

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/250615085>

A novel blind reversible method for watermarking relational databases

Article in *Journal of the Chinese Institute of Engineers* · May 2013

DOI: 10.1080/02533839.2012.726041

CITATIONS

24

READS

318

1 author:



Mahmoud Farfoura

Applied Science Private University

12 PUBLICATIONS 507 CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:



Relational Database Watermarking [View project](#)

A novel blind reversible method for watermarking relational databases

Mahmoud E. Farfoura^{1*}, Shi-Jinn Horng^{1,2}, Xian Wang¹

¹Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei 106, Taiwan

²Department of Electronic Engineering, National United University, Miao-li 360, Taiwan

Abstract

Digital Watermarking technology has been introduced in the past few years, not only to ensure the ownership of digital media, but also to ensure the integrity of those digital media. Reversible watermarking (which is also called invertible watermarking, or erasable watermarking) enables one to recover the original data after the contents have been authenticated. Such reversibility is highly desired in some sensitive database applications, e.g. in military and medical data. Permanent distortion is one of the main drawbacks of the entire irreversible relational database watermarking schemes. In this paper, we introduce a novel blind reversible watermarking method that ensures ownership protection in the field of Relational Database watermarking (RDB). Whereas previous techniques have been mainly concerned with introducing permanent errors into the actual data, our approach ensures 100% recovery of the original database relation after the watermark has been detected and authenticated. In the proposed method, we utilize a reversible data embedding technique called Prediction-error Expansion (PE) on integers to achieve reversibility. The watermark detection can be completed successfully even when seventy percent of watermarked relation tuples are deleted. The experimental result shows that the blindness and robustness of this approach can withstand several kinds of database attacks.

Keywords:

relational database; reversible watermarking; copyright protection; robustness; blindness; database attacks

* Corresponding author {TEL: +886975729435; EMAIL: mfarfora@yahoo.com}

1. Introduction

Copyright protection for owners is becoming more and more necessary due to the rapid growth of the Internet, the wide development of digital multimedia contents, and easier distribution. As an important area in information hiding [1-3], digital watermarking provides a promising method of protecting digital data from illicit copying, and manipulation by embedding a secret code directly into the data. Digital watermarking allows the user to add a layer of protection to the digital media content by identifying copyright ownership and delivering a tracking capability. Accordingly, it monitors and reports where the user's digital media contents are being used.

Cryptography provides a mean for secure delivery of content to the consumer only. Not all legitimate consumers are trustworthy and an untrustworthy consumer may alter or copy the decrypted content in a manner that is not permitted by the content owner. However, cryptography provides no protection once the content is decrypted, which is required for human perception. Watermarking complements cryptography by embedding a message within the content [10].

The desired properties of the proposed watermarking method:

1. Prevents illegal embedding and verification: The whole process is governed by a key; only an authorized person who has a key can embed, extract, and verify watermarks. This prevents unauthorized persons from inserting a false watermark or illegally verifying watermarks.

2. Blind detection: At the time of watermark extraction, the watermark can be extracted without need for the original database. Solely the key, which is typically used during the watermark insertion, is required.
3. Invisibility: The watermarked tuples and attributes i.e. the position of watermark cannot be detected by an attacker.
4. Robustness: The watermarking method should be able to survive and resist several database attacks such as deletion, insertion, and modification attacks.
5. Reversibility: The original non-watermarked database relation can be fully recovered after the watermark has been detected and authenticated.

Significant research efforts [1-9] have been expended on signal processing and multimedia works (e.g., images, video and audio). The existing watermarking techniques for multimedia cannot be applied directly to watermark relational databases. There are differences in the nature of the data properties. As we know, multimedia data are highly correlated; however tuples in a relational database are independent and discrete. The tuples can be added, deleted or modified frequently in either benign updates or malicious attacks.

Comparatively, watermarking of relational databases started to receive attention due to the increasing use of database systems in many real-life applications [11-17]. Before now, almost all watermarking schemes have been irreversible (the original relation cannot be restored from the watermarked relation). Medical and military data are examples where reversible database watermarking might have a crucial importance including protecting rights of outsourced relational databases and because loss of these types of data is not acceptable. Another purpose of reversible watermarking is providing shareware or trial versions of specific database applications where the original database version can be recovered after the consumer buys the license for that

application.

In this paper, we propose a novel blind reversible method for watermarking relational databases which proves the true ownership of the database owner. In the proposed method, an identification image is converted into a bit stream which is embedded in the fraction portion of numeric attributes by adopting a reversible data embedding technique called Prediction-error Expansion (PE) proposed by Diljith M. Thodi *et al.* [19]. After the watermarked database has been authenticated, the erasable watermark can be removed and the original database can be recovered.

The rest of this paper is organized as follows: Section 2 gives an overview of related research and preliminary background. Section 3 explains in detail the proposed watermarking method, including watermark insertion and watermark extraction. In section 4, we present experiments and analysis. Section 5 concludes this paper with summaries and suggestions for future work.

2. Related researches and preliminary background

The first well-known irreversible database watermarking scheme was proposed by Agrawal and Kiernan [11] for watermarking numerical values in relational databases. The fundamental assumption is that the watermarked database can tolerate a small amount of errors. It utilizes the pseudorandom number generator algorithm to identify the marked tuples and attributes, and utilizes the degree of error of the marked attributes. The private key, used for copyright verifiability, is the seed for the pseudorandom number generator algorithm. The watermarking scheme appears to be satisfactorily secure against several database attacks. However, there is no discussion on query preservation or data usability. Also, the scheme suffers from certain drawbacks in terms of database usability and correctness.

Agrawal and Kiernan's scheme [11] cannot be directly applied to watermarking categorical data since any bit change to a categorical value may render the value meaningless. To solve this problem, Sion [12] proposed a scheme to watermark a categorical attribute by changing some of its values to other values of the attribute (e.g., 'red' is changed to 'green') if such change is tolerable in certain applications. In [13], Li *et al.* extended the work by Kiernan, Agrawal [11] to provide for multi-bit watermarks in a direct domain encoding which, in turn, increased the length of embedded watermarks. Another related approach is to be found in [16], where the authors have proposed a fragile watermark technique to detect and localize alterations made to a database relation with categorical attributes. The main drawback of this scheme is that it relies on the physical location of tuples in the relation, so any reordering occurring will destroy the embedded watermark.

Sion *et al.* [14] presented a different approach to a robust watermarking scheme for relational databases. In their scheme, all tuples are securely sorted and divided into non-intersecting subsets. A single watermark bit is embedded into some tuples of a subset by modifying the distribution of tuple values. Watermark bits are embedded repeatedly and an error control coding scheme is employed to recover the embedded bits. This scheme is claimed to be robust against attacks such as data re-sorting and data transformations. However, the scheme is not suitable for database relations that need frequent updates, since it is very expensive to re-watermark the updated database relations.

Several reversible image watermarking schemes have been proposed [20- 23]. Tian [23] proposed the Difference Expansion (DE) techniques which usually generate some small values to represent the features of the original image. Then, they expand (enlarge) the generated values to embed the bits of watermark information. The watermark information is usually embedded in the

Least Significant Bit (LSB) of the expanded values. Then the watermarked image is reconstructed by using the modified values. In Tian's scheme, an integer transformation is defined as

$$g = \left\lfloor \frac{(x+y)}{2} \right\rfloor, \quad (1)$$

$$d = x - y. \quad (2)$$

For example, $x = 106$ and $y = 104$ are two adjacent pixels. Then the difference d and the average g can be computed as follows:

$$d = 106 - 104 = 2, \text{ and } g = \left\lfloor \frac{(106+104)}{2} \right\rfloor = 105. \text{ Here, } \lfloor x \rfloor \text{ denotes the greatest integer smaller}$$

than x (floor function). To embed a watermark bit $i = 1$ into the pixel pair, the difference d is represented using the binary format. Shift it left by one bit and append the watermark bit i into the vacant LSB. If l is the bit length of d (i.e., $d = b_{l-1}b_{l-2} \dots b_0$), then the new difference value d' can be obtained as

$$d' = b_{l-1}b_{l-2} \dots b_0i = 2 \times d + i = 2 \times 2 + 1 = 5. \quad (3)$$

Then, the new pixel values are given by

$$x' = g + \left\lfloor \frac{(d'+1)}{2} \right\rfloor = 105 + 3 = 108, \quad (4)$$

$$y' = g - \left\lfloor \frac{d'}{2} \right\rfloor = 105 - 2 = 103. \quad (5)$$

On the decoder side, the watermark bit can be extracted from the LSB of the difference value, and the original difference value can be restored.

$$d' = x' - y' = 108 - 103 = 5, \quad (6)$$

$$i = \text{LSB}(d') = \text{LSB}(101_2) = 1,$$

$$d = \left\lfloor \frac{d'}{2} \right\rfloor = \left\lfloor \frac{5}{2} \right\rfloor = 2 . \quad (7)$$

Then the inverse integer transform is given by Equations (8) and (9). Thus original pixel values x and y can be recovered completely.

$$x = \left\lfloor \frac{(x'+y')}{2} \right\rfloor + \left\lfloor \frac{(d+1)}{2} \right\rfloor = 105 + 1 = 106 , \quad (8)$$

$$y = \left\lfloor \frac{(x'+y')}{2} \right\rfloor - \left\lfloor \frac{d}{2} \right\rfloor = 105 - 1 = 104 . \quad (9)$$

Gupta *et al.* [24] proposed a reversible watermarking scheme adopting the DE to embed watermark bits into numeric attributes. The major drawback of this scheme is that it requires two attributes to embed a single watermark bit and the amount of distortion introduced in the attributes is directly proportional to the numerical difference of the two attributes. Since data in a relation is discrete and values are not related, it is expected this will cause much distortion in the final watermarked relations. Another problem with this scheme is that it is more subject to modification attack. Thus, any attempt to modify any one of the contributed attributes will fail the watermark extraction process plus failing to recover the original data. A single attribute carrying a watermark bit would be a much better option, and this is precisely our objective in this work.

Prediction-error Expansion (PE) proposed in [19] is employed instead of DE in this work. In PE a predictor instead of a difference operator is used, to create the feature elements into which expansion embedding will be done. A predictor better exploits the correlation inherent in the neighborhood of a pixel. The prediction errors are the feature elements into which expansion embedding is done. Consider a pixel with intensity y in a grayscale image and a watermark bit i to be embedded. A predictor operates on the neighborhood of pixel y and predicts its

intensity y^\wedge . The prediction error p_e is given

$$p_e = y - y^\wedge. \quad (10)$$

To embed a watermark bit i into prediction error p_e , p_e is represented using the binary format. Shift it left by one bit and append the watermark bit i into the vacant LSB. If l is the bit length of d (i.e., $p_e = b_{l-1}b_{l-2} \dots b_0$), then the new prediction error p'_e can be obtained as

$$p'_e = b_{l-1}b_{l-2} \dots b_0i = 2 \times p_e + i. \quad (11)$$

The modified prediction-error, p'_e and the predicted intensity y^\wedge , are combined to calculate the embedded pixel intensity y' (i.e. the watermarked value), where

$$y' = y^\wedge + p'_e = y - p_e + p'_e = y + p_e + i. \quad (12)$$

On the detector side, the intensity of the pixel is predicted using the same predictor as employed by the embedder. The predicted intensity will be y^\wedge (if the neighborhood has not been altered). The prediction error at the decoder is $y' - y^\wedge = p'_e$. The embedded bit is the LSB of p'_e $i = \text{LSB}(p'_e)$. The true pixel intensity is restored after calculating the original prediction error as

$$p_e = \left\lfloor \frac{p'_e}{2} \right\rfloor, \quad (13)$$

$$y = y' - p_e - i. \quad (14)$$

Since data in a database relation are discrete and do not correlate to each other, the notion of pixel prediction cannot be applied directly, so a slight variation of PE is applied. Hence, we assume that y is the fraction portion of any numeric attribute to be watermarked, and y^\wedge is any given value known at watermark embedding and extraction time. In the following section, we present more details about the proposed watermarking method.

3. The proposed watermarking method

In our method, an identification image will be converted into a stream of bits 0's, and 1's and embedded into numeric attributes to represent the watermark information and to prove the true ownership of a database relation. A tolerable change in the attributes values can be accepted, and will not affect the overall data usability [11]. Assume the data set R is a database relation and its schema has the form $R(P, A_0, A_1, \dots, A_{\alpha-1})$, where P is the primary key, $A_0, A_1, \dots, A_{\alpha-1}$ are the α attributes of R and η is the number of tuples in R . We assume that the primary key P cannot be changed by an attacker, otherwise data integrity, usability, and availability will be violated.

This is unlike [11], where the embedding affects every part of a numeric attribute, so a high distortion may be induced, which may influence the query preservation and data usability. We minimize the amount of distortion that will be incurred by restricting the embedding only into the fraction portion of the candidate numeric attributes. Hence, the amount of distortion is minimized, thus data usability is maintained. Table 1 summarizes the important parameters used in our algorithms.

Table 1. Notations used in this paper.

η	Number of tuples in the relation
R	Database relation to be watermarked
RW	Watermarked relation
W	Watermark bits to be embedded
A_i	Attribute i in relation R
t_i	Tuple i in relation R
α	Attribute numbers to be marked
γ	Fraction of tuples marked
K	The embedding key
sb	The length of watermark

We use a one-way hash cryptographic function result determined by the primary key (P) and the embedding key (K) to determine which tuples and attributes to mark. Usually it has the

form of $h = H(M)$, where M is the message. Besides, it bears the following characteristics: given M , it is easy to compute h ; yet when given h , it is hard to compute M , such that $H(M) = h$; given M , it is hard to find another message $M0$ such that $H(M) = H(M0)$. Several hash functions such as MD5 and SHA are good choices for this purpose [18]. For security, the embedding key (K) should be selected from a large enough key space so that it is computationally infeasible for an attacker to guess the key. We suggest the embedding key be chosen as in Equation (15):

$$K = H(\text{ID} \parallel \text{DB name} \parallel \text{version} \parallel \dots), \quad (15)$$

where ID is the database owner identity, \parallel denotes concatenation, DB name is the database name, version is the database version and $H()$ is the cryptographic hash function.

3.1 Watermark insertion

In this step, we embed an identification image into the relational data for representing the copyright information [17]. To be more precise, this copyright image should be embedded in the fraction portion of the candidate numeric attribute. First, we transform a meaningful image as the copyright information into a bit flow $W = \{W(i) | W(i) \in \{0,1\}, 0 \leq i < (m \times n) - 1\}$ (m, n are the image width and height). Figure 1 shows the Taiwan Tech. emblem to be embedded, the size of which is 32×32 pixels. We use this emblem as the copyright information, and the length of the corresponding binary sequence is $sb = 32 \times 32$.



Figure 1. Watermark image (Taiwan Tech. emblem).

Bit_encoding algorithm depicted in Figure 2, describes the process of embedding one watermark bit into fraction portion of the candidate numeric attributes. Get_int() method used to extract the integer part of a real number while Get_frac() retrieves the fraction portion. As

mentioned in section 2, Prediction-error Expansion (PE) assumes two intensities for each pixel, the original intensity y and the predicted one \hat{y} . Here we assume that y is the fraction portion of an attribute $t_i.A_j$ and \hat{y} is any value known at watermark insertion and extraction time and related to tuple t_i , say $H(t_i.P \parallel K)$ - denotes the hashed value of concatenated primary key of tuple i with the embedding key K . After subtracting \hat{y} from y to obtain the difference, and based on the watermark bit to be embedded by expanding the obtained difference, then by adding the expanded difference to $H(t_i.P \parallel K)$, we can achieve the encoding method.

As a working example, let the value to be watermarked be $t_i.A_j = 253.101$ and the watermark bit $i = 1$. Using `Get_frac(253.101)` sets $y = 101$ while `Get_int(253.101)` returns 253. Assuming the value of $\hat{y} = H(t_i.P \parallel K) = 5$, then $p_e = 101 - 5 = 96$. After embedding the watermark bit i into p_e , we have $p'_e = 2 \times p_e + i = 2 \times 96 + 1 = 193$. Then, to obtain the watermarked value y' we add p'_e to \hat{y} , so we have $y' = p'_e + \hat{y} = 193 + 5 = 198$. Next, the new (watermarked) value of $A = 253.198$ then we update the value to database.

```
=====
Bit_encoding( $K, w[idx], t_i.A_j$ )
Input: Embedding key  $K$ , watermark bit, numeric value for attribute  $j$  of tuple  $i$ 
Output: updated value for attribute  $j$  of tuple  $i$ 
1   $frac = \text{Get\_frac}(t_i.A_j);$            // returns the fraction portion of  $t_i.A_j$ 
2   $int = \text{Get\_int}(t_i.A_j);$            // returns the integer portion of  $t_i.A_j$ 
3  if ( $frac \geq H(t_i.P \parallel K)$ ) then
4     $diff = frac - H(t_i.P \parallel K);$ 
5     $temp = \text{Bin}(diff);$                // convert  $diff$  to binary value
6     $temp = temp \parallel w[idx];$        // || denote concatenate
7     $newdiff = \text{To\_integer}(temp);$      // convert  $temp$  back to integer value
8     $newval = newdiff + H(t_i.P \parallel K);$ 
9     $newval = \text{To\_number}(int, newval);$  // convert  $int$  and  $newval$  to number
10  $\text{Reflect\_update}(t_i.A_j, newval);$  //update field  $j$  of tuple  $i$  by  $newval$ 
11 end if;
```

12 **end;**

Figure 2. Bit_encoding algorithm.

Our approach inspires the method of [11] to pseudo randomly select some tuples and attributes for watermark insertion and extraction. Now, after the watermark information is randomized, it should be ready to be embedded into the data. Figure 3 shows the final watermark insertion algorithm.

Watermark insertion algorithm (Watermark W , Relation R , Embedding key K , γ , α)

Input: Watermark W , Original Relation R , Embedding key K , fraction of tuples $1/\gamma$, candidate attributes α

Output: Watermarked Relation RW

```
1  w[0, ..., sb - 1] = Load_watermark(W); // the size of bit flow of the watermark is sb
2  Randomize(w[0, ..., sb - 1]);
3  for each tuple  $t_i \in R$ 
4  loop
5     $d = H(t_i.P \parallel K)$ ;
6    if  $d \bmod \gamma = 0$  then                                // mark this tuple
7      attribute_index  $j = d \bmod \alpha$ ;                    // mark attribute  $A_j$ 
8      mark_bit_idx =  $d \bmod sb$ ;
9      Bit_encoding( $K$ ,  $w[idx]$ ,  $t_i.A_j$ );
10 end if;
11 end loop;
12 end;
```

Figure 3. Watermark insertion algorithm.

As can be seen, a sorting operation is not required in our method, since tuples in the database relation are hashed based on their primary keys, concatenated with the embedding key, and mapped to a specific index of the watermark vector. This property ensures that incremental updatability [11] can be satisfied where the modified tuples can be individually re-watermarked. Thus, the proposed method solves the sorting problem found in [14], which means that the

proposed method is preferred for databases with frequent updates.

3.2 Watermark extraction

Here are two situations where the watermark extraction can be invoked: 1) If the owner of the relational database suspects that some data sets are illegally copied from his/her relational database R , the owner or the third party can use the watermark extraction algorithm to verify the ownership of the suspicious database. 2) The database user wants to buy the full version of a trial database application. In order to extract the watermark from the database relation R , we need to know the embedding key K , γ , and α .

Similarly, Bit_decoding algorithm is used in watermark extraction algorithm to extract one encoded bit. The fraction portion will be processed, by subtracting the fraction portion from $H(t_i.P \parallel K)$ to calculate the expanded difference, and the watermark bit i is the LSB of the expanded difference. Next, by using Equation (13) we can obtain the original prediction error (difference). After that, by using Equation (14), we can recover the original value of y . Continuing with the above mentioned working example, the watermarked value of $t_i.A_j = 253.198$. By using $\text{Get_frac}(253.198)$ we have $y' = 198$. Then, by subtracting $H(t_i.P \parallel K)$ from y' we have $p'_e = 198 - 5 = 193$. After that, the embedded bit $i = \text{LSB}(p'_e) = \text{LSB}(11000001_2) = 1$. By using Equations (13) and (14), we obtain the original prediction error $p_e = \lfloor 193/2 \rfloor = 96$, and original $y = 198 - 96 - 1 = 101$, respectively. Thus, we have recovered the watermark bit $i = 1$ and the original value of $A = 253.101$ successfully. Figure 4 shows the Bit_decoding algorithm used in watermark extraction algorithm.

```
=====
Bit_decoding( $K, b, t_i.A_j$ )
Input: Embedding key  $K$ , numeric value for attribute  $j$  of tuple  $i$ 
```

Output: decoded watermark bit b , updated value for attribute j of tuple i

```

1  newfrac = Get_frac( $t_i.A_j$ );           // returns the fraction portion of  $t_i.A_j$ 
2  int = Get_int( $t_i.A_j$ );               // returns the integer portion of  $t_i.A_j$ 
3  if (newfrac >= H( $t_i.P \parallel K$ )) then
4    newdiff = newfrac - H( $t_i.P \parallel K$ );
5     $b$  = LSB(bin(newdiff));
6    diff = Floor(newdiff / 2);
7    newval = newfrac - diff -  $b$ ;
8    newval = To_number(int, newval);    // convert int and newval to number
9    Reflect_update( $t_i.A_j$ , newval);    // update field  $j$  of tuple  $i$  by newval
10 end if;
11 end;

```

Figure 4. Bit_decoding algorithm.

Because of the identical distribution of the hash function when seeded by the same embedding key, the selected tuples are of the same order as in the watermark insertion algorithm. By using the Bit_decoding algorithm, we can reconstruct the embedded watermark and recover the original database relation completely in the absence of attacks. After finishing the extraction process, we will have several watermarks each belonging to a certain point. A majority voting mechanism is applied to find the final watermark information. For each marked bit, we count the numbers of its zeroes and ones respectively, and then the higher will be the final value of this bit. The detected result is a binary sequence which includes the copyright information of the relation R . Then, we transform the binary sequence again into the watermark image. The copyright holder can make use of the detected watermark image to prove the copyright ownership. The detailed algorithm used for watermark extraction is reported in Figure 5.

Watermark extraction algorithm (RW, K, γ, α)

Input: Watermarked relation RW , Embedding key K , fraction of tuples $1/\gamma$, candidate attributes α

Output: Watermark image $W[]$, recovered relation R

```

1  for  $i = 0$  to  $sb - 1$  do

```

```

2   w[i] = '';                                // initialize detected mixed watermark
3   count[i][0] = 0; count[i][1] = 0;         // initialize 1's and 0's counter
4   end loop;
5   for each tuple  $t_i \in RW$  loop
6      $d = H(t_i.P \parallel K)$ ;
7     if  $d \bmod \gamma = 0$  then                // this tuple was marked
8       attribute_index  $j = d \bmod \alpha$ ;    // marked attribute  $A_j$ 
9       mark_bit  $idx = d \bmod sb$ ;
10       $b = \text{Bit\_decoding}(K, w[idx], t_i.A_j)$ ;
11      count[idx][b] = count[idx][b] + 1;      // increment the 0 or 1 counter
12    end if;
13  end loop;
14  for  $i = 0$  to  $sb - 1$  do                    // majority voting mechanism
15    if (count[i][0] >= count[i][1]) then  $W[i] = 0$ ; // extracted watermark bits
16    else  $W[i] = 1$ ;
17  end if;
18  end for;
19  Re-randomize( $W[0], \dots, sb - 1$ );        // this method will re-shuffle the voted  $W[]$ 
20  Transform_watermark( $W[]$ );                // this method will transform the voted  $W[]$  to bitmap image of size  $m*n$ 
21  Save_watermark();                          // this method saves the transformed watermark
22  end;

```

=====

Figure 5. Watermark extraction algorithm.

4. Experiments and analysis

In this section, we report the intensive experiments of this study. The purpose of our experiments was to test the proposed method in terms of robustness against several database attacks such as deletion, insertion, and modification attacks in addition to checking its imperceptibility and the time overhead. The experimental setup included a 2.4 GHz CPU and 1GB RAM PC running Windows Vista Professional. Algorithms were implemented in Microsoft .NET environment using ADO component to visit Microsoft Office Access 2003. We applied our algorithms to a generated synthetic data set of nine attributes, one is the primary key and the others are numerical attributes. The size of the generated set was 40,000 tuples. As we mentioned in Section 3, the size of the watermark information is $32*32$ pixels, so the watermark size sb is

1024 bit. We investigated several embedding scales ($\gamma = 6, 12$, and 18) to embed the watermark bits into the data set.

4.1 Imperceptibility test

We report on the effect of the distortion introduced to the data after the watermark insertion process, and we used the mean and variance as statistical metrics to measure the quantitative change introduced to the data. In this experiment, the value of γ varied from 6 to 18. Table 2 shows the result of this experiment after rounding the values to the nearest integer. Blank entries in the table indicate very little or no change. We can see that the distortion is minuscule and does not damage the data usability.

Table 2. Change in mean (ΔM) and variance (ΔV) introduced by watermarking ($\gamma = 6, 12$, and 18).

Attribute	Mean Value	Variance	$\gamma = 6$		$\gamma = 12$		$\gamma = 18$	
			ΔM	ΔV	ΔM	ΔV	ΔM	ΔV
A1	134.97	28786.46	-1	-2		-1		-1
A2	299.64	141881.20	-1	+3	-1	+2		+1
A3	2.54	3.24×10^{-2}						
A4	54.30	478.04		-1				
A5	156.92	31006.56	-1	+2		+1		+1
A6	35.49	204.23						
A7	4.54	3.03×10^{-2}						
A8	38.92	1201.80		+1		+1		

4.2 Robustness test

We use S as a similarity indicator to measure the similarity of the detected watermark to the original one. S is calculated as shown in Equation (16).

$$S = 100 - \frac{\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} pic1(i, j) \oplus pic2(i, j)}{m \times n} \times 100\% , \quad (16)$$






where $pic1(i, j)$ represents the original watermark image and $pic2(i, j)$ represents the extracted

watermark image. We tested the validity and robustness of our proposed method against deletion, insertion, and modification attacks. Next we report our findings against these attacks.

4.2.1 Deletion attack

In this attack, violators randomly drop α tuples from the watermarked relation RW . To simulate this attack, we randomly deleted different ratios of the watermarked data. Table 3 shows that the embedded watermark can be 100% detected after deleting 70% of the watermarked relation. Even when 85% of the watermarked data is deleted, the watermark image can be fairly recovered. We can see that even a small portion of the watermarked relation is enough for a successful detection. This important property combined with the high efficiency of our watermark extraction algorithm makes it possible to develop a tool which is able to efficiently search the web to detect illegal copies of data. Such a technique only needs to inspect about 30% of the watermarked data for successful watermark detection.

Table 3. Results using PE algorithm under deletion attack ($\gamma = 6$).

Tuples deleted %	10 -70%	75%	80%	85%	90%
Watermark detected					
Similarity (S)	100%	98.33%	95.01%	86.91%	76.46%

We implemented the Difference Expansion (DE) method adopted in [24]. In table 4, we report on the simulated deletion attack based on DE algorithm in the same testing setup. As we can see, the results of PE and DE under deletion attack are quite similar, since the delete operation affects all the attributes $t_i.A_j$ in a tuple t_i in RW with equal probability.

Table 4. Results using DE algorithm under deletion attack ($\gamma = 6$).






Tuples deleted %	10-70%	75%	80%	85%	90%
Watermark detected					
Similarity (S)	96.67%	93.19%	88.18%	82.71%	73.33%

Figure 6 depicts the resilience under the deletion attack for several values of the embedding scale ($\gamma = 6, 12, \text{ and } 18$) for the PE and DE algorithms. We can see that both PE and DE shows higher resilience when γ decreases. Which means more bits are embedded.

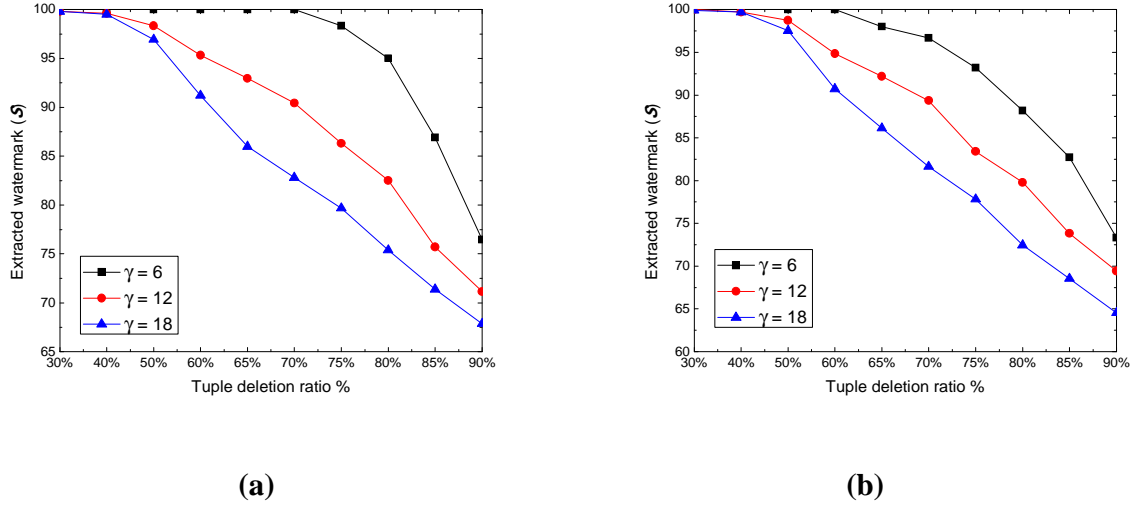






Figure 6. Resilience under deletion attack for (a) PE and (b) DE algorithms.

4.2.2 Insertion attack





In this attack, violators try to insert α tuples from other sources to the current watermarked relation RW , hoping to weaken the embedded watermark; however those inserted tuples have little effect in our method. Even when 90% new tuples are added, the watermark can be successfully detected. The results show that our method is resilient against this attack, owing to the fact that the newly added tuples have equal probability to affect a watermark bit being one or zero due to the majority voting mechanism. For a given watermark bit of value one, it needs to convert 50% or more of the embedded bits to zero in order to be converted from one to zero, and vice versa. So this implies a need to at least duplicate the current data set size for the attack to succeed, which, in turn, leads to loss of data usability. Table 5 shows the result of this attack.

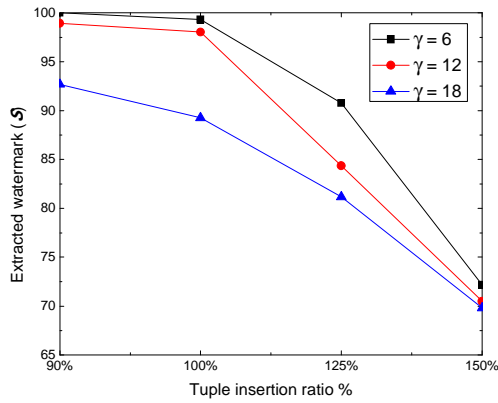
Table 5. Results using PE algorithm under insertion attack ($\gamma = 6$).

Tuples Inserted %	10-90%	100%	125%	150%
Watermark detected				
Similarity (S)	100%	99.31%	87.79%	72.16%

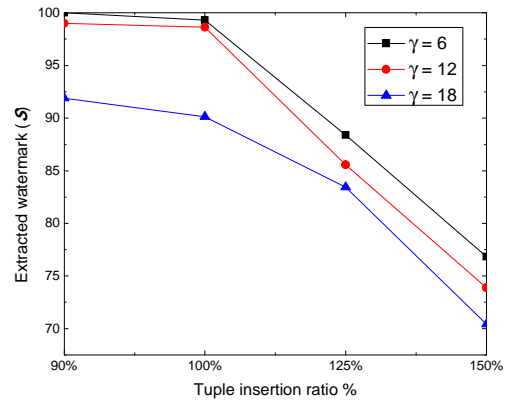
We simulated the insertion attack using the DE [24] also. Table 6, reports the result for this attack. We can see that in both PE and DE the results are quite similar for the same reason explained in the deletion attack. Figure 7 depicts the resilience under insertion attack using several values of the embedding scale ($\gamma = 6, 12$, and 18) for the PE and DE algorithms. As in deletion attack, we can see that both PE and DE show higher resilience when γ decreases.

Table 6. Results using DE algorithm after insertion attack ($\gamma = 6$).

Tuples Inserted %	10-90%	100%	125%	150%
Watermark detected				
Similarity (S)	100%	99.31%	87.40%	75.83%



(a)











(b)

Figure 7. Resilience under Insertion attack for (a) PE and (b) DE algorithms.

4.2.3 Modification attack

In this attack, violators try to randomly select and modify random attributes $t_i.A_j$ in α tuples in the watermarked relation RW . Those modified tuples may perturb the watermark extraction process, but in our method, the result shows high resilience against this attack. In this experiment, we modified two attributes randomly for the whole data set. Table 7 shows the result of this attack. It can be seen that the results in this attack are very high. This can be explained due to the majority voting mechanism. Given that for a watermark bit to be flipped, more than 50% of the embedded bits must be flipped. Thus, the proposed method is robust against this attack.

Table 7. Results using PE algorithm under modification attack ($\gamma = 6$).

Tuples Modified %	10-30%	40%	50%	60%	70%	80%	90%	100%
Watermark detected								
Similarity (S)	100%	99.90%	99.51%	98.73%	97.94%	96.19%	94.72%	92.57%

Similarly, we simulated the modification attack using the DE [24]. Table 8, reports the result for this attack. We can see that the proposed PE method outperforms the DE method. As mentioned in section 2, the main drawback in DE that is one embedded watermark bit is embedded in two attributes $t_i.A_j$. Thus, any attempt to modify any of the contributed attributes will lead to watermark bit loss and failing to recover the original data.

Table 8. Results using DE algorithm under modification attack ($\gamma = 6$).









Tuples Modified %	10-30%	40%	50%	60%	70%	80%	90%	100%
Watermark detected								
Similarity (S)	100%	98.14%	95.11%	89.45%	83.88%	78.71%	72.94%	67.87%

Figure 8 depicts resilience under modification attack using several values of the embedding scale ($\gamma = 6, 12$, and 18) for the PE and DE algorithms. We can see that the PE algorithm shows higher resilience against this attack, owing to the fact that an embedded bit using DE has a higher probability to be flipped than is the case in the PE algorithm under this attack. The results

showed in this figure demonstrate that the proposed PE algorithm is superior to the DE algorithm employed in [24].

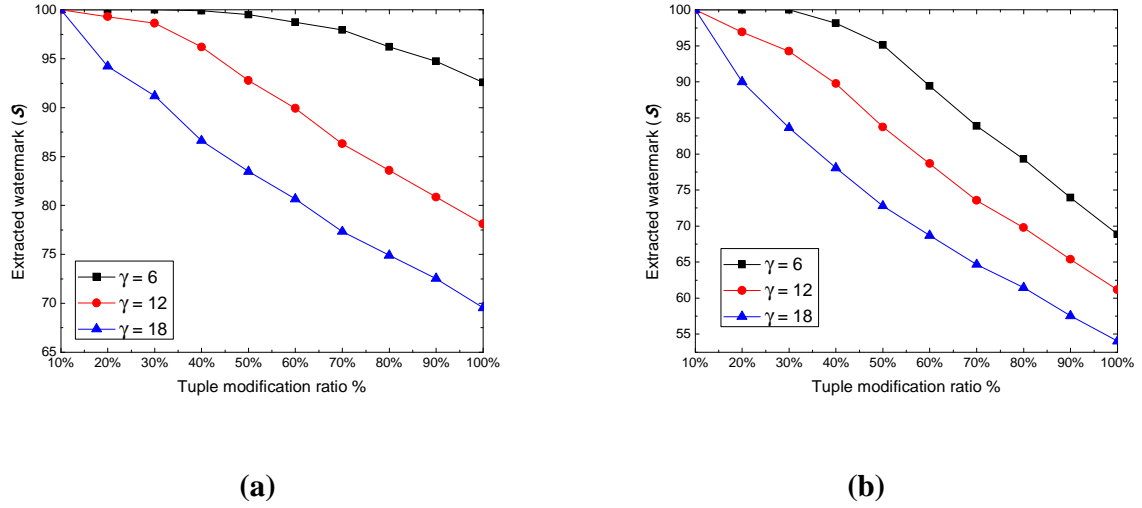


Figure 8. Resilience under modification attack for (a) PE and (b) DE algorithms.

4.3 Overhead cost test

We report the computational cost results for watermark insertion and extraction algorithms for the embedding scales ($\gamma = 6, 12$, and 18) in Table 9.

Table 9. Time overhead cost.

	$\gamma = 6$	$\gamma = 12$	$\gamma = 18$
Watermark insertion time (s)	22	18	13
Watermark extraction time (s)	13	11	8

As we can see, in the worst case when $\gamma = 6$, where one tuple out of six is selected to be watermarked, it took around 22 seconds on average for the tests to be carried out. This time is quite small compared to the major benefits available from watermarking. On the other hand, the watermarking process is usually done offline, which will not affect the overall performance of the database operations.

5. Conclusions and future work

In this paper, we have proposed a novel blind reversible relational database watermarking method. This method can prove the true ownership of the database's owner and allows full recovery of the original database relation once the watermark information is extracted and authenticated. The watermarks are embedded into a database relation under the control of a secure embedding key. A majority voting technique is applied to correct the watermark bits extracted from the data at the watermark extraction phase.

The reversibility of our method makes it feasible to apply it to watermark sensitive database applications like medical and military systems. Based on its imperceptibility, robustness against attacks, and the overhead cost tests, the experimental results show that the proposed method is feasible, blind, and robust. A full-fledged commercial watermarking application could be derived from our method. Our future research is directed towards increasing the level of attack resilience in a reversible and blind watermarking method.

References

- [11] R., Agrawal and J., Kiernan, 2002. Watermarking relational databases. In: *Proceedings of the 28th very large data bases VLDB conference*, Aug. 20-23, 2002, Hong Kong, China, 155-166.
- [9]. M., Arnold, 2000. Audio watermarking: features, applications and algorithms. In: *Proceedings of the 5th IEEE international conference on computer and multimedia and expo*, Jul. 30, 2000, New York, NY, USA, 1013-1016.
- [4] J.T., Brassil, S., Low, N.F., Maxemchuk, 2002. Copyright protection for the electronic distribution of text documents. In: *Proceedings of the IEEE*, Aug. 06, 2002, 1181-1196.
- [8] I.J., Cox, *et al.*, 1997. Secure spread spectrum watermarking for multimedia. *IEEE transaction image processing*, 6(12), 1673-1687.

- [6] I.J., Cox, M.L., Miller, and J.A., Bloom, 2001. *Digital watermarking*. Morgan Kaufmann. Menlo Park.
- [10] I.J., Cox, G., Doerr, and T., Furon, 2006. Watermarking is not cryptography. In: *Proceedings of 5th international workshop, IWDW 2006 Jeju Island, Korea, Nov. 8-10, 2006*, 1-15.
- [24] G., Gupta, and J., Pieprzyk. 2008. Reversible and blind database watermarking using difference expansion. In: *Proceedings of eForensics*, Jan. 2008, 1-6.
- [7]. G., Langelaar, and I., Setyawan, 2000. Watermarking digital image and video data. *IEEE signal processing magazine*, 17(5), 20-43.
- [13] Y., Li, V., Swarup, and S., Jajodia, 2003. A robust watermarking scheme for relational data. In: *Proceedings of the 13th workshop on information technology and systems WITS*, Dec. 2003, 195-200.
- [16] Y., Li, H., Guo, and S., Jajodia, 2004. Tamper detection and localization for categorical data using fragile watermarks. In: *Proceedings of the 4th ACM workshop on digital rights management DRM '04*, 2004, 73-82.
- [1] W.H., Lin, *et al.*, 2008. An efficient watermarking method based on significant difference of Wavelet coefficient quantization. *IEEE transactions on multimedia*, 10(5), 746-757.
- [3] W.H., Lin, *et al.*, 2009. Image copyright protection with forward error correction. *Expert systems with applications*, 36(9), 11888-11894.
- [5] F.A.P., Petitcolas, 2000. Watermarking schemes evaluation. *IEEE signal processing magazine*, 17(5), 58-64.
- [18] B., Schneier, *Applied Cryptography*, John Wiley. New York: 1996.

- [15] M., Shehab, E., Bertino, and A., Ghafoor, 2008. Watermarking relational databases using optimization based techniques. *IEEE transactions on knowledge and data engineering*, 20(1), 116-129.
- [12] R., Sion, 2004. Proving ownership over categorical data. In: *Proceedings of 20th IEEE international conference on data engineering ICDE*, Apr. 2004, 584-596.
- [14] R., Sion, M., Atallah, and S., Prabhakar, 2004. Rights protection for relational data. *IEEE transactions on knowledge and data engineering*, 16(12), 1509-1525.
- [19] D.M., Thodi, and J.J., Rodriguez, 2004. Prediction-error-based reversible watermarking. In: *Proceedings of IEEE conference on image processing*, Singapore, Oct. 2004, 1549-1552.
- [23] J., Tian, 2003. Reversible data embedding using a difference expansion. *IEEE transactions on circuits and systems for video technology*, 13(8), 890-896.
- [20] S., Weng, *et al.*, 2008. Reversible watermarking based on invariability and adjustment on pixel pairs. *IEEE Signal Processing Letters*, 15, 721-724,
- [2] R.B., Wolfgang, C.I., Podilchuk, and E.J., Delp, 1999. Perceptual watermarks for digital images and video. In: *Proceedings of SPIE- The International Society for Optical Engineering*, San Jose, CA. Jan. 1999, 40-51.
- [22] C., Yang, W., Hu and C., Lin, 2010. Reversible data hiding by coefficient-bias algorithm. *Journal of information hiding and multimedia signal processing*, 1(2), 91-100.
- [21] C., Yang, C., Lin, and W., Hu, 2011. Reversible data hiding by adaptive IWT-coefficient adjustment. *Journal of information hiding and multimedia signal processing*, 2(1), 24-32.
- [17] Z., Zhang, *et al.*, 1996. Watermarking relational database using image. In: *Proceedings of the third international conference on machine learning and cybernetics*, Shanghai, Aug. 26-29, 2004, 1739-1744.