Proving Ownership over Categorical Data *

Radu Sion
Computer Sciences and The Center for Education and Research
in Information Assurance and Security, Purdue University
West Lafayette, IN, 47907, USA
sion@cs.purdue.edu

Abstract

This paper introduces a novel method of rights protection for categorical data through watermarking. We discover new watermark embedding channels for relational data with categorical types. We design novel watermark encoding algorithms and analyze important theoretical bounds including mark vulnerability. While fully preserving data quality requirements, our solution survives important attacks, such as subset selection and data re-sorting. Mark detection is fully "blind" in that it doesn't require the original data, an important characteristic especially in the case of massive data. We propose various improvements and alternative encoding methods. We perform validation experiments by watermarking the outsourced Wal-Mart sales data available at our institute. We prove (experimentally and by analysis) our solution to be extremely resilient to both alteration and data loss attacks, for example tolerating up to 80% data loss with a watermark alteration of only 25%.

1 Introduction

The relational data framework would benefit from an intrinsic method of rights protection especially in scenarios where sensitive, valuable content is to be sold or made directly accessible. Good examples are data mining applications (e.g. sales database, oil drilling data, financial data), where a set of data is usually produced/collected by a data collector and then sold in pieces to parties specialized in mining that data. Other applications include scenarios such as online B2B interactions (e.g. airline reservation and scheduling portals) in which data is made available for direct, inter-active use.

Different avenues for rights protection are available, each with its own advantages and drawbacks. Enforcement by legal means is usually ineffective in preventing theft of copyrighted works, *unless* augmented by a digital counter-part. Watermarking provides one avenue for rights protection. The main purpose of Digital Watermarking is to protect a certain content from unauthorized duplication and distribution by enabling provable ownership over the content. Digital Watermarking has traditionally [3] [4] relied upon the availability of a large noise domain within which the object can be altered while retaining its essential properties. For example, the least significant bits of image pixels can be arbitrarily altered with little impact on the visual quality of the image (as perceived by a human). In fact, much of the "bandwidth" for inserting watermarks in multi-media Works (such as in the least significant bits) is due to the inability of human sensory system (especially sight and hearing) to detect certain changes.

More recently, the focus of watermarking for digital rights protection is shifting toward different data domains such as natural language text [1], and software [2] [7]. Since these data types often have very well defined semantics (as compared to those of images, video, or music) and may be designed for machine ingestion, the identification of

*Portions of this work were supported by Grants EIA-9903545, ISS-0219560, IIS-9985019 and IIS-9972883 from the National Science Foundation, N00014-02-1-0364 from the Office of Naval Research, and by sponsors of the Center for Education and Research in Information Assurance and Security.

the available "bandwidth" for watermarking is as important a challenge as the algorithms for inserting the watermarks themselves.

Watermarking works by deploying resilient information hiding techniques to [11] insert an indelible mark in the data such that (i) the insertion of the mark does not destroy the value of the data (i.e. the data is still useful for the *intended purpose*); and (ii) it is difficult for an adversary to remove or alter the mark beyond detection without destroying the value of the data. Clearly, the notion of value or utility of the data is central to the watermarking process. This value is closely related to the type of data and its intended use. For example, in the case of software the value may be in ensuring equivalent computation, and for text it may be in conveying the same meaning (i.e. synonym substitution is acceptable). Similarly, for a collection of numbers, the utility of the data may lie in the actual values, in the relative values of the numbers, or in the distribution (e.g. normal with a certain mean).

An important point about watermarking should be noted. By its very nature, a watermark modifies the item being watermarked. If the data to be watermarked cannot be modified then a watermark cannot be inserted. The critical issue is not to avoid changing the data, but to limit the change to acceptable levels with respect to the intended use of the data. Clearly, one can always identify some use of the data that is affected by even a minor change to the any portion of the data. It is therefore necessary that the intended purpose of the data that should be preserved be identified during the watermarking process.

Existing related research [6] [12] explores the issue of watermarking *numeric* relational content. [5] provides theoretical constructs linking query preservation to allowable data alteration bounds.

Here we propose and analyze the issue of rights protection for *categorical* relational content through watermarking. Important new challenges are associated with this domain. One cannot rely on 'small" alterations to the data in the embedding process. Any alteration is going to necessarily be significant. The discrete characteristics of the data require discovery of fundamentally new bandwidth channels and associated encoding algorithms. Our solution proves to be resilient to various important classes of attacks, including data re- shuffling/sorting, massive subset selection, linear data changes, random item(s) alterations etc. The main contributions of this paper include: (i) the proposal and definition of the problem of watermarking categorical data, (ii) the discovery and analysis of new watermark embedding channels for relational data with categorical types, (iii) the design of novel associated encoding algorithms.

The paper is structured as follows. In Section 2 we present our main data and adversary models. Section 3 introduces the main solution and outlines alternatives. Section 4 analyzes aspects of our algorithms and proposes improvements for particular scenarios. It also discusses the inherent algorithm vulnerability to data altering attacks. Section 5 outlines our experimental setup and results. Section 6 concludes.

2 Model

We choose to keep our model concise but representative. Our data schema includes a set of discrete attributes $\{A, B\}$ and a primary data key K, not necessarily discrete. Any attribute $X\ in\{A, B\}$ can yield a value out of n_X possibilities. (e.g. city names, airline names). Thus our schema is (K,A,B).

2.1 Notation

Let the number of tuples in the database be N. By notation, for any entity X (e.g. relation attribute), let $b(X)$ be the number of bits required for its representation and $msb(X, b)$ the most significant b bits. If $b(X) < b$ we left-pad X with $(b-b(X))$ zeroes to form a b-bit result.For any categorical attribute X we naturally have $b(n_X) \leq b(X)$. Let $T_j(X)$ be the value of attribute X in tuple j. Let $\{a_1, ..., a_{n_A}\}$ the discrete values of attribute A. These are distinct and can be sorted (e.g. by ASCII value). Let $f_A(a_j)$ be the normalized (to 1.0) occurrence frequency of value a_j in attribute A. $f_A(a_j)$ models the de-facto occurrence probability of value a_j in attribute A. Let wm be a $|wm|$-bit watermark to be embedded, $wm[i]$ the i-th bit of wm. In any following mathematical expression the symbol "&" signifies a *bit-AND* operation. Also, let $set\_bit(d, a, b)$ be a function that returns d with the bit position a set to value b.

2.2 One-Way Hashing

In fighting court-time exhaustive key search claims, we are leveraging a special de-facto secure construct, the one-way cryptographic hash. Let $crypto\_hash()$ be a cryptographic secure one-way hash. In our solution we are using the fact that there exists an assumption that it is computationally unfeasible, for a given value V' to find a V such that $crypto\_hash(V) = V'$. This assumption of one-wayness lies at the heart of many current security protocols. Examples of potential candidates for $crypto\_hash()$ are the MD5 or SHA hash. For more details on cryptographic hashes consult [9]. By notation, let $H(V, k) = crypto\_hash(k; V; k)$ (where ";" denotes concatenation).

2.3 The Adversary

There is a set of attacks that can be performed by evil Mallory with the purpose of defeating the watermark while preserving the value in the data. More-over these perceived attacks may be the result of normal use of the data by the intended user. In order to be effective, the watermarking technique has to consider these scenarios and be able to survive them. In [12] we outlined attacks in the numeric relational data framework. Here we present a summary and discuss challenges associated with categorical data.

A1. Horizontal Data Partitioning Mallory can randomly select and use a subset of the original data set that might still provide value for its intended purpose.

A2. Subset Addition Mallory adds a set of tuples to the original data. This addition is not to significantly alter the useful (from Mallory's perspective) properties of the initial set versus the resulting set.

A3. Subset Alteration Altering a subset of the items in the original data set such that there is still value associated with the resulting set. In the categorical data framework, subset alteration is intuitively quite expensive from a data-value preservation perspective. One has also to take into account semantic consistency issues that become immediately visible because of the discrete nature of the data.

A4. Subset Re-sorting If a certain order can be imposed on the data then watermark retrieval/detection should be resilient to re-sorting attacks and should not depend on this predefined ordering.

A5. Vertical Data Partitioning In this attack, a valuable subset of the attributes are selected (by vertical partitioning) by Mallory. The mark has to be able to survive this partitioning. The encoding method has to feature a certain attribute-level property that could be recovered in such a vertical partition of the data. We believe that while vertical data partitioning attacks are possible and also very likely in certain scenarios, often value is to be found in the association between a set of relation attributes. These attributes are highly likely to survive such an attack, as the final goal of the attacker is to produce a still-valuable result.

A6. Attribute Remapping If data semantics allow it, re-mapping of relation attributes can amount to a powerful attack that should be carefully considered. In other words, if Mallory can find an, at least partial, value-preserving mapping from the original attribute data domain to a new domain, a watermark should hopefully survive such a transformation. The difficulty of this challenge is increased by the fact that there naturally are an infinity of transformations available for a specific data domain. Determining a value-yielding one is both data and consumer dependent. This is thus an intractable task for the generic case. One special case is primary key re-mapping. In Section 4.5 we discuss the particular case of bijective mappings.

Given the attacks above, several properties of a successful solution surface. For immunity against **A1**, the watermark has to be embedded in overall data properties that survive subset selection. If the assumption is made that the attack alterations do not destroy the value of the data, then **A3** should be defeat-able by embedding the primitive mark in resilient global data properties. Since it adds new data to the set, defeating **A2** seems to be the most difficult task, as it implies the ability to identify potential uses of the data (for the attacker). This is especially so in the case of categorical data where we suspect the main attack will focus not as much on expensive data alterations but more on data addition.

2.4 Embedding Limits

By its very nature, watermarking modifies its input. If data constraints on the input - output relation-ship are too restrictive and do not allow enough change, watermarking could potentially fail due to lack of bandwidth. While it is necessary that the intended purpose of the data should be identified and preserved during the watermarking process, sometimes this is not possible.

The extreme case occurs when *any* alterations are allowed to the data. In this case the available bandwidth is directly related to the data entropy. As this requirement is becoming increasingly restrictive, the watermark is necessarily becoming more vulnerable. Often we can express the available bandwidth as an increasing function of allowed alterations. In Section 4.4 we discuss the relation-ship between attack vulnerability and embedding bandwidth.

3 Categorical Data

The discrete nature of our data domain results in an inherent limitation in the associated entropy. In order to enable watermarking, we first aim to discover appropriate embedding channels. Next, we propose new encoding methods, able to leverage the newly discovered bandwidth.

3.1 Bandwidth Channels

If the discrete attribute A has a finite set of possible values (n_A), unless this value is really high, the associated $log_2(n_A)$ bits entropy is not going to be enough for direct-domain embedding of a reasonable watermark length/convince-ability. For example in the case of departure cities, a value of $n_A = 16000$ is going to yield only 14 bits.

In the case of categorical data however (and not necessarily in any other continuous data domain) there exists a natural, solid semantic association between A, the rest of the schema's categorical attributes (e.g. B) and the data's primary key K. This association derives from the fact that in most cases there exists no concept of "minor" changes. *Any* change is going to necessarily be significant (e.g. change departure city from "Chicago" to "San Jose"). A comparatively large potential encoding bandwidth can be found in these *associations between categorical attributes* (including possibly the primary key). We propose to make use of it in our embedding algorithm.

Additionally, while direct-domain embedding does not seem to have enough entropy potential, we will leverage a related dimension, the *value occurrence frequency-transform*, (attribute frequency histogram) as an additional (or alternate) encoding channel.

Our next objective is to provide an embedding method that is able to resiliently hide information in the attribute association outlined above (while preserving guaranteed data distortion bounds) and then, if necessary, augment it with a direct-domain watermark.

3.2 Algorithms

Surviving vertical partitioning attacks is important and requires a careful consideration of the attribute association used in the embedding process. Selecting the appropriate attributes is challenging as one has to determine many possible valuable features to be found in the data that would still be preserved after vertical partitioning.

This is why we propose an initial user-level assessment step in which a set of attributes are selected that are likely to survive vertical partitioning attacks (see Section 3.3 for an extended discussion). In the extreme case (i), just one attribute and the primary key are going to survive. A milder alternative (ii) assumes that several (e.g. two) categorical attributes and the primary key survive the partitioning process. Apparently, a watermarking method for (i) presents the disadvantage of a direct primary key-dependency. In Section 3.3 we further expand on this.

Let us propose an encoding method for (i), in which we encode a watermark in the bandwidth derived from the association between the primary key and a categorical attribute A. In Section 4 we analyze (ii).

3.2.1 Mark Encoding

At mark encoding time we assume the following input: a relation with at least a categorical type attribute A (to be watermarked), a watermark wm and a set of secret keys (k_1, k_2) and other parameters (e.g. e) used

wm_embed(K,A,wm,k_1,k_2,e,ECC)
$wm\_data \leftarrow ECC.encode(wm, wm.length)$
for ($j \leftarrow 1; j < N; j \leftarrow j + 1$)
if ($H(T_j(K), k_1)$) mod $e = 0$) **then**
$t \leftarrow set\_bit(H(T_j(K), k_1), 0,$
$wm\_data[H(T_j(K), k_2)])$
$T_j(A) \leftarrow a_t$

wm_embed_alternate(K,A,wm,k_1,e,ECC)
$wm\_data \leftarrow ECC.encode(wm, wm.length)$
$idx \leftarrow 0$
for ($j \leftarrow 1; j < N; j \leftarrow j + 1$)
if ($H(T_j(K), k_1)$ mod $e = 0$) **then**
$t \leftarrow set\_bit(H(T_j(K), k_1), 0, wm\_data[idx])$
$T_j(A) \leftarrow a_t$
$embedding\_map[T_j(K)] \leftarrow idx$
$idx \leftarrow idx + 1$
return $embedding\_map$

Figure 1. (a) Embedding Algorithm (b) Alternative using embedding map (bit size adjustments left out for formatting purposes)

in the embedding process. The algorithm starts by discovering a set of "fit" tuples determined directly by the association between A and the primary relation key K. These tuples are considered for mark encoding.

We say that a tuple T_i is "fit" for encoding iff $H(T_i(K), k_1)$ mod $e = 0$, where e is an adjustable encoding parameter determining the percentage of considered tuples [1] and k_1 is a secret $max(b(N), b(A))$-bit key. In other words, a tuple is considered "fit" if its primary key attribute satisfies a certain secret criteria [2].

Error Correction. Because often the available embedding bandwidth $\frac{N}{e}$ is greater than the watermark bit-size $|wm|$, we can afford the deployment of an error correcting code (ECC) that, upon embedding takes as input a desired watermark wm and produces as output a string of bits wm_data of length $\frac{N}{e}$ containing a redundant encoding of the watermark, tolerating a certain amount of bit-loss, $wm\_data = ECC.encode(wm, \frac{N}{e})$. At decoding time, the ECC takes as input (a potentially altered) wm_data and produces the (most likely) corresponding wm, $wm = ECC.decode(wm\_data, |wm|)$. There are a multitude of error correcting codes to choose from. As this does not constitute the main contribution of this research, in our implementation we deploy majority voting codes. Let $wm\_data[i]$ be the i-th bit of wm_data. Thus, before embedding, our algorithm starts by deploying the error correcting code first to compute the bits to be embedded $wm\_data = ECC.encode(wm, \frac{N}{e})$.

For each "fit" tuple T_i, we encode one bit by altering $T_i(A)$ to become $T_i(A) = a_t$ where $t = set\_bit(msb(H(T_i(K), k_1), b(n_A)), 0, wm\_data[msb(H(T_i(K), k_2), b(\frac{N}{e}))])$, where k_2 is a secret key $k_2 \neq k_1$. In other words, we are generating a secret value of $b(n_A)$ bits (depending on the primary key and k_1) and then force its least significant bit to a value according to a corresponding (random, depending on the primary key and k_2) position in wm_data.

The use of a second different key here ensures that there is no correlation between the selected tuples for embedding (selected also by k_1) and the corresponding bit value positions in wm_data (selected by k_2). Such a correlation would potentially cause certain bits to be never considered in the embedding process. In summary, the new attribute value is selected by the secret key k_1, the associated relational primary key value and a corresponding bit from the watermark data wm_data.

The "fitness" selection step provides several advantages. On the one hand this ensures the secrecy and resilience of our method, on the other hand, it effectively "modulates" the watermark encoding process to the actual attribute-primary key association. Additionally, this is the place where the cryptographic safety of the hash one-wayness is leveraged to defeat court-time attacks in which Mallory claims that the data in dispute is not actually watermarked but that rather certain values for k_1,k_2 were searched for to yield the watermark.

Note: When computing t (i.e. selecting a new value for $T_i(A)$) there can be (arguably rare) cases when we select the same wm_data bit to embed The pseudo-random nature of $H(T_i(K), k_2)$ guarantees on average

[1] The set of fit tuples contains roughly $\frac{N}{e}$ elements. The parameter e can be controlled at embedding time to adjust the trade-off between the level of data alteration and mark resilience. See Section 4.4 for a more detailed analysis.

[2] Similar criteria are found in various frameworks, such as [6].

that a large majority of the bits in wm_data are going to be embedded at least once. The ulterior step of error correction can tolerate such small changes.

Alternately, we could keep an on-the-fly hash-table/mapping (with $(\frac{N}{e})$ entries, See Figures 1 (b) and 2 (b)) between $T_i(K)$ values and the actual considered bit index in wm_data. This mapping can be used at detection time to accurately detect *all* wm_data bits. In this case, also we do not require an extra watermark bit selection key (k_2). Although we use this alternative in our implementation, for simplicity and conciseness reasons we are not going to discuss it here.

The advantage of using $H(T_i(K), k_2)$ in selecting the wm_data bit to embed becomes clear when we discuss data loss alterations. Because the selected bit is directly related only to the *currently* considered tuple, this method naturally survives subset selection and data addition attacks. More on this in section 5.

While it does a good job in watermark embedding, data alteration is an expensive operation because it effectively destroys valuable data. There are also other data transformations that we can make use of, each with a different degree of associated data distortion and benefits. For a discussion on an alternative (i.e. data addition) see Section 4.6.

3.2.2 Mark Decoding

In the decoding phase we assume the following input: the potentially watermarked data, the secret keys k_1, k_2 and e. We then use the same criteria for discovering "fit" tuples. That is, we say that a tuple T_i is "fit" for encoding iff $H(T_i(K), k_1)$ mod $e = 0$.

The first aim of the decoding algorithm is to discover the embedded wm_data bit string. For each "fit" tuple T_i, with $T_i(A) = a_t$, we set $wm\_data[msb(H(T_j(K), k_2), b(\frac{N}{e}))] = t\&1$.

Once wm_data (possibly altered) is available, the error correcting mechanism is invoked to generate the ("closest", most likely) corresponding watermark wm, $wm = ECC.decode(wm\_data, |wm|)$.

3.3 Multiple Attribute Embeddings

We introduced a watermarking method making use of the bandwidth present in the association between the primary key and the categorical type attribute A.

---

wm_decode(K,A,k_1,k_2,e,ECC)
  for ($j \leftarrow 1$; $j < N$; $j \leftarrow j + 1$)
    if ($H(T_j(K), k_1)$ mod $e = 0$) **then**
    determine t such that $T_j(A) = a_t$
    $wm\_data[msb(H(T_j(K), k_2), b(\frac{N}{e}))] = t\&1$
  $wm \leftarrow ECC.decode(wm\_data, wm.length)$
  return wm

wm_decode_alternate(K,A,k_1,e,ECC,$embedding\_map$)
  for ($j \leftarrow 1$; $j < N$; $j \leftarrow j + 1$)
    if ($H(T_j(K), msb(k, b(K)))$ mod $e = 0$) **then**
    determine t such that $T_j(A) = a_t$
    $wm\_data[embedding\_map[T_j(K)]] = t\&1$
  $wm \leftarrow ECC.decode(wm\_data, wm.length)$
  return wm

Figure 2. (a) Decoding Algorithm (b) Alternative using embedding map

This method did *not* touch the primary key attribute but rather relied on modulating A through minor alterations (and data additions, see Section 4.6).

In the following we extend this algorithm to provide more generality and resilience, in particular to attacks of the type A5 (vertical data partitions). In a possible attack scenario Mallory partitions the data in such a way as to preserve only two attributes and no primary key. Moreover, if one of the remaining attributes can act as a primary key, this partitioning results in no duplicates-related data loss (in the two attributes).

Defeating this scenario leads to a natural extension. Instead of relying on the association between the primary key and A, the extended algorithm considers *all* pairs [3] of attributes and embed a watermark separately in *each* of these associations. In other words, if the original watermarking method read $mark(K, A)$ for a schema composed of the primary key K and A, in the case of a (K, A, B) schema we apply the watermark several times, for example $mark(K, A)$; $mark(K, B)$; $mark(A, B)$. In each case, we treat one of the attributes as a primary key in Section 3.2, while maintaining the rest of the algorithm in place. This provides protection against

[3] For simplicity we consider pairs for now, but believe that an arbitrary number of attributes could be considered.

A5 attacks and allows for more resilience in the rest of the scenarios (as there are more rights "witnesses" to testify). In addition, this effectively "breaks" the previous algorithm's dependency of the primary key.

Several issues need to be resolved. One apparent problem is the issue of interference. If we watermark the pair (K, A) and then aim to watermark (K, B) everything seems to work out fine as the modified attributes A, B are different. With the exception of semantic consistency issues that would need to be handled (as they would also be in the initial case, see Section 4) the two encodings seem to be independent. But in the case of additionally watermarking the pair (A, B), modifying B suddenly interferes with the modifications occurred in the (K, B) case.

Although the level of interference is likely to be very low [4], there exists a solution to this problem. Maintaining a hash-map at watermarking time, "remembering" modified tuples in each marking pass, allows the algorithm (extended accordingly) to avoid tuples and/or values that were already considered.

Additionally, when considering the association between two attributes A, B as an encoding channel for a watermark, if values in B were already altered during a previous encoding, instead of deploying $mark(A, B)$ (which would result in further alterations to B), we propose the deployment of $mark(B, A)$. While still encoding the mark in the association between A and B, by modifying A (assumed un-modified yet, otherwise it doesn't matter anyway) we effectively "spread" the watermark throughout the entire data, increasing its level of resilience.

Moreover, if data constraints allow, we propose watermarking each and every attribute pair by first building a closure for the set of attribute pairs over the entire schema that minimizes the number of encoding interferences while maximizing the number of pairs watermarked. Due to limited space constraints we are not elaborating on this here.

Note: The discrete nature of categorical attributes complicates the watermarking process of a pair (A, B) in which a categorical attribute A is used as a primary key (in the initial algorithm). In the extreme case, A can have just one possible value which would upset the "fit" tuple selection algorithm. It remains to be investigated if a pair-closure can be constructed over the schema such that no categorical attributes are going to be used as primary key place-holders.

[4] As the probability of the same tuple to be considered again in the second encoding is low, especially in large data sets, see Section 4.4 for a related analysis.

4 Discussion

4.1 On-the-fly Quality Assessment

In the relational framework it is important to preserve structural and semantic properties of the data. Because by its very nature, watermarking alters its input, we have to provide a mechanism ensuring that these alterations are not degrading the data beyond usability. Preserving data quality requires the ability to express and enforce data constraints. Sometimes it is undesirable or even impossible to directly map higher level semantic constraints directly into low level (combined) change tolerances for individual tuples or attributes [5]. The practically infinite set of potential semantic constraints that can be desired/imposed on a given data set makes it such that versatile, "data goodness" (i.e. semantically) assessment method is required. Thus, we propose to extend the marking algorithm with semantic data constraints awareness. We introduced and successfully analyzed this idea in [12]. Each property of the database that needs to be preserved is written as a constraint on the allowable change to the dataset. The watermarking algorithm is then applied with these constraints as input and re-evaluates them continuously for each alteration. A rollback log (see Figure 3) is kept to allow undo operations in case certain constraints are violated by the current watermarking step. Due to space considerations, we are not going to elaborate on this further. In the following we are going to focus on the encoding method itself.

4.2 Frequency Domain Encoding

While most vertical partitioning attacks can be handled by a multiple attribute embedding solution as

[5] It should be noted that not all constraints of the database need to be specified. A practical approach would be to begin by specifying a upper bound on the percentage of allowable data alterations. Further semantic or structural constraints that the user would like to preserve can be added to these basic constraints.

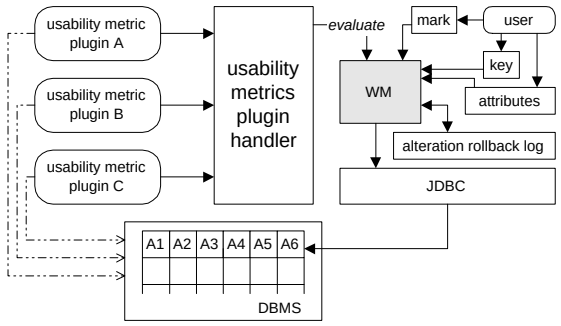

Figure 3. Data quality is continuously evaluated. A rollback log aids undo operations in cases where the watermark embedding would violate quality constraints.

described in Section 3.3, consider an extreme vertical partitioning attack scenario in which Mallory only preserves a single (categorical) attribute A.

An intuitive assumption is that n_A (the number of possible values in A) is much smaller than N, thus A (by itself) is naturally containing many duplicate values. Because there is probably very little value associated with knowing the set of possible values of $\{a_1, ..., a_{n_A}\}$, the main value of A (in Mallory's eyes) is (arguably) to be found in one of the only remaining characteristic properties, namely the value occurrence frequency distribution $[f_A(a_i)]_{i\in(1,n_A)}$. If we could devise an alternative watermark encoding method for this set we would be able to associate rights also to this aspect of the data, thus surviving this extreme partitioning attack.

Note: If the data value occurrences are uniformly distributed (often unlikely, imagine airport or product codes) distinguishing among these values will not work and (arguably) there is nothing one can do to watermark that result.

In [10] we introduced a watermarking method for numeric sets that is able to minimize the absolute data alteration in terms of distance from the original data set. We propose to apply this method here to embed a mark in the occurrence frequency distribution domain. One concern we should consider is the fact that in the categorical domain we are usually interested in minimizing the *number* of data items altered whereas in the numeric domain we aim to minimize the absolute data change. It is surprising and fortunate that, because $[f_A(a_i)]_{i\in(1,n_A)}$ are values modeling occurrence frequency, a solution minimizing absolute data change in this (frequency) domain naturally minimizes the *number* of items changed in the categorical value domain. Other concerns include issues such as mark interference (with the other encodings), which can be solved by an approach similar to the one in Section 3.3 using embedding markers.

4.3 Blindness and Incremental Updates

Our watermarking method is blind, in that it doesn't require the original data in the detection process. This is important, because it is un-realistic to assume the original data available after a longer time elapses, especially in the case of massive data sets. Also, our method supports incremental updates naturally. As updates occur to the data, the resulting tuples can be evaluated on the fly for "fitness" and watermarked accordingly.

4.4 Attack Vulnerability

In order to fight false-positive claims in court we ask: *What is the probability of a given watermark of length $|wm|$ to be detected in a random data set of size N ?* The assumption is of course that $|wm| < \frac{N}{e}$ (enough bandwidth).

It is easy to prove that this probability is $(\frac{1}{2})^{|wm|}$. In case multiple embeddings are used (e.g. majority voting) and all available bits are utilized, this probability decreases even more to $(\frac{1}{2})^{\frac{N}{e}}$. For example, in the case of a data set with $N = 6000$ tuples and with $e = 60$, this probability is approximately 7.8×10^{-31}.

In the absence of additional information, Mallory, faced with the issue of destroying the watermark while preserving the value of the data, has only one alternative available, namely a random attack (Here we are concerned with data alteration attacks). We ask: what is the probability of success of such an attack ? In other words, if an attacker randomly alters a total number of a data tuples and succeeds in each case to flip the embedded watermark bit with a success rate p, *what is the probability of success*

of altering at least $r, r < a$ watermark bits in the result, $P(r,a)$? It can be shown that $P(r,a) = \sum_{i=r}^{a} [aCi] \times p^a \times (1-p)^{a-i}$.

Remember that only every e-th tuple (on average) is watermarked, thus Mallory effectively attacks only an average of $\frac{a}{e}$ tuples actually watermarked. If $r > \frac{a}{e}$ then $P(r,a) = 0$. In the case of $r < \frac{a}{e}$ we have we have the corrected version

$$P(r,a) = \sum_{i=r}^{(\frac{a}{e})} C_i^{\frac{a}{e}} \times p^{(\frac{a}{e})} \times (1-p)^{(\frac{a}{e})-i} \quad (1)$$

Consider $r = 15$, $p = 70\%$ (it is quite likely that when Mallory alters a watermarked tuple, it will destroy the embedded bit), $a = 1200$ (20% of the tuples are altered by the attacker[6], $|wm| = 10$ and $e = 60$ ($|wm\_data| = 100$).

Because we have an effectively binomial distribution experiment with $X_i = 1$, with probability p and $X_i = 0$, with probability $1-p$. $E[X_i] = p$, $var(X_i) = E[X_i^2] - (E[X_i])^2 = ... = p \times (1-p)$, by using the central limit theorem [8], we can derive that $f(\sum X_i)$, where

$$f(\sum X_i) = \frac{\sum X_i - \frac{a}{e} \times p}{\sqrt{\frac{a}{e} \times p \times (1-p)}} \quad (2)$$

effectively behaves like a normal distribution $N(0,1)$ (when $\frac{a}{e} \times p \geq 5$ and $\frac{a}{e} \times (1-p) \geq 5$). In other words, the probability that $(\sum X_i) > r$ (attack altering at least r bits) can be rewritten as the probability of $f(X_i) > f(r)$. Because of the normal behavior of $f(x)$ (we know $f(r)$) we can estimate this probability by normal distribution table lookup. Thus, we get $P(15, 1200) \approx 31.6\%$.

If we assume that the error correcting code tolerates an average of $t_{ecc} = 5\%$ alterations to the underlying data and that the alteration propagation is uniform and stable [7] then the final watermark is going to incur only an average fraction of

$$(\frac{r}{\frac{N}{e}} - t_{ecc}) \times \frac{|wm|}{|wm\_data|}$$

alteration. In our case this is only 1.0%, corresponding to an average of 1.0 bit in the watermark. *Thus in order to modify one bit in the watermark Mallory has to alter at least 20% of the data and even then has only a success rate of 31.6% !* This analysis was done in a highly attack-favorable scenario in which error correction can only handle 5% alterations in wm_data.

Because data alteration is expensive, naturally we aim to minimize the number of altered tuples in the watermarking process. If we define attack vulnerability as the probability $P(r,a)_1$ to succeed in altering one bit in the final watermark (wm), and the number of altered tuples is defined by the ratio $\frac{N}{e}$, we ask: *what is the relationship between the required number of fit tuple encodings (i.e. available bandwidth) and attack vulnerability ?* In other words, what is the minimum number of alterations we have to allow (and perform) in the watermarking phase that would guarantee a certain upper bound on the overall attack vulnerability. While this relationship is somewhat defined by equation (1), we are interested here in an actual estimate for a likely scenario.

If we assume that Mallory cannot afford to modify more than 10% of the data items ($a = 600$) and we set a maximum tolerable threshold $\tau = 10\%$ for $P(r,a)_1$ ($P(r,a)_1 < \tau$), let us compute the minimum required e to guarantee these bounds (the other values are as above). By using equation (2) and doing a normal distribution table lookup we derive that (for $\tau = 10\%$) we have to satisfy $\frac{r - \frac{a}{e} \times p}{\sqrt{\frac{a}{e} \times p \times (1-p)}} = 1.28$, which results in $e \approx 23$. In other words, we have to alter only $\approx 4.3\%$ of the data to guarantee these bounds !

4.5 Bijective Attribute Re-mapping

Consider the scenario of an attack in which the categorical attribute A is re-mapped through a bijective function to a new data domain. In other words, the $\{a_1, ..., a_{n_A}\}$ values are going to be mapped into a different set $\{a'_1, ..., a'_{n_A}\}$. The assumption here is that from Mallory's perspective, the re-mapped data still features enough value that can be banked upon [8].

The problem of remapping becomes clear in the mark detection phase when, after tuple fitness

[6] This is likely a highly value-damaging operation overall. Such an attack is unlikely because Mallory cannot afford destroying the data beyond use. We present it for illustration purposes.

[7] These are intuitive terms we use to denote the fact that if one bit in wm_data is altered above the t_{ecc} bound then a stable average of $\frac{|wm|}{|wm\_data|}$ are altered in the resulting error corrected watermark $wm = ECC.decode(wm\_data, |wm|)$.

[8] Even more, Mallory could sell a secret secure black-box "reverse mapper" together with the re-mapped data to third parties, still producing revenue

selection, the bit decoding mechanism will fail, being unable to determine t such that $T_j(A) = a_t$. It will instead determine a t value that maps to the $\{a'_1, ..., a'_{n_A}\}$ value set. Thus our main challenge is to discover the mapping (or a major part of it) and apply its inverse in the detection phase.

Unless the items in the initial set $\{a_1, ..., a_{n_A}\}$ feature a peculiar distinguishing property, intuitively this task is impossible for the general case, as there are a large number of possible mappings. Nevertheless, over large data sets we argue that a such distinguishing property might exist, namely the value occurrence frequency for the items in $\{a_1, ..., a_{n_A}\}$. We propose to sample this frequency in the suspected (remapped) dataset and compare the resulting estimates ($E[f_A(a'_j)]_{j\in(1,n_A)}$) with the known occurrence frequencies ($(f_A(a_j))_{j\in(1,n_A)}$). Next, we sort both sets and associate items by comparing their values. For example if the closest value to $E[f_A(a'_i)]$ (in the set $E[f_A(a'_j)]_{j\in(1,n_A)}$) is $f_A(a_j)$ (in the set $(f_A(a_j))_{j\in(1,n_A)}$), then we add $i \rightarrow j$ to the inverse mapping to be used at watermark decoding time.

4.6 Data Addition

While it does a good job in watermark embedding, data alteration is an expensive operation because it effectively destroys valuable data. There are also other data transformations that we can make use of, each with a different degree of associated data distortion and benefits. In particular, *data addition* seems to be a promising candidate. Intuitively it features a much lower data distortion rate (no actual alterations) and thus presents potentially higher benefits. On the other hand there likely exists an upper bound on the number of tuples that can be added to the data. Let p_{add} be the upper bound on the allowed additional percentage of tuples to be added. We propose that in addition to the initial data-altering step, we artificially "inject" watermarked tuples that conform to the "fitness" criteria (while conforming to the overall data distribution, in order to preserve stealthiness).

But isn't data addition of "fit" tuples inhibited by the one-way nature of the used cryptographic hash ? Not exactly. Because c effectively "reduces" the fitness criteria testing space to a cardinality of c, we can afford to massively produce random tuple values (within the appropriate attribute data domain) and test for "fitness". On average (depending on the randomness of the tuple producing mechanism), one in every c tuples should conform (as the values are evaluated modulo c).

If a percentage of p_{add} artificially produced tuples are to be added to the data, the watermark is effectively enforced with an additional $p_{add} \times N$ bits. See Section 4.4 for an analysis on the impact of watermark bits on the encoding resilience.

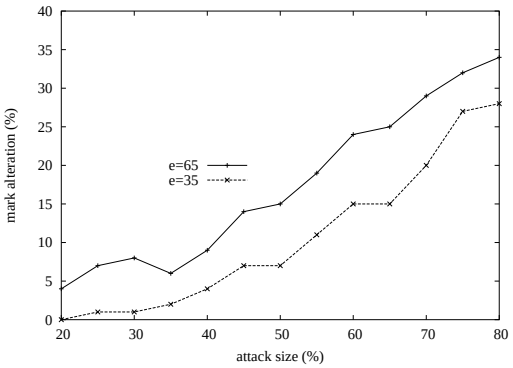

Figure 4. The watermark degrades gracefully with increasing attack size ($e = 65$).

5 Experiments

We implemented a Java proof-of-concept of the watermarking algorithm and deployed it on categorical attributes in the Wal-Mart Sales Database. The Wal-Mart Sales Database contains most of the information regarding item sales in Wal-Mart stores nationwide. In the following we present some of our experiments in watermarking categorical attributes within this database. Our experimental setup included access to the 4 TBytes Wal-mart data, (formerly) hosted on a NCR Teradata machine, one 1.6GHz CPU Linux box with Sun JDK 1.4 and 384MB RAM. The amount of data available is enormous. For example, the *ItemScan* relation contains over 840 million tuples. For testing purposes, we deployed our algorithm on a randomly selected subset of size equal to a small percentage of the original data size (e.g. just a maximum of 141000 tuples for relation

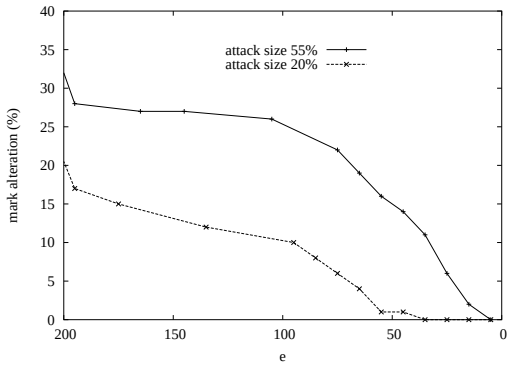

Figure 5. More available bandwidth (decreasing e) results in a higher attack resilience.

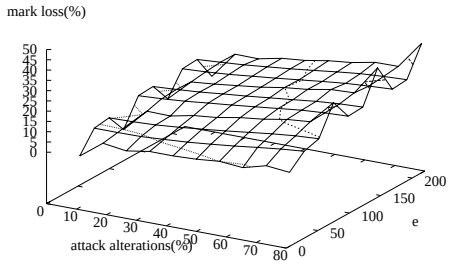

Figure 6. The watermark alteration surface with varying c and attack size a. Note the lower-left to upper-right tilt.

UnivClassTables.ItemScan). The relational schema included the attributes:

```
Visit_Nbr INTEGER PRIMARY KEY,
Item_Nbr INTEGER NOT NULL
```

To illustrate and test our watermarking algorithm we choose Item_Nbr, a categorical attribute, uniquely identifying a finite set of products. The watermark considered was 10 bits long, all the presented data is the result of an averaging process with 15 passes (each seeded with a different key), aimed at smoothing out data-dependent biases and singularities.

In the first experiment we analyzed the behavior of the embedded watermark in the presence of massive data alterations. As the attack size grows (random alterations to the data), the watermark distortion increases. The error correction mechanism (majority voting in this case) does a good job in error recovery. This is particularly so in the case of random alterations to the underlying data, the only available data altering attack option as discussed in Section 4.4. Figure 4, depicts this phenomena for two values of e.

The next experiment aims at exploring the relationship between the amount of alterations required in the watermarking phase and a minimum guaranteed watermark resilience. It can be seen in Figure 5 that as e increases (decreasing number of encoding alterations) the vulnerability to random alteration attacks increases accordingly. This illustrates the trade-off between the requirement to be resilient and the preservation of data quality (e.g. fewer alterations).

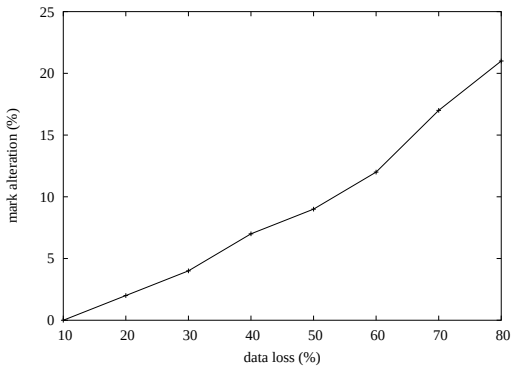

Figure 7. The watermark degrades almost linearly with increasing data loss.

Figure 6 represents the composite surface for both experiments.

An experiment analyzing resilience to data loss is depicted in Figure 7. We observe here the compensating effect of error correction. Compared to data alteration attacks, the watermark survives even better with respect to the attack size (in this case loss of data).

6 Conclusions

In this paper we defined the problem of watermarking categorical data. We proposed a solution and analyzed it both in theory and in practice. We

outlined a set of extensions (e.g. an alternative for occurrence frequency encoding to survive extreme vertical partitioning attacks) and discussed main associated attacks and challenges. We implemented a proof-of-concept for our algorithm and deployed it in experiments on real Wal-Mart sales data. Our method proves (experimentally and by analysis) to be extremely resilient to both alteration and data loss attacks, for example tolerating up to 80% data loss with a watermark alteration of only 25%.

Various issues remain to be explored. Additive watermark attacks need to be analyzed and handled. Also, while the concept of on-the-fly quality assessment (see Section 4.1) has a good potential to function well, as confirmed also by experiments in [12], another interesting avenue for further research would be to augment the encoding method with direct awareness of semantic consistency (e.g. classification and association rules). This would likely result in an increase in available encoding bandwidth, thus in a higher encoding resilience. One idea would be to define a generic language (possibly subset of SQL) able to naturally express such constraints and their propagation at embedding time.

Acknowledgments

We would like to thank Murat Kantarcioglu for the useful insights and discussions.

References

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

[2] Christian Collberg and Clark Thomborson. Software watermarking: Models and dynamic embeddings. In *Principles of Programming Languages*, San Antonio, TX, January 1999.

[3] Ingemar Cox, Jeffrey Bloom, and Matthew Miller. Digital watermarking. In *Digital Watermarking*. Morgan Kaufmann, 2001.

[4] Stefan Katzenbeisser (editor) and Fabien Petitcolas (editor). Information hiding techniques for steganography and digital watermarking. In *Information Hiding Techniques for Steganography and Digital Watermarking*. Artech House, 2001.

[5] D. Gross-Amblard. Query-preserving watermarking of relational databases and xml documents. In *Proc. Nineteenth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 2003.

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

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

[8] Sheldon Ross. A first course in probability. In *A First Course in Probability*. Prentice Hall, 2001.

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

[10] Radu Sion, Mikhail Atallah, and Sunil Prabhakar. On watermarking numeric sets. In *Proceedings of IWDW 2002, Lecture Notes in Computer Science, CERIAS TR 2001-60*. Springer-Verlag, 2002.

[11] Radu Sion, Mikhail Atallah, and Sunil Prabhakar. Power: Metrics for evaluating watermarking algorithms. In *Proceedings of IEEE ITCC 2002, CERIAS TR 2001-55*. IEEE Computer Society Press, 2002.

[12] Radu Sion, Mikhail Attalah, and Sunil Prabhakar. Rights protection for relational data. In *CERIAS-TR 2002-28, Proceedings of ACM SIGMOD*, 2003.