Hindawi
Mathematical Problems in Engineering
Volume 2017, Article ID 1387375, 14 pages
https://doi.org/10.1155/2017/1387375

Research Article

A New Reversible Database Watermarking Approach with Firefly Optimization Algorithm

Mustafa Bilgehan Imamoglu, Mustafa Ulutas, and Guzin Ulutas

Department of Computer Engineering, Karadeniz Technical University, 61080 Trabzon, Turkey

Correspondence should be addressed to Mustafa Bilgehan Imamoglu; bilgehan@ktu.edu.tr

Received 5 September 2016; Revised 25 December 2016; Accepted 30 January 2017; Published 6 March 2017

Academic Editor: Lotfi Senhadji

Up-to-date information is crucial in many fields such as medicine, science, and stock market, where data should be distributed to clients from a centralized database. Shared databases are usually stored in data centers where they are distributed over insecure public access network, the Internet. Sharing may result in a number of problems such as unauthorized copies, alteration of data, and distribution to unauthorized people for reuse. Researchers proposed using watermarking to prevent problems and claim digital rights. Many methods are proposed recently to watermark databases to protect digital rights of owners. Particularly, optimization based watermarking techniques draw attention, which results in lower distortion and improved watermark capacity. Difference expansion watermarking (DEW) with Firefly Algorithm (FFA), a bioinspired optimization technique, is proposed to embed watermark into relational databases in this work. Best attribute values to yield lower distortion and increased watermark capacity are selected efficiently by the FFA. Experimental results indicate that FFA has reduced complexity and results in less distortion and improved watermark capacity compared to similar works reported in the literature.

1. Introduction

The rapid development in Information Technologies (IT) has simplified data sharing through the web and enabled collaboration among people, organizations, and governments worldwide. The collaborative environment necessitates either free or commercial sharing of information as in medicine, science, and stock quotes. Most of the time, centralized or distributed relational databases hold shared information where authorized users can access and use it by applications. One of the problems in sharing information is that information can be copied, altered, or modified easily by authorized users and may be distributed to unauthorized users for reuse. Therefore, ensuring copyright and preventing temper of shared data have become a serious problem.

Cryptography can be used to share confidential information without compromising security. However, it does not associate ciphered information with original content and cannot be used for copyright protection. Watermarking is proposed by [1] and provides solutions for authentication, fingerprinting, copy control, and copyright protection during distribution of digital data. Many watermarking techniques have been proposed in the literature to share images, video files, audio files, and relational databases without violating copyrights. Robustness, fidelity, and blindness properties of watermarking techniques have been used for copyright protection of relational databases recently.

In 2003, Kiernan et al. showed that watermarking could be used to provide copyright protection of relational databases [2]. Their method embeds watermark data at bit level on numeric data fields. A secure Message Authentication Code (MAC) is utilized by their work to select some tuples and attributes on the selected tuples to embed the watermark bits. LSB embedding technique is used by their work and tampers the relational database permanently. A pseudo random number generator is used to generate the watermark bits. When an attacker implements simple shifting operation on the LSB bits of numeric fields in the watermarked database, a significant amount of watermark will be lost. After this, many works in the field have been proposed by other researchers. In 2004 [3], Sion et al. proposed a method to insert a *virtual* watermark. Their method first sorts tuples in ascending order

by n most significant bits of normalized values and then puts markers at tuples where MAC mod m yield zero to create partitions. The watermark is then embedded by modifying tuple values close to standard deviation boundary which results in altered partition statistics. Their work has some extra payload information and is dependent on a single attribute.

Li et al. used categorical attribute to detect malicious alterations and used fragile watermarking approach [4]. The tuples in the database are grouped and watermarks are embedded into groups. Verification procedure is applied on group level. Guo et al. proposed a fragile watermarking scheme to detect malicious modifications on a database in 2006 [5]. The method divides the database into partitions according to the primary key value of tuples. However, two LSB bits are modified by their approach to embed the watermark information. In 2006, Zhang et al. proposed the first reversible watermarking scheme in their work [6]. Their method utilized histogram expansion approach to reversibly watermark the selected nonzero initial digits of errors. In 2007, Zhou et al. embedded a bitmap image into the relational database for copyright protection [7]. Their method also implements an error correction mechanism to correct the errors in the detected watermarks. Difference Expansion Based Watermarking (DEW) technique is used in [8] to watermark a database in a reversible manner. DEW was proposed by Alattar in 2004 and used average values with expanding or enlarging to embed the watermark bits [9]. The method described in [8] uses distortion tolerance to determine the distortion tolerance of the attribute. Bhattacharya and Cortesi created the watermark after partitioning tuples as a permutation of tuples in 2009 [10]. A hash function is used by their method for grouping purposes. The proposed method is distortion-free due to the ordering of tuples. Hanyurwimfura et al. used nonnumeric attributes that contain multiwords [11]. Their method uses Levenshtein distance during the embedding procedure. Location of a word is shifted horizontally according to the watermark bit. Tuples and attributes that are used for embedding procedure are chosen dynamically. Farfoura et al. [12] used reversible watermarking technique in their method. Reversible watermarking approach recovers the original data from the watermarked data after watermark extraction. Primary key dependent technique cannot resist against linear transformation attacks because when the watermarked tuple is deleted, watermark cannot be detected. Their work utilized Prediction Error Expansion watermarking technique proposed by [13]. In 2012, Arif et al. emphasized rewatermarking attack and used date stamp with watermark to overcome this problem [14]. Khan and Husain proposed a database watermarking technique based on fragile watermarking [15]. Their method is a kind of zero watermarking approach and utilizes characteristics of database relation. Numeric attributes are evaluated by the method and it is not resilient to attribute value substitution attack. Camara et al. suggested a fragile zero watermarking method to authenticate numerical relational data [16]. Their method is distortion-free and generates the watermark from the original database. The method partitions the database into groups and generates some mathematical values from each group. Results show that their method is robust to some modification attacks.

Jawad and Khan used genetic algorithm with difference expansion watermarking to propose a robust and reversible database watermarking approach called GADEW [17]. GADEW minimizes distortion induced by the embedding procedure and increases watermarking capacity. Their method used DEW technique to embed the watermark bits into the database with a reversible manner.

In this work, we proposed a new reversible database watermarking method using Firefly Algorithm to both minimize distortion and improve robustness of DEW called FFADEW. The method uses a bioinspired algorithm, firefly proposed by [18] to select the best attributes for watermark embedding. Each firefly in a population is a candidate solution for watermarking process and brightest firefly on the current population is the best solution for the current iteration. Other fireflies are moved to the best firefly with the proposed moving operation. Generation of the new populations is ended when the algorithm reaches the desired condition proposed by the algorithm. Brightest firefly in the last population guides the watermarking process. The algorithm contains floating-point values that construct a firefly designating the location of attributes to be watermarked in each tuple. Experimental results show that the method modifies attribute values in a less observable manner (standard deviation and average values of attributes are used) and the method is more robust against popular database watermarking attacks compared to GADEW. Complexity of the proposed method is also less than GADEW as shown in the results by the run times of both algorithms.

The rest of the article is organized as follows: Section 2 gives background information about DEW method and Firefly Algorithm used in FFADEW. The details of the method are explained in Section 3 and experimental results are given in Section 4. Section 5 concludes the article with suggestions.

2. Related Works

The details of the difference expansion watermarking technique and firefly optimization algorithm used in the proposed method are given in this section.

2.1. Difference Expansion Watermarking Technique. Alattar proposed a reversible watermarking approach called difference expansion watermarking (DEW) in 2004 to apply the images [9]. DEW algorithm gets a pair of adjacent pixels and modifies their difference values to show the watermark bit. The algorithm can reconstruct the original image after extracting the watermark. DEW is used to embed a watermark bit into two numeric attributes in the proposed method. The proposed method uses two numeric attributes denoted by A_1 and A_2 instead of pixel values. Average value of them and difference between them are calculated as in (1) and denoted by avg and d, respectively.

$$\text{avg} = \left\lfloor \frac{A_1 + A_2}{2} \right\rfloor, \qquad d = A_1 - A_2. \tag{1}$$

The symbol $\lfloor x \rfloor$ denotes the floor function which returns the greatest integer less than or equal to argument x. The difference value d is modified to carry the watermark bit b as in (2). The attribute values are also modified to give the new difference value $\tilde{d}$.

$$\begin{aligned} \tilde{d} &= 2d + b, \\ \widetilde{A_1} &= \text{avg} + \left\lfloor \frac{\tilde{d}+1}{2} \right\rfloor, \\ \widetilde{A_2} &= \text{avg} - \left\lfloor \frac{\tilde{d}}{2} \right\rfloor. \end{aligned} \tag{2}$$

$\widetilde{A_1}$ and $\widetilde{A_2}$ denote the modified values of the attributes. The method will use the following steps to recover the original values of attributes and corresponding watermark bit value.

Step 1. Compute the average of the attributes: $\acute{\text{avg}} = \lfloor (\widetilde{A_1} + \widetilde{A_2})/2 \rfloor$ and the difference $\tilde{d} = \widetilde{A_1} - \widetilde{A_2}$.

Step 2. Extract the watermark bit b from the difference: $b = \tilde{d} - 2\lfloor \tilde{d}/2 \rfloor$.

Step 3. Reconstruct the original values of A_1 and A_2: $A_1 = \acute{\text{avg}} + \lfloor (\lfloor \tilde{d}/2 \rfloor + 1)/2 \rfloor$; $A_2 = \acute{\text{avg}} - \lfloor (\lfloor \tilde{d}/2 \rfloor)/2 \rfloor$.

For example, let $A_1 = 16$ and $A_2 = 24$ be the contents of two numeric attributes in a database. The difference and average values are calculated as $d = 16 - 24 = -8$ and avg $= \lfloor (16 + 24)/2 \rfloor = 20$. Assume that current watermark bit is 1. New difference value will be $\tilde{d} = 2(-8) + 1 = -15$ and modified attribute values will be $\widetilde{A_1} = 20 + \lfloor (-15+1)/2 \rfloor = 13$ and $\widetilde{A_2} = 20 - \lfloor -15/2 \rfloor = 28$. Watermark extraction algorithm calculates new difference value and new average value as −15 and 20, respectively. Watermark bit b is calculated as $-15 - (2(-8)) = 1$ and original attribute values are reconstructed as in

$$\begin{aligned} A_1 &= 20 + \left\lfloor \frac{(\lfloor -15/2 \rfloor + 1)}{2} \right\rfloor = 16, \\ A_2 &= 20 - \left\lfloor \frac{(\lfloor -15/2 \rfloor)}{2} \right\rfloor = 24. \end{aligned} \tag{3}$$

2.2. Firefly Algorithm. Biologically inspired algorithms have become popular in solving global optimization problems such as the Travelling Salesman Problem (TSP) recently. Multiple agents that are affected by each other constitute the base of these algorithms. These algorithms sometimes are referred to as Swarm Intelligence (SI) based algorithms since they simulate the Swarm Intelligence characteristics of biological agents such as fish, birds, and ants. Particle swarm optimization proposed in 1995 used the swarming behavior of fish and birds [19]. Ant colony optimization and artificial bee colony optimization are some examples in this field [20, 21].

Yang proposed Firefly Algorithm, one of the nature inspired algorithms, to solve NP-hard problems in 2008 [18]. Their algorithm uses flashing patterns and behavior of fireflies. Firefly is one of the stochastic optimization algorithms in which a global solution is searched using partially randomized movements in order not to get stuck in one of many local solutions. Being a stochastic method, firefly cannot guarantee optimal solution in deterministic time but it will eventually converge to a solution in a reasonable amount of time. In fact, firefly is a metaheuristic method where a trade-off between randomization and local search is controlled by the parameters. Firefly's heuristic depends on the survival of the population whereas randomized movement avoids local optima. The search for the optimum solution continues unless improvements in the objective function are possible. The solution in the proposed method corresponds to the selection of both tuples and attributes that minimizes distortion even though a single objective function is optimized.

There are nearly two thousand firefly species and each has a unique pattern of flashes. Flashing characteristics of the fireflies can be summarized with three rules given below.

Rule 1. Each firefly can attract others regardless of its sex. This means that all fireflies are unisex.

Rule 2. Brightness of a firefly determines its attractiveness. If two fireflies are flashing, less bright one will move towards the brighter one. If none firefly is brighter than a particular one, it will move away randomly.

Rule 3. The objective function determines the brightness of a firefly.

Brightness can be defined in a similar way with fitness function in genetic algorithm (GA). The brightness is simply proportional to the value of objective function for a maximization problem.

The brightness I of a firefly for a particular location x could be chosen as $I(x) \propto f(x)$, where $f(x)$ is the objective function and the attractiveness β should be judged by other fireflies. Distance between the fireflies changes the attractiveness parameter. Light intensity decreases in relation to the distance from the source. The light intensity $I(r)$ can be given in (4) in the simplest form. Intensity is inversely proportional to the square of distance as shown in the formulation. I_s denotes the intensity at the source.

$$I(r) = \frac{I_s}{r^2}. \tag{4}$$

When the fixed light absorption coefficient γ for a medium is considered, the light intensity changes according to (5). I_0 represents the initial light intensity.

$$I = I_0 e^{-\gamma r^2}. \tag{5}$$

Light intensity determines the attractiveness of the firefly for other fireflies. β given in (6) defines the attractiveness of the firefly. β_0 in (3) is the attractiveness at the start position.

$$\beta = \beta_0 e^{-\gamma r^2}. \tag{6}$$

The method assumes that each firefly resides on a point at d dimensional space. Assume that coordinates of any two

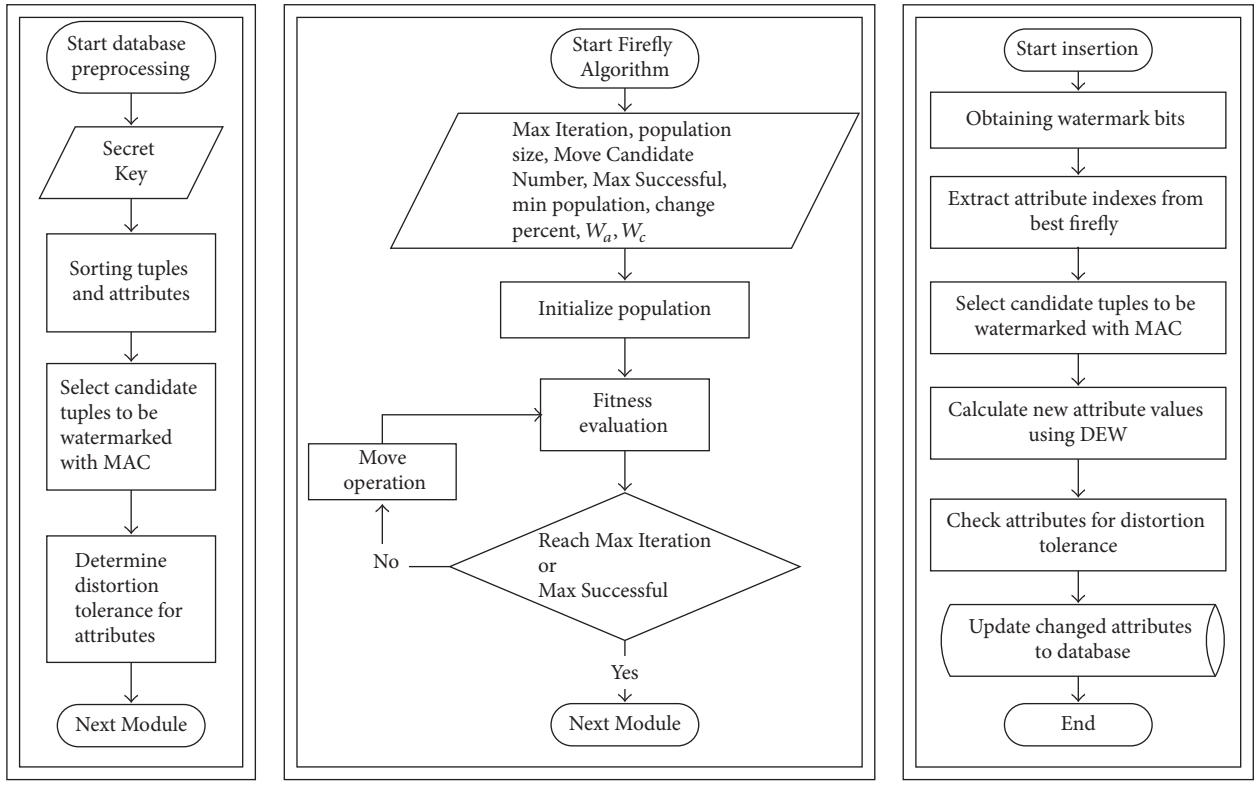

Figure 1: Watermark insertion algorithm: (a) preprocessing, (b) firefly determination, and (c) watermark insertion.

fireflies i and j are represented by x_i and x_j. The Cartesian distance r_{ij} between two fireflies is calculated as in

$$r_{ij} = \left|x_i - x_j\right| = \sqrt{\sum_{k=1}^{d} \left(x_{i,k} - x_{j,k}\right)^2}. \quad (7)$$

The movement of one firefly in d dimensional space at time $t + 1$ towards another attractive firefly is defined as in (8) at time t,

$$x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma} r_{ij}^2 \left(x_j^t - x_i^t\right) + \alpha\varepsilon_i. \quad (8)$$

Attractiveness is considered in the second term and random movement is ensured by the third term in (8). α is a randomization factor and ε_i is a vector of random numbers (rand − 1/2), where rand is a pseudo random number generator that generates uniformly distributed random numbers in [0 − 1]. The equation given in (8) shows that fireflies realize simple random walk for $\beta_0 = 0$.

3. Proposed Method

A new reversible database watermarking algorithm using Firefly Algorithm, a bioinspired optimization algorithm, is proposed here. The method uses DEW algorithm to reverse the original database after watermark extraction and it also uses optimization algorithm called firefly to determine the best candidate pairs to embed the watermark. The main advantage of the Firefly Algorithm compared to genetic algorithms is its easy implementation and its run time efficiency. In this regard, we used Firefly Algorithm in this work.

The method consists of two algorithms: watermark insertion and watermark extraction. Watermark insertion algorithm embeds the watermark information into specially selected tuples with DEW. Firefly Algorithm determines which attributes are more appropriate for watermark embedding on the selected tuples. Watermark extraction algorithm extracts the specially embedded watermark information from the database and compares it with the original one. This algorithm also reverses the watermarked database to original after watermark extraction and verification. The details of the two algorithms are given in the following sections.

3.1. Watermarking Algorithm. Watermarking algorithm consists of three modules as shown in Figure 1: preprocessing, firefly determination, and watermark embedding. Preprocessing module prepares the database by sorting tuples and columns. Firefly determination module outputs the best firefly for dataset. Watermark embedding module embeds watermark data into dataset according to the best firefly.

3.1.1. Preprocessing Module. This module prepares the database for watermarking process as the first step. The tuples of the database are sorted according to the primary key and then the columns of the database are sorted according to the names of attributes. Sorting operation ensures algorithm's robustness

against reshuffling of attributes/tuples attacks. Second step of the preprocessing stage is selection of candidate tuples from the database using a Message Authentication Code (MAC) function. These selected tuples that are evaluated during watermark insertion create a subset, DS. The equation given in (9) is used for selection purposes where S denotes the Secret Key, P_k shows the primary key of the current tuple, $\|$ represents concatenation operation and function, and H is 512-bit version of Secure Hash Algorithm (SHA). The tuple is selected as a member of subset DS if it satisfies (9). The value of γ determines the number of selected tuples.

$$\mod\left(H\left(S \parallel H\left(P_k \parallel S\right)\right), \gamma\right) \equiv 0. \tag{9}$$

Assume that selected subset DS has M rows and A attributes. A distortion tolerance range for each attribute in current database should also be determined during watermark insertion algorithm. FFADEW selects the min and the max values of each attribute as corresponding distortion tolerance ranges if no distortion tolerance is specified. Assume the defined ranges in a database D with attributes A denoted by $(DT_x^{\min}, DT_x^{\max})$, $x = 1, \ldots, A$. For example, values in the xth attribute of the database can get values in $[DT_x^{\min}, \ldots, DT_x^{\max}]$ range.

3.1.2. Firefly Determination Module. Firefly Algorithm is utilized in this step to choose the best attribute pairs for the tuples in the selected subset DS. This module gets the subset from the preceding module as input and outputs the chosen firefly. Determined solution (firefly) designates the locations (attributes positions for the selected rows) to embed the watermark. Firefly determination module consists of two phases and the details of them are given in below.

Phase 1 (creation of the initial population). The algorithm creates initial population with P fireflies, $F^1, \ldots, F^P$. A firefly that constitutes an example solution for the current selected database DS can be represented by a floating-point array. Each element of the array represents the index values of the attributes for each row that is chosen for embedding the watermark information. While integer part of the element represents the index value for the first attribute, fractional part denotes the index value of second attribute at the current row. For example, assume that a database accommodates ten attributes and one firefly is determined as the solution during the optimization algorithm. If the first element of the firefly is 7.2, the algorithm uses seventh and second attributes of the first row that is chosen for watermark embedding.

jth candidate solution for the current database contains M floating-point numbers and is denoted by $F^j = \{F_i^j \mid F_i^j = x \cdot y,\ x \in [1, \ldots, A],\ y \in [1, \ldots, A],\ x \neq y,\ i \in [1, \ldots, M]\}$. The maximum value for an element in the current firefly can be $A \cdot (A-1)$ and it shows that A and $A-1$th attributes are chosen for the current row for watermark embedding. Steps for the creation of initial population are given below.

Function. Initial_Population

Input. DS

Output. Initial population

Step 1. Repeat Step 2 for $j \in [1, \ldots, P]$.

Step 2. Repeat the steps from 2.1 to 2.4 for $i \in [1, \ldots, M]$.

Step 2.1. Select an attribute randomly At_x from the kth row of DS.

Step 2.2. Apply DEW algorithm on At_x and the other attributes At_y, $y \in [1 \ldots A]$, $x \neq y$, at the kth row. $(M\_At_x, M\_At_y) = ApplyDEW(At_x, At_y, w)$.

Step 2.3. Fitness function selects best pair (At_x, At_y) whose elements are changed less compared to other pairs after embedding the watermark bit with DEW. Selected pair gives minimum value for $|M\_At_x - At_x| + |M\_At_y - At_y|$.

Step 2.4. Index values of these attributes constitute the corresponding kth element of the initial population, $F_i^j = (x \cdot y)$ where x denotes the integer part and y denotes the fractional part. The information extracted from the kth row determines the kth element of the current firefly F^j.

Used fitness function for generating the initial population aims to minimize distortion after embedding watermark. Therefore, it selects best pair for each row that results in minimum modification after DEW. Fitness function aims to improve two criteria: minimization of distortion on the selected attributes caused by DEW and improving the watermark capacity. Therefore, FFADEW selects the best pair for each row that gives minimum absolute difference after embedding watermark during the generation of the initial population. Figure 2 shows an example for the creation of initial population.

Figure 2(a) gives the ith row of DS. The algorithm randomly selects an attribute, here the seventh in this example. Figure 2(b) shows all possible pairs that can be created with (At_7) and the result of DEW algorithm on all possible pairs. The sixth pair (At_7, At_6) yields minimum difference if total cost (minimum absolute difference) is considered for each pair. Thus, the algorithm chooses (At_7, At_6) and embeds the value 7.6 into ith element of current firefly.

Phase 2 (determination of the best firefly). In this step, FFADEW determines brightness value for each firefly in initial population that was created in the previous phase and chooses the brightest one among them. The other fireflies are moved towards to the best one and the best one is also moved randomly after other fireflies were moved. Phase 2 of the firefly determination module has two terminating conditions: Step 1 or Step 2.3 in the *Brightness Value Determination* algorithm. First condition controls the number of iteration. The algorithm terminates and returns the brightest firefly if the algorithm exceeds $\max_{\text{iteration}}$ iteration. Second condition controls the number of watermarked tuple and distortion

PK	At_1	At_2	At_3	At_4	At_5	At_6	At_7	At_8	At_9
	34	21	67	18	445	213	188	104	531

PK	At_1	At_2	At_3	At_4	At_5	At_6	At_7	At_8	At_9
	34	21	67	18	445	213	188	104	531

(a) Select first value for DEW randomly

At_7	At_1
188	34

At_7	At_2
188	21

At_7	At_3
188	67

At_7	At_4
188	18

At_7	At_5
188	445

At_7	At_6
188	213

At_7	At_8
188	104

At_7	At_9
188	531

(b) Apply DEW to the selected attribute and other attributes, respectively

M_At_7	M_At_1
265	−43

M_At_7	M_At_2
271	−63

M_At_7	M_At_3
248	6

M_At_7	M_At_4
273	−67

M_At_7	M_At_5
59	573

M_At_7	M_At_6
175	225

M_At_7	M_At_8
230	62

M_At_7	M_At_9
16	702

New values of attributes after DEW

M_At_7	M_At_1
154	

M_At_7	M_At_2
167	

M_At_7	M_At_3
121	

M_At_7	M_At_4
170	

M_At_7	M_At_5
257	

M_At_7	M_At_6
25	

M_At_7	M_At_8
84	

M_At_7	M_At_9
343	

Calculate total distortions on the attribute pair after DEW and select the tuple that gives minimum distortion

Figure 2: An example creation of initial population.

on the watermarked tuple. The algorithm terminates if the brightness value for the current firefly is greater than a predefined threshold value t_{end}, and the current firefly is saved.

This phase consists of two algorithms: Brightness Value Determination and firefly moving algorithms. The first one determines brightness values for each firefly and run the other algorithm to determine the new coordinates of fireflies. The details of the algorithms are given below.

The brightness value Br^i for ith firefly, F^i, is determined with following algorithm. The algorithm evaluates subdataset DS with M rows.

Function. Brightness_Value_Determination

Input. DS and initial population

Output. F^{best}

Step 1. Repeat the steps from 2 to 3 for $k = 1, \ldots, \max_{\text{iteration}}$.

Step 2. Repeat the steps from 2.1 to 2.3 for $i = 1, \ldots, P$.

Step 2.1. Repeat the following steps from 2.1.1 to 2.1.5 for $j = 1, \ldots, M$.

Step 2.1.1. Extract jth row of DS, R^j.

Step 2.1.2. $x = IntegerPart(F_j^i)$; $y = FractionalPart(F_j^i)$.

Step 2.1.3. $At_x = R_x^j$, $At_y = R_y^j$.

Step 2.1.4. $(M\_At_x, M\_At_y) = ApplyDEW(At_x, At_y, w)$.

Step 2.1.5

$$\begin{aligned} &If \ \left(DT_x^{\min} \le M\_At_x \le DT_x^{\max}\right) \\ &\quad \wedge \left(DT_y^{\min} \le M\_At_y \le DT_y^{\max}\right) \\ &\text{mdf}_x = \text{mdf}_x + \left(M_{Atx} - At_x\right) \\ &\text{mdf}_y = \text{mdf}_y + \left(M\_At_y - At_y\right) \\ &\text{row}_w = \text{row}_w + 1. \end{aligned} \tag{10}$$

Step 2.2. $Br^i = ((\sum_{t=1}^{A} |\text{mdf}_t|)/\text{tuple}_{\text{count}})w_a + (\text{row}_w/M)w_c$.

Step 2.3. If $Br^i \ge t_{\text{end}}$ save the current firefly F^i into F^{best} and go to Step 4.

Step 3. Implement *moving operation*.

Step 4. Export F^{best} to the following phase (watermark insertion).

$IntegerPart(F_j^i)$ and $FractionalPart(F_j^i)$ functions get the jth element of ith firefly and return the integer part and fractional part of the element. These values show the index values of attributes that are chosen for watermark embedding during population generation. The function given in Step 2.1.4, $ApplyDEW(At_x, At_y, w)$, applies DEW algorithm on two selected attributes At_x, At_y and embeds one bit watermark. The function returns the new values of At_x, At_y after watermark embedding (M_At_x, M_At_y). Distortions are calculated if modified attribute values fall into the desired ranges. Distortions on the attributes induced from the watermarking process are denoted by mdf_x, $x = 1, \ldots, A$, and they are accumulated independently as shown in Step 2.1.5. The variable row_w is used to hold information about how tuples in the selected subset can be watermarked. Current element of firefly is not considered in the calculation of brightness value if modified attribute values do not fall into the desired ranges, and the algorithm returns to Step 2.1 to consider the next element of the current firefly.

The algorithm assigns a brightness value for each firefly. The brightness values are affected from two factors: total distortions on the attributes and the number of tuples that can be watermarked. First factor is the ratio between the sum of the distortions on the attributes mdf_x, $x = 1, \ldots, A$, and total record count in the database, $\text{tuple}_{\text{count}}$. Second factor indicates how many records in the selected subset can be watermarked. The two factors are weighted with w_a and w_c to determine the brightness value. Figure 3 shows an example for calculation of $((\sum_{t=1}^{A} |\text{mdf}_t|)/\text{tuple}_{\text{count}})$ on an example

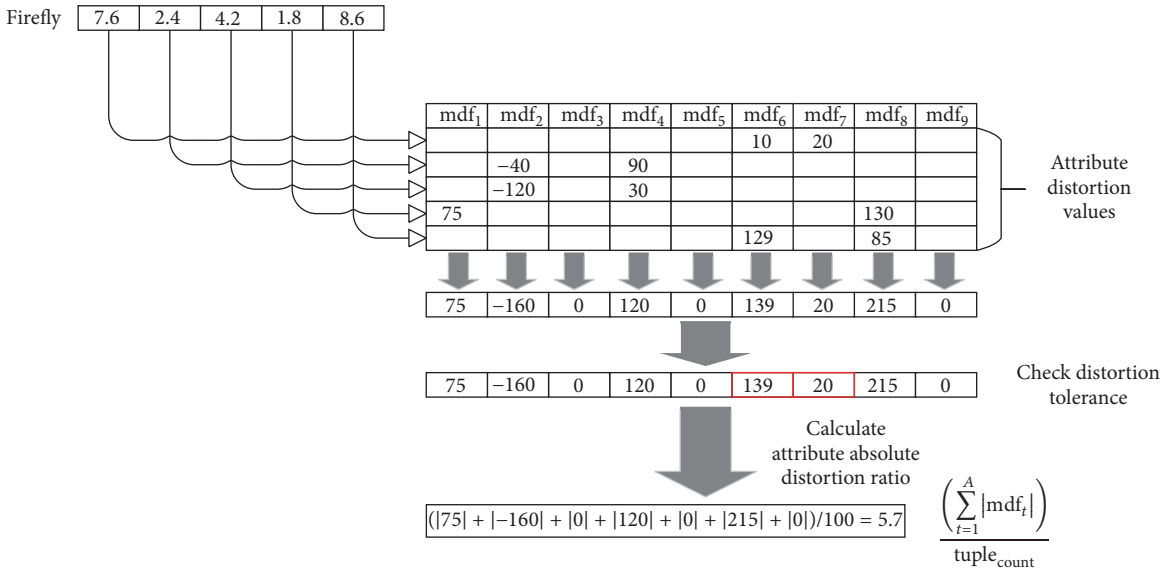

Figure 3: An example calculation first part of the brightness function.

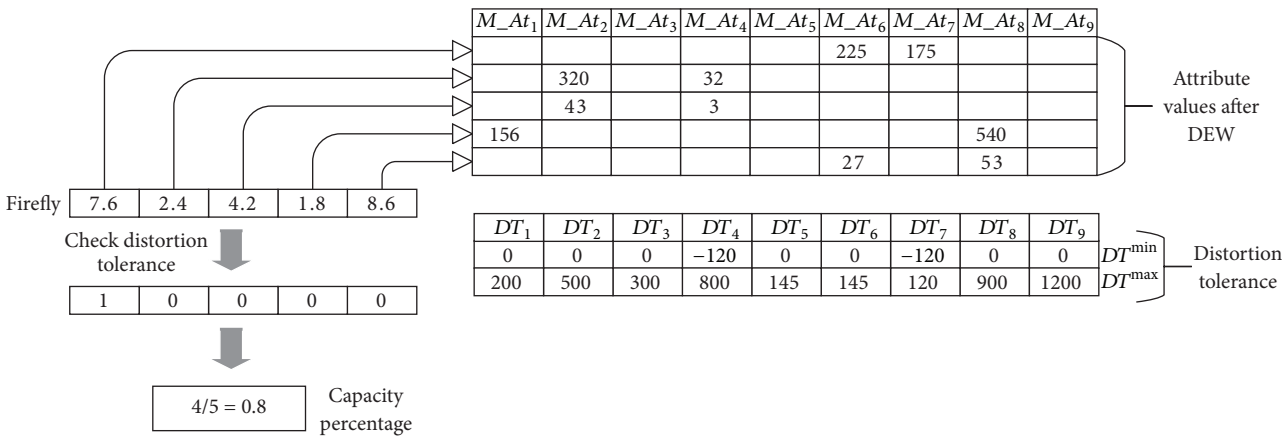

Figure 4: An example calculation second part of the brightness function.

dataset. First row of this dataset is the calculated one in Figure 2. This example assumes that firefly consists of only five elements. First part of the brightness function is calculated as given in Figure 3. mdf values and $(|\mathrm{mdf}_t|)/\mathrm{tuple}_{\mathrm{count}}$ ratios are also reported in the figure. Figure 4 also shows the calculation of the other parts of brightness function (row_w/m). Four out of five elements of the current firefly can be watermarked because watermarked values fall into the desired range. Only first row is not watermarked.

The equation in Step 2.2 that corresponds to the brightness of ith firefly in the population is the objective function and also used to compute β given in (6). Weights W_a and W_c in the objective function correspond to normalized distortion and amount of embedded watermark, respectively. These weights are selected such that their sum is one. Higher values of W_a indicate increased importance of normalized distortion compared to the amount of embedded watermark whereas higher values of W_c indicate increased importance of the amount of embedded watermark compared to normalized distortion in the objective function. Embedded watermark ratio and distortion are calculated for W_c values in $[0.1, \ldots, 0.9]$ and reported in Figure 18. Tests indicated that $W_a = 0.4$ and $W_c = 0.6$ yield highest embedded watermark and lowest distortion for the dataset. Because of this W_a and W_c are selected as 0.4 and 0.6, respectively.

The distance between two fireflies in the original Firefly Algorithm is computed by the Cartesian distance given in (7) whereas the proposed method uses total number of different corresponding element as the distance between two vectors. The fireflies in the current population are sorted according to their brightness values and the brightest firefly is chosen as the best firefly. Other fireflies in the current population are moved towards to the best firefly. Assume that F^{best} denotes the brightest firefly. The element-by-element difference between a firefly and the best firefly is used to guide the moving process. The main purpose of the moving

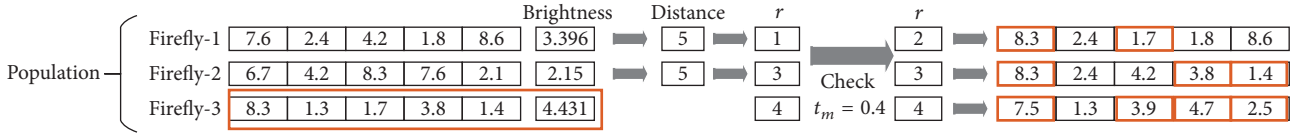

Figure 5: An example moving operation.

process is to minimize the difference between two vectors that corresponds to two fireflies. The distance between the ith firefly F^i and the best firefly F^{best} is calculated using (11). Since each firefly has M elements, distance $D(F^i, F^{\text{best}})$ is in $[0, \ldots, M]$ range.

$$D\left(F^i, F^{\text{best}}\right) = \sum_{k=1}^{M} \left(F_k^i \neq F_k^{\text{best}}\right). \tag{11}$$

The movement of fireflies given in (8) is modified such that fireflies in the population improve their similarity to the best firefly by copying a randomly selected number of best firefly's component. Movement equation given in (8) has 2 parts as information based and random movement. Both of these movements are responsible for finding an optimal solution in the search space without getting stuck at local extremes. The information based movement relies on the positions and brightness of the population whereas random movement acts like mutation in the genetic algorithms where there is no guarantee that the movement is in the correct direction. Carefully selected ε_i parameter is important in order to achieve acceptable results. The value of parameters depends on the problem and may require some trial and error before successfully generating an optimal solution.

The algorithm moves fireflies to the firefly corresponding to the best solution, by generating a random number r and replaces randomly chosen r elements of F^i with corresponding elements of F^{best}. The random number $r \in [1, \ldots, D(F^i, F^{\text{best}})]$ and it must also satisfy $r/M \geq t_m$, where t_m denotes the amount of modification and can get values in $[0, \ldots, 1]$ range. All elements of F^i are replaced with corresponding elements of F^{best} if t_m is one and no elements of F^i are replaced with corresponding elements of F^{best} if t_m is zero.

Randomness of moving operation given in (8) is used to determine the number of elements to be changed in the current solution vector (firefly) with the best solution vector (firefly) to make it similar (moving current firefly closer to the best firefly) in our method. The criterion $r/M \geq t_m$ given in Step 3.2 which is specific to our algorithm is used in the selection of the randomization parameter and $[1, \ldots, D(Ftemp^j, F^{\text{best}})]$ range is used to generate information based random values.

The following source code realizes moving operation for all fireflies except from the best.

Function. Moving_Operation

Input. DS, F^{best}

Output. New F^{best}

Step 1. Repeat the steps from 2 to 4 for $i = 1, \ldots, P,\ i \neq \text{best}$.

Step 2. Calculate the distance between the current firefly and the best one (F^i, F^{best}).

Step 3. Repeat the steps from 3.1 to 3.4 for $j = 1, \ldots, c_{\text{move}}$.

Step 3.1. Store the current firefly into temporary firefly $Ftemp^j$.

Step 3.2. Choose a random number $r \in [1, \ldots, D(Ftemp^j, F^{\text{best}})]$ and it must also satisfy the rule, $r/M \geq t_m$. If the rule is not satisfied choose another random number r.

Step 3.3. Replace randomly selected r elements of $Ftemp^j$ with corresponding elements of F^{best}.

Step 3.4. Calculate the brightness value for the current $Ftemp^j$ and store it into an array TB^j.

Step 4. Choose the brightest firefly using TB from $Ftemp$ and replace F^i with it.

Figure 5 shows an example of moving operation for $t_m = 0.4$, or 40% replacement. Three fireflies from the population are shown in the figure. Third and best firefly is the brightest one and its brightness value is 4.431. The distances between the other fireflies and the brightest one are calculated as 5 and 5. Random numbers (1 and 3) are chosen for two fireflies. However, the rule given in Step 3.2 is not satisfied for the first firefly, $1/5 \not\geq 0.4$, and a new random number 2 is chosen. Two and three elements' contents of two fireflies are replaced by corresponding elements of the best firefly and thus moving process is realized.

The best firefly is also moved after moving all other fireflies. Only Step 3.3 is different from the above code to determine the new coordinates of the best firefly. Randomly selected r elements of current firefly are modified randomly as given below.

Step 3.3 integer and fractional parts of each randomly selected r elements of the best firefly are modified randomly and stored in $Ftemp^j$. Randomly selected values are floating-point numbers in $x \cdot y$ form and both x and y must satisfy $x \in [1, \ldots, A-1],\ y \in [1, \ldots, A-1],\ x \neq y$ conditions.

3.1.3. Watermark Embedding Module. Each element of the brightest firefly F^{best} of the firefly determination module is consulted and watermark embedding process is realized. Elements of the firefly designate the attribute locations of the

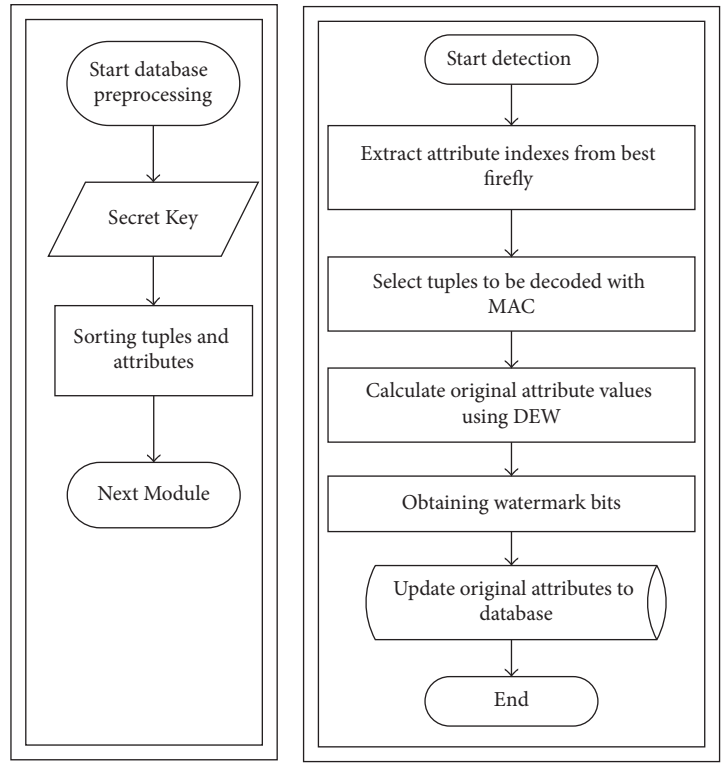

Figure 6: General structure of the algorithm.

current row in the selected dataset. Attributes are modified to embed watermark bit using DEW. The following code inserts the watermark information into corresponding attributes in the selected dataset.

Function. Watermark_Embedding

Input. DS, $W = w_1, \ldots, w_n$, F^{best}

Output. Watermarked DS, DS'

Step 1. Repeat the following steps for $j = 1, \ldots, M$.

Step 1.1. Extract jth row of DS, R^j.

Step 1.2. $x = IntegerPart(F_j^{\text{best}})$; $y = FractionalPart(F_j^{\text{best}})$.

Step 1.3. $At_x = R_x^j$, $At_y = R_y^j$.

Step 1.4. $(At_x, At_y) = ApplyDEW(At_x, At_y, w_i)$.

Step 1.5. $i = i + 1$.

3.2. Watermark Extraction Algorithm. This algorithm extracts watermark information from the database and reconstructs the original database using DEW. Figure 6 shows general structure of the algorithm.

Rows of the database are sorted according to the primary keys and columns are sorted according to the names of the attributes. After sorting operation, selected subdataset DS that was used during watermark embedding procedure must also be constructed in this algorithm to extract watermark correctly. The equation given in (9) is applied on the database with same secret value S to determine DS. The secret value S and the best firefly F^{best} must be transmitted to watermark extraction algorithm as the side information.

Since best firefly F^{best} has M elements, DS also has M rows. Each element of F^{best} points to attributes of current row that are used for watermark embedding. DEW is applied on designated attributes by F^{best} and watermark information is extracted. DEW algorithm also reconstructs the original values of attributes. The source code for watermark extraction algorithm is given below.

Function $ExtracWithDEW(At_x{}', At_y{}')$ returns the extracted watermark information and original values of $(At_x{}', At_y{}')$.

Function. Watermark_Extracttion

Input. DS', F^{best}

Output. Original DS and reconstructed watermark W'

Step 1. Repeat the following steps for $j = 1, \ldots, M$.

Step 1.1. Extract jth row of DS, R^j.

Step 1.2. $x = IntegerPart(F_j^{\text{best}})$; $y = FractionalPart(F_j^{\text{best}})$.

Step 1.3. $At_x{}' = R_x^j$, $At_y{}' = R_y^j$.

Step 1.4. $(At_x, At_y, w'_i) = ExtracWithDEW(At_x{}', At_y{}')$.

Step 1.5. $i = i + 1$.

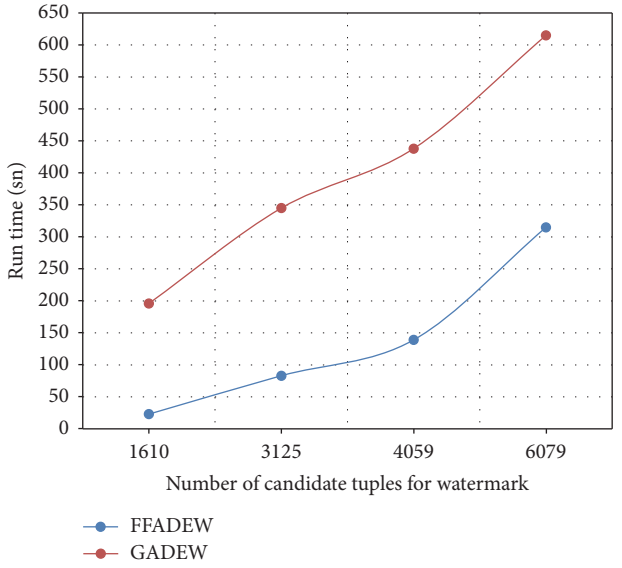

Figure 7: Average running times for FFADEW and GADEW with different number of tuples.

If the extracted watermark W' is consistent with the current watermark, this means that the database is authentic and no modification is applied on it.

4. Results and Discussion

Simulation results of the proposed method are given in this section. The method is coded in C# using MicroSoft Visual Studio IDE. MicroSoft SQL Server 2012 is used for modification operations on the database. Forest Cover Type (FCT) dataset provided by University of California is used as the database to watermark by the method. The dataset contains 581.012 tuples and consists of 54 numeric attributes. However, nine attributes are used during the experiments. Experiments are given in three sections. First section compares the method with similar works reported in the literature [9, 17]. Second section shows the performance of the method under some well-known attacks used to test the effectiveness of watermarking. The last section analyses effects of some of the parameters on the performance of the algorithm.

4.1. Performance Analysis. In this section, three criteria are used to compare the method with GADEW [17]: running time, watermarking capacity, and distortion effect. The details of the experiments are given below.

The first experiment compares the run time efficiency of the method with GADEW [17]. The number of tuples in the selected dataset *DS* is determined to be 1610, 3125, 4059, and 6079, respectively, and two algorithms are used to watermark *DS* ten times. Average running times for two methods with different number of tuples are given in Figure 7. The result shows that FFADEW algorithm runs much faster than GADEW for the same number of tuples. For example, FFADEW determines the best firefly in 23 s whereas GADEW determines the best solution in 196 s for a *DS* with 1610 tuples. The figure also shows that the method gives better results even if *DS* contains more tuples.

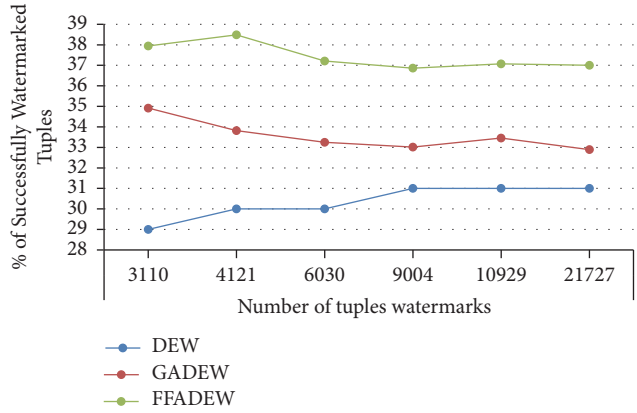

Figure 8: Watermarking capacity performance of DEW, GADEW, and FFADEW methods.

Second experiment tests watermarking capacity of methods for comparison. Watermarking capacity is the ratio of tuples to total number of tuples in the *DS* that can be used for watermarking. Higher watermarking capacity means more watermark information can be embedded into the same *DS*. Six different *DS* are created with 3110, 4121, 6030, 9004, 10929, and 21727 tuples to test the watermarking capacity of the proposed method and others. Figure 8 shows the watermarking capacity performance of DEW, GADEW, and FFADEW methods. For example, when the number of tuples in *DS* becomes 21727, DEW and GADEW can use 31% and 32% of *DS* for watermark embedding. However, FFADEW watermarks 37% of *DS*. The figure also shows that the method has better watermarking capacity compared to others for larger *DS*. The method has approximately 37% watermarking capacity for different *DS* sizes.

Third experiment analyses distortion effect of the proposed watermarking approach. The attacker aims to determine the distortion on the dataset, an indication of an embedded watermark. Distortion on the attributes is a clue about the locations of the watermark. A potential attacker needs to estimate the locations of the watermark first and then try to destroy it. Thus two metrics are important to evaluate the distortion effect of the method: alteration on the standard deviation and average of each attribute in the selected dataset. Figure 9 gives the amount of alteration on the standard deviation of attributes. In this test, *DS* has 3110 tuples. GADEW alters 24% and 31% of the standard deviation of $A3$ and $A8$ attributes, respectively, whereas FFADEW alters only 0.0742% and 0.0168% of the standard deviation for the same attributes. Likewise, Figure 10 shows alteration on the average values of attributes. Average values of $A3$ and $A5$ are altered 1.1% and 0.3%, respectively, for DEW or GADEW whereas FFADEW altered only 0.01% and 0.0025% of the average values of the same attributes.

4.2. Watermarking Attacks. The robustness of the method under well-known database watermarking attack scenarios is reported in this section. The main purposes of the attacks are to destroy the watermark information or to embed their own watermark information. The main attack types used and

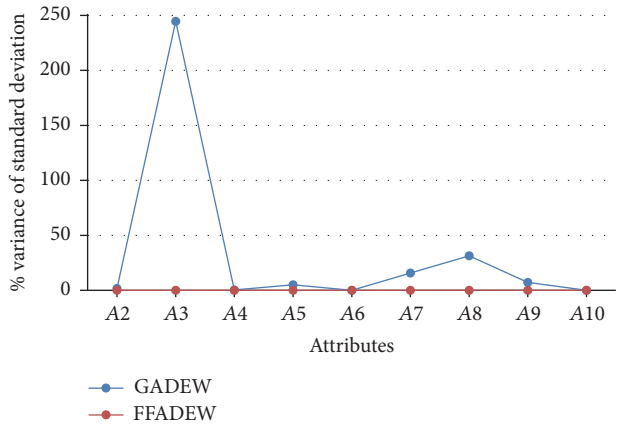

Figure 9: Alteration on the standard deviation of each attribute in the FCT dataset.

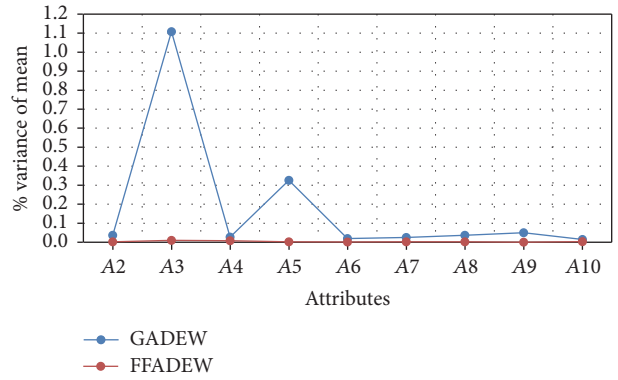

Figure 10: Alteration on the average of each attribute in the FCT dataset.

reported in the literature are considered and comparative results are given. A total of 1172 tuples from the selected *DS* are watermarked to make a fair comparison with GADEW reported in [17]. Successfully extracted watermarked tuple ratios reported are the average of 10 runs of FFADEW since both methods rely on stochastic optimization. The proposed method yields better detection rate on the same dataset with approximately equal number of watermarked tuples.

The first experiment is used to evaluate the robustness of the method when addition attack is applied on the watermarked database. Randomly created tuples are added to the database to destroy the watermark in this type of attack. Figure 11 shows that FFADEW is robust to watermark addition attack.

Even if half of the watermarked number of tuples are added to the database, the watermark is extracted with a true manner. Deletion attack is applied as the second experiment to show the performance of FFADEW. This type of attack deletes randomly selected tuples to destroy the watermark. Figure 11 shows that, as the number of deleted tuples increases, ratio of the extracted watermark decreases. Watermark cannot be recovered if a tuple that contains watermark information is deleted. Thus, deletion attack has the worst effect on the number of watermarked tuples as

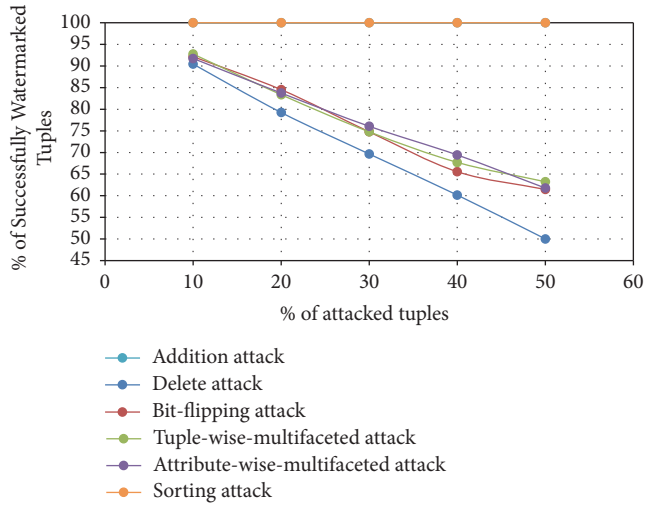

Figure 11: Comparison of FFADEW addition, delete, bit-flipping, tuple-wise-multifaceted, attribute-wise-multifaceted, and sorting attack.

shown in the figure. Bit-flipping attack is applied on the database as the third test. Bit-flipping attack complements LSBs of all attributes in randomly selected tuples. FFADEW yields similar results with deletion attack for bit-flipping attack. Since this attack modifies the content of the attributes it may also destroy watermark information in a tuple if it has watermark bit embedded in it. Figure 11 shows that the ratio of successfully extracted watermark information decreases considerably as the destroyed number of tuples increases. However, we only modify half of the attributes to discriminate the deletion attack from the bit-flipping attack visually. The result of the sorting attack is also shown in Figure 11. This type of attack rearranges the attributes in database. The method is robust against the sorting attack because the proposed method sorts the attributes alphabetically before embedding watermark and extraction algorithms. Watermark extraction ratio does not change even if an attacker modifies the original sequence of attributes. The last two attack types are tuple-wise multifaceted attack and attribute-wise multifaceted attack. The first one modifies the original database by applying three different attacks sequentially: addition, deletion, and bit-flipping attacks. Figure 11 shows the performance of the method under this attack and the method is also compared with methods in [9, 17] to show the effectiveness of it as shown in Figure 12.

DEW based method yields a detection ratio above 100% because addition attack causes false positives. GADEW can extract only half of the watermark information if half of the tuples in the database are affected by the attack. The proposed method detects 63.24% of the watermark for the same test. The last attack called attribute-wise multifaceted attack applies two different types of attacks to modify the original database, bit-flipping, and update attacks. Update attack modifies the content of an attribute with another attribute's content. Figure 13 shows the comparison of the methods. The proposed method extracts 63.24% of the watermark for 50% attack whereas GADEW extracts only

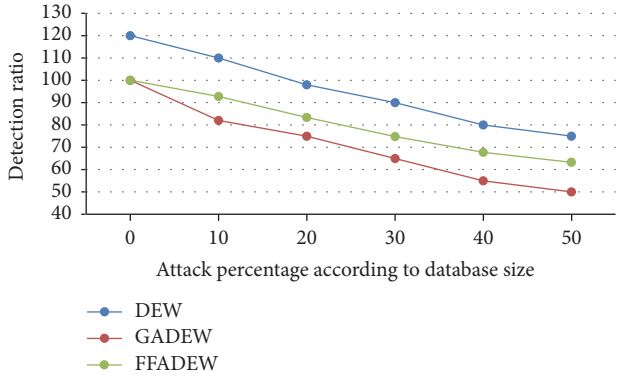

Figure 12: Comparison of DEW, GADEW, and FFADEW after tuple-wise-multifaceted attacks.

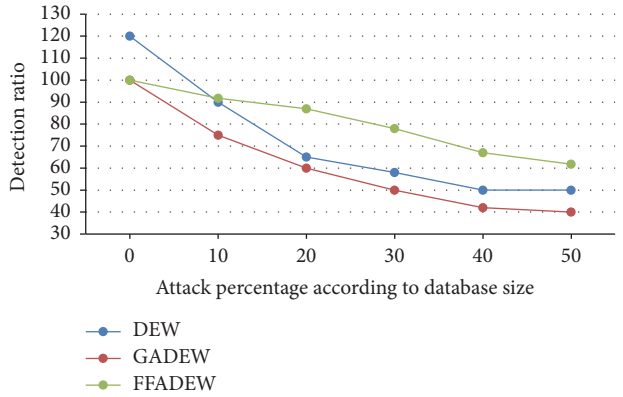

Figure 13: Comparison of DEW, GADEW, and FFADEW after attribute-wise-multifaceted attacks.

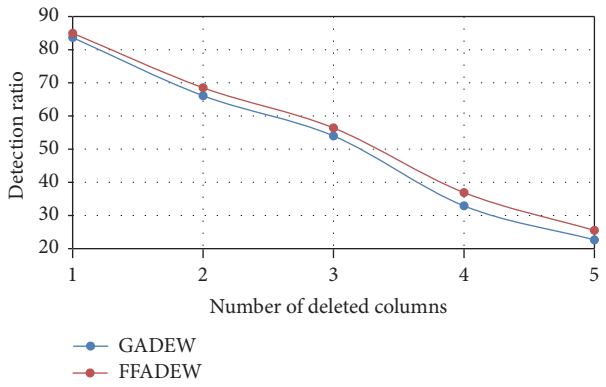

Figure 14: Comparison of FFADEW performance for attribute deletion attack.

40% of the original watermark for this test. Figure 13 also indicates that the proposed method yields better detection for other attack ratios.

Attribute deletion attack removes attributes from database to destroy the watermark. Figure 14 shows that the ratio of the extracted watermark decreases as the number of deleted attributes increased. Watermark cannot be recovered if an attribute that contains watermark information is deleted.

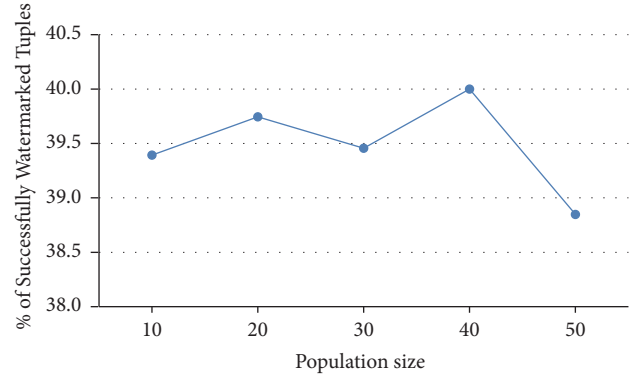

Figure 15: Comparison of FFADEW performance for different size of population.

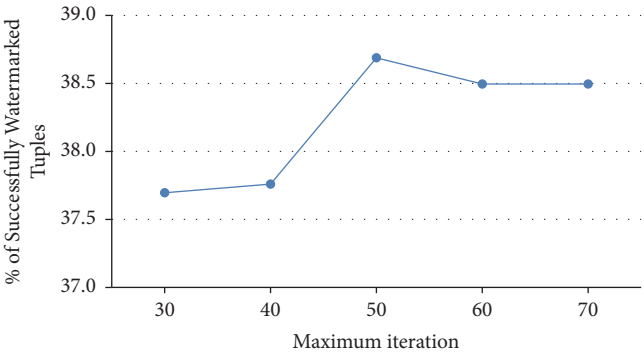

Figure 16: Comparison of FFADEW performance for different $\max_{\text{iteration}}$ values.

4.3. Algorithm Parameter Analysis. The effect of parameters on the performance of the proposed method is evaluated in this section. The effects of population size P, maximum number of iteration $\max_{\text{iteration}}$, and number of movement operation c_{move} are evaluated during the tests.

The first experiment tests the proposed method for 10, 20, 30, 40, and 50 fireflies as the population size P. Figure 15 shows the performance of the proposed method for P values. The y-axis of the graph shows the ratio of watermarked number of tuples to the total number of tuples in DS. The graph shows that the ratio changes in 38.8% and 40%. The result indicates that FFADEW yields high watermarking ratio even for small populations. Execution time of algorithm also increases as the population size increases. Thus, it is important to seek for the same performance with a smaller population.

The effect of maximum number of iteration $\max_{\text{iteration}}$ is evaluated in the second experiment. The method is allowed to iterate 30, 40, 50, 60, and 70 times during the test. Figure 16 indicates that the method has watermarking ratio between 37.7% and 38.7% in five different runs. The algorithm has a high watermarking ratio even for less number of iterations. It means that FFADEW has almost the same performance with better execution times.

Figure 17 shows the effects of the number of candidate move operations c_{move} on the watermarking ratio. Movement

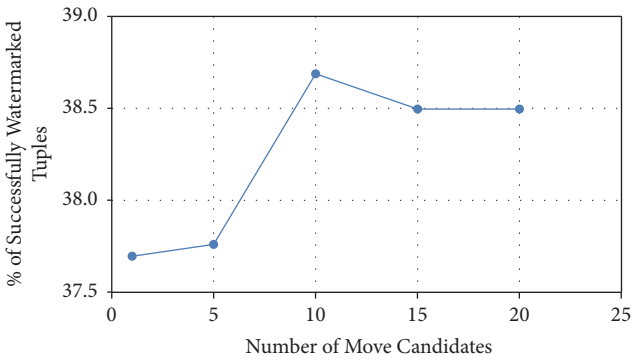

FIGURE 17: Comparison of FFADEW performance for different c_{move} values.

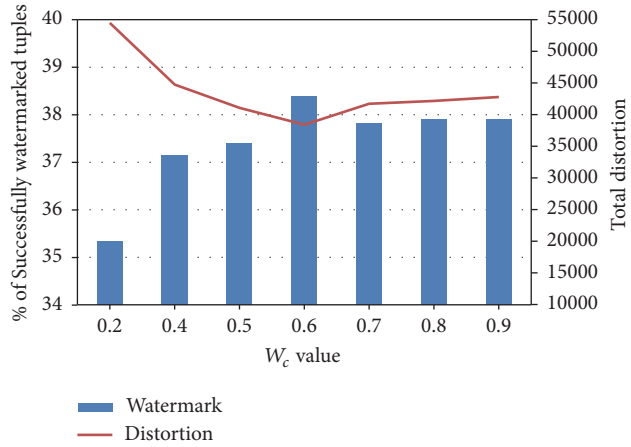

FIGURE 18: Comparison of FFADEW performance for different W_c values.

operation is repeated for c_{move} times, and the algorithm selects the best movement operation between them. Watermarking ratio also increases as c_{move} increase. However, watermarking ratio does not change considerably as shown in Figure 17 for c_{move} values greater than 15. It also shows that FFADEW converges the best solution faster with less number of iterations.

Total distortion and successfully embedded watermark ratios as a function of weights W_a and W_c used in the objective function are given in Figure 18. Highest watermark ratio and lowest distortion values are observed for $W_c = 0.6$ and $W_a = 0.4$ ($W_a = 1 - W_c$) during tests for the same dataset.

5. Conclusion

Database watermarking has become an active research as the demand for sharing information increases. Many reversible watermarking techniques have been proposed to watermark the database and to retrieve the original version of the database after watermark extraction. Researchers proposed using GA to minimize distortion and improve watermark capacity for database watermarking. However, optimization techniques such as GA increase complexity of the watermarking process. Firefly, a new bioinspired optimization algorithm, is adopted and used with DEW in this work to both minimize distortion and reduce complexity during database watermarking. Experimental results indicate less distortion with improved watermark capacity faster than similar works reported in the literature.

Conflicts of Interest

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] I. Cox, "Digital watermarking," *Journal of Electronic Imaging*, vol. 11, no. 3, 2002.

[2] J. Kiernan, R. Agrawal, and P. J. Haas, "Watermarking relational data: framework, algorithms and analysis," *The VLDB Journal*, vol. 12, no. 2, pp. 157–169, 2003.

[3] R. Sion, M. Atallah, and S. Prabhakar, "Rights protection for relational data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 12, pp. 1509–1525, 2004.

[4] Y. Li, H. Guo, and S. Jajodia, "Tamper detection and localization for categorical data using fragile watermarks," in *Proceedings of the 4th ACM Workshop on Digital Rights Management (DRM '04)*, pp. 73–82, October 2004.

[5] H. Guo, Y. Li, A. Liu, and S. Jajodia, "A fragile watermarking scheme for detecting malicious modifications of database relations," *Information Sciences*, vol. 176, no. 10, pp. 1350–1378, 2006.

[6] Y. Zhang, B. Yang, and X.-M. Niu, "Reversible watermarking for relational database authentication," *Journal of Computers*, vol. 17, no. 2, pp. 59–66, 2006.

[7] X. Zhou, M. Huang, and Z. Peng, "An additive-attack-proof watermarking mechanism for databases' copyrights protection using image," in *Proceedings of the ACM Symposium on Applied Computing (SAC '07)*, pp. 254–258, Seoul, Korea, March 2007.

[8] G. Gupta and J. Pieprzyk, "Reversible and blind database watermarking using difference expansion," in *Proceedings of the 1st International Conference on Forensic Applications and Techniques in Telecommunications, Information, and Multimedia and Workshop*, Adelaide, Australia, January 2008.

[9] A. M. Alattar, "Reversible watermark using the difference expansion of a generalized integer transform," *IEEE Transactions on Image Processing*, vol. 13, no. 8, pp. 1147–1156, 2004.

[10] S. Bhattacharya and A. Cortesi, "A distortion free watermark framework for relational databases," in *Proceedings of the 4th International Conference on Software and Data Technologies (ICSOFT '09)*, pp. 229–234, Springer, Sofia, Bulgaria, July 2009.

[11] D. Hanyurwimfura, Y. Liu, and Z. Liu, "Text format based relational database watermarking for non-numeric data," in *Proceedings of the International Conference on Computer Design and Applications (ICCDA '10)*, pp. V4-312–V4-316, IEEE, Qinhuangdao, China, June 2010.

[12] M. E. Farfoura, S.-J. Horng, J.-L. Lai, R.-S. Run, R.-J. Chen, and M. K. Khan, "A blind reversible method for watermarking relational databases based on a time-stamping protocol," *Expert Systems with Applications*, vol. 39, no. 3, pp. 3185–3196, 2012.

[13] D. M. Thodi and J. J. Rodriguez, "Prediction-error based reversible watermarking," in *Proceedings of the International Conference on Image Processing (ICIP '04)*, Singapore, October 2004.

[14] F. Arif, S. Suhail, and M. Kamran, "Watermarking of relational databases with emphasis on re-watermarking attack," *International Journal of Computer Science Issues*, vol. 9, no. 3, pp. 521–526, 2012.

[15] A. Khan and S. A. Husain, "A fragile zero watermarking scheme to detect and characterize malicious modifications in database relations," *The Scientific World Journal*, vol. 2013, Article ID 796726, 16 pages, 2013.

[16] L. Camara, J. Li, R. Li, and W. Xie, "Distortion-free watermarking approach for relational database integrity checking," *Mathematical Problems in Engineering*, vol. 2014, Article ID 697165, 10 pages, 2014.

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

[18] X.-S. Yang, *Nature-Inspired Metaheuristic Algorithms*, Luniver Press, 2008.

[19] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 1–6, pp. 1942–1948, Perth, Australia, December 1995.

[20] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed optimization by ant colonies," in *Proceedings of the 1st European Conference on Artificial Life*, Paris, France, December 1991.

[21] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Tech. Rep. TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.