CERIAS Tech Report 2004-05

RESILIENT RIGHTS PROTECTION FOR SENSOR STREAMS

by Radu Sion, Mikhail Atallah, Sunil Prabhakar

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

Resilient Rights Protection for Sensor Streams *

Radu Sion, Mikhail Atallah, Sunil Prabhakar
Center for Education and Research in Information Assurance
Computer Sciences, Purdue University
West Lafayette, IN, 47907, USA
[sion, mja, sunil]@cs.purdue.edu

Abstract

Today's world of increasingly dynamic computing environments naturally results in more and more data being available as fast streams. Applications such as stock market analysis, environmental sensing, web clicks and intrusion detection are just a few of the examples where valuable data is streamed to its consumer. Often, streaming information is offered on the basis of a non-exclusive, single-use customer license. One major concern, especially given the digital nature of the valuable stream, is the ability to easily record and potentially "re-play" parts of it in the future. If there is value associated with such future re-plays, it could constitute enough incentive for a malicious customer (Mallory) to duplicate segments of such recorded data, subsequently re-selling them for profit. Being able to protect against such infringements becomes a necessity.

In this paper we introduce the issue of rights protection for streaming data through watermarking. This is a novel problem with many associated challenges including: the inability to perform multiple-pass random accesses to the entire data set, the requirement to be fast enough to keep up with the incoming stream rate, to survive instances of extreme sparse sampling and summarizations, while at the same time keeping data alterations within allowable bounds. We propose a solution and analyze its resilience to various types of attacks as well as some of the important expected domain-specific transforms, such as sampling and summarization. We implement a proof of concept software (wms.*) for the proposed solution and perform experiments on real sensor data from the NASA Infrared Telescope Facility at the University of Hawaii, to assess these resilience levels in practice. Our method proves to be well suited for this new domain. For example, we can recover an over 97% confidence watermark from a sampled (e.g. less than 8%) stream. Similarly, our encoding ensures survival to stream summarization (e.g. 20%) and random alteration attacks with very high confidence levels, often above 99%.

1 Introduction

Digital Watermarking aims to protect a certain content from unauthorized duplication and distribution by enabling provable ownership over the content. It has traditionally [8] [11] [17] relied upon the availability of a large noise domain within which the (usually multimedia) object can be altered while retaining its essential properties. For example, the least significant bit of image pixels can usually be arbitrarily altered with little impact on the visual quality of the image (as perceived by a human). In fact, much of the "bandwidth" for inserting watermarks in multimedia objects (such as in the least significant bits) is due to the inability of the human sensory system (especially sight and hearing) to detect minor changes.

Protecting rights over outsourced digital content becomes essential when considering areas where the data is sensitive and valuable. One example is the outsourcing of data for data mining. In this scenario data is

*Portions of this work were supported by Grants EIA-9903545, IIS-0325345, IIS-0219560, IIS-0312357, IIS-9985019 and IIS-0242421 from the National Science Foundation, Contract N00014-02-1-0364 from the Office of Naval Research, by sponsors of the Center for Education and Research in Information Assurance and Security, and by Purdue Discovery Park's e-enterprise Center.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the VLDB copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Very Large Data Base Endowment. To copy otherwise, or to republish, requires a fee and/or special permission from the Endowment.

Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004

produced/collected by a data collector and then sold to parties specialized in mining it. Given the nature of most of the data, it is hard to resiliently associate rights of the originator over it. Watermarking can be used to solve this issue.

Watermarking, as a rights protection method, works by inserting an indelible mark in the object such that (i) the insertion of the mark does not destroy the value of the object (i.e. the object is still useful for the *intended purpose*); and (ii) it is difficult for an adversary to remove or alter the mark beyond detection without destroying the value of the object. Clearly, the notion of value or utility of the object is central to the watermarking process. This value is closely related to the type of data and its intended use. For example, in the case of software the value may be in ensuring equivalent computation, and for text it may be in conveying the same meaning (e.g. synonym substitution is acceptable).

A considerable amount of effort has been invested in the problem of watermarking multimedia data (images, video and audio). More recently, the focus of watermarking for digital rights protection is shifting toward other data domains such as natural language text [2], software, algorithms [7] [16] and relational data [12] [20], [19]. Since these data domains often have very well defined restrictive semantics (as compared to those of images, video, or music) and may be designed for machine ingestion, the identification of the available "bandwidth" for watermarking is as important a challenge as the algorithms for inserting the watermarks themselves.

In this paper we introduce and study the problem of watermarking streaming data, which to the best of our knowledge, has not been addressed. Streaming data sources represent an important class of emerging applications [3] [4]. These applications produce a virtually endless stream of data that is too large to be stored in a given system. Examples of streaming data include output from environmental sensors such as temperature, pressure, and brightness readings, and stock prices. Recent efforts in the broader area of streaming data, deal with the database challenges of its management [5] [9] [10] [14]. Existing work on discrete data watermarking relies upon the availability of the entire dataset during the watermarking process. While this is generally a reasonable assumption, it does not hold true for the case of streaming data [3]. Moreover, since the streamed data is typically available as soon as it is generated, it is desirable that the watermarking process be applied immediately on subsets of the data. Due to this limitation, earlier work on watermarking relational databases is not applicable to streams.

But why is watermarking streaming data important? Couldn't we simply apply a watermarking technique on the data once it is stored? While this surely would work and enable rights protection for the stored result, it would not deter a mallicious customer (Mallory), with direct stream access, to duplicate segments of the stream and re-sell them or simply re-stream the data for profit. Thus the main rights protection scenario in this framework (see Figure 1) is to prevent exactly such leaks from a licensed customer to a third party.

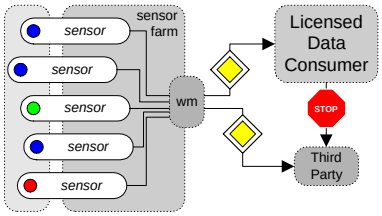

Figure 1: Sensor Streams Watermarking Scenario.

Our contributions include (i) the proposal and definition of the problem of watermarking sensor streams, (ii) the discovery and analysis of new watermark embedding channels for such data, (iii) the design of novel associated encoding algorithms, (iv) a proof of concept implementation of the algorithms and (v) their experimental evaluation. The algorithms introduced here prove to be resilient to important domain-specific classes of attacks, including stream re-sampling, summarization (replacing a stream portion by its average value) and random changes. For example, sampling the data stream down to less than 8% still yield a court-time confidence of watermark embedding of over 97%. Summarization (20%) and random data alterations are also survived very well, often with a false-positive detection probability of under 1%.

The paper is structured as follows. Section 2 outlines the major challenges in this new domain. It proposes an appropriate data and transform model, discusses associated attacks and overviews related work. In Section 3 an initial solution is provided. Further resilience-enhancing improvements and attack handling capabilities are gradually introduced in Section 4. Section 5 analyzes court-time convince-ability of our solution, discusses various aspects of the algorithms and proposes improvements for particular scenarios. Section 6 presents **wms.***, a proof-of-concept java implementation of our solution; our experimental setup and results are introduced. Section 7 concludes.

2 Challenges

2.1 Model

For the purpose of simplicity let us define a simple data stream as an (almost) infinite timed sequence of $(x[t])$ values "produced" by a set of data sources of a particular type (e.g. temperature sensors, stock market data). We do not consider simultaneous types of data sources here. $x[t]$ is a notation for the value yielded

by our source(s) at time t. Unless specified otherwise, lets denote a stream as $(x[], \varsigma)$ where ς is the number of incoming data values per time unit, i.e. the stream *data rate*.

Note: While a time-stamp t can be assigned naturally to each and every data value when produced by a data source, it often becomes irrelevant after such domain-specific transformations as sampling and summarization which destroy the exact association between the value $x[t]$ and the time it was initially generated, t. Thus, the notation $x[t]$ is merely used to distinguish separate values in the stream and is not intended for suggesting the preservation of the time-stamp-value in the resulting stream which is ultimately just a sequence of values.

Any stream processing performed is necessarily both time and space bound. The time bounds derive from the fact that the processing has to keep up with incoming data. We are going to model the space bound by the concept of a window of size ϖ. At each given point in time, no more than ϖ of the stream $(x[t])$ values (or equivalent amounts of arbitrary data) can be stored locally, at the processing point. Unless specified otherwise, as more incoming data becomes available, the default behavior of the window model is to "push" older items out (i.e. to be transmitted further, out of the processing facility) and "shift" the entire window (e.g. to the right) to free up space for new entries.

Note: For simplicity purposes, in this paper we are considering streams with fixed data rates. Although, intuitively, all the methods and approaches introduced apply also in the variable data rates case, a validation of this claim is to be subject to future research.

For the purpose of the current framework, we define the *uniform random sampling* of degree χ of a stream $(x[], \varsigma)$ as another stream $(x'[], \varsigma')$ with $\varsigma' = \frac{\varsigma}{\chi}$ such that for each sample data item $x'[t]$, there exists a contiguous subset of $(x[])$, $(x[t_1], x[t_2])$ such that $x'[t] \in (x[t_1], x[t_2])$, $\{x'[t-1], x'[t+1]\} \nsubseteq (x[t_1], x[t_2])$, and t is uniformly distributed in (t_1, t_2). In other words, a uniform random sampling is constructed by randomly choosing one value out of every χ values in the original stream.

A subtle variation of *uniform random sampling* is the case when $x'[t]$ is not randomly chosen but rather always the first element in it's corresponding χ sized subset (e.g. $t = t_1$). We call this *fixed random sampling* of degree χ.

We define the *summarization* of degree ν of a stream $(x[], \varsigma)$ as another stream $(x'[], \varsigma')$ with $\varsigma' = \frac{\varsigma}{\nu}$ such that for each two adjacent sample data items $x'_1[t], x'_2[t+\nu]$, there exist two contiguous, adjacent, non-overlapping ν-sized subsets of $(x[])$, $(x[t-\nu+1], x[t-\nu+2], ..., x[t])$, $(x[t+1], x[t+2], ..., x[t+\nu])$ such that $x'_1[t] = \frac{\sum_{i\in(1,\nu)} x[t-\nu+i]}{\nu}$ and $x'_2[t+\nu] = \frac{\sum_{i\in(1,\nu)} x[t+i]}{\nu}$. In other words, for each continuous chunk of ν elements from the original stream summarization outputs its average.

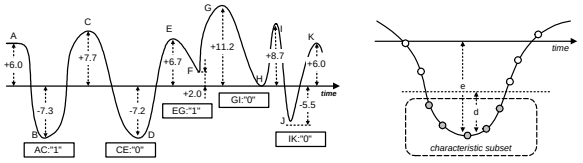

Figure 2: (a) A sample stream. If all the extremes are considered to be major, then the resulting label bits for K are shown (for $\varrho = 2$, Section 4.1) (b) δ-Radius characteristic subset of extreme η.

We define an *extreme* η in a stream simply as either a local minimum or local maximum value. If the stream is considered normalized within the (-0.5,0.5) interval, then we define the extreme's *characteristic subset of radius* δ, noted $\Xi(\eta, \delta)$ (see Figure 2 (b)), as the subset of stream items forming complete "chunks", immediately adjacent to η and conforming to the following criteria: item i, with value $v_i \in \Xi(\eta, \delta)$ iff $|\eta - v_i| < \delta$ and all the items "between" i and the extreme η, also belong to $\Xi(\eta, \delta)$.

A *major extreme* of degree χ and radius δ is defined as an extreme η such that at least one item in $\Xi(\eta, \delta)$ can be found in *any* uniform random sampling of degree χ of $(x[])$ (i.e. some items in $\Xi(\eta, \delta)$ "survive" sampling of χ degree). For example, in Figure 2 (a), intuitively, it seems likely that extremes such as F,I and J are going to have less chance of surviving sampling than C,E or G. This is so because of the temporal shape of the stream's evolution. C,E,G seem to yield characteristic subsets much "fatter" than F,I,J.

Note: Intuitively, δ needs to be chosen such that the characteristic subsets are going to be of an average size greater than χ in order to guarantee survival to a sampling of degree χ. Remember that in a sampling of degree χ every χ elements in the original stream are replaced by a single one in the result.

To model the "fluctuating" nature of a stream, we define $\varepsilon(\chi, \delta)$ as the average number of stream data items encountered/read per *major extreme* (i.e. before encountering a major extreme) of degree χ and radius δ. $\frac{1}{\varepsilon(\chi,\delta)}$ defines the average "frequency of major extremes" in terms of the number of observed data items.

By notation, for any value x let $b(x)$ be the number of bits required for its accurate representation and $msb(x, b)$ its most significant b bits. If $b(x) < b$ we left-pad x with $(b - b(x))$ zeroes to form a b-bit result. Similarly, $lsb(x, b)$ is used to denote the least significant b bits of x. Let wm be a watermark to be embedded, $wm[i]$ the i-th bit of wm.

In our solution we leverage a special de-facto secure construct, the one-way cryptographic hash. Let $crypto\_hash()$ be a cryptographic secure one-way hash.

Of interest are two properties of such a crypto-hash: (i) it is computationally infeasible, for a given value V' to find a V such that $crypto\_hash(V) = V'$; this assumption of one-wayness lies at the heart of many current security protocols, and (ii) changing even one bit of the hash input causes random changes to the output bits (i.e. roughly half of them change even if one bit of the input is flipped). Examples of potential candidates for $crypto\_hash()$ are the MD5 (used in the proof of concept implementation) or SHA hash. For more details on cryptographic hashes consult [18]. By notation, let $H(V, k) = crypto\_hash(k; V; k)$ (where ";" denotes concatenation).

2.2 Attacks

As outlined above, the nature of most "fast" time-series data applications imposes a set of strict requirements on any on-the-fly data processing method, such as watermarking. For one, it has to be able to keep up with the incoming data rate and, the fact that only a finite window of memory (e.g. of size ϖ) is available for processing makes certain history-dependent computations difficult or simply impossible. At the same time, any quality preservation constraints can be formulated only in terms of the current available data window; including any history information will come at the expense of being unable to store as much new incoming data.

Moreover, the effectiveness of any rights protection method is directly related to its ability to survive normal domain specific transformations as well as malicious attacks. There are several transforms relevant in a streaming scenario, including the following: (A1) summarization, (A2) sampling, (A3) segmentation (we would like to be able to recover a watermark from a finite segment of data drawn from the stream), (A4) linear changes (there might be value in actual *data trends*, that Mallory [1] could still exploit, by scaling the initial values), (A5) addition of stream values and (A6) random alterations.

While we discuss most of these and other attacks in the next sections, let us note here that a scaling attack (A4) can be handled by an initial normalization step, e.g. yielding values in the $(-0.5, 0.5)$ interval. If the data distribution is assumed to be known, normalization can also be easily performed at detection time. If data distribution is not known, then we propose an initial "discovery" run in which data is simply read and a reference data distribution is constructed and updated on the fly. This will yield a certain data-dependent inaccuracy in the initial phases of detection but will likely quickly converge as more data is read. If detection is performed offline on a static segment of data, normalization is eased by the ability to read the data multiple times. In the following, unless specified otherwise, we consider this normalization step to have been performed, yielding a normalized version of the stream, with values in the interval $(-0.5, 0.5)$. To survive sampling and other minor stream transformations, several improvements to the normalization process are proposed in Section 3.2. With respect to (A5), one observation to be made is that Mallory is bound to add only a limited amount of data (in order to preserve the value in the original stream) and these new values are to be drawn from a similar data distribution, lest they become easy to identify in the detection process as not conforming to the original, known, distribution. (A6) is naturally modeled by a combination of (A2) and (A5).

Apparently, data re-sorting might be also of concern as an attack. At closer inspection however, if value is to be found in the stream, it is assumed to lie in two aspects of it: the data values and their relative ordering. In other words, in most applications, a recorded stream (even sampled) is only valuable if its re-play is preserving the relative ordering of the values (with exception of some extreme cases). Re-ordering the sequence of values in the stream is going to significantly alter its core value. For example consider the case of stock market data. If the evolution of a given stock is modeled by a stream of values, a recording of it is only valuable if the sequence ordering is preserved. Also, significant on-the-fly data re-sorting, is simply not possible given the finite processing window and speed assumptions. In this paper we consider data re-sorting to significantly alter the core value of the data set, not a successful attack choice Mallory would consider. Our method does however handle minor data re-sorting gracefully.

2.3 Related Work

Could existing work in non-media data sets watermarking such as relational data [12] [20] be adapted to the new domain? The work by Sion et al in [20] requires access to the entire data set in an almost random access model, which is certainly not possible here at embedding time. Also, both efforts seem to make extensive use of the existence of a primary key, thus rendering a direct adaptation impossible. Nevertheless it might be worth noting that, if a primary key is assumed to exist, e.g. if there is a guarantee that the time-stamp information for each stream value is going to be preserved in the result, then the bit alteration method proposed by Kiernan et al in [12] could be adapted to work on a single attribute, namely the stream value. The result would likely be resilient to (time-stamp preserving) sampling, but not to any of the other attacks.

But what about multimedia watermarking? Given the "streaming" nature of our data, would it not be possible to simply adapt an existing audio (or media) watermarking algorithm [6] [8] [13] [17] [21] since audio data is also an example of a data stream? In other

[1] The traditional name of *the* maliciously acting party.

words, why is our problem different? While there seem to be similarities between watermarking audio and sensor data for example, at a closer inspection these similarities prove to be just appearances. A multitude of differences are to be found between the two frameworks mainly deriving from different data models and associated semantic scopes.

While, in theory, a sensor stream could be viewed as an audio signal for example and processed as such, for all practical purposes such an approach would not suit reality and/or often yield undesired results. For example, while in sensor data streams, summarization and sampling are routinely expected natural operations, audio streams are not to be summarized, and sampling in the audio domain entails an entirely different semantic. Data quality to be preserved in audio streaming is usually related to the human auditory system and its limitations. Any watermark-related alteration can be induced as long as the stream still "sounds" good [2]. In the case of sensor streams (e.g. temperature) on the other hand, many scenarios involve widely different quality metrics, that often need to also consider overall stream characteristics [3].

Space constraints prevent a more in depth comparison. In summary, while experiences in the multi-media domain are valuable, due to the nature of this new application domain, a solution for watermarking sensor streams needs to be naturally suited to handle attacks and transformations such as the ones outlined in Section 2.2.

3 An Initial Solution

This Section outlines the main solution and then gradually improves it to a more robust and resilient version, by identifying potential flaws and their associated fixes.

3.1 Overview

The first issue to be considered when watermarking in such a framework are the data assumptions that the detection process is expected to handle. More specific, are we concerned with (i) an on-the-fly streaming detection process or (ii) the ability to detect a watermark offline, in a static "chunk" of data (with associated multiple-pass, random access), likely a subset of the original stream? These two different scenarios apparently feature distinct challenges. Intuitively a watermarking solution for (ii) could potentially yield an increased detection accuracy (with respect to the same amount of data), due to the ability to repeatedly iterate on the entire data set, without restrictive time bounds. Because any on-the-fly solution can be directly applied to (ii), for the time being let us consider a solution for (i). In Section 5.6 we analyze the offline case.

At an overview level, watermark embedding proceeds as follows: (a) first a set of "major" extremes (actual stream items) are identified in the data stream, extremes that feature the property that they (or a majority thereof) can be recovered after a suite of considered alterations (possibly attacks) such as (random) sampling and summarization. Next (b) a certain criteria is used to select some of these extremes as recipients for parts of the watermark. Finally (c), the selected ones are used to define subsets of items considered for 1-bit watermark embedding of bits of the global watermark. The fact that these extremes can be recovered ensures a consistent overlap (or even complete identity) between the recovered subsets and the original ones (in the un-altered data). In the watermark detection process (d) *all* the extremes in the stream are identified and the selection criteria in step (b) above is used once again to identify potential watermark recipients. For each selected extreme, (e) its corresponding 1-bit watermark is extracted and ultimately the global watermark is gradually re-constructed, by possibly also using an error correction mechanism such as majority voting.

Thus, one of the main ideas behind our solution is the use of extreme values in the stream's evolution as watermark bit-carriers. The intuition here lies in the fact that much of the stream value lies in exactly its fluctuating behavior and the associated extremes, more likely to be preserved in value-preserving, domain-specific transforms.

3.2 Embedding

Using the notation in Section 2.1, let $\alpha, \beta \in \mathbb{Z}$ such that $\alpha + \beta \leq b(x[])$, where $b(x[])$ is the bit-size of the values in the considered stream $(x[])$. Let $\delta, \chi \in (0, 1)$. δ will be chosen such that all elements in any characteristic subset $\Xi(\eta, \delta)$ have the same most significant α bits ($\delta < 2^{(b(x[])-\alpha)}$). $\alpha, \beta, \delta, \chi$ are secret. We use the term "advance the window" to denote reading in more new data items while discarding old ones from the current data window.

```
wm_embed(δ,α,β,wm,k1,φ)
    while (true) do
        η ← first major extreme in win[]
        compute Ξ(η, δ)
        i ← H(msb(η, α), k1) mod φ
        if i ≤ b(wm) then
            bit ← H(msb(η, α), k1) mod β
            foreach v ∈ Ξ(η, δ) do
                v[bit − 1] ← false
                v[bit] ← wm[i]
                v[bit + 1] ← false
        advance win[] past η
```

Figure 3: Initial Embedding Algorithm

[2] Incidentally, these types of quality metrics are also well suited for "window only" data processing.

[3] e.g. the total alteration introduced per data item should not exceed a certain threshold.

In the initial step of our embedding algorithm we first identify the first major extreme of degree χ and radius δ in the current window. The assumption here is that there exists a major extreme in the current window. If this is not the case, we can simply advance the window until we find one. The "majority" of an extreme can be easily evaluated by comparing the size of its characteristic subset $\Xi(\eta, \delta)$ with the sampling degree χ. The characteristic subset containing at least χ elements guarantees that in a random sampling of degree χ, at least one of those elements is going to survive. If no major extremes can be found for given δ and χ values, one could consider instead extremes with characteristic subsets smaller than χ that guarantee an acceptable chance (e.g. 70%) of survival in case of sampling (i.e. $\frac{subset\_size}{\chi} > 70\%$?).

Note: δ and the desired values for χ can be adjusted such that eventually (in the extreme) all characteristic subsets feature enough elements to survive a sampling of degree χ. We should not forget though that we also aim to minimize the amount of change introduced. Thus an ideal choice for δ would yield just enough major extremes with characteristic subsets large enough to survive the required level of sampling but no more. This is a fine data dependent trade-off that needs to be considered in practice.

Once a major extreme (η) is identified in the current window, in the second step, a *selection criterion* is used in determining weather η is going to be used in the embedding process or not. If $H(msb(\eta, \alpha), k_1) \bmod \phi = i$ and $i \leq b(wm)$, then η is considered for embedding bit i of the watermark, $wm[i]$. $\phi \in (b(wm), b(wm) + k_2)$ ($k_2 > 0$) is a secret unsigned integer fixed at embedding time, ensuring that only a limited number (a ratio of $\frac{b(wm)}{\phi}$) of these major extremes are going to be selected for embedding. We used this "fitness" selection criteria also in [19]. It is a powerful tool, deriving strength from both the one-wayness and randomness properties of the deployed one-way cryptographic hash, forcing Mallory into a "guessing" position with respect to watermark encoding location. The reason behind the use of the most significant bits of η in the above formula, is resilience to minor alterations and errors due to sampling. As discussed above, the assumption is that for any value $x \in \Xi(\eta, \delta)$, $msb(x, \alpha) = msb(\eta, \alpha)$.

If the previous step resulted in η being selected, the next step embeds the $wm[i]$ bit into $\Xi(\eta)$. This process is performed as follows. First, a certain bit position $bit = H(msb(\eta, \alpha), k_1) \bmod \beta$. is selected for embedding. Next, for each value $v \in \Xi(\eta, \delta)$ and in η itself, that bit position is set to $wm[i]$ and the adjacent bits are set to false (to prevent overflow in case of summarization). In other words $v[bit - 1] = false$, $v[bit] = wm[i]$ and $v[bit + 1] = false$. The reasoning behind modifying an entire subset of items ($\Xi(\eta, \delta)$) is to survive summarizations. This is the case if the bit encoding is such that the average of any combination of ($\nu < |\Xi(\eta)|$ or less) items in $\Xi(\eta, \delta)$, would preserve the embedded bit. It is easy to show that this is indeed the case. Finally, the window is advanced past η and the process re-starts.

3.3 Detection

For each bit $wm[i]$ in the original watermark wm, let $wm[i]^T$ and $wm[i]^F$ be "buckets" (unsigned integers) which are incremented accordingly each time we recover a corresponding true/false bit $wm^{det}[i]$ from the stream. In other words, if the detection process yields at some point $wm^{det}[i] = false$, then the $wm[i]^F$ value is incremented. Similarly, for $wm^{det}[i] = true$, $wm[i]^T$ is incremented. At the end of the detection process, the actual $wm[i]$ will be estimated by the difference between $wm[i]^T$ and $wm[i]^F$, i.e. if $wm[i]^T - wm[i]^F > \upsilon$ then the estimated value for this particular bit becomes $wm^{est}[i] = true$ and conversely if $wm[i]^F - wm[i]^T > \upsilon$ then $wm^{est}[i] = false$, where $\upsilon > 0$. It is to be noted that if detection would be applied on random, un-watermarked data, the probability of detecting $wm^{det}[i] = false$ would equal the probability of $wm^{det}[i] = true$, thus yielding virtually identical (υ is used to distinguish this exact case) values for $wm[i]^T$ and $wm[i]^F$. In this case, $wm^{est}[i]$ would be un-defined, thus the data considered un-watermarked. The watermark detection process effectively relies on discovering a statistical bias in the $true/false$ distribution for each detected watermark bit.

```
wm_detect(δ,α,β,wm,k1,φ)
 while (true) do
   η ← first extreme in win[]
   i ←H(msb(η, α), k1)modφ
   if i ≤b(wm) then
     bit ←H(msb(η, α), k1)modβ
     if (η[bit]==true) then
       wm[i]^T ←wm[i]^T + 1
     else
       wm[i]^F ←wm[i]^F + 1
   advance win[] past η
```

```
wm_construct(wm[]^T,wm[]^F,υ)
 for (i ← 0;i < b(wm);i ← i + 1)
   if (wm[i]^T − wm[i]^F > υ) then
     wm[i] ←true
   else
   if (wm[i]^F − wm[i]^T > υ) then
     wm[i] ←false
   else
     wm[i] ←undefined
 return wm[]
```

Figure 4: Initial Detection Algorithm

Detection starts by identifying the first extreme η in the current window. The selection criteria deployed in the embedding phase is tested on η. If $H(msb(\eta, \alpha), k_1) \bmod \phi = i$ and $i \leq b(wm)$, then η was likely used in embedding bit i of the watermark, $wm[i]$. This bit is then extracted from bit-position $H(msb(\eta, \alpha), k_1) \bmod \beta$ and depending on its value, the corresponding bucket $wm[i]^T$ or $wm[i]^F$ is incremented. Finally, as in the embedding case, the window is advanced past η and the process re-starts.

The detection process does not consider only "major" extremes but rather any and all extremes that can be identified in the stream. The reason behind this is the fact that the stream could have been subjected to sampling (A2) and/or summarization (A1)

in the meantime. Considering "major" extremes only and their corresponding characteristic subsets in the embedding phase was a means to ensure survival to exactly such transformations. Nevertheless, the detection process apparently suffers now from the fact that it also considers extremes that were potentially not watermarked in the first place, possibly yielding false watermark readings. At a deeper insight, it becomes clear that this does not constitute a problem. As the watermark reconstruction problem relies on a statistical bias and as this bias is zero in the case of random data (as discussed above), introducing new, random, un-watermarked data points into the detection does not affect the watermark-induced bias at all. This is yet another reason why this embedding will prove resilience to data addition (A5).

4 Improvements

We now discuss improvements to the initial solution, aimed at boosting its resilience level.

4.1 De-correlation

One particular issue of concern in the above solution is the fact that because there exists a correlation between the watermarking alteration (the $wm[i]$ bit) and its actual location (determined by $H(msb(\eta, \alpha), k_1)$)), Mallory can mount a special attack with the undesirable result of revealing the mark embedding locations. The attack proceeds by first realizing that, despite the one-wayness of the deployed hash function $H()$, in fact, η is the only variable that determines *both* the bit embedding location as well as its value. If Mallory would be able to check for this correlation for each encountered extreme, it would quickly lead to exposing each one carrying a watermark bit. But how does he check for the correlation? Mallory can simply build a set of "hash buckets" for each separate value of $msb(\eta, \alpha)$ (if α is secret the job becomes harder but not impossible) and count, for each extreme η encountered, which of the lower β bits of η is set (resp. reset) more often. For each η for which a bias in one bit position is determined, that particular bit position is considered as carrying a watermark. Mallory can then simply randomize that position throughout the considered set of extremes, effectively erasing the global watermark.

Thus, the problem lies here in the correlation between the actual bit location and the bit value, correlation induced by the fact that a single variable (η) determines both of these. A fix would ideally rely on a separate source of information to determine for example the location of the embedded bit, independently of the bit value. Also, this source of information would need to be consistently recoverable at detection time. For example, if time-stamp information would be assumed available, i.e. if all the processing and the attacks on the data stream could be assumed to preserve the time-stamp to value association then the actual time-stamp would present an ideal candidate, effectively labeling each and every stream extreme uniquely while at the same time not being correlated (directly) to these extremes. This unique label could then be used in computing the bit position for watermark embedding. In the selection of the bit embedding location, instead of using $bit = H(msb(\eta, \alpha), k_1) \bmod \beta$ which yields a result correlated to the actual embedded bit value ($wm[i]$, where $i = H(msb(\eta, \alpha), k_1) \bmod \phi$) we propose to use $bit = H(msb(label(\eta), \alpha), k_1) \bmod \beta$ where $label(\eta)$ is the (virtually) unique label of extreme η. A labeling scheme like this would make "bucket counting" attacks impossible. In our model however, timestamps are not assumed to be preserved. Can we maybe envision a different labeling scheme (at least) for extremes, that would survive the attacks and transformations outlined in Section 2.2? We propose to build it from scratch.

Because the data can be subject to both sampling and summarization and we would like to enable watermark detection also from a finite segment of the data (see Section 2.2), this task becomes especially challenging. Sampling and summarization are already survived (by design) by the extremes selected using the "majority" criteria in Section 2.1. We could maybe make use of this fact in the labeling scheme. The more challenging aspect becomes clear when one considers data segmentation. To support segmentation, the labeling scheme needs to function based solely on information available close (in terms of stream location) to the considered to-be-labeled extreme. Also, labels computed at detection time from potential segments of sampled and/or summarized data, need to (at least) converge to the original ones, as more and more watermarked data is available.

Let λ be the bit length of the labels resulting in our labeling scheme. Let $\varrho > 1$ be an unsigned integer. λ and ϱ are to be secrets, fixed at embedding time. We propose the following labeling scheme for extremes. For each extreme i (denoted by its index in the set of extremes seen so far), let $msb(abs(val(i)), \alpha)$ be the α most significant bits of that extreme's absolute (normalized) data value. Given two extremes i and a subsequent $i + \varrho$, we define $label\_bit(i, i + \varrho) = true$ iff $msb(abs(val(i)), \alpha) < msb(abs(val(i + \varrho)), \alpha)$ and $false$ otherwise. Then the label for extreme $i + \lambda$, $label(val(i + \lambda))$ is defined by the bit string composed of the concatenation of "1" (binary true) followed by each and every $label\_bit(j, j + \varrho)$ in ascending order of $j \in (i, i + \lambda)$. In other words, an extreme is labeled by a certain differential interpretation of some of the preceding extremes' values. For example, in Figure 2 (a), the label for extreme **K** becomes "110100" ($\varrho = 2$). The main role of ϱ's secrecy is to hide the actual labeling scheme locations from a potential attacker, making a random-alteration attack necessarily more damaging to the value of the data, thus increas-

ingly un-successful. To illustrate this, consider for example the case where Mallory knows that $\varrho = 2$. Now all it needs to do is alter any and only two successive extremes (in any continuous chunk of 2λ extremes), just enough to flip one label bit. But now, if ϱ is secret, Mallory *has* to alter a larger, arbitrary number of successive extremes. Further improvements are discussed in Section 5.7.

Before going any further, let us analyze what happens if an important extreme is "lost". In other words, if one extreme i is altered so much that its α most significant bits flip the $msb(abs(val(i)), \alpha) < msb(abs(val(i + \varrho)), \alpha)$ inequality, corrupting its corresponding label bit. What happens is in fact not too damaging. The labels that were constructed considering this particular extreme will be corrupted, until the detection process encounters again a continuous sequence of extremes not altered beyond recognition. We have to realize that Mallory cannot afford altering extremes to such extents, and the fact that ϱ is secret makes a random alteration attack the only choice.

In summary, the main purpose of such a labeling scheme is to ensure that Mallory cannot mount the "bucket counting" type of statistical analysis attack as outlined above. Different labels for adjacent extremes together with the use of one-way hashing completely defeat such an attack. The labeling scheme provides an independent, un-correlated source of information for determining the bit position to be altered. Remember that our ability to survive "bucket counting" type of attacks was dependent on the labels being un-correlated with respect to the actual extreme values, while at the same time being virtually unique for each extreme.

4.2 Repeating Labels

But the finite nature of the considered bit size of the label poses a certain problem in this respect by necessarily allowing for duplicates (e.g. in the optimal case only due to "wrap-around" of the λ-sized space) if the considered data segment is small. For example if $\lambda = 10$ and we label 2000 extremes, on average, if we are lucky we will have each label repeated only roughly twice. A more complex analysis needs to also include data-time behavior, e.g. what is the likelihood of low to high vs. high to low transitions, given the considered ϱ? If there is a bias in this data behavior then the resulting labels are going to contain possibly more one-bits than zeroes etc. Nevertheless, in summary our problem is now that, because some labels might repeat themselves, an unfortunate circumstance could make it such that enough data for a particular label becomes available for Mallory to mount yet again a "bucket counting" attack.

There are two fixes for the above issue. First (i) the selected size of the considered labels could be kept secret, within a certain range (e.g. $\lambda \in (10, 20)$). There is a trade-off here between the ability to converge in case of data loss and a higher λ value, but for $\lambda = 20$ and $\varrho = 3$ for example, roughly 3 million extremes need to pass by before a label is going to be repeated. Second (ii) once the un-correlated nature of the labels has been established by their independent information source, we can re-consider the use of the most significant bits of the extreme values. If we redefine the labels as a concatenation between the initial $label\_bit(j, j + 1)$-derived labels bit string and $msb(abs(val(i)), \alpha)$ we (arguably) significantly decrease the probability of duplicates.

4.3 Reconstructing Labels

In the initial algorithm, the detection process relied entirely on proving a certain statistical bias in the underlying data. Labeling, while providing a defense for the correlation attack, introduces the requirement to be able to identify major extremes at detection times, possibly in a summarized and/or sampled stream. This becomes a challenge as the definition of "major" does not make sense anymore in the context of a sampled version of the original stream.

We propose the following solution. In a first stage, the degree of the transformation performed is determined. In a second stage, the definition of majority of an extreme is updated to reflect the fact that the considered stream is already transformed. A major extreme of degree χ and radius δ in the original stream $(x[], \varsigma)$, becomes a major extreme of degree $\frac{\chi}{\gamma}$ and radius δ in the transformed stream $(x'[], \frac{\varsigma}{\gamma})$, where γ is the degree of the transformation (e.g. summarization, sampling) applied to $(x[], \varsigma)$. Once we know γ identifying major extremes in the transformed stream is simply a matter of considering this updated definition.

But how do we determine γ, the degree of the transformation applied to the stream? In a dynamic stream, with consistent stream data rates, γ can be determined by simply dividing the original stream rate to the current (transformed) stream rate, $\gamma = \frac{\varsigma}{\varsigma'}$. The more challenging scenario is to determine the value of γ corresponding to a stream for which only a segment is available. In other words, given a certain segment of a transformed stream $(x'[], \varsigma')$, corresponding to an original stream $(x[], \varsigma)$, how do we determine the degree of the transform(s) applied to $(x[], \varsigma)$?

A reasonable assumption that can be made is that the transform was applied uniformly to the entire stream, in other words, the entire segment is consistently transformed throughout (with respect to the original stream data). In this case, one solution would start by preserving some information about the initial stream, namely the average size of the characteristic subsets of extremes, for a given δ. Then, in the transformed segment, extremes are identified and their average characteristic subset size for the same δ is computed. It is to be expected (arguably) that in

a transformed (sampled and/or summarized) stream these sizes would shrink according to the actual transform degree. Dividing the original average characteristic subset size by the sampled stream average would thus yield an estimate of the transform degree γ. In our proof of concept implementation this method is used successfully. Space considerations prevent further elaboration.

4.4 Hysteresis

The labeling features yet another interesting challenge. While ϱ's secrecy indeed makes it more difficult on Mallory to precisely alter extremes so as to flip label bits, what is to stop him from still altering a large number of consecutive extremes with the same purpose? This attack is likely not of much concern as the assumption is that Mallory cannot afford such modifications throughout the data as the required modifications to flip several consecutive bits are likely quite significant. Unfavorable data distribution and data semantics preservation are further arguments that Mallory would not be able to deploy such an attack.

Nevertheless, a solution is available and we propose its use. It proceeds by changing the labeling scheme as follows: given two extremes i and $i + \varrho$, we define $label\_bit(i, i + \varrho) = true$ iff $(msb(abs(val(i))) - msb(abs(val(i+\varrho)))) < \iota^- < 0$ and $label\_bit(i, i+\varrho) = false$ iff $0 < \iota^+ < (msb(abs(val(i))) - msb(abs(val(i+\varrho))))$. As can be seen, these new formulas induce a hysteresis (defined by (ι^-, ι^+)). Now Mallory is not only presented with the dilemma of which extremes to alter but also unable to determine what the minimum change is that would flip the label's corresponding bit.

4.5 Detecting Bias

But what prevents Mallory from identifying all the major extremes for which there exists a majority of (possibly all) items in the characteristic subset with a certain bit position set to the same identical value? These extremes would then be (rightfully so) considered watermark carrying and Mallory could mount a simple attack of randomizing those bit positions. This is a serious attack and threatens the validity of the entire watermarking scheme.

How can we fix this while surviving summarization? Remember that the main reason behind embedding the same bit multiple times at the same position in different items in the characteristic subset was directly mandated by the requirement to survive summarization. We propose a new approach that survives summarization and results in alterations effectively appearing random to the eyes of an attacker. Let $\Xi(\eta, \delta) = \{x_1, x_2, ..., x_a\}$. For each $i \leq j \in [1, a]$, let $m_{ij} = \frac{\sum_{u \in [i,j]} x_u}{|j-i+1|}$. Then we define the *characteristic subset bit encoding convention* as follows: (i) we say that a bit value of "true" is embedded in $\Xi(\eta, \delta)$ iff $\forall j, i$ we have $lsb(H(lsb(m_{ij}, \beta), label(\eta)), \zeta) = 2^\zeta - 1$; similarly, (ii) we say that "false" is embedded iff $\forall j, i$ we have $lsb(H(lsb(m_{ij}, \beta), label(\eta)), \zeta) = 0$, where $\zeta > 0$ is a secret fixed at embedding time. The embedding method simply alters the least significant β bits in the values in $\Xi(\eta, \delta)$ until the criteria is satisfied for the desired to-be-embedded $wm[i]$ bit value. It is to be noted that these alterations should aim to minimize the Euclidean distance (or possibly any other desired distance metric) from the starting point defined by $\{x_1, x_2, ..., x_a\}$. We call this a "multi-hash encoding".

The use of m_{ij} ensures survival to summarization, while the cryptographic hash provides the appearance of randomness. But is it feasible to assume that one could find such a point in the a-dimensional space defined by the items in $\Xi(\eta, \delta)$? How many computations are required to at least find one? There are $\frac{a(a+1)}{2}$ possible m_{ij} averages (including all $m_{ii} = x_i$ values). For each we consider the last ζ bits of its hash, thus we effectively have an output space of $\zeta\frac{a(a+1)}{2}$ bits. The probability that a desired pattern occurs in this space is then $2^{-\zeta\frac{a(a+1)}{2}}$. Thus, on average, the expected number of configurations in the input space that would need to be tested in an exhaustive search before yielding one that results in the desired output, is $2^{\zeta\frac{a(a+1)}{2}}$. For example if $\zeta = 1$ and $a = 5$ we have 2^{15}, that is, approx. 32,000 computations would need to be performed (for each considered major extreme in the window).

If enough computation power is available with respect to the incoming stream data rate, larger values for ζ and a could be handled, resulting in an increased level of court-time persuasiveness. Nevertheless, given the exponential nature of the increase in required computations for an increasing number of items in the characteristic subset, it is probably not likely to be able to exhaustively handle subsets with more than $8 - 10$ items efficiently. While out of the scope of the current paper, the design and use of efficient pruned-space algorithms would be required to significantly reduce these requirements. Alternately, we could deploy a computation-reducing technique that limits the number of m_{ij} averages for which (i) or (ii) needs to hold in the subset bit encoding convention above. In other words, the search process (in the $\{x_1, x_2, ..., x_a\}$ space) will be stopped once a certain number of the m_{ij} averages feature the desired encoding convention ((i) or (ii)). We call these m_{ij} values "active". The resulting decrease in required computation time comes at the expense of decreased resilience to transforms. More specifically, the fact that the bit-embedding can only be "seen" through a limited number of "good" m_{ij}'s (which feature the appropriate subset bit encoding) makes it such that detecting the corresponding watermark bit in a transformed stream will fail if the stream does not contain at least one of the "good" m_{ij} values.

Note: If such a reducing technique is applied, a de-

sired property would be the ability to survive to as many levels of summarization as possible. Thus, after ensuring the subset bit encoding convention for every m_{ii} (original items, so as to survive also sampling), we propose to "divide" the remaining computing cycles so as to enable a non-zero probability of bit detection for any degree of summarization. This would be achieved, if for any considered summarization degree ν to be survived, there would exist at least one m_{ij} with $|j-i| = \nu$ (ensuring a non-zero probability of this average to appear in a ν-degree summarized stream) that allows the extraction of the associated watermark bit.

5 Discussion

5.1 Analysis

In this Section we are exploring a theoretical analysis of the vulnerability of our scheme under the following attack model: Mallory starts to modify randomly every a_1-th ($a_1 > 1$) extreme (η) in such a way as to alter a ratio of $a_2 \in (0,1)$ of the items in the extreme's characteristic subset of radius a_3, $\Xi(\eta, a_3)$. (Thus, on average, Mallory alters only one in every $a_1' = a_1\phi$ bit-carrying extremes).

The assumption here is that these alterations do not impact the asociated labeling scheme, in other words, they don't change the "greater than" relationship between extremes used in the labeling process. An extension considering this case is out of the current limited-space scope. Due to space constraints we are only going to focus on a more "informed" Mallory, aware of the characteristic subset radius used at encoding time. This will strengthen our derived bounds. In other words, we assume that $a_3 = \delta$ is known to Mallory, see Section 3.2.

We propose two ways to analyse the vulnerability of the proposed solution: (i) looking at how much an attack "weakens" the encoding, i.e. how many of the *active* m_{ij} values are actually destroyed divided by the total number of active ones (making it thus proportionally harder to detect a watermark in court) and (ii) what is the probability that *all* of the active ones are obliterated ? It can be proven that, for a given extreme η, for which $\Xi(\eta, a_3) = \{x_1, x_2, ..., x_a\}$ the number of corresponding m_{ij} values altered is $c_m = \frac{1}{2}aa_2(2a - aa_2 + 1)$.

Now, for (i) the "weakening" of the encoding can be defined as $c_m \times \frac{2}{a(a+1)}$, the ratio of m_{ij} values that are altered from the total number of potential active ones for each altered extreme. Because one in every $a_1' = a_1\phi$ bit-carrying extremes gets impacted, the overall "weakening" factor can be defined as $a_1 \times c_m \times \frac{2}{a(a+1)}$. To answer (ii) we first model this scenario by a sampling experiment without replacement. In this experiment, $x + t, t > 0$ balls are randomly removed from a bowl with a total of y balls. The question answered is: if the bowl contained exactly x balls what is the probability that the $x + t$ removals emptied the bowl of all y black balls. It can be shown that this is $P(x + t, x, y) = \frac{\binom{y-x}{t}}{\binom{y}{x+t}}$. In our model $(x + t) = c_m$, $y = a(a + 1)\frac{1}{2}$ and if $x = a_4 y$ (only a ratio of a_4 of the $a(a + 1)\frac{1}{2}$ m_{ij} values are active) we can compute the probability that *all* of them are altered.

Thus, for each attacked extreme we have a non-zero probability of altering all active m_{ij} values and removing the corresponding watermark bit. Next we ask, how do these alterations impact our ability to convince in court and detect a watermark bias in the resulting data? Because the alteration is necessarily random (the randomness of the one-way hashes in the encoding in Section 4.5 guarantee this) we can model the attack as essentially a random noise addition attack. Evaluating the resilience of any watermark bias becomes now a matter of asking how many of the embeddings actually survive until detection time. Are there enough of them to actually convincingly reconstruct the multi-bit watermark after error correction? In Section 5.3 we look at how the watermark bias becomes more convincing in time (and seen data). Loosing a fraction of the mark bit encoding extremes can be in fact seen as a reduction of the ϕ value (see Section 3.2). If for each of the $a_1' = a_1\phi$ bit carrying extremes that are altered by Mallory, the attack success probability is given by $P(x + t, x, y)$, we can perform a similar reasoning (Section 5.3) with a new $\phi' = \phi + a_1' \times P(x + t, x, y)$. What now happens is that the persuasiveness (court-time convince-ability) converges proportionally slower. In other words, we need to see $a_1 \times P(x + t, x, y)$ more stream data to be able to provide an equally convincing proof in court.

For example, for $a_1 = 5$, $a = 6$, $a_4 = 50\%$, $a_2 = 50\%$ we get the average probability $P(15, 10, 21) \approx 0.85\%$ of a complete alteration of all the active m_{ij} values at each extreme. This effectively translates in the need to see only an average of $a_1 \times P(x + t, x, y) \approx 4.25\%$ more data to be equally convincing at detection time.

5.2 Surviving Transforms

By construction the method introduced above certainly survives sampling (A2) up to a degree of $\chi_{max} = |\Xi(\eta, \delta)|$. Indeed this is so if at least one element in the characteristic subset of η is to be found in a sampling of degree χ_{max}. This element can be used in the detection process to recover the corresponding watermark bit for η. Higher degrees of sampling are also quite likely to be survived as there is a non-zero probability of elements in $\Xi(\eta, \delta)$ to be in the sampled stream even for $\chi > \chi_{max}$. Due to space constraints we do not elaborate further. The phenomenon is experimentally illustrated in Section 6.

Summarization (A1) up to a degree of $\nu_{max} =$

$|\Xi(\eta, \delta)|$ is also handled well by design, for example due to the use of m_{ij} in the bit-encoding procedure illustrated in Section 4.5. Any summarization of a degree $\nu \leq \nu_{max}$ naturally results in at least one of the m_{ij} averages being in the summarized stream. Even in the initial algorithm, the bit encoding pattern used on the elements in the characteristic subset ensured survival of the pattern in the process of averaging (thus surviving summarization) within the subset. Summarization is experimentally analyzed in Section 6.

How well is segmentation (A3) survived by our solution? More specifically, what is the minimum size of a stream segment from which we are able to recover the watermark? For simplicity let us assume a one-bit watermark, i.e. $b(wm) = 1$. In the following we are trying to determine the minimum required size of a contiguous watermarked stream segment that would enable a proof more "convincing" than a coin-flip stating that a watermark is embedded in the data. This proof would be obtained if we can correctly detect at least two consistent bits (equal to $wm[0]$) from two different extremes found in the segment. In that case, the probability of a false-positive becomes lower than a random coin-flip.

But what is the minimum amount of data we need to see to be able to decode two bits? In the best case, the two extremes are adjacent and we need to see enough data to build correct labels for those two extremes. To build the labels correctly, we need to have seen all the previous $\lambda\varrho$ major extremes correctly. Further qualitative analysis must be data dependent, for example if the fluctuating nature of the stream features a major extreme of degree χ and radius δ for every $\varepsilon(\chi, \delta)$ data items, then the minimum required size of a segment enabling watermark detection is $\varepsilon(\chi, \delta)\lambda\varrho$.

5.3 Persuasiveness

In this section we analyze the ability of our method to convince in court. This can be naturally expressed as follows: given a one bit (e.g. true) watermark, what is the probability of false positives (P_{fp}) for the watermark encoding? In other words, we ask: *What is the probability of a one-bit (true) watermark to be detected in a random data stream?* If this probability is low enough, then a positive detection would constitute a strong proof of rights, with a "confidence" of $1 - P_{fp}$. Here we define confidence as the probability that a given detected watermark was indeed purposefully embedded in the data by the rights owner.

Using the notation in Section 4.5, for each considered extreme η, the occurrence probability of a "good" corresponding m_{ij} (i.e. encoding "true" with respect to the bit encoding convention) in a random stream is naturally $\frac{1}{2}$, because of the cryptographic hash used in the encoding. There are $\frac{a(a+1)}{2}$ possible m_{ij} averages (including all $m_{ii} = x_i$ values). Because for each we consider the last ζ bits of its hash, we effectively have an output space of $\zeta\frac{a(a+1)}{2}$ bits. Thus the probability of the bit "true" being encoded consistently by all of these becomes $2^{-\zeta\frac{a(a+1)}{2}}$ (per extreme). Now, for each $\varepsilon(\chi, \delta)$ items there is a potential major extreme recipient of a one-bit encoding. Out of these how many are actually selected for encoding ? As discussed in Section 3.2 only a fraction of $\frac{1}{\phi}$ (because now b(wm)=1) of them are actually selected for embedding. Thus if ς is the stream data rate, we can determine the relationship between the time elapsed since we started reading the incoming stream (t) and the reached level of persuasiveness, as follows.

If $\varepsilon(\chi, \delta)$ models the average number of items that need to be read before a major extreme is encountered, then $\frac{\varepsilon(\chi,\delta)}{\varsigma}$ represents the average time-interval "between" major extremes. But only $\frac{1}{\phi}$ of the major extremes are selected for embedding, and so the time-interval between two major extremes that encode the watermark is $\frac{\phi\varepsilon(\chi,\delta)}{\varsigma}$. In a time interval of t we are thus likely to see $\frac{t\varsigma}{\phi\varepsilon(\chi,\delta)}$ extremes.

As discussed above, each major extreme has an associated probability of false positives of $2^{-\zeta\frac{a(a+1)}{2}}$, thus if we discover a consistent pattern of embedding in a time interval t, the probability of a false-positive becomes $P_{fp}(t) = (2^{-\zeta\frac{a(a+1)}{2}})^{\frac{t\varsigma}{\phi\varepsilon(\chi,\delta)}}$. For example if $\zeta = 1$, $a = 5$, $\varsigma = 100Hz$, $\phi = 20\%$, $\varepsilon(\chi, \delta) = 50$, after detecting a bit "true" for only $t = 2$ seconds we have $P_{fp}(2) = (2^{-15})^{20} \approx 0$ and an associated proof of rights, with a confidence of close to 100%. Even, at the limit, when due to transforms such as sampling and summarization, for each extreme, only one single m_{ij} average survives and the probability of false positives for each extreme becomes only $\frac{1}{2}$, $P_{fp}(2)$ becomes roughly only "one in a million".

Thus, the method persuasiveness proves to quickly converge in time to a comfortable level. Space constraints do not allow for a more extensive quantitative analysis. In Section 6 we provide experimental results for watermark resilience to various transforms, including random attacks.

5.4 On-the-fly Quality Assessment

In any watermarking framework, it is important to preserve structural and semantic properties of the watermarked data. Because by its very nature, watermarking alters its input, one has to provide a mechanism ensuring that these alterations do not degrade the data beyond usability. Preserving data quality also requires the ability to express and enforce data constraints. Sometimes it is undesirable or even impossible to directly map higher level semantic constraints into low level (combined) change tolerances for individual data items. The practically infinite set of semantic constraints that can be desired of a given data set makes it such that a versatile "data goodness" (i.e.

semantically) assessment method is required. Sion et. al. in [20] introduced this concept in the relational data watermarking framework. We also propose to extend our marking algorithm with semantic data constraints awareness. Each data property that needs to be preserved is written as a constraint on the allowable change to the dataset, the watermarking process is then applied with these constraints as input and re-evaluates them continuously for each alteration. An "undo" log (quite like the "rollback" log in [20]) is kept to allow undo operations in case certain constraints are violated by the current watermarking step (see Figure 5).

The new challenges in this framework are related to the fact that now, due to storage limitations, any data quality preservation constraints can only be formulated in terms of the current available data window. Likely only few window slots can be used to store data aggregates, possibly including some history information to be used in the quality evaluation process but this will all come at the expense of being unable to store and process as much new incoming data. Space considerations prevent us to elaborate further.

5.5 Speed and Finite Window

The proposed watermarking solution is highly adaptive to both speed and space constraints. By far the most computationally intensive operation is the one-bit encoding operation which alters the characteristic subset data to conform to the bit encoding convention defined in Section 4.5. At the expense of embedding resilience, this operation can be sped up significantly by both pruning of the search space or, more importantly, deployment of a computation-reducing technique as described in Section 4.5. Depending on the actual stream rate, these speed-ups can be gradually deployed to be able to keep up with the incoming data. Alternately, in Section 5.9 we present a (likely faster) encoding convention. Additionally, the average amount of computation to be performed per window-load of data is defined also by the actual fraction of extremes "selected" to be bit-carriers. This fraction is determined by $\frac{b(wm)}{\phi}$. If the incoming data rate is too high, ϕ can be increased to reduce the workload. In Section 5.5 we analyse this issue also from an experimental point of view.

With respect to space constraints, we believe the solution is an ideal fit for an on-the-fly finite-window processing model. The only requirements are: (i) to be able to detect at least one major extreme at a time for each window, (ii) to be able to fit its characteristic subset (or parts thereof) within the same window and (iii) to have enough remaining space to store some insignificant amounts of information such as the past $\varrho\lambda$ encountered major extremes. Even if the stream behavior is such that entire characteristic subsets (if it is just one of them, we can simply ignore it, we are concerned here with a majority of the extremes) cannot fit in the current window, the embedding gracefully handles subsets with fewer elements. Yet another solution would be to adjust δ so as to result in smaller characteristic subsets. Nevertheless, we believe this is (arguably) not a concern as most likely the window would contain severals extremes.

5.6 Offline Detection

As outlined above, the detection process is designed to function on-the-fly, in one pass over the data and compute the statistical bias for the embedded watermark bits. Time and storage space permitting, would a offline detection process possibly yield more accuracy? In other words, could there be any advantages to having more memory (e.g. $2 \times \varpi$) and unlimited amounts of time in the detection process? The answer is no. The only improvement that could be achieved would be in the normalization process. If the actual data distribution is not known, on-the-fly normalization (as discussed in Section 2.2) suffers from the need to perform an initial (non-detection) "discovery" run in which (hopefully) enough data is seen so as to construct a reasonable accurate reference data distribution. Some of the data read in this process would be lost for detection purposes due to storage space limitations. In the offline detection scenario, if multiple-pass access is assumed, this data can be used in detection, effectively enforcing the overall watermark.

5.7 Labeling Made Safer

The safety of the labeling process with respect to an attack in which Mallory purposefully alters previous extreme values adjacent to a considered extreme (in the hope of flipping one bit in the corresponding label), could be improved as follows. Instead of using ϱ as a sequential "step" factor in selecting some previous extremes to construct the current extreme's (η) label bits, we could use $H(msb(\eta, \alpha), k_1)$ as a bit-mask, to select a subset of the past extremes to define the label. For example out of the past 20 extremes we select 10 to be used in the 10-bit label computation, selection based on the last 20 bits of $H(msb(\eta, \alpha), k_1)$ (if a bit in the bit-mask is "true", the corresponding past extreme value is used in the label computation). This process yields both the benefits of shorter labels (more resilient overall, see Section 6) and forcing Mallory to consider all 20 bits (instead of 10) in his alteration attack, likely significantly more damaging to the data. For example, in Figure 2 (a), if the last 5 bits of $H(msb(val(K), \alpha), k_1) =$ "01101" then the 4-bit label of extreme K would be "1010".

Yet another resilience enhancing idea for labeling would be the use of multiple labels instead of just one, labels constructed using several different subsets of previously seen extremes. Then embedding/detection

proceed by enforcing the bit encoding convention considering both labels.

5.8 Summarization Revisited

Massive summarization is often used in scenarios involving storage and processing of streaming data. Summarization can be viewed as a normalized integration process. High summarization degrees (ν) are likely destroying much of the high frequency domain in the original stream. Often there exists a trade-off between preserving data of high-granularity in the recent past and of increasingly lower granularity in the distant past. The watermarking solution introduced here survives summarization very well up to high degrees. However, naturally, distant past data, if summarized to a higher degree would yield a more degraded version of the watermark than recent data. One solution to this issue would be to embed multiple layers of watermarks for different ν values, e.g. one layer for the low frequency domain (i.e. small ν values) and another layer for the high frequency domain (i.e. higher ν values). This would ensure an increasing accuracy on detection for both higher and lower degree summarizations.

5.9 Alternative Bit Encoding

An (arguably) fast(er) encoding than the use of cryptographic hashes in Section 4.5 could be adapted from [1]. The method works by altering the β least significant bits until every one of the longest k pre-fixes of the whole value (most significant bits included), when treated as an integer, becomes a quadratic residue modulo a secret large prime, for embedding a "true" value and a quadratic non-residue modulo the secret prime for embedding a "false" value.

6 Experimental Results

We implemented **wms.*** a Java proof-of-concept of the watermarking solution. Our experimental setup included one 1.8GHz CPU Linux box with Sun JDK 1.4 and 384MB RAM.

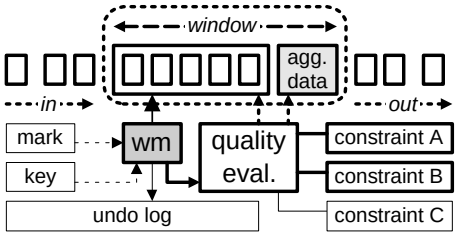

Figure 5: Overview of proof of concept implementation.

We implemented also a temperature sensor synthetic data stream generator with controllable parameters, including the ability to adjust the data stream distribution, fluctuating behavior (e.g. $\varepsilon(\chi, \delta)$) and rate ($\varsigma$). This sensor was used in the initial design phase of some of our experiments because of the ability to produce various fine-tuned data inputs impacting specific strengths of the encoding.

We explored extensive experiment scenarios, modeling both the behavior of sub-systems such as the on-the-fly labeling module as well as the overall watermark resilience. Synthetic (temperature sensor model) and real-world data was used in evaluating our method.

Because, as discussed in Section 3.3, watermark encoding relies on altering a certain secret statistical bias within the data, when we present resilience results we refer to the ability to detect and reconstruct this bias as an overall measure of encoding performance. In this case, the notion of a "watermark bias" refers to the number of instances of extremes for which statistically, the characteristic subset bit encoding convention (see Section 4.5), yields a positive true-bit embedding bias. For example, a detected watermark bias of 10 yields a false-positive probability of $\frac{1}{2^{10}}$, and an associated proof of rights with a confidence of roughly 99.9%, as discussed in Section 5.3.

Unless specified otherwise, the experimental results presented here refer to an underlying normalized stream with values distributed normally with a mean of 0 and a standard deviation of 0.5. The fluctuating behavior of the stream was determined by an average $\varepsilon(\chi, \delta) = 100$ (100 items per each major extreme) and $\varsigma = 100Hz$ (100 items per second). Other parameters include: $\phi = 3$, $\alpha = 16$, $\beta = 16$, $\upsilon = 2$, k_1 was chosen by a random number generator. Whenever exact quantitative results are shown, they refer to a data set drawn from about 50 seconds of stream data (i.e. roughly 5000 data values).

Additionally, when experiments were performed on real-life test data this is specified in the figure captions. The real life data sets [15] were obtained from the environmental monitors of the NASA Infrared Telescope on the summit of Mauna Kea, at the University of Hawaii. They represent multiple sets of once-every-two-minutes environmental sensor (i.e. temperature) readings at various telescope site locations. The reference data set used is referring to 30 days worth of data from the month of September 2003, totalling a number of 21630 temperature readings (with values on the Celsius scale roughly between 0 and 35 degrees).

Some of the figures presented in this Section feature a "spikey" behavior. This is a result of the adaptive data-dependent nature of the encoding. Different input data sets react differently to sampling for example, yielding slightly varying behavior at distinct points. Averaging over multiple inputs would provide a solution for this issue. Nevertheless, we believe that, while it might soften the spikes it would also (arguably) tone down distinct features for a given data set, features

that inter-relate figures. Instead of focusing on local variations, the figures should be interpreted as qualitative samples of global governing trends.

6.1 Random Alterations

In [20] Sion et al defined the *epsilon-attack* in the relational data framework, a transformation that modifies a percentage τ of the input data values within certain bounds defined by two variables ϵ (amplitude of alteration) and μ (mean of alteration). Epsilon-attacks can model any uninformed, random alteration – often the only available attack alternative. A *uniform altering* epsilon-attack (as defined in [20]) modifies τ percent of the input tuples by multiplication with a uniformly distributed value in the $(1 - \epsilon + \mu, 1 + \epsilon + \mu)$ interval. We believe this attack closely resembles (A6), a very likely combination of (A5) and (A2). In Figures 6 and 7 ($\mu = 0$) we analyze the sensitivity of both our labeling module and overall watermarking scheme to such randomly occurring changes, as direct measures for encoding resilience. In Figure 6 (a), label alteration increases with an increasing degree of data change. Smaller label bit sizes seem to better survive such an attack. In Figure 6 (b), as the percentage of altered data items increases, the labeling scheme naturally degrades.

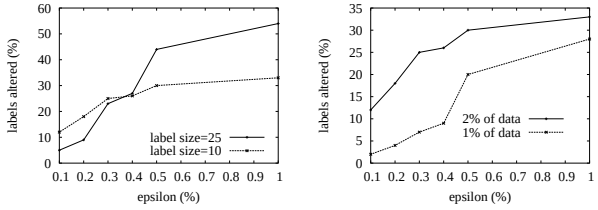

Figure 6: Label alteration for increasingly aggressive uniform altering epsilon attacks. (a) Different label bit sizes shown. A smaller label size seems to survive better. (b) Different altered data percentages shown. Naturally, altering a larger amount of data results in increasing label alterations.

In Figure 7, an embedded watermark (bias) is detected in a randomly altered stream. Naturally, an increasing distortion results in a decreasing bias detection. Nevertheless, it is to be noted that the encoding scheme proves to be quite resilient by design, for example for $\tau = 50\%$ of the data altered within $\epsilon = 10\%$ (Figure 7 (b)), the detected bias is still above 25, yielding a false-positive rate of less than "one in thirty million".

6.2 Sampling and Summarization

The ability to survive summarization (A1) and sampling (A2) is of extreme importance as both are expected common domain-specific transformations occurring for example in the process of data storage.

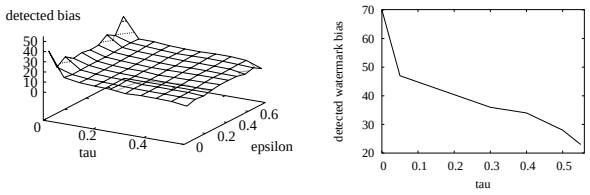

Figure 7: Watermark survival to epsilon-attacks. (a) Naturally, increasing τ and ϵ values result in a decreasing watermark bias. (b) Same phenomena shown for $\epsilon = 10\%$. (real data)

In Figure 8 the labeling algorithm is evaluated with respect to (a) sampling and (b) summarization. Intuitively, a higher label bit-size results in an increased fragility to sampling. Summarization seems to be naturally survived by our design. For example, a summarization of the data down to 5% ($\nu = 20$) still preserves over 20% of the original label values, thus conferring a strong back-bone to the watermark embedding process.

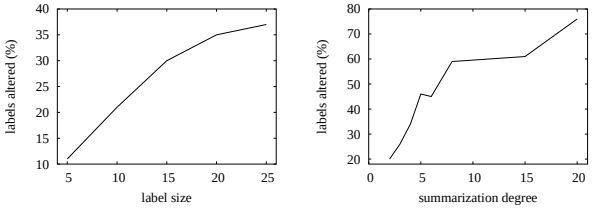

Figure 8: (a) Label resilience under sampling conditions. A higher label bit-size naturally yields an increased fragility to sampling. (b) Label alteration for summarization of increasing degree.

The behavior of the watermark encoding algorithm to sampling and summarization is outlined in Figure 9. Both transformations are survived extremely well, likely due to the design of the characteristic subset bit encoding which handles them naturally.

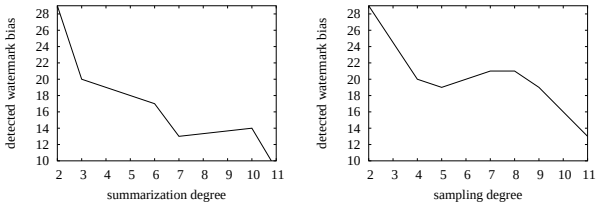

Figure 9: (a) Watermark survival to summarization. An increasing summarization degree results in a decreasing detected watermark bias. (b) Watermark survival to sampling. An increasing sampling degree results in less watermark bias, still enough to convince in court (e.g. a bias of 10 ensures a true-positive probability of 99.999%)

6.3 Segmentation. Combinations

In Section 5.2 we theoretically assessed the ability of our scheme to survive segmentation (A3), by answering the question: what is the minimum size of a stream segment from which we are able to recover the watermark? In Figure 10 (a) we analyze the impact of actual recovered segment size on the detected watermark bias. From a segment of only 2000 stream values we can detect a watermark bias of 10, corresponding to a very convincing low false positive rate of roughly 0.001.

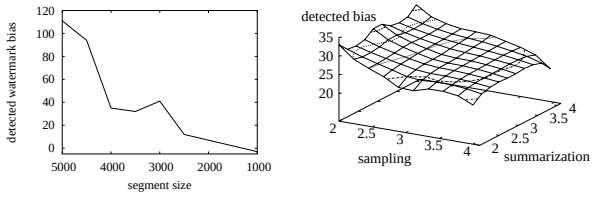

Figure 10: (a) Watermark survival to segmentation. With decreasing segment size, the detected watermark bias is degraded. Nevertheless, for example, a bias of 50 ensures a virtually zero false positive rate (2^{-50}), thus being entirely convincing in court. (b) Watermark survival to combined sampling and summarization (real data).

In Figure 10 (b) we outline the impact of a *combined* transformation (sampling and summarization) on the watermark embedding. Because of the nature of both transformations and of the resilience featured in each case, the combination seems to be survived equally well. For example, a 25% sampling, followed by a 25% summarization process still yields a watermark bias of up to 20, corresponding to a low false-positive rate of "one in a million".

6.4 Overhead and Impact on Data Quality

As mentioned in Section 5.5 our solution is naturally designed for stream processing. It is of importance to assess this ability also in practice. We performed experiments aimed at evaluating the introduced watermarking computation overhead. Unless specified otherwise, we used the multi-hash encoding discussed in Section 4.5 and parameters set such that the resulting watermark survives 100% any combined sampling and summarization up to a degree of 6.

First, we compared the computing times required by the watermarking process with the times spent in a simple read and copy model in which each stream item is read and copied to an output port (with fixed writing time-cost). We obtained consistent value classes clearly identifying each of the separate encoding methods presented. It became clear that, as expected, the majority of time is spent in the actual bit encoding convention routine (and not as much in the labeling module). Not surprising, the encoding convention introduced in Section 3.2 performed fastest with an average of only 5.7% increase in processing times per stream item. The poorest performer was the more complex multi-hash routine in Section 4.5 with an average increase of over 1000%, as expected decreasing almost perfectly exponential with the decrease of the guaranteed resilience (see Figure 11 (a)).

There are two lessons to be learned here. First, different encodings should be used for different scenarios with associated value models. For example for a temperature stream with a likely average reading rate of under 1Hz, deploying the multi-hash encoding routine for high resilience would be best suited whereas in a very fast streaming scenario the encoding in Section 3.2 would perform much better. Additionally, subject to future research is the issue of better pruning algorithms as discussed in Section 4.5.

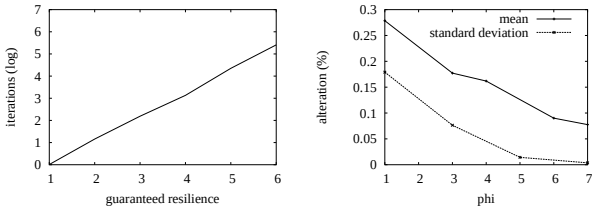

Figure 11: (a) Computation overhead (iterations) in multi-hash encoding increases with increasing guaranteed resilience (e.g. sampling degree) levels (logarithmic scale). (b) Decreasing the number of considered bit-encoding extremes (increasing ϕ) decreases the impact on mean and standard deviation in the watermarked data.

We also performed experiments evaluating the impact of our encoding on data quality. More specifically we analysed the alterations incurred by the mean and standard deviation of the stream data. For the above parameter settings, over a large number (12000+) of runs over the real (and synthetic) data sets, the value of the mean of the watermarked stream varied less than a mere 0.21% average from the original. The alteration to the standard deviation also maintained itself nicely within 0.27% of the original data. There exists a certain tunable trade-off between attack/transformation resilience and the incurred alterations. A lower level of resilience would definitely yield less required modifications to the data and an associated lower impact in the global statistics. In Figure 11 (b) we show how decreasing the number of considered bit-encoding major extremes decreases the impact on the average and standard deviation in the resulting stream.

Due to the random nature (with respect to the stream data values) of the encoding specifics we expected a virtually zero impact on such statistics over the longer term. While we observed a certain convergence to zero, it had not as fast a pace as expected; we were actually not able to actually reach the zero-

impact point. We suspect this is due to a bias introduced by the MD5 hash implementation used in our proof of concept, although the complex nature of the multi-hash embedding used (see Section 4.5) might also hold some of the answers. We are further investigating this. Space constraints do not allow for more details.

7 Conclusions.

In the present paper we introduced the issue of rights protection for sensor streams. We proposed a watermarking solution, based on novel ideas such as on-the-fly labeling and watermark encoding, resilient to important domain-specific transforms. We implemented a proof of concept of the proposed solution and evaluated it experimentally on real data. The method proves to be extremely resilient to all considered transforms, including sampling, summarization, random alterations and combined transforms. In upcoming research we propose to analyze streams of categorical data, to investigate other aggregates (instead of averages) in the summarization process (e.g. min, max, most likely value) and to experiment with alternative resilient and fast(er) bit-encodings.

References

[1] M. J. Atallah and Jr. S. S. Wagstaff. Watermarking with quadratic residues. In *Proc. of IS-T/SPIE Conf. on Security and Watermarking of Multimedia Contents, SPIE Vol. 3657, pp. 283–288.*, 1999.

[2] M.J. Atallah, V. Raskin, C. F. Hempelmann, M. Karahan, R. Sion, K. E. Triezenberg, and U. Topkara. Natural language watermarking and tamperproofing. In *Lecture Notes in Computer Science, Proc. 5th International Information Hiding Workshop 2002*. Springer Verlag, 2002.

[3] B. Babcock, S. Babu, M. Datar, and Motwani R. Models and issues in data stream systems. In *Proc. ACM Symp. on Principles of Database Systems (PODS)*, page 1.

[4] D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, N. Stonebraker, M.and Tatbul, and S. Zdonik. Monitoring streams – a new class of data management applications. In *Proceedings of the Int. Conf. on Very Large Data Bases (VLDB)*, 2002.

[5] S. Chandrasekaran and M. J. Franklin. Streaming queries over streaming data. In *Proceedings of the Int. Conf. on Very Large Data Bases (VLDB)*, pages 203–214, 2002.

[6] B. Chen and G. W. Wornell. Quantization index modulation: A class of provably good methods for digital watermarking and information embedding. *IEEE Transactions on Information Theory*, 47(4), 2001.

[7] Christian Collberg and Clark Thomborson. On the limits of software watermarking, August 1998.

[8] I. Cox, J. Bloom, and M. Miller. Digital watermarking. In *Digital Watermarking*. Morgan Kaufmann, 2001.

[9] M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining stream statistics over sliding windows. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 635–644, 2002.

[10] J. Kang, J. F. Naughton, and S. D. Viglas. Evaluating window joins over unbounded streams. In *Proceedings of ICDE*, 2003.

[11] S. Katzenbeisser and F. Petitcolas (editors). Information hiding techniques for steganography and digital watermarking. Artech House, 2001.

[12] J. Kiernan and R. Agrawal. Watermarking relational databases. In *Proceedings of the 28th International Conference on Very Large Databases VLDB*, 2002.

[13] D. Kirovski and H.S. Malvar. Spread-spectrum watermarking of audio signals. *IEEE Transactions on Signal Processing*, 51(4), 2003.

[14] F. Korn, S. Muthukrishnan, and D. Srivastava. Reverse nearest neighbor aggregates over streams. In *Proceedings of the Int. Conf. on Very Large Data Bases (VLDB)*, 2002.

[15] NASA. NASA infrared telescope facility (http://irtfweb.ifa.hawaii.edu/).

[16] J. Palsberg, S. Krishnaswamy, M. Kwon, D. Ma, Q. Shao, and Y. Zhang. Experience with software watermarking. In *Proceedings of ACSAC, 16th Annual Computer Security Applications Conference*, pages 308–316, 2000.

[17] F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn. Attacks on copyright marking systems. In David Aucsmith, editor, *Information Hiding: Second International Workshop*, volume 1525 of *Lecture Notes in Computer Science*, pages 218–238, Portland, 1998. Springer-Verlag.

[18] Bruce Schneier. Applied cryptography: Protocols, algorithms and source code in c. In *Applied Cryptography*. John Wiley and Sons, 1996.

[19] Radu Sion. Proving ownership over categorical data. In *Proceedings of the IEEE International Conference on Data Engineering ICDE*, 2004.

[20] Radu Sion, Mikhail Atallah, and Sunil Prabhakar. Rights protection for relational data. In *Proceedings of ACM SIGMOD*, 2003.

[21] M. D. Swanson, B. Zhu, and A. H. Tewfik. Audio watermarking and data embedding – current state of the art, challenges and future directions. In J. Dittmann, P. Wohlmacher, P. Horster, and R. Steinmetz, editors, *Multimedia and Security Workshop at ACM Multimedia*, volume 41 of *GMD*, Bristol, United Kingdom, September 1998. ACM.