A Hybrid Watermarking Scheme for Relational Databases Copyright Protection and Tamper Proofing

Ali Hamadou, Xingming Sun, Saeed Arif Shah, Lingyun Gao

Abstract

To date, database watermarking schemes are classified to be either robust or fragile depending on whether they are designed for copyright protection or data integrity checking. In this paper, we propose a novel approach that combines robust and fragile watermarking for ownership proofing and tamper detection and localization of relational data. The scheme first securely embeds a multi-bits robust watermark into the database relation. For fragile watermark embedding, the robustly marked relation is partitioned into groups. Then, a watermark is computed and inserted in each group independently in a way that preserves the robustness and the reliability of the robust component. All watermarks are securely detected without requiring access to the original database relation. Through rigorous theoretical analysis, we demonstrate that our proposed hybrid approach is secure against common database attacks. Furthermore, our experiments conducted on real life data showed that the proposed method can be effectively used in real world applications.

Keywords: Hybrid Watermarking, Copyright Protection, Tamper Proofing, Relational Databases

1. Introduction

Nowadays, with the rapid development of information technology and the widespread use of peer-to-peer file sharing applications, the security of digital content has generated significant research and commercial interest. Digital watermarking provides a promising method of protecting digital data such as image, audio, or video, and recently relational databases, from illegal copying and malicious tampering. The main identified application scenarios in the security of relational databases watermarking are: a) copyright protection, i.e. protecting ownership and usage rights, and b) tamper proofing and authentication, which aim at detecting and localizing malicious modifications. Therefore, database watermarking schemes can be classified to be either robust or fragile according to their applications. Robust schemes are usually used for copyright protection, ownership proof, or traitor tracing, while fragile schemes are designed for tamper proofing and database integrity checking. Recently, the problem of securing relational databases through digital watermarking has been extensively studied [1~16]. However, all proposed techniques are either robust [1~7] or fragile [8~15]. From the best of our knowledge, approaches that combine both copyright protection and authentication for relational databases have not been proposed yet. In this paper, we propose a hybrid watermarking technique that embeds a robust watermark and a fragile watermark into relational data, thus joining copyright protection and tamper proofing. Moreover, since the existing multiple watermarks algorithms [6] focus only on joint ownership proofing, the proposed scheme can be considered as a novel application of multiple watermarks approach.

The rest of this paper is organized as follows. In section 2 we review the background and related work. Our proposed hybrid scheme is described in section 3. The algorithms for watermark embedding and detection are explained in detail. The security analysis and experimental results of the proposed scheme are discussed in sections 4 and 5 respectively. Finally, we conclude the paper in section 6, and provide some directions for the future work.
2. Related Work

A number of robust and fragile techniques have been proposed for watermarking relational databases [16].

2.1. Robust Schemes

The first well known database watermarking scheme has been proposed by Agrawal and colleagues [1]. In this method, for embedding the watermark, some least significant bits (LSBs) of selected numeric attributes in some of the tuples are altered by the means of a secure pseudo-random sequence generator seeded with a secret key and the primary key of the tuples. In [6], the authors extended the scheme in [1] by embedding a multiple-bits watermark into numerical attributes. Li et al. [5] have also extended the scheme in [1] for fingerprinting relational data. The embedded multi-bits watermark is used to represent different buyers who purchased the database relation. Sion et al. [2] proposed a group-based scheme that embeds a multi-bits watermark in the data statistics rather than in data itself. In this scheme, one bit is securely encoded into each subset, while continuously assessing data quality. Shehab et al developed an robust scheme using optimization techniques [3]. This method which also embeds the watermark in data statistics is claimed to introduce less distortion than the above schemes. In [4], Sion et al. proposed a robust scheme for watermarking categorical attribute by changing some of its values to other values of the attributes (e.g., “red” is changed to “green”) if such change is tolerable in certain applications. Recently, Ali et al. [7] proposed an ownership proofing scheme based on threshold generator. The scheme simply combines register odd number using a secret key.

2.2. Fragile Schemes

Regarding fragile watermarking, in [8], Li et al. proposed a group-based fragile scheme for watermarking database relations with categorical data. In this technique, a watermark is embedded and verified in each group independently by manipulating the order of tuples. Another example of group-based fragile scheme has been proposed by Guo et al. [9] for database integrity checking. For watermarking embedding, two LSBs of the of all attributes in all tuples are altered. Guo and colleagues [10] also proposed another fragile watermarking scheme for streaming data. In this scheme, the watermark insertion is done by altering one LSB of some numerical data streams. In [13], the author proposed a zero-distortion fragile database watermarking scheme which appends a new attribute containing a checksum of all other attributes and an aggregate value for anyone of the relation numeric attributes. Clearly the scheme introduces a space overhead due to storage of the appended attribute. Zhang et al. [14] proposed a reversible watermarking scheme for database authentication using expansion on data error histogram. This scheme is claimed to do not cause any permanent distortion because of attribute restoration. In [11], the author proposed a method for protecting data integrity by watermarking database indexes. The scheme is based on zero distortion as it does not alter attribute values. In [14], a watermarking algorithm for integrity checking of non-numeric data is proposed. The scheme is based on the concept of eigen values which are used for watermark embedding in non-numeric attributes. In [12], we proposed a content-based fragile zero-watermarking scheme for database authentication without fear of any constant on the type of attributes. This scheme is distortion free because the watermark bits are not actually encoded in the data. Instead, the watermark is registered and notarized with a trusted third party for authentication purpose. Recently, in another work [15], we proposed a clustering-based semi-fragile scheme for integrity verification of relational data. This scheme associates a weight to each database attribute that reflects it sensitivity to benign updates. As a result, the scheme can detect and localize malicious modifications, while allowing legitimate distortions.

3. Proposed hybrid scheme

Our proposed hybrid watermarking scheme is a multiple watermarks application that has two main components: a robust component and fragile component. These components can be used independently, depending on the dispute nature: ownership claim or tamper proofing.
Suppose $R(P_k, A_0, \ldots, A_{m-1})$ is the database relation being watermarked, where $P_k$ is the primary key attribute, and $A_i$ (i = 0, ..., m-1) is a numeric attribute candidate for marking. We assume that it is acceptable to alter $\xi$ LSBs of all attributes without affecting the data usefulness. Table 1 summarizes the parameters and notations used in this paper.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\eta$</td>
<td>Number of tuples in the relation</td>
</tr>
<tr>
<td>$m$</td>
<td>Number of attributes in the relation</td>
</tr>
<tr>
<td>$1/\lambda$</td>
<td>Fraction of robustly marked tuples</td>
</tr>
<tr>
<td>$r_i.A_j$</td>
<td>$j^{th}$ attribute of the $i^{th}$ tuple</td>
</tr>
<tr>
<td>$r_i.p_k$</td>
<td>Primary key of $i^{th}$ tuple</td>
</tr>
<tr>
<td>$h_i$</td>
<td>Hash value of $i^{th}$ tuple’s primary key</td>
</tr>
<tr>
<td>$h_{n_i}$</td>
<td>Hash value of $i^{th}$ attribute’s name</td>
</tr>
<tr>
<td>$K_s$</td>
<td>Secret key</td>
</tr>
<tr>
<td>$g$</td>
<td>Number of groups in the relation</td>
</tr>
<tr>
<td>$G_q$</td>
<td>$q^{th}$ group</td>
</tr>
<tr>
<td>$v$</td>
<td>Average number of tuples in a group</td>
</tr>
<tr>
<td>$W_r$</td>
<td>Robust watermark</td>
</tr>
<tr>
<td>$W_f$</td>
<td>Fragile watermark embedded in $q^{th}$ group</td>
</tr>
<tr>
<td>$w_b$</td>
<td>Basic watermark used to compute the robust watermark</td>
</tr>
<tr>
<td>$\tau$</td>
<td>Threshold parameter in robust watermark detection</td>
</tr>
</tbody>
</table>

### 3.1. Hybrid watermark embedding

The framework of the hybrid watermark embedding is shown in Figure 1. The watermark embedding consists of two phases: robust embedding and fragile embedding. The fragile component has to be applied after the robust one in order to hash the robust watermark with the data. As a result, the integrity of the robust watermark is guaranteed by the fragile component. Any attack aimed to destroy or remove the robust watermark will be correctly detected and localized as malicious modification. For the hashing purpose, any cryptographic one way hash function such as MD5 or SHA-1 can be used [26]. To ensure the robustness and reliability of the robust watermark, we propose to use the LSB of every attribute for embedding the fragile watermark, while the robust watermark bits can be encoded into the remainder ($\xi$-1) LSBs.

#### 3.1.1 Robust watermark embedding

The embedding algorithm of the robust watermark is shown in Figure 2. This procedure is an extension of Li et al.’s algorithm [5]. The first difference between the two algorithms is that our robust component is designed for copyright protection in a hybrid context, whereas Li et al.’s algorithm is proposed for traitor tracing. Furthermore, in Li et al’s algorithm it is assumed that the relation being watermarked has a fixed order of attributes that either never change or else can be recovered. In other words, Li et al’s scheme critically depends on the original order of attributes. As a result, a simple modification of the order of attributes will randomize the embedded watermark bits. Therefore, the scheme is vulnerable to attribute related attacks such as attribute sorting attacks which do not actually change data content. In our proposed technique, to deal with this limitation, we propose a technique that enables the data owner to define a secret order of the attributes that can be efficiently recovered at watermark detection. Indeed, as described in lines 1 through 3 in Figure 2, our solution consists to first compute a name hash value for each attribute of the database. Then the attributes are reordered in ascending order of their name hash values. Notice that this sorting process is a virtual operation and does not physically change the original attributes order. For watermark insertion, a (multiple bits) robust watermark $W_r$ is computed from the hash value of a basic watermark information $w_b$ which is decided by the owner. Next, for each tuple, a primary key hash value is computed. Accordingly, the
algorithm selects the marking tuples, as well as the marking attributes and marking bit positions. To accommodate the existence of the fragile watermark, whenever the selected marking bit index equals to 0 (i.e., LSB position), the index is moved to the second LSB.

```
// attributes reordering
1. for each attribute Aᵢ ∈ R repeat steps 2 to 3
2. Compute attribute name hash: hᵢ = Hash (Kₛ || Aᵢ || Kₛ)
3. Sort attributes in ascending order of attribute name hash values
4. Compute multiple bits watermark: Wᵣ = s MSBs from Hash (Kₛ || wᵦ || Kₛ)
5. for each tuple rᵢ ∈ R repeat steps 6 to 14
6. Compute primary key hash value: hᵢ = Hash (Kₛ || rᵢ.Pk || Kₛ)
7. Select marking tuple: if (hᵢ mod λ == 0) then
8. Select marking attribute: attr_ind_j = hᵢ mod m
9. Select marking bit index: bit_ind_k = hᵢ mod ξ
10. if (k == 0) then
11. k = k+1 // move to 2nd LSB
12. Calculate mask bit: tᵢ = hᵢ mod 2
13. Select watermark bit index: bᵢ = hᵢ mod s
15. Return R'
```

**Figure 2. Robust watermark embedding algorithm**

### 3.1.2. Fragile watermark embedding

The embedding algorithm of the fragile watermark is shown in Figure 3. First, the robustly marked relation $R'$ is partitioned into $g$ groups using the same secret key $Kₛ$ used in the robust component. Next, all tuples in a group are sorted in ascending order of the corresponding primary key hash values. Notice that the grouping and sorting operations do not change the physical positions of the database tuples. Instead, these two operations are useful for enforcing some relationship between the relation tuples so that to assure the watermarks synchronization. Thereafter, a keyed hash value is computed for each attribute while ignoring the least significant bit since it is intended for encoding the watermark bits. Next, a watermark is embedded in each group as shown in lines 12 through 17 in Figure 3. Let $W_f$ be the watermark associated to a particular group. For each attribute, a bit vector of size $v$, computed from its hash value, is inserted in $W_f$. After all attributes are processed, the resultant watermark $W_f$ will be a Boolean matrix of $v$ rows and $m$ columns. Finally, for each $r_i.A_j$ in the group, the corresponding watermark bit $W_f[i,j]$ is embedded in the least significant bit of $r_i.A_j$.

### 3.2. Hybrid watermark detection

Figure 4 highlights the detection process of our proposed hybrid scheme. The robust and the fragile watermarks are extracted separately using the same secret key as the embedding algorithms do. We
consider the relation \( D \) to be a subset of the relation \( R \). However, \( D \) may also contain additional tuples not included in \( R \).

**Figure 3.** Fragile watermark embedding algorithm

**Algorithm: Embed_fragile_watermark**

**Input:** Robustly Watermarked relation \( R' \), Secret key \( K_s \) and Number of groups \( g \)

**Output:** Hybirdly watermarked relation \( R'' \)

// data grouping
1. for each tuple \( r_i \in R' \) repeat steps 2 to 3
2. Determine group index: \( \text{group}_\text{ind}_q = h_i \mod g \) // \( h_i \) primary key hash, see figure 2
3. Insert tuple \( r_i \) into group \( G_q \)
4. for each group \( G_q \in R' \) repeat steps 5 to 6
5. Sort all tuples in \( G_q \) in increasing order of their primary key hash
6. Generate watermark bits for \( G_q \): \( W_f = \text{genBits}(G_q, v) \) // see lines 11-16
// Insert watermark in \( G_q \)
7. for each tuple \( r_i \in G_q \) repeat steps 8 to 9
8. for each attribute \( A_j \)
9. Embed mark bit: LSB of \( r_i.A_j = W_f[i,j] \)
10. Return \( R'' \)
11. genBits(\( G,d)\) [\( \text{for each attribute } A_j \in G \) repeat steps 13 to 15
12. Calculate attribute \( H_j = \text{Hash}(K_s||r_0.A_j||...||r_{d-1}.A_j||K_s) \) // exclude the LSB for all values
13. Extract a bit string \( S_j \) from \( H_j \): \( S_j = d \text{ MSBs from } H_j \) // assume \( d \leq \text{length}(H_j) \)
14. Insert \( S_j \) in \( W \)
15. Return \( W \)

**Figure 4.** Hybrid watermarking detection process

### 3.2.1. Robust watermark detection

The robust watermark detection algorithm is shown in Figure 5. In this phase, since attributes original order may have been changed, the algorithm first recovers the secret order of attributes defined at the embedding stage. Then, three voting variables are defined for extracting the watermarking bits. The counting variable \( \text{matchCount}_0[i] \) determines the number of times that a watermark bit \( w_i \) is extracted to be 0, while the variable \( \text{matchCount}_1[i] \) indicates the number of times the watermark bit \( w_i \) is extracted to be 1. Next, according to the primary key hash values, the algorithm selects the marked tuples, marked attributes as well as the marked bits. Thereafter, by majority voting, the value (0 or 1) is assigned to the recovered watermark bit \( W_r[i] \), according to a threshold \( \tau \in [0.5, 1) \) which is a real parameter related to the watermark detection process. Finally, to determine whether the suspected relation \( D \) is pirated or not, the original watermark \( W_r \) is computed and compared to the extracted watermark \( W_r' \). If the two watermarks match then the algorithm concludes that there is no piracy, and suspects the existence of piracy otherwise.
3.2.2. Fragile watermark detection

Figure 6 shows the detection process of the fragile watermark. Since the integrity of the suspicious relation $D$ can be verified independently from the robust part, the fragile watermark detection phase also starts by recovering the secret order of attributes. Then, the relation $D$ is securely divided into $g$ partitions, as in Figure 3. The watermark verification is performed for each group independently as described in lines 7 through 14 in Figure 6. The watermark bits hidden in each group are extracted from the least significant bit of the corresponding attributes. Thereafter, the extracted watermark $W'_j$ is compared with its corresponding original one $W_j$ and the result is assigned to a verification vector $V$, with $V[j]$ denoting the verification result for the $j^{th}$ group. An element $V[j]$ is set to true if the $j^{th}$ group original watermark matches its associated extracted watermark; otherwise it is set to false. As a result, any change to the watermarked relation can be detected and localized at the group level.

```
Algorithm: Detect_robust_watermark
Input: Relation $D$, Secret key $K_s$, Basic watermark $w_b$, Secret parameters $m$, $s$, and $\lambda$
Output: Detected robust watermark $W'_r$

1. Recover secret order of attributes // see Figure lines 1-3 in figure 2
2. totalCount = 0
3. for $i = 0$ to $s-1$ repeat step 4
4. matchCount_0[i] = matchCount_1[i] = 0;
5. for each tuple $r_i \in D$ repeat steps 6 to 21
6. Compute primary key hash value: $h_i = Hash(K_s || r_i.P || K_s)$
7. Select marked tuple: if ($h_i \mod \lambda = 0$) then
8. Select marked attribute: $attr_{ind} = h_i \mod m$
9. Select marked bit index: $bit_{ind} = h_i \mod \xi$
10. if ($k == 0$) then
11. $k = k+1$
12. Calculate mask bit: $t = h_i \mod 2$
13. Determine watermark bit index: $x = h_i \mod s$
14. Determine watermark bit: $w_x = kth$ LSB of $r_i.A_j \oplus t$
15. totalCount = totalCount + 1
16. if ($w_x == 0$) then
17. matchCount_0[x] = matchCount_0[x] + 1
18. else
19. matchCount_1[x] = matchCount_1[x] + 1
// Majority voting
20. for $i = 0$ to $s-1$ repeat steps 20 to 24
21. if matchCount_0[i] / totalCount > $\tau$ then
22. $W'_r[i] = 0$
23. else
24. if matchCount_1[i] / totalCount > $\tau$ then
25. $W'_r[i] = 1$
// Check piracy
26. Compute original watermark:
27. $W_r = s$ MSBs from Hash($K_s || w_b || K_s$);
28. if $W'_r \neq W_r$ then
29. piracy suspected
30. else
31. no piracy suspected
32. Return $W'_r$
```

Figure 5. Robust watermark detection algorithm

3.3. Discussions

3.3.1. Benign updates

Benign updates performed over the hybridly watermarked relation do not require the re-watermarking of the whole relation. When a new tuple is inserted, the robust algorithm only selects this
row for marking or not based on the same marking criteria used for the existing tuples. The deletion of tuples does not require any processing by the robust component. For the fragile component, only the group(s) affected by the modification should be re-watermarked.

3.3.2. Tamper detection at attribute level

Suppose that the value \( r_i.A_j \) of a given group has been maliciously modified. Since the watermark is made up with bit vectors extracted from each attribute hash value, this change can be detected and localized by just checking the integrity of the relevant attribute hash \( H_j \). In this scenario, a verification vector is constructed for each group, and each element of this vector is associated to a specific attribute.

```
Algorithm: Detect_fragile_watermark
Input: Relation D, Secret key Ks, and Number of groups g
Output: Verification vector for groups

1. Recover secret order of attributes:  // see lines 1-3 in Figure 2
2. Data grouping        // see lines 1-3 in Figure 3
3. For each group \( G_q \in D \) repeat steps 4 to 6
4. Sort all tuples in \( G_q \) in increasing order of their primary key hash
5. Generate watermark bits: \( W_f = \text{genBits}(G_q,v) \) // see lines 11-16 in Figure 3
6. Check the authenticity of \( G_q \): checkWatermark(\( G_q,W_f \)) // see lines 7-14
7. checkWatermark(\( G_k,W \) ) {
   8. for each tuple \( r_i \in G_k \) repeat steps 2 to 3
   9. for each attribute \( A_j \) repeat step 3
   10. \( W'[i,j] = \text{LSB of } r_i.A_j \)
   11. if (\( W' \neq W \)) then
      12. return \( V[k] = \text{false} \)   // \( G_j \) tampered with
      13. else
      14. return \( V[k] = \text{true} \)   // \( G_j \) is authentic }
```

Figure 6. Fragile watermark detection algorithm

4. Security Analysis

The watermarked database relation can be subject to many attacks either targeting at destroying the robust watermark while keeping data useful, or aiming at maliciously modifying data while keeping the fragile watermark untouched. We consider three typical database attacks which are common to both robust and fragile watermarks: attribute value alteration, subset deletion, and subset insertion.

4.1. Robustness

Regarding the robust component, we analyze the failure probability in recovering the embedded watermark. We use the following notation. Let \( X \) be a random variable that follows the binomial distribution with parameters \( n \) and \( p \). The probability of getting exactly \( k \) successes in \( n \) Bernoulli trials is given by the binomial distribution function:

\[
f(k;n, p) = \binom{n}{k} p^k q^{n-k}
\]

where \( p \) is the probability of success, \( q=1-p \) is the probability of failure.. The probability of getting more than \( k \) successes out of \( n \) independent Bernoulli trials is given by the cumulative distribution function
4.1.1 Attribute value alteration

In this kind of attack, the pirate tries to destroy the embedded watermark bits by randomly flipping-backs their values. Assume that a marked bit $w_i$ is embedded and extracted $\alpha_i$ times. If we further assume that each marked bit is toggled with probability $p > 0.5$. Since there are $s$ bits in $W$, due to the majority voting, the probability that the watermark is completely destroyed is:

$$P = 1 - \prod_{i=0}^{s-1} (1 - F(\frac{\tau \alpha_i}{2}, p))$$

Figures 7a shows the failure rate related to this attack. We can see that the chance of success of this attack is very low. Indeed, the attack cannot succeed unless more than 40% of the embedded bits are destroyed, and doing so will seriously affect the data usability.

4.1.2. Subset deletion

Suppose that the attacker deleted some tuples from the marked relation with the probability $p$. The detection algorithm will fail to recover the embedded watermark only if all relevant embedded bits of at least one watermark bit $w_i$ are deleted by the attack. If we assume that $w_i$ is embedded $\alpha_i$ times, the probability that the watermark is not recovered is:

$$P = 1 - \prod_{i=0}^{s-1} (1 - F(\alpha_i, \tau, p))$$

Figure 7b shows the failure probability for the robust watermark under subset deletion attack. We can see that this kind of attack cannot completely destroy the embedded watermark even if 90% of the tuples were deleted.

4.1.3. Subset insertion

Suppose that the attacker added some tuples to the watermarked database. Let $\alpha_i$ denotes the number of times a watermark bit $w_i$ is embedded in the original tuples, and $\beta_i = \alpha_i p_i$ the number of times it is extracted from the additional tuples, where $p_i$ ($p_i > 0$) is the insertion rate. Due to the majority voting, the probability of failure to recover the whole watermark is given by

$$P(X > k) = \sum_{i=k}^{n} f(i; n, p) = F(k; n, p)$$
\[ P = 1 - \prod_{i=0}^{s-1} \left( 1 - F\left( \bar{r}(\alpha_i, \beta_i) \right) \right) \]  \hspace{1cm} (5)

where \( p = 0.5 \) is the probability of success in a Bernoulli trial. Obviously, this attack is less effective than both attribute value alteration and subset deletion attacks.

4.2. Tamper proofing

4.2.1 Attribute value alteration

Suppose that only a single attribute value \( r_iA_j \) has been modified by the attack. This change could randomize the attribute hash as well as the relevant bit vector in the group watermark. Since there are \( v \) bits in the vector, the probability that these bits remain unchanged after the attack is

\[ P = \frac{1}{2^v} \]  \hspace{1cm} (6)

We can see that, the chance of success of this attack is monotonic decreasing with the group size \( v \).

4.2.2 Subset deletion

Suppose that \( \gamma \) tuples are deleted from a single group of \( v \) tuples. As a result, the change could randomize all attribute hash values and the watermark as well. Since, only \( mv-m\gamma \) bits remain in the watermark after the attack, the corresponding failure rate can be written as

\[ P = \frac{1}{2^{m(v-\gamma)}} \]  \hspace{1cm} (7)

Clearly, the error rate for this attack decreases with the watermark, but increases with larger \( \gamma \).

4.3.3 Subset insertion

Suppose that \( \gamma \) new tuples are added to a particular group. As a result, the relevant watermark would be randomized and its size is increased of \( m\gamma \) bits. Therefore, the corresponding failure probability is

\[ P = \frac{1}{2^{m(v+\gamma)}} \]  \hspace{1cm} (8)

Observe that this error rate decreases with the group size and the number of inserted tuples as well.

5. Experiments

In this section, we report some experimental results to demonstrate the feasibility of our hybrid watermarking technique. We essentially focus on the distortions introduced by watermarks embedding in two important statistical indicators of every marked attribute, namely mean and variance. The experiments were conducted on a computer running Microsoft SQL Server 2005, with a 2.2 GHz Intel Dual CPU, and 1GB of RAM. We tested our algorithms on a real life dataset, the Forest Cover Type dataset (http://kdd.ics.uci.edu/databases/covertype/covertype.html). There are 581,108 tuples in the dataset, each with 54 attributes. For watermark insertion, we used the 10-first integer attributes, and added a new attribute called \( id \) to serve as primary key. We set the robust watermark size \( s \) to 60, the threshold \( \tau \) to 0.5, and \( \xi \) to 4 LSBs. We used three different values (25, 50, and 100) for \( \lambda \). Notice that, since every attribute in all tuples are fragilely marked, the parameter \( g \) has no impact on the alterations...
induced by the insertion of the watermarks. For this reason, we set $g$ to 5,810 so that there were roughly 100 tuples in each group.

Table 2 shows the impact of watermarks insertion on the mean and variance of marked attributes. The values are rounded to the nearest integer. We observed that the statistical indicators values remained identical for the three different values of $\lambda$ except that, for $\lambda=25$, the variance increased by 1 for the attribute 4. This result is important because $\lambda$ can be used to balance between robustness and imperceptibility. Furthermore, it’s easy to see that the mean only increases by 1 for attributes 1 and 4. Though the change in the variance is higher than the one in the mean for some attributes, these errors are still minor, and can be tolerated in many real world applications.

### 6. Conclusion

A novel hybrid watermarking method for protecting relational databases is presented in this paper. Two watermarks: one robust and the other fragile are embedded into data for ownership verification and tamper detection, respectively. The fragile watermark is inserted in a way that assures the reliability and the integrity of the robust watermark. As a result, our hybrid scheme can be considered as a new class of multiple watermarks methods for databases. Security analysis showed that the scheme is secure against various database attacks, and experimental results demonstrated that the distortions caused by watermarks insertion are minuscule. In the future, we will investigate hybrid solution for watermarking database relations with non-numeric attributes.

Table 2: Errors introduced in mean and variance by watermarks embedding

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Before embedding</th>
<th>After embedding</th>
<th>Error in mean</th>
<th>Error in variance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Variance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2959</td>
<td>78391</td>
<td>+1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>156</td>
<td>12525</td>
<td>0</td>
<td>+1</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>56</td>
<td>0</td>
<td>+1</td>
</tr>
<tr>
<td>4</td>
<td>269</td>
<td>45177</td>
<td>+1</td>
<td>-37</td>
</tr>
<tr>
<td>5</td>
<td>46</td>
<td>3398</td>
<td>0</td>
<td>-2</td>
</tr>
<tr>
<td>6</td>
<td>2350</td>
<td>2431272</td>
<td>0</td>
<td>-74</td>
</tr>
<tr>
<td>7</td>
<td>212</td>
<td>717</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>223</td>
<td>391</td>
<td>0</td>
<td>+1</td>
</tr>
<tr>
<td>9</td>
<td>143</td>
<td>1465</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>1980</td>
<td>78391</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

7. Acknowledgements

This work is supported by the National Natural Science Foundation of China (60736016, 60973128, 60973113, 61070196, 61070195, and 61073191), National Basic Research Program 973 (2009CB326202, 2010CB334706, 2011CB311808), PAPD fund.

8. References

A Hybrid Watermarking Scheme for Relational Databases Copyright Protection and Tamper Proofing  
Ali Hamadou, Xingming Sun, Saeed Arif Shah, Lingyun Gao


