Don't Abandon the Primary Key: A High-Synchronization and Robust Virtual Primary Key Scheme for Watermarking Relational Databases

Ke Yang[1,2,3], Shuguang Yuan[1,2,3], Jing Yu[1,2,3], Yuyang Wang[1,2,3], Tengfei Yang[4], and Chi Chen[1,2,3](✉)

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
chenchi@iie.ac.cn

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

[3] Key Laboratory of Cyberspace Security Defense, Beijing, China

[4] National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing, China

Abstract. A relational database is an infrastructure that manages and shares structured data. To safeguard the copyrights of data within such databases, database watermarking is an effective technique. Most watermarking schemes rely on the primary key (PK) to locate and embed watermarks, which preserves high watermark synchronization during detection. However, these schemes become invalid once PK is erased or changed. To avoid this vulnerability, virtual primary key (VPK) schemes are proposed to replace PK. Nevertheless, duplicate values in virtual primary keys compromise synchronization. Besides, current VPK schemes fail to utilize the primary key, even when it remains unchanged. This strategy decreases detection accuracy in many cases. In addition, we find an attribute name attack. It is a common challenge of existing watermarking schemes, which distort the link between detected attributes and watermarked attributes. In this paper, we propose a high-synchronization and robust VPK scheme. It introduces a classifier to maintain the original order and number of attributes to resist attribute name attacks. To resist primary key erasure or change and mitigate synchronization problems, it generates distinct virtual primary keys. Moreover, to improve detection accuracy, our watermarking scheme integrates the primary key when it remains unchanged. Experiments demonstrate that our scheme achieves high watermark synchronization. It is robust against various attacks, even when 98% of tuples or over half of the attributes are distorted. It is also practical in terms of data distortion and overhead.

Keywords: Relational databases · Watermarking · Virtual primary key

S. Katsikas et al. (Eds.): ICICS 2024, LNCS 15057, pp. 289–309, 2025.
https://doi.org/10.1007/978-981-97-8801-9_15

1 Introduction

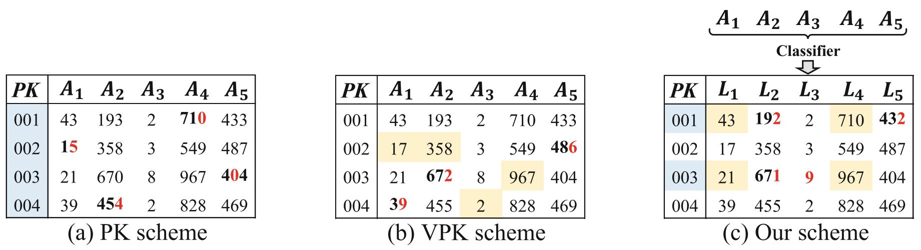
Relational databases continue to be the preferred data storage solution for a majority of enterprises and applications, owing to their stability, maturity, and ease of maintenance. In the context of the current digital era, driven by advances in cloud computing and big data technology, the demand for data sharing in these databases has increased significantly. Data copyright of relational databases is then becoming a significant concern.

Database watermarking techniques protect copyright by embedding watermark information into preprocessed relational databases before sharing, and extracting it to prove ownership if needed. Watermark detection and extraction require the provision of all correct secret parameters set by the database owner. The process of aligning the detected watermark with the embedded watermark is referred to as watermark synchronization.

Most watermarking schemes (denoted as PK schemes) rely on the primary key to select, embed, and detect each mark of watermark. Considering the uniqueness of primary key, each mark is more likely to be chosen uniformly, facilitating high watermark synchronization. These schemes assume that the primary key of the relational database cannot be erased or changed since the primary key contains valuable information, and changing it will render the database less useful from the user's point of view [1]. Nevertheless, there is a vulnerability. These schemes cannot perform the watermarking process without primary key. Attackers might bypass watermark detection by erasing or changing the primary key (denoted as primary key attacks) if they disregard its information.

To avoid the vulnerability, virtual primary key (VPK) schemes have been proposed. These schemes are robust against primary key attacks by providing new values for watermarking instead of primary key. They generate virtual primary keys using non-primary key attributes [3,12,14,21,22], or pseudo-random selecting values in tuples to construct new ones [4,7–9,19]. However, these schemes face synchronization problems [8]. The redundancy of the set of virtual primary keys causes a nonuniform distribution of chosen marks, affecting the embedding quality and the scheme's robustness. For example, consider a set of virtual primary keys $\{11, 22, 22, 22, 22, 33, 33\}$, where each unique virtual primary key selects a distinct mark. It is observed that the virtual primary key 22 selects and embeds a mark four times, whereas 11 embeds only once. Therefore, the mark selected by 11 is more likely to be distorted, rendering the watermark incomplete. Besides, existing VPK schemes fail to utilize the primary key, which results in synchronization problems even when it remains unchanged. This decreases detection accuracy in many cases.

All these watermarking schemes aim to protect copyright and resist attacks. The existing literature primarily discusses subset attacks (distortion of tuples), attribute attacks, and additive (re-watermarking) attacks [15,17,26]. While attribute attacks typically involve altering attribute columns by deletion, insertion, or shuffling, we also identify a significant risk from attacks on attribute names, which have not yet been considered in existing schemes. Such attacks can bypass watermark detection because many schemes embed and detect watermarks based on the original order of attributes. This order depends on the attribute names. For instance, attacks such as substituting, deleting, or shuffling attribute names can bypass watermark detection without modifying the data values.


Classifier

PK	A_1	A_2	A_3	A_4	A_5
001	43	193	2	710	433
002	15	358	3	549	487
003	21	670	8	967	404
004	39	454	2	828	469

(a) PK scheme

PK	A_1	A_2	A_3	A_4	A_5
001	43	193	2	710	433
002	17	358	3	549	486
003	21	672	8	967	404
004	39	455	2	828	469

(b) VPK scheme

A_1 A_2 A_3 A_4 A_5

PK	L_1	L_2	L_3	L_4	L_5
001	43	192	2	710	432
002	17	358	3	549	487
003	21	671	9	967	404
004	39	455	2	828	469

(c) Our scheme

Fig. 1. Schematic of the general PK/VPK and our schemes. Blue and yellow cells represent the selected PK and generated VPK, respectively, used to locate and embed watermarks (indicated by red numbers). Our scheme uses both PK and VPK for watermarking and employs a classifier to label attributes to resist attribute name attacks. (Color figure online)

In this paper, we introduce attribute name attacks and mix-match attribute attacks, and propose a novel virtual primary key scheme that achieves high watermark synchronization and robustness. The scheme can resist the primary key, subset, and attribute attacks. Our scheme also exhibits high watermark capacity, reversibility, and enhanced practicality compared to the benchmarks. Figure 1 illustrates watermarking a database by the general PK/VPK scheme and our scheme. The contributions of this paper are as follows:

– We extend attribute attacks with attribute name attacks and mix-match attribute attacks. To resist these attacks, we train a classifier for attribute classification and attribute name recognition, which is used to maintain the original order and number of attributes.
– A novel virtual primary key scheme is proposed, which resists primary key attacks and mitigates synchronization problems. Specifically, it generates distinct virtual primary keys for locating and embedding marks.
– The proposed watermarking algorithm integrates primary key to guarantee high watermark synchronization when the primary key remains unchanged.
– Our algorithm performs comprehensive experiments to assess its robustness, distortion, and overhead. Compared to both VPK schemes (RRWC [3], M [19], HRW [7]) and PK schemes (AHK [1], RDMT [16], S^2R^2W [18]), our algorithm achieves stronger robustness, even when 98% of tuples or over half of the attributes are distorted.

The rest of this paper is organized as follows: Sect. 2 introduces previous related works. Section 3 shows an overview of our scheme. Section 4 details the process of the scheme. Section 5 presents the performance. Finally, Sect. 6 concludes our work with a prospect of future work.

2 Related Works

To protect the copyrights of databases, the first relational database watermarking scheme was proposed by Agrawal and Kiernan [1]. This scheme applies the primary key to decide which bit of the Least Significant Bits (LSB) from the selected attribute to be marked. While the algorithm is robust to subset attacks, it embeds a meaningless watermark. Subsequent studies [10,20,25] have enhanced the scheme, extending its use to fingerprints, embedding meaningful watermarks, and enabling reversibility. Sion et al. [24] proposed a statistical-based watermarking scheme to survive attacks while preserving data quality. The watermarks are embedded into actual data distribution properties (as opposed to [1] directly into the LSB of data). Subsequent studies [5,6,11,13,16,23] also studied the embedding method under a data distortion constraint. Optimization techniques, difference expansion, histogram shift, and other techniques are used to embed watermarks. Li et al. [18] analyzed attribute semantics and introduced two semantic-based watermarking schemes for numeric and non-numeric data.

The first virtual primary key (VPK) scheme was proposed by Agrawal and Kiernan [1] that considered their algorithm without primary key. Some schemes use non-primary key attributes for watermarking, which can be regarded as VPK schemes. Sebé et al. [22] proposed a watermarking scheme to preserve means and variances without applying primary key. The algorithm is performed repeatedly to embed a watermark to every attribute to address the attribute attacks. Odeh and Al-Haj [21] chose the ‘Time’ field to embed the watermark. Kamran and Farooq [14] ranked attributes for watermarking by assessing their classification potential and used Particle Swarm Optimization (PSO) to create a watermark that maintains statistical integrity. Chai et al. [3] assumed that some attributes are important and cannot be destroyed, and the watermark is embedded in others. Li et al. [19] proposed VPK schemes, including S-Scheme, E-Scheme, and M-Scheme. These schemes create virtual primary keys and locate the attribute values to be embedded by pseudo-random selecting attributes in tuples. These schemes do not depend on the primary key and the order of attributes. However, they introduced serious synchronization problems. [4,7–9] continued this study. Chang et al. [4] proposed a virtual primary key watermarking algorithm for textual relation. Gort et al. [7] proposed a novel method for decreasing duplicated values of virtual primary keys, but the algorithm is more vulnerable to attribute attacks than previous works. Gort et al. [8] introduced a high-quality virtual primary key generation principle and a VPK scheme under the principle. Gort et al. [9] proposed double fragmentation of a watermark by using the redundancy in the set of virtual primary keys, improving watermark synchronization. Nevertheless, the approach relies on tuple order; thus, subset attacks are ignored.

Research into watermarking schemes invariably includes an evaluation of watermark attacks. Kamran et al. [15] broadly classified these attacks as A_1 (Insertion attacks), A_2 (Deletion attacks), A_3 (Alteration attacks), A_4 (Multi-faceted attacks), A_5 (Additive (Re-watermarking) attacks), which the $A_1 \sim A_3$ are attacks on the tuples (records) of database relations. Kamran et al. [17] mentioned attacks like substitution, addition, alteration, vertical partition, invertibility, and Mix-Match, which contain the attacks on attributes (vertical partition). Yuan et al. [26] classifies basic attacks, including subset attacks (insertion attack, alteration attack, deletion attack) and attribute attacks.

3 Scheme Overview

3.1 Notations

In this section, the notations used in this paper are shown in Table. 1.

Table 1. Notations

Notation	Description	Notation	Description
D	Original database relation	PK	Primary key
D_s	Suspicious database relation	VPK	Virtual primary key
D_W	Watermarked database relation	LSB	Least significant bit
t	tuples of database relation	LSBF	LSB fraction
$\mathcal{A}^D$	Numeric attributes of D	K_s	Secret key
$\mathcal{A}$	Numeric attributes for watermark embedding	W	Watermark
$\mathcal{C}$	Candidate attributes for VPK generation	W_r	Recovered watermark
$\mathcal{C}'$	Candidate attributes of D_s after analyzing	ζ	Watermark length
$\Phi_{\mathcal{C}}$	Concatenated candidate attribute values	γ	Embedding density
$\Phi_{\mathcal{C}'}$	Concatenated candidate attribute values of D_s	ξ	Length of LSB

3.2 Process Overview

In this paper, our virtual primary key scheme for watermarking database relations will be used for the data owner's copyright verification of the shared database. Our scheme is divided into two parts: Watermark Embedding Process and Watermark Detection Process. Figure 2 illustrates the process overview.

Watermark Embedding Process. (i) In data preprocessing, it trains a classifier for attribute classification and recognition. This classifier is then utilized to maintain the original order and number of attributes, thereby mitigating attribute name attacks. (ii) After data preprocessing, it generates more distinct virtual primary keys than the existing literature, mitigating the synchronization problems. (iii) Then, two types of Embedding are performed: Logical Embedding and Physical Embedding. Logical Embedding applies the virtual primary keys to generate embedding records, which allows detecting watermarks using virtual primary key when suffering from primary key attacks. Physical Embedding applies primary key to embed and detect marks when primary key is available. In addition, a record table is generated to store auxiliary information for watermark detection. It will decide where and how to locate and extract marks. After

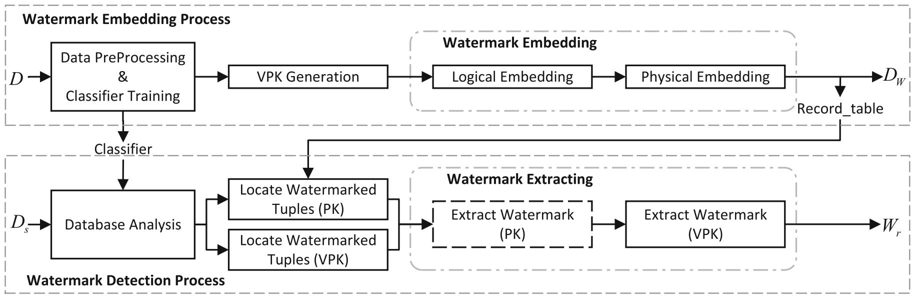


Fig. 2. Process Overview

embedding, the database relation D_w can be shared with the public. Thus, our embedding scheme owns the advantages of PK and VPK schemes, addressing synchronization problems and being robust against primary key attacks.

Watermark Detection Process. Before detecting the watermark for the suspicious database relation D_s, our scheme will analyze D_s. It determines whether or not to use the classifier to restore the attribute labels, and thereby maintain the original order and number of attributes. Furthermore, combining different locating methods, i.e., using the primary key and virtual primary key, it applies the record table for locating watermarked tuples. After that, the watermark is extracted using the reverse watermark embedding process.

3.3 Threat Model

The considered scenario involves two parties in the threat model: the *defender*, who is the data owner, and the *attacker*, a malicious user distributing data without authorization. The defender holds the copyright of the database relation and embeds a watermark before sharing it. The attacker may attack the watermarked database relation to remove the watermark. The scenario is shown in Fig. 3.

Defender's Assumption. The defender confidentially sets the secret parameters, a secret key K_s, and watermark information W. The defender employs the watermarking scheme to embed the watermark in database relation D, and can control the robustness and data distortion by parameters. During watermark detection, the defender can obtain the suspicious database relation D_s and use the watermarking scheme to extract watermarks for copyright verification.

Attacker's Goal. The attacker aims to remove the watermark while maintaining the data quality required for their purpose. The attacker then profits by selling databases without authorization or providing paid data access services (e.g., Paid APIs).

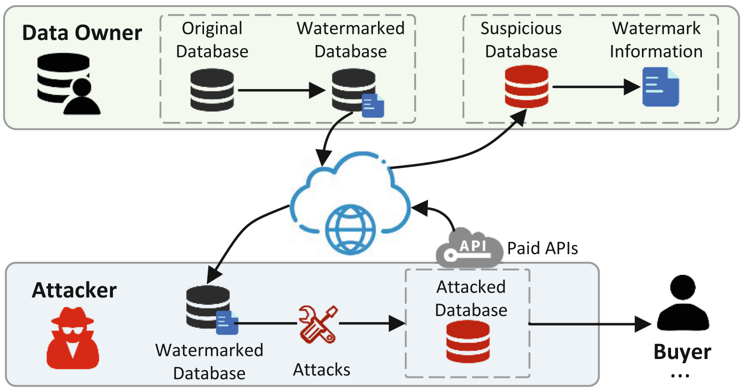


Fig. 3. Threat Scenario

Attacker's Capabilities. The attacker is aware that the database relation may contain a watermark and can modify the database as desired. The attacker can access the watermarking scheme but can not access the defender's secret parameters and secret key. In order to evade the copyright verification of database relation by the defender, the attacker may apply subset, additive, primary key, attribute attacks (including attribute name attacks and mix-match attribute attacks), alone or in combination, to remove the watermark.

4 Proposed Watermarking Scheme

In this section, we will detail attribute name attacks and the virtual primary key watermarking algorithm.

4.1 Attribute Attacks Extension

In this paper, we propose attribute name attacks and mix-match attribute attacks. We rename the current attribute attacks as attribute column attacks. Thus, attribute attacks can be classified into three types.

Attribute Column Attacks can be partially resisted by existing watermarking schemes (conscious or unconscious). They can be further mitigated by preserving the original order of attributes. Note that the primary key attacks constitute a form of attribute column attack, resulting in erroneous watermark detection in schemes reliant on the primary key.

Attribute Name Attacks can be divided into attribute name shuffling, deletion and substitution, which will affect the original order of attributes. These attacks can invalidate watermarking schemes that rely on the order of the attributes in watermark detection (e.g., most PK schemes, and part of VPK schemes [3,7,14,21,22]). Several VPK schemes [8,19] can resist these attacks due to the constructed virtual primary key, and the selected attributes to embed the watermark do not depend on the attribute order.

Mix-Match Attribute Attacks are combinations of attribute column attacks and attribute name attacks. These attacks are more complex and can affect both the original order and the number of attributes. Even these schemes [8,19] can not resist all attacks in mix-match attribute attacks.

To our knowledge, none of the existing watermarking schemes have fully addressed all the proposed attacks. The attackers can simply apply attribute name attacks to bypass watermark detection without modifying the data. We will further analyze the impact of these attacks on the existing watermarking schemes in the Sect. 5.

4.2 Data Preprocessing and Classifier Training

To enhance our scheme's resistance to the attribute attacks discussed in Sect. 4.1, we assign labels to attribute names and introduce a multi-classification task. This task aims to classify attributes and restore their labels (attribute names) in any database relation suspected of such attacks, preserving the original attribute order and count. Importantly, classifiers are utilized as an auxiliary measure only when subject to attribute name attacks. Datasets unaffected by these attacks do not require classifiers for attribute label restoration during detection. To train the classifier for the database relation D, data preprocessing is performed first.

Suppose the database relation D has a primary key attribute PK and numeric attributes $\mathcal{A}^D = \{A_0^D, A_1^D, ..., A_{n-1}^D\}$, the relation can be represented as $D = \{PK, \mathcal{A}^D\}$. The symbol A_i^D can be regarded as a label of ith attribute. We then choose each attribute's minimum, maximum, mean, variance, skewness, kurtosis, entropy, and label (symbol) as the feature vector to train the classifier. We apply random sampling, noise addition, and other operations to the tuples of D to create multiple datasets. Subsequently, feature extraction is conducted on these datasets to augment the training sample size, enhancing the classifier's generalization ability. In this paper, because the novelty of classification is not our contribution, we use the Random Forest [2] Classifier of scikit-learn to train the classifier. The classifier is then used to classify the attributes of the suspicious database relation D_s and restore the labels of the attributes. We tested the model using the datasets in Sect. 5, which includes both the watermarked and the post-attack watermarked datasets as test sets, achieving an accuracy of 96%.

During prediction, the attributes that do not belong to the original database may be misclassified. To address this problem, our scheme sets a threshold τ on the prediction probabilities of labels for the classifier to determine whether the classification result is reliable. This can mitigate the risk of erroneous watermark detection due to misclassification. To verify the effectiveness of τ, we selected numerical attributes from other real-world datasets and employed classifiers to calculate prediction probabilities. The highest prediction probabilities for misclassification labels are around 50%, while most of the highest prediction probabilities of the attributes of the test set are above 80%. Thus, users can empirically set τ to trade off misclassification and false negatives.

4.3 Virtual Primary Key Generation

In this proposed scheme, The attributes $\mathcal{A}^D$ of database relation $D(PK, \mathcal{A}^D)$ are divided vertically into two parts $\mathcal{A}$ and $\mathcal{C}$, where $\mathcal{A}^D = \mathcal{A} + \mathcal{C}$. The data owner secretly selects several attributes $\mathcal{C} = \{C_0, C_1, ..., C_{\nu-1}\}$ as the candiate attributes for virtual primary key generation. The remaining attributes $\mathcal{A} = \{A_0, A_1, ..., A_{\mu-1}\}$ are used to embed the watermark.

The virtual primary key generation method is shown in Algorithm 1. In order to reduce the number of duplicate values, we concatenate the candidate attribute values $t.\mathcal{C}$ with their corresponding labels, using a concatenation function $\varphi(a, b) = a \circ b$, where $\circ$ is the concatenation operator. These concatenated values are represented as $\Phi_{\mathcal{C}}$ as the input of the algorithm. In the lines 2-6, All possible combinations of the elements e in $\Phi_{\mathcal{C}}$ are generated and then hashed using a common one-way hash function $H(\cdot)$ to generate the virtual primary keys. The virtual primary key *vpk* associated with its corresponding element e is then stored in a dictionary *vpk_dict* with e as the key.

Algorithm 1. *vpk* generation

```
Input: $\Phi_{\mathcal{C}}$
Output: $vpk\_dict$
1: Initialize Dictionary $vpk\_dict$ with keys as elements of $\Phi_{\mathcal{C}}$ and empty sets as values
2: for $r$ from 1 to Len($\Phi_{\mathcal{C}}$) do
3:     for each combination $c$ of size $r$ from $\Phi_{\mathcal{C}}$ do
4:         for each element $e$ in combination $c$ do
5:             $vpk = H(c)$
6:             $vpk\_dict[e]$.append(vpk)
7: return $vpk\_dict$
```

In this way, multiple virtual primary keys can be constructed in a tuple, increasing the total count and the number of unique values of virtual primary keys. Therefore, watermark capacity is increased, and synchronization problems are mitigated. The database relation may suffer from attribute column deletion. As long as the $\mathcal{C}$ or $\mathcal{A}$ are not entirely deleted, the watermark can be extracted using the virtual primary key.

4.4 Watermark Embedding

Watermarking schemes that utilize the primary key are known for high synchronization. Therefore, when the primary key is available, its use in watermark detection can enhance the practicality of the virtual primary key scheme. The proposed VPK scheme aims to integrate the primary key for watermarking. To achieve this, three critical considerations must be addressed in the watermark embedding process: 1) The selection of the watermark position for a tuple t should not be linked to the primary or virtual primary keys; 2) The embedding of the watermark using both the primary key and virtual primary key must not interfere with each other, preventing the detection of incorrect marks; 3) Integrating watermarking with the primary key should avoid excessive data distortion, as significant distortion can compromise the scheme's usability.

Algorithm 2. watermark embedding

```
Input: $D$, $K_s$, $\mathcal{C}$, $\mathcal{A}$, $W$, $\zeta$, $LSBF$
Output: $D_W$
 1: function WM_EMBED($D$, $K_s$, $\mathcal{C}$, $\mathcal{A}$, $W$, $\zeta$, $LSBF$)
 2:     for each $t \in D$ do
 3:         if $H(K_s \circ t.PK) \mod \gamma \neq 0$ then
 4:             Continue
 5:         $t.A_{w_0}, t.A_{w_1}$ ←Select values according to Eq.(1)
 6:         $vpk\_dict$ = vpk_generation($\Phi_{\mathcal{C}}$)
 7:         $idx = H(K_s \circ t.PK) \mod \zeta$
 8:         $leb\_dict$ = LOGICAL_EMBED($vpk\_dict$, $idx$, $K_s$, $\zeta$)
 9:         $x\_dict$ = PHYSICAL_EMBED($t.\mathcal{A}$, $idx$, $W$, $K_s$, $LSBF$)
10:         record $\{t.PK, leb\_dict, x\_dict\}$ into record_table
11:     return $D_W$
12: function LOGICAL_EMBED($vpk\_dict$, $idx$, $K_s$, $W$, $\zeta$)
13:     Initialize Dictionary $leb\_dict$ with keys of $vpk\_dict$ as keys and an empty binary
        string as value
14:     for $e$, $vpks$ in $vpk\_dict$ do
15:         for each $vpk$ in $vpks$ do
16:             $idx' = H(K_s \circ vpk) \mod \zeta$
17:             if $W[idx] == W[idx']$ then
18:                 $leb\_dict[e]$.join('0')
19:             else
20:                 $leb\_dict[e]$.join('1')
        return $leb\_dict$
21: function PHYSICAL_EMBED($t.\mathcal{A}$, $idx$, $W$, $K_s$, $LSBF$)
22:     Initialize Dictionary $x\_dict$ to record $t.A_{w_i}$ and the change of $t.A_{w_i}$
23:     for each $t.A_{w_i}$ in $\{t.A_{w_0}, t.A_{w_1}\}$ do
24:         $\xi$ = ROUND(Len($t.A_{w_i}$) × $LSBF$)
25:         $bitIndex = H(K_s \circ \text{Len}(t.A_{w_i})) \mod \xi$
26:         Set the bit in LSB of $t.A_{w_i}$ to $W[idx]$ according to $bitIndex$
27:         Set a marker $ch = True$ if $t.A_{w_i}$ changed else $ch = False$
28:         $x\_dict[t.A_{w_i} \circ A_{w_i}] = ch$
        return $x\_dict$
```

The watermark embedding process is shown in Algorithm 2. In this paper, we integrate watermarking with primary key that embeds the watermark into the least significant bit (LSB) of attribute values. This algorithm alters the bit-level values with minimal data distortion and ensures high watermark synchronization. To satisfy the first consideration, two attribute values for a tuple t are selected to embed a watermark bit according to Eq.(1).

$$\begin{aligned} \mathcal{H}(t.\mathcal{A}) = \{H(K_s \circ t.A_0), H(K_s \circ t.A_1), ..., H(K_s \circ t.A_{\mu-1})\} \\ t.A_{w_0} = \min(\mathcal{H}(t.\mathcal{A})), t.A_{w_1} = \min(\mathcal{H}(t.\mathcal{A}) \setminus \{t.A_{w_1}\}) \end{aligned} \tag{1}$$

Each attribute value of $t.\mathcal{A}$ is hashed with secret key K_s, and the two smallest values are selected for watermark embedding. The algorithm's security hinges on the one-way hash function and the secret key; the watermark remains secure as long as K_s is not disclosed. The rationale for selecting these two values will be detailed in the Watermark Detection Sect. 4.5.

The watermark embedding process is divided into two parts: Logical Embedding and Physical Embedding, in order to satisfy the second and third considerations. Watermark bits selected by primary key are physically embedded, while those selected by virtual primary key are logically embedded. The reason for applying Logical Embedding is the proposed scheme generates multiple virtual primary keys per tuple. Physically embedding watermark bits selected by virtual primary keys would result in significant data distortion. Line 7 of Algorithm 2 selects the position idx of a watermark bit from the watermark W through $t.PK$, K_s, and the length ζ of W. Then, line 8 performs the Logical Embedding according to $W[idx]$ and the virtual primary keys. The detail about Logical Embedding is shown in the function LOGICAL_EMBED. This function first initializes a dictionary *leb_dict* for logical embedding: *leb_dict* utilizes the same keys as *vpk_dict*, establishing a direct mapping between the two. For each key in *leb_dict*, the corresponding value is initialized as an empty binary string, whose length is determined by the number of values (*vpks*) associated with that key in *vpk_dict*. After that, each *vpk* for each key e in the *vpk_dict* is traversed to select the bit position idx' from the watermark W using *vpk*. Then, comparing $W[idx']$ with $W[idx]$, if the two bits are the same, the logical embedding bit is 0; otherwise, it is 1. The logical embedding bit is then stored sequentially in the binary string of the *leb_dict*. Therefore, each character of the string can be mapped to a vpk. Line 9 performs the Physical Embedding. The detail about Physical Embedding is shown in the function PHYSICAL_EMBED. $W[idx]$ is embedded into the LSB of the attribute value $t.A_{w_i}$ according to the *bitIndex*. Noting that the length ξ of LSB is not fixed, it is computed according to Eq.(2).

$$\xi = \text{ROUND}(\text{Len}(t.A_{w_i}) \times LSBF) \tag{2}$$

The length of LSB is proportional to the length of the attribute value $t.A_{w_i}$ and the least significant bit fraction (LSBF). In this way, each $t.A_{w_i}$ own its corresponding ξ. Thus, the watermark bits are embedded in distinct positions for different attribute values. The *bitIndex* is computed according to Eq.(3).

$$bitIndex = H(K_s \circ \text{Len}(t.A_{w_i})) \mod \xi \tag{3}$$

During embedding, the change to $t.A_{w_i}$ is stored in a dictionary x_dict, with keys formed by concatenating $t.A_{w_i}$ and its label A_{w_i}. A record comprising $t.PK$, leb_dict, and x_dict is then stored in a record_table. This table is kept locally as metadata by the data owner and provided for watermark detection when needed for copyright verification. After traversing the database, it is watermarked.

4.5 Watermark Detection

Watermarked database relation D_W is used for sharing. However, most sharing channels are insecure, and D_W may face various modifications and attacks. Data owners need to verify the copyright of the suspicious database D_s. Detecting the embedded watermark W in D_s serves as essential proof of copyright.

In the proposed scheme, an initial analysis of D_s is required before watermark detection. This analysis determines if D_s has undergone attribute attacks and if a classifier is needed to restore attribute labels. And finalize the availability of $\mathcal{C}$ and $\mathcal{A}$, along with the primary key attributes. After analysis, the candidate attributes of D_s are denoted as $\mathcal{C}'$.

During the watermark detection process, the crux is locating the correct watermarked tuples. Each tuple t of D_s is traversed to match the record r in the record_table. According to the analysis, matches will be divided into the following two cases: primary key is available, and primary key is unavailable.

- **The Primary Key is Available:** If the primary key $t.PK$ matches a record r, and at least one of the elements of the $x\_dict.key$ of r equals the corresponding $t.A_{w_i} \circ A_{w_i}$ in t, then tuple t is identified as a watermarked entity and subsequently employed in the extraction of watermark bits.
- **The Primary Key is Unavailable:** False-matched problems need to be considered. If matching process relies solely on $\mathcal{C}'$ to align with $leb\_dict.key$ in record_table, it may result in false matching tuples (the more candidate attributes affect, the more records may falsely match). To minimize false-matched tuples, matching $x\_dict.key$ to the corresponding $t.A_{w_i} \circ A_{w_i}$ in t is essential. Only tuple t that $t.\mathcal{C}'$ and $t.A_{w_i} \circ A_{w_i}$ match a record can extract the watermark. The purpose of selecting two attribute values for embedding is to mitigate the false-matched problems. Despite the possibility that this matching method may miss some watermarked tuples, the integrity of the watermark can still be preserved. This is because our scheme's ample and distinct virtual primary keys ensure watermark redundancy.

The given examples of the two cases are shown in Table. 2 and Table. 3.

The pseudocode of watermark detection is shown in Algorithm 3. The algorithm first matches the record_table to locate the record r corresponding to the tuple t. Line 5 selects the $t.A_{w_i}$ for watermark extraction only if the $x\_dict[t.A_{w_i} \circ A_{w_i}]$ of r is true, which ensures that the extracted bits are definitely the correct embedded watermark bits. Therefore, our scheme obviates the need for iterating through all tuples and employing a majority voting mechanism to restore the correct watermark. The algorithm terminates once the watermark W_r

Table 2. Matching Process if Primary Key is Available

Matched PK	A_{w_i} of x_dict	$t.A_{w_i}$ of x_dict	$t.A_{w_i}$ of tuple t	Located watermarked tuple
9	A_{w_0}, A_{w_1}	7,3	7,3	True
12	A_{w_0}, A_{w_1}	24,5	22,7	False
37	A_{w_0}, A_{w_1}	13,2	13,3	True

Table 3. Matching Process if Primary Key is Unavailable

$\Phi_{\mathcal{C}'}$	$\Phi_{\mathcal{C}}$ of leb_dict	A_{w_i} of x_dict	$t.A_{w_i}$ of x_dict	$]t.A_{w_i}$of tuple t	Located watermarked tuple
$234 \circ C_i$	$234 \circ C_i$,$32 \circ C_j$	A_{w_0}, A_{w_1}	7,3	7,3	True
$256 \circ C_i$	$256 \circ C_i$,$54 \circ C_j$	A_{w_0}, A_{w_1}	24,5	22,7	False
$197 \circ C_i$	$197 \circ C_i$,$69 \circ C_j$	A_{w_0}, A_{w_1}	13,2	13,3	False

Algorithm 3. watermark detection

```
Input: $D_s$, $K_s$, $\mathcal{C}'$, $\mathcal{A}$, $\zeta$, $LSBF$, record_table
Output: $W_r$
 1: for each tuple $t$ in $D_s$ do
 2:     if $W_r$ is recovered then
 3:         Break
 4:     Query the $record\_table$ to locate $r$
 5:     Select $t.A_{w_i}$ where $r.x\_dict[t.A_{w_i} \circ A_{w_i}] = True$
 6:     $\xi = \text{ROUND}(\text{Len}(t.A_{w_i}) \times LSBF)$
 7:     $bitIndex = H(K_s \circ \text{Len}(t.A_{w_i})) \mod \xi$
 8:     Gain the bit $b$ in LSB of $t.A_{w_i}$ according to $bitIndex$
 9:     if $PK$ attribute is available then
10:         $idx = H(K_s \circ t.PK) \mod \zeta$
11:         if $W_r[idx]$ is empty, set $W_r[idx] = b$
12:     $vpk\_dict = \text{vpk\_generation}(r.leb\_dict.keys)$
13:     for each element $e$ in $\Phi_{\mathcal{C}'}$ do
14:         for each $vpk$ in $vpk\_dict[e]$ do
15:             $idx' = H(K_s \circ vpk) \mod \zeta$
16:             if $W_r[idx']$ is empty, do the following steps
17:             $leb\_str = r.leb\_dict[e]$
18:             $leb \leftarrow$Map the $vpk'$ to the corresponding position in $leb\_str$
19:             if $leb == 0$ then
20:                 $W_r[idx'] = b$
21:             else
22:                 $W_r[idx'] = \neg b$
23: return $W_r$
```

has been successfully recovered. Lines 6-8 locate the watermark bit b in the LSB of $t.A_{w_i}$ according to the $bitIndex$. If the primary key is available, the algorithm will compute the idx and set the $W_r[idx]$ to b. Lines 9-18 show the watermark extraction by virtual primary keys. The vpk_dict is generated according to the $leb\_dict.keys$ of r. $\Phi_{\mathcal{C}'}$ is calculated according to $\mathcal{C}'$. For each element in $\Phi_{\mathcal{C}'}$, traversing each vpk in $vpk\_dict[e]$ and map the vpk to the corresponding position in $r.leb\_dict[e]$ to get the logical embedding bit leb. If leb is 0, the $W_r[idx']$ is set to b, otherwise it is set to $\neg b$. It is worth noting that the watermarked values can be restored to their original value by $ch = True$ and the $bitIndex$ in record_table. Thus, the proposed scheme is reversible.

5 Performance Experiment Results

In this section, we design experiments to validate the proposed scheme's performance, focusing on attack robustness, overhead, and data distortions.

Experiments were conducted on a computer with a 1.9GHz CPU and 16GB of RAM, running Windows 11. The Forest Cover Type[1] dataset is used to perform experiments, which has 581,012 tuples, each with 55 attributes. We added an extra attribute ID as the primary key and then chose the first ten numerical attributes and one categorical attribute for experiments. To test the robustness of our algorithm under demanding conditions, we used settings of LSBF $= 0.3$ and $\gamma = 200$. Thus, around 2900 tuples are selected by the algorithm for watermarking. To ensure comparability, we adjusted parameters in benchmark watermark algorithms to maintain a similar count of watermarked tuples. We randomly selected 2 numerical attributes and 1 categorical attribute $\{\text{Aspect}, \text{Hillshade\_Noon}, \text{Cover\_Type}\}$ as candidate attributes $\mathcal{C}$.

Attack resistance is quantified using the Correction Factor (CF), which represents detection accuracy by comparing the binary watermark string extracted from the suspect database to the original at the embedding stage. CF is calculated as CF $= \frac{\sum_{i=0}^{\zeta-1} W[i] \oplus W_r[i]}{\zeta} \times 100\%$, where i represents the ith bit of the watermark string. A CF near or at 100% confirms the copyright of the suspected database, demonstrating the efficacy of the watermark detection algorithm.

5.1 Robustness Verification

Results of Virtual Primary Key Watermarking Algorithm. As a VPK scheme, the proposed algorithm is compared with Chai's [3] scheme (denoted by RRWC) and M-scheme of Li's [19] scheme (denoted by M) and Gort's scheme [7] which based on M (denoted by HRW). In this section, attacks are performed, and the primary key attribute is assumed to be unavailable by default.

To assess synchronization problems [8], we compared our scheme's number of virtual primary keys (VPKs) and unique VPK values with the benchmark schemes. Furthermore, Let ω_i be the actual times that each watermark bit $W[i]$

[1] Forest CoverType: http://kdd.ics.uci.edu/databases/covertype/covertype.html.

is embedded $(i = 0, ..., \zeta)$, and $\omega_{max} = \max \omega_i, \omega_{min} = \min \omega_i$. We use ω_{max} and ω_{min} to measure the robustness of the algorithm. The smaller the ω_{min} and ω_{max}, the more easily the marked bits are corrupted by users' conscience updates or various attacks by attackers. Table. 4 shows the results of the number of VPK/UniqueVPK/$\omega_{\max}$/$\omega_{\min}$ values of algorithms.

Table 4. The number of VPK/Unique VPK/ω_{max}/ω_{min} of algorithm.

Algorithm	VPK	Unique VPK	ω_{max}	ω_{min}
Proposal	34761	3340	2331	689
M	3191	83	1326	0
HRW	2872	1181	161	33
RRWC	2906	2904	78	78

Table 4 indicates that our scheme generates more (and unique) virtual primary keys compared to the benchmarks, enhancing watermark capacity and mitigating synchronization problems. It is worth noting that the RRWC uses the Euclidean Distance $CalDistance(t)$ of tuple t's *important attributes* as a virtual primary key. RRWC groups the database by tuples and embeds a mark in each group, achieving $\omega_{max} = \omega_{min}$.

We then performed subset and attribute column attacks on the database marked by our algorithm and the benchmarks. Preserving the original attribute order during watermark detection neutralizes shuffling and insertion attacks and equates attribute substitution to deletion. Thus, our experiments concentrated on attribute deletion attacks. The results of the CF are shown in Figure. 4.

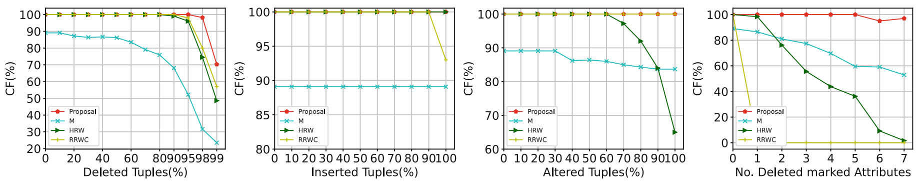


Fig. 4. Results of algorithms against subset and attribute deletion attacks.

As observed, the CF of our algorithm, RRWC, and HRW remain over 95%, even if 95% of tuples are deleted. The CF of M is below 90% at the beginning because its $\omega_{min} = 0$, indicating the significant synchronization problem leads to incomplete watermark embedding. Subset insertion and alternation only slightly affect our proposal, RRWC, and M. However, the CF of HRW steeply decreases when over 60% of tuples are altered. This is caused by major data modifications that disrupt the consistency between the virtual primary keys used in detection and embedding, leading to incorrect watermark bits recovery. Our algorithm outperforms M, HRW, and RRWC in attribute column deletion. The CF is still close to 100% when over half of the attributes are deleted. Nevertheless, with the deletion of 7 attributes, the CF of our algorithm exhibits a slow upward trend due to disruptions in tuple locating, which increases false matches. HRW is susceptible to severe attribute attacks, again due to the generation of incorrect virtual primary keys during detection. Moreover, the CF of RRWC drops to zero when *important attributes* are affected, failing to resist attribute attacks.

The robustness of algorithms against multifaceted attacks (subset attacks combined with attribute column attacks) was evaluated, as shown in Fig. 5. Our algorithm is still robust compared to the benchmarks. However, when over 80% of tuples are deleted and more than 5 attribute columns are affected, the CF significantly decreases. This indicates the challenge of fully mitigating the impact of a missing primary key on watermark synchronization.

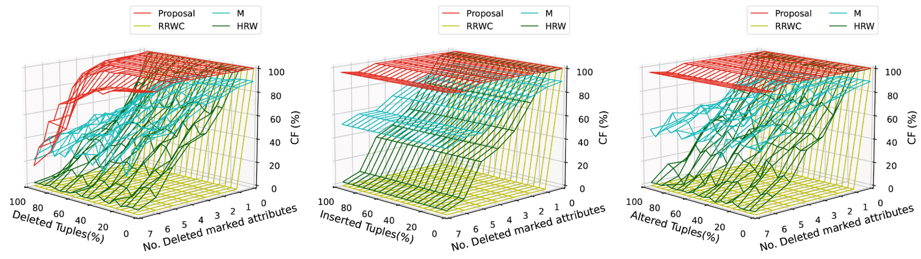


Fig. 5. Results of algorithms against multifaceted attack.

Attribute name attacks are invalid for our algorithm since the classifier restores attribute labels, and maintains their order and count. M is also immune to these attacks because it does not rely on the order of attributes. However, HRW and RRWC rely on the original order of attributes, and such attacks disrupt their watermark detection process.

The resistance of our algorithm to mix-match attribute attacks, facilitated by the classifier, is shown in Fig. 6. HRW and RRWC, vulnerable to attribute name attacks, cannot withstand these attacks. For M, the effects of combined attribute name and deletion/substitution attacks align with those from sole attribute deletion attacks. However, when attribute name attacks are combined with insertion attacks, impacting both the order and number of attributes, M struggles to maintain the original attribute count, resulting in a decreased CF.

Results with Primary Key Watermarking Algorithm. Given that our scheme integrates watermarking with the primary key, we compared it to PK schemes to evaluate its performance when primary key is available. Our scheme is compared with the Agrawal-Kiernan's [1] scheme (denoted by AHK), Kamran's [16] scheme (denoted by RDMT) and Li's [18] scheme (denoted by S^2R^2W).

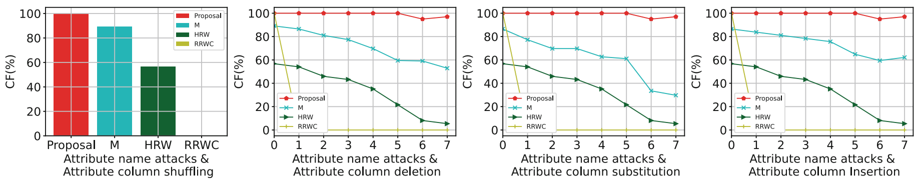


Fig. 6. Results of algorithms against mix-match attribute attacks.

Subset and attribute deletion attacks are performed the same in Sect. 5.1. Figure 7 shows the results of the experiments. Our algorithm slightly outperforms benchmark schemes in resistance to these attacks, thanks to its dual use of primary and virtual primary keys for watermark detection when the primary key is available. This allows our scheme to extract more marks than traditional PK schemes, thus enhancing its robustness. We can see that PK schemes like AHK and RDMT are vulnerable to attribute column deletion attacks.

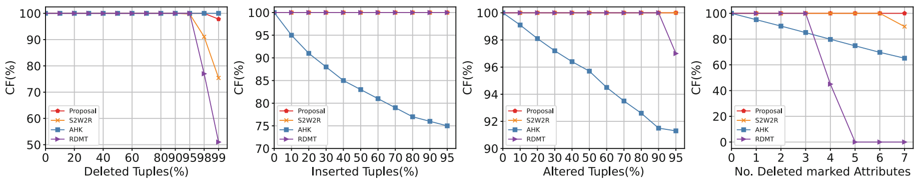


Fig. 7. Results of algorithms against subset and attribute deletion attacks.

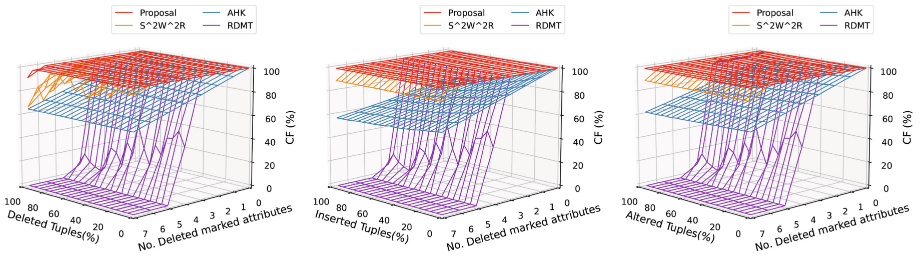


Fig. 8. Results of algorithms against multifaceted attack.

Figure. 8 illustrates the robustness of algorithms against multifaceted attacks. Our algorithm remains robust, maintaining a CF of 90% despite the deletion of 7 attributes and over 95% of tuples. Insertion and alternation attacks with attribute attacks only slightly affect the CF. This desirable behavior is attributed to the vast marks embedded during the embedding process, as well as the record table storing auxiliary information to exclude false detection. S^2R^2W also shows improved performance compared with AHK and RDMT but is slightly less effective than our proposal. RDMT shows a steep decrease in CF due to attribute attacks destroying the once-for-all constraints, and the CF of AHK decreases as the number of affected attributes and tuples increases.

In addition, it is worth noting that these benchmark algorithms can not resist attribute name attacks, as do the mix-match attribute attacks, because they all rely on the original order of attributes.

5.2 Usability Verification

Data Distortions. Data distortions between the original and watermarked datasets are measured using statistical values like mean difference ($\Delta\mu$) and standard deviation difference ($\Delta\sigma$) of each marked attribute, shown in Table. 5. One can easily notice that the $\Delta\mu$ and $\Delta\sigma$ are small enough to be undetectable, suggesting that our proposed scheme minimally impacts data availability.

Table 5. Distortions introduced in the attributes.

Attribute	$\Delta\mu$	$\Delta\sigma$
Elevation	-1.89e-4	2.33e-4
Slope	-1.6e-4	7.98e-05
Horizontal_Distance_To_Hydrology	1.31e-4	-2.57e-4
Vertical_Distance_To_Hydrology	1.39e-4	-6.24e-6
Horizontal_Distance_To_Roadways	1.08e-4	-4.66e-7
Hillshade_9am	8.9e-5	8.56e-5
Hillshade_3pm	-9.8e-5	2.78e-5
Horizontal_Distance_To_Fire_Points	4.6e-5	1.81e-4

Overhead. Two experiments in Fig. 9 assess the computational costs of watermark embedding and detection across different numbers of tuples on selected configurations and datasets. Disk read and write operations are excluded from execution times. Our scheme shows lower execution times compared to benchmarks, and its detection time is shorter than the embedding time because the detection process stops once the watermark is successfully recovered. Execution times increase with the number of tuples. For $0 \leq$ tuples ≤ 500000, our scheme, AHK, and RDMT have execution times under 1 s, whereas RRWC, HRW, M, and S^2W^2R exhibit longer execution times.

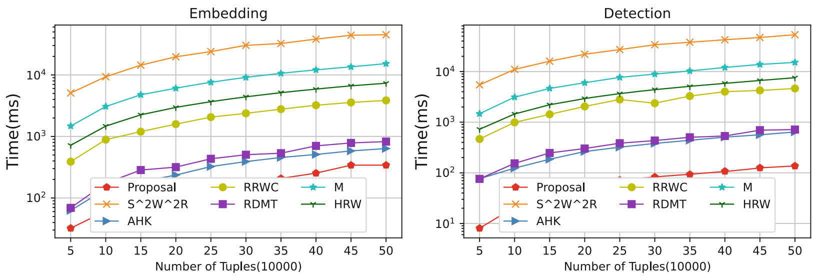


Fig. 9. Execution times of algorithms.

6 Conclusion

In this paper, attribute name attacks and mix-match attribute attacks are introduced, and a virtual primary key scheme is proposed. In order to resist the introduced attacks, the scheme introduces a classifier to maintain the original order and number of attributes. Our scheme generates ample and unique virtual primary keys for resisting primary key attacks and mitigating synchronization problems. The scheme integrates watermarking with the primary key to ensure high detection accuracy when the primary key is available. The results of our performance experiments on a real-world dataset substantiate our claims. The scheme features high watermark synchronization and capacity, robustness, and reversibility. These characteristics make it practical in various scenarios.

In future research, we aim to optimize the watermark embedding method by incorporating data distribution properties. Additionally, we plan to extend the scheme to include categorical and non-numeric data.

Acknowledgments. This work was supported by the Strategic Priority Research Program of the Chinese Academy of Sciences, Grant No. XDB0690303.

References

1. Agrawal, R., Kiernan, J.: Watermarking relational databases. In: VLDB'02: Proceedings of the 28th International Conference on Very Large Databases, pp. 155–166. Elsevier (2002)
2. Breiman, L.: Random forests. Mach. Learn. **45**, 5–32 (2001)
3. Chai, H., Yang, S., Jiang, Z.L., Wang, X.: A robust and reversible watermarking technique for relational dataset based on clustering. In: 2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), pp. 411–418. IEEE (2019)

4. Chang, C.-C., Nguyen, T.-S., Lin, C.-C.: A blind robust reversible watermark scheme for textual relational databases with virtual primary key. In: Shi, Y.-Q., Kim, H.J., Pérez-González, F., Yang, C.-N. (eds.) Digital-Forensics and Watermarking: 13th International Workshop, IWDW 2014, Taipei, Taiwan, October 1-4, 2014. Revised Selected Papers, pp. 75–89. Springer International Publishing, Cham (2015). https://doi.org/10.1007/978-3-319-19321-2_6
5. 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)
6. Franco-Contreras, J., Coatrieux, G., Cuppens, F., Cuppens-Boulahia, N., Roux, C.: Robust lossless watermarking of relational databases based on circular histogram modulation. IEEE Trans. Inf. Forensics Secur. **9**(3), 397–410 (2013)
7. Gort, M.L.P., Díaz, E.A., Uribe, C.F.: A highly-reliable virtual primary key scheme for relational database watermarking techniques. In: 2017 International Conference on Computational Science and Computational Intelligence (CSCI), pp. 55–60. IEEE (2017)
8. Gort, M.L.P., Feregrino-Uribe, C., Cortesi, A., Fernández-Peña, F.: Hqr-scheme: a high quality and resilient virtual primary key generation approach for watermarking relational data. Expert Syst. Appl. **138**, 112770 (2019)
9. Gort, M.L.P., Feregrino-Uribe, C., Cortesi, A., Fernández-Peña, F.: A double fragmentation approach for improving virtual primary key-based watermark synchronization. IEEE Access **8**, 61504–61516 (2020)
10. Hou, R., Xian, H.: A graded reversible watermarking scheme for relational data. Mobile Netw. Appl. **26**, 1552–1563 (2021)
11. Hu, D., Zhao, D., Zheng, S.: A new robust approach for reversible database watermarking with distortion control. IEEE Trans. Knowl. Data Eng. **31**(6), 1024–1037 (2019)
12. 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 (2014)
13. Jawad, K., Khan, A.: Genetic algorithm and difference expansion based reversible watermarking for relational databases. J. Syst. Softw. **86**(11), 2742–2753 (2013)
14. Kamran, M., Farooq, M.: An information-preserving watermarking scheme for right protection of emr systems. IEEE Trans. Knowl. Data Eng. **24**(11), 1950–1962 (2011)
15. Kamran, M., Farooq, M.: A comprehensive survey of watermarking relational databases research. arXiv preprint arXiv:1801.08271 (2018)
16. Kamran, M., Suhail, S., Farooq, M.: A robust, distortion minimizing technique for watermarking relational databases using once-for-all usability constraints. IEEE Trans. Knowl. Data Eng. **25**(12), 2694–2707 (2013)
17. Kumar, S., Singh, B.K., Yadav, M.: A recent survey on multimedia and database watermarking. Multimedia Tools Appl. **79**(27), 20149–20197 (2020)
18. Li, W., Li, N., Yan, J., Zhang, Z., Yu, P., Long, G.: Secure and high-quality watermarking algorithms for relational database based on semantic. IEEE Transactions on Knowledge and Data Engineering (2023)
19. Li, Y., Swarup, V., Jajodia, S.: Constructing a virtual primary key for fingerprinting relational data. In: Proceedings of the 3rd ACM Workshop on Digital Rights Management, pp. 133–141 (2003)
20. Li, Y., Swarup, V., Jajodia, S.: Fingerprinting relational databases: schemes and specialties. IEEE Trans. Dependable Secure Comput. **2**(1), 34–45 (2005)

21. Odeh, A., Al-Haj, A.: Watermarking relational database systems. In: 2008 First International Conference on the Applications of Digital Information and Web Technologies (ICADIWT). pp. 270–274. IEEE (2008)
22. Sebé, F., Domingo-Ferrer, J., Solanas, A.: Noise-Robust Watermarking for Numerical Datasets. In: Torra, V., Narukawa, Y., Miyamoto, S. (eds.) MDAI 2005. LNCS (LNAI), vol. 3558, pp. 134–143. Springer, Heidelberg (2005). https://doi.org/10.1007/11526018_14
23. Shehab, M., Bertino, E., Ghafoor, A.: Watermarking relational databases using optimization-based techniques. IEEE Trans. Knowl. Data Eng. **20**(1), 116–129 (2007)
24. Sion, R., Atallah, M., Prabhakar, S.: Rights protection for relational data. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of data, pp. 98–109 (2003)
25. Wang, H., Cui, X., Cao, Z.: A speech based algorithm for watermarking relational databases. In: 2008 International Symposiums on Information Processing, pp. 603–606. IEEE (2008)
26. Yuan, S., Chen, C., Yang, K., Yang, T., Yu, J.: An attribute-attack-proof watermarking technique for relational database. In: 2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 1136–1143. IEEE (2022)