Expert Systems with Applications 39 (2012) 3185–3196

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

A blind reversible method for watermarking relational databases based on a time-stamping protocol

Mahmoud E. Farfoura [a], Shi-Jinn Horng [a,*], Jui-Lin Lai [b], Ray-Shine Run [b], Rong-Jian Chen [b], Muhammad Khurram Khan [c]

[a] Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, 106 Taipei, Taiwan
[b] Department of Electronic Engineering, National United University, 36003 Miao-Li, Taiwan
[c] Center of Excellence in Information Assurance, King Saud University, Saudi Arabia

A R T I C L E I N F O

Keywords:
Relational database
Digital watermarking
Copyrights
Blindness
Robustness
Database attacks
Reversible
Cryptography

A B S T R A C T

Digital watermarking technology has been adopted lately as an effective solution to protecting the copyright of digital assets from illicit copying. Reversible watermark, which is also called invertible watermark, or erasable watermark, helps to recover back the original data after the content has been authenticated. Such reversibility is highly desired in some sensitive database applications, e.g. in military and medical data. Permanent distortion is one of the main drawbacks of the entire irreversible relational database watermarking schemes. In this paper, we design an authentication protocol based on an efficient time-stamp protocol, and we propose a blind reversible watermarking method that ensures ownership protection in the field of relational database watermarking. Whereas previous techniques have been mainly concerned with introducing permanent errors into the original data, our approach ensures one hundred percent recovery of the original database relation after the owner-specific watermark has been detected and authenticated. In the proposed watermarking method, we utilize a reversible data-embedding technique called prediction-error expansion on integers to achieve reversibility. The detection of the watermark can be completed successfully even when 95% of a watermarked relation tuples are deleted. Our extensive analysis shows that the proposed scheme is robust against various forms of database attacks, including adding, deleting, shuffling or modifying tuples or attributes.

© 2011 Elsevier Ltd. All rights reserved.

1. Introduction

Copyright protection of owners is becoming more and more necessary due to the rapid growth of the Internet, the wide development of digital multimedia contents, and the easier distribution. As an important area in information hiding (Petitcolas, Anderson, & Kuhn, 1999), digital watermarking provides a promising method of protecting digital data from illicit copying, and manipulation by embedding a secret code directly into the data. Digital watermarking allows the user to add a layer of protection to the digital media content by identifying copyright ownership and delivering a tracking capability. Accordingly, it monitors and reports where the user's digital media contents are being used.

Cryptography provides a means of secure delivery of content to the consumers only. Not all legitimate consumers are trustworthy, and untrustworthy consumers may alter or copy the decrypted content illegally. However, cryptography provides no protection once the content is decrypted, which is required for human perception. Watermarking complements cryptography by embedding a message within the content (Cox, Doerr, & Furon, 2006).

There is a rich body of literature on watermarking multimedia objects, starting from watermarking still images (Cox, Kilian, Leighton, & Shamoon, 1997; Wong & Memon, 2001) and later extended to digital video (Hartung & Girod, 1998) and audio (Arnold, 2000) and text (Brassil, Low, & Maxemchuk, 1999). Digital watermarking was exploited in other digital media like protecting software (Collberg & Thomborson, 2002), natural language (Atallah, Raskin, & Hempelmann, 2002) and sensor data (Sion, Atallah, & Prabhakar, 2004a).

The basic watermarking procedures like watermark insertion and detection to multimedia objects cannot be applied directly to watermarking relational databases due to the differences in the characteristics of multimedia and relational data. Unlike highly correlated multimedia data whose relative positions are fixed, database relations contain independent tuples with little redundancy. The tuples can be added, deleted or modified frequently in either benign updates or malicious attacks.

In the conventional irreversible watermarking schemes, only the embedded watermark information can be extracted from the

* Corresponding author.
E-mail addresses: mfarfora@rss.gov.jo (M.E. Farfoura), horngsj@yahoo.com.tw (S.-J. Horng), jllai@nuu.edu.tw (J.-L. Lai), run5116@ms16.hinet.net (R.-S. Run), rjchen@nuu.edu.tw (R.-J. Chen), mkhurram@ksu.edu.sa (M.K. Khan).

0957-4174/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.09.005

suspected data; however, in reversible watermarking schemes, the original objects can be recovered as well as the embedded watermarks.

Comparatively, watermarking of relational databases started to receive attention due to the increasing use of database systems in many real-life applications (Agrawal & Kiernan, 2002; Li & Deng, 2003; Li, Guo, & Jajodia, 2004; Li, Guo, & Wang, 2008; Li, Swarup, & Jajodia, 2003; Shehab, Bertino, & Ghafoor, 2008; Sion, 2004; Sion, Atallah, & Prabhakar, 2004b). Nonetheless, most watermarking schemes have been irreversible (the original relation cannot be restored from the watermarked relation). One major motivation for reversible watermarking is some real-life applications such as outsourced medical and military data. These kinds of data do not allow any losses. Another purpose of reversible watermarking is providing shareware or trial versions of specific database applications where the original database version can be recovered after the consumer buys the license of that application.

Since most the previous relational database watermarking schemes (Agrawal & Kiernan, 2002; Li & Deng, 2003; Li et al., 2003, 2004, 2008; Shehab et al., 2008; Sion et al., 2004b) suffer from an intrinsic problem, where an attacker can falsely claim ownership, since the previous schemes rely only on the robustness of the embedding algorithms, apparently an attacker can succeed to destroy the embedded watermarks under some conditions. Our new authentication protocol comes to solve this problem by introducing a trusted time-stamping service (TSS) to the watermarking scheme which plays a significant role in authenticating the watermarked relation and the embedded watermark besides to relation owners.

In this work, we propose a blind reversible method for watermarking relational database which proves the true ownership of the database owner. In the proposed scheme, we embed a stream of owner-specific watermark bits in the fractional portion of numeric attributes by utilizing a reversible data-embedding technique called prediction-error expansion proposed by Thodi and Rodriguez (2004). Once a database relation is outsourced and the owner suspects any intentional misuse, he/she can use the detection procedure to check whether the outsourced relation is abused by detecting and verifying the embedded owner-specific watermark information.

The rest of this paper is organized as follows: Section 2 gives an overview of related researches and preliminary background. Section 3 explains in detail the proposed watermarking scheme, including watermark insertion and watermark detection. In Section 4, we present robustness analysis. In Section 5, we present experiment details and Section 6 concludes this paper with summaries and suggestions for future works.

2. Related researches and preliminary background

2.1. The previous irreversible schemes

The first well-known conventional (irreversible) database watermarking scheme was proposed by Agrawal and Kiernan (AK for short) (Agrawal & Kiernan, 2002) for watermarking numerical values in relational databases. The fundamental assumption is that the watermarked database can tolerate a small amount of errors. In AK (Agrawal & Kiernan, 2002) scheme, the private key $\mathcal{K}$, used for copyright verifiability, concatenated with the primary key of the tuple, is the seed for the pseudorandom number generator algorithm which decides the tuples, attributes within a tuple, and bit positions within an attribute to be marked. Only when the attacker has access to the private key, it can detect the watermark with high probability. The technique survives several attacks and preserves mean and variance of all numerical attributes.

Agrawal and Kiernan's scheme (Agrawal & Kiernan, 2002) cannot be directly applied to watermarking categorical data since any change, no matter how small it is, to a categorical value may render the value meaningless. To solve this problem, Sion (2004) proposed a scheme to watermark a categorical attribute by changing some of its values to other values of the attribute (e.g., 'red' is changed to 'green') if such change is tolerable in certain applications. In Li et al. (2003) extended the work by Agrawal and Kiernan (2002) to provide for multi-bit watermarks in a direct domain encoding which in turn increases the length of embedded watermarks. Since some sensitive types of applications like medical and military cannot bear any permanent data distortion, the aforementioned works cannot be applied to watermark those relations.

2.2. The previous reversible schemes

Several reversible watermarking schemes in the field of digital multimedia have been proposed. Data compression based reversal (Celik, Sharma, Tekalp, & Saber, 2005) involves methods to losslessly compress a set of selected features from an image and embed the watermark information in the space saved due to the compression. They use a binary-to-*L*-ary conversion to embed the secret payload. In Celik et al. scheme, the payload capacity greatly relies on two factors: the number of the *L*-levels and the number of smooth regions in the cover image. The larger *L* and the smoother regions, the higher payload an embedding method has. However, data compression based schemes lack robustness because the applied compression methods cannot resist the distortions.

Histogram shifting techniques (Vleeschouwer, Delaigle, & Macq, 2001) rely on the fact that adjacent pixels in grayscale images are close to each other, so pixels in an image can be divided into distinct zones, and a histogram for each zone can be calculated. After that, if the watermark bit is 1 then the histogram circularly upgraded, or downgraded if the watermark bit is 0. This technique is not suitable in the case of relational database watermarking because, as we mentioned in Section 1, the data in a relation are discrete and not related.

Difference expansion (DE) techniques (Alattar, 2004; Tian, 2003) schemes usually generate some small values to represent the features of the original image. Then, expand or enlarge the generated values to embed the bits of watermark information. The watermark information is usually embedded in the LSB (least significant bit) parts of the expanded values. Then the watermarked image is reconstructed by using the modified values. In Tian's scheme, an integer transformation is defined as

$$g = \left\lfloor \frac{(x+y)}{2} \right\rfloor, \tag{1}$$

$$d = x - y \tag{2}$$

For example, $x = 106$ and $y = 104$ are two adjacent pixels. Then the difference d and the average g can be computed as follows: $d = 106 - 104 = 2$, and $g = \lfloor \frac{(106+104)}{2} \rfloor = 105$. Here, $\lfloor x \rfloor$ denotes the greatest integer smaller than x (floor function). To embed a watermark bit $i = 1$ into the pixel pair, the difference d is represented using the binary format, shift it left by one bit and append the watermark bit i into the vacant LSB. If l is the bit length of d (i.e., $d = b_{l-1}b_{l-2}\cdots b_0$), then the new difference value d' can be obtained as

$$d' = b_{l-1}b_{l-2}\cdots b_0 i = 2 * d + i = 2 * 2 + 1 = 5 \tag{3}$$

Then, the new pixel values is given by

$$x' = g + \left\lfloor \frac{(d'+1)}{2} \right\rfloor = 105 + 3 = 108 \tag{4}$$

$$y' = g - \left\lfloor \frac{d'}{2} \right\rfloor = 105 - 2 = 103 \tag{5}$$

At the decoder side, the watermark bit can be extracted from the LSB of the difference value, and the original difference value can be restored

$$\begin{aligned} d' &= x' - y' = 108 - 103 = 5 \\ i &= \mathrm{LSB}(d') = \mathrm{LSB}(101_2) = 1 \end{aligned} \tag{6}$$

$$d = \left\lfloor \frac{d'}{2} \right\rfloor = \left\lfloor \frac{5}{2} \right\rfloor = 2 \tag{7}$$

Then the inverse integer transform is given by Eqs. (8) and (9). Thus, original pixel values x and y can be recovered completely

$$x = \left\lfloor \frac{(x' + y')}{2} \right\rfloor + \left\lfloor \frac{(d+1)}{2} \right\rfloor = 105 + 1 = 106 \tag{8}$$

$$y = \left\lfloor \frac{(x' + y')}{2} \right\rfloor - \left\lfloor \frac{d}{2} \right\rfloor = 105 - 1 = 104 \tag{9}$$

Gupta and Pieprzyk (2008) have proposed a reversible watermarking scheme based on DE. The major drawback of his scheme is that it requires two attributes to embed a single watermark bit, and the amount of distortion introduced in the attributes becomes directly proportional to the numerical difference of the two attributes. And since data in a relation are discrete and values are not related, it is trivial to have a high distortion in the final watermarked relations. Another problem that this scheme faces is more vulnerable to alteration attack; thereby any alteration to any one of the participated attributes will fail detecting the watermark and recovering the original data. Our objective in this paper is to show that embedding a watermark bit in a single attribute will be a better choice and will solve the previous problems raised in Gupta and Pieprzyk (2008).

Instead of using DE, prediction-error expansion (PE) proposed in Thodi et al. (2004) is utilized in this work. In PE, a predictor instead of a difference operator is used to create the feature elements into which expansion embedding will be done. A predictor better exploits the correlation inherent in the neighborhood of a pixel. The prediction errors are the feature elements into which expansion embedding is done. Consider a pixel with intensity y in a gray-scale image and a watermark bit i to be embedded. A predictor operates on the neighborhood of pixel y and predicts its intensity $y^\wedge$. The prediction error p_e is given by

$$p_e = y - y^\wedge \tag{10}$$

To embed a watermark bit i into prediction-error p_e, p_e is represented by using the binary format. Shift it left by one bit and append the watermark bit i into the vacant LSB. If l is the bit length of d (i.e., $p_e = b_{l-1}b_{l-2}\cdots b_0$), then the new prediction-error p'_e can be obtained as:

$$P'_e = b_{l-1}b_{l-2}\cdots b_0 i = 2 * p_e + i \tag{11}$$

The modified prediction-error, p'_e and the predicted intensity $y^\wedge$ are combined to calculate the embedded pixel intensity y' (i.e. the watermarked value), where

$$y' = y^\wedge + p'_e = y - p_e + p'_e = y + p_e + i \tag{12}$$

At the detector side, the intensity of the pixel is predicted by using the same predictor as employed by the embedder. The predicted intensity will be $y^\wedge$ (if the neighborhood has not been altered). The prediction error at the decoder is $y' - y^\wedge = p'_e$. The embedded bit is the LSB of p'_e, i.e. $i = \mathrm{LSB}(p'_e)$, the true pixel intensity is restored after calculating the original prediction error as

$$p_e = \left\lfloor \frac{p'_e}{2} \right\rfloor \tag{13}$$

$$y = y' - p_e - i \tag{14}$$

In our approach, we assume that y is the fractional part of any numeric attribute to be watermarked, and $y^\wedge$ is any given value known at watermark insertion and detection time. In Section 3 we will give more details about the proposed watermarking method.

2.3. The previous time-stamping protocols for digital watermarking

Hwang, Hwang, and Chang (2005) proposed a time-stamping protocol for digital watermarking. The main purpose of a time-stamping technique is to ascertain whether a certain digital medium was created or signed at a certain time. Later, Chou, Chen, and Chan (2007) proposed another time-stamping protocol based on Hwang et al. (2005), they claimed that the former protocol is not secure enough against attacks, such like off-line attack, the reader is referred to Chou et al. (2007) and Hwang et al. (2005) for more details about their protocols. Both the protocols done by Hwang et al. (2005) and Chou et al. (2007) were mainly based on symmetric and asymmetric key cryptography. Hence it is proved that asymmetric key cryptography is more secure than symmetric one. Based on Chou et al. (2007) and Hwang et al. (2005), we propose a new time-stamping protocol relying only on asymmetric key cryptography, to solve the problem of multiple claims of ownership in relational database. In the following section we give the details of the proposed time-stamping protocol.

3. The proposed watermarking method

3.1. Time-stamping protocol for relational database watermarking

Firstly, we identify some of the notations used in our time-stamping protocol:

$\sigma = \mathrm{Sign}_{\mathrm{ks}}(\mathrm{RW})$: denotes applying the signing function Sign() to the relation RW using the private key ks.
$\mathrm{Ver}(\sigma, \mathrm{kp})$: denotes applying the verifying function Ver() to σ using the public key kp.
H(RW): denotes the hash value of relation RW.
$\mathrm{E}_{\mathrm{kp}}(X)$: denotes the encryption of input X using public key kp.
$\mathrm{D}_{\mathrm{ks}}(X)$: denotes the decryption of input X using private key ks.
$\mathrm{OID}_{\mathrm{i}}$: denotes the owner's ID.
t: denotes the timestamp.

The proposed protocol consists of two phases, the signing phase and the verification phase.

Our protocol runs in two passes as in Chou et al. (2007), but using asymmetric key cryptography only, which is a noteworthy improvement over (Chou et al., 2007). The trusted time-stamp service (TSS) acts as a legal trusted third party in which any owner can trust to register the watermarked database relation and some other important information such like the owner $\mathrm{OID}_{\mathrm{i}}$, and the user-specific watermark W. Assuming that each participant has their private keys, and the corresponding public key certificates are valid and verifiable with the presence of a public key infrastructure (Housley et al., 1999). Given keys and certificates, parties are able to authenticate their peers following standards such as Standard Specifications for Public Key Cryptography (2000).

The detailed procedures for our watermarking time-stamping protocol in the signing phase are shown in Fig. 1 and presented as follows:

Step 1. Owner $\mathrm{OID_i}$ encrypts the watermarked database relation RW, the watermark W, and his $\mathrm{OID_i}$, using TSS's public key $\mathrm{T_{kp}}$, to compute the following value c (=$\mathrm{E_{Tkp}(RW,W,OID_i)}$) and sends it to TSS.
Step 2. TSS decrypts the value c using his private key $\mathrm{T_{ks}}$ to obtain RW, W, and $\mathrm{OID_i}$. Then TSS uses a secure one-way hash function $\mathcal{H}()$ Schneier (1996) to compute the value of d (=$\mathrm{H(RW,W,OID_i},t)$) where t is the current time. After that, TSS signs on d by using his private key, obtaining s_1. Finally, TSS uses the owner's public key $\mathrm{OID_{i_{kp}}}$ to encrypt s_1, obtaining T_1, and to encrypt t to obtain T_2 and he sends the encrypted T_1 and T_2 to the owner $\mathrm{OID_i}$.
Step 3. When the owner receives T_1 and T_2, he uses his private key $\mathrm{OID_{i_{ks}}}$ to decrypt T_1 and T_2 obtaining the TSS's signature s_1, and timestamp t respectively. After that, the owner $\mathrm{OID_i}$ validates s_1 by verifying the value of s_1 by using the TSS's public key, to obtain d and he checks whether the result is equal to $\mathrm{H(RW,W,OID_i},t)$.

$$\mathrm{Ver}(s_1,\mathrm{kp}) = \mathrm{d}? = \mathrm{H}(\mathrm{RW},\mathrm{W},\mathrm{OID_i},t) \tag{15}$$

The time-stamping verification procedures are described as follows:
Step 1. The notary requests the owner $\mathrm{OID_i}$ to verify the time-stamp of database relation RW and the corresponding watermark W.
Step 2. The owner $\mathrm{OID_i}$ sends $(\mathrm{RW,W,OID_i},s_1,t)$ to the notary for arbitration.
Step 3. The notary verifies s_1, obtaining d and compares d to the computed $\mathrm{H(RW,W,OID_i},t)$ in Eq. (15).

The time-stamp verification phase is completed after step 3 is done.

For the situation of multiple owners claiming the ownership of a watermarked database relation, the problem could be solved by comparing the time-stamps, the owner who has the smaller time-stamp is the legal owner.

3.1.1. Security analysis

From the security perspectives, the protocol is more secure since all the messages sent between both TSS and owner $\mathrm{OID_i}$ are encrypted using public and private keys. For an attacker to succeed attacking a time-stamped watermarked database relation, based on this protocol, he should be able to forge RW, W, $\mathrm{OID_i}$, s_1, and t which are impossible.

We summarize the strength points of our watermarking time-stamping protocol:

- The protocol is efficient since it works on only two passes.
- The TSS is not required to store any messages.
- The protocol is impervious to forgery.
- The original database relation R is not required for verification, hence it is kept secret.

3.2. Assumptions

Our proposed reversible watermarking method is motivated from the AK scheme (Agrawal & Kiernan, 2002); we examine the following assumptions that are used in the AK scheme:

1. Primary key criticality: A database relation must have a primary key which is fixed at the time of watermark embedding and detection. Any alteration to it may decline the relation integrity, usability, and availability.
2. Error tolerance: A small modification to some attributes in a database relation that is being watermarked will not affect the whole data usability and availability.

3.3. Model

For a database owner Alice, in order to prove the ownership of a given relation R, she should be able to retrieve the embedded user-specific watermark W. The following desired properties should be satisfied:

1. Prevent illegal embedding and verification: The whole process is governed by a key; only an authorized person who has the key can embed, detect, and verify watermarks. This prevents unauthorized persons from inserting a false watermark or illegally verifying watermarks.
2. Blind detection: At the time of watermark detection, the watermark can be detected without the need of original database. Solely the key, which is typically used during the watermark insertion, is required.
3. Imperceptibility: The watermarked tuples and attributes, i.e. the position of watermark cannot be detected by an attacker.
4. Robustness: The watermarking scheme should be able to survive and resist against benign database updates and malicious attacks.

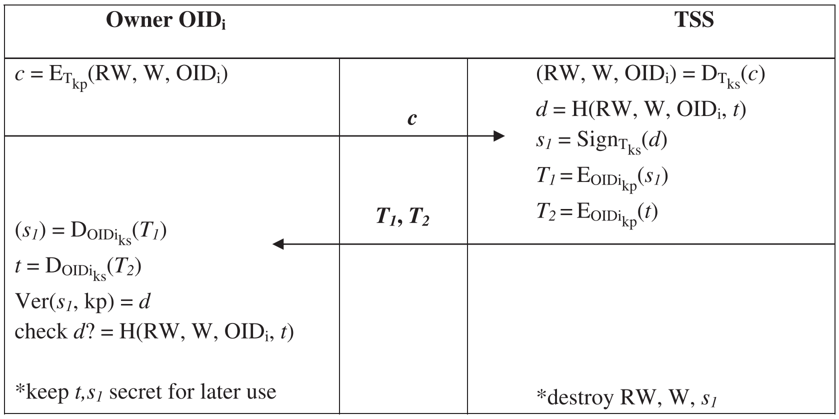
Owner $\mathrm{OID_i}$
TSS
$c = \mathrm{E_{T_{kp}}(RW, W, OID_i)}$
c
$\mathrm{(RW, W, OID_i) = D_{T_{ks}}}(c)$
$d = \mathrm{H(RW, W, OID_i}, t)$
$s_1 = \mathrm{Sign_{T_{ks}}}(d)$
$T_1 = \mathrm{E_{OIDi_{kp}}}(s_1)$
$T_2 = \mathrm{E_{OIDi_{kp}}}(t)$
T_1, T_2
$(s_1) = \mathrm{D_{OIDi_{ks}}}(T_1)$
$t = \mathrm{D_{OIDi_{ks}}}(T_2)$
$\mathrm{Ver}(s_1, \mathrm{kp}) = d$
check $d? = \mathrm{H(RW, W, OID_i}, t)$
*keep t, s_1 secret for later use
*destroy RW, W, s_1

Fig. 1. Time-stamping protocol for relational database watermarking.

5. Limited distortion: The amount of introduced distortion should not decline the usability of the database relation. The statistical data metrics like the change in “mean” and “variance” can be measured and maintained to the minimum.
6. Incremental updatability: Where each tuple is marked independently of all other tuples.
7. Reversibility: The original non-watermarked database relation can be fully recovered after the watermark has been detected and authenticated.
8. Low false positives rates: The probability of detecting a watermark from non-watermarked relation should be negligible.
9. Randomness: The watermark information should be spread to the whole relation in order to resist any localized attacks.

3.4. Malicious attacks

The following malicious attacks should be considered in the proposed watermarking scheme:

1. Tuple alteration attack (bit-flipping attack): attacker tries to alter some values randomly.
2. Tuple deletion attack: attacker tries to delete some tuples from the watermarked relation aiming to weaken the embedded watermark.
3. Tuple insertion attack (mix-and-match attack): attacker tries to replace some tuples from other sources.
4. Attribute attack: attacker may add, delete, or modify an attribute in order to delete some parts of the watermark.
5. Tuple sorting attack: attacker rearranges the tuples based on one or more attributes hoping to weaken the embedded watermark.
6. Additive attacks: attacker inserts additional watermarks to watermarked data so as to confuse ownership proof.
7. Invertibility attacks: attacker suddenly discovers a fake watermark from the watermarked relation by guessing some values of the secret key $\mathcal{K}$.

3.5. Notation and parameters

Consider a database relation R and its schema has the form R(P,A_0,A_1,...,$A_{\alpha-1}$), where P is the primary key, A_0,A_1,...,$A_{\alpha-1}$ are the α attributes of R, where $A_i \in \mathbb{R}$ and η is the number of tuples in R. Any of the A_0,A_1,...,$A_{\alpha-1}$ attributes is a candidate for marking. To minimize the amount of distortion that will be introduced to the original data, watermark bits will be embedded in the fraction portion of the numeric attributes. Gap γ is a control parameter that determines the fractions of tuples marked. The bigger value of γ, the lower embedding rates, and the lower change introduced to relation data, also the lower robustness, and vice versa. Table 1 summarizes the important parameters used in our algorithms.

In our watermarking algorithms, we use a one-way hash cryptographic function result determined by the primary key and the embedding key $\mathcal{K}$ to determine which tuples and attributes to mark. Usually it has the form of $\langle = \mathcal{H}(M)$, where M is the message. Besides, it bears the following characteristics: given M, it is easy to compute $\langle$; yet when given $\langle$, it is hard to compute M, such as $\mathcal{H}(M) = \langle$; given M, it is hard to find another message M′ such as $\mathcal{H}(M) = \mathcal{H}(M')$. Several hash functions such as MD5 and SHA are good choices for this purpose (Schneier, 1996). We use $\mathcal{F}(t_i \cdot \mathcal{P}) = \mathcal{H}(\mathcal{K}||t_i \cdot \mathcal{P})$ as a message authentication code (MAC) considered to be secured (Schneier, 1996), where $t_i \cdot \mathcal{P}$ is the value of the primary key attribute P of tuple t_i in relation R. For security reason, the embedding key $\mathcal{K}$ should be selected from a large enough key space so that it is computationally infeasible for an attacker to guess the key. We suggest the embedding key be chosen as the following:

Table 1
Notations used in this paper.

η	Number of tuples in the relation
R	Database relation to be watermarked
RW	Watermarked relation
W	Watermark bits to be embedded
A_i	Attribute i in relation R
t_i	Tuple i in relation R
α	Number of attributes in relation R
$1/\gamma$	Fraction of tuples marked
τ	The detection parameter
ω	Number of tuples marked
$\mathcal{K}$	The embedding secret key
L	The length of watermark

$$\mathcal{K} = \mathcal{H}(\text{OID}||\text{DBname}||\text{version}||\ldots) \tag{16}$$

where OID is the database owner identity, || denotes concatenation, DB name is the database name, version is the database version, and $\mathcal{H}()$ is a cryptographic hash function (Schneier, 1996).For the watermark bits W to be embedded into the database relation R, W should be generated in a secure and random way to thwart several attacks. Eq. (17) shows the components that constitute W:

$$\text{W} = \mathcal{H}(\mathcal{K}||OID||\text{DB name}||\text{DB specific information}||\ldots) \tag{17}$$

As mentioned before, $\mathcal{K}$ is the embedding key, OID is the database owner identity, DB specific information is any specific characteristic information related to the current relation being watermarked such as the number of tuples, and $\mathcal{H}()$ is a cryptographic hash function (Schneier, 1996). The main purpose of this process is to make it difficult for the attacker to guess the embedded watermark W and to thwart the invertibility attacks. On the other hand, the watermark length L should be big enough to reduce the probability of guessing the embedded watermark W by the attacker; in the robustness analysis section a detailed discussion will be given.

Sorting operation is not required in our scheme since tuples in a database relation will be hashed based on their primary keys concatenated with the embedding key to a specific index to the watermark vector. This property ensures the incremental updatability can be satisfied.

3.6. Watermark Insertion

In this section, we give details of the watermark insertion process. We list five basic methods used in the watermark insertion algorithm.

1. Get_int(): used to extract the integral portion of a real number.
2. Get_frac(): used to retrieve the fractional portion of a real number.
3. Bin(): used to convert a value to the equivalent binary value.
4. Lsb(): used to return the least significant bit of a value.
5. Gen_watermark(): used to generate the user-specific watermark as shown by Eq. (17).

Algorithm 1 describes the process of encoding one watermark bit in an attribute. As mentioned in Section 2, prediction-error expansion (PE) assumes two intensities for each pixel, the original intensity y and the predicted one $y^\wedge$. Here we assume that y is the fractional portion of an attribute and $y^\wedge$ is any value known at encoding and decoding time that is related to that tuple, say $\mathcal{F}(t_i \cdot \mathcal{P})$. After subtracting y from $y^\wedge$ to obtain the difference, and based on the watermark bit to be embedded by expanding the obtained difference, we can achieve the encoding method.

Algorithm 1. Bit_encoding

```
Input: $\mathcal{F}(t_i \cdot \mathcal{P})$, watermark bit b, the value for attribute $j$ of
  tuple $i$: $t_i \cdot A_j$
Output: the updated value for attribute $j$ of tuple $i$
1   frac = Get_frac($t_i \cdot A_j$);
2   int = Get_int($t_i \cdot A_j$);
3   if (frac > 0) then
4   diff = frac $- \mathcal{F}(t_i \cdot \mathcal{P})$);
5   temp = Bin(diff);
6   temp = temp||b;
7   newdiff = To_integer(temp);
8   newval = newdiff + $\mathcal{F}(t_i \cdot \mathcal{P})$;
9   newval = To_number(int,newval);
10  Reflect_update($t_i \cdot A_j$,newval); //update attribute $t_i \cdot A_j$ by
    newval
11  end if;
12  end;
```

Algorithm 2 describes the process of decoding one watermark bit in an attribute. The Get_frac( ) method is used to retrieve the fractional portion. The new fractional portion will be processed by subtracting the fractional portion from $\mathcal{F}(t_i \cdot \mathcal{P})$ to calculate the expanded difference. The watermark bit i is the Lsb of the expanded difference. Now by using Eq. (13), we can obtain the original prediction error (difference). After that, using Eq. (14) we can recover the original value of y.

Algorithm 2. Bit_decoding

```
Input: $\mathcal{F}(t_i \cdot \mathcal{P})$, the value for attribute $j$ of tuple $i$: $t_i \cdot A_j$
Output: the decoded watermark bit b, the updated value for
  attribute $j$ of tuple $i$: $t_i \cdot A_j$
1  newfrac = Get_frac($t_i \cdot A_j$);
2  int = Get_int($t_i \cdot A_j$);
3  newdiff = newfrac $- \mathcal{F}(t_i \cdot \mathcal{P})$;
4  b = Lsb(Bin(newdiff));
5  diff = Floor(newdiff/2);
6  newval = newfrac − diff − b;
7  newval = To_number(int, newval);
8  Reflect_update($t_i \cdot A_j$, newval);    //update attribute $t_i \cdot A_j$
   by newval
9  end;
```

We extend the watermark insertion algorithm of AK scheme (Agrawal & Kiernan, 2002) to embed multiple watermark bits W, where a vector of watermark bits is embedded in the relation attributes, one bit for each tuple determined algorithmically under the control of the embedding key $\mathcal{K}$. The final watermark insertion algorithm is given in Algorithm 3.

Algorithm 3. Watermark insertion

```
// The embedding key $\mathcal{K}$ is known only to the owner of the
  database.
// The parameters $\gamma$ and $\alpha$ are also private to the owner
Input: Watermark W, Original Relation R, Embedding Key $\mathcal{K}$,
  fraction of tuples $1/\gamma$, candidate markable attributes $\alpha$
Output: Watermarked Relation RW
1   w[$w_0,\ldots,w_{L-1}$] = Gen_watermark(W);
2   for each tuple $t_i \in$ R
3   loop
4   if $\mathcal{F}(t_i \cdot \mathcal{P})$ mod $\gamma$ = 0 then // mark this tuple
5     attribute_index $j = \mathcal{F}F(t_i \cdot \mathcal{P})$ mod $\alpha$;    // mark
    attribute $A_j$
6   mark_bit idx = $\mathcal{F}(t_i \cdot \mathcal{P})$ mod $L$;
7   b = w[idx];
8   Bit_encoding($\mathcal{F}(t_i \cdot \mathcal{P}), b, t_i \cdot A_j$);
9   end if;
10  end loop;
11  end;
```

3.7. Watermark detection

There are two situations where the watermark detection can be invoked: (1) if Alice suspects that some data sets are illegally copied or tampered from her relation R, Alice or a third party can use watermark detection algorithm to verify the ownership of the suspicious database. (2) Alice uses a trial version of a database application and she wants to buy the full version of that database application.

In order to detect the watermark from the database relation R, we have to know the embedding key $\mathcal{K}$, γ, and α. The watermark detection algorithm deals with the same subset of tuples used in the watermark insertion because of the identical distribution of the hash function. When seeded by the same embedding key $\mathcal{K}$, then using the Bit_decoding algorithm, we can reconstruct the embedded watermark and recover the original database relation. After finishing the detection process, we will have several watermarks each belongs to a certain watermark bit index. A majority voting mechanism is applied to obtain the final watermark information. For each marked bit, we count the numbers of its value to be zeroes or ones respectively, and if the number of ones is higher than the detection τ then the final value of that bit is one; otherwise, it is zero. The detected result is a binary sequence of bits w′[$w_0,\ldots,w_{L-1}$], then we compare them with the original watermarks w[$w_0,\ldots,w_{L-1}$] and we count the matches among them, if all w′[$w_0,\ldots,w_{L-1}$] = w[$w_0,\ldots,w_{L-1}$] then Alice retrieves his watermark from the suspected relation R is recovered successfully. The detailed algorithm used for watermark detection is reported in Algorithm 4.

Algorithm 4. Watermark detection

```
Input: Watermarked relation RW, key $\mathcal{K}$, fraction of tuples $1/\gamma$,
  markable attributes $\alpha$
Output: Watermark Status $\in$ {true,false}, recovered relation R
1   OldRW = RW;    // backup the suspected relation
2   w[$w_0,\ldots,w_{L-1}$] = Gen_watermark(W);
3   for $i$ = 0 to $L - 1$ do
4   w′[$i$] = ";    // reset watermark information
5   count[$i$][0] = 0; count[$i$][1] = 0; // reset votes for $w_i$ to be
    0, 1, respectively
6   end for;
7   for each tuple $t_i \in$ RW
8   loop
9   if $\mathcal{F}(t_i \cdot \mathcal{P})$ mod $\gamma$ = 0 then    // this tuple was marked
10    attribute_index $j = \mathcal{F}(t_i \cdot \mathcal{P})$ mod $\alpha$;    // attribute $A_j$
    was marked
11    mark_bit $idx = \mathcal{F}(t_i \cdot \mathcal{P})$ mod $L$;
12    Bit_decoding($\mathcal{F}(t_i \cdot \mathcal{P})$, b, $t_i \cdot A_j$);
13    count(idx, b) = count(idx, b) + 1;
14  end if;
```

```
15 end loop;
16 for i = 0 to $L-1$ do      //Majority Voting Mechanism
17   if count[$i$][0] + count[$i$][1] = 0 then
18     w′[$i$] = −1;
19 end if;
20 if count[$i$][1]/count[$i$][1] + count[$i$][0] > $\tau$ then
21       w′[$i$] = 1
22     else w′[$i$] = 0;
23   end if;
24 end for;
25 for $i$ = 0 to $L-1$ do      // find matches between original
   and detected watermark
26   if w[$i$] = w′[$i$] then matchcount = matchcount + 1;
27     end if;
28 end for;
29 if matchcount = $L$ then
30     return true; // Alice retrieves his watermark from the
   suspected relation and R is recovered successfully
31 else
32 return false; // suspected relation
33     RW = OldRW; // R cannot be recovered
34 end if;
35 end;
```

4. Robustness analysis

In this section we present the detailed analysis of our proposed watermarking method. Our proposed method statistically follows AK scheme (Agrawal & Kiernan, 2002) since the same number $\omega \approx \eta/\gamma$ of attribute values is modified during the watermark insertion, so the number of introduced errors is the same, the only difference is that every watermark bit w_i is embedded $\omega \approx \eta/(\gamma \cdot L)$ times. We analyze the robustness of our scheme using the same method (i.e., binomial probability) as it was used in AK scheme (Agrawal & Kiernan, 2002). We analyze the robustness of our scheme under a range of malicious attacks mentioned in Section 3.3. A similar work can be found in Li et al. (2008) where the authors extended AK scheme to embed a multiple watermark bits in an irreversible scheme, the reader is referred to Li et al. (2008) for more analysis. We use the following robustness measure probabilities to analyze our watermarking method:

1. False hit rate: where a valid watermark is detected from non-watermarked relation.
2. False miss rate: where no valid watermark is detected from watermarked relation in the presence of various types of attacks.
3. The smaller values of these probabilities, the more robust the watermarking method.

4.1. Cumulative binomial probability

We use Bernoulli trials in our robustness analysis. Repeated independent trails are called Bernoulli trials if there are only two possible outcomes for each trial and their probabilities remain the same throughout the trials. Let $b(k; n, p)$ be the probability of obtaining k successes out of n Bernoulli trials each with probabilities p for success and $q = 1 - p$ for failure result in k successes and $n - k$ failures. Then,

$$b(k; n, p) = \binom{n}{k} p^k q^{n-k} \tag{18}$$

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{19}$$

Denote the number of successes in n trials as S_n. The probability of having at least k successes in n trials and the cumulative binomial probability can be written as:

$$P\{S_n \geqslant k\} = \sum_{i=k}^{n} b(i; n, p) \tag{20}$$

For brevity, define

$$B(k; n, p) := \sum_{i=k}^{n} b(i; n, p) \tag{21}$$

4.2. Detecting non-watermarked relations

It is obvious that watermark information can be suddenly detected from a non-watermarked database relation by an attacker even by cheer chance, or by guessing some information like the embedding key $\mathcal{K}$. This rarely happens, but a watermark detection algorithm should be designed to take this issue into consideration; otherwise, attacker can falsely claim the ownership of a relation.

4.2.1. False hit rate

It is the probability of detecting valid watermark from non-watermarked relation. The lower the false hit, the better robustness. We show that the false hit is under control in our scheme and can be made highly negligible. For a watermark bits $w[w_{0,\dots,w_{L-1}}]$, let w_i be extracted from data ω_i times ($\omega_i \approx \eta/(\gamma \cdot L) > 0$). After the watermark detection algorithm is applied to a non-watermarked relation, for each detected bit w_i, it has the same probability 0.50 to match or not to match the original bit in the watermark. The probability that at least τ portion out of ω_i bits can be detected from the non-watermarked data by cheer chance is $B(\tau\omega_i; \omega_i, 0.5)$. For a watermark of size L, the false hit will be equal to $\prod_{i=0}^{L-1} B(\tau\omega_i; \omega_i, 0.5)$ (Li et al., 2008).

Fig. 2 shows the change of the false hit when the watermark insertion parameter γ increases from 6 to 96 for fixed η = 10,000 and various values of the watermark detection parameter τ. The figure illustrates that the false hit is monotonic increasing with both watermark insertion parameter γ and detection parameter τ. On the one hand, the larger value of gap γ – which means lower fractions of tuples marked – the higher the false hit. On the other

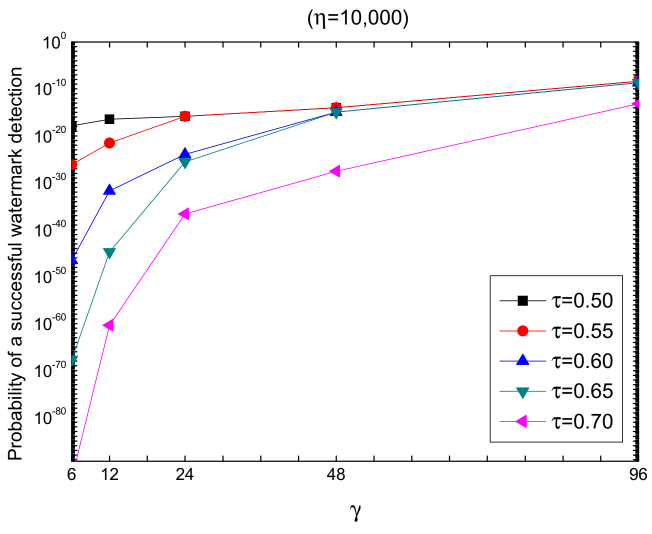

Fig. 2. False hit as a function of γ.

hand, the false hit can be decreased by increasing the detection parameter τ, which means the minimum threshold that is needed for the correctly detected watermarks is to be ascertained. We can see that the false hit rate is very low (less than 10^{-10}), even for high value of $\gamma = 96$ and low value of $\tau = 0.50$, which means that our method is robust, secure, and cannot be pirated.

Fig. 3 shows the change of the false hit when the number of tuples η increases from 2000 to 10,000 for fixed $\gamma = 6$ based on various values of the watermark detection parameter τ. The trend is that the false hit is monotonic decreasing with η. We observe that the bigger η, the lower false hit. However, the false hit can be decreased by increasing the detection parameter τ. The conclusion drawn from these two figures is that with small values of γ and reasonably high values of τ and η, the false hit can be made extremely low and even negligible.

4.3. Detecting watermarked relations

A watermarking scheme should be robust against malicious attacks, or benign database operations and updates that may destroy or affect the embedded watermark. The effect of those updates on the process of watermark detection should be limited to the minimum. It should be made hard for the attacker to modify the embedded watermark so that the innocent owner will not be in need to claim his ownership rights of a relation, nor will he/she be considered a pirate.

4.3.1. False miss rate

It is the probability of not detecting a valid watermark from a watermarked relation that has been modified in some benign updates or malicious attacks. We consider tuple alteration, tuple deletion, mix and match, and attribute deletion attacks for the false miss analysis.

4.3.2. Tuple alteration attacks

Sometimes this attack is called bit-flipping attack. In this form of attacks, Mallory tries to destroy the embedded watermark by altering randomly some bits and toggles their values (Agrawal & Kiernan, 2002). We assume that the attacker does not know the embedding key, so he/she does not know the positions where the watermark was embedded.

Fig. 4 shows the result of simulating this attack by randomly selecting different ratios of the watermarked relation and toggling the least significant bits (LSB) of all their attributes. We can see that the lower values of γ – the higher fraction of tuples watermarked – the higher resilience to this attack, even when up to 80% of the tuples are randomly altered (for $\gamma = 6$), the watermark can be detected completely. This shows that for the attacker to completely erase the embedded watermarks, he/she should alter more than 80% of the watermarked relation, causing a perceptible change to the pirated relation.

Due to the majority voting, watermark detection fails to detect watermark bit w_i only if at least $\omega_i/2$ embedded bits that correspond to w_i are toggled, where $\omega_i > 0$ is the number of times the watermark bit w_i is embedded in the data. Thus, the probability that the watermark bit is not recovered is $B(\frac{\omega_i}{2}; \omega_i, p_a)$, where p_a is the probability that a tuple is altered in the attack. Now the probability that the entire watermark is not recovered (i.e., the false-miss rate) is $1 - \prod_{i=0}^{L-1} 1 - B\frac{\omega_i}{2}; \omega_i, p_a$.

Fig. 5 shows the false miss in the case of random tuples alteration. For fixed ($\eta = 80{,}000$, $\tau = 0.50$), the lower value of γ, the harder it is for tuple alteration attacks to succeed. The general trend shown in this figure is that false miss is monotonic increasing with probability of alteration p_a, and with the lower value of γ, the false miss can be made extremely low.

Fig. 6 depicts the relation between γ and the detected watermark; for fixed ($\eta = 80{,}000$, $p_a = 0.50$), and variable detection parameter τ ($\tau = 0.50$–0.70), we can observe that the lower values of γ and τ, the higher watermark detection rates which means higher robustness against this attack.

4.3.3. Tuple deletion attacks

Sometimes this attack is called subset attack. In this form of attacks, Mallory tries to destroy the embedded watermark by selecting and deleting randomly some tuples from a watermarked relation (Agrawal & Kiernan, 2002). We also assume that the attacker does not know the embedding key, so he/she does not know the positions (tuples) where the watermark was embedded.

Fig. 7 shows the result of simulating this attack by randomly selecting and deleting different ratios of tuples from the watermarked relation. We can see that the lower the values of γ, the higher resilience to this attack. Even when up to 95% of the tuples are deleted (for $\gamma = 6$), the watermark can be detected completely. This shows that for the attacker to completely erase the embedded watermarks; he should delete more than 95% of the watermarked relation, which means that he will lose almost the whole relation. We conclude that this attack is not effective for the attacker.

Suppose that the attacker examines each tuple independently and selects it with probability p_d for inclusion from the pirated

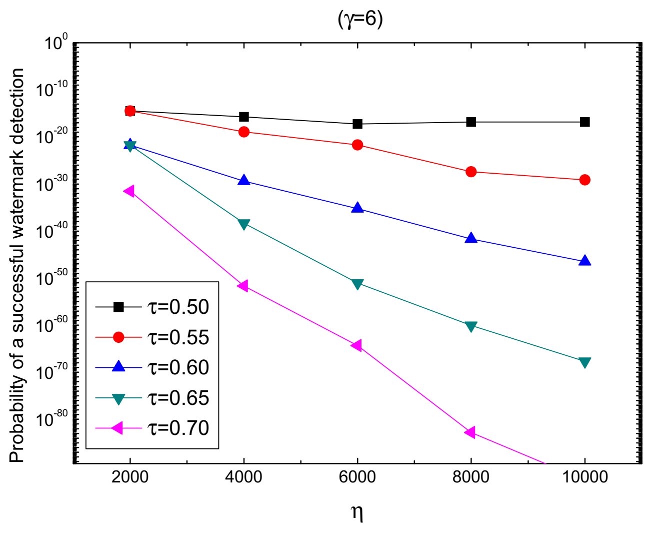

Fig. 3. False hit as a function of η.

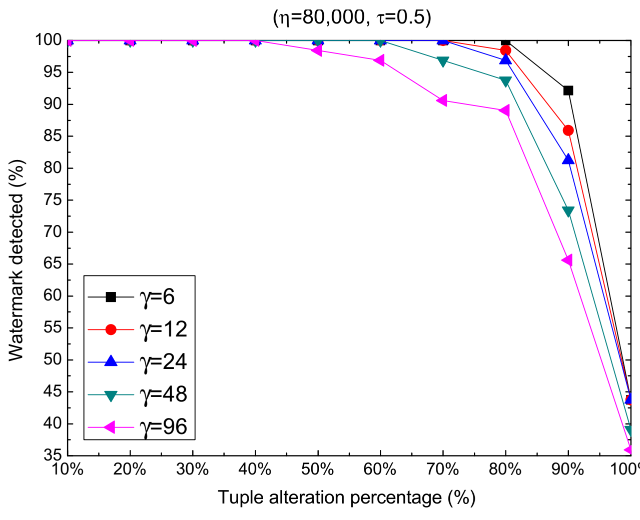

Fig. 4. Resilience to tuple alteration attack.

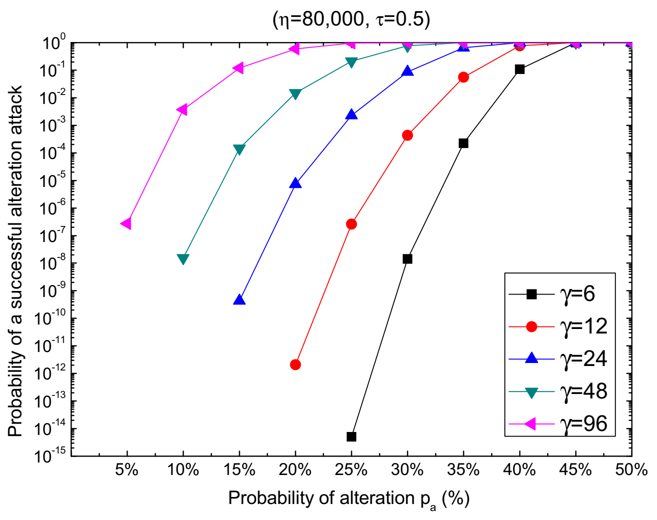

Fig. 5. False miss for tuple alteration attack as a function of p_a.

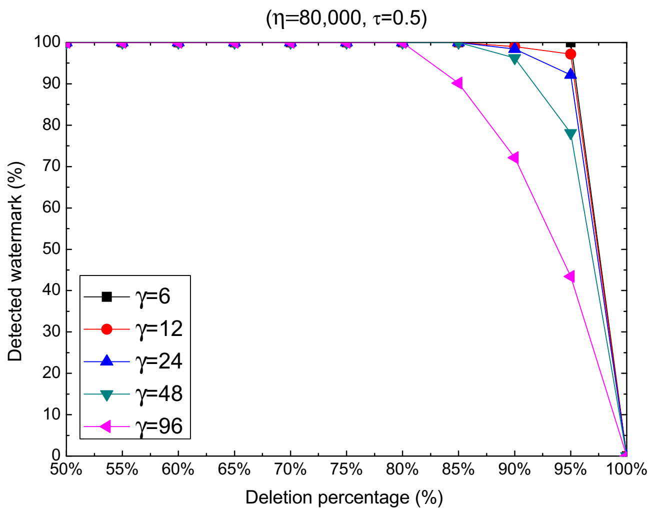

Fig. 7. Resilience to tuple deletion attack.

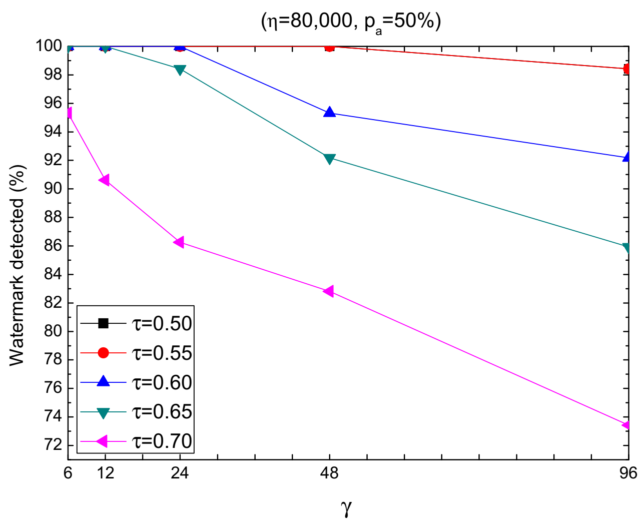

Fig. 6. Tuple alteration attack as a function of γ.

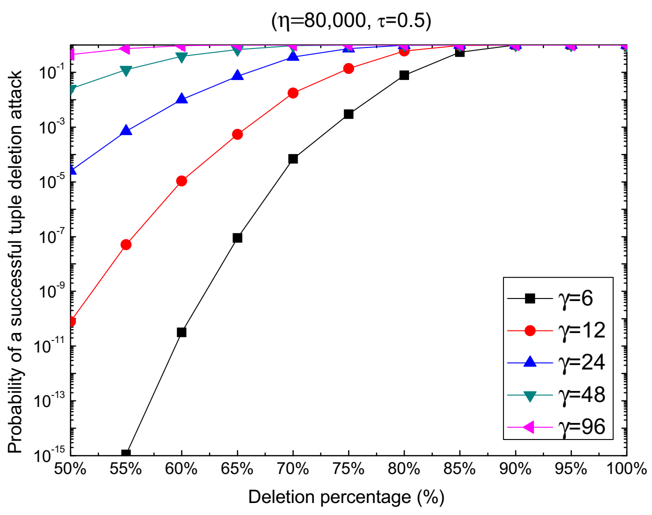

Fig. 8. False miss for tuple deletion attack.

relation. For the attack to be successful, it must delete all embedded bits for at least one watermark bit. We know that each watermark bit w_i is embedded ω_i times, so the probability that all the embedded bits for w_i are deleted is $B(\omega_i;\omega_i,p_d)=p_d^{\omega_i}$.

Therefore, for a watermark of size L, the false miss will be equal to $1-\prod_{i=0}^{L-1}(1-p_d^{\omega_i})$.

Fig. 8 shows the false miss in the case of random tuples deletion. For fixed (η = 80,000, τ = 0.50), the lower the value of γ, the harder it is for tuple deletion attacks to succeed. The general trend shown in this figure for this attack is similar to that shown in the previous Fig. 5 for tuple alteration. The lower values of γ have lower false miss (lower probability of success), which means more robustness against this attack. The figure also shows that even if 60% of watermarked tuples are deleted, the false miss is as low as 10^{-11} for γ = 6 when τ = 0.50. The false miss is close to one only if more than 95% of watermarked tuples are deleted. This finding supports the practical experiment in Fig. 7.

4.3.4. Tuple insertion attacks

Sometimes this attack is called mix-and-match attack (Agrawal & Kiernan, 2002). In the mix-and-match attack, Mallory takes the watermarked relation R and mixes it with η. p_m fraction of tuples from other sources to create his relation S of the same size as R, where η is the number of tuples in the original relation and $p_m > 0$ is the mixing rate.

Fig. 9 shows the result of simulating this attack, by randomly selecting different ratios of the watermarked relation and mixing them with tuples from other sources. We can see that for γ = 6 and p_m = 50% when τ = 0.50, the detection rate is about 98%. This is rational due to the majority voting mechanism being applied in the watermark detection algorithm. This shows that for the attacker to modify 55% of the embedded watermarks, he/she should mix more than 80% of the watermarked relation, causing a perceptible change to the relation.

In watermark detection, each watermark bit w_i is extracted from those additional tuples roughly ω_i. p_m times, where ω_i is the number of times the watermark is extracted from the original data and p_m is the probability of mixing where $p_m > 0$. Then the probability that this watermark bit is not recovered due to this attack is $B\left(\frac{\omega_i(1+p_m)}{2};\omega_i(1+p_m),p_m\right)=p_m^{\omega_i}$.Therefore, for a watermark of size L, the false miss will be equal to $1-\prod_{i=0}^{L-1}(1-p_m^{\omega_i})$.Fig. 10 shows the false miss in the case of mix-and-match. For fixed (η = 80,000, τ = 0.50), the lower value of γ, the harder it is to

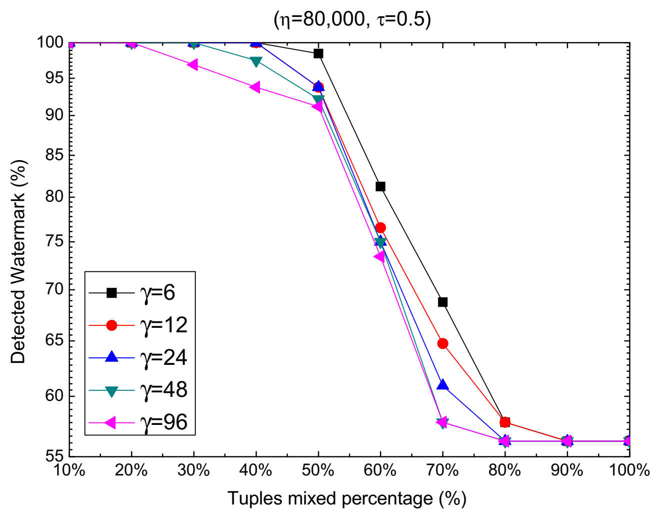

Fig. 9. Resilience to mix-and-match attack.

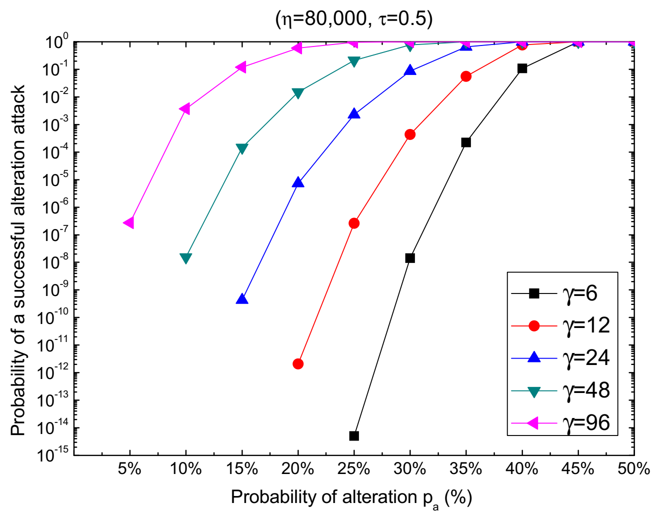

Fig. 10. False miss for mix-and-match attack.

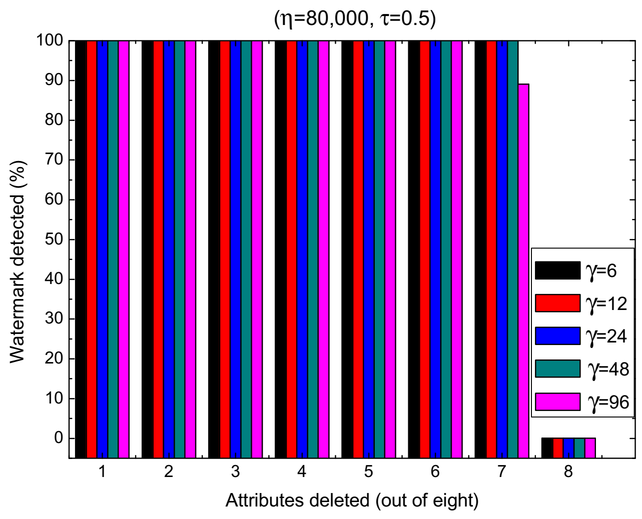

Fig. 11. Resilience to attribute deletion attack.

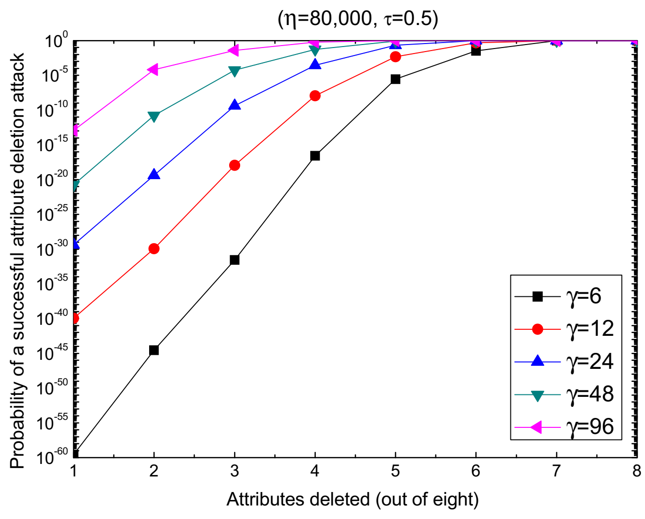

Fig. 12. False miss for attribute deletion attack.

succeed. The general trend shown in this figure is similar to that shown in the previous Fig. 5 for tuple alteration. Fig. 10 shows that even if 35% of watermarked tuples are mixed, and the false miss is as low as 10^{-4} for $\gamma = 6$ when $\tau = 0.50$. The false miss is close to one only if more than 45% of watermarked tuples are mixed.

4.3.5. Attribute attacks

Consider a situation where Mallory applies his attack to an attribute or more from a pirated relation. He may add, delete, and modify the contents of those attributes. We assume that the watermark detection algorithm can mount the relation attributes in a dynamic way at run time. Thus, the newly added attributes can be manipulated and take the effect at the watermark detection. In the case of adding new attribute(s) to the watermarked relation RW, the same effect of tuple insertion can be applied, so this follows the tuple insertion attack analysis. For the case of altering certain attribute(s), the same situation of tuple alteration can be considered. Finally, for the case of deleting certain attribute(s), tuple deletion analysis can be applied. To avoid the repetition, we present only attribute deletion attack in this context.

Now consider a situation where Mallory deletes an attribute or more from a pirated relation. Only those marked tuples in the deleted attributes will be affected. The same effect can be generalized when deleting tuples from a relation. Accordingly, the same false miss analysis can be drawn.

Fig. 11 shows the result of simulating this attack by gradually selecting and deleting different ratios of the attributes of the watermarked relation. We can see that our method has a high resilience to such an attack with complete watermark detection even when deleting seven out of eight attributes with variable values of $\gamma = (6–96)$ and $\tau = 0.50$.

Fig. 12 depicts the false miss for attribute deletion attack. For fixed ($\eta = 80{,}000$, $\tau = 0.50$), the lower value of γ, the harder it is for this attack to succeed. The general trend shown in this figure for this attack is similar to that shown in the previous Fig. 5 above for tuple alteration. Fig. 12 shows that even if five out of eight of attributes are deleted, the false miss is as low as 10^{-6} for $\gamma = 6$ when $\tau = 0.50$. The false miss is close to one only if seven attributes are deleted, and this result supports the practical experiment in Fig. 11.

4.3.6. Tuple sorting attack

The proposed watermarking method treats each tuple independently at the time of watermark insertion and detection. The pseudo hashed values of the primary key of each tuple maps to a designated watermark bit index used at both watermark insertion and detection without the need for sorting, and, as a result, this attack is not effective.

4.3.7. Additive attacks

In additive attacks, the attacker may insert another watermark to a relation before distribution, which results in confusing the detection algorithm. Later he/she will claim the ownership of the pirated relation. This kind of attacks is discussed in Agrawal and Kiernan (2002) and their proposed solution can be applied to our method. Our proposed watermarking scheme withstands this attack in two ways (1) by the setting of the proper values of the detection parameter τ and the gap γ and (2) by registering the RW using our proposed time-stamping protocol.

4.3.8. Invertibility attacks

An invertibility attack (Craver, Memon, Yeo, & Yeung, 1998) discovers a fictious secret key that extracts the embedded watermark information from a pirated relation. Later the attacker uses the discovered watermark to claim the ownership of the relation. This attack cannot proceed with setting the proper size of both the secret key $\mathcal{K}$ and the length of watermark L, by making use of Eqs. (16) and (17), it will make it harder to the attacker to work out the secret key $\mathcal{K}$. Given the probability of guessing the secret key is $\frac{1}{2^{|\mathcal{K}|}}$ and the probability of guessing the watermark is $\frac{1}{2^L}$, the bigger size of both $\mathcal{K}$ and L the harder for the attacker to succeed. On the other hand, signing the watermark W using the proposed time-stamping protocol will prevent the invertibility attack, since both the database owner OID_i, RW and W are time-stamped with the TSS.

5. Experiments

The experimental setup included a 2.4 GHz Pentium CPU and 1 GB RAM PC running Windows Vista Professional. Algorithms were implemented in Microsoft .NET environment using ADO component to visit Microsoft SQL Server database. We applied our algorithms to a generated synthetic data set of nine attributes, one is the primary key and the others are numerical attributes, and the eight attributes were used as candidate for watermarking. The size of the generated set was 80,000 tuples. The size of the watermark was $L = 64$ bit, and we investigated five embedding gaps $\gamma = 6$, 12, 24, 48 and 96 to embed the watermark bits into the data set. Throughout the experiments, several values of detection parameter were investigated $\tau = 0.50$, 0.55, 0.60, 0.65, and 0.70. For each conducted experiment, the test was repeated 100 times and the average of successfully detected watermark for each trial was calculated.

5.1. Imperceptibility

We report the effect of the errors introduced to the data after the watermark insertion process, and we used the mean and variance as statistical metrics to measure the quantitative change introduced to the data, in this experiment the value of γ varied from 6 to 96. We found a negligible change in the mean value for all the attributes. For this reason we omit it. Table 2 shows the result of this experiment after rounding the values to the nearest integer. Blank entries in the table indicate very little or no change.

As expected, greater changes in variance occur when gap $\gamma = 6$ and 12 because of larger perturbations in a greater fraction of tuples. The minuscule change in these statistics validates our assertion that our watermarking method is imperceptible.

5.2. Overhead

We report the computational cost results for watermark insertion and detection algorithms for several values of the gap γ in Table 3.

As we can see, in the worst case when $\gamma = 6$, where one tuple out of six is selected to be watermarked, it took about 32 s on average of 100 tests carried out. This time is quite small regarding the major benefits of the watermarking. On the other hand, the watermarking process is usually done offline, which will not affect the over whole performance of the database operations.

5.3. Tradeoffs

In our watermarking method, we have three tunable parameters: (i) γ, the gap which controls the fractions of tuples to be watermarked, (ii) τ, the detection parameter which controls the least fraction of watermark bits required for watermark detection and (iii) α, the number of attributes in the relation available for watermarking. The three parameters can control the balance between the overhead and the robustness. Based on the analysis reported in Sections 4 and 5, we summarize our findings in Table 4 as follows:

6. Conclusions and future works

In this paper, we have proposed a blind reversible relational database watermarking scheme. This scheme can prove the true ownership of the database's owner, and attains full recovery of the original database relation once the watermark information is detected and authenticated. We have extended the AK scheme to be applied for watermarking some sensitive applications such as medical and military systems. The watermarks are embedded into a database relation under the control of a secure embedding key. A

Table 2
Change in variance introduced by watermarking.

Attribute	Mean	Variance	$\gamma = 96$	$\gamma = 48$	$\gamma = 24$	$\gamma = 12$	$\gamma = 6$
A1	135	28,787				−1	−2
A2	300	141,878					+1
A3	121	2090				−1	−1
A4	54	478					+1
A5	157	31,006				+1	
A6	35	204					−1
A7	15	33					
A8	39	1202				−2	+2

Table 3
Time overhead cost.

	$\gamma = 6$	$\gamma = 12$	$\gamma = 24$	$\gamma = 48$	$\gamma = 96$
Insertion time (s)	32	27	20	16	10
Detection time (s)	17	14	11	9	5

Table 4
Tradeoffs.

Parameter	False hit (H)	False miss (M)	Data errors	Robustness	Time overhead
$\gamma \downarrow$	H ↓	M ↓	E ↑	↑	↑
$\tau \uparrow$	H ↓	M ↑	–	↑ M ↓ H	–
$\alpha \uparrow$	H ↓	M ↓	–	↑	↑

majority voting mechanism was applied to correct the watermark bits detected from the data at watermark detection phase.

An efficient and secure time-stamping protocol for digital watermarking of relational databases has been proposed. This protocol generates a timestamp and a specific signature for each distinct owner OID_i, watermark bits W, and a watermarked relation RW through a trusted-third party. Several identity and integrity attacks have been thwarted through this time-stamping protocol. The proposed protocol can be applied orthogonally with any watermarking scheme for other digital media types.

Based on the empirical and experimental security and robustness analysis against benign updates and malicious attacks, the results show that the proposed scheme of watermarking relational databases is secure, blind, and robust. The experiments show that the amount of errors introduced to the relation is minuscule and negligible. Our future research will be directed towards increasing the level of attack resilience against several sources of attacks in a typical blind reversible watermarking method, and proposing new techniques for watermarking database relations without primary keys.

References

Agrawal, R., & Kiernan, J., 2002. Watermarking relational databases. In *Proceedings of the 28th very large data bases VLDB conference, Hong Kong, China* (Vol. 28, pp. 155–166).

Alattar, A. M. (2004). Reversible watermark using the difference expansion of a generalized integer transform. *IEEE Transactions on Image Processing, 13*(8), 1147–1156.

Arnold, M. (2000). Audio watermarking: Features, applications and algorithms. In *Proceedings of the 5th IEEE international conference on computer and multimedia and expo* (pp. 1013–1016).

Atallah, M. J., Raskin, V., & Hempelmann, C. F. (2002). Natural language watermarking and tamperproofing. In *Proceedings of fifth international workshop on information hiding* (pp. 196–212).

Brassil, J. T., Low, S., & Maxemchuk, N. F. (1999). Copyright protection for the electronic distribution of text. *Proceedings of the IEEE, 87*(7), 1181–1196.

Celik, M. U., Sharma, G., Tekalp, A. M., & Saber, E. (2005). Lossless generalized-lsb data embedding. *IEEE Transactions on Image Processing, 14*(2), 253–266.

Chou, J., Chen, Y., & Chan, C. (2007). Cryptanalysis of Hwang-Chang's a time-stamp protocol for digital watermarking. <http://eprint.iacr.org/2007/004> Cryptology ePrint Archive Vol. 2007(001-050), Issue, 2007-01-01.

Collberg, C. S., & Thomborson, C. (2002). Watermarking, tamper-proofing, and obfuscation: Tools for software protection. *IEEE Transactions on Software Engineering, 28*(8), 735–746.

Cox, I. J., Doerr, G., & Furon, T. (2006). Watermarking is not cryptography. In *Proceedings of the 5th international workshop, IWDW 2006 Jeju Island, Korea, November 8–10, 2006* (Vol. 4283, pp. 1–15).

Cox, I. J., Kilian, J., Leighton, T., & Shamoon, T. G. (1997). Secure spread spectrum watermarking for multimedia. *IEEE Transaction Image Processing, 6*(12), 1673–1687.

Craver, S., Memon, N., Yeo, B. L., & Yeung, M. M. (1998). Resolving rightful ownerships with invisible watermarking techniques: Limitations, attacks, and implications. *IEEE Journal of Selected Areas in Communications, 16*(4), 573–586.

Gupta, G., & Pieprzyk, J. (2008). Reversible and blind database watermarking using difference expansion. In *Proceedings of the eForensics, January 2008* (pp. 1–6).

Hartung, F., & Girod, B. (1998). Watermarking of uncompressed and compressed video. *Signal Processing, 66*(3), 283–301.

Housley, R., Ford, W., Polk, W., & Solo, D. (1999). Internet x.509 public key infrastructure certificate and crl profile. RFC 2459. January 1999.

Hwang, M. S., Hwang, K. F., & Chang, C. C. (2005). A time-stamping protocol for digital watermarking. *Applied Mathematics and Computation, 169*(2), 1276–1284.

Li, Y., & Deng, R. H. (2003). Constructing a virtual primary key for fingerprinting relational data. In *Proceedings of the ACM digital rights management workshop (DRM)* (pp. 133–141).

Li, Y., Swarup, V., & Jajodia, S. (2003). A robust watermarking scheme for relational data. In *Proceedings of the 13th workshop on information technology and systems WITS* (pp. 195–200).

Li, Y., Guo, H., & Jajodia, S. (2004). Tamper detection and localization for categorical data using fragile watermarks. In *Proceedings of the 4th ACM workshop on digital rights management DRM'04* (pp. 73–82).

Li, Y., Guo, H., & Wang, S. (2008). A multiple-bits watermark for relational data. *Journal of Database Management, 19*(3), 1–21.

Petitcolas, F. A. P., Anderson, R. J., & Kuhn, M. G. (1999). Information hiding – A survey. *Proceedings of the IEEE, 87*(7), 1062–1078. special issue on protection of multimedia content.

Schneier, B. (1996). *Applied cryptography*. New York: John Wiley.

Shehab, M., Bertino, E., & Ghafoor, A. (2008). Watermarking relational databases using optimization based techniques. *IEEE Transactions on Knowledge and Data Engineering, 20*(1), 51–52.

Sion, R. (2004). Proving ownership over categorical data. In *Proceedings of the 20th IEEE international conference on data Engineering ICDE, April 2004* (pp. 584–596).

Sion, R., Atallah, M., & Prabhakar, S. (2004a). Resilient rights protection for sensor streams. In *Proceedings of the 30th international conference on very large data bases (VLDB)* (pp. 732–743).

Sion, R., Atallah, M., & Prabhakar, S. (2004b). Rights protection for relational data. *IEEE Transactions on Knowledge and Data Engineering, 16*(12), 1509–1525.

Standard specifications for Public Key Cryptography, IEEE standard. 1363-2000, 2000.

Thodi, D. M., & Rodriguez, J. J. (2004). Prediction-error-based reversible watermarking. In Proceedings of the IEEE conference on image processing, Singapore, October 2004 (pp. 1549–1552).

Tian, J. (2003). Reversible data embedding using a difference expansion. *IEEE Transactions on Circuits and Systems for Video Technology, 13*(8), 890–896.

Vleeschouwer, C. D., Delaigle, J. E., & Macq, B. (2001). Circular interpretation of histogram for reversible watermarking. In *Proceedings of the IEEE 4th workshop on multimedia signal processing, France, October 2001* (pp. 345–350).

Wong, P. W., & Memon, N. (2001). Secret and public key image watermarking schemes for image authentication and ownership verification. *IEEE Transactions on Image Processing, 10*, 1593–1601.