorized user trying to extract the watermark with an incorrect private key. In this situation, no valid watermark can be extracted. The output would be a random collection of bits; no meaningful QoS assessment can be done, and the assessment comprises random values of percentages.

The second attack is an attack on the watermark itself by replacing the LSB 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 PIN) 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 in the QoS, otherwise the QoS 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 (i.e., focuses only on one part of the data), but a non-malicious attack cannot be selective, and would not differentiate between audio data and watermark data.

6.3 Performance Analysis

The system was deployed on a 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, and the others were just running clients. Each client chose a random audio format for adaptation. The server was clearly a bottleneck, but with only 15 clients the audio streaming was uninterrupted and was processed and played back in real time by all the clients. The quality of the audio clearly decreases with transcoding to low levels, but this fact is not due to anything within the novel three-level watermarking method, rather the result of the fact that transcoding itself drops large amounts of audio data making it of a lesser quality.

The decrease in audio quality 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 $DR(dB) = 20 \log_{10} (2^n) = 6.02n$ where "n" is the sample size, the three-level watermarking changes the two LSBs 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 were replaced by similar bits. Practically, because of the use of a private key, not all the LSBs were changed; at the same time the probability of a match (i.e., of a replacing bit with 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 with either 1 or 0.

An SNR calculation for 100 watermarked images using the three-level

watermarking by randomly generated watermarks shows that the decrease in quality for 8-bit sample audio ranges between 2 and 8 dB with a mean of 4 dB, which is around an 8% audio alteration on a 48 dB range. We should, however, keep in mind that this quality decrease is much less perceived with a higher sample size. We have tested the worst case scenario: 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 a 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 method is described in the implementation section). In fact, 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. In any case, during the testing, the system was deployed on a dedicated LAN in which the network delay was not a factor.

Before presenting and analysing the measurements, there should be a clear definition of a real-time expectation. 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 3 shows the expected performance for a real-time system.

Audio specifications	Data rate	Real time performance limit for 24 KB buffer (needs half the time for 12 KB)
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

Table 3: List of frequencies and sample sizes with the corresponding data rate and needed performance for real time limits.

The following pictures all represent graphs of sorted 500 consecutive measurements for the performance of the selected module.

6.3.1 Server-side Performance

6.3.1.1 Transcoding Speed: The Effect of the Audio Specification Choice

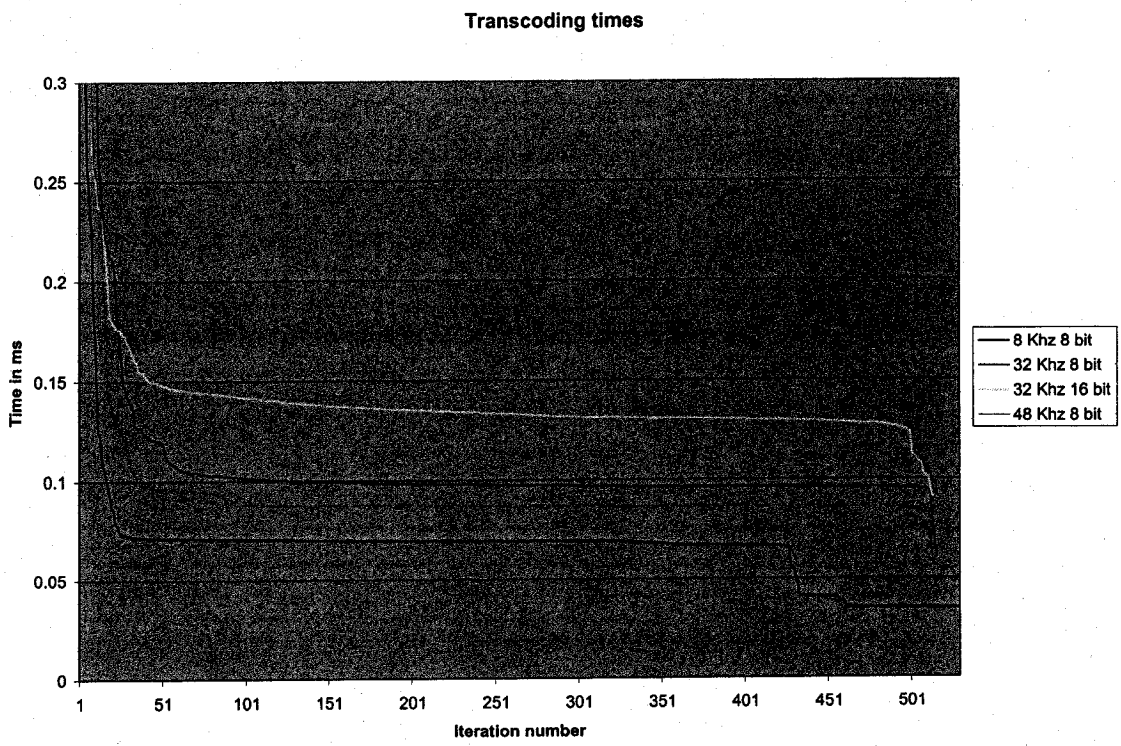

Figure 21: Total Transcoding Times for Different Frequencies and Bit Rates

We can see from the results in Figure 21 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.

6.3.1.2 Watermark Embedding Speed: The Effect of SHA

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. However, at this stage we are looking at the first- and second-level watermarking stages:

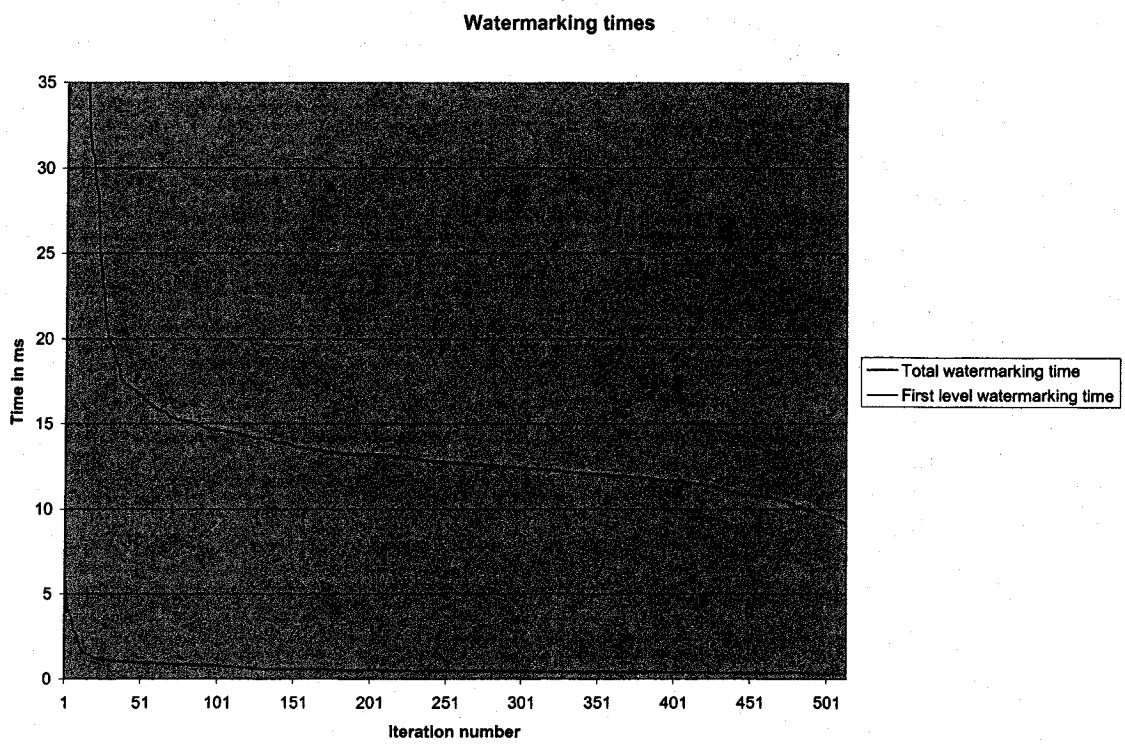

Figure 22: First Level and Total Watermarking Times

Figure 22 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.

6.3.1.3 Total Time Spent on Server Side

Two measurement specimens for the total time spent on the server side are shown in Figures 23.

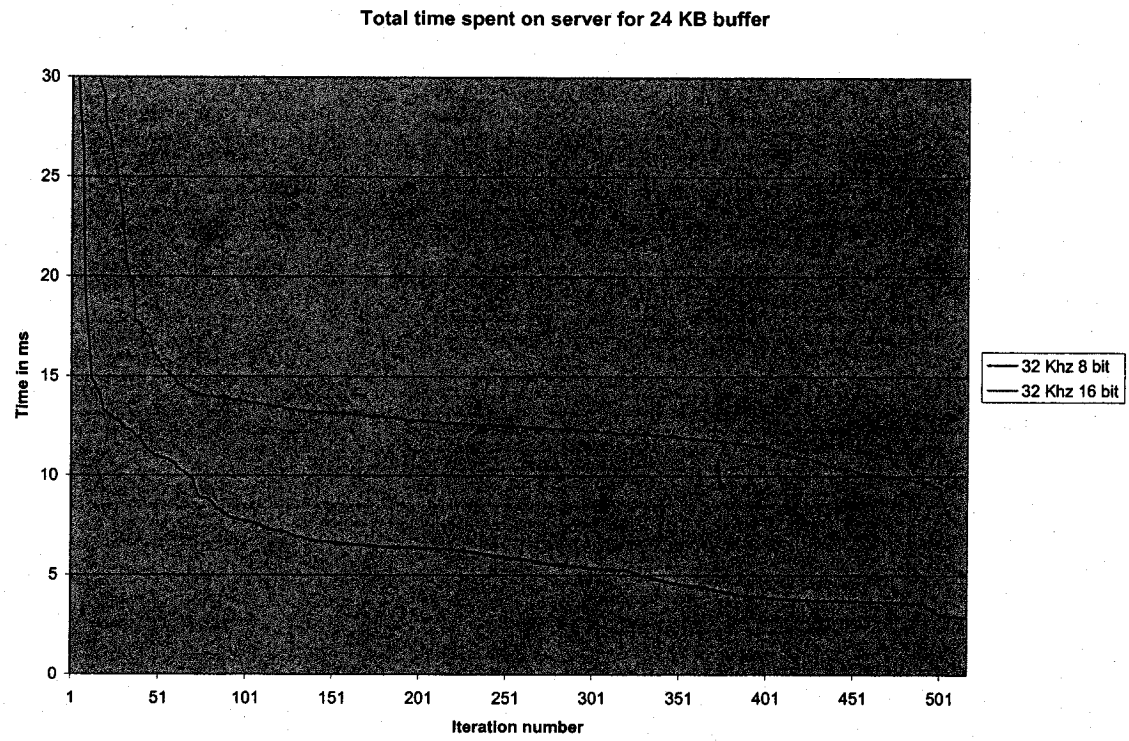

Figure 23: Total Time Spent on Server Side

These measurements show that the performance of the server is very high and is in the order of 10 to 15 ms in the worst case scenarios. These values are much higher than the values necessary for a real-time performance. Also, the server is shown to be capable of acting as a soft real-time system without any major issues.

6.3.2 Client-Side Performance

6.3.2.1 The Effect of the QoS Option

The client side performs the watermark extraction and the QoS verification. The performance of the watermark extraction can be seen in Figure 24.

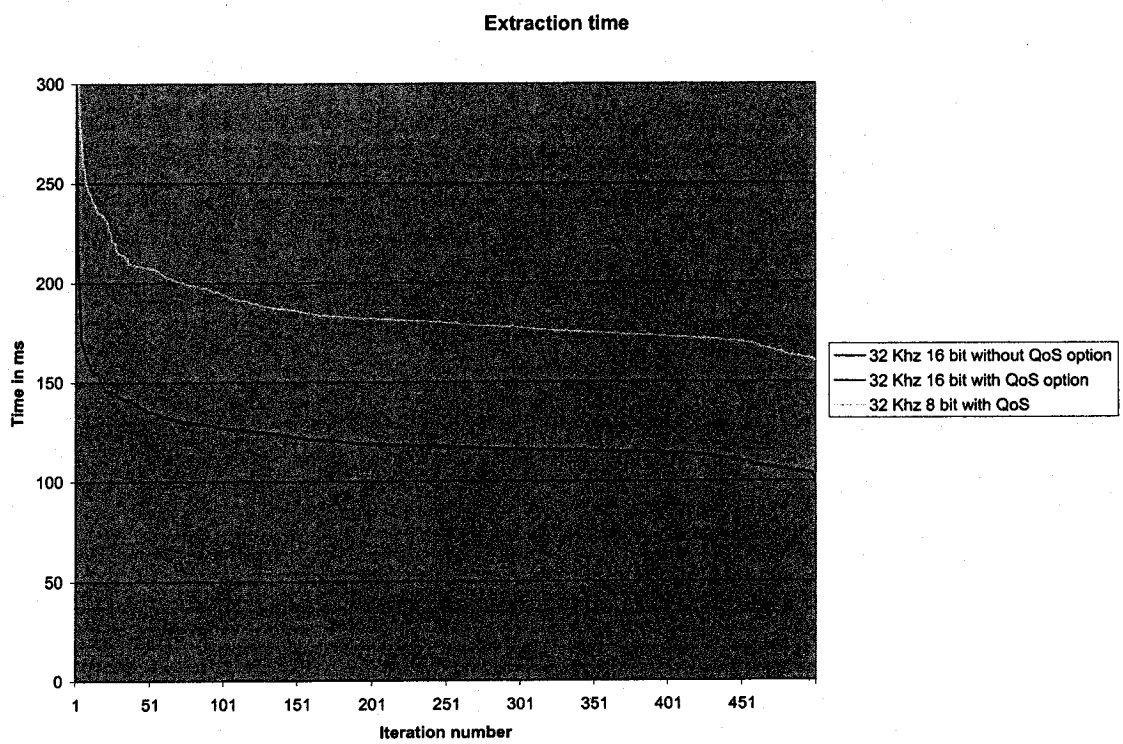

Figure 24: Extraction Times under Different Options

Figure 24 shows the watermark extraction with an additional QoS assessment. The measurements are taken from the same 500 iterations. The first measurement measured the watermarking extraction alone; the second 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 differences in

the order of 100 nanosecond, which was very negligible when compared to the magnitude of the overall measurements (170 ms).

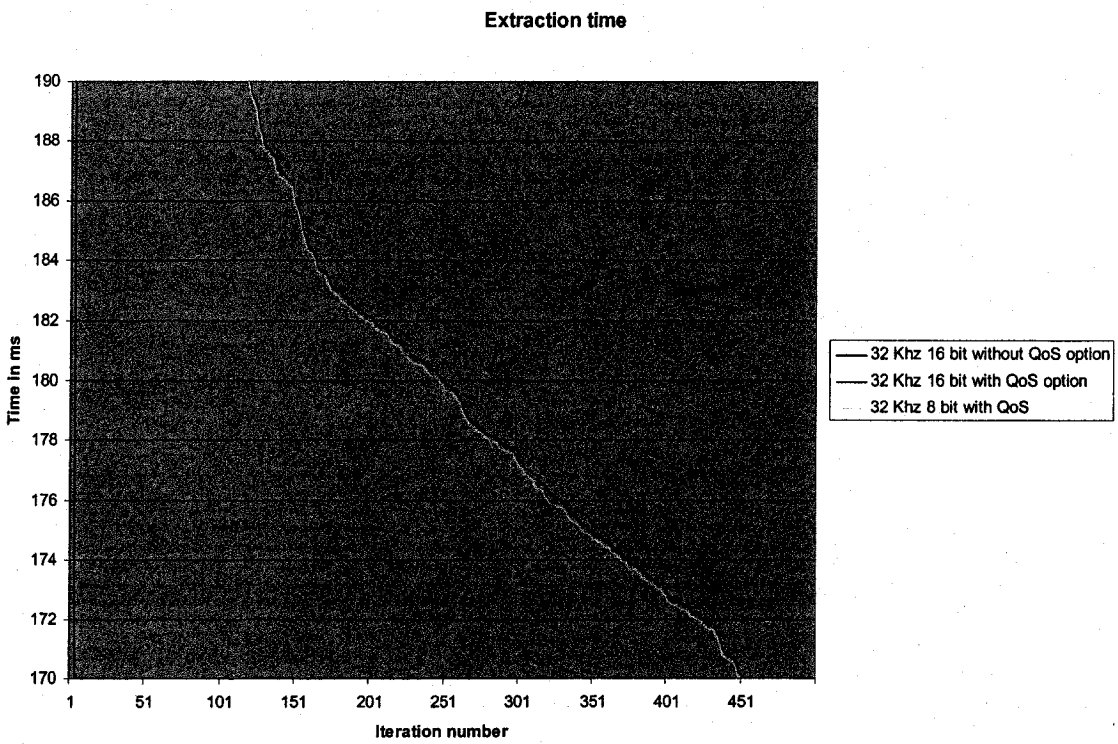

Figure 25: Detailed Image of Extraction times

Although the QoS assessment did not affect the client performance, the watermark extraction seemed to be the more costly action. It consumed up to 170 ms of the client's time. The time limits for a real time 32 KHz 16-bit audio transmission is 375 ms. Even though the client is much less efficient than the server, it is still twice as fast as the time needed for a real-time performance.

Another measurement was done for a 32 KHz 8-bit audio transmission, it is included in Figure 25, which showed that client's total time averaged 120 ms – much faster than the needed 750 ms for a real-time performance.

6.3.3 Effect of the Buffer Size.

The following measurements were done for 500 iterations in a similar manner as before, but using a 12Kb-sized buffer instead of a 24Kb buffer.

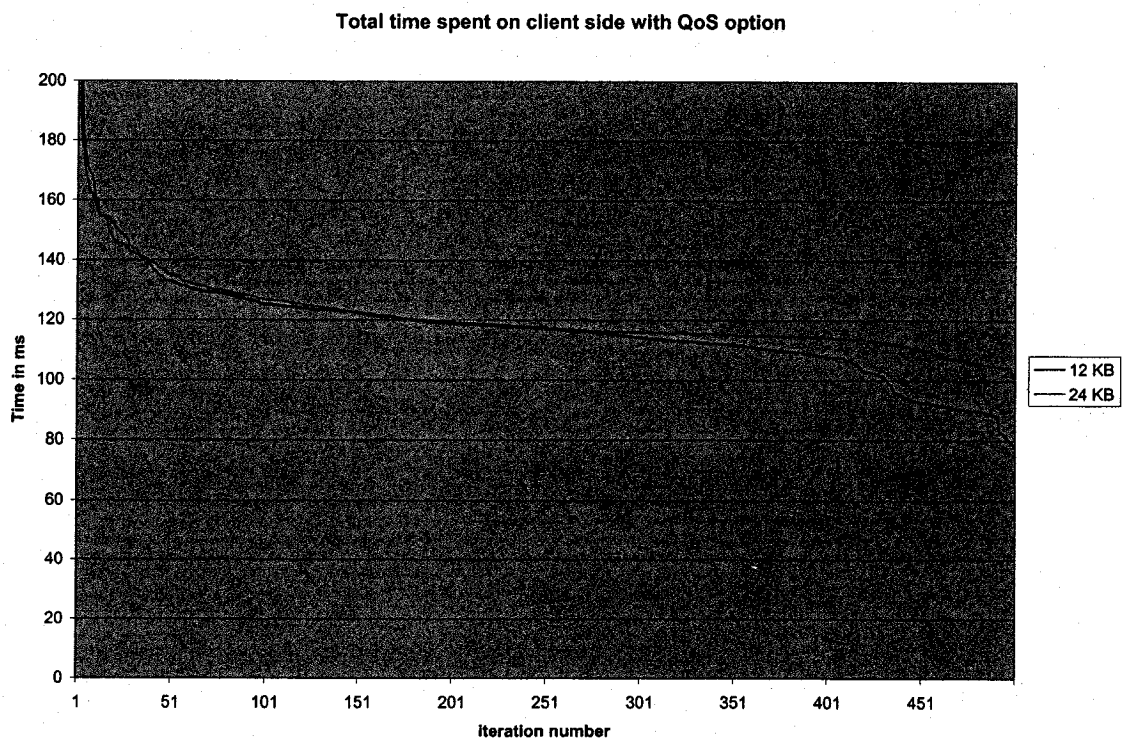

Figure 26: Total Times Spent on Client Side with Different Buffer Sizes

Figure 26 shows that the client side is unaffected by the change in the buffer size.

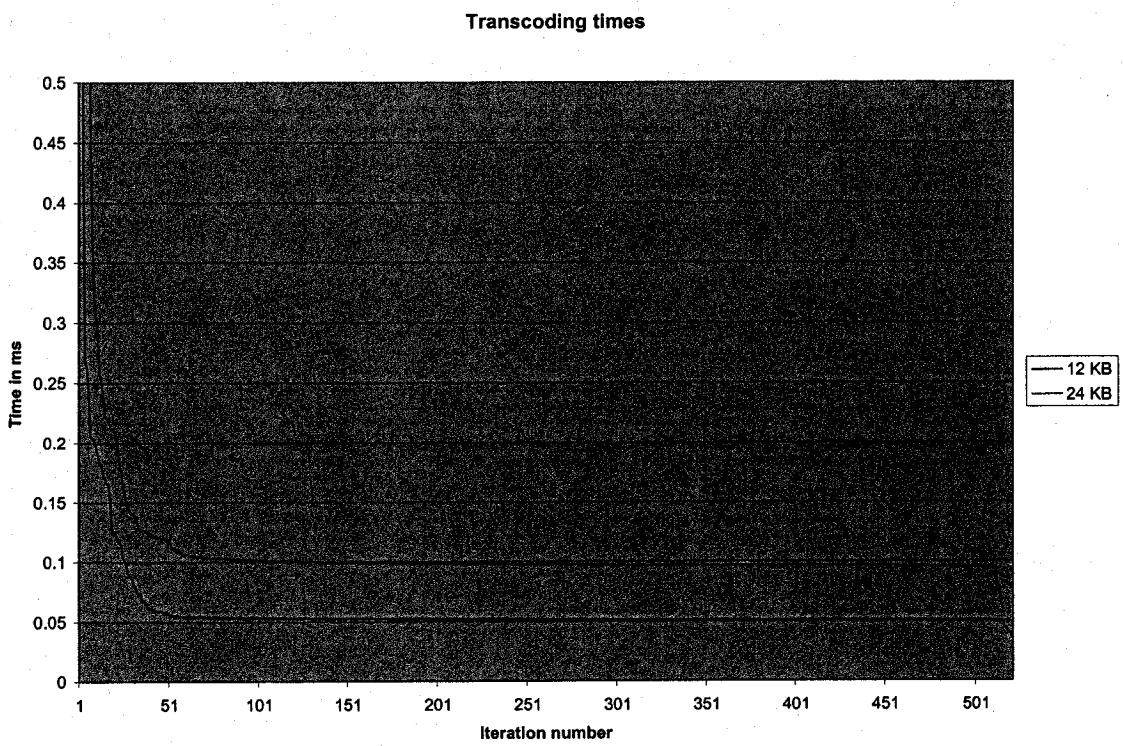

Figure 27: Transcoding Times for Different Buffer Sizes

As expected, Figure 27 shows that it takes half the time to transcode a buffer that is half the size. The efficiency is therefore not affected since it is twice as fast, but only half the amount of data is being transcoded. But if we look at Figure 28, the second-level watermarking seems to have acquired a more chaotic behaviour, and has not increased in performance even if the data being used was 50% less in size.

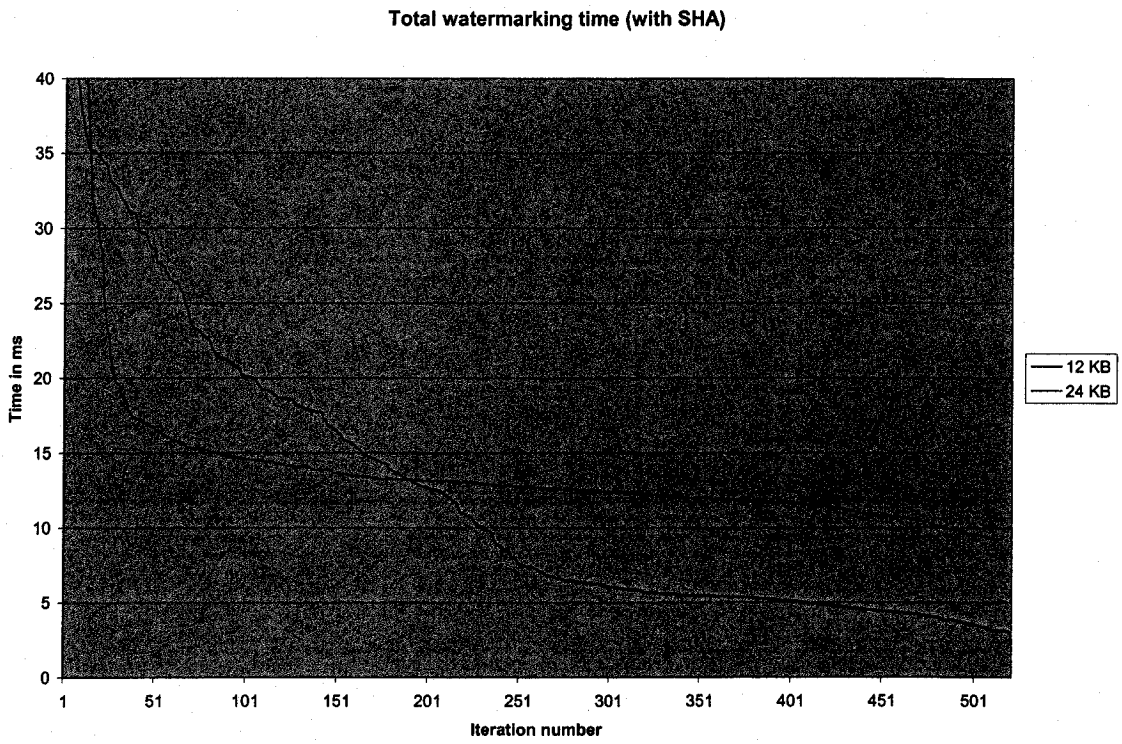

Figure 28: Effect of Buffer Size on Watermarking Time

This behaviour is also expected since the message digest is designed to output the same seamlessly random digest regardless of the amount of data that it is being fed.

These measurements show that decreasing the buffer size would decrease proportionally the transcoding time, but would not decrease the second-level watermarking time, making the system less efficient. Therefore, increasing the buffer size makes the system more efficient. The boundary of the buffer size is then drawn by external factors such as the data transfer limits in the network and the efficiency of the data transfer and the network. The designed system is best with the biggest buffer size available by the network connection between the client and the server.

6.3.4 Performance Evaluation Conclusion and Improvements.

Our system is capable of behaving as a soft real-time system on an average user computer without any special settings. The only limitation for this conclusion is the fact that the tests were done on a small, dedicated LAN with very high network connectivity and low jitter and delay.

As the number of connections increase, the TCP/IP 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.

6.4 Limitations

One limitation of the system was found while developing the experimental prototype. Although the private watermark (first-level watermarking) is only known by the party who embedded it, it needs to be of a certain known format because the third-level watermarking watermarks it with a black square using a specific watermarking method that needs to know the format and that works only if the format of the image to watermark is part of a certain class of formats. To solve this problem, we adopted a default format to be used for the private watermark images. This way the third-level watermarking can be done on any private watermark as long as it follows the default format. It is possible to use other images with different formats, but in this case the new format should be specified at both server and client machines. Moreover, although different formats can be used, they still need to be from a very tight class: the image cannot be compressed. In other words, the third-level watermarking, using the adopted algorithms, works with bitmap images (tested with 2bits, 4bits, 16bits, 24bits, 32bits, and 64bits per sample) but does not work with jpeg, gif, tiff images, or any other compressed format.

Another limitation is the size of the watermark used for the QoS assessment. In fact, if we were to try to embed a large image, the buffer sizes used would not be big enough to hold the watermark, resulting in half-embedded watermarks. We should note that (as mentioned in the transcoding section) buffers can be made larger only to a certain point where the system will lose its real-time ability. We should also keep in mind that if we are using a private key in order to privatize the QoS, the same watermark would need more space to be embedded because it would not reside at every sample, but at randomly selected samples. Therefore, sometimes with certain types of private keys, the audio data will not be enough to embed the entire watermark, resulting in the same effect mentioned above: a half-embedded watermark (see Figure 30) instead of a fully embedded watermark (see Figure 29). However, the integrity of the half-embedded watermark and the audio data would still be verifiable. But the QoS of the system would show a decrease. The watermark from Figure 30 was extracted with a 44% QoS indication, but with no flags raised for a breach of integrity.

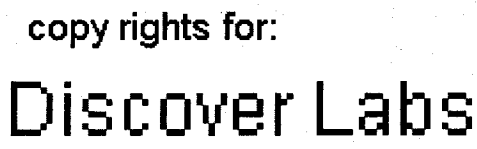

Figure 29: Selected Image for Embedding

Figure 30: Partially Embedded Watermark

On the other hand, the experimental testing was done in a LAN environment of 15 computers, which is a very tight and noise-free environment. If this system is deployed in a noisy environment, loss of bits will result in flags that would be raised constantly, indicating the violation of the integrity of the data. The functionality of the system would be paralyzed since the watermarks would not be able to be verified. However, since the system uses a lot of redundancy by watermarking every buffer, one unaffected buffer

would be enough for the validity of the watermark to be asserted. The values of the QoS module would be very useful to track the noisiness of the environment.

7. Conclusion and Future Work

During audio communication, many of the devices receiving the audio can either be underutilized (e.g., a device has better capabilities) or overloaded (e.g., an audio signal's resolution is too high). There can be cases where users do not get what they long for, or on the contrary, have more than what their device can handle. Users have to accept the audio quality as it is offered to them without knowing if it is actually what they are offered by their service provider or if what they are receiving is a corrupted version of the stream. Copyright and integrity are the other issues that are important in these applications.

This work suggested a way to solve these problems without having to change the original audio at the source, and without having to change the way the audio is captured or played back. The designed system is a transparent system that works ubiquitously on generically captured audio and delivers generically playable audio. The user's experience is not changed in any way at any time. The method suggested in this paper buffers audio streams and transcodes the streams in a divide-and-conquer approach by attacking each buffer and transcoding it according to the needs of the receiver. Every buffer is then subject to a novel three-level watermarking method that relies on the most effective watermarking techniques to deliver a privately watermarked audio and to track the integrity of the data as well as the QoS of the audio streaming itself. The successful image and message digest embedding in an audio stream as watermarks as well as a lower layer image embedding within the watermark itself make this system to be the first of its kind. In fact, any user can extract an image out of an audio stream, and any publisher can embed an image into an audio stream. They can also make sure that the image is only changed by the authorized people and can be viewed only by authorized people due to the second watermark put in parallel with the image that depends on both the image and the audio stream. A successful quality tracking of the audio is achieved with yet another image watermark that can be extracted from the first watermark.

This way users not only receive authentic and watermarked audio that is tailored to their playback needs, but also are able to know whether the quality of the received audio corresponds to what they are supposed to receive. Applications of this framework can extend from average online music listeners to secure military communications and police audio evidence that must be proven authentic in a court of law.

This system's design and successful implementation demonstrated good results which can lead to other future work. The transcoding module currently takes the needed parameters for transcoding as inputs from the receiver side. Future work can be done to make the transcoding totally pervasive by letting it detect the device's characteristics and decide by itself on the necessary playback format. More work can also be done on extending the system to work with other compressed audio formats like MP3s. In the latter 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 an MP3 audio format. The three-level watermarking framework proposed can still be used, but different watermarking techniques would need to be adopted. Any future work will focus 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. On the other hand, future work can also be done on improving and widening the choice window of the private watermark images, as well as finding a solution to avoid the need to specify the format of the private watermark for the third-level watermarking. Users can therefore embed jpeg or gif images without the need to decompress them first. Last but not least, users are expected to have their private key already. There are many private key exchange algorithms (such as Diffie-Hellman) and we will look into adding a secure key exchange algorithm suitable to the proposed framework and its application scenarios.

References

[1] MPEG-7 Content Description for Universal Multimedia Access, ISO/IEC JTC1/SC29/WG11/M4749, MPEG99, Vancouver, BC July 1999.

[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] Yu Chen, Xing Xie, Wei-Ying Ma, & Hong-Jiang Zhang. "Detecting web page structure for adaptive viewing on small form factor devices", Internet Computing, IEEE, 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 & 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, September 2004.

[10] R. Tachibana. "Audio watermarking for live performance", *Security and Watermarking of Multimedia Contents V*, vol. 5020 of Proceedings of SPIE, pp. 32–43, Santa Clara, California, 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, December 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.

[15] Changsheng Xu, Yongwei Zhu, David Dagan Feng , *"Digital audio watermarking based-on multiple-bit hopping and human auditory system"*, International Multimedia Conference; Vol. 9. Proceedings of the ninth ACM international conference on Multimedia Pages: 568 – 571., Ottawa, Canada, 2001

[16] David Megias, Jordi Herrara-Joancomarti, & Julia Minguillon Alfonso. "A Robust Audio Watermarking Scheme Based on MPEG 1 Layer 3 Compression", Seventh IFIP Conference on Communications and Multimedia Security CMS2003, October 2-3, 2003, Turin, Italy.

[17] Rainer Bohme, & Andreas Westfeld. "Statistical Characterisation of MP3 Encoders for Steganalysis", ACM MM&Sec'04, September 2004, Magdeburg, Germany.

[18] David Megias, J. Herrera-Joancomarti, & Julia Minguillon. "An Audio Watermarking Scheme Robust Against Stereo Attacks", ACM MM&Sec'04, September 2004, Magdeburg, Germany.

[19] R. Tachibana, S. Shimizu, S. Kobayashi, & T. Nakamura. Tokyo Research Laboratory IBM Japan, "An audio watermarking method robust to time and frequency fluctuation."

[20] Michael Arnold. "Attacks on Digital Audio Watermarks and Countermeasures", proceedings of the Third International Conference WEB Delivering of Music (WEDELMUSIC'03), IEEE, 2003.

[21] Libin Cai, & Jiying Zhao. "Audio Quality Measurement by Using Digital Watermarking", CCECE 2004 – CCGEI 2004, IEEE, Niagara Falls, Canada May 2004.

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

[23] Sha Wang, Dong Zheng, Jiying Zhao, Wa James Tam, and Filippo Speranza, *"An Accurate Method for Image Quality Evaluation Using Digital Watermarking"*, IEICE Electronics Express (ELEX), Vol.2, No.20, pp.523-529, October 2005.