Verify a Valid Message in Single Tuple: A Watermarking Technique for Relational Database

Shuguang Yuan[1,2], Jing Yu[1,2](✉), Peisong Shen[1], and Chi Chen[1,2]

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Science, Beijing 100089, China
{yuanshuguang,yujing,shenpeisong,chenchi}@iie.ac.cn

[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100043, China

Abstract. The leakage of sensitive digital assets is a major problem which causes huge legal risk and economic loss. Service providers need to ensure data security for owners. Robust watermarking techniques play a critical role in ownership protection of relational databases. In this paper, we proposed a new Double-layer Ellipse Model called DEM that embeds a valid message in each candidates tuples. Each watermark can independently prove ownership. The main idea of DEM is to use watermarks itself to locate and make the most of contextual information to verify validity of watermarks. Under the framework of DEM, we propose a robust and semi-blind reversible watermarking scheme. Our scheme handles non-significant data for locating and embedding. The scheme generates watermark by exchanging data groups extracted from scattered attributes using Computation and Sort-Exchange step. Key information such as primary key, most significant bit (MSB) become a assistant feature for verifying the validity of embedded watermark. Our scheme can be applied on all type of numerical attributes (e.g. Integer, float, double, boolean). In robust experiments, the scheme is proved to be extremely resilient to insertion/detection/alteration attacks in both normal and hard situations. From a practical point of view, our solution is easy to implement and has good performance in statistics, incremental updates, adaptation.

Keywords: Watermarking · Relational database · Robust

1 Introduction

With the arrival of data era, business interest in database are mined and analyzed. At the same time, attackers also covet the values of database. Thus, copyright protection of relational database is an essential requirement for data owners and service providers. Watermarking is mainly used for ownership protection and tamper proofing. Watermarking techniques are widely used in many fields like images [14] , multimedia [6], text [11], databases [1] and applications [20]. In relational database, the robust watermarking [3,16,18,21–23] and fragile watermarking [4,8,10,13] are used for ownership and integrity checking respectively.

The watermarking process mainly comprised of locating phase, embedding phase, and detection phase. In locating phase, positions of watermarks are identified. There are three ways of locating: pseudorandom sequence generator (PSG), hash function (HF), and statistical feature. The [1] relys on fixed order generated by PSG and constant starting/end point. A number of techniques like [2,3,17,21–23] use HF (Example of candidates for HF are the MD5 and SHA) with key information (e.g. Primary Key, Virtual Primary Key, MSBs) for locating. The [7] embeds watermarks on non-significant effect features which locates by mutual information(MI). When it comes to embedding phase, there are two typically approaches. The type of approaches in [2,21–23] is that one embedding validates one bit (OEVOB). In detection phase, OEVOB is a pattern that every iteration on one tuple validates one bit. The other approaches in [3,16,18] are multiple embedding validates one bit (MEVOB). One bit of watermarks is embedded per group of tuples by modifying the values of one or several digits (according to the variables that are bounded by constraints).

In this paper, we propose a new idea that embeds a valid message in every selected single tuple and declares ownership by counting numbers of valid message. The detection can be reduced to counting problem which includes many standalone matching process on embedded tuples. This type model is one embedding validates one message (OEVOM). We propose a innovative Double-layer Ellipse Model (DEM) based on OEVOM. The main idea of DEM is that each watermark will have an independent process which includes actual embedding/detection and verification. DEM has flexible mechanism that can adjust single or combined feature with different weight according to the application scenarios.

Based on DEM, we implement a robust and semi-blind reversible watermarking scheme. The scheme can be applied on all type of numerical attributes (e.g. Integer, float, double, boolean). The embedding phase has two major processes. Firstly, the scheme extract digits from scattered attributes. The selected digits are scattered and unsignificant. Secondly, watermark will be generated in Computation and Sort-Exchange step. Based on the Computation result by key, two data groups are exchanged. The values in the corresponding original tuples are modified. Meantime, the scheme extracts context information for verification. Our watermarking scheme has following characteristics: **1.** Our scheme use watermarks itself to locate. Contextual information including key attributes becomes an enhanced factor of watermark. In our case, the extra locating placeholders (e.g. Primary Key, Virtual Primary Key, MSBs) are optional. Thus, our scheme has weak dependence on key information and improved robustness. **2.** Our scheme can generate a valid message of ownership on each selected tuple. These embedded watermarks are independent of each other. The embedding process supports on-demand incremental updating while has changing previous record. Data can be watermarked separately and stored centrally. Or it is to embed watermark centrally, store separately. In detection phase, our scheme counts the number of valid message. **3.** Our scheme has little impact on statistical result of data. By setting proper positions of watermarking, experiment shows the statistical impact can be reduced to negligible. **4.** Our scheme is a reversible watermarking technique. It can recover watermarked data to original data by inverse embedding progress.

Our contributions are summarized as follows:

1. We propose a model DEM for OEVOM. It includes complete locating, embedding and detection processes. DEM abstracts components of watermarking and evaluation framework.
2. We implement a new watermarking scheme which addresses practical factors such as robust, low false hit rate, incremental updates and low statistical impact.

The paper is organized as follows: In Sect. 2, we describe the related work. In Sect. 3, we present Double-layer Ellipse Model. In Sect. 4, the implementation of scheme including detail of process is shown. In Sect. 5, data-driven experiments are showed. In Sect. 6, the performance of the robustness against subset addition/deletion/alteration attack is presented. In Sect. 7, we conclude our work.

2 Related Work

The watermarking techniques can be classified into three broad categories [9]: Bit-Resetting Techniques (BRT), Data Statistics-Modifying Techniques (DSMT) and Constrained Data Content-Modifying Techniques (CDCMT). The first relational databases watermarking technique was published by Agrawal *et al.* [1,15], which depends on embedding bit string as watermarks. By Hash and secret key, it selects a fraction of tuples, attributes and bit locations for embedding. A number of following techniques like [2,21–23] continue to improve this model method. Cui *et al.* [22] proposed a weighted watermarking algorithm which assigns different weights to attributes. Guo *et al.* [2] proposed a twice-embedding scheme for identifying both the owner and the traitor. Zhou *et al.* [23] presented a scheme that embeds image (BMP file) into the relational databases, and an error correction approach of BCH (BoseChaudhuri-Hocquenhem) coding is used for enhancing the robustness of the algorithms. Wang *et al.* [21] introduced speech signal as watermark. The speech signal is more meaningful and correlative to the data owner. In [17,19], Sion *et al.* provided a new idea for categorical data. It establishes a secret correspondence between category attributes according to a certain rule. Above mentioned techniques belongs to Bit-Resetting Techniques (BRT).

Recently, Data Statistics-Modifying Techniques have evolved gradually. In [18], Sion *et al.* proposed a method that encoding of the watermark bit relies on altering the size of the “positive violators” set. This solution addresses data re-sorting, subset selection, linear data changes attacks. Shehab *et al.* [16] formulated the watermarking techniques of relational database as a constrained optimization problem. They embedded watermarks with constraints on partitioned tuple groups. They presented two techniques to solve the formulated optimization problem based on genetic algorithms and pattern-searching techniques. In terms of [7], Saman *et al.* developed an information techniques that embeds watermarks on non-significant effect features. The attribute selection relys on mutual information(MI) for controlling data distortions. In [3], Javier *et al.* raised a scheme that modulates the relative angular position of the circular histogram center of mass of one numerical attribute for message embedding. Compared with the other two categories of watermarking techniques, fewer Constrained Data Content-Modifying Techniques (CDCMT) are proposed. Li *et al.* [12] proposed a publicly verified watermarking scheme of which detection and ownership proof can be effectively performed publicly by anyone. The idea is that the sets of selected MSBs (selected by pseudorandom sequence generator) of attributes are jointed together to form the watermarks.

3 Model

3.1 Attacker Model

Alice marks database D to generate a watermarked database D_w. The attacker Mallory can operate several types of attacks. The target of attacks is deleting the watermarks. In terms of robust watermarking techniques, survivability is important gist of technological choices of watermarking. We assumes Mallory haven't secret key of watermarking process and original data. Mallory's malicious attack may take various forms: **1. Deletion Attacks.** For the purpose of destroying watermarks, Mallory can delete subset of watermarked tuples from database D_w. Attackers haven't secret key for locating position of watermarks. Deletion will damage tuples which may has watermarks. **2. Alteration Attacks.** In this type of attack, Mallory can change the value of tuples on database D_w. Mallory randomly select 1 to 5 attributes and modify their values. We assume that the attacker dose not know the real positions where the watermarks was embedded. **3. Insertion Attacks.** Mallory adds similar tuples that may disturb detection.

3.2 Double-Layer Ellipse Model

The main components of DEM is presented in Fig. 1. The embedding and detection use the same model but different components. As a model for OEVOM, the main proposal of DEM is to use watermarks itself to locate positions and use contextual information to verify watermarks.

The model is divided into two layer. The watermark should be generated primarily by inner layer. The inner layer defines six components: **Filter**, **Preprocessor**, **Calculator**, **Indexer**, **Modifier**, **Recover**. The components of Filter and Preprocessor select tuple and generate processed data. Calculator computes processed data by parameters and key to generate watermarks. Indexer stores watermarks with affiliated feature extracted from outer layer. Modifier

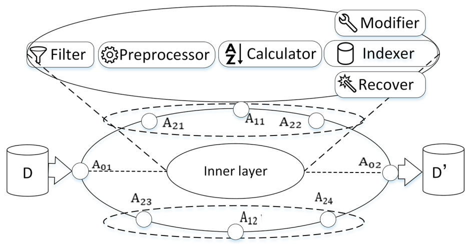

Fig. 1. The Components of DEM.

and Recover both execute SQL to finish modification task on databases. The locating and embedding use the same position of data, which needs a trade-off that makes as few changes as possible for embedding but uniqueness for locating. How to find a suitable bandwidth on tuples for both locating and embedding becomes a priority issue.

The outer layer extracts contextual information for verification. The score functions on outer layer ranks features with different weight. Facing different usage scenario, developers can adapt different strategy of features selection and weight assignment. In outer layer, DEM defines vertex, co-vertex, normal points as affiliated feature according to relative position compared with inner ellipse. To do so, strong dependence on key information become a enhanced verification instead of necessity. In general, unique feature, like primary key, has high credibility for verifying watermark. Correspondingly, fractional or circumjacent information has low credibility with low weight.

The below formula measures probability of watermarking of database D_w by verifying every tuples from 0 to R. When inner layer detects a watermark with $p_i = 1$, the score of every feature is calculated by $w_0 q_{i0}, ..., w_n q_{in}$. The score of watermarked tuple is $w_{pi} + max\left\{w_0 q_{i0}, ..., w_n q_{in}\right\}$. However, if current tuple can't be identified as watermarks with $p_i = 0$, the score of current tuple is 0. The q_i is the score function of feature A_j with weight w_i. The result of weighted accumulative detection is compared with the threshold τ. The synthetic evaluation weight of watermarking is given by

$$Score(D_w) = \frac{\sum_{i=0}^{R} p_i \left(w_{pi} + max\left\{w_0 q_{i0}, ..., w_n q_{in}\right\}\right)}{S} * 100\%. \tag{1}$$

The $f(A, K)$ should be piecewise function where value is 0/1 to determine whether the condition of the feature is satisfied. The vertex and co-vertex verify current and non-neighbor information of watermarked tuple respectively. The normal points have geometrically symmetric. A couple of normal points can represent adjacent feature in opposite directions of context of watermarked tuple in the vertical. And in the horizontal, a couple of normal points have the relation of "AND" or "OR". For example, the relation of A_{j1} and A_{j2} have relation of "And", so they obtain one weight when both conditions are met. Secret key K

is added to salt the result for protecting leakage of feature information. The feature A_j is computed as follows

$$q_{ij} = \begin{cases} f_{j1}\left(A_{j1}, A_{j2}, K\right) & j = 0, 1 \\ \frac{f_{j1}(A_{j1}, A_{j2}, K) + f_{j2}(A_{j3}, A_{j4}, K)}{2} & j \neq 0, 1. \end{cases} \quad (2)$$

The m is total number of feature. Besides features can be combined for obtaining a higher weight value

$$q_{ik} = \sum_{1 \le n \le m} \sum_{1 \le i_1 i_2 \dots i_n \le m} \left(q_{i_1} + q_{i_2} + \dots + q_{i_n}\right). \quad (3)$$

The notations present in Table 1 for reference.

Table 1. Notations

Sym	Description	Sym	Description
D	Original database	A	The feature of contextual information
D_w	Watermarked database	N_w	Watermark groups
R	Number of tuples	p_i,q_j	Score function
S	Number of watermarked tuples	τ	Threshold of detection
r	A tuple	w_i	Weight of i-th feature
r_w	A watermarked tuple	W_i	Watermark of i-th entity
r_{adj}	Adjacent tuples of r	n	Number of feature of outer layer
K	Secret key	l	Number of attributes in a tuple
C	Configuration of watermarking process	l_g	Number of digits in a group
M	The value of modulo	L_g	Number of groups
I	Interval of watermark	K_g	Key groups
Pos_w	Available length of attributes	N_g	Data groups

4 Proposed Scheme

We proposed a robust and semi-blind reversible watermarking scheme under DEM. The scheme consist consists of three subsystems: Watermark Embedding, Watermark detection and Data Recovery. Scheme simplifies six components for unfolding our algorithm. The scheme embeds private message which stands on exchanging positions of scattered digits by key, and generating index for locating. Our scheme uses the scattered data on tuples as the carrier of watermarks. It includes a configure set $C = \{l_g, I, M, P_1, P_2, Pos_w\}$ and secret key K.

4.1 Watermark Embedding

The embedding process is showed below.

Algorithm 1 Preprocessor

```
Require: $Pos_w, l_g, l, r$
Ensure: $N_g$
 1: $N_g \leftarrow Array[], result \leftarrow ""$
 2: for $i = 0 \rightarrow l$ do
 3:     if $r[i]$ is not integer/float or $Pos_w[i] <= 0$ then
 4:         continue
 5:     end if
 6:     $spliteDigit \leftarrow extractDigits(r[i])$
 7:     if $length(splitDigit) > Pos_w[i]$ then
 8:         $result.concat(spliteDigit.lastSubString(Pos_w[i]))$
 9:     else
10:         $result.concat(spliteDigit)$
11:     end if
12: end for
13: $N_g \leftarrow groupBySize(result, l_g)$
14: return $N_g$
```

Algorithm 2 Modifier

```
Require: $W_i, Pos_w, r, l$
Ensure: $r_w$
 1: $r_w \leftarrow r, iter \leftarrow 0$
 2: for $i = 0 \rightarrow l$ do
 3:     if $r[i]$ is not integer/float or $Pos_w[i] <= 0$ then
 4:         continue
 5:     end if
 6:     $spliteDigit \leftarrow extractDigits(r[i])$
 7:     $len \leftarrow length(spliteDigit)$
 8:     if $len > Pos_w[i]$ then
 9:         $subMark \leftarrow W_i.subString(iter, iter + Pos_w[i])$
10:         $value \leftarrow value.subString(0, len - Pos_w[i]) + subMark$
11:         $iter \leftarrow iter + Pos_w[i]$
12:     else
13:         $subMark \leftarrow W_i.subString(iter, iter + len)$
14:         $value \leftarrow subMark$
15:         $iter \leftarrow iter + len$
16:     end if
17:     $r_w[i] \leftarrow fillDigit(value)$
18: end for
19: $replaceTuple(D, r_w)$
20: return $r_w$
```

1. Filter. Scheme generates random number in the range of 0 to I to pick a candidate every I tuples. The selected tuple will be embedded a watermark. The parameter I controls the density of watermark in watermarked database D_w. The number of watermarked tuples S is R divided by the interval I. By comparing the actual quantity with the theoretical quantity, scheme has a measurable baseline by interval I for watermarking detection.

2. Preprocessor. In Fig. 2 A, the process of Preprocessor is described. The embedding uses digits across different attributes. For preserving statistic, the last few place of integer/decimals of attributes are considered. Only one or two digits are extracted in a single attribute. Pos_w is a predetermined list that contains available length of each attribute. Extraction work makes reference to Pos_w. The extracted digits are grouped into the N_g of size l_g. Algorithm 1 describes the steps involved in Preprocessing. l is the number of attributes in tuple r. In line 3, the attribute $r[i]$ is determined whether it is suitable for extracting digits. This ensures that suitable type (e.g. Integer, float, double, boolean) under restrained available length will not be skipped. Line 6 splits the decimal place of float/integer. Finally, integrated digits are divided into groups. Meanwhile, context of selected tuple enters outer layer of DEM.

3. Calculator. The watermark W will be generated in this component. The groups N_g forms watermark W which is resulted by Computation and Sort-Exchange steps. During Calculator on inner layer, scheme has four parameters key K, modulo M, and exchange position P_1, P_2. The key K is transformed to K_g as the same format of N_g.

Firstly, Computation step defines operator $\bigoplus$ that each element in N_g is multiplied by same-position element in K_g respectively, then result of $\bigoplus$ is added up. Because the number of K_g is less than N_g, K is recycled on rounds in $\bigoplus$ operation. The modulo operator with the value M is prevent from order-preserving. Secondly, according Computation result, the Sort-Exchange step swaps P_1-th largest value and P_2-th largest value using Quick-sort algorithm. The spliced result forms watermark W_i. Only if attackers possess C and K, can this process be repeated. Figure 2 B describes the steps involved in Calculator.

After generating W_i on inner layer, affiliated features are extracted from contextual tuples. Four features are selected with score function. Feature A_{01} defines q_1 that relies on hash function of MSB (Most Significant Bit) with key. This feature is assigned a high score because of uniqueness of MSB. A_{21} and A_{23} compare significant information of adjacent tuples and are defined by function q_2, q_3. A_{11} records the distance from the previous same entry during traversal. When two embedded tuples where surrounding tuples have same content, feature A_{11} helps distinguish correct one. A_{21}, A_{23} and A_{11} together form a high score evaluation function q_4. All score functions use K for salting in order to prevent original information disclosure.

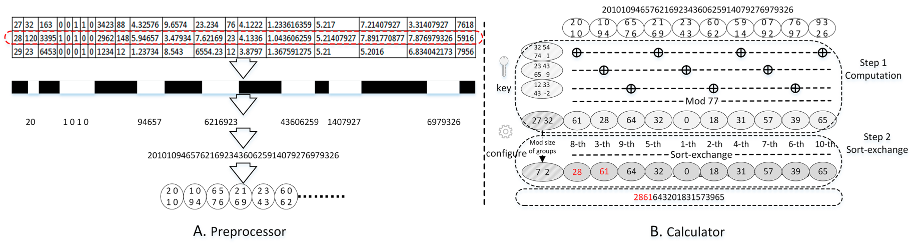

Fig. 2. The schematic diagram of Preprocessor and Calculator.

4. Indexer and Modifier. Receiving the watermark from Calculator, Modifier continues the work to change the database. The Algorithm 2 shows Modifier process. Firstly, algorithm measures the usable length of attributes. Then algorithm takes substrings of watermark fill back into the tuple in turn on the basis of usable length Pos_w and actual length of attributes. It plays a role in producing and executing SQL for embedding on database. Indexer stores/extracts the record of watermarks.

The watermarking process is shown in algorithm 3. In embedding phase, the algorithm traverses the whole database. Because A_{11} requires comparing the distance between the current tuple and the most recent same one. In every interval I, algorithm generates a random number between 0 and I. This random number picks a tuple to embed watermark. After Preprocessor step, in line 8–13, watermark will be generated by Calculator. Finally, modification of tuple is transformed to SQL for executing on D. In Fig. 3 A, the whole process is presented.

4.2 Watermark Detection and Recovery

In the watermark detection process, the first step is to locate the watermarks. The scheme uses adaptable data structure HashMap for matching extracted watermarks. Because the same watermark exists but has different affiliated feature A. Thus, the key of HashMap is watermarks, and the value is a list of different A_i. This process contains only Preprocessor and Calculator components. Because detection is a traverse process on database without modifying the original content. After extracting, based on the result of matching, detection compute the score function by predefined weight using Eq. 1. For example, when watermark $W^{'}$ and affiliated A are both matched, the watermark is 100% confidence. If A can be matched partially, the watermark has the confidence of the corresponding part. If only the watermark is detected and no attached features are fond, confidence is low. Our scheme defines High, Middle and Low scores. The detection process is presented in Fig. 3B.

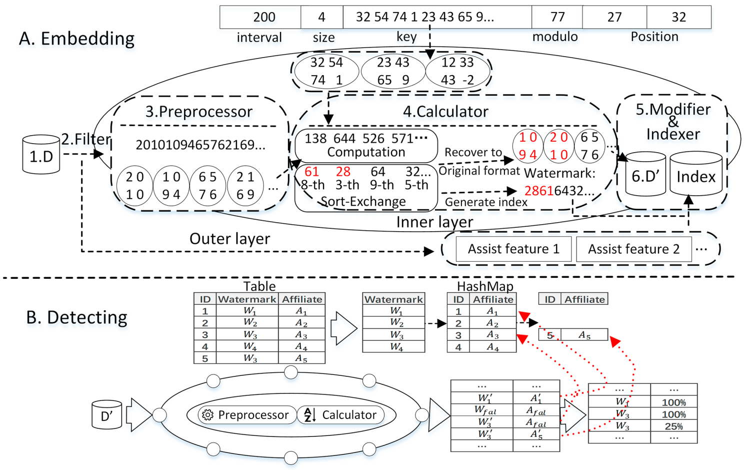

Fig. 3. The process of Detection.

The detection algorithm is presented in Algorithm 4. The detection mainly consists of three steps: 1. Scheme extracts the index of watermarks saved in the database as matching reference. 2. After building hashMap data structure, scheme traverses all tuples to calculate watermark W_i for a preliminary verification. If W_i can't be match with hashMap of matching reference, program will move on to the next one. If W_i can be matched on map_W, the extracted affiliated feature A is compared with every items using Eq. 1 in the list of $HashMap(W)$ to get the highest score. Finally, comparing with τ, the program returns a deterministic result. In our scheme, the assigned value of τ is 50%. Whether to perform a recovery operation is optional. And this action only works on the tuples of which are 100% confidence of detection. If user chooses to runs this operation, based on the detection, one more component (Recover) is added. The function of Recover is similar to Modifier that executes SQL for recovering the embedded tuples.

5 Experiments and Discussion

Experiments are conducted on Intel Core i7 with CPU of 3.60 GHz and RAM of 16 GB. Algorithms were implemented on Postgresql11.0 with JDK 1.7. Experiments were performed using the Forest Cover Type data set, downloaded from Archive.[1] The data set has 581,012 rows with 61 attributes. Each tuple contains 10 integer attributes, 1 categorical attribute, and 44 Boolean attributes. We added an extra attribute ID as the primary key, eight attributes of float type, eight attributes of double type. The data set has four type: boolean, integer, float, double. We chose 7 integer attributes, 16 float/double attributes, and 2 boolean attributes as candidates for watermarking. The experiments were repeated 15 times and the average of result for each trial was calculated.

5.1 Overhead

Equation 4 explains the total computational time of watermarking. The R/I represents the number of embedded tuples, $(|W_i|/l_g)^2$ is the time of single embedding/detection. $|W_i|$ is the length of watermark W_i. l_g is the number of digits in a group. So $|W_i|/l_g$ is equal to the number of groups. Because Sort-Exchange operation needs Quick-Sort algorithm, execution time is $(|W_i|/l_g)^2$. For every embedded tuple are calculated and added together as $Time(D_w)$.

$$Time(D_w) = (R/I) * (|W_i|/l_g)^2 \tag{4}$$

There are three experiments for assessing the computational cost of embedding, detection and recover. In Fig. 4, two aspects (tuple number and interval) of the execution times are shown. In Fig. 4(a), figure shows the computational cost of embedding, detection, recover for different number of tuples with interval $I = 200$. The range of tuple numbers is 100000 to 581012. The time consumption increases along with the number of embedded tuples. As the number of watermarks increases, the trend is not a straight line but a gradually steep curve because of $(|W_i|/l_g)^2$ operation. In Fig. 4(b), figure displays the effect of Interval I for executing time. The larger the interval, the less time it takes because the number of embedded tuples becomes less. As value of interval goes above 500, the gradient of time curve decreases because of watermarks decreases. Thus, the batch size of processing tuples should not be large, which results in a disproportionate increase in time consumption. These results shows that our scheme perform well enough to be used in off-line processing.

5.2 False Hit Rate

Watermarking techniques may detect message of ownership from un-embedded database without correct key. In our scheme, detected watermarks without correct affiliated feature dose not affect score because of score *Low* has weight 0.

[1] kdd.ics.uci.edu/databases/covertype/covertype.html.

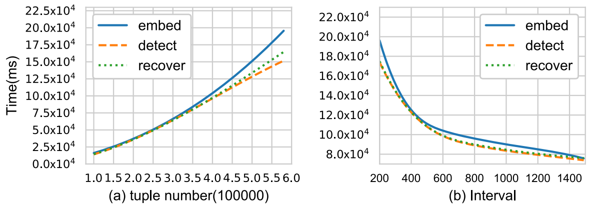

Fig. 4. The computation time of watermarking.

Therefore, only when watermark and affiliated feature are detected correctly can scores be obtained. In Table 2. We represents the experiments in different parameters. D_1, D_2, D_3 are test databases that embedded watermarks with parameter I of $200, 500, 1000$ and independent key. SCDK is the abbreviation of Same Configure but Different Key. And DCDK is the abbreviation of Different Configure and Different Key. All the tests scored low which is in line with expectations. The difference between SCDK and DCDK can be neglected. As parameter I increases, the error rate decreases slightly because the total number of watermarks is reduced at the same time. In practice, the False Hit Rate can be completely ignored in our watermarking techniques.

Table 2. False hit rate experiments

Database		D_1	D_2	D_3
I		200	500	1000
SCDK	Suc	N	N	N
	Score	1.093%	1.054%	0.861%
DCDK	Suc	N	N	N
	Score	1.452%	0.813%	0.807%

5.3 Effect on Statistics

We evaluated statistical distortion through the variations of mean and standard deviation on embedded database. Experiments were performed on independent database with parameters $I = 200, 500, 1000$ and $Pos_w = 18, 24, 36$. The 24 attributes are selected as candidates for embedding. We choose 6 attributes of different numeric data type for presenting distortion. The type of $a1$ and $a2$ is a three-digit and four-digit integer respectively. The $a3$ is a boolean. The $a4$,$a5$ are float type, $a6$ is double. In Table 3, table shows the variation rate of Mean and Std deviation between watermarked data and original data. The variation in attributes $a1$, $a2$ was tiny in all cases. But in $a3$, the fluctuation increases significantly because of the length of $a3$ is 1. Thus, boolean attributes should not be an option for embedding watermarks. According to the experiments on three attributes of $a1$, $a2$ and $a3$, the larger the length of integer data is, the more suitable it is to embed watermarks. Attributes like $a4$, $a5$, $a6$ are the best candidates for watermarking. Embedding has almost no statistical effect on $a4$, $a5$, $a6$. In summary, if watermark is suitably embedded in the last few digits, which has little impact on the individual and the overall statistics for almost all type attributes.

The measures of Mean and Std decrease with the number of interval I but the change is not significant. Thus, we adjust the parameter I as needed for practical application rather than reduce statistical perturbations. With increase of Pos_w, perturbations decreased significantly. But because the perturbations are so small, sometimes there's a little bit of anomaly. Perturbation increased with a maximum value in attribute $a4$ at $I = 200$ and Pos_w.

5.4 Incremental Updates and Adaptation

Incremental updates is an important feature for watermarking technique. The data is continuously produced in many environments. The watermark depends only on groups N_g extracted from scattered attributes and secret key. Thus a tuple can be deleted, inserted without examining or altering any other tuples. When updating candidates attributes of embedding, we recompute current tuple using Calculator component and update new watermark index in the database. When updating non-candidates attribute, nothing need be done. Our scheme defines a parameter I that controls the interval of each embedding. When the amount of incremental data is much larger than I, it won't be any problems. However, if the amount of each incremental update is less than I, a batch may not be embedded with a watermark which will lead to the decrease of watermarks R/I. Thus, the basic requirement is that batch size of increments should be greater than I.

The data is usually stored on multiple nodes, which organization and management is not in a single server. Therefore, the watermarking techniques adapted to the application need to meet data cutting or consolidation problems. Our scheme relys on the watermarks to locate. Scheme needs to share the index records generated by the watermark in the storage node to the detection server, which can use a single key to complete the ownership identification for different databases. In Table 4. We use a secret key to detect data that is fused across multiple database sources. First column is the number of database from multiple data sources. The second column is R/I that the right number of total watermarks. Then next 3 columns are detection results of High/Middle/Low score. The column of Suc is Y or N which represents the successful or failed detection. And column of Score is the detection score, Recover is the percentage of recovered watermarks. The result shows that Multiple databases embedded separately which do not affect accuracy and recovery for detection with the same key. In Table 5. We tested the ability of detection for partial data. The first column is the split ratio on a original embedded database. The results shows that the accuracy and recovery rates were impressive in this case. The result represents our watermarking scheme has ability of facing partial and consolidated data for adaptation.

Table 3. Statical distortion experiments

I		200			500			1000		
Pos_w		18	24	36	18	24	36	18	24	36
Mean.	a1	0.001202744%	0.000647662%	0.000122770%	0.000266471%	0.000194822%	0.000054743%	0.000286597%	0.000130015%	0.000011270%
	a2	0.000025436%	0.000011588%	0.000017904%	0.000002723%	0.000005359%	0.000005736%	0.000003215%	0.000001680%	0.000003273%
	a3	3.508439731%	3.512046454%	1.509413546%	2.303794272%	1.280386640%	0.865613503%	1.028817716%	1.082918560%	0.516663059%
	a4	0.000062976%	0.000015910%	0.000018888%	0.000009228%	0.000008995%	0.000009036%	0.000020242%	0.000018056%	0.000002479%
	a5	0.000045576%	0.000004733%	0.000005368%	0.000033804%	0.000010773%	0.000032949%	0.000021699%	0.000005888%	0.000001224%
	a6	0.000000459%	0.000000952%	0.000000063%	0.000000015%	0.000000001%	0.000000049%	0.000000086%	0.000000005%	0.000000012%
Std.	a1	0.000601960%	0.000515174%	0.000537451%	0.000332161%	0.000161452%	0.000115328%	0.000150423%	0.000001484%	0.000098659%
	a2	0.000015376%	0.000002622%	0.000017030%	0.000003017%	0.000003500%	0.000001998%	0.000000007%	0.000004535%	0.000000061%
	a3	11.024910177%	11.285514891%	4.953004830%	7.338794350%	4.280060337%	2.923639995%	3.371868926%	3.638831796%	1.718041449%
	a4	0.000021213%	0.000027466%	0.000001033%	0.000027849%	0.000049301%	0.000018759%	0.000036018%	0.000042024%	0.000005003%
	a5	0.000021884%	0.000021995%	0.000025646%	0.000007663%	0.000001051%	0.000051026%	0.000013529%	0.000002588%	0.000038540%
	a6	0.000000770%	0.000002712%	0.000000057%	0.000000005%	0.000000000%	0.000000305%	0.000000056%	0.000000003%	0.000000039%

Table 4. Detection for consolidated data

databases	R/I	RESULT			Suc	Score	Recover
		High	Mid	Low			
2	5810	5810	0	0	Y	100%	100%
3	8715	8715	0	0	Y	100%	100%
4	11620	11620	0	0	Y	100%	100%
5	14525	14525	0	0	Y	100%	100%

Table 5. Detection for partial data

%	R/I	RESULT			Suc	Score	Recover
		High	Mid	Low			
10%	290	289–291	0	0	Y	100%	100%
20%	581	580–582	0	0	Y	100%	100%
50%	1452	1451–1453	0	0	Y	100%	100%
70%	2033	2032–2034	0	0	Y	100%	100%
90%	2614	2613–2615	0	0	Y	100%	100%

6 Robustness

In this section, we experiment the robustness of our watermarking scheme under three attacks. Mallory will try their best to remove watermarks to disturb detection. The robustness requirement should be higher to adapt to more rigorous environment. For example, after stealing order data of market, Mallory deletes sensitive information such as order id (most likely primary key), name, phone number, and keeps the non-sensitive valuable information. Our scheme embeds watermarks on non-significant data for locating and embedding. Thus, under a more severe condition, our scheme has a stronger ability of robustness. Attacks are conducted in two situations: 1. Normal situation which refers to an attack test under original watermarked database. 2. Hard situation where key information is deleted in experiments. Database D was watermarked using different parameter I and ratios of influence for attacks. Figures presented in this section have score on vertical axis, which represents the scores of detection using Eq. 1.

6.1 Deletion Attacks

Mallory can randomly delete subset of watermarked tuples from database D_w. Figure 5(a)(c) shows the experiment result with parameter $I = 200, 500, 1000$ by randomly deleting different ratios of tuples from D_w in normal situation and hard situation. The detection threshold is $\tau = 50\%$ with red dotted line in (a)(c). Score is greater than the τ which has a successful detection. In Fig. 5(a), even when up to 90% of the tuples are deleted, the detection is successful. The Fig. 5 (b) shows the proportions of the high/Middle/Low scores, with the percentage of high scores being 100%. In Fig. 5(c), suppose that we obtained a file containing suspected part of our business data. But the order id and sensitive attributes (e.g. name, age, address) were deleted. It still has semi-structured or structured formats that can separate data. In this hard case, our watermarking method still works. When the deletion ratio is less than 50% with $I = 200, 500, 1000$, the detection is successful. In Fig. 5(d), the proportion of Middle score increases first and then decreases. And the ratio of high score keeps going down but low score increases. When the ratio of deletion is more than 50%, almost all of mutual relation for tuples are destroyed which converts High/Middle score to Low score.

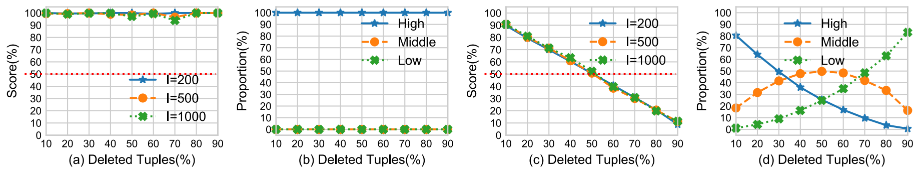

Fig. 5. The experiments of deletion attacks

6.2 Insertion Attacks

The source of new tuples for insertion is original database or similar fake data. The experiments didn't disturb the original tuples of database D_w. The degree of insertion ranges from 10% to 250%. Since the effect is consistent in both normal situation and hard situation, only two diagrams are shown. In Fig. 6(a), when Mallory tries to insert more than 100% tuples, the detection can't sure who owns database because the result is smaller than value of τ. The curves with different I follow a similar trend in figure. In experiments, the value of τ is 50%. But 50% is not a fixed value. If obtaining a large number of watermarks, detection can claim ownership. If detecting 10,000 watermarks in a database with tens of thousands of tuples, owner may still claims ownership even if the ratio of watermarks is less than τ. In Fig. 6(b), the radio of Middle is 0%. The ratio of Low goes up, the ratio of High goes down with increase of insertion radio. Although the new tuples are replica of the original tuples, there is only a watermark W_i with Low score.

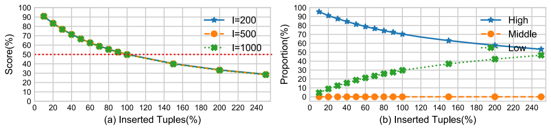

Fig. 6. The experiments of insertion attacks.

6.3 Alteration Attacks

Mallory changes the value of tuples on database D_w. Mallory randomly selects 1 to 5 attributes and modify their values. The experiments assume that the attacker dose not know the real positions where the watermarks was embedded. Figure 7(a) shows the result of attacks under different ratio of alteration and parameter $I = 200, 500, 1000$ in normal situation. All results of experiment are higher than τ which represents successful detections for ownership identification. Figure 7(b) shows that alteration can disrupt the watermark locating by decreasing the amount of watermarks in normal situation. Figure 7(c) shows that the result curve drops more steeply because of lacking of key information for verification of watermarks in hard situation. Even the ratio of alteration is up to 100%, all detection are successful. In Fig. 7(d), the proportion of High score keeps going down over increase of altered tuples. And the ratio of Middle/Low keeps going up.

Table 6 shows a comparison among our scheme and Sion's [18] technique and DEW [5] for robustness in normal situation. Our scheme is highly robust as compared to Sion's and DEW techniques in three types of attacks.

Table 6. Comparison among DEM and other techniques

	Our technique	Sion's technique	DEW
Deletion attack	Resilient to random tuple insertion attacks; 95% watermark score even when 90% of the tuples are deleted	Not resilient to random tuple deletion attack; watermark accuracy deteriorates to 50% when only 10% of the tuples are deleted	Not resilient to random tuple deletion attack; watermark accuracy deteriorates to 50% when only 50% of the tuples are deleted.
Insertion attack	Resilient to random tuple insertion attacks; 50% watermark score even when 200% of the tuples are inserted	Not resilient to random tuple insertion attacks; watermark accuracy deteriorates to 50% when only 10% of the tuples are inserted	Resilient to random tuple insertion attacks; 85% watermark accuracy even when 50% of the tuples are inserted.
Alteration attack	Resilient to random tuple Alteration attacks; 70% watermark score even when 90% of the tuples are altered		Resilient to random tuple Alteration attacks; 90% watermark accuracy even when 50% of the tuples are altered

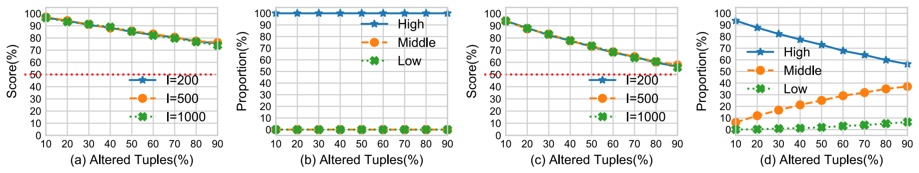

Fig. 7. The experiments of alteration attacks.

7 Conclusion

In this paper, we defined a new method of watermarking and designed a general model DEM. A novel robust and semi-blind reversible watermarking scheme on numerical attributes was proposed. Experiments presented the robustness under three attacks and excellent properties in many aspects such as implementation, false hit rate, incremental updates, statistic distortion and adaptation.

Acknowledgments. This work has been supported by National Key R&D Program of China (No.2017YFC0820700), National Science and Technology Major Project (No.2016ZX05047003), the Beijing Municipal Science & Technology Commission Funds for Cyberspace Security Defense Theory and Key Technology Project (No.Z191100007119003).

References

1. Agrawal, R., Kiernan, J.: Watermarking relational databases. In: Very Large Data Bases, pp. 155–166 (2002)
2. Fei, G., Wang, J., Li, D.: Fingerprinting relational databases. In: ACM Symposium on Applied Computing (2006)
3. Franco-Contreras, J., Coatrieux, G.: Robust watermarking of relational databases with ontology-guided distortion control. IEEE Trans. Inf. Forensics Secur. **10**(9), 1939–1952 (2015)
4. Guo, H., Li, Y., Liu, A., Jajodia, S.: A fragile watermarking scheme for detecting malicious modifications of database relations. Inf. Sci. Int. J. **176**(10), 1350–1378 (2006)
5. Gupta, G., Pieprzyk, J.: Reversible and blind database watermarking using difference expansion. Int. J. Digit. Crime Forensics **1**(2), 42–54 (2009)
6. Hartung, F., Kutter, M.: Multimedia watermarking techniques. Proc. IEEE **87**(7), 1079–1107 (1999)
7. Iftikhar, S., Kamran, M., Anwar, Z.: RRW a robust and reversible watermarking technique for relational data. IEEE Trans. Knowl. Data Eng. **27**(4), 1132–1145 (2015)
8. Kamel, I.: A schema for protecting the integrity of databases. Comput. Secur. **28**(7), 698–709 (2009)
9. Kamran, M., Farooq, M.: A comprehensive survey of watermarking relational databases research. arXiv preprint arXiv:1801.08271 (2018)
10. Khan, A., Husain, S.A.: A fragile zero watermarking scheme to detect and characterize malicious modifications in database relations. Sci. World J. **2013**, 796726 (2013)
11. Kim, Y.W., Moon, K.A., Oh, I.S.: A text watermarking algorithm based on word classification and inter-word space statistics. In: International Conference on Document Analysis & Recognition (2003)
12. Li, Y., Deng, R.H.: Publicly verifiable ownership protection for relational databases. In: ACM Symposium on Information (2006)
13. Li, Y., Guo, H., Jajodia, S.: Tamper detection and localization for categorical data using fragile watermarks. In: ACM Workshop on Digital Rights Management (2004)
14. Podilchuk, C.I., Zeng, W.: Image-adaptive watermarking using visual models. IEEE J. Sel. A. Commun. **16**(4), 525–539 (2006). https://doi.org/10.1109/49.668975
15. Rakesh, A., Peter, H., Jerry, K.: Watermarking relational data: framework, algorithms and analysis. VLDB **12**, 157–169 (2003). https://doi.org/10.1007/s00778-003-0097-x
16. Shehab, M., Bertino, E., Ghafoor, A.: watermarking relational databases using optimization-based techniques. IEEE Trans. Knowl. Data Eng. **20**(1), 116–129 (2007)

17. Sion, R., Atallah, M., Prabhakar, S.: Rights protection for categorical data. IEEE Trans. Knowl. Data Eng. **17**(7), 912–926 (2005)
18. Sion, R., Atallah, M., Prabhakar, S.: Rights protection for relational data. IEEE Trans. Knowl. Data Eng. **16**(12), 1509–1525 (2004)
19. Sion, R.: Proving ownership over categorical data (2004)
20. Stern, J.P., Hachez, G., Koeune, F., Quisquater, J.-J.: Robust object watermarking: application to code. In: Pfitzmann, A. (ed.) IH 1999. LNCS, vol. 1768, pp. 368–378. Springer, Heidelberg (2000). https://doi.org/10.1007/10719724_25
21. Wang, H., Cui, X., Cao, Z.: A speech based algorithm for watermarking relational databases. In: International Symposiums on Information Processing (2008)
22. Cui, X., Qin, X., Sheng, G.: A weighted algorithm for watermarking relational databases. Wuhan Univ. J. Nat. Sci. **12**(1), 79–82 (2007). https://doi.org/10.1007/s11859-006-0204-0
23. Zhou, X., Huang, M., Peng, Z.: An additive-attack-proof watermarking mechanism for databases' copyrights protection using image. In: ACM Symposium on Applied Computing (2007)