A Perceptual Audio Hashing Algorithm: A Tool for Robust Audio Identification and Information Hiding

M. Kıvanç Mıhçak[1] and Ramarathnam Venkatesan[2]

[1] University of Illinois, Urbana-Champaign
mihcak@ifp.uiuc.edu
[2] Microsoft Research
venkie@microsoft.com

Abstract. Assuming that watermarking is feasible (say, against a limited set of attacks of significant interest), current methods use a secret key to generate and embed a watermark. However, if the same key is used to watermark different items, then each instance may leak partial information and it is possible that one may extract the whole secret from a collection of watermarked items. Thus it will be ideal to derive content dependent keys, using a *perceptual* hashing algorithm (with its own secret key) that is resistant to small changes and otherwise having randomness and unpredictability properties analogous to cryptographic MACs.
The techniques here are also useful for *synchronizing* in streams to find fixed locations against insertion and deletion attacks. Say, one may watermark a frame in a stream and can synchronize oneself to that frame using keyed perceptual hash and a known value for that frame. Our techniques can be used for *identification* of audio clips as well as *database lookups* in a way resistant to formatting and compression. We propose a novel audio hashing algorithm to be used for audio watermarking applications, that uses signal processing and traditional algorithmic analysis (against an adversary).

1 Introduction

Information hiding methods such as watermarking (WM) use secret keys, but the issue of choosing keys for a large set of data is often not addressed. Using the same key for many pieces of content may compromise the key in the sense that each item may leak some partial information about the secret. A good defense is not to rely on the requirement that the same secret key is used in watermarking different data. But using a separate key for each content would blow up the WM verification work load. Since adversarial attacks and WM insertion are expected to cause little or minor perceptual alterations, any hash function (with a secret key K) that is resistant to such unnoticeable alterations can be used to generate input–dependent keys for each piece of content, analogous to cryptographic MACs. For an attacker (without K), the hash value of a given content will be unpredictable.

I. S. Moskowitz (Ed.): IH 2001, LNCS 2137, pp. 51–65, 2001.

Further motivation stems from hiding information in streams (e.g. video or audio), assuming we are given a method for hiding a WM in a single frame or element (e.g. image or a 30 second audio clip) of the stream. Within this context, the hash values can be used to select frames pseudo-randomly with a secret key, and locate them later after modifications and attacks; this yields a synchronization tool, whereby one can defend against de-synch attacks such as insertion, deletion and time dilation. This approach also will reduce the number of watermarked frames which in turn reduces the overall perceptual distortions due to embedded WMs, as well as the work load of WM detection if the hash functions are faster or incremental. Alternate way to synchronize is to use embedded information, but this may lead to circular situations or excessive search as attack methods evolve. In the context of streams, consider a relatively weak information hiding method that survives with probability 0.01 on each segment of the stream (e.g. each frame of a video sequence) after attacks. Provided that we can synchronize to the locations where information is hidden, even such a weak method would be adequate for applications with long enough streams (since it is possible to hide the same or correlated information in a neighborhood whose location is determined by hash values). Viewed as a game against an adversary, an embedding step (not present in hashing) has to first commit to a move, whereby the adversary has extra information in the form of the watermarked content to attack. Hashing appears to be a simpler problem to study first and enable one to better understand the more complex WM problem [1].

Other applications of hash functions include *identification* of content that need copyright protection, as well as searching (in *logn* steps) in a database (of size n), and sorting in a way that is robust to format changes and compression type common modifications.

Conventional hashing :The uses of hash functions, which map long inputs into short random-looking outputs, are many and indeed wide-ranging: compilers, checksums, searching and sorting algorithms, cryptographic message authentication, one-way hash functions for digital signatures, time stamping, etc. They usually accept binary strings as inputs and produce a fixed length hash value (say L). They use some random seeds (keys) and seek the following goals:

(**Randomness**)For any given input, the output hash value must be uniformly distributed among all possible L-bit outputs

(**Approximate pairwise independence**) For two distinct inputs, the corresponding outputs must be statistically almost independent of each other.

Note that the term "randomness" above refers to having uniform (maximal entropy) or almost uniform random hash values. It can be shown that the collision probability (i.e. the probability that two distinct inputs yield the same output) is minimized under the these two conditions. It is well known that for the purposes of minimizing the collision probability, one needs to consider the algorithm's behavior only on *pairs* of inputs. Clearly, the utility of conventional hash functions depend on having minimal number of collisions and *scalability* (a direct result of the two requirements above) as the data set size grows. Such

a scalability in the multimedia applications remains an open problem and may need explicitly randomized algorithms (rather than assuming that images have entropy and thus contribute to the randomness of hash values); here we need to treat two perceptually similar objects as the same, which leads to the additional constraint:

(**Perceptual similarity**) For a pair of perceptually similar inputs, the hash values must be the same (with high probability over the hash function key).

For example, we term two audio clips as "perceptually similar" if they sound the same. For simplicity one may use a standard Turing test approach where a listener is played two audio clips at random order, and one should not be able to distinguish them.

A corollary of the perceptual requirement is that our hash values must remain invariant before and after watermarking, and it should remain the same even after malicious attacks (that are within reasonable bounds). This requirement considerably complicates the matters. Nevertheless we propose an algorithm to achieve these goals. The proposed algorithm has shown itself to be quite successful in our tests. In particular, we consider the problem *audio hashing*. We present design algorithms and some simulation results; our designs take cue from the design of the similar image hashing function described in a paper by Venkatesan *et.al.* [2]. Our hash functions produce **intermediate hash** values that can be used if two given items are similar.

2 Definitions and Goals

Let X denote a particular audio clip, $\hat{X}$ denote a modified version of this clip which is "perceptually same" as X and Y denote a "perceptually different" audio clip. Let L be the final length of the hash, K be the secret key used and $H_K(.)$ represent a hash function that takes audio clips as inputs and produces length L binary strings using the secret key K. We state our goals as below; formalizing them would need a notion of metric (here the standard metrics (without randomizations as we do here) may pose problems) and addressing questions if L can be increased at will. We do not address them here.

(***Randomization :***) For all $\alpha, X : \Pr[H_K(X) = \alpha] \approx 2^{-L}$
(***Pairwise independence of perceptually different inputs)*** For all α, β, X, Y :
$\Pr[H_K(X) = \alpha | H_K(Y) = \beta] \approx \Pr[H_K(X) = \alpha]$
(***Collision on perceptually similar inputs:***) For all $X, \hat{X}$:
$\Pr\left[H_K(X) = H_K\left(\hat{X}\right)\right] \approx 1$

Thus, apart from the randomization issue, our goal can be viewed as (given a distance metric $D(.,.)$)

$$D\left(H_K(X), H_K\left(\hat{X}\right)\right) = 0, \quad D\left(H_K(X), H_K(Y)\right) > 0, \tag{1}$$

with high probability for all possible different audio clips X, Y and for all possible "perceptually inaudible" modifications on X that yield $\hat{X}$. Throughout this paper, we shall use normalized Hamming distance as the distance metric D (the normalization is done by the length of the hash).

In order to simplify the presentation, we divide the problem into two stages:

1. **Intermediate hash value**: At the end of the first stage, we aim to obtain hash values that are of length M, where $M > L$ and have the following separation property:

$$D\left(h_K\left(X\right), h_K\left(\hat{X}\right)\right) < 0.2, \quad D\left(h_K\left(X\right), h_K\left(Y\right)\right) > 0.35, \tag{2}$$

   where h_K is the intermediate hash function that takes audio clips as inputs and produces length l binary strings.
2. Given the intermediate hash, we use some list-decoding procedures to generate a binary string of length L with desired properties (similar tools were employed in [2]).

This paper focuses on the "intermediate hash" part of the problem. In the rest of the paper, we shall drop the subscript K in the representation of the intermediate hash function for convenience; it will be denoted by h_X for an input signal X. Typically, we design h_X such that $5L < l < 10L$. We experimentally show that the present version of the algorithm achieves (2) for an extensive range of attacks and audio clips. The ongoing research focuses on proposing a complete solution to the problem, in particular we currently concentrate on developing an algorithm for solving Stage 2 and augmenting the robustness properties of the proposed algorithm for Stage 1.

3 Proposed Algorithm

The block diagram of our proposed methodology is shown in Fig. 1. An algorithmic description is given below (secret key K is used as the seed of the random number generator in each of the randomized steps):

1. Put the signal X in canonical form using a set of standard transformations (in particular MCLT (Modulated Complex Lapped Transform) [3]). The result is the time–frequency representation of X, denoted by T_X.

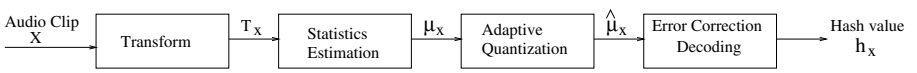


Fig. 1. Block diagram of the proposed audio hashing algorithm. X is the input audio clip, T_X is the time–frequency representation using MCLT (Modulated Complex Lapped Transform), $\mu_{\mathbf{X}}$ represents estimated statistics from the transform domain, $\hat{\mu}_{\mathbf{X}}$ represents the quantized value of the statistics and h_X is the final hash value of the audio clip

2. Apply a randomized interval transformation to T_X in order to estimate audible statistics, μ_X, of the signal.
3. Apply randomized rounding (i.e. quantization) to μ_X to obtain $\hat{\mu}_X$.
4. Use the decoding stages of an error correcting code on $\hat{\mu}_X$ to map similar values to the same point. The intermediate hash, h_X, is produced as a result of this stage.

Each of aforementioned steps shall be explained in detail in subsequent sections.

3.1 MCLT

MCLT ([3]) is a complex extension of MLT (Modulated Lapped Transform). MLT was introduced in [4] and is used in many audio processing applications, such as Dolby AC-3, MPEG-2. Characteristics of time-varying versions of MLT and audio processing applications are discussed inn [5]. MCLT basis functions are found in pairs to produce real and complex parts separately. These basis functions are derived from MLT and they are phase–shifted versions of each other. MCLT has perfect reconstruction and approximate shift invariance properties. For further details of the MCLT, we refer the reader to [3].

Fig. 2 shows the implementation. Let $2M$ be the length of the analysis and synthesis filters. Audio input sequence X is broken into overlapping "blocks" of length $2M$ (Fig. 2a), so that neighboring blocks overlap by 50%. The number of frequency bands for each block is M. After the transform is applied to each block independently(Fig. 2b), the *magnitudes of transform domain coefficients* are combined into a matrix to obtain the time–frequency representation of X, denoted by T_X (Fig. 2(c)). T_X is of size $M \times N$ where N is the number of blocks. In the notation below, let $A(i, j)$ represent the (i, j)th element of a 2–dimensional matrix A. MCLT can be used to define a "hearing threshold matrix" H_X which is of the same size T_X, such that if $T_X(i, j) \geq H_X(i, j)$, then $T_X(i, j)$ is audible, inaudible otherwise. Such hearing thresholds in the MCLT domain have proven to be useful in audio compression [6] and audio watermarking [7] applications.

We now introduce *significance map* S_X, defined as $S_X(i, j) = 1$ if $T_X(i, j) \geq H_X(i, j)$ and 0 otherwise. The time-frequency representations and corresponding significance maps for two different audio clips are shown in Fig. 3. Note that there exists a striking pattern in time–frequency representation of an audio clip (See Fig. 3). Furthermore this pattern has a slowly varying structure both in time and frequency. Our purpose is to capture this existing structure in a compact fashion via randomized interval transformations (also termed as statistics estimation) which is explained in the next section.

3.2 Randomized Interval Transformation (Statistics Estimation)

Our goal is to estimate signal statistics that would reflect its characteristics in an irreversible manner, while introducing robustness against attacks. We carry out statistics estimation in the time–frequency domain and exploit both local and global correlations. Note that correlations exist both along time axis and

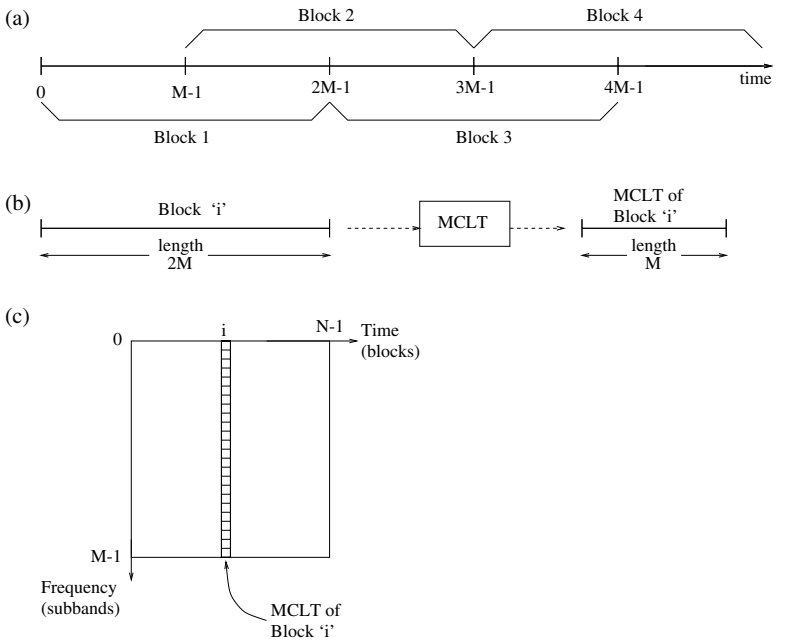


Fig. 2. MCLT. (a) The input audio clip is split into blocks that have a 50% overlap with their neighbors. (b) MCLT is applied independently to each block to produce spectral decomposition of size M. (c) The spectral decomposition of the blocks are combined together in order to form the time–frequency decomposition, T_X

frequency axis(Fig. 3). These correlations constitute different characteristics of audio. In general, it is not clear what type of characteristics are more robust and representative and it is a non–trivial task to localize both in time and frequency. These observations suggest a trade–off between time and frequency in terms of statistics estimation. Hence we propose 3 methods for statistics estimation. Method I exploits correlations in frequency localized in time; method II uses correlations in time localized in frequency and method III uses correlations both in time and frequency via randomized rectangles in the time–frequency plane. Each one of these methods could be useful for different applications (for different strong attacks considered). The common property shared by all 3 is that for perceptually similar audio clips, estimated statistics are likely to have close values (under suitable notions of metric) whereas for different audio clips they are expected be far apart. The secret key K is used as the seed of random number generator in each of randomized steps of the proposed methods.

Method I : The algorithmic description is given below.

1. For each block (each column of T_X), determine if there exist sufficiently many entrees exceeding the hearing thresholds. If not pass to the next block, else collect the "significant" coefficients of the ith block into vector $\mathbf{v}_i$ of size

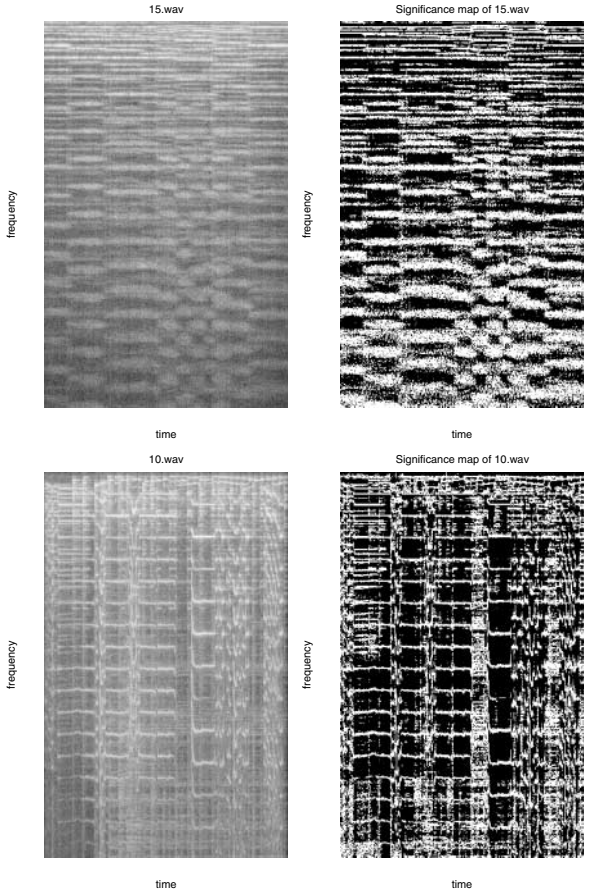


Fig. 3. Time–frequency representations (left side) and corresponding significance maps (right side) for two different audio clips

$\tilde{M}_i \leq M, 0 \leq i < N$. The steps **2.** and **3.**, that are explained below, are repeated for each $\mathbf{v}_i$.

2. Randomized Interval Transformation : Refer to Fig. 4(a) for a single step of splitting. At a single level of randomized splitting, splitting point is picked randomly around the "randomization region" of the midpoint (of a vector or subvector). As a result of a single split, two new subvectors are formed. For each $\mathbf{v}_i$, this procedure is carried out recursively a certain number of times (level) on each new–born subvector (Fig. 4(b) shows 2 level recursive splitting). The relative length of the randomization region and the level of splitting are user parameters.

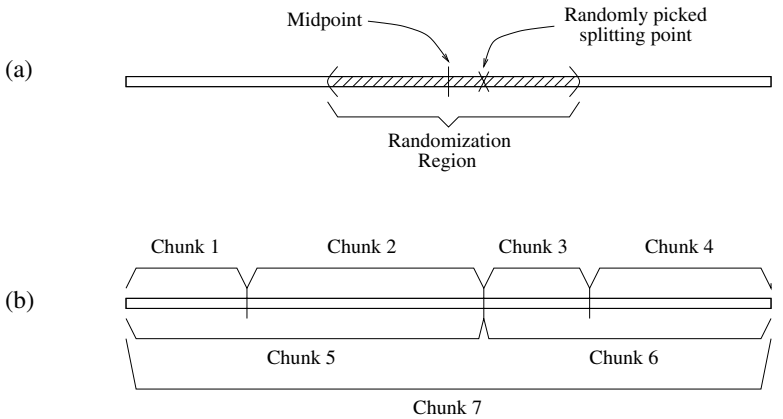


Fig. 4. Randomized splitting and the formation of subvectors (also termed as chunks) in order to perform 1st order statistics estimation. In (a), we show how a single step randomized splitting is carried out. The procedure shown in (a) is repeated a finite number of times in a recursive manner. In (b), randomized subvectors are formed for a 2–level recursion in randomized splitting. The length of the statistics vector in case of 2 level splitting would be 7

3. Compute 1st order statistics (empirical mean) of $\mathbf{v}_i$ and each subvector produced from it in the process of splitting. Gather these statistics in a vector, called μ_i.
4. Repeat steps **2.** and **3.** for all $\mathbf{v}_i$ for which $\tilde{M}_i$ is sufficiently large. Collect all μ_i obtained in a single vector, to form total statistics vector μ_X.
Method II : In this method, we collect 1st order statistics for each "significant" frequency subband (whereas in Method I, statistics are obtained from each "significant" time block). Hence, the machinery explained above is applied to each row of T_X in Method II (with possibly different parameters). The difference between methods I and II is depicted in the left panel of Fig. 5.
Method III : Let ll be the length of the total statistics vector that is desired to be obtained as a result of this method (a user parameter). The algorithmic description is given next.
1. For each rectangle i $(1 \leq i \leq ll)$, first randomly generate its width, ww_i and its height, hh_i. ww_i and hh_i are realizations of uniform distributions in the intervals of $[ww - \Delta_w\,,\, ww + \Delta_w]$ and $[hh - \Delta_h\,,\, hh + \Delta_h]$ respectively, where ww, hh, Δ_w, Δ_h are user parameters. Next, randomly generate the location of center of gravity of each rectangle, cc_i, such that it resides within the range of T_X.
2. For each rectangle i $(1 \leq i \leq ll)$, the corresponding 1st order statistic is given by the sum of "significant" coefficients within that rectangle (the transform coefficients that are larger than hearing threshold) divided by the area of the rectangle ($ww_i \times hh_i$).
3. Collect all such statistics in a single vector, to form total statistics vector μ_X.

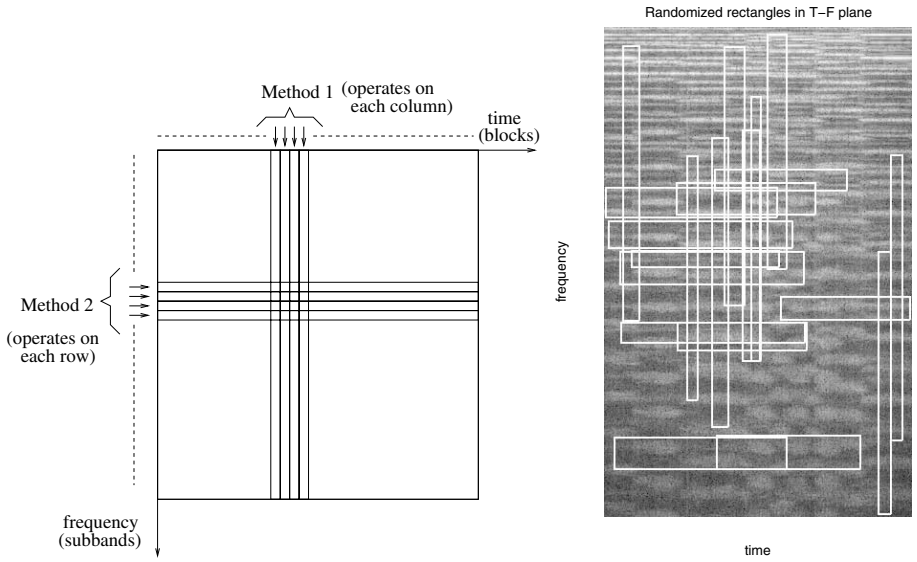


Fig. 5. The operation of statistics estimation in proposed methods in the time–frequency plane. Left: Method I operates on each time block, exploits correlations in frequency; method II operates on each frequency band, exploits correlations in time. Right: Method III exploits correlations both in time and frequency via random rectangles

Remarks :

a. We propose to include "significant" coefficients only in the statistics estimation in all the proposed methods. The rationale is that most acceptable attacks would easily alter inaudible portions of audio clips in huge amounts, possibly erase them, whereas significantly audible portions should not be varied to a high extent.

b. Note that methods I and II collect statistics that naturally include redundancies (i.e. given the statistics at the lowest level of splitting recursion, it is possible to uniquely determine the statistics at higher levels). Such a mechanism uses error correction encoding flavors that are naturally tailored for multimedia signals. As a result, redundancy is added such that both local and semi–global signal features are compactly captured.

c. In method I, by localizing in time, we capture dominant note(s) for each time block that hints about the global frequency behavior at that time instant. On the other hand, in method II, by localizing in frequency, we capture the temporally global behavior of particular frequency bands. As result, method I is, by construction, more robust against frequency domain linear filtering type attacks, whereas method II is more robust again time-stretching type attacks, again by construction. This motivates us to get the best of both worlds: in method III, 2 types of rectangles are employed; tall&narrow rectangles that

localize in time and short&wide rectangles that localize in frequency (see right panel of Fig. 5).

d. Although our methods use 1st order statistics in local regions of the time–frequency plane, our approach is inherently flexible in the sense that estimates of any order statistics from regions of various shapes and locations could possibly be employed. In particular, any representative of an audio clip, that is believed to compactly capture signal characteristics while maintaining robustness, could be used in the latter stages of our algorithm as well.

3.3 Adaptive Quantization

At this stage of the algorithm, our goal is to discretize μ_X. While accomplishing this task, we also want to both enhance robustness properties and increase randomness to minimize collision probabilities. The conventional way of discretizing a continuous signal is termed as "quantization". While we are going to use basic techniques of quantization, slight modifications will take place in order to achieve our goal.

Let Q be the number of quantization levels, $\hat{\mu}_X$ denote the quantized μ_X, $\mu_X(j)$ and $\hat{\mu}_X(j)$ denote the jth elements of μ_X and $\hat{\mu}_X$ respectively. In conventional quantization schemes, the quantization rule is completely deterministic and given by $\Delta_i \leq \mu(j) < \Delta_{i+1} \quad \Leftrightarrow \quad \hat{\mu}(j) = i, \quad 0 \leq i < Q$, where the interval $[\Delta_i, \Delta_{i+1})$ is termed as ith quantization bin. (Unlike the compression problem, the reconstruction levels are not crucial for hashing problem as long as the notion of being "close" is preserved at the quantized output. Therefore, without loss of generality, we choose $\hat{\mu}_X(j) = j$.)

Our observations reveal that, μ_X often comes from a distribution that is highly biased at some points. This "colored" nature of the statistics distribution motivates us to employ an "adaptive quantization" scheme which takes into account possible arbitrary biases at different locations of the distribution of the statistics. In particular, we use the normalized histogram of μ_X as an estimate of its distribution. Note that normalized histogram is usually very resistant against "slightly inaudible" attacks. Hence, we propose to design quantization bins $\{\Delta_i\}$ such that $\int_{\Delta_{i-1}}^{\Delta_i} p_\mu(t)\,dt = 1/Q, \quad 0 \leq i < Q$, where p_μ stands for the normalized histogram of μ_X. Next, we define the "central points", $\{C_i\}$, such that $\int_{\Delta_{i-1}}^{C_i} p_\mu(t)\,dt = \int_{C_i}^{\Delta_i} p_\mu(t)\,dt = 1/(2Q), \quad 0 \leq i < Q$. Around each Δ_i, we introduce a randomization interval $[A_i, B_i]$ such that $\int_{A_i}^{\Delta_i} p_\mu(t)\,dt = \int_{\Delta_i}^{B_i} p_\mu(t)\,dt, \quad 0 \leq i < Q$, i.e. the randomization interval is symmetric around Δ_i for all i in terms of distribution p_μ. We also impose the natural constraint $C_i \leq A_i$ and $B_i \leq C_{i+1}$. Our proposed p.d.f.–adaptive randomized quantization rule is then given by

$$A_i \leq \mu_X(j) \leq B_i \quad \Leftrightarrow \quad \hat{\mu}_X(j) = \begin{cases} i & \text{with probability } \dfrac{\int_{A_i}^{\mu_X(j)} p_\mu(t)\,dt}{\int_{A_i}^{B_i} p_\mu(t)\,dt} \\ i-1 & \text{with probability } \dfrac{\int_{\mu_X(j)}^{B_i} p_\mu(t)\,dt}{\int_{A_i}^{B_i} p_\mu(t)\,dt} \end{cases}$$

and

$$C_i \leq \mu_X(j) \leq A_i \Leftrightarrow \hat{\mu}_X(j) = i - 1 \text{ with probability 1,}$$
$$B_i \leq \mu_X(j) < C_{i+1} \Rightarrow \hat{\mu}_X(j) = i \text{ with probability 1.}$$

The denominator term $\int_{A_i}^{B_i} p_\mu(t)\,dt$ in the random region is a normalization factor. The probabilities are assigned in accordance with the "strength" of the p.d.f. Note that if $\mu_X(j) = \Delta_i$ for some i, j, then it is a fair coin toss; conversely as $\mu_X(j)$ approaches A_i or B_i for some i, j, quantization decision becomes more biased. The amount of randomness in quantization in bin i is controlled by $(\int_{L_i}^{\Delta_i} p_\mu(t)\,dt)/(\int_{\Delta_{i-1}}^{\Delta_i} p_\mu(t)\,dt)$, which is a user parameter and which we choose to be the same for all i due to symmetry.

Remark :

The choice of this parameter offers a trade–off: As it increases, the amount of randomization at the output increases, which is a desired property to minimize collision probability, however this also increases the chances of being vulnerable to attacks (slight modifications to the audio clip would change the probability rule in quantization). Hence, we would like to stress that choosing a suitable value for this parameter is a delicate issue.

3.4 Error Correction Decoding

At this step of the algorithm, the goal is to to convert $\hat{\mu}_X$ into a binary bit string and shorten the length such that "perceptually similar" audio clips are mapped to binary strings that are close to each other and "perceptually different" audio clips are mapped to binary strings that are far away from each other. The resulting hash values' being close and far away are measured in the sense of $D(.,.)$ which was defined in Sec. 2.

In order to achieve this purpose, we employ 1st order Reed-Muller codes. Reed-Muller codes are a class of linear codes over GF(2) that are easy to describe and have an elegant structure. The generator matrix $\mathbf{G}$ for the 1st order Reed-Muller code of codeword length 2^m is defined as an array of blocks: $\mathbf{G} = \begin{bmatrix} \mathbf{G}_0 \\ \mathbf{G}_1 \end{bmatrix}$, where $\mathbf{G}_0$ is a single row consisting of all ones and $\mathbf{G}_1$ is a matrix of size m by 2^m. $\mathbf{G}_1$ is formed in such that each binary m–tuple appears once as a column. The resulting generator matrix is of size $m + 1$ by 2^m. For further details on error correcting codes and Reed–Muller codes in particular, we refer the reader to [8].

Unlike traditional decoding schemes that use Hamming distance as the error metric, we propose to use a different error measure which we call "Exponential Pseudo Norm" (EPN). This error measure has proven to be effective in the image hashing problem [2] and we believe that it is inherently more suitable than traditional error metrics (such as Hamming distance) for multimedia hashing problems. In the next paragraph, we give a description of EPN.

Let $\mathbf{x}_D$ and $\mathbf{y}_D$ be 2 vectors of length z such that each component of these vectors belongs to the set $\{0, 1, \ldots, Q-1\}$. Similarly let $\mathbf{x}$ and $\mathbf{y}$ be the binary representations of the vectors $\mathbf{x}_D$ and $\mathbf{y}_D$ respectively, where each decimal

component is converted to binary by using $\lceil \log_2 Q \rceil$ bits. Note that the lengths of $\mathbf{x}$ and $\mathbf{y}$ are therefore going to be both $Z\lceil \log_2 Q \rceil$. EPN is defined between the binary vectors $\mathbf{x}$ and $\mathbf{y}$ as $\mathrm{EPN}(\mathbf{x}, \mathbf{y}) \stackrel{\triangle}{=} \sum_{i=1}^{Z} \Gamma^{|x_D(i) - y_D(i)|}$, where $x_D(i)$ and $y_D(i)$ denote the ith elements of the vectors $\mathbf{x}_D$ and $\mathbf{y}_D$ respectively. Note that $\mathrm{EPN}(\mathbf{x}, \mathbf{y})$ is actually a function of Q and Γ as well, however for the sake of having a clean notation we are embedding these values in the expression and assuming that these values are known within the context of the problem.

In the hashing problem, Q is the number of quantization levels, and Γ is the "exponential constant" that determines how EPN penalizes large distances. Based on our experiments, the results are approximately insensitive to the value of Γ provided that it is chosen large enough. We believe that EPN is more favorable for the hashing problem since most attacks would cause small perturbations and thus we wish to distinguish between close and far values with an emphasis (stronger than linear).

The algorithmic explanation of this step is given next:

1. Divide $\hat{\mu}_X$ into segments of a certain length (user specified parameter).
2. Convert the contents of each segment into binary format by using $\lceil \log_2 Q \rceil$ bits for each component, where Q is the number of quantization levels.
3. Form the generator matrix of 1st order Reed–Muller code where the length of the codewords is as close as possible to the length of each segment.
4. For all possible input words (there are a total of 2^{m+1} possible input words for a generator matrix of size $m+1$ by 2^m), generate the corresponding codewords.
5. For all possible input words and for all segments, find the EPN between the corresponding codeword and the quantized data in that segment.
6. For each segment, pick up the input word that yields the minimum EPN.
7. Concatenate the chosen input words to form the intermediate hash h_X.

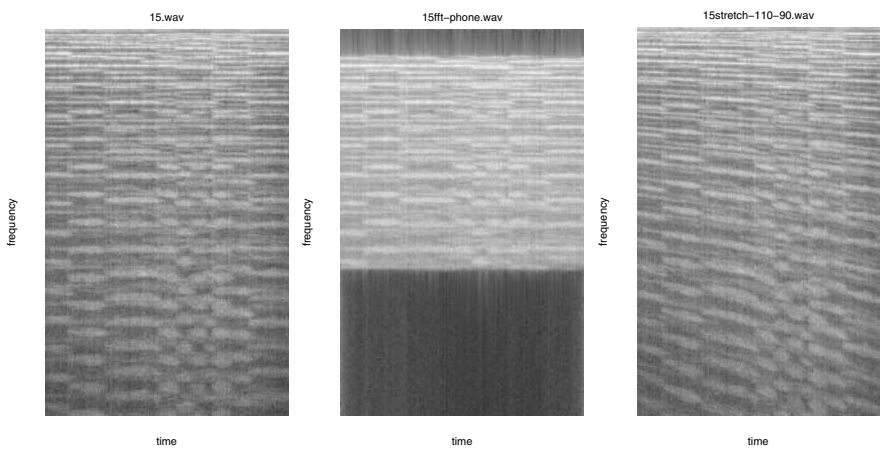
4 Testing under Attacks

In our simulations, we used 15 second audio clips that were subjected to approximately 100 different attacks performed by a commercial software [9]. We assume that the input audio clips are in ".wav" format. In Fig. 6, we show an audio clip and two attacked versions of this clip that have inaudible or slightly audible modifications. The attacks we considered can roughly be classified into the following categories:

1. **Silence Suppression:** Remove inaudible portions that have low amplitudes.
2. **Amplitude Modification: (inaudible or slightly audible)**
   (a) Apply amplification factors that are either constant or slowly time–varying.
   (b) Dynamic range processing type attacks that modify the audio clip components based on their values. For instance medium amplitude can be expanded and high and low amplitude values can either be cut.
   (c) Echo effects are one of the most significant attacks or modifications in audio signal processing. Echos can be explained as repetitions of signal

peaks with exponentially decaying magnitudes. Echo hiding, echo cancellation and producing echo chamber effects usually produce inaudible effects whereas the signal values change significantly.

3. **Delays:** An audio clip can be delayed by some percentage of its duration. Furthermore the original clip and the slightly delayed versions can be "mixed" yielding slightly audible effects. These are some of the most potent attacks.
4. **Frequency Domain Effects:** These attacks usually involve modifications in the spectrum of the signal.
   (a) Filtering effects usually involve low pass filters, band pass filters and equalizers. Human beings are most sensitive to a certain group of frequencies only (0.5–7 kHz) which makes such attacks effective.
   (b) Denoising and hiss reduction techniques usually operate in the spectrum domain. The main aim of such techniques is to remove the undesired background noise. However in case of attacks, the noise threshold can deliberately be set to be high such that only the major signal components that create the melody survive.
5. **Stretching and Pitch Bending:** The length of the audio clip can be changed slightly without causing too much audible distortion. The basic procedure is to apply downsampling and upsampling in an adaptive fashion. By using such techniques it is possible to play audio clips slightly faster or slightly slower of even with slowly changing speed. Such attacks cause "bending" effects in the spectrum representation of the signal.



(a) (b) (c)

Fig. 6. (a) Original audio clip, (b) Attacked with heavy band pass filtering, (c) Another attack that includes stretching and pitch bending. Note that the "fibers" in (a) are bent in (c)

In order to overcome some of the de–synch effects, we apply a few simple synchronization techniques within our proposed methods. These techniques include:

- **Silence Deletion :** Before applying the hashing algorithms, we completely remove "silent" or "approximately silent" parts of the audio clip.
- **Amplitude Normalization :** Before applying MCLT, we normalize the contents of each block such that the dynamic range is precisely $[-1, 1]$ within a local neighborhood. The normalization is done via scaling.
- **Frequency Band Selectivity :** We apply our statistics estimation methods to a frequency band, to which human ears are sensitive. We choose this band as 50 Hz – 4 KHz range.

Our results reveal that, Method I yields hash values that achieve the goal expressed in (2) for all of the inaudible attacks, 50–60 percent of which achieve zero error. For some of the slightly audible attacks, Method I fails to achieve (2). These cases include too much amplification, too much delay, too much time stretching. We observed that Method II is inferior to Method I over a broad class of attacks. However within the class of attacks that Method I fails, particularly delay and time stretching type of attacks, Method II produces superior results and achieves (2). Method III produces the best results among the three over a broad class of attacks and achieves (2) under most acceptable attacks as long as they are not too severe. This is intuitively clear since Method III is designed such that it captures (at least partially) signal characteristics captured by both Method I and II. For a particular class of attacks, the superiority of Method III is not clear. For instance Method I provides superior performance for frequency domain modification type attacks, whereas Method II provides superior performance for temporal displacement type attacks.

5 Conclusions and Future Work

Our approach to the hashing problem takes its principles from collecting both *robust* and *informative* features of the multimedia data. Note that due to the well–known problem of lacking suitable distortion metrics for multimedia data, this is a non–trivial and tough task. Furthermore, in general there is a trade–off between *robustness* and *being informative*, i.e., if very crude features are used, they are hard to change, but it is likely that one is going to come across collision between hash values of perceptually different data. Robustness, in particular, is very hard to achieve. It is clear that there is going to be clustering between hash value of an input source and hash values of its attacked versions. In principle, a straightforward approach would be to use high–dimensional quantization where quantization cells are designed such that their centers coincide with centers of clusters. However, since original data are unknown, this does not seem to be plausible unless input adaptive schemes are used [10].

In this paper, we introduced the problem of randomized versions of audio hashing. Robust hash functions could be quite useful in providing content dependent keys for information hiding algorithms. Furthermore such hash values

would be very helpful against temporal de-synchronization type attacks in watermarking streaming multimedia data. Our novel perceptual audio hashing approach consists of randomized statistics estimation in the time–frequency domain followed by random quantization and error correction decoding.

In addition to adapting and testing our algorithms in the applications mentioned earlier, our future work includes using additional steps involving more geometric methods for computing hash values, as well as using the ideas from here to develop new types of WM algorithms. See [11] for any further updates.

Acknowledgments : We thank Rico Malvar of Microsoft Research for his generous help with audio tools, testing and valuable suggestions. We also thank M. H. Jakubowski, J. Platt, D. Kirovski, Y. Yacobi as well as Pierre Moulin (U. of Illinois, Urbana–Champaign) for discussions and comments.

References

1. M. K. Mıhçak, R. Venkatesan and M. H. Jakubowski, "Blind image watermarking via derivation and quantization of robust semi–global statistics I," *preprint.* 52
2. R. Venkatesan, S.-M. Koon, M. H. Jakubowski and P. Moulin, "Robust image hashing," *Proc. IEEE ICIP*, Vancouver, Canada, September 2000. 53, 54, 61
3. H. S. Malvar, "A modulated complex lapped transform and applications to audio processing,", *Proc. IEEE ICASSP*, Phoenix, AZ, March 1999. 54, 55
4. H. S. Malvar, *Signal Processing with Lapped Transforms.* Norwood, MA: Artech House, 1992. 55
5. S. Shlien, "The modulated lapped transform, its time-varying forms, and applications to audio coding," *IEEE Trans. Speech Audio Processing*, vol. 5, pp. 359–366, July 1997. 55
6. Windows Media Player 55
7. D. Kirovski, H. S. Malvar and M. H. Jakubowski, "Audio watermarking with dual watermarks," *U. S. Patent Application Serial No. 09/316,899,* filed on May 22, 1999, assigned to Microsoft Corporation. 55
8. R. Blahut, *Theory and Practice of Error Control Codes*, 1983. 61
9. See http://www.syntrillium.com/cooledit/. 62
10. M. K. Mıhçak and R. Venkatesan, "Iterative Geometric Methods for Robust Perceptual Image Hashing," *preprint.* 64
11. See http://www.research.microsoft.com/˜venkie. 65