Self-Identifying Data for Fair Use

Stephen Chong
Christian Skalka
and
Jeffrey A. Vaughan

TR-01-14

Computer Science Group
Harvard University
Cambridge, Massachusetts

Self-Identifying Data for Fair Use

Stephen Chong
Harvard University

Christian Skalka
The University of Vermont

Jeffrey A. Vaughan
LogicBlox, Inc.

Abstract

Public-use earth science datasets are a useful resource with the unfortunate feature that their provenance is easily disconnected from their content. "Fair-use policies" typically associated with these datasets require appropriate attribution of providers by users, but sound and complete attribution is difficult if provenance information is lost. 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 time-series 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, and show how our system can be adapted to various data representation types. Our algorithms are probabilistic in nature and are characterized by both combinatorial and empirical analyses. Mark embedding can be applied at any point in the data lifecycle, allowing adaptation of our scheme to social or scientific concerns.

1 Introduction

Open sharing of datasets is beneficial for scientific research, but is often complicated by issues related to attribution, i.e. who should be acknowledged, and how, if a particular dataset is used? Two major problems present themselves: *how* to associate provenance metadata with datasets, and *what* is the nature of metadata. In environmental sciences datasets in particular, dataset usage often brings with it an informal obligation to attribute organizations that manage areas from which data is collected, e.g. field stations, as well as collectors of data, e.g. University research groups. These are typically called "fair use policies", and different datasets are subject to different, possibly complex policies. And although standards such as eml [2] have been proposed, different repositories use different techniques to associate data and metadata such as provenance information. Even if researchers are well-intentioned, metadata and datasets can become disassociated during usage, and provenance information lost, leading to incomplete or inaccurate attributions in publications and other research products.

In this paper, we explore a scheme for embedding a *provenance identifier* in environmental datasets, that associates metadata with datasets in a tightly coupled manner that does not rely on external structure such as xml formats or database schema. We say that such datasets are "self identifying". Further, this identifier provides a level of indirection allowing referenced metadata to be structured and organized in an arbitrary manner. Our technique is suited to environmental datasets because it exploits noise common to instruments used in environmental sciences data gathering, in particular, margin of error in instrumentation. It can be viewed as a type of watermarking scheme, where the watermark does not alter data beyond the bounds of its error bars, and is thus "imperceptible" by data end-users, but is nevertheless retrievable by algorithmic means. And while these provenance identifiers are not impervious to attack as would be a security mechanism, they are robust to corruptions—including reordering, sampling, truncation, and rounding—that can be expected during the course of normal dataset analysis by end users. Thus, our approach is well-suited to supporting attribution and fair use policies in environmental datasets.

1.1 A Proposed Application Setting

The Sagehen Creek Field Station (CA) provides both a setting for environmental field research and data gathering, and an online data repository for associated datasets. On its data repository web portal [19], a blanket fair use policy is stated requiring acknowledgment of the field station and data owners, and a request to contact data owners to discuss attribution details. Similarly, the Hubbard Brook Experimental Forest (NH) also provides both a research setting and online repository [13]. In this case, specific fair use policies are associated with individual datasets (although a general default policy appears to cover most), and users must electronically sign a fair use agreement prior to downloading data. In any case it is clear that use policies are sufficiently complex that repositories must clearly enunciate them, and encourage compliance with them. And in fact, although research scientists are typically well-intentioned, anecdotal evidence from station managers suggests that fair use policies are often violated.

We argue that these violations are often caused by the decoupling of datasets and their provenance information during normal scientific usage. Once datasets are downloaded, they are often reformatted, entered into local databases with arbitrary schema, and passed around between users. Policies that were reviewed when data was downloaded may be forgotten or not communicated between users, and data structure imposed in xml or similar forms may be discarded once datasets are extracted from them, and metadata lost. We envision that our scheme can be used to mark datasets intended for public use, for example by data repository managers. This mark, when recovered from datasets, can be interpreted as a pointer to online provenance information and fair use policies. Thus, any end-user, or reader of their work, could copy-paste a subset of their dataset into a recovery tool and be immediately directed to relevant provenance information for the dataset. The scheme therefore provides much greater ease of access to provenance information, and user accountability for proper attribution.

It is important to note that the addition of a provenance mark does alter datasets, however imperceptibly. But an appealing feature of our scheme is that a provenance mark can be applied at any point in the data lifecycle. Thus, marking strategies can be flexible, and unmarked, raw datasets can be retained by data producers, or made available on repositories along with marked datasets. This will allow data providers to negotiate marking strategies with repository managers, and determine the best tradeoff between ease of attribution and fidelity to original data. In most cases, we expect that marks will be applied post-processing, e.g. after raw sensor voltage readings are converted to physical units and smoothed.

To provide a user experience that actively illustrates these ideas, we have developed a prototype application, described in Section 8 and available online. It shows how data providers can embed provenance marks in datasets without significant alterations to existing web interfaces or data distribution schemes. It also illustrates how users of datasets can easily retrieve associated metadata from a sampling of datapoints. Furthermore, the application shows how data can be marked post-processing, and that marking does not preclude access to unmarked datasets.

1.2 Technical 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. In the digital realm, it has 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 encoded provenance information.

Watermarking of digital media is typically treated as a signal processing issue [9]. Watermarking techniques have also been studied in application to relational databases [1] and streaming data [21]. Usually these approaches leverage the database schema, or the sequential structure of a data stream, in order to embed watermarks in the data itself.

Our main contribution is a new watermarking technique for environmental sciences data. Similar to

database watermarking it is combinatorial in nature, and allows typical datasets to be marked in a manner that does not interfere with normal scientific use. But unlike database watermarking, the technique makes minimal assumptions about the structure of datasets and is compatible with workflows that do not, for example, support rich schemas. While it is not a security mechanism per se, it is robust to corruptions including reordering, sampling, truncation, and rounding, which we argue is sufficient to support fair use policies in the scientific user community.

We have evaluated our system from both empirical and analytic perspectives. Using a prototype implementation of our system and a simulation of watermarked dataset corruptions, we have characterized the robustness of our approach in practice. And, a combinatorial analysis yields an upper bound probability for recovering a watermark from a dataset with our technique. Results obtained from these exercises demonstrate its flexibility and resilience. Further underscoring the practical applicability of our approach, we formulate a general method for adapting it to a variety of data representation types, e.g. integer values, floating point values, and geographic datapoints.

To illustrate how our system can be used in practice to support fair use policies, we have applied our method to encode a 32-bit provenance mark in an existing public use dataset, available online. We have also made available online an interface for extracting 32-bit provenance marks and reporting provenance information associated with marked datasets. URLs for these tools, and further discussion of them, are provided in Section 8.

1.3 Characterization of the Technique

As described above, our goal is to design a system that can embed provenance information into, and retrieve such information from, scientific datasets. Furthermore our techniques are intended to be robust against certain natural, non-adversarial dataset corruptions. 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* [11].

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 environmental datasets, we propose a definition of perceptibility in this context, under the assumption that any dataset is a collection of numeric values, with possibly significant ordering. To guide intuition, we (informally) assume that individual data points have a precision that is coarser than the precision of the datatypes used to represent the numeric values. We say that a watermark is imperceptible in a 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 datasets. Indeed, our algorithm for the latter relies on the former for increased probability of correct mark extraction.

1.4 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 corruptions. Our basic theory treats datasets as sequences of bit vectors representing natural numbers; in Section 6 we formulate a methodology for extending our approach to various datatypes such as integers, doubles, and GPS coordinates. In Section 7 we report results of combinatorial analysis and empirical tests that illustrate robustness of our technique. We have developed a prototype implementation of our technique, which is deployed as a web application, allowing consumers of self-identifying datasets to easily check or discover provenance information. We describe the implementation in Section 8. In Section 9 we summarize deployment issues and how we might address them in future work. We discuss related work in Section 10, and conclude in Section 11.

This paper substantially revises earlier work by same authors. [8] The present paper contains an greatly expanded analysis of non-integral data including floating point and GPS values, describes the implementation of a new prototype web application that decodes concise provenance identifiers from datasets and associates those identifiers with descriptive provenance text, discusses the application of this tool to an environmental dataset, reports on additional experimental results with synthetic data, and provides an appendix with an explicit derivation of Section 7's combinatorial results.

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* $\mathcal{DS} \in \mathit{list}(\mathbb{V})$ is a list of bit vectors; each bit vector is a *datapoint*. A *transformation* is a function of type $\mathit{list}(\mathbb{V}) \to \mathit{list}(\mathbb{V})$; transformation f is a *corruption* if $f(\mathcal{DS}) \neq \mathcal{DS}$. An *encoding* E is a function of type $\mathit{list}(\mathbb{V}) \times \mathbb{V} \to \mathit{list}(\mathbb{V})$ with the property that there exists a corresponding *retrieval function* R of type $\mathit{list}(\mathbb{V}) \to \mathbb{V}$ 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. Retrieval of v corresponds to *blind* watermarking, defined in the Introduction: provenance information v can be retrieved from the transformed dataset $E(\mathcal{DS}, \mathsf{v})$. *One-bit* watermarking is supported if there exists a *checking function* C of type $\mathit{list}(\mathbb{V}) \times \mathbb{V} \to \mathbb{B}$ such that for all $\mathcal{DS}$ and $\mathsf{v}, \mathsf{v}' \in \mathbb{V}$:

$$C(E(\mathcal{DS}, \mathsf{v}), \mathsf{v}') = \begin{cases} \text{True} & \text{if } \mathsf{v} = \mathsf{v}' \\ \text{False} & \text{if } \mathsf{v} \neq \mathsf{v}' \end{cases}$$

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 $\mathsf{v} \in \mathbb{V}$. Similarly, an encoding-checking function pair (E, C) is *robust* to a corruption f iff:

$$C(f(E(\mathcal{DS}, \mathsf{v})), \mathsf{v}') = \begin{cases} \text{True} & \text{if } \mathsf{v} = \mathsf{v}' \\ \text{False} & \text{if } \mathsf{v} \neq \mathsf{v}' \end{cases}$$

for all $\mathcal{DS}$ and $\mathsf{v}, \mathsf{v}' \in \mathbb{V}$. We say that encoding-retrieval pair (E, R), or encoding-checking pair$(E, C$ is robust to a set of corruptions C by generalizing f in the above definitions to range over all functions in C.

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) and an encoding-checking pair (E, C) that are 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 $\mathsf{v} \mathbin{@} \mathsf{v}'$ 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:

- *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]
- *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 a provenance mark 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

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

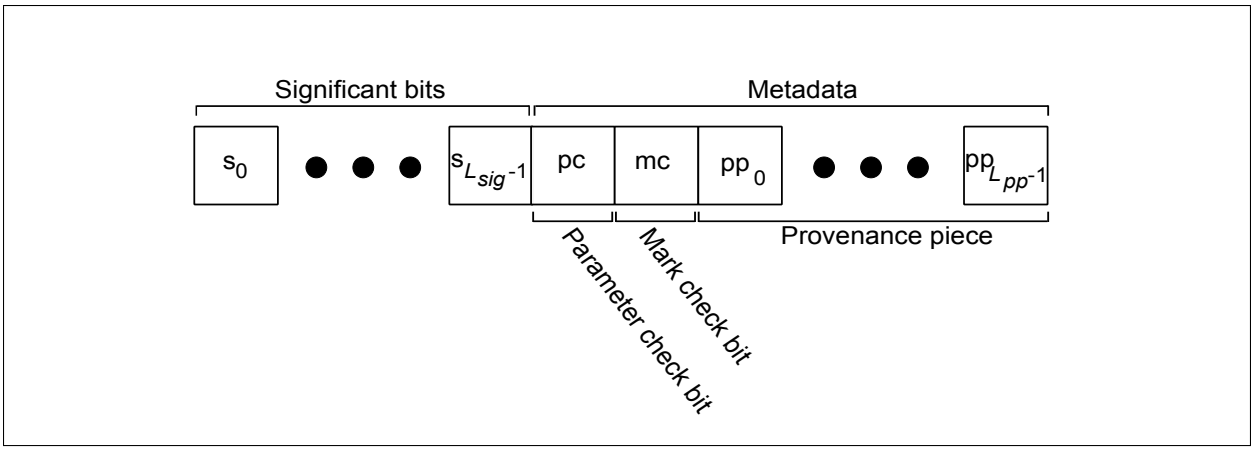

Figure 1: Anatomy of an Annotated Datapoint

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 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.

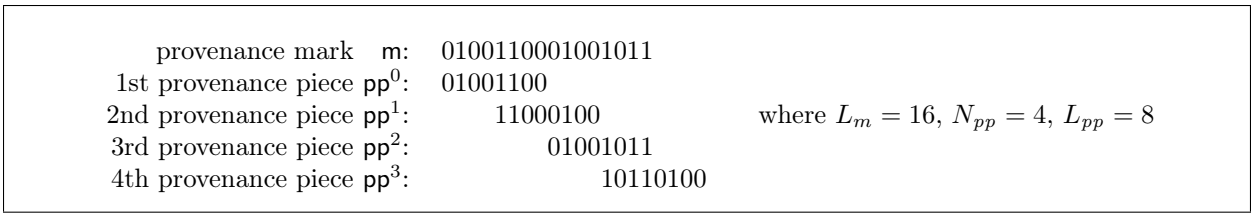

Figure 2: Provenance pieces example.

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 $k = \text{hash}(N_{pp}, \mathsf{s})$ and $\text{hash}(n, \mathsf{v})$ is a cryptographic hash of bit vector v that returns a value in $\mathbb{Z}_n$. Provided 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:

$$\mathsf{pc} = \text{hash}(2, \mathsf{s} \mathbin{@} N_{pp}) \quad (1)$$

$$\mathsf{mc} = \text{hash}(2, \mathsf{s} \mathbin{@} \mathsf{m}). \quad (2)$$

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.

A corrupted annotated datapoint d is defined to be *pc-consistent* with guesses L_{sig} and N_{pp} iff

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

Note that if the guesses for L_{sig} and N_{pp} are correct, and bits $\mathsf{d}_0 \ldots \mathsf{d}_{L_{sig}}$ 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 $\mathsf{d}_0 \ldots \mathsf{d}_{L_{sig}}$ 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 ($\{1 \ldots \max\{|\mathsf{d}| \mid \mathsf{d} \in \mathcal{DS}\}\}$ and $\{1 \ldots L_m\}$ 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 $\mathsf{d} \in \mathcal{DS}$, we say that d is *mc-consistent* with L_{sig} and m iff

$$\text{hash}(2, \mathsf{d}_0 \ldots \mathsf{d}_{L_{sig}-1} @ \mathsf{m}) = \mathsf{d}_{L_{sig}+1}.$$

Note that if the guesses for L_{sig} and m are correct, and the bits $\mathsf{d}_0 \ldots \mathsf{d}_{L_{sig}+1}$ have not been corrupted, then $\mathsf{d}_{L_{sig}+1}$ 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 $split(\mathsf{d}) = (\mathsf{s}, \mathsf{pc}, \mathsf{mc}, \mathsf{pp}, k)$ where

- $\mathsf{s} = \mathsf{d}_0 \ldots \mathsf{d}_{L_{sig}-1}$; and
- $\mathsf{pc} = \mathsf{d}_{L_{sig}}$ and
- $\mathsf{mc} = \mathsf{d}_{L_{sig}+1}$; and
- $\mathsf{pp} = \mathsf{d}_{L_{sig}+2} \ldots \mathsf{d}_{|\mathsf{d}|-1}$; and
- $k = \text{hash}(N_{pp}, \mathsf{s})$.

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, $\mathit{suggest}(\mathsf{d}, i)$ 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:

$$\mathit{suggest}(\mathsf{d}, i) = \begin{cases} (\mathsf{pp}_j, j) & \text{if } \; j + k \bmod L_m = i \;\text{ and }\; 0 \leq j < |\mathsf{pp}| \\ * & \text{otherwise} \end{cases}$$

where $(\mathsf{s}, \mathsf{pc}, \mathsf{mc}, \mathsf{pp}, k) = \mathit{split}(\mathsf{d})$.

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 $\mathit{suggest}(\mathsf{d}, i) = *$ for $0 \leq i \leq 7$ (since pp^2 contains no information about the first 8 bits of the provenance mark), and $\mathit{suggest}(\mathsf{d}, 10) = (\mathsf{pp}^2_2, 2) = (0, 2)$.

We lift the definition of $\mathit{suggest}(\mathsf{d}, i)$ from datapoints to datasets: $\mathit{suggest}(\mathcal{DS}, i)$ 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 $\mathit{suggest}(\mathsf{d}, i) = *$). Formally, we define $\mathit{suggest}(\mathcal{DS}, i)$ as follows.

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

Given $\mathit{suggest}(\mathcal{DS}, i)$, 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 round$(\cdot)$ rounds real numbers to the nearest integer):

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

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.

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

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.

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

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

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

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

Figure 3: Directed search algorithm.

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.

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 $\frac{\sum_{(\mathsf{b},c)\in L}\mathsf{b}}{|L|}$ 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*$(10, \mathsf{m})$ only the best 2^{10} possible provenance marks will be considered, equivalent to $sb = 1024$.

Section 7 discusses the robustness of the retrieval process, and the efficacy of directed search.

6 Extension to Various Datatypes and Low-Entropy Datasets

Our presentation has focused on embedding and recovery techniques for watermarking collections of bit vectors, with an underlying assumption that lower order bits correspond to less significant data. However, in practice, datasets needing marking may not conform to these assumptions. Here we consider extensions of our framework that can accommodate datasets that may be encountered in the wild.

One precondition for our encoding is a certain level of entropy in data, since significant bits in datapoints serve as provenance piece indices. Insufficient entropy will result in an insufficient distribution of provenance pieces in the dataset. However, we note that surprisingly little entropy is needed to support statistically reliable recovery of a mark; as demonstrated in Section 7 our encoding is reasonably robust for datasets with as few as 50–100 unique elements. Datasets with fewer unique values are uncommon and when they do arise, it is typically when measuring small, discrete quantities. These are a poor fit for our scheme, since encoding in the low order bits of an integral data representation would result in unacceptably large changes in value. On the other hand, increasing entropy in a dataset is always possible, either by randomly manipulating low

order bits in datapoints, or by padding real-numbered datapoints with additional, randomized lower order bits. However, this sort of manipulation is so closely tied to the needs and tolerances of data users that we consider such strategies highly application specific and thus out of scope for this presentation.

Real datasets will also employ different representation types of data. For example, geographic coordinates are common in environmental datasets, and are fundamentally different from integral values in their representation. This matter is more readily considered here, since in particular we can develop a general terminology and method for dealing with different data types. Our approach is to introduce a notion of data preserving, bidirectional transformation between representation types and bit vectors. This allows us to retain the previously described method for directly embedding and recovering marks from sets of bit vectors, and providing a technical framework in which to evaluate the effect of applying provenance marks.

We start by formalizing our provenance mark encoding functionality in pointwise manner. For the issue under consideration, it is acceptable to fix the parameters of the encoding.

Definition 6.1 *Assume given a type* bitvec *populated by bit vectors of fixed length L. Let PM be our provenance mark embedding function defined on single bit vectors with fixed L_{pp}, L_{sig}, and provenance mark m : bitvec. Hence:*

$$PM : \text{bitvec} \to \text{bitvec}$$

An important property of the encoding is that it only manipulates low-order bits, which we characterize as follows.

Proposition 6.1 *Let $\mathit{preservedBits}(PM) = \{\, i \mid (PM(b))[i] = b[i]$ for all b : bitvec$\}$. Then $\mathit{preservedBits}(PM) \supseteq \{0 \ldots L_{sig} - 1\}$.*

Now, we introduce a general notion of representation type T, whose members are data points. The following will show how to effectively mark elements of a type T using a provenance mark embedding function PM. As we will see, it is sufficient for T to be endowed with an equivalence relation and a distance function. Distance is expressed as a real number, and allows us to measure the perturbation of datapoints under an embedding.

Definition 6.2 *We let T range over arbitrary* representation types *which must admit an equivalence relation $=_T$ and a distance function:*

$$\ominus : T \times T \to \text{real} \qquad \textit{symmetric distance function}$$

Now we formally characterize the transformation between representation types and bit vectors, which must be data preserving.

Definition 6.3 *Given fixed provenance mark function PM and a representation type T, an* encoding for T*, denoted e, is a pair of functions $\mathit{vectorize}_T$ and $\mathit{devectorize}_T$ such that:*

$$\mathit{vectorize}_T : T \to \text{bitvec} \qquad\qquad \mathit{devectorize}_T : \text{bitvec} \to T$$

and these must be inverses, i.e. for any $p : T$ it must be the case that:

$$\mathit{devectorize}_T(\mathit{vectorize}_T(p)) =_T p$$

We will write $e(p)$ to denote $\mathit{devectorize}_T(PM(\mathit{vectorize}_T(p)))$ for $p : T$.

Ultimately, we would like to objectively evaluate encoding schemes to determine if they are suitable for specific applications. Intuitively, this is measured in terms of the resulting perturbation thresholds of datasets—less is better. In general, we expect desirable thresholds to be within the margin of error of a given dataset.

Definition 6.4 *An encoding e* respects a threshold n in T *iff $p \ominus_T e(p) \leq n$ for all $p : T$.*

Now, some examples. We consider likely common representation types: two's complement integers, fixed-point numbers, IEEE floating points, and geographic coordinates. For both two's complement and IEEE floating point values, transformation is trivial since both representation types are already bit vectors with less significant bits in lower order positions.

Example 6.1 (Two's complement encoding) *Take T to be a two's complement bit vector representation of integer values, with maximal bit vector length n; we call this type $2s$. We define $vectorize_{2s}$ and $devectorize_{2s}$ to be the identity function, and $=_{2s}$ and $\ominus_{2s}$ to be ordinary two's complement equality and absolute difference respectively, with results interpreted as real numbers. Suppose that the provenance mark embedding function PM manipulates the L_{md} low-order bits. Then encoding e respects a threshold $2^{L_{md}} - 1$, since $p \ominus_{2s} e(p) \leq 2^{L_{md}} - 1$ for all two's complement numbers p.*

Example 6.2 (Fixed-point encoding) *Take T to be the set of real numbers that, when represented in base-2, have at most k bits after the radix point. Intuitively, by multiplying each datapoint by 2^k, we obtain an integer value, and can use the two's complement encoding defined above. Specifically, we can define $vectorize_T$ and $devectorize_T$ as follows.*

$$vectorize_T(p) = vectorize_{2s}(p \times 2^k)$$
$$devectorize_T(v) = devectorize_{2s}(v) \div 2^k$$

If we suppose that the provenance mark embedding function PM manipulates the L_{md} low-order bits then encoding e respects a threshold $2^{L_{md}-k} - 2^{-k}$, which follows immediately for the threshold for the two's complement encoding.

The fixed-point encoding works well if all datapoints in the dataset have approximately the same magnitude. However, if the datapoints can vary in magnitude, then a fixed-point encoding is typically not suitable. This is because the noise added to datapoints to encode provenance information is of the same magnitude for all datapoints, and thus the relative change needed to encode provenance information may be vastly different for different datapoints. This issue can be avoided by using a floating-point encoding to encode numbers of varying magnitude.

Example 6.3 (IEEE floating point encoding) *Take T to be the set of real numbers encodable as IEEE binary64 "double precision" floating point numbers; we call this type* double*. We define $vectorize_{\text{double}}$ and $devectorize_{\text{double}}$ to be the standard IEEE binary64 encoding and interpretation. Furthermore, we take $=_{\text{double}}$ and $\ominus_{\text{double}}$ to be ordinary IEEE binary64 equality and absolute difference respectively, with results interpreted as real numbers.*

In the standard IEEE binary64 encoding, 64 bits are used to represent numbers, with the left-most bit indicating sign s (either 1 or 0), the next 11 bits indicating the exponent e (covering all integer values between -1023 and 1023 inclusive) and the remaining 52 bits indicating the mantissa man. Thus, the number represented is $(-1)^s \times man \times 2^e$ (with a few exceptions for encoding ± 0, $\pm\infty$, and not-a-numbers).

Restricting attention to a well-behaved subset of T yields stronger properties. Let $T' \subseteq T$ be the set of non-zero, normal *numbers. This is all positive and negative doubles with exponent e greater than -1023 and excludes smaller* subnormal *numbers, ± 0, $\pm\infty$, and not-a-number codes. Call the corresponding representation type* normdouble*. Here we take $a \ominus_{\text{normdouble}} b = \max(a/b, b/a)$. Assuming $L_{sig} > 12$ (i.e., L_{sig} encompasses the sign bit and all exponent bits), applying a provenance mark does not change a number's sign or exponents bits, and we can calculate that that this encoding respects threshold $1 + 2^{13-L_{sig}}$ in T'.*

More interesting is the example of GPS coordinates, which are typically represented as pairs of floating point numbers. Perturbation distance here is measured as the geographic distance between pre- and post-embedding datapoints. Noting that each component of the coordinate pair is equally significant, a successful strategy is to *interleave* the coordinate pair in vectorization.

Example 6.4 (GPS coordinates) *Take T to be pairs of normal real numbers, representing GPS coordinates in WGS84 format up to 64 bit (double) precision; we call this type GPS. We define $=_{GPS}$ to be ordinary equality on pairs, and define $\ominus_{GPS}$ as the physical distance between two points on the Earth, calculated using the haversine formula. The result of vectorize$_{GPS}$ is a 128 bit vector obtained by* interleaving *the elements of each coordinate pair represented as binary64 values. That is, the least significant bits of the latitude and longitude elements becomes the 0th and 1st bits in the bit vector respectively, the second least significant bits the 2nd and 3rd elements, etc. We just take devectorize$_{GPS}$ to be the inverse of this. Modifying the real-number bounds to account for the fact the latitude and longitude each have half as many bits flipped as a single real number, allows us to compute that this encoding satisfies threshold*

$$R \arcsin \sqrt{\sin^2 \frac{k\pi}{2} + \sin^2 2k\pi}$$

where R is the radius of the Earth and $k = 2^{(13-L_{sig}/2)}$. Note that, assuming a constant amount of metadata, L_{sig} will be more than twice a large for a GPS coordinate—a pair of doubles—as for a single double. As a point of references, if we have 12 bits of metadata then $L_{sig} = 128 - 12 = 116$ and the encoding respects a threshold of 2.9×10^{-7} meters, or 0.29 micrometers.

The essential characteristic of the above transformations from representation types to bit vectors is that less significant data in the representation type end up as less significant bits in the resulting bit vector. Since our encoding modifies the less significant bits of bit vectors, this scheme tends to modify less significant data in the representation type. We therefore propose the following property as a heuristic for creating encoding schemes that are likely to respect low thresholds.

Property 6.1 *Given an encoding $e = (\mathit{vectorize}_T, \mathit{devectorize}_T)$ for a representation type T, we say that e is* significant-bit order preserving *iff for all b : bitvec we have:*

$$\begin{array}{l} j < L_{sig} \text{ and } j < k \implies \\ \quad \mathit{devectorize}_T(\mathit{flip}\ j\ b) \ominus \mathit{devectorize}_T(b) \leq \\ \quad \mathit{devectorize}_T(\mathit{flip}\ k\ b) \ominus \mathit{devectorize}_T(b) \end{array}$$

where flip i b logically negates b's ith bit.

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

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.

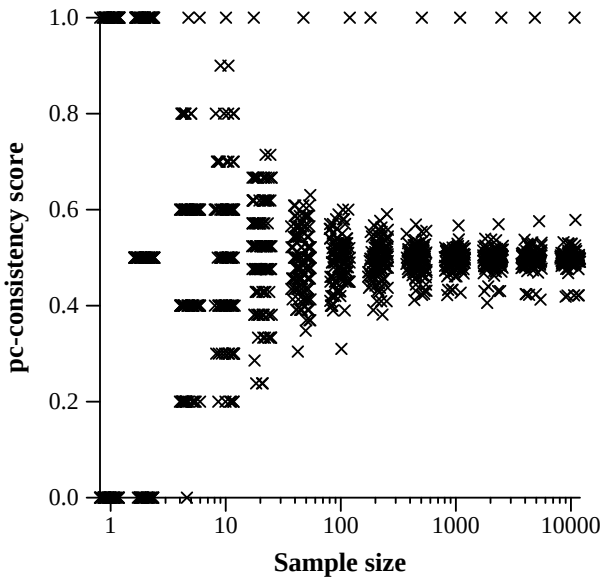

Figure 4: pc-consistency scores vs. sample size. Each thick column represents pc-consistency scores from a single run of the parameter recovery algorithm.

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.

Corruption via sampling. Figure 4 shows 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 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}.

Figure 5 shows the proportion of provenance marks recovered when computed from a sampled data set. Consistent with Figure 4, a high likelihood of successful provenance mark recovery was observed for samples with about 50 elements.

Effect of low-entropy data. We consider a dataset with many identical values to be a low-entropy dataset.

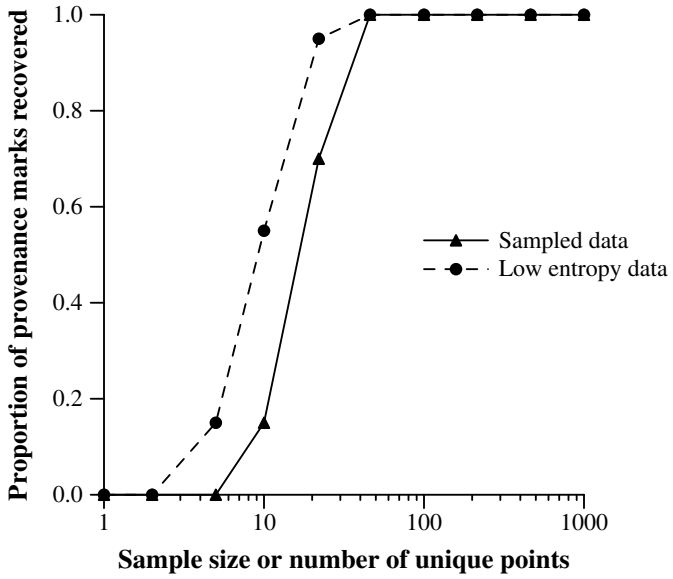

Figure 5: Fraction of provenance marks recovered from a data set vs. number of unique datapoints. The solid series describes data sets with k elements and the dashed series describes data sets with 1,000 elements, k-many of which are unique. Prior to decoding, low-order bits are flipped with low probability. The difference between sampled and low-entropy behavior appears to vanish if the encoded data is not corrupted.

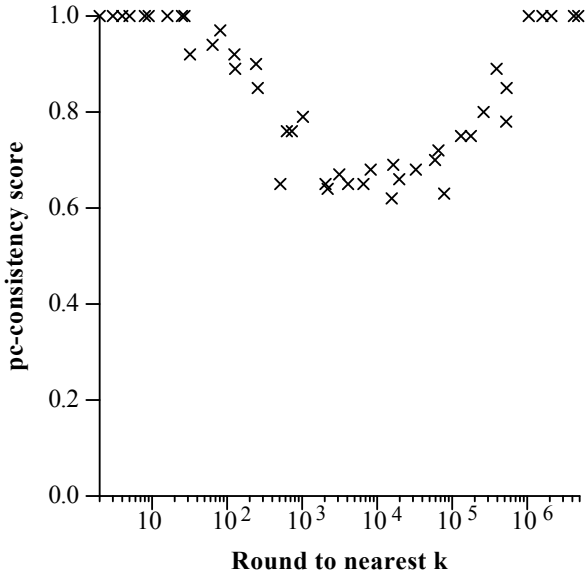

Figure 6: Max pc-consistency scores vs. degree of rounding.

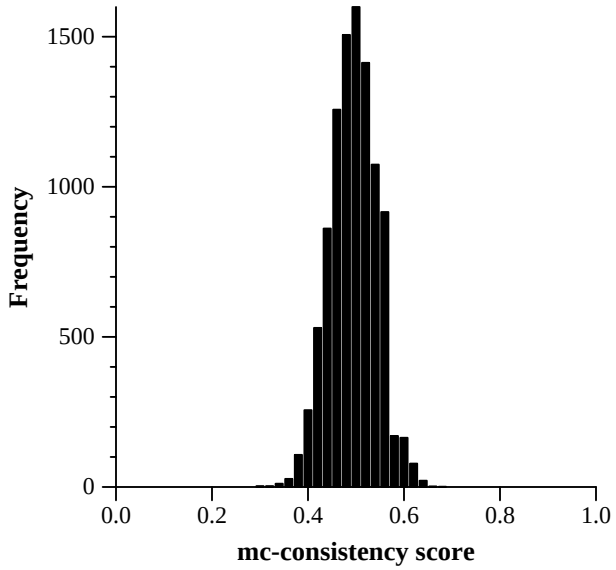

Figure 7: mc-consistency scores of incorrect provenance marks.

Such a dataset may occur when, for example, a sensor measures a slowly changing quantity. Provence mark recovery for a data set with many samples but only k unique elements behaves similarly to recovery for a data set with k-many elements total, all of which are unique. Figure 5 shows that a low entropy data set can be more resilient to random corruption than a small sample of a data set. A point is corrupted by flipping each of its 4 least significant bits with probability 0.2.

Corruption via rounding. Figure 6 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 has 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.

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

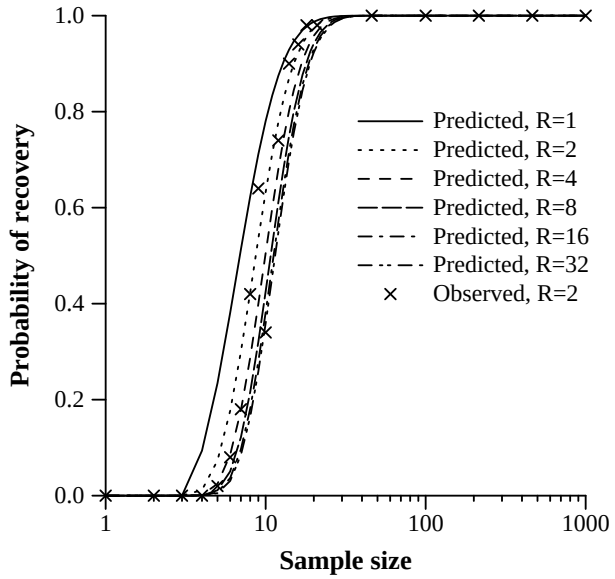

Figure 8: Predicted and observed probability of recovering provenance mark.

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 7 demonstrates this with a histogram of mc-consistency scores for about 10,000 incorrect provenance marks, in a dataset of 100 datapoints.

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 an (uncorrupted) annotated dataset of size $|D|$, with a given redundancy R, can be calculated analytically. Each datapoint of the annotated dataset contains one of the N_{pp} distinct provenance pieces, chosen uniformly at random by the encoding mechanism. Thus, there are $(N_{pp})^{|D|}$ equally probable ways of choosing (with replacement) $|D|$ provenance pieces from the N_{pp} distinct provenance pieces. Given $|D|$ provenance pieces, the provenance mark can be recovered if every bit of the provenance mark occurs in at least one of the $|D|$ provenance pieces. We can count the number of ways that $|D|$ provenance pieces can be chosen such that the provenance mark can be recovered. The combinatorial argument is described in detail in Appendix A. Figure 8 presents the predicted probability of

recovering a provenance mark from annotated datasets of various size (with $N_{pp} = 8$). 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 8 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 8 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.

Corruption via 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.

Truncation may occur in one of two ways. Under *zeroing* truncation, low bits of a datapoint are overwritten with zeros, losing precision but maintaining the datapoint's order of magnitude. For instance applying three bits of zeroing truncation to 53 (binary 110101) yields 48 (binary 110000). In contrast *shifting* truncation simply removes the truncated bits, reducing values by factors of two. For instance, three bits of shifting truncation takes 53 to 6 (binary 110).

Figure 9 shows the effect of zeroing 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 $0 \leq n \leq 9$, and the mean is plotted.

The graph indicates that we "fall off a cliff". Blind checking can tolerate some amount of zeroing 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.

Comparing Figures 9 and 10 indicates zeroing and shifting truncation have similar effects. Furthermore, our experiments indicate that the effects of rounding are also similar.

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 measurement. 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

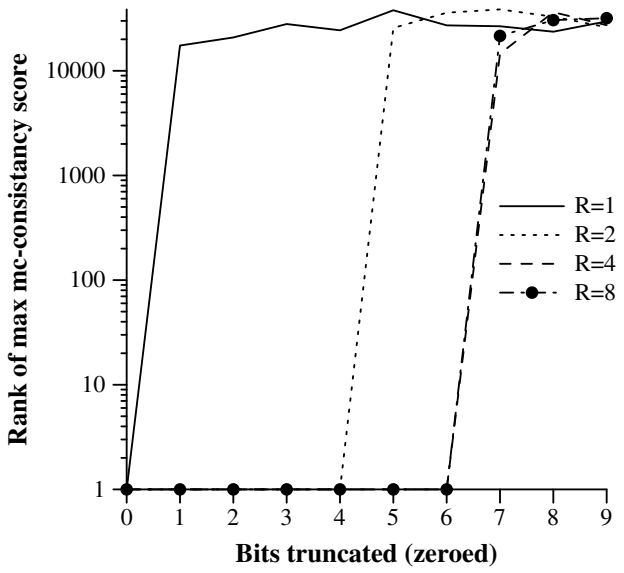

Figure 9: Rank of max mc-consistency score vs. zeroing truncation to nearest k. Sample contains 100 elements.

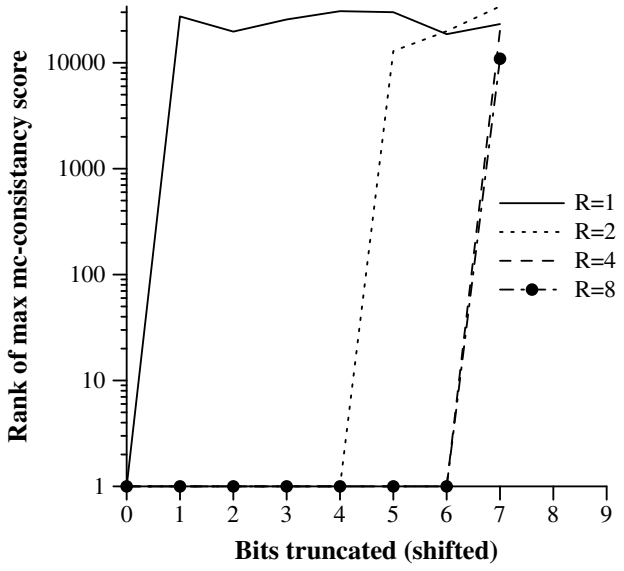

Figure 10: Rank of max mc-consistency score vs. shifting truncation to nearest k. Sample contains 100 elements. Recovery fails completely when points are shifted by $L_{pp}(=8)$ bits.

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 errors in measurement 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

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

The variances of A and X are given by

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

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 Jacobson [14]). 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 The Technique in Practice

In practice, we envision that provenance information will be encoded in data at its source, and online or offline tools will be available to retrieve provenance information from encoded datasets. Since the encoding and decoding algorithms are completely described in this paper, it is straightforward to provide encode and decode utilities. To illustrate and explore possible implementations, we have made a prototype decode utility available online at `http://tinyurl.com/ar6498f`. This utility requires that input data is in tab-delimited multi-column format. Encoded data may be either integer or floating point numbers. The utility will decode any provenance mark in the dataset using techniques described in Section 4, Section 5, and Section 6, and then report either that no known mark was successfully retrieved, or report the provenance information associated with the mark.

Our implementation assumes that provenance marks are 32-bit integer identifiers and maintains a database associating those identifiers with provenance information. The decode utility retrieves associated provenance information from the database, and reports it to the user via a webpage redirect. Note the implicit semantics of provenance marks in this implementation, i.e. as database lookup keys. Another appealing option is to

interpret provenance marks as URLs directly. However, even URL minimization services such as Bitly and TinyURL have more than 32 bits of entropy in their addresses, and it appears that 64 bit marks would be necessary to interpret marks as (e.g.) Bitly url references. Our core encoding and decoding tools support such longer provenance marks, and updating the web frontend to would be an an easy extension to our system.

Experience with an encoded dataset. We have encoded a 32-bit provenance mark in a publicly available dataset. This is a time-series dataset of snow depth readings from a deployment adjacent to route 395 near Mammoth Lakes, CA. These readings were taken at hourly intervals during the Winter of 2012. The data is available via an interface available online at: `http://tinyurl.com/ckbavcf`. To access the encoded data, choose the "encoded data" panel, and select Snow Depth data from any subset of Towers 1, 2, and 3 specifying start and end times during the period 1/11/12-4/30/12. The Table output format should be selected, which will enumerate data in a tab delimited multicolumn format. An adequate selection of data (50 datapoints or so) can be copy-pasted directly into the decode tool described above.

Using one data set, Tower 2 snow depth, we examined continuous data windows of different length, and considered how this impacts mark recovery. We attempted to decode a provenance mark for 100 random selected intervals of each duration, and found that mark recovery was reliable for periods of 60 hours, with successful recovery occurring in 98% of windows. Recovery succeed in all trials with windows of 84 or more hours. The data is summarized below:

Period duration (hours)	12	24	36	48	60	72	84	96	108
Marks recovered	24%	59%	83%	91%	98%	99%	100%	100%	100%

9 Deployment Issues and Future Work

In this paper we develop a foundational theory comprising an abstract notion of "dataset," and provide a prototype implementation. However, 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 environmental sciences 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 [13] and the Sagehen Creek Field Station repositories [19]. 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 feel strongly 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. In the online data repository described in Section 8, we have made both marked and unmarked datasets available through separate interfaces. It would be straightforward for the data producer to rescind public access to the unmarked dataset interface.

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. This is the approach we have used in our prototype online tool as described in Section 8.

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, e.g. as part of data collection in an embedded system. 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 [23]. 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 efficiency. A 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 8 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.

10 Related Work

The issue of how to represent and manage metadata and provenance in environmental 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 [17] and during curation of large-scale, interconnected data repositories [3], as well as leveraging provenance information to increase the utility of environmental data [15]. Research has resulted in online tools for sensor data storage such as SensorBase [6] 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 [11, 9]. 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, 20]. Such relational watermarking techniques tend use primary keys in the same way we use significant bits—as important, unalterable parts of a tuple, but this is not essential and our datasets may be viewed as simple, unary relations. 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 [7] and in raw video data [12]. Our work accomplishes similar goals for environmental 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 [12], 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.

The work of Sion et al. [21] also considers watermarking streaming data. Their work is motivated by the desire for a tamper-resistant algorithm, and they assume datasets must be ordered streams. In contrast, our work is non-adversarial and makes no particular assumptions about streaming. Toward tamper-resistance Sion et al. present several watermarking schemes that modify the significant bits of some datapoints, so that removing the watermark fundamentally damages the data and enables detection of the edit. This watermarking technique is robust to a different class of transformations than ours. For example, in Sion's work a contiguous stream segment can be replaced by an average, but datapoints cannot be reordered.

Shehab et al. [20] make explicit the importance of not alternating most significant bits when adding provenance metadata to a dataset, identifying that preserving MSBs is useful both to maintain data utility and to provide stable input for metadata encoding and decoding algorithms. Shehab's technique works over relational data and defines a tuple's primary key as its MSBs, and a hash of these bits is used to split relations into disjoint partitions, each of which is able to encode a single bit of provenance information. Similarly, we use MSBs to determine a provenance piece associated with a datapoint. However one of our provenance pieces carry multiple bits of metadata, while the entire set of datapoints in one of Shehab's partition may carries at most one bit. Unlike us, Shehab assumes an adversarial setting.

Tracking provenance in curated databases [5] 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 [4, 22, 10, 16, 18, 24]. This work mostly focuses on combining provenance annotations on data warehouses in a systematic manner; by contrast, we aim to directly associate provenance information with unstructured data.

11 Conclusion

In this paper we have a presented a system for generating *self-identifying data*, which is environmental sciences data marked with its own metadata in an automatically recoverable manner. Our system is intended to support fair-use policies in the environmental sciences 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. Finally, we built a prototype implementation and applied our technique to time-series data collected in the field.

Acknowledgments

We thank James Cheney for inviting us to a meeting at the University of Edinburgh where we started this work, and are grateful for travel support from the UK eScience Institute Theme Program on Principles of Provenance. Thanks to Evan Davis-Drennan and Rebecca Norton for their work on implementing the online decoder utility. Thanks also to Jeff Brown of Sagehen Field Station for inspiring the idea, and to the reviewers and shepherd for useful feedback on a previous version of this work. Christian Skalka's work was supported by a grant from the Air Force Office of Scientific Research.

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] Hans K. Arndt, Thomas Bandholtz, Oliver Günther, Maria Rüther, and Thomas SchützI. EML - the Environmental Markup Language. In *Workshop Symposium on Integration in Environmental Information Systems ISESS 2000*, pages 1–9, 2000. URL `http://www.bandholtz.info/publications/2000-ISESS-EML.pdf`.

[3] 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.

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

[5] 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. ISBN 1-59593-434-0. doi: http://doi.acm.org/10.1145/1142473.1142534.

[6] K. Chang, N. Yau, M. Hansen, and D. Estrin. Sensorbase.org - a centralized repository to slog sensor network data. Technical report, Center for Embedded Network Sensing, 2006.

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

[8] Stephen Chong, Christian Skalka, and Jeffrey A. Vaughan. Self-identifying sensor data. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, IPSN '10, pages 82–93, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-988-6. doi: 10.1145/1791212.1791223. URL `http://doi.acm.org/10.1145/1791212.1791223`.

[9] Ingemar J. Cox and Matt L. Miller. The first 50 years of electronic watermarking. *EURASIP J. Appl. Signal Process.*, 2002(2):126–132, 2002. ISSN 1110-8657. doi: http://dx.doi.org/10.1155/S1110865702000525.

[10] Yingwei Cui and Jennifer Widom. Lineage tracing for general data warehouse transformations. In *27th International Conference on Very Large Data Bases (VLDB 2001)*, 2001. URL `http://ilpubs.stanford.edu:8090/525/`.

[11] 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.

[12] 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. ISBN 0-7695-3058-3.

[13] Hubbard Brook Ecosystem Study. `http://www.hubbardbrook.org/`. Last visited 8/30/13.

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

[15] 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.

[16] Thomas Lee, Stéphane Bressan, and Stuart E. Madnick. Source attribution for querying against semi-structured documents. In Sadri [18], pages 33–39.

[17] 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. URL `http://www.isi.edu/~johnh/PAPERS/Park08b.html`.

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

[19] Sagehen Creek Field Station Data Repository. `http://sagehen.ucnrs.org/resources.htm`. Last visited 8/30/13.

[20] M. Shehab, E. Bertino, and A. Ghafoor. Watermarking relational databases using optimization-based techniques. *Knowledge and Data Engineering, IEEE Transactions on*, 20(1):116–129, 2008.

[21] Radu Sion, Mikhail Atallah, and Sunil Prabhakar. Rights protection for discrete numeric streams. *IEEE Transactions on Knowledge and Data Engineering*, 18(5):699–714, 2006. ISSN 1041-4347. doi: http://doi.ieeecomputersociety.org/10.1109/TKDE.2006.82.

[22] 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.

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

[24] 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.

A Combinatorial analysis

In this section we describe the combinatorial analysis to compute the probability of recovering a provenance mark from a dataset with a given size and redundancy.

Let L_m be the length (in bits) of the provenance mark, L_{pp} be the length (in bits) of each provenance piece, and let N_{pp} be the number of distinct provenance pieces. Let $R = \frac{L_{pp} \times N_{pp}}{L_m}$ be the *redundancy* of the encoding. We assume L_{pp}, N_{pp} and L_m are chosen such that R is a whole number. For example, in Figure 2, with $L_m = 16$, $L_{pp} = 8$, and $N_{pp} = 4$, we have redundancy $R = \frac{8 \times 4}{16} = 2$.

Due to the construction of the provenance pieces, each bit of the provenance mark occurs in the R provenance pieces $\mathsf{pp}^i, \mathsf{pp}^{(i+1) \mod N_{pp}}, \ldots, \mathsf{pp}^{(i+R-1) \mod N_{pp}}$ for some i. For example, in Figure 2, the 6th bit of the provenance mark occurs in provenance pieces pp^0 and pp^1.

Let D be an (uncorrupted) annotated dataset of size $|D|$. In order to be able to recover the provenance mark (without performing any search), the dataset must contain, for each bit of the provenance mark, at least one provenance piece in which that bit occurs. Equivalently, we will be unable to reconstruct the provenance mark if there is some i such that the dataset does *not* contain any of the provenance pieces $\mathsf{pp}^i \mod N_{pp}, \mathsf{pp}^{(i+1) \mod N_{pp}}, \ldots, \mathsf{pp}^{(i+R-1) \mod N_{pp}}$.

Since the provenance piece to annotate a datapoint is chosen uniformly at random (through the use of a cryptographic hash function), we can compute the probability that we will be able to reconstruct the provenance mark from an annotated dataset of size $|D|$: it is the probability that we can select $|D|$ elements from the set $\{0, ..., N_{pp} - 1\}$, with replacement, such that for all i we selected at least one of $i \mod N_{pp}, (i+1) \mod N_{pp}, \ldots, (i+R-1) \mod N_{pp}$. In the remainder of this section, we show recurrence relations to compute this probability. We assume for simplicity that $N_{pp} < |D|$.

The probability of being able to reconstruct the provenance mark from an annotated dataset of size $|D|$ is

$$\frac{\#\mathsf{good}}{(N_{pp})^{|D|}},$$

where $(N_{pp})^{|D|}$ is the total number of ways a dataset of size $|D|$ can choose $|D|$ provenance pieces, and $\#\mathsf{good}$ is the number of ways a dataset of size $|D|$ can choose $|D|$ provenance pieces such that every bit of the provenance mark appears in at least one of the chosen provenance pieces.

We compute $\#\mathsf{good}$ by computing, for each $i \in 1..N_{pp}$, the number of subsets of $\{0, \ldots, N_{pp} - 1\}$ of size i that do not have R consecutive (modulo N_{pp}) numbers missing, and multiply that by the number of ways the dataset could have exactly i distinct provenance pieces occurring in it. The latter number is equal to the number of surjective functions from a domain of size $|D|$ to a domain of size i, given by the formula $(i! \cdot \mathcal{S}(|D|, i))$, where $\mathcal{S}(|D|, i)$ denotes the Stirling number of the second kind.

The number of subsets of size i that do not have R consecutive numbers missing is equal to the total number of subsets of size i minus the number of subsets of size i that have R consecutive numbers missing, which is equal to the subsets of size $N_{pp} - i$ that have R consecutive numbers. Thus, we have

$$\#\mathsf{good} = \sum_{i=1}^{N_{pp}} \left(\binom{N_{pp}}{i} - \#\mathsf{Rconsecutive}(N_{pp} - i) \right) \times (i! \cdot \mathcal{S}(|D|, i))$$

where $\#\mathsf{Rconsecutive}(j)$ is the number of subsets of size j that have R consecutive numbers.

To compute $\#\mathsf{Rconsecutive}(j)$, we let k range from 0 to $N_{pp} - 1$, and count the number of subsets that contain all of $\{k, k+1 \mod N_{pp}, k+2 \mod N_{pp}, \ldots, (k+R-1) \mod N_{pp}\}$. To avoid double counting, we also require that for each k, and for all $0 \leq k' < k$, the subsets do not contain all of $\{k', k'+1 \mod N_{pp}, k'+2 \mod N_{pp}, \ldots, (k'+R-1) \mod N_{pp}\}$.

Let k be fixed. Thus we are considering subsets of size j that contain the R elements $\{k, k+1 \mod N_{pp}, k+2 \mod N_{pp}, \ldots, (k+R-1) \mod N_{pp}\}$. Since the subset is of size j, we must choose another $j-R$ elements. For each $l \in 0..(j-R)$, we choose an additional l elements from the set $\{0, \ldots, (k-1)\}$ and $(j-R-l)$ elements from the set $\{(k+R), \ldots, (N_{pp}-1)\}$. For the l additional elements from the set $\{0, \ldots, (k-1)\}$, we must choose them so that there is no k' such that $0 \leq k' < k$ and the set contains all elements $\{k', k'+1 \mod N_{pp}, k'+2 \mod N_{pp}, \ldots, (k'+R-1) \mod N_{pp}\}$. To ensure this condition holds, there are several restrictions:

- Clearly, we cannot choose the element $k-1$, since $k' = k-1$ would then violate the conditions. So we must choose the l additional elements in the range $0..(k-2)$.
- If $(k+R-1) \mod N_{pp} \geq 0$, then the subset already contains the elements $0, 1, \ldots, (k+R-1 \mod N_{pp})$. Thus, we must choose the l additional elements in the range $a..(k-2)$, where $a = (k+R) \mod N_{pp}$ if $k+R-1 \geq m$, and $a = 0$ otherwise.

  Also, to ensure that $k' = 0$ does not violate the conditions, when choosing the l additional elements from $a..(k-2)$, we cannot choose the first $u = R - (k+R \mod N_{pp})$ elements of the range.
- The l additional elements in the range $0..(k-2)$ cannot contain R consecutive elements (i.e., elements $k', k'+1, \ldots, k'+R-1$ for some $k' \in 0..(k-R)$.

So we need to count the number of ways we can choose l elements in the range $a..(k-2)$ such that we do not choose any R consecutive elements, and we do not choose all of the the first u elements. Note that there are $k-2-a+1$ elements in the range $a..(k-2)$. We define $\#\mathsf{noRconsecutive}(l, k-2-a+1, u)$ to count the number of possible ways we can choose these l elements appropriately.

There are no restrictions on how we choose the $(j-R-l)$ elements in the range $(k+R)..(N_{pp}-1)$. Note that there are $N_{pp}-k-R$ elements in the range $(k+R)..(N_{pp}-1)$, and so there are $\binom{N_{pp}-k-R}{j-R-l}$ ways of choosing $(j-R-l)$ elements in the range $(k+R)..(N_{pp}-1)$.

Note that we assume that $\binom{x}{y} = 1$ if $y = 0$, and $\binom{x}{y} = 0$ if $y \neq 0$ and either $x < 0$ or $y > x$.

$\#\mathsf{Rconsecutive}(j)$

$$= \begin{cases} 0 & \text{if } j < R \\ \sum_{k=0}^{N_{pp}-1} \sum_{l=0}^{j-R} \#\mathsf{noRconsecutive}(l, k-2-a+1, u)\binom{N_{pp}-k-R}{j-R-l} & \text{otherwise} \end{cases}$$

where

$$a = \begin{cases} (k+R \mod N_{pp}) & \text{if } k+R-1 \geq N_{pp} \\ 0 & \text{otherwise} \end{cases}$$

$$u = \begin{cases} R-(k+R \mod N_{pp}) & \text{if } k+R-1 \geq N_{pp} \\ 0 & \text{otherwise} \end{cases}$$

$\#\mathsf{noRconsecutive}(j, b, u)$ is the number of size j subsets of $\{0, \ldots, b-1\}$ such that the subset does not contain R consecutive elements and does not contain all of the elements $\{0, \ldots, (u-1)\}$, i.e, the first u elements. If $u = 0$, then the only restriction is that the subset does not contain R consecutive elements. We assume that u is not negative, and that $u < R$.

If $j = 0$ then there is exactly one size j subset, and this subset satisfies all of the conditions. If $j > b$, then there is no way of choosing a size j subset from $0..(b-1)$.

To compute the number of subsets that satisfy the conditions we consider all possible ways of choosing a size j subset from $0..(b-1)$, and subtract the number of subsets that violate the conditions. $\binom{b}{j}$ is the total number of ways of choosing a size j subset from $0..(b-1)$. We let v be the number of subsets that violate the condition of containing the first u elements. Thus, $v = 0$ if $u = 0$, otherwise, v is equal to the number of ways that the remaining $j-u$ numbers can be chosen from the range $(u-1)..(b-1)$, which is equal to $\binom{b-u}{j-u}$. Finally, we consider the number of subsets that do not contain the first u elements, but that do contain R consecutive elements.

To count the number of subsets that do not contain the first u elements, but that do contain R consecutive elements, we let k range from 0 to $b-R$, and count the number of subsets that contain $k, k+1, k+2, \ldots, k+R-1$, and for all $0 \leq k' < k$, do not contain all of $k', k'+1, k'+2, \ldots, k'+R-1$. Note that if $k > b-R$, then $k+R-1 > b-1$, and so $k, k+1, k+2, \ldots, k+R-1$ would not be a legal set of elements to choose.

Let k be fixed. Thus we are considering subsets that contain the R elements $k, k+1, k+2, \ldots, k+R-1$. Since the subset is of size j, there must be another $j-R$ elements. For $l \in 0..(j-R)$, we consider choosing an additional l elements in the range $0..(k-1)$ and $(j-R-l)$ elements in the range $(k+R)..(b-1)$. For the l additional elements in the range $0..(k-1)$, we must choose them so that there is no k' such that $0 \leq k' < k$ and $k', k'+1, k'+2, \ldots, k'+R-1$, and so that the first u elements are not chosen. So the element $k-1$ cannot be in the set (since $k' = k-1$ would violate the conditions), and so we are actually choosing l elements in the range $0..(k-2)$. Note that there are $k-1$ elements in the range $0..(k-2)$. Thus, we use $\#\mathsf{noRconsecutive}(l, k-1, u)$ to count the number of ways of choosing l additional elements in the range $0..(k-2)$ without having a consecutive sequence of size R or selecting the first u elements. Note that this recurrence is well-founded in the second argument.

There are no restrictions on how we choose the $(j-R-l)$ elements in the range $(k+R)..(b-1)$. Note that there are $b-k-R$ elements in the range $(k+R)..(b-1)$, and so there are $\binom{b-k-R}{j-R-l}$ ways of choosing $(j-R-l)$ elements in the range $(k+R)..(b-1)$.

$\#\mathsf{noRconsecutive}(j, b, u)$

$$= \begin{cases} 1 & \text{if } j = 0 \\ 0 & \text{if } j > b \\ \binom{b}{j} - v - \sum_{k=c}^{b-R} \sum_{l=0}^{j-R} \#\mathsf{noRconsecutive}(l, k-1, u)\binom{b-k-R}{j-R-l} & \text{otherwise} \end{cases}$$

where

$$v = \begin{cases} 0 & \text{if } u = 0 \text{ or } u > j \\ \binom{b-u}{j-u} & \text{otherwise} \end{cases}$$

$$c = \begin{cases} 0 & \text{if } u = 0 \\ 1 & \text{otherwise} \end{cases}$$