6th International Conference on Internet Technology and Secured Transactions, 11-14 December 2011, Abu Dhabi, United Arab Emirates

A Robust Watermarking Scheme for Relational Databases

Theodoros Tzouramanis
Department of Information & Communication Systems Engineering,
University of the Aegean,
Karlovassi, Samos, 83200, Greece
ttzouram@aegean.gr

Abstract—**Research into the issue of the rights protection of digital data is of critical importance since legal measures have proved ineffective against digital piracy. Digital watermarking tops the list of technical countermeasures. Its process involves the incorporation of a set of some data in the item to be protected; this set of data is called watermark. The accuracy of the item is slightly degraded but the watermark acts as a seal that henceforth identifies the intellectual owner. This paper proposes a novel watermarking scheme for relational data which is efficient against a range of attacks that may be issued to remove or destroy the watermark. The paper provides experimental results for a variety of parameter settings, revealing the robustness of the proposed scheme in numerous possible attacks.**

Keywords-digital rights protection; digital crime and forensics; relational databases; robust digital watermarking method.

I. Introduction

The advance of digital communications and the internet has opened new horizons in the social and business domain and has re-defined traditional perceptions of fields such as trade, banking and social welfare. On the other hand, the exponential increase of internet users, together with the accessibility of technological knowledge on a scale that makes it impossible to control, have led to new proportions of crime. A major weakness of digital technology is how easily unauthorized and illegal reproduction and distribution of digital objects is achieved and this activity is threatening to become the worst enemy of the digital era [1].

Watermarking adds a level of protection to the copyright of digital assets. It is the process of making deliberate alterations in a digital object, providing that they can be detected in the future. This requirement is meant to determine the paternity of the object. Watermarking is based on the existence of a region that can harbour noise in the object, inside which it is possible to produce small changes that degrade slightly the object without, however, causing damage to its fundamental attributes.

The deliberate alterations to the digital object are the *marks* and the set of embedded marks is the *watermark*. The marks are applied by an *encoder*. The *detection* of the watermark is achieved through the use of a key. Since the term *digital watermarking* was coined in 1993 when [2] presented two techniques to hide data in images, a large number of watermarking methods has been proposed for multimedia [3], digital documents [4], software [5] and, more recently, databases [6].

Two categories, depending on application, distinguish watermarks: *robust* watermarks for ownership verification and *fragile* watermarks for tamper detection. The purpose of a robust watermark is to resist a variety of attacks and legitimate users' data modifications, and categorically determines intellectual property. The purpose of a fragile watermark is for it to be damaged or destroyed by even the slightest data manipulation, thus determining categorically (and possibly localizing) any attack directed at the integrity of a digital object.

In the context of databases, the copyright protection is essential where it concerns sensitive data or data to be sold from a collecting institution *A* to an institution *B* (outsourcing), for uses such as data mining. Independently of the sale, institution *A* retains the copyright, while institution *B* holds the right to use, but not to sell the data to another institution.

The main contribution of this paper is the proposition of a novel watermarking scheme for numeric database attributes which is efficient in defeating a range of attacks that may be used to remove or destroy the mark. The proposed scheme is *multipurpose* because it can be used for both watermarking (*i.e.*, the same bit string is embedded and detected in every database copy) and fingerprinting (*i.e.*, a different bit string is embedded and detected in each database copy). This paper focuses on studying the performance of the proposed model as a watermarking scheme. The watermark might be any digital object related to the underlying data, for example an image, a logo, a text message, a sound, a speech signal, *etc.* The encoding algorithm can be applied to each tuple independently, therefore, the proposed method has the property of *incremental updateability*, *i.e.*, the watermarked database can support normal user modifications (insertions, deletions and updates) by simply applying the encoding algorithm to those involved tuples, without affecting any other ones.

The paper presents experimental results for a variety of parameter settings that show that the method can thwart efficiently a number of possible attacks, pointing to its practical robustness in real-world database watermarking applications.

Section II discusses previous work in watermarking relational data and draws a list of possible attacks against a relational database. Section III introduces the notations and parameters used in the paper. Section IV defines and describes the proposed method. Section V provides a brief discussion on the efficiency of the proposed method in real-world database applications, in comparison to the efficiency of other existing watermarking techniques. Section VI reports on an extensive experimental performance analysis of watermark tolerance to several possible attacks against the database. Section VII summarizes and suggests directions for further research.

II. RELATED WORK AND POTENTIAL ATTACKS

In the rich body of literature on watermarking multimedia data, most of the techniques were initially developed for still images [3] and later extended to video and audio sources [7]. These methods do not apply in the context of relational data because an important parameter in their operating lies in the fact that multimedia and software objects are of value only when they are entire: it is not possible to maintain the usefulness of the objects if parts are arbitrarily removed from them or added to them. In the case of databases, the insertions, deletions and updates of tuples constitute the more familiar processes in the framework of their operation. Also, in a database relation every tuple is a separate object (entity) and should be protected independently.

The fundamental objective of watermarking methods for relational data is to deliver efficient performance with respect to the following important metrics:

- the storage cost for the maintenance of the secret keys and other useful information (if any) that are required to maintain secrecy for the detection phase,
- the time cost required to embed the watermark and detect it, subsequently, in a suspicious relation,
- the ability of a relation to remain watermarked after modification operations (insertions, updates and deletions of database tuples), and,
- the sensitivity of the method to malicious attacks.

Perhaps the most well-known robust watermarking scheme for relational data is the one proposed in [8] whereby a small portion of numeric data is changed according to a secret key in such a way that this change can be detected for the purpose of ownership proof. Since the method just embeds a meaningless watermark, so that it can only determine whether the database is indeed watermarked, it cannot be used for meaningful embedding information. Another drawback is that a very high or very low percentage of marks has to be detected in a suspicious database to verify ownership, otherwise the method cannot decide whether the "unlike" watermark is a result of an attack or because no certain watermarks exists. This work has been extended in [9, 10] to allow meaningful multiple-bit watermarks to be embedded as well.

Another popular robust multibit watermark scheme for numeric data is proposed in [11] in which the tuples are securely divided into nonintersecting subsets. A single watermark bit is embedded into each subset, by modifying the distribution of tuple values. However, the capacity of the watermark is limited and the method has to record an extra subset classifying information, which is much larger than the size of the watermark, and safe storage, as well as space needed, are at question. Also, the scheme is not suitable for database relations that need frequent updates, since frequent data modifications may destroy the watermark and it is very expensive for the watermarking method to re-watermark modified database relations. Reference [12] extended this work, making it resistant to modifications and alteration attacks, however the subset information that needs to be stored and be given as input to the watermark detection process remains high.

In [13] a gray image is used as a watermark. There is no guarantee here that the marked data are still usable because the method reset their whole decimal fraction: this alteration may be so significant that normal application of data will be affected. Also, in [14] a speech signal is used as a watermark. Recent progress has expanded types of cover data to non-numeric data [15], categorical data [16], XML data [17], streaming data [18], data cubes [19], *etc*. Reference [6] makes a detailed survey of the literature on watermarking methods for relational data.

Regarding the robustness of a watermarking scheme, both malicious actions and normal user modifications (insertions, deletions and updates) should not wipe-off the watermark. Lets assume Alice is a database owner and Mallory is a hypothetical malicious user. Mallory plans to acquire rights over the protected data. His best-known approaches to achieve this are the following:

Data-weathering attacks: These attacks aim at the destruction of the watermark by making changes in the values of some of the least-significant bits of the data. Examples of this attack are the *deterministic bit-flipping* (which is performed by changing the value x of selected bits to $1 - x$), the *randomized bit-flipping* (which is performed by setting the value of selected bits randomly to 0 or 1, according to the independent toss of a fair coin), the *bit-setting* (which is performed by setting the value of all the selected bits to 0 or 1, independently of their original value) and the *roundin*g of the values of the watermarked data.

Subset-deletion attacks: Mallory deletes some tuples from the watermarked relation, aiming, on the one hand, to ensure that the remaining tuples acquire a high degree of importance and, on the other hand, to ensure that the watermark detection process will fail.

Pseudo-property statement attacks: Mallory incorporates his own watermark Y' in the relation R that is already protected by a legal watermark Y and claims that the watermark Y' pre-existed the watermark Y, and thus that the data are his own intellectual property. In most cases it is easy to defeat this attack by detecting both watermarks and analysing the existing distortion in the tuples that have been marked by both watermark encoding procedures.

Comparative attacks: Mallory compares different versions of the same relation that are likely to bring different watermarks; on the basis of the differences between the versions of the relation, he locates and removes the watermark.

III. NOTATIONS

Consider a database relation R which scheme is $R(P, A_0, A_1, \ldots, A_s)$, where P is the primary key and A_i ($i = 1, \ldots, s$) is a numeric attribute candidate for watermarking. Let there be n tuples in R, a fraction $1/\gamma$ of which will be used for the encoding of the watermark, where γ is a user-defined parameter.

Since a robust watermarking scheme inevitably introduces small distortions to the data, it is assumed that each attribute value can tolerate modifications of at least ξ least-significant bits. For the sake of simplicity we assume that ξ is a constant number that is independent of the attribute value, although it could depend on the number of bits of the binary representation of the attribute value as well. Table I lists the notations and parameters that will be used throughout in this paper.

TABLE I. NOTATIONS

Notation	Explanation
R	a database relation to be watermarked
R	a tuple in *R*
r.P	the primary key value of tuple *r*
$r.A_i$	the numeric attribute A_i of tuple *r*, where $i = 1, \ldots, s$
W	the watermark, represented as a binary string of length $L = \lvert W \rvert$ bits
Ξ	the number of least-significant bits in an attribute
$1/\gamma$	the fraction of tuples that are selected for watermarking
K	a secret key
S()	a cryptographic hash function

IV. THE PROPOSED APPROACH

In this section the proposed method for watermarking numeric relational data is presented. Without loss of generality, the watermark is assumed to be a meaningful binary string (*e.g.*, a logo) with length $L \in \aleph$ which is a power of 2[1]. An example of a watermark is illustrated in Fig. 1. The watermark is presented in its $w_0w_1 \ldots w_i \ldots w_{L-1}$ binary string form, where each black (white) square represents the bit value 1 (0) and the position *i* of each watermark bit w_i ($i = 0, \ldots, L - 1$) appears also in the figure, under its corresponding black or white square. As Fig. 1 shows, it the rest of the paper the position of each watermark bit will be represented in its binary form, which has length $\log_2 L$; these binary strings are called *watermark bits addresses* (WBAs). For example, in Fig. 1, the value of watermark bit w_3 is 1, while its WBA is 011.

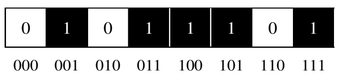


Figure 1. The binary representation of a watermark *W*.

On the other hand, the bits of every database attribute value are separated into two groups. The first group contains the *most-significant bits* (msbs) while the second group contains the ξ *least-significant bits* (lsbs) of the data. Fig. 2 shows the binary representation of an attribute value, where $\xi = 4$. It is assumed that the msbs cannot be modified by Alice or Mallory without rendering the data useless. The lsbs contain useful information which however can be changed to a limited extent. Therefore it is assumed that Mallory cannot reset all lsbs from an attribute value, even if he is able to predict which the lsbs are, without significantly degrading the value of the data.

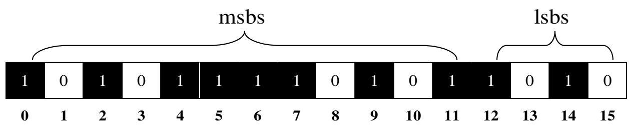


Figure 2. The binary representation of an attribute value with $\xi = 4$.

The proposed watermarking method consists of two algorithms, the encoding and the decoding algorithm, which are presented in the following subsections.

A. Watermark Encoding Algorithm

To simplify the discussion it is assumed that the database relation *R* contains two attributes: the primary key *R.P* and a secondary numeric attribute *R.A*. The encoding algorithm will watermark the attribute *R.A*, although it can be easily extended to spread the watermark among more than one attribute or relation. Fig. 3 show a snapshot of this hypothetical relation *R*, assuming that the attribute *R.A* has $\xi = 4$ lsbs.

Relation *R*					
R.P	***R.A* ($\xi = 4$)**	***R.A* in binary form**	**watermarked *R.A* in binary form**	**water mark-ed *R.A***	**disto-rtion (%)**
10	44,890	10101111 01011010	1010111101011100	44,892	0,00
13	2,842	00001011 00011010	00001011 00010011	2,835	0,24
14	65,000	11111101 11101000	11111101 11101001	65,001	0,00
18	570	00000010 00111010	00000010 00110000	560	1,75
25	3,672	00001110 01011000	00001110 01011001	3,673	0,02
28	20	00000000 00010100	00000000 00010100	20	0
42	7,625	00011101 11001001	00011101 11001111	7,631	0,07
47	54,540	11010101 00001100	11010101 00001010	54,538	0
48	3,608	00001110 00011000	00001110 00011011	3,611	0,08
51	2,544	00001001 11110000	00001001 11110111	2,551	0,27
				Average distortion (%):	0,24

Figure 3. An example of a database relation *R* which is watermarked.

The encoding algorithm is illustrated in Algorithm I. As an input, Alice has to provide the private key *K*, the number ξ of lsbs and the fraction $1/\gamma$ of tuples which will be *marked*. In the beginning, in Line 3, the algorithm selects the tuples that will be marked. A tuple *r* is selected if $S(K \parallel r.P) \bmod \gamma = 0$, where *K* is the secret key, *P* is the primary key of *r*, *S*() is a cryptographically secured hash function *S* [20] (*e.g.* SHA) and $\parallel$ denotes concatenation. Due to the uniqueness of the primary key, roughly one out of every γ tuples is selected for marking.

Every selected tuple *r* will be modified to store a part of the watermark. In particular, one of the bits of its *r.A* value will be selected to store a watermark bit *w*; in the sequel this bit of *r.A* will be called *marked bit*. However, a set of $\log_2 L$ other *r.A* bits will be chosen to store the WBA of *w* in the binary representation of the watermark *W* (recall Fig. 1). This set of *r.A* bits is called *WBA set* and it is consisting of msbs and/or lsbs of *r.A*.

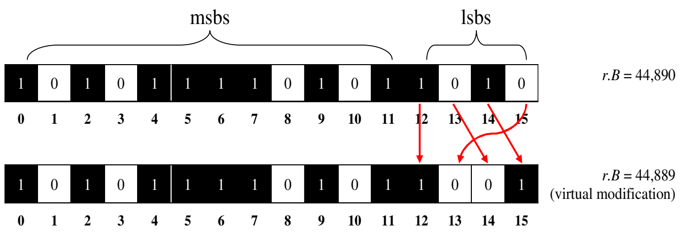


Figure 4. Rearranging the lsbs of the *r.B* attribute value of tuple *r*.

Supposing Mallory is aware of this selection strategy, he might attack randomly one or more lsb columns in order to destroy the watermark. To defend against this, the encoding algorithm, firstly, in Line 4 copies *r.A* into a temporary attribute *r.B* and, secondly, in Line 5 it creates a secret rearrangement for the lsbs of *r.B* value, making thus impossible for Mallory to locate which lsb is the marked bit, with high probability. This re-arrangement is carried out as follows: 1/ a hash digest *S*() for each *r.B* lsb is calculated, based on the primary key, the secret key and the position of each *r.B* lsb. 2/ the hash digests are sorted in an increasing order and the *r.B* lsbs are re-arranged according to this order. For example, let's assume that *r.B*'s

[1] If the watermark's length is not a power of 2, some random bits can be added appropriately, for example on its rightmost side.

value is the one that appears in Fig. 4 and that the hash digest for the lsb in the 12th / 13th / 14th / 15th position of *r.B* is correspondingly S(K||P||12) / S(K||P||13) / S(K||P||14) / S(K||P||15) = FFFA3300 / AA4F4127 / 00001044 / F4450182. By sorting these values, we find that the lsb value on the 12th / 13th / 14th / 15th position will be moved accordingly to the 12th / 14th / 15th / 13th position, *i.e.* the value of the lsb in the 12th position will not move, while all the other lsbs will be transferred into different positions. Fig. 4 illustrates the final result of this rearrangement. It must be stressed that these modifications in the *R.B* attribute values do not harm the original data, which have been kept safe in the *R.A* attribute.

```
Algorithm I: the Encoding() operation
Input:   a relation R, the watermark W, the secret key K
         and the parameters • and • (which are known only
         to the database owner).
Output:  the watermarked relation R, the lsb column of the
         rearranged version of R.A that was selected per
         tuple to store a watermark bit w, the (msbs and
         lsbs) bit columns of the rearranged version of
         R.A that store the WBA set of w.
1:    Add the temporary attributes B and C into R;
          // R.A, R.B & R.C share the same domain value
2:    FOR each tuple r ∈ R DO {
3:       IF S(K || r.P) mod • = 0) THEN {
4:          r.B = r.A;
5:          Re-arrange bits in r.B;
         }
      }
6:    FOR i = 0 TO |R.B|-1 DO
7:       Calculate the distribution of 1s in every
            bit column i of R.B attribute;
8:    Select the top $\log_2|W|$ msbs columns of R.B with 1s
         distribution as closer as possible to 50%,
         to form the WBA columns set;
9:    i = 1;
10:   WHILE (i ≤ ξ) AND (the assignment from r.B to W
            is not uniform) DO {
11:      Replace a msb column in the WBA column set with
            the next available lsb column of r.B with 1s
            distribution as closer as possible to 50%;
12:      IF (the assignment from r.B to W is not
            uniform) THEN {
13:         FOR each tuple r ∈ R DO {
14:            IF S(K || r.P) mod • = 0) THEN {
15:              r.C = r.B;
16:              Modify appropriately the lsbs of r.C that
                    belong to the WBA columns set in order
                    to achieve uniform assignment of the
                    data tuples to the watermark W's bits;
               }
            }
         }
17:      i++;
      }
18:   IF S(K || r.P) mod • = 0) THEN {
19:      Mark r.C;
20:      Perform in r.C the opposite re- arrangement of
            bits that was performed in r.B value in Line 5;
21:      r.A = r.C;
      }
22:   Drop the temporary attributes B and C from R;
23:   END;  // R is now watermarked
```

Algorithm I. The encoding algorithm.

The bits' re-arrangement process that appears in Line 5 can be extended to re-arrange also the *r.B*'s msbs as well, however this is optional since it is assumed that any alteration attack on the msbs will totally destroy the value of the data.

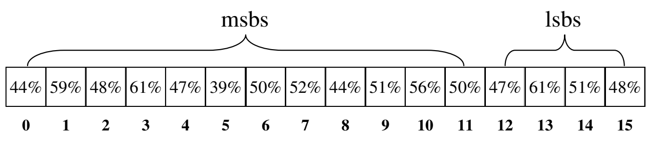


Figure 5. The distribution of 1s in each bit column in the re-arranged *R.B* attribute.

In the next step, the columns of the 'rearranged' *r.B* value which will store a watermark bit *w* needs to be selected, together with its WBA set. The bit columns of the rearranged *R.B* attribute that will be selected to store the WBA set will be the ones with bit values distribution of 1s and 0s as closer as possible to 50%. To understand this necessity we may assume an extreme case where this procedure mistakenly selects the *R.B* bit columns that have 0% or 100% distribution of 1s. Therefore, all tuples in *R* will have the same bit value for the selected columns (for example, '0'/'1'/'1', correspondingly, in the first/second/third selected column, assuming that $\log_2 L = 3$). Therefore the selected columns will define the same WBA columns set for every tuple (for example, '011' for every tuple) and each tuple will be assigned to the same watermark bit *w* (for example, to the watermark bit in the '011' position in the watermark *W*). This strategy would produce an extremely unbalanced assignment of data tuples to the watermark bits; in particular, all the tuples would point to the same watermark bit. Fig. 5 shows the distribution of 1s for a hypothetical example of a rearranged attribute *R.B* with data values to a maximum length of 16 bits (please recall that rearrangement might have been performed also in the msbs of *R.B*).

Therefore, in Line 8, the encoding algorithm selects separately the msbs columns with 1s distribution that is as closer as possible to 50%. Regarding the example of Fig. 5, assuming that three bits of *R.B* are required for storing a WBA, the algorithm will select the set of msbs in columns: 6th, 11th and 9th, starting by the column with 1s distribution closer to 50%. The marked bit, however, can be anyone of the lsbs; for the running example we may randomly select the bit column in the 12th position. The positions of the selected *R.B* bit columns need to be stored in a safe place since they have to be provided as an input to the decoding algorithm, which will be presented in the next subsection.

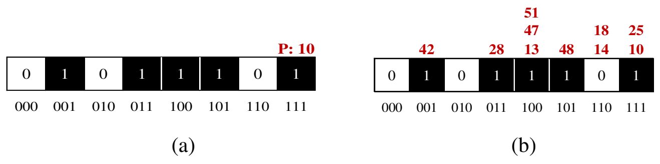


Figure 6. (a) A tuple *r* with primary key *r.P* = 10 is attached to a watermark bit, and (b) every data tuple in *R* is attached to a watermark bit.

In the next step we shall use the selected WBA columns set to actually route every tuple (using this set of bits) to point to a specific watermark bit. In the rearranged *r.B* example of Fig. 4, we may see that the binary string which is constructed by the data bits in the 6th, 11th and 9th bit columns is the '111'. As it is illustrated in Fig. 6a, this value directs the tuple to the 7th position of the watermark. For illustrative purposes the tuple's primary key is also "attached" on top of this watermark bit. Performing the same operation for each tuple of the relation *R* of Fig. 3, the result may look like the one in Fig. 6b.

In the next step, in Line 10, it is checked if the database tuples that will be marked, have been distributed uniformly to the watermark bits. For example, assuming that g tuples will be marked, this step checks if at least $\lfloor g/L \rfloor$ database tuples have been assigned to each one of the L watermark bits. The goal here is to assign an almost equal number of tuples to each watermark bit. If this is not the case (an example appears in Fig. 7a), and if $\xi \geq 1$ then in Line 11 the last selected msb (according to the procedure described in Line 8) is replaced with one lsb (since $\xi \geq 1$, we have at least one lsb). Afterwards, it is checked again if the tuple distribution to the watermark bits is uniform. The lsb that Line 11 chooses is the one with distribution of 1s as close as possible to 50% among the lsbs. In our running example of Fig. 5 this step would be translated into a replacement of the 9th bit column of *R.B* with the 14th bit column, which is one of the rearranged *R.B* lsb columns.

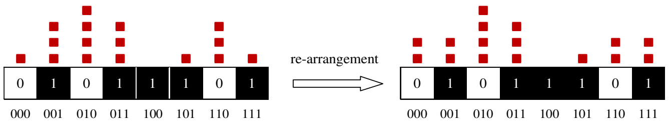


Figure 7. Modifying one lsb address bit.

In Line 12, if the tuples still have not been assigned uniformly to the watermark bits (like in the example of Fig. 7a), the algorithm in Line 16 will try to achieve this by modifying accordingly the lsb(s) that participate in the WBA group. The goal is to transfer some tuples from watermark bits with too much tuples to watermark bits with less tuples, by modifying the selected *R.B* lsb(s). If this will still not produce a uniform tuple distribution, then (in Line 12 again) if there are still available lsbs for selection, the previous step is repeated and one more msb is replaced by the next available lsb. In the example of Fig. 5, since $\xi = 4$, in this step we would replace the 11th (msb) bit column by the 15th (lsb) bit column of *r.B*.

We have to note that the more lsbs participate in the WBAC columns set, the further away we can move a tuple from one watermark bit to another, e.g. if one lsb participates in the WBAC set, then every tuple can move one position on the left or on the right, depending on the original value of the selected lsb (*i.e.*, if necessary, we may change the value of the lsb from '1' to '0', or vice-versa, thus moving the tuple from one position to another). However, if the WBAC set is constituted of one msb and two lsbs, then we may change the value of these two lsbs (*i.e.*, we may change the two selected lsbs values, for example, from a hypothetical '01' value to '11' or '00' or '10', thus moving the tuple from one position to three other alternative neighbouring positions). Of course, only the lsbs (if any) in the WBAC set are candidates for modification. Fig. 8a shows a hypothetical tuple distribution with one msb and two lsbs in the WBA columns set and Fig. 8b shows the uniform final tuple distribution on the watermark bits after the appropriate modification of the two lsbs in the WBA set.

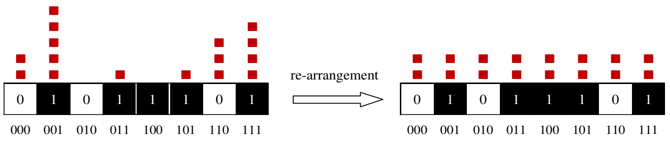


Figure 8. Modifying two lsbs address bits.

After the final assignment of each tuple to a watermark bit, in Line 19 the marked bit of the tuple is modified by setting it to be equal to the corresponding watermark bit. In the example of Fig. 6a, since it had previously been decided that the marked bit of the tuple would be the one in the 12th column of the rearranged *R.B* attribute, here the bit in the 12th column will be set to 1, which is the value of the corresponding watermark bit. In Line 20, the (lsb) bits of the rearranged watermarked tuples are re-ordered into their original positions, carrying any modifications in the WBA bits set and in the marked bit.

B. *Watermark Decoding*

During the decoding process, an operation reverse to the encoding operation is performed, to extract the watermark *W*' from a suspicious database *R*' and compare it to the encoded watermark *W*. Therefore, Alice has to recall the parameters *K* and γ, together with the position of the WBAs columns set and the position of the marked bit in the rearranged version of *R.A* attribute. The algorithm is formally illustrated in Algorithm II.

```
Algorithm II: the Decoding() operation
Input:   a suspicious relation R', the secret key K, the
         parameter •, and the position of the WBAs columns
         set together with the position of the marked bit
         in the rearranged version of R.A.
Output:  an extracted watermark W'
1:   FOR each tuple r ∈ R DO
2:     IF S(K || r.P) mod • = 0) THEN {
3:       Re-arrange bits in r.A;
4:       Based on the positions of the WBA columns set,
            assign tuple r to a watermark bit of
            an unknown watermark W' with length
            equal to 2^|WBA column set| bits;
         }
5:     Based on the tuples which were assigned to each
         watermark W' bit and also based on a majority
         voting rule, construct the W' watermark;
6:   RETURN W';  // W' is the extracted watermark
```

Algorithm II. The decoding algorithm.

In order to discover whether a tuple has been marked, in the beginning, the decoding algorithm performs the same hash operation as the encoding algorithm. If the tuple has been marked, in Line 3 the decoding algorithm performs the same re-arrangement of bits as in the encoding. The rearranged bits that construct the WBA bits set are then extracted (for example, '011') and on this basis the selected tuple is assigned to the corresponding bit of an unknown watermark *W'* with length equal to the number of columns in the WBA columns set. In Line 5, the marked bit value is then extracted using a *majority voting rule*, and it is stored to construct a possible watermark *W*'. The watermark *W'* in the sequel needs to be compared to the marked watermark *W* and if they match or are similar, then ownership can be claimed. The similarity percentage between *W'* and *W* which provides a certainty with regard to whether or not the embedded watermark exists in the database is decided by the user, however, our experiments indicated that a 75% similarity between the extracted and the embedded watermark is a good choice.

As an example, in Fig. 9 we may consider the tuple *r* with a watermarked *r.A* value which in binary form is 10101111 01011100. We may assume also that the WBA column set is constructed by the 6h, 15th and 14th columns of the appropriately rearranged *r.A* attribute and that also the marked bit is in the 12th column. The rearranged version of *r.A* has then to be constructed, which is depicted in Fig. 9. We now use the WBA column set {6, 15, 14} to get the address to the watermark bit: '101'. We also retrieve the marked bit from the 12th column to get the watermark bit value: 1. Therefore, the watermark bit in the $(101)_2 = 5^{th}$ position of *W′* should have value 1.

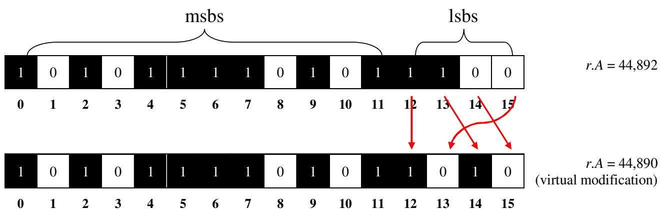


Figure 9. Rearranging the lsbs of the *r.A* attribute value of tuple *r*.

If the inspected database has been maliciously modified, two different tuples assigned to the same watermark bit may have different marked bit values. To deal with this problem the decoding algorithm in Line 5 uses a majority voting rule of 1s or 0s, in order to estimate the correct value of the assigned watermark bit. It does so by creating a counter for every watermark bit which is incremented if a tuple assigned to this watermark bit indicates that the watermark bit's value should be 1 or decremented if the watermark bit's value should be 0. Once all tuples have been processed, the watermark bit's value is set to 1 if the counter is positive and greater than a specified threshold value, or it is set to 0 if its counter is negative and smaller than the negative threshold value; otherwise the watermark bit has an unknown value.

V. DISCUSSION

A. *Encoding by grouping without the need to store large group-related information*

The proposed scheme embeds the watermark on the group basis. The tuples are uniformly divided into $|W|$ groups, using a mixed sequence of $\log_2|W|$ msbs and lsbs of the attribute that will be watermarked and, afterwards, one bit of watermark information is stored in each group. Therefore, the only information which needs to be saved in a safe storage regarding this process is $\log_2|W| + 1$ short integer values. It is obvious that a great advantage of the proposed method against other group-based watermarking techniques (such as [11, 12]) is that there is no need to store large quantities of information related to the constructed groups of tuples, like the number of groups, the number of tuples in each group, the tuples that define the borders of each group, the parameters of the function which distributed the tuples in the groups, and any other related information regarding the groups' content. Therefore the proposed method offers an almost blind decoding process.

B. *Database updates*

If the need to update the watermarked database arises, it will definitely affect the encoded watermark. To ensure the robustness of the watermark, we only need to watermark the modified tuples in order to keep the distribution of tuples per watermark bit as uniform as possible. This operation will not affect the rest of tuples in the database. This is another advantage of the proposed scheme over other similar schemes which need to re-watermark the whole database relation or at least the tuples that are enlisted in the same group with the modified tuples (for example, [11, 21]).

C. *Regarding databases without a primary key*

In the proposed scheme it is assumed that the database to be watermarked has a primary key. The primary key is used for sorting the tuples' (msbs and) lsbs columns. For databases without a primary key, we can easily incorporate the scheme in [9], where a virtual primary key can be constructed from some msbs of each tuple's attributes.

D. *For error-intolerant databases*

It is assumed that the database relation to be watermarked has only numeric attributes and can tolerate small errors introduced by watermark encoding algorithm. The proposed scheme can easily be modified to be applicable if a database relation does not have numeric attributes or if the numeric attributes cannot tolerate any modifications. For example, instead of changing the lsbs values, an extra attribute can be created to store the watermark.

VI. EXPERIMENTS

Some experimental results that illustrate the robustness of the proposed method are presented in the sequel. The experiments we performed using database relations containing two integer attributes with uniformly distributed synthetic data, one attribute serving as the primary key and one as the attribute to be watermarked. We ran experiments on an Oracle Database 11g Release 2 using Oracle Call Interface (OCI) connectivity on a Windows 7 workstation. Each experiment cycle is comprised of three phases: a watermark encoding operation, an attack and a watermark decoding operation. The performed attacks are the subset-deletion attack, the randomised-flipping attack and the bit-setting attack. Comparative attacks are not effective because we assume that the same watermark is embedded in every version of a relation, therefore there is no meaning in the search for differences between versions of the same watermarked relation. The output of each experiment cycle is the detection success rate, *i.e.* the fraction of bits of the extracted watermark *W′* that match with the embedded watermark *W*. Every experiment was repeated ten times, each time with a different synthetic database.

A. *The Role of the Size of the Watermark*

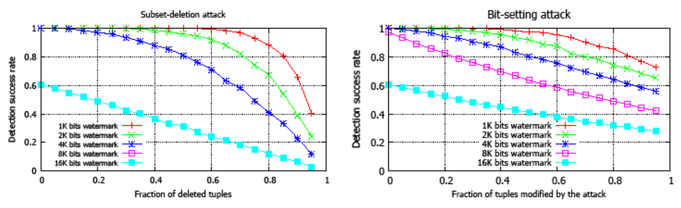
In this experiment the parameters that appear in Table II remain constant. We ran experiments with the following watermark sizes (in bits): 1K, 2K, 4K, 8K, and 16K. The benchmark database contains 10,000 tupes and we mark all of them, *i.e.* $\gamma = 1$.

The first attack to be reported in Fig. 10a is the subset-deletion attack. By deleting the 0.2 fraction of the tuples we mean that the malicious user keeps only 80% of the initial tuples. The results are somehow expected since each tuple in the remaining database relation will make a correct match because every tuple is marked and also 'untouched' by the attack. For example if we delete the 0.7 fraction of the dataset, the remaining database will contain about 3,000 tuples which are uniformly distributed over the watermark bits. Thus, the detection success rate is analogous to the number of watermarked tuples attached to each watermark bit. Therefore, the smaller the watermark, the higher the retrieval success, for every fraction of deleted tuples. For example, for a small watermark of 1K bits and by deleting 95% of the database tuples, 40% of the watermark will survive.

TABLE II. PARAMETERS USED IN THE EXPERIMENTS STUDYING THE ROLE OF THE SIZE OF THE WATERMARK.

Database size (in tuples):	10,000
Number ξ of lsbs:	6
Percentage of marked tuples:	100%
Number of bits modified by the attack per tuple (applies only to randomized bit-flipping and bit-setting attacks):	2

Another interesting conclusion is regarding the importance of the appropriate selection of the watermark size in respect to the size of the database. For example, in the graph we may see that for a watermark of 16K bits, the detection success rate is low, mainly because the database size in tuples is smaller than the watermark size in bits, which means that not all the watermark bits were recorded into the database.



(a) (b)

Figure 10. Evaluation of the robustness of the method with regard to the watermark size in (a) subset-deletion attack, and (b) bit-setting attack.

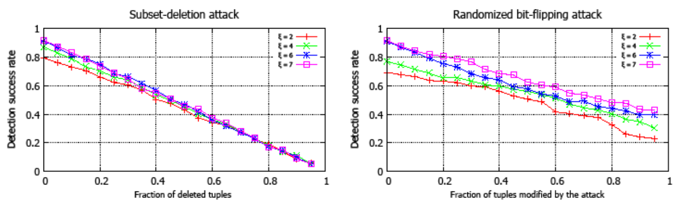
The results in both randomized bit-flipping and bit-setting attacks are quite similar; therefore, we illustrate in Fig. 10b only the results for the bit-setting attack. When the fraction of tuples modified by the attack is small, we get a high detection success rate. However, when the tuples modified by the attack increase, the graph indicates that the watermark size affects almost linearly the watermark retrieval. Therefore, the higher the percentage of tuples modified by the attack, the smaller the watermark needs to be in order to survive.

B. The Role of the Number of Available lsbs

The values for the most important parameters are depicted in Table III. The graph in Fig.11b for the randomized bit-flipping attack shows that the efficiency of the watermarking method improves slightly when the number ξ of lsbs increases. This happens because the increase of available number ξ of lsbs allows the watermarking method to spread the marked bit and the WBA bits in more bit columns in order to achieve uniform assignment of tuples in groups (Line 10 of the encoding algorithm). Therefore, the more lsbs are used by the encoding algorithm, the higher the percentage of the survived watermark fraction is, since it is also more likely to not invert a bit that is used by the algorithm. In the subset-deletion attack, by definition the bits are not modified, therefore this conclusion does not apply in Fig. 11a; hence the watermark retrieval success rate depends only in the fraction of the tuples modified by the attack.

TABLE III. PARAMETERS USED IN THE EXPERIMENTS STUDYING THE ROLE OF THE NUMBER OF AVAILABLE LSBS.

Database size (in tuples):	10,000
Watermark size (in bits):	2,048
Percentage of marked tuples:	100%
Number of bits modified by the attack per tuple (applies only to randomized bit-flipping and bit-setting attacks):	2



(a) (b)

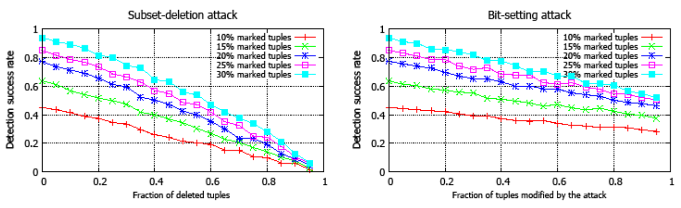
Figure 11. Evaluation of the robustness of the method with regard to the number of available lsbs in (a) subset-deletion attack, and (b) randomized bit-flipping attack.

C. The Role of the Percentage of Marked Tuples

The values for the most important parameters are depicted in Table IV. We ran experiments for the following percentages of marked tuples: 10%, 15%, 20%, 25% and 30%. The results in Fig. 12a and Fig. 12b for this experiment are as expected. The overall conclusion is that when everything else remains unchanged, the percentage of the watermarked tuples affects linearly the detection success rate of the watermark.

TABLE IV. PARAMETERS USED IN THE EXPERIMENTS STUDYING THE ROLE OF THE PERCENTAGE OF MARKED TUPLES.

Database size (in tuples):	5,000
Number ξ of lsbs:	6
Watermark size (in bits):	1,024
Number of bits modified by the attack per tuple (applies only to randomized bit-flipping and bit-setting attacks):	2



(a) (b)

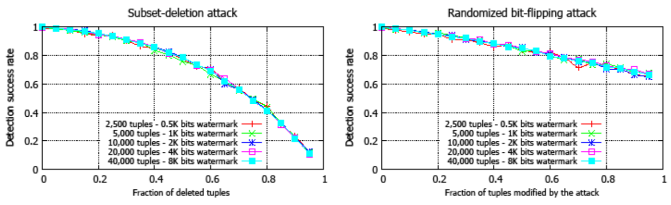
Figure 12. Evaluation of the robustness of the method with regard to the percentage of the marked tuples in (a) subset-deletion attack, and, (b) bit-setting attack.

D. The Role of the Ratio of Marked Tuples Over the Watermark Size

Table V illustrates the selected values for the parameters that were used in this experiment. Regarding to the number of tuples in the database and the watermark size, the following pair of values (in the <*number of tuples*, *watermark size in bits*> form) were used: <2,500, 0.5K>, <5,000, 1K>, <10,000, 2K>, <20,000, 4K>, <40,000, 8K>.

TABLE V. PARAMETERS USED IN THE EXPERIMENTS STUDYING THE ROLE OF THE RATIO OF MARKED TUPLES OVER THE WATERMARK SIZE.

Number ξ of lsbs:	6
Percentage of marked tuples:	50%
Number of bits modified by the attack per tuple (applies only to randomized bit-flipping and bit-setting attacks):	2



(a) (b)

Figure 13. Evaluation of the robustness of the method with regard to the ratio of marked tuples over the watermark size in (a) subset-deletion attack, and, (b) randomized bit-flipping attack.

The graphs in Fig. 13a and Fig. 13b show that as long as the ratio of marked tuples over the watermark size remains constant, the algorithm's robustness remains constant as well. The important consequence of this observation is that in case we have a very large database to watermark, we can estimate the robustness of the proposed algorithm with a random fraction of the database tuples and a correspondingly smaller random fraction of the watermark bits.

VII. CONCLUSION

The paper proposes an algorithm for watermarking numeric relational data. The algorithm sorts the bits of each tuple in a secret order and selects some of its data bits to route the tuple to a specific watermark bit and one data bit to be marked by the value of the assigned watermark bit. We showed that the algorithm can be easily implemented. Experimental study pointed to the robustness of the proposed scheme for a variety of parameter settings and of possible attacks. The study shows that the proposed scheme can be widely used in copyright protection. It is worth noting that, for the sake of simplicity, in the experimental study we did not adopt any error correction mechanism [22] to make the watermark more robust against the various kinds of attacks. In real-world applications it is expected that this methodology will protect the embedded information against noises; error correcting codes exist that have a correcting ability up to approximately 25% of the occurred errors [23].

In the future we aim at making correct watermark recovery decisions in view of other types of attacks, for example, brute force and mix-and-match attacks. We also plan to extent the proposed method to be used as a fragile watermarking scheme as well, for data authentication proposes, for example, by assigning properly a single data tuple to each watermark bit. Further research should also investigate new, non-numeric encoding domains, that is, categorical and alphanumeric attributes.

ACKNOWLEDGMENT

The author would like to thank Mr. Konstantinos Robotis for his invaluable contribution with regard to the implementation and evaluation process of this work.

REFERENCES

[1] CBS News: "Digital piracy stronger than ever", Online link: http://goo.gl/Ws2rZ, valid as of October 2010

[2] Tirkel A., Rankin G., Schyndel R., Ho W., Mee N., and Osborne C.: "Electronic watermark". *Proceedings of Digital Image Computing, Technology and Applications, DICTA 93*, pp. 666-673, 1993.

[3] Langelaar G.C., Setyawan I., and Lagendijk R.L.: "Watermarking digital image and video data: a state-of-the-art overview". *IEEE Signal Processing Magazine*, Vol.17, pp.20-46, 2000.

[4] Low S.H., Maxemchuk N.F. and Lapone A.M.: "Document identification for copyright protection using centroid detection". *IEEE Transactions on Communications*, Vol.46, No.3, pp.372-383, 1998.

[5] Collberg C.S, and Thomborson C.: "Watermarking, tamper- proofing and obfuscation – tools for software protection". *IEEE Transactions on Software Engineering*, Vol. 28, No. 8, pp. 735-746, August 2002.

[6] Halder R., Pal S., and Cortesi A.: "Watermarking techniques for relational databases: survey, classification and comparison". *Journal of Universal Computer Science (JUCS)* Vol.16, No.21, pp.3164-3190, 2010.

[7] Boney L., Tewfik A.H., and Hamdy K.N.: "Digital watermarks for audio signals". *Proceedings of the International Conference on Multimedia Computing and Systems*, pp.473-480, 1996.

[8] Agrawal R., Haas P.J., and Kiernan J.: "Watermarking relational data: framework, algorithms and analysis". *The VLDB Journal*, Vol. 12, pp 157-169, 2003.

[9] Li Y., Swarup V., and Jajodia S.: "Constructing a virtual primary key for fingerprinting relational data". *Proceedings of the ACM Workshop on Digital Rights Management*, pp. 133–141, 2003.

[10] Li Y., Guo H., and Wang S.: "A multiple-bits watermark for relational data". *Proceedings of the Principle Advancements in Database Management Technologies*, pp.1-22, 2010.

[11] Sion R., Atallah M.J., and Prabhakar S.: "Rights protection for relational data". *IEEE Trans. Knowl. Data Eng.* Vol.16,No.12,pp.1509-1525, 2004

[12] Shehab M., Bertino E., and Ghafoor A.: "Watermarking relational databases using optimization-based techniques". *IEEE Trans. Knowl. Data Eng. (TKDE),* Vol. 20, No.1, pp.116-129, 2008.

[13] Zhang Z., Jin X., Wang J., and Li D.: "Watermarking relational database using image". *Proceedings of the Third International Conference on Machine Leaning & Cybernetics*, Vol.3, pp.1739-1744, Shanghai, 2004.

[14] Wang H., Cui X., and Cao Z.: "A Speech Based Algorithm for Watermarking Relational Databases". *Proceedings of the ISIP 2008*, pp.603-606, 2008.

[15] Al-Haj A., and Odeh A.: "Robust and blind watermarking of relational database systems". *Journal of Computer Science*, Vol.4, pp.1024–1029, 2008.

[16] Sion R.: "Proving ownership over categorical data". *Proceedings of ICDE 2004*, pp.584-596, 2004.

[17] Gross-Amblard D., "Query-preserving watermarking of relational databases and Xml documents". *ACM Trans. Database Syst. (TODS)* Vol.36, No.1, Article No.3, 2011.

[18] Sion R., Atallah M., and Prabhakar S., "Resilient rights protection for sensor streams". *Proceedings of the Very Large Databases Conference*, pp.732–743, 2004.

[19] Guo J., Li Y., Deng R. H., and Chen K.., "Rights protection for data cubes". *Proceedings of the ISC*, pp.359–372, 2006.

[20] Schneier B.: *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, 1995.

[21] Sion R., Atallah M.J., and Prabhakar S., "On watermarking numeric sets". *Proceedings of the IWDW 2002*, pp.130-146, 2002

[22] Ambroze A., Wade G., Serdean C., Tomlinson M., Stander J., and Borda M., "Turbo code protection of video watermark channel". *IEEE Proceedings-Vision, Image and Signal Processing*, Vol.148, No.1, pp.54–58, 2001.

[23] Zhou X., Huang M., and Peng Z., "An additive-attack-proof watermarking mechanism for databases' copyrights protection using image". *Proceedings of the SAC 2007*, pp.254-258, 2007.