Audio Watermark Attacks: From Single to Profile Attacks

Andreas Lang[1], Jana Dittmann[1], Ryan Spring[2], Claus Vielhauer[1]
Otto-von-Guericke University of Magdeburg
Advanced Multimedia and Security Lab
Magdeburg, Germany
[1]{andreas.lang, jana.dittmann, claus.vielhauer}@iti.cs.uni-magdeburg.de
[2]ryan.spring@student.uni-magdeburg.de

ABSTRACT

A wide range of watermarking evaluation approaches and especially image benchmarking suites have been described in the literature. Our paper sets the main focus on the evaluation of digital audio watermarking with StirMark Benchmark for Audio (SMBA). Here we describe the currently implemented single geometric attacks in detail and introduce our so-called attack profiles. Profiles reflect an application oriented point of view ranging from the normal usage of audio content like internet radio or music shops up to typical attacker scenarios. In particular we present a definition of an extended profile which is composed of three basic profiles specific for annotation watermarks. Furthermore, we demonstrate how SMBA attacks can be used to evaluate the transparency of digital watermarking algorithms regarding the embedding strength. Test results based on an example audio watermarking algorithm and the measurement of transparency and capacity are presented.

Keywords: digital watermarking, audio, attack, stirmark, smba

1. MOTIVATION

The field of image watermarking covers a wide variety of attacks, for example in [8] are three main categories suggested: JPEG compression, geometric transformation and enhancement techniques. The JPEG compression performs a lossy compression on the image signal which attacks an embedded watermark. The category of geometric transformation changes for example the rotation, crops an image part, scales the image or makes a random geometric distortion (the StirMark attack). Enhanced attack techniques include for example low pass filtering, sharpening, noise addition or print-scanning like a digital to analog conversion. Recent publications can be classified into four large categories: [9] distinguishes between removal attacks, geometrical attacks, cryptographic attacks and protocol attacks. Removal attacks try to estimate the watermark (for example by building statistical models) and filtering to remove the watermark itself. Geometrical attacks try to either destroy it or prevent its detection. Both the removal and the geometrical attack are mostly aimed at the robustness of the watermark. Cryptographic attacks cover for example attacks to find the secret key, to perform coalition or collusion attacks. As summarized in [13], attacks in the last group, the protocol attacks, do neither aim at destroying the embedded information nor at preventing the detection of the embedded information (deactivation of the watermark).
Most actual benchmarking approaches already contain a variety of attacks and allow combinations of removal and geometrical attacks. Main challenges are to find the appropriate attack strength and relevant attack combinations depending on the application and the security goals. The attackers can have a different motivation and different strategy to attack the watermark. We distinguish between two types of attackers. The first type of an attackers (α), use a single specific or a combination of audio signal modifications to destroy the watermark or to confuse the detection process explicitly. These attackers are malicious and can be classified as powerful. The other type of attackers (β) use the audio signal in a normal environment, they produce single or combined a audio signal modifications implicitly without the goal to destroy the watermark (for example lossy compression or DA/AD). These attackers are non-malicious and they are not interested in attacking the digital audio watermark explicitly.

In our paper, we summarize and classify the current *SMFA* attacks for audio which could be used by both types of attackers (α and β). Based on the single attacks we design application oriented attack profiles. The paper is organized as follows:
The StirMark Benchmark for Audio (SMBA) tool contains currently a set of 39 single geometric attacks, described in detail in section 2. Section 3 describes profile attacks and discusses profile applications. Furthermore, the extended annotation profile and the assigned basic profiles are introduced and discussed. Section 4 introduces the test results regarding the basic profiles transparency and capacity measurement for an example watermarking algorithms. We summarize the paper in section 5 with conclusions and directions for further work.

MM&Sec'05, August 01-02, 2005, New York, USA.

2. SMBA ARCHITECTURE AND SINGLE ATTACKS

This section introduces briefly the SMBA architecture and the concept of single attacks. Our general design goal was to simulate the most relevant types of signal modifications known from the audio editing tools and audio production. These modifications could be used from our both types of attacker α and $beta$, as malicious and non-malicious attacks.

Note, that in this paper we use the following notation: SMBA stands for the overall benchmarking system and SMFA denotes the single attack module. The architecture of SMBA consists of four different types of modules. First, the attack module *SMFA* itself, second the *read_write* stream module to convert audio files into streams and back into files, which is needed for input and output of audio signals. The third module *SM-Bell* is a wrapper for *SMFA* and *read_write* to make it easier to use. The fourth module *SM-Bell_GUI* is a graphical user interface for *SM-Bell*. Figure 1 shows the modules and the module dependencies. The line between the *read_write* process and *SMFA* and the other *SMFA* processes is a symbol for a pipe. The audio file is read by the *read_write* module. The audio data are given to the first *SMFA* process which runs the first attack. The resulting audio signal of this process is the new input for the second *SMFA* process and so on. At the end of the pipes can be the *read_write* module to save the audio signal in an audio file. If the user does not want to store the audio signal in a file, then the audio stream can be sent to the sound device to play it.

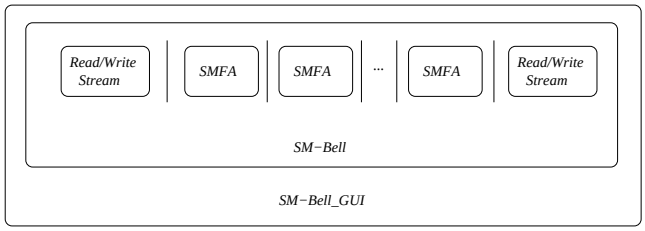

Figure 1: Modules of SMBA

The main benchmarking process is *SMFA*, the attack process, where a selected single attacks is running (or concatenations of single attacks) to change the audio signal. In this paper, we set our main focus on this module (*SMFA*).
Due to the design of our system, by running one single process, it is only possible to run one single attack on the audio signal. If it is feasible to run more than one attack on the audio signal, it is simple to use multiple instances of *SMFA*. This is possible by connecting the *stdout* of one *SMFA* process with the *stdin* of the following *SMFA* process by using a pipe. Another advantage of using *SMFA* for multiple attacks is, that each attack runs in its own *SMFA* process and the operating system can independently allocate each process to a processing unit. This is especially advantageous if there are multiple processing units available.

The following subsection classifies and introduces the single attacks of the *SMFA* module in more detail.

2.1 Current single attacks in SMFA

In this subsection, we introduce the single attacks of *SMFA*. In particular, we show the available attack functions in detail and classify these attacks into three different attack classes.

From the overall point of view, a digital audio signal S depends on different parameters based on the capturing and sampling processes (with the following default values for SMBA):

- Sampling Frequency: $f_{SR} = 44.1$ kHz
- Sampling Quantization: 16 bits ($MaxQanti-zation = 2^{16}$)
- Number of channels: 2 (stereo)

Based on the digital audio representation, we differ between time and frequency domain. The frequency domain representation can be provided by transforming the time domain audio signal into the frequency domain for example by using a Fourier transformation [1]. As notation for the attacks of *SMFA* working in time domain, we use $S_i(x)$ as input signal for *SMFA* (marked audio signal) and $S_o(x)$ as output signal from *SMFA* which is the attacked, modified, marked audio signal. The value x is the sample value at a discrete point of time in the input and output stream, we use $x = x(t_i)$. The value f denotes one frame that contains n sample values x, $f = \{x(t_0), x(t_1), \cdots, x(t_{n-1})\}$, $t_i, i = 0 \ldots n-1$, $n \in \mathbb{N}$, n=framelength. $Avg(f)$ is the average of all sample values for all channels within the frame f. As notation for the attacks of *SMFA* working in frequency domain, we use $F_i(x)$ to identify the frequency input signal and $P_i(x)$ to identify the phase of the signal represented in the frequency domain. Furthermore, we use $F_o(x)$ and $P_o(x)$ as the corresponding output signal in frequency domain. The attacks in the frequency domain are parameterized with a window size ($FFTSize$) equal to frame size f for the Fast Fourier Transformation used for processing. As a default value we use a size of 1024 samples. To scale the attacks, we introduce an attack strength scalar value A_S which specifies the strength of the attack performed on the audio signal.

The motivation for all attacks in *SMFA* is to destroy or weaken the embedded watermark signal, as Kutter et. all [9] described for geometric attacks. From the signal processing point of view, we can classify the *SMFA* attacks into three attack classes. The first class adds or removes a signal k to or from $S_i(x)$: $S_o(x) = a * S_i(x) + b * k(t)$. The value a scales the input audio signal and the value b scales $k(t)$ to a specific limit. The second class can be described as filter attacks: $S_o(x) = F_{Attack}(S_i(x))$, where F_{Attack} is the corresponding attack from this attack class. The third attack class can be seen as modification attacks, by modifying the overall structure of the signal representation for example the overall length of the audio signal or shifting audio samples: $S_o(x) = M_{Attack}(S_i(x))$. Table 1 summarizes all current single attacks of *SMFA* into these three classes by indicating time and frequency domain.

In the following, we describe the single attacks of the three attack classes in more detail. The indicated range of the parameters is the normal used parameter range, derived from

Table 1: Classification of *SMFA* attacks

Add/Remove Attacks	Domain	Filter Attacks	Domain	Modification Attacks	Domain
$S_o(x) = a * S_i(x) + b * k(t)$		$S_o(x) = F_{Attack}(S_i(x))$		$S_o(x) = M_{Attack}(S_i(x))$	
AddBrumm	Time	Amplify	Time	Invert	Time
AddSinus	Time	Normalizer1	Time	FFT_Invert	Frequency
AddNoise	Time	Normalizer2	Frequency	CopySample	Time
AddDynNoise	Time	Compressor	Time	FlippSample	Time
AddFFTNoise	Frequency	BassBoost	Time	CutSample	Time
NoiseMax	Time	RC-HighPass	Time	ZeroCross	Time
Denoise	Time	RC-LowPass	Time	ZeroLength1	Time
LSBZero	Time	FFT_HLPassQuick	Frequency	ZeroLength2	Time
Echo	Time	Stat1	Time	ZeroRemove	Time
		Stat2	Time	PitchScale	Frequency
		FFT_Stat1	Frequency	DynamicPitchScale	Frequency
		Smooth1	Time	TimeStretch	Frequency
		Smooth2	Time	DynamicTimeStretch	Frequency
				Exchange	Time
				Resampling	Time
				ExtraStereo	Time
				VoiceRemove	Time

empirical tests. Of course, there are values out of range possible, but it makes no sense because of the disturbed audio output signal.

2.1.1 Add/Remove Attacks

The attack *AddBrumm* adds a buzz as sinus tone to $S_i(x)$ to simulate the buzz of a power supply unit of analog devices. The distortion signal can be described as $k(t) = \sin(2 * \Pi * Frequency)$, $b = A_S$, $a = 1$ and $Frequency$ is the frequency of the buzz sound in range of $\left[1, \frac{f_{SR}}{2}\right]$. As default value, we chose $Frequency = 55$ Hz.

In contrast, *AddSinus* works with a different frequency ($Frequency = 3000$ Hz): it adds a sinus tone to $S_i(x)$ in a higher frequency band. This attack is useful to simulate various sinusoidal audio phenomena or to add a distortion signal in exactly the same frequency range like the watermark embedder.

The attacks *AddNoise, AddDynNoise, AddFFTNoise* and *NoiseMax* add a noise to $S_i(x)$. *AddNoise* adds a white noise generated from a pseudo noise random generator (PN). The attack can be described as $k(t) = PN(t) \bmod A_S$, $a = \left(1 - \frac{A_S}{MaxQuantization}\right)$ and $b = 1$. The strength of this attack is constant for the whole duration and specified by the attack parameter. The attack *AddDynNoise* is similar to *AddNoise*. The difference is, that *AddDynNoise* adds a dynamic noise, generated from PN that depends on the signal amplitude of $S_i(x)$. This attack is useful for simulating certain noise effects found in analog audio equipment and can be described as $k(t) = PN(t) \bmod \left(\left|S_i(x) * \frac{A_S}{100}\right| + 1\right)$ where t is the time, which controls PN and $a = 1, b = 1$. The *AddFFTNoise* is similar to the *AddNoise* attack by changing the domain from time to frequency domain: $F_o(x) = F_i(x) + PN(t) \bmod (A_S)$. The attack *NoiseMax* adds a Maximum Length Sequence (MLS) based on [14] to $S_i(x)$: $k(t) = MLS(Mask, Length)$, $b = A_S$ with $Mask = 23$ and $Length = 1365$. In this class of attacks, we plan to add attacks, which use alternative noise models like pink or brown noise.

The *Denoise* attack is the opposite of *AddNoise* and removes noise from $S_i(x)$.

LSBZero removes the least significant bit plane by setting all least significant bits (LSB) to zero: $S_o(x) = S_i(x) \wedge (MaxQuantitization() - 1)$.

Echo adds an echo to $S_i(x)$. The echo produces an echo delay of *Distance* samples and adds them with half of loudness: $a = 2$, $b = 2$, $k(t) = S_i(x + Distance)$. The *Distance* can be in a range $[1, x_n]$ where x_n is the last sample value in $S_i(x)$. a and b set the loudness of the echo and $\frac{1}{a} + \frac{1}{b} \leqq 1$ to avoid distortion. This attack meant to test watermarking algorithms that hide information in echos in the signal or test the watermark against echo attacks.

2.1.2 Filter Attacks

All filter attacks perform a kind of filtering operation $F_{Attack}(S_i(x))$ depending on the attack itself and on the audio signal $S_i(x)$.

The *Amplify* attack increases or decreases the amplitude of the audio signal to simulate an amplification: $S_o(x) = S_i(x) * \frac{A_S}{100}$. A_S is the percent increase or decrease percentage with 100 being neutral (default $A_S = 50$).

The *Normailzer1* and *Normalizer2* change also the amplitude, but introduce a normalization. *Normalizer1* normalizes $S_i(x)$ based on local or cumulative peak information and simulates various real-time normalizer devices in time domain. The value *Level* is the quantitization step that the samples are to be normalized to (in a range of $[1, MaxQuantization]$ with a default value of $Level = 28000$ for a 16 bit audio signal) and *Zeroing* is a boolean switch to switch between local and cumulative peak finding (0 = local (default), 1 = cumulative). The function $Peak(x)$ finds the

maximum peak value in the buffer of the audio signal and $PreviousPeak$ specifies the found maximum from all buffers before: $S_o(x) = S_i(x) * \frac{Level}{Peak(S_i(x_j), Zeroing, PreviousPeak)}$ and $j = 0, 1, \ldots, BufferSize$.
The attack *Normalizer2* is similar to *Normalizer1*, but works in the frequency domain. Additionally, the audio signal is split into two frequency bands, which are normalized independently of each other and combined afterwards.

The *Compressor* attack scales all samples over a given threshold by a given amount. This attack is similar to various analog compressor devices that are available on the market

$$S_o(x) = \begin{cases} S_i(x) & S_i(x) \leq (-ThresholdDB) \\ S_i(x) * CV & S_i(x) > (-ThresholdDB) \end{cases}$$

and $CV = CompressorValue$ specifies the amount of increasing (or decreasing) the sample values (default $CV = 2.1$) and $ThresholdDB$ specifies the threshold, which defines the attenuation of the maximum possible signal whenever the compressor amplification works (default 6.1 dB). If the CompressorValue CV is less than 1, this attack works like a compressor. Otherwise, if CV is larger as 1, then this attack works like an expander.

The attack *BassBoost* boosts the bass range of the audio signal, similar to analog bass boosting circuits that are prevalent in consumer audio devices depending on a given frequency threshold. This attack uses code from the Open-Source Media Editor Audacity [20] to compute the coefficients of the used infinite impulse response filter (IIR).

There are three attacks (*RC-HighPass, RC-LowPass* and *FFT_HLPassQuick*), which filter out a special frequency selected by the attack parameters.
RC-HighPass and *RC-LowPass* simulate an analog Resistor/Capacitor (RC) Pass filter working in time domain and *FFT_HLPassQuick* works in frequency domain. The algorithm to compute *RC-HighPass* comes from [18] and is given as follows: $S_o(x) = A_0 * S_i(x) + B * S_o(x-1) + A_1 * S_i(x-1)$ and $B = e^{\frac{-2\pi * Threshold}{SampleRate}}$ and $A_0 = \frac{1+B}{2}$, $A_1 = -\frac{1+B}{2}$ where $Threshold$ specifies the high pass frequency threshold and is in range of $\left[1, \frac{f_{SR}}{2}\right]$ with a default of $Threshold = 150$.
The *RC-LowPass* attack is similar to *RC-HighPass*, but filters out low frequencies from $S_i(x)$ by computing $S_o(x) = A * S_i(x) + B * S_o(x-1)$ and $B = e^{\frac{-2\pi * Threshold}{SampleRate}}$ and $A = 1 - B$. For this attack, the default value of $Threshold$ is 15000.
The attack *FFT_HLPassQuick* has a sheer flank on the frequency threshold like the other two attacks:

$$F_o(x) = \begin{cases} F_i(x) & F_i(x) < LowPassFrequency \\ F_i(x) & F_i(x) > HighPassFrequency \\ 0 & \text{otherwise} \end{cases}$$

The three attacks (*Stat1, Stat2* and *FFT_Stat1*) apply a statistical average distortion depending on the current and neighbor sample values of $S_i(x)$.
For *Stat1* the current sample value depends on itself and the following sample value: $S_o(x) = \frac{S_i(x)+S_i(x+1)}{2}$. The *Stat2* attack computes the current sample value depending on the following and predecessor sample value with a given priority: $S_o(x) = \frac{S_i(x-1)+3*S_i(x)+S_i(x+1)}{5}$. The attack *FFT_Stat1* is similar to *Stat1* but works in frequency domain: $F_o(x) = \frac{F_i(x-1)+F_i(x+1)}{2}$.

The two attacks *Smooth1* and *Smooth2* apply a simple smoothing to $S_i(x)$ depending only from the neighbors of the current sample value. For *Smooth1*, the computation is

$$S_o(x) = \begin{cases} \frac{S_i(x-1)+S_i(x+1)}{2} & S_i(x-1), S_i(x+1) > S_i(x) \\ \frac{S_i(x-1)+S_i(x+1)}{2} & S_i(x-1), S_i(x+1) < S_i(x) \\ S_i(x) & \text{otherwise} \end{cases}$$

where both neighbors of $S_i(x)$ ($S_i(x-1)$ and $S_i(x+1)$) have to be higher or lower than $S_i(x)$ to perform the attack. Otherwise, the attack *Smooth2* is defined as:

$$S_o(x) = \begin{cases} \frac{S_i(x-1)+S_i(x+1)}{2} & S_i(x-1) < S_i(x) < S_i(x+1) \\ \frac{S_i(x-1)+S_i(x+1)}{2} & S_i(x-1) > S_i(x) > S_i(x+1) \\ S_i(x) & \text{otherwise} \end{cases}$$

where one neighbor of $S_i(x)$ ($S_i(x-1)$ or $S_i(x+1)$) is higher and the other one is lower than $S_i(x)$ to perform the attack.

2.1.3 Modification Attacks

All modification attacks perform a modification operation $M_{Attack}(S_i(x))$ depending on the audio signal $S_i(x)$. The general structure of the audio signal is changed after performing an attack from this attack class.

The attack *Invert*, inverts all sample values in time domain by replacing it with its opposite (phase shift of 180°): $S_o(x) = -S_i(x)$.

The same does the attack *FFT_Invert*, but in frequency domain. *FFT_Invert* inverts (phase shift 180°) the frequencies $F_o(x) = -F_i(x)$ and phases $P_o(x) = -P_i(x)$. Both attacks (*Invert* and *FFT_Invert*) evaluate the watermarking algorithm sensitivity to the right phase of the audio signal.

The *CopySample* attack inserts copies of a group of samples at a later point in $S_i(x)$ in the time domain. The goal is to test the watermarking algorithm dependency on quantity and order of samples. The value C is the number of sample values, which are copied (range $[1, lengthof(S_i(x))]$ with default $C = 2000$). DI specifies the distance between the copy and paste point in the audio signal and is in range of $[C, lengthof(S_i(x))]$ and the default value $DI = 6000$. The value $Period$ defines the distance between two copies in the audio signal and is in range of $[DI, lengthof(S_i(x))]$ with a default value of $Period = 10000$.

$$S_o(x) = \begin{cases} S_i(z) & 0 \leq z < DI \\ S_i(z - DI) & DI \leq z < DI + C \\ S_i(z - C) & DI + C \leq z < Period + C \end{cases}$$

and $z = x \bmod (Period + C)$. The length of the audio signal increases by using this attack.

The *FlippSample* attack flips a group of samples at a later point in time of $S_i(x)$. This is meant to test watermarking algorithms for dependance on the order of sample values.

$$S_o(x) = \begin{cases} S_i(z + DI) & 0 \leq z < Count \\ S_i(z) & Count \leq z < DI \\ S_i(z - DI) & Distance \leq z < DI + Count \\ S_i(z) & DI + Count \leq z < Period \end{cases}$$

and $DI = Distance$ and $z = x \bmod (Period + Count)$. By using this attack the length of $S_i(x)$ is equal to the length

of $S_o(x)$. The range of the parameter values and the default parameters are the same like in the case of *CopySample*.

The *CutSample* attack drops a certain number of samples periodically in the time domain. The value $RemoveNumber$ specifies the number of sample values, which are droped every $Remove$ distances. The range of $RemoveNumber$ is $[1, lengthof(S_i(x))]$ and has a default value of 7. The parameter $Remove$ has a range of $[RemoveNumber, lengthof(S_i(x))]$ and a default value of 1000. $S_o(x) = S_i(z+RemoveNumber)$ and $z = x \bmod (Remove - RemoveNumber)$. This attack decreases the length of $S_i(x)$ depending on the attack parameters.

The *ZeroCross* attack sets all samples with an absolute value less than the given threshold $ZeroCross$ to zero. The range of $ZeroCross$ is $[1, MaxQuantization]$ and has a default value of 1000.

$$S_o(x) = \begin{cases} 0 & |S_i(x)| < ZeroCross \\ S_i(x) & \text{otherwise} \end{cases}$$

The *ZeroLength1* attack sends a given number of zero samples as output after the detection of a zero value in the input, treating each channel independently. The value Z specifies the number of sample values, which are intermediated at the detection point (range $[1, lengthof(S_i(x))]$ and default $Z = 10$):

$$S_o(x) = \begin{cases} 0 = S_o(x+1) = \ldots = S_o(x+Z) & S_i(x) = 0 \\ S_i(x) & \text{otherwise} \end{cases}$$

The *ZeroLength2* attack is similar to *ZeroLength1* but sends a given number of zero samples to all channels after the detection of a zero value in any channel of $S_i(x)$. The value Z specifies the number of intermediated sample values and has the same range and default value like *ZeroLength1*:

$$S_o(x) = \begin{cases} 0 = S_o(x+1) = \ldots = S_o(x+Z) & S_i(x) = 0 \\ S_i(x) & \text{otherwise} \end{cases}$$

Both attacks (*ZeroLength1* and *ZeroLength2*) increase the length of the audio signal.

The *ZeroRemoves* attack removes all zero samples from $S_i(x)$ and is the opposite to *ZeroLength1* and *ZeroLength2*.

$$S_o(x) = \begin{cases} S_i(x) & S_i(x) \neq 0 \\ S_i(x+1) & S_i(x) = 0 \end{cases}$$

The length of the audio signal is decreased.

The *PitchScale* attack scales the frequency linear up or down without changing the tempo of the audio signal. This attack uses the library SoundTouch [16] to alter the signal.

Instead of the linear pitch scaling of *PitchScale*, the attack *DynamicPitchScale* scales the frequency nonlinear up or down by using the library SoundTouch [16]. This attack has 5 different modes [11], where the type and form of nonlinearity is specified.

The *TimeStretch* attack stretches or shrinks the signal playing time linear without changing the pitch. This attack uses the library SoundTouch [16] to alter the signal.

Instead of the linear time stretching attack *TimeStretch*, the *DynamicTimeScale* attack stretches the time of $S_i(x)$ nonlinear without changing the pitch. This attack has 5 different modes [11], where the type and form of nonlinearity is specified.

The *Exchange* attack swaps consecutive samples in the audio signal to test watermarking algorithms for their sensitivity to the exact order of the samples.

$$S_o(x) = \begin{cases} S_i(x+1) & z = 0 \\ S_i(x-1) & z = 1 \end{cases}$$

Where $z = x \mod 2$.

The *Resampling* attack adjusts the sample rate f_{SR} to a new sample rate. This attack uses the libsamplerate library [15] to compute the sample values regarding the changed f_{SR} (default $f_{SR} = 22050$).

The attack *ExtraStereo* works only for stereo audio signals. This attack adds the average between the channels in a frame to all samples in the frame. The idea is to test watermarking algorithms that use channel difference in multichannel systems. This attack is based on the extra stereo function of XMMS [22]: $S_o(x) = S_i(x) + (S_i(x) - Avg(f)) * A_S$. A_S has a default value of 20.

The opposite of the *ExtraStereo* attack is the *VoiceRemove* attack, which subtracts the average between the channels in a frame from all samples in the frame also to test watermarking algorithms that use channel difference in multi-channel systems. This attack is therefore also based on the voice remove function of XMMS [22]: $S_o(x) = S_i(x) - Avg(f) * A_S$.

By evaluating a digital audio watermark, it is possible to attack the watermark itself by using one of the single attacks in *SMFA* to weak or destroy it. The user can specify the attack and attack parameters (like strength). Another and more comfortable way is the usage of attack profiles instead of single attacks which will be discussed in the next section. The goal is to pre-define the relevant single attacks, attack parameters and attack strength and predefined application scenarios for audio watermarking.

3. PROFILE ATTACKS

This section introduces the profile attacks which are expected from attackers *alpha* and β and their classification as well as their assignment. Furthermore, we introduce three basic and one extended profile definition and select two basic profiles for our tests.

From the motivation, the most simple way to attack a digital watermark is a brute force attack by using all possible attacks against the watermark. For each attack, SMBA has default attack parameters, which can be used to evaluate the digital audio watermark very quickly. It is also possible to change and optimize the attack parameters to improve the attack strength or attack transparency. Each attack is part of a single evaluation process to determine the watermarking algorithm weakness e.g. with which attack a watermark can be broken. These single attacks are atomic signal modification processes. This scenario is also called *single attack process* [3]. In this mode, the watermarked audio file undergoes many separate instances of attacks and for each attack

a separate output audio file is produced. Each of these audio files is only modified by a single attack (e.g. AddNoise, PitchScale, Amplify or CutSample). This is useful to find a single specific weakness of a watermark algorithm. By using this attack method, the problem is twofold: Firstly, for each attack we produce and evaluate an attacked audio file to test the watermarking detection/verification computing a huge amount of data. Secondly, weaknesses caused by combinations of audio effects or artefacts are not tested.

Therefore, another attack mode, called *profile attack* is introduced by [3, 10] to run more than one attack in serial order. An evaluation profile is an ordered sequence of processes that may be applied to a signal, as shown in figure 2. Each of the individual processes in the profile is defined by its own set of parameters. While a profile may seem to be merely an attack or process macro, profiles serve a very useful purpose in benchmarking. Profiles allow the evaluation system to model or simulate scenarios of interest to particular applications, like internet radio, audio production or audio archives. Based of our classification of profiles [12], an evaluation profile may be defined in terms of other (existing) profiles, which allow a complex process or attack to be modeled as a sequence of previously-defined (or elementary) processes (for example the DA/AD conversion).

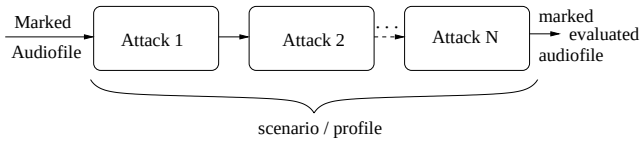

Figure 2: Evaluation Profile [12]

In [12] we defined a general profile assignment and classification, which could be used from our both types of attackers α and β.

Based on the variety of digital watermarking applications, e.g. to protect the copyrights of users, to guarantee the integrity of content or to provide additional information embedded in the media, each individual watermarking scheme has specific parameters for robustness, transparency, capacity, security and complexity. As the first two are relevant for most applications we defined in [10] and shown in figure 3 the following main profiles with their assignment of sub-profiles: High Quality Robust, Low Quality Robust, High Quality Fragile and Low Quality Fragile. For example: For an integrity watermark, we would evaluate the *Degree of Fragility* for High Quality or Low Quality requirements. For copyright protection, it is for example essential that the watermarking algorithm is robust against digital analog conversion and depending on the application scenario, we have High and Low Quality requirements.

Annotation watermarking (sometimes also called caption watermarking or illustration watermarking) is used to embed supplementary information directly in the media, so that the additional information is directly integrated and cannot get easily lost (e.g. meta data like the audio description fields). Today, we find a wide range of applications for annotation watermarking, especially also to watermark specific media objects like song sequences. As discussed in [21],

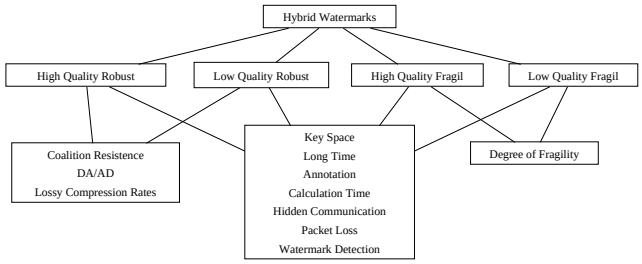

Figure 3: Assignment of Profiles [14]

in comparison to copyright watermarking we do not expect any dedicated removal, cryptographic or protocols attacks in general. As the annotated data would lose value in most cases there is no attack motivation and most security properties of the watermarking scheme have a minor relevance. Even only a limited set of expected geometric attacks play an important role, like robustness against add noise, cutting and compression seems to be the most the important one. Highly of interest are the parameters:

1. Capacity: how many bits can be spread over the whole audio signal or directly stored in the related parts within the audio.
2. Transparency: if there are noticeable artefacts in the marked media the user will probably not accept the media for professional and semi-professional purposes like publishing and presentation.

Therefore capacity and transparency of the marked media are fundamental requirements for annotation watermarks.

Requirements for annotation watermarking can be summarized in an Annotation Watermarking Scheme (AWS) based on the formalization of [21] for image annotation watermarking. AWS is called annotation watermarking scheme if, and only if

- AWS is a transparent watermarking scheme,
- AWS is (add noise, cutting, compression)-robust [21],
- AWS is a sequence of data (for audio) or object-based (for images)

Therefore, we can formalize the annotation profile based on [12] for benchmarking watermarking algorithms as follows: We define the extended annotation profile as:
$P_{E-Annotation}(S_i||S_o||parameters)$
$parameters = (algorithm||options||quality)$
The parameter *algorithm* defines the scheme used for embedding process. The parameter *options* depends on the embedding scheme and specifies the parameters for the embedding process. To define the quality of the marked signal (embedding transparency), the parameter *quality* is used. This profile is associated to the main profiles High- and Low-Quality Robust, as well as High- and Low-Quality Fragile because the profile does not touch specific properties of robust and fragile watermarking schemes.

The annotation profile itself is an "Extended Profile", because it is composed of the "Basic Profiles" *Robustness Measurement, Capacity Measurement* and *Transparency Measurement*. A basic profile provides a simple functionality, which can be a part of an extended profile or a simple measuring process. For the extended annotation profile, the associated basic profiles provide the measuring functionality as is shown in figure 4 and introduced below.

In particular, the profile *Capacity Measurement* $P_{B-Capacity\_Measurement}$ is a basic profile and needed to determine the maximum capacity for embedding a message in a signal. The transparency regarding the embedding process is not considered in this profile. It is defined as:
$P_{B-Capacity\_Measurement}(S_i||S_o||parameters)$
$parameters = (algorithm||options)$
The *algorithm* defines the watermarking algorithm used for embedding process and *options* are optional parameters which depend and are specific for the embedding scheme. Furthermore, it is important to differ between the overall and total capacity representing the watermark and the payload for the watermark information. For example, if a watermarking algorithm introduces an error correction code (ECC), the payload is less than the capacity, due to the lost space of ECC. The measurement of the capacity of a watermarking algorithm depends on the algorithm (working domain) and the parameters. For capacity measurement, there are different measure methods possible like bits per second, bits per frame, bits per frequency range or bits per frequency frame.

The profile *Transparency Measurement* $P_{B-Transparency\_Measurement}$ is a basic profile and needed to determine the transparency of a signal processing operation like an embedding process. It is defined as:
$P_{B-Transparency\_Measurement}(S_i||S_o||parameters)$
$parameters = (algorithm||options)$
The *algorithm* defines the watermarking algorithm used for embedding process and *options* are optional parameter which depends and are specific for the embedding scheme. The measurement of the transparency can be done by hearing tests (subjective measurement of transparency) or computer based measurement. To measure the audible distortion of the embedding process, we use the ITU-R Recommendation standard BS.1387 [6]. This standard provides the definition for Perceptual Evaluation of Audio Quality (PEAQ) [19] and can be used for rating of the quality of audio signal modifications, which can be performed by a watermark embedder. As implementation of the PEAQ, we use the tool EAQUAL [4], which computes the Objective Difference Grade (ODG) by means of a neuronal network. The computed ODG values are in a range from $[-4, 0]$, where -4 is the worst (very annoying) 0 is the best (imperceptible).

The profile *Robustness Measurement* $P_{B-Robustness\_Measurement}$ is a basic profile and needed to determine the robustness of a watermark. It is defined as:
$P_{B-Robustness\_Measurement}(S_i||S_o||attack)$
The parameter *attack* is defined as the set of single attacks $(A_{i1}(x_1)||A_{i2}(x_2)||\cdots||A_{in}(x_n)), n > 0$. There are all attacks from SMBA possible, but as introduced for AWS the focus can be set on add noise, cut and compress attacks.

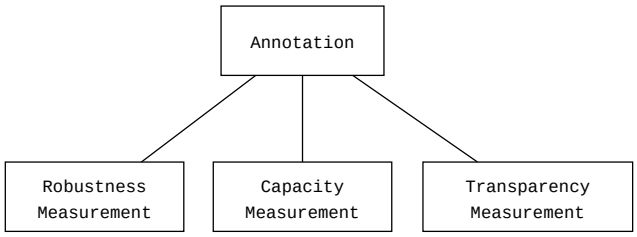

Figure 4: Annotation profile

Each of the three basic profiles, which are needed to define the extended annotation profile measures a single property of the magic watermarking triangle (robustness, capacity and transparency). These properties are in concurrency to each other and the improvement of one implies effects on the other two.

In this paper, we set our focus on this annotation profile, which is selected for implementing and evaluation of watermarking algorithms for audio signals. The following section introduces the current test set, test environment and the principle work of our implementation. Based on our definitions, we show on the example of annotation profile in the next section how a profile can be implemented and tested within SMBA.

4. TEST ENVIRONMENT FOR ANNOTATION PROFILE

In this section, we present a possible test environment for $P_{E-Annotation}()$.

The annotation profile gives users the opportunity to evaluate the embedding parameters (like strength or frequency range) of a digital audio watermark to determine the audible distortion introduced into the audio signal by a certain amount of embedded data.

As test material, we are using the well known Sound Quality Assessment Material (SQAM) audio files [17], which have a sampling frequency of $f_{SR} = 44.1$ kHz, 2 channels and a quantization of 16 bits.

In our test scenario, we set our focus on the basic profiles *Capacity Measurement* and *Transparency Measurement* to measure the embedding capacity regarding the transparency and otherwise. We suppress the output for robustness by setting the respective attack in SMBA to "none". To embed a watermark information, we use a watermarking algorithm, which is a spread spectrum watermarking algorithm working in frequency domain [5] based on the general algorithm [2, 7].

The watermarking algorithm transforms the audio input signal in the frequency domain by performing a Fourier transformation. The watermark information is spread over 128 bits and embedded in the frequency coefficients. The frequency range, which is used for embedding, are parameters of our implementation of the watermarking algorithm and ranges from 1 Hz to 22050 Hz. After embedding the information, the signal is transformed back to the time domain

to store it. For our test scenario, we decided to define the "middle" of the frequency range at 10 kHz. To increase the frequency range, we expand the frequency band symmetrically. This means for example, when we set the frequency range to 200 Hz, the watermark is embedded in the frequency range of $[9.9, 10.1]$ kHz.

The following figure 5 shows the general principle of the tests. The original audio signal S_i will be watermarked by using the watermarking algorithm. The watermark algorithm needs additional parameters $p1$ like watermarking message, embedding strength and frequency range (both effect the final watermarking transparency). From the watermarked output signal S_o the transparency will be measured by computing the ODG value. Depending on the predefined threshold $p2$, which defines the embedding transparency, the embedding parameters are modified $p'1$ and the watermark is embedded in S_i with an increased or decreased embedding strength. As we set our focus on the *Capacity Measurement* and *Transparency Measurement* and do not perform *Robustness Measurements* to demonstrate the overall profile based testing, the retrieval (detection) of the watermark information can be neglected.

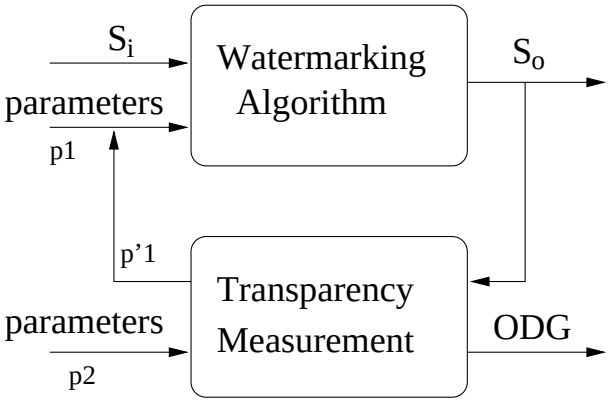

Figure 5: Test environment

We define three test scenarios to evaluate the two basic profiles *Capacity Measurement* and *Transparency Measurement*, which are like follow:
In the first scenario (a) we demonstrate the basic profile *Capacity Measurement* and measure the possible embedding frequency range for the watermarking algorithm depending on the used audio content. We defined $p2$ with an ODG threshold of -3 and the iteration step size, which increases the frequency range, by 200 Hz.
The second test scenario (b) focus also the basic profile *Capacity measurement*, use the same watermark algorithm and the same ODG threshold of -3, but each iteration cycle increases the frequency range by 2000 Hz.
The last test scenario (c) uses the same watermarking algorithm, but the frequency range is fixed for all tests and set to 1000 Hz, which means a fixed embedding capacity. The beginning of embedding frequencies is different (lower and upper frequency bound). The frequency ranges are $[1000, 2000]$, $[2000, 3000]$, $\cdots$, $[15000, 16000]$ and so on. We measure the transparency of the embedding process by computing the ODG values to perform the basic profile *Transparency Measurement*.

For all tests, we estimate an audio content depended transparency of the embedding process with a constant embedding capacity as well as an audio content depended embedding capacity with constant transparency (constant embedding strength).

In the following, we introduce our test results for our scenarios (a), (b) and (c) as described above.

Table 2 contains our test results for test scenario (a), which measures the frequency range of the watermarking algorithm by using a predefined ODG value of -3 and an increased frequency range of 200 Hz. This table shows the computed ODG values for the audio SQAM files in an increased frequency range. Depending on the audio content, the quality level of an ODG value $= -3$ has a different frequency range for different audio contents. The iteration cycle is for example for *vioo10* 21, which corresponds with a bandwidth of $[7900, 12100]$ Hz and for *bass47* 20 (bandwidth $[8000, 12000]$ Hz) to get a quality of ODG ≈ -3. For the audio file *quar48* it requires 34 iterations (bandwidth $[6600, 13400]$ Hz). Furthermore, the frequency range, which can be used to embed the watermark, is larger for the *quar48* $[6600, 13400]$ Hz) compared to *vioo10* $[7900, 12100]$ Hz) or *bass47* $[8000, 12000]$ Hz). A larger frequency range, where the watermark information is embedded, implies a higher embedding capacity. Furthermore the test results show, that different audio contents with identical watermarking algorithm and watermarking algorithm parameters distort the host audio to different degrees. Leading to varying maximum values before the distortion crosses our threshold of -3 and ends the execution. The following figure 6 visualizes the test results for all SQAM audio files, which are shown in table 2. The measuring process stopped after crossing the threshold of the ODG$=-3$.

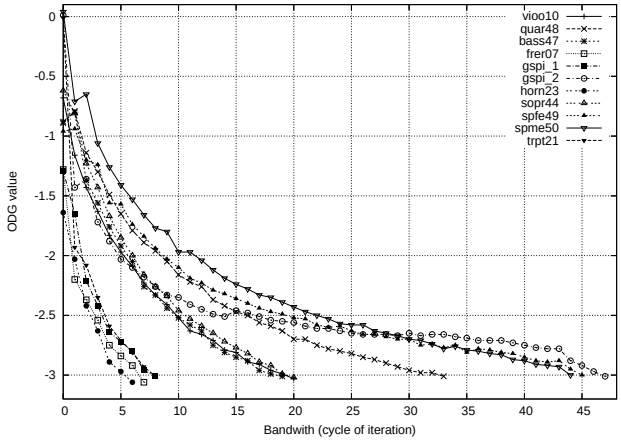

Figure 6: Iterations by a frequency range of 200 Hz

Table 3 summarizes the test results regarding the test scenario (b), which use the same watermarking algorithm with a different granularity in the iteration steps (increasing the bandwidth). The frequency range is increasing by a factor of 10 (2000 Hz) per cycle rather than the previous 200 Hz in test scenario (a). Since each cycle requires approximately

equal execution time, this can greatly decrease execution time, if an exact answer is not required. The complexity of the measurement of the transparency (ODG) regarding the capacity (frequency range) is decreased for example from 34 to 4 cycles for the audio file *quar48*. This means, that the computation time is decreased by the factor of 8.5 but the correctness of the test results is decreased too. The threshold of the measured transparency is set to ODG $=-3$ and by expanding the frequency range of 2000 Hz per cycle, the ODG value of *vioo10* is -3.24 in the 3 iteration step. The audio files *frer47, gspi_1, horn23* and *trpt21* have in the first iteration an ODG value less than -3. These audio files, which have as audio content single instruments, have a very sensitive transparency. Other audio files like *spfe49* or *spme50* need 5 iteration steps in test scenario (b) and the ODG values are -3.18 for both compared to an ODG value of -3.0 and iteration steps of about 45 in test scenario (a). Although within test scenario (b) we could reduce the overall computation time, this caused higher errors to determine the optimal frequency range. The dependence of the audio content is the same like in test scenario (a): speech is more insensible than single instruments. The following figure 7 shows the visualization of our test results based on table 3. The single instruments *frer47, gspi_1, horn23* and *trpt21* are shown on the Y-axis, because there was no second iteration step necessary, which expands the frequency range by decreasing the transparency.

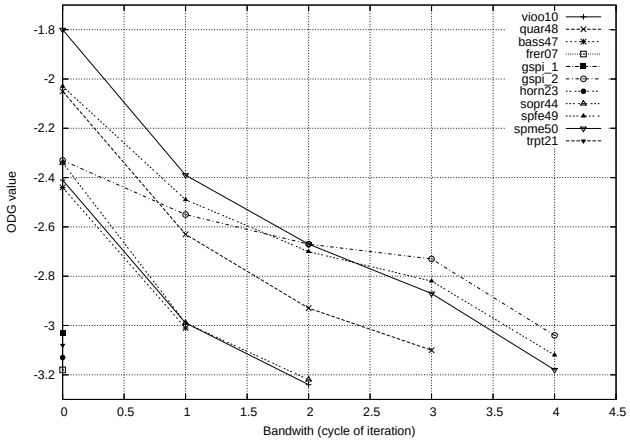

Figure 7: Iterations by a frequency range of 2000 Hz

Table 4 shows our test results for the test scenario (c), which measured the embedding transparency by a fixed embedding frequency range which implies a fixed embed capacity (basic profile *Transparency Measurement*). The frequency range for the embedding process for all audio files is as embedding parameter set to the fix value of 1000 Hz. The difference between the iteration steps is the used frequency band, which is shifted from 1000 Hz up to 15000 Hz. For all audio files, the embedding transparency decrease in the lower frequency band (less than 2000 Hz) or higher frequency band (higher than 7000 Hz). If the embedding frequency band begins in the range of 2000 Hz to 7000 Hz, the embedding transparency is more sensitive than lower than 2000 Hz or higher than 7000 Hz. This basic global characteristic of all graphs such as the overall trend and the minimum in the area around 3 kHz and local maximum in the area of 4-6 kHz are typical characteristics of a psycho acoustic model which may be explained by the psycho acoustic model used by ODG [23]. Figure 8 visualizes the test results presented in table 4. It can be seen, that the audio files, where the audio content are single instruments (*frer47, gspi_1, horn23* and *trpt21*), have a worse transparency than the other audio files with for e.g. spoken text.

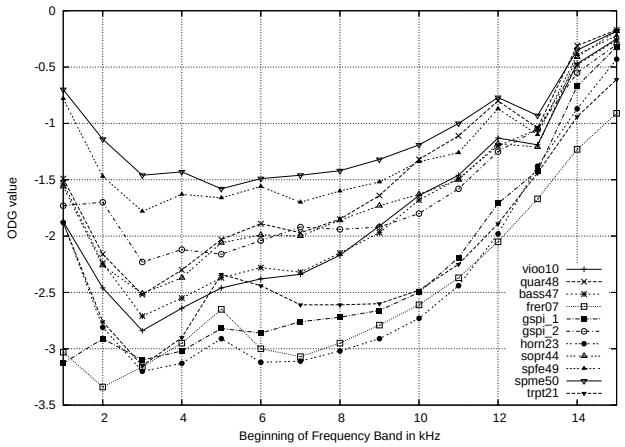

Figure 8: Shifted frequency band

Our three tests demonstrate the usability of the two basic profiles (*Capacity Measurement* and *Transparency Measurement*) to evaluate digital audio watermarking algorithms for the annotation applications. For our example watermarking algorithm we could determine, that the characteristic of the audio content has an influence to the computed capacity and transparency values.

The follow section concludes the paper and introduces future work.

5. CONCLUSION AND FUTURE WORK

In this paper, firstly we introduced and classified the single attacks from StirMark for Audio. Secondly, the profile attacks for real world scenarios are introduced and enhanced by new defined extended profile $P_{E-Annotation}$ and basic profiles
$P_{B-Capacity\_Measurement}$, $P_{B-Transparency\_Measurement}$ and $P_{B-Robustness\_Measurement}$. The transparency and capacity measurement profiles are selected to demonstrate the methodology how to evaluate a watermarking algorithm within profiles of SMBA. Our example watermarking algorithm is a spread spectrum algorithm working in frequency domain. The contribution of this article is threefold. Firstly, we have introduced a reference profile for one specific application: annotation watermarking. Secondly, we have suggested an evaluation methodology for this scenario, which is based on a fixed capacity boundary and iterative parameterization of embedding bandwidth, controlled by a feedback of ODG transparency measure. Our experimental evaluations are further based on three different settings of iteration step sizes as the third contribution can be summarized as follows: In all three test scenarios the transparency of audio watermarks showed a high dependency on the characteristics of

the audio material and it is shown, that single instruments are in most cases more effected than the other audio test files from SQAM.
Future work will include the entire process chain, including a robustness measurement and the watermark detector. With this extension and the methodology presented in this paper, future investigations with respect to the transparency of annotation watermarks will be performed. In context of this task, we will also consider to evaluate alternative watermark techniques, profiles and also subjective transparency tests.

6. ACKNOWLEDGMENTS

The work about single SMFA attacks described in this paper has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT. The information in this document is provided as is, and no guarantee or warranty is given or implied that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

Effort for the profile based evaluation is sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant number FA8655-04-1-3010. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research or the U.S. Government.

Furthermore, basic ideas presented here come from the previous work done in Illustration Watermarking and we would like to thank Thomas Vogel for his stimulating discussions and his work.

7. REFERENCES

[1] Emmanuel C. Ifeachor, Barrie W. Jervis, *Digital Signal Processing*, Prentice Hall, ISBN 0201 59619 9, 2002

[2] Nedeljko Cvejic, *Algorithms for Audio Watermarking and Steganography*, Department of Electrical and Information Engineering, Information Processing Laboratory, University of Oulu, 2004

[3] Jana Dittmann, Martin Steinebach, Andreas Lang, Sascha Zmudzinsky, *Advanced audio watermarking benchmarking*, San Jose, CA, USA Bellingham, Washington, USA, SPIE 2004, vol. 5306

[4] EAQUAL, http://home.wanadoo.nl/ w.speek/eaqual.htm, 2004

[5] Zaid Ghazal, *Umsetzung und Vergleich von digitalen Audio Wasserzeichenalgorithmen*, diploma thesis at Otto-von-Guericke University of Magdeburg, 2005

[6] ITU-R Recommendation BS.1387, *Method for Objective Measurements of Perceived Audio Quality*, Dec. 1998

[7] F.Y. Duan, I. King, *A Short Summary of Digital Watermarking Techniques for Multimedia Data*, Department of Computer Science and Engineering, Chinese University of Hong Kong, Shatin, N.T., Hong Kong, China

[8] M. Kutter, F.A.P. Petitcolas, *A fair benchmark for image watermarking system*, In SPIE Electronic Imaging, Security and Watermarking of Multimedia Contents, Vol. 3657, San Jose, CA, USA, 25-27 January 1999

[9] M. Kutter, S. Voloshynovskiy and A. Herrigel, *Watermark copy attack*, In Ping Wah Wong and Edward J. Delp eds., IS&T/SPIE's 12th Annual Symposium, Electronic Imaging 2000: Security and Watermarking of Multimedia Content II, Vol. 3971 of SPIE Proceedings, San Jose, California USA, 23-28 January 2000

[10] Andreas Lang, Jana Dittmann, *StirMark and profiles: from high end up to preview scenarios*, Reviewed Paper, IFIP/GI Workshop on Virtual Goods, Ilmenau (Germany), 28-29 May 2004, online publication available from http://virtualgoods.tu-ilmenau.de/2004/program.html

[11] Andreas Lang, Ryan Spring, *StirMark Benchmark for Audio - Attack Description*, internal report, 2004

[12] A. Lang, J. Dittmann, E. T. Lin, E. J. Delp, *Application Oriented Audio Watermark Benchmark Service*, to appear in SPIE 2005, San Jose

[13] Benoit Macq, Jana Dittmann, Edward J. Delp, *Benchmarking of Image Watermarking Algorithms for Digital Rights Management*, Proceedings of the IEEE, Special Issue on: Enabling Security Technology for Digital Rights Management, pp. 971-984, Vol. 92 No. 6, June 2004

[14] D.D.Rife, J.Vanderkooy, *Transfer-Function Measurement with Maximum-Length Sequences*, JAES, Vol. 37, June 1989

[15] Secret Rabbit Code (aka libsamplerate), http://www.mega-nerd.com/SRC/, 2004

[16] SoundTouch Sound Processing Library, http://sky.prohosting.com/oparviai/soundtouch/, 2004

[17] SQAM - Sound Quality Assessment Material, http://sound.media.mit.edu/mpeg4/audio/sqam/, 2004

[18] Steven W. Smith, *The Scientist and Engineer's Guide to Digital Processing*, California Technical Publishing, ISBN 0 9660176 3 3, 1997

[19] T. Thiede, W. C. treurniet, R. Bitto, C. Schmidmer, T. Sporer, J. G. Beerends, C. Colomes, M. Keyhl, G. Stoll, K. Brandeburg, B. Feiten, *PEAQ – The ITU Standard for Objective Measurement of Perceived Audio Quality*, J. Audio Eng. Soc., vol. 48, pp. 3-29, Jan-Feb. 2000

[20] Visit Audacity, Inc, http://www.audacity.com/, 2004

[21] Thomas Vogel, Jana Dittmann, *Illustration Watermarking: An Object Based Approach for Digital Images*, to appear in SPIE 2005, San Jose

[22] X Multimedia System, http://www.xmms.com/, 2004

[23] Eberhard Zwicker, Hugo Fastl, *Psychoacoustics. Facts and Models*, Springer, Berlin, November 2001, ISBN 3 540 65063 6

Table 2: Annotation with increased bandwidth of 200 Hz with unspecified watermark

Cycle	Frequency [Hz]	vioo10	quar48	bass47	frer07	gspi_1	gspi_2	horn23	sopr44	spfe49	spme50	trpt21
1	9900-10100	-0.68	-0.89	-0.88	-1.28	-1.29	0.01	-1.64	-0.62	-0.96	0.04	-0.01
2	9800-10200	-1.16	-0.79	-0.80	-2.2	-1.65	-1.43	-2.03	-0.81	-0.94	-0.71	-1.93
3	9700-10300	-1.43	-1.14	-1.37	-2.37	-2.21	-1.36	-2.42	-1.23	-1.2	-0.65	-2.08
4	9600-10400	-1.63	-1.30	-1.56	-2.54	-2.42	-1.72	-2.63	-1.43	-1.24	-1.06	-2.35
5	9500-10500	-1.83	-1.49	-1.76	-2.75	-2.64	-1.88	-2.89	-1.67	-1.56	-1.26	-2.59
6	9400-10600	-1.97	-1.65	-1.92	-2.84	-2.72	-2.03	-2.97	-1.85	-1.57	-1.41	-2.71
7	9300-10700	-2.09	-1.79	-2.05	-2.92	-2.8	-2.1	-3.06	-2.0	-1.74	-1.53	-2.81
8	9200-10800	-2.23	-1.89	-2.26	-3.06	-2.96	-2.18		-2.16	-1.84	-1.66	-2.93
9	9100-10900	-2.33	-1.96	-2.33		-3.01	-2.26		-2.26	-1.94	-1.77	-3.01
10	9000-11000	-2.41	-2.05	-2.44			-2.33		-2.34	-2.03	-1.8	
11	8900-11100	-2.52	-2.16	-2.52			-2.35		-2.46	-2.1	-1.97	
12	8800-11200	-2.63	-2.22	-2.58			-2.41		-2.53	-2.19	-1.97	
13	8700-11300	-2.66	-2.26	-2.63			-2.45		-2.59	-2.23	-2.04	
14	8600-11400	-2.71	-2.37	-2.75			-2.49		-2.65	-2.29	-2.12	
15	8500-11500	-2.79	-2.42	-2.81			-2.51		-2.72	-2.32	-2.19	
16	8400-11600	-2.81	-2.47	-2.85			-2.46		-2.77	-2.36	-2.24	
17	8300-11700	-2.89	-2.50	-2.88			-2.48		-2.82	-2.4	-2.28	
18	8200-11800	-2.91	-2.56	-2.96			-2.51		-2.89	-2.44	-2.33	
19	8100-11900	-2.96	-2.59	-2.99			-2.54		-2.92	-2.47	-2.35	
20	8000-12000	-2.99	-2.63	-3.01			-2.55		-2.99	-2.49	-2.39	
21	7900-12100	-3.03	-2.70				-2.56		-3.02	-2.52	-2.43	
22	7800-12200		-2.70				-2.59			-2.53	-2.47	
23	7700-12300		-2.75				-2.61			-2.58	-2.5	
24	7600-12400		-2.78				-2.61			-2.6	-2.53	
25	7500-12500		-2.80				-2.63			-2.59	-2.57	
26	7400-12600		-2.82				-2.65			-2.63	-2.58	
27	7300-12700		-2.85				-2.66			-2.66	-2.58	
28	7200-12800		-2.87				-2.66			-2.66	-2.63	
29	7100-12900		-2.90				-2.66			-2.68	-2.65	
30	7000-13000		-2.93				-2.67			-2.7	-2.67	
31	6900-13100		-2.96				-2.65			-2.7	-2.7	
32	6800-13200		-2.98				-2.67			-2.75	-2.71	
33	6700-13300		-2.98				-2.66			-2.74	-2.74	
34	6600-13400		-3.01				-2.66			-2.77	-2.78	
35	6500-13500						-2.68			-2.75	-2.76	
36	6400-13600						-2.69			-2.81	-2.79	
37	6300-13700						-2.71			-2.78	-2.8	
38	6200-13800						-2.71			-2.79	-2.82	
39	6100-13900						-2.71			-2.81	-2.83	
40	6000-14000						-2.73			-2.82	-2.87	
41	5900-14100						-2.75			-2.85	-2.88	
42	5800-14200						-2.77			-2.88	-2.91	
43	5700-14300						-2.78			-2.89	-2.92	
44	5600-14400						-2.78			-2.88	-2.93	
45	5500-14500						-3.88			-2.95	-3.0	
46	5400-14600						-2.92			-3.0		
47	5300-14700						-2.97					
48	5200-14800						-3.01					

Table 3: Annotation with increased bandwidth of 2000 Hz with unspecified watermark

Cycle	Frequency [Hz]	vioo10	quar48	bass48	frer47	gspi_1	gspi_2	horn23	sopr44	spfe49	spme50	trpt21
1	9000-11000	-2.41	-2.05	-2.44	-3.18	-3.03	-2.33	-3.13	-2.34	-2.03	-1.8	-3.08
2	8000-12000	-2.99	-2.63	-3.01			-2.55		-2.99	-2.49	-2.39	
3	7000-13000	-3.24	-2.93				-2.67		-3.22	-2.7	-2.67	
4	6000-14000		-3.10				-3.04			-2.82	-2.87	
5	5000-15000									-3.18	-3.18	

Table 4: Annotation with moved frequency band

Cycle	Frequency [Hz]	vioo10	quar48	bass48	frer47	gspi_1	gspi_2	horn23	sopr44	spfe49	spme50	trpt21
1	1000-2000	-1.87	-1.49	-1.53	-3.03	-3.13	-1.73	-1.88	-1.56	-0.78	-0.7	-1.88
2	2000-3000	-2.46	-2.16	-2.24	-3.34	-2.91	-1.7	-2.81	-2.26	-1.47	-1.14	-2.76
3	3000-4000	-2.84	-2.52	-2.71	-3.16	-3.1	-2.23	-3.2	-2.51	-1.78	-1.46	-3.15
4	4000-5000	-2.64	-2.3	-2.55	-2.95	-3.02	-2.12	-3.13	-2.37	-1.63	-1.43	-2.9
5	5000-6000	-2.46	-2.03	-2.37	-2.65	-2.82	-2.16	-2.91	-2.06	-1.66	-1.58	-2.34
6	6000-7000	-2.38	-1.89	-2.28	-3.0	-2.86	-2.04	-3.12	-1.99	-1.56	-1.49	-2.44
7	7000-8000	-2.34	-1.97	-2.32	-3.07	-2.76	-1.92	-3.11	-2.0	-1.7	-1.46	-2.61
8	8000-9000	-2.17	-1.85	-2.15	-2.95	-2.72	-1.94	-3.02	-1.86	-1.6	-1.42	-2.61
9	9000-10000	-1.91	-1.64	-1.97	-2.79	-2.66	-1.92	-2.91	-1.73	-1.52	-1.32	-2.6
10	10000-11000	-1.64	-1.32	-1.68	-2.61	-2.5	-1.8	-2.73	-1.63	-1.34	-1.19	-2.48
11	11000-12000	-1.46	-1.11	-1.49	-2.37	-2.19	-1.58	-2.44	-1.5	-1.26	-1.0	-2.25
12	12000-13000	-1.13	-0.8	-1.2	-2.05	-1.71	-1.25	-1.98	-1.18	-0.87	-0.77	-1.89
13	13000-14000	-1.19	-1.04	-1.05	-1.67	-1.42	-1.05	-1.38	-1.21	-1.1	-0.93	-1.44
14	14000-15000	-0.47	-0.31	-0.48	-1.23	-0.67	-0.55	-0.87	-0.41	-0.39	-0.35	-0.94
16	15000-16000	-0.25	-0.17	-0.26	-0.91	-0.32	-0.3	-0.43	-0.18	-0.22	-0.18	-0.61