Journal of Data Mining and Management

Volume 1 Issue 3

Rights Protection for Relational Data Using Least Significant Bit Method

R. Prabhu[1], S. Rajesh[2]

[1]Department of Computer Science & Engineering and Technology, AAA College of Engineering and Technology, Sivakasi, India

[2]Department of Information Technology, Mepco Schlenk Engineering College, Sivakasi, India

E-mail: prabhu_24_1985@yahoo.co.in

Abstract

A solution for computer database content rights protection through watermarking. Rights protection for relative information is of ever-increasing interest, particularly considering areas wherever sensitive, valuable content is to be outsourced. A decent example could be a data processing application, wherever, information is sold in items to parties specialized in mining it. Totally different avenues are on the market, every with its own benefits and disadvantages. Social control by legal suggests that is sometimes ineffective in preventing thievery of proprietary works, unless increased by a digital counterpart, for instance, watermarking. Whereas, having the ability to handle higher level linguistics constraints, like classification preservation, our resolution additionally addresses necessary attacks, like set choice and random and linear information changes. We introduce wmdb., a proof-of-concept implementation and its application to real-life information, namely, in watermarking the outsourced Wal-Mart sales information that we have on the market at our institute.*

Keywords: *Rights protection, relational data, watermarking, information hiding*

INTRODUCTION

The main purpose of Digital Watermarking is to shield a definite content from unauthorized duplication and distribution by facultative demonstrable possession over the content. It is historically relied upon the provision of an oversized noise domain at intervals that the item may be altered whereas holding its essential properties. As an example, the smallest amount

vital bits of image pixels may be haphazardly altered with very little impact on the visual quality of the image (as perceived by a human). In fact, abundant of the "bandwidth" for inserting watermarks (such as within the least vital bits) is owing to the shortcoming of human sensory system (especially sight and hearing) to observe bound changes. A lot of recently, the main target of watermarking for digital rights protection is shifting toward totally different knowledge sorts like text, software, and algorithms. Since these knowledge sorts have terribly well-defined linguistics (as compared to those of pictures, video, or music) and will be designed for machine consumption, the identification of the out there "bandwidth" for watermarking is as vital a challenge because the algorithms for inserting the watermarks themselves. A challenge of watermarking is to insert an indelible mark in the object such that 1) the insertion of the mark does not destroy the value of the object (i.e., the object is still useful for the intended purpose) and 2) it is difficult for an adversary to remove or alter the mark beyond detection without destroying the value of the object. Clearly, the notion of value or utility of the object is central to the watermarking process. This 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 malie in the actual or the relative values of the numbers, or in the distribution (e.g., normal with a certain mean).

Although, a substantial quantity of try has been endowed within the downside of watermarking multimedia system information (images, video, and audio), there is comparatively very little work on watermarking different varieties of information [1, 2]. Recent work has addressed the issues of software system watermarking and tongue watermarking. Here, we tend to study the problem of watermarking numeric relative content. Protective rights over outsourced relative information are of ever-increasing interest, particularly considering areas wherever sensitive, valuable information is to be outsourced. Sensible examples are data processing applications (e.g., Wal-Mart sales information, oil drilling information, monetary information, etc.), wherever, a group of information is sometimes produced/collected by an information collector then sold-out in items to parties specialised in mining that data. Given the character of most of the information, it is onerous to associate rights of the creator over it. Watermarking will be accustomed solve this issue [3].

An important point about watermarking should be
An important point about watermarking should be
noted. By its very nature, a watermark modifies the item being watermarked. If the object 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 would be affected by even a minor change to any portion of it. It is, therefore, necessary that the intended purpose of the data to be

preserved is identified during the watermarking process. Whereas, extensive research has focused on various aspects of DBMS security, including access control techniques as well as data security issues, little has been done to secure proof of rights over relational data. Only one related simultaneously published effort is available for comparison numerous fundamental differences distinguish our results from this effort [4, 5].

In this paper, we explore the issue of securing valuable outsourced data through watermarking, enabling court proofs assessing proper rights over the content. Thus, the main contributions of the present work include:

1. A resilient watermarking method for relational data,
2. A technique for enabling user-level runtime control over properties that are to be preserved as well as the degree of change introduced,
3. a complete, user-friendly implementation for numeric relational data, and
4. The deployment of the implementation on real data, in watermarking the Wal-Mart Sales Database and the analysis thereof.

Our solution starts by receiving as user input a reference to the relational data to be rights-protected, a watermark to be embedded as a copyright proof, a secret key used to protect the embedding, and a set of data quality constraints be preserved in the result. It then proceeds to watermark the data while continuously assessing data quality, potentially backtracking, and rolling back undesirable alterations that do not preserdat quality. Watermark embedding is composed of two main parts: In the first stage, the input data set is securely partitioned into subsets of items; the second stage then encodes one bit of the watermark into each subset. If more subsets (than watermark bits) are available, error correction is deployed to result in an increasingly resilient embedding. The algorithms introduced here prove to be resilient important classes of attacks, including subset selection, linear data changes, and random alterations.

CHALLENGES

While analysis associated with the difficulty of embedding info into a group of numbers associate degree be found (sometimes implicitly) in several frameworks, related to varied info concealment techniques (e.g., frequency domain embedding, DCT, and ripple watermarking), relational information presents a special set of challenges and associated constraints [6]. These challenges area unit novel and directly associated with the specifics of the domain, namely, massive sets of things organized in a very relative framework, with associated linguistics to be preserved. This can be not the case for transmission (mostly time-series kind of) information, wherever, linguistics area unit related to the info stream solely at a way higher composite level. As an example, in a very multi-megabit audio channel of stories broadcast, the linguistics to be protected area unit seemingly to be within the broadcast speech text instead of directly within the underlying audio stream bits; so, a

basically totally different and broader noise band becomes accessible for watermark embedding, and with it totally different (possibly less accurate) cryptography and analysis ways. In contrast, the low noise information measure of major relative framework information uses (e.g., information mining) need a special approach, taking a lot of careful consider the particular tolerated changes on the given information [7].

Whereas, within the multimedia system case, the info quality model is sometimes at the best fuzzy as a result of the Einstein's theory of relativity of any model of human perception, one resolution here is to outline the noise channel expressly as a part of the watermarking resolution, in terms of needed client constraints to be preserved on the ultimate information. At watermarking time, information quality is often unceasingly assessed as AN intrinsic a part of the marking algorithmic program in itself. During this respect, we are able to claim that, as against different watermarking algorithms in varied domains (e.g., image watermarking), we have a tendency to maintain 100% of the associated information price with relation to a collection of given needed information "goodness" constraints. We believe this is an essential part of any watermarking application in this low-noise, high-fragility domain of relational data, especially considering data mining issues, such as classification and JOIN results preservation.

Additionally, the watermark coding methodology has to feature a style suited to the new constraints, namely, the flexibility to survive a most level of attacks and, at a similar time, accommodate the existence of needed knowledge "usability" conditions to be glad by the result. Our algorithmic program, deploying means that for knowledge distribution manipulation and coding the particular data in distribution properties of the information instead of directly into the information itself, is best fitted to its purpose, and nearly optimally therefore. For, whereas permitting Associate in Nursing adjustable degree of freedom in alteration points choice, it provides at a similar time a astonishingly high level of resilience as proved by our in depth validation experiments.

Available Bandwidth

An important first step in inserting a watermark into a relational database (and thereby altering it), is to identify changes that are acceptable. As was mentioned earlier, the acceptable nature and level of change is dependent upon the application for which the data is to be used. With respect to particular data uses and metrics of quality, it is of utmost importance that the watermarking process not interferes with the final data consumer requirements. This is why these requirements need to be considered as an integral part of the watermarking process, providing a feedback loop, in assessing the quality of the final result.

To the best of our knowledge, our solution is the first to recognize the importance of these essential desiderata and provide a direct algorithm for it. In the following, we define a functionality that will enable us to determine the watermarking result as being valuable and valid, within permitted/guaranteed error bounds. The available

"bandwidth" for inserting the bits of the watermark text is, therefore, not defined directly. Instead, we define allowable distortion bounds for the input data in terms of consumer-defined metrics. If the water-marked data satisfies the metrics, then the insertion of watermark is considered to be successful. This quality assessment mechanism is part of the marking process.

Example: One simple but relevant example is the maximum allowable mean squared error case, in which the usability metrics are defined in terms of mean squared error tolerances as

$$ðs_i \_ v_iÞ^2 < t_i \quad 8i ¼ 1; \ldots ; n$$

and $^P ðs_i \_ v_iÞ^2 < t_{max}$, where **S** ¼ $fs_1; :::; s_ng \_$ IR is the data to be watermarked, **V** ¼ $fv_1; \ldots ; v_ng$ is the result, **T** ¼ $ft_1; \ldots ; t_ng \_$ IR, and t_{max} 2 IR define the guaran-teed error bounds at data distribution time. In other words, **T** defines the allowable distortions for individual elements in terms of mean squared error (MSE) and t_{max} the overall permissible MS

Database Semantics

Specifying solely allowable modification limits on individual values associate degreed presumably an overall limit, fails to capture necessary linguistics options related to knowledge-especially if the data is structured. Consider, as an example, age data. Whereas, a little modification to the age values could also be acceptable, it should be vital that people that are younger than twenty one stay thus even when watermarking if the information are would not to verify behavior patterns for under-age drinking. Similarly, if an equivalent knowledge were to be used for distinguishing legal voters, the cut-off would be eighteen years. Moreover, for a few different application, it should be necessary that the relative ages (in terms of that one is younger) not modification. Different samples of constraints embody:

1. Uniqueness-each value must be unique;
2. Scale-the ratio between any two number before and after the change must remain the same; and
3. Classification-the objects must remain in the same class (defined by a range of values) before and after the watermarking.

As is clear from the above examples, simple bounds on the change of numerical values are often not enough.

Structured Data

Structured collections, for example, a collection of relations, present further constraints that must be adhered to by the watermarking algorithm. Consider a data warehouse organized using a standard Star schema with a fact table and several dimension tables. It is important that the key relationships be preserved by the watermarking algorithm. This is similar to the "Cascade on update" option for foreign keys in SQL and ensures that tuples that join before watermarking also join after water-marking. This requires that the new value for any attribute should be unique after the watermarking process. In other words, we want to preserve the relationship between the various tables. More generally, the relationship could be expressed in terms of an arbitrary join condition, not just a natural join. In addition to relationships between tuples, relational data may have constraints within tuples. For example, if a relation contains the start

and end times of a Web interaction, it is important that each tuple satisfies the condition that the end time be later than the start time.

Also, an adversary attempting to destroy a watermark becomes much more effective if he can identify the values in which the watermark has been embedded. In addition to specifying properties of the data that should be preserved for usability, constraints can be used to prevent easy detection of watermark locations. For example, a tuple with a start time later than its corresponding end time, or a customer with an age less than 12 years is very likely to be detected as resulting from watermarking.

Model of the Adversary

In order to be effective, the watermarking technique must be able to survive a wide variety of attacks. These attacks may be malicious with the explicit intent of removing the watermark, or may be the result of normal use of the data by the intended user.

Subset Selection

The attacker (Mallory) can randomly select and use a subset of the original data set that might still provide value for its intended purpose (subtractive attack).

Subset Addition

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

Subset Alteration

Altering a subset of the items in the original data set such that there is still value associated with the resulting set. A special case needs to be outlined here, namely, (A3.a) a linear transformation performed uniformly to all of the items. This is of particular interest as such a transformation preserves many data mining related properties of the data, while actually altering it considerably, making it necessary to provide resilience against it. Given the attacks above, several properties of a successful solution surface. For immunity against A1, the water-mark has to be embedded in overall collection properties that survive subset selection (e.g., confidence intervals). If the assumption is made that the attack alterations do not destroy the value of the data, then A3 should be defeatable by embedding the primitive mark in resilient global data properties. As a special case, A3.a can be defeated by a preliminary normalization step in which a common divider to all the items is first identified and applied. For a given item X, for notation purposes, we are going to denote this "normalized" version of it by NORMðXÞ. 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 Mallory).

Subset Recovery

Another interesting requirement is the ability to "recognize" all (or at least most) of the collection items before and after watermarking and/or an attack. That is, how do we "recognize" an item and its corresponding subset after it has been changed slightly?

SIMPLIFIED PROBLEM: NUMERIC COLLECTIONS

This section deals with the foundations of a primitive numeric collection watermarking procedure that will be later deployed as a subroutine in the main watermarking algorithm. S be a set of n real numbers **S** ¼ fs$_1$; . . . ; s$_n$g _ IR. Then, the general simplified problem of watermarking the set **S** can be defined as the problem of finding a transforma-tion from **S** to another item set **V**, such that, given all imposed usability metrics sets **G** ¼ [G$_i$ for any and all subsets S$_i$ _ **S**, that hold for **S**, then, after the transformation yields **V**, the metrics should hold also for **V**.[1] we call **V** the "water-marked" version of **S**. Thus, **V** ¼ fv$_1$; . . . ; v$_n$g _ IR is the result of watermarking **S** by minor alterations to its content. Let a string of bits w of size m << n is the desired watermark to be embedded into the data (jwj ¼ m). We will use the notation w$_i$ to denote the ith bit of w.

But, how much of a change is to be allowed to the content? For a numeric collection, a natural starting point for defining the allowed change is to specify an absolute (or relative) change in value. For example, each value may be altered by no more than 0.0005 or 0.02 percent. Moreover, a bound on the cummulative change may be specified. Our solution for the simplified problem consists of several steps. First, we deploy a resilient method for item labeling, enabling the required ability to "recognize" initial items at watermarking detection time (i.e., after watermarking and/ or attacks). In the next step, we ensure attack survivability by "amplifying" the power of a given primitive water-marking method. The amplification effect is achieved by deploying secrets in the process of selecting the subsets to become input for the final stage, in which a primitive encoding method is deployed. Before watermarking, e.g., being identified with a certain label L, then, hopefully, at watermark detection time the same item is identified with the same label L or a known mapping to the new label. More generally, we would like to be able to identify a majority of the initial elements of a subset after watermarking and/or attacks. As we will see, our technique is resilient to "missing" a small number of items.

Our solution relies on lexicographically sorting the things within the assortment, sorting occurring supported a unidirectional, in secret keyed, cryptographical hash of the set of most important bits (MSB) of the normalized version of the things. The key unidirectional hashing ensures that Mallory cannot probably confirm the ordering. Within the next step, set "chunks" of the things area unit elite supported this secret ordering. Chunk-boundaries ("subset markers") area unit then computed and hold on for detection time More formally, given a collection of items as above, **S** ¼ fs$_1$; . . . ; s$_n$g _ IR, and a secret "sorting key" k$_s$, we induce a secret ordering on it by sorting according to a cryptographic keyed hash of the most significant bits of the normalized items. Thus, we have:

$$\text{index}(s_i) = H(k_s; \text{MSB}(\text{NORM}(s_i), k_s))$$

The MSB space here is assumed to be a domain where minor changes on the collection items (changes that still satisfy the given required

usability metrics) have a minimal impact on the MSB labels. This is true in many cases (as usually the usability metrics are related to preserving the "important" parts of the original data). If not suitable, a different labeling space can be envisioned, one where, as above, minor changes on the collection items has a minimal impact.

Note: In the relational data framework, the existence of a primary key associated with the given attribute to be watermarked can make it easier to impose a secret sorting

Solution Summary

A summary of the solution for the simplified problem reads as follows:

Encoding Phase: (E.1)

Select a maximal number of unique, nonintersecting (see below) subsets of the original set, using a set of secrets, as described in Section 3.3. (E.2) For each considered subset, (E.2.1) embed a watermark bit into it using the encoding convention in Section 3.3 and (E.2.2) check for data usability bounds. If usability bounds are exceeded, (E.2.3) retry different encoding parameter variations or, if still no success, (E.2.3a) try to mark the subset as invalid (i.e., see encoding convention in Section 3.3), or if still no success, (E.2.4) ignore the current set.[2] We repeat step E.2 until no more subsets are available for encoding. This results in multiple embeddings in the data.

Decoding Phase: (D.1)

Using the secrets from step E.1, recover a majority of the subsets considered in E.1, (or all if no attacks were performed on the data). (D.2) For each considered subset, using the encoding convention in Section 3.3, recover the embedded bit value and reconstruct watermarks. (D.3) The result of D.2 is a set of copies of the same watermark with various potential errors. This last step uses a set of error correcting mechanisms (e.g., majority voting schemes) to recover the highest likelihood initial mark.

Selecting Subsets

Watermarking a collection of data items requiresth ability to "recognize" (i.e., rediscover, at detection time) most of the items before and after watermarking and/or a security attack. In other words, if an item was accessed/modified

1. In other words, if **G** is given and holds for the initial input data, **S**, then **G** should also hold for the resulting data **V**.
2. This leaves an invalid watermark bit encoded in the data that will be corrected by the deployed error correcting mechanisms (e.g., majority voting) at extraction time.

Amplifying Watermark Power

Current watermarking algorithms draw most of their court-persuasion power from a secret that controlled water-mark embedding (i.e., watermarking key). Much of the attack immunity associated with a watermarking algorithm is based on this key and its level of secrecy. Given a weak partial marking technique (e.g., (re)setting a bit), a strong marking method can be derived by a method of "mark amplification"—repeatedly applying the weak technique in a keyed fashion on different parts of the data being watermarked.

Generic Solution

Let **K** ¼ fk$_1$; . . . ; k$_m$g be a set of m keys of n bits each. We define

$$S_i \text{ ¼ } fs_j \text{ 2 } \mathbf{S}jðk_i Þ_{bitj} \text{ ¼ } 1g; \text{ i ¼ } 1; \ldots ; \text{ m:}$$

In other words, each S_i _ **S** is defined by selecting a subset of **S** fully determined by its corresponding key k_i 2 **K**.

In most scenarios, watermarking outsourced relational content happens only once, at outsourcing time. The main purpose of watermarking in this framework is rights-protection and/or traitor tracing through fingerprinting. Thus, there seems to be little to be gained from an ability to watermark at runtime, in the presence of updates. More-over, because watermarking inherently alters the data, it is unreasonable to assume that a certain party would keep an altered (i.e., watermarked) copy of the data as replacement for the original.

Nevertheless, our solution naturally supports on-the-fly watermarking, especially in the presence of updates. Let us analyze several different update scenarios:

1. Updates that add fresh tuples to the already water-marked data set,
2. Updates that remove tuples from the already ter-marked data, and
3. Updates that alter existing tuples.

In each of the cases, we assume that the watermarking mechanism runs continuously as a dormant process and is notified for each update,

```
detect(attribute, wm_key, db_primary_key, subset_size, v_false, v_true, c, embedding_map[], subset_boundaries[])
    sorted_attribute ← sort_on_normalized_crypto_hash(wm_key,db_primary_key,wm_key)
    read_pipe ← null
    do { tuple ← next_tuple(sorted_attribute) }
    until (exists idx such that (subset_boundaries[idx] == tuple))
    current_subset ← idx
    while (not(sorted_attribute.empty())) do
        do {
            tuple ← next_tuple(sorted_attribute)
            read_pipe = read_pipe.append(tuple)
        } until (exists idx such that (subset_boundaries[idx] == tuple))
        subset_bin ← (at most subset_size elements from read_pipe, not including last read element)
        read_pipe.remove_all_remaining_elements_but_last_read()
        if (embedding_map[current_subset]) then
            mark_data[current_subset] ← decode (subset_bin, v_false, v_true, confidence)
            if (mark_data[current_subset] != DECODING_ERROR)
                    then detection_map[current_subset] ← true
        current_subset ← idx
    return mark_data, detection_map
```

Fig. 1: *Watermark Detection Algorithm (version using Subset Markers and Detection Maps Shown*

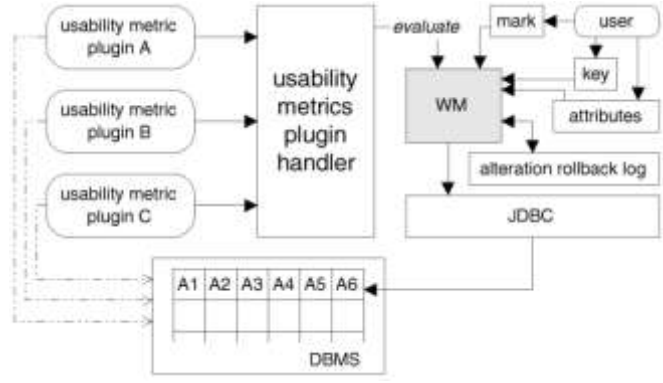
HðK^0; keyÞÞ mod e ¼ 0) is performed on a MSB portion of the primary key K, i.e., K^0 ¼ MSBðKÞ. This is to be subject to further investigation, hopefully resulting in primary key independence.

EXPERIMENTAL RESULTS

This section presents our implementation and the experimental results of watermarking real-life, commercial, data, namely, the Wal-Mart Sales relational database.

*Implementation: wmdb.**

wmdb.* is our test-bed implementation of the algorithms presented in this paper. It is written using the Java language and uses the JDBC API in accessing the data. The package receives as input a watermark to be embedded, a secret key to be used for embedding, a set of relations/attributes to consider in watermarking as well as a set of external usability plugin modules. The role of the plugin modules is to allow user-defined query metrics to be deployed and queried at runtime without recompilation and/or software restart. Once usability metrics are defined and all other parameters are in place, the watermarking module initiates the process of watermarking. An undo/rollback log is kept for each atomic step performed (i.e., 1-bit encoding) until data usability is assessed and confirmed (by querying the currently active usability plugins). This allows for rollbacks in the case when data quality is not preserved by the current atomic operation. Watermark recovery takes as input the watermarking key used in embedding, the set of attributes known to contain the watermark as well as various other encoding specific parameters.



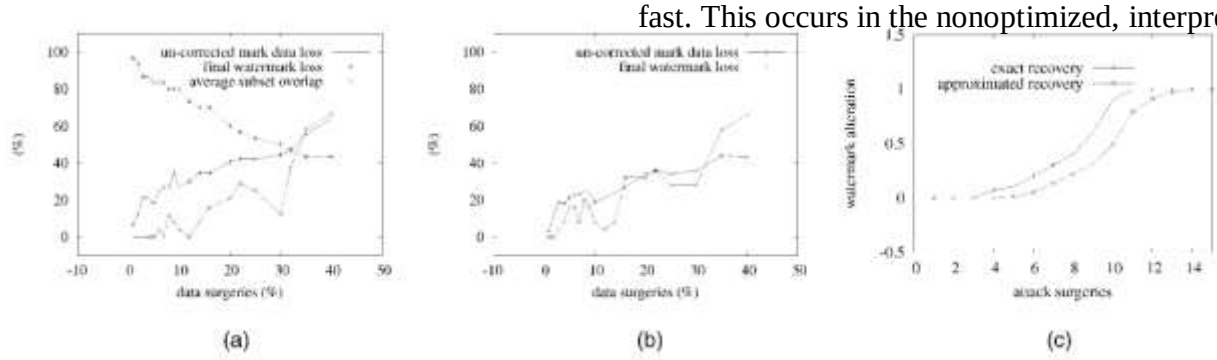


Fig. 2: *Resilience to Data Surgeries: (a) Uniform Distribution, (b) Normal Distribution, and (c) Single Subset (1-bit) Encoding.*

Our experimental setup included access to the 4 TBytes Wal-mart data, (formerly) hosted on a NCR Teradata machine, one 1.8GHz CPU Linux box with Sun JDK 1.4 and 384MB RAM. The amount of data available is enormous. For example, the ItemScan relation contains more than 840 million tuples. For testing purposes, we deployed our algorithm on a randomly selected subset of the original data (e.g., just a maximum of 141,075 tuples for relation UnivClassTables.Store Visits).

We assessed computation times and observed an intuitively (according to the O(n) nature of the algorithm) linear behavior, directly proportional with the input data size. Given the setup described above, in single-user mode, with a local database we obtained an average of around 350-400 tuples/second for watermark embedding, while detection turned out to be approximatively twice as fast. This occurs in the nonoptimized, interpreted Java proof-of-concept implementation. We expect major speedups (orders of magnitude) in a real-life deployment version. In the following, we present experiments involving attacks (data loss, data alterations, linear changes, data resorting) as well as the evaluation of the available bandwidth in the presence of different data goodness metrics (tolerable absolute change and data classification preservation).

Data Loss Attacks ("Surgeries")

In this attack scenario, we study the distortion of the watermark as the input data is subjected to gradually increasing levels of data loss. In Figure 2c, the analysis is performed repeatedly for single bit encoding using the "confidence-violators" encoding method outlined. The results are then averaged over multiple runs. The "confidence-violators" primitive set encoding proves to be resilient to a consider-able amount of randomly occurring uniformly distributed surgeries (i.e., item

removals by Mallory, with no extra knowledge) before watermark alterations occur. Even then, there exists the ability to "trace" or approximate the original watermark to a certain degree (i.e., by trying to infer the original mark value from an invalid set). The set size considered was 35, experiments were performed on 30 different sets of close to normally distributed data. Other parameters for the experiment include:

$$v_{false} ¼ 5\%; v_{true} ¼ 9\%; c ¼ 88\%:$$

The average behavior is plotted in the graphs. Up to 25 percent and above data loss was tolerated easily by the tested data, before mark alteration (i.e., bit-flip) occurred.

Figures 2a and b depict more complex scenarios in which a real multibit watermark is embedded into a larger data set (both (a) uniform and (b) normal distributions in Figure 2 were considered). The input data contained 8,000 tuples, subset size was 30, and the considered watermark was 12 bits long. Other parameters: v_{false} ¼ 35%; c ¼ 85%. This set is then subjected to various degrees of data loss and the watermark distortion is observed. The encoding method again proves to be surprisingly resilient by allowing up to 45-50 percent data loss while still 40-45 percent of the watermark survives. Also, in Figure 2a, as data alteration increases, the subset (i.e., secretly selected for encoding 1-bit) overlap (i.e., the "resemblance" to the original content, the number of same elements in resulting subsets) degrades.

Note on Data Dependency in Figures. Some of the figures presented in this section feature "spikes." This is a result of the adaptive data-dependent nature of the encoding. Different input data reacts differently to data surgeries (for example) and feature slightly varying behavior at distinct points. Averaging over multiple inputs provides a solution for this issue. Nevertheless, we believe that, while it might soften the spikes, it would also (arguably) tone down distinct features for a given data set, features that interrelate figures. Instead of focusing on local variations, the figures should be interpreted as an illustrative sample of the global governing trends.

Data Alteration Attacks (Epsilon-Attack)

Presented with the watermarked data Mallory is faced with two contradictory tasks: preserving the inherent value of the data while, at the same time, removing the hidden water-mark. Given no knowledge of the secret watermarking key nor of the original data, the only available choice is to attempt (minor) random data modifications in the hope that, at some point, the watermark will be destroyed. Because the original data is unknown (thus, also the current watermark-related distortion is unknown), it is impossible for Mallory to determine the real "minority" of changes he/she performs.

In other words, because of the goal of preserving the data value, Mallory cannot afford to perform significant change to the data.

In this experiment, we analyze the sensitivity of our watermarking scheme to randomly occurring changes, as a direct measure for watermark resilience. To do this, we define a transformation that modifies a percentage _ of the input data within certain bounds defined by two variables _ and _.

We called this transformation epsilon-attack. Epsilon-attacks can model any uninformed, random alteration—the only available attack alternative. A normal epsilon-attack modifies roughly ⁻₂ percent of the input tuples by multiplication with ð1 þ _ þ _Þ and the other ⁻₂ percent by multiplication with ð1 þ _ _ _Þ. A uniform altering epsilon-attack modifies _ percent of the input tuples by multi-plication with a uniformly distributed value in the ð1 þ _ _ _; 1 þ _ þ _Þ interval.

A comparison is made between the case of uniformly distributed (i.e., values are altered randomly between 100 and 120 percent of their original value) and fixed alterations (i.e., values are increased by exactly 20 percent). In the case of fixed alterations, the behavior demonstrates the effectiveness of the encoding convention: As more and more of the tuples are altered linearly, the data distribution comes increasingly closer to the original shape. For example, when 100 percent of the data is modified consistently and linearly, the mark data suffers only 6 percent alterations. A peak around 50 percent data alterations can be observed indicating that an attack changing roughly 50 percent of the data might have a greater chance of success. This is also intuitively so (in the case of randomly distributed alterations) as a maximal change in distribution is expected naturally when close to half of the data set is skewed in the same "direction" (by addition or subtraction).

Parameter_models the average of the data alteration distribution while _ controls its width. Naturally, a zero-average epsilon-attack (_ ¼ 0) is a transformation that modifies roughly ⁻₂ percent of the input tuples by multi-plication with ð1 þ _Þ and the other ⁻₂ percent by multi-plication with ð1 _ _Þ.

The behavior of our encoding algorithm to this type of attack. This is particularly intriguing as it clearly reveals a special feature of the watermarking method: Since the bit-encoding convention relies on altering the actual distribution of the data, it survives gracefully to any distribution-preserving transformation. Randomly changing the data, while it can definitely damage the watermark (e.g., especially when altering around 50 percent of the data), proves to be, to a certain extent, distribution-preserving. A zero-average epsilon-attack is survived very well. For example, altering 80 percent of the input data within 20 percent of the original values still yields over 70 percent of the watermark.

Note: One could argue that, after all, if the watermark encoding relies too much on the distribution of the data, one successful attack could be the one that alters exactly this distribution. But, this is not possible, as the power of the watermarking scheme lies not only in the distribution itself but also in the secrecy of the encoding subsets. In other words, where the bits are encoded (i.e., subsets) as important as how. Altering global data characteristics would not only destroy probably much of the value of the data but, as shown above, achieve little in destroying the watermark.

As the percentage of tuples altered and the alteration factor goes up, so does the watermark distortion. Nevertheless, it turns out to be surprisingly resilient. For example, altering 100 percent of the data within 1 percent of the original values can yield a distortion as low as 5-6 percent

in the resulting watermark. The watermark distortion increases with increasing (b) alteration factor or (c) percentage of data presents a comparison between the curves corresponding to the alteration of 40 percent of the tuples versus 80 percent of the tuples. Naturally, the curve for the higher tuples percentage appears "above." A comparison is made between curves for the alteration factor 1 or 5 percent. The higher alteration curve is intuitively "above." Note that the curves are slightly increasing but not very steep: Mark alteration is less dependent on the percentage of data altered than on the alteration factor. Thus, the watermarking scheme proves a natural resilience to uninformed attacks (modeled by epsilon-attack transformations).

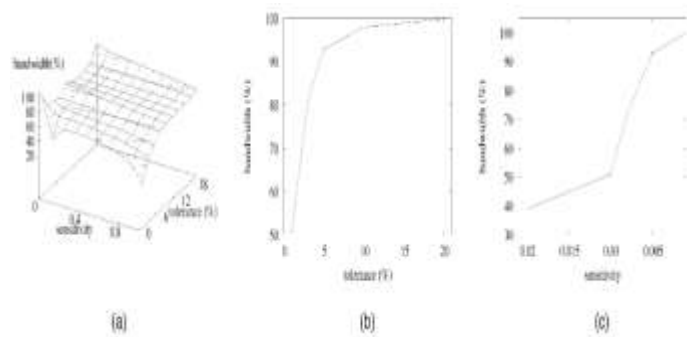
Data Quality (Goodness) Metrics

Here, we analyze the impact of data goodness preservation on the available watermark encoding bandwidth. Intuitively, the more restrictive data constraints one imposes, the less available bandwidth there is, as allowable data changes are directly impacted.

In the following, we present two results. The first analyzed goodness metric is a commonly considered one, namely, upper bounds imposed on the total and local tolerable absolute change (i.e., of the new data with respect to the original).

Note: An identical experimental result was obtained for a related metric, the maximum allowable mean squared error.

As data goodness metrics are increasingly restrictive, the available bandwidth (guaranteeing higher resilience) decreases. In the illustrated experiment, the allowed absolute change in the watermarked data (i.e., from the original) is decreased gradually (from 0.1 to 0.02 percent) and the decrease in available encoding bandwidth is observed (depicted as a percent of total potential bandwidth). The upper limit (approximately 90 percent) is inherently data imposed and cannot be



a. defining a new suitable mark encoding method marking, S. Katzenbeisser and F. Petitcolas, eds. nd
b. building an algorithmic secure mapping (i.e., mark amplification) from a simple encoding method to a more complex watermarking algorithm, and

applied the concept to numeric relational databases.

We thus provided a solution for resiliently watermarking relational databases. We also developed a proof of concept implementation of our algorithms under the form of a Java software package, wmdb.* which we then used to water-

mark a commercial database, extensively used for data-mining in the area of customer trends and buying patterns. In upcoming research, we are investigating new, nonnumeric encoding domains. Furthermore, a model of attacks in this new domain needs to be devised and a more detailed attack analysis performed. A full-fledged commercial watermarking application could be derived from our proof-of-concept software.

ACKNOWLEDGMENT

Portions of this work were supported by Grants EIA-9903545, IIS-0325345, IIS-0219560, IIS-0312357, IIS-9985019, IIS-9972883, and IIS-0242421 from the US National Science Foundation, Contract N00014-02-1-0364 from the US Office of Naval Research, by sponsors of the Center for Education and Research in Information Assurance and Security, and by Purdue Discovery Park's e-enterprise Center.

REFERENCES

1. M.J. Atallah, S.S. Wagstaff Jr. Watermarking with quadratic residues. *Proc. IS-T/SPIE Conf. Security and Watermarking of Multimedia Contents.* 1999; 3657: 283–288p.
2. M.J. Atallah, V. Raskin, C.F. Hempelmann, et al. Natural language water-marking and tamperproofing. *Proc. Fifth Int'l Information Hiding Workshop*; 2002.
3. E. Bertino, M. Braun, S. Castano, et al. A java-based system for XML data protection. *Proc. IFIP Workshop Database Security*. 2000; 15–26p.
4. E. Bertino, S. Jajodia, P. Samarati. A flexible authorization mechanism for relational data management systems. *ACM Trans. Information Systems*. 1999; 17(2).
5. C. Collberg, C. Thomborson. On the limits of software watermarking. *Technical Report;* 1998.
6. Kiernan, R. Agrawal. Watermarking relational databases. *Proc. 28th Int'l Conf. Very Large Databases VLDB*; 2002.