3-2006

Publicly Verifiable Ownership Protection for Relational Databases

Yingjiu LI
Singapore Management University, yjli@smu.edu.sg

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

DOI: https://doi.org/10.1145/1128817.1128832

Publicly Verifiable Ownership Protection for Relational Databases

Yingjiu Li
School of Information Systems
Singapore Management University
80 Stamford Road, Singapore 178902
yjli@smu.edu.sg

Robert Huijie Deng
School of Information Systems
Singapore Management University
80 Stamford Road, Singapore 178902
robertdeng@smu.edu.sg

ABSTRACT

Today, watermarking techniques have been extended from the multimedia context to relational databases so as to protect the ownership of data even after the data are published or distributed. However, all existing watermarking schemes for relational databases are *secret key based*, thus require a secret key to be presented in proof of ownership. This means that the ownership can only be proven once to the public (e.g., to the court). After that, the secret key is known to the public and the embedded watermark can be easily destroyed by malicious users. Moreover, most of the existing techniques introduce distortions to the underlying data in the watermarking process, either by modifying least significant bits or exchanging categorical values. The distortions inevitably reduce the value of the data. In this paper, we propose a watermarking scheme by which the ownership of data can be publicly proven by anyone, as many times as necessary. The proposed scheme is distortion-free, thus suitable for watermarking any type of data without fear of error constraints. The proposed scheme is robust against typical database attacks including tuple/attribute insertion/deletion, random/selective value modification, data frame-up, and additive attacks.

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems—*relational databases*

Keywords

Relational database, ownership protection, public verifiability, watermark, certificate

1. INTRODUCTION

Ownership protection of digital products after dissemination has long been a concern due to the high value of these assets and the low cost of copying them (i.e., piracy problem). With the fast development of information technology, an increasing number of digital products are distributed through the internet. The piracy problem has become one of the most devastating threats to networking systems and electronic business. In recent years, realizing that "the law does not now provide sufficient protection to the comprehensive and commercially and publicly useful databases that are at the heart of the information economy" [12], people have joined together to fight against theft and misuse of databases published online (e.g., parametric specifications, surveys, and life sciences data) [32, 4].

To address this concern and to fight against data piracy, watermarking techniques have been introduced, first in the multimedia context and now in relational database literature, so that the ownership of the data can be asserted based on the detection of watermark. The use of watermark should not affect the usefulness of data, and it must be difficult for a pirate to invalidate watermark detection without rendering the data much less useful. Watermarking thus deters illegal copying by providing a means for establishing the original ownership of a redistributed copy [1].

In recent years, researchers have developed a variety of watermarking techniques for protecting the ownership of relational databases [1, 28, 26, 29, 13, 19, 20, 2] (see Section 5 for more on related work). One common feature of these techniques is that they are secret key based, where ownership is proven through the knowledge of a secret key that is used for both watermark insertion and detection. Another common feature is that distortions are introduced to the underlying data in the process of watermarking. Most techniques modify numerical attributes [1, 28, 29, 13, 19, 20], while others swap categorical values [26, 2]. The distortions are made such that the usability of data for certain applications is not affected and that watermark detection can be performed even in the presence of attacks such as value modification and tuple selection.

The above two features may severely affect the application of watermarking techniques for relational databases. First, the secret key based approach is not suitable for proving ownership to the public (e.g., in a court). To prove ownership of suspicious data, the owner has to reveal his secret key to the public for watermark detection. After being used one time, the key is no longer secret. With access to the key, a pirate can invalidate watermark detection by either removing watermarks from protected data or adding a false watermark to non-watermarked data.

Second, the distortions that are introduced in the process of watermarking may affect the usefulness of data. Even though certain kind of error constraints (e.g., means and variances of watermarked attributes) can be enforced prior to or during the watermarking process, it is difficult or even impossible to quantify all possible constraints, which may include domain constraint, uniqueness constraint, referential integrity constraint, functional dependencies, semantic integrity constraint, association, correlation, cardinality constraint, the frequencies of attribute values, and statisti-

ASIACCS'06, March 21-24, 2006, Taipei, Taiwan.

cal distributes. In addition, any change to categorical data may be considered to be significant. Another difficulty is that the distortions introduced by watermarking cannot be reduced arbitrarily. A tradeoff has to be made between watermark distortions and the robustness of watermark detection (roughly speaking, the more distortions introduced in the watermarking process, the more likely that a watermark can be detected in the presence of database attacks).

In this paper, we attempt to design a new database watermarking scheme that can be used for publicly verifiable ownership protection and that introduces no distortions. Our research was motivated in part by certain aspects of public key watermarking schemes in the multimedia context, yet it is fundamentally different and particularly customized for relational databases (see also Section 5 for related work). Our scheme has the following unique properties. First, our scheme is publicly verifiable. Watermark detection and ownership proof can be effectively performed publicly by anyone as many times as necessary. Second, our scheme introduces no errors to the underlying data (i.e., it is distortion-free); it can be used for watermarking any type of data including integer numeric, real numeric, character, and Boolean, without fear of any error constraints. Third, our scheme is efficient for incremental updating of data. It is designed to facilitate typical database operations such as tuple insertion, deletion, and value modification. Fourth, our scheme is robust. It is difficult to invalidate watermark detection and ownership proof through typical database attacks and other attacks. With these properties, we believe that our watermarking technique can be applied practically in the real world for the protection of ownership of published or distributed databases.

The rest of the paper is organized as follows. Section 2 presents our watermarking scheme, which includes watermark generation and detection. Section 3 studies how to prove ownership publicly using a watermark certificate. It also investigates certificate revocation and incremental update in our scheme. Section 4 analyzes the robustness of our scheme and the tradeoff between its robustness and overhead. Section 5 comments on related work, and section 6 concludes the paper.

2. THE SCHEME

Our scheme watermarks a database relation R whose schema is $R(P, A_0, \ldots, A_{\nu-1})$, where P is a primary key attribute (later we discuss extensions for watermarking a relation that does not have a primary key attribute). There is no constraint on the types of attributes used for watermarking; the attributes can be integer numeric, real numeric, character, Boolean, or any other types. Attributes are represented by bit strings in computer systems. Let η denote the number of tuples in relation R. For each attribute of a tuple, *the most significant bit* (MSB) of its standard binary representation may be used in the generation of a watermark. It is assumed that any change to an MSB would introduce intolerable error to the underlying data value. For ease of referencing, Table 1 lists the symbols that will be used in this paper.

2.1 Watermark Generation

Let the owner of relation R possess a *watermark key* K, which will be used in both watermark generation and detection. The watermark key should be capable of publicly proving ownership as many times as necessary. This is contrast to traditional watermarking, where a watermark key is kept secret so that the database owner can prove his ownership by revealing the key for detecting the watermark. However, under that formation, the ownership can be publicly proved only once. In addition, the key should be long enough to thwart brute force guessing attacks to the key.

Algorithm 1 $genW(R, K, \gamma)$ // Generating watermark W for DB relation R

```
1: for each tuple r in R do
2:    construct a tuple t in W with the same primary key t.P = r.P
3:    for i=0; i < γ; i= i+1 do
4:       j = G_i(K, r.P) mod (the number of attributes in r)
5:       t.W_i = MSB of the j-th attribute in r
6:       delete the j-th attribute from r
7:    end for
8: end for
9: return W
```

In our scheme, the watermark key is public and may take any value (numerical, binary, or categorical) selected by the owner. There is no constraint on the formation of the key. To reduce unnecessary confusion, the watermark key should be unique to the owner with respect to the watermarked relation. We suggest the watermark key be chosen as

$$K = h(ID|DB\_name|version|...) \quad (1)$$

where ID is the owner's identity, '|' indicates concatenation, and $h()$ is a cryptographic hash function (e.g., SHA-512) [22]. In the case of multiple owners, the public key can be extended to be a combination of all the owners' IDs or generated from them using a threshold scheme. For simplicity, we assume that there is a single owner of DB relation R in the following.

Our concept of public watermark key is different from that of a public key in public key infrastructure (PKI) [16]. In the cryptography literature, a public key is paired with a private key such that a message encoded with one key can be decoded with its paired key; the key pair is selected in a specific way such that it is computationally infeasible to infer a private key from the corresponding public key. In our watermarking scheme, there is no private key, and the public watermark key can be arbitrarily selected. If the watermark key is derived from the owner's ID as suggested, it is similar to the public key in identity based cryptography [25, 3, 5], though the owner does not need to request a private key from a key distribution center (KDC).

The watermark key is used to decide the composition of a public *watermark* W. The watermark W is a database relation whose scheme is $W(P, W_0, \ldots, W_{\gamma-1})$, where $W_0, \ldots, W_{\gamma-1}$ are binary attributes. Compared to DB relation R, the watermark (relation) W has the same number η of tuples and the same primary key attribute P. The number γ of binary attributes in W is a control parameter that determines the number ω of bits in W, where $\omega = \eta \cdot \gamma$ and $\gamma \leq \nu$. In particular, we call γ the *watermark generation parameter*.

Algorithm 1 gives the procedure $genW(R, K, \gamma)$ for generating the watermark W. In the algorithm, a cryptographic pseudorandom sequence generator (see chapter 16 in [24]) $\mathcal{G}$ is seeded with the concatenation of watermark key K and the primary key $r.P$ for each tuple r in R, generating a sequence of numbers $\{\mathcal{G}_i(K, r.P)\}$. The MSBs of selected values are used for generating the watermark. The whole process does not introduce any distortions to the original data. The use of MSBs is for thwarting potential attacks that modify the data. Since the watermark key K, the watermark W, and the algorithm $genW$ are publicly known, anyone can locate those MSBs in R that are used for generating W. However, an attacker cannot modify those MSBs without introducing intolerable errors to the data.

In the construction of watermark W, each tuple in relation R

R	database relation to be watermarked
η	number of tuples in relation R
ν	number of attributes in relation R
W	database watermark (relation) generated in watermarking
γ	(watermark generation parameter) number of binary attributes in watermark W
ω	number of bits in W; $\omega = \eta\gamma$
τ	(watermark detection parameter) least fraction of watermark bits required for watermark detection
K	watermark key

Table 1: Notation in watermarking

Algorithm 2 $detW(R', K, \gamma, W, \tau)$ // Detecting watermark for DB relation R'

```
1: match_count=0
2: total_coutn=0
3: for each tuple r in R' do
4:    get a tuple t in W with the same primary key t.P = r.P
5:    for i=0; i < γ; i= i+1 do
6:       total_count = total_count +1
7:       j = G_i(K, r.P) mod (the number of attributes in r)
8:       if t.W_i = MSB of the j-th attribute in r then
9:          match_count = match_count +1
10:      end if
11:      delete the j-th attribute from r
12:   end for
13: end for
14: if match_count/total_count > τ then
15:    return true
16: else
17:    return false
18: end if
```

contributes γ MSBs from different attributes that are pseudo-randomly selected based on the watermark key and the primary key of the tuple. It is impossible for an attacker to remove all of the watermark bits by deleting some but not all of the tuples and/or attributes from the watermarked data. The larger the watermark generation parameter γ, the more robust our scheme is against such deletion attacks.

2.2 Watermark Detection

Our watermark detection is designed to be performed publicly by anyone as many times as necessary. This is a notable difference compared from previous approaches, which are secret key based. In watermark detection, the public watermark key K and watermark W are needed to check a suspicious database relation R'. It is assumed that the primary key attribute has not been changed or else can be recovered. If the primary key cannot be relied on, one can turn to other attributes, as will be discussed in Section 2.4.

Algorithm 2 gives the procedure $detW(R', K, \gamma, W, \tau)$ for detecting watermark W from relation R', where γ is the watermark generation parameter used in watermark generation, and τ is the *watermark detection parameter* that is the least fraction of correctly detected watermark bits. Both parameters are used to control the assurance and robustness of watermark detection, as will be analyzed in Section 4. The watermark detection parameter τ is in the range of $[0.5, 1)$. To increase the robustness of watermark detection, we do not require that all detected MSBs in R' match the corresponding bits in W, but that the percentage of the matches is more than τ (i.e., match_count/total_count $> \tau$ in algorithm 2).

2.3 Randomized MSBs

Most modern computers can represent and process four primitive types of data besides memory addresses: integer numeric, real numeric, character, and Boolean. Regardless of its type, a data item is represented in computer systems as a bit string. The MSB of the bit string is the leftmost digit, which has the greatest weight. In a signed numeric format (integer or real), the MSB can be the sign bit, indicating whether the data item is negative or not[1]. If the sign bit is not chosen (or there is no sign bit), the MSB can be the high order bit (next to the sign bit; in floating point format, it is the leftmost bit of exponent). For character or Boolean data, any bit can be an MSB and we simply choose the leftmost one.

We assume that watermark bits generated from selected MSBs are randomly distributed; that is, each MSB has the same probability of 1/2 to be 1 or 0. This randomness is important in our robustness analysis (see Section 4). If this is not the case, then we randomize the MSBs by XOR'ing them with random *mask bits*. For the MSB of the j-th attribute of tuple r, the corresponding mask bit is the j-th bit of hash value $h(K|r.P)$ if $j \leq \ell$, where ℓ is the bit-length of hash output. In general, if $(k-1)\ell < j \leq k\ell$, the mask bit is the $(j-(k-1)\ell)$-th bit of hash value $h^k(K|r.P)$. Since the hash value is computed from the unique primary key, the mask bit is random; thus, the MSB after masking is random. The randomized MSBs are then used in watermark generation and detection in our scheme.

2.4 Discussion on Relations without Primary Keys

Most watermarking schemes (e.g., [1, 20, 26, 2]) for relational databases, including ours, depend critically on the primary key attribute in the watermarking process. In the case that there is no primary key attribute, or that the primary key attribute is destroyed in malicious attacks, one can turn to other attributes and construct a virtual primary key that will be used instead of the primary key in the watermarking process. The virtual primary key is constructed by combining the most significant bits of some selected attributes. The actual attributes that are used to construct the virtual primary key differ from tuple to tuple, and the selection of the attributes is based on a key that could be the watermark key in the context of this paper. The reader is referred to [19] for more details on the construction of a virtual primary key.

Since the virtual primary key is constructed from the MSBs of selected attributes, it is difficult to destroy the virtual primary key through value modification or attribute deletion. However, unlike a real primary key, the virtual primary key may not be unique for each tuple; consequently, there could be multiple tuples in both R and W sharing the same value of the primary key. In watermark detection, the exact mapping between pairs of these tuples needs

[1] In most commonly used storage formats, the sign bit is 1 for a negative number and 0 for a non-negative number.

to be recovered (see line 4 in algorithm 2). This can be done as follows. For each tuple $r \in R$ with primary key $r.P$, compute a tuple t the same way as in watermark generation, then choose a tuple $t' \in W$ such that t' is the most close (e.g., in terms of Hamming distance) to t among the multiple tuples in W that share the same primary key $r.P$. The number of tuples sharing the same primary key value (i.e., the severity of the duplicate problem) can be minimized, as shown in the above-mentioned work [19].

3. PUBLIC OWNERSHIP PROOF

We now investigate how to publicly prove ownership as many times as necessary. If the watermark key K is kept secret with the owner, the ownership proof can be done secretly; however, it can be done only once in public since the key has to be revealed to the public during this process.

The problem of public ownership proof was originally raised in the multimedia context [15] (see section 5 for details); it has not been studied in the literature of database watermarking. We note that the requirements for watermarking relational data are different from those for watermarking multimedia data. The former must be robust against typical database alterations or attacks such as tuple insertion, deletion, and value modification, while the latter should be robust against multimedia operations such as compression and transformation. An additional requirement for watermarking relational data is that a watermarked relation should be updated easily and incrementally.

Public ownership proof in our scheme is achieved by combining watermark detection with a certificate.

3.1 Watermark Certificate

DEFINITION 3.1. *A watermark certificate C of relation R is a tuple $\langle ID, K, h(W), h(R), T, \text{DB-CA}, Sig\rangle$, where ID is the identity of the owner of R, K is the owner's watermark key, W is the public watermark, T is the validity information, DB-CA is the trusted authority who signs the certificate by generating a signature Sig.*

Similar to the identity certificate [16] in PKI (or attribute certificate [10] in PMI), which strongly binds a public key (a set of attributes) to its owner with a validity period, the watermark certificate strongly binds a watermark key, a watermark, and a DB relation to its owner's ID with validity information. The validity information is a triple $T = \langle T_{origin}, T_{start}, T_{end}\rangle$ indicating the original time T_{origin} when the DB relation is first certified, the starting time T_{start}, and the ending time T_{end} of this certificate in the current binding. When the DB relation is certified for the first time, T_{origin} should be the same as T_{start}. Compared with the identity certificate or attribute certificate, the watermark certificate not only has a validity period defined by T_{start} and T_{end}, but also contains the original time T_{origin}. The original time will be useful in thwarting possible attacks that confuse ownership proof.

A comparison of the watermark certificate with the traditional identity certificate is illustrated in Figure 1. The two kinds of certificates share a similar structure except that the public key information in the identity certificate is replaced by the watermark key, watermark hash, and database hash in the watermark certificate. In traditional identity certificate, the subject's public key is paired with a private key known only to the subject. In the case of damage or loss of the private key (e.g., due to collision attacks), the identity certificate needs to be revoked before the expiration of the certificate. In the watermark certificate, since there is no private key associated with the public watermark key, it seems that there is no need

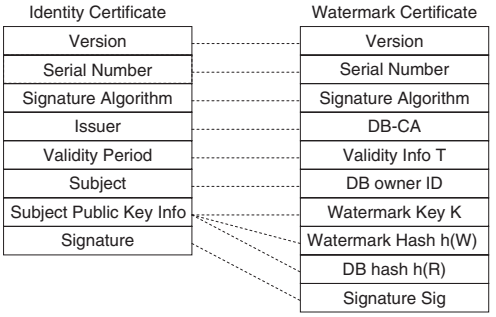

Figure 1: Relation between watermark and identity certificate

of certificate revocation. Nonetheless, certificate revocation and recertification may be needed in the case of identity change, ownership change, DB-CA signature compromise, and database update.

The role of DB-CA is similar to that of the traditional CA in PKI in terms of authentication of an applicant's identity. The differences are: (i) it binds the applicant's ID to the watermark key, watermark, and watermarked data; and (ii) it confirms the original time when the watermarked data was first certified. The original time is especially useful in the case of recertification so as to thwart false claims of ownership by a pirate. This is addressed in the following subsection.

3.2 Public Verifiability

While the watermark detection process can be performed by anyone, voluntarily or in delegation, who has access to the public watermark and watermark key, the ownership is proven by further checking the corresponding watermark certificate. This involves checking (i) if the watermark certificate has been revoked (see the next subsection for details); (ii) if the watermark key and (the hash of) the watermark used in watermark detection are the same as those listed in the watermark certificate; (iii) if the signature is correctly signed by the DB-CA stipulated in the watermark certificate (this is done in traditional PKI and may involve checking the DB-CA's public key certificate, a chain of CA's certificates, and a certificate revocation list); and (iv) the similarity of suspicious data R' to the original data R as published by the owner of watermark certificate. If all are proven, the ownership of the suspicious data is publicly claimed to belong to the owner of the watermark certificate for the time period stipulated in the certificate. The original time that the data was certified is also indicated in the certificate.

The last requirement is optional, depending on whether *data frame-up attack* is of concern. In a data frame-up attack, an attacker modifies the watermarked data as much as possible while leaving the watermarked bits (i.e., MSBs of selected values) untouched. Note that in our scheme, an attacker can pinpoint the watermarked bits since the watermark key, watermark, and watermark algorithm are all public. Since the ownership is publicly verifiable, such "frame-up" data may cause confusion and damage to the legitimate ownership.

The data frame-up attack has not been discussed before, even though it is also possible in secret key based schemes. For example, in Agrawal and Kiernan's watermarking scheme [1], the watermark information is embedded in one of ξ least significant bits of some selected values. Data frame-up attack is possible if an attacker modifies all significant bits except the last ξ least significant bits in each value. However, this attack is less serious in secret key

based schemes because the owner of watermarked data may choose not to claim the ownership for "frame-up" data. In our scheme, this attack is thwarted by requiring that the suspicious data is similar enough to the original data (the authenticity of the original data R can be checked with $h(R)$ in the watermark certificate).

The rationale is that when an attacker forges a low quality data R' with the MSBs given in the public watermark W, such R' will be significantly different from the original R due to its low quality. The similarity between R and R' may be measured, for example, by the portion of significant bits that match for each pair of values in R and R' whose watermarked MSBs match. The similarity may also be measured in terms of the usefulness of data, such as the difference of individual values, means, and variances.

3.3 Certificate Management

Once publicly proven based on a valid watermark certificate, the ownership of watermarked data is established for the owner of the certificate. The current ownership is valid for a time period $[T_{start}, T_{end}]$ stipulated in the certificate. The original time T_{origin} when the data was first certified is also indicated in the certificate.

The use of original time is to thwart *additive attack*. Additive attack is a common type of attacks to watermarking schemes in which an attacker simply generates another watermark for watermarked data so as to confuse ownership proof. The additional watermark can be generated using a watermark key that is derived from the attacker's ID. It is also possible for the attacker to obtain a valid watermark certificate for this additional watermark.

We solve this problem by comparing the original time T_{origin} in the certificate of real owner with the original time T'_{origin} in the certificate of the attacker. We assume that the owner of data will not make the data available to potential attackers unless the data is watermarked and a valid watermark certificate is obtained. Therefore, one always has $T_{origin} < T'_{origin}$ by which the legitimate ownership can be proven in the case of an ownership dispute. After this, the attacker's valid certificate should be officially revoked.

Besides revocation upon losing an ownership dispute, a certificate may be revoked before its expiration based on the following reasons: (1) identity change; (2) ownership change; (3) validity period change; (4) DB-CA compromise; and (5) database update. When the owner of a valid certificate changes his identity, he needs to revoke the certificate and, at the same time, apply for a new certificate to replace the old one. Upon the owner's request, the DB-CA will grant a new validity period $[T_{start}, T_{end}]$ according to its policy while keeping the original time T_{origin} unchanged in the new certificate. The case of ownership change is handled in a similar manner, except that the DB-CA needs to authenticate the new owner and ensure the ownership change is granted by the old owner. In both cases, a new watermark key and a new watermark may be derived and included in the new certificate.

Sometimes the owner wants to prolong or shorten the validity period of his certificate. In this case, the watermark certificate needs to be re-certified with a new validity period. The watermark key or watermark does not need to change in the recertification process.

In our scheme, the DB-CA is trusted, similar to the CA in traditional PKI. A traditional PKI certificate would need to be revoked for a variety of reasons, including key compromise and CA compromise. Since a watermark key is not paired with a private key in our scheme, there is no scenario of watermark key compromise. However, there is a possibility of DB-CA compromise if any of the following happens: (i) DB-CA's signature is no longer safe (e.g., due to advanced collision attacks); (ii) DB-CA loses its signature key; (iii) DB-CA ceases its operation or business; or (iv) any CA who certifies the DB-CA's public key is compromised (the public key is used to verify the DB-CA's signature in our scheme). In the case of DB-CA compromise, all related watermark certificates must be revoked and re-examined by a valid DB-CA and recertified with new validity periods but unchanged original times.

Due to the similarity between the watermark certificate and the traditional identity certificate, many existing standards and mechanisms regarding certificate management, such as certification path constraints and CRL distribution points, can be borrowed from PKI with appropriate adaptations. For simplicity and convenience, the functionality of a DB-CA may be performed by a CA in traditional PKI.

3.4 Efficient Revocation of Watermark Certificate

Micali proposed an efficient public key certificate revocation scheme [23] called CRS (for certificate revocation status). Compared with the CRL-based solution, CRS substantially reduces the cost of management of certificates in traditional PKI. This scheme can easily be adapted to our scheme for efficient revocation of watermark certificates.

As pointed out in [23], the costs of running a PKI are staggering and most of the costs are due to CRL transmission. The major reason is that each time a user queries the status of a single certificate, he needs to query a directory, an agent receiving certificate information from a CA and handling user queries about it, and the directory sends him the whole CRL list that has been most recently signed by the CA. Since the CRL list tends to be very long and transmitted very often, the CRL solution is extremely expensive. In CRS, however, the directory responds to a user's query by sending a 100-bit value only, instead of the whole CRL. The 100-bit value is employed by the user to verify whether the relative certificate is valid or has been revoked.

In our watermarking scheme, the DB-CA selects a secret 100-bit value Y_0 for a watermark certificate, and recursively applies on it a one-way function F 365 times, assuming that the validity period of the certificate is a normal year. The DB-CA then includes the 100-bit value $Y_{365} = F^{365}(Y_0)$ in the watermark certificate $C = \langle ID, K, h(W), h(R), T, \text{DB-CA}, Y_{365}, Sig\rangle$.

Assume that the current day is the i-th day in the validity period of the certificate. The DB-CA generates a 100-bit value $Y_{365-i} = F^{365-i}(Y_0)$ and gets it published through the directory. It is the DB owner's responsibility to obtain Y_{365-i} from the directory and publish it together with the watermark certificate C. Anyone can verify the validity of the certificate by checking whether $F^i(Y_{365-i}) = Y_{365}$, where i is the number of days since the start of the validity period (i.e., T_{start} in T). If this is the case, the certificate is valid; otherwise, it has been revoked before the i-th day, in which case the DB-CA did not get Y_{365-i} published. Note that Y_{365-i} cannot be computed from previously released Y_{365-j} ($j < i$) due to the one-way property of function F.

In this scheme, the DB owner needs to query the directory and update Y_{365-i} every day. To make the transition from Y_{365-i} to Y_{364-i} smooth, one more hour may be granted for the validity period of Y_{365-i} (i.e., 25 hours). To avoid high query load at certain hours, the validity period of Y_{365-i} should start at a different time each day for a different certificate. A policy stating this may also be included in the watermark certificate.

Note that Micali's original scheme requires a CA to (i) sign another 100-bit value besides Y_{365-i} to explicitly indicate a certificate being revoked; and (ii) sign a updated list indicating all and only the series numbers of issued and not-yet-expired certificates. The signed value and list are sent to the directory so that *any* user

query can be answered by the directory. In our scheme, it is the DB owner's responsibility (for his own benefit, namely anti-piracy) to query the directory and publish the updated Y_{365-i} online together with DB, watermark, and certificate. A user who wants to verify the certificate will obtain the validity information from the owner rather than from the directory. This separation of duty simplifies the scheme and clarifies the responsibility of the DB owner.

It is relatively straightforward to analyze the communication cost of our scheme as compared with the CRL based solution. The analysis is very similar to that given in [23] for comparing CRS with CRL (CRS is about 900 times cheaper than CRL in terms of communication cost with the Federal PKI estimates). We omit this analysis due to space limitations.

3.5 Incremental Updatability

The proposed scheme is also designed to facilitate incremental database update. In relational database systems, database update has been tailored to tuple operations (tuple insertion, deletion, and modification), where each tuple is uniquely identified by its primary key. In our scheme, both watermark generation and detection are tuple oriented; each tuple is processed independently of other tuples, based on its primary key.

The watermark is updated as follows. If a set of new tuples is inserted into the watermarked data, the watermark generation algorithm 1 can be performed on those new tuples only. As a result, a set of corresponding new tuples is generated and inserted into the watermark. If a set of tuples is deleted from the watermarked data, the corresponding tuples with the same primary keys are simply deleted from the watermark. In the case that a set of values is modified, only the related tuples need to be updated in the watermark. This can be done in a similar manner as in the tuple insertion case. Note that if a modified value does not contribute any MSB to the watermark, then no update is needed for that value.

The update of the watermark certificate follows the update of the watermark. To update a watermark certificate, the owner of watermarked data needs to authenticate himself to a DB-CA, revoke the old certificate, and get a new certificate for the updated DB and watermark. The new certificate may have an updated validity period, but the original time will not be altered. As this process involves interactions with a DB-CA, it may not be efficient if executed frequently. Fortunately, our scheme is very robust against database update, as will be indicated in the next section. Therefore, the update of the watermark and watermark certificate may lag behind the update of the watermarked data; it can be done periodically after a batch of data updates. The lag-behind watermark and certificate can still be used for checking the ownership of the updated data as long as the updates do not severely degrade the robustness of our scheme.

3.6 Discussion

Like traditional PKI, the certificate revocation in our scheme is handled only by the trusted party (i.e., the DB-CA). An alternative solution is to let the DB owner himself handle the certificate revocation. After the DB-CA signs a watermark certificate $C = \langle ID, K, h(W), h(R), T, DB-CA, Y_{365}, Sig\rangle$, where $Y_{365} = F^{365}(Y_0)$, it gives Y_0 to the DB owner through a secure channel. The DB owner keeps Y_0 secret. On the i-th day in the validity period of the certificate, the DB owner himself can generate and publish $Y_{365-i} = F^{365-i}(Y_0)$, based on which anyone can verify the validity of the certificate. This solution further simplifies our scheme in the sense that the DB-CA does not need to generate Y-values for all valid certificates each day, and that all DB owners do not need to query a directory to update the Y-values. The communication cost is thus reduced substantially. Whenever the DB owner deems it appropriate (e.g., after database is updated), he can refuse to release new Y-values to the public, thus revoking the certificate in a de facto manner, and apply a new certificate if necessary. This solution works well for database updates because it is to the benefit of the DB owner to maintain the certificate status. However, it may not work well in the case of DB-CA compromise or loss of Y_0, but this fortunately would not happen very often as compared with database updates. It is possible to develop a hybrid solution that combines the merits of both DB-owner-handled revocation and CA-handled revocation.

4. ROBUSTNESS AND OVERHEAD

For a watermarking scheme to be useful, it must be robust against typical attacks and be efficient in practice. In this section, we first present a quantitative model for the robustness of our watermarking scheme. We analyze the robustness of our scheme by the same method (i.e., binomial probability) as was used in [1]. We then investigate the overhead of our watermarking scheme. We also study the tradeoffs between the robustness and overhead in terms of the watermarking generation parameter γ and watermarking detection parameter τ.

4.1 Survival Binomial Probability

Consider n Bernoulli trials of an event, with probability p of success and $q = 1 - p$ of failure in any trial. Let $P_p(k; n)$ be the probability of obtaining exactly k successes out of n Bernoulli trials (i.e., the discrete probability of binomial distribution). Then

$$P_p(k;n) = \binom{n}{k} p^k q^{n-k} \quad (2)$$

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \quad (3)$$

Let $C_p(k; n)$ denote the probability of having more than k successes in n Bernoulli trials; that is, $C_p(k; n)$ is the survival binomial probability. According to the standard analysis of binomial distribution, we have

$$C_p(k;n) = \sum_{i=k+1}^{n} P_p(i;n) \quad (4)$$

In many widely available computation software packages such as Matlab and Mathematica, the survival binomial probability can be computed by $C_p(k; n) = 1 - binocdf(k, n, p)$, where $binocdf(k, n, p)$ is the binomial cumulative distribution function with parameters n and p at value k. When n is large, the binomial distribution can be approximated by a normal distribution with mean np, standard deviation $\sqrt{npq}$, at value $k + 0.5$, where 0.5 is the correction of continuity (for $p = 0.5$, the normal is a good approximation when n is as low as 10; see chapter 7.6 in [31]). Thus, $C_p(k; n) = 1 - normcdf(k + 0.5, np, \sqrt{npq})$, where $normcdf$ is the normal cumulative distribution function.

4.2 Detecting Non-Watermarked Data

First consider the robustness of our scheme in terms of *false hit*, which is the probability of a valid watermark being detected from non-watermarked data. The lower the false hit, the better the robustness. We show that the false hit is under control in our scheme and can be made highly improbable.

Recall that in watermark detection, a collection of MSBs are located in suspicious data and compared with the corresponding

bits recorded in the public watermark. When the watermark detection is applied to non-watermarked data, each MSB in data has the same probability $1/2$ to match or not to match the corresponding bit in the watermark. Assume that the non-watermarked data has the same number η of tuples (and the same primary keys) as the original data. Let $\omega = \eta\gamma$ be the total number of bits in the watermark, where γ is the watermark generation parameter. The false hit is the probability that at least τ portion of ω bits can be detected from the non-watermarked data by sheer chance, where τ is the watermark detection parameter. The false hit H can be written as

$$H = C_{1/2}(\lfloor \tau\omega \rfloor, \omega) = C_{1/2}(\lfloor \tau\gamma\eta \rfloor, \gamma\eta) \quad (5)$$

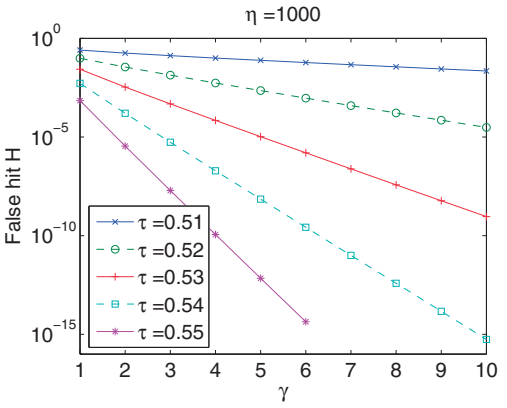

Figure 2: False hit as function of γ

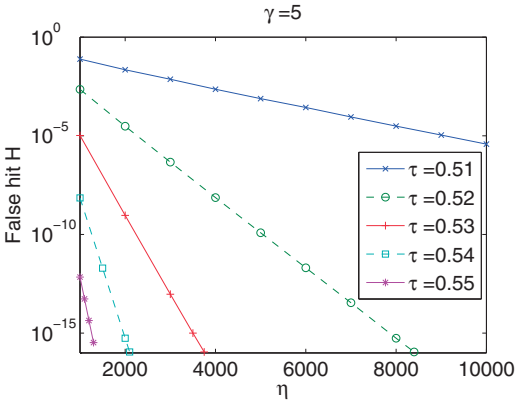

Figure 3: False hit as function of η

Figure 2 shows the change of the false hit when the watermark insertion parameter γ increases from 1 to 10 for fixed $\eta = 1000$ and various values of the watermark detection parameter τ. The figure illustrates that the false hit is monotonic decreasing with both watermark insertion parameter γ and detection parameter τ. On the one hand, the larger the insertion parameter γ, the more MSBs are included in the watermark and the smaller the false hit. On the other hand, the false hit can be decreased by increasing the detection parameter τ, which is the least fraction of watermark bits required for ownership assertion.

Figure 3 illustrates the trend of false hit when the number η of tuples is scaled up from 1000 to 10,000. The trend is that the false hit is monotonic decreasing with η. This trend is linear, which is similar to that of increasing γ, as indicated in figure 2. A conclusion drawn from these two figures is that with reasonably large values of γ, τ, and/or η, the false hit can be made extremely low.

4.3 Detecting Watermarked Data

We now consider the robustness of our scheme in terms of *false miss*, which is the probability of not detecting a valid watermark from watermarked data that has been modified in typical attacks. The robustness can also be measured in terms of the error introduced by typical attacks. The less the false miss, or the larger the error introduced by typical attacks, the better the robustness. The typical attacks include database update, selective value modification, and suppression. Other typical attacks include the data frame-up attack and the additive attack which have been addressed in a previous section.

4.3.1 Typical Database Update

Typical database update includes tuple insertion, tuple deletion, attribute deletion, and value modification. For tuple deletion and attribute deletion, the MSBs in the deleted tuples or attributes will not be detected in watermark detection; however, the MSBs in other tuples or attributes will not be affected. Therefore, all detected MSBs will match their counterparts in the public watermark, and the false miss is zero.

Though the deletion of tuples or attributes will not affect the *false miss*, it will make the *false hit* worse. The more the tuples or attributes are deleted, the larger the false hit, as indicated in Section 4.2. The effect to the false hit of deleting tuples is equivalent to that of decreasing η as shown in Figure 3, while the effect of deleting attributes is equivalent to decreasing γ proportionally as shown in Figure 2.

Since the watermark detection is primary key based, a newly inserted tuple should have a valid primary key value; otherwise, there is no corresponding tuple in the public watermark. We thus consider tuple insertion to be "mix-and-match" [1]; that is, an attacker inserts ξ new tuples to replace ξ watermarked tuples with their primary key values unchanged. For watermark detection to return a false answer, at least $\gamma\eta - \lfloor \tau\gamma\eta \rfloor$ MSBs in those newly added tuples (which consists of $\gamma\xi$ MSBs) must not match their counterparts in the public watermark (which consist of $\gamma\eta$ bits). Therefore, the false miss M_ξ for inserting ξ tuples in mix-and-match can be written as

$$M_\xi = C_{1/2}(\gamma\eta - \lfloor \tau\gamma\eta \rfloor - 1, \gamma\xi) \quad (6)$$

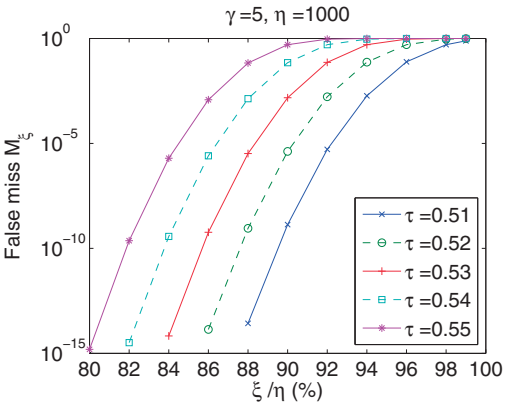

Figure 4: False miss (tuple insertion) as function of ξ

Figures 4, 5, and 6 show the false miss in the case of tuple insertion. The default parameters in these figures are $\xi/\eta = 90\%$ (i.e.,

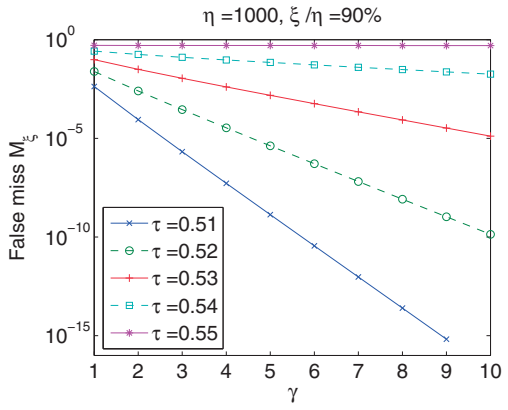

Figure 5: False miss (tuple insertion) as function of γ

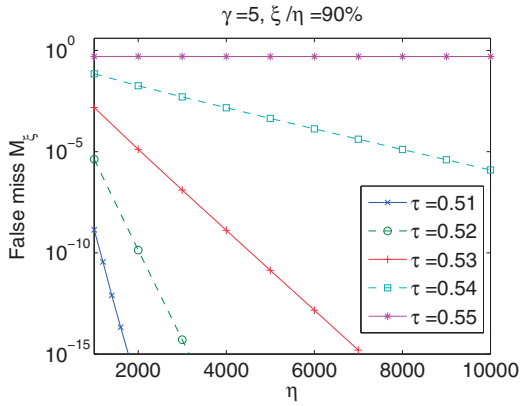

Figure 6: False miss (tuple insertion) as function of η

90% of the new tuples are inserted into the data to replace the watermarked tuples), $\gamma = 5$, and $\eta = 1000$. A general trend shown in these figures is that the false miss is monotonic increasing with watermark detection parameter τ. This trend is opposite to that of the false hit, which is monotonic decreasing with τ as indicated in Figures 2 and 3. Therefore, there is a tradeoff between false hit and false miss with respect to τ.

Figure 4 shows that even if 80% of watermarked tuples are replaced with new tuples, the false miss is as low as 10^{-15} for all τ values greater than or equal to 51%. The false miss is close to one only if more than 90% of watermarked tuples are replaced in this figure.

Figures 5 and 6 illustrate that the false miss is monotonic decreasing with γ and η, which is similar to the trend of false hit as indicated in Figures 2 and 3. With reasonably large γ and/or η, the false miss can be made extremely low.

For value modification, we assume that the modified values are randomly chosen. We leave the selective modification targeted on watermarked values to the next subsection. Recall that there are ν attributes in the original data in which γ attributes are watermarked for each tuple. When a random modification happens, it has probability γ/ν that a watermarked value is chosen. When a watermarked value is modified, its MSB has probability $1/2$ to change (i.e., the value is modified randomly). In watermark detection, a detected MSB has probability $\gamma/(2\nu)$ not to match its counterpart in the public watermark. The false miss M_ζ for randomly modifying ζ values can be written as

$$M_\zeta = C_{\gamma/2\nu}(\gamma\eta - \lfloor\tau\gamma\eta\rfloor - 1, \zeta) \tag{7}$$

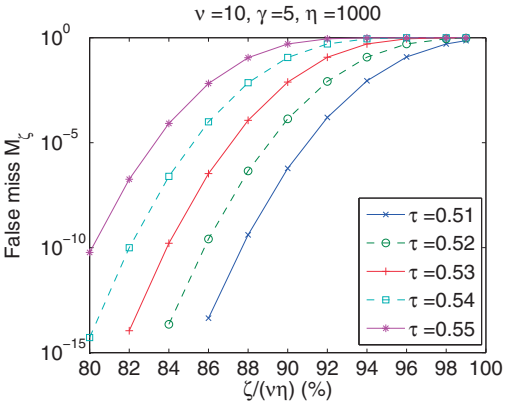

Figure 7: False miss (value modification) as function of ζ

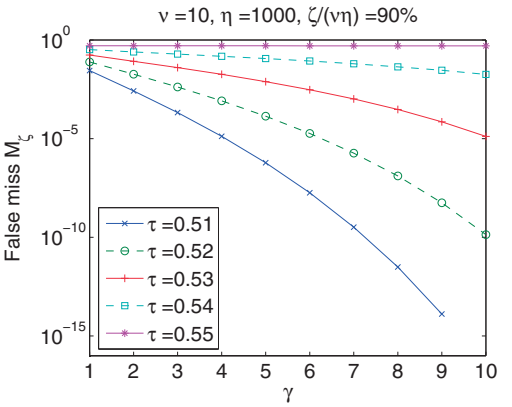

Figure 8: False miss (value modification) as function of γ

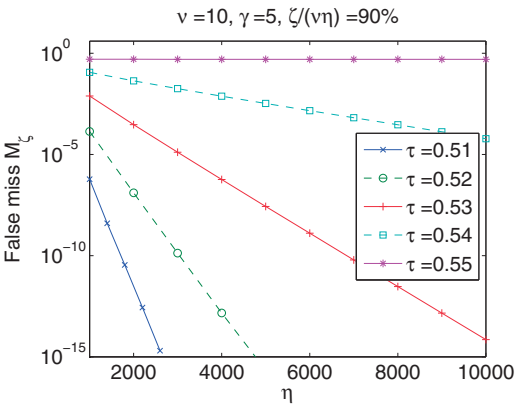

Figure 9: False miss (value modification) as function of η

Figures 7, 8, and 9 show the false miss in the case of random value modification. The default parameters in these figures are $\zeta/(\gamma\eta) = 90\%$ (i.e., 90% of the values are modified randomly), $\nu = 10$, $\gamma = 5$, and $\eta = 1000$. The general trend shown in these

figures for value modification is similar to that shown in previous Figures 4, 5, and 6 for tuple insertion. The difference in calculation is due to the use of probability $\gamma/2\nu$ in Equation 7 instead of probability $1/2$ in Equation 6. Figure 7 shows that even if 80% of values are modified randomly, which would make the data less useful, the false miss rate in detection is less than 10^{-10} in our computation.

4.3.2 *Selective Value Modification and Suppression*

Since both the watermark key and the watermark are public in our scheme, an attacker can pinpoint the MSBs of watermarked values. A simple attack would be to flip some of those MSBs so that the watermark detection will detect no match. Assuming that ς watermarked MSBs are flipped in selective value modification, the false miss M_ς can be written as

$$M_\varsigma = \begin{cases} 1 & \text{if } \varsigma \geq \gamma\eta - \lfloor\tau\gamma\eta\rfloor \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

If no less than $\gamma\eta - \lfloor\tau\gamma\eta\rfloor$ watermarked MSBs are flipped, the watermarked data will no longer be detected. The robustness of our scheme can then be measured in terms of the error introduced by this attack. The larger the error introduced for defeating the watermark detection (i.e., achieving $M_\varsigma = 1$), the better the robustness.

Recall that any change to an MSB would introduce intolerable error to the related data value. To defeat the watermark detection, no less than $\gamma\eta - \lfloor\tau\gamma\eta\rfloor$ MSBs have to be flipped; this would introduce intolerable errors to no less than $\gamma\eta - \lfloor\tau\gamma\eta\rfloor$ data values. We thus measure the robustness in terms of *failure error rate*, which is the least fraction F of total data values that need to be intolerably modified for defeating the watermark detection. This failure error rate can be written as

$$F = \frac{\gamma\eta - \lfloor\tau\gamma\eta\rfloor}{\eta\nu} \approx (1-\tau)\frac{\gamma}{\nu} \quad (9)$$

A larger failure error rate (or better robustness) can be achieved by increasing γ (watermark generation parameter) or decreasing τ (watermark detection parameter). There is a tradeoff between the robustness of our scheme and the size of the public watermark (which has γ binary attributes). To achieve the best robustness in terms of thwarting the selective modification attacks, one may choose $\gamma = \nu$ and $\tau \approx 0.5$. (However, this would increase the false hit as indicated in Section 4.2.) In this extreme case, approximately 50% of data values have to be intolerably modified so as to defeat the watermark detection.

To avoid the intolerable error, an attacker may choose to suppress some watermarked values rather than flipping their MSBs. Since this attack causes no mismatch in watermark detection, the *false miss* is zero. However, it will increase the *false hit* because those MSBs will be missed in watermark detection. It is easy to know that the effect of suppressing ς MSBs to the false hit is the equivalent of decreasing the total number of MSBs by ς in the computation of false hit. Thus, the false hit formula (see section 4.2) changes from $C_{1/2}(\lfloor\tau\gamma\eta\rfloor, \gamma\eta)$ to $C_{1/2}(\lfloor\tau(\gamma\eta-\varsigma)\rfloor, \gamma\eta-\varsigma)$ for selective suppression of ς watermarked values.

Figure 10 shows the influence of selective value suppression to the false hit for fixed $\gamma = 5$, $\eta = 1000$, and various τ from 0.51 to 0.55. In the figure, we change the rate $\varsigma/(\gamma\eta)$ (the percentage of watermarked bits are suppressed) from 0% to 99%. Even if the rate $\varsigma/(\gamma\eta)$ increases up to 50%, the false hit is still below 15.4% for $\tau = 0.51$, below 2.2% for $\tau = 0.52$, below 0.13% for $\tau = 0.53$, below $3 * 10^{-5}$ for $\tau = 0.54$, and below $2.6 * 10^{-7}$ for $\tau = 0.55$.

4.4 Overhead

We now analyze the time and space overhead for both watermark

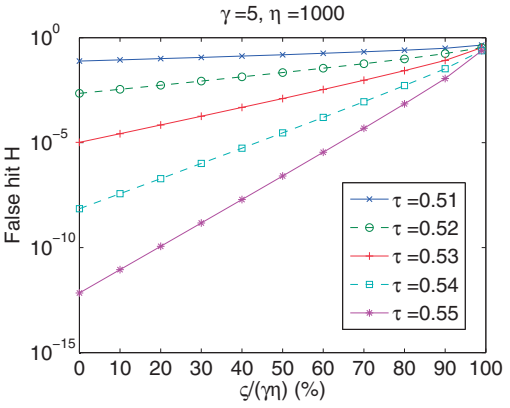

Figure 10: False hit (value suppression) as function of ς

generation and watermark detection. Throughout the analysis, we ignore the IO cost (i.e., reading and writing tuples). Table 2 describes the symbols that will be used in this section.

Consider watermark generation. For each of η tuples to be processed, a random sequence generator $\mathcal{G}$ is first seeded, then γ MSBs are determined based on γ random numbers generated by $\mathcal{G}$. The MSBs are assigned to the corresponding attributes in the public watermark. For each MSB to be determined, one mod operation is involved and one attribute is deleted from the copy of related tuple. The memory requirement for the process of a tuple is to keep the copy of the tuple, γ MSBs, and the watermark key in concatenation with the tuple's primary key. Therefore, the time overhead t_{genW} and space overhead m_{genW} for watermark generation are

$$\begin{aligned} t_{genW} &= \eta t_{seed} + \eta\gamma(t_{genS} + t_{mod} + t_{bit} + t_{delA}) \\ &= O(\eta\gamma) \quad (10) \\ m_{genW} &= m_{tuple} + \gamma + m_{wkey} = O(\gamma) \quad (11) \end{aligned}$$

In watermark detection, the time and space overheads are the same as in watermark generation except for the cost of processing the count information. Let t_{if} denote the cost of the last operation "if $match\_count/total\_count > \tau$." The time overhead t_{detW} and space overhead m_{detW} for watermark detection can be written as

$$\begin{aligned} t_{detW} &= 2t_{count} + \eta t_{seed} + \eta\gamma(t_{genS} + t_{mod} + t_{bit} + \\ &\quad t_{delA} + 2t_{count}) + t_{if} = O(\eta\gamma) \quad (12) \\ m_{detW} &= 2m_{count} + m_{tuple} + \gamma + m_{wkey} = O(\gamma) \quad (13) \end{aligned}$$

The generated watermark W will be stored on disk. The disk storage requirement m_{disk} is thus

$$m_{disk} = |W| = \eta m_{pkey} + \eta\gamma = O(\eta\gamma) \quad (14)$$

4.5 Tradeoffs

In our watermark scheme, we have two parameters: watermark generation parameter γ and watermark detection parameter τ. The two parameters can be used to balance between the robustness and the overhead of our scheme. Table 3 summarizes the tradeoffs that can be made when choosing the two parameters.

The watermark generation parameter γ is used to balance between robustness and overhead. The larger the γ, the better the robustness of our scheme and the worse the time and space overhead. While the watermark detection parameter τ has no effect on

Table 2: Symbols used in the analysis of overhead

t_{seed}	cost of seeding random sequence generator $\mathcal{S}$ with public key and a tuple's primary key
t_{genS}	cost of generating a random number from $\mathcal{S}$
t_{mod}	cost of mod operation
t_{delA}	cost of deleting an attribute from a copy of a tuple
t_{bit}	cost of assigning/comparing a bit value to/with the public watermark
t_{count}	cost of assigning/updating a count in watermark detection
m_{count}	number of bits required to store a count in watermark detection
m_{tuple}	number of bits required to store a copy of a tuple
m_{wkey}	number of bits to store a watermark key
m_{pkey}	number of bits to store a primary key value

Table 3: Tradeoffs

parameter	false hit	false miss	failure error rate	robustness (summary)	overhead (time)	overhead (space)
$\gamma \uparrow$	$H \downarrow$	$M \downarrow$	$F \uparrow$	$\uparrow$	$\uparrow$	$\uparrow$
$\tau \uparrow$	$H \downarrow$	$M \uparrow$	$F \downarrow$	$\uparrow$ in terms of H $\downarrow$ in terms of M, F	$--$	$--$

the overhead, it is used as a tradeoff between false hit, false miss, and failure error rate. Increasing τ will make the robustness better in terms of false hit, but worse in terms of false miss and failure error rate.

5. RELATED WORK

Watermarking has been extensively studied in the context of multimedia data for the purpose of ownership protection and authentication [7, 17, 18]. Most watermarking schemes proposed so far are *secret key based*, which require complete disclosure of the watermarking key in watermark verification. These watermarking schemes can be further classified as *private* (both the secret key and original data are required in watermark verification), *blind* (only the secret key is needed for watermark bit decoding), and *semi-blind* (it requires both the secret key and watermark bit sequence in watermark detection). Watermarking schemes can also be classified as being *robust* (the watermark is hardly destroyed in attacks), or *fragile* (the watermark is hardly untouched if the watermarked data is modified). The robust watermark may be used for ownership proof while the fragile watermark is suitable for data authentication and integrity check.

As database piracy increasingly becomes a serious problem, watermarking techniques have been extended to protect the ownership of published or distributed databases [1, 13, 28, 29, 26, 19, 20, 2]. Agrawal and Kiernan [1] first proposed a robust watermarking scheme for database relations. Their scheme modifies a collection of least significant bits of numerical attributes. The locations of those least significant bits, and the values to which those bits are modified, are all determined by a secret key. With the same secret key, those modified values can be localized in watermark detection, and ownership is claimed if a large portion of the detected values are as expected.

As noted by Agrawal and Kiernan [1], database relations differ from multimedia data in significant ways and hence require a different class of watermarking techniques. A major difference is that a database relation is composed of a set of tuples; each tuple represents an independent object which can be added, deleted, and modified frequently in either benign updates or malicious attacks. In contrast, a multimedia object consists of a large number of bits; portions of a multimedia object are bound together in fixed spatial or temporal order that cannot be arbitrarily changed. It is also noted that the frequency domain watermarking being used in the multimedia context is not suitable for watermarking relational data. The reason is that the error introduced in frequency domain will spread over all attribute values (i.e., the whole "image"), which may not be acceptable in certain database applications.

There have been other schemes proposed for watermarking relational data. In Sion et al.'s scheme [28], an arbitrary bit is embedded into a selected subset of numeric values by changing the distribution of the values. The selection of the values is based on a secret sorting. In another work, Gross-Amblard [13] designs a query-preserving scheme which guarantees that special queries (called local queries) can be answered up to an acceptable distortion. Recent work also includes watermarking categorical data [26], streaming data [29], XML data [27], and medical databases [2]. The watermarking schemes for categorical data [26, 2] exchange pairs of categorical values so as to embed watermark information. In this case, there is no insignificant change and the error constraint is considered at aggregation level (e.g., k-anonymity).

A common feature of this class of work is that a watermark is embedded and detected based on a secret key. Without knowing the key, an attacker is not able to locate exactly where the watermark is embedded, nor does he destroy the embedded watermark unless too many errors are introduced. A drawback of such a solution is that the ownership of watermarked data can be proven only once. After the key is revealed to the public (e.g., to the court) in the proof, anyone knowing the key can easily locate and remove the embedded watermark. Another common feature of these schemes is that the watermarking process introduces errors to the underlying data. This may severely affect database applications unless error constraints are carefully enforced in the watermarking process. In addition, a tradeoff between the watermarking error and the robustness of watermarking schemes has to be made.

The concept of *public key based* watermark (or asymmetric watermark) was first conceived in the multimedia context. Hachez and Quisquater summarized the work in this area in [14]. As mentioned in [14], one of the first ideas was proposed by Hartung and Girod

[15] for watermarking compressed video. The basic idea is to make a part of the embedded watermark public such that a user can check the presence of this part of watermark. However, an attacker is able to remove this part of watermark and thus invalidate a public detector. Another idea is to embed private key information into a host signal and detect a correlation between the signal and a transformation of the signal using a public key [33]. Other correlation-based public watermarking schemes include [9, 30, 11]. However, such watermarks can be removed by certain attacks such as a sensitivity attack [6, 21] or confusing attack [34].

Craver and Katzenbeisser [8] used a zero knowledge protocol to prove the presence of a watermark in a signal "without revealing the exact location and nature of the watermark (specified by a private key)." As in most zero knowledge protocols, the proposed scheme requires many rounds of interactions between prover and verifier, which may not be efficient in practice. It is also not clear how to extend this scheme to watermarking relational databases. Because the original watermark is not certified and because a verifier is allowed to perform the protocol multiple times, this scheme may be subject to oracle attack (an attacker uses a public detector repeatedly to test modified signals so as to remove the watermark), plain-text chosen attack (a special case of oracle attack in which the tested signals are chosen by an attacker), or ambiguity attack (also called invertibility attack, in which a fake watermark is discovered from the watermarked signal). In comparison, our scheme requires no interaction between a verifier and the owner of data, thus is immune to both oracle attack and plain-text chosen attack. The watermark is certified in our scheme for thwarting the ambiguity attack (which we call additive attack in this paper). In addition, our scheme is both efficient and robust for typical database operations.

6. CONCLUSION

In this paper, we proposed a public watermarking scheme for relational databases. The scheme is unique in that it has the following properties.

- **Public verifiability** Given a database relation to be published or distributed, the owner of data uses a public watermark key to generate a public watermark, which is a relation with binary attributes. Anyone can use the watermark key and the watermark to check whether a suspicious copy of data is watermarked, and, if so, prove the ownership of the data by checking a watermark certificate officially signed by a trusted certificate authority, DB-CA. The watermark certificate contains the owner's ID, the watermark key, the hashes of both the watermark and DB relation, the first time the relation was certified, the validity period of the current certificate, and the DB-CA's signature. The watermark certificate may be revoked and re-certified in the case of identity change, ownership change, DB-CA compromise, or data update. Therefore, the revocation status also needs to be checked in ownership proof. To our best knowledge, our scheme is the only one to achieve public ownership proof in database literature. In contrast, all existing schemes are based on secret key, by which ownership cannot be proven more than once in public.

- **Distortion free** Different from typical watermarking schemes (e.g., [1]) for database ownership proof that hide watermark information in data by modifying least significant bits (LSBs), our scheme generates a public watermark from a collection of the most significant bits (MSBs). Our scheme does not modify any MSBs; therefore, it is distortion-free. The public watermark is a database relation that has the same primary key attribute as the original data, plus one or more binary attributes to store the MSBs. Even though the MSBs are publicly known, an attacker cannot modify them without introducing intolerable error to the underlying data. In comparison, all previous watermarking schemes for databases introduce some kind of distortion to the watermarked data. They either modify LSB's for numerical data (e.g., [1, 19, 20]), or exchange values among categorical data (e.g., [26, 2]). Those schemes work well for particular types of data only, while our scheme can be applied for any type of data distortion-free.

- **Incremental updatability** Following the line of [1], each tuple in a database relation is independently processed in our scheme. Neither watermark generation nor detection depends on any correlation or costly sorting among data items as required in [28, 26, 2]. Therefore, the scheme is particularly efficient for typical database operations, which are mostly tuple oriented. In the case of tuple insertion, deletion, or modification, the watermark can be easily updated by processing those relating tuples only, with simple computation of random sequence numbers and modulus operations. Due to the robustness of our scheme, the update of watermark certificate can be performed periodically after a batch of data updates.

- **Robustness** Since the ownership of data is proven after the data is published or distributed, it is crucial that our scheme is robust against various attacks that intend to invalidate watermark detection or ownership proof. The robustness of our scheme is measured in terms of: (i) false hit, the probability of detecting a valid watermark from non-watermarked data; (ii) false miss, the probability of not detecting a valid watermark from watermarked data due to attacks; and (iii) failure error rate, the least portion of data that has to be intolerably modified so as to defeat our watermark detection. Typical database attacks considered in this paper include tuple/attribute insertion, deletion, and random/selecitive value modification/suppression. Both theoretical analysis and experimental study show that our scheme is robust in terms of these measures, which can be adjusted by the watermark generation and detection parameters. We have also studied the tradeoff between the robustness and the overhead of our scheme. Our scheme is robust against the data frame-up attack and additive attack that may be more perilous to public watermarking schemes.

The major contribution of this paper is the proposal of a public watermarking scheme that has the above properties. Though our scheme may not necessarily supersede secret key based schemes due to the overhead of using certificate and public watermark, we believe that it can be applied more practically in the real world for database ownership protection. Our future plan includes extending our scheme to other types of data such as XML and streaming data.

7. REFERENCES

[1] R. Agrawal and J. Kiernan. Watermarking relational databases. In *Proceedings of VLDB*, pages 155–166, 2002.

[2] E. Bertino, B. C. Ooi, Y. Yang, and R. Deng. Privacy and ownership preserving of outsourced medical data. In *Proceedings of IEEE International Conference on Data Engineering*, pages 521–532, 2005.

[3] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *Proceedings of CRYPTO'2001, LNCS 2139, Springer-Varlag*, pages 213–229, 2001.

[4] Coalition Against Database Piracy (CADP). Piracy is unacceptable in the information age or any other age, July 2, 2005. http://cadp.net/default.asp.

[5] C. Cocks. An identity based encryption scheme based on quadratic residues. In *Cryptography and Coding - Institute of Mathematics and Its Applications International Conference on Cryp- tography and Coding – Proceedings of IMA 2001, LNCS 2260*, pages 360–363, 2001.

[6] I. J. Cox and J. M. G. Linnartz. Public watermarks and resistance to tampering. In *Proceedings of International Conference on Image Processing*, pages 3–6, 1997.

[7] I. J. Cox, M. L. Miller, and J. A. Bloom. *Digital Watermarking: Principles and Practice*. Morgan Kaufmann, 2001.

[8] S. Craver and S. Katzenbeisser. Security analysis of public-key watermarking schemes. In *SPIE Vol. 4475, Mathematics of Data/Image Coding, Compression, and Encryption IV*, pages 172–182, 2001.

[9] J. J. Eggers, J. K. Su, and B. Girod. Public key watermarking by eigenvectors of linear transforms. In *Proceedings of European Signal Processing Conference (EUSIPCO)*, 2000.

[10] S. Farrell and R. Housley. An internet attribute certificate profile for authorization, internet draft, April, 2002. http://www.ietf.org/rfc/rfc3281.txt.

[11] T. Furon, I. Venturini, and P. Duhamel. A unified approach of asymmetric watermarking schemes. In *SPIE Vol. 4314, Security and Watermarking of Multimedia Contents III*, pages 269–279, 2001.

[12] B. Gray and J. Gorelick. Database piracy plague. *The Washington Times*, March 1, 2004. http://www.washingtontimes.com.

[13] D. Gross-Amblard. Query-preserving watermarking of relational databases and xml documents. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 191–201, 2003.

[14] G. Hachez and J. Quisquater. Which directions for asymmetric watermarking. In *Proceedings of XI European Signal Processing Conference (EUSIPCO), Vol. I*, pages 283–286, 2002.

[15] F. Hartung and B. Girod. Fast public-key watermarking of compressed video. In *Proceedings of IEEE International Conference on Speech and Signal Processing*, 1997.

[16] R. Housley, W. Ford, W. Polk, and D. Solo. Internet x.509 public key infrastructure certificate and crl profile, July 2, 2005. http://www.ietf.org/rfc/rfc2459.txt.

[17] N. F. Johnson, Z. Duric, and S. Jajodia. *Information Hiding: Steganography and Watermarking–Attacks and Countermeasures*. Kluwer Publishers, 2000.

[18] S. Katzenbeisser and F. A. Petitcolas, editors. *Information Hiding Techniques for Steganography and Digital Watermarking*. Artech House, 2000.

[19] Y. Li, V. Swarup, and S. Jajodia. Constructing a virtual primary key for fingerprinting relational data. In *Proceedings of ACM Workshop on Digital Rights Management (DRM)*, October 2003.

[20] Y. Li, V. Swarup, and S. Jajodia. Fingerprinting relational databases: Schemes and specialties. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 2(1):34–45, 2005.

[21] J. M. G. Linnartz and M. van Dijk. Analysis of the sensitivity attack against electronic watermarks in images. In *Proceedings of 2nd Workshop on Information Hiding Workshop*, 1998.

[22] A. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[23] S. Micali. Efficient certificate revocation. In *Technical Report: TM-542b.* Massachusetts Institute of Technology. Cambridge, MA, USA, 1996.

[24] B. Schneier. *Applied Cryptography*. John Wiley & Sons, Inc., 1996.

[25] A. Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of CRYPTO'84, LNCS 196, Springer-Varlag*, pages 47–53, 1984.

[26] R. Sion. Proving ownership over categorical data. In *Proceedings of IEEE International Conference on Data Engineering*, pages 584–596, 2004.

[27] R. Sion, M. Atallah, and S. Prabhakar. Resilient information hiding for abstract semi-structures. In *Proceedings of the Workshop on Digital Watermarking*, 2003.

[28] R. Sion, M. Atallah, and S. Prabhakar. Rights protection for relational data. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 98–108, 2003.

[29] R. Sion, M. Atallah, and S. Prabhakar. Resilient rights protection for sensor streams. In *Proceedings of the Very Large Databases Conference*, pages 732–743, 2004.

[30] J. Smith and C. Dodge. Developments in steganography. In *Proceedings of 3rd International Workshop on Information Hiding*, pages 77–87, 1999.

[31] G. W. Snedecor and W. G. Cochran. *Statistical Methods*. 8th edition, Iowa State Press, 1989.

[32] L. Vaas. Putting a stop to database piracy. *eWEEK, enterprise news and reviews*, September 24, 2003. http://www.eweek.com/print_article/0,3048,a=107965,00.asp.

[33] R. G. van Schyndel, A. Z. Tirkel, and I. D. Svalbe. Key independent watermark detection. In *Proceedings of IEEE International Conference on Multimedia Computing and Systems, Vol. 1*, 1999.

[34] Y. Wu, F. Bao, and C. Xu. On the security of two public key watermarking schemes. In *Proceedings of 4th IEEE Pacific-Rim Conference on Multimedia*, 2003.