Self-Identifying Sensor Data

Stephen Chong
Harvard University
Cambridge, MA
chong@seas.harvard.edu

Christian Skalka
The University of Vermont
Burlington, VT
skalka@cs.uvm.edu

Jeffrey A. Vaughan
Harvard University
Cambridge, MA
jeff@seas.harvard.edu

ABSTRACT

Public-use sensor datasets are a useful scientific resource with the unfortunate feature that their provenance is easily disconnected from their content. To address this we introduce a technique to directly associate provenance information with sensor datasets. Our technique is similar to traditional watermarking but is intended for application to unstructured datasets. Our approach is potentially imperceptible given sufficient margins of error in datasets, and is robust to a number of benign but likely transformations including truncation, rounding, bit-flipping, sampling, and reordering. We provide algorithms for both one-bit and blind mark checking. Our algorithms are probabilistic in nature and are characterized by a combinatorial analysis.

Categories and Subject Descriptors

E.m [**Data**]: Miscellaneous; H.3.m [**Information Systems**]: Information Storage and Retrieval—*Miscellaneous*

General Terms

Design, Documentation, Reliability, Security.

Keywords

Provenance, Self-identifying data

1. INTRODUCTION

Datasets generated by sensor networks often benefit not just the producer but also subsequent generations of users. Many datasets are produced expressly for the public domain. However, conditions of use are typically attached that include the expectation of acknowledgment; that is, data producers expect that users will cite their contributions. Unfortunately the realities of data usage complicate this, as data is passed around and used nth hand. The original source is often forgotten, without malicious intent.

For example, the Hubbard Brook Ecosystem Study (HBES) in New Hampshire provides an on-line interface for downloading a myriad of ecological datasets generated in their experimental forest [18]. Users must complete a "Conditions of Use" form before data is delivered, that entails an obligation to acknowledge the data's provenance. However, there is no direct, irrevocable association of the data itself with its provenance.

In this paper we propose such an association via a technique we call *self-identifying data*. This technique allows a mark containing provenance information to be embedded in a dataset, and later checked or extracted from the dataset. Associated forms, signatures, files headers, or other types of annotations are unnecessary. Our technique is most reminiscent of watermarking.

1.1 Background and Contributions

Watermarking refers to any number of techniques for marking a real or virtual item with some sort of provenance identifier, in a manner that does not interfere with normal use of the item and is "indelible" by some measure. An analog example is the dollar bill, which contains a watermark that is visible when held against a bright light, using a technique developed in the 13th century to protect against counterfeit. Watermarking has more recently been adapted to the digital realm in a number of guises. For example, blowing up a Google Maps satellite image to the maximum will reveal a faint logo pattern that is invisible at lower (and more normal) resolutions. Watermarking has also been adapted to audio and video data, wherein signals are altered to carry the mark in subtle ways that are undetectable by human senses. In contrast to analog watermarking, digital watermarking usually does not prevent copying of data, but ensures that copying retains the watermark and thus the encoded provenance information.

Watermarking of digital media is typically treated as a signal processing issue [7]. Watermarking techniques in a more "combinatorial style" have also been studied in application to relational databases [1]. Usually these approaches leverage the structure of database relations to embed watermarks in the data itself.

Despite the myriad of digital watermarking techniques, none are appropriate for environmental sensor data. This sort of data typically does not lend itself to signal processing-style watermarking, and because it is unstuctured database watermarking techniques are not applicable. Also, sensor data usage patterns are unique to their user communities, and watermarking must be adapted to this, since watermarks should be impercievable in and robust to normal usage. Our contribution can be described as a new watermarking technique for sensor data. Similar to database watermarking it is combinatorial in nature, and allows typical sensor datasets to be marked in a manner that does not interfere with normal scientific use. But unlike database watermarking, the technique does not rely on the structure of datasets. While it is not a security mechanism per se, it is robust to manipulations such as reordering, sampling, and rounding, which we argue is sufficient to support fair-use policies in the scientific user community.

1.2 Characterization of the Technique

As described above, our goal is to design a system that can embed provenance information into, and automatically retrieve such information from, scientific datasets. Furthermore our techniques are intended to be robust against certain natural, non-adversarial dataset transformations. The remainder of the paper comprises a detailed description and analysis of our embedding and retrieval algorithms; now we provide a high-level characterization of our technique, using common parlance of the watermarking literature for clarity. In particular, we classify our system under the standard categories of *perceptibility*, *robustness*, and *capacity* [9].

Perceptibility.

This classification refers to whether a watermark can be "perceived" in the marked object; its meaning is media-dependent and defined with respect to human perception. Since watermarking has not been previously studied in the context of sensor datasets, we propose a definition of perceptibility in this context, under the assumption that any sensor dataset is a numeric representation of data. We say that a watermark is imperceptible in a sensor dataset if the watermark does not inhibit standard uses of the dataset. Specifically, watermarking should not significantly affect the scientific use of the dataset. Consider, as an example, the Sensirion SHT11 temperature sensor, which is accurate to at best $\pm 0.5°$C. Altering data produced by a Sensirion SHT11 by amounts orders of magnitude smaller than $0.5°$C does not affect use of that data, since such alterations are well within the sensor data error.

Robustness.

Robustness refers to how well a watermark stands up to transformations of data, benign or malicious. Our scheme is *semi-fragile*, in the sense that it is capable of withstanding many, but not all, transformations. It is not intended as a security mechanism per se; rather our goal is to withstand benign transformations that scientists might impose in the natural course of dataset usage. These transformations include *truncation* and *quantization* (rounding) of digits, as well as *random bit flips*. Our scheme is also robust to *sampling* of datasets, i.e. where a new dataset is generated by a selection of datapoints from the original dataset; this is also called a *subset attack* in the database watermarking literature [1]. *Reordering* of data within the dataset also has no effect on the reliability of our mechanisms.

Capacity.

This category refers to whether a mechanism allows to check if an object is marked with a given watermark (so-called *one-bit* watermarking), or whether it allows a watermark to be extracted from an object with no prior knowledge of the embedded watermark (so-called *blind* watermarking). Since we expect both capacities to support the user community, we define techniques for both one-bit and blind marking of sensor datasets. Indeed, our algorithm for the latter relies on the former for increased probability of correct mark extraction.

1.3 Outline of the Paper

The remainder of the paper is structured as follows. In Section 2 we provide a formal problem statement, define our corruption model, and summarize our embedding and retrieval technique at a high level. In Section 3 we describe our embedding algorithm in detail. In Section 4 we define our one-bit checking technique, and in Section 5 we define our blind retrieval algorithm. In both of these Sections we discuss how our approach is robust to various corrup-

tions. Our basic theory treats datasets as sequences of bit vectors representing natural numbers; in Section 6 we describe extensions allowing treatment of e.g. negative and floating point values. In Section 7 we report results of combinatorial analysis and empirical tests that illustrate robustness of our technique. In Section 8 we summarize deployment issues and how we might address them in future work. We discuss related work in Section 9, and conclude in Section 10.

2. PROBLEM DESCRIPTION AND TECHNICAL SUMMARY

In this section we provide a formal description of the problem of interest, and also an informal description of how we address it. For simplicity we will consider bit vector representations of numerals, where $\mathbb{V}$ denotes the set of all finite-length bit vectors.

2.1 Formal Problem Statement

A *dataset* $\mathcal{DS} \in list(\mathbb{V})$ is a list of bit vectors; each bit vector is a *datapoint*. A *transformation* is a function of type $list(\mathbb{V}) \to list(\mathbb{V})$; transformation f is a *corruption* if $f(\mathcal{DS}) \neq \mathcal{DS}$. An *encoding* E is a function of type $list(\mathbb{V}) * \mathbb{V} \to list(\mathbb{V})$ with the property that there exists a corresponding *retrieval function* R of type $list(\mathbb{V}) \to \mathbb{V}$ such that for all $\mathcal{DS}$ and $\mathsf{v} \in \mathbb{V}$:

$$R(E(\mathcal{DS}, \mathsf{v})) = \mathsf{v}$$

Intuitively, the encoding function $E(\mathcal{DS}, \mathsf{v})$ encodes bit vector v into dataset $\mathcal{DS}$, and the retrieval function retrieves v from that dataset.

We say that an encoding-retrieval function pair (E, R) is *robust* to a corruption f iff:

$$R(f(E(\mathcal{DS}, \mathsf{v}))) = \mathsf{v}$$

for all $\mathcal{DS}$ and $\mathsf{v} \in \mathbb{V}$. We say that (E, R) is robust to a set of corruptions C by generalizing f in the above definition to range over all functions in C and their compositions.

A *provenance mark* (or simply, *mark*) is a bit vector of fixed length L_m. The mark is an identifier that may refer to more elaborate metadata via some standard such as DOI. In this paper, we are not concerned with the allocation of provenance marks, or how a provenance mark is linked to more detailed metadata. We assume that the length of provenance marks L_m is universally agreed upon and known. Our goal is to define algorithms that implement an encoding-retrieval pair (E, R) that is robust to transformations in our corruption model described below.

Notation.

For bit vector $\mathsf{v} \in \mathbb{V}$, we write $|\mathsf{v}|$ for the length (size) of v, and write v_i to refer to the ith bit of v, where $i = 0$ is the most significant (left-most) bit and $i = |\mathsf{v}| - 1$ is the least significant (right-most) bit. We write $\mathsf{v} @ \mathsf{v}'$ for the concatenation of bit vectors v and v'. Finally, we write $|L|$ for the length of list L.

2.2 Corruption Model

In our corruption model we allow the following transformations and any composition thereof:

- *Rounding.* Datapoints are rounded to the nearest multiple of some integer n.

- *Truncation.* Some number of least-significant bits may be removed from any datapoint in the given dataset.[1]

---

[1] We do not regard truncation as a special case of rounding. Trun-

- *Deletion/Sampling.* Datapoints may be removed from the given dataset.

- *Reordering.* Datapoints in a dataset may be permuted.

- *Bit flip.* One or more bits of any datapoint in the given dataset may be changed.

We assume that truncation, rounding, sampling, and reordering are more likely to occur than bit flips. For example, copying data (for example, from one spreadsheet to another) is more likely to truncate or round data values than it is to randomly flip bits. Note that rounding and sampling tend to corrupt or remove the less significant bits from datapoints. That is, more significant bits in datapoints are more likely to be unaffected by truncation and rounding. Our embedding and retrieval processes are designed to take advantage of this behavior.

There are of course limits on the robustness of any encoding-retrieval pair within this corruption model. For example removing all datapoints from a dataset will disallow retrieval. Similarly, replacing the entire dataset with random data through bit flips will also prevent retrieval.

2.3 Informal Summary of the Technique

Our technique embeds provenance information in the lower-order bits of datapoints in a dataset. The embedding is designed to be robust to corruption in the lower-order bits of datapoints. That is, our technique assumes that lower-order bits are more likely to be corrupted than higher-order bits.

To support one-bit checking (wherein given dataset $\mathcal{DS}$ and provenance mark m, we want to answer the question "was m embedded in $\mathcal{DS}$?") we use two of the lower-order bits of each datapoint in the dataset as *check bits*. One of the check bits is used to help the retrieval process determine the parameters used for the embedding; the other check bit is a hash of the encoded provenance mark concatenated with the more significant bits of the datapoint. Given sufficiently many datapoints whose check bits (and more significant bits) are uncorrupted, the encoding parameters can be retrieved, and one-bit checking supported.

To support blind checking (wherein given dataset $\mathcal{DS}$, we want to answer the question "which provenance mark was embedded in $\mathcal{DS}$?"), our encoding technique breaks a provenance mark into a number of smaller pieces, and replaces the least significant bits of each datapoint with one of these *provenance pieces*. We leverage *redundancy*, allowing a bit of the provenance mark to appear in more than one provenance piece, which provides greater robustness to truncation and rounding. The significant bits of a datapoint are used to determine which provenance piece replaces the least significant bits. During retrieval, the least significant bits of the datapoints are examined to retrieve a "best guess" at the provenance mark, which can then be tested using one-bit checking. If the best guess is incorrect, the retrieval process permits limited search of possible provenance marks, where more likely marks are considered first.

3. EMBEDDING A PROVENANCE MARK IN A DATASET

In this section we describe how we embed a provenance mark m of length L_m into a dataset. We assume that each datapoint in the

cation of the least-significant k bits will affect only k bits; rounding to the nearest $n = 2^k$ may affect an arbitrary number of bits. For example, 10111 (=23) rounded to the nearest 4 is 11000 (=24), with 5 bits affected.
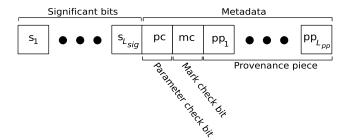

Figure 1: Anatomy of an Annotated Datapoint

dataset contains *insignificant bits*: a non-zero number of least significant bits that can be safely manipulated within the error bounds of the device that generated the data. We embed a provenance mark into a dataset by encoding *metadata* into the insignificant bits of each datapoint. We further assume that each datapoint in a dataset is the same length, contains the same number of insignificant bits, and represents non-negative integers. If datapoints are of different length, we can, without loss of generality, pad datapoints with leading zeros until they are the same length. The number of insignificant bits is inherent in the physical characteristics of the sensor generating the dataset, and so all datapoints in the dataset should have an identical number of insignificant bits. In Section 6 we describe how our technique can be generalized to floating point and negative values.

Let L_{md} denote the number of insignificant bits in a datapoint, and L_{sig} denote the number of significant bits; thus the length of any datapoint is $L_{md} + L_{sig}$. Clearly the values of L_{md} and L_{sig} are dependent on the given dataset, since the number of significant bits is a property of the sensors used to collect the data.

We refer to a datapoint in which the insignificant bits have been replaced with metadata as an *annotated datapoint*. The process of embedding a provenance mark in a dataset produces an *uncorrupted annotated dataset* containing uncorrupted annotated datapoints.

Figure 1 shows the structure of an annotated datapoint. The L_{md} insignificant bits have been replaced with a *parameter check bit*, a *mark check bit*, and a *provenance piece*. We thus require that each datapoint contains at least 3 insignificant bits (i.e., $L_{md} \geq 3$). The provenance piece is a subset of bits of the provenance mark; Section 3.1 describes how provenance pieces are derived from a provenance mark. The parameter check bit and mark check bit are derived from the datapoint's significant bits and the parameters for the embedding, described in Section 3.2. The check bits enable the retrieval process to verify, with high probability, which provenance mark was embedded in the dataset.

3.1 Provenance Pieces

Each annotated datapoint contains $L_{md} - 2$ bits of the provenance mark, referred to as a provenance piece. If the length of the provenance mark L_m is greater than $L_{md} - 2$, then a single datapoint cannot contain all bits of the provenance mark. The embedding process chooses provenance pieces to ensure that an annotated dataset contains all bits of the provenance mark with high probability.

The embedding process takes as a parameter N_{pp}, the number of distinct provenance pieces. For each datapoint one of the N_{pp} provenance pieces is chosen to be embedded into it. Specifically, given a datapoint with significant bits s, we choose the kth provenance piece, where $k = \text{hash}(N_{pp}, \text{s})$ and $\text{hash}(n, \text{v})$ is a cryptographic hash of bit vector v that returns a value in $\mathbb{Z}_n$. Provided

provenance mark m:	0100110001001011
1st provenance piece pp^0:	01001100
2nd provenance piece pp^1:	11000100
3rd provenance piece pp^2:	01001011
4th provenance piece pp^3:	10110100

$$L_m = 16, \ N_{pp} = 4, \ L_{pp} = 8$$

Figure 2: Provenance pieces example.

that a dataset has sufficient variation in the significant bits of its constituent datapoints, the N_{pp} provenance pieces will be embedded uniformly at random throughout the dataset. Thus, any sufficiently large random subset of an annotated dataset is likely to include every distinct provenance piece. This supports robustness in the retrieval process. Section 6 discusses an extension to this technique that treats low entropy datasets.

Given provenance mark m of length L_m, and given the number of provenance pieces N_{pp} and insignificant bits L_{md}, then the N_{pp} distinct provenance pieces are defined as follows. Let $L_{pp} = L_{md} - 2$ be the length of each provenance piece. For $k \in 0..(N_{pp} - 1)$ and $i \in 0..(L_{pp} - 1)$, let the ith bit of the kth provenance piece, denoted pp_i^k, be defined as

$$\mathsf{pp}_i^k = \mathsf{m}_j \text{ where } j = \left(k\frac{L_m}{N_{pp}} + i \right) \mod L_m.$$

Note that a given bit of provenance mark m may appear in more than one provenance piece. This redundancy means that not all distinct provenance pieces need to be available during the retrieval process. Furthermore, since the same mark bit may occur in a more significant position in one provenance piece than another, redundancy provides robustness to truncation and rounding.

Figure 2 gives an example of how a provenance mark of length 16 is split into four distinct provenance pieces of length 8 (i.e., $L_m = 16$, $N_{pp} = 4$, and $L_{pp} = 8$). Note that each provenance piece contains 8 contiguous bits of the provenance mark. (The fourth provenance piece, pp^3, contains the last four bits of m followed by the first 4 bits.) Note also that each bit of the provenance mark appears in exactly two provenance pieces. Thus, provenance mark m could be retrieved if the retrieval process has either provenance pieces pp^0 and pp^2, or provenance pieces pp^1 and pp^3. This redundancy means that a subset of the provenance pieces may suffice to retrieve the provenance mark. Furthermore, if a dataset was truncated by, say, a single bit, then the least significant bit of piece pp^0 (corresponding to the 7th bit of m) would be lost from every instance, but the 7th bit of m would still available in all instances of piece pp^1.

To ensure that each bit of the provenance mark appears in the same number of distinct provenance pieces, we require that $N_{pp} \times L_m$ is divisible by L_{pp}. Moreover, since each provenance piece should contain distinct bits of the provenance mark, we require that $N_{pp} \leq L_m$.

3.2   Parameter and Mark Check Bits

The parameter check bit and mark check bit are derived from the encoding parameters, and are used in one-bit checking and during retrieval to determine with high probability if the correct provenance mark has been retrieved. The parameter check bit pc for a datapoint with significant bits s is computed by taking the hash of s and the number of distinct provenance pieces, N_{pp}. The mark check bit mc is the hash of s and the provenance mark m. Formally,

we have:

$$\mathsf{pc} = \operatorname{hash}(2, \mathsf{s} @ N_{pp}) \tag{1}$$
$$\mathsf{mc} = \operatorname{hash}(2, \mathsf{s} @ \mathsf{m}). \tag{2}$$

Section 4 describes how the parameter and mark check bits are used by one-bit checking.

4.   CHECKING A PROVENANCE MARK

The embedding process embeds a provenance mark m into a dataset to produce an uncorrupted annotated dataset. As the dataset is disseminated and used, it may be corrupted, for example by datapoints being rounded or deleted as described in Section 2.2. However, we can use a corrupted annotated dataset for both *one-bit watermarking* and *blind watermarking*. In one-bit watermarking, we can determine with high probability whether a provenance mark m$'$ is the provenance mark m that was embedded into the dataset. Blind watermarking allows us to retrieve m with high probability.

In this section, we describe how to perform one-bit watermarking using a corrupted annotated dataset. In Section 5 we describe how to perform blind watermarking, and our technique there relies on the one-bit checking described here to increase probabilities of correctness.

4.1   Retrieving L_{sig} and N_{pp}

The embedding process adds a parameter check bit to each datapoint to assist the retrieval of embedding parameters L_{sig} (the number of significant bits of a datapoint) and N_{pp} (the number of distinct provenance pieces used in the embedding process). The retrieval process guesses values for L_{sig} and N_{pp}, and uses the parameter check bits to determine (with high probability) when it has guessed the values correctly.

Given a corrupted annotated datapoint d, we say that d is *pc-consistent* with guesses L_{sig} and N_{pp} iff

$$\operatorname{hash}(2, \mathsf{d}_0 \dots \mathsf{d}_{L_{sig}-1} @ N_{pp}) = \mathsf{d}_{L_{sig}}.$$

Note that if the guesses for L_{sig} and N_{pp} are correct, and bits $\mathsf{d}_0 \dots \mathsf{d}_{L_{sig}}$ have not been corrupted, then $\mathsf{d}_{L_{sig}}$ is the parameter check bit as computed by Equation 1. If the guesses for L_{sig} and N_{pp} are incorrect, or one or more bits $\mathsf{d}_0 \dots \mathsf{d}_{L_{sig}}$ have been corrupted, then the probability of the datapoint being pc-consistent is approximately $\frac{1}{2}$, due to the properties of the cryptographic hash function used to generate the parameter check bit.

We define the *pc-consistency score* of corrupted annotated dataset $\mathcal{DS}$ for guesses L_{sig} and N_{pp} to be the proportion of datapoints in $\mathcal{DS}$ that are pc-consistent for guesses L_{sig} and N_{pp}. If the guesses are correct, and the first $L_{sig} + 1$ bits of each datapoint have not been corrupted, then the pc-consistency score will be 1. If the guesses are incorrect, then (regardless of the corruption of the datapoints) the expected pc-consistency score is $\frac{1}{2}$. In general, the probability of an incorrect guess having pc-consistency score of 1 is 2^{-n}, where $n = |\mathcal{DS}|$, which is vanishingly small for an annotated corrupted dataset $\mathcal{DS}$ of reasonable size.

Because guesses L_{sig} and N_{pp} are drawn from limited domains ($\{1 \dots \max\{|\mathsf{d}| \mid \mathsf{d} \in \mathcal{DS}\}\}$ and $\{1 \dots L_m\}$ respectively) it is feasible to enumerate all possible guesses (L_{sig}, N_{pp}) and calculate their pc-consistency score. The parameters with the best pc-consistency score are used for the following step, checking a provenance mark. In Section 7, we investigate the effects of corruption on the pc-consistency score.

4.2   Checking a Provenance Mark

Suppose we have a corrupted annotated dataset $\mathcal{DS}$ for which we know the embedding parameter L_{sig}, and we have a guess at

the provenance mark m. (We describe in Section 5 how we retrieve one or more guesses for the provenance mark for blind checking.) The embedding process adds a mark check bit to each datapoint to determine with high probability when the correct provenance mark has been guessed.

Given a corrupted annotated datapoint $d \in \mathcal{DS}$, we say that d is *mc-consistent* with L_{sig} and m iff

$$\text{hash}(2, d_0 \ldots d_{L_{sig}-1} @ m) = d_{L_{sig}+1}.$$

Note that if the guesses for L_{sig} and m are correct, and the bits $d_0 \ldots d_{L_{sig}+1}$ have not been corrupted, then $d_{L_{sig}+1}$ is the mark check bit as computed by Equation 2. Otherwise, the probability of the datapoint being mc-consistent is approximately $\frac{1}{2}$.

The *mc-consistency score* of dataset $\mathcal{DS}$ for L_{sig} and m is the proportion of datapoints in $\mathcal{DS}$ that are mc-consistent for L_{sig} and m. As with the pc-consistency score, if L_{sig} and m are correct and the first $L_{sig} + 2$ bits of each datapoint are uncorrupted, the mc-consistency score will be 1, otherwise the expected mc-consistency score is $\frac{1}{2}$. In Section 7, we investigate the effects of corruption on the mc-consistency score, and given an mc-consistency score, when that implies the provenance mark is correct.

5. RETRIEVING THE PROVENANCE MARK

In addition to the parameter check bit and mark check bit, the embedding process adds to each point in the dataset a provenance piece containing bits of the provenance mark m. To retrieve a provenance mark from a corrupted annotated dataset (also known as *blind watermarking*), we extract provenance pieces from the dataset and combine them to construct a best guess at the provenance mark. Because the dataset may have been corrupted, the provenance pieces embedded into datapoints may not contain all the correct bits of the embedded provenance mark. However, due to the construction of the provenance pieces, it is likely that some information about provenance mark m can be recovered as a best guess. This guess can be checked for correctness using the mark check bits, as described in Section 4.2.

For presentation purposes, we define a useful function $split(\cdot)$ that takes datapoint d and, based on parameters L_{sig} and N_{pp}, returns information about the significant bits, check bits, and provenance piece retrieved from d. Formally, for datapoint d, and parameters L_{sig} and N_{pp}, we define $split(d) = (s, pc, mc, pp, k)$ where

- $s = d_0 \ldots d_{L_{sig}-1}$; and

- $pc = d_{L_{sig}}$ and

- $mc = d_{L_{sig}+1}$; and

- $pp = d_{L_{sig}+2} \ldots d_{|d|-1}$; and

- $k = \text{hash}(N_{pp}, s)$.

Assuming that the datapoint has not been corrupted and L_{sig} and N_{pp} were the parameters used in the embedding process, then s is the significant bits of d, pc and mc are the parameter check bit and mark check bit respectively, pp is the provenance piece that was embedded into the datapoint, and k indicates which provenance piece was chosen for this datapoint. If the datapoint is corrupted, then one or more bits may be incorrect.

However, we provide redundancy in the encoding of the provenance mark in two ways. First, each datapoint contains a provenance piece; even if some datapoints are corrupted, there are likely to be other datapoints that contain the same provenance piece, possibly uncorrupted. Second, the encoding parameter N_{pp} can be set so as to ensure that each bit of the provenance mark appears in

more than one distinct provenance piece; if some of the less significant bits of a provenance piece are corrupted due to rounding or truncation, those bits will appear in another provenance piece in a more significant position, making the retrieval process more robust to corruptions such as rounding or truncation. For example, in Figure 2, the first bit of the provenance mark appears in provenance piece pp^3 in the fourth least-significant position, and in provenance piece pp^0 in the most significant position.

The *suggestion for bit i of the provenance mark* of datapoint d is the information that datapoint d contains about the ith bit of the provenance mark. Specifically, $suggest(d, i)$ is either $*$ (if d contains no information about the ith bit of the provenance mark) or a pair (b, c), where b is the bit (0 or 1) that d suggests for the ith bit of the provenance mark, and $c \in \mathbb{N}$ is the confidence of that suggestion. Formally, we define:

$$suggest(d, i) = \begin{cases} (pp_j, j) & \text{if } j + k \mod L_m = i \text{ and} \\ & \quad 0 \le j < |pp| \\ * & \text{otherwise} \end{cases}$$

where $(s, pc, mc, pp, k) = split(d)$.

The confidence of suggestions is a natural number, where higher numbers indicate less confidence. We use the index within the provenance piece at which the ith bit of the provenance mark occurs. Thus, the most confident suggestion is one where the ith bit of the provenance mark appears in the most-significant position of the provenance piece. This reflects our corruption model, where less-significant bits of a datapoint are more likely to be corrupted than more-significant bits.

For example, if datapoint d is uncorrupted and contains provenance piece pp^2 from Figure 2, then $suggest(d, i) = *$ for $0 \le i \le 7$ (since pp^2 contains no information about the first 8 bits of the provenance mark), and $suggest(d, 10) = (pp^2_2, 2) = (0, 2)$.

We lift the definition of $suggest(d, i)$ from datapoints to datasets: $suggest(\mathcal{DS}, i)$ is a list of suggestions for bit i of the provenance mark, derived from the datapoints of dataset $\mathcal{DS}$. We ignore datapoints d that contain no information about the ith bit of the provenance mark (i.e., we ignore datapoints d such that $suggest(d, i) = *$). Formally, we define $suggest(\mathcal{DS}, i)$ as follows.

$$suggest(\mathcal{DS}, i) = \{suggest(d, i) \mid d \in \mathcal{DS} \wedge suggest(d, i) \ne *\}.$$

Given $suggest(\mathcal{DS}, i)$, there are many ways to compute the "best guess" for the ith bit of the provenance mark. One possibility is that we select the bit that the majority of suggestions propose, regardless of the confidence of any suggestion. We call this *allVote* defined as follows (we use L to range over lists of suggestions of the form (b, c), and $\text{round}(\cdot)$ rounds real numbers to the nearest integer):

$$allVote(L) = \text{round}\left(\frac{\sum_{(b,c) \in L} b}{|L|}\right)$$

Another possibility is that we choose the bit that the majority of the most confident suggestions propose. Function *bestVote* considers only the most confident suggestions.

$$bestVote(L) = \text{let } L' = \{(b, c) \mid (b, c) \in L \wedge$$
$$c = \min\{c' \mid (b', c') \in L\}\}$$
$$\text{in round}\left(\frac{\sum_{(b,c) \in L'} b}{|L'|}\right)$$

Other functions are possible, such as weighting the vote b from suggestion (b, c) based on the confidence c—more confident suggestions receive greater weight.

```
search(n, m):
    if n = 0 then
        check possible provenance mark m
    else
        search(n − 1, m)
        flip bit iₙ of m
        search(n − 1, m)
```

Figure 3: Directed search algorithm.

Now, given some function f (such as *allVote* or *bestVote*) for computing a bit from a list of suggestions, we can compute a "best guess" m at a provenance mark from a dataset. The ith bit of the best guess m is computed as

$$\mathsf{m}_i = f(suggest(\mathcal{DS}, i)).$$

Given thusly computed best guess m, we can verify whether it is the originally embedded provenance mark via one-bit checking as described in Section 4.2. The construction of the best guess is robust to many transformations of the dataset; Section 7 presents related analysis in detail.

However, if the dataset is too corrupted, the best guess may be incorrect. If the mark check bits of the dataset are also severely corrupted, then there is insufficient information in the dataset to determine if we have recovered the correct provenance mark. But if the mark check bits are mostly uncorrupted, then we can *search* for the correct provenance mark, using mc-consistency to determine when we have succeeded.

5.1 Directed Search

The space of possible provenance marks is too large to search exhaustively. However, given a best guess for a provenance mark, and some measure of confidence in each bit of that guess, we can direct the search of possible provenance marks so that we check more likely marks first. This allows allocation of a budget for searching, with the budget being used to check the best candidates first.

We assume that we have a best guess m for the provenance mark, and for each i, confidence c_i in the ith bit of m. Confidence c_i may be derived from the suggestions of datapoints, for example, using the average of confidences of suggestions used to compute the ith bit of m. However, other measures of confidence are possible, such as using the entropy of the suggestions for the ith bit. For example, when using *allVote* to compute the ith bit of the provenance mark, if $\frac{\sum_{(b,c)\in L} b}{|L|}$ is close to 0.5, then there is much entropy in the suggestions for bit i, and as a result, there should be low confidence for that bit.

Let $i_1, \ldots, i_{L_m}$ be a permutation of $1..L_m$ such that $c_{i_1} \geq c_{i_2} \geq \cdots \geq c_{i_{L_m}}$. That is, $i_1, \ldots, i_{L_m}$ orders the L_m bits of the provenance mark guess m by increasing confidence, with i_1 being the index of the bit that we have the least confidence in, and i_{L_m} being the index of the bit that we have the most confidence in. The recursive algorithm sketched in Figure 3 checks possible provenance marks in decreasing order of confidence, starting with a call *search(L_m, m)*, where m is the best guess at the provenance mark. Note that m is the first provenance mark checked, and subsequent possible provenance marks are obtained by varying the bits in which we have the least confidence. An implementation could stop once the mc-consistency score of a possible provenance mark is above a certain threshold. Alternatively, given a *search bound sb*, it could consider only the first *sb* possible provenance marks and choose the best provenance mark of those considered. For example, by calling *search*(10, m) only the best 2^{10} possible provenance

marks will be considered, equivalent to $sb = 1024$.

We discuss the robustness of the retrieval process, and the efficacy of directed search, in Section 7.

6. EXTENSION TO NEGATIVE, DECIMAL NUMBERS, LOW-ENTROPY DATASETS

Our theory is based on view of datasets as lists of bit vectors. It is easy to see how this covers positive integral data, but sensor data may also be negative and/or floating point. However, note that bit vectors serve as an internal representation for the algorithm and need not correspond to the original numeric representation of data. Thus, we are free to perform pre- and post-processing on datasets during embedding and retrieval provided this processing is imperceptible. Pre-processing yields a bit vector that is treated via our standard technique. We have already anticipated this approach in Section 3, where we have discussed padding variable-length bit vectors with zeroes to obtain fixed length. This processing must be performed during retrieval as well as embedding.

Negative integers.

Negative integral numbers are easily dealt with: we transform them into a signed integer bit vector representation, and process (both embedding and retrieval) as normal. The embedding process does not alter the most significant bits, and so leaves the sign unchanged. If we are required to pad variable-length bit vectors, we preserve the sign bit, and pad the magnitude with zeros. Other bit representations for negative integers could also be used.

Floating point numbers.

Treatment of floating point numbers is slightly more involved, but despite the variety of digital representations of floating point numbers our proposed technique is uniform and fairly simple. Assuming the data is in decimal notation, we pre-process by determining the smallest number n such that multiplying all datapoints by 10^n will produce a dataset with integral values. We then multiply each datapoint by 10^n, and perform the embedding in the standard manner on the transformed dataset in bit vector representation. Finally, we post-process by multiplying each datapoint by 10^{-n} to obtain the annotated dataset for dissemination. Pre-processing and post-processing during retrieval is the same. Note if corruption has truncated or rounded datapoints, then the minimal multiplier 10^n that makes all datapoints integral may be different from the multiplier used in encoding, and we may need to try several nearby values: 10^{n+1}, 10^{n+2}, etc. Note also that this technique relies on the floating point numbers in the dataset having similar magnitude, which ensures that different datapoints have approximately the same number of insignificant bits. For negative floating point numbers, we compose these processing steps with the above-described processing steps for negative numbers.

Other schemes for encoding floating point numbers are possible, and the choice of encoding technique may have a measurable performance impact. We leave investigation of this issue to future work.

Low-entropy datasets.

Some datasets may contain few distinct values. In the extreme, every datapoint in a dataset may be the same value. This presents a challenge for our embedding technique, since it uses the significant bits of the datapoint to determine the check bits and which provenance piece to use for the insignificant bits of the datapoint. Datapoints with the same value will have the same check bits and provenance piece.

We can address this challenge during pre-processing of the embedding, by adding entropy to the dataset in the insignificant bits, and treating these as significant in a standard embedding. This will however reduce the number of insignificant bits available for encoding metadata. What is a sufficient amount of noise will depend on the entropy of the dataset and N_{pp}.

7. EVALUATION AND ANALYSIS

We have developed prototype tools for encoding provenance marks into datasets, and for performing one-bit and blind mark checking. The tools are implemented in approximately 1,300 lines of non-comment, non-blank lines of Perl code. We have also developed tools that corrupt datasets by rounding, truncating, and sampling datapoints. In this section we empirically evaluate the effectiveness of our techniques using the prototype tools on artificially generated datasets. We also consider analytically how our encoding techniques affect statistical properties of datasets.

7.1 Mark checking

7.1.1 Retrieving encoding parameters

Recall from Section 5 that the first step in both one-bit and blind checking is to retrieve the encoding parameters L_{sig} and N_{pp} by computing the pc-consistency score of all possible parameters, and selecting the candidate with the greatest score. For an uncorrupted annotated dataset, the correct encoding parameters yield a pc-consistency of 1, and all other candidates will have a pc-consistency score of approximately $\frac{1}{2}$.

For corrupted datasets, the correct encoding parameters may have a pc-consistency score less than 1. Retrieving these parameters is still possible if there exists a significant gap between the greatest and second-greatest pc-consistency scores, which correspond to the correct and best-looking incorrect answer respectively. If the encoding parameters cannot be retrieved, then neither one-bit nor blind checking can be performed.

The retrieval algorithm examines only the significant bits and the parameter check bit of a datapoint; it does not examine the other $L_{md} - 1$ bits of metadata. Thus truncation and bit flips do not effect parameter recovery so long as the corruption is limited to low-order bits. Furthermore, pc-consistency scoring is unaffected by dataset reordering. Below we consider how sampling and rounding affect parameter retrieval.

Effect of sampling.

Figure 4 show the pc-consistency scores for all non-trivial encoding parameter candidates for annotated datasets of various sizes. Samples were drawn from a synthetic dataset with 10,000 datapoints generated by choosing elements uniformly at random, with replacement, from the set $\{1, 2, \ldots, 5 \times 10^6\}$. We assume that there are 13 significant bits ($L_{sig} = 13$), and, since the synthetic data can be represented in 23 bits, there are 10 least significant bits that we can use to encode metadata ($L_{md} = 10$). A 32-bit provenance mark ($L_m = 32$) was encoded in the dataset by replacing the 10 least significant bits; there were 8 distinct provenance pieces ($N_{pp} = 8$), and the length of each provenance piece was 8 ($L_{pp} = L_{md} - 2 = 8$). Unless otherwise stated, all experiments in this section used the same parameters. When retrieving the encoding parameters, degenerate encoding parameters (with $L_{sig} \leq 2$) were ignored.

In all cases, the correct parameters have a pc-consistency score of 1. For sufficiently large samples (say, at least 50), the pc-consistency scores of incorrect parameters cluster around $\frac{1}{2}$, since incorrect parameters are consistent with a given parameter check bit half the
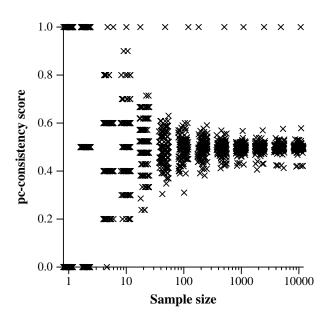

Figure 4: pc-consistency scores vs. sample size. Each thick column represents pc-consistency scores from a single run of the parameter recovery algorithm.

time, and thus the pc-consistency scores of incorrect parameters are binomially distributed. Thus, in a dataset with n distinct values, the probability of an incorrect parameter have a pc-consistency score of 1 is approximately 2^{-n}. For small sized samples, there is a much greater chance of an incorrect parameter having a high pc-consistency score.

Thus, the correct encoding parameters can be retrieved with high probability given a sufficiently large sample of an otherwise uncorrupted annotated dataset, say at least 50 datapoints. This result is independent of the length of the provenance mark L_m, length of the metadata L_{md}, and number of provenance pieces N_{pp}.

Effect of rounding.

Figure 5 shows maximum pc-consistency scores computed when decoding a dataset corrupted by rounding datapoints to the multiples of k, for varying values of k. The dataset contained 10,000 datapoints, but the pc-consistency scores were calculated using a randomly chosen 100 element sample.

As we round by larger quantities, top scores fall and it becomes harder to distinguish the correct encoding parameters from the incorrect parameters. Rounding to the nearest multiple of k for $2 \leq k \leq 128$ leaves top pc-consistency scores far above those expected for arbitrary parameters, thus allowing the successful retrieval of the encoding parameters. At first, large values of k yield substantially lower top pc-consistency scores, however once rounding as eliminated enough significant bits, it becomes easy to overfit during parameter recovery.

The value of k for which it becomes difficult to determine the correct encoding parameters is independent of the length of provenance mark L_m and number of provenance pieces N_{pp}. It is however dependent on the length of the metadata L_{md}: using more bits for metadata provides robustness for larger values of k.

7.1.2 One-bit checking

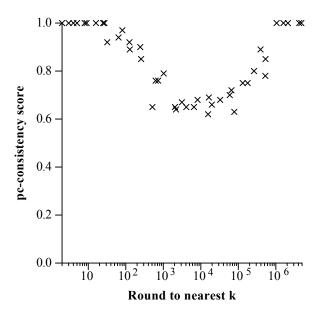One-bit checking evaluates the mc-consistency score of a single

Figure 5: Max pc-consistency scores vs. degree of rounding.

Figure 6: mc-consistency scores of incorrect provenance marks.

provenance mark. A sufficiently high mc-consistency score indicates that it is likely that the same provenance mark was embedded in the dataset.

Since one-bit checking uses just the mark check bit, and does not use the $L_{pp} = L_{md} - 2$ bits of the provenance piece, its robustness with respect to various corruption is very similar to that of recovering the encoding parameters, which uses the parameter check bit. It is unaffected by truncation or bit-flipping that affect only the $L_{md} - 2$ least-significant bits of datapoints. It is also unaffected by reordering of datapoints. The effects of sampling and rounding on one-bit checking are the similar to their on parameter recovery.

What mc-consistency score indicates that we have the same provenance mark that was used during encoding? The acceptance threshold for mc-consistency scores—especially in view of potential corruption—is best determined empirically. In our experience, 0.85 is a conservative threshold; given a sufficiently large dataset (containing, say, 100 distinct values), the probability that an incorrect provenance mark will have a mc-consistency score of 0.85 or higher is approximately 2.41×10^{-13}—about two chances in 10 trillion. Intuitively, this is because the mc-consistency scores of incorrect provenance marks are binomially distributed. Figure 6 demonstrates this with a histogram of mc-consistency scores for about 10,000 incorrect provenance marks, in a dataset of 100 datapoints.

7.1.3 Blind checking

Blind checking attempts to retrieve a provenance mark from a corrupted annotated dataset knowing only the length of the mark. It uses a heuristic to generate one or more guesses, and uses one-bit checking as a subroutine to evaluate the guesses. Blind checking is not affected by reordering of datapoints. It is affected by sampling, since too few datapoints may not contain all bits of the encoded provenance mark. Blind checking can be affected by bit flips, truncation, and rounding; however, the encoding scheme is designed to be robust to corruption in lower order bits, and truncation and rounding are more likely to corrupt lower order bits.
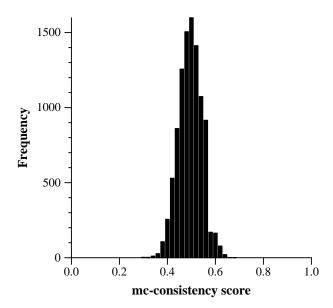
Effect of redundancy and sampling.

A single bit of the provenance mark may occur in many provenance pieces. We write $R = k$ to indicate an annotated dataset has redundancy k—that is, each bit of the provenance mark appears in k provenance pieces. As shown in Figure 2, encoding a provenance mark of length 16 using $N_{pp} = 4$ and $L_{pp} = 8$ results in each bit of the provenance appearing in two distinct provenance pieces: $R = 2$. Increasing redundancy substantially increases likelihood that decoding will succeed in the presence of corruption of lower-order bits. However, more redundancy also increases the probability that a small sample won't contain all bits of a provenance mark.

The probability of recovering a provenance mark from a dataset with a given size and redundancy can be calculated analytically using a combinatorial argument. Figure 7 presents the predicted probability of recovering a provenance mark from annotated datasets of various size. Here, recovery means that the sample contains datapoints with enough provenance pieces so that every bit of the provenance mark appears in at least one provenance piece. Other than sampling, the annotated dataset is not corrupted (i.e., no bit-flipping, rounding, etc.). Note that the probabilities for recovery depend only on the redundancy and sample size, and are independent of provenance mark length L_m, and provenance piece length L_{pp}; however, not all redundancies are possible with given encoding parameters. For example, R cannot be greater than L_{pp}.

Figure 7 demonstrates that the probability of blind mark recovery undergoes a "phase transition" around sample size 10. For smaller samples, there are not enough distinct provenance pieces present to recover all the bits of the provenance mark; for larger samples, recovery is very likely.

Also plotted on Figure 7 are observed recovery rates for a series of experiments using $L_m = 32$, $L_{pp} = 8$ and $N_{pp} = 8$ (which implies $R = 2$). Taking samples of various sizes from a large uncorrupted annotated dataset, we measured how often the sample contained sufficient information to reconstruct all bits of the provenance mark (without any search). The observed success rate matches the predicted success rate well.
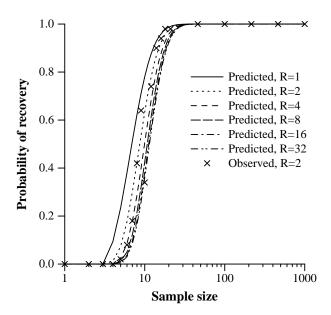
8

Figure 7: Predicted and observed probability of recovering provenance mark.

Figure 8: Rank of max mc-consistency score vs. truncation to nearest k. Sample contains 100 elements.

Effect of truncation and rounding.

Unlike one-bit checking and parameter retrieval, blind checking uses low-order bits to identify a best-guess provenance mark; corruption via truncation or rounding can impede blind checking. Figure 8 shows the effect of truncation of blind checking. All datapoints of an annotated dataset had the n least-significant bits set to zero; blind and directed search performed, with a search bound of $2^{16} = 65,536$. That is, at most 2^{16} provenance marks were considered. The x-axis indicates n, the number of least-significant bits truncated from each datapoint; the y-axis shows how many provenance marks were considered before the provenance mark with the highest mc-consistency score was found (which may not be the correct mark). Ten trials were performed for each possible combination of $R \in \{1, 2, 4, 8\}$ and $0 \le n \le 9$, and the mean is plotted.

The graph indicates that we "fall off a cliff". Blind checking can tolerate some amount of truncation very well, and is able to retrieve the provenance mark without search (rank = 1). However at some point, too many bits have been truncated, and we are unable to recover the provenance mark. Higher redundancy results in more robustness to truncation. The correct provenance mark, if not found at rank 1, was typically not in the first 2^{16} provenance marks considered.

Our experiments indicate that the effect of rounding is similar to that of truncation.

7.2 Perceptibility

Embedding of a provenance mark into a dataset should not affect the scientific use of the dataset. Our embedding technique is clearly detectable: algorithms such as one-bit checking can distinguish annotated datasets from unannotated datasets. However, we alter only the least significant bits of datapoints—those within the noise of the sensor measurements. As such there is little impact on various statistical measures of the dataset and no significant impact on many of the scientific uses of the dataset.

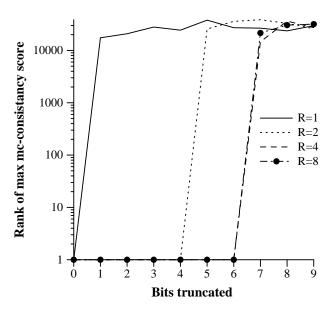For specific descriptive statistics, we can analytically bound the effect of embedding marks into datasets. Let L_{md} be the number of bits of metadata (parameter check bit, mark check bit, and provenance piece) that we are adding to each datapoint.

Mean.

By adding metadata to a datapoint, the value of a datapoint can change by at most $2^{L_{md}} - 1$. Thus, the mean over the dataset changes by at most $2^{L_{md}} - 1$. However, typically the actual change to the mean would be at least an order of magnitude smaller, since the first two bits of the metadata (the parameter and mark check bits) are uniformly distributed. Additionally, if the lower order bits of the original datasets are fairly uniformly distributed over the interval $[0, 2^{L_{md}} - 1]$, then the change in the mean will be close to zero.

Variance.

We assume that the value of the L_{md} least significant bits of an unannotated dataset are not correlated with the value of the more significant bits; this is a reasonable assumption since the error of the sensor measurements are greater than $2^{L_{md}}$.

Let X be the random variable corresponding to the distribution from which the unannotated datapoints are sampled. Let $\epsilon = 2^{L_{md}}$. The following equivalence holds.

$$X = \epsilon \left\lfloor \frac{X}{\epsilon} \right\rfloor + X \bmod \epsilon.$$

Let M be the random variable corresponding to the distribution from which the metadata is taken, which ranges over the interval $[0, \epsilon - 1]$. Note that since the metadata is selected using hash functions, the metadata added to a datapoint is independent of the datapoint.

The annotated dataset is obtained by replacing the last L_{md} bits of each datapoint with metadata. The annotated dataset is modeled by random variable A where

$$A = \epsilon \left\lfloor \frac{X}{\epsilon} \right\rfloor + M.$$

9

The variances of A and X are given by

$$\begin{aligned}
\mathrm{Var}(X) &= \mathrm{Var}\left(\epsilon\left\lfloor\frac{X}{\epsilon}\right\rfloor\right) + \mathrm{Var}(X \bmod \epsilon) \\
&\quad + \mathrm{Cov}\left(\epsilon\left\lfloor\frac{X}{\epsilon}\right\rfloor, X \bmod \epsilon\right) \\
\mathrm{Var}(A) &= \mathrm{Var}\left(\epsilon\left\lfloor\frac{X}{\epsilon}\right\rfloor\right) + \mathrm{Var}(M) + \mathrm{Cov}\left(\epsilon\left\lfloor\frac{X}{\epsilon}\right\rfloor, M\right).
\end{aligned}$$

Distribution $\epsilon\left\lfloor\frac{X}{\epsilon}\right\rfloor$ is independent of $X \bmod \epsilon$ and of M, so both covariance terms are 0. Thus,

$$\mathrm{Var}(X) - \mathrm{Var}(A) = \mathrm{Var}(X \bmod \epsilon) - \mathrm{Var}(M).$$

Both $X \bmod \epsilon$ and M sample $\{0, \ldots, \epsilon - 1\}$, so their variances are at most $(\epsilon - 1)^2/4$ (see [11]). Thus the worst case change in variance is $\mathrm{Var}(X) - \mathrm{Var}(A) = \pm(\epsilon-1)^2/4$. If both X's low order bits and distribution M are uniformly random, then $\mathrm{Var}(X) - \mathrm{Var}(A) = 0$.

8. DEPLOYMENT ISSUES AND FUTURE WORK

In this paper we develop a foundational theory comprising an abstract notion of "dataset," but our scheme is intended for real-world deployment and this raises a variety of issues. Here we discuss salient ones and how they can be approached in future work.

Social issues.

As we have stressed, our scheme is not a security mechanism per se, but rather is intended to (1) support the "fair use" policies typical of publicly available sensor network data, and (2) provide a means to mark data with its own metadata. The importance of both fair use policies and metadata in the environmental sciences community is evidenced by online archives such as the aforementioned HBES [18] and the Sagehen Creek Field Station repositories [15]. Both incorporate policies expecting that data producers *and* archivers should be acknowledged in and informed of publications that use their data. These sites also clearly associate metadata with datasets, which is crucial to contextualize environmental data.

Thus, the tools and techniques we propose should be freely available and "open source," and our provenance encoding need not be irreversible in a cryptographic sense; they are intended to support scientists and promote good citizenship. In particular, we envision publicly accessible web-based tools for embedding and retrieving provenance marks. Indeed, probably the most sensitive aspect of our scheme is that it alters datasets themselves. Data producers often have strong feelings about the integrity of their data and may look askance at manipulation of their data, arguments about imperceptibility notwithstanding. However, in our scheme it is always the case that unmarked data is available to the data producer and can be privately archived.

The meaning of provenance marks.

Intuitively, provenance encoding provides a means to answer the questions "where did this data come from?" and "is this data mine?", given just a dataset. Analogously to typical watermarking schemes, blind checking addresses the former question, while one-bit checking addresses the latter. Which question is more important may determine what meaning is carried by the provenance marks. That is, if checking data ownership is paramount, then it would suffice that each data producer uses a unique provenance mark, and a producer's single mark may be embedded in many datasets. On the other hand, if provenance is the dominant issue, then marks would

more appropriately encode or point to the dataset's metadata. For example, a mark could be a url for a webpage containing extensive provenance information for the marked dataset; a level of indirection allows marks to be shorter than metadata, allowing a more robust encoding.

Embedding and data lifecycles.

A central issue is at what point in the data lifecycle should provenance marks be embedded. In particular, note that raw sensor data is usually obtained as a straight voltage reading or as a ratio to a benchmark voltage. Some formula is then applied to obtain a standard unit measurement based on the calibration of the instrument. Since this transformed reading is typically the data that is actually disseminated, we argue that transformed data in standard units is more appropriate to be marked than raw sensor data. Otherwise, our provenance encoding scheme would have to be robust to arbitrary algebraic transformations, or the retrieval algorithm would need to "know" the precise transformation function, neither of which is realistic.

At what point the embedding occurs is also a function of who intends to mark the data; when data is included as part of a larger repository then both the producer and the archiver may wish to mark it with provenance information. If the latter, the embedding could be performed upon data entry into or retrieval from the repository in a straightforward manner. If the former, it may be desirable to perform the embedding before the data is introduced to the repository. This may require that the embedding be performed within the sensor network itself. Given empirical observations regarding the efficiency of our scheme we believe this is practical. The most computational expensive component of our embedding technique is the computation of a hash; MD5, a commonly-used cryptographic hash function, has been implemented as a MAC algorithm for TinyOS [19]. Indeed, since our scheme is not a security mechanism, we do not require our hash to be hard to invert, and could just as well use a block cipher algorithm instead of a hash; implementations of, for example, both AES and Skipjack are available in TinyOS.

Retrieval: format and efficiency.

We have implemented a retrieval algorithm that takes as input newline-separated lists of numeric values that would be straightforward to provide as a form-based tool on the web. This would allow users to copy-paste lists of data for automated provenance mark retrieval. However, a more realistic option is to develop a simple parsing tool that takes as input data in standard domain-specific formats such as EML.

Another problem related to retrieval tools is larger datasets. Obviously, the larger the dataset, the longer the retrieval process. Very large datasets could pose a significant problem for our technique unmodified. However, we have observed that our technique is robust to sampling in both our combinatorial analysis and empirical observations. In particular, Figure 7 shows that approximately 100 datapoints with 10 bits of metadata provides high reliability of recovery of a 32-bit provenance mark. Thus, one-bit and blind checking could be implemented efficiently for large datasets by sampling a relatively small subset of datapoints.

9. RELATED WORK

The issue of how to represent and manage metadata and provenance in sensor network data repositories has received increasing attention as these repositories grow in size and popularity. Previous related work has considered tracking provenance of data as it is republished [14] and during curation of large-scale, interconnected

data repositories [2], as well as leveraging provenance information to increase the utility of sensor data [12]. Research has resulted in online tools for sensor data storage such as SensorBase [17] which have even introduced the notion of "slogging", or logging of sensor data via systems that automate metadata annotation and support sharing with the broader community. While this previous work reflects the importance of issues we have addressed in our work, it mainly considers management and dissemination of metadata and provenance annotations, rather than introducing means to *directly* associate data with its metadata as in our system.

Our provenance encoding technique can be viewed as a new type of digital watermarking [9, 7]. A vast amount of research exists in this field, with watermarking techniques proposed for a variety of media; most related to our work are techniques for watermarking relational databases [1]. However, relational database watermarking typically relies on schema structure, whereas our technique treats unstructured datasets. The watermarking literature has also considered the use of watermarks to associate data with its provenance information. Previous work has treated this issue in multimedia DBMS [5] and in raw video data [10]. Our work accomplishes similar goals for sensor datasets. However, a key difference with most work on watermarking is that we do not regard the user of the data as an adversary who is attempting to detect or remove the watermark. As such, our technique does not aim to be imperceptible (although we do ensure that embedding does not affect scientific uses of the data), and we do not aim for the metadata to be inalterable. A notable exception is the VEIL system for certifying video provenance [10], which allows an adversary to remove metadata, but aims to make it difficult for an adversary to introduce false metadata, or to alter data while still associating it with the same metadata.

Tracking provenance in curated databases [4] is another related problem of burgeoning interest. Here the issue is keeping track of the provenance of data originating from multiple sources, manually constructed by domain experts. But work in this area treats structured data and focuses on the "history" of data and how it is manipulated to obtain the curated database. Somewhat more related is work on tracking provenance in automatically-generated data warehouses [3, 6, 8, 13, 16, 20]. This body of work is mostly focused on combining provenance annotations on data warehouses in a systematic manner; by contrast, we aim to directly associate provenance information with unstructured data.

10. CONCLUSION

In this paper we have a presented a system for generating *self-identifying data*, which is sensor network data marked with its own metadata in an automatically recoverable manner. Our system is intended to support fair-use policies in the scientific community. Our technique for marking data is similar to previous watermarking techniques for other media, in that we define mark embedding and both one-bit and blind retrieval algorithms that are robust to a number of transformations. Since our system is not intended as a security mechanism per se, the transformations we consider are benign and comprise modifications we expect data users to make in the course of normal study, including sampling, reordering, truncation, and rounding of data. We have performed combinatorial and empirical analysis of our system characterizing its robustness in various scenarios and providing insight into its best use.

Acknowledgments

11. REFERENCES

[1] Rakesh Agrawal and Jerry Kiernan. Watermarking relational databases. In *VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases*, pages 155–166. VLDB Endowment, 2002.

[2] Karen S. Baker and Lynn Yarmey. Data stewardship: Environmental data curation and a web-of-repositories. *The International Journal of Digital Curation*, 4(2), 2009.

[3] Deepavali Bhagwat, Laura Chiticariu, Wang Chiew Tan, and Gaurav Vijayvargiya. An annotation management system for relational databases. *VLDB J.*, 14(4):373–396, 2005.

[4] Peter Buneman, Adriane Chapman, and James Cheney. Provenance management in curated databases. In *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 539–550, New York, NY, USA, 2006. ACM.

[5] Richard Chbeir and David Gross-Amblard. Multimedia and metadata watermarking driven by application constraints. In *MMM*. IEEE, 2006.

[6] Wang chiew Tan. Containment of relational queries with annotation propagation. In *In Proceedings of the International Workshop on Database and Programming Languages (DBPL*, pages 37–53, 2003.

[7] Ingemar J. Cox and Matt L. Miller. The first 50 years of electronic watermarking. *EURASIP J. Appl. Signal Process.*, 2002(2):126–132, 2002.

[8] Yingwei Cui and Jennifer Widom. Lineage tracing for general data warehouse transformations. In *27th International Conference on Very Large Data Bases (VLDB 2001)*, 2001.

[9] Jessica Fridrich and Miroslav Goljan. Comparing robustness of watermarking techniques. In *Security and Watermarking of Multimedia Contents*, volume 3657 of *Proceedings of Spie–the International Society for Optical Engineering*, 1999.

[10] Ashish Gehani and Ulf Lindqvist. Veil: A system for certifying video provenance. In *ISM '07: Proceedings of the Ninth IEEE International Symposium on Multimedia*, pages 263–272, Washington, DC, USA, 2007. IEEE Computer Society.

[11] Harold I. Jacobson. The maximum variance of restricted unimodal distributions. *Ann. Math. Statist.*, 40(5):1746–1752, 1969.

[12] Jonathan Ledlie, Chaki Ng, David A. Holland, Kiran-Kumar Muniswamy-Reddy, Uri Braun, and Margo Seltzer. Provenance-aware sensor data storage. In *Proceedings of the 1st IEEE International Workshop on Networking Meets Databases*, 2005.

[13] Thomas Lee, Stéphane Bressan, and Stuart E. Madnick. Source attribution for querying against semi-structured documents. In Sadri [16], pages 33–39.

[14] Unkyu Park and John Heidemann. Provenance in sensornet republishing. In *Proceedings of the 2nd International Provenance and Annotation Workshop*, pages 208–292, Salt Lake City, Utah, USA, June 2008. Springer-Verlag.

[15] Sagehen Creek Field Station Data Repository. `http://sagehen.ucnrs.org/resources.htm`. Last visited 10/29/09.

[16] Fereidoon Sadri, editor. *CIKM'98 First Workshop on Web Information and Data Management (WIDM'98), Bathesda, Maryland, USA, November 6, 1998*. ACM, 1998.

[17] SensorBase. `http://sensorbase.org/`. Last visited 10/29/09.

[18] Hubbard Brook Ecosystem Study. `http://www.hubbardbrook.org/`. Last visited 10/29/09.

[19] UbiSec&Sens Hmac-MD5 Implementation. `http://www.ist-ubisecsens.org/downloads/hmac-md5/hmac-md5.php`. Last visited 10/29/09.

[20] Jennifer Widom. Trio: A system for integrated management of data, acuray and lineage. In *Proc. of the International Conference of Data Systems Research (CIDR)*, 2005.