Rights Protection for Categorical Data

Radu Sion, Mikhail Atallah, and Sunil Prabhakar

Abstract—A novel method of rights protection for categorical data through watermarking is introduced in this paper. New watermark embedding channels are discovered and associated novel watermark encoding algorithms are proposed. While preserving data quality requirements, the introduced solution is designed to survive important attacks, such as subset selection and random alterations. Mark detection is fully "blind" in that it doesn't require the original data, an important characteristic, especially in the case of massive data. Various improvements and alternative encoding methods are proposed and validation experiments on real-life data are performed. Important theoretical bounds including mark vulnerability are analyzed. The method is proved (experimentally and by analysis) to be extremely resilient to both alteration and data loss attacks, for example, tolerating up to 80 percent data loss with a watermark alteration of only 25 percent.

Index Terms—Rights protection, categorical data, relational data, watermarking, information hiding.

1 Introduction

Rights protection for categorical data is important in scenarios where it is sensitive and valuable and about to be outsourced. A good example is a data mining application, where data is sold in pieces to parties specialized in mining it (e.g., sales patterns database, oil drilling data, financial data). Other scenarios involve, for example, online B2B interactions (e.g., airline reservation and scheduling portals) in which data is made available for direct, interactive use (see Fig. 2). Given the nature of these scenarios, it is hard to associate originator rights with the data in use. Information Hiding and Watermarking can be used to solve this issue and provide a tool for resilient Rights Assessment.

1.1 Rights Assessment through Information Hiding

Digital Watermarking as a method of Rights Assessment deploys Information Hiding to conceal an indelible "rights witness" ("rights signature," watermark) within the digital Work to be protected (see Fig. 1). The soundness of such a method relies on the assumption that altering the Work in the process of hiding the mark does not destroy the value of the Work, and that it is difficult for a malicious adversary ("Mallory") to remove or alter the mark beyond detection without destroying the value of the Work. The ability to resist attacks from such an adversary (mostly aiming at removing the embedded watermark) is one of the major concerns in the design of a sound watermarking solution.

But how does the ability to prove rights in court relate to our final desiderata, namely to *protect* those rights? Why not simply publish a digest of the Works to be protected in a newspaper, just before releasing them, enabling us to prove later on in court that at least they were in our possession at the time of publication? In the following, we address these and other related issues.

1.1.1 Rights Protection through Assessment

The ability to prove/assess rights convincingly in court constitutes a deterrent to Mallory. It thus becomes a tool for rights protection if counter-incentives and legal consequences are set high enough. But because information hiding does not provide means of actual access control, the question of rights protection still remains. *How* are rights protected here?

It is intuitive that such a method would only work if the rightful rights-holder (Alice) actually knows about Mallory's misbehavior **and** is able to prove to the court that: 1) Mallory possesses a certain Work X and 2) X contains a "convincing" (e.g., very rare with respect to the space of all considered similar Works) and "relevant" (e.g., a string stating "© by Alice") watermark.

What watermarking does not offer is a direct deterrent. If Alice does not have the knowledge of Mallory's illicit possession of the Work and/or if it is impossible to actually prove this possession in court beyond reasonable doubt, then watermarking cannot be deployed directly to prevent Mallory.

If, however, Information Hiding is aided by additional access control level levers, it can become very effective. For example, if, in order to derive value from the given Work (e.g., watch a video tape), Mallory has to deploy a known mechanism (e.g., use video player), information hiding could be deployed to enable such a proof of possession, as follows. One simple example would involve modifying the video player so as to detect the existence of a watermark and match it with a set of credentials and/or "viewing tickets" (that can be purchased) associated with the player's owner. If no match is found, the tape is simply not played back.

This is just one of many scenarios where watermarking can be deployed in conjunction with other technologies to aid in managing and protecting digital rights. Of course, this scenario is simplistic and relies on the assumption that the cost of reverse engineering this process is far higher than the potential derived illicit gain. However, this is

- R. Sion is with the Department of Computer Sciences, Stony Brook University, Stony Brook, NY 11794. E-mail: sion@cs.stonybrook.edu.
- M. Atallah and S. Prabhakar are with the Department of Computer Sciences, 250 N. University St., Purdue University, West Lafayette, IN 47906-2066. E-mail: {mja, sunil}@cs.purdue.edu.

Manuscript received 7 Jan. 2004; revised 17 Aug. 2004; accepted 23 Dec. 2004; published online 18 May 2005.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0007-0104.

1041-4347/05/$20.00 © 2005 IEEE Published by the IEEE Computer Society

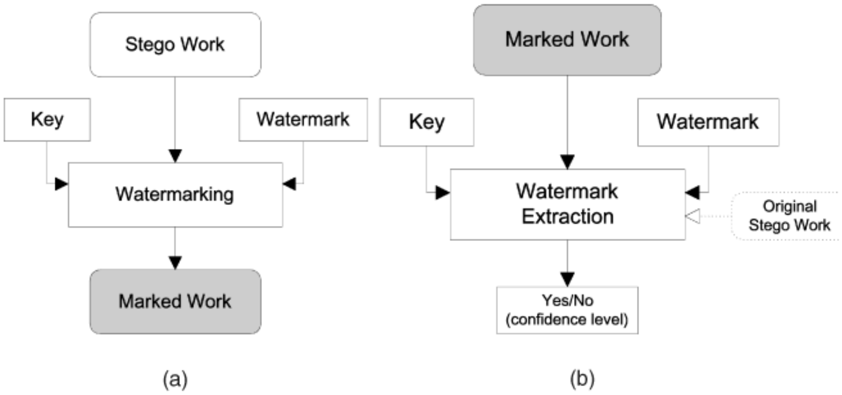


Fig. 1. (a) *Digital Watermarking* conceals an indelible "rights witness" ("rights signature," watermark) within the digital Work to be protected. (b) In court, a detection process is deployed to prove the existence of this "witness" beyond reasonable doubt (confidence level) and, thus, assess ownership.

essential in that it illustrates the game theoretic economic nature at the heart of the watermarking proposition and of information security in general.

Watermarking is a game with two adversaries, Mallory and Alice. At stake lies the value inherent in a certain Work X, over which Alice owns certain rights. When Alice releases X, she deploys watermarking for the purpose of ensuring that one of the following holds:

- She can always prove rights in court over any copy or valuable derivate of X (e.g., segment),
- any existing derivate Y of X, for which she cannot prove rights does not preserve any significant value (derived from the value in X), and
- the cost to produce such an unwatermarked (for which she cannot prove rights) derivate Y of X that is still valuable (with respect to X) is higher than its value.

1.2 Information Hiding versus Newspaper Digests

Apparently, Alice could simply publish a (cryptographic) digest of X in a newspaper, thus being able to at least claim a time stamp of possession of X later on. Why not deploy this as a rights assessment tool instead of information hiding? There are many reasons why it would not work, including: 1) scalability issues associated with the need for a trusted third party (newspaper), 2) the cost of publishing a digest for each released Work, and 3) scenarios when the fact that the Work is watermarked should be kept secret (stealthiness), etc.

Maybe the most important reason is that Mallory can now claim that his ownership of the Work precedes X's publication date and that Alice simply (modified it and) published a digest. It would then be up to the court to decide if Mallory is to be believed or not, which is hardly an encouraging scenario for Alice. This could work if there existed a mechanism for the mandatory publication of digests for each and every valuable Work, which is again probably impractical due to both costs and lack of scalability.

It becomes clear that deploying such aids (digests) as rights assessment tools makes sense only in the case of the Work being of value only unmodified. In other words, if it does not tolerate any changes (without losing its value) and Mallory is caught in possession of an identical copy, Alice can successfully prove in court that she possessed the original at the time of its publication, but she cannot prove more.

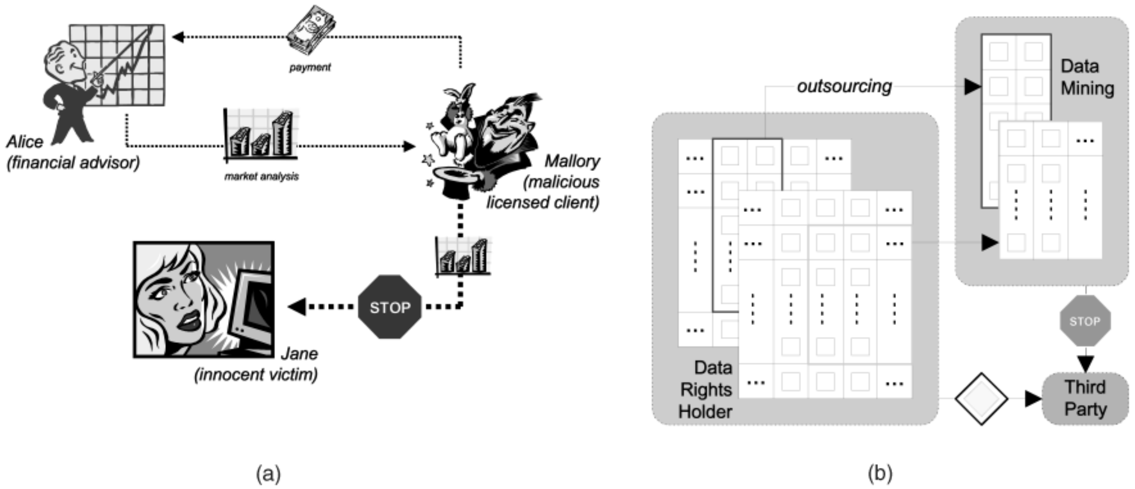


Fig. 2. Rights Assessment is useful when valuable content is to be sold/outsourced to potentially untrusted parties, even if rightfully licensed. (a) Mallory breaches the data license and sells the data or derivatives thereof to Jane, a third party. (b) Data partitions are outsourced (e.g., for data mining).

1.3 Relational Data

While extensive efforts have focused on various aspects of DBMS security, including access control techniques as well as data security issues [1], [2], [3], [10], [12], [13], [14], [16], [17], [18], [19], [21], little has been done to enable the ability to assert rights over outsourced relational data. Notably, Kiernan and Agrawal in [15] and Sion et al. in [25] introduced solutions for rights protection of *numeric* relational content through information hiding.

A natural continuation of these efforts derives from the realization that there are a multitude of applications that operate with categorical data and would benefit from a method of rights protection for such data types. In this paper, we propose and analyze this issue of rights protection for *categorical* relational content.

Main challenges in this new domain derive from the fact that one cannot rely on "small" alterations (such as the ones deployed in the work on numeric data types) to the data in the embedding process. Any alteration can be potentially significant. This discrete characteristic of the data requires the discovery of fundamentally new bandwidth channels and associated encoding algorithms. Additionally, since the associated data types do not have fixed, well-defined semantics (as compared to multimedia) and may be designed for machine ingestion, identifying the available "bandwidth" for watermarking becomes as important as the actual encoding algorithms.

Our solution proves to be resilient to various important classes of attacks, including subset selection and random item(s) alterations. The main contributions of this work include:

1. the proposal and definition of the problem of watermarking categorical data,
2. the discovery and analysis of new watermark embedding channels for relational data with categorical types,
3. the design of novel associated encoding algorithms, and
4. the experimental evaluation thereof.

The paper is structured as follows: In Section 2, we present our main data and adversary models. Section 3 introduces the main solution, outlines alternatives, and discusses the basic algorithm vulnerability to data altering attacks. Section 4 gradually introduces algorithm improvements handling particular scenarios while Section 5 outlines and explores various additional issues. Section 6 presents our experimental setup and results and Section 7 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, which is 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).

Let the number of tuples in the database be N. 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 $\{a_1, \ldots, a_{n_A}\}$ be the discrete potential 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 defacto occurrence probability of value a_j in attribute A.

2.1 Notations and Primitives

Throughout this paper, we repeatedly use a set of notations and security primitives. In the following, we summarize some of the more frequent ones.

For any value (e.g., numeric) x, let $b(x)$ be the number of bits required for its accurate representation and $msb(x, b)$ be its most significant b bits. If $b(x) < b$, we left-pad x with $(b - b(x))$ zeroes to form a b-bit result. Similarly, $lsb(x, b)$ is used to denote the least significant b bits of x. If by wm we denote a watermark to be embedded of length $|wm|$, $wm[i]$ will then be the ith bit of wm. Let $set\_bit(d, a, b)$ be a function that returns value d with the bit position a set to the truth-value of b. In any following mathematical expression, let the symbol "&" signify a *bit-AND* operation.

A special defacto secure construct we are leveraging is the one-way cryptographic hash. If $crypto\_hash()$ is a cryptographic secure one-way hash, of interest are two of its properties: 1) it is computationally infeasible, for a given value V', to find a V such that $crypto\_hash(V) = V'$ (onewayness) and 2) changing even one bit of the hash input causes random changes to the output bits (i.e., roughly half of them change even if one bit of the input is flipped). Examples of potential candidates for $crypto\_hash()$ are the MD5 or the SHA class of hashes. For more details on cryptographic hashes, consult [22]. Let $H(V, k) = crypto\_hash(k; V; k)$ (where ";" denotes concatenation).

2.2 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. Moreover, 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 the following, we discuss challenges specifically associated with categorical data types.

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 result.

A3. Subset Alteration. Altering a subset of the items in the original data set such that there is still value associated with the result. 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. 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.

A5. Attribute Remapping If data semantics allow it, remapping 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 remapping. In Section 4.3, 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 defeatable 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.

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. Then, we propose new encoding methods able to leverage the newly discovered bandwidth.

3.1 Challenges

Given our research in the numerical domain, the first impulse was to build an extension of it for nonnumeric attributes. This would start by establishing a mapping between the nonnumeric domain and a numeric one, followed by a translation of the input data A to a set of numbers N (after all, any data can be represented as a string of bits). In the next step, the numeric watermarking method is deployed on the translated data N and a watermarked version of it (N') is obtained. If the mapping features certain properties (e.g., has an inverse), the algorithm can then translate this watermarked version (N') back into the original data domain (according to the inverse mapping) and produce A', a watermarked version of A. The assumption here is that there exists a mapping that is stable and is suitable. For example, in the case of A being an attribute containing multimedia JPEG images (possibly under the form of BLOB fields), this mapping might be exactly the DCT[1] (or a combination of the significant DCT coefficients). The detection algorithm will function similarly, by translating the suspected watermarked input data to the numeric domain (using the inverse transform) and deploying the numeric detection process on the translation.

Unfortunately, depending on the actual data domain, nonnumeric relational data will feature a different set of data value and quality metrics and associated uses. For many applications, this will make it difficult to directly apply the above idea. Let us consider, for example, the case of changing the value of a categorical DEPARTURE_CITY attribute from "Chicago" to "Bucharest." This is likely to affect the data quality of the result more than a simple change in a numeric domain. There are no "epsilon" changes in this domain. This discrete characteristic of the data requires discovery of fundamentally new bandwidth channels and associated encoding algorithms.

3.2 Bandwidth Channels

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. We propose to make use of the encoding bandwidth found in these *associations between categorical attributes* (including possibly the primary key). 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).

Because the discrete nature of the data domain makes it such that any watermark-related data alteration can be potentially significant, intuitively, one would desire to minimize the *number* of such alterations while maximizing the resilience of the encoding to potential attackers and transforms. Additionally, if there exists a certain distance metric in the attribute value domain, then another desiderata of interest would be to upper bind some function of the distances of all performed alterations (see Section 5.2).

3.3 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 3.4 for an extended discussion). In the extreme case, 1) just one attribute and the primary key are going to survive. A milder alternative, 2) assumes that several (e.g., two) categorical attributes and the primary key survive the partitioning process. Apparently, a watermarking method for 1) presents

[1] Discrete Cosine Transform. A frequency-domain transform used in the compression process of JPEG images, quantifying an image into a set of coefficients.

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

(a)

```
wm_embed_alt($K$,$A$,$wm$,$k_1$,$e$,ECC)
  $wm\_data \leftarrow ECC.encode(wm, wm.len)$
  $idx \leftarrow 0$
  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 $embedding\_map$
```

(b)

Fig. 3. (a) Embedding algorithm. (b) Alternative using embedding map (bit size adjustments omitted).

the disadvantage of a direct primary key-dependency. In Section 3.4, we further expand on this.

Let us propose an encoding method for 1), in which we encode a watermark in the bandwidth derived from the association between the primary key and a categorical attribute A (see Fig. 4). In Section 5, we analyze 2).

3.3.1 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 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 then considered for mark encoding.

Step One. 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 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 value satisfies a certain secret criteria (similar criteria are found in various frameworks, e.g., [15]).

Note: The fit tuples set 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 3.5 for a more detailed analysis.

Note on 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 ith bit of wm_data. Thus, before embedding, our algorithm starts by deploying the error correcting code first to compute the bits to be embedded $wm\_data = ECC.encode(wm, \frac{N}{e})$.

Step Two. For each "fit" tuple T_i, we encode one bit by altering $T_i(A)$ to become $T_i(A) = a_t$, where

$$t = set\_bit\Bigg(msb(H(T_i(K), k_1), b(n_A)), 0, wm\_data\Bigg[msb\Bigg(H(T_i(K), k_2), b\bigg(\frac{N}{e}\bigg)\Bigg)\Bigg]\Bigg)$$

and 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 forcing its least significant bit to a value according to a corresponding (random, depending on the primary key and k_2) position in wm_data.

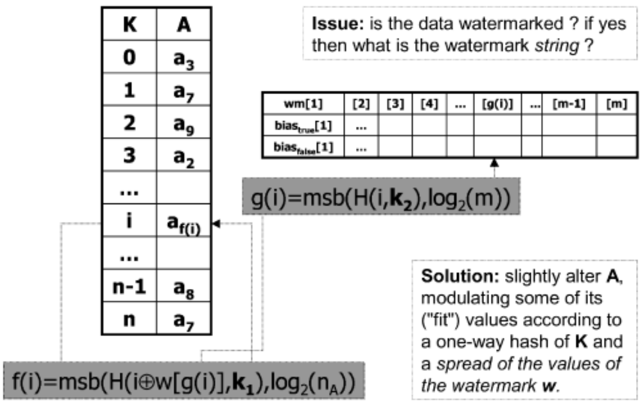
Note: The use of a second different key here ensures that there is no correlation between the selected tuples for embedding (selected also by k_1) and the corresponding bit value positions in wm_data (selected by k_2). Such a correlation would potentially cause certain bits to be never considered in 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 pseudorandom nature of $H(T_i(K), k_2)$ guarantees, on average, that a large majority of the bits in wm_data are going to be embedded at least once.[2]

Alternately, we could keep an on-the-fly hash-table/mapping (with $\left(\frac{N}{e}\right)$ entries, see Figs. 3b and 5b) 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

[2] The ulterior step of error correction can tolerate such small changes.



Issue: is the data watermarked ? if yes then what is the watermark *string* ?

Solution: slightly alter **A**, modulating some of its ("fit") values according to a one-way hash of **K** and a *spread of the values of the watermark* **w**.

Fig. 4. Overview of multibit watermark encoding.

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 6.

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 4.4.

3.3.2 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 $H(T_i(K), k_1) \bmod e = 0$.

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

$$wm\_data\left[msb\left(H(T_j(K), k_2), b\left(\frac{N}{e}\right)\right)\right] = t\&1.$$

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

3.4 Multiple Attribute Embeddings

The above encoding method makes use of the bandwidth present in the association between the primary key and the categorical type attribute A. It does *not* touch the primary key attribute but rather relies on modulating A through minor alterations (and data additions, see Section 4.4).

In the following, we extend this algorithm to provide more generality and resilience, in particular to attacks of the type A4 (vertical data partitions). In a possible attack scenario, Mallory partitions the data in such a way as to preserve only two attributes (multisets) and no primary key (see Fig. 6).

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 embeds a watermark separately in *each* of these associations. In other words, if the original watermarking method reads $mark(K, A)$ 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 (see Section 3.3), while maintaining the rest of the algorithm in place. This provides protection against A4 attacks and allows for more resilience in the rest of the scenarios (as there are more rights "witnesses" to testify). In addition, it 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 5), the two encodings seem to be independent. But, in the case of additionally watermarking the pair (A, B), modifying B suddenly interferes with the alterations occurring in the (K, B) case.

Although the level of interference is likely to be very low, especially in large data sets,[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 unmodified 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. The construction of such an "interference graph" is shown in Fig. 7.

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. See Section 4.1 for a related analysis and extension.

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

[4] As the probability of the same tuple to be considered again in the second encoding is low, see also Section 3.5.

```
wm_dec($K$,$A$,$k_1$,$k_2$,$e$,ECC)
  for ($j \leftarrow 1$; $j < N$; $j \leftarrow j+1$)
    if ($(H(T_j(K), k_1) \bmod e = 0)$ 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 $wm$
```

(a)

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

(b)

Fig. 5. (a) Decoding Algorithm (b) Alternative using embedding map.

3.5 Analysis: False Positives and Vulnerability to Attacks

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

It can be easily proved 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 $N = 6{,}000$ 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 discuss 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, $P(r, a)$?* This metric illustrates the relationship between attack vulnerability and embedding bandwidth. It can be shown that

$$P(r,a) = \sum_{i=r}^{a} \binom{a}{i} \times p^a \times (1-p)^{a-i}.$$

Remember that only every eth 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 the corrected version

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

Consider $r = 15$, $p = 70$ percent (it is quite likely that when Mallory alters a watermarked tuple, it will destroy the embedded bit), $a = 1{,}200$ (20 percent of the tuples are altered by the attacker, $|wm| = 10$, and $e = 60$ ($|wm\_data| = 100$). 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 as it makes the case even stronger.

Because we have an effectively binomial distribution experiment with $X_i = 1$, with probability p and $X_i = 0$, with probability $1 - p$. $E[X_i] = p$, $var(X_i) = E[X_i^2] - (E[X_i])^2 = \ldots = p \times (1-p)$, by using the central limit theorem [20], we can derive that $f(\sum X_i)$, where

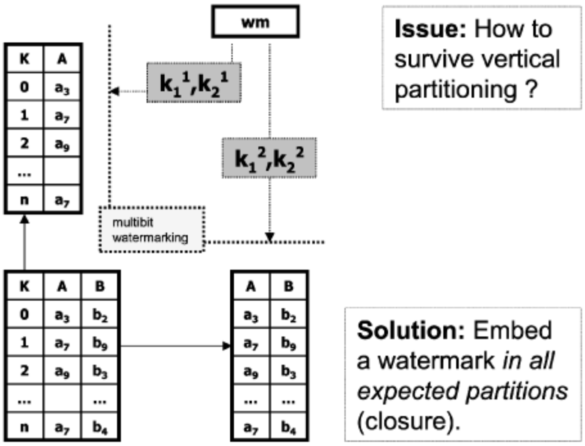
$$f\left(\sum X_i\right) = \frac{\sum X_i - \frac{a}{e} \times p}{\sqrt{\frac{a}{e} \times p \times (1-p)}} \tag{2}$$

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 $P(15, 1{,}200) \approx 31.6$ percent.

Let us assume that the error correcting code tolerates an average of $t_{ecc} = 5$ percent alterations to the underlying data and that the alteration propagation is uniform and stable. Intuitively, what we mean is that if one bit in wm_data is altered above the t_{ecc} bound, then a stable average of $\frac{|wm|}{|wm\_data|}$ bits are altered in the resulting error corrected watermark $wm = ECC.decode(wm\_data, |wm|)$. The final watermark is then incurring only an average fraction of

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

alteration. In our case, this is only 1.0 percent, 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 percent of the data and even then has only a success rate of 31.6 percent!* This analysis was done in a highly attack-


wm
$k_1{}^1, k_2{}^1$
$k_1{}^2, k_2{}^2$
multibit watermarking
Issue: How to survive vertical partitioning ?
Solution: Embed a watermark *in all expected partitions* (closure).

K	A
0	a_3
1	a_7
2	a_9
...	
n	a_7

K	A	B
0	a_3	b_2
1	a_7	b_9
2	a_9	b_3
...		...
n	a_7	b_4

A	B
a_3	b_2
a_7	b_9
a_9	b_3
...	...
a_7	b_4

Fig. 6. Defeating vertical partitioning.

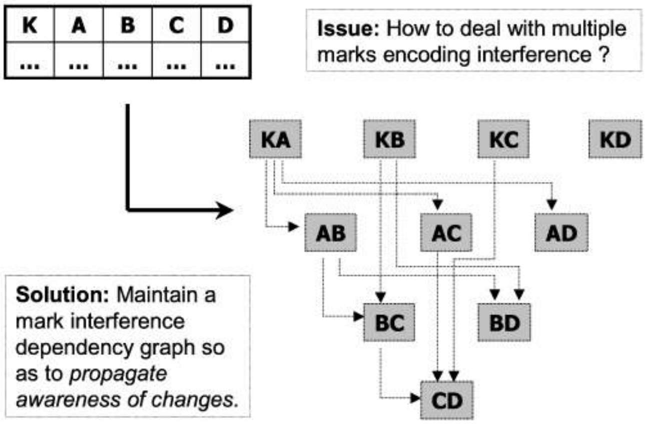

K
A
B
C
D
...
Issue: How to deal with multiple marks encoding interference ?
KA
KB
KC
KD
AB
AC
AD
BC
BD
CD
Solution: Maintain a mark interference dependency graph so as to *propagate awareness of changes.*

Fig. 7. Handling multiple marks interference.

favorable scenario in which error correction can only handle 5 percent 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?

If we assume that Mallory cannot afford to modify more than 10 percent of the data items ($a = 600$) and we set a maximum tolerable threshold $\tau = 10$ percent 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 (2) and doing a normal distribution table lookup, we derive that (for $\tau = 10$ percent) 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 ≈ 4.3 percent of the data to guarantee these bounds!

4 Improvements

4.1 Correlation Attacks

The solution above features a particular issue of concern in certain cases of multiattribute embeddings where two nonkey attributes are used in the encoding, i.e., mark(A,B). Because of the correlation between the watermarking alteration (the newly selected value $T_i(B) = b_t$) and its actual location (determined by the fitness selection, $H(T_i(A), k_1)$, and e), sometimes Mallory can mount a special attack with the undesirable result of revealing some of the mark bit embedding locations. This occurs if the fitness criteria decides that a particular value of A yields a tuple fit and that value of A appears then in multiple (statistically significant number of) different tuples. This is possible only if A is not a primary key but rather another categorical attribute (with repeating duplicate values).

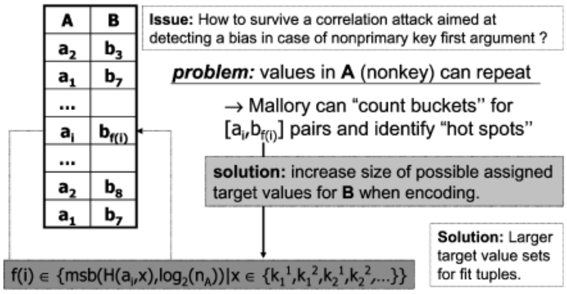

A
B
a_2 b_3
a_1 b_7
...
a_i $b_{f(i)}$
...
a_2 b_8
a_1 b_7
Issue: How to survive a correlation attack aimed at detecting a bias in case of nonprimary key first argument ?
problem: values in **A** (nonkey) can repeat
→ Mallory can "count buckets" for $[a_i, b_{f(i)}]$ pairs and identify "hot spots"
solution: increase size of possible assigned target values for **B** when encoding.
$f(i) \in \{msb(H(a_i,x), \log_2(n_A)) | x \in \{k_1^1, k_1^2, k_2^1, k_2^2, \ldots\}\}$
Solution: Larger target value sets for fit tuples.

Fig. 8. Defeating correlation attacks.

The attack then proceeds by first realizing that, despite the one-wayness of the deployed hash function $H()$, in fact, A is the only variable that determines *both* the bit embedding location (tuple "fitness") *and* its value. If Mallory is able to detect this correlation for potential candidates (tuples), it would quickly lead to exposing some of the ones carrying a watermark bit. But how does he check for the correlation? Mallory can simply build a set of "hash buckets" (see Fig. 8) for each separate value of A (yielding the same value of $H(T_i(A), k_1)$) and count (for all matching tuples) if there is a statistical bias for a certain value (e.g., "true") of the least significant bit of t (see Section 3.3.1). If such a bias is discovered, e.g., if a majority of LSB values are "true," then Mallory suspects (rightfully so) that the respective tuples are "fit" and a watermark bit of "true" (for example) is embedded in those locations. Mallory can now obliterate the embedding in these tuples by randomization, leading to a loss of the corresponding watermark bit.

In other words, if, as a result of the extension proposed in Section 3.4, two attributes A and B are used in a watermarking process, $mark(A, B)$ (and the data set contains many "fit" tuples with repeated values for attribute A), Mallory can discover the association between the individual unique values of A and the bit-embeddings in B. He can then use this discovered association to randomize the embeddings and effectively remove the corresponding watermark bits. Thus, the problem lies here in the correlation between the actual bit location and the bit value, correlation induced by the fact that a single variable (A) determines both of these and this variable can have repeated values for different tuples, allowing for a "bucket counting" attack as described above.

One solution to this issue would be to simply restrict the fitness selection criteria for tuples so as to only include the ones with attribute $T(A)$ values that do not have a significant number of repeats throughout the data. A more radical idea would be to simply search the space of potential k_1 values until the fitness criteria results in selecting tuples with different values for A.

Note: These solutions work only if A indeed does contain a significant number of nonrepeating values in the data. In any case, this problem has the potential to introduce a significant reduction in available encoding bandwidth. An interesting extreme scenario occurs in the case of binary attributes, i.e., attributes with only two possible values. If, for example, A can only take values in the $\{0, 1\}$ set (e.g., quite likely in many data mining sets), intuitively, it cannot be used as a first argument in an $mark(X, Y)$ operation (that is, it cannot play the pseudo primary key role). This is so

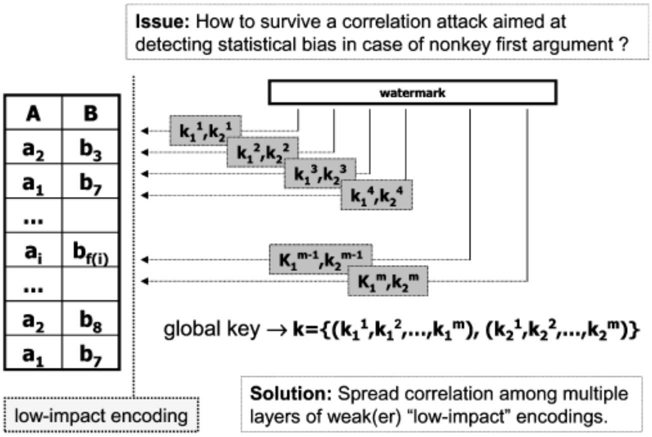

Issue: How to survive a correlation attack aimed at detecting statistical bias in case of nonkey first argument ?
watermark
A | B
a_2 | b_3
a_1 | b_7
...
a_i | $b_{f(i)}$
...
a_2 | b_8
a_1 | b_7
k_1^1,k_2^1
k_1^2,k_2^2
k_1^3,k_2^3
k_1^4,k_2^4
K_1^{m-1},k_2^{m-1}
K_1^m,k_2^m
global key → $\mathbf{k=\{(k_1^1,k_1^2,\ldots,k_1^m),(k_2^1,k_2^2,\ldots,k_2^m)\}}$
low-impact encoding
Solution: Spread correlation among multiple layers of weak(er) "low-impact" encodings.

Fig. 9. Defeating correlation attacks revisited (multiple embeddings).

due to the fact that the fitness selection criteria (controlled by X) is going to either 1) fail to discover any fit tuples (i.e., resulting in no available encoding bandwidth), 2) possibly partition the entire data set into two, namely, a subset of "fit" tuples (e.g., corresponding to the $T_i(A) = 1$ values) and the rest or 3) deem all tuples fit in which case the entire data set is going to be altered in a potential watermarking operation. This case is interesting because it illustrates the impact of n_X (the cardinality of the first attribute's possible values set) on the $mark(X, Y)$ operation.

4.1.1 *Multiple Embeddings*

There exists a refinement that would overcome many of the above. What if the actual watermark were to consist of a combination of several different embeddings, each, in turn, being an encoding using a different k_1 value (see Fig. 9). While each of these "low impact" encodings would be weaker than the original solution, their combined "sum" can be made (arguably) arbitrarily strong(er) by increasing their number. At the same time, correlation attacks would be defeated.

If, for example, we embed two watermarks with different keys (k_1^1, k_2^1) and (k_1^2, k_2^2), the correlation attack cannot be performed "across" the encodings, as the $H(T_i(A), k_1^1)$ and $H(T_i(A), k_1^2)$ values are not going to be consistent with each other, making "bucket counting" impossible.

One price to pay is the amount of computation required at each step to decode all of the potential watermarks. Another issue of (arguably) minor concern could be the fact that the same tuple might be considered in multiple encodings, in which case, some of these will suffer a mark loss. A mark interference graph (see Fig. 7) can be deployed to avoid such collisions.

The new algorithm defeating correlation detection through multiple embeddings is illustrated in Fig. 10. In Fig. 10a, the val_seen_cnt (together with $local\_seen\_cnt[]$) hash-tables are used to avoid multiple uses of the same values of K (within the same embedding round), thus simply inhibiting the appearance of biases that can be detected by Mallory. This mechanism also avoids the interference between different (successive) watermark embeddings as it "skips" all instances which were already "touched" by previous encodings and only considers "fresh" tuples. The extended algorithm in Fig. 10b then applies this "low-impact encoding" in Fig. 10a an arbitrary number of times (determined by its parameter M) with separate keys resulting in an output without any detectable correlations.

The separate keys $\{k_1^1, k_1^2, k_1^3, \ldots, k_1^i\}$ and $\{k_2^1, k_2^2, k_2^3, \ldots, k_2^i\}$ are generated from the initial k_1, k_2 pair by a classic cryptographic key construction used for encryption in block ciphers, namely, the counter mode (CTR). In this description, E is any arbitrary keyed encryption method (e.g., AES) and $nonce$ is a random one-time use secret (to be considered part of the algorithm input secrets, including the k_1, k_2 pair). For more details, please refer to [6].

Note: We included this key generation mechanism here as an implementation suggestion. From a rights assessment point of view, how these keys are generated is out of the current scope. In the simplest case, they could as well be directly provided as input to the algorithm instead.

4.2 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 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 methods are required. Thus, we propose to extend the marking algorithm with semantic data constraints awareness.

We introduced and successfully analyzed this idea (an instance of *consumer driven* encoding) in [24]. Each property of the database that needs to be preserved is written as a constraint on the allowable change to the data set. The watermarking algorithm is then applied with these constraints as input and reevaluates them continuously for each alteration. A backtrack log (see Fig. 11) is kept to allow undo operations in case certain constraints are violated by the current watermarking step.

4.3 Bijective Attribute Remapping

Consider the scenario of an attack in which the categorical attribute A is remapped through a bijective function to a new data domain. In other words, the $\{a_1, \ldots, a_{n_A}\}$ values are going to be mapped into a different set $\{a_1', \ldots, a_{n_A}'\}$. The assumption here is that, from Mallory's perspective, the remapped data still features enough value that can be banked upon.[6]

The problem of remapping becomes clear in the mark detection phase when, after tuple fitness selection, the bit decoding mechanism will fail, being unable to determine t

[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.

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

```
wm_embed_X2($K$,$A$,$wm$,$k_1$,$k_2$,$e$,ECC,$val\_seen\_cnt[]$)
  $wm\_data \leftarrow ECC.encode(wm, wm.len)$
  initialize $local\_seen\_cnt[]$
  for $(j \leftarrow 1; j < N; j \leftarrow j+1)$
    if $(H(T_j(K), k_1))$ mod $e = 0)$ then
      $local\_seen_cnt[T_j(K)] \leftarrow local\_seen\_cnt[T_j(K)] + 1$
      if $(local\_seen\_cnt[T_j(K)] > val\_seen\_cnt[T_j(K)])$ then
        $t \leftarrow set\_bit(H(T_j(K), k_1), 0, wm\_data[H(T_j(K), k_2)])$
        $T_j(A) \leftarrow a_t$
        $val\_seen\_cnt[T_j(K)] \leftarrow val\_seen\_cnt[T_j(K)] + 1$
```

(a)

```
wm_embed_X($K$,$A$,$wm$,$k_1$,$k_2$,$e$,ECC,$M$,$nonce$)
  $k_1^0 \leftarrow k_1$
  $k_2^0 \leftarrow k_2$
  initialize $val\_seen\_cnt[]$
  for $(j \leftarrow 1; j < M; j \leftarrow j+1)$
    $k_1^j \leftarrow E(k_1^{j-1}, (nonce; j))$
    $k_2^j \leftarrow E(k_2^{j-1}, (nonce; j))$
    wm_embed_X2($K$,$A$,$wm$,$k_1^j$,$k_2^j$,$e$,ECC,$val\_seen\_cnt$)
```

(b)

Fig. 10. Extended Algorithm: (a) Awareness of previous values is included in the low-impact encoding (the val_seen_cnt hash-table). (b) Handling correlation detection through multiple layers of embeddings (embedding algorithm shown, base decoding is similar to Fig. 5a).

such that $T_j(A) = a_t$. It will instead determine a t value that maps to the $\{a_1', \ldots, a_{n_A}'\}$ 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 $\{a_1, \ldots, a_{n_A}\}$ 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 $\{a_1, \ldots, a_{n_A}\}$. We propose to sample this frequency in the suspected (remapped) data set 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 (see Fig. 12).

4.4 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 e effectively "reduces" the fitness criteria testing space to a cardinality of e, 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 e tuples should conform (as the values are evaluated modulo e).

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 3.5 for an analysis on the impact of watermark bits on the encoding resilience.

5 Discussion

5.1 Vertical Partitioning Revisited

While most vertical partitioning attacks can be handled by a multiple attribute embedding solution as described in Section 3.4, consider an extreme vertical partitioning attack scenario in which Mallory only preserves a single (categorical) attribute A (a multiset).

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 $\{a_1, \ldots, a_{n_A}\}$, the main market-able 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.

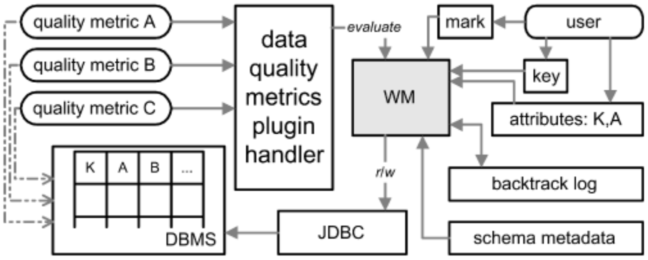


Fig. 11. Data quality is continuously evaluated. A backtrack log aids undo operations in cases where the watermark embedding would violate quality constraints (see also [24]).

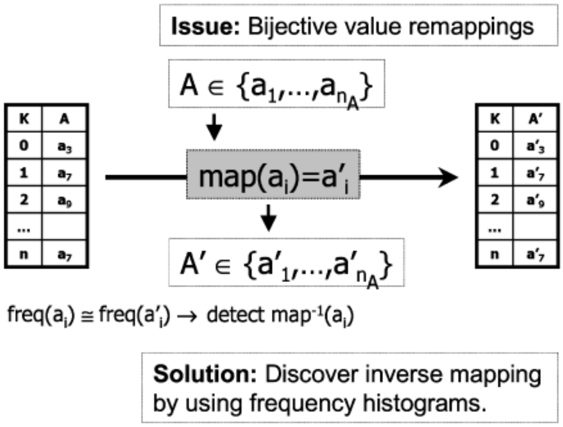

Issue: Bijective value remappings
A ∈ {a1,...,anA}
K A
0 a3
1 a7
2 a9
...
n a7
map(ai)=a'i
A' ∈ {a'1,...,a'nA}
K A'
0 a'3
1 a'7
2 a'9
...
n a'7
freq(ai) ≅ freq(a'i) → detect map-1(ai)
Solution: Discover inverse mapping by using frequency histograms.

Fig. 12. Handling attribute remapping.

In [24], 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 (see Fig. 13). 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 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 multimark interference (with the other encodings), which can be solved by an approach similar to the one in Section 3.4 using embedding markers and/or deploying a mark interference graph/tree (see Fig. 7).

5.2 Minimizing Alteration Distance

An interesting problem to consider is the case when, for a given "fit" tuple, certain alterations would be preferred to others. For example, if the given attribute represents airport names, intuitively, it is likely that an alteration changing "Chicago, O'Hare" into "Las Vegas" produces more damage overall than one that would result in "Chicago, Metro." In other words, what if there exists a certain distance metric model for the values within a categorical attribute and the encoding is to minimize a (e.g.,) sum (for each change in the data) of these associated alteration distances.

This scenario can be easily dealt with through the design of a data quality plugin that continuously evaluates the amount of damage already performed and allows only the alterations that conform. However, such a solution suffers from several issues. For one, upon encountering a "fit" tuple that doesn't conform to this quality metric, the only alternative available to the data quality plugin is to simply veto the proposed modification. Depending on the restrictiveness of the desired alteration distance upper bound, this will yield fewer tuples that can be used in the marking process, thus resulting in a reduced bandwidth. Additionally, a condition like the one above (sum of all alteration distances to not exceed maximum) is not easy to implement so as to result in a resilient embedding. A trivial imple-mentation would simply allow all alterations until the sum exceeds a certain upper bound. But, in this case, if the data is read sequentially, this will result in a data set watermarked only in a portion at its beginning, after which most of the alterations will be denied. A smarter version would be to "spread" the allowed modifications throughout the data, which, in turn, would require the data quality plugin to allow encoding only to some of the good "fit" tuples, to "save" some of the allowed alteration distance for future ones. This would hopefully result in the coverage of the entire data.

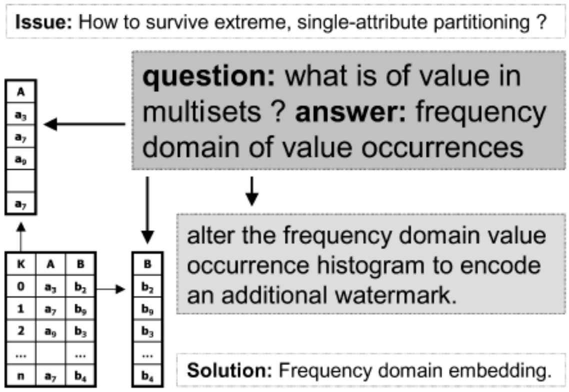

Issue: How to survive extreme, single-attribute partitioning ?
question: what is of value in multisets ? answer: frequency domain of value occurrences
alter the frequency domain value occurrence histogram to encode an additional watermark.
Solution: Frequency domain embedding.
A
a3
a7
a9
a7
K A B
0 a3 b2
1 a7 b9
2 a9 b3
... ...
n a7 b4
B
b2
b9
b3
...
b4

Fig. 13. Handling extreme multiset partitioning.

But what if we could modify the encoding method so as to naturally accommodate such a case? We propose to modify the tuple alteration criteria to result not in one value for t in the selection of $T_i(A) = a_t$ (see Section 3.3.1), but rather in an entire set of $\gamma < n_A$ potential candidate values. Let us define

$$t' = set\_bit\Bigg(msb(H(T_i(K), k_1), b(n_A - \gamma)), 0,$$
$$wm\_data\Bigg[msb\Bigg(H(T_i(K), k_2), b\Bigg(\frac{N}{e}\Bigg)\Bigg)\Bigg]\Bigg)$$

and

$$S_{t'} = \{a_t | msb(t, n_A - \gamma) = msb(t', n_A - \gamma)\}.$$

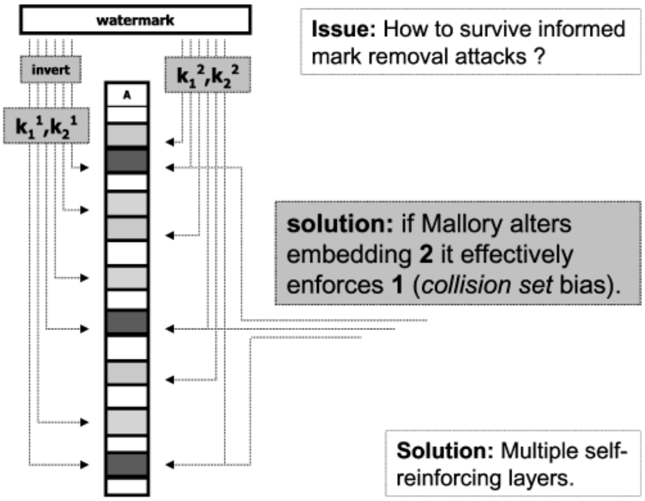
Then, at each encoding step for a fit tuple i, its new value $T_i(A)$ is selected from $S_{t'}$ so as to minimize the alteration distance.

In other words, we "divide" the set of potential discrete values for A into $\frac{n_A}{\gamma}$ subsets of γ elements (sharing the first $b(n_A - \gamma)$ most significant bits). Each t' value then selects a certain subset $S_{t'}$, and the final corresponding $T_i(A)$ is constructed by selecting the "closest" (in terms of the above discussed alteration distance metric) data value in $S_{t'}$.

The number and size of the subsets are controlled through a choice of an appropriate γ. To the extreme, if $\gamma = n_A - 1$, only two such subsets exist (i.e., one subset with values having the most significant bit of their index "true" and the other subset with the rest).

5.3 Blindness, Incremental Updates, and Streams

Our watermarking method is blind in that it doesn't require the original data in the detection process. This is important, because it is unrealistic to assume the original data available



Issue: How to survive informed mark removal attacks ?

solution: if Mallory alters embedding **2** it effectively enforces **1** (*collision set* bias).

Solution: Multiple self-reinforcing layers.

Fig. 14. Handling an informed Mallory.

after a longer time elapses, especially in the case of massive data sets.

The 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. Therefore, the encoding also (gracefully) handles data streaming in which tuples are available only in a one-pass streaming model with limited storage space and processing power. This makes it ideal for a transparent deployment scenario in which a black box "sits" at the data "exit" point and continuously watermarks outgoing information.

5.4 MultiLayer Self-Reenforcing Marks

The above solution (and any symmetric/single-key watermarking method) is vulnerable to a scenario in which Mallory (who might have participated in a public court hearing) finds out the key that was used to watermark a given Work. He can then use the key to remove the watermark and have illicit access to the original version.

Work in asymmetric multimedia watermarking [4], [5], [7], [8], [9], [11], [26] deploys different keys for detection (public) and embedding (secret). The design of an asymmetric version of our solution is to be subject to future research. Now, however, we propose a draft idea that seems to handle this scenario reasonably well.

The idea is to simply embed multiple weak watermarks with different secret keys and reveal in court only a certain subset of these, enough to satisfy the convince-ability requirements. Having these keys would only enable the removal of the corresponding watermarks and nothing more. The data will still feature the remaining ones, which will hopefully be enough for the next court hearing instance.

Yet another idea would be to embed multiple self-reenforcing pairs of watermarks $(w_1, w_2)_i$ with different keys $(k_1^1, k_2^1, k_1^2, k_2^2)_i$, such that altering w_2 will result in enforcing w_1 (see Fig. 14). The feasibility, details, and benefits of such a method are to be subject to future investigation. Space constraints prevent further elaboration here.

5.5 Categorical and Numerical Data Types

In many applications, a combination of categorical and numerical relational data types are manipulated. Can we design a method for rights assessment that considers this potential type-mix directly? While the trivial approach would be to treat these data types separately (e.g., deploy numeric methods [15], [25] for numeric types, etc.), it is (arguably) of interest to explore how/if these methods can be combined, if a new method is to be designed, and if it is of benefit to do so.

Why would one provide a rights protection method that works on a data set composed of two different types of data? Let us assume, for example, a schema (A, B) which contains an attribute A of a numeric continuous type and an attribute B of a categorical type. What could be achieved by such a (combined) proof mechanism that could not be achieved by the individual application of the separate methods for each type (e.g., numeric for attribute A and categorical for B)? One answer is that a rights assessment mechanism for combined data types would possibly also prove that the associated data sets were actually produced "together." Achieving such a proof could be of significant interest, for example, if the intrinsic value of the data lies in the actual *combination* of the two data types. Another answer would have to do with the existence of a primary key, in the case when the considered data appears under the form of associated multisets and does not contain a primary key with unique values.

How could one go about designing such a solution? While extensive details are out of scope here, let us briefly indulge in this exploration. One idea would be to "associate" the two attributes by encoding a watermark in B using A as a first argument in a normal encoding procedure for categorical types, $mark(A, B)$. In fact, it seems there are no modifications necessary to enable this. Thus, a solution is provided by the current work, in effect, watermarking the (A, B) pair by using B as a watermark carrier and A as associative reference.

However, there might be scenarios where A cannot be used as a first argument (e.g., in multiple layers of numeric watermarks it would be changed by another encoding). It would also be (arguably) of interest to modify A instead of B in the watermarking process because of the potential of less significant damage in the case of minor numeric alterations to A (as opposed to discrete changes in the categorical domain). Because the existing numeric domain methods [15], [25] require a resilient, unchanging primary key, this becomes especially challenging. Could somehow the values in B (maybe combined with information from the associated A values) offer enough entropy for the construction of a primary key? In other words, can we use a combination of $B + msb(A)$ to construct a primary key to deploy numeric watermarking for A? Or does this require more complex constructs? This is a subject for future research.

5.6 Multiple Data Sources

Let us now discuss how our solution handles the case of data sets derived from multiple data sources. This scenario could be of significant interest for example in the case of an EquiJOIN performed between two data sets, (A, B), unwatermarked and (B, C), watermarked (plain new data addition was already discussed previously). Does the watermark survive? The short answer is *likely yes*. Because the watermark encoding relies on a bias in the association

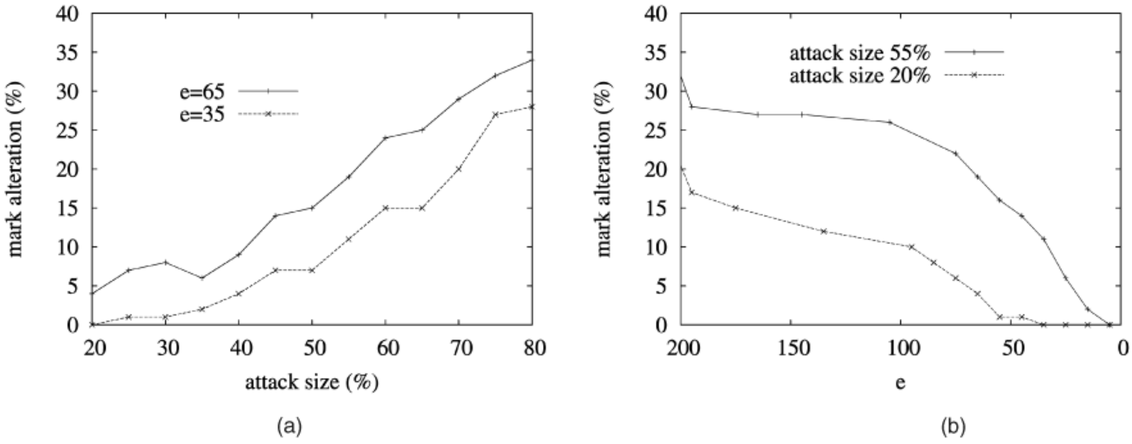


Fig. 15. (a) The watermark degrades gracefully with increasing attack size ($e = 65$). (b) More available bandwidth (decreasing e) results in a higher attack resilience.

between B and C, it can be naturally retrieved from the JOIN result under certain reasonable assumptions.

The first assumption is that B is preserved in the JOIN result. Otherwise, the watermark cannot be retrieved. The second assumption has to do with the fact that, due to the effect of the JOIN, the detected encoding will be likely weaker than the original as some values of B in the (B, C) data set would not survive in the JOIN result (if these values don't appear in (A, B)). However, in a reasonable scenario, significantly much of (B, C) appears in the result, thus allowing for detection of the original watermark.[7] However, if too few values from the (B, C) data set are not finding their way into the result, the watermark will not survive either.

Now, if (A, B) was also watermarked, the reasoning is similar. In a favorable scenario, both of the original marks can be recovered from the join result. On the other hand if the join result size is (statistically speaking) much smaller than the original, no significant watermark bias can be detected.

6 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 of 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 141,000 tuples for relation *{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 chose Item_Nbr, a categorical attribute, uniquely identifying a finite set of products. The watermark considered was 10 bits long and 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.

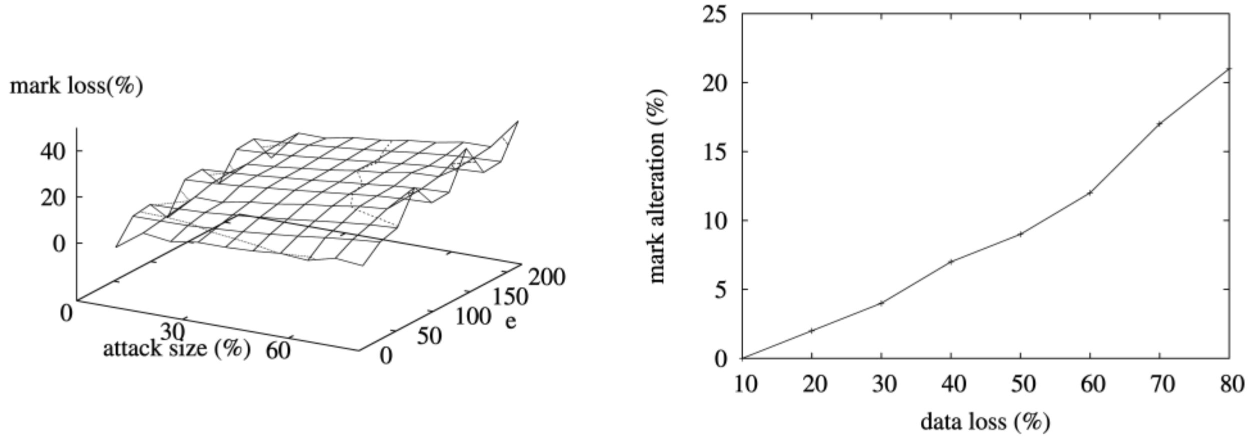
In the first experiment, we analyzed the behavior of the embedded watermark in the presence of massive data alterations. As the attack size grows (random alterations to the data), the watermark distortion increases. The error correction mechanism (majority voting in this case) does a good job in error recovery. This is particularly so in the case of random alterations to the underlying data, with the only available data altering attack option as discussed in Section 3.5. Fig. 15a depicts this phenomena for two values of e.

In the next experiment, we explored the relationship between the amount of data modifications required in the watermarking phase and a minimum guaranteed watermark resilience. It can be seen in Fig. 15b that as e increases (decreasing number of encoding modifications), 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). Fig. 16a represents the composite surface for both experiments.

An experiment analyzing resilience to data loss is depicted in Fig. 16b. 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).

In Fig. 17, various aspects of the implementation execution times are illustrated. In Fig. 17a, it becomes clear that there seems to be a minimal dependency of e of the embedding (detection graph is virtually identical, thus

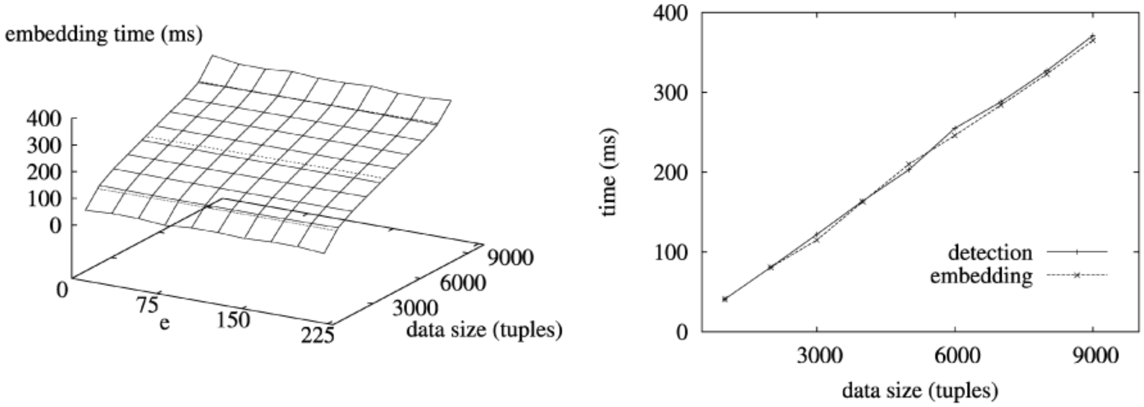
[7] This is possible due to the detection process being independent of actual data location (as long as some of the "fit" values of B are preserved in the join result).



(a)

(b)

Fig. 16. (a) The watermark alteration surface with varying c (watermark modifications) and attack size. Note the lower-left to upper-right tilt. (b) The watermark degrades almost linearly with increasing data loss.



(a)

(b)

Fig. 17. (a) Embedding time dependency as a function of e and N. (b) Detection time requirements are similar to embedding and linear in the size of the data.

omitted) times. Execution time seems to be mainly linear in the data size, as also expected. In Fig. 17b, it can be seen that detection yields almost identical times as the embedding process. The linear dependency is clear. An average of 25K tuples can be processed per second by our proof of concept implementation. We expect a speed-up of orders of magnitude in an optimized industry-level version.

7 Conclusions

In this work, we defined the problem of rights protection for categorical data through watermarking. We proposed a solution and analyzed it both in theory and in practice. We outlined improvements and 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 percent data loss with a watermark alteration of only 25 percent.

Various issues remain to be explored. Additive watermark attacks need to be analyzed in more depth. Self-reenforcing multilayered watermarks seem to be of important potential for handling informed attackers and should be explored. Also, while the concept of on-the-fly quality assessment (see Section 4.2) has a good potential to function well, as confirmed also by experiments in [25], 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 provide a way to express such constraints and their propagation at embedding time (e.g., ”IF A.product ==’APPLES’ THEN ASSERT(A.price=’2.99’)”).

Acknowledgments

The authors would like to thank the anonymous reviewers for helpful comments. 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 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. An abbreviated version of these results appeared in [23].

REFERENCES

[1] E. Bertino, M. Braun, S. Castano, E. Ferrari, and M. Mesiti, "Author-X: A Java-Based System for XML Data Protection," *Proc. 14th Ann. Int'l Federation Information Processing Workshop Database Security*, pp. 15-26, 2000.

[2] E. Bertino, S. Jajodia, and P. Samarati, "A Flexible Authorization Mechanism for Relational Data Management Systems," *ACM Trans. Information Systems*, vol. 17, no. 2, 1999.

[3] C. Clifton and D. Marks, "Security and Privacy Implications of Data Mining," *Proc. Workshop Data Mining and Knowledge Discovery*, pp. 15-19, 1996.

[4] L.d.C.T. Gomes, M. Mboup, M. Bonnet, and N. Moreau, "Cyclostationarity-Based Audio Watermarking with Private and Public Hidden Data," *Proc. Convention Audio Eng. Soc.*, Sept. 2000.

[5] J. Eggers, J. Su, and B. Girod, "Public Key Watermarking by Eigenvectors of Linear Transforms," *Proc. European Signal Processing Conf.*, Sept. 2000.

[6] N. Ferguson and B. Schneier, *Practical Cryptography.* Wiley & Sons, 2003.

[7] T. Furon and P. Duhamel, "Audio Asymmetric Watermarking Technique," *Proc. Int'l Conf. Audio, Speech and Signal Processing*, June 2000.

[8] T. Furon and P. Duhamel, "Robustness of an Asymmetric Technique," *Proc. Int'l Conf. Image Processing*, Sept. 2000.

[9] T. Furon, I. Venturini, and P. Duhamel, "Unified Approach of Asymmetric Watermarking Schemes," *Security and Watermarking of Multimedia Contents III*, P.W. Wong and E. Delp, eds. 2001.

[10] J. Hale, J. Threet, and S. Shenoi, "A Framework for High Assurance Security of Distributed Objects," *Proc. 10th Ann. Int'l Federation for Information Processing WG11.3 Working Conf. Database Security*, 1997.

[11] F. Hartung and B. Girod, "Fast Public-Key Watermarking of Compressed Video," *Proc. IEEE Int'l Conf. Image Processing*, Oct. 1997.

[12] E. Hildebrandt and G. Saake, "User Authentication in Multidatabase Systems," *Proc. Ninth Int'l Workshop Database and Expert Systems Applications*, pp. 281-286, 1998.

[13] S. Jajodia, P. Samarati, and V.S. Subrahmanian, "A Logical Language for Expressing Authorizations," *Proc. IEEE Symp. Security and Privacy*, pp. 31-42, 1997.

[14] S. Jajodia, P. Samarati, V.S. Subrahmanian, and E. Bertino, "A Unified Framework for Enforcing Multiple Access Control Policies," *Proc. SIGMOD*, 1997.

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

[16] N. Li, J. Feigenbaum, and B.N. Grosof, "A Logic-Based Knowledge Representation for Authorization with Delegation," *PCSFW: Proc. 12th Computer Security Foundations Workshop*, 1999.

[17] M. Nyanchama and S.L. Osborn, "Access Rights Administration in Role-Based Security Systems," *Proc. Int'l Federation Information Processing Workshop Database Security*, pp. 37-56, 1994.

[18] S.L. Osborn, "Database Security Integration Using Role-Based Access Control," *Proc. Int'l Federation Information Processing Workshop Database Security*, pp. 245-258 2000.

[19] D. Rasikan, S.H. Son, and R. Mukkamala, "Supporting Security Requirements in Multilevel Real-Time Databases," citeseer.nj.nec.com/david95supporting.html, 1995.

[20] S. Ross, *A First Course in Probability.* Prentice Hall, 2001.

[21] R.S. Sandhu, "On Five Definitions of Data Integrity," *Proc. Int'l Federation Information Processing Workshop Database Security*, pp. 257-267, 1993.

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

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

[24] R. Sion, M. Atallah, and S. Prabhakar, "Rights Protection for Relational Data," *Proc. ACM SIGMOD*, 2003.

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

[26] J. Stern and J.-P. Tillich, "Automatic Detection of a Watermarked Document Using a Private Key," *Proc. 4th Int'l Workshop Information Hiding*, vol. 2137, Apr. 2001.

Radu Sion received the PhD degree in computer sciences from Purdue University in 2004 and the BSc and MSc degrees also in computer sciences from the "Politehnica" University of Bucharest in 1998 and 1999, respectively. In 2004, he held a visiting researcher position with the IBM Almaden Research Center. In August of 2004, he joined the Stony Brook University Computer Science Department. His current research interests are centered around interconnected entities that access data and need to do so with assurances of security, privacy, and functionality, preferably fast. His research lies at the intersection of security, databases, and distributed systems. Applications include: authentication, rights protection and integrity proofs, trusted reputation and secure storage in peer to peer and ad hoc environments, data privacy and bounds on illicit inference over multiple data sources, security in computation/data grids, and detection of intrusions by access profiling for online web portals.

Mikhail ("Mike") Atallah did his graduate studies at Johns Hopkins University from 1979 to 1982, obtaining the MS degree after two semesters (in 1980), and the PhD degree two years after that (in 1982). He joined the Computer Sciences Department at Purdue University immediately upon obtaining his PhD, and has been with Purdue since then. He was promoted to associate professor in 1986, to full professor in 1989, and to distinguished professor in 2004. His current research interests are in information security (in particular, software security, secure protocols, and watermarking).

Sunil Prabhakar's research focuses on performance and security issues in large-scale, modern database applications such as multimedia, moving-object, and sensor databases. The efficient execution of I/O is a critical problem for these applications. The scope of this research ranges from main memory to disks and tertiary storage devices. Sensor and moving object applications are also faced with the need to process large volumes of data in an online manner. The current research effort addresses efficient continuous query evaluation and novel techniques for managing the inherent lack of accuracy for these applications. His interests also lie in the design and development of digital watermarking techniques for structured (e.g., relational databases) and semistructured (e.g., XML) data. Prior to joining Purdue, Dr. Prabhakar held a position with Tata Unisys Ltd. from 1990 to 1994.