A reversible database watermarking method non-redundancy shifting-based histogram gaps

Yan Li¹,², Junwei Wang¹ and Xiangyang Luo¹

Abstract

In relational databases, embedding watermarks in integer data using traditional histogram shifting method has the problem of large data distortion. To solve this problem, a reversible database watermarking method without redundant shifting distortion is proposed, taking advantage of a large number of gaps in the integer histogram. This method embeds the watermark bit by bit on the basis of grouping. First, an integer data histogram is constructed with the absolute value of the prediction error of the data as a variable. Second, the positional relationship between each column and the gap in the histogram is analyzed to find out all the columns adjacent to the gap. Third, the highest column is selected as the embedded point. Finally, a watermark bit is embedded on the group by the histogram non-redundant shifting method. Experimental results show that compared with existing reversible database watermarking methods, such as genetic algorithm and histogram shift watermarking and histogram gap–based watermarking, the proposed method has no data distortion caused by the shifting redundant histogram columns after embedding watermarks on forest cover type data set and effectively reduces the data distortion rate after embedding watermarks.

Keywords

Reversible database watermark, data distortion, histogram gap, low distortion, non-redundancy shifting

Date received: 31 August 2019; accepted: 30 March 2020

Handling Editor: Hongxin Hu

Introduction

In the era of big data, how to protect data security in the database is an important issue. Therefore, researchers introduced digital watermarking technology¹–⁴ into the database as an effective means and then the database watermarking technology⁵,⁶ came into being. The technology protects the rights of the database owner by modifying the data in the database and embedding the watermark containing the copyright information. In 2002, IBM Almaden research center introduced digital watermarking technology in the relational database for the first time.⁷ Thereafter, a variety of database watermarking technologies successively appeared, such as watermark technology based on special marker tuples,⁸ fingerprint technology based on block idea,⁹ and optimization technology based on verification.¹⁰

In order to further protect the authenticity and integrity of data, reversible database watermarking technology emerges as the times require.¹¹,¹² This technique can not only hide the watermark information in the original carrier data but also recover the original carrier data without loss after extracting the watermark. In 2006, a reversible database watermarking method histogram shifting watermarking (HSW) based on histogram
shifting was first proposed by Zhang et al.\textsuperscript{13} Since then, researchers have successively proposed different types of reversible watermarking technologies, such as the difference expansion-based watermarking (DEW),\textsuperscript{14} the differential expansion watermark technology combined with genetic algorithm (GADEW),\textsuperscript{15} the differential expansion watermark technology combined with firefly algorithm (FFADEW),\textsuperscript{16} and the predictive error extended watermarking (PEEW).\textsuperscript{17} Although these techniques have solved the problems of copyright protection and data recovery in different degrees, they have some shortcomings in data distortion rate (DDR) and watermark robustness.

In 2018, DH Hu et al.\textsuperscript{18} proposed a reversible database watermarking method called genetic algorithm and histogram shift watermarking (GAHSW) based on numerical relational data distortion control. This method uses a genetic algorithm to group tuples and then uses the HSW method to embed watermarks. This method improves the robustness of reversible database watermarking while reducing data distortion. Li et al.\textsuperscript{19} proposed a low-distortion reversible database watermarking method called histogram gap-based watermarking (HGW). Compared with GAHSW, this method reduces data distortion without weakening the robustness of watermark.

However, through the analysis, we found that the HGW is unnecessary to modify part of the data when embedding watermarks, which is caused by the shifting of redundant columns. Therefore, we propose a reversible database watermarking method without redundant shifting distortion, which is abbreviated as NSHGW (non-shifting histogram gaps based on watermarking). The traditional histogram shifting method is improved using the characteristics of a large number of gaps in the database histogram, the positional relationship between each column and gap in the histogram is analyzed, and all adjacent columns are found; then choose the highest one of these columns as the embedding point to embed watermark information, so as to reduce the amount of modification to the carrier database data, reduce the redundant shifting of histogram columns, so as to reduce data distortion.

The structure of the article is as follows: Section “Related work” describes the related work in the early stage of the paper; section “The proposed method” elaborates the basic ideas and main contents of the proposed method; experimental verification and result analysis were carried out in section “Results and analysis”; and section “Conclusion” gives conclusions.

**Related work**

In this section, histogram shifting method and HGW reversible database watermarking method will be introduced in detail.

**Histogram shifting method**

The histogram shifting method (HS, histogram shifting) is mainly used in the early stage to process the reversible watermarking of digital images, which hides the watermarking information with the vertex value of the histogram. In Zhang et al.’s study,\textsuperscript{13} the histogram shifting method was introduced into the database. This method obtains non-zero prediction errors from two adjacent local raw values in the database, uses these error values as the horizontal axis, and uses the frequency of each value of prediction error as the vertical axis to construct a histogram. The prediction errors are calculated by equation (1.1), where $p_e$ is the prediction error, and $x_i, x_{i+1}$ are two adjacent attribute values

$$p_e = x_{i+1} - x_i, \quad x_i < x_{i+1} \quad (1.1)$$

A non-zero frequency peak bit is found in the histogram, and the corresponding prediction error is marked as $p$. Then shift all the bits except $p$ by one cell to create a vacancy near $p$. Finally, each prediction error is scanned one by one, and when $p$ will be encountered, a watermark bit $w(w = 0$ or $w = 1$) is embedded. After embedding the watermark, the prediction error changes and the new prediction error value $p'_e$ is calculated by equation (1.2)

$$p'_e = \begin{cases} p_e + 1, & p_e > p; \\ p_e + w, & p_e = p; \\ p_e, & \text{otherwise} \end{cases} \quad (1.2)$$

After $p'_e$ is calculated, the inverse integer Haar wavelet transform is used to calculate the new attribute value. The calculation formulas are equations (1.3) and (1.4), respectively, where $x'_i$ and $x'_{i+1}$ are the watermark values of attributes $x_i$ and $x_{i+1}$ after embedding, and $x_m$ is the mean value of $x_i$ and $x_{i+1}$, which are calculated by equation (1.5)

$$x'_m = x_m - \left\lfloor \frac{p'_e}{2} \right\rfloor \quad (1.3)$$

$$x'_{i+1} = x_m + \left\lfloor \frac{p'_e + 1}{2} \right\rfloor \quad (1.4)$$

$$x_m = \frac{x_i + x_{i+1}}{2} \quad (1.5)$$

For example, $x_i = 102$ and $x_{i+1} = 108$ are adjacent properties, prediction error $p_e = 108 - 102 = 6$, $x_m = \lfloor(102 + 108)/2\rfloor = 105$. If the peak bit $p = 4$, embedded watermark bit $w = 1$, then the new prediction error value $p'_e = 4 + 1 = 5$, the values of $x_m$ and $p'_e$ are respectively substituted into equations (1.3) and (1.4) to obtain:

$$x'_m = 105 - \lfloor 4/2 \rfloor = 103 \quad x'_{i+1} = 105 + \lfloor (4 + 1)/2 \rfloor = 107.$$
**HGW reversible database watermarking method**

Li et al.\(^1\) propose a low-distortion reversible database watermarking method which is HGW. This method improves the traditional histogram shifting method. It uses histogram gaps to reduce the number of columns shifted when embedding a watermark to reduce data distortion.

HGW first divides the database tuples into \(N_g\) groups according to the watermark length \(N_w\), and then inserts a watermark bit into each group by histogram translation method. The basic steps of HGW embedding watermark are as follows:

Step 1. Determine the relevant parameters: for the first \(i\) grouping, found in the histogram peak location \(p_i\), were about to search, to find the first frequency to zero position, respectively for \(p_{iL}, p_{iR}\) and \(p_{iL}\) to \(p_{i}\) all column height of Hutchison for \(h_{siL}\) and between \(p_{i}\) and \(p_{iR}\) all column height of Hutchison for \(h_{siR}\), then respectively between \(p_{i}\) and \(p_{iL}, p_{iR}\), the number of columns for \(d_{iL}, d_{iR}\). The position relationship between \(p_{i}, p_{iL},\) and \(p_{iR}\) is shown in Figure 1.

Step 2. Determining the shifting direction: this method judges the sizes of \(h_{siL}\) and \(h_{siR}\) and distinguishes between different cases for shifting and watermark embedding according to the judgment results:

\[
\text{if } h_{siL} \geq h_{siR} \\
\quad p^{'i} = \begin{cases} 
    p_{i} + 1, & p_{i} < p_{h} < p_{i} + d_{iR} \\
    p_{i} + w, & p_{h} = p_{i} \\
    p_{h}, & \text{otherwise}
\end{cases} \quad (1.6)
\]

\[
\text{if } h_{siL} < h_{siR} \\
\quad p^{'i} = \begin{cases} 
    p_{h} - 1, & p_{h} - d_{iL} < p_{i} < p_{h} \\
    p_{h} - w, & p_{h} = p_{i} \\
    p_{h}, & \text{otherwise}
\end{cases} \quad (1.7)
\]

According to the reference, when \(p_{c} \geq 0, p_{h} = p_{c}\); when \(p_{c} < 0, p_{h} = -p_{c}\); during watermark embedding, we should consider not only the conditional comparison of the sum of heights but also the distinction between \(p_{c} \geq 0\) and \(p_{c} < 0\). Finally, histogram translation and watermark embedding will be processed according to the differential formula.

**The proposed method**

The proposed NSHGW mainly consists of three sections: preprocessing, watermark embedding, and watermark extraction. This method is applicable to the integer numerical data of relational databases. The algorithm principle is shown in Figure 2. Among them, \(D_{O}\) is the original database, \(D_{WR}\) is the database with embedded watermarks, \(D_{AW}\) is the watermarked database after suffered attacks, and \(D_{R}\) is the recovered database.

**Preprocessing**

In the preprocessing stage, the step should be completed before watermark embedding and watermark extraction. The specific pretreatment process is as follows:

Step 1.1 Determining candidate attribute columns: selecting multiple integer columns from the database that can identify the features of things as Candidate attribute columns for embedding watermarks. The data in the candidate attribute column usually indicate the characteristics of things (such as hair color). When watermarks are embedded in these data, the data distortion caused by the data has less impact on the data quality.

Step 1.2. Sorting candidate attribute columns: This method sorts the column names of the candidate attribute columns in ascending order, in order to enhance the robustness of the algorithm when it is attacked by the attribute columns.

Step 1.3. Determining the range of semantic distortion and calculating tolerance: the value range of the candidate attribute column is usually the semantic distortion range of the attribute column. In the watermark embedding process, in order to ensure that the modification of the data does not exceed the range of semantic distortion defined by each attribute column, the method uses the minimum and maximum values in the data of each column as the lower and upper bounds of the value range, respectively. This method defines the range of semantic distortion of the \(j\)th attribute column in \([\text{min}(j), \text{max}(j)]\). In the process of embedding the watermark, the value of the modified data cannot exceed the range of semantic distortion of the
The tolerance of the \( j \)th column can be calculated using equation (2.1)

\[
y_j = \frac{\max[j] + \min[j]}{2}
\]  

(2.1)

In equation (2.1), \( \max[j] \) and \( \min[j] \) individually represent the maximum and minimum of the \( j \)th column.

Step 1.4 Grouping tuples: in the method, \( K_s \), a secret grouping key, is set by a random method, and the tuples in the relational database are divided into several non-overlapping groups \( \{G_i\} i = 1, 2, \ldots, N_g \) according to equation (2.2)

\[
n_u = H(K_s|H(K_s|t_u.PK))) \mod N_g
\]  

(2.2)

In equation (2.2), \( n_u \) is a serial number of some group, “\( | \)” represents the concatenation operation, \( H() \) represents a hash function, \( K_s \) represents the secret grouping key, and \( t_u \cdot PK \) represents the primary key of the tuple as parameters. \( N_g \) is determined by the number of bits of the watermark to be embedded. For example, if the watermark to be embedded contains 48 watermark bits, then \( N_g = 48 \).

**Watermark embedding**

The method requires embedding a watermark 1 bit in each grouping. Therefore, the process of embedding a watermark is an iterative process of performing steps 2.1–2.4 for each grouping. For the convenience of expression, group \( i \) is used here to represent any grouping. The process of embedding watermark is shown in Figure 3.
Step 2.1. Construct a histogram of group i. First, the method uses equations (2.3) and (2.4) to calculate all \( p_c \) and \( p_h \) in the group. Then, it uses the value of \( p_h \) as the horizontal axis and the frequency of each value of \( p_h \) as the vertical axis to construct a histogram

\[
\begin{align*}
    p_c &= y - \bar{y} \quad (2.3) \\
    p_h &= |p_c| \quad (2.4)
\end{align*}
\]

In equation (2.3), \( y \) represents the value located in \( j \)th column and in certain tuple, \( \bar{y} \) is the tolerance of the \( j \)th column. \( p_c \) represents the prediction error value corresponding to \( y \). In equation (2.4), \( p_h \) is the absolute value of the prediction error of the original database.

Step 2.2. Selecting eligible columns in group \( i \): it searches from the very left side of the histogram to the right in turn to find each column \( HC_j \) which meets the following conditions.

If \( HC_j > 0 \), then \( HC_{j-1} = 0 \) or \( HC_{j+1} = 0 \). \( HC_j \) is the height of the \( j \) column in the histogram.

Step 2.3. Determining the position of the watermark embedding: the method selects the highest column from all eligible \( HC_j \), records it as \( HC_i \), and uses \( HC_i \) as the position of watermark embedding. Record the \( p_h \) value of the position where the column \( HC_i \) is located as \( p_i \), and use \( GP_i \) to mark the relationship between the blank position and \( HC_i \). If the blank position is on the left of \( HC_i \), make \( GP_i = -1 \); if the blank position is on the right side of \( HC_i \), make \( GP_i = 1 \).

Step 2.4. Embedding watermark bit.

It performs the watermark embedding operation based on the value of \( GP_i \) which indicates the relative relationship between \( HC_i \) and the adjacent blank position. It can embed watermarks according to the following two formulas, respectively.

if \( GP_i = -1 \)

\[
\begin{align*}
    p'_h &= \begin{cases} 
        p_h - w, & p_c = p_i \\
        p_h, & \text{otherwise}
    \end{cases} \quad (2.5)
\end{align*}
\]

if \( GP_i = 1 \)

\[
\begin{align*}
    p'_h &= \begin{cases} 
        p_h + w, & p_c = p_i \\
        p_h, & \text{otherwise}
    \end{cases} \quad (2.6)
\end{align*}
\]

It can be concluded from equation (2.4) that when \( p_c \geq 0 \), \( p_h = p_c \); when \( p_c < 0 \), \( p_h = -p_c \). Therefore, equations (2.5) and (2.6) can be further distinguished into two cases, respectively. By substituting \( p_c \geq 0 \) and \( p_c < 0 \) into formulas (3.5) and (3.6) respectively, the following four formulas can be obtained.

if \( GP_i = -1 \) and \( p_c \geq 0 \)

\[
\begin{align*}
    p'_e &= \begin{cases} 
        p_e - w, & p_e = p_i \\
        p_e, & \text{otherwise}
    \end{cases} \quad (2.7)
\end{align*}
\]

if \( GP_i = -1 \) and \( p_c < 0 \)

\[
\begin{align*}
    p'_e &= \begin{cases} 
        p_e + w, & p_e = -p_i \\
        p_e, & \text{otherwise}
    \end{cases} \quad (2.8)
\end{align*}
\]

if \( GP_i = 1 \) and \( p_c \geq 0 \)

\[
\begin{align*}
    p'_e &= \begin{cases} 
        p_e - w, & p_e = p_i \\
        p_e, & \text{otherwise}
    \end{cases} \quad (2.9)
\end{align*}
\]

if \( GP_i = 1 \) and \( p_c < 0 \)

\[
\begin{align*}
    p'_e &= \begin{cases} 
        p_e + w, & p_e = -p_i \\
        p_e, & \text{otherwise}
    \end{cases} \quad (2.10)
\end{align*}
\]

In summary, the histogram shifting and watermark embedding performed by this method can be finally processed according to the formulas (3.7)–formula (3.10). In addition, in order to extract the watermark and recover data from the database, an array needs to be used here to store the \( p_i \) and \( GP_i \) obtained in each group. After the watermark embedding process is completed, the \( pa \) will be combined with the \( Ks \) generated in step 1.4 as parameters for the process of watermark extraction and data recovery.

**Watermark extracting**

Detailed algorithm for watermark extraction and data recovery is reported below. The following steps can be used to extract the watermarks and restore the database after the database is attacked by insertion, deletion, and alteration. Before the extraction watermarking, it is necessary to preprocess the database according to the pretreatment Steps 1.1–1.4, and then perform the procedure of Step 3.1–3.3.

Step 3.1. Constructing histogram: equation (2.11) was used to calculate all \( p'_c \) and \( p'_h \) in group \( i \). Then, the values of \( p'_h \) were used as the horizontal axis, and the frequency of occurrence of each \( p'_h \) value was used as the vertical axis to construct the histogram. Scan each \( p'_h \) one by one to determine the position of \( j \) in the histogram

\[
\begin{align*}
    p'_c &= y' - \bar{y} \quad (2.11) \\
    p'_h &= |p'_c| \quad (2.12)
\end{align*}
\]
Among them, $y'$ is any attribute value in the $D_{yr}$, and $\hat{y}$ is the tolerance of the attribute column of $y', p_e$ which can be calculated by equation (2.11) is the prediction error corresponding to $y'$.

Step 3.2. Extracting watermark: extracting watermark with the $p_i$ and $GP_i$ stored in array $pa$ corresponding to group $i$. Perform the following detection operation on all tuples in group $i$, and record the number of all watermark bits 0 and 1 detected in this group, then the use majority voting mechanism to determine the final watermark information of this group, and regard the watermark bits with a large number as the final detected watermark bits.

The specific test operation method is as follows: when $|p'_e| = p_i$, the property value has not been altered, the watermark bit is 0. When $GP_i = -1$, if $p'_e > 0$ and $p'_e = p_i - 1$, the watermark bit detected is 1, if $p'_e < 0$ and $p'_e = p_i + 1$, the watermark bit detected is 1; when $GP_i = 1$, if $p'_e > 0$ and $p'_e = p_i + 1$, the watermark bit detected is 1, if $p'_e < 0$ and $p'_e = p_i - 1$, the watermark bit detected is 1.

After repeated experiments, the watermark extracted by the majority voting mechanism can be used to prove the copyright owner. If the extracted watermark is the same as the current one, it means that the database is real and that the extraction process does not modify it.

Step 3.3. Recovering data: Figure 1 shows watermark extraction and data recovery. According to the positive and negative values of $p'_e$ and $GP_i$, and the size relationship of sum, the following data recovery is carried out:

\[
\text{if } GP_i = -1 \text{ and } p'_e \geq 0\\
\quad y' = \begin{cases} 
\ y' + 1, & p'_e = p_i - 1 \\
\ y', & \text{otherwise}
\end{cases}
\] (2.13)

\[
\text{if } GP_i = -1 \text{ and } p'_e < 0\\
\quad y' = \begin{cases} 
\ y' - 1, & p'_e = -p_i + 1 \\
\ y', & \text{otherwise}
\end{cases}
\] (2.14)

\[
\text{if } GP_i = 1 \text{ and } p'_e \geq 0\\
\quad y' = \begin{cases} 
\ y' - 1, & p'_e = p_i + 1 \\
\ y', & \text{otherwise}
\end{cases}
\] (2.15)

\[
\text{if } GP_i = 1 \text{ and } p'_e < 0\\
\quad y' = \begin{cases} 
\ y' + 1, & p'_e = -p_i - 1 \\
\ y', & \text{otherwise}
\end{cases}
\] (2.16)

Among them, $y'$ is the restored attribute value. Here is a summary of formula symbols (Table 1).

**Experimental results and analysis**

In this article, in order to evaluate the performance of the watermark and validation method, we conduct experiments in an Intel Core i5 environment with a 2.40 GHz CPU and 8GB RAM. The selected database was the forest cover type dataset provided by the University of California as the database for testing the watermark. This data set contains 581,012 tuples and 54 attributes. According to the experimental requirements, we selected 10 integer numeric attributes for the experiment and compared them with the existing HGW reversible watermarking method.

The experiment was set up in three parts. The first part is DDR analysis, which compares this method with similar methods reported by Hu et al. and Li et al. The second part is the quantitative analysis of histogram distortion, which shows the influence of this parameter on the performance of the algorithm. The third part is the robustness analysis, which shows the performance of the method under some well-known attacks in the database to test the effectiveness of the watermark. The method used is set as the HGW experiment by Li et al., which generates a composite column as the primary key, and the length (number of packets) of the watermark is 48. For the convenience of comparison, all the following experiments are embedded with watermarks under the same conditions.

**DDR analysis**

The loss and harm caused by data distortion are huge, so data distortion must be strictly prevented. In this experimental study, in order to better understand the effect of DDR on data quality, we not only evaluated and calculated the watermark length of 48-bit but also selected the watermark length of 24-bit and 72-bit to compare the distortion rate of NSHGW and HGW watermarking methods.

We use the DDR to assess NSHGW and HGW embedded watermark after the distortion of the effects on the database. The larger the DDR, the greater the distortion of the data will be and vice versa. The calculation formula of DDR is as follows

\[
\text{DDR} = \frac{T_{dis}}{TD}
\] (3.1)

In the formula above $T_{dis}$ is the total data amount of distortion, and $TD$ is the total amount of data in the database. Experiments show that the distortion rate of NSHGW is much smaller than that of HGW.

In the experiment, the distortion rates of NSHGW and HGW were verified when the total tuples were 1000, 1200, 1400, 1600, 1800, and 2000, respectively. We use different lengths of watermarks (the number of packets) and get different results. Figure 4 shows the
DDRs of the two methods when the watermark length is 24, 48, and 72, respectively.

The horizontal axis represents the number of different tuples in the database, while the vertical axis is DDR, representing the rate of distortion. When DDR is 0, the data in the database is not distorted. Because both NSHGW and HGW rely on stochastic optimization, the distortion rate is the average of the 10 runs of the two methods, respectively.

As can be seen from Figure 4, while embedding the same watermark information into the same database in the experiment, the DDR generated by NSHGW method is much lower than that of HGW, both less than 0.42%.

**Quantitative analysis of histogram distortion**

In the experiment, in order to identify redundancy shifting brought about by the distortion effect more clearly, we not only for 48 to evaluate watermark length calculation, but also with the above DDR analysis experiments, selected the additional 24 and 72—bit watermark value, the length of the two watermarking methods and NSHGW HGW distortion of histograms multi-faceted contrast. It is important to note that it is necessary to evaluate the distortion effect of NSHGW- and HGW-embedded watermarks on the database by the amount of distorted data caused by shifting (excluding the distortion caused by embedding watermark bits).

In the experiment, we compare the two methods of NSHGW and HGW each time, and verifying the total amount of distorted data caused by the shifting of the two methods of NSHGW and HGW when the total number of tuples is 1000, 1200, 1400, 1600, 1800, and 2000, respectively, under the condition of the same embedded watermark. We use different lengths of watermarks (the number of packets) and get different results. Figure 5 shows the total amount of distorted data caused by the shifting of the two methods when the watermark length is 24, 48, and 72, respectively.

In the figure, the horizontal axis represents the number of different tuples in the database, while the vertical axis represents the total amount of distorted data and represents the number of distorted data. When the total amount of shifting distortion data is 0, it means that the data of the database is not distorted. Since both NSHGW and HGW depend on random optimization, the total amount of distortion data is the average of the results of 10 runs of the two methods. As can be seen from the above figure, when the same watermark information is embedded in the same database, the total amount of redundant translation distortion data generated by NSHGW method is close to 0 and far lower than that of HGW, with a minimum of 12 and a maximum of 29, almost overlapping with the horizontal axis.

**Robustness analysis**

In this experiment, we mainly study the robustness of NSHGW watermarking method under malicious attack of well-known database. The watermarked data in the experiment will be attacked by different types, such as insertion attack, altered attack, and deleted attack. In order to facilitate the comparison with other methods, NSHGW watermarking method is consistent with the experiments of HGW, GAHSW, and other watermarking methods by Gupta and Pieprzyk,14 Jawad and Khan,15 Farfoura et al.,17 Hu et al.,18 and Li et al.,19 and the length of the same watermark is set to 48 for this experimental test. The value of bit error rate (BER) is used to evaluate the robustness of the watermarking method. From the following equation, we can easily see that the lower the BER value is, the higher the watermark robustness will be. BER is the ratio of the number of error-extracted bits to the number of

| Symbol | Description |
|--------|-------------|
| D_W   | Watermarked database |
| D_WM  | Watermarked database after suffered attacks |
| P_e   | Prediction error of original database |
| P_r   | Prediction error of watermarked database |
| HC_i  | Screen out the highest column |
| HC_j  | All columns that meet the conditions |
| y[j]  | Tolerance of the j-th column |
| y[e]  | Attribute value |
| K_s   | Secret grouping key |
| t_e   | Primary key of the tuple |
| W_det | Detected watermark bits |
| P_h   | Absolute value of new prediction error of watermarked database |
| P_i   | Prediction error value of HC_i corresponding position |
| D_R   | Recovered database |
| D_O   | Original database |

---

It is determined by the number of a bits of the watermark to be embedded.
embedded watermark bits. The calculation equation of BER is as follows

\[
\text{BER} = \frac{\sum_{i=1}^{N} w_i \oplus w_i^{\text{det}}}{N_r}
\]  

(3.2)

where \(w_i\) is the embedded watermark bit and \(w_i^{\text{det}}\) the detected watermark bit. Through the analysis of different types of attack data in the experiment, it is verified that the robustness of NSHGW is equal to that of HGW and GAHSW, and obviously higher than that of other watermarking methods. Next, we will carry out the attack experiment in the best case and the worst case, respectively, demonstrate the watermark detection and data recovery results, and compare the current popular watermark methods.

We present the results as a graph, with the horizontal axis representing the percentage (%) of attacks based on the size of the database. This means we need to change the specific rate of tuples which exist in the data set. Zero (0)% attack means no detection of any attack on the data set. We simulate the attacker trying to insert, delete, and modify 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, and 90% of the data in the data set. If the data set contains 1000 tuples, changing 10% means 100 tuples will be attacked. In the figure, the vertical axis is BER, indicating the rate of unsuccessful detection of watermark. When BER approaches 0, the watermark is correctly detected from the database. Because NSHGW relies on random optimization, the BER extracted from the watermark is the average of the 10 runs of the method. The results of the deletion and alteration are drawn as shown in Figures 6 and 7, respectively.

Insertion attack is an attack by inserting a new tuple to affect the watermark detection process. In the database, the attacker adds new tuples to increase the number of false positives and affect the process of detecting data. But the experimental results show that the BER value is 0 before insertion attack. After the insertion attack, BER remains 0 even if 60% of the new tuples are added to the watermarked data set. It is verified that insertion attack will not cause the problem of watermark robustness because our technology only involves tuples with watermark, and tuples without watermark or newly added tuples will not affect the process of watermark detection. Therefore, the three watermarking methods of NSHGW, HGW, and GAHSW are robust for insertion attack. No matter how many tuples

---

**Figure 4.** The comparison of data distortion rates caused by watermark of different bits: (a) 24-bit watermark, (b) 48-bit watermark, and (c) 72-bit watermark.
are inserted, the BER under the insertion attack is always 0, so the comparison diagram of insertion attack will not be drawn here.

Deletion attack is a process in which an attacker destroys the watermark by deleting tuples randomly. Attackers randomly remove some tuples from the watermarked data set in the hope that some of the watermarked tuples will also be deleted, resulting in the loss of watermark bits. Deleted attack is a process in which an attacker destroys the watermark by deleting tuples randomly. Attackers randomly remove some tuples from the watermarked data set in the hope that some of the watermarked tuples will also be deleted; resulting in the loss of watermark bits. Deleted attack is more damaging than other attack because there is a chance that watermark bits will not be detected.

**Figure 5.** The comparison of the total amount of the data distortion caused by shift with watermark of different bits: (a) 24-bit watermark, (b) 48-bit watermark, and (c) 72-bit watermark.

**Figure 6.** Deleted attack BER comparison.

**Figure 7.** Alteration attack BER comparison.
correctly when removing a tuple with a watermark. As the percentage of delete attack increases, the rate of correct detection plummets and is more damaging than other attacks because there is a chance that watermark bits will not be detected correctly when removing a tuple with a watermark. As the percentage of delete attack increases, the rate of correct detection plummets.

Figure 6 shows the BER of the watermark extracted by NSHGW, HGW, GAHSW, PEEW, DEW, and GADEW methods after removing the attack. GAHSW, DEW, GADEW, and PEEW extracted watermarks with BER values of 0.477, 0.814, 0.9550, and 0.915, respectively, when the number of deleted tuples in the database increased by 90% when the database was severely attacked by deletion. However, the BER extracted from NSHGW watermark is 0.388 and that from HGW watermark is 0.472. NSHGW can also recover at least half of the watermark. If most of the watermarks contained in the terra-tuple are removed from the database, the watermark cannot be recovered. The harm of deleting attack to database watermark is obvious.

Alteration attack is a process in which an attacker randomly altered the tuples of a database to destroy the watermark. Attackers randomly or permanently change some tuple values from a watermarked data set in the hope that some watermarked attributes will be changed altogether, never increasing the difficulty of detecting watermark bits. As the percentage of alteration attack increases, the rate of correct detection plummets.

Figure 7 shows the BER of the watermark extracted by NSHGW, HGW, GAHSW, PEEW, DEW, and GADEW methods after alteration attack. The BER of extracted watermarks will increase with the increase of data altered. amount, and the number of altered attributes in the database reaches 90%. GAHSW, DEW, GADEW and PEEW extracted watermarks with BER values of 0.411, 0.434, 0.584 and 0.526, respectively. The BER of NSHGW watermark extraction is 0.356, and that of HGW watermark extraction is 0.383. We can see that NSHGW can recover at least half of the watermark at this point. However, if a large number of watermark attributes are altered, it is difficult to extract the watermark tuples affected by this attack from the remaining unaffected data.

In general, in the deletion attack and altered attack, Figures 6 and 7 show that the robustness of NSHGW, HGW, and GAHSW methods is roughly the same. With the increase of attack intensity, the BER of extracting watermark by these three methods also increases. But compared with other methods, NSHGW has better watermark detection rate under this attack. Experiments also show that the NSHGW method produces a lower BER when embedding the same watermark into the same database.

Conclusions

We propose a new method of reversible database watermarking with low distortion and the originality of the method lies in the grate reduction of the distortion rate of the carrier data. NSHGW method is to use the characteristics of database histogram gap to improve the histogram shifting method, find the best embedding point in the gap, and realize watermark embedding. While embedding watermark, the quantity of redundant shifting of histogram column is avoided, the data quality is guaranteed (the amount of changed data is greatly reduced), and the data distortion caused by watermark embedding is effectively reduced. Experiments show that NSHGW method is better than the existing HGW method in distortion rate analysis and histogram shifting quantitative analysis. The robustness analysis of NSHGW method is similar to that of HGW and GAHSW. Based on the current histogram embedding method and on the basis of reducing data distortion, the future work is to find better and smaller data distortion methods, which is also my research direction.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) received no financial support for the research, authorship, and/or publication of this article.

ORCID iD

Yan Li https://orcid.org/0000-0003-1208-1226

References

1. Unnikrishnan K and Pramod KV. Robust optimal position detection scheme for relational database watermarking through HOLPSOFA algorithm. Inf Secur Tech Rep 2017; 35: 1–12.
2. Laskar RH, Choudhury M, Chakraborty K, et al. A joint DWT-DCT based robust digital watermarking algorithm for ownership verification of digital images. In: 5th international conference on information processing, Bangalore, India, 5–7 August 2011, pp.482–491. Berlin: Springer.
3. Ahmed F. Integrated fingerprint verification method using a composite signature-based watermarking technique. Opt Eng 2007; 46(8): 087005.
4. Gong DF, Chen Y, Lu HY, et al. Self-embedding image watermarking based on combined decision using pre-offset and post-offset blocks. *Comput Mater Con* 2018; 57(2): 243–260.

5. Zhang ZH, Jin XM, Wang JM, et al. Watermarking relational database using image. In: *International conference on machine learning and cybernetics*, Shanghai, China, 26–29 August 2004, pp.1739–1744. New York: IEEE.

6. Xiao XG, Sun XM and Chen MG. Second-LSB-dependent robust watermarking for relational database. In: *Third international symposium on information assurance and security*, Manchester, 29–31 August 2007, pp.292–300. New York: IEEE.

7. Agrawal R and Kiernan J. Watermarking relational databases. In: *VLDB'02: proceedings of the 28th international conference on very large databases*, Hong Kong SAR, China, 20–23 August 2002, pp.155–166. Amsterdam: Elsevier.

8. Rajneeshkaur B, Anita T and Wadhai VM. A new watermarking approach for non-numeric relational database. *Int J Comput Appl Technol* 2010; 13(7): 37–40.

9. Liu SY, Wang SH, Deng RH, et al. A block oriented finger printing scheme in relational database. In: *International conference on information security and cryptology*, Seoul, Koreal, 2–3 December 2004, pp.455–466. Berlin: Springer.

10. Chandankhede C. A robust technique for relational database watermarking and verification. In: *ICCCICT international conference on communication*, Mumbai, India, 15–17 January 2015, pp.1–7. New York: IEEE.

11. Chang CC, Nguyen TS and Lin CC. A blind reversible robust watermarking scheme for relational databases. *Sci World J* 2013; 2013(3): 717165.

12. Hao W, Xiang LY, Li Y, et al. Reversible natural language watermarking using synonym substitution and arithmetic coding. *Comput Mater Con* 2018; 55(3): 541–559.

13. Zhang Y, Yang B and Niu XM. Reversible watermarking for relational database authentication. *J Comput* 2006; 17(2): 59–65.

14. Gupta G and Pieprzyk J. Reversible and blind database watermarking using difference expansion. *Int J Digital Crime Forensics* 2009; 1(2): 42–54.

15. Jawad K and Khan A. Genetic algorithm and difference expansion based reversible watermarking for relational databases. *J Syst Software* 2013; 86(11): 2742–2753.

16. Imamoglu MB, Ulutas M and Ulutas G. A new reversible database watermarking approach with firefly optimization algorithm. *Math Prob Eng* 2017; 2017(Pt. 3): 1387375.

17. Farfouar ME, Horng SJ and Wang X. A novel blind reversible method for watermarking relational databases. *J Chin Inst Eng* 2013; 36(1): 87–97.

18. Hu DH, Zhao D and Zheng SL. A new robust approach for reversible database watermarking with distortion control. *IEEE Trans Knowl Data Eng* 2018; 31(6): 1024–1037.

19. Li Y, Wang JW, Ge SK, et al. A reversible database watermarking method with low distortion. *Math Biosci Eng* 2019; 16(5): 4053–5068.