Rights Protection for Discrete Numeric Streams

Radu Sion, *Member*, *IEEE*, Mikhail Atallah, *Fellow*, *IEEE*, and Sunil Prabhakar, *Senior Member*, *IEEE*

Abstract—Today's world of increasingly dynamic 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. Often, streaming information is offered on the basis of a nonexclusive, single-use customer license. One major concern, especially given the digital nature of the valuable stream, is the ability to easily record and potentially "replay" parts of it in the future. If there is value associated with such future replays, it could constitute enough incentive for a malicious customer (Mallory) to record and duplicate data segments, subsequently reselling them for profit. Being able to protect against such infringements becomes a necessity. In this work, we introduce the issue of rights protection for discrete streaming data through watermarking. This is a novel problem with many associated challenges including: operating in a finite window, single-pass, (possibly) high-speed streaming model, and surviving natural domain specific transforms and attacks (e.g., 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.*) and perform experiments on real sensor data from the NASA Infrared Telescope Facility at the University of Hawaii, to assess encoding resilience levels in practice. Our solution proves to be well suited for this new domain. For example, we can recover an over 97 percent confidence watermark from a highly down-sampled (e.g., less than 8 percent) stream or survive stream summarization (e.g., 20 percent) and random alteration attacks with very high confidence levels, often above 99 percent.

Index Terms—Rights protection, discrete streams, sensor networks, watermarking.

1 Introduction

Digital Watermarking as a method of Rights Assessment deploys Information Hiding to conceal an indelible "rights witness" ("rights signature," watermark) within the digital Work to be protected. The soundness of such a method relies on the assumption that altering the Work in the process of hiding the mark does not destroy the value of the Work, and that it is difficult for a malicious adversary ("Mallory") to remove or alter the mark beyond detection without destroying the value of the Work. The ability to resist attacks from such an adversary (mostly aiming at removing the embedded watermark) is one of the major concerns in the design of a sound watermarking solution.

In this paper, we introduce and study the problem of watermarking discrete (sensor) streams data, which, to the best of our knowledge, has not been addressed. Streaming data sources represent an important class of emerging applications [4], [6]. These applications produce a virtually endless stream of data that is too large to be stored directly. Examples include output from environmental sensors such as temperature, pressure, brightness readings, stock prices, etc. Recent efforts in the broader area of streaming data deal with the database challenges of its management [7], [9], [12], [15].

Our work on discrete/itemized data types (e.g., [21], [22], [23]) and related efforts by Agrawal [1], Kiernan and Agrawal [14], and Li et. al. [16] all rely upon the availability of the entire data set during the watermarking process. While this is generally a reasonable assumption, it does not hold true for the case of streaming data [4]; 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. Additionally, the attack and transformation models in existing research does not apply here. For example, a process of summarization would defeat any of the above schemes. Yet, another difference from previous research is the lack of a "primary key" reference data set (an essential, required, part in both [23] and [14]). Due to these differences, earlier work on watermarking relational data sets is not applicable to streams.

Let us understand now why watermarking streaming data important. After all, could we not simply watermark the data once it is stored? This surely would work and enable rights protection for the stored result. But, it would not deter a malicious customer (Mallory), with direct stream access, to duplicate segments of the stream and resell them or simply restream the data for profit. The main rights protection scenario here is to prevent exactly such leaks from a licensed customer.

Our contributions include

1. the proposal and definition of the problem of watermarking sensor streams,
2. the discovery and analysis of new watermark embedding channels for such data,
3. the design of novel associated encoding algorithms,
4. a proof of concept implementation of the algorithms, and
5. their experimental evaluation.

- *R. Sion is with the Department of Computer Sciences, Stony Brook University, Stony Brook, NY 11794. E-mail: sion@cs.stonybrook.edu.*
- *M. Atallah and S. Prabhakar are with the Department of Computer Sciences, Purdue University, 250 N. University Street, West Lafayette, IN 47906. E-mail: {mja, sunil}@cs.purdue.edu.*

Manuscript received 27 June 2005; revised 20 Oct. 2005; accepted 1 Nov. 2005; published online 17 Mar. 2006.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0248-0605.

The algorithms introduced here prove to be resilient to important domain-specific classes of attacks, including stream resampling, summarization (replacing a stream portion by its average value) and random changes. For example, sampling the data stream down to less than 8 percent still yields a court-time confidence of watermark embedding of over 97 percent. Summarization (e.g., 20 percent) and random data alterations are also survived very well, often with a false-positive detection probability of under 1 percent.

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 the ability to convince in court to survive attacks and natural domain transformations. 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 The Adversary

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, see below) is available for processing makes certain history-dependent computations difficult or simply impossible. At the same time, metrics of quality can only be handled within this space; any 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. In summary, the nature of this new domain is such that only a limited amount of time is available to be spent in processing each incoming data item and only a limited number of such items can be considered at a time (limited window).

Moreover, the effectiveness of any rights protection method is directly related to its ability to deal with 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[1] (there might be value in actual *data trends*, that Mallory 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), 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 known original distribution. Also, it can be seen that (A6) is naturally modeled by a combination of (A2) and (A5).

Apparently, data resorting might be also of concern as an attack. We claim, however, that in a significant number of scenarios, 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 many applications, a recorded stream (even sampled) is likely valuable if its replay preserves the relative ordering of the values (with exception of some extreme cases). Reordering 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 resorting, is simply not possible given the finite processing window and speed assumptions. Here, we consider data resorting 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 resorting gracefully.

2.2 Model

For the purpose of simplicity, let us define a 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 and stock market data). $x[t]$ is a notation for the value yielded by our source(s) at time t. Unless specified otherwise, let us denote a stream as $(x[], \varsigma)$, where ς is the number of incoming data values per time unit (*data rate*).[2]

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

1. Taken care of by the initial normalization step.

2. The proposed solution does not rely on any characteristic of the actual stream data rate. For space and simplicity purposes, in this paper, we are discussing streams with fixed data rates.

resulting stream (which is ultimately just a sequence of values).

Data Model. One assumption underlying this solution is that the the stream data is assumed to tolerate a small number of alterations to its values' least significant bits without significant degradation of its overall worth. On the other hand, any large number of alterations is assumed to render the stream data unusable.

Processing Model. Any stream processing is necessarily both time and space bound. The time bounds derive from the fact that it 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.

Definitions. 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 $x'[t]$ (considered for the purpose of this definition as uniquely defined by its value *and* its timestamp), there exists a contiguous subset of $(x[])$, $(x[t_1], x[t_2])$ such that

$$x'[t] \in \{x[i] : t_1 \leq i \leq t_2\}$$

and

$$\{x'[t-1], x'[t+1]\} \subsetneq \{x[i] : t_1 \leq i \leq t_2\}$$

and t is uniformly distributed in (t_1, t_2). In other words, it is constructed by randomly choosing one value out of every χ values in the original. A subtle variation of *uniform random sampling* is the case when $x'[t]$ is not randomly chosen but rather always the first element in its 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, nonoverlapping ν-sized subsets of $(x[])$, $(x[t-\nu+1], x[t-\nu+2], \ldots, x[t])$, $(x[t+1], x[t+2], \ldots, x[t+\nu])$ such that

$$x'_1[t] = \frac{\sum_{i \in (1,\nu)} x[t-\nu+i]}{\nu}$$

and

$$x'_2[t+\nu] = \frac{\sum_{i \in (1,\nu)} x[t+i]}{\nu}.$$

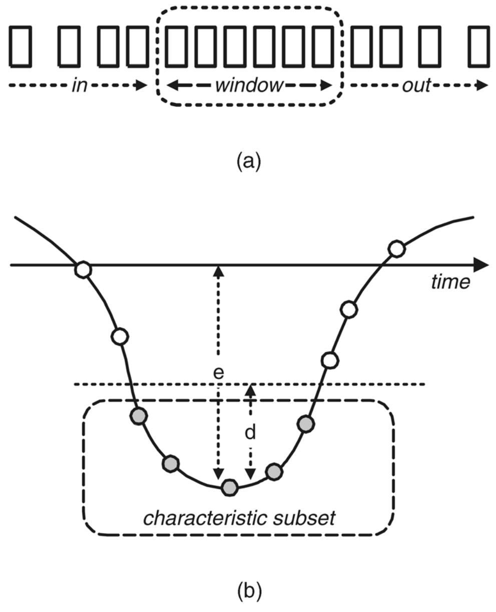


Fig. 1. (a) Processing is necessarily bound in both time (stream rate) and space (window). (b) δ-radius characteristic subset of η.

In other words, for a continuous chunk of ν elements from the original stream summarization outputs its average. Various other similar aggregates could be envisioned here (e.g., min/max, most likely value). We believe that in the current scope, considering averaging summarization is both illustrative and qualitatively identical while not complicating the analysis too much.

We define an *extreme* η in a stream simply as either a local minimum or local maximum value. We define the extreme's *characteristic subset of radius* δ, noted $\Xi(\eta, \delta)$ (see Fig. 1b), 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 Fig. 5, intuitively, it seems likely that extremes such as F, I, and J have a smaller chance of surviving sampling than C, E, or G. This is so because of the temporal shape of the stream's evolution. C, E, and G seem to yield characteristic subsets much "fatter" than F, I, and J. Intuitively, δ needs to be chosen such that the characteristic subsets are of an average size greater than χ (to handle a sampling of degree χ).

To model the "fluctuating" nature of a stream, let $\varepsilon(\chi, \delta)$ be 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.

Notations. For any value (e.g., numeric) 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,

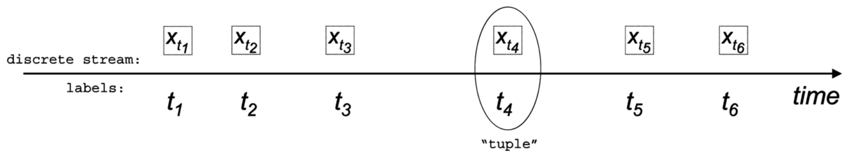

discrete stream:
labels:
x_{t_1} x_{t_2} x_{t_3} x_{t_4} x_{t_5} x_{t_6}
t_1 t_2 t_3 t_4 t_5 t_6
time
"tuple"

Fig. 2. If trusted time-stamp information would be available, we could treat the time-stamp information as a primary key and each (timestamp, value) pair as a relational tuple. Existing methods could then be applied to a limited extent, surviving (only) sampling transformations.

$lsb(x, b)$ is used to denote the least significant b bits of x. If by wm we denote a watermark to be embedded, $wm[i]$ will then be the ith bit of wm. Let $set\_bit(d, a, b)$ be a function that returns value d with the bit position a set to the truth-value of b. In any following mathematical expression, let the symbol "&" be the *bit-AND* operation, let "|" be the *bit-OR* operation. Let "$(a \ll b)$" be a notation for $(a \times 2^b)$ by definition (left bit shift). Let "$(a \gg b)$" be a notation for $\lfloor \frac{a}{2^b} \rfloor$. Let $x; y = ((x \ll b(y))|y)$ be the concatenation of the two bitstrings x and y.

2.3 Crypto-Hashes

A special defacto secure construct we are leveraging in our solution, is the *one-way cryptographic hash*. If $crypto\_hash()$ is a cryptographic secure one-way hash, of interest are two of its properties: 1) It is computationally infeasible, for a given value V' to find a V such that $crypto\_hash(V) = V'$ (one-wayness), and 2) 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 hashes. For more details on cryptographic hashes, consult [19]. Let $H(V, k) = crypto\_hash(k; crypto\_hash(V; k))$ (where ";" denotes concatenation).

2.4 Related Work

Let us understand here why existing results in nonmedia data sets watermarking such as relational data [1], [14], [21], [22], [23], [16] cannot be adapted for discrete streams. These solutions require access to the entire data set in an almost random access model, which is certainly not possible here at embedding time. Also, these efforts seem to make extensive use of the existence of a primary key (or an additional attribute, e.g., in [21]), thus rendering a direct adaptation impossible. Moreover, the expected attacks and transformations are different. For example, a process of summarization would defeat any of the above schemes. 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 both the bit alteration method proposed in [1] (for numeric types), its extension in [16] and our solution in [21] (for discrete data) could be all adapted to work on a single attribute, namely, the stream value. The result would likely be resilient to (time-stamp preserving) sampling, but fail with respect to any other attack or transformation (see Fig. 2).

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 [2], [5], [8], [10], [11], [13], [17], [20] since audio data is also an example of a data stream? In other words, why is our problem different? While there seems to be similarities between watermarking audio and sensor data, for example, at a closer inspection these similarities prove to be merely superficial. A multitude of differences are to be found between the two frameworks mainly deriving from different data models, associated semantic scopes and the itemized nature of sensor stream data.

In theory, a sensor stream could be viewed as an audio signal, for example, and processed as such. However, 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. Summarization, for example, would not be survived by any of the existing results. Moreover, 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. 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]

In summary, while experiences in the multimedia domain are valuable, due to the nature of this new application domain, a solution for watermarking discrete sensor streams needs to naturally handle attacks and transformations such as the ones outlined in Section 2.1.

3 An Initial Solution

This section outlines the main solution and then gradually improves it to a more robust and resilient version, by identifying and fixing potential flaws.

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 specifically, we identify two scenarios apparently featuring distinct challenges: 1) an on-the-fly streaming detection process *and* 2) 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. Intuitively a watermarking solution for 2) 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

3. For example, the total alteration introduced per data item should not exceed a certain threshold.

```
embed_bit($\Xi$, $loc$, $val$)
  foreach $v \in \Xi$ do
    $v[loc-1] \leftarrow false$
    $v[loc] \leftarrow val$
    $v[loc+1] \leftarrow false$

embed($\delta$,$\alpha$,$\beta$,$wm$,$k_1$,$\phi$)
  while (true) do
    $\eta \leftarrow$ next major extreme in win[]
    compute $\Xi(\eta, \delta)$
    $i \leftarrow H(msb(\eta, \alpha), k_1) \bmod \phi$
    if $i \leq b(wm)$ then
      $bit \leftarrow H(msb(\eta, \alpha), k_1) \bmod \beta$
      embed_bit($\Xi(\eta, \delta), bit, wm[i]$)
    advance win[] past $\eta$

detect($\delta$,$\alpha$,$\beta$,$wm$,$k_1$,$\phi$)
  while (true) do
    $\eta \leftarrow$ next extreme in win[]
    $i \leftarrow H(msb(\eta, \alpha), k_1) \bmod \phi$
    if $i \leq b(wm)$ then
      $bit \leftarrow H(msb(\eta, \alpha), k_1) \bmod \beta$
      if ($\eta[bit]$==true) then
        $wm[i]^T \leftarrow wm[i]^T + 1$
      else
        $wm[i]^F \leftarrow wm[i]^F + 1$
    advance win[] past $\eta$

construct($wm[]^T$,$wm[]^F$,$\upsilon$)
  for ($i \leftarrow 0$;$i < b(wm)$;$i \leftarrow i+1$)
    if ($wm[i]^T - wm[i]^F > \upsilon$) then
      $wm[i] \leftarrow true$
    else
    if ($wm[i]^F - wm[i]^T > \upsilon$) then
      $wm[i] \leftarrow false$
    else
      $wm[i] \leftarrow undefined$
  return $wm[]$
```

Fig. 3. Initial algorithm.

can be directly applied to 2), for the time being let us consider a solution for 1). In Section 4.4, we analyze the offline case.

At an overview level, watermark embedding proceeds as follows:

1. 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.
2. A certain criterion is used to select some of these extremes as recipients for parts of the watermark. And,
3. 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 unaltered data). In the watermark detection process 4), *all* the extremes in the stream are identified and the selection criteria in step 2) above is used once again to identify potential watermark recipients. For each selected extreme, 5) its corresponding 1-bit watermark is extracted and ultimately the global watermark is gradually reconstructed, by possibly also using error correction.

Thus, one of the main insights 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, likely to survive value-preserving, domain-specific transforms.

3.2 Embedding

In this and the next section, we will introduce an initial encoding and corresponding detection algorithm. As shown in subsequent sections, however, these algorithms are somewhat flawed. We will then gradually fix them. Using the notation in Section 2.2, let $\alpha, \beta \in \mathbb{N}$ such that $\alpha + \beta \leq b(x[])$, where $b(x[])$ is the length of the representation of the values in the considered stream $(x[])$. Let χ be a secret integer and $\delta \in (0, 1)$ chosen such that all elements within a characteristic subset $\Xi(\eta, \delta)$ have the same most significant α bits. $\alpha, \beta, \delta, \chi$ are secret. We use the term "advance the window" to denote reading in more new data items while discarding old ones.

In the **initial step** of our embedding algorithm (see Fig. 3), 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. Whether an extreme is "major" (Section 2.2) 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 percent) of survival in case of sampling (i.e., $\frac{subset\_size}{\chi} >$ 70 percent).

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 to determine whether η is going to be used in the embedding process. If $H(msb(\eta, \alpha), k_1) \; mod \; \phi = i$ 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. A related "fitness" selection criteria was used by us and others in [14], [23]. Its power derives 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

$$msb(x, \alpha) = msb(\eta, \alpha), \quad \forall x \in \Xi(\eta, \delta).$$

If η is the result of the previous selection step, in the **third step**, we embed bit $wm[i]$ into $\Xi(\eta)$. This is done by first, selecting a certain bit position $bit = H(msb(\eta, \alpha), k_1)\ mod\ \beta$ 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 restarts.

3.3 Detection

We are going to illustrate a specific flavor of the detection process, namely, the case when majority voting is deployed as an error correction mechanism (see Fig. 3).

In the detection process, the watermark is gradually reconstructed as more and more of the stream data is processed. The reconstruction process relies on an array of majority voting "buckets" as follows: 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. In the end, 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 $wm^{est}[i] = true$[4] (and, conversely, if $wm[i]^F - wm[i]^T > \upsilon$, then $wm^{est}[i] = false$) where $\upsilon > 0$. If detection would be applied on random, unwatermarked data, the probability of detecting $wm^{det}[i] = false$ would equal[5] 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 undefined, thus the data considered unwatermarked. The watermark effectively lies in a statistical bias in the $true/false$ distribution for each bit encoding.

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 $H(msb(\eta, \alpha), k_1)\ mod\ \phi = i$ 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, the window is advanced past η and the process restarts. It is to be noted that, because of the infinite nature of the stream, detection is a continuous process. This is why it is enclosed in a while loop. At the same time, it shares the $wm[]$ array with the watermark reconstruction process (**construct()**).

4. To make this completely accurate, we need to consider the associated random walk probability. Thus, in evaluating the formula, υ needs to be adjusted with roughly $\sqrt{\frac{2r}{\pi}}$, where $r = wm[i]^T + wm[i]^F$ is the number of recovered bits so far.

5. Because the embedding locations are secret and randomly distributed throughout the data, and because the data is assumed to be numeric (thus, any bias will likely be also numeric—and not necessarily consistent in the bit-representation domain), it is highly unlikely that a corresponding data bias will exactly "follow" these locations so as to impact the detected watermark. In other words, we assume it is unlikely, *for example*, for temperature values to have a certain bit set to "1" more often than "0" (bias in bit-domain). If this is the case, however, possibly an initial bias detector can be deployed in conjunction with the watermarking module.

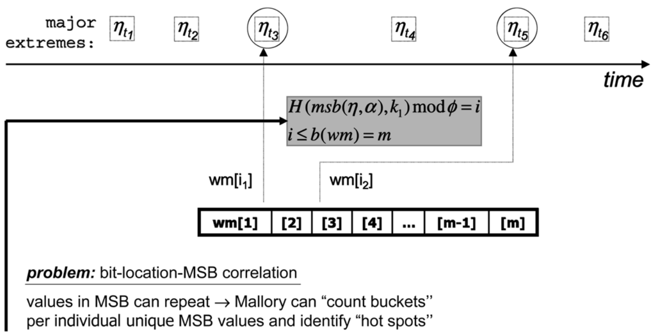

major extremes:
η_{t_1}
η_{t_2}
η_{t_3}
η_{t_4}
η_{t_5}
η_{t_6}
time
$H(msb(\eta,\alpha),k_1) \bmod \phi = i$
$i \leq b(wm) = m$
wm[i1]
wm[i2]
wm[1] [2] [3] [4] ... [m-1] [m]
problem: bit-location-MSB correlation
values in MSB can repeat → Mallory can "count buckets" per individual unique MSB values and identify "hot spots"

Fig. 4. A special attack is possible, exploiting the correlation between the watermarking alteration (the $wm[i]$ bit) and its actual location (determined by $H(msb(\eta, \alpha), k_1)))$. One solution would be to use an alternate source of information to determine the bit location (i.e., "labels," see below).

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, unwatermarked 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

Various resilience enhancing improvements are possible with respect to the initial solution introduced above. These are discussed here.

4.1 Defeating Correlation Detection

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 (see Fig. 4). 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. Mallory can now 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 (respectively, reset) more often. For each η for which a bias in a bit position is discovered, that particular bit position is considered mark-carrying and randomized.

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 could possibly rely on a separate source of information to determine 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 their values. This unique label could then be used in computing the bit position for embedding. In the selection of the bit embedding location, instead of using $bit = H(msb(\eta, \alpha), k_1)$ mod β which yields a result correlated to the actual embedded bit value ($wm[i]$, where $i = H(msb(\eta, \alpha), k_1)$ mod ϕ) we propose to use

$$bit = H(msb(label(\eta), \alpha), k_1) mod \beta,$$

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 envision a different labeling scheme (at least) for extremes, that would survive the attacks and transformations outlined in Section 2.1? 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 2.1), this task becomes especially challenging. Sampling and summarization are already survived (by design) by the extremes selected using the "majority" criteria in Section 2.2. We could maybe make use of this fact in the labeling scheme.

One of the challenging aspects of such a labeling scheme becomes clear when one considers data segmentation. To support segmentation, it 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 (secret) bit length of the labels resulting in our labeling scheme. Let $\varrho > 1$ be a (secret) unsigned integer. We propose the following labeling scheme: Given two extremes i and a subsequent $i + \varrho$, we define $label\_bit(i, i + \varrho) = true$ iff

$$msb(abs(val(i)), \alpha) < msb(abs(val(i + \varrho)), \alpha), \qquad (1)$$

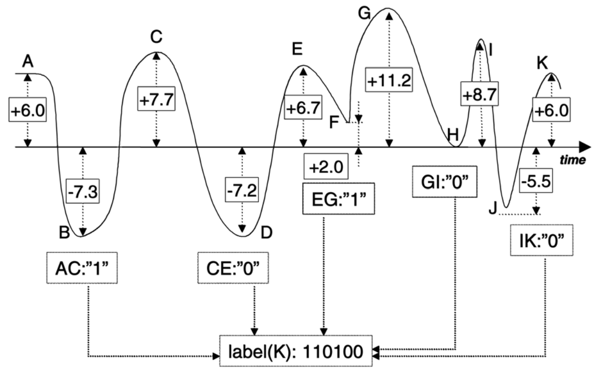


Fig. 5. Basic Label construction for extreme **K** by concatenation of individual label bits ($\varrho = 2$).

and $false$ otherwise. We define the label for extreme $i + \lambda$, $label(val(i + \lambda))$ as 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 $j \in \{i - \varrho, i, i + \varrho, \ldots, i - \varrho + \lambda\}$. In other words, an extreme is labeled by a certain differential interpretation of some of the preceding extreme values (see Fig. 5).

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 increasingly unsuccessful. 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 4.5.

Before going any further, let us analyze what happens if an important extreme is "lost," e.g., if one extreme i is altered so much that its α most significant bits flip inequality (1), corrupting its corresponding label bit. What happens is in fact not too damaging: Labels that were constructed using this particular extreme will be corrupted, until the detection process encounters again a continuous sequence of extremes not altered beyond recognition. But, Mallory cannot afford altering extremes to such extents, and the secrecy of ϱ 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, uncorrelated source of information for determining the bit position to be altered (see Fig. 6). Remember that our ability to survive "bucket counting" type of attacks was dependent on the labels being uncorrelated with respect to the actual extreme values, while at the same time being virtually unique for each extreme.

4.1.1 Repeating Labels

But, the finite nature of the considered bit size of the label poses a certain problem in this respect by necessarily allowing

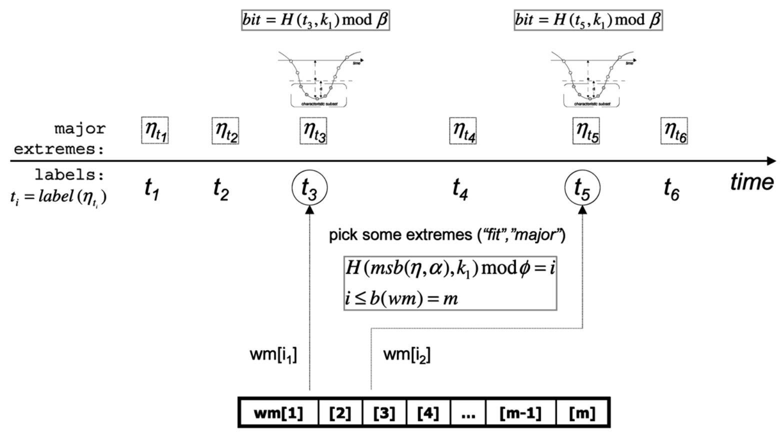

$bit = H(t_3, k_1) \bmod \beta$
$bit = H(t_5, k_1) \bmod \beta$
major extremes:
η_{t_1} η_{t_2} η_{t_3} η_{t_4} η_{t_5} η_{t_6}
labels: $t_i = label(\eta_{t_i})$
t_1 t_2 t_3 t_4 t_5 t_6
time
pick some extremes ("fit", "major")
$H(msb(\eta, \alpha), k_1) \bmod \phi = i$
$i \leq b(wm) = m$
wm[i1]
wm[i2]
wm[1] [2] [3] [4] ... [m-1] [m]

Fig. 6. Overview of the extended labeling-based algorithm. The watermark bits are "spread" throughout a set of "fit" extremes. For each extreme, the characteristic subset encoding bit location is determined by a one-way hash of the extreme's label.

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 2,000 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 versus 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: 1) 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. 2) Once the uncorrelated nature of the labels has been established by their independent information source, we can reconsider 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 $msb(abs(val(\eta)), \alpha)$, we significantly decrease the probability of duplicates.

4.1.2 Reconstructing Labels

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 and 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? 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 (possibly transformed) stream $(x'[], \varsigma')$ 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 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 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 of this method is used successfully.

4.1.3 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

```
label_bit($i, j$)
  if $((msb(abs(val(i))) - msb(abs(val(j)))) < \iota^-)$ then
    return true
  if $(\iota^+ < (msb(abs(val(i))) - msb(abs(val(j)))))$ then
    return false
  return undefined

label_extreme($\eta, prev\_ext[], \varrho, \lambda$)
  l ← true
  for $(i \leftarrow 0; i < \lambda; i \leftarrow i+2)$
    bit ← label_bit(prev_ext[$i$],prev_ext[$i+\varrho$])
    l ← (l;bit)
  return l

embed($\delta,\alpha,\beta,wm,k_1,\phi,\varrho,\lambda$)
  prev_ext[] ← {}
  while (true) do
    $\eta$ ← next major extreme in win[]
    prev_ext.append($\eta$)
    compute $\Xi(\eta,\delta)$
    $i \leftarrow H(msb(\eta,\alpha),k_1)\bmod\phi$
    if $i \leq b(wm)$ then
      $l \leftarrow$ label_extreme($\eta, prev\_ext[], \varrho, \lambda$)
      $l \leftarrow (l; msb(abs(val(\eta)),\alpha))$
      $bit \leftarrow H(msb(l,\alpha),k_1)\bmod\beta$
      embed_bit($\Xi(\eta,\delta), bit, wm[i]$)
    advance win[] past $\eta$
```

Fig. 7. Extended labeling-based algorithm.

$$(msb(abs(val(i))) - msb(abs(val(i+\varrho)))) < \iota^- < 0$$

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.

Given the improvements introduced in the previous sections, the extended embedding algorithm is illustrated in Fig. 7.

4.2 Defeating Bias Detection

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 naturally be considered watermark carrying and Mallory could mount a simple attack of randomizing those bit positions. This attack 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 $\Xi(\eta,\delta) = \{x_1, x_2, \ldots, x_a\}$. 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: 1) we say that a bit value of "true" is embedded in $\Xi(\eta,\delta)$ iff

$$\forall(j,i), lsb(H(lsb(m_{ij},\beta), label(\eta)), \zeta) = 2^\zeta - 1.$$

Similarly, 2) we say that "false" is embedded iff we have

$$\forall(j,i), lsb(H(lsb(m_{ij},\beta), label(\eta)), \zeta) = 0,$$

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 $wm[i]$ bit value. It is to be noted that these alterations should aim to minimize the Euclidean distance (or possibly any other distance metric) from the point defined by $\{x_1, x_2, \ldots, x_a\}$. We call this a "multihash 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? For each item in $\Xi(\eta,\delta)$, we consider its β least significant bits, thus we effectively operate over an input space of $a\beta$ bits. 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, effectively getting 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).

We validated these required computation times results experimentally with the aim to understand any potential deviations from the theoretical case introduced by less than ideal behavior of the deployed constructs. In Fig. 8, we illustrate an experiment in which an unoptimized exhaustive search was deployed. The exponential nature of the required amount of computation becomes clear by the linear behavior in the logarithmic graph. See also Section 6.4 for a related experimental analysis.

If enough computation power is available with respect to the incoming stream data rate, larger values for ζ and a

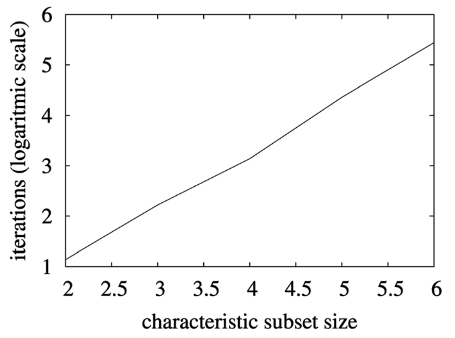


Fig. 8. Average exhaustive search iterations required in computing the closest point that satisfies the characteristic subset bit encoding convention (logarithmic scale).

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 scope here, 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 1) or 2) needs to hold in the subset bit encoding convention above. In other words, the search process (in the $\{x_1, x_2, \ldots, x_a\}$ space) will be stopped once a certain number of the m_{ij} averages feature the desired encoding convention (1) or 2)). 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 active m_{ij} values.

If such a reducing technique is applied, a desired 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 nonzero 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 nonzero probability of this average to appear in a ν-degree summarized stream) that allows the extraction of the associated watermark bit.

With these modifications, we can further extend the algorithm in Section 4.1 (Fig. 7). The modified bit-encoding function **embed_bit**(Ξ, bit, wm_bit_value) is illustrated in Fig. 9. Note the additional ζ parameter, to become part of the global embedding secret key.

Also, a likely fast(er) encoding than the use of cryptographic hashes above could be adapted from [3]. The method works by altering the β least significant bits until every one of the longest k prefixes of the whole value (most significant bits included), when treated as an integer,

```
**embed_bit**($\Xi = \{x_1, ..., x_a\}, loc, val, \zeta$)
  $m_{ij} \leftarrow \{\{\}\}$
  **forall** $(i,j)$ **compute** $m_{ij} \leftarrow \frac{\sum_{u \in [i,j]} x_u}{|j-i+1|}$
  **find** $\Xi' = \{x'_1, ..., x'_a\}$ **such that** $\forall (j,i)$
    $lsb(H(lsb(m_{ij}, \beta), label(\eta)), \zeta) = (2^{\zeta} - 1)$ for (val=*true*)
    **and**
    $lsb(H(lsb(m_{ij}, \beta), label(\eta)), \zeta) = 0$ for (val=*false*)
    **and**
    $\nexists\ \Xi''$ **with** euclidean_distance($\Xi, \Xi''$) < euclidean_distance($\Xi, \Xi'$)
```

Fig. 9. Modified partial-sums-based bit-encoding handling bias detection attacks.

becomes a quadratic residue modulo a secret large prime, for embedding a "true" value and a quadratic nonresidue modulo the secret prime for embedding a "false" value.

4.3 On-the-Fly Quality Assessment

In the process of altering the data for the purpose of information hiding, it is important to preserve its structural and semantic properties. 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. We propose to augment our sensor stream marking algorithm with such semantic constraints awareness. Each data property that needs to be preserved is written as a constraint on the allowable change to the data set, the watermarking process is then applied with these constraints as input and reevaluates them continuously for each alteration ("consumer driven encoding"). An "undo" log (quite like the "rollback" log in [23]) is kept to allow undo operations in case certain constraints are violated by the current watermarking step (see Fig. 10). 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.

4.4 Offline Detection

As outlined above, the detection process is designed to function on-the-fly, in one pass over the data and compute

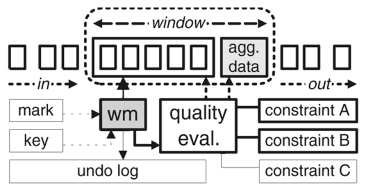


Fig. 10. Overview of proof of concept implementation.

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 2.1) suffers from the need to perform an initial (nondetection) "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.

4.5 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 Fig. 5, if we have the last 5 bits of $H(msb(val(K), \alpha), k_1)$ equal "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.

4.6 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 degrees.

5 Analysis

In this section, we analyze the ability of our method to convince in court, survive attacks, and transforms.

Court-convince-ability 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 4.2, 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 (per extreme)

$$2^{-\zeta\frac{a(a+1)}{2}}.$$

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 $\frac{\phi\varepsilon(\chi,\delta)}{\varsigma}$, thus the number of extremes that we are likely to see in a time interval of size t, is

$$\frac{t\varsigma}{\phi\varepsilon(\chi, \delta)}.$$

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 $\zeta = 1$, $a = 5$, $\varsigma = 100Hz$, $\phi = 20$ percent, $\varepsilon(\chi, \delta) = 50$, 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 200 percent. 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 persuasion power of our method quickly converges to a comfortable level. In Section 6, we provide experimental results for watermark resilience to various transforms, including random attacks.

Next, we explore a theoretical analysis of the vulnerability of our scheme under the following attack model: Mallory starts to modify randomly every a_1th $(a_1 > 1)$ 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 associated labeling scheme, in other words, they do not 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. To strengthen our derived bounds, we are going to focus directly on a more challenging, "informed," Mallory, aware of the characteristic subset radius used at encoding time. In other words, we assume that $a_3 = \delta$ is known to Mallory, see Section 3.2.

We propose two ways to analyze this vulnerability: 1) 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 2) what is the probability that *all* of the active ones are obliterated? It is easy to see that, for a given extreme η, for which $\Xi(\eta, a_3) = \{x_1, x_2, \ldots, x_a\}$, the number of corresponding m_{ij} values altered is

$$c_m = \frac{1}{2} a a_2 (2a - a a_2 + 1).$$

Now, for 1), the "weakening" of the encoding can be defined as

$$c_m \times \frac{2}{a(a+1)},$$

which is 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 (2), 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 black balls, what is the probability that the $x + t$ removals emptied the bowl of all of them? It can be shown that this is

$$P(x + t, x, y) = \frac{\binom{y-x}{t}}{\binom{y}{x+t}}. \tag{2}$$

In our model, $(x + t) = c_m$, $y = a(a+1)\frac{1}{2}$ and if $x = a_4 y$ (a_4 is the ratio of active m_{ij} values), we can compute the probability that *all* of them are altered.

Thus, for each attacked extreme we have a nonzero 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 4.2 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 multibit watermark after error correction? At the beginning of the section, we looked at how the watermark bias becomes more convincing in time (and seen data). Losing 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 $P(x + t, x, y)$ (2) we can perform a similar reasoning 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 $a_1 = 5$, $a = 6$, $a_4 = 50$ percent, $a_2 = 50$ percent, we get the average probability $P(15, 10, 21) \approx 0.85$ percent 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 $a_1 \times P(x + t, x, y) \approx 4.25$ percent more data to be equally convincing at detection.

But, how does our encoding handle transforms? By construction, it 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 nonzero probability of elements in $\Xi(\eta, \delta)$ to be in the sampled stream even for $\chi > \chi_{max}$. This is experimentally analyzed 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 4.2. 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.

But, how well is segmentation (A3) survived? 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 $\varepsilon(\chi, \delta)\lambda\varrho$.

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.

We also implemented 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 experiment scenarios modeling both the behavior of subsystems 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 our evaluation.

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 *active* extremes for which the characteristic subset bit encoding (see Section 4.2), survives with a positive true-bit embedding bias.

With respect to court-time confidence, 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 percent, as discussed in Section 5.

Unless specified otherwise, the bit-encoding used is the multi-hash encoding discussed in Section 4.2; 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, were 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 $5,000$ data values). Additionally, when experiments were performed on real-life test data this is specified in the figure captions. The real-life data sets [18] 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 refers to 30 days worth of data from the month of September 2003, totaling a number of 21,630 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 tone down distinct features for a given data set, features that interrelate figures. Instead of focusing on local variations, the figures should be interpreted as qualitative samples of global governing trends.

6.1 Random Alterations

In [22], we defined the *epsilon-attack* in the relational data framework (related attacks are defined also in [1], [14]), 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 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 Fig. 11 and Fig. 12 ($\mu = 0$), 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 Fig. 11a, label alteration increases with an increasing degree of data change. Smaller label bit sizes seem to better survive such an attack. In Fig. 11b, as the percentage of altered data items increases, the labeling scheme naturally degrades. Similar behavior can be observed for sampling (Fig. 11c) and summarization (Fig. 11d).

In Fig. 12, 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 $\tau = 50$ percent of the data altered within $\epsilon = 10$ percent (Fig. 12b), the detected bias is still above 25, yielding a false-positive rate of less than "one in 30 million".

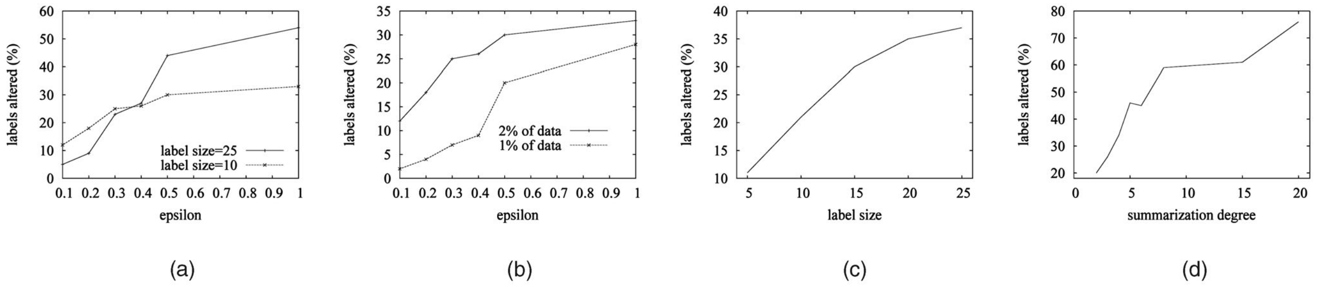


Fig. 11. (a) Label alteration for increasingly aggressive uniform altering epsilon attacks. Different label bit sizes shown. A smaller label size seems to survive better. (b) Different altered data percentages shown. (c) Label resilience under sampling conditions. A higher label length naturally yields an increased fragility to sampling. (d) Label alteration for summarization of increasing degree.

6.2 Sampling and Summarization

The ability to survive summarization (A1) and sampling (A2) is of extreme importance as both are expected attacks. In Fig. 11, the labeling algorithm is evaluated with respect to (Fig. 11c) sampling and (Fig. 11d) summarization. Intuitively, a higher label length results in increased fragility to sampling (shown is a sampling degree of 3). Summarization seems to be naturally survived by our design. For example, a summarization of the data down to 5 percent ($\nu = 20$) still preserves more than 20 percent of the original label values, thus conferring a strong back-bone to watermark embedding.

The behavior of the watermark encoding algorithm to sampling and summarization is outlined in Fig. 13. The natural strength of the bit encoding convention is clearly illustrated here. Both transformations are survived extremely well.

6.3 Segmentation. Combinations

In Section 5, 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 Fig. 13c, we analyze the impact of actual recovered segment size on the detected watermark bias. From a segment of only $2,000$ stream values we can detect a watermark bias of 10, corresponding to a very convincing low false positive rate of roughly 0.001.

In Fig. 13d, 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 percent sampling, followed by a 25 percent summarization process still yields a watermark bias of up to 20, corresponding to a low false-positive rate of "one in a million."

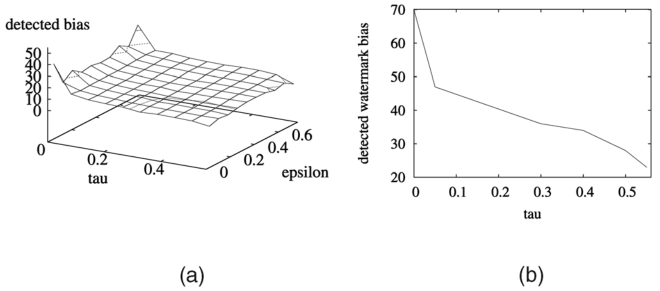


Fig. 12. Watermark survival to epsilon-attacks. (a) Naturally, increasing τ and ϵ values result in a decreasing watermark bias. (b) Same shown for $\epsilon = 10$ percent (real data).

6.4 Overhead and Impact on Data Quality

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 4.2. 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 4.2. Depending on the actual stream rate, these speed-ups can be gradually deployed to be able to keep up with the incoming data. 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.

While 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 multihash encoding discussed in Section 4.2 and parameters set such that the resulting watermark survives 100 percent 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 a practically insignificant 5.7 percent increase in processing times per stream item. The experimental setup could easily handle stream data rates of up to 3.5-4 million readings per second.

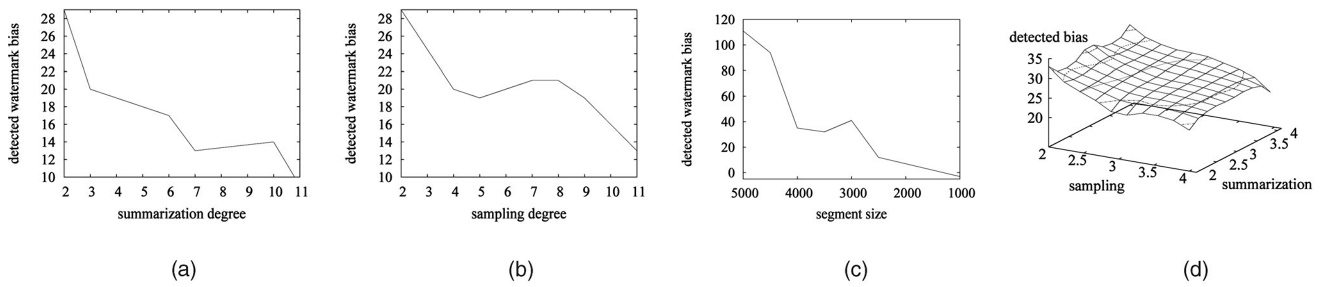


Fig. 13. Watermark survival to: (a) Summarization. An increasing summarization degree results in a decreasing detected watermark bias. (b) Sampling. A bias of 10 ensures a true-positive probability of 99.999 percent. (c) Segmentation. (d) Combined sampling and summarization (real data).

The poorest performer was the more complex multihash routine in Section 4.2 with an average increase of over 1,000 percent, as expected decreasing almost perfectly exponential with the decrease of guaranteed resilience (see Fig. 14a). In the high-resilience, multihash setting, an average watermarking throughput of 330 sensor readings per second was achieved; deploying such a setting makes sense exactly in such low data rate scenarios when the lack of sufficient data makes it important to offer a maximal encoding resilience.

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 multihash 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 4.2.

We also performed experiments evaluating the impact of our encoding on data quality. More specifically, we analyzed the alterations incurred by the mean and standard deviation of the stream data. For the above parameter settings, over a large number $(12,000+)$ 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 percent average from the original. The alteration to the standard deviation also maintained itself nicely within 0.27 percent of the original data. There exists a tunable trade-off between attack/transformation resilience and the incurred alterations. A lower level of resilience would definitely require less modifications to the data and have a lower impact on global statistics. In Fig. 14b, we show how decreasing the number of considered bit-encoding major extremes decreases the impact on the average and standard deviation in the result.

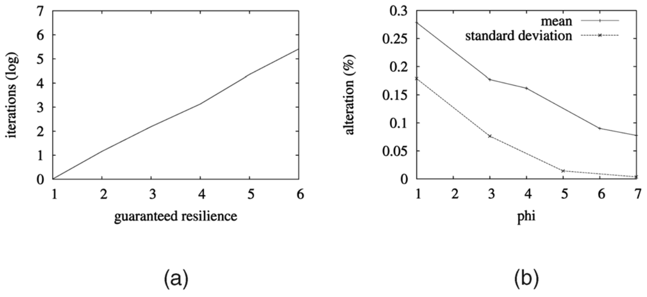


Fig. 14. (a) Computation overhead (iterations) in multihash encoding increases with increasing guaranteed resilience (e.g., sampling degree) levels (logarithmic scale). (b) Decreasing the number of considered bit-encoding extremes (increasing ϕ) decreases the impact on mean and standard deviation in the watermarked data.

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 did not have 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 multihash embedding used (see Section 4.2) might also hold some of the answers. We are further investigating this.

7 Conclusions

In this 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. Published research results of this work include [24].

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, and most likely value) and to experiment with alternative bit-encodings.

Acknowledgments

Portions of this work appeared in the Proceedings of the Very Large Databases Conference (VLDB) 2004 [24].

References

[1] R. Agrawal, P.J. Haas, and J. Kiernan, "Watermarking Relational Data: Framework, Algorithms and Analysis," *The VLDB J.*, vol. 12, no. 2, pp. 157-169, 2003.

[2] M. Arnold, S.D. Wolthusen, and M. Schmucker, *Techniques and Applications of Digital Watermarking and Content Protection.* Artech House Publishers, 2003.

[3] M.J. Atallah and S.S. Wagstaff Jr., "Watermarking with Quadratic Residues," *Proc. IS-T/SPIE Conf. Security and Watermarking of Multimedia Contents, SPIE,* vol. 3657, pp. 283-288, 1999.

[4] B. Babcock, S. Babu, M. Datar, and R. Motwani, "Models and Issues in Data Stream Systems," *Proc. ACM Symp. Principles of Database Systems (PODS),* pp. 1-16, 2002.

[5] M. Barni and F. Bartolini, *Watermarking Systems Engineering: Enabling Digital Assets Security and Other Applications.* Marcel Dekker, 2004.

[6] D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, N. Stonebraker, M. Tatbul, and S. Zdonik, "Monitoring Streams—A New Class of Data Management Applications," *Proc. Int'l Conf. Very Large Data Bases (VLDB),* 2002.

[7] S. Chandrasekaran and M.J. Franklin, "Streaming Queries over Streaming Data," *Proc. Int'l Conf. Very Large Data Bases (VLDB),* pp. 203-214, 2002.

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

[9] M. Datar, A. Gionis, P. Indyk, and R. Motwani, "Maintaining Stream Statistics over Sliding Windows," *Proc. ACM-SIAM Symp. Discrete Algorithms,* pp. 635-644, 2002.

[10] J. Eggers and B. Girod, *Informed Watermarking.* Kluwer Academic Publishers, 2002.

[11] N.F. Johnson, Z. Duric, and S. Jajodia, *Information Hiding: Steganography and Watermarking-Attacks and Countermeasures.* Kluwer Academic Publishers, 2001.

[12] J. Kang, J.F. Naughton, and S.D. Viglas, "Evaluating Window Joins over Unbounded Streams," *Proc. Int'l Conf. Design Eng.,* 2003.

[13] *Information Hiding Techniques for Steganography and Digital Watermarking,* S. Katzenbeisser and F. Petitcolas, eds., Artech House, 2001.

[14] J. Kiernan and R. Agrawal, "Watermarking Relational Databases," *Proc. 28th Int'l Conf. Very Large Databases (VLDB),* 2002.

[15] F. Korn, S. Muthukrishnan, and D. Srivastava, "Reverse Nearest Neighbor Aggregates over Streams," *Proc. Int'l Conf. Very Large Data Bases (VLDB),* 2002.

[16] Y. Li, V. Swarup, and S. Jajodia, "A Robust Watermarking Scheme for Relational Data," *Proc. Workshop Information Technology and Systems (WITS),* pp. 195-200, 2003.

[17] C.-S. Lu, *Multimedia Security: Steganography and Digital Watermarking Techniques for Protection of Intellectual Property,* Idea Group Publishing, 2004.

[18] NASA, The Hawaii Univ. Infrared Telescope Facility, http://irtfweb.ifa.hawaii.edu, 2004.

[19] B. Schneier, *Applied Cryptography: Protocols, Algorithms and Source Code in C.* Wiley & Sons, 1996.

[20] H.T. Sencar, M. Ramkumar, and A.N. Akansu, *Data Hiding Fundamentals and Applications: Content Security in Digital Multimedia.* Elsevier Science and Technology Books, 2004.

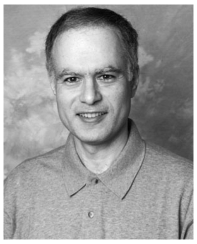
[21] R. Sion, "Proving Ownership over Categorical Data," *Proc. IEEE Int'l Conf. Data Eng. (ICDE),* 2004.

[22] R. Sion, M. Atallah, and S. Prabhakar, "Rights Protection for Relational Data," *Proc. ACM Special Interest Group on Management of Data Conf. (SIGMOD),* 2003.

[23] R. Sion, M. Atallah, and S. Prabhakar, "Relational Data Rights Protection through Watermarking," *IEEE Trans. Knowledge and Data Eng.,* vol. 16, no. 6, June 2004.

[24] R. Sion, M. Atallah, and S. Prabhakar, "Resilient Rights Protection for Sensor Streams," *Proc. Very Large Databases Conf. (VLDB),* 2004.

Radu Sion is an assistant professor of computer sciences at Stony Brook University. His research interests are in the areas of data security and privacy in distributed networked environments. Applications include: secure data outsourcing, queries over encrypted data, secure reputation systems, authentication, rights protection and integrity proofs in sensor networks, secure storage in peer-to-peer and ad hoc environments, data privacy and bounds on illicit inference over multiple data sources, security and policy management in computation/data grids, and detection of intrusions by access profiling for online Web portals. He is a member of the IEEE.



Mikhail ("Mike") Atallah received the PhD degree in 1982 from the Johns Hopkins University and joined the Computer Sciences Department at Purdue University, where he currently holds the rank of distinguished professor. He served on the editorial boards of many top journals (including *SIAM Journal on Computing*, *Journal of Parallel and Distributed Computing*, *IEETC*, etc.), and on the program committees of many top conferences and workshops (including PODS, SODA, SoCG, WWW, PET, DRM, SACMAT, etc.). He was keynote and invited speaker at many national and international meetings, and a speaker in the Distinguished Colloquium Series of six top Computer Science Departments. He was selected in 1999 as one of the best teachers in the history of Purdue University and included in *Purdue's Book of Great Teachers*, a permanent wall display of Purdue's best teachers past and present. He is a cofounder of Arxan Technologies Inc. He is a fellow of the IEEE.

Sunil Prabhakar is an associate professor of computer sciences at Purdue University. He received the Bachelor of Technology in electrical engineering from the Indian Institute of Technology, Delhi in 1990, and the MS and PhD degrees in computer science from the University of California, Santa Barbara in 1998. His research interests are in databases. In particular, he is currently working on management of uncertain data, probabilistic databases, sensor and streams databases, data privacy, and biological databases. He is a recepient of the US National Science Foundation CAREER award. He is a senior member of the IEEE and serves on the editorial board of the *Distributed and Parallel Databases Journal*. He has served on the program committees of leading database conferences including ACM SIGMOD, VLDB, ICDE, and ICDCS.