Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/mbe

MBE, 20(12): 21315–21336.
DOI: 10.3934/mbe.2023943
Received: 18 August 2023
Revised: 03 November 2023
Accepted: 22 November 2023
Published: 30 November 2023

Research article

Low distortion reversible database watermarking based on hybrid intelligent algorithm

Chuanda Cai[1], Changgen Peng[1,2,*], Jin Niu[1], Weijie Tan[1,2,3] and Hanlin Tang[4]

[1] State Key Laboratory of Public Big Data, Guizhou University, Guiyang 550025, China

[2] College of Computer Science and Technology, Guizhou University, Guiyang 550025, China

[3] Key Laboratory of Advanced Manufacturing Technology, Ministry of Education, Guizhou University, Guiyang 550025, China

[4] Guizhou ShuJuBao Network Technology Co.Ltd, Guiyang 550025, China

* **Correspondence:** Email: peng_stud@163.com.

Abstract: In many fields, such as medicine and the computer industry, databases are vital in the process of information sharing. However, databases face the risk of being stolen or misused, leading to security threats such as copyright disputes and privacy breaches. Reversible watermarking techniques ensure the ownership of shared relational databases, protect the rights of data owners and enable the recovery of original data. However, most of the methods modify the original data to a large extent and cannot achieve a good balance between protection against malicious attacks and data recovery. In this paper, we proposed a robust and reversible database watermarking technique using a hash function to group digital relational databases, setting the data distortion and watermarking capacity of the band weight function, adjusting the weight of the function to determine the watermarking capacity and the level of data distortion, using firefly algorithms (FA) and simulated annealing algorithms (SA) to improve the efficiency of the search for the location of the watermark embedded and, finally, using the differential expansion of the way to embed the watermark. The experimental results prove that the method maintains the data quality and has good robustness against malicious attacks.

Keywords: reversible watermarking; data recovery; firefly algorithm; simulated annealing algorithm; robustness

1. Introduction

In the era of exponential data growth, sharing information has become an important part of business, industry and academic research and these activities involve the trading of data. Examples include weather

forecasting, medical data analysis and credit reports of financial institutions. Therefore, protecting data copyrights and preventing leakage of shared data have become challenging issues today [1], and database watermarking, as an important technique for database security, can stop the illegal dissemination of data.

At present, scholars for the database watermarking problem put forward the least significant bit (LSB), the use of histogram statistical analysis, text steganography, digital fingerprinting, optimization algorithms based on the reversible database watermarking, blockchain-based watermarking technology and other methods, which are simple and easy to implement. The hidden nature of the LSB is high, but the LSB is susceptible to a variety of attack means of destruction, limited watermarking capacity and poor robustness in practical applications. Embedded watermarking using histogram statistical analysis is reversible and has a large watermarking capacity, but it is susceptible to random noise interference and it is difficult to cope with attack analysis. Text steganography has a high degree of covertness and can cope with attack analysis but its capacity is limited, which restricts the application in large-capacity information transmission [2]. Digital fingerprints are robust and fast to recognize and retrieve, but the original data cannot be recovered after the data is converted into fingerprints. Blockchain technology utilizes distributed database technology, which is slow to process and requires complex operation and maintenance development for large databases [3]. Reversible database watermarking based on optimization algorithms inserts watermarks to protect data in the form of finding the optimal location for embedding watermarks, which can protect intellectual property rights, prove ownership and ensure authentication and security of the content [4], and has the advantages of watermarking capacity, robustness and recovery of the original data in large databases.

Addressing the shortcomings inherent in traditional watermarking schemes, which often encompass issues such as irreversibility, limited capacity, inadequate robustness and suboptimal optimization, this paper introduces an innovative approach to reversible database watermarking. Our principal objective is to bolster algorithmic robustness across a spectrum of database attack scenarios, while concurrently addressing the deficiencies that plague conventional watermarking techniques. The primary contributions of this paper can be summarized as follows:

- A new reversible database watermarking algorithm is proposed, and after a series of experimental evaluations, it is found that the method exhibits better performance than traditional algorithms while maintaining the watermark embedding tolerance. Experiments on the database with 50% attribute or tuple deletion still maintain the watermark extraction rate above 75% stably, and the extraction rate can be maintained at 100% under the attack of adding tuples and attributes. This result strongly proves that our method can better protect the integrity of the database watermark information.
- The weighting coefficients are introduced into the design of the optimization function (or brightness function) as an indicator of the performance of the algorithm. At the data embedding watermark preparation stage, the data copyright owner can change the weighting parameters in combination with the actual application requirements. It not only determines the capacity size of the watermark embedding, but also directly affects the degree of data distortion, further enhancing the autonomy and flexibility of the data owner to protect the intellectual property rights of the data.
- In order to cope with the problem of huge data distortion that can be caused by a large number of watermark embeddings, a differential extension technique is used to embed watermarks into individual tuples. The aim is to decentralize the embedding of the watermark information into various sub-parts of the tuple while keeping the data relatively stable. The problem of serious data distortion caused by centralized large-scale embedding is avoided, while the reversibility

and integrity of data manipulation is also ensured. A symmetric encryption algorithm is used to realize fast watermark verification under the premise of protecting data security. Compared with the traditional large number judgment method, this strategy has significantly improved in efficiency and accelerated the watermark verification process.

2. Related work

In 2003, Kiernan et al. [5] proposed a database watermarking technique to protect relational database copyright issues. The approach is to embed the watermark data on the numeric fields of the database, classify each tuple using a message authentication code (MAC), thereby obtaining the embedding position of each watermark bit and embedding the watermark using LSB. The disadvantage of this approach is that the tuples are not uniformly categorized and, in extreme cases, there are many candidate tuples in some categorization groups and only a few tuples to choose from in other categorization groups. When an attacker attacks the lowest valid bit of a numeric field in a database, a large number of watermarks will be lost, and the watermarking capacity of this approach is also very small. In 2006, Zhang et al. [6] proposed a reversible database watermarking scheme based on the attribute difference to construct a histogram. In the same year, Zhang and Niu [7] proposed a reversible watermarking scheme based on heteroskedastic operation for relational databases. In 2008, Shehab et al. [8] proposed an optimization algorithm based on the genetic algorithm in order to reduce the distortion of the data caused by embedded watermarks, which turns the search for the location of the watermark to be embedded into an optimization problem. Gupta et al. [9] proposed a database watermarking embedding technique based on differential extension watermarking (DEW) in 2009, because of the DEW in this scheme is based on MAC and is randomly selected to be embedded in a watermark location, so this embedding watermarking technique will make the data distortion too large; in practical use it will be selected by setting a threshold to insert or not, resulting in the reduction of watermarking capacity. In 2013, Chang et al. [10] proposed histogram transform based on the blind reversible robust watermarking (BRRW) technique, which realizes the insertion of reversible watermarks against relational databases. In 2015, Iftikhar et al. [11] proposed a new scheme for selecting the basis of watermark embedding location based on the concept of information according to the characteristics of genetic algorithm (GA). In 2017, Imamoglu et al. [12] used the one way hash function Secure Hash Algorithm (SHA) to sort the tuple of the database according to the primary key as well as the input key, the firefly optimization algorithm (FA) based on the database to find the best location for embedding the watermark, the DEW technique to achieve the embedding of the watermark and recovery of the data and finally stored the location of the watermark and extracted the watermark location in the extraction phase. In 2019, Hu et al. [13] used the genetic algorithm to select the best key for grouping of the database and embedded the watermark after offsetting the histogram of prediction error.

In the process of reversible watermarking, the distortion constraint ensures the imperceptibility of the watermark and avoids a large amount of data quality loss. The severe loss of data quality makes it an easy target for attacks, which deteriorates the robustness of watermarking. Therefore, there may be a potential conflict between robustness and distortion constraints [14]. Although genetic algorithm differential extension watermarking (GADEW) and firefly algorithm differential extension watermarking (FADEW) also use optimization methods to select the best location to minimize watermark embedding distortion, the distortion caused by the embedded watermark is still significant. In order to maximize the robustness and

minimize the distortion and seek the optimal solution as much as possible within the required distortion range, so as to determine the appropriate watermark embedding location [15–17], this paper proposes a combination of the FA and the SA to solve this problem.

3. Definitions for DE, FA and SA

3.1. Differential extension

Differential extension (DE) technology was initially widely used in the field of audio decoding and audio processing to improve audio compression, transmission and quality. Later, DE technology gradually migrated to the field of database watermarking to form DEW, which protects data in databases by embedding almost imperceptible digital signals to ensure that copyrighted information is not illegally exploited, in addition to authenticating the source of the data. After embedding watermarks in databases, distortions are usually difficult to recognize by observation, but in certain application scenarios with high-precision data, even small data variations may lead to undesirable results. In order to preserve the robustness of data usage, reversible database watermarking techniques based on differential extensions have emerged. These techniques not only allow watermarking to be extracted during the data validation phase, but also enable full recovery of the original data. For a deeper understanding of the transformation and recovery process of differential expansion, some sample data demonstrations are provided below:

Suppose x, y are the data of some tuple in the database where x = 206, y = 201, and the bit of information to be embedded b = 1.

Step 1: Calculate the average of x and y, $avg = \lfloor \frac{x+y}{2} \rfloor = 203$, difference: $d = x - y = 5$.

Step 2: Defining the new margin $\tilde{d} = 2d + b = 11$, calculate the new $\tilde{x}$ and $\tilde{y}$, $\tilde{x} = avg + \lfloor \frac{\tilde{d}+1}{2} \rfloor = 209$, $\tilde{y} = avg - \lfloor \frac{\tilde{d}}{2} \rfloor = 198$

Above is the forward step of DE. After getting $\tilde{x}$ and $\tilde{y}$, use them to replace the original x and y to get the data containing watermark. The method to recover the original data and the corresponding bit b is as follows:

Step 1: Calculate the average of $\tilde{x}$ and $\tilde{y}$, $a\tilde{v}g = \lfloor \frac{\tilde{x}+\tilde{y}}{2} \rfloor = 203$. Calculate the difference $\tilde{d} = \tilde{x} - \tilde{y} = 11$

Step 2: Extract a watermark bits $b = \tilde{d} - \lfloor \frac{\tilde{d}}{2} \rfloor \times 2 = 1$

Step 3: The original data can be recovered from d and b obtained from steps one and two, $x = avg + \lfloor \frac{\lfloor \tilde{d}/2 \rfloor + 1}{2} \rfloor = 206$, $y = avg - \lfloor \frac{\lfloor \tilde{d}/2 \rfloor}{2} \rfloor = 201$.

3.2. Firefly brightness and behavior

Biologically inspired algorithms have become popular in solving global optimization problems such as the traveling salesman problem (TSP), also known as swarm intelligence (SI)-based algorithms due to their simulation of the group characteristics of living organisms. A particle swarm optimization algorithm was proposed in 1995 based on the swarming behaviors of birds and fish [18]. Subsequently, ant colony optimization [19] and artificial bee colony optimization [20] were proposed. In 2008, Yang et al. [21] proposed the FA to solve the nondeterministic polynomial (NP) problem based on the glowing behavior of fireflies in nature. The algorithm, as a stochastic algorithm, cannot guarantee an optimal solution in a certain time, but it can converge to a value in a certain time, so as to find the local optimal embedding location of the watermark in the database. Set the firefly population in a certain spatial range S as F, the attraction function is A if two fireflies are attracted to each other, then $A = 1$; otherwise, $A = 0$, the

brightness function is I, the attraction is β, the movement function is M, then the FA algorithm has the following three properties:

$$\forall f_1, f_2 \in F, A(f_1, f_2) = 1 \tag{1}$$

$$\forall f_1, f_2 \in F, M(f_1, f_2) = \begin{cases} f_1 \quad to \quad f_2, & \mathrm{I}(f_1) < I(f_2) \\ f_1, f_2 \quad notmove, & \mathrm{I}(f_1) = I(f_2) \\ f_2 \quad to \quad f_1, & \mathrm{I}(f_1) > I(f_2) \end{cases} \tag{2}$$

$$if \;\; I(f) = max, M(f) = random(S). \tag{3}$$

In the FA, the two fireflies' relative brightness is determined by initial brightness and light absorption coefficient as well as distance, assuming the light absorption coefficient is γ, the distance of any pair of fireflies is r_i^j and the brightness of selected fireflies is I_0. The firefly's relative brightness is calculated by the formula:

$$I = I_0 \mathrm{e}^{-\gamma \mathrm{r}_i^j}. \tag{4}$$

The mutual attraction between fireflies determines the flight direction of fireflies, in the space region. Suppose there are three fireflies F_1, F_2, F_3 ; the attraction of F_1 to F_2 is greater than the attraction of F_3 to F_2 and F_2 tends to fly to F_1. The mutual attraction of the fireflies is directly proportional to the relative luminance of the fireflies, and the attraction is determined by the maximum attraction (attraction when the distance is zero) with the light absorption function and the distance, assuming that the maximum attraction is β_0, the light absorption coefficient is γ and the distance of the two fireflies is r_i^j, then the formula of the mutual attraction of the fireflies is as follows:

$$\beta(\mathbf{r}) = \beta_0 \mathbf{e}^{-\gamma \mathbf{r}_\mathbf{i}^{\mathbf{j}^2}}. \tag{5}$$

Different moments correspond to different positions of the same firefly, and their position is not only determined by mutual attraction, but also by the random movement of fireflies. Assuming that the position of firefly i at time t is $X_i(t)$ and the random movement of firefly i is $\alpha\epsilon_i$, then the position function of firefly i at time t+1 can be expressed as follows:

$$X_i(t+1) = X_i(t) + \beta(\mathrm{X_j(t)} - \mathrm{X_i(t)}) + \alpha\varepsilon_\mathrm{i}, \tag{6}$$

where α is a randomization parameter and ϵ_i is a random number generator. $x_j(t) - X_i(t)$ denotes the distance between the best firefly and the current firefly at time t, i.e. r_i^j is defined as follows:

$$r_i^j = \mid x_j - x_i \mid = \sqrt{\sum_{k=1}^{d} \left(x_{j,k} - x_{i,k}\right)^2}. \tag{7}$$

3.3. Simulated annealing

The earliest idea of the simulated annealing algorithm (SA) was proposed in 1953 by Metropolis et al. It is a stochastic optimization algorithm based on the Monte-Carlo iterative solution strategy, and its starting point is based on the similarity between the annealing process of solids in physics

and general combinatorial optimization problems. The essence of the SA is to randomly search for a globally optimal solution of the objective function in the solution space, accepting the new solution if it is better than the current one or else accepting the poor solution with a certain probability. Let the initial temperature be T_0, the solution be x, the objective function be E and the cooling coefficient be k. The steps are as follows:

Step 1: Initialize t, x_0, and E, $t = 0$, $x_0 = random(x)$, $E = E(x_0)$.

Step 2: $x_1 = random(x)$, $E_1 = E(x_1)$, $\Delta E = E(x_1) - E(x_0)$.

Step 3: If $\Delta E > 0, x = x_1$. Otherwise, accept the new solution with probability $e^{-\Delta E/kT}$ and lower the temperature T=kT, $k \in [0, 1]$.

Step 4: Repeat steps one to three until the temperature reaches the set threshold or the temperature drops to zero.

4. Design and implementation of database watermarking scheme

This section describes the use of firefly algorithm and simulated annealing differential extension watermarking (FASADEW) to embed reversible watermarks in relational databases. FASADEW improves the robustness of watermarks while reducing data distortion. The main architecture of FASADEW is shown in Figure 1. The algorithm consists of three modules: (1) Data preprocessing; (2) watermark embedding; (3) watermark extraction.Data preprocessing consists of two parts: Reordering tuples based on tuple primary key with key and determining the distortion range of attribute columns. The key is used for watermark embedding and extraction.

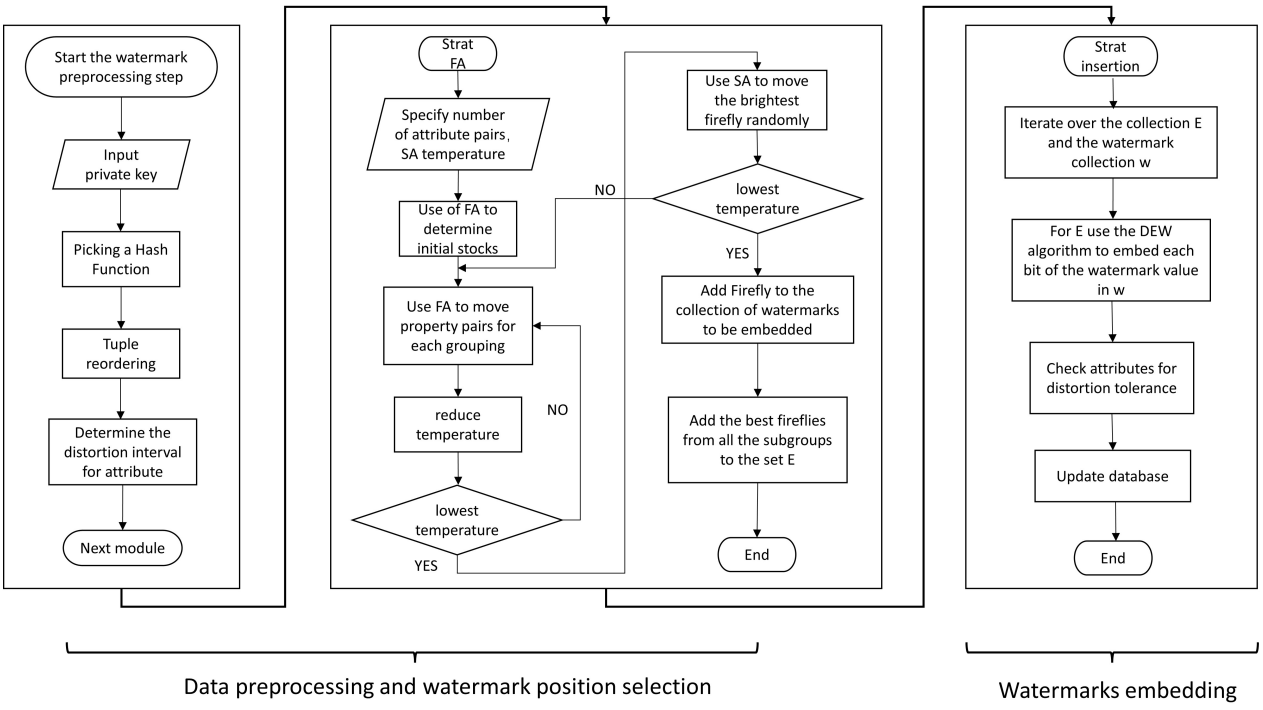


Figure 1. Watermark preprocessing and embedding process.

The watermark embedding phase, which is the focus of this research, is concerned with maintaining a balance between watermark capacity and data distortion prior to embedding and ensuring database availability and watermark imperceptibility.

The extraction phase and the data recovery phase include the efficient extraction of the watermark and the reversible recovery of the data. This study examines the security of embedding watermarks into a database and analyzing the proof of FASADEW through robustness experiments, assuming the presence of an insecure computer network. The possible error rates in extracting watermarks against common attacks, such as tuple insertion and deletion and attribute insertion and deletion, are analyzed. For readers' convenience, we list the notations used in this paper in Table 1.

Table 1. Notations used in the paper.

Symbol	Description
D_i	Group i grouped using the MAC function
τ	Total number of subgroups
$ATmin_x$	The minimum value in attribute x
$ATmax_x$	The maximum value in the attribute x
w	watermark collection
row_w	Number of attribute pairs in Firefly that can be embedded in watermarks
F_j^i	The i-th element in the j-th firefly
n	Customize the number of firefly attribute pairs
M	Number of attributes
$tupleUnit$	Number of tuples in Firefly
M_AT_x	Value of At_x after embedding watermarks
$Distort\_f$	Total distortion of fireflies after embedding watermarks
Ts	brightness threshold
E	A collection of the brightest fireflies in the game
D'	Databases with embedded watermarks
len	Length of watermark

4.1. Data preprocessing

Data preprocessing in the database is used as the first step in embedding the watermark, in order to better distribute the watermark randomly in different tuples with different attributes in the database, to prevent a large number of watermarks from being lost due to tuple attacks and to ensure that the insertion of the watermark is independent of the way the watermark is stored in the database. The primary key is extracted to reorder the database tuples using MAC, and the attributes are reordered according to the attribute name. This approach ensures that the database watermark is robust to the reordering attack on tuple and attribute columns and effectively improves the efficiency of the subsequent creation of the initial population. The tuple grouping operation creates non-repeating groups$\{D_i\}_{(i=1,\cdots,\tau)}$, and the group number *num* is determined by the primary key S and the user-defined private key:

$$\text{num} = H(S \parallel H(S_k \parallel S)) \bmod \tau, \tag{8}$$

where || is the concatenation notation and H is the secure hash function algorithm such as (Message-Digest Algorithm 5) MD5, SHA-1, SHA-2, SHA-512 etc. In this paper, scheme SHA-512 is used to be applied to the grouping to ensure secure grouping.

The reordering groups the tuples, the tuples with the same computation result are put into the set D, the tuples with different computation results are separated and one bit of the watermark needs to be inserted into each set so the length of the watermark determines the size of the symbol τ. Define the distortion tolerance range of each attribute, which is decided by the user. If no distortion tolerance range is set, the algorithm automatically selects the minimum and maximum values of the attribute as the corresponding distortion tolerance range, and the default distortion tolerance range DT is in the interval $[ATmin_x, ATmax_x]$.

4.2. Watermark embedding phase

This module uses the FA to create an initial population for the set screened in the preprocessing stage and customizes the firefly brightness function, moves the fireflies iteratively through the brightness function and the SA and, ultimately, the fireflies are gathered in a cluster and are selected by their high brightness, which is the optimal solution given by the algorithm, i.e., the optimal embedding location of the watermark.

Algorithm 1 Creating the initial population method

Input: D_i, w[i]
Output: Initial population F
Step 1: Loop the method j of step 2, $j \in [1, \cdots, P]$.
Step 2: Method k times of steps 2.1-2.3 of the cycle, $k \in [1, \cdots, M]$.
Step 2.1: Number of user-defined firefly attribute columns n.
Step 2.2: Randomly select the nth attribute At_x in the jth row, ensuring that the selection is not the same each time.
Step 2.3: Loop the method of steps 2.3.1-2.3.3 N times.
Step 2.3.1: At_x requires a DEW operation with all attributes except it, $(M\_At_x, M\_At_y) = DEW(At_x, At_y, w[i])$, $y \in [1, \cdots, A]$, $x \neq y$.
Step 2.3.2: Calculate the distortion after applying DEW algorithm, $distort = \mid M\_At_x - At_x \mid + \mid M\_At_y - At_y \mid$.
Step 2.3.3: Select the attribute pair with the least distortion to add to F_j.
Step 3: Getting the initial population F in D_i.

4.2.1. Creating the initial population

Create an initial population $F = \{F_1, F_2, \cdots, F_p\}$ using P fireflies. The size of P is equalized by the number of tuples in the grouping D_i and each firefly uses a floating point number to represent the position of each firefly $F_1 = 3.2$ represents the third attribute of the first firefly and the second attribute of the first firefly, and the steps to create the initial population are as follows.

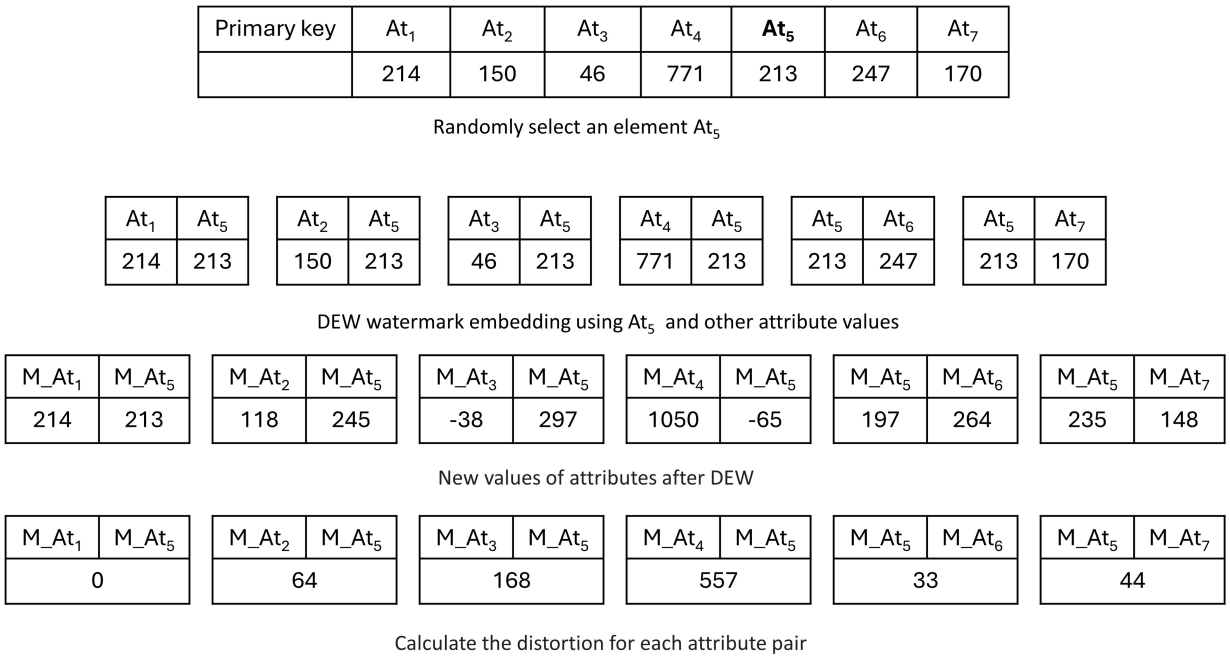


Figure 2. Partial schematic of the algorithm for creating the initial population.

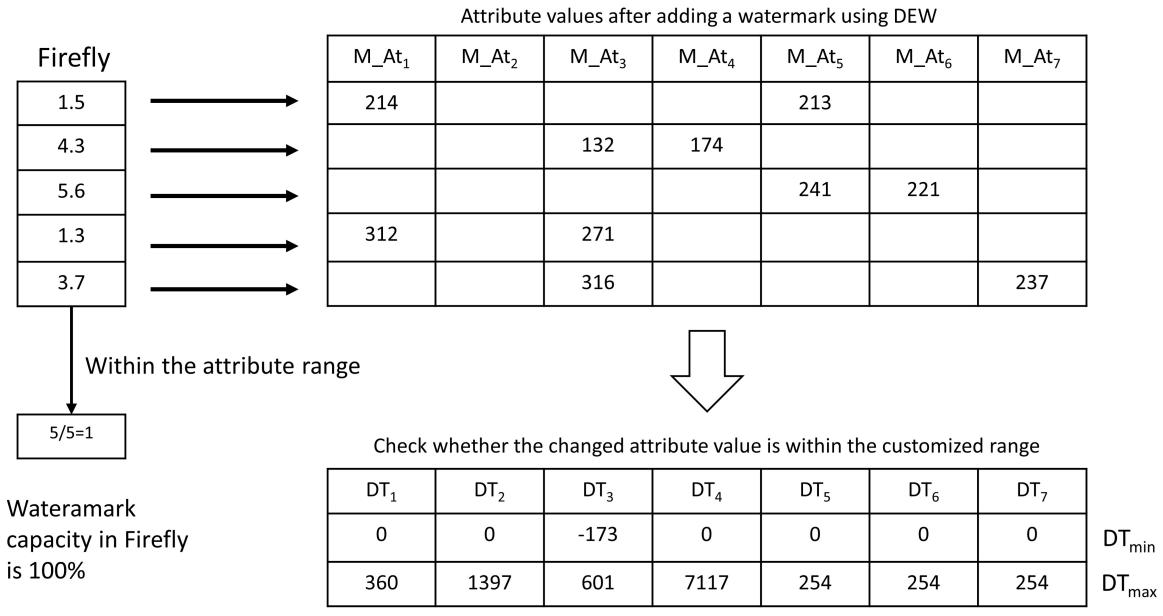


Figure 3. Luminance function - watermark capacity part.

The initial population of fireflies in a set D_i is given in Figure 2, and a random element in the tuple

is selected, assuming a watermark of one. This element is embedded with other attribute values using the DEW algorithm, as shown in Figure 3. After embedding, each attribute pair distortion is calculated as shown in Figure 4 and, finally, a least distorted one is selected to be added to F_j^i. The loop is repeated n times. The randomized algorithm selects some attribute values that are not the global minimum distortion but the local minimum distortion, so it cannot be used to embed the watermark.

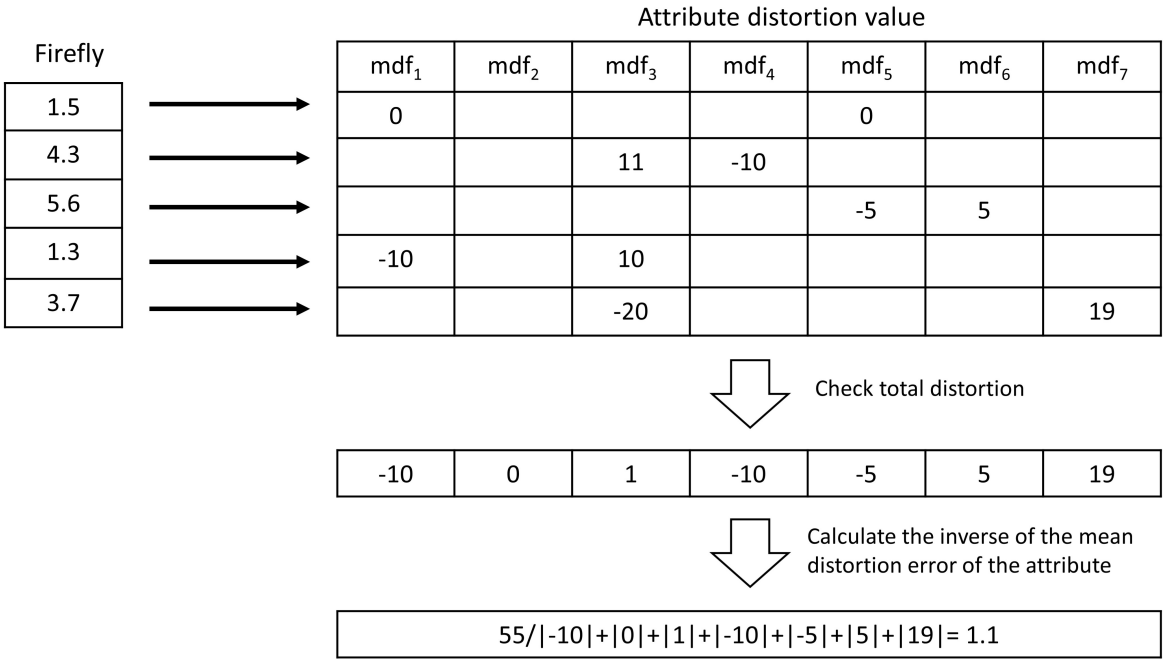


Figure 4. Luminance function-distortion value part.

4.2.2. Define the luminance function of a firefly

The previous step creates the initial population and selects an attribute pair with the smallest distortion as an element in the firefly. In this step the luminance function needs to be defined to ensure the moving direction of the firefly. The luminance function is defined as the sum of the watermark capacity and the weight of the average attribute distortion, i.e., $Br_i = (row_w/n)w_1 + \mid tupleUnit/All\_Distort \mid w_2$. The weight can be adjusted appropriately by the specific engineering requirements and the calculation of the brightness function is shown in Figures 3 and 4. The sum of the weights of the brightness function Br_i weights w_1 and w_2 is one. The higher value of w_1 indicates that a higher watermarking capacity is needed in the project, and the higher value of w_2 indicates that a normalized distortion is more important in the project.

4.2.3. Mobile fireflies

The brightest firefly in each group is judged before moving, and if the brightness of the brightest firefly is greater than the set brightness threshold Ts, then the firefly can be directly added to the set

E. In the original FA, the distance between two fireflies is calculated based on Cartesian coordinates, which are not available in the database, so the distance of fireflies needs to be redefined. In this paper, we use the number of different attribute pairs to define the distance between two fireflies. The movement of fireflies is the movement of ordinary fireflies toward the brightest fireflies. The main purpose of the movement of fireflies is to reduce the differences between fireflies. The movement process needs to be guided by the distance of fireflies to be complete. Since there are m attribute pairs in each firefly, we define the distance of fireflies as follows:

$$D(F_i, F_{\text{best}}) = \sum_{k=1}^{M} (F_k^i \neq F_k^{\text{best}}). \tag{9}$$

How to move to the brightest firefly is very important. You can use a random number to move the firefly and if the firefly contains m attribute pairs, it is different from the brightest firefly in the attribute pairs of t; that is, it is the distance of t with the brightest firefly, then in the firefly to select the range [1,t] attribute pairs for replacement, panning to replace into the attribute pairs of the brightest firefly, the specific steps are shown in Figure 5.

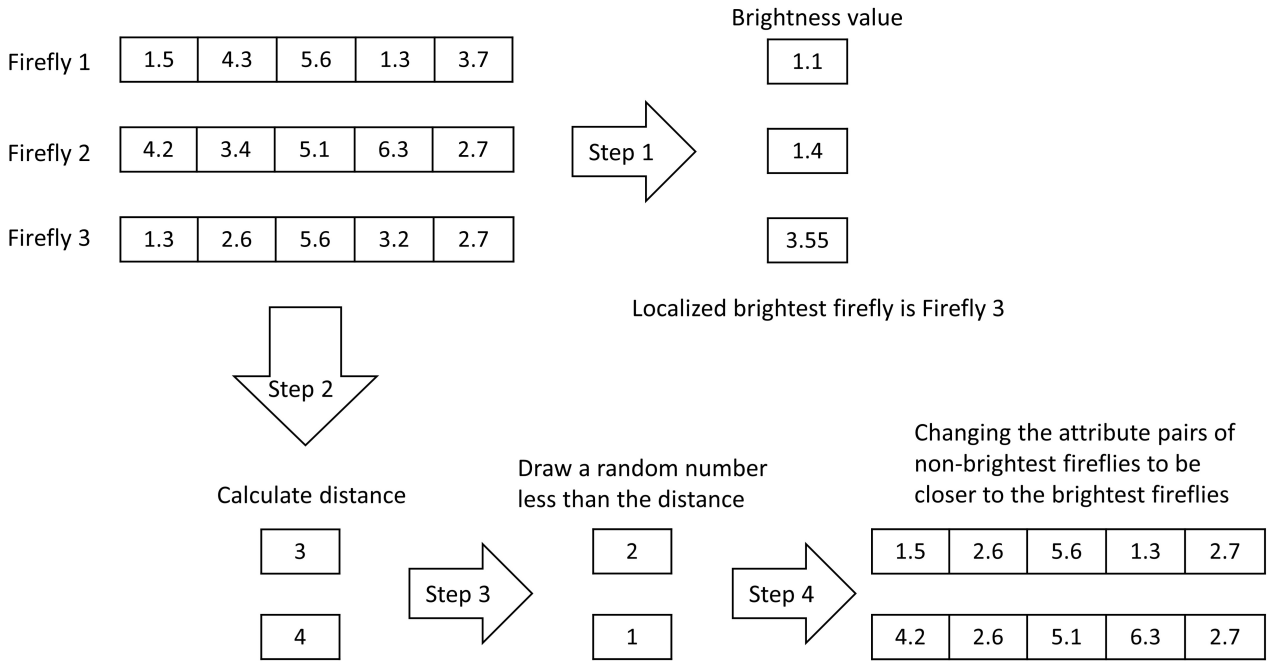


Figure 5. Schematic diagram of firefly movement.

The random movement of the firefly in the algorithm is done by taking a random value for a part of the remaining attribute pairs so that it ensures the randomness of the movement. Next, we use the SA to update the movement of the firefly, set an initial temperature T and the cooling coefficient dT in the firefly after a movement to calculate its brightness value. If the brightness value of the firefly is greater than the brightness value before the movement, then accept the value and then drop the temperature once and use

the SA to move again on a randomly moving attribute pairs to move and repeat the firefly's movement operation. Finally, the cycle stops when the temperature drops to the lowest value of the set temperature.

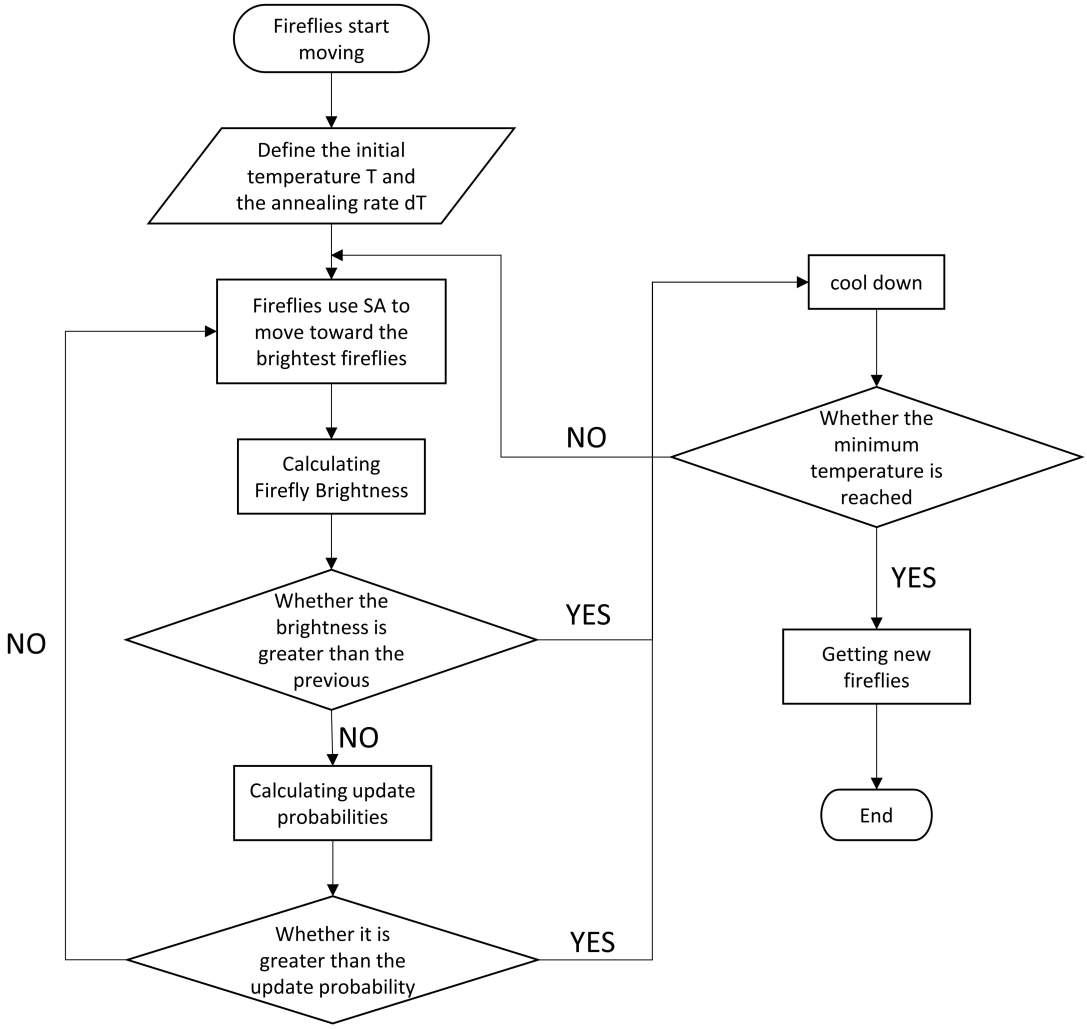


Figure 6. Firefly iterative mobile process.

After all non-brightest fireflies use simulated annealing to complete the move, the SA is applied to the brightest fireflies. After moving the brightest fireflies once, the brightest fireflies in the group are re-judged and the brightest fireflies are iterated until the global optimal solution is found, and the non-brightest fireflies are also moved to the location of the brightest fireflies in a similar position through the hybrid algorithm of fireflies and simulated annealing to form a cluster. The brightest fireflies in the found clusters are added into the set E.

4.2.4. Watermark embedding phase

The brightest fireflies from all groups are added to the set E by a hybrid firefly and SA, and each firefly in the set E specifies the attribute position of the watermark embedding. The first firefly in the

set E represents the first group computed by the MAC function, which contains the best embedding position of the watermark. Each watermark bit is embedded into the database by traversing the set E. The following pseudo-code specifically demonstrates embedding the watermark information into the corresponding set of the database.

Algorithm 2 Watermark Embedding Algorithm

Input: Database D, E, watermark set w, watermark length *len_w*
Output: Database with watermarked values $D^{'}$
Step 1: Repeat steps 2-4 *len_w* times.
Step 2: Extract the *ith* element (x_i, y_i) in the set E and the *ith* element in the watermark set, and embed the watermark $w[i]$ on all attribute pairs of the *ith* element.
Step 3: Embed the watermark into At_x and At_y using DEW algorithm to get M_At_x and M_At_y.
Step 4: Re-write M_At_x and M_At_y back to the database.
Step 5: Getting a new database $D^{'}$.

Eventually, the set E is encrypted and saved locally using the symmetric encryption algorithm advanced encryption standard (AES) to ensure the covertness and security of the watermark embedding.

4.2.5. Watermark extraction

The encrypted set E is decrypted, the tuples are grouped using the MAC function to obtain a series of tuples, the best fireflies in the set E are extracted and each firefly points to an attribute of the current row used for watermark embedding. Next, the watermark information is extracted by applying the DEW inverse operation to the specified attribute through E and the reconstruction restores the original value of the attribute. The pseudo-code of the watermark extraction algorithm is as follows.

Algorithm 3 Watermark Extraction Algorithm

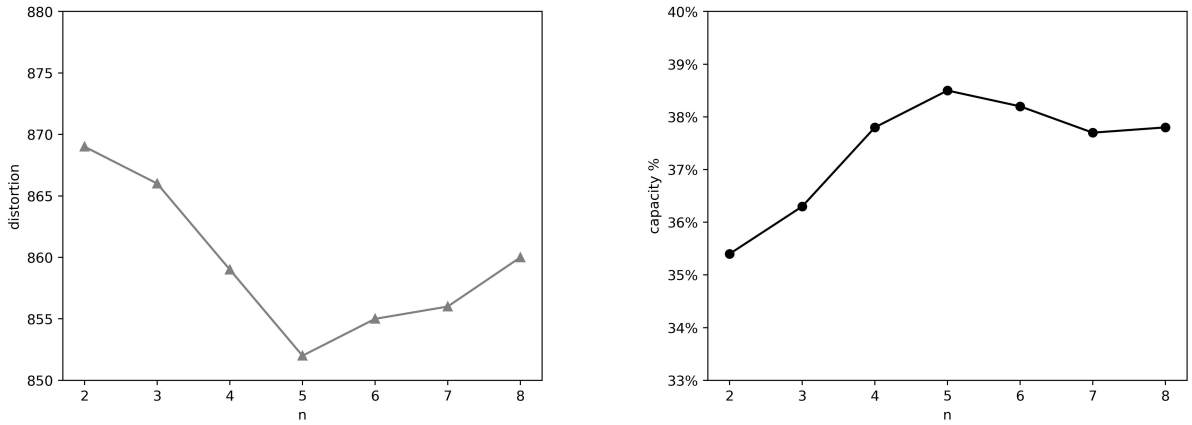
Input: Embedded watermark in database $D^{'}$, encrypted file *K_doc*, private key *sk*
Output: Original database D and watermark
Step 1: Use *sk* to decrypt *K_doc* to get the best firefly set E. Get the number of elements of set set E *E_num*.
Step 2: Use *sk* with $D^{'}$ to reorder the database tuples into groups, aligning the watermark array with the reordered tuples.
Step 3: Repeat step 4 for *E_num* times.
Step 4: The ith reordered tuple and the ith firefly in E are selected, and the watermark is extracted by performing the inverse operation of DEW on all attribute pairs in the fireflies.
Step 5: Obtaining the original database D and watermark.

5. Experimental results

In this paper, we use the University of California Forest Coverage dataset to conduct experimental tests on the dataset, which includes 581,012 tuples and 54 attributes, most of which have the value

of zero. Therefore, only 10 attributes in the dataset are used in the actual experiments to test the watermarking capacity, the degree of distortion and the robustness under some attacks, as well as to determine whether the method of this paper is adaptable to the database with an arbitrary number of tuples by varying the use of the tuples in the experiments. Any number of tuples of the database are adapted. With comparison experiments with the algorithms in [12, 16, 22–26], set the watermark length to 96, and all the experiments are carried out under the same embedded watermarking conditions. The experimental environment is Intel core i5-13400 processor, 16GB DDR4 memory and the operating system is windows 11 host, database mysql. The experiment is divided into four parts. The first part analyzes the watermarking capacity between each algorithm, the second part analyzes the distortion of each algorithm after completing the embedding of the watermark, the third part is the comparison of the classical algorithm and the proposed algorithm in this paper under the comparison of robustness under some known attacks.

As the size of the population in this paper is decided based on the number of tuples in the MAC function grouping, the number of population is constant in the experiment. In the experiment, the number of custom attributes n is taken in the interval [2, 8], respectively, for testing the proposed method in this paper, which is randomly taken in the database 10176 for testing. The performance of this paper's method for the value of n is shown in Figure 7, where the x-axis is the number of customized attributes n and the y-axis in (a) denotes the total database distortion using the FASADEW method. The y-axis in (b) is the watermark capacity using the FASADEW method.



(a) **(b)**

Figure 7. Data distortion and watermark capacity for different number of attribute pairs. (a) Data distortion, (b) Watermark capacity.

The results show that when n = 5, the watermarking capacity is the largest and the data distortion is low, so we set n to five. After determining n, keep n unchanged to change with dT to find the most suitable dT, and do not change T in the experiment because we can determine the annealing rate with dT as the condition. At this time, n = 5, T = 2,000, the method in this paper adjusts the dT value and the watermarking performance is shown in Figure 8. The x-axis in Figure 8 is the cooling factor dT, the y-axis in (a) is the watermarking capacity, the y-axis in (b) is the total distortion of the data and the y-axis in (c) is the time required by the algorithm.

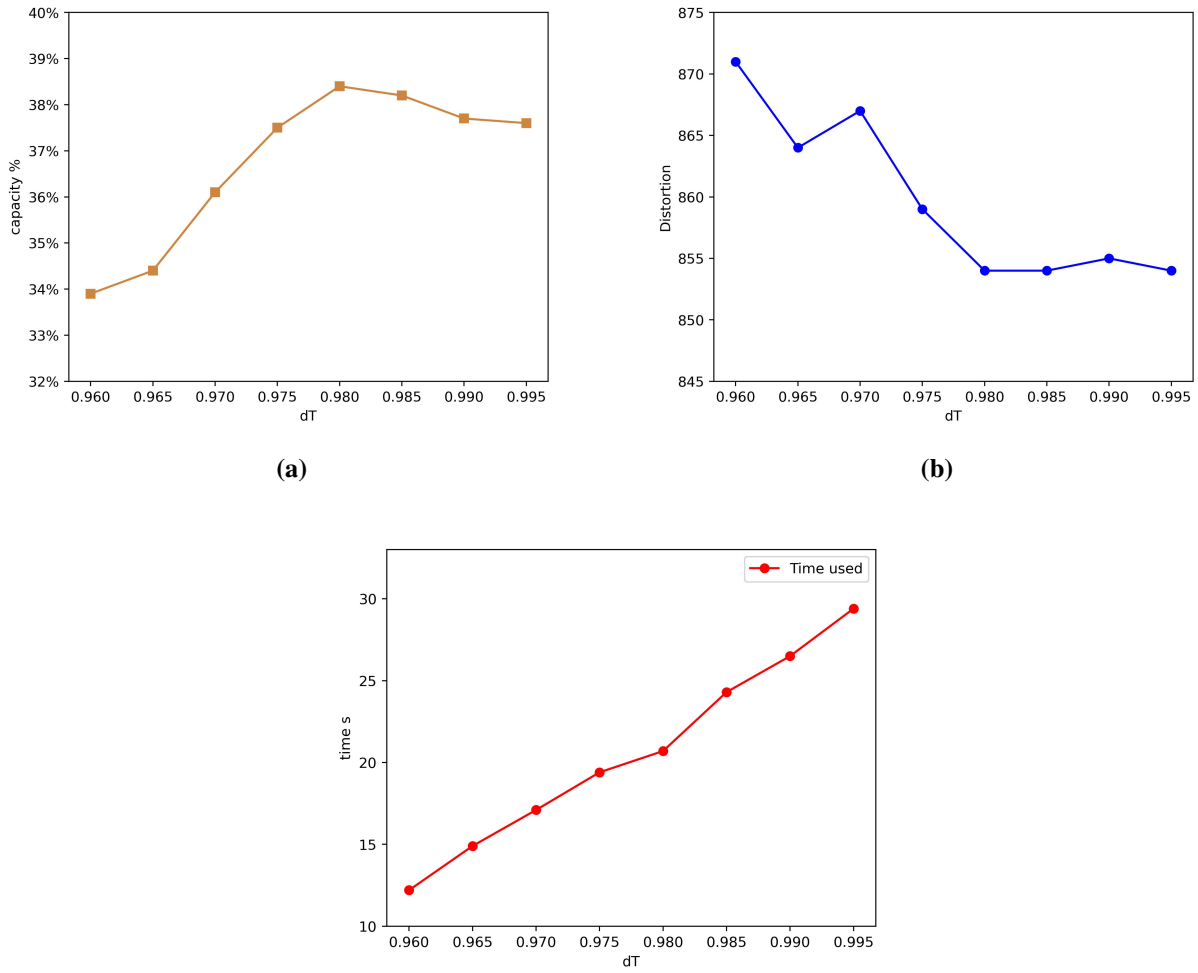


Figure 8. Algorithm performance with different cooling factors. (a) Watermark capacity, (b) Data distortion, (c) Time spent.

From Figure 8 it can be seen that in the testing process the dT = 0.98 watermark capacity reaches the highest, dT = 0.99 when the minimum data distortion, dT = 0.96 when the algorithm takes the least amount of time and dT = 0.99 when the time required is the longest. At this time, to calculate the average of data distortion, we found that the average distortion of dT = 0.99 and dT = 0.98 is not much of a difference considering the efficiency of the algorithm in this paper, so we choose dT = 0.98.

Next, we test the influence of two weight parameters w_1 and w_2 in the luminance function corresponding to the watermark capacity and data distortion weights, and the sum of the two weights is one. The higher value of w_1 in the luminance function indicates that the capacity of embedded watermarks in each group is more important and relative to the distortion, and the higher value of w_2 indicates that the capacity of embedded watermarks in each group is more important and relative to the distortion. Keeping the previous n = 5, T = 2,000, dT = 0.98, the size of data distortion and watermark capacity of w_1 at 0.3, 0.4, 0.5, 0.6 and 0.7 (w_2=1-w_1) were calculated and the results are shown in Figure 9, where the highest embedded watermark was produced by w_1 = 0.6 among the 5 w_1

tested, while the lowest distortion was produced. Therefore, n = 5, T = 2,000, dT = 0.98, $w_1 = 0.6$ and $w_2 = 0.4$ were used in the next experiments.

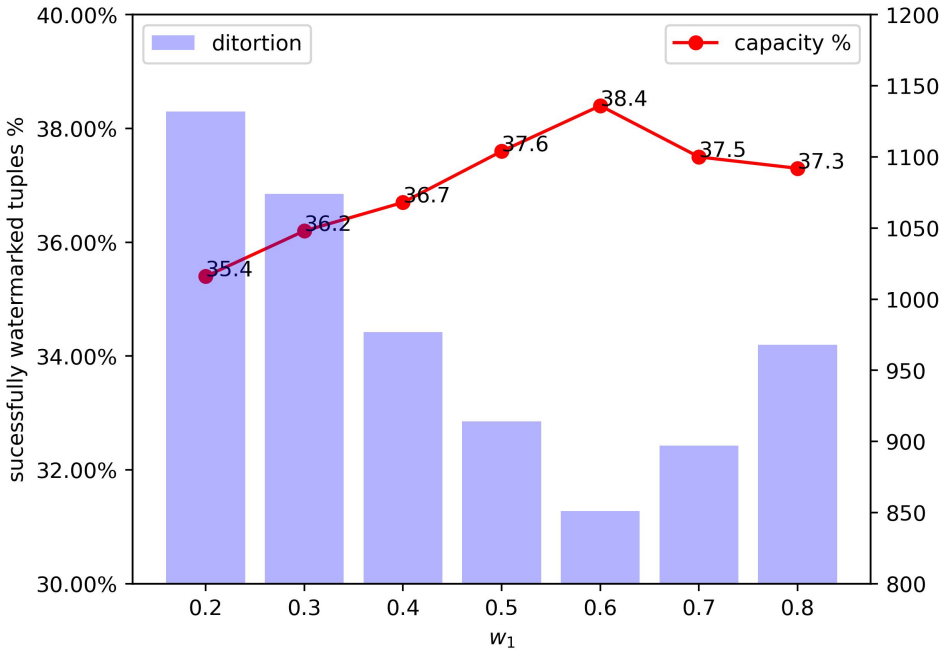


Figure 9. Distortion and watermark capacity with different weights.

5.1. Watermark capacity

Watermark capacity is the maximum number of watermarks or the maximum amount of information that can be embedded in a given database or dataset. A higher watermark capacity means that more watermarks can be embedded or a larger amount of watermark information can be embedded. The higher the capacity, the more robust and stealthy the watermark is, which is very useful for protecting data integrity, tracking the origin of data and resisting malicious tampering [27–29]. In this paper, the watermark capacity is defined as:

$$P_{capacity} = \frac{row_w}{M}. \tag{10}$$

To evaluate the performance of the algorithm, we conducted experiments using 214, 756, 1478, 3986, 5769 and 10176 tuples in the dataset and compared the results with FADEW, particle swarm optimization (PSODEW), DEW and GADEW.

The experimental results show that the watermarking capacity of the FASADEW algorithm is comparable to that of the FADEW algorithm when faced with small datasets. However, in larger datasets (>1478), the watermarking capacity of the FASADEW algorithm has a large boost compared to the FADEW algorithm. This boost is due to the fact that the FASADEW algorithm, guided by the SA, accepts fireflies with a brightness less than the original brightness with some probability. The watermarking capacity using the FASADEW algorithm shows a significant improvement compared to the PSODEW and DEW algorithms, which is mainly due to the fact that the FASADEW algorithm has

more embeddable locations.

Table 2. Comparison of watermarking capacity between algorithms.

Average distortion / Test Tuples / Algorithm	214	756	1478	3986	5769	10176
FASADEW	**32.7%**	**33.6%**	**34.9%**	**37.1%**	**38.3%**	**38.5%**
FADEW [12]	32.5%	33.4%	33.2%	34.6%	35.3%	36.0%
PSODEW [22]	31.1%	32.1%	32.4%	31.8%	31.7%	32.4%
DEW [23]	29.0%	30.1%	30.4%	31.2%	31.3%	31.2%
GADEW [16]	32.1%	33.5%	34.4%	36.5%	37.2%	37.4%

5.2. Distortion value analysis

Watermarking causes distortion of data after embedding, which can be harmful in subsequent use by the authorizer, and when the data is very different from the original data, it can lead to some wrong prediction results or the obtained data not being in line with the expectation [30, 31]. In this paper, the experiments take the average distortion method to calculate the distortion. The formula is as follows:

$$Distort = \frac{\sum_{i=0}^{D} \mid M\_At_x - At_x \mid + \mid M\_At_y - At_y \mid}{DataNum}. \tag{11}$$

where D represents the tuple to be embedded in the watermark and $DataNum$ represents the number of all tuples in the database. Since the DEW algorithm is too long, it is not comparable in the distortion value analysis. The watermark distortion of each algorithm is shown in Table 3 and it can be found that the distortion using in the FASADEW algorithm is less than using FADEW, PSODEW, GADEW algorithms.

Table 3. Comparison of average distortion values by algorithm.

Average distortion / Test Tuples / Algorithm	214	756	1478	3986	5769	10176
FASADEW	**16.3**	**17.2**	**16.9**	**18.4**	**20.3**	**25.4**
FADEW [12]	25.6	26.2	29.4	31.3	31.6	32.9
PSODEW [22]	24.6	36.1	35.6	37.3	37.6	39.8
DEW [23]	31.5	35.7	35.5	37.1	38.3	40.7
GADEW [16]	21.8	25.3	25.9	27.2	29.7	30.9

5.3. Robustness analysis

The robustness of watermarking is mainly shown in some attacks, such as anti-tuple deletion and modification attacks, addition attacks, attribute deletion attacks, tuple reordering attacks, bit reversal, etc., and the main purpose of the attacks is to destroy the watermarking message w or embedding their own watermarks [32, 33]. Under these attacks, whether the watermark can be extracted correctly or not

is a critical issue. In this paper, scheme experiments on some common attacks observe whether the watermark can be extracted correctly under these common attacks. Before this paper, we defined the watermark extraction rate

$$P_{detect} = \frac{Num_w}{Num_{tuple}}, \tag{12}$$

where Num_w is the number of watermarks detected in each type of experiment and Num_{tuple} is the total number of successfully embedded watermarks. The robustness experiments include the cases in which the degree of attack is 10, 20, 30, 40 and 50% of the common attack methods, the watermarks are embedded for 10176 tuples in the experiments and the average value is taken after running 10 times. As shown in Figure 10, the experimental results show that FASADEW has the best performance in the case of tuple reordering attack and tuple addition attack, which can keep all the watermarks extractable, but in the case of attribute deletion attack, the results are not satisfactory. This is due to the fact that the watermarks are embedded in different attributes of a tuple. When the attribute deletion is too much, it will lead to, the watermarks being unable to be extracted.

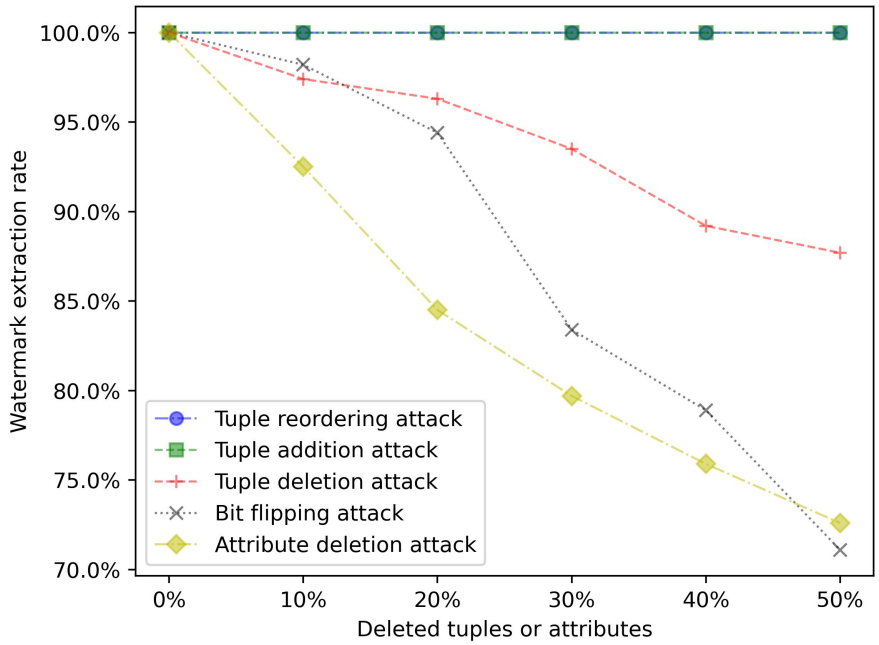
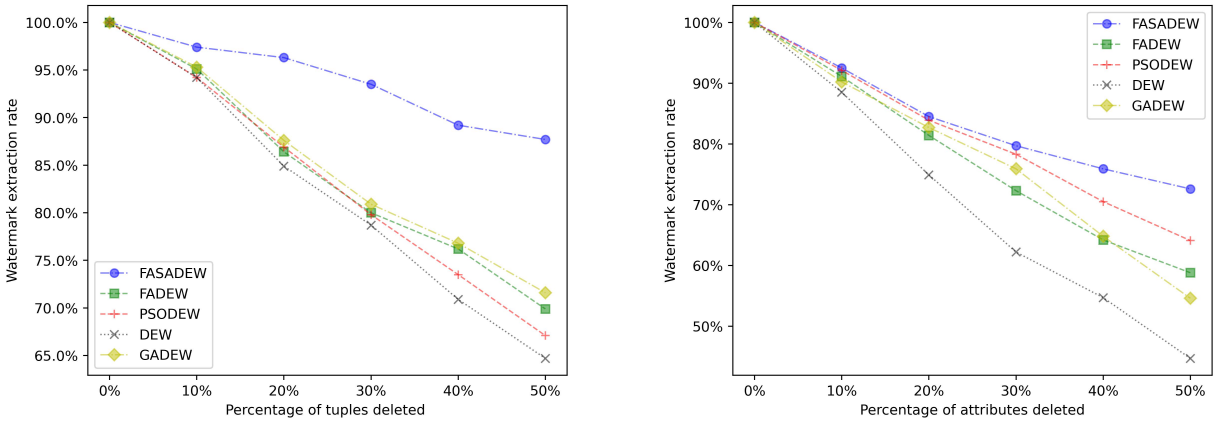


Figure 10. FASADEW's detection rate of watermarking under common attacks.

In the following, the FASADEW algorithm of this paper is compared with FADEW, PSODEW, DEW and GADEW to compare the robustness under tuple deletion attack, attribute deletion attack and tuple reordering attack. Figures 11 demonstrate the detection rate of each algorithm watermark under both attacks.

Experimental results from (a) show that the FASADEW anti-tuple deletion attack is better than FADEW, PSODEW, DEW, GADEW algorithms. In the deletion of 50% of the tuples under the original DEW

algorithm, the FASADEW algorithm improves the watermark extraction rate by 5%, where the enhancement comes from the use of a MAC function to pick out tuples are randomly selected. With the use of a different MAC function, randomly selected tuples are also different and in the end, it will affect the tuple deletion of the watermark extraction rate, so the choice of a good MAC function is also very important.



(a) **(b)**

Figure 11. Watermark detection rate of various algorithms under tuple deletion attack and attribute deletion attack. (a) Tuple deletion attack. (b) Attribute deletion attack.

Attribute deletion is a more serious attack that deletes all the watermarks on the attribute columns while deleting the attribute columns, which is a disastrous way of embedding watermarks using the DEW algorithm. In this regard, in the FASADEW algorithm, in order to increase the capacity of watermark in the attribute values, it is proposed to use fireflies to find the optimal location for embedding the watermark. The optimal location has more than one attribute that can be embedded with the watermark, and we embed all of the multiple locations so as to achieve the effect of defending against attribute deletion attacks. From Figure 11(b), comparing the FADEW, PSODEW, DEW and GADEW algorithms, it can be found that when the proportion of attribute deletion is large, the FASADEW algorithm has strong robustness.

The tuple adding attack is to add tuples to the database using a certain amount of data. Since all the above schemes use DEW for watermark embedding in the embedding phase of the watermark, this approach takes into account the synchronization of the watermark embedding and extracting, so in the tuple adding attack, the watermark extracting rate of these schemes is 100%.

The reordering attack of tuple means that the tuple is disrupted and reordered in a certain way, while for the MAC used by the FASADEW algorithm for tuple filtering, the reordering did not disrupt the sorting of the subsequent MAC function, so the FASADEW algorithm has a watermark extraction rate of 100% under the attack of the reordering of the tuple. This means that the FASADEW is able to ensure that it is completely resistant to the attack of the tuple reordering to ensure the safety of the watermark.

6. Conclusions and future directions

In this paper, we proposed a reversible and robust digital data watermarking technique for relational databases. We combined the FA with the SA method to reduce distortion and improve the robustness of

database watermarking. FASADEW is characterized by selecting the initial population by the FA and using the SA to find the optimal embedding location of the watermark, which guarantees the quality of the watermarked database. Experimental results show that FASADEW has less distortion and improves the robustness of watermarking. Because of the grouping mechanism, the method can extract most of the watermark information and recover most of the data, even if it is maliciously attacked. A large number of experiments are conducted for different attack scenarios. The experimental results show that FASADEW recovers the embedded watermark and the original data well, no matter what percentage of tuples the attacker adds. FASADEW is compared with recently proposed techniques such as DEW, GADEW, FADEW, PSODEW, GADEW etc. and the results show that FASADEW outperforms all of them. Future work is to develop reversible and robust watermarking for non-digital data and to propose watermarking schemes for shared databases in a distributed environment.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work was supported in part by the State's Key Project of Research and Development Plan under Grant2022YFB2701400, in part by the National Natural Science Foundation of China under Grant 62272124.62002080.62361010, in part by The Natural Science Researching Program of D.o.E. of Guizhou (No.Qian Edu. & Tech. [2023] 065), in part by The Research Project of Guizhou University for Talent Introduction (No.[2020]61), in part by the Cultivation Project of Guizhou University(No.[2019]56) and in part by the Open Fund of Key Laboratory of Advanced Manufacturing Technology, Ministry of Education(GZUAMT2021KF[01]).

Conflict of interest

The authors declare there is no conflict of interest.

References

1. X. Tang, Z. Cao, X. Dong, J. Shen, PKMark: a robust zero-distortion blind reversible scheme for watermarking relational databases, in *2021 IEEE 15th International Conference on Big Data Science and Engineering (BigDataSE)*, (2021), 72–79. https://doi.org/10.1109/BigDataSE53435.2021.00020
2. M. L. P. Gort, M. Olliaro, A. Cortesi, C. F. Uribe, Semantic-driven watermarking of relational textual databases, *Expert Syst. Appl.*, **167** (2021), 114013. https://doi.org/10.1016/j.eswa.2020.114013
3. A. S. Alghamdi, S. Naz, A. Saeed, E. Al Solami, M. Kamran, M. S. Alkatheiri, , A novel database watermarking technique using blockchain as trusted third party, *Comput. Mater. Con.*, **70** (2022), 1585–1601. https://doi.org/10.32604/cmc.2022.019936
4. K. E. Drandaly, W. Khedr, A. M. Mostafa, I. Mohamed, A digital watermarking for relational database: state of Art techniques, *Int. J. Adv. Sci. Tech.*, **29** (2020), 870–883.

5. R. Agrawal, P. J. Haas, J. Kiernan, Watermarking relational data: framework, algorithms and analysis, *VLDB J.*, **12** (2003), 157–169. https://doi.org/10.1007/s00778-003-0097-x
6. Y. Zhang, B. Yang, X. M. Niu, Reversible watermarking for relational database authentication, *J. Comput.*, **17** (2006), 59–65.
7. Y. Zhang, X. M. Niu, Reversible watermark technique for relational databases, *Acta Electr. Sin.*, **34** (2006), 2425–2428.
8. M. Shehab, E. Bertino, A. Ghafoor, Watermarking relational databases using optimization-based techniques, *IEEE Trans. Knowl. Data Eng.*, **20** (2008), 116–129. https://doi.org/10.1109/TKDE.2007.190668
9. G. Gupta, J. Pieprzyk, Reversible and blind database watermarking using difference expansion, *Int. J. Digital Crime and Forensics*, **1** (2009), 42–54. https://doi.org/10.4018/jdcf.2009040104
10. C. C. Chang, T. S. Nguyen, C. C. Lin, A blind reversible robust watermarking scheme for relational databases, *Sci. World J.*, **2013** (2013). https://doi.org/10.1155/2013/717165
11. S. Iftikhar, M. Kamran, Z. Anwar, RRW—A robust and reversible watermarking technique for relational data, *IEEE Trans. Knowl. Data Eng.*, **27** (2014), 1132–1145. https://doi.org/10.1109/TKDE.2014.2349911
12. M. B. Imamoglu, M. Ulutas, G. Ulutas, A new reversible database watermarking approach with firefly optimization algorithm, *Math. Probl. Eng.*, **2017** (2017). https://doi.org/10.1155/2017/1387375
13. D. Hu, D. Zhao, S. Zheng, A new Robust approach for reversible database watermarking with distortion control, *IEEE Trans. Knowl. Data Eng.*, **3** (2019), 1024–1037. https://doi.org/10.1109/TKDE.2018.2851517
14. S. Rani, R. Halder, Comparative analysis of relational database watermarking techniques: an empirical study, *IEEE Access*, **10** (2022), 27970–27989. https://doi.org/10.1109/ACCESS.2022.3157866
15. S. Xiang, G. Ruan, H. Li, J. He, Robust watermarking of databases in order-preserving encrypted domain, *Front. Comput. Sci.*, **16** (2022) 162804. https://doi.org/10.1007/s11704-020-0112-z
16. K. Jawad, A. Khan, Genetic algorithm and difference expansion based reversible watermarking for relational databases, *J. Syst. Softw.*, **86** (2013), 2742–2753. https://doi.org/10.1016/j.jss.2013.06.023
17. K. E. Parsopoulos, M. N. Vrahatis, Particle swarm optimization method for constrained optimization problems, in *Intelligent Technologies: From Theory to Applications*, IOS Press, (2002), 214–220.
18. J. Kennedy, R. Eberhart, Particle swarm optimization, in *Proceedings of ICNN'95 - International Conference on Neural Networks*, **4** (1995), 1942–1948. https://doi.org/10.1109/ICNN.1995.488968
19. A. Colorni, M. Dorigo, V. Maniezzo, and others, Distributed optimization by ant colonies, in *Proceedings of the first European conference on artificial life*, (1991), 134–142.
20. D. Karaboga, An idea based on honey bee swarm for numerical optimization, in *Technical report-tr06, Erciyes university, engineering faculty, computer engineering department*, 2005.
21. K. Unnikrishnan, K. V. Pramod, Prediction-based robust blind reversible watermarking for relational databases, *Int. J. Inform. Comput. Secur.*, **14** (2021), 211–228. https://doi.org/10.1504/IJICS.2021.114702

22. X. S. Yang, *Nature-inspired Metaheuristic Algorithms*, Luniver Press, Frome, 2008.

23. A. M. Alattar, Reversible watermark using the difference expansion of a generalized integer transform, *IEEE Trans. Image Process.*, **13** (2004), 1147–1156. https://doi.org/10.1109/TIP.2004.828418

24. Y. Li, J. Wang, X. Luo, A reversible database watermarking method non-redundancy shifting-based histogram gaps, *Int. J. Distrib. Sens. Netw.*, **16** (2020). https://doi.org/10.1177/1550147720921769

25. A. Khan, S. A. Husain, A fragile zero watermarking scheme to detect and characterize malicious modifications in database relations, *Sci. World J.*, **2013** (2013). https://doi.org/10.1155/2013/796726

26. L. Camara, J. Li, R. Li, W. Xie, Distortion-free watermarking approach for relational database integrity checking, *Math. Probl. Eng.*, **2014** (2014). https://doi.org/10.1155/2014/697165

27. Y. Li, J. Wang, S. Ge, X. Luo, B. Wang, A reversible database watermarking method with low distortion, *Math. Biosci. Eng.*, **16** (2019), 4053–4068. https://doi.org/10.3934/mbe.2019200

28. S. Xiao, X. Zuo, Z. Zhang, F. Li, Large-capacity reversible image watermarking based on improved DE, *Math. Biosci. Eng.*, **19** (2022), 1108–1127. https://doi.org/10.3934/mbe.2022051

29. A. Alqassab, M. Alanezi, Relational database watermarking techniques: A survey, *J. Phys.: Conf. Ser.*, **1818** (2021), 012185. https://doi.org/10.1088/1742-6596/1818/1/012185

30. X. Shen, Y. Zhang, T. Wang, Y. Sun, Relational database watermarking for data tracing, in *2020 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, (2020), 224–231. https://doi.org/10.1109/CyberC49757.2020.00043

31. H. M. Sardroudi, S. Ibrahim, A new approach for relational database watermarking using image, in *5th International Conference on Computer Sciences and Convergence Information Technology*, (2010), 606–610. https://doi.org/10.1109/ICCIT.2010.5711126

32. A. Phadikar, S. P. Maity, B. Verma, Region based QIM digital watermarking scheme for image database in DCT domain, *Comput. Electr. Eng.*, **37** (2011), 339–355. https://doi.org/10.1016/j.compeleceng.2011.02.002

33. S. Melkundi, C. Chandankhede, A robust technique for relational database watermarking and verification, in *2015 International Conference on Communication, Information and Computing Technology (ICCICT)*, (2015), 1–7. https://doi.org/10.1109/ICCICT.2015.7045676

AIMS Press