Multimed Tools Appl (2015) 74:813–839
DOI 10.1007/s11042-013-1700-9

Watermarking relational databases using bacterial foraging algorithm

Vidhi Khanduja · Om Prakash Verma · Shampa Chakraverty

Published online: 13 October 2013

Abstract The main aspect of database protection is to prove the ownership of data that describes who is the originator of data. It is of particular importance in the case of electronic data, as data sets are often modified and copied without proper citation or acknowledgement of originating data set. We present a novel method for watermarking relational databases for identification and proof of ownership based on the secure embedding of blind and multi-bit watermarks using Bacterial Foraging Algorithm (BFA). Feasibility of BFA implementation is shown in the framed watermarking databases application. Identification of owner is cryptographically made secure and used as an embedded watermark. An improved hash partitioning approach is used that is independent of primary key of the database to secure ordering of the tuples. Strength of BFA is explored to make the technique robust, secure and imperceptible. BFA is implemented to give nearly global optimal values bounded by data usability constraints and thus makes database fragile to any attack. The parameters of BFA are tuned to reduce the execution time. BFA is experimentally proved to be better solution than Genetic Algorithm (GA). The technique proposed is experimentally proved to be resilient against malicious attacks.

Keywords Bacterial foraging · Digital watermarking · Genetic algorithm · Relational database · Copyrights protection

1 Introduction

With the technological advancements evident in every sphere possible, today, data in the form of digital images, audio, video or even content from an outsourced relational database, is available for access with no inconvenience whatsoever. One of the biggest challenges

V. Khanduja (✉) · S. Chakraverty
Department of Computer Engineering, Netaji Subhas Institute of Technology, Dwarka, Delhi, India
e-mail: vidhikhanduja9@gmail.com

S. Chakraverty
e-mail: apmahs.nsit@gmail.com

O. P. Verma
Department of Information Technology, Delhi Technological University, Delhi, India
e-mail: opverma.dce@gmail.com

posed by this rapid increase in data availability on the web is identification of the data and details such as the source of its origin, ownership details, etc. The problem worsens with the fact that the internet is one of the most convenient means to gather information and data providers are encouraged to promote effortless and uncomplicated access to their data. Copyright infringement has become more rampant over the years. Now the main issue for the owners of web data is to find a way to authenticate their data, be able to trace tampered data, prevent misuse of copyrighted data [9] and be able to embed their own identity in the data outsourced. Digital watermarking provides evident solution in establishing owner of data. By embedding the identity of owner as watermark, the proof of ownership can be resolved as in case of any dispute, the watermarks can be extracted from the database that contains disputed data and acts as a proof. After embedding, the watermark and the data are inseparable. Other applications of digital watermarking include authentication, fingerprinting, copy control, and broadcast monitoring, etc. [6]. Now days, most of the web data resides on databases we mainly focus on the copyright protection problem of relational databases. For this kind of application, digital watermarking should have properties such as robustness, fidelity, privacy, accuracy, blindness etc. [1]. The need for watermarking database relations to deter data piracy has been identified and explained by [1]. In this paper, we present a mechanism for database protection i.e. proof of ownership and ownership identification based on the secure embedding of blind and multi-bit watermark on web database. Owner identity is first cryptographically made secure and then used as watermark to be embedded. An improved hash partitioning approach is employed that is independent of primary key attribute of database. BFA is used to enhance robustness and imperceptibility. The technique proposed is experimentally proved to be resilient against various malicious attacks. The problem of watermarking relational databases is optimized using BFA. BFA [18] is nature-inspired optimization algorithm, which is based on the foraging behavior of E. coli bacteria present in the human intestine. In this scheme foraging is modeled as an optimization process where bacteria seek to maximize the energy obtained per unit time spent during foraging. In this paper, we have used BFA to solve the watermarking problem framed.

The paper is organized as follows: Section 2 discusses the related work that includes available watermarking relational databases techniques. This section briefly explains various steps involved in BFA. Section 3 describes the proposed database watermarking technique using BFA. The technique includes procedure for embedding and extraction of watermarks in relational databases. The various steps i.e. Watermark Preparator, Hash Partitioner, and Watermark Embedder involved in the embedding of watermarks in relational database are also discussed. Section 4 presents the experimental results for tuning the parameters of BFA, comparison of GA and BFA and analysis of the robustness against various attacks and finally, conclusions are drawn in Section 5.

2 Related work, motivation and preliminary background

R. Agrawal et.al. [1] identified the need of protecting rights over relational databases by digital watermarking. They proposed that relational database can be watermarked in some algorithmically selected attributes out of several candidates attributes in a tuple. This algorithm embeds watermarks in LSBs of a selected attribute, such that changes in these values will not affect their applicability. The watermarking bits are generated using pseudorandom generators and secret key. The technique does not provide mechanism for multibit watermarks. These bits are then embedded into specific bit locations determined under the control of a secret key known only to the owner of the data. The LSB based data hiding technique is not resilient as simple shifting of the least significant bit by one position leads to a significant loss of watermark without much loss to the database hence are prone to bit-based attacks.

V. Khanduja and O. P. Verma [13] proposed a more imperceptible embedding mechanism that securely and randomly selects multiple attributes out of the selected candidate attributes for inserting watermarks in varying number of least significant bits. The technique resolves the two important concerns namely: owner identification and proof of ownership. This algorithm embeds the identity of a work's copyright holder as a watermark and this watermark can be used to provide evidence in ownership disputes making it useful in applications where proof of ownership is required. However, the resilience of this technique is also affected by changing LSB.

Several image-based watermarking mechanisms [3, 5, 17, 19, 25, 29] have been proposed utilizing various types of attributes. Zhi-Hao Zhang et.al. [29] proposed a novel image-based watermarking method for numerical data. In their method, an image is embedded into relational data which represents copyright information. In this technique the Relational database is divided into various chunks of uniform size. The pixel values of the image were embedded into corresponding locations of attributes by lowering the planer image dimension. Due to the fact that the pixels of image have relative position, and they are sure to be placed in order, this method is vulnerable to conflicting order of embedded marks (pixels) by some subset attacks [19]. J. Sun et.al. [25] used two identification images that are embedded into numeric attribute of relational data in least significant bits of a selected attributes. This technique is also not resilient to LSB attacks. Ali Al-Haj and Ashraf Odeh [3] proposed similar technique by inserting a binary image watermark in the non-numeric attribute of database tuples. The embedding process of each short string of the binary image is based on creating double-space at a location determined by the decimal equivalent of the short string. However, if kerckhoffs principal for public-system is followed i.e. embedding algorithm is publically known, then it is easy to detect the places where watermarks are embedded i.e. where double-spaces are inserted and can be easily removed.

S. Bhattacharya and A. Cortesi [4] build watermark after partitioning tuples as a permutation of tuples. A hash function is built on the top of this grouping. As the ordering of tuples does not affect the original database, this technique is distortion free. C. Jiang et.al. [12] proposed a watermarking algorithm, which can embed the watermark into a relational database transformed in the DWT domain. They provide an analysis of the wavelet's high frequency coefficients, defined an intensive factor and employed the linear correlation detecting method. The watermark can be distributed to different parts of the relational database. H. Cui et.al. [7] proposed a public key cryptography based algorithm. In this algorithm, asymmetric keys are used in inserting and detecting database watermarks in LSB. Private keys are decided by users and public key by trusted center IPR. Users can not destroy the database watermark through public key. Watermark detection can be completed by the third party using public key without secret leaking. D. Hanyurwimfura et.al. [11] embed watermarks in non-numeric multi words data based on lavenshtein distance. A mark is embedded in the selected attribute of selected tuples by horizontally shifting the location of a word depending on watermark bit. The lavenshtein distance between two successive words within an attribute decides the location where the mark is to be inserted. Certain cloud based techniques [14, 30] were proposed in literature. Based on the concept of similar clouds and N-D normal compatibility cloud generators, the numeric attributes in relational database whose schema is persistent is watermarked. R. Sion et.al. [23] proposed technique for watermarking numeric attribute that selects subsets of the relational database and for each subset; a watermark bit is embedded under data usability bounds. The technique stores the marker tuples in order to accurately recover the partitions. This technique violates the principle of blind watermark detection. Most of the techniques discussed above are based on the use of special marker tuples, which makes them vulnerable to watermark synchronization errors resulting from tuple deletion and tuple insertion. Thus, such techniques are not resilient to deletion and insertion attacks. M. Shehab et.al. [21] formulated watermarking of relational databases as a constrained optimization problem and discussed techniques based on Pattern Search and Genetic Algorithm to solve the optimization problem. This makes it resilient to watermark synchronization errors because it uses a partitioning approach that does not require marker tuples. However the technique is primary key dependent, not resilient to linear transformation attack and is not computationally efficient. One of the recent works carried out by Farfoura et.al. [8] proposes reversible technique for watermarking databases. The proposed technique is primary key dependent and not resilient to linear transformation attack. Moreover, technique selects certain tuples into which watermark is to be embedded, thus if those tuples are altered or deleted, the watermark will be lost. Hence such techniques are prone to subset alteration and subset deletion attacks. This technique is also prone to attribute re-order attack.

In order to increase efficiency of the watermarking system, we have applied BFA in this work. To the best of our knowledge the BFA is used for the first time in watermarking databases application. In this work we have designed a novel, improved watermarking relational database system with enhanced robustness, efficiency and imperceptibility using BFA. Moreover, improved hash partitioning approach independent of primary key is proposed in this work. Watermark to be embedded is cryptographically made secure and then embedded into multiple attributes within a tuple of a partition. Further, various parameters of BFA are tuned to make the watermarking system computationally efficient. To make this paper self explanatory, we briefly explain the bacterial foraging algorithm in the following sub-section.

2.1 Bacterial foraging algorithm

BFA is evolutionary algorithm proposed by Passino [18] in 2002 inspired by the group foraging behaviour of bacteria such as E.coli. BFA mimics the four principal mechanisms observed in a real bacterial system: chemotaxis, swarming, reproduction, and elimination-dispersal to solve the non-gradient optimization problem framed in this paper. We define N_b as total population of bacteria, $\theta_i(j,k,l)$ is position of ith bacteria at jth chemotactic step, kth reproduction step and lth elimination-dispersal step. The E.coli cells when stimulated by a high level of succinate, release an attractant called aspertate, which helps them to aggregate into groups and thus move as concentric patterns of swarms with high bacterial density. In the present application, swarming step does not play any major role. Therefore, we ignore the swarming step for our optimization problem and explain the rest of the steps as follows:

A. Chemotaxis: The process, in which a bacterium moves by taking small steps while searching for nutrients, is called chemotaxis. It decides the direction in which the bacterium should move. Depending upon the rotation of the flagella, each bacterium decides whether it should swim (move in same direction) or tumble (change in direction). Change in direction (tumble) of ith bacteria at $(j + 1)$th chemotactic step is given by [18].

$$\theta_i(j+1,k,l) = \theta_i(j,k,l) + C(i).\phi(j) \tag{1}$$

Where, initial value of position for all bacteria is application dependent. It must be initialized such that it lies randomly within the solution domain. $\theta_i(j,k,l)$ is position of ith bacteria at jth chemotactic step, kth reproduction step and lth elimination-dispersal step i.e. previous step. C(i) is basic chemotactic step size and ϕ(j) is unit length random direction for tumble and is defined as

$$\phi(j) = \frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}} \tag{2}$$

Δ(i) is random number between [−1,1].

An objective function is defined as the effort or a cost incurred by the bacteria in search of food. Cost of the ith bacteria at (j + 1)th chemotactic step, kth reproduction step and lth elimination-dispersal step is denoted as J(i,j + 1,k,l). This hiding function value is compared with value of hiding function before tumble J(i,j,k,l) i.e. at jth chemotactic step. If tumble has produced minimum valued hiding function for minimization problem then we make bacteria to swim in same direction N_s times else will move to next bacteria. N_s define number of steps bacteria takes in the same direction after a tumble.

B. Reproduction: The cumulative health of each bacterium during its entire life time is calculated and each of the healthier bacteria will reproduce and split into two bacteria, which are placed in the same location as their parents. If the bacterium reaches a nutrient rich area easily, it will imply that it is indeed easy for the bacterium to survive; hence it will be healthier. Half of the population is eliminated in this step while new healthy ones are added. For the given k and l, and for each i=1, 2, ., N_b, where N_b, is the total population of bacteria. Let health of ith bacteria be

$$J^i_{health} = \sum_{j=1}^{N_c+1} J(i,j,k,l) \tag{3}$$

Where N_c represents total number of chemotactic steps. It measures how many nutrients bacterium got over its lifetime and how successful it was to avoiding noxious substances.

C. Elimination and Dispersal: There may be instances when entire population of bacteria get destroyed or dispersed to new region. To simulate this, in BFA some bacteria are liquidated at random with a very small probability while the new replacements are randomly initialized over the search space.

Thus, each bacterium produces a solution iteratively for a set of optimal values of parameters. Gradually, all the bacteria converge to the global optimum. The information processing strategy of the algorithm is to allow cells to stochastically and collectively swarm towards optima. Algorithm followed in bacterial foraging for calculating optimal value of hiding function for minimization problem is shown in Fig. 1.

BFA is widely used in various applications for solving different optimization problems. The BFA is implemented successfully in edge detection [28], color image enhancement [10], harmonic estimation [15], design of unified power flow controllers [26], transmission loss reduction [27] etc. We have applied the BFA for optimizing the attribute values depending on watermarking bits. As, bringing the watermarked data close to usability vicinity limits will make the watermarked database fragile to any attack [23]. Hence, any attempt to distort watermark will have more risk of making the

```
1. Initialize variables: Number of chemotactic steps $N_c$, Number of swim steps $N_s$,
   Number of reproduction steps $N_{re}$ , Number of elimination dispersal steps $N_{ed}$,
   population of bacteria $N_b$.
2. for each $l \varepsilon N_{ed}$ repeat steps 3 to 15
3.   for each $k \varepsilon N_{re}$ repeat steps 4 to 14
4.     for each $j \varepsilon N_c$ repeat steps 5 to13
5.       for each $i \varepsilon N_b$ repeat steps 6 to13
6.         Evaluate objective function $J(i,j,k,l)$ and assign $J_{last}=J(i,j,k,l)$
7.         Tumble in the direction calculated using eq(1) and evaluate $J(i,j+1,k,l)$
8.         for each $m \varepsilon N_s$ repeat steps 9 to13
9.           if $J(i,j+1,k,l) < J_{last}$
10.            Assign $J_{last}=J(i,j+1,k,l)$ and move further in same direction.
11.          else
12.            exit for loop at step 8
13.          end if
14.    $J_{health}$ is calculated using eq (3) and healthy bacteria will reproduce.
15.  Eliminate and disperse each bacterium with probability $P_{ed}$ to random location on
     optimization domain.
```

Fig. 1 Algorithm of bacterial foraging algorithm for minimization problem

data useless with respect to the usability constraints. BFA always gives global optima solution of our problem in very less time as compared to GA which is mentioned in [21]. Hence BFA improves the resilience of approach and lessen the execution time.

3 Proposed watermarking technique

In general, any watermarking database algorithm can be represented as shown in Fig. 2. While designing the watermarking database system the major concern is not to avoid data alteration, but to limit the change within usability limits i.e. to acceptable levels with respect to the intended use of the data [23]. Watermark insertion is performed by watermark encoder system and detection of embedded watermarks is performed by watermark decoder. The proposed watermarking system consists of two subsystems: 1. Watermark Encoder and 2. Watermark Decoder.

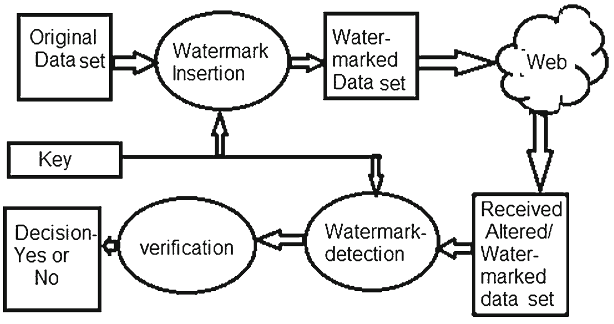


Fig. 2 Block diagram of digital watermarking scheme

3.1 Watermark encoder

Table 1 enlists the notations and parameters that will be used in this paper.

The watermark encoder embeds desired watermarks into a given relational database. The relational database S is a database relation with scheme S (A_0, . . . $A_{N_a -1}$), where N_a is the total number of attributes in the relation S out of which, ν attributes are candidates for watermarking such that $\nu < N_a$ and N_t is the number of tuples in relation S such that $N_t = |S|$. The target attributes are selected by the owner of the database in a manner such that these attributes can tolerate a small amount of error without affecting the usage of the database. The task is achieved using four steps as shown in Fig. 3

A. *Watermark-Preparator*

This step converts the identity of database's copyright holder to the watermark, thus ensuring owner's identification. In order to ensure proof of ownership and owner identification, the watermark (W) must be chosen such that it reflects owner's identity. The watermark to be inserted is selected by owner of the database. Owner creates a final watermark (W′) to be embedded by applying a hash function to the selected watermarking text (W). A cryptographic hash function is a deterministic procedure that takes an arbitrary block of data and returns a fixed-size string; the (cryptographic) hash value, such that any intentional change to the data will change the hash value [24]. The hash values are called the message digest or simply digest. One of the most important properties of hash function is that it is infeasible to find a message that has a given hash. We employ Message-Digest 5 algorithm as a cryptographic hash function with a 128-bit (16-byte) hash value. However, any of the secure hash functions can be used e.g. RIPE-MD, SHA, HAVAL, SNEFRU etc. The selection criteria to choose among various hash functions include encryption speed and hash length [20]. The final watermark created in this step is a set of s bits $W' = b_{s-1}.....b_0$ that are to be embedded in database. The length of watermark s is selected such that $s << m$, to enable multiple embeddings of the watermark in the database where m is number of partitions to be created.

Table 1 Notations used

Symbol	Description
S	Relational database
N_a	Number of attributes in S
ν	Number of numeric attributes candidate for watermarking
N_t	Number of tuples in S
W′	Watermark bit sequence = $b_{s-1} . . . b_0$
s	Length of watermark sequence
m	Total number of data partitions
M[i][p][d]	Multidimensional partition vector
K_s	Secret key
X_{min}	Minimum embedding statistics
X_{max}	Maximum embedding statistics
U	Usability constraints
δ[i][p][d]	Multidimensional manipulation vector
Γ	Set of secret parameters chosen by owner

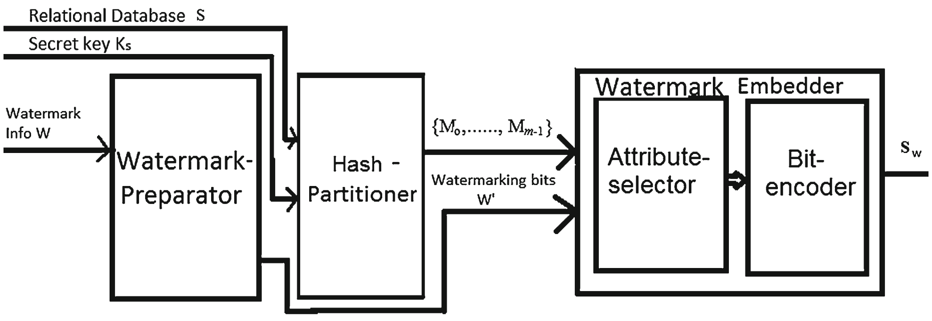


Fig. 3 Watermark encoder

According to kerckhoff's principle, embedding algorithms are publically known; security lies in secret key [1]. In such cases, if direct watermark i.e. we say owners identity is taken as watermark; then an attacker can easily guess the watermark knowing the owner of the database. Therefore, guessing the closest watermark and the algorithm, it is not impossible to crack the keys used. Hence, attacker can claim that this is his database by successfully extracting the correct watermark in case of dispute with the owner. *Watermark-Preparator* step acts as an additional protective shield to entire embedding process.

Most of the watermarking techniques available in literature lack this important step of watermark preparation and does not contain information on how watermark is selected, secured and prepared [7, 11, 12, 21, 23, 25, 29]. In some of the available work, the watermark does not contain the information about the owner. In certain cases either an image is ciphered into the watermark [3, 17, 19] or the random meaningless bit stream [1, 2, 30] acts as a watermark.

We summarize the strength points of our *Watermark-Preparator*:

1. Watermark created contains owner's identification.
2. Watermark is cryptographically made secure using hash function.
3. In addition to security, hash fixes the length of variable-sized watermark. In our work we have given owner the liberty to select his own watermark. Owner can select large files such as an image, audio or even video also to be embedded as watermark. Keeping in mind the restriction on the length of the watermark to be embedded (as watermarks are embedded repeatedly for robustness), we consider the hash to represent the shortened reference to original selected watermark.
4. Secure multi-bit watermark is prepared and embedded multiple times depending on number of partitions.
5. This step is analogous to digital signature using hash where encryption of the digest is replaced by secure embedding of watermark into the database that robustly provides authentication of the owner/ownership identification.

B. *Hash-Partitioner*

An improved hash-partitioning technique is used to partition the database S into *m* non-overlapping partitions M = $\{M_0,.....,M_{m-1}\}$ in-order to secure the ordering of tuples. This strategy designates one or more attributes from the database S as the partitioning attribute. A hash function is chosen in the range $\{0,1,......,m\text{-}1\}$. Each tuple on the original relation is hashed on partitioning attributes [22]. If the hash function returns i, then the tuple is placed on partition M_i. The secret key K_s which is selected by the owner of the database, is concatenated with Most Significant Bits (MSB) of the normalised partitioning attributes in order to make tuple-to-partition assignment more secure. Since owner knows K_s, he can reproduce result but Mallory, an attacker does not know K_s, can't. This makes it more difficult for an attacker to predict the assignment. Such one-way hash function with secret key is referred as Message Authentication Code (MAC) [20]. Figure 4 shows the flowchart for the function create_partitions() that implements the Hash-Partitioner.

In this flowchart, the MAC for each tuple t ε S is computed and used to assign tuples to partitions. For any tuple t, its partitioning index, I_d is given by [23]:

$$I_d = H(K_s \| p_k \| K_s) \bmod m \tag{4}$$

Where, p_k is derived key that acts as a primary key and is defined as

$$p_k = MSB(NORM(t_{ctr}.A1)) \| MSB(NORM(t_{ctr}.A2))$$

MSB(t.A1) and MSB(t.A2) are most significant bits (MSB) of partitioning attributes in relations S, K_s is secret key selected by owner, H is a secure hash function, and || is the concatenation operator. MSB of partitioning attributes are considered because MSB will not be affected even if attacker tries to alter the value of attribute within usability constraints. The MSB space is assumed to be a domain where minor changes on the collection items have a minimal impact on the MSB labels.

To defeat linear transformation attack, normalization is performed in which a common divider to all the items is identified and applied [23]. In this attack, numeric values are linearly transformed e.g. attacker may convert the data to different unit of measurement (like Fahrenheit to Celsius). In such a case, entire attribute get changed to

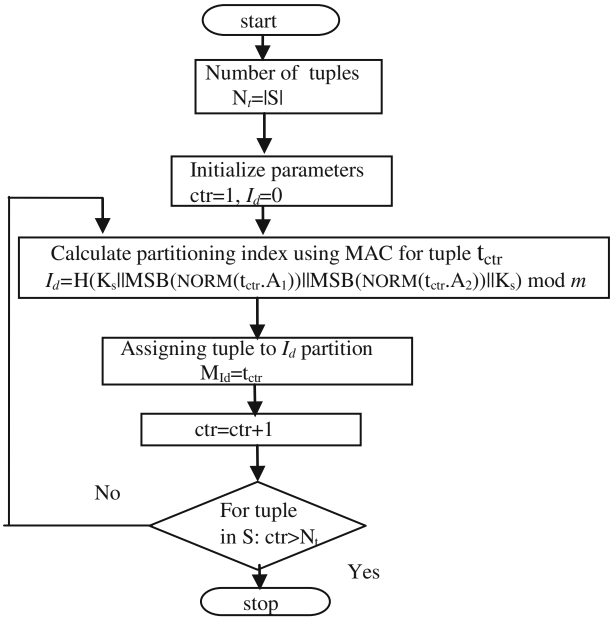

start
Number of tuples
N_t=|S|
Initialize parameters
ctr=1, I_d=0
Calculate partitioning index using MAC for tuple t_ctr
I_d=H(K_s||MSB(NORM(t_ctr.A_1))||MSB(NORM(t_ctr.A_2))||K_s) mod m
Assigning tuple to I_d partition
M_Id=t_ctr
ctr=ctr+1
For tuple in S: ctr>N_t
No
Yes
stop

Fig. 4 Flowchart for create_partitions(.)

different value and hence watermark is removed. Thus, normalising the attribute values and then using them defeats this attack. Normalization of partitioning attribute of a tuple is denoted as NORM (t_{ctr}.A1). Hence, tuple t_{ctr} will be assigned partition M_{Id}.

Partitioning attribute and hash function has to be carefully selected to avoid partition skew. This technique creates *m* partitions which are not overlapping i.e. for any two partitions M_i and M_j, if $i \neq j$ then $M_i \cap M_j = \{\Phi\}$ as shown in Fig. 5. So far, the scheme ensures three levels of protection against attacks. Firstly, an attacker cannot predict the tuples-to-partition assignment without the knowledge of the secret key, number of partitions *m* and the selection of partitioning attributes, which are kept secret and are known only to owner of the database. Security is further enhanced as hash partitioning technique is independent of the primary key. Lastly, normalised partitioning attributes values are used to defeat linear transformation attack.

C. *Watermark-embedder*

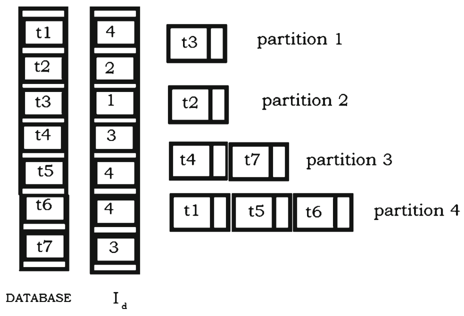
The *Watermark-embedder* embeds multi-bit watermarks into each of the partitions created by the Hash-Partitioner. It is assumed that the tuples in a partition M_i contain multiple numeric attributes out of which ν attributes are candidates for watermarking. The Attribute-Selector further chooses attributes to be watermarked that varies for each partition. Multi-bit watermarks are embedded because for each tuple multiple attributes can be selected and a single bit is embedded multiple numbers of times in earmarked attributes of a tuple. Therefore, partition M_i can be represented as a multidimensional partition vector M[i][p][d] where i is the partition index, p is the attributes selected for a i^{th} partition into which watermarking bit is to be embedded and d is the tuples selected in the i^{th} partition. Values of p and d may vary for each partition as attribute selector selects different attributes for each partition. Finally, the bit-encoder uses BFA to embed watermarks into respective partitions and thus, enhances robustness. The *Watermark-Embedder* consists of following subcomponents:

i. **Attribute-Selector:** A Cryptographic Pseudorandom Sequence Generator (CPSG) is created with initial seed, I_s as

$$I_s = H(K_s \| I_d \| K_s) \tag{5}$$

Where, H is secured hash function, K_s is secret key selected by owner, || is the concatenation operator and I_d is calculated using (1). Output of CPSG is retrieved as a vector with number of states equal to ν. These states decide what all attributes in a partition are selected for watermarking [13]. Attributes selected are those for which there is corresponding ones present in the output vector of CPSG. The number of bits in

Fig. 5 Example of data partitioning



the output vector of CPSG can be controlled programically. Thus, multidimensional partition vector M_i can be represented as a M[i][p][d]. We represent A_i as attributes selected in ith partition and T_i as tuples in ith partition. Since, the output of the attribute-selector depends on the pseudorandom generator; this increases the level of security as for each partition a different number of attributes (A_i) is selected out of a total of ν candidate attributes as shown in Fig. 6. In order to erase a watermark, the attacker has to correctly guess the tuple-to-partition assignment and then corresponding selected attributes. Candidate attributes are selected by owner of database and hence can be made independent of attribute re-ordering attack by first sorting them alphabetically and then selecting.

ii. **Bit-Encoder:** The Bit-Encoder embeds watermarks into the partitions based on the watermarking bits generated by Watermark-Preparator. Watermarking is done based on statistical properties of the database. Advantage of embedding watermark bits into actual data distribution properties (as opposed to directly into the data itself) includes increased resilience to various types of attacks and the tolerance of considerable data loss as compared to fragility of direct data domain encoding[23].

Bit-encoder converts the multidimensional partition vector M[i][p][d] to a new multidimensional data vector M^W[i][p][d] = M[i][p][d] + δ[i][p][d], where δ[i][p][d] is known as the multidimensional manipulation vector. Manipulations for all the values(1 to T_i) of a particular attribute(say, pth) within a particular partition(say, ith) i.e. δ(i,p,1:T_i) are bounded by the data usability constraints U(i,p,1:T_i). Usability constraints are the constraints imposed on each attribute of the database that it can tolerate without affecting its applicability. The set of usability constraints U represents the bounds on the tolerated change for the elements of M. These constraints are selected by owner of database and are different for different applications and the corresponding data. Examples of constraints include: i) uniqueness—each value must be unique ii) scale—the ratio between any two number before and after the change must remain the same and iii) classification—the objects must remain in the same class (defined by a range of values) before and after the watermarking [23]. These constraints define the feasible space for the manipulation vector δ(i,p,1:T_i). We employ usability constraints that are classification preserving constraints for implementation. The values altered must be such that they belong to same classification as the original data. The Bit-Encoder selects the optimal value of δ(i,p,1:T_i) depending on the watermarking bit and statistical properties of database. For optimization the following steps are used to calculate the hiding function [21]:

Step1: Calculate the reference point for each selected attribute (p) within ith partition

$$R_i = \mu_i + c^*\sigma_i \tag{6}$$

where, c ε [0,1]; is a secret real number that is selected by owner and is part of γ, μ_i is mean of M^W(i,p,1:T_i) and σ_i is variance estimates of the set M^W(i,p,1:T_i).

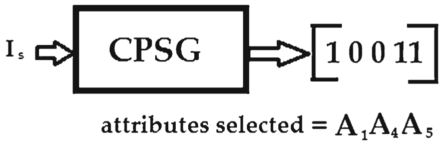


Fig. 6 Attribute selector

```
Step 2:  Initialize sum=0
Step 3:  For all the elements of a particular attribute (p) in ith partition $M^W(i,p,1:T_i)$ i.e.
         j=1 to $T_i$
    a)   If $M(i,p,j) > R_i$ then sum = sum + 1
         [End of for loop]
Step 4:  Hiding function is defined as
```

$$J_\gamma\left(M_i^w\right) = \frac{sum}{T_i} \tag{7}$$

Where, T_i is the total number of values in a particular attribute (pth) of ith partition and γ is the set of secret parameters which is decided by the owner of the database. Thus, the values of the selected attributes vary according to statistical properties of the values of that attribute within a particular partition. Basically, the hiding function calculated depends on number of positive violators [23] of an attribute within a particular partition. All the elements that lie beyond R_i are considered as positive violators [23] as shown in Fig. 7.

Our motive is to vary values of all the elements of an attribute in a partition within the usability constraints such that either the number of the elements that lies beyond R_i increases or decreases. This limits to the amount of values altered. Since the values of all the elements are varied we prefer to solve this problem using optimization. The hiding function in Eq. (7) is defined as an objective function for this constrained optimization problem. Thus, the watermarking bit (b_i) decides whether the objective function is maximized or minimized.

If the bit b_i is equal to 1, then the Bit-Encoder maximizes the objective function without violating usability constraints. However, if the bit b_i is equal to 0, then the objective function is minimized. The solution to the problem of optimization generates the manipulation vector $\delta^*[i][p][d]$ at which $J\gamma(M^w(i,p,1{:}T_i))$ is optimal. The new data partition set $M^W(i,p,1{:}T_i)$ is computed as

$$M^w(i,p,1:T_i) = M(i,p,1:T_i) + \delta^*(i,p,1:T_i) \tag{8}$$

This process is repeated for all the partitions created and within a partition for all the attributes selected. The function created for this step is named bit_encode(). Flowchart for bit_encode() is shown in Fig. 8.

The maximization and minimization solution statistics are recorded in X_{max}, X_{min} for each encoding step. These values are used by Threshold-Evaluator to compute decoding parameters. The formulated optimization problem is nonlinear constrained optimization problem. BFA is used to solve this optimization problem. Cost is calculated using hiding function defined in (7). Our goal is to let the bacterium search for the manipulation values

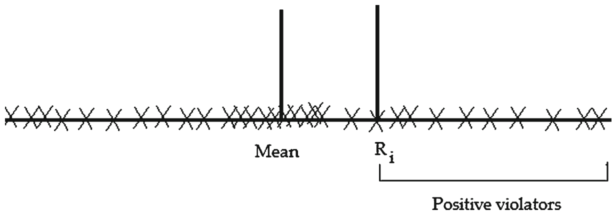


Fig. 7 Distribution of $M(i,p,1{:}T_i)$ showing positive violators

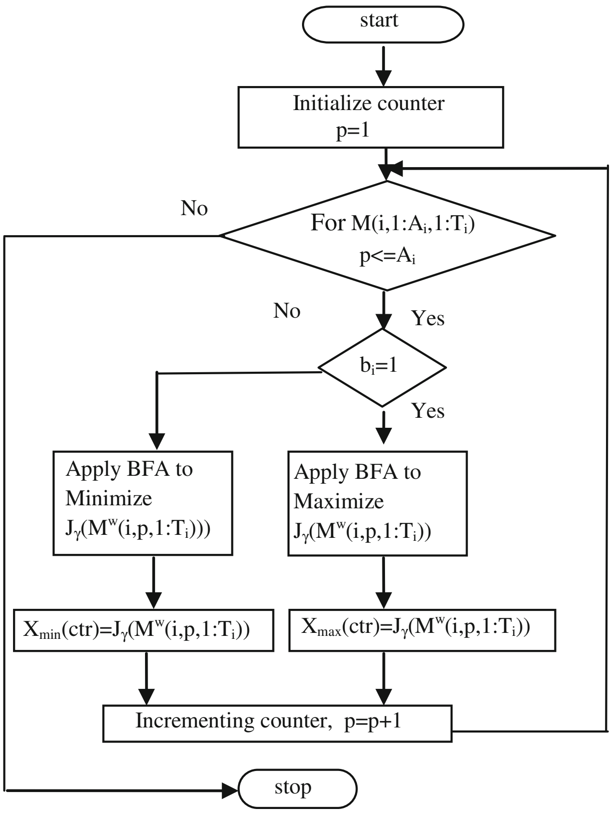

start
Initialize counter
p=1
No
For M(i,1:A_i,1:T_i)
p<=A_i
No
Yes
b_i=1
Yes
Apply BFA to Minimize J_γ(M^w(i,p,1:T_i)))
Apply BFA to Maximize J_γ(M^w(i,p,1:T_i))
X_min(ctr)=J_γ(M^w(i,p,1:T_i))
X_max(ctr)=J_γ(M^w(i,p,1:T_i))
Incrementing counter, p=p+1
stop

Fig. 8 Flowchart of bit_encode()

which when added to their respective attribute values will lie below reference point and hence minimize the objective function. Reverse process can be attained for maximization. Position of bacteria's must be initialized such that it lies randomly within the solution domain of $\delta(i,p,1{:}T_i)$. The T_i-dimensional search space for bacteria consists of the optimal values of $\delta(i,p,1{:}T_i)$ which when added to partition vector will yield optimized data values. Nutrient rich area is modelled as the one in which more values out of T_i in particular partition will be before reference value for minimization. Least healthy bacteria are ignored. First reference point for a particular attribute of a partition is calculated and then objective function is calculated. Similarly for all the attributes in the partition $M(i,1{:}p,1{:}d)$ and then finally for every partition objective function is calculated.

In this approach the objective function optimizes the vector consisting of number of elements in a particular attribute of a partition. However various parameters of BFA are experimentally tuned to get optimal results in reduced time. An experiment was conducted using normally distributed data, where the objective function was maximized and minimized using BFA with classification preserving constraints. Further, Genetic Algorithm and BFA are compared and results of the experiments are recorded in Section 4.2. BFA always gives the global optimal solution under usability constraints and thus database values lies in the usability vicinity limits. Bringing the watermarked data close to usability vicinity limits will make the watermarked database fragile to any attack [23]. Any attempt to distort watermark will have more risk of making the data invalid with respect to the guaranteed usability metrics, thus removing or diminishing its value. Moreover, as the optimization algorithm is to be applied to each of the attribute selected within a partition separately, the chosen optimization technique should not take more time. As, BFA takes less execution time, the algorithm has an upper edge than GA. We summarize optimization problem framed in this step as follows:

1. We have framed watermarking optimization problem as changing the data such that either they increase or decrease the number of positive violators. All the data items within a partition are optimised to permissible limits. Thus, all the elements of the selected attribute within a partition are parameters for optimization. Values of all the elements are optimized in such a way they increase or decrease the number of positive violators depending on watermarking bit.
2. The changes in the data items have to be such that it should not violate the usability constraints making data useless. In this work, we have considered classification preserving constraint for example; consider the grading system in an organization. If a person scores 58 marks and according to classification rules, marks between 50 and 70 gets 'B' grade. Then after optimization his marks will remain in same category i.e. between 50 and 70. In case objective function is maximized and initial value of reference point is say, 62. Then optimization process will make his marks greater than 62 in order to maximize the sum. His marks can attain maximum of 70, beyond which category will be changed and hence, constraints will get violated.

   For example, in this work we have considered classification preserving constraint i.e. in case an element belongs to 'A' grade whose range is 80–100 then modified optimised value will remain in same category. If maximized will be close to 100 else close to 80.
3. Every data has its own predefined limitations up to which it can tolerate distortions, beyond which data is useless. We have used this in our approach. We are embedding the watermark such that embedded data reaches its boundary. Further modification in that direction will make data useless i.e. data value will exceed permitted bandwidth. Hence, attacker is left with no choice. Thus watermark is secured. By optimizing objective function, we are optimizing data items. Thus, data items reaching their maximum/minimum permissible values (global optima solution) will improve the watermarking quality.

3.2 Watermark decoder

Watermark decoding is the reverse of encoding. It is process of extracting the embedded watermark bits using the watermarked database S_W' and the secret key K_s. Process assumes that even if the database is altered or distorted, embedded watermarks are successfully retrieved. Therefore, S_W' is used instead of S_W. Statistics (X_{max}, X_{min}) are stored after embedding of each bit and are used by the Threshold-Evaluator of decoder to compute the decoding threshold. The decoding algorithm is blind as the original database S is not required for the successful decoding of the embedded watermark. The watermark decoder is divided into four main steps as shown in Fig. 9.

A. *Watermark-Preparator*

   By following same technique as used in Watermark-Encoder watermarks are prepared. Watermark to be inserted is selected by owner of the database. The watermark must be chosen such that it reflects owner's identity. Owner selects the watermarking text W to create a watermark to be embedded (W') consisting of s bits. Else watermark prepared by encoder W' can directly be used in this step. This will save computation time and will not affect security of the process.

B. *Hash-Partitioner*

By implementing data partitioning algorithm used in Encoder, the data partitions are generated.

Input to function: Database S_W', Secret key K_s, number of partitions m.

Output of function: Data partitions M_0', ..., M_{m-1}'.

C. *Watermark-Extractor*

This technique is based on threshold. First the threshold is evaluated based on statistical values of X_{max} and X_{min} and then the bits decoded are recovered. It consists of following subcomponents:

i. **Attribute-selector**: By following same technique as used in Watermark Encoder attributes is selected using cryptographic pseudorandom sequence generator (CPSG) with initial seed as in (5) whose output is a vector that decides what all attributes in a partition are selected for watermarking. Thus each partition M_i' can be represented as a multidimensional partition vector such that M'[i][p][d]) where p is the attributes selected for ith partition into which watermarking bit is to be embedded and d is the tuples selected in ith partition.

ii. **Threshold-Evaluator:** The value of the threshold T is calculated in order to minimize the probability of an embedded bit decoded incorrectly i.e. known as probability of bit decoding error. The maximized hiding function values corresponding to b0 is equal to 1 are stored in the set X_{max}. Similarly, the minimized hiding function values are stored in X_{min}. Considering following terms:

P_{err} i.e. probability of decoding error is calculated to get threshold T using [30]:

$$\frac{\sigma_0^2-\sigma_1^2}{2\sigma_0^2\sigma_1^2}T^2+\frac{\mu_0\sigma_1^2-\mu_1\sigma_0^2}{\sigma_0^2\sigma_1^2}T+\ln\left(\frac{P_0\sigma_1}{P_1\sigma_0}\right)+\frac{\mu_1^2\sigma_0^2-\mu_0^2\sigma_1^2}{2\sigma_0^2\sigma_1^2}=0 \tag{9}$$

P_0	probability of encoding a bit as 0.
P_1	probability of encoding a bit as 1.

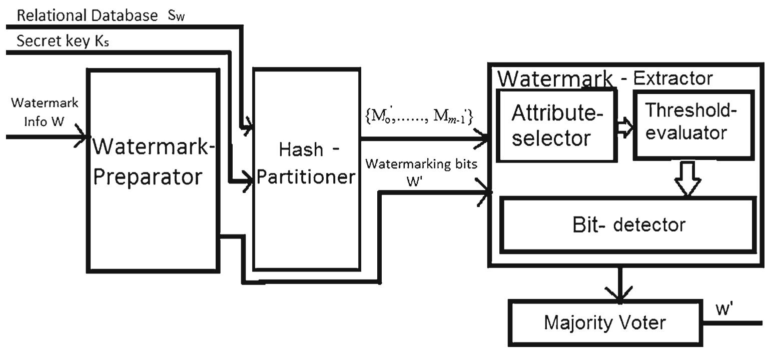


Fig. 9 Watermark decoder

The threshold T increases the effectiveness of the embedded watermark by raising the chances of successful decoding.

iii. ***Bit-Detector*****:** Using watermarked data partition $M^w[i][p][d]$, the bit detector computes hiding function $J_\gamma(M_i^w)$ and compares it to the decoding threshold T. If $J_\gamma(M_i^w)$ is greater than T, then the decoded bit is 1; else the decoded bit is 0.

D. *Majority-Voter*

The watermark bits are decoded using a majority voting technique [23]. Each watermark bit is extracted many times as each of the watermark bit W' = $b_{s-1}......b_0$ is embedded several times in the different data partitions. Advantage of embedding single watermark bit multiple times is that when extracted using majority voting, probability of decoding bit incorrectly almost nullifies.

4 Experiments and analysis

In this section, we report the results of an extensive experimental study that analyzes the resilience of the proposed watermarking scheme to the attacks. All the experiments were performed on 2.13 GHz Intel Core i3 CPU with 2GB of RAM. Experiment was performed to execute proposed method by BFA. Various parameters of the algorithms were tuned to get best results. A national geochemical survey database of the US [16] is taken for experiment. These data composes a complete, national-scale geochemical coverage of the US, and enables construction of geochemical maps, refine estimates of baseline concentrations of chemical elements in the sampled media, and provide context for a wide variety of studies in the geological and environmental sciences. In the database the stream sediments and soils in the US, from existing data are analyzed. Database of 287 attributes with 77,212 records is taken for the experiment. The usability constraint considered is classification preserving constraints that are used to control the magnitude of the alteration for δ[i][p][d]. BFA is executed to optimize the various partitions created of the sample relational database. c=0.75 is chosen, 128-bit watermark is created and number of partitions, m=1100 is considered.

Experiment is conducted by initializing the parameters and the results obtained are shown in Fig. 10.

The initial values of the BFA algorithm paramaters selected are:

N_b=50	Total number of bacteria in the population
N_c=100	The number of chemotactic steps
N_s=4	The swimming length.
N_{re}=4	The number of reproduction steps
N_{ed}=2	The number of elimination-dispersal events
P_{ed}=0.25	Elimination-dispersal probability.

The BFA always converges to global minima and global maxima for our problem. This strength of BFA is explored to make the technique robust and imperceptible. Since better the separation between minimal and maximal values, the database is more resilient to subset alteration attack. Bringing the watermarked data close to usability vicinity limits will make the watermarked database fragile to any attack. Any attempt to distort watermark will have more risk of making the data invalid with respect to the guaranteed usability metrics, thus removing or diminishing its value.

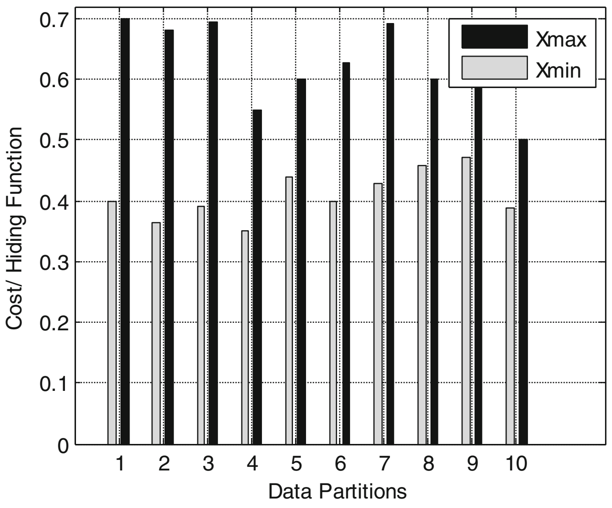


Fig. 10 Optimized Hiding function values attained at various partitions

4.1 Tuning of BFA parameters

To overcome execution time constraint, various experimental tuning of parameters is performed as follows.

$\mathbf{N_b}$**:** Keeping all other parameters constant and varying population size of bacteria (N_b), the results observed are mentioned in Table 2 for single partition.

The experiment is performed on 4 different partition sets as shown in Fig. 11. Different partitions were taken and minimum cost is calculated by varying total bacteria population $N_{b.}$ The optimal results can be obtained with minimum execution time by reducing the total number of bacteria to 10. In fact by reducing bacteria population to 10, the execution time reduces by approximately 81 %.

$\mathbf{N_c}$: The cost i.e. value of hiding function of a single bacterium at various chemotactic steps is observed, keeping other parameters constant. It is found that at N_c=40 optimized values are obtained when population size is reduced to 20 (Table 3).

This results in 60 % reduction in execution time when compared with execution time at N_c=100 as shown in Fig. 12.

Experiment is performed on four different partition sets. Figure 13 reveals that for N_b=10, the optimal value can be obtained by reducing the N_c to 40 when the execution time is further reduced by 60 %.

Table 2 Cost and execution time by varying N_b (For minimization)

N_b	Cost(Minimization)
50	0.4091
40	0.4091
30	0.4091
20	0.4091
10	0.4091
04	0.4545

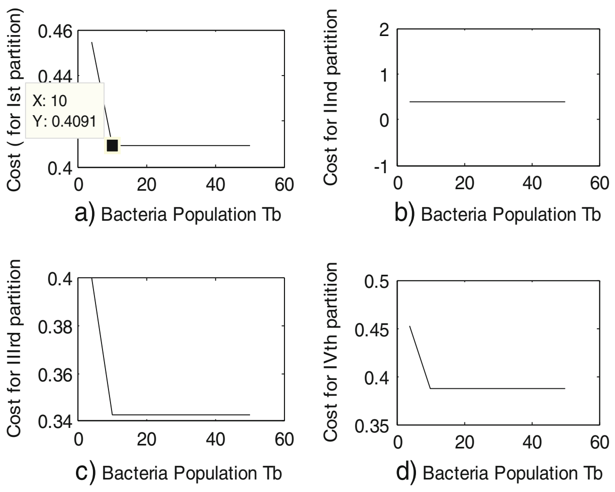


Fig. 11 Minimization cost value at various bacteria population for 4 different partitions

4.2 Comparison with the existing optimization based approach

The experiment was performed using Genetic Algorithm (GA) and Pattern Search (PS) [21]. The systematic behaviour of PS leads to the fast convergence to optimal feasible solutions. However, PS is not guaranteed to find a global optimum. To attain optimal points starting points must be initialized properly. However, in our problem large number of parameters is optimized multiple times according to number of partitions, hence randomly generated starting points are to be given for which PS does not explore the global structure of cost function and so it can get attracted by local optima in most of the cases. Unlike PS technique, GA has the ability to converge at the global solution point, as it can handle the search space from different directions simultaneously. GA has very less probability to get trapped at local optima because of Crossover and Mutation operators between chromosomes. However, when the system has a highly epistemic objective function (i.e. where parameters being optimized are highly correlated), and number of parameters to be optimized is large, then GA has been reported to exhibit degraded efficiency [26]. However [21] discussed that solving the optimization problem does not necessarily require finding a global solution because finding such solution may require a large number of computations. We aim at finding global optima within usability constraints as it improves robustness.

As global solutions in our problem will make the values in the usability vicinity limits and thus any attempt to make changes in watermarked database can diminish its value. Thus

Table 3 Cost and execution time by varying N_c (For minimization)

N_c	Cost(Minimization)
100	0.4091
80	0.4091
60	0.4091
40	0.4091
20	0.4545
10	0.4545

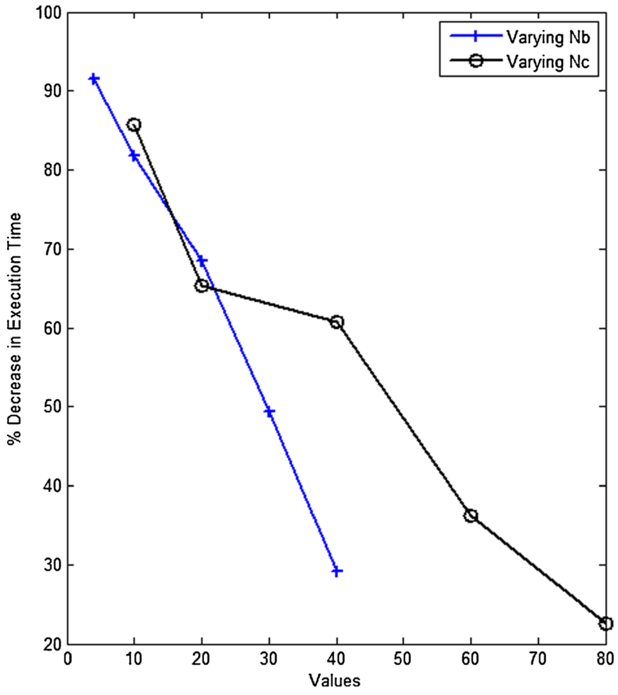


Fig. 12 Decrease in execution time by varying values of N_b and N_c

we have explored the strength of BFA i.e. it always finds global optima to solve our optimization problem. For the sake of comparison, same fitness function is applied to both GA and BFA approaches and the initial population for both of them are randomly initialized. We performed the experiment on different partitions of normally distributed data and the results are shown in Fig. 14. Figure shows the minimized and the maximized objective function obtained at different partitions by GA and BFA. For GA, the population size and number of generations both are taken as 1,000 for optimal results.

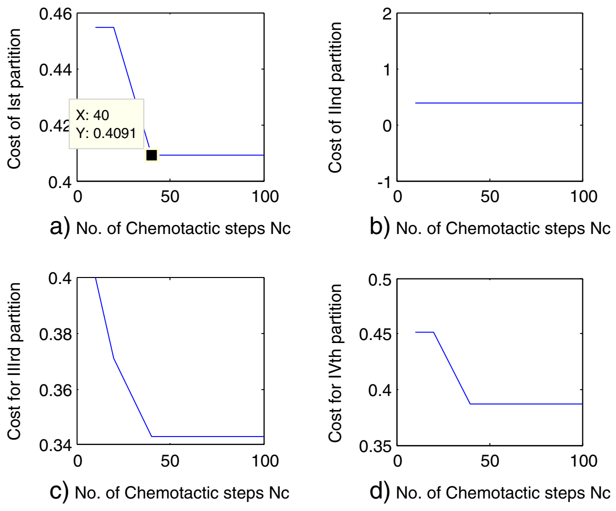


Fig. 13 Cost of Bacterium for different partitions at N_b=10 (Min. value attained at N_c=40)

Execution time of BFA after tuning all the parameters is approximately 56.9 % less than the execution time required by GA for obtaining the same quality of solutions. The choice of the technique will vary with problem domain. For faster, optimal and secure performance of our constrained optimized problem the use of BFA technique is recommended.

In watermarking system, the time to embed the watermark should not be very large. Otherwise, more often updations to the database will create incremental watermarking practically infeasible. However, in this work we mainly focused on time criticality and robustness of the proposed watermarking system. Experiments revealed that BFA gives best result and after tuning various parameters, time has significantly reduced as compared to GA. Moreover, we have improved the robustness of the watermarking system by making it more resilient to subset addition, subset alteration, subset deletion, subset reorder and attribute re-ordering attack (discussed in section 4.3). Further, our proposed technique is resilient to the linear transformation attack while [21] is not. We have made our technique primary key independent by implementing improved hash partitioning technique. We summarize major difference in Table 4.

4.3 Analysis of robustness

Our algorithm embeds the watermark by changing the entire attribute value within usability constraints. Thus, altering LSB will have negligible effect on watermarking bits. This makes the algorithm robust against various bit-based attacks e.g. randomization, zero out and bit flipping attack. In these attacks, attacker assigns random values to certain bit positions, sets least significant bit (LSB) to zero or simply inverts values at LSB in order to destroy the watermark [1]. Moreover, most of the techniques proposed in literature [1, 2, 8, 11, 13, 19, 21] are primary key dependent and hence algorithm does not works for databases without primary key. In our approach the primary key is derived using MSBs of the two attributes selected by owner of the database. Hence, works for any type of relational database.

We now analyze the robustness of our watermarking technique against various forms of attacks.

i. *Attribute Re-ordering attack*: In this attack, an attacker may change the order of attributes or tuples in an attempt to destroy watermark bits [23]. Attribute Selector in our technique, first sorts the attributes alphabetically and then use them for selection of final attributes where watermarking bits are to be embedded. As, the candidate attributes

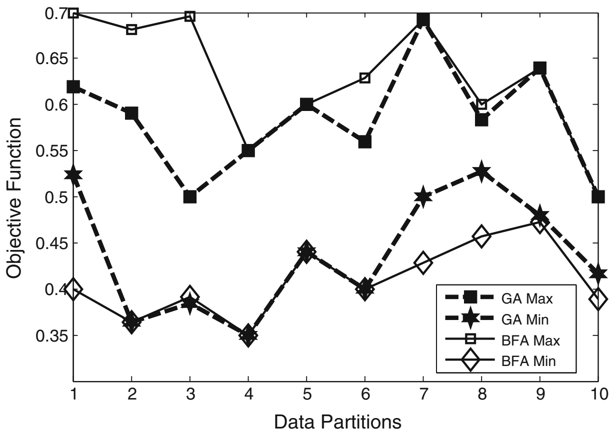


Fig. 14 Comparison of GA [21] & BFA by plotting value of X_{min} & X_{max} at different data partitions

Table 4 Comparision between our technique and Shehab's technique [21]

Our approach	Shehab's [21]
Watermark represents owner's identity and hence ensures owner identification.	No discussion on Watermark to be embedded. Watermark is considered as bit stream.
Watermark Preparator step is introduced to secure the watermark to be embedded and hence acts as an additional protective shield to entire embedding process. Embedding of watermark directly may help an attacker to crack the key in case owner is known to him.	Watermark not secured.
Owner can select large files to create watermark such as an image, audio or even video also. Hash of selected watermark is calculated that represents the shortened reference to selected watermark.	Restriction on the length of the watermark to be embedded because watermark bits are embedded repeatedly to improve robustness.
Normalization is performed before embedding watermark in which a common divider to all the items is identified and applied to defeat linear transformation attack which is a special case of subset alteration attack.	It is not resilient to linear transformation attack.
Sorting of attributes is performed by attribute selector makes the technique resilient to Attribute Re-ordering attack.	Technique not resilient to attribute re-ordering attack.
Proposed technique is Primary key independent and hence works on all type of relational databases.	Technique will not work for relational databases without primary key.
BFA is implemented to solve the framed constrained optimization problem.	Pattern Search and Genetic Algorithm are suggested.
For our problem, BFA always gives global optima solutions (Fig. 14). This makes embedded data items close to their boundaries i.e. permitted bandwidth. Further modification in that direction by an attacker will make data useless. Hence, attacker is left with no choice. Thus watermark is secured and **robustness** of the watermarking system is enhanced.	PS and GA do not guarantee to give global optima solution.
After tuning BF parameters i.e. population size and number of chemotactic steps, same optimized results are obtained. Also, execution time has reduced significantly. **Execution time** of BFA after tuning all the parameters is approximately 56.9 % less than the execution time required by GA for obtaining the same quality of solutions. In watermarking system, the time to embed the watermark should not be very large. Otherwise, more often updations to the large database create incremental watermarking practically infeasible.	Execution time found to be more.

are selected by owner of database, thus can be made independent of attribute re-ordering attack by first sorting them alphabetically and then selecting. Moreover, sorting of tuples are done based on cryptographically secure hash function

ii. *Subset Deletion Attack*: In this attack, attacker randomly drops α tuples from the watermarked database, the watermark is then decoded and watermark loss is measured for different α values. Technique used embeds same watermarking bits multiple times within database, increases the robustness against this attack as bits loss in deleted tuples may be recovered from rest of the tuples. Figure 15 shows the experimental results. Results revealed that the proposed watermarking technique is resilient to the random deletion attack as the watermark was successfully extracted with 100 % accuracy even when more than 80 % of the tuples were deleted. Beyond this, if an attacker further deletes more tuples, then the chances of the database becomes useless increases. However, the robustness of this attack depends on number of watermarking bits(s) and the number of places where watermarks are to embed. Since our technique embeds watermarks into multiple attributes within a tuple, hence increasing the places to hide watermark bits. Moreover, another factor that influences is number of partitions. As one bit is embedded in a partition, the number of times same bit is embedded into different partitions increases and hence efficient to extract. However in case of marker-based watermarking techniques proposed earlier, deletion of tuples may delete marker tuples and result in deletion of bits in between the embedded watermark sequence. Consequently, this results in a watermark synchronization error.

iii. *Subset Alteration Attack*: In this attack, attacker alters the data value of α tuples. However, altering the data for disturbing the watermark can easily violate the usability constraints and make the data useless. Since our technique uses BFA that gives best optimal values hence maximizes the distance between the hiding function values in both minimization and maximization cases; thus, it makes it more difficult for the attacker to modify the embedded watermark bit. As global solutions in our problem will make the values in the usability vicinity limits and thus any attempt to make changes in watermarked database can diminish its value. In addition, by the repeated embedding of the watermark, this attack can easily be defeated. Hence technique proposed is resilient towards alteration attack. Experiment reveals that our technique extracts watermarks with 100 % accuracy even if an attacker succeeds to alter 100 % of tuples without harming usefulness of database (Fig. 16). Since in our technique every partition will have varying number of attributes into which watermarking bit is to be embedded, it is difficult for an attacker to find what all attribute values are watermarked in a particular tuple.

A special case of subset alteration attack is linear transformation attack. In this attack, numeric values are linearly transformed e.g. attacker may convert the data to different unit of measurement (like Fahrenheit to Celsius). In such a case, entire attribute get changed to different value and hence watermark is removed [23]. In the proposed work, the normalized values of the attributes are taken. A common divisor to candidate attributes are identified and then applied in partitioning step to defeat this attack.

Fig. 15 Resilience to deletion attack

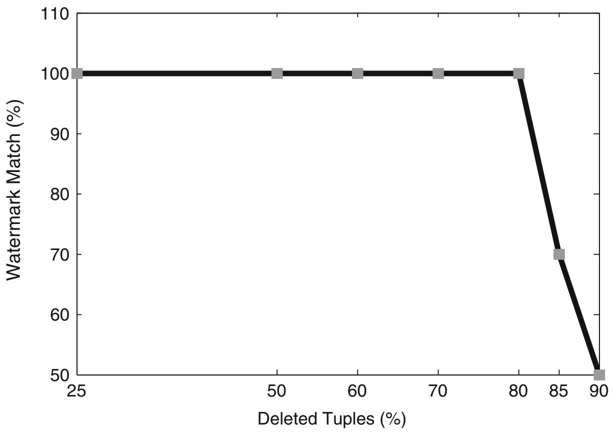
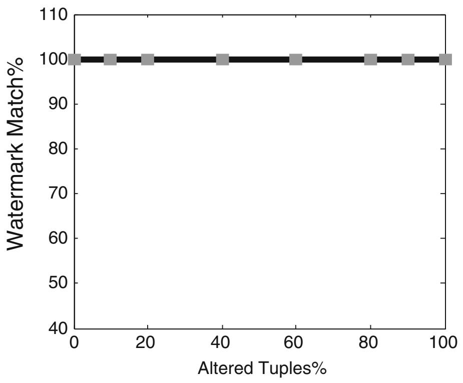


Fig. 16 Resilience to alteration attack



iv. *Subset Insertion Attack*: In this attack, an attacker decides to insert α tuples to the database S_W hoping to perturb tihe embedded watermark. The new tuples acts a noise to the embedded watermark. However, use of Hash-Partitioner and Watermark-Embedder in our proposed technique makes the watermark embedding more secure as is based on a cumulative hiding function that operates on all the tuples in the partition. Thus, the effect of adding tuples will not affect much to the value of the hiding function and the watermark bit embedded. The watermark was recovered with 95 % accuracy even when up to 100 % percent of the database size tuples were inserted as shown in Fig. 17.

However in case of marker-based watermarking techniques proposed earlier, addition of tuples may introduce new markers in the database and result in addition of new bits in the embedded watermark sequence. Consequently, this results in a watermark synchronization error. As discussed in case of deletion attack, effect of addition of new tuples within a partition can be defeated by embedding watermarks into multiple attributes within a tuple and by increasing number of partitions.

v. *Synchronization Error*: Synchronization is an important problem in a watermarking scheme. In a watermarking system, watermark extracted should be in same order as embedded. If synchronization is lost, even if no modifications have been made, the embedded watermark cannot be correctly verified. Technique presented is resilient to watermark synchronization errors because it uses a partitioning approach that does not require any type of marker tuples thus, tuple deletion or tuple insertion will not affect order of extracted watermarking bits.

Fig. 17 Resilience to Insertion attack

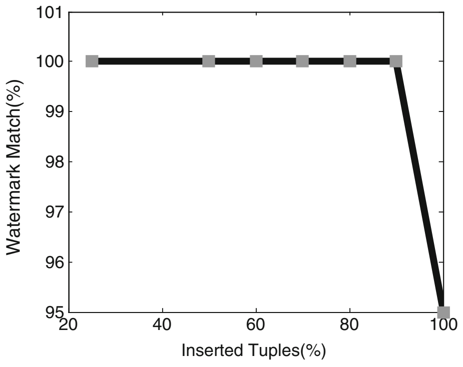


Table 5 Effect of watermarking on mean and variance of data

Watermark	Attribute	Percentage change in mean	Percentage change in variance
Equal number of 1's and 0's	A1	−1.23	0.70
	A2	−0.28	1.2
More number of 1's	A1	−1.37	−0.20
	A2	−0.58	1.8
More number of 0's	A1	−1.3	0.80
	A2	−0.02	1.7
All 1's	A1	−1.30	0.69
	A2	−0.03	1.2
All 0's	A1	−1.07	1.16
	A2	−0.04	0.9

4.4 Measuring the impact of watermarking on data distribution

In an experiment, we analyzed the perturbation in the data-set caused due to watermarking. We monitored the mean and variance of the values of an integer-valued attribute first before watermarking and later after watermarking. Different watermark patterns were generated varying the number of 1's in the watermark. These watermarks were applied on two different attributes. The observed values are recorded in Table 5.

The results reveal that the net effect on mean and variance is negligible, irrespective of the pattern of watermarking bits inserted. The percentage change in the mean value of data pertaining to the two attributes chosen, ranged from −1.37 % to −0.02 %. The change in the variance of data ranged from 1.8 % to 0.2 %.

5 Conclusion and future scope

In this paper, we presented a mechanism for data protection with proof of ownership and ownership identification based on the secure embedding of blind and multi-bit watermarks on a given web database. In this work **time criticality** and **robustness** of the watermarking system are considered. The proposed approach is independent of primary key as Hash Partitioner uses the most significant bits of the normalized database instead of primary key. The main contributions of our work are summarized below:

1) We have significantly enhanced the degree of robustness of the watermarked database to a variety of security threats and attacks by applying an intelligent mix of techniques. Specifically:
    i. The Identification of owner is cryptographically made secure and then used as an embedded watermark.
    ii. An improved hash partitioning technique is presented that does not depend on either primary key or marker tuples to locate the partitions thus, making it resilient to watermark synchronization errors.
    iii. Watermarking of multiple numerical attributes based on normalized database is performed and hence is resilient to linear transformation attack.
    iv. Sorting of attributes is performed by attribute selector makes the technique resilient to Attribute Re-ordering attack.

2) To compliment the secure watermarking scheme, the recovery process is also made robust to ensure that watermarks can be extracted back accurately even from a distorted or altered watermarked database. Firstly, for each tuple multiple attributes can be selected. Next, a bit is watermarked multiple numbers of times in the earmarked attributes of a tuple. Finally, a majority voting technique is used to recover embedded watermarked bits reliably.
3) We presented a comparison of BFA and GA as optimizing techniques in our watermarking scheme and concluded that bacterial foraging give better results in terms of processing time as well as the optimality of results obtained. BFA always converges to global optima. The global solutions will make the values in the usability vicinity limits and thus any attempt to make changes in watermarked database can diminish its value.

We have experimentally demonstrated that the proposed watermarking technique is resilient to subset insertion, subset deletion, subset alteration, attribute re-ordering and linear transformation attacks and shown that the change in mean and variance of data after watermarking is negligible. The proposed watermarking technique is thus a secure, robust and imperceptible algorithm for numeric attributes that provides reliable ownership identification.

Our future research will be directed towards increasing the level of attack resilience against several sources of attacks in the watermarking method, and proposing new techniques for watermarking database relations for the attributes other than numeric. Other evolutionary algorithms available in the literature to solve constraint optimization problem framed in this work. Finally, different applications for our proposed technique could be envisioned and enforced.

References

1. Agrawal R, Haas PJ, Kiernan J (2003) Watermarking relational data: framework, algorithms and analysis. VLDB J 12(2):157–169
2. Ali YH, Mahdi BH (2011) Watermarking for relational database by using threshold generator. Eng Tech J 29(1):33–43
3. Ali A-H, Odeh A (2008) Robust and blind watermarking of relational database systems. J Comput Sci 4(12):1024–1029
4. Bhattacharya S, Cortesi A (2009) A distortion free watermark framework for relational databases. In: Shishkov B, Cordeiro J, Ranchordas A (eds) ICSOFT 2009: Proceedings of the 4th international conference on software and data technologies, vol 2, Sofia, Bulgaria, INSTICC Press, pp 229–234
5. Chen X, Chen P, He Y, Li L (2008) A self-resilience digital image watermark based on relational database. Int Symp Knowl Acquis Model 698–702
6. Cox IJ, Miller ML, Bloom JA (2002) Digital watermarking. Morgan Kaufmann, Academic Press
7. Cui H, Cui X, Meng M (2008) A public key cryptography based algorithm for watermarking relational databases. Proc IEEE International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Harbin, 1344–1347. doi:10.1109/IIH-MSP.2008.194
8. Farfoura ME, Shi-Jinn H, Jui-Lin L, Ray-Shine R, Rong-Jian C, Khan MK (2012) A blind reversible method for watermarking relational databases based on a time-stamping protocol. Expert Syst Appl 39(3):3185–3196
9. Gupta VK (1997) Copyright issues relating to database use. DESIDOC Bull Inf Technol 17(4):11–16
10. Hanmandlu M, Verma OP, Kumar NK, Kulkarni M (2009) A novel optimal fuzzy system for color image enhancement using bacterial foraging. IEEE Trans Instrum Meas 58(8):2867–2879
11. Hanyurwimfura D, Liu Y, Liu Z (2010) Text format based relational database watermarking for non-numeric data. Proc IEEE International Conference on Computer Design and Applications (ICCDA), vol 4. Qinhuangdao, 312–316. doi:10.1109/ICCDA.2010.5541119
12. Jiang C, Chen X, Li Z (2009) Watermarking relational databases for ownership protection based on DWT. Proc Fifth Int Conf Fifth International Conference on Information Assurance and Security, vol 1. Xi'an, 305–308. doi:10.1109/IAS.2009.220
13. Khanduja V, Verma OP (2012) Identification and proof of ownership by watermarking relational databases. Int J Inf Electron Eng 2(2):274–277

14. Min H, Jia-heng C, Zhi-yong P, Cheng Z (2004) A new mechanism based on similar clouds watermark for database's information security. Wuhan University. J Nat Sci 9(4):415–419. doi:10.1007/BF02830434
15. Mishra S (2005) A hybrid least square-fuzzy bacterial foraging strategy for harmonic estimation. IEEE Trans Evol Comput 9(1):61–73
16. National geochemical survey database of the US, http://tin.er.usgs.gov/geochem/
17. Odeh A, Al-Haj Ali (2008) Watermarking relational database systems. First International Conference on the Applications of Digital Information and Web Technologies, Ostrava, 270–274. doi:10.1109/ICADIWT.2008.4664357
18. Passino KM (2002) Biomimmicry of bacterial foraging for distributed optimization and control. IEEE Control Syst Mag 22(3):52–67
19. Sardroudi HM, Ibrahim S (2010) A new approach for relational database watermarking using image. 5th International Conference on Computer Sciences and Convergence Information Technology (ICCIT), Seoul, 606–610. doi:10.1109/ICCIT.2010.5711126
20. Schneier B (2008) Applied cryptography, protocols, algorithms and source code in C. Wiley-India
21. Shehab M, Bertino E, Ghafoor A (2008) Watermarking relational databases using optimization-based techniques. IEEE Trans Knowl Data Eng 20(1):116–129
22. Silberschatz A, Korth HF, Sudarshan S (2005) Database system concepts. McGraw-Hill Int. Edition
23. Sion R, Atallah M, Prabhakar S (2004) Rights protection for relational data. IEEE Trans Knowl Data Eng 16(12):1509–1525
24. Stalling W (2006) Cryptography and network security- principles and practices. PHI
25. Sun J, Cao Z, Hu Z (2008) Multiple watermarking relational databases using image. Proc IEEE Int Conf MultiMedia Inf Technol 373–376
26. Tripathy M, Mishra S (2006) Bacteria foraging: a new tool for simultaneous robust design of UPFC controllers. International Joint Conference on Neural Networks (IJCNN). Vancouver, BC, 2274–2280. doi:10.1109/IJCNN.2006.247025
27. Tripathy M, Mishra S, Lai LL, Zhang QP (2006) Transmission loss reduction based on FACTS and bacteria foraging algorithm. Lect Notes Comput Sci 4193:222–231
28. Verma OP, Hanmandlu M, Kumar P, Chhabra S, Jindal A (2011) A novel bacterial foraging technique for edge detection. Pattern Recognit Lett 32(8):1187–1196
29. Zhang Z-H, Jin X-M, Wan J-M (2004) Watermarking relational database using image. Proc IEEE Third International Conference on Machine Learning and Cybernetics, vol 3. 1739–1740. doi:10.1109/ICMLC.2004.1382056
30. Zhang Y, Niu X, Zhao D (2005) A method of protecting relational databases copyright with cloud watermark. World Acad Sci Eng Technol 3:170–174

Vidhi Khanduja received her B.Tech degree in Information Technology from Guru Govind Singh Indraprashtha University, Delhi, India and M.E degree in Computer Technology and Applications from Delhi College of Engineering under University of Delhi, Delhi, India and is currently pursuing her PhD from Department of Computer Engineering, Netaji Subhas Institute of Technology, University of Delhi, India.

She worked as an Assistant Professor in Department of Information Technology, Delhi College of Engineering (now Delhi Technological University) Delhi, India for more than 4 years. She is currently working as Teaching Cum Research Fellow at Netaji Subhas Institute of Technology, University of Delhi, India. She is a member of IEEE. Her research interests include Database protection, Information security, Evolutionary algorithms, artificial intelligence.

Om Prakash Verma received his B.E. degree in Electronics and Communication Engineering from Malaviya National Institute of Technology, Jaipur, India in 1991 and M. Tech. degree in Communication and Radar Engineering from Indian Institute of Technology (IIT), Delhi, India, in 1996 and PhD from University of Delhi, Delhi, India in 2011.

From 1992 to 1998 he was assistant professor in Department of Electronics & Communication Engineering, at Malaviya National Institute of Technology, Jaipur, India. He joined Department of Electronics & Communication Engineering, Delhi College of Engineering (now Delhi Technological University) Delhi, India, as Associate Professor in 1998. Currently, he is Head of Department of Information Technology at Delhi Technological University, Delhi. He is also the author of more than 35 publications in both international journal and conference proceedings. He has guided more than 26 M. Tech. student for their thesis. He has authored a book on Digital Signal Processing in 2003. He is a Principal investigator of an Information Security Education Awareness project, sponsored by Department of Information Technology, Government of India. His research interests include image processing, application of fuzzy logic in image processing, application of evolutionary algorithm in image processing, artificial intelligent and digital signal processing.

Shampa Chakraverty is Professor in the Department of Computer Engineering in Netaji Subhas Institute of Technology, New Delhi, India. She is B.E (Electonics and Communication-83, Delhi Univ.), M.Tech (Integrated Electronics and Circuits'92, IIT-Delhi), Ph.D (Delhi University). Her research interests include Design space exploration of hardware software architectures, Information Retrieval and Computer Security.