symmetry

MDPI

*Article*

# LRW-CRDB: Lossless Robust Watermarking Scheme for Categorical Relational Databases

**Chia-Chen Lin [1,*], Thai-Son Nguyen [2] and Chin-Chen Chang [3,*]**

[1] Department of Computer Science and Information Engineering, National of Chin-Yi University of Technology, Taichung 411030, Taiwan
[2] Department of Information Technology, Tra Vinh University, Tra Vinh 940000, Vietnam; thaison@tvu.edu.vn
[3] Department of Information Engineering and Computer Science, Feng Chia University, Taichung 40724, Taiwan
* Correspondence: ally.cclin@ncut.edu.tw (C.-C.L.); ccc@o365.fcu.edu.tw (C.-C.C.)

**Abstract:** In 2002, Agrawal and Kiernan defined six basic requirements, including preventing illegal watermark embedding and authentication, reversibility, robustness, and others, which must be satisfied when a reversible watermark is designed for relational databases. To meet these requirements, in this paper, a lossless watermarking scheme for a categorical relational database called LRW-CRDB (lossless robust watermarking for categorical relational databases) is proposed. In our LRW-CRDB scheme, the database owner needs to generate two secret embedding keys, $K_1$ and $K_2$, in advance. Then, two reference sets are generated based on two different secret embedding keys and a symmetry-based data hiding strategy, and then these are used for the watermark embedding phases. Experimental results confirmed that our LRW-CRDB scheme successfully detects 100% of hidden watermarks, even when more than 95% of the watermarked relational database has been deleted. In other words, the robustness of our proposed LRW-CRDB scheme outperforms other existing schemes under a variety of possible attacks, such as alteration, sorting, deletion, and mix-match attacks.

**Keywords:** blind; lossless watermark; relational database; robustness; categorical data



**Citation:** Lin, C.-C.; Nguyen, T.-S.; Chang, C.-C. LRW-CRDB: Lossless Robust Watermarking Scheme for Categorical Relational Databases. *Symmetry* **2021**, *13*, 2191. https://doi.org/10.3390/sym13112191

Academic Editor: Yu-Chi Chen

Received: 27 September 2021
Accepted: 8 November 2021
Published: 17 November 2021



## 1. Introduction

With the rapid development of the Internet and digital processing technologies, the ownership protection of digital content, such as an image, audio, videos, and so on, has become a crucial issue. As an important research topic, various approaches are employed, i.e., data hiding [1–5] and watermarking techniques [6–13], to protect the legal owner and protect digital data from illegal manipulations. The main objectives of the above approaches can be achieved by concealing watermarks, such as a company's brand logo or an owner's personal signature, in the original content, so that the authenticated owners can claim their ownership of the digital content by successfully extracting the hidden watermark.

Basically, watermarking techniques can be divided into three types. The first type is a robust watermarking technique [9–11], in which the hidden watermark can successfully withstand malicious attacks. The second type is the fragile watermarking technique [6–8], in which the hidden watermark will be easily affected by various operations, no matter if they are conducted by malicious attackers or innocent users. The third type of watermarking technique [12,13] is called semi-fragile watermarking, and it can resist benign transformations, such as compression, but not malignant transformation.

Over the past decade, the applications of watermarking have been extended to databases, and many watermarking schemes designed for the numerical part of a relational database have been proposed [14–25]. The relational database can be a customer database that contains various customer attributes that a company holds of its customers, such as the customer's name, e-mail, phone number, Zip code, address and so on. Most of the existing watermarking schemes [14–16,19–21,23–25] are based on irreversible watermarking; in

such schemes, the original relational databases can be distorted permanently and cannot be reconstructed, even if the hidden watermarks have been authenticated. However, in certain types of application, e.g., military troop location information, medical x-rays, and satellite aerial photographs, it is important that the original relational database can be restored without any distortion after the secret message has been extracted. As a result, various reversible watermarking techniques [17,18,22] have been proposed, by applying reversible data hiding techniques. Reversible watermarking allows the restoration of the relational database to its original situation after the embedded watermark has been extracted and verified. In addition to the above application, another aim of reversible watermarking techniques is to provide shareware versions of specific database applications, so that the original database can be restored only when the customer buys a regular license for his/her given application.

In 2002, the first well-known database watermarking scheme for relational databases was proposed by Agrawal and Kiernan [14]. The basic idea of their scheme is to modify the numerical attributes in a relational database to embed a watermark. Their requirement was that the watermarked relational database can tolerate a small number of errors after the embedding of the watermark. Agrawal and Kiernan's scheme is robust under four types of attack, i.e., alteration, sorting, deletion, and mix-match attacks. In addition, they also guarantee that the mean and variance of all the numerical attributes will be tiny. In other words, the modified attributes' values will be slightly different from the original attributes' values. However, their scheme cannot be directly used for embedding watermarks into categorical data, because any bit modification of a categorical value may make the attribute's value meaningless. To solve the weaknesses of Agrawal and Kiernan's scheme, in 2004, Sion [15] proposed a watermarking technique for categorical attributes of the relational database, by modifying their current values to different values of the attribute. In his scheme, small modifications to the database are acceptable if the small modifications do not substantially affect the content of the categorical attributes. In 2008, Shehab et al. [17] proposed a new watermarking technique that was based on an optimization-based technique for numerical attributes in the relational database. The relational database was divided into non-overlapping partitions by using a secret key, K. Then, one watermark bit was embedded into each partition by modifying the partition statistics. Their scheme can protect against alteration, deletion, and insertions attacks. Moreover, Shehab et al.'s scheme is efficient when the relational data in certain applications allow a small change in some of the numerical values. In the same year, Al-Haj and Odeh [18] proposed a new database watermarking scheme that based on inserting a binary image watermark into the non-numeric multi-word attributes of selected tuples. Their scheme is a blind technique, which means that their scheme does not need the original database for extracting the hidden watermark. However, their scheme successfully detects the watermarked data with 100% accuracy only when the modification is smaller than 10% of the content of the watermarked relational database. To obtain reversibility, and further enhance the robustness when the database contains more than 10% destroyed content, in 2012, Farfoura et al. [22] proposed a blind, reversible, watermarking scheme based on a reversible data hiding technique called 'prediction-error expansion' on integers. Their scheme was designed only for numerical attributes, but it successfully detects the watermarked data with 100% accuracy, even when more than 60% of the content of the watermarked relational database has been modified. Moreover, Farfoura et al.'s scheme resists mix-match attacks. Their scheme successfully detects 100% of the hidden watermark only when less than 50% of the tuples that are selected from other database sources are mixed with the current watermarked relational database. In addition, in Farfoura et al.'s scheme, only a fractional portion of the numerical attributes are used for the embedding watermark. Therefore, if the numerical attributes do not contain the fractional portion, the watermark scheme cannot be applied.

To overcome the above-mentioned issues (that it is non-blind and only designed for numerical attributes in the relational database), and to further improve the robustness

(i.e., to resist removing or degrading the hidden watermark), in this paper, a LRW-CRDB watermarking scheme for categorical relational databases is proposed, instead of using numerical attributes. Experimental results confirmed that our proposed LRW-RDB scheme obtained a stronger robustness than previous schemes, even when more than 95% of the watermarked relational database has been deleted.

The remainder of this paper is organized as follows. Section 2 defines five basic requirements of a relational database-based watermarking scheme. Next, the details of the proposed LRW-CRDB watermarking scheme are shown in Section 3. A robustness analysis and the details of our experiments using the proposed scheme are presented in Section 4. Finally, our conclusions and suggested future research are presented in Section 5.

## 2. Five Basic Requirements for Relational Databased-Based Watermarking Scheme

When a watermarking scheme is defined for a relational database, there are five basic requirements that must be fulfilled [14]. To give a clear description of these five requirements, a scenario is given in the following paragraphs:

Let assume Kait owns a relational database *K-RDB* that contains *n*-tuples, and an attacker, called Evil, wants to tamper with Kait's database *K-RDB*. To protect the integrity of her relational database *K-RDB*, Kait embeds a watermark into her database *K-RDB* and obtains a watermarked database, called $K\text{-}RDB_W$. When Kait wants to claim ownership of her watermarked relational database $K\text{-}RDB_W$, she must be able to extract the hidden watermark *W* from database $K\text{-}RDB_W$ for verification.

i. *Preventing illegal watermark embedding and authentication*: To embed a watermark, some parameters that are used to control the number of selected tuples and attributes for generating the watermark must be determined in advance and kept secret. In other words, only the authorized database owner, such as Kait, who knows all of the above parameters, can generate and embed watermark during the watermark embedding phase and then extract the hidden watermark and verify it during the watermark extracting and verifying phase. Therefore, a good relational database-based watermarking scheme must prevent illegal watermark embedding by malicious attackers.
ii. *Reversibility*: The original relational database, such as *K-RDB*, should be completely restored after the hidden watermark *W* has been detected or identified by its owner, Kait.
iii. *Incremental updatability*: A database may need to be updated frequently, the amount of either tuples or attributes may be updated frequently. To maintain the scalability and availability of a relational database's watermark, the selected tuples and attributes for watermark generation and embedding must be independent from the remaining tuples and attributes.
iv. *Robustness*: The designed watermark scheme must be able to resistant various attacks, e.g., deletion, alternation, mix-match, and sorting attacks.
v. *Imperceptibility*: The watermarked tuple and the watermarked attributes are chosen randomly. Therefore, a malicious attacker Evil cannot guess which tuple and attribute were used to carry the watermark during the watermark embedding phase.
vi. *Blind system*: When either the owner or a third-party wants to verify the database, such as $K\text{-}RDB_W$, they neither need knowledge of the original database nor the original watermark.

Beside the above six basic requirements defined by Agrawal and Kiernan [14], the following five attacks have been included in various existing relational database-based watermarking:

## 3. The Proposed LRW-CRDB Watermarking Scheme

After carefully exploring some of the previous watermarking schemes [17,18,22,25], we observed that, in most of the numerical watermarking schemes, the fractional values of the numerical attributes in the relational databases are modified to carry the watermark [17]. However, when the numerical attributes do not exist in the relational database, no watermarking can be applied. Some watermarking schemes have been proposed for cat-

egorical attributes [15]. Moreover, the existing schemes can identify the hidden watermark successfully when only 10% of the tuples of the watermarked relational database $R'_W$ were altered. In this section, we propose a new relational database-based watermarking scheme, called LRW-CRDB, that uses categorical attributes. First, the watermark is calculated based on the information in the relational database, while referring to two embedding keys, $K_1$ and $K_2$. Then, the generated watermark is hidden in the chosen categorical attributes, so that later they can be used to claim ownership of the relational database by authorized users. In the proposed scheme, we embed only the watermark into the categorical attributes of the relational database, instead of into numerical attributes. Following the same requirements used in previous schemes [17,18,22,25], a small change in the value of the categorical attributes is acceptable in the proposed scheme, and the slight modification will not influence the entire relational database's usability. Figure 1 shows a flowchart of our proposed LRW-RDB scheme.

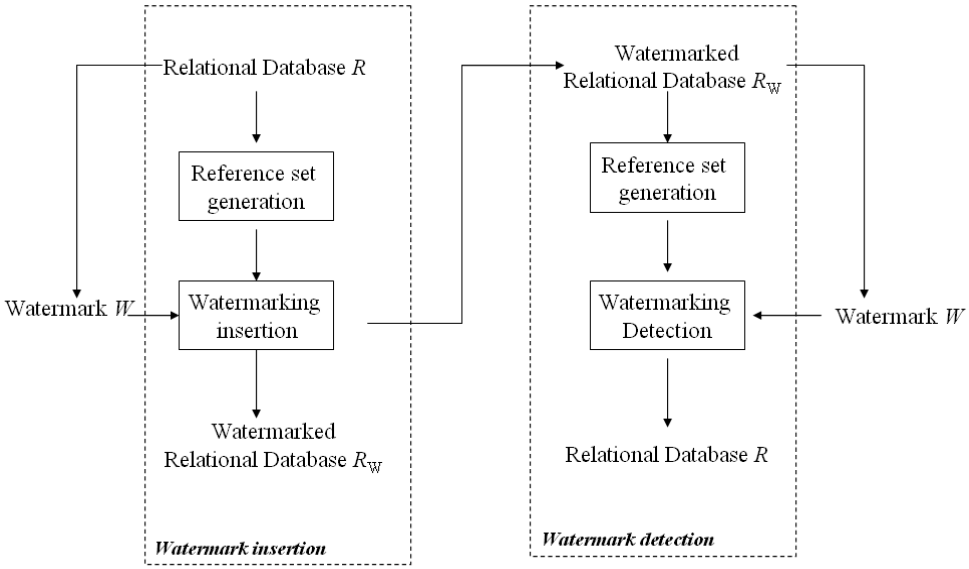


**Figure 1.** Flowchart of our proposed LRW-RDB scheme.

Assume that the relational database is the data set $R$ and is defined as $R(PK, A_0, \ldots, A_{\alpha-1})$. In the data set $R$, $PK$ is defined as the primary key attribute, and the other attributes, $A_0, A_1 \ldots, A_{\alpha-1}$, are candidates for watermark embedding. For simplicity, assume that all of the $\alpha$ attributes are categorical attributes and that their values are size values of fashion products, i.e., T-shirts, shoes. These size values are in the range of the set *OriginalSet* = {S, M, X, XL}. Let us set the index of each value in the *OriginalSet* to be from 0 to 3, respectively. There is a strong assumption in our proposed LRW-RDB scheme, that is that the primary key $PK$ cannot be modified by malicious attackers because the primary key contains valuable information, and once it is changed, the integrity of the relational database is compromised. This assumption is reasonable, especially for a relational database. To prevent the embedded watermark from being extracted by malicious attackers, the result of a one-way hash function, which is derived from the corresponding primary key $PK$ and the embedding secret keys $K_1$ and $K_2$, is used to determine which tuples are selected for watermark embedding. As the embedding keys $K_1$ and $K_2$ play a crucial role in both watermark insertion and extraction phases, they should be chosen from a large key space, and they are only known by the database owner; as a result, it is difficult for malicious attackers to guess these two keys when determining which tuples are to be chosen for watermark embedding. In our scheme, the secret keys $K_1$ and $K_2$ are computed by using Equations (2) and (3), respectively, and the watermark $W$ is generated by using Equation (4), to withstand guessing attacks for the hidden watermark $W$. Here, the one-way hash function that is used is defined as $h = H(M)$, where $M$ is a message, i.e., information from the relational database, and $M$ has three characteristics, i.e., (1) given $M$,

it is easy to compute $h$; (2) given $h$, it is hard to obtain $M$, such that $H(M) = h$; and (3) given $M$, it is also hard to find any input message $M'$, so that $H(M) = H(M')$. To meet the above three requirements, some hash functions [19] can be considered for the proposed scheme, i.e., MD5 [19] and SHA [19]. Equation (1) is used to compute the message authentication code (MAC) of a primary key and two embedding keys, $K_1$ and $K_2$. This MAC value is used to generate two reference sets and then is used to decide the chosen tuples in the watermark insertion phase. Table 1 presents several important parameters of our scheme.

$$F(t_i.PK) = H(K_1||K_2||t_i.PK) \tag{1}$$

where $K_1$ and $K_2$ are two embedding keys, $||$ denotes the concatenation function, and $t_i.PK$ is the primary key attribute of the tuple $t_i$ in the relational database $R$.

$$K_1 = H(DB\_name||Version||DOI||DB\ \text{inf.}||\ldots) \tag{2}$$

$$K_2 = H(K_1||H(K_1||DB\_name||Version||DOI||DB\ \text{inf.}||\ldots)) \tag{3}$$

$$W = H(K_2||H(K_1||H(DB\_name||Version||DOI||DB\ \text{inf.}||\ldots))) \tag{4}$$

where *DB_name* represents the name of a database; *Version* represents the version of the database; *DOI* represents the database owner's identity; *DB inf. re*presents the database information (i.e., the number of attributes, the number of tuples of the relational database $R$); and $H()$ is a one-way hash function. To help readers understand the proposed scheme presented in the following subsections more clearly, the related notations are given in Table 2:

**Table 1.** Five attacks included in existing relational database-based watermarking.

| Attack Types | Definitions |
|---|---|
| Alteration attack | Malicious attackers, such as Evil, who want to destroy the hidden watermark in the watermarked relational database by randomly modifying partial values of the chosen tuples, i.e., $K\text{-}RDB_W$. |
| Deletion attack | Malicious attackers, such as Evil, try to randomly delete partial chosen tuples in the watermarked relational database, i.e., $K\text{-}RDB_W$. to weaken the hidden watermark. |
| Sorting attack | Malicious attackers, such as Evil, try to re-sort the selected tuples of the watermarked relational data, to damage the hidden watermark. |
| Mix-match attack | The mix-match attack is also called an insertion attack. Malicious attackers, such as Evil, try to insert a certain number of selected tuples from other database into all of the tuples of the watermarked relational database, and Evil hopes the hidden watermark will not be detected. |
| Benign database updates | Assume that malicious attackers, such as Evil, try to copy Kait's watermarked relational database $K\text{-}RDB_W$, so that the attackers can sell this valuable database. However, the attackers do not know the hidden watermark $W$ in Kait's database. Therefore, the malicious attackers insert new tuples or new attributes into Kait's database before he/she uses it, so that the attackers can claim they also own Kait's database. Although the attackers update the relational database several times, Kait's watermark might not be deleted from $K\text{-}RDB_W$. Therefore, when Kait suspects that one database entry is illegally copied from her watermarked relational database, Kait can extract her watermark $W$ from the suspected database to prove her copyright.In other words, benign database updates can also be referred to the subset attack, as defined by Agrawal and Kiernan [2]. |

**Table 2.** Notations used in the proposed scheme.

| Parameters | Descriptions |
|---|---|
| $R$ | Database relation which is used to be watermarked |
| $R_W$ | Relational database with hidden watermark |
| $t_iA_j$ | Attribute $j$ of tuple $i$ in the relational database $R$. It is noted that $t_iA_j$ is the smallest inserting unit in our scheme. |
| $N$ | Number of tuples in the relational database $R$ |
| $K_1$ and $K_2$ | Two secret embedding keys |
| $1/g$ | Fraction of tuples selected for watermark embedding |
| $\alpha$ | Number of attributes in the relational database available for carrying a watermark |
| $L$ | The length of a given watermark $W$ |

### *3.1. Reference Set Generation*

In this section, we demonstrate how our method generates two reference sets, i.e., $RS_1$ and $RS_2$, which are based on the secret embedding keys $K_1$ and $K_2$. These two reference sets of each selected tuple are generated to refer to when a secret bit is hidden in the chosen tuple. They are used for the watermark phases of both insertion and detection. The reference set generation algorithm (Algorithm 1) is shown in the following:

**Algorithm 1: Reference set generation**

*Input:* MAC value $F(t_i.PK)$ of primary key $t_i.PK$, two embedding keys $K_1$ and $K_2$
*Output:* two reference sets, $RS_1$ and $RS_2$
Generate two random sequences $S_1$ and $S_2$;//using a pseudo-random generator with the seed value $F(t_i.PK)$*$K_1$ and $F(t_i.PK)$*$K_2$, respectively. $S_1$ and $S_2$ have four distinct elements, and their values are from 0 to 3.
*OriginalSet1* = {S, M, L, XL};
*OriginalSet2* = {s, m, l, xl};
**for** $i$ = 0 to 3 do
  $RS_1$[i] = *OriginalSet1*[$S_1$[$i$]];
  $RS_2$[i] = *OriginalSet2*[$S_2$[$i$]];
**end for**;
**end**

For each tuple $t_i$, the MAC value, $F(t_i.PK)$, of the primary key is calculated using Equation (1). Then, two random sequences for each tuple, $S_1$ and $S_2$, are generated using a pseudo-random generator with the corresponding seed values, $F(t_i.PK)$*$K_1$ and $F(t_i.PK)$*$K_2$, respectively. Each sequence contains four distinct values in the range (0, 3). For a better explanation, we can demonstrate the above algorithm by using the example given in the second paragraph in Section 3. Following our example, all of the $\alpha$ attributes are categorical attributes, and their values are the size values of fashion products, i.e., T-shirts, shoes. Their size values are in the range of the set *OriginalSet* = {S, M, X, XL}. Define the sets *OriginalSet1* = {S, M, X, XL} and *OriginalSet2* = {s, m, x, xl}. Assume that, with the two seeds $F(t_i.PK)$*$K_1$ and $F(t_i.PK)$*$K_2$, two random sequences, $S_1$ = {1, 0, 3, 2} and $S_2$ = {3, 2, 0, 1}, are generated using a pseudo-random generator. For each value in the random sequence $S_1$, the corresponding value of the reference set $RS_1$ is set as $RS_1$[i] = *OriginalSet1*[$S_1$[$i$]]. As a result, the reference set $RS_1$ is generated as $RS_1$ = {M, S, XL, X}. Following the same procedure, the reference set $RS_2$ is constructed as $RS_2$ = {xl, x, s, m}. In this algorithm, reference sets $RS_1$ and $RS_2$ are generated in uppercase and in lowercase, respectively, to be used for embedding the secret bit "0" or "1". To choose which reference set will be used to represent the secret bit 0 or 1 for the corresponding tuple, the choice is based on the MAC value that is computed in Equation (1).

### *3.2. Watermark Insertion*

In this section, the details of the watermark insertion process of our proposed LRW-RDB scheme are given. Our proposed LRW-RDBhiding strategy, implemented in the watermark insertion algorithm, is based on the symmetry concept, which means the pattern of the hidden watermark should be random and the distribution of the hidden watermark remains uniform. The watermark insertion algorithm describes the process of embedding one watermark bit into the categorical attributes. The two reference sets $RS_1$ and $RS_2$ for each tuple, based on the MAC value $F(t_i.PK)$ of the primary key and two predetermined secret embedding keys $K_1$ and $K_2$, are required for the watermark insertion. A *ReferenceSetGeneration*() function, which is presented in Section 3.1, is used to generate the two reference sets $RS_1$ and $RS_2$. Therefore, we skip the related description here. To embed a watermark into the selected attribute of the selected tuple in the relational database $R$, two algorithms are designed in this section. The watermark insertion algorithm describes how to determine which tuples and which attributes will be used for carrying watermark bits. The bit-encoding algorithm describes our embedding rule for hiding the watermark

bit in the selected attribute. The watermark insertion algorithm (Algorithm 2) is first given as follows:

**Algorithm 2: Watermark embedding**

*Input:* Relational database $R$, parameter $g$
*Output:* Watermarked relational database $R_W$, embedding keys $K_1$ and $K_2$
Generate the embedding keys $K_1$ and $K_2$ // using Equations (2) and (3)
Generate watermark $W$; // using Equation (4)
**for each** tuple $t_i \in R$ **do**
  **if** $F(t_i.PK)$ mod $g$ is equal to 0, **then** // the current tuple is selected
    attribute_index $j = F(t_i.PK)$ mod $\alpha$ // the current attribute $A_j$ is selected
      mark_bit_idx = $F(t_i.PK)$ mod $L$
      $b$ = $W$[mark_bit_idx];
      *bit_encoding*($F(t_i.PK)$, $K_1$, $K_2$, $b$, $t_iA_j$);
  **end if**;
**end for each tuple**;
**end**

Based on the watermark embedding algorithm mentioned above, it is obvious that each tuple must conduct ($F(t_i.PK)$ modulo $g$) to derive its MAC value first, and only the tuple whose MAC value equals 0 is selected for the concealed watermark. Thus, in $g$ continuous tuples, only one tuple is selected. For example, if $g$ = 6, then in 6 tuples, there is only one selected tuple used to embed the watermark bit. It is noted, that among $\alpha$ attributes of the selected tuples, only one attribute will be used to conceal the watermark bits. To modify the attribute values of the selected tuple, a bit-encoding algorithm (Algorithm 3) is presented below:

**Algorithm 3: Bit-encoding**

*Input:* $F(t_i.PK)$, embedding keys, $K_1$ and $K_2$, watermark bit $b$, the value of attribute $j$ of tuple $i$: $t_iA_j$
*Output:* the update value of attribute $j$ of tuple $i$: $t_iA_j$
[$RS_1$ $RS_2$] = ReferenceSetGeneration($F(t_i.PK)$, $K_1$, $K_2$) // determine two reference sets, $RS_1$ and $RS_2$
Get index $idx$ of attribute value $t_iA_j$ // index of attribute value in *OriginalSet* set
**if** $F(t_i.PK)$%2 = 0, **then**
  **if** $b$ = 0, **then** $t_iA'_j = RS_1$[idx];
  **else** $t_iA'_j = RS_2$[idx];
  **end if**;
**else**
  **if** $b$ = 0, **then** $t_iA'_j = RS_2$[idx];
  **else** $t_iA'_j = RS_1$[idx];
  **end if**;
**end if**;
*Update_Attr*(); // Update attribute value $t_iA_j$ by value $t_iA'_j$

In our proposed bit-encoding algorithm, the original attribute value $t_iA_j$ is used to extract its index $idx$ from the set *OriginalSet*. Then, the MAC value $F(t_i.PK)$ derived by Equation (1) is used to determine which reference set is assigned to conceal the watermark, bit 0 or bit 1. The watermark attribute value $t_iA'_j$ can be extracted from reference sets $RS_1$ or $RS_2$, such as $RS_1[idx]$ or $RS_2[idx]$, respectively, and based on the watermark bit $b$ and the index $idx$, for the selected attribute value $j$ of the selected tuple $i$ in the relational database $R$. After completing the process of bit-encoding, the *Update_Attr*() function is used to update the attribute value $j$ of the tuple $i$ in the relational database $R$ for the given watermark attribute value $t_iA'_j$.

To clarify the process of the bit-encoding algorithm, an example is presented in this paragraph. Let us assume the original attribute value of the selected tuple is $t_iA_j$ = "X", and the watermark bit $b$ = 0, two reference sets are constructed as $RS_1$ = {M, S, XL, X} and $RS_2$ = {xl, x, s, m}, respectively. Here, we assume that the corresponding value of

$F(t_i.PK)\%2 = 0$, which means that $RS_1$ is assigned to carry the watermark bit 0 and $RS_2$ is assigned to carry watermark bit 1, respectively. Based on *OriginalSet* = {S, M, X, XL}, the index of the current attribute value "X" is extracted as $idx = 2$. Then, to embed the watermark bit $b = 0$, the reference set $RS_1$ is utilized. Following the proposed bit-encoding rule, the corresponding value of index, $idx = 2$, is extracted from the reference set $RS_1$ as "XL". Finally, the watermark attribute value is set as "XL".

### 3.3. Watermark Detection

Assume that Kait suspects that a published relational database has been illegally copied from her watermarked relational database $R_{W.}$ Here, we assume that the attacker does not drop the primary key attributes and does not modify the values of the primary key. This assumption is made because the primary key values contain valuable information, and once the primary key values in a database have been modified, the integrity of the database is compromised.

To extract and verify the hidden watermark, the parameters $g$ and $\alpha$ are required in addition to two secret embedding keys, $K_1$ and $K_2$; as they use the same one-way hash function, with the same parameters $g$ and $\alpha$ and the embedding secret keys, $K_1$ and $K_2$, and the same tuples and attributes are selected as in the watermark insertion algorithm. The proposed watermark detection procedure contains two algorithms: one is the watermark detection algorithm (Algorithm 4), and the other is the bit-decoding algorithm (Algorithm 5). The prior algorithm not only performs the inverse operations of the watermark insertion algorithm introduced in Section 3.2, but also conducts a majority voting strategy to determine the final watermark, because the watermark $W$ is hidden in the relational database $R$ several times. The latter algorithm performs the inverse operations of the bit encoding algorithm mentioned in Section 3.2. After the watermark detection is finished, the relational database owner can make use of the extracted watermark to prove copyright ownership. The watermark detection algorithm shows details of the watermark detection process for the watermark relational database $R_W$, as follows:

**Algorithm 4: Watermark detection**

*Input:* Relational database $R_W$, parameters $g$, and $\alpha$
*Output:* Watermarked status $\in$ {true, false}, and recover relational database $R$Generate embedding keys, $K_1$ and $K_2$//using Equations (2) and (3)
Generate watermark $W$;//using Equation (4)
**for** $i = 0$ to $L - 1$ **do**
count[$i$][0] = 0; count[$i$][1] = 0; //reset the votes of the watermark
**end for**;
for each tuple $t_i \in$ R do
    **if** $F(t_i.PK)$ mod $g$ equal 0, then//select this tuple
    attribute_index $j = F(t_i.PK)$ mod $\alpha$//select this attribute $A_j$
    mark_bit_idx = $F(t_i.PK)$ mod $L$
    $b = bit\_decoding(F(t_i.PK), K_1, K_2, t_iA_j)$;
    count[mark_bit_idx][$b$] = count[mark_bit_idx][$b$] + 1;
    **end if**;
**end for each tuple**;
**for** $i = 0$ to $L - 1$ **do**
    **if** count[i][0] + count[i][1] = 0 **then** $W'$[i] = $-1$;
    **end if**;
    **if** count[i][1] > count[i][1] **then** $W'$[i] = 1;
    **else** $W'$[i] = 0;
    **end if**;
**end for**;
**for** $i = 0$ to $L - 1$ do//find the match result between the original and the detected watermarks
**if** $W'$ [i] = $W$[i] **then** matchamount + = 1;
**end if**;
**end for**;

**if** matchamount = $L$, **then**
    **return** true;//relational database $R$ is restored successfully
**else**
    **return** false;//relational database $R$ can not be restored
**end if**;

The details of the bit-decoding algorithm is depicted as follows:

**Algorithm 5: Bit-decoding**

*Input:* $F(t_i.PK)$, embedding keys, $K_1$ and $K_2$, the value of the watermarked attribute $j$ of tuple $i$: $t_iA_j$
*Output:* watermark bit b and the update value of attribute $j$ of tuple $i$: $t_iA_j$
[$RS_1$ $RS_2$] = ReferenceSetGeneration($F(t_i.PK)$, $K_1$, $K_2$)//determine two reference sets, $RS_1$ and $RS_2$
**if** $F(t_i.PK)\%2 = 0$, **then**
    **if** $t_iA'_j \in RS_1$, **then**
        $b = 0$;
        Get index $idx$ of attribute value $t_iA_j$ from the set $RS_1$;
    **else**
        $b = 1$;
        Get index $idx$ of attribute value $t_iA_j$ from the set $RS_1$;
    **end if**;
**else**
    **if** $t_iA'_j \in RS_2$ **then**
        $b = 0$;
        Get index $idx$ of attribute value $t_iA_j$ from the set $RS_1$;
    **else**
        $b = 1$;
        Get index $idx$ of attribute value $t_iA_j$ from the set $RS_1$;
    **end if**;
    recon_value = *OriginalSet*[idx];
**end if**;
*Update_Attr*(); //Update attribute value $t_iA_j$ by value recon_value

From the above steps, we can see that the index value $idx$ shall be determined to reconstruct the original categorical attribute value. If $t_iA'_j \in RS_1$, then the index $idx$ of $t_iA'_j$ is extracted from the set $RS_1$, and the watermark bit is recovered as an assigned bit of the set $RS_1$. Otherwise, the index $idx$ of $t_iA'_j$ is extracted from the set $RS_2$, and the watermark bit is recovered as the assigned bit of the set $RS_2$. Finally, the original attribute value $t_iA_j$ is reconstructed from the set *OriginalSet*, such as *OriginalSet* [$idx$].

## 4. Experimental Results

In this section, we present two experiments to confirm our performance robustness and one comparison to prove the usability of our proposed LRW-RCDB scheme. First, we discuss the robustness analysis of our watermarking scheme with different parameters. Second, we compare the proposed scheme with Shehab et al.'s scheme [17], Al-Haj and Odeh's scheme [18], and Farfoura et al.'s scheme [22]. All of the experiments were performed on a PC with an Intel(R) Core™ i7-3770 CPU and 8-GB RAM. Windows 7 Professional 64-bit was the operating system used in our experiments. In all experiments, all algorithms were programmed using Microsoft Visual Studio 2005 C#, and the Microsoft SQL Server database served as the database test platform. A test relational database $R$ with nine attributes, one of which contained the primary key attribute, and where the other eight contained the categorical attributes, were generated in advance. The eight categorical attributes were considered to be candidates for embedding the watermark in the proposed schemes. In the last comparison, we compare our proposed LRW-CRDB scheme with the latest scheme proposed by Shah et al. [25], to evaluate the usability of our scheme; although their scheme belongs to semi-fragile type, instead of robustness.

### *4.1. Robustness Analysis*

In this section, an analysis of the proposed scheme's robustness against malicious attacks and benign database update operations is demonstrated.

A blind, robust watermark scheme, not only detects the embedded watermark without the knowledge of the original relational database, but also must withstand various attacks and benign database update operations. These attacks are implemented by attackers with the hope of destroying or adversely affecting the watermark that is hidden in the relational database. Such attacks can be classified into four types: deletion, alteration, mix-match, and sorting attacks. The related analyses of our proposed LRW-RDB scheme for these four types of attack are discussed in the following subsections.

In the following experiments, we assumed that attackers did not know all of the secret information, i.e., the parameters $g$ and $\alpha$ and the two embedding keys; thus, they did not know which tuples and attributes were chosen for the embedding watermark.

#### 4.1.1. Alternation Attack

In this attack, attackers attempt to randomly select and modify random attributes $t_iA_j$ in $\beta$ tuples in the watermarked relational database $R_W$. Assume that the malicious attackers do not know any secret information. Hence, they do not know which tuples and attributes were chosen for embedding the watermark.

Figure 2 presents the performance of the proposed scheme under alternation attack with various parameters of $g$, such as $g$ = 6, 12, 24, 48, and 96. As can be seen from Figure 2, the smaller the value of parameter $g$, the higher the number of tuples that were selected for the watermark embedding. When more than 95% of the tuples were altered randomly, the proposed scheme could detect the hidden watermark successfully with $g$ = 6. Thus, attackers must alter more than 95% of the watermarked relational database to have the possibility of removing the watermark completely. Obviously, with a small portion of the watermarked relational database, the watermark is successfully detected in the proposed scheme. As a result, the proposed scheme obtains a high efficiency for watermark detection for relational databases. This fact makes it possible for our proposed LRW-RDB scheme to build an efficient tool for searching and detecting illegal copies of the database on the web.

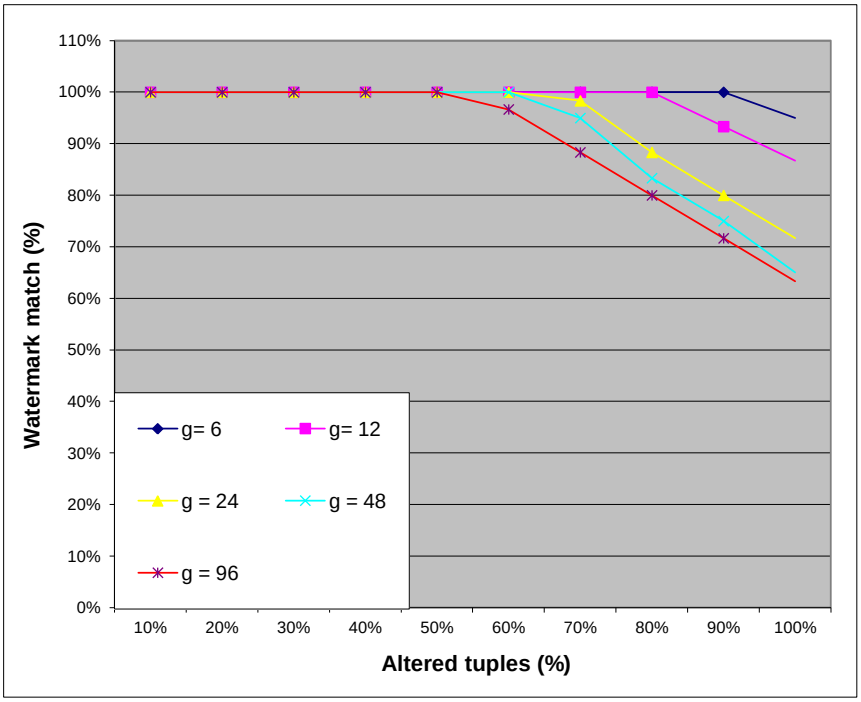


**Figure 2.** Resilience to alteration attacks for various values of the $g$ parameter.

In our proposed LRW-RDB scheme, the watermark $W$ is hidden in the relational database several times. Then, with the majority MVT, for each watermark bit, the numbers of its values that are ones or zeroes will be counted, respectively. Therefore, the watermark detection of our proposed LRW-RDB scheme is unable to reconstruct the true watermark bit $w_i$ only if the number of times of the extracted true watermark bit is smaller than $N_w/2$ times that of the embedded watermark bit $w_i$, where $N_w$ is the number of times that the watermark bit $w_i$ was embedded into the chosen tuples.

#### 4.1.2. Deletion Attack

In a deletion attack, attackers randomly drop $\beta$ tuples from the watermarked relational database. To simulate this attack, we randomly deleted various ratios of the watermarked relational database with different parameters of $g$ ($g$ = 6, 12, 24, 48, and 96). Figure 3 shows that our proposed LRW-RDB scheme ($g$ = 6) can obtain 100% of the watermark. This means the hidden watermark can be extracted successfully with our proposed LRW-RDB scheme, even when more than 95% of the tuples have been deleted from the watermarked relational database. In addition, we can see that our proposed LRW-RDB scheme provides a higher resilience when the parameter $g$ decreases. Due to using a smaller value for the parameter $g$, more tuples are selected for the embedding watermark. In other words, the more bits that can be hidden, the more time the watermark is hidden. This relationship makes the MVT technique more effective for determining the correctly extracted bit.

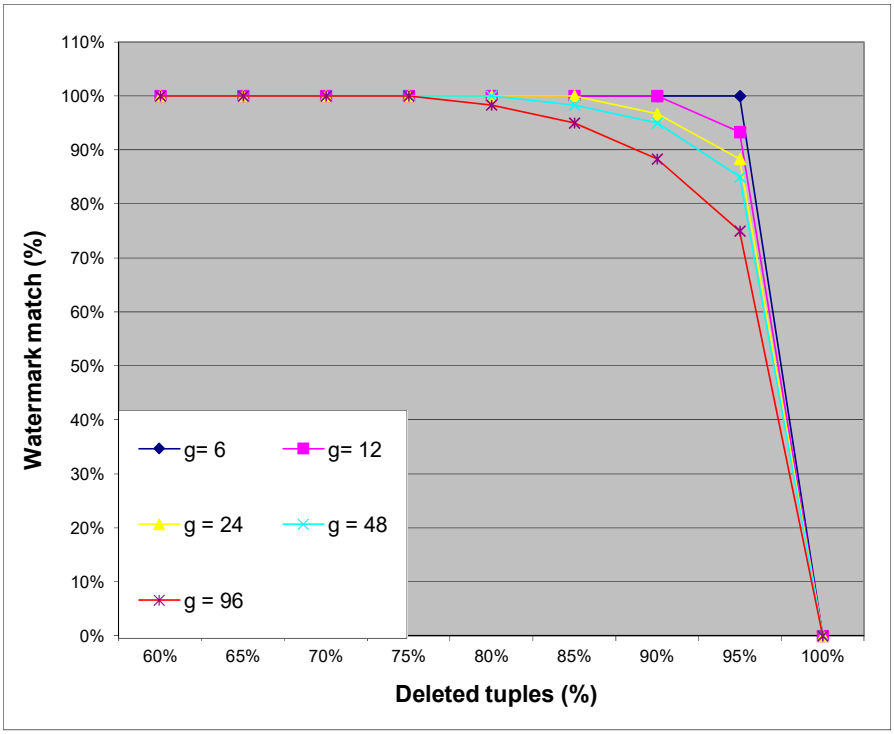


**Figure 3.** Resilience to deletion attacks for various values of $g$ parameter.

#### 4.1.3. Mix-Match Attack

In this attack, which is also known as an insertion attack, attackers attempt to weaken the embedded watermark by inserting $\beta$ tuples from other data sources into the watermarked relational database $R_W$. However, these attacks have little effect on our proposed LRW-RDB scheme. Figure 4 shows the good results of our proposed LRW-RDB scheme against a mix-match attack. This attack is implemented by randomly selecting difference ratios of the other data sources and mixing them with those of the watermarked relational database $R_W$. As shown in Figure 4, when as much as 80% of the tuples are newly inserted into the watermarked relational database, the watermark can be successfully detected in

our proposed LRW-RDB scheme (for $g = 6$). As a result, our proposed LRW-RDB scheme is resilient to this type of attack. In this attack, new tuples are inserted with the purpose of changing a watermark bit from 0 to 1, and vice versa. However, with the MVT technique, our proposed LRW-RDB scheme can guarantee the high accuracy ratio of each extracted bit. For example, when a watermark bit is 1, with the aim of converting the value of the embedded watermark bit from 1 to 0, the attackers must embed 50% or more of the embedded bits to 0, to reduce the effect of the MVT technique. In other words, the attackers must insert at least the duplicate of the current watermark relational database to succeed in this attack. However, in this scenario, the usability of the watermark relational database is lost.

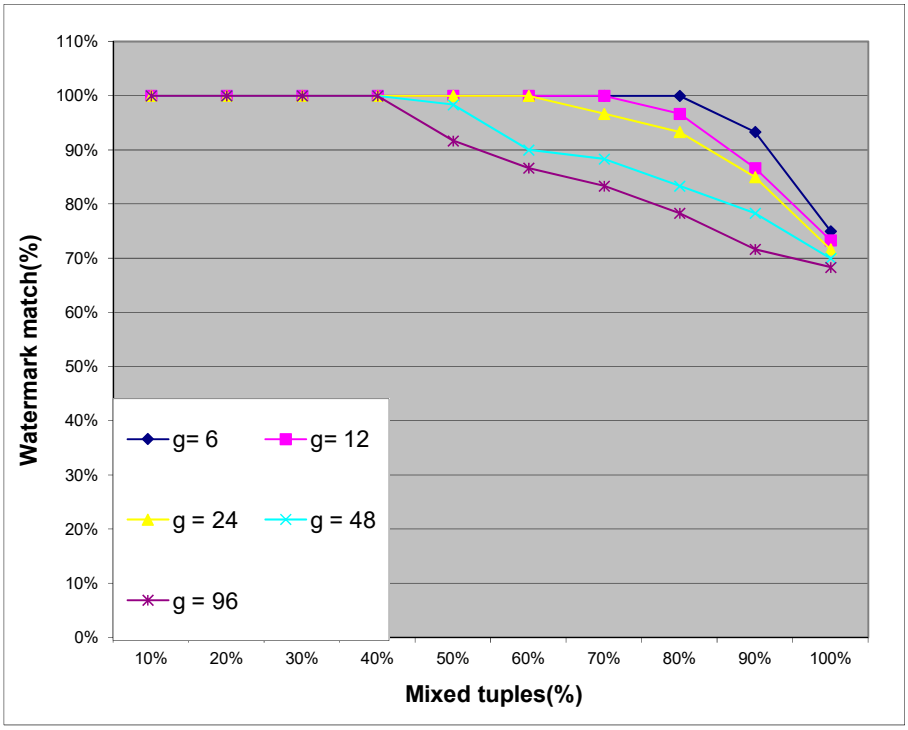


**Figure 4.** Resilience to mix-match attacks for various values of $g$ parameter.

#### 4.1.4. Sorting Attack

In our proposed LRW-RDB scheme, each tuple and its attribute are chosen independently for carrying the watermark. The pseudo hash value of the primary key of each tuple and the two embedding secret keys $K_1$ and $K_2$ are used to determine whether these tuples and attributes are chosen for watermark embedding. Additionally, the hash value is also used to select the index of the watermark bit in both of watermarking insertion and detection phases. In other words, our proposed LRW-RDB scheme is not affected by sorting attacks.

#### 4.1.5. Combination Attack

To further prove the robustness of our proposed LRW-RDB scheme against combination attacks, such as a combination of mix-match attack and sorting attack, some discussions are also given in paragraph. It is noted that each tuple and attribute is selected independently for inserting the watermark with our proposed LRW-CRDB scheme. In other words, the smallest inserting unit is a cell, which is defined as "Attribute $j$ of tuple $i$ in the relational database $R$" in Table 2. As Equation (1) defines, a tuple is selected with the combinations of primary key $PK$ for a given tuple and two pre-determined secret keys. This means that, when selecting an attribute to insert a watermark bit with our proposed LRW-CRDB scheme, its neighboring tuples and its neighboring attributes are not involved.

Therefore, as long as the primary key has not be compromised in a tuple, our proposed scheme can withstand various combination attacks.

### *4.2. Performance Comparison*

To show the advantages of our method, we also compared our proposed scheme with three other schemes, against alternation, deletion, mix-match, and sorting attacks, in this subsection. For the three other schemes by Shehab et al. [17], Al-Haj and Odeh [18], and Farfoura et al. [22], the eight other attributes are numerical attributes, which are used as candidate attributes. The size $n$ of the generated relational database $R$ was 80,000 tuples, and the parameter $g$ = 6 was used in our experiments. To ensure the accuracy of the obtained results, each test was repeated 100 times. Then, for each trial, the average was calculated of all of the successful watermark matches.

#### 4.2.1. Alternation Attack

Figure 5 presents the results of resilience against alteration attack of the proposed LRW-RDB scheme, Shehab et al.'s scheme [17], Al-Haj and Odeh's scheme [18], and Farfoura et al.'s scheme [22]. As seen from Figure 5, our proposed scheme outperformed the schemes of Shehab et al., Al-Haj and Odeh., and Farfoura et al.

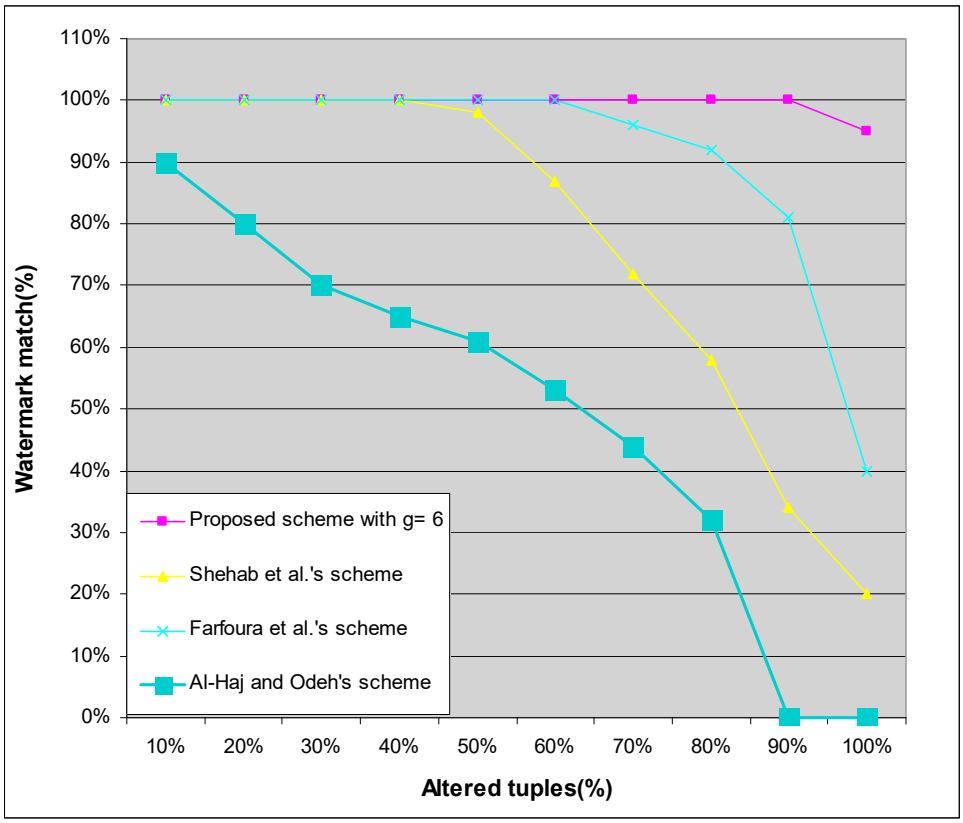


**Figure 5.** Comparison of the results of resilience to an alteration attack among the four schemes with $g$ = 6.

This finding arose because, in our proposed scheme, the MVT technique was used; thus, the watermark data were embedded in the relational database several times. When the watermark detection was processed completely, the MTV technique was used to determine the best watermark data.

4.2.2. Deletion Attack

Figure 6 gives the results of resilience to a deletion attack using the proposed LRW-RDB scheme, Shehab et al.'s scheme [17], Al-Haj and Odeh's scheme [18], and Farfoura et al.'s scheme [22]. In our proposed scheme with g = 6, the embedded watermark can be 100% detected after deleting more than 95% of the watermarked relational database, which is superior to the results of Shehab et al.'s scheme, Al-Haj and Odeh's scheme, and Farfoura et al.'s scheme. However, in Farfoura et al.'s scheme, the embedded watermark can be detected 100% successfully only when less than 70% of the watermarked relational database is deleted.

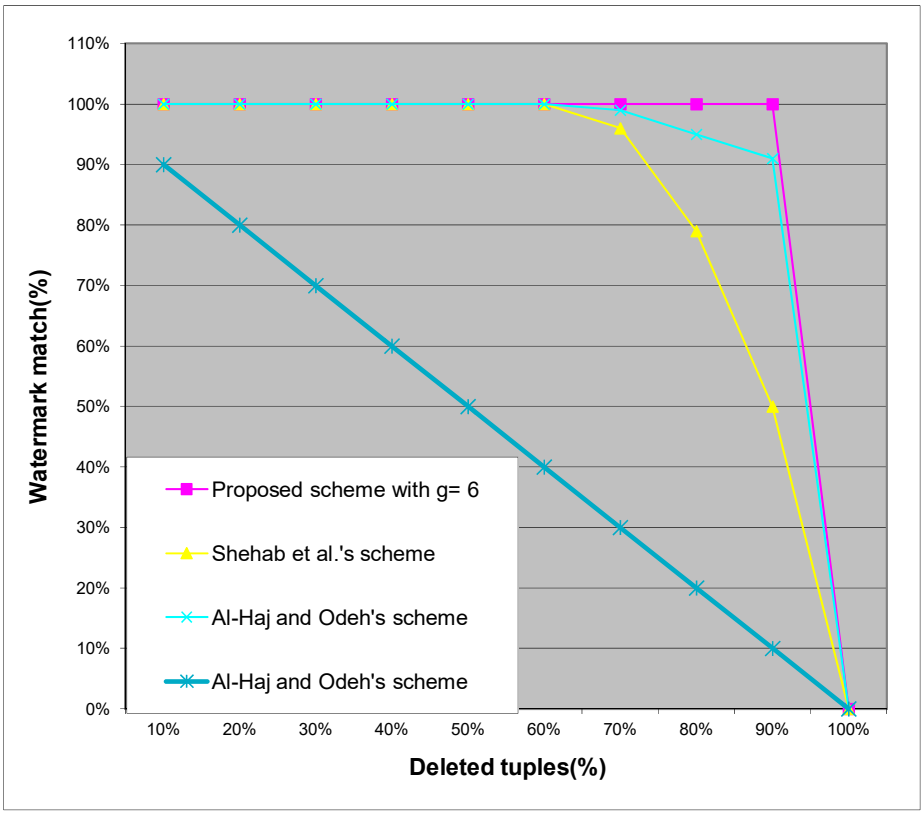


**Figure 6.** Comparison of the results of resilience to a deletion attack among the four schemes, with $g = 6$.

We can easily see that, when the value of parameter $g$ equals 6 and the MVT technique is used in our proposed scheme, greater robustness against deletion attacks is obtained. When smaller values of the parameter $g$ and MVT were used, more bits could be hidden. In other words, the watermark is carried for a longer time. Such an arrangement makes our proposed scheme more effective for determining the correctly extracted bit with the MVT technique.

4.2.3. Mix-Match Attack

To simulate a mix-match attack, we inserted various ratios of the relational database from other sources into the watermarked relational database $R_W$. In this attack, the relational database from other sources was generated artificially with the same nine attributes as the watermarked relational database $R_W$. Figure 7 shows the resilience to a mix-match attack of our proposed scheme compared with the schemes of Shehab et al. [17], Al-Haj and Odeh [18], and Farfoura et al. [22]. It is obvious that, for a mix-match attack, our proposed scheme is more robust than Shehab et al.'s scheme [17], Al-Haj and Odeh's scheme [18], and Farfoura et al.'s scheme [22].

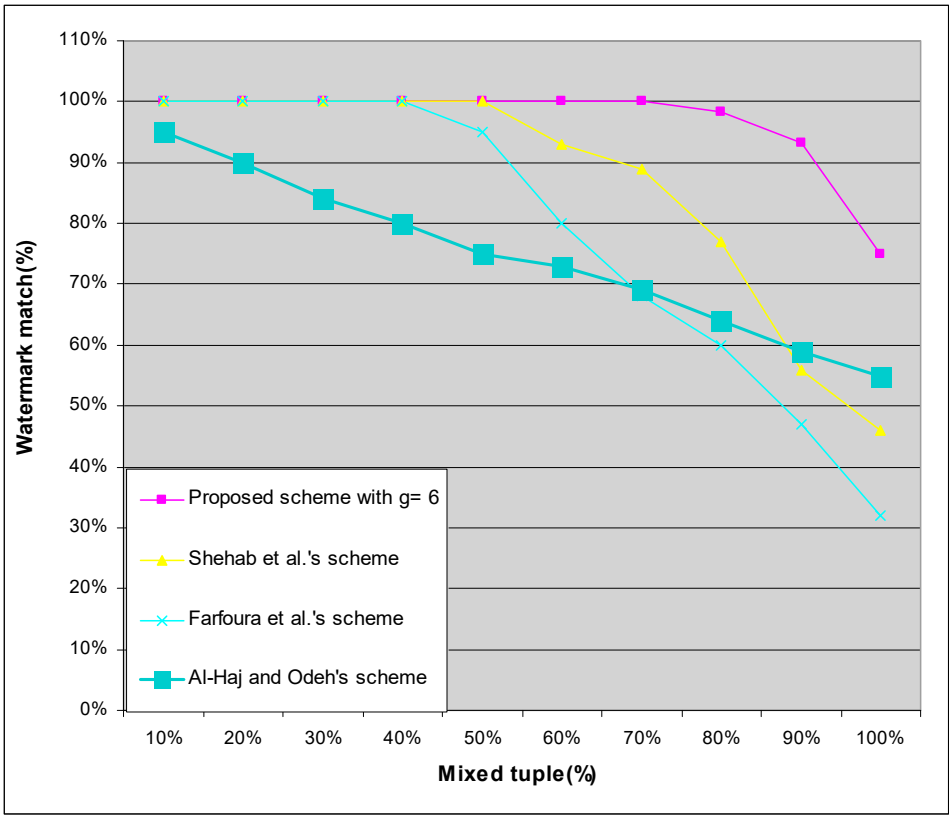


**Figure 7.** Comparison of the results of resilience to a mix-match attack among the four schemes, with $g = 6$.

#### 4.2.4. Sorting Attack

In a sorting attack, malicious attackers try to randomly sort the tuples in the database. The sorted tuples can perturb the watermark extraction process, but in our scheme and the existing schemes [17,18,22], the results show the same amount of resilience to such an attack. This finding arose because the pseudo hash value of the primary key of each tuple was used to determine whether the tuples and attributes were selected for watermark embedding in all schemes.

### *4.3. Usability Analysis*

In the relational database scenario, the data owner always wants to generate a watermarked database which has a strong ownership, so that his/her ownership can be maintained. In other words, data owners may want to have larger bandwidths for manipulations during watermark embedding. By contrast, the data recipient may want minimum distortions caused during watermark embedding, so that the usability of the watermarked database is ensured. It is concluded, therefore, that the data owner and data recipient have a conflict of requirements, and the usability of a watermarked database can be maintained, only if a balance between above two requirement is achieved. Among fragile, semi-fragile, and robust watermarking schemes, the latter type a has larger bandwidth for manipulations during watermark embedding than the former two types; because more watermark embedding makes it more difficult for the hidden watermark to be removed.

To further prove that the usability of our proposed LRW-CRDB scheme is acceptable, Shah et al.'s scheme [25] was chosen because their scheme was published recently and represents a good example of finding a balance between the data owner's and recipient's requirements. In Table 3, we compare our proposed LRW-CRDB scheme and Shah et al.'s semi-fragile watermarking scheme for relational databases.

**Table 3.** Comparison between our proposed LRW-RDB scheme and Shah et al.'s semi-fragile watermarking scheme.

| Criteria | LRW-RDB Scheme | Shah et al.'s Semi-Fragile Watermarking Scheme [25] |
|---|---|---|
| Database Type | Relational database | Relational database |
| Embedding strategy | Switching between two difference reference set + majority voting | Changing text case |
| Attack types | Alternation, Deletion, Mix-match (insertion), Sorting | Insertion with error in detection, Detection with error detection |
| Watermarking type | Reversibility | Semi-fragile |
| Blind system | Yes | Yes |
| Benign database updates | Yes | No |
| Imperceptibility | Yes | Yes |
| Incremental updatability | Yes | Yes |
| Reversibility | Yes | No |
| Prevent illegal watermark embedding and authentication | Yes | Yes |
| % of watermark match when 50% tuples' deletion | 100% | 60% when group size is 25, 100% when group size is 75 |
| % of watermark match when 90% tuples' deletion | 100% | 5% when group size is 25,50% when group size is 75,100% when group size is 150 |
| Usability | + | − |

From Table 3, it is noted that the purpose of Shah et al.'s semi-fragile watermarking scheme is to ensure that the hidden watermark can withstand legal updates, such as legal insertion/deletion of tuples or the update of attribute values. Their expected performance depends on the number of tuple groups. Only when the number of the group size is up to 75, after 50% of tuples are deleted, can the detection rate of legally deleted hidden watermarks remain 100%. When 90% of the tuples are removed, the number of tuple groups must be increased to 150 to maintain a 100% detection rate. In other words, to maintain its detection performance, the number of tuple groups plays a crucial role. Since their embedding strategy is to change the text case of the selected attribute values, the distortion caused by watermark embedding is very small. However, the scheme of Shah et al. needs to group the tuples in the database in advance. A larger group size helps to increase the detection rate, but the cost is that the usability of the watermark database will therefore be reduced. In contrast, our proposed LRW-CRDB scheme embeds watermarks according to two pre-defined reference sets and does not depend on the number of tuple groups in the database. In addition, the semantics of the embedded watermark data generated by our proposed scheme are unchanged. The watermark data generated by our LRW-CRDB scheme can also meet the requirements of data owners and data recipients regarding the degree of data modification. In general, our proposed LRW-CRDB scheme has a high availability for the watermark database generated, compared with Shah et al.'s scheme [25].

## 5. Conclusions

In this paper, we proposed a LRW-CRDB scheme for categorical relational databases, which is used to prove the ownership of the relational database. The authorized owner can obtain the fully reconstructed relational database after the watermark has been extracted and verified with our proposed scheme. Sufficient experiments demonstrated that our proposed scheme can withstand a variety of attacks. Moreover, comparisons among our proposed scheme and three existing schemes showed that the performance of the proposed scheme in resisting various attacks is superior to those of the three previous schemes.

Based on the experimental results and our robustness analysis, we can conclude that the proposed LRW-CRDB scheme provides more secure and robust protection of the hidden watermark and the content of the relational database than the three previous schemes. This is because when selecting an attribute to insert a watermark bit with our proposed LRW-CRDB scheme, only the corresponding primary key and two secret keys are involved, the neighboring tuples and the neighboring attributes are not involved. In other words, as long as the primary key in a relational database has not be compromised, the robustness of our proposed LRW-RDB scheme is guaranteed.

Certainly, a scheme such as ours, based on a strong assumption and primary key, does not exist in the NoSQL database. In the future, we will try to design a new scheme that does not depend on primary key, so that our research results can be extended to include the NoSQL database.

**Author Contributions:** Conceptualization, C.-C.L. and T.-S.N.; methodology, C.-C.L. and T.-S.N.; software, T.-S.N.; validation, C.-C.L. and C.-C.C.; writing—original draft preparation, T.-S.N.; writing—review and editing, C.-C.L. and C.-C.C.; supervision, C.-C.L. and C.-C.C.; funding acquisition, C.-C.L. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by Ministry of Science and Technology, grant number 110-2410-H-167-004.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Data available on request.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Swanson, M.; Kobayashi, M.; Tewfik, A. Multimedia data-embedding and watermarking technologies. *Proc. IEEE* **1998**, *86*, 1064–1087. [CrossRef]
2. Chang, C.-C.; Nguyen, T.S.; Lin, C.-C. A reversible data hiding scheme for VQ indices using locally adaptive coding. *J. Vis. Commun. Image Represent.* **2011**, *22*, 664–672. [CrossRef]
3. Shiu, P.-F.; Tai, W.-L.; Jan, J.-K.; Chang, C.-C.; Lin, C.-C. An Interpolative AMBTC-based high-payload RDH scheme for encrypted images. *Signal Process. Image Commun.* **2019**, *74*, 64–77. [CrossRef]
4. Xie, X.-Z.; Lin, C.-C.; Chang, C.-C. Data Hiding Based on a Two-Layer Turtle Shell Matrix. *Symmetry* **2018**, *10*, 47. [CrossRef]
5. Su, G.-D.; Chang, C.-C.; Lin, C.-C. A High Capacity Reversible Data Hiding in Encrypted AMBTC-Compressed Images. *IEEE Access* **2020**, *8*, 26984–27000. [CrossRef]
6. Chen, M.; He, Y.; Lagendijk, R. A fragile watermark error detection scheme for wireless video communications. *IEEE Trans. Multimed.* **2005**, *7*, 201–211. [CrossRef]
7. Lin, C.-C.; He, S.-L.; Chang, C.-C. Pixel-based fragile image watermarking based on absolute moment block truncation coding. *Multimed. Tools Appl.* **2021**, *80*, 29497–29518. [CrossRef]
8. Sergio, B.S.; Gan, L.; Nandi, A.K.; Aburdene, M.F. Secure private fragile watermarking scheme with improved tampering localization accuracy. *IET Inf. Secur.* **2010**, *4*, 137–148.
9. Dharwadkar, N.V.; Amberker, B.B. An Efficient Non-blind Watermarking Scheme for Color Images using Discrete Wavelet Transformation. *Int. J. Comput. Appl.* **2010**, *2*, 60–66. [CrossRef]
10. Bhatnagar, G.; Wu, Q.J.; Raman, B. Robust gray-scale logo watermarking in wavelet domain. *Comput. Electr. Eng.* **2012**, *38*, 1164–1176. [CrossRef]
11. Song, C.; Sudirman, S.; Merabti, M. A robust region-adaptive dual image watermarking technique. *J. Vis. Commun. Image Represent.* **2012**, *23*, 549–568. [CrossRef]
12. Preda, R.O. Semi-fragile watermarking for image authentication with sensitive tamper location in the wavelet domain. *Measurement* **2013**, *46*, 367–373. [CrossRef]
13. Chang, C.-C.; Lin, C.-C.; Su, G.-D. An effective image self-recovery based fragile watermarking using self-adaptive weight-based compressed AMBTC. *Multimed. Tools Appl.* **2020**, *79*, 24795–24824. [CrossRef]
14. Agrawal, R.; Kiernan, J. Watermarking Relational Databases. In Proceedings of the 28th International Conference on Very Large Databases, Hong Kong, China, 20–23 August 2002; pp. 155–166. [CrossRef]
15. Sion, R. Proving ownership over categorical data. In Proceedings of the 20th IEEE International Conference on Data Engineering ICDE, Boston, MA, USA, 2 April 2004; pp. 584–596.

16. Li, Y.; Swarup, V.; Jajodia, S. Fingerprinting relational databases: Schemes and specialties. *IEEE Trans. Dependable Secur. Comput.* **2005**, *2*, 34–45. [CrossRef]
17. Shehab, M.; Bertino, E.; Ghafoor, A. Watermarking Relational Databases Using Optimization-Based Techniques. *IEEE Trans. Knowl. Data Eng.* **2007**, *20*, 116–129. [CrossRef]
18. Al-Haj, A.; Odeh, A. Robust and Blind Watermarking of Relational Database Systems. *J. Comput. Sci.* **2008**, *4*, 1024–1029. [CrossRef]
19. Bhattacharya, S.; Cortesi, A. A Generic Distortion Free Watermarking Technique for Relational Databases. In *International Conference on Information Systems Security*; Springer: Berlin, Germany, 2009; pp. 252–264. [CrossRef]
20. Prasannakumari, V. A robust tamperproof watermarking for data integrity in relational databases. *Res. J. Inf. Technol.* **2009**, *1*, 115–121.
21. Hamadou, A.; Sun, X.; Gao, L.; Shah, S.S.A. A Fragile Zero-Watermarking Technique for Authentication of Relational Databases. *Int. J. Digit. Content Technol. Its Appl.* **2011**, *5*, 189–200. [CrossRef]
22. Farfoura, M.E.; Horng, S.-J.; Lai, J.-L.; Run, R.-S.; Chen, R.-J.; Khan, M.K. A blind reversible method for watermarking relational databases based on a time-stamping protocol. *Expert Syst. Appl.* **2012**, *39*, 3185–3196. [CrossRef]
23. Rao, U.P.; Patel, D.R.; Vikani, P.M. Relational Database Watermarking for Ownership Protection. *Procedia Technol.* **2012**, *6*, 988–995. [CrossRef]
24. Khan, A.; Husain, S.A. A Fragile Zero Watermarking Scheme to Detect and Characterize Malicious Modifications in Database Relations. *Sci. World J.* **2013**, *2013*, 796726. [CrossRef]
25. Shah, S.A.; Khan, I.A.; Kazmi, S.Z.H.; Nasaruddin, F.H.B.M. Semi-fragile watermarking scheme for relational database tamper detection. *Malays. J. Comput. Sci.* **2021**, *34*, 1–12.