A New Robust Approach for Reversible Database Watermarking With Distortion Control

Donghui Hu, Dan Zhao, Shuli Zheng

Abstract—Nowadays information is crucial in many fields such as medicine, science and business, where databases are used effectively for information sharing. However, the databases face the risk of being pirated, stolen or misused, which may result in a lot of security threats concerning ownership rights, data tampering and privacy protection. Watermarking is utilized to enforce ownership rights on shared relational databases. Many reversible watermarking methods are proposed recently to protect rights of owners along with recovering original data. Most state-of-the-art methods modify the original data to a large extent, result in data quality degradation, and cannot achieve good balance between robustness against malicious attacks and data recovery. In this paper, we propose a robust and reversible database watermarking technique, Genetic Algorithm and Histogram Shifting Watermarking (GAHSW), for numerical relational data. The genetic algorithm is used to select the best secret key for grouping database, where the watermarking can be embedded with balanced distortion and capacity. The histogram of the prediction error is shifted to embed the watermark with good robustness. Experimental results demonstrate the effectiveness of GAHSW and show that it outperforms state-of-the-art approaches in terms of robustness against malicious attacks and preservation of data quality.

Index Terms—Reversible watermarking, genetic algorithm, database, data recovery, robustness, histogram shifting.

1 Introduction

Currently, relational databases are used and shared extensively due to the increasing use of the Internet and cloud computing [1]. However, the remarkable increase in the creation, transfer and sharing of databases simultaneously incurs security risks, such as data theft, illegal copying and copyright violation. Therefore information security problems have become increasingly prominent. Indeed, outsourced data can be redistributed or modified without permission from owners. Accidents of database leakage have been reported frequently in recent years, even in domains such as healthcare where data are sensitive [2].

Historically, watermarking techniques have been utilized to ensure ownership protection and tamper proofing for various data formats (e.g., image, video and audio). Database watermarking is a relatively new technique and embeds the message into database records that are often independent and discrete. Database watermarking was firstly introduced in 2002 by Agrawal and Kiernan [3]. Since then, several methods have been proposed [4–7].

In [4], a fragile and robust persistent watermarking method that embeds both private and public watermarks is proposed. Based on whether the watermarking introduces any changes to the data of the database, the watermark technology can be categorized into two types: distortion-based watermarking and distortion-free watermarking [8]. Next we mainly introduce distortion-based and robust watermarking methods. Most of the distortion-based watermarking schemes were proposed to handle distortion constraints. For instance, in [5] the embedding process does not modify numerical attributes if the availability conditions for certain data are not satisfied, [7, 9, 10] focus on preserving the database statistics, and [11] takes into account the full database semantics that should also be preserved. However, these methods modify the data to a large extent, which may result in the loss of data quality. To overcome the problems of these distortion-based methods, reversible database watermarking has been introduced to recover the original data fully after extracting embedded watermarks from the watermarked databases.

The first reversible watermarking scheme for relational databases was proposed in 2006 [12], where histogram expansion is used for reversible database watermarking. However, this method is not robust against heavy attacks. In 2008, the technique called Difference Expansion based Watermarking (DEW) [13] was utilized to watermark a database in a reversible way. Bhattacharya and Cortesi proposed a method in 2009 [14] to embed the watermark after partitioning tuples as a permutation. However, when taking the distortion constraint into account, the embedding rate of DEW is greatly reduced. In [15], the genetic algorithm (GA) was used to design a robust steganographic method. Then the genetic algorithm based on difference expansion watermarking (GADEW) technique was proposed as a robust and reversible databases watermarking solution [16]. In [17], a new reversible database watermarking method is proposed by combining DEW with firefly algorithm (FFADEW). Best attribute values are selected by the FFA to yield lower distortion and increase watermark capacity. Although FFADEW minimizes data distortions and increases watermark capacity, information distortion still remain serious.

Another reversible watermarking technique proposed in [18], Prediction-Error Expansion Watermarking (PEEW), is fragile against malicious attacks because the watermarking information is embedded only in the fractional part of numeric features. For integer values, this method violates the imperceptibility of the watermarking. When the watermark information is embedded directly into the integer

The authors are with the College of Computer Science and Information Engineering, Hefei university of technology, Hefei 230009, China (e-mail: hudh@hfut.edu.cn; 2296377616@qq.com; zsl251@163.com).

values, the data distortion will be very large. Therefore, this method degrades data quality and lacks robustness.

In a more recent work [19], a Robust and semi-blind Reversible Watermarking (RRW) technique for numerical relational data was proposed. In this method, the value of Mutual Information (MI) for each feature is calculated and the features with low MI are selected for watermarking. An optimum watermark string is created by employing GA. Then the watermark information is embedded in the selected features. Two parameters, the optimized value from the GA (β) and a change matrix, are used in the watermark encoding and decoding phases. RRW can preserve data quality and is effectively against malicious attacks. However, this method has the following drawbacks: 1) the auxiliary data that needs to be passed to users are very large; 2) in the data preprocessing stage of RRW, it takes a lot of time to calculate the value of MI for each feature and watermark information; 3) the watermark information is uncertain and needs to be obtained by the genetic algorithm, so that different databases have different watermark information; and 4) when dealing with integer data, the robustness of RRW is poor because β, which is used to watermark a feature, is a decimal.

In summary, the state-of-the-art reversible relational databases watermarking methods, including DEW, GADEW, FFADEW, PEEW and RRW, have their own weakness as described above. Besides, these methods were proposed to handle floating point values, while dealing with integer data, these methods will greatly degrade data quality and thus are not applicable in real-world applications. Therefore, we aim to develop an appropriate watermark method that can ensure watermark robustness without significant loss of data quality for databases containing integer and floating point data.

Based on the granularity level of the watermark embedding, [20] classify watermarking techniques into four types: ATSASB (all tuples, single attribute and single bit), MTSASB (multiple tuples, single attribute and single bit), MTSAMB (multiple tuples, single attribute and multiple bits) and MTMAMB (multiple tuples, multiple attributes and multiple bits). From the security analysis in the paper, we can see that the security of ATSASB is the best in the four cases. Therefore our method is designed to be ATSASB.

In the process of reversible watermarking, it is challenging to achieve robustness (attack resilience) and small distortion of data after embedding watermarks. Distortion constraints can ensure the imperceptibility of watermarking and avoid a large loss of data quality. Significant loss of data quality makes it an easy target for attack, thus, the robustness of watermarking will be poor. So the robustness and the distortion constraints may be potentially conflicting. To tackle this challenge, we propose a robust and reversible database watermarking technique, Genetic Algorithm and Histogram Shifting Watermarking (GAHSW), to both maximize robustness and minimize distortion. Although GADEW and FFADEW also use the optimization method to select the best positions to minimize distortion for watermark embedding, the distortion caused by the embedded watermarks is still large. Our proposed GAHSW method solves this problem by combining GA with a new proposed method called Histogram Shifting of prediction error expansion Watermarking (HSW).

GAHSW mainly includes three phases, watermark preprocessing phase, watermark embedding phase, and watermark extraction and data recovery phase. In the data preprocessing phase, attribute columns are sorted according to the attribute name, the range of semantic distortion constraint for each attribute column is determined, and the approximately optimal secret key information is obtained through the GA. The secret key is used for watermark embedding and extraction. The watermark embedding phase embeds watermark information into database using the HSW. HSW embeds the watermark mainly through shifting the left and right sides of the histogram, and increases the watermark embedding rate without violating the defined distortion constraints. Thus, this method can preserve the data quality of the watermarked database very well. Three parameters, the optimized secret key from the GA, peak values of the histogram for each group, and the primary keys corresponding to the tuples where the selected attributes are prohibited be modified in the embedding phase, are used in the watermark embedding and extraction phases. Finally, the watermarked data are generated. The attack types comprise subset alteration, subset deletion and subset insertion attacks generated by the adversary. These malicious attacks modify the original data and try to destroy watermarks. In the watermark extraction and data recovery phase, the embedded watermark is extracted from the suspicious data. Some works of the preprocessing step (such as sorting attribute columns according to the attribute name and determining the range of semantic distortion constraint for each attribute column) are performed again, and extraction strategies of HSW are used to recover the watermark.

The subsequent sections of the paper are organized as follows. Section 2 gives background information about database watermarking. The details of the proposed scheme are described in Section 3, which is followed by experimental results and analysis in Section 4. Section 5 concludes the article and presents future research.

2 Preliminaries

We present the preliminaries of the proposed work in this section.

2.1 Histogram Shifting

The histogram shifting (HS) method was first proposed for digital image reversible watermarking in [21] and then attracted researchers' attention [22–32]. HS uses the peak value of the histogram to hide watermark information. In terms of HS used in digital image reversible watermarking, the gray value corresponding to the peak value of the histogram is used to expand, and the gray values ranging from the peak to zero are used to shift. HS was first used for database watermarking in 2006 [12]. In this scheme, only a part of each original value (instead of the whole one) is extracted. The capacity in this scheme is gained from the partial error between two neighbouring original partial values. Assume an arbitrary pair of neighbouring original

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2851517, IEEE Transactions on Knowledge and Data Engineering

partial values x_i, $x_{i+1}(x_i < x_{i+1})$, the predictive error is calculated as

$$p_e = x_{i+1} - x_i. \quad (1)$$

Then errors with nonzero initial digits are used to form a histogram. Then a peak bin with nonzero frequency in the histogram is determined and denoted as p. This method then shifts the histogram bins by one unit to create a vacant position near p. Finally, each predictive error is scanned to embed 1-bit information w (w = 1 or 0) when p is encountered. Let p_e and p'_e represent an original predictive error and the corresponding new one, respectively. HS is performed by

$$p'_e = \begin{cases} p_e + 1, & p_e > p; \\ p_e + w, & p_e = p; \\ p_e, & otherwise. \end{cases} \quad (2)$$

The method uses the inverse integer Haar wavelet transform to derive watermarked database values x'_i and x'_{i+1} from the watermark information carrier p'_e. Define the median value of x'_i and x'_{i+1} as

$$x_m = \lfloor (x'_i + x'_{i+1})/2 \rfloor . \quad (3)$$

The watermarked attribute values can be obtained by the inverse integer Haar wavelet:

$$x'_i = x_m - \lfloor (p'_e)/2 \rfloor . \quad (4)$$

$$x'_{i+1} = x_m + \lfloor (p'_e + 1)/2 \rfloor . \quad (5)$$

For example, $x_i = 104$, $x_{i+1} = 106$ are two adjacent attributes. Then the predictive error p_e and the average x_m can be computed as follows: $p_e = 106 - 104 = 2$, and $x_m = \lfloor (104 + 106)/2 \rfloor = 105$. Here, $\lfloor x \rfloor$ denotes the greatest integer less than x (floor function). If the peak value p = 2 and the embedded watermark bit w = 1, the new predictive error p'_e can be obtained as: p'_e= 2 +1= 3. Then, the new attribute values are given by

$$x'_i = x_m - \lfloor (p'_e)/2 \rfloor = 105 - 1 = 104. \quad (6)$$

$$x'_{i+1} = x_m + \lfloor (p'_e + 1)/2 \rfloor = 105 + 2 = 107. \quad (7)$$

This method uses histogram shift and the inverse integer Haar wavelet transform to embed watermark. We can see that this histogram expansion based method leads to relatively small data distortion. However, the robustness of the method is poor. This is because if any one of the two adjacent attribute values embedded the watermark is attacked by the adversary, the watermark cannot be extracted correctly. Therefore, we propose a new histogram shifting (HS) based reversible database watermarking algorithm shown in Section 3.2.1.

2.2 Genetic algorithm

Th genetic algorithm (GA) is an optimization technique inspired by the biological evolution process [33]. It is often used to solve optimization and search problems. GA uses a collection of data structures called chromosomes, which provides a possible solution for the problem. It is initialized with a population of random chromosomes. The population size of GA is determined by the number of chromosomes. Consecutive populations are known as generations. GA evolves a potential solution to optimization problem in search space. It uses the operators, such as selection, crossover and mutation, to generate new generation to preserve essential information. Consequently, the chromosomes gain multiple reproductive opportunities. In addition, the fitness value of each candidate chromosome is evaluated by the fitness function. GA follows an iterative mechanism to evolve a population of chromosomes in the search of optimal solution. Operations continue for a number of generations, until the required criteria of fitness are achieved or the number of generations is completed. Finally, the chromosome with the best fitness provides approximate optimal solution. In general, GA consists of two major parts that are dependent on the problem: the problem encoding and the fitness function.

In GADEW [16], GA based optimization is also used to improve robustness of reversible watermarking for relational databases. However, the purpose of GA used in GADEW is to select the best embedding positions in order to minimize distortion for watermark embedding. Besides, the distortion caused by the embedded watermark through this method is still large. In FFADEW [17], FFA based optimization has the same usage and purpose as GADEW. In this paper, GA is used in the data preprocessing phase to find the optimal grouping in order to maximize watermark robustness without significant information loss. Besides, our method combines GA with a new proposed Histogram Shifting of prediction error method. As a result, our method can solve the problem of large distortion caused by the embedded watermark very well.

3 The Proposed GAHSW Method

This section presents GAHSW for reversible watermarking of relational databases. GAHSW improves the data quality and the watermark robustness. The main architecture of GAHSW is presented in Figure 1. It includes the following three major phases: (1) watermark preprocessing; (2) watermark embedding; (3) watermark extraction and data recovery.

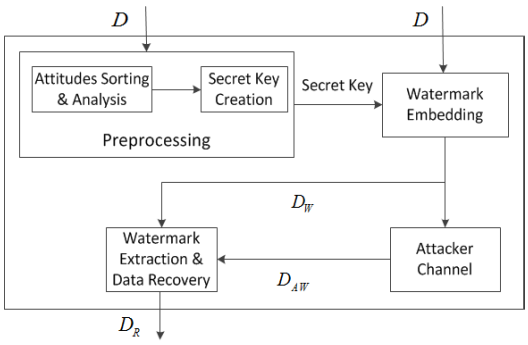


Fig. 1. Main architecture of GAHSW, where D is a database, D_W is the watermarked database, D_{AW} is the watermarked database after suffered attacks, and D_R is the recovered database.

The watermark preprocessing phase also consists of three parts, sorting the attribute column according to the attribute name, determining the semantic distortion range

of the attribute column, and obtaining the approximately optimal secret key information through GA. The secret key is used for watermark embedding and extraction.

The main focus of watermark embedding phase is to embed watermark information in such a way that ensures the usability and the imperceptibility of the watermarked database. We first use GA to select the best secret key for grouping and then embed watermark information (in the form of binary) into database using the new proposed HSW technique.

After the watermarking, the database is released on a communication channel that is assumed to be insecure. These data may suffer several malicious attacks in the attacker channel. The effectiveness of GAHSW is presented through robustness analysis. We analyze the bit error rate of extracted watermarks for well-known attacks, such as subset insertion, deletion and alteration. The watermark extraction and data recovery phase mainly comprises effective extraction of the embedded watermark and successful recovery of the original data. For a quick reference, Table 1 lists the notations used in this paper.

In subsequent sections, these three phases of GAHSW will be elaborated.

3.1 Watermark Preprocessing Phase

Data preprocessing is needed before embedding watermark information. The purpose of data preprocessing is to make the watermark insertion/reading independent of the way the database is stored. One solution is to perform the tuple grouping operation prior to watermark insertion. The tuple grouping operation creates a set of non-overlapping groups $\{G_i\}_{i=1,\ldots,N_g}$. As shown in [7], the group number $n_u \in [0, N_g - 1]$ of the tuple t_u is given in Eq. (8),

$$n_u = H(Ks|H(Ks|t_u.PK)) \; mod \; N_g \tag{8}$$

where '|' represents the concatenation operator and N_g is the number of groups to build. $H()$ is a cryptographic hash function, e.g., Secure Hash Algorithm (SHA), which is applied to the primary key of the tuple ($t_u.PK$) and a secret watermarking key (Ks) to ensure secure grouping.

Then, one bit or symbol of the watermarks is embedded into each group and every tuple in the same group is embedded with the same bit to improve the robustness of the watermarking. Thus, one may expect to embed watermarks corresponding to a sequence of N_g symbols. Watermark extraction works in a similar manner. Tuples are first divided into N_g groups according to Eq. (8) and one watermark bit is then extracted from each group.

In the preprocessing phase, three important tasks should be accomplished: (1) sorting the attribute columns; (2) determining the range of attribute values; (3) calculating an approximately optimal secret key string with the help of GA. The procedure is shown in Figure 2 (a).

Sorting the attribute columns is the first step in preparing the database for watermarking process. The columns of the database are sorted in ascending order according to the attribute names. This operation ensures the algorithm's robustness against attacks of reshuffling of attribute columns. The second step is to query the range of attribute values in each attribute column of the database to determine the distortion tolerance so as to ensure the robustness of the watermarking. The last step of the preprocessing stage is to calculate an optimal secret key using the GA algorithm to obtain the approximately optimal grouping for the database watermark.

We use GA to find the optimal grouping to maximize watermark embedding rate and minimize distortion. By Eq. (8), we know that the grouping is related to the primary key and the secret key. Since the primary key is immutable, we can obtain the best secret key information using GA so as to gain the optimal grouping. Randomly initial population of binary strings called chromosomes is created. Gene values of each chromosome represent l-bit string of the secret key.

The fitness function, used for chromosome evaluation of GA, can have more than one objectives. It is named as the multi-objective fitness function. Multi-objective GA deals with concurrent optimization of two or more objectives, such as cost and performance [33, 34]. These objectives may conflict with each other and cannot be optimized simultaneously, so a proper tradeoff should be found. Fitness for each chromosome is evaluated by using two objectives related to capacity and distortion for each group. The two factors used in the fitness function are discussed in detail in Section 3.2.2. Finally, optimal secret key information string is returned after completing the maximum number of iterations of the GA.

3.2 Watermark Embedding Phase

The procedure of the watermarking algorithm is shown in Figure 2 (b). Note that the GA outputs the best secret key for grouping in the preprocessing phase. Each group will be embedded with 1-bit watermark using HS. It is worth pointing out that HS is independent with any particular database. Next, we will elaborate the HS scheme and introduce the fitness function of the GA.

Our proposed HS scheme is based on histogram expansion introduced by Zhang and Yang in 2006 [12]. Although the robustness of the histogram expansion method is poor and the embedding rate is low, the distortion of each data is very small (modified with the minimal perturbation, i.e., ± 1). We have illustrated the histogram shifting principle for database watermarking in Section 2.1. In the following part, we discuss how the histogram expansion algorithm is improved to embed one symbol of message into numerical attributes for each group of tuples.

3.2.1 histogram shifting (HS) of prediction error method

Histogram shifting provides a novel approach to robust watermarking scheme for databases. In this scheme, all tuples are securely divided into non-overlapping subsets. A single watermark bit is embedded into some tuples of a subset by modifying attribute values in the group. A watermark bit is embedded repeatedly into one group. Thus, in each group, HS is executed repeatedly. The details of the method are described as follows.

In HS, a predictor is used to create the feature elements for expansion embedding. The prediction errors are the feature elements into which expansion embedding will be done. Consider an attribute value y and a watermark bit w that need to be embedded. A predictor operates on the average value shown in Eq. (9), which is calculated according to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2851517, IEEE Transactions on Knowledge and Data Engineering

TABLE 1
Notations Used in the Paper

Symbol	Description	Symbol	Description
D	Original database	D_W	Watermarked database
D_R	The recovered database	D_{AW}	The watermarked database after suffered attacks
N	Total number of tuples in a database	N_g	Total number of groups
l	The length of the watermark	w	The watermark bit
A_{ij}	An attribute located in ith row and jth column of original database	p_e	Prediction error of original database
A_{ij}^w	An attribute located in ith row and jth column of watermarked database	p_e'	New prediction error of watermarked database
p	The peak value	z	Total number of columns in a database
pa	A one-dimensional array for storing peak points	mp	A one-dimensional array stores the special primary key information
y	The attribute value	y'	The watermarked attribute value
$\hat{y}$	The value determined by the maximum and minimum of the column	y^r	The restored attribute value
y_i^w	The watermarked attribute value in ith tuple	A	A feature/column/attribute of original database
W	Watermark bits	A^w	A feature/column/attribute of watermarked database
W^{det}	The detected watermark bits	w^{det}	A detected watermark bit
K_S	The secret key	$t_u.PK$	The primary key of the tuple
$max[i]$	The maximum of ith column	$min[i]$	The minimum of ith column
A_{cco}	Classification accuracy of original database	A_{ccw}	Classification accuracy of watermarked database
S_{no}	Sensitivity of original database	S_{nw}	Sensitivity of watermarked database
S_{po}	Specificity of original database	S_{pw}	Specificity of watermarked database

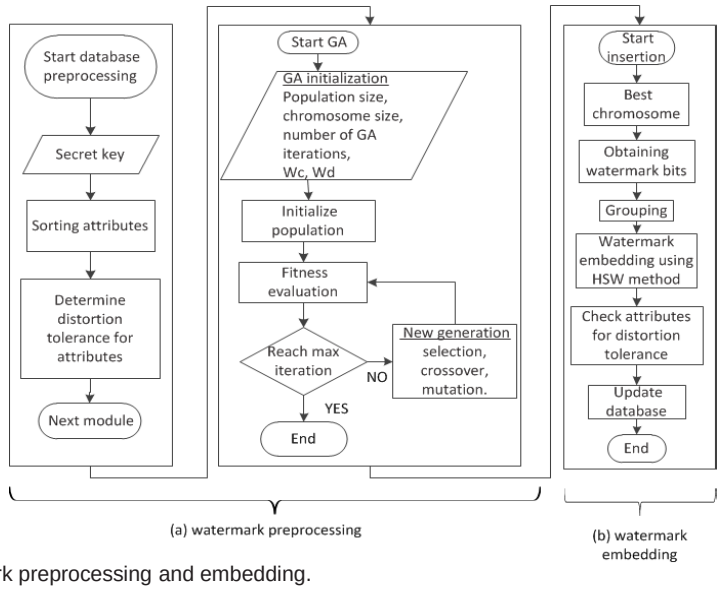


Fig. 2. The procedure of watermark preprocessing and embedding.

the maximum and minimum values in the range of attribute column where the attribute value y is located.

$$\hat{y} = \left\lfloor \frac{min[j] + max[j]}{2} \right\rfloor. \tag{9}$$

The prediction error p_e is given in Eq. (10)

$$p_e = y - \hat{y}. \tag{10}$$

The prediction error p_e is computed to construct a histogram. Before HS embedding, a peak bin with nonzero frequency is determined on the absolute value of prediction error in the histogram. We denote the peak bin as p, and use an array pa to store all of the p. pa is then shared by extractors for watermark extraction and data recovery. We shift p_e to both the right and the left by one bin to create vacant positions near p. Finally, each predictive error is scanned to embed 1-bit watermark information w when p or $-p$ is encountered. Let p_e' represents the corresponding new prediction error of p_e, which is calculated by Eq. (11).

$$p_e' = \begin{cases} p_e + 1, & p_e \geq p + 1; \\ p_e - 1, & p_e \leq -(p+1); \\ p_e, & p_e = p, w = 0; \\ p_e + 1, & p_e = p, w = 1; \\ p_e, & p_e = -p, w = 0; \\ p_e - 1, & p_e = -p, w = 1. \end{cases} \tag{11}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2851517, IEEE Transactions on Knowledge and Data Engineering

Then, the new attribute value y' is given by

$$y' = p'_e + \hat{y}. \tag{12}$$

We can see that compared with Eq. (2), Eq. (11) can increase the embedding rate of watermarks. Because in Eq. (2), p'_e can only shift to the right side of p_e, while in Eq. (11), p'_e can shift to both sides of p_e. According to Eq. (11), p'_e only changes with three possible values of -1, 0 and 1, so y' only changes with -1, 0 or 1 in Eq. (12). By this means, the proposed HSW can watermark the integer data in database with small changes.

For each group, the above procedure is repeated until all tuples of the database have been watermarked. Algorithm 1 describes the steps involved in the watermark embedding phase.

Algorithm 1 Watermark embedding

Input: Variables:D, W, z.
Output: D_W, pa, mp

1: group the database by using Eq. (8);
2: **for** $w = 1$ to l **do**
3: //loop will iterate for all watermark bits w from 1 to length l of the watermark
4: **for** $i = 1$ to N/N_g **do**
5: //loop will iterate for all tuples of each group
6: $j = \mathrm{H}(\mathrm{K_S}|\mathrm{t_u.PK})\%z$;
7: //identify marked attribute column
8: **if** $A_{ij} = max[j]$ or $A_{ij} = min[j]$ **then**
9: $j = (j+1)\%z$;
10: insert $\mathrm{t_u}$.PK of A_{ij} into mp;
11: **end if**
12: $\hat{y} = \left\lfloor \frac{min[j]+max[j]}{2} \right\rfloor$;
13: p_e is calculated by using Eq. (10);
14: **end for**
15: p is determined by the peak point of the histogram of the absolute value of p_e;
16: insert p into pa;
17: **for** $i = 1$ to N/N_g **do**
18: p'_e is calculated by using Eq. (11);
19: the corresponding attribute value is watermarked by using Eq. (12);
20: **end for**
21: **end for**
22: return D_W, pa, mp.

We illustrate our method using an example. Consider a database, for brevity, only has three features (A1, A2 and A3) with six tuples as shown in Table 2. According to Algorithm 1, first, the maximum and minimum values of each column are determined to calculate $\hat{y}$. $\hat{y}$ is then used to calculate the prediction error p_e. In step 6 to step 11 of Algorithm 1, the marked attribute column is determined. In step 6, z is a constant, determined by users, and is smaller than the number of database columns. In Table 2, the marked attribute column for each tuple is shown in blue. Finally, the peak value is determined according to the absolute value of p. In this example, $p = 1$, the watermark is embedded in the prediction error p_e, as shown in Figure 3. When the prediction error is 0, the prediction error keeps unchanged. When the prediction error is 1, the prediction error is revised to 1 or 2 to hide the watermark according to the watermark bit. Similarly, when the prediction error is -1, the prediction error is changed to -1 or -2 according to the watermark value. When the prediction error is greater than 1 or less than -1, one bin should be shifted to the right or the left, respectively. Finally, the watermarked database is shown in Table 3. When the watermark bit w is 1, attribute values take the value in the bracket. When the watermark bit w is 0, attribute values take the blue value without the bracket.

TABLE 2
Original Data for Watermarking (D)

	A1	A2	A3	
...	5	0	2	...
...	4	2	6	...
...	7	3	8	...
...	1	6	7	...
...	4	2	9	...
...	2	3	4	...

TABLE 3
Watermarked Database (D_w)

	A1	A2	A3	
...	5 (or 6)	0	2	...
...	4	2	6 (or 7)	...
...	7	3	9	...
...	1	6	8	...
...	4	2 (or 1)	9	...
...	1	3	4	...

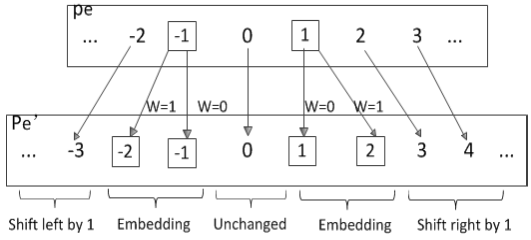


Fig. 3. A simple example of HS embedding algorithm.

Note that the attribute values for certain tuples in the group are increase of +1 (or -1) under the distortion constraints. More clearly, one value is not modified if its value is the maximum or the minimum. The primary key of this tuple should be recorded to pass to the recipient for extracting the watermark information and recovering data successfully (the recorded and passed primary key information is denoted as mp in Algorithm 1). In this situation, the embedding algorithm jumps to the next property column in the same tuple and repeats the above procedure.

3.2.2 HS using GA

In the proposed scheme, the GA is populated with a fitness function to obtain an optimal modification of data that can ensure data quality when embedding the watermark. The fitness values of chromosomes are utilized to decide which individual will be selected. The purpose of gaining an optimum string of secret key is to justify grouping that can make

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2851517, IEEE Transactions on Knowledge and Data Engineering

a tradeoff between the compromising of data quality and the embedding rate of each group. Based on this, fitness for each chromosome is evaluated using two objectives (called fitness parameters), the capacity related cost (CrC) and the distortion of tuples (ToD). For each chromosome, fitness parameters are evaluated and resulting values are added to total cost (TC). TC represents the main fitness function. At the end iteration of GA, chromosome having minimum TC is selected and passed to the watermark embedding phase for watermark insertion.

Capacity related Cost The capacity related cost is relevant to the total number of peak bins in one group that can be inserted into a watermark bit. The involved selection mechanism tends to choose the most appropriate individuals with lower fitness value as parent chromosomes. Thus the frequency of peak bins (PF) itself cannot be taken directly as the fitness value. Let NoG be the number of tuples contained in each group and N be the total number of tuples in the relational database. Each group will approximately contain N/N_g tuples, so we have $NoG = \mathrm{N}/\mathrm{N}_g$. CrC, which represents the capacity that cannot be used to embed the watermark, is defined in Eq. (13).

$$CrC = \sum_{i=1}^{N_g} (NoG - PF_i). \tag{13}$$

Distortion of tuples The second cost added to TC is the distortion of tuple, which is introduced by watermark embedding. Distortion represents the change of each attribute value caused by HS. In a group, some attributes of tuples are changed to embed 1-bit watermark. In addition, we should check for the change of attribute value, which should be satisfied with distortion constraint. ToDs are calculated for all the tuples of each chromosome. Distortion is measured by taking absolute difference of an attribute value before and after watermarking, which is finally added together as the distortion of all tuples. Eq. (14) explains process of distortion of tuples (ToD) for each chromosome.

$$ToD = \sum_{i=1}^{N} |y_i - y_i^w| \tag{14}$$

where N represents the total number of watermarked tuples; y_i and y_i^w correspond to the attribute value before and after watermarking in ith tuple, respectively. Our objective is to minimize distortion of attribute values to ensure that the watermarking may not degrade data quality obviously.

These fitness parameters are evaluated for each chromosome of the GA population. At the end of algorithm, chromosome with minimum TC is selected as the solution of our problem.

Total cost Eq. (15) explains the process of calculating the total cost (TC). CrC and ToD presented in Eq. (13) and Eq.(14), respectively, are used to calculate TC. W denotes a weight vector consisting of W_c and W_d, W_c+W_d = 1. W_c and W_d are the weights assigned to CrC and ToD, respectively. These weights can be adjusted accordingly.

$$TC = W_c \times CrC + W_d \times ToD. \tag{15}$$

For every chromosome, CrC and ToD are calculated and added together as TC. Better chromosomes have lower value for TC, while worse chromosomes have higher one. This process is executed to find the best possible chromosome for GA population. Chromosome with the minimum value of TC is ranked as the most suitable one.

These two fitness parameters can be combined in a single fitness function. GA can be executed to maximize capacity and minimize distortion.

3.3 Watermark Extraction and Data Recovery Phase

The procedure of watermark extraction and data recovery is shown in Figure 4. The detailed algorithm is reported below. In order to extract the embedded watermark and recover the original database, attribute sorting is performed in alphabetic order according to the attribute name in the database.

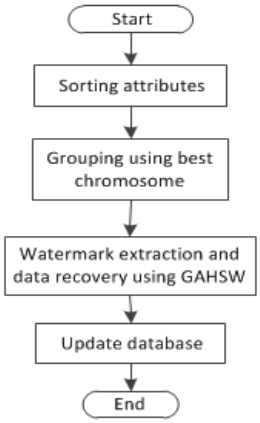


Fig. 4. Watermark detection process.

Concerning the watermark extraction stage, groups of tuples are constructed using Eq.(8). Since the distribution of the hash function is identical when seeded by the same secret key, tuples are in the same groups as in the watermark insertion algorithm. Tuples are divided into N_g groups, which need the primary key and the secret key. It is important to remark that the secret key should be known by the detector so as to make it possible to reconstruct the groups. The peaks of each group, as well as the primary keys corresponding to the tuples where the selected attributes are prohibited be modified in the embedding phase, are sent to readers as a part of the watermarking key. Watermark bits extracting and the original attribute value recovery using the HS detection technique will be elaborated below.

Consider an attribute value y' in the watermarked database. A predictor operates on $\hat{y}$, which is the same predictor as employed by the embedder. The new prediction error $\mathrm{p_e}'$ is given as Eq.(16).

$$p_e{}' = y' - \hat{y}. \tag{16}$$

The watermark bits are then extracted according to $p_e{}'$. If $p_e{}' = p$, the detected watermark bit will be 0. If $p_e{}' = p+1$ or $p_e{}' = -(p+1)$, the detected watermark bit will be 1. The value of a numeric attribute is recovered using Eq.(17).

$$y^r = \begin{cases} y' - 1, & p_e{}' \geq p+1; \\ y', & p_e{}' = p; \\ y' + 1, & p_e{}' \leq -(p+1). \end{cases} \tag{17}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2851517, IEEE Transactions on Knowledge and Data Engineering

where p is the peak bin of the histogram in each group, y^r is the corresponding restored attribute value of y. Each new predictive error is scanned to extract watermark and recover the original data when p, $p+1$ or $-(p+1)$ is encountered. In other cases, the original data value is restored according to the condition it meets shown in Eq.(17). Note that in Algorithm 2, pa is an array has stored all of the p, and $pa[s]$ is the p related to each group.

Algorithm 2 Watermark Extraction and Data Recovery

Input: Variables:D_W, z, pa, mp.
Output: $D_R, W^{\det}$

group the database by using Eq. (8);
$a = 0$;$b = 0$;
for $s = 1$ to l **do**
  for $i = 1$ to N/N_g **do**
    //loop will iterate for all tuples of each group
    $j = \mathrm{H}(\mathrm{K_S}|\mathrm{t_u}.\mathrm{PK})\%z$;
    //identify marked attribute column
    if ($A_{ij}^w = max[j]$ or $A_{ij}^w = min[j]$) and find the $\mathrm{t_u}$.PK of A_{ij} in mp **then**
      $j = (j+1)\%z$;
    end if
    $\hat{y} = \left\lfloor \frac{min[j]+max[j]}{2} \right\rfloor$;
    ${p_e}'$ is calculated by using Eq. (16);
    if ${p_e}' = pa[s]$ **then**
      the detected watermark bit ($w^{\det}$) is 0;
    else if ${p_e}' = pa[s]+1$ or ${p_e}' = -(pa[s]+1)$ **then**
      the detected watermark bit ($w^{\det}$) is 1;
    end if
    the value of numeric attribute is recovered by using Eq. (17);
    if $w^{\det} = 0$ **then**
      $a = a + 1$; //count the number of detected watermark bit taking zero
    else
      $b = b + 1$; //count the number of detected watermark bit taking one
    end if
  end for
  if $a > b$ **then**
    $w^{\det} = 0$;
  else
    $w^{\det} = 1$;
  end if //the final watermark bit is determined by majority voting mechanism
  $W^{\det} = W^{\det} + w^{\det}$;
end for
return $D_R, W^{\det}$.

Regarding data recovery, the primary key information passed by the embedder is checked. If the primary key of a tuple, where attribute value y' is located, is identical to the received primary key information (mp in Algorithm 2), we will skip to the next attribute value of the same tuple and repeat the same operation. The true attribute value is then restored. A majority voting mechanism is applied to find the final watermark information of each group. For each bit extracted by each tuple in a group, we count the sum of the bit taking zero and the sum of the bit taking one, respectively. The bit with the higher sum is taken as the final watermark bit. This mechanism makes watermark more robust. The watermark extraction and data recovery algorithm is presented in Algorithm 2.

4 Experimental Results and Analysis

In this section, we report experimental results. The main purpose of experiments is to test the robustness against malicious attacks and evaluate the effect on the data quality of proposed scheme. We compare GAHSW with the state-of-the-art reversible database watermarking schemes, such as DEW [13], GADEW [16], FFADEW [17], PEEW [18] and RRW [19]. We also compare GAHSW with the scheme of HSW proposed in this paper to evaluate the effect of GA in GAHSW. In RRW, the watermark information is obtained by GA, which results in difficulty to compare RRW with our method with the same watermark. Therefore, in the following experiments, we only compare GAHSW with DEW, GADEW, FFADEW, PEEW and HSW, and leave the comparison between GAHSW and RRW in 4.5.

Experiments were conducted on a worksation with Intel Core i5 with CPU of 2.30 GHz and RAM of 4 GB. The test database is the Forest Cover Type dataset provided by University of California. The dataset contains 581,012 tuples and 54 attributes. 1,000 tuples and 10 numerical attributes are selected to conduct the experiments. We think 1000 tuples of database is sufficient to verify the effectiveness of GAHSW method and compare with the existing reversible watermarking methods. We generate a synthetic column as the primary key. So totaly we have 11 attributes. The length of watermark (the number of groups) is set to 48. All of the following experiments are carried out under the same embedded watermarks condition.

In this paper, the length of chromosome is kept with 16-bits. Roulette wheel selection mechanism is applied to select the most appropriate chromosome from the population and then pass them to the next generation. The operations of crossover and mutation are performed on parent chromosomes to create offspring. A single point crossover operator is used by exchanging segments of two or more chromosomes to generate a new chromosome inheriting parental characteristics. The mutation operator randomly alters values of one or more cell in gene values of binary chromosomes to create new chromosome and bring diversity in population. The proposed Algorithm 1 and Algorithm 2 are used to embed and extract watermarks, respectively.

Experiments are given in three sections. The first part analyses the effect of some parameters on the performance of the proposed method. Then, the robustness of GAHSW is compared with the state-of-the-art works and HSW scheme under some well-known attacks. Next, the performance on statistical distortion and the effect on classification of the database before and after watermarking are analyzed. Finally, other performance on GAHSW is analyzed, such as the watermarking capacity and the computational time.

4.1 Algorithm Parameter Analysis

In this section, we evaluate the impact of different parameter values of GA on the performance of the proposed GAHSW.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2851517, IEEE Transactions on Knowledge and Data Engineering

Parameters include the population size P, the maximum number of iteration $max_{iteration}$ [17], and weight vectors (W_c and W_d). Note that different databases have different parameter values. We have carried out experiments 10 times to find the most reliable set of parameters of GA.

In the first experiment, the population size P of GA in the proposed method is tested with 10, 20, 30, 34 and 40, respectively. The performance for P values of the proposed method is shown in Figure 5. The x-axis of the graph shows the population size of GA and the y-axis shows the fitness value in the database when using GAHSW watermarking method. The result indicates that GAHSW yields the lowest fitness value when P is 30. Thus we set P to 30.

In the second experiment, the effect of the maximum number of iteration $max_{iteration}$ is evaluated. The technique is allowed to iterate 20, 40, 60, 80, 100, 120 and 140 times during the test. GAHSW is used to watermark database 10 times for each number of iteration. Figure 6 indicates that GAHSW reaches the lowest fitness value when 100 is encountered during tests. So this value is set to 100.

In the third experiment, we test the impact of the two weight vectors. W_c and W_d in the objective function correspond to the capacity of embedded watermark in each group and distortion, respectively. These two weights are selected such that their sum is one. In the objective function, the higher value of W_c indicates the more importance of the capacity of embedded watermark in each group compared to distortion, whereas the higher value of W_d indicates the more importance of distortion compared to the capacity of embedded watermark in each group. CrC and ToD are calculated for W_c values in 0.3, 0.4, 0.5, 0.6 and 0.7 ($W_d = 1 - W_c$), with the results reported in Figure 7. In the tested five values of W_c, $\mathrm{W_c}= 0.6$ yields the highest embedded watermark, and at the same time yields the lowest distortion. Thus, in the following experiments, the weight vectors in Eq. (15) are set to $\mathrm{W_c}= 0.6$ and $\mathrm{W_d}= 0.4$.

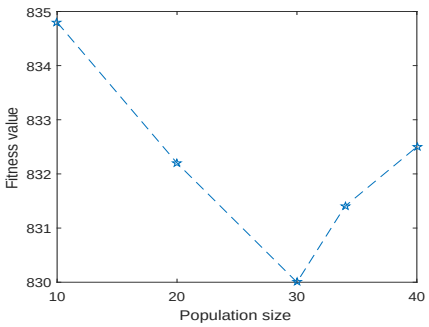


Fig. 5. Comparison of fitness value for different size of population.

4.2 Robustness Analysis

In this section, the robustness of GAHSW watermarking method under well-known database attacks is reported. Robustness is demonstrated through attack analysis. Three types of attacks are considered: insertion, deletion and alteration. Robustness is evaluated by means of the bit error rate (BER), i.e., the ratio of the number of incorrectly extracted

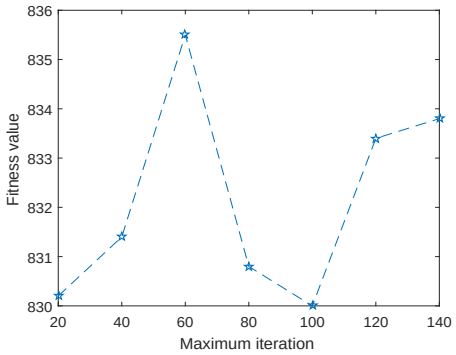


Fig. 6. Comparison of fitness value for different $max_{iteration}$ values.

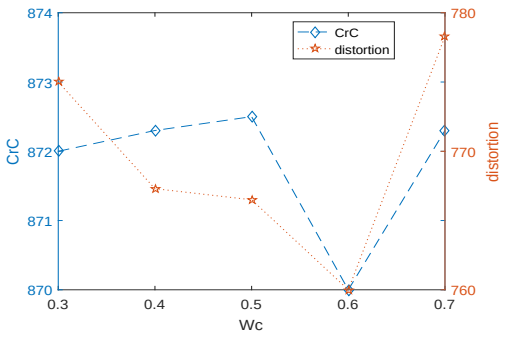


Fig. 7. Comparison of fitness value for different W_c values.

bits to the number of the watermark bits. BER is computed as:

$$BER = \frac{\sum_{i=1}^{N_g} w_i \oplus w_i^{\mathrm{det}}}{N_g} \tag{18}$$

where w_i is an embedded watermark bit and w_i^{det} is the detected watermark bit. We can see that lower value of BER means higher watermarking robustness. The robustness of GAHSW is comparable to DEW, GADEW, FFADEW, PEEW and HSW. These attacks are conducted and comparative results are given below. The experiments are performed to demonstrate watermark detection and data recovery in best-case and worst-case scenarios.

An attacker tries to insert, delete and alter 10, 20, 30 up-to 90 percent of the data in worst-case scenarios. The results are plotted in the charts. Figures presented in this part have BER on vertical axis, which represents the ratio of unsuccessful detection of watermark. Watermark is correctly detected from the database when BER approaches zero. Horizontal axis represents the change of tuples in attack percentage (%) according to the size of the database. The BER of watermark extraction reported is the average of 10 runs of GAHSW since the method relies on stochastic optimization. The proposed method yields lower BER than comparable methods when identical watermark information is embedded into same database. Note that the proposed method sorts attribute columns alphabetically and groups the tuple according to the primary key of the tuples before the watermark embedding and extraction, so rearranging the database does not have any influence.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2851517, IEEE Transactions on Knowledge and Data Engineering

The first experiment to evaluate the robustness of the method is the insertion attack of the watermarked database. In this type of attack, tuples are created randomly and inserted into the database to destroy the watermark. Experimental results show that watermark information can be extracted correctly and completely no matter how many tuples are added to the watermark database. Thus GAHSW is robust to watermark insertion attack. That is to say, no matter how many tuples are inserted, BER under the insertion attack is always 0, and the same is true of other comparative methods.

Deletion attack is performed in the second experiment to show the robustness of GAHSW. This type of attack deletes tuples randomly to destroy the watermark. Figure 8 shows BERs of the extracted watermark of DEW, GADEW, FFADEW, PEEW, HSW and GAHSW methods under deletion attack. When the database suffers from heavy deletion attack, e.g., deleted tuples in database become 90%, DEW, GADEW, FFADEW and PEEW extract the watermark with the BER value of 0.82, 0.95, 0.89 and 0.9, respectively. However, HSW and GAHSW can extract watermarks with BER of 0.52 and 0.5, respectively. The watermark cannot be recovered if the most of tuples containing watermark information are deleted. Therefore, deletion attack has the worst impact on database watermarks. The experiments also show that BER of the extracted watermark increases significantly as the number of deleted tuples increases, and the performance of GAHSW has a better watermark detection rate under this attack compared with other methods.

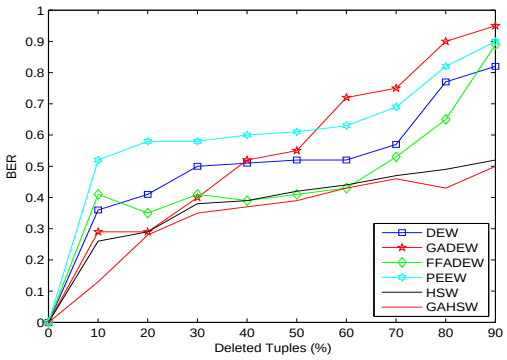


Fig. 8. A Comparison of watermark extraction BER of GAHSW with DEW, GADEW, FFADEW and PEEW after deletion attacks.

Alteration attack is applied on the database as the third experiment. This type of attack modifies attributes of the database randomly. Figure 9 shows BERs of the extracted watermarks of DEW, GADEW, FFADEW, PEEW, HSW and GAHSW methods under alteration attack. We can see that BER of the extracted watermark increases as the number of modified tuples increases. Since watermarked tuples are affected by such attacks, it is difficult to extract the watermark from the remaining unaffected tuples. Figure 9 also shows that the performance of GAHSW is superior to that of other methods under alteration attack, and that even after an attacker alters up to 90 percent of tuples, the BER of extracted watermarks is just 0.4.

In conclusion, a comparative study of GAHSW with other state-of-the-art techniques and HSW is given above. The results of experiments indicate that no matter how

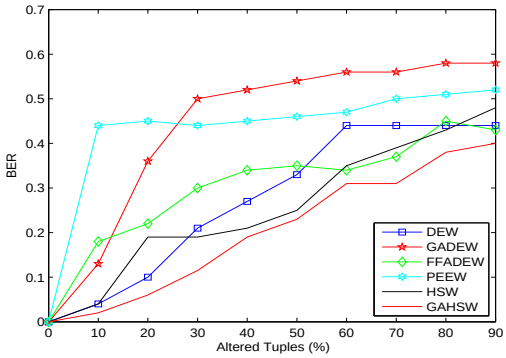


Fig. 9. A Comparison of watermark extraction BER of GAHSW with DEW, GADEW, FFADEW and PEEW after alteration attacks.

many percent of tuples attackers are added, GAHSW can recover both the watermark information and the original data, and that even attackers delete or alter up to 90% of tuples, GAHSW is able to recover at least half of the watermark information. Therefore, GAHSW is highly robust as compared to DEW, GADEW, FFADEW, PEEW and HSW.

4.3 Statistical Distortion and Effect on Classification Analysis

The data quality of GAHSW is evaluated with statistical distortion. The effect of statistical measures is analyzed with the original database and watermarked database. Results are compared with effect on the statistical measures of DEW, GADEW, FFADEW, PEEW and HSW before and after watermarking.

For this purpose, we quantify its statistical distortion through the variations of mean and standard deviation between the attribute before and after watermark embedding, as well as the mean absolute error (MAE). Denote A_i the original attribute and A_i^w its watermarked version. We have:

$$MAE = \frac{\sum_{i=1}^{n} |A_i - A_i^w|}{n} \tag{19}$$

where n is the total number of tuples in the database.

It is worth noting that all the following experimental results are given in average over the 10 simulation experiments. As stated above, we evaluate the statistical distortion through the variations of the mean, the standard deviation, and the MAE of the attribute taken from database before and after watermarking. Table 4 provides the results we obtained from each attribute in all of the tuples of the database (48 groups). The proposed GAHSW is also compared with other techniques by computing statistical measures. In order to see the change of mean and variance more intuitively, the difference in mean (DM) and the difference in standard deviation (DS) for each attribute are given in Table 5. DM and DS are calculated with the following equations:

$$DM = |Mean_D - Mean_{D_W}| \tag{20}$$

$$DS = |Std_D - Std_{D_W}| \tag{21}$$

where $Mean_D$ and Std_D represent Mean and Std value of original database, respectively. $Mean_{D_W}$ and Std_{D_W} represent mean and standard deviation values of watermarked

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2851517, IEEE Transactions on Knowledge and Data Engineering

database, respectively. Besides, the MAE values by using different techniques are shown in the last row.

As shown in Table 5, GAHSW has very small variations of mean and standard deviation for each attribute between the original database and the watermarked database. The biggest variation it causes is 0.101, which is negligible as compared to that of DEW, GADEW, FFADEW and PEEW. Note that, the variation of some attribute values yielded by DEW and PEEW are 0, such as the second attribute shown in Table 5. This is because watermarks are not embedded into such attribute column. Generally, the proposed method has very little impact on the data quality. It is the same for the MAE which quantifies the distortion of the attribute's distribution. GAHSW has the lowest MAE value, that is 0.764, while the MAE values of DEW, GADEW, FFADEW and PEEW are not satisfactory. DEW, GADEW, FFADEW and PEEW yield 28.385, 7.734, 27.334 and 120.896, respectively, which means that large distortion of each attribute is caused by those methods. HSW yields 0.865, which is much lower than the state-of-the-art schemes while higher than GAHSW. This verifies the effectiveness of our proposed GAHSW. In conclusion, GAHSW outperforms the other methods on the performance of statistical distortion significantly.

Furthermore, the effect of GAHSW on classification of the database before and after watermarking is evaluated. For this purpose, various classification methods are used, such as Naive Bayes, Bagging, Adaboost and Decision Tree classifier to classify the Forest Cover Type data. The data is classified into three classes, 1, 2 and 5, which correspond to three different types of cover type. The classification results of the database obtained before and after watermarking with GAHSW are reported in Table 6, where A_{cco}, S_{no} and S_{po} are the classification accuracy, sensitivity and specificity of original database, respectively; A_{ccw}, S_{nw} and S_{pw} are the classification accuracy, sensitivity and specificity of watermarked database, respectively. It indicates that the classification measures remain almost the same before and after watermarking with GAHSW. To summarize, our scheme incurs low statistical distortions and does not bias data mining operations.

TABLE 6
Effect of GAHSW on various classification measures (Accuracy A_{cc}, Sensitivity S_n and Specificity S_p)

classifier	classes	A_{cco}	A_{ccw}	S_{no}	S_{nw}	S_{po}	S_{nw}
NB	1	80.08	79.98	60.62	60.62	85.77	85.64
	2	66.37	66.27	61.88	61.88	72.46	72.46
	5	79.28	80.08	60.64	60.1	83.6	83.6
Bagging	1	92.39	92.39	73.01	72.12	98.06	98.32
	2	91.79	91.79	96.75	97.09	84.78	84.3
	5	78.8	98.8	97.34	97.34	99.14	99.14
Adaboost	1	79.58	79.58	64.6	64.6	83.96	83.96
	2	60.76	60.76	78.8	78.8	35.27	35.27
	5	81.18	81.18	0	0	100	100
DT	1	79.58	79.58	64.6	64.6	83.96	83.96
	2	60.76	60.76	78.8	78.8	35.27	35.27
	5	81.18	81.18	0	0	100	100

4.4 Watermarking Capacity and Time Complexity Analysis

Here, the remaining two criteria, watermarking capacity and time complexity, are utilized to analyze the performance of GAHSW. The details are given below.

Watermarking capacity is evaluated by the ratio of tuples that can be used for watermarking to the total number of tuples in the database. Higher watermarking capacity means that more watermark information can be embedded. The watermarking capacity of our proposed method can reach 100%, that is to say, each tuple can be considered as a group, and each group embeds a watermark bit. In this case, the embedding rate is the maximum, but the robustness will be reduced. DEW and PEEW can also adjust the parameter to make the embedding rate reach 100%. However, GADEW and FFADEW cannot reach 100% because of the limitation of the distortion range. The embedding rate of GADEW and FFADEW are dependent on the database.

The computational time of GAHSW is $(C * G * R * A)$, where C is the number of chromosomes, G is the number of generations, R and A correspond to the total number of tuples and the total number of feature columns in the database, respectively. The computational complexity of GADEW is also $(C * G * R * A)$. However, the computational time of GADEW is much higher than that of GAHSW. In GADEW [16], the size of the chromosome is twice as large as the number of tuples since two attribute values are selected from each tuple for watermark embedding. While in GAHSW, the size of the chromosome C is the length of the key, which is set by the database owner, and in this paper the value is 16, which is much smaller than C in GADEW. The computational complexity of FFADEW is $(R * A + C * G * R * A)$, where the first item $(R * A)$ is the time complexity for initiating population, and the seconde item $(C * G * R * A)$ is similar as that of GADEW.

For a large database, the time complexity of GAHSW for watermark embedding and extraction is O(n). It is worth mentioning that the time for computing the optimal secret key through GA is not included in this calculation because it is a part of preprocessing and is the same as the case of GADEW. For DEW and PEEW, the time complexity is also O (n).

4.5 Further Discussion

First, we discuss the advantage of GAHSW that it can can handle integer values of database. The state-of-the-art reversible relational databases watermarking methods can handle floating point values with less distortion. However, according to the value of MAE as shown in the last row of Table 5, we can see that when dealing with integer data, these techniques will greatly degrade data quality. However, GAHSW can handle the database containing integer and floating point data without causing much distortion. This is because it leads to invisible distortion for each attribute (just modified with 0 or 1) within distortion constraint.

Second, we discuss the reason why GA plays an important role in our scheme. Figure 10 and Figure 11 show the histogram of prediction error of a random group in the two cases: HSW and GAHSW. In HSW, the group is constructed by the secret key generated by random numbers;

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2851517, IEEE Transactions on Knowledge and Data Engineering

TABLE 4
Results of statistical distortion of the database.

Attribute name	original database		DEW		GADEW		FFADEW		PEEW		HSW		GAHSW	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
Aspect	141.844	108.431	141.604	108.598	141.393	108.573	141.940	108.535	142.513	109.803	141.792	108.510	141.801	108.499
Elevation	2867.562	173.698	2867.562	173.698	2867.543	173.631	2867.335	173.752	2867.562	173.698	2867.567	173.776	2867.559	173.785
Hillshade_3pm	139.768	29.329	139.677	29.469	139.518	29.541	139.484	29.482	140.086	29.638	139.777	29.395	139.78	29.382
Hillshade_9am	218.274	20.804	218.203	21.115	218.182	21.221	218.385	20.946	218.491	22.022	218.349	20.824	218.351	20.832
Hillshade_Noon	225.413	14.616	225.499	14.660	225.370	14.872	225.449	14.613	225.413	14.616	225.501	14.646	225.47	14.651
H_D_To_Fire_Points	3184.821	1746.286	3182.716	1751.167	3183.546	1746.616	3186.040	1753.956	3293.577	2044.210	3184.801	1746.370	3184.782	1746.379
H_D_To_Hydrology	236.588	189.965	236.608	190.505	236.876	190.376	237.852	190.698	236.242	190.127	236.498	189.996	236.509	190.003
H_D_To_Roadways	3351.21	1678.140	3353.614	1682.323	3352.767	108.573	3350.611	1688.907	3360.865	1681.792	3351.218	1678.260	3351.207	1678.241
Slope	11.262	6.023	11.262	6.023	11.244	6.015	11.258	6.023	11.279	6.035	11.207	6.052	11.229	6.055

TABLE 5
Results of difference in Mean, difference in Std and MAE.

	DEW		GADEW		FFADEW		PEEW		HSW		GAHSW	
	DM	DS	DM	DS	DM	DS	DM	DS	DM	DS	DM	DS
Aspect	0.24	0.167	0.451	0.142	0.096	0.104	0.669	1.372	0.052	0.079	0.043	0.068
Elevation	0	0	0.019	0.067	0.227	0.054	0	0	0.005	0.078	0.003	0.087
Hillshade_3pm	0.091	0.14	0.25	0.212	0.284	0.153	0.318	0.309	0.009	0.066	0.012	0.053
Hillshade_9am	0.071	0.311	0.092	0.417	0.111	0.142	0.217	1.218	0.075	0.020	0.077	0.028
Hillshade_Noon	0.086	0.044	0.043	0.256	0.036	0.003	0	0	0.088	0.030	0.057	0.035
H_D_To_Fire_Points	2.105	4.881	1.275	0.33	1.219	7.67	108.756	297.924	0.020	0.084	0.039	0.093
H_D_To_Hydrology	0.02	0.54	0.288	0.411	1.264	0.773	1.264	0.733	0.090	0.031	0.079	0.038
H_D_To_Roadways	2.404	4.183	1.557	2.785	0.599	10.767	9.655	3.652	0.008	0.246	0.003	0.101
Slope	0	0	0.018	0.008	0.004	0	0.017	0.012	0.055	0.029	0.033	0.034
MAE	28.385		7.734		27.334		120.896		0.865		0.746	

in GAHSW, the group is constructed by the GA. We can see that the frequency of the peak point in the histogram of GAHSW is higher than that of HSW, so the embedding rate of watermarking in GAHSW is greater than that in the HSW, this also explains the reason why GAHSW improves the robustness of watermarking compared with HSW.

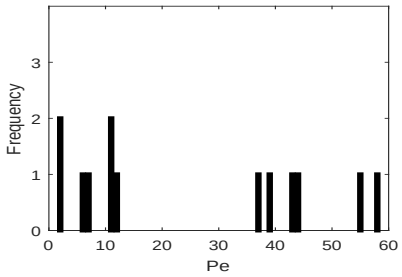


Fig. 10. The histogram of prediction error of a random group in HSW.

Third, we further compare GAHSW with RRW [19]. By comparison, the major drawback of RRW is that the auxiliary data that need to be passed to users is very large. The size of auxiliary data that need to deliver is l times as large as the number of tuples in the original database, where l is the length of the watermark information and usually is large. Besides, the time complexity of RRW is larger than that of GAHSW. Because both methods use GA, so we just compare the time consumed in other phases of the methods. In RRW, there is a data preprocessing phase, where MI of one feature with all other features are calculated, so the time

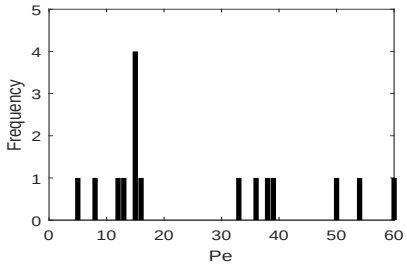


Fig. 11. The histogram of prediction error of a random group in GAHSW.

complexity in this phase is $O(n^2)$. And the time complexity of RRW for watermark insertion and detection is $O(n)$. Thus the final time complexity of RRW is $O(n^2)$. While the time complexity of GAHSW is $O(n)$. Unlike other reversible watermarking methods that randomly select the column to embed the watermark, RRW selects the column with small MI value to embed the watermark. In this method, the watermark information needs to be obtained by GA, and different databases have different watermark information, which is decided by the characteristics of database. Because the watermark information in RRW is not user-defined, while in DEW, GADEW, PEEW and our method, the watermark information is user-defined, we do not conduct comparative experiments on RRW.

Lastly, we further compare GAHSW with the method proposed by R.Halder et al. [4] in the following three aspects. (1) In [4], public watermarking and private wa-

termarking algorithm are proposed to embed watermarks into static attributes and non-static attributes, respectively. The public watermarking allows any end-user to verify the originality and integrity of the database by using the owner's public key, while private watermarking allows the owner to prove her ownership by using her secret key and private key. In our method, we only embed watermarks into static attributes by using a shared key. Besides, the owner and any end-user can extract the watermarks if they have the shared key. (2) The method in [4] is distortion-free. It achieves robustness by choosing the most significant bit positions of the data for watermarking, and achieves fragile by using a cryptographic hash value of each tuple. When attackers modify any attribute data in the database, the watermark will be untouched and the user will get the wrong verification claim. Therefore, the watermarking method on the whole is fragile. However, our method is designed to be distortion-based and robustness, and experimental results prove our method is robust against many malicious attacks. (3) Because the method in [4] is a zero-watermarking algorithm for relational database, any end-users can obtain the original database. By comparison, our method is reversible, and only the legitimate users who have the shared key can extract the watermarks and obtain the original database.

5 Conclusion

Database watermarking has become a hot research as the increasing demand of ownership protection when sharing database information. Traditional watermarking techniques make changes in the database, which greatly compromise the data quality. Reversible watermarking techniques are used to to solve this problem because they can recover original data from the watermarked database and ensure data quality. Many reversible watermarking techniques have been proposed, but these techniques are not robust enough against malicious attacks.

In this paper, a novel reversible and robust technique for watermarking numerical data of relational databases has been proposed. We combine GA with a new proposed Histogram Shifting of prediction error Watermarking (HSW) method to minimize distortion and improve robustness for database watermarking. The characteristics of GAHSW are embedding a watermark bit repeatedly in one group optimally by GA and ensuring the quality of watermarked database by HSW. Experimental results show GAHSW causes less distortion and improves robustness of watermarking. Since the grouping and the majority voting mechanism are used, the proposed method allows extracting most of the watermark information and recovering a large portion of the data even after being subjected to malicious attacks. A number of experiments have been conducted with different attack scenarios. The results of the experimental study show that no matter how many percent of tuples an attacker adds, GAHSW can recover both the embedded watermark and the original data very well; and that even if an attacker deletes or alters up to 90 percent of tuples, GAHSW is able to recover at least half of the watermark information. GAHSW is compared with recently proposed techniques such as DEW, GADEW, FFADEW, PEEW and RRW and the results demonstrate that GAHSW outperforms all of them. Our future work is to develop reversible and robust watermarking for non-numeric data and propose watermarking schemes for the shared databases in distributed environments.

References

[1] Y.-C. Liu, Y.-T. Ma, H.-S. Zhang, D.-Y. Li, and G.-S. Chen, "A method for trust management in cloud computing: Data coloring by cloud watermarking," *International Journal of Automation and Computing*, vol. 8, no. 3, pp. 280–285, Aug 2011.

[2] M. McNickle, "Top 10 data security breaches in 2012 [online]," http://www.healthcarefinancenews.com/news/top-10-data-security-breaches-2012, 2013, Apr. 17.

[3] J. Kiernan, *Chapter 15 - Watermarking Relational Databases*, 2002.

[4] R. Halder and A. Cortesi, "A persistent public watermarking of relational databases," in *Information Systems Security - International Conference, Iciss 2010, Gandhinagar, India, December 17-19, 2010. Proceedings*, 2010, pp. 216–230.

[5] M. J. A. R. Sion and S. Prabhakar, "Rights protection for relational data," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 12, pp. 1509–1525, 2004.

[6] D. Gross-Amblard, "Query-preserving watermarking of relational databases and xml documents," *Acm Transactions on Database Systems*, vol. 36, no. 1, pp. 1–24, 2011.

[7] E. B. M. Shehab and A. Ghafoor, "Watermarking relational databases using optimization-based techniques," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 1, pp. 116–129, 2008.

[8] R. Halder, S. Pal, and A. Cortesi, "Watermarking techniques for relational databases: Survey, classification and comparison." *J. UCS*, vol. 16, no. 21, pp. 3164–3190, 2010.

[9] S. S. M. Kamran and M. Farooq, "A robust, distortion minimizing technique for watermarking relational databases using once-for-all usability constraints," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 12, p. 2694C2707, 2013.

[10] M. Kamran and M. Farooq, "A formal usability constraints model for watermarking of outsourced datasets," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 6, pp. 1061–1072, 2013.

[11] J. Franco-Contreras and G. Coatrieux, "Robust watermarking of relational databases with ontology-guided distortion control," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 9, pp. 1939–1952, 2015.

[12] Y. Zhang, B. Yang, and X. M. Niu, "Reversible watermarking for relational database authentication," *ŁXW*, vol. 17, no. 2, pp. 59–65, 2006.

[13] G. Gupta and J. Pieprzyk, "Reversible and blind database watermarking using difference expansion," 2008, pp. 1–6.

[14] S. Bhattacharya and A. Cortesi, "A distortion free watermark framework for relational databases," in *Icsoft 2009 - Proceedings of the International Conference on Soft-*

ware and Data Technologies, Volume 2, Sofia, Bulgaria, July, 2013, pp. 229–234.

[15] Y. Wu and F. Y. Shih, "Genetic algorithm based methodology for breaking the steganalytic systems," *IEEE Transactions on Systems Man & Cybernetics Part B Cybernetics A Publication of the IEEE Systems Man & Cybernetics Society*, vol. 36, no. 1, pp. 24–31, 2006.

[16] K. Jawad and A. Khan, "Genetic algorithm and difference expansion based reversible watermarking for relational databases," *Journal of Systems & Software*, vol. 86, no. 11, pp. 2742–2753, 2013.

[17] M. B. Imamoglu, M. Ulutas, and G. Ulutas, "A new reversible database watermarking approach with firefly optimization algorithm," *Mathematical Problems in Engineering,2017,(2017-3-6)*, vol. 2017, no. 2, pp. 1–14, 2017.

[18] M. E. Farfoura and S. J. Horng, "A novel blind reversible method for watermarking relational databases," in *International Symposium on Parallel and Distributed Processing with Applications*, 2010, pp. 563–569.

[19] S. Iftikhar, M. Kamran, and Z. Anwar, "Rrwła robust and reversible watermarking technique for relational data," *Knowledge & Data Engineering IEEE Transactions on*, vol. 27, no. 4, pp. 1132–1145, 2015.

[20] V. Khanduja, "Database watermarking, a technological protective measure: Perspective, security analysis and future directions," *Journal of Information Security and Applications*, vol. 37, pp. 38–49, 2017.

[21] Y. Q. Shi, *Reversible Data Hiding*. Springer Berlin Heidelberg, 2005.

[22] G. Xuan, Q. Yao, C. Yang, J. Gao, P. Chai, Y. Q. Shi, and Z. Ni, "Lossless data hiding using histogram shifting method based on integer wavelets," in *International Conference on Digital Watermarking*, 2006, pp. 323–332.

[23] C. C. Lin, W. L. Tai, and C. C. Chang, "Multilevel reversible data hiding based on histogram modification of difference images," *Pattern Recognition*, vol. 41, no. 12, pp. 3582–3591, 2008.

[24] W. L. Tai, C. M. Yeh, and C. C. Chang, "Reversible data hiding based on histogram modification of pixel differences," *IEEE Transactions on Circuits & Systems for Video Technology*, vol. 19, no. 6, pp. 906–910, 2009.

[25] D. G. Yeo, H. Y. Lee, and B. M. Kim, "High capacity reversible watermarking using differential histogram shifting and predicted error compensation," *Journal of Electronic Imaging*, vol. 20, no. 1, pp. 40–43, 2011.

[26] H. T. Wu and J. Huang, "Reversible image watermarking on prediction errors by efficient histogram modification," *Signal Processing*, vol. 92, no. 12, pp. 3000–3009, 2012.

[27] L. Luo, Z. Chen, M. Chen, X. Zeng, and Z. Xiong, "Reversible image watermarking using interpolation technique," *IEEE Transactions on Information Forensics & Security*, vol. 5, no. 1, pp. 187–193, 2010.

[28] X. Gao, L. An, Y. Yuan, D. Tao, and X. Li, "Lossless data embedding using generalized statistical quantity histogram," *IEEE Transactions on Circuits & Systems for Video Technology*, vol. 21, no. 8, pp. 1061–1070, 2011.

[29] X. Li, W. Zhang, X. Gui, and B. Yang, "A novel reversible data hiding scheme based on two-dimensional difference-histogram modification," *IEEE Transactions on Information Forensics & Security*, vol. 8, no. 7, pp. 1091–1100, 2013.

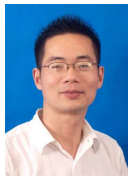
[30] B. Yang, M. Schmucker, C. Busch, X. Niu, and S. Sun, "Approaching optimal value expansion for reversible watermarking," in *The Workshop on Multimedia and Security*, 2005, pp. 95–102.

[31] G. Xuan, Y. Q. Shi, P. Chai, X. Cui, Z. Ni, and X. Tong, "Optimum histogram pair based image lossless data embedding," in *International Workshop on Digital Watermarking*, 2008, pp. 264–278.

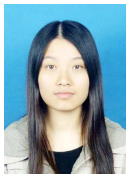
[32] J. Wang, J. Ni, X. Zhang, and Y. Q. Shi, "Rate and distortion optimization for reversible data hiding using multiple histogram shifting," *IEEE Transactions on Cybernetics*, vol. 47, no. 2, p. 315, 2017.

[33] A. Khan, S. F. Tahir, A. Majid, and T. S. Choi, "Machine learning based adaptive watermark decoding in view of anticipated attack," *Pattern Recognition*, vol. 41, no. 8, pp. 2594–2610, 2008.

[34] M. Yoo, "Real-time task scheduling by multiobjective genetic algorithm," *Journal of Systems & Software*, vol. 82, no. 4, pp. 619–628, 2009.



Donghui Hu received the B.S. degree from Anhui Normal University, Wuhu, China, in 1995, the M.S. degree in computer science and technology from University of Science and Technology of China, Hefei, China, in 2004, and Ph.D. degree in information security from Wuhan University, Wuhan, China, 2010. Currently, he is an Associate Professor in the School of Computer Science and Information at Hefei University of Technology. His research interests include information security, machine learning methods in network security detection, and privacy protection.



Dan Zhao received her B.S. degree from Anhui University of Finance and Economics, China, in 2016. She is currently pursuing the M.S. degree in information security from Hefei University of Technology, China. Her research interests include privacy and security in database.

Shuli Zheng received the B.S. degree from the Department of Electrical Engineering (1996), the M.S. and Ph.D. of Computer Applications (1998 - 2003) from the School of Computer Science and Information Engineering, Hefei University of Technology. She is an Associate Professor in the School of Computer Science and Information Engineering, Hefei University of Technology, Her current research interests include information hiding and multimedia content security.