Digital Communications and Networks 10 (2024) 1851–1863

Contents lists available at ScienceDirect

Digital Communications and Networks

journal homepage: www.keaipublishing.com/dcan

Consistency preserving database watermarking algorithm for decision trees

Qianwen Li [a], Xiang Wang [b,*], Qingqi Pei [c], Xiaohua Chen [d], Kwok-Yan Lam [e]

[a] School of Telecommunications Engineering, Xidian University, Xi'an, 710071, China
[b] School of Cyber Engineering, Xidian University, Xi'an, 710071, China
[c] School of Telecommunications Engineering, Xidian University, Xi'an, China
[d] 2012 Laboratories, Huawei Technologies, Huawei Technology Co., Ltd, Hangzhou, 310000, China
[e] The School of Computer Science and Engineering, Nanyang Technological University, 639798, Singapore

ARTICLE INFO

Keywords:
Consistency preserving
Decision tree
Database watermarking
Data mining

ABSTRACT

Database watermarking technologies provide an effective solution to data security problems by embedding the watermark in the database to prove copyright or trace the source of data leakage. However, when the watermarked database is used for data mining model building, such as decision trees, it may cause a different mining result in comparison with the result from the original database caused by the distortion of watermark embedding. Traditional watermarking algorithms mainly consider the statistical distortion of data, such as the mean square error, but very few consider the effect of the watermark on database mining. Therefore, in this paper, a consistency preserving database watermarking algorithm is proposed for decision trees. First, label classification statistics and label state transfer methods are proposed to adjust the watermarked data so that the model structure of the watermarked decision tree is the same as that of the original decision tree. Then, the splitting values of the decision tree are adjusted according to the defined constraint equations. Finally, the adjusted database can obtain a decision tree consistent with the original decision tree. The experimental results demonstrated that the proposed algorithm does not corrupt the watermarks, and makes the watermarked decision tree consistent with the original decision tree with a small distortion.

1. Introduction

As the main carrier of data storage, a database plays an important role in data analysis and mining, such as regression analysis, decision trees, and neural networks [1]. However, because a database often stores sensitive and confidential information, its widespread use results in new data security challenges [2–4]. For example, if unauthorized leakage occurs during database sharing, this can pose a great threat to the information security of a government, as well as the public [5–7]. Database watermarking technology has emerged in this scenario, and can achieve copyright identification, traceability, and content authentication of the database by embedding specific watermark information in the carrier data [8].

Database watermarking was proposed by Agrawal and Kiernan [9] in 2002, who modified the least significant bits (LSB) of the tuple attribute value to achieve the watermark imperceptibly. Database watermarks can be divided into two categories according to whether watermark embedding causes data distortion: distortion-based and distortion-free methods [10,11]. In distortion-free algorithms [12–18], watermark information is usually embedded by changing the order of database tuples. Watermark information can be correctly extracted while keeping the order of data tuples unchanged. However, when there are operations, such as element deletion and addition, it is difficult to extract watermark information correctly. Moreover, watermark information is related to the way in which data are stored, which limits its application to a great extent. Distortion-free watermarking is less useable in practical applications because of its fragility. In a distortion-based database watermarking algorithm [19–27], the most important challenge in watermark embedding is how to preserve knowledge stored in features or attributes. Datasets are often used for different purposes, such as statistical characterization and data mining analysis. Therefore, watermarking algorithms should consider not only the robustness of the watermark but also the availability constraints of data.

Regardless of whether a database watermarking algorithm is distortion-free or distortion-based, it is concerned with the statistical distortion of data and whether semantic information is preserved, and

* Corresponding author.
E-mail addresses: liqw880@gmail.com (Q. Li), wangxiang@xidian.edu.cn (X. Wang), qqpei@mail.xidian.edu.cn (Q. Pei), chenxiaohua1@huawei.com (X. Chen), kwokyan.lam@ntu.edu.sg (K.-Y. Lam).

https://doi.org/10.1016/j.dcan.2022.12.015
Received 6 June 2022; Received in revised form 15 November 2022; Accepted 26 December 2022
Available online 16 January 2023
2352-8648/© 2024 Chongqing University of Posts and Telecommunications. Production and hosting by Elsevier B.V. on behalf of KeAi Communications Co. Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

does not consider the consistency of watermarked data when used in data mining models. For data mining algorithms, some minor changes in the data can make the training model parameters change significantly. A classification and regression tree (CART) [28] is an easy to understand and implement classification algorithm used in data mining algorithms that can effectively classify and predict unknown data by constructing a decision tree using existing data, which has the advantages of good readability and high efficiency. Compared with other data mining algorithms, a decision tree algorithm can classify data from large data sources in a shorter time and can process unrelated feature data. Therefore, in this study, we used original data and watermarked data to construct CART decision trees separately, and analyzed the influence of watermarking on the construction of decision tree models. We found that the decision tree model was greatly changed by the embedding of the watermark, and the consistency of the decision tree was destroyed compared with the original decision tree, which resulted in the construction of a model that could not be used for unknown data prediction. Although traditional database watermarking algorithms effectively solve the problem of copyright authentication and tracking after data leakage, the small distortion caused by watermark embedding can result in the loss of the preservation of the consistency of the decision tree. Therefore, we propose a consistency preserving database watermarking algorithm for decision trees. Unlike traditional watermarking algorithms that only consider the statistical distortion of data, the proposed algorithm not only preserves the performance of the watermarking algorithm but also ensures that the consistency of watermarked data is not destroyed when constructing decision trees after embedding the watermark. The main contributions of this paper are summarized as follows:

1) In this paper, we analyze the impact of traditional watermarking algorithms for decision tree model construction, and propose a consistency preserving database watermarking algorithm. The algorithm not only solves problems such as copyright authentication in data sharing and transactions but also ensures the consistency of data when used for decision tree model construction.
2) To make the structure of the watermarked decision tree consistent with the original decision tree, we propose label classification statistics (LCS) and label state shifting methods to adjust the structure of the watermarked decision tree. Using the LCS of the original decision tree as the basis, we adjust the watermarked data to accomplish label state shifting using the method of split value mapping and data shifting so that the watermarked data can produce a decision tree with the same structure as the original data.
3) To make the split values of the watermarked decision tree the same as the original data, we divide the split values into independent and continuous values according to their position status, and adjust the data using defined constraint equations so that the split values of the watermarked decision tree are consistent with those of the original decision tree.
4) The proposed database watermarking algorithm can be effectively combined with other quantization-based watermarking algorithms. It has universal applicability.

The remainder of this paper is organized as follows: In Section 2, we present the CART classification algorithm and LSB watermarking method. We provide the motivation for the approach and an overview in Section 3. In Section 4, we introduce consistent preserving methods for the watermarked decision tree and discuss the consistency preserving theory analysis in Section 5. In Section 6, we analyze and discuss the experimental results. We present a summary of this paper and future research in Section 7.

2. Preliminaries

2.1. CART classification algorithm

The CART algorithm was proposed by Breiman et al. [23]. The core idea of this algorithm is to use the concept of bipartite division to divide a dataset into two samples recursively so that the obtained tree is a binary tree. The CART algorithm uses the *Gini* coefficient for feature selection, which is the selection of split nodes. The smaller the *Gini* value, the better the feature. In our study only embedding watermark information in the continuous numerical attributes of a database is considered.

Given database data $D = (K_v, A, C)$, where K_v represents the primary key of the tuple, $A = \{A_1, A_2, \ldots, A_n\}$ is the feature attribute, C is the label attribute, and $|\mathscr{Y}|$ is the number of labels in the database, $p_k (k = 1, 2, \ldots, |\mathscr{Y}|)$ represents the probability of the k-th label in D. The *Gini* value of data D can be calculated using the following formula:

$$Gini(D) = 1 - \sum_{k=1}^{|\mathscr{Y}|} p_k^2 \tag{1}$$

where $\sum_{k=1}^{|\mathscr{Y}|} p_k = 1$. If there is only one label in D, the *Gini* value is 0, which indicates that the purity of the data label is very high.

When constructing a decision tree, to increase the purity of data divided by the split value, it is necessary to select an optimal feature for each division. For an attribute $A_i \in A$, assume that A_i has n different continuous values (d_1, d_2, …, d_n). First, arrange the n continuous values in ascending order and calculate a split value between two different values so that $(n-1)$ split values are generated. Next, use the split value to divide the continuous value discretely into two datasets denoted by D_1 and D_2. The *Gini* value after dividing the data using the split value is:

$$Gini_{A_i,j}(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2) \tag{2}$$

where $Gini_{A_i,j}$ represents the *Gini* value when the j-th split value of attribute A_i divides the data.

The minimum *Gini* for each attribute A_i is:

$$\min_j \left(Gini_{A_i,j}(D) \right) \tag{3}$$

where $j = 1, \ldots, n-1$, i is i-th feature attribute in attribute set A. Hence, the attribute with the smallest *Gini* is selected as the optimal partition attribute:

$$\arg\min_{A_i \in A} \left(\min_j \left(Gini_{A_i,j}(D) \right) \right) \tag{4}$$

Using the above steps, D is divided iteratively until it cannot be divided. There are two cases in which the data cannot be divided: (1) the label type in the data segment is consistent, that is, the *Gini* value is equal to 0; and (2) all attributes of the data have the same value.

2.2. LSB watermarking algorithm

The LSB watermarking algorithm has the advantages of large embedding capacity, fast embedding speed, and low data distortion; hence, most existing database watermarking algorithms are based on LSB [5,24–27].

The LSB watermark embedding algorithm embeds watermark information by replacing the LSB of the carrier data with watermark information $W = (w_1, w_2, \ldots, w_n)$. We assume that the element (d_1, d_2, …, d_n) of the feature attribute in D is the carrier data. LSB watermark embedding can be achieved using the following formula:

$$d_i^{'} = 2\lfloor d_i/2 \rfloor + w_i \tag{5}$$

where $\lfloor \bullet \rfloor$ represents the lower rounding function. Using the above

Original data

Primary key	Weight	Diabetes
1	60.43	NO
2	61.54	NO
3	70.53	NO
4	71.32	Yes
5	72.18	NO
6	80.42	Yes
7	80.58	NO
8	81.47	Yes

Watermark

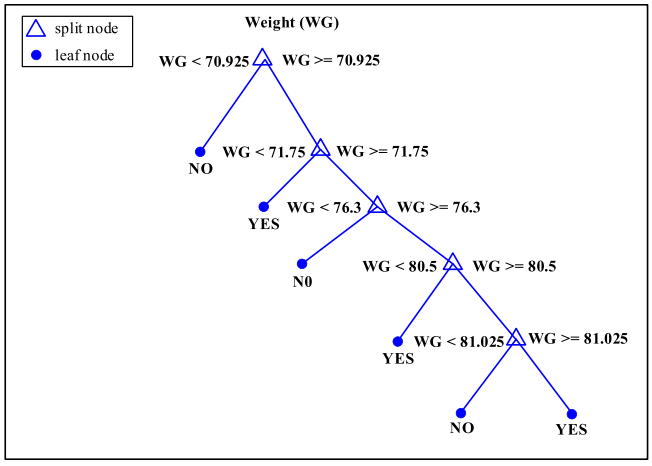
W
0
1
1
1
0
0
0
0

Embedding data

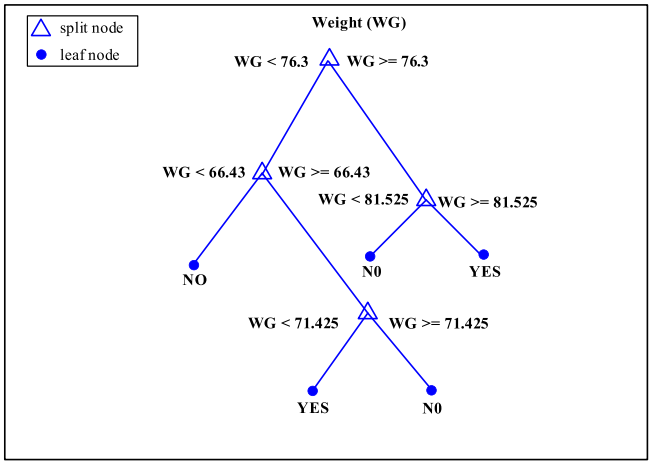
Watermarking data

Primary key	Weight	Diabetes
1	60.43	NO
2	61.54	NO
4	71.32	Yes
3	71.53	NO
5	72.18	NO
6	80.42	Yes
8	81.47	Yes
7	81.58	NO

Fig. 1. Embed watermark information through LSB watermark algorithm.



(a) Decision tree of Original data



(b) Decision tree of Watermarked data

Fig. 2. CART decision trees for original data and watermarked data.

formula, watermark data $D_w = (d_1^{'}, d_2^{'}, \ldots, d_n^{'})$ can be obtained after watermark information is embedded. For example, if the original data are denoted by $d_i = 51$ and watermark information $w_i = 0$ is to be embedded, then watermarked data denoted by $d_i^{'} = 50$ can be obtained. The watermark extraction process is the reverse process of embedding, and watermark w_i can be obtained using the following formula:

$$w_i = d_i^{'} \bmod (2) \tag{6}$$

where mod (•) is a modular operation.

3. Approach motivation and overview

3.1. Motivation

Traditional database watermarking algorithms minimize data distortion by optimizing the watermark embedding algorithm and setting the constraint function. However, the minimized distortion can also destroy the consistency of the decision tree model. Traditional watermarking algorithms are no longer applicable to existing data mining techniques. We illustrate the influence of watermark embedding on decision tree model construction with an example. The LSB watermarking algorithm is used to embed the watermark in data, and we present the change of the decision tree before and after the watermark is embedded. As shown in Fig. 1, the primary key of the tuple, the weight, and whether a user has diabetes is stored in databases. The watermark is embedded in the integer part of the weight data using the LSB algorithm. The decision trees generated by the application of the CART algorithm to the original data and watermarked data are presented in Fig. 2. As shown in Fig. 2(a) and (b), the original data embedded with the watermark generates a different tree from the original data. Thus, if the tree is used to classify unknown data, a different classification result will be obtained.

From this example, we can conclude that the decision tree generated from the watermarked data is completely different from that generated from the original data; hence, the usability of data in decision tree classification is destroyed. Traditional watermark embedding algorithms introduce less distortion to the original data and can effectively trace the source of data leakage when data have been leaked illegally. However, traditional watermarking algorithms do not consider the consistency of the decision tree when data are applied to decision tree construction. Thus, watermark embedding results in meaningless data.

3.2. Overview

The framework of the proposed algorithm is shown in Fig. 3. The specific steps can be summarized as follows:

(1) Watermark embedding. First, encode the watermark using the Bose–Chaudhuri–Hocquenghem (BCH) codes for error correction. Then, embed it into different groups according to the traditional LSB watermarking algorithm to obtain the watermarked data.
(2) Lable Classification Statistics (LCS). Count the number of labels in the original data segment where the split values are located, and prestore the statistical results of the lables and split values.
(3) Label State Transition (LST). Iteratively adjust the watermarked data. First, locate the original split value in the watermarked data. Then, adjust the label state of the split value in the watermarked data segment to be consistent with the original state through data shifting.
(4) Split Value Adjustment (SVA). The watermarked tree structure obtained after completing the above two steps is consistent with the original tree structure. Then, divide the split values into two cases: independent split values and continuous split values. Furthermore, adjust the split values by solving constraint equations and shifting data. After the SVA, the split values in the watermarked tree are consistent with those in the original tree.
(5) Watermark Extraction. According to the watermark extraction algorithm, extract the watermark into different groups. Vote for the watermark obtained in the group to obtain the final watermark.

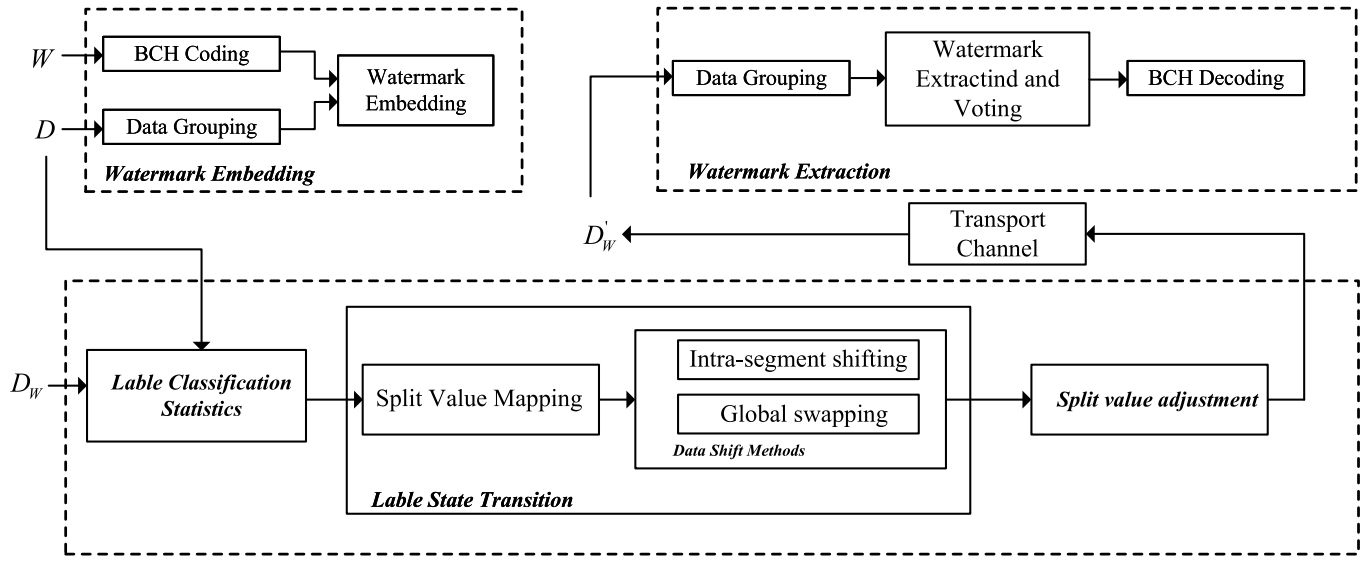


Fig. 3. The framework of the proposed scheme.

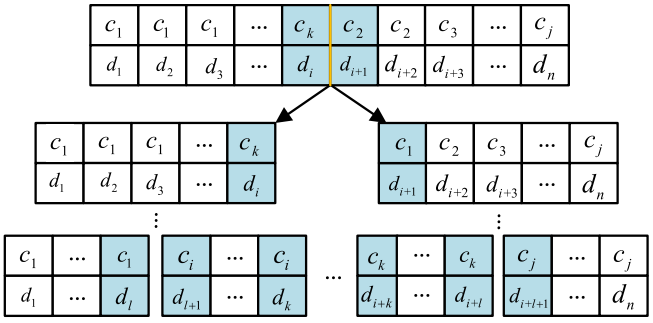


Fig. 4. The process of generating a CART decision tree for two-dimensional data.

4. Proposed method

4.1. Watermark embedding

The watermark embedding steps are as follows:

Step 1. Watermark Generation. Given length l of watermark information, use a pseudo-random function to generate a 0–1 string of length l as the watermark for subsequent embedding.

Step 2. Increase the robustness of the watermark. The BCH error correction code has been commonly used. It represents a type of cyclic code and has strong error correction capability, which is usually represented by (n, l), where n represents the code length after encoding and l represents the length of the watermark. Code length n is related to the length of information bits and the number of error corrections. When information bit l is constant, the longer code length n, the better the error correction capability. For instance, BCH code (31,16) can correct three-bit errors and BCH code (63,16) can correct seven-bit errors.

Step 3. Watermark embedding. First, use the *hash* function to group the data according to the primary key of the tuple:

$$t = H(k \| r_{PK}) \bmod l \tag{7}$$

where k is the given key, r_{PK} is the primary key value of the tuple, $\|$ represents a parallel connection, and H is the *hash* function. We use the MD5 hash function in this paper. We consider the number of groups equal to l. After the tuple in the database is finished grouping, we embed watermark information in different groups in turn according to Eq. (5) and embed the same watermark information into data in the same group.

We need to consider two special cases in the watermark embedding process: 1) watermark embedding that causes differences in equal data; and 2) watermark embedding that causes unequal data to become equal. Based on Subsection 2.2, when a decision tree is generated, the selection of a split value requires the calculation of the *Gini* value of two different data elements; hence, whether the data are equal affects the generation process of a decision tree. The following methods are used to solve these two special cases. For 1), the same watermark is embedded into the equal data, which solves the problem of differences between equal data. For 2), once the unequal data become equal, the data are further adjusted to regain the unequal status.

As mentioned previously, after the watermark is embedded, a decision tree is produced by the watermarked data that is inconsistent with the original data. There are three cases in which the two trees are considered consistent: 1) the tree structures are the same; 2) the leaf nodes of the decision trees are the same; and 3) the partition values of the trees are equal. After these three cases are considered, consistency preserving watermarked decision trees can be reconstructed using the LCS, LST, and SVA.

4.2. Label classification statistics

First, a decision tree is generated from the original data using the CART algorithm. Then, the split value of each layer of the classification tree and the numbers of labels on its left and right sides are recorded in P. When database data $D(K_v, A, C)$ generates a CART tree, it is assumed that there are m split values: $S = \{s_1, s_2, \ldots, s_m\}$. The label classification statistical process is performed by recording these m split values and the numbers of labels on their left and right sides in P. Fig. 4 is an example diagram that shows a tree generated from two-dimensional data. If $(d_1, \ldots, d_n)$ are the feature attribute elements, the subscripts in $(d_1, \ldots, d_n)$ indicate the position of the data after sorting in ascending order, and $c_i \in C$ is the label type corresponding to d_i. Thus, the label statistical value $P = \{p_1, p_2, \ldots, p_m\}$, p_i is:

$$p_i = \left[s_i \middle| \left(\sum c_{l1}, \ldots, \sum c_{l\not y} \middle| \sum c_{r1}, \ldots, \sum c_{r\not y} \right) \right] \tag{8}$$

where s_i represents the split value. It is assumed that s_1 is calculated by d_{i+1} and d_i; hence, $s_i = (d_i + d_{i+1})/2$. As shown in Fig. 4, the optimal split

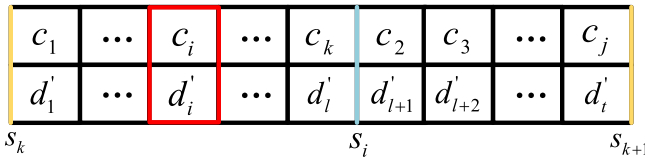


Fig. 5. Intra-segment shifting.

value is chosen according to the *Gini* calculation method in Subsection 2.1. Additionally, the first generated split value is ordered as s_1. $\sum c_{li}$ counts the number of labels c_i in data (d_1, …, d_n) on the left side of s_1. Similarly, $\sum c_{ri}$ counts the number of labels c_i in data (d_1, …, d_n) on the right side of s_1. The blue data in Fig. 4 are used to calculate the split value s_i, and the yellow line denotes the location of s_i. According to s_1, the data are divided into two parts. For the two parts, the split values s_2 and s_3 are calculated. Then, the numbers of labels p_2 and p_3 of the two parts are obtained. The data are divided in sequence until the leaf node of the original data is reached. Finally, the labels on the left and right sides of all the divided values are counted and recorded in P.

4.3. Label state transition

The watermarked data are adjusted using split value mapping and data shifting methods so that the label state of the watermarked data is consistent with that of the original decision tree after using the original split values to split the watermarked data. Then two decision trees are obtained with the same structure. In Subsection 4.3.1, how to map the original split value to the watermark data is explained. In Subsection 4.3.2, how to shift data so that the label state of the split value in the watermark data segment is consistent with the label state of the original data is described.

4.3.1. Split value mapping

First, the feature attributes of the watermarked data segment generated by the i-th split value are arranged in ascending order to obtain $(d_1^{'},\ldots,d_t^{'})$, where t is the number of data elements in the current segment. Then s_i in p_i of the LCS stage is used for the split value mapping. Two values are found in the current watermarked data segment such that:

$$d_j^{'} < s_i < d_{j+1}^{'} \tag{9}$$

where $1 \le j < j+1 \le t$. Fig. 4 shows that all the data are divided into different segments after the split value mapping.

Then the position of the original s_i in the watermarked data is determined to be in the middle of data $d_j^{'}$ and $d_{j+1}^{'}$, which is expressed as $(d_j^{'},d_{j+1}^{'})$. If there are no two data $d_j^{'}$ and $d_{j+1}^{'}$ that satisfy Eq. (11), s_i must belong to the following three special cases: (1) $d_j^{'} \le s_i < d_{j+1}^{'}$, (2) $d_1^{'} > s_i$, and (3) $d_t^{'} \le s_i$. Once these three cases occur, the position of the split value in the watermarked data is determined as $(d_j^{'},d_{j+1}^{'})$, $([],d_1^{'})$, and $(d_t^{'},[])$, where $([],d_1^{'})$ and $(d_t^{'},[])$ indicate the position of the original split value in the watermarked data at the boundary of the data segment. [] represents empty, which indicates that the split value is at the beginning or end of the dataset.

4.3.2. Data shift methods

After the split value mapping, $p_i^{'}$ of the current segment is obtained according to the position of s_i in the watermarked data. Considering the left label adjustment as an example, the difference between the number of left labels in the watermarked data and original data is given by:

$$dc_{li} = \sum c_{li}^{'} - \sum c_{li} \tag{10}$$

where $\sum c_{li}^{'}$ indicates the amount of data labeled with c_i to the left of the split value s_i in the watermarked data. When $dc_{li} = 0$, the data labeled as c_i do not need to be adjusted. If $dc_{li} > 0$, the data labeled as c_i on the left side of the split value need to be adjusted to the right side of the split value, and if $dc_{li} < 0$, the data labeled as c_i need to be adjusted from the right side of the split value to its left side.

In the data shifting process, because of the difference in the data redundancy space in the watermarked data segment, the data shifting methods include intra-segment shifting and global swapping. Intra-segment shifting considers only the data shift in the current data segment, and the shift is simple. When intra-segment shifting cannot guarantee that the label satisfies $dc_{li} = 0$, global swapping is required. Global swapping requires traversing other data segments to identify data that can be swapped, which is a more complex process than intra-segment shifting.

The two data shifting methods are introduced in the following.

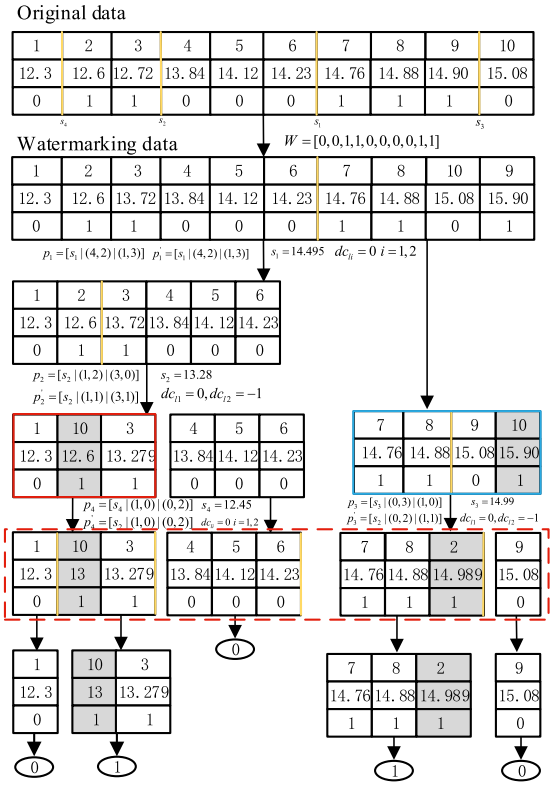


Fig. 7. Watermarked data label state transition.

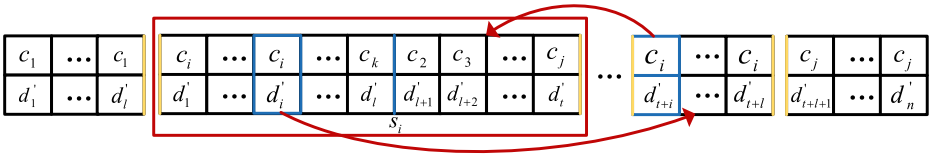
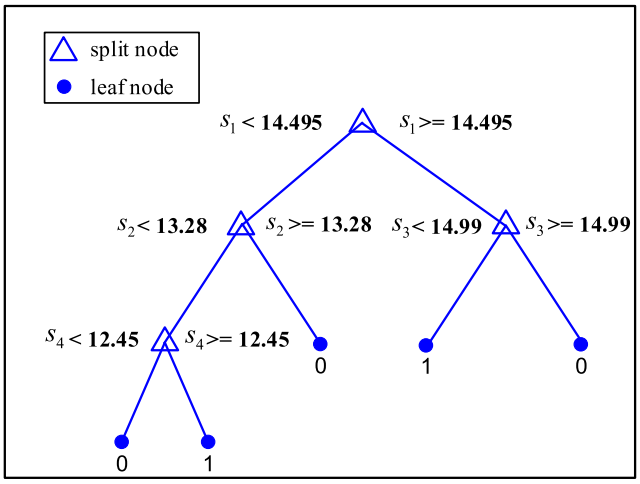
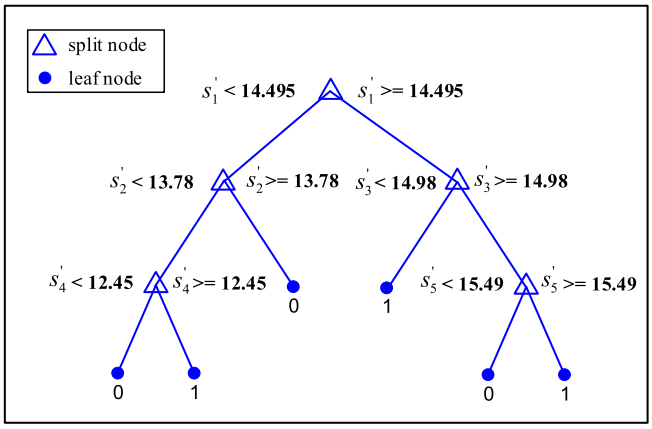


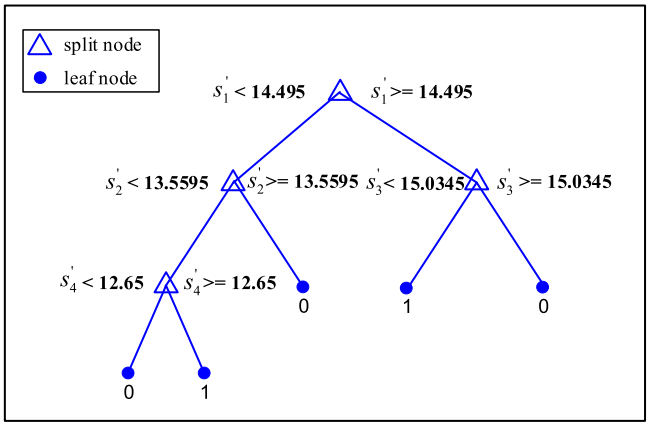
Fig. 6. Global swapping.



(a) Decision tree of original data



(b) Decision tree of watermarked data



(c) Decision tree of watermarked data after LST

Fig. 8. Decision trees of different data.

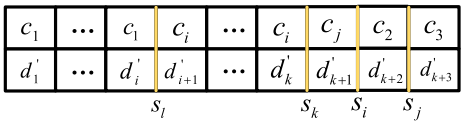


Fig. 9. Different states of the split value.

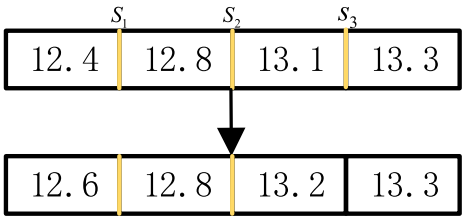


Fig. 10. Continuous split value adjustment.

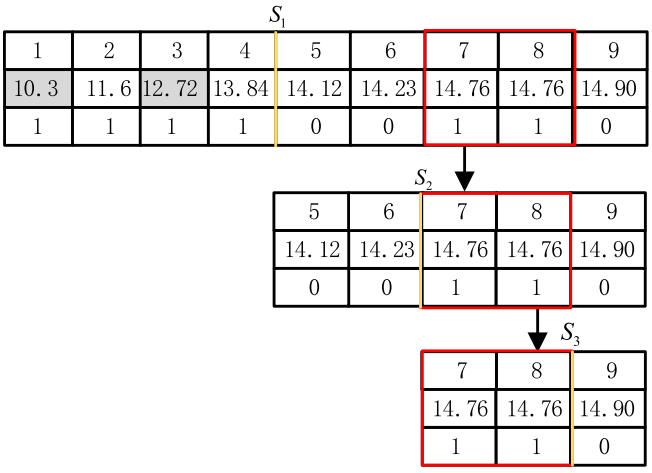


Fig. 11. Label 'false consistent'.

a) Intra-segment shifting

Considering $dc_{li} > 0$ as an example, the number of dc_{li} data tuples with the label c_i should be shifted from the left side of the split value to the right side. When shifting data with the label c_i on the left to the right, the closest data with the label c_i of s_i is selected from the left shifted to the nearest redundant space on the right of s_i to reduce data distortion. An example of intra-segment shifting is shown in Fig. 5. The blue line indicates the mapping position of s_i in the watermarked data and d_i' in the red box denotes data with the label c_i closest to s_i. Assuming that the yellow line in Fig. 5 shows the location of the mapping of the other split values s_k and s_{k+1}, the data adjustment should not exceed the value of these two values, otherwise it will cause the *Gini* values of the split values to change. To reduce distortion when d_i' is modified to the right of s_i, d_i' is modified as follows:

$$d_i'' = d_i' + \delta + n\lambda \tag{11}$$

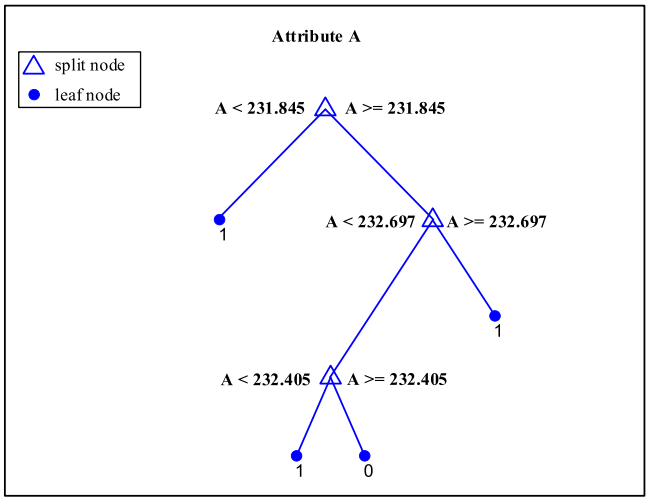
where the values of m and n are given by

$$\begin{aligned} \min \quad & m, n \\ \text{s.t.} \quad & d_i' + 2m + n\lambda \neq d_j' \\ & s_k < d_i' + 2m + n\lambda < s_{k+1} \\ & -df < n\lambda < 1 - df \\ & m, n \in Z \\ & j = 1, \ldots, t \end{aligned} \tag{12}$$

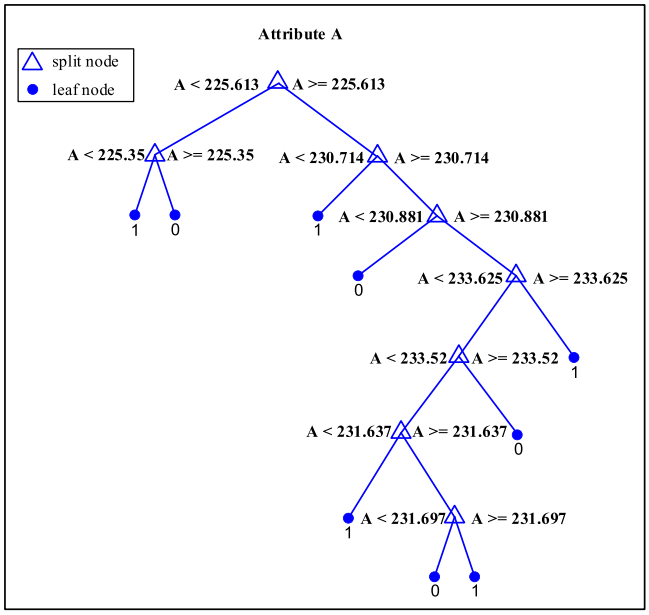
where λ is the step size of the fractional part, $\delta = 2\,m\,(m = 1, 2, 3\cdots)$ is the step of the integer part, and df is the value of the fractional part of d_j':

$$df = d_j' - f\left(d_j'\right) \tag{13}$$

where $f(d_j')$ is the largest integer that does not exceed d_j'. $f(d_j')$ can be expressed as:

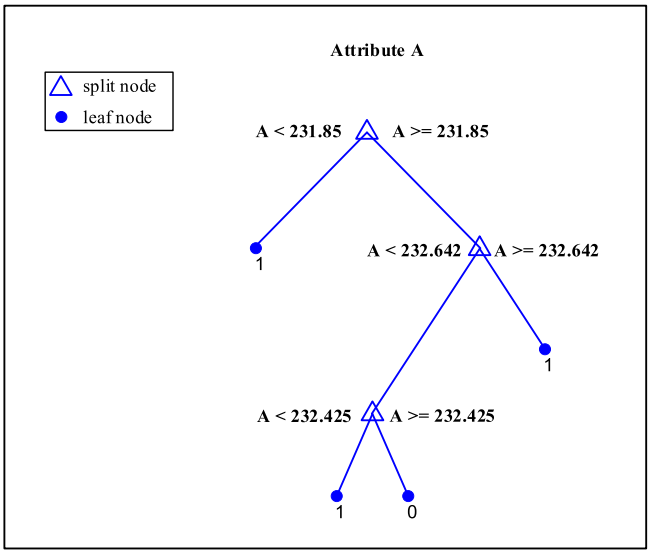


(a) Decision tree of original data

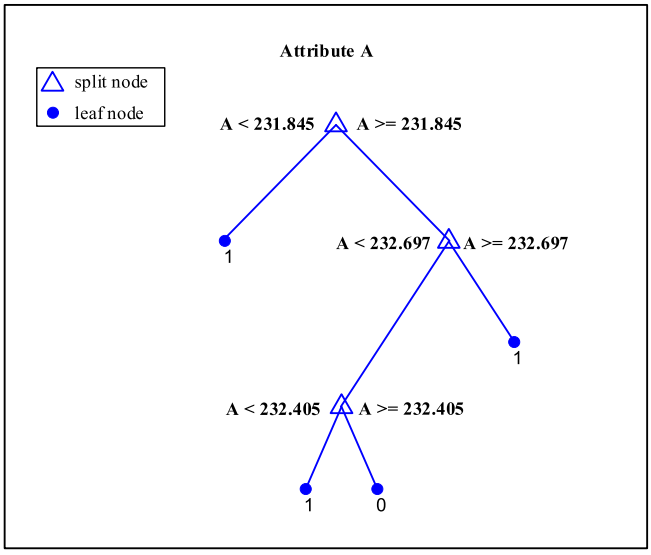


(b) Decision tree of watermarked data

Fig. 12. Changes in decision tree when embedding 64bits watermark information.



(a) Decision tree of watermarked data after LST



(b) Decision tree of watermarked data after SVA

Fig. 13. Decision tree of watermarked data with 64bits watermark information after LST and SVA.

$$f\left(d_j^{'}\right) = \lfloor d_j^{'} \rfloor \tag{14}$$

In the proposed algorithm, λ is set to be 1 bit larger than the smallest decimal precision of the data. For instance, if the decimal precision of data is 0.001, the value of λ is set to 0.0001. This ensures that there is redundant space in the decimal part for two different elements during the initial data shifting. The values of m and n are set to satisfy the condition by traversing the search. If no m and n values satisfy Eq. (12) when $d_j^{'}$ is shifted, then the algorithm searches for the next element with the label c_i on the left of s_i to shift until the number of shifted elements satisfies the number of dc_{li}. After traversing all data in the current segment, if the number of shifts that meets Eq. (12) is less than the number of dc_{li}, then the data need to be shifted using the global swapping method.

b) Global swapping

Global swapping is not limited to data in the current segment. When the label state of the i-th split value is adjusted, the first $i-1$ split values are mapped to the data. As shown in Fig. 6, data are divided into disjoint segments; $i-1$ split values can generate i data segments. In Fig. 6, the red box is the data segment where the current split value s_i is located and the yellow line indicates the location of the other split values. In global swapping, it is necessary to find a segment that meets the following condition 1 where s_i is not located:

Condition 1. The remaining data segments have tuples with the label c_i and the data to be modified satisfy Eqs. (11) and (12).

After the data segment that satisfies condition 1 is found, an element labeled with c_i should be found on the left of the data segment where s_i is located that satisfies the following condition 2:

Condition 2. A data element that is swapped should satisfy Eqs. (11) and (12) when it is shifted to a data segment that satisfies condition 1.

If there is a data segment that satisfies conditions 1 and 2, then global swapping can be achieved. The blue box in Fig. 6 shows the data selected for swapping.

When the number of labels in a data segment is not sufficient to apply

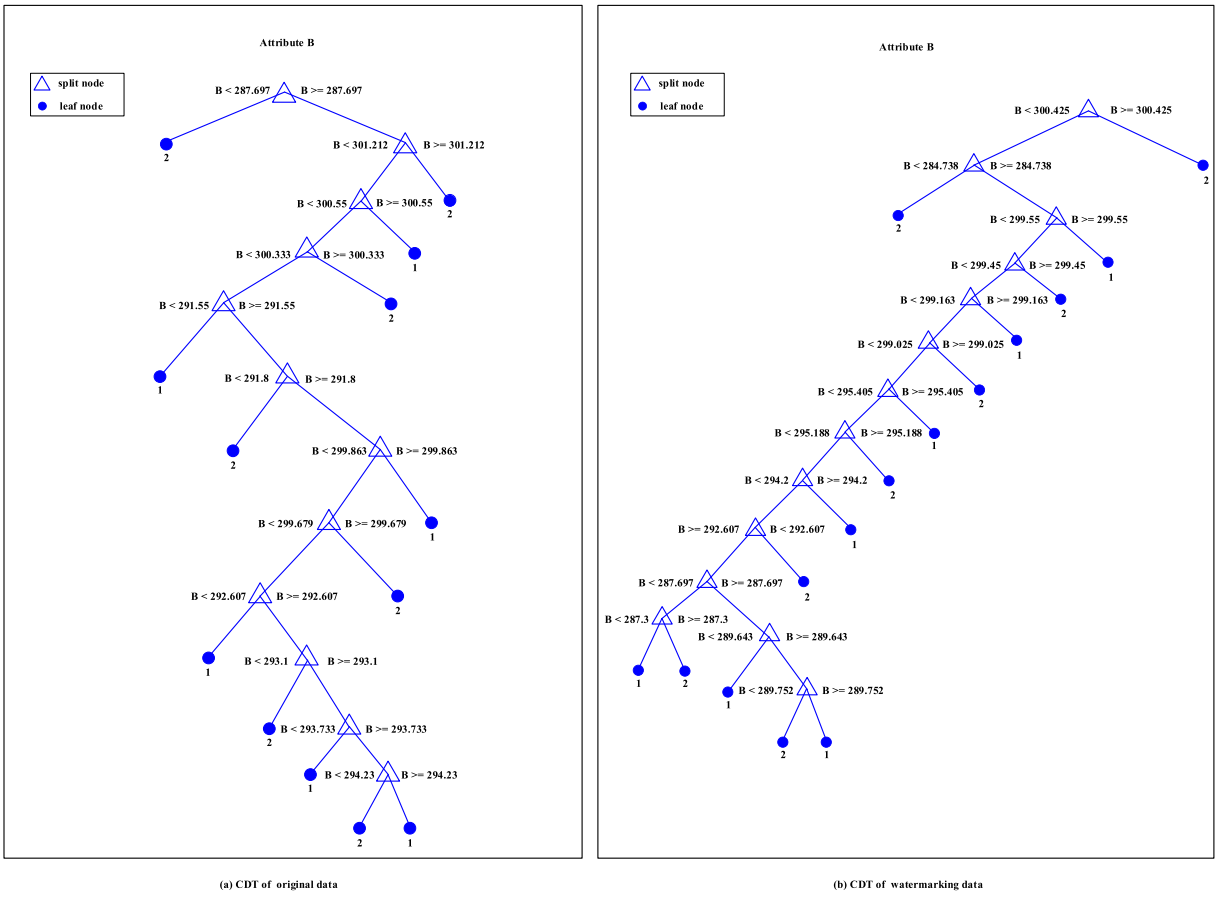


Fig. 14. Changes in decision tree when embedding 128bits watermark information.

shifting after using methods a) and b), then step δ in Eqs. (11) and (12) can take any value when the data are shifted.

To better illustrate the LST, a schematic diagram of the watermarked data label adjustment process is presented in Fig. 7. The first row of the data table in Fig. 7 represents the primary key value of the data, the second row is the characteristic attribute, and the third row is the label attributes, denoted by $c_1 = 0$ and $c_2 = 1$. W is watermark information to be embedded. The yellow line in Fig. 7 indicates the position of the split value. After the original data are embedded with watermark information, s_1 is mapped to the watermarked data, and the amount of labels dc_{li} to the left of s_1 should be used for the adjustment. According to the recorded values of p_1 and $p_1^{'}$, the number of labels in the watermarked data satisfies $dc_{li} = 0$; hence, the watermarked data are divided by s_1. Then, s_2 is mapped in the left data segment; the position is shown by the yellow line. It can be calculated that $dc_{l2} = -1$, and data with a label of 1 should be moved from the right of the split value to the left. In this example, $\delta = 2$, $\lambda = 0.001$, and according to Eq. (12), $m = 0$ and $n = 441$; hence, 13.72 is changed to 13.279. According to the recorded p_3, $dc_{l2} = -1$. Data with label 1 should be moved from the right side of s_3 to the left side of s_3. According to Eqs. (11) and (12), no m and n values exist that can meet the constraint in the data segment; hence, global swapping is required. To map s_1 and s_2 to data, the data are divided into three segments. The blue box in Fig. 7 represents the currently adjusted data segment. The solid red boxes represent the data segments that can be swapped according to conditions 1 and 2. The gray part represents the data to be swapped globally. For two values with primary keys 10 and 2, the data should be modified so that the sorted position of the data is changed during the global exchange. Therefore, to minimize the distortion of the data, the difference between the modified value and original value is minimized. For example, if the modified value of 15.9 is closest to 12.6, and the modified value of 12.6 is closest to 15.9, the overall distortion after the swap is guaranteed to be minimal. According to Eqs. (11) and (12), when $\delta = 2$ and $\lambda = 0.001$, 12.6 is changed to 14.989, and 15.9 is changed to 13, thus completing the global swapping process. Finally, the label status of s_4 is adjusted. The red dashed box in Fig. 7 represents the adjusted watermarked data after all split values have been adjusted. Sorting the data in ascending order according to the characteristic attribute value can verify that the label status of the watermarked data is consistent with the label status of the original data.

When the data are modified by steps δ and λ, the watermark information in the data is not destroyed. The specific analysis of watermark preserving is shown in Subsection 5.2. If the label status cannot be consistent when using Intra-segment shifting and Global swapping, the data are adjusted by changing step length δ, which can destroy the original watermark information. Thus, error correction codes are introduced in the watermark generation process to solve this problem.

In Fig. 8(a) and (b), the CART trees generated by the original data and watermarked data are presented, respectively. The watermarked tree after the LST is presented in Fig. 8(c). Fig. 8 shows that, after the LST, the tree structure of the watermarked data is the same as that of the original data, except for the split value. The consistency theory analysis of the decision tree structure is in Subsection 5.1.

4.4. Split value adjustment

The watermarked data can generate a CART tree with the same structure as the original data after the LST algorithm is applied. Therefore, in this section, a method is proposed to reconstruct the split value. During the SVA, each split value is divided into an independent split value and continuous split value according to the status of the split value. As shown in Fig. 9, s_l is independent, and s_k, s_i, and s_j are

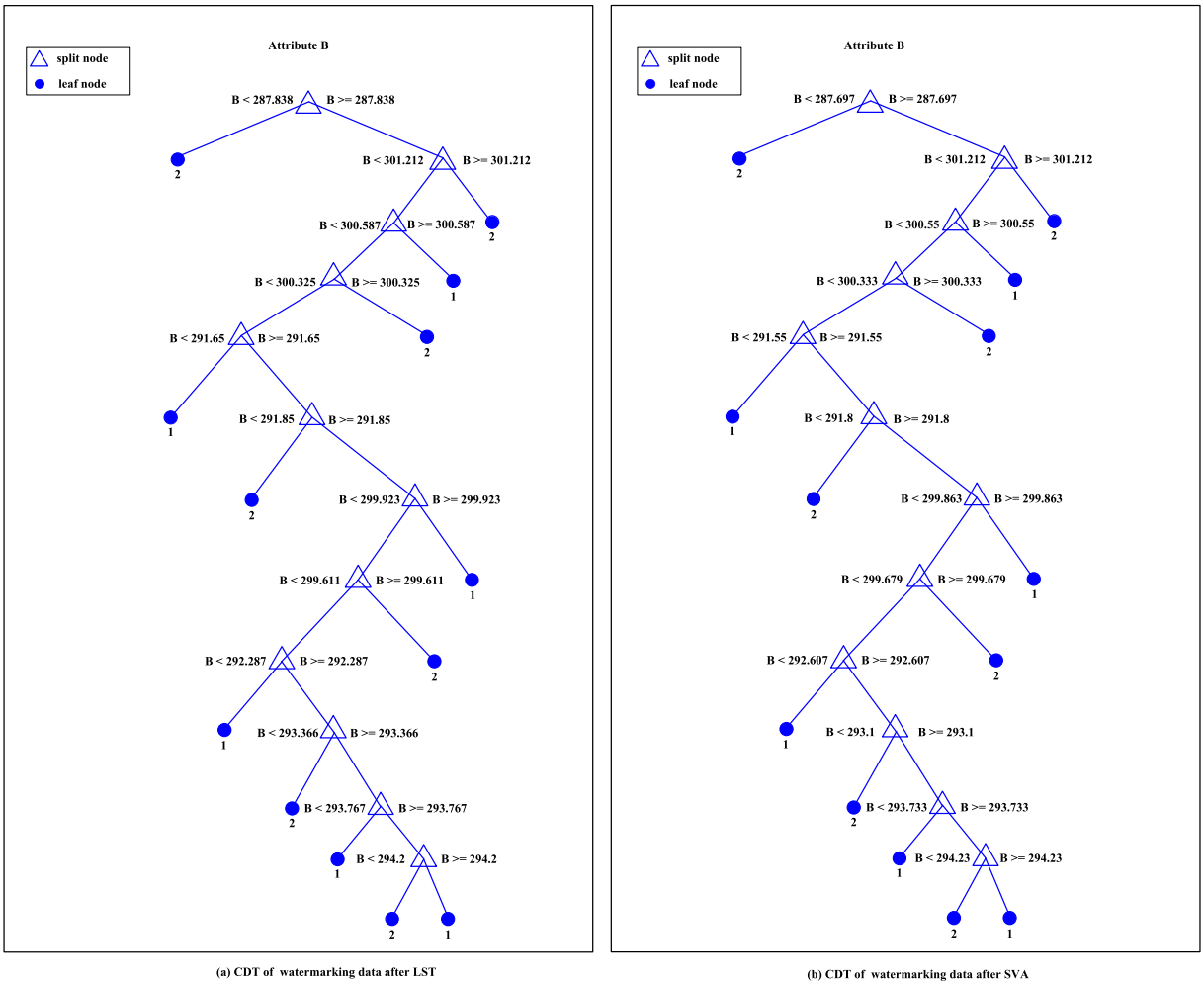


Fig. 15. Decision tree of watermarked data with 128bits watermark information after LST and SVA.

continuous. The adjustment of the independent split value needs to consider only that value. By contrast, the continuous split values are mutually restricted and need to be considered simultaneously. According to Subsection 4.3, the original split value is mapped to the watermarked data to satisfy Eq. (9). Thus, for independent split values, when the relative position of the watermarked data is unchanged, there are two values to satisfy:

$$\frac{d_j^{'} + d_{j+1}^{''}}{2} = s_i \tag{15}$$

From previous knowledge, the split value of the decision tree is calculated using two different numbers; hence, $d_j^{''} \neq d_{j+1}^{''}$.

Unlike the adjustment of the independent split value, the adjustment of one continuous split value affects the other continuous split values. Assuming that the continuous split values are not considered simultaneously, there may be scenarios in which the watermark split value cannot be adjusted. In Fig. 10, a case is presented in which the adjustment cannot be completed because the continuous split values are considered independently. In the figure, the original split values are assumed to be $s_1 = 12.7$, $s_2 = 13$, *and* $s_3 = 13.2$, which satisfy Eq. (9). First, the first split value of the watermarked data is adjusted. Then two values (12.6, 12.8) that satisfy Eq. (15) are found. When the second split value is adjusted, the two numbers that produce s_1 do not change. At this time, only 13.1 can be adjusted to make the watermark split value equal to s_2; hence, the modified values (12.8, 13.2) can be obtained. Next, the split value s_3 is adjusted while the previous data remain unchanged; in this case, it is impossible to determine two different numbers equal to s_3. However, if the split values are considered simultaneously, the data (12.5, 12.9, 13.1, 13.3) can be found to complete the adjustment. Therefore, the continuous division value adjustment problem is converted into a constrained indeterminate equation solving problem:

$$\begin{cases} d_1^{''} + d_2^{''} = 2s_1, \\ \vdots \\ d_n^{''} + d_{n+1}^{''} = 2s_n. \\ d_0^{'} < d_1^{''} < s_1 < d_2^{''} < \cdots < d_{n+1}^{''} < d_{n+2}^{'} \end{cases} \tag{16}$$

where $\{d_1^{''}, \ldots, d_{n+1}^{''}\}$ are unknown variables, and $d_0^{'}$, $d_{n+2}^{'}$, and $s_i (1 \leq i \leq$

Table 1
MAE, VAR, RDM, AND RDV when 64bits watermark information is embedded.

Method	MAE	VAR	RDM	RDV
Original	220.1258	123.1110	/	/
LSB	220.1542	124.4771	0.013%	1.110%
Proposed Method	220.1397	127.2665	0.006%	3.345%

Table 2
MAE, VAR, RDM, AND RDV when 128bits watermark information is embedded.

Method	MAE	VAR	RDM	RDV
Original	264.8221	631.1322	/	/
LSB	264.7816	631.1565	0.015%	0.004%
Proposed Method	264.8466	627.8256	0.005%	0.524%

Table 3
Ber of LSB watermarking algorithm and our proposed algorithm under tuple deletion attack.

Deletion	0	20%	30%	40%	50%	60%	70%	80%
LSB	0	0.0058	0.0582	0.0864	0.1239	0.1533	0.1783	0.2016
Proposed Method	0	0.0067	0.0575	0.0789	0.1071	0.1447	0.1782	0.1940

n) are known variables. For n original split values, an indeterminate equation system with $n+1$ unknown variables can be constructed. For an independent split value, according to Eq. (9), there must be a solution to the equation system under the above constraints, and it can be guaranteed that at least one of the two solutions satisfies Eqs. (11) and (12). The proof is provided in Appendix A. Regarding continuous split values, in some cases, the equation may not have a solution under the above constraints because the data are related to each other. In such a case, $d_0^{'}$ and $d_{n+2}^{'}$ are no longer considered. To obtain an optimal solution in these cases, two aspects are mainly considered:1) changes in data should satisfy Eqs. (11) and (12) as much as possible; and 2) changes of data should be minimized under 1).

According to the decision tree generation principle presented in Subsection 2.1, the split values of a decision tree are calculated from two different data elements. Therefore, the same data elements are considered as a whole for the adjustment in Subsections 4.3 and 4.4. However, when the same data are regarded as a whole and shifted using data shifting, the labels in some segments may not be consistent. This is because the label 'false consistent' state can appear in the previous LST process. As shown in Fig. 11, the LST of s_1 and s_2 in the watermarked data is complete. When adjusting s_3, the split value is first mapped to the yellow line of the last row of data. Suppose data with label 1 should be shifted from the left side to the right side of the split value. Because the same data elements are shifted as a whole, the status of the label cannot be consistent regardless of the shift type. Once this occurs, it is necessary to backtrack to find exchangeable data segments and then exchange the data with the same labels. The requirement for judging whether data is exchangeable is that the exchange of data does not affect the consistent state of the labels of the other split values. As shown in Fig. 11, when reversing back to the data segment in which s_1 is located, data (marked in gray) can be found that can be shifted and swapped with the data in the red box.

4.5. Watermark extraction

Watermark extraction is the reverse of the embedding process. The specific steps are as follows:

Step 1. Watermark Extraction. First, the watermarked data are grouped using Eq. (7). Then watermark information is extracted from the data using Eq. (6). Because the data will encounter non-malicious attacks during data transmission, the watermark information will change. To improve the robustness of the watermark, it is necessary to 'vote' on watermark information extracted from the same group; the larger number of watermark information after voting is selected as watermark information embedded in the group.

Step 2. Watermark Error Correction Decoding. The final watermark information can be obtained by decoding the BCH.

4.6. Generalization in quantization-based watermarking algorithms

The proposed algorithm can also be extended to watermarking algorithms based on different quantization steps. In this paper, the LSB watermarking algorithm can be regarded as a quantization step of $q=2$.

When quantization step q is greater than two, it is only necessary to change the watermark embedding method in Eq. (5) to:

$$d_i^{'} = 2q\lfloor d_i/2q \rfloor + qw_i \tag{17}$$

The extraction of w_i is changed from Eq. (6) to:

$$w_i = \frac{d_i}{q} \bmod (2) \tag{18}$$

When the LCS, LST, and SVA methods are applied, if q is greater than two, only the value of δ in Subsections 4.3 and 4.4 needs to be modified to $q\delta$ to complete the adjustment of the decision tree. It should be noted that when q is greater than two, the proof in Appendix A is no longer applicable.

5. Consistency preserving analysis

5.1. Decision tree structure consistency analysis

In this section, we analyze the consistency of the decision tree structure after the LST method.

When the LST method is applied, we first sort the watermarked data in ascending order by the value of the feature attributes and map the original split values to the watermarked data; the specific mapping method is described in Subsection 4.3.1. Then, we obtain the label statistics value $p_i^{'} = [s_i|(\sum c_{l1}^{'}, \sum c_{l2}^{'}, \ldots \sum c_{lk}^{'}|\sum c_{r1}^{'}, \sum c_{r2}^{'}, \ldots, \sum c_{rk}^{'})]$ at the location of s_i in the watermarked data; $\sum c_{li}^{'}$ and $\sum c_{ri}^{'}$ indicate the amounts of data labeled with c_i to the left and right of the split value s_i in the watermarked data. Because watermark embedding only modifies the data and does not affect the number of labels, the $p_i^{'}$ of watermarked data and p_i of original data satisfy:

Table 4
Ber of LSB watermarking algorithm and our proposed algorithm under tuple insertion attack.

Insertion	0	20%	30%	40%	50%	60%	70%	80%
LSB	0	0.0005	0.0025	0.0072	0.0152	0.0259	0.0385	0.0526
Proposed Method	0	0.0004	0.0024	0.0064	0.0131	0.0221	0.0332	0.0464

Table 5
Ber of LSB watermarking algorithm and our proposed algorithm under feature attribute value modification.

Modification	0	20%	30%	40%	50%	60%	70%	80%
LSB	0	0.00006	0.0040	0.0309	0.0747	0.1196	0.1691	0.2282
Proposed Method	0	0.00005	0.0040	0.0319	0.0745	0.1190	0.1686	0.2281

$$\sum \overset{'}{c}_{l1} + \cdots + \sum \overset{'}{c}_{l\mathscr{y}} + \sum \overset{'}{c}_{r1} + \cdots + \sum \overset{'}{c}_{r\mathscr{y}} = \sum c_{l1} + \cdots + \sum c_{l\mathscr{y}} + \sum c_{r1} + \cdots + \sum c_{r\mathscr{y}} \tag{19}$$

The numbers of left and right labels of s_i in the watermarked data segment are adjusted using data shifting so that:

$$\begin{aligned} \sum \overset{'}{c}_{l1} + \cdots + \sum \overset{'}{c}_{l\mathscr{y}} &= \sum c_{l1} + \cdots \sum c_{l\mathscr{y}} \\ \sum \overset{'}{c}_{r1} + \cdots + \sum \overset{'}{c}_{r\mathscr{y}} &= \sum c_{r1} + \cdots \sum c_{r\mathscr{y}} \end{aligned} \tag{20}$$

When Eq. (20) is satisfied, that is, $p_i^{'} = p_i$, s_i in the watermarked data segment has the same label status as that in the original data. Next, the watermarked data can be split into two segments using s_i, and the original split values s_{i+1} and s_{i+2} are mapped to these two segments to adjust the label state so that $p_{i+1}^{'} = p_{i+1}$ and $p_{i+2}^{'} = p_{i+2}$. At the end of the LST, the label statistics P' of the watermarked data and the label statistics P of the original data satisfy:

$$P^{'} = P \tag{21}$$

According to Subsection 2.1, the two different datasets can generate CART trees with a consistent structure when they satisfy Eq. (21).

5.2. Watermark preserving analysis

In the embedding process of the LSB, the watermark is embedded in the LSB of the integer part of the data. Therefore, the watermark carried by the data remains unchanged when the modification step of the integer part of the data is $\delta = 2\,m$.

Modified data d'' are given by:

$$d^{''} = d^{'} + \delta \tag{22}$$

According to Eq. (6), watermark information w extracted from d'' is:

$$\begin{aligned} w &= d^{''} \bmod (2) \\ &= (d^{'} + 2m) \bmod (2) \\ &= d^{'} \bmod (2) \end{aligned} \tag{23}$$

Based on Eq. (5), the extracted watermark information remains unchanged when the integer part is modified by step δ.

Additionally, according to Eqs. (11)–(13), the value of $n\lambda$ is always less than 1, regardless of the value of n. Thus, the integer part of d' does not change.

6. Experiment results

The proposed algorithm was applied to the UCI public dataset 'occupancy detection' for verification. This dataset is a binary classification dataset. The feature and label attributes were selected to construct a two-dimensional dataset. The dataset for the test contained 1000 tuples. For a 128-bit watermark, 3-bit BCH error correction codes were used. For a 64-bit watermark, a 1-bit BCH error correction code was used. The watermark embedding algorithm was applied to the feature attributes, and the LCS, LST, and SVA methods were used to preserve decision tree consistency. The experimental results verified that the proposed algorithm effectively reconstructed the decision tree and kept the data distortion low. Additionally, the proposed algorithm did not weaken the robustness of the watermark.

In the first experiment, the consistency preservation of the decision tree was verified. First, 64-bit and 128-bit watermark information were embedded in different two-dimensional datasets. The original tree and the tree generated by the data embedded with 64-bit and 128-bit watermark information are presented in Figs. 12 and 14, respectively. Because the generated decision tree was too large, the changes in the decision tree are illustrated by representing the local branches where the changes were obvious. The changes in the other branches were similar to the changes in the presented branches. A triangle in the figures represents the split value of the feature attribute, a circle represents the leaf node, and the values one and zero indicate two label types. Figs. 12(b) and 14(b) show that, because of watermark embedding, the decision tree obtained after watermarking changed significantly compared with the original decision tree. Figs. 13(a) and 15(a) show the decision trees generated after the LST watermarking algorithm was applied with 64-bit and 128-bit watermark information, respectively. Figs. 13–15 show that the watermarked data generated a tree with the same structure as the original data after the LST method. After the SVA, as shown in Figs. 13 (b) and 15(b), the tree generated using watermarked data was consistent with the original tree. The experiment verified the effectiveness of the proposed algorithm.

In the second experiment, the statistical distortion of data was quantified using the mean and variance metrics. The changes in the mean and variance values of the data after the proposed algorithm was applied were evaluated. After many repeated tests, the average data distortion results after the 64-bit and 128-bit watermark information were embedded are presented in Tables 1 and 2, respectively. The second line in Tables 1 and 2 is the average absolute error (*MAE*), variance (*VAR*), mean change rate (*RDM*), and variance change rate (*RDV*) of the original data. *RDM* and *RDV* can be calculated using the following formulas, respectively:

$$RDM = \frac{MAE_D - MAE_{D_W}}{MAE_D} \tag{24}$$

$$RDV = \frac{VAR_D - VAR_{D_W}}{VAR_D} \tag{25}$$

where D and D_W denote the original data and watermarked data, respectively. The third row of the table shows the value after the watermark is embedded using the LSB watermarking algorithm, and the last row shows the *MAE*, *VAR*, *RDM*, and *RDV* after the CDTR method using the proposed algorithm. The tables show that the *MAE* values of the proposed algorithm and traditional LSB watermarking algorithm were similar, whereas the *VAR* value of the proposed algorithm was slightly higher than that of the traditional LSB watermarking algorithm.

In the final experiment, the LSB watermarking algorithm was compared with the proposed algorithm in terms of robustness when embedding 128-bit watermark information. Three attacks were tested separately: tuple deletion, tuple insertion, and feature attribute value modification. The percentage of attacks on the tuple and attribute values ranged from 0% to 80%. The bit error rate (BER) results of the watermark extracted from the data encountered under the three attack methods are presented in Tables 3–5. As shown in these tables, the robustness of the proposed algorithm was comparable with that of the traditional LSB watermarking algorithm. There are two main reasons for this. First, the proposed algorithm is based on the LSB watermarking algorithm and the data are adjusted by the LCS, LST, and SVA methods to obtain a consistent preserved decision tree. Therefore, the proposed algorithm has the same robustness as the LSB watermarking algorithm. Second, the data were adjusted using constraint equations to ensure that the watermark was not destroyed when the watermark decision tree was consistently adjusted. In rare cases in which the watermark was damaged, a BCH error correction code was introduced to correct the

false watermark information. It is also clear from the experimental results that the proposed algorithm did not affect the watermark information. The experimental results verified that the proposed algorithm was not only effective in preserving the robustness of the original watermarking algorithm but also preserving the consistency of the decision tree.

7. Conclusion

In this paper, we proposed a consistency preserving database watermarking algorithm. Unlike the traditional LSB watermarking algorithm, the proposed algorithm considers the impact on the data mining model after the watermark is embedded in two-dimensional data. We proposed the LCS, LST, and SVA methods to adjust the watermarked data so that the decision tree generated from the watermarked data was consistent with the original decision tree. Our proposed algorithm eliminated the corruption of the embedded watermark information by setting constraint equations to ensure the error-free extraction of the watermark without attacks. The experimental results also verified that, through our proposed algorithm, we not only effectively embedded watermarking information into the data but also ensured that the watermarked data could be used effectively for decision tree model construction, which solved the problem of inconsistent decision trees caused by traditional watermarking algorithms. Additionally, our proposed decision tree consistency algorithm can be combined with other quantization-based watermarking algorithms. Therefore, the proposed algorithm is universal.

Acknowledgements

This work is supported by the National Key Research and Development Program of China under Grant 2021YFB2700600, the National Natural Science Foundation of China under Grant 62132013 and 61902292, the Key Research and Development Programs of Shaanxi under Grants 2021ZDLGY06-03, the Truth-Seeking Research Scholarship Fund of Xidian University.

Appendices A.

After we adjust the split value, the values $d_j^{''}$ and $d_{j+1}^{''}$ satisfy Eq. (20). We assume that these two data and the watermarked data $d_j^{'}$, $d_{j+1}^{'}$ do not satisfy Eq. (16) under the condition of steps δ and λ. To clarify that at least one value can satisfy Eq. (16), we consider the following two cases:

Case 1. $d_j^{''} < d_j^{'}$

If we modify $d_j^{''}$ to satisfy Eq. (16) with $d_j^{'}$, we can modify it in two directions, that is, decrease and increase it under the condition of step sizes δ and λ. The values with the smallest change can be obtained as $f(d_j^{''}) - \lambda$ and $f(d_j^{''}) + 1$.

We consider the new value of $d_j^{'''}$ as $d_j^{'''} = f(d_j^{''}) + 1$, which must satisfy $d_j^{'''} \le d_j^{'}$. $d_j^{'''}$ increases relative to $d_j^{''}$; hence, $d_{j+1}^{''}$ needs to increase less to satisfy Eq. (17):

$$d_{j+1}^{'''} = d_{j+1}^{''} - \left(d_j^{'''} - d_j^{''}\right) \tag{26}$$

We can obtain two new values $d_j^{'''}$ and $d_{j+1}^{'''}$, which makes $(d_j^{'''} + d_{j+1}^{'''})/2 = s_i$ while keeping the relative position of the data unchanged.

Case 2. $d_j^{''} > d_j^{'}$

If $d_j^{''}$ and $d_j^{'}$ satisfy Eq. (16) under the condition of steps δ and λ, the value changes are the same as those in case 1. We consider $d_j^{'''} = f(d_j^{''}) - \lambda$, and from previous knowledge, there must be a step λ such that $d_j^{'} \le d_j^{'''} < s_i$. $d_j^{'''}$ is smaller than $d_j^{''}$; hence, $d_{j+1}^{''}$ needs to move in a larger direction. Hence,

$$d_{j+1}^{'''} = d_{j+1}^{''} + \left(d_j^{''} - d_j^{'''}\right) \tag{27}$$

In case 2, we can also obtain two new values $d_j^{'''}$ and $d_{j+1}^{'''}$, which makes $(d_j^{'''} + d_{j+1}^{'''})/2 = s_i$ while keeping the relative position of the data unchanged. Through the above two cases, we can prove that at least one value can satisfy Eq. (16) when the split value is adjusted.

References

[1] Z. Ruan, Y. Miao, L. Pan, et al., Visualization of big data security: a case study on the KDD99 cup data set, Digital Commun. Networks 3 (4) (2017) 250–259.
[2] S. Yu, P. Muller, A. Zomaya, Special issue on “big data security and privacy”, Digital Commun. Networks 3 (4) (2017) 211–212.
[3] A.O. Almagrabi, A.K. Bashir, A classification-based privacy-preserving decision-making for secure data sharing in Internet of Things assisted applications[J], Digital Commun. Networks 8 (4) (2022) 436–445.
[4] D. Wu, B. Yang, R. Wang, Scalable privacy-preserving big data aggregation mechanism[J], Digital Commun. Networks 2 (3) (2016) 122–129.
[5] X. Wu, X. Zhu, G.Q. Wu, et al., Data mining with big data, IEEE Trans. Knowl. Data Eng. 26 (1) (2013) 97–107.
[6] J. Feng, L. Liu, Q. Pei, K. Li, Min-max cost optimization for efficient hierarchical federated learning in wireless edge networks, IEEE Trans. Parall. Distr. 33 (11) (2022) 2687–2700.
[7] H. Hamdoun, A. Sagheer, Information security through controlled quantum teleportation networks, Digital Commun. Networks 6 (4) (2020) 463–470.
[8] B.B. Mehta, H.D. Aswar, Watermarking for security in database: a review, in: 2014 Conference on IT in Business, Indus-try and Government (CSIBIG), 2014, pp. 1–6.
[9] R. Agrawal, J. Kiernan, Watermarking relational databases, in: VLDB '02: Proceedings of the 28th International Conference on Very Large Databases, 2002, pp. 155–166.
[10] M.K. Rathva, G.J. Sahani, Study on watermarking relational databases, Theory Pract. Math. Comput. Sci. 9 (2021) 161–171.
[11] S. Rani, R. Halder, Comparative analysis of relational database watermarking techniques: an empirical study, IEEE Access 10 (2022) 27970–27989.
[12] W. Wang, A. Men, B. Yang, et al., A novel robust zero watermarking scheme based on DWT and SVD, in: 2011 4th International Congress on Image and Signal Processing, vol. 2, IEEE, 2011, pp. 1012–1015.
[13] I. Kamel, M. AlaaEddin, W. Yaqub, et al., Distortion-free fragile watermark for relational databases, Int. J. Big Data Intell. 3 (3) (2016) 190–201.
[14] I.K. Waheeb Yaqub, Z. Aung, Distortion-free watermarking scheme for compressed data in columnar database, Proceed. 15th ICETE (2018) 343–353. Porto, Portugal.
[15] C.C. Lin, T.S. Nguyen, C.C. Chang, LRW-CRDB: lossless robust watermarking scheme for categorical relational databases, Symmetry 13 (11) (2021) 2191.
[16] S. Yan, S. Zheng, B. Ling, et al., Lossless database watermarking based on order-preserving encryption, in: ACM Turing Award Celebration Conference-China (ACM TURC 2021), 2021, pp. 216–223.
[17] C.C. Lin, T.S. Nguyen, C.C. Chang, LRW-CRDB: lossless robust watermarking scheme for categorical relational databases, Symmetry 13 (11) (2021) 2191.
[18] A.A. Ismail, S.M. Darwish, Context-based zero database watermarking scheme based on evolutionary mechanism, in: International Conference on Advanced Machine Learning Technologies and Applications, Springer, Cham, 2021, pp. 1005–1015.
[19] A.K. Dwivedi, B.K. Sharma, A.K. Vyas, Watermarking techniques for ownership protection of relational databases, Int. Conf. Ad. Develop. Eng. Technol. 6 (1) (2012) 988–995.
[20] B.B. Mehta, U.P. Rao, A novel approach as multi-place watermarking for security in database, in: International Conference on Sam, 2014 arXiv.

[21] S. Melkundi, C. Chandankhede, A robust technique for relational database watermarking and verification, in: 2015 International Conference on Communication, Information & Computing Technology (ICCICT), 2015, pp. 1–7.
[22] R. Sion, M. Atallah, S. Prabhakar, Rights protection for relational date, IEEE Trans. Knowl. Data Eng. 16 (6) (2004) 1509–1525.
[23] M. Shehab, E. Bertino, A. Ghafoor, Watermarking relational data-bases using optimization-based techniques, IEEE Trans. Knowl. Data Eng. 20 (1) (2008) 116–129.
[24] Y. Li, R. Deng, Publicly verifiable ownership protection for relational databases, in: Proc. ACM Symp. Information, Computer and Comm. Security, 2006, pp. 78–89.
[25] S. Bhattacharya, A. Cortesi, A distortion free watermark framework for relational databases, in: Proc. Fourth Int'l Conf. Software and Data Technologies (ICSOFT '09), 2009, pp. 229–234.
[26] M. Kamran, S. Suhail, M. Farooq, A robust, distortion minimiz-ing technique for watermarking relational databases using once-for-all usability constraints, IEEE Trans. Knowl. Data Eng. 25 (12) (2013) 2694–2707.
[27] M. Kamran, M. Farooq, A formal usability constraints model for watermarking of outsourced datasets, IEEE Trans. Inf. Forensics Secur. 8 (6) (2013) 1061–1072.
[28] L. Breiman, J.H. Friedman, R.A. Olshen, et al., Classification and regression trees (CART), Biometrics 40 (3) (1984) 358.