2010 International Conference on Advances in Communication, Network, and Computing

Persistent Watermarking of Relational Databases

Raju Halder
Dipartimento di Informatica
Università Ca' Foscari di Venezia, Italy
halder@unive.it

Agostino Cortesi
Dipartimento di Informatica
Università Ca' Foscari di Venezia, Italy
cortesi@unive.it

***Abstract*—Digital watermarking for relational databases emerged as a candidate solution to provide copyright protection of relational data, maintaining integrity of the database information, tamper detection, traitor tracing etc. In this paper, we introduce the notion of persistent watermarking that serves as a way to recognize the integrity and ownership proof of the database bounded with a set of queries. It allows the evaluation of the database while applying the queries. We preserve the persistency of the watermark by exploiting two invariants of the database state *w.r.t.* the set of queries: *Stable Cells* and *Semantics-based Properties* of the data. We discuss how we can improve the existing techniques in terms of the persistency of the watermark. Moreover, we propose a novel persistent watermarking scheme that strictly improves the algorithm proposed by Li and Deng.**

***Keywords*-Watermarking, Databases, Security**

I. Introduction

In service provider models, a third-party entity manages and distributes various services and solutions to their end-users. For instance, an application service provider (ASP) model [1] provides the end-users various software-based services whereas the "database as a service" [2] model provides users the power to create, store, retrieve, and modify data from anywhere in the world through a given set of applications interacting with the databases. However, in the latter case, the database content may suffer from various challenges like illegal redistribution, ownership claims, forgery, theft, etc. The encryption of the database information stored on the service-provider site is one way to prevent attacks to the data from outsiders, and even from the service-providers themselves when they are not trusted. However, this approach is too restrictive and does not constitute a general solution to the challenges mentioned above. A more effective approach is using digital watermarking technologies where a watermark is some kind of information that is embedded into underlying data for tamper detection, localization, ownership proof, traitor tracing etc. There are many application areas where database watermarking might be of a crucial importance that include protecting rights and ensuring the integrity of outsourced databases, data publishing, data mining technology where data are sold in pieces to parties specialized in mining it, online B2B interactions etc [3], [4], [5]. The idea to secure a database of map information (represented as a graph) by digital watermarking technique was first coined by Khanna and Zane [6] in 2000. In 2002, Agrawal et al. [7] proposed the idea of digital watermarking for relational databases.

In general, the database watermarking techniques consist of two phases: watermark embedding and watermark verification. During the embedding phase, a private key K (known only to the owner) is used to embed the watermark W into the original database. The watermarked database is then made publicly available. To verify the ownership of a suspicious database, the verification process is performed where the suspicious database is taken as input, and by using the key K (the same which is used during the embedding phase) the embedded watermark (if present) is extracted and compared with the original watermark information.

The existing watermarking techniques can be broadly categorized into two: distortion-based and distortion-free.

The distortion-based watermarking techniques introduce tolerable changes in the underlying data of the database while embedding the watermark. The marking can be performed at bit-level or character-level or higher such as attribute or tuple level over the attribute values of the types numeric, string, categorical, or any. Agrawal et al. [7] first proposed a scheme to embed the watermark at bit level over numeric data type attributes. The data-part of the image in [8] is first converted into a watermark which is embedded at algorithmically chosen bit positions of the numeric data type attribute values. The watermarking schemes in [9], [10] are based on the content of the database itself. Information which is extracted from the database content is used as watermark information and is embedded into some/all (multi-)bit positions of the data in the same database. Unlike the aforementioned watermarking schemes where the marking is based on numeric attribute, the right protection scheme proposed

978-0-7695-4209-6/10 $26.00 © 2010 IEEE
DOI 10.1109/CNC.2010.18

by Sion et al. [11] is based on categorical type data. Authors in [12], [13] proposed watermarking schemes based on the insertion of the fake tuples and virtual attributes that are used as watermark information.

In contrast, the distortion-free watermarking schemes do not introduce any distortion to the underlying data. The existing techniques include (*i*) extraction of the hash value from the database content and use it as watermark [14], (*ii*) combining owner's mark and database features as watermark [15], (*iii*) converting the database relation into binary form which is used as watermark [16], [17], etc.

The watermark verification phase in the techniques discussed above completely rely on the content of the database. In other words, the success of the watermark detection is content dependent. *Benign Updates* or any other intensional processing of these content may damage or distort the existing watermark which leads to the watermark detection almost infeasible. For instance, suppose a publisher is offering 20% discount to the price of all articles. The modification of the price information may yield to the watermark detection phase almost infeasible, if the price values are marked at bit-level or if any information (*viz*, hash value) is extracted based on this price information and is used in the embedding phase. Therefore, most of the previous techniques are designed to face *Value Modification Attacks*, and therefore they are unable to resolve the persistency of the watermark under intentional operations. In addition, none of the existing techniques addresses the issue like how the watermarking impacts the query processing over it.

The aim of this paper is to introduce the notion of, and to design an algorithm for persistent watermarking, that serves as a way to recognize the integrity and ownership proof of the database bounded with a set of queries Q and that allows the evaluation of the database while applying queries in Q over it, focussing on the stable part and invariant properties of the data that have to be preserved in this process.

The rest of the paper is organized as follows: Section 2 introduces the notion of the persistent watermark. Section 3 discusses how to extract invariant information from the database state *w.r.t.* a given set of queries. In section 4 we discuss how we can exploit these invariants to the existing techniques and we propose a novel persistent watermarking technique. Section 5 illustrates the proposed scheme by an example. In sections 6 and 7, we compute time and space complexity of the proposed scheme and we provide a brief discussions to relate the scheme with the existing one. Finally, in section 8 we draw our conclusions.

II. The notion of persistent watermark

Given a database DB and the set of applications A interacting with the DB. Let Q be the set of queries issued by the applications in A. We denote the database model by the tuple $\langle DB, Q\rangle$. It is worthwhile to note that by Q we assume the set of all data manipulation operations *SELECT*, *UPDATE*, *DELETE*, *INSERT* over the database. We may avoid the discussion of the *SELECT* in this literature as it does not affect the existing watermark.

Let the initial state of the database DB be d_1. When queries in Q are issued over the DB, its initial state d_1 changes and goes through a number of valid states $d_2, d_3, \ldots, d_n$. Let W be the watermark that is embedded in the state d_1. The watermark is persistent if we can extract and verify the watermark W blindly from any of the following $n-1$ states successfully.

Definition 1: (Persistent Watermark)
Let $\langle DB, Q\rangle$ be a database model where Q represents the set of queries issued to the database DB. Suppose the initial state of DB is d_1 and application of queries in Q over d_1 yield to a set of valid states $d_2, \ldots, d_n$. A watermark W in the state d_1 is called persistent if

$$\forall i \in [2..n], \quad \text{verify}(d_1, W) = \text{verify}(d_i, W)$$

where verify(d, W) is a boolean function such that the probability of "verify(d, W) = *true*" is negligible when W is not the watermark embedded in d.

III. Obtaining Invariants from Database States

Given a database schema DB in initial state d_1 and a set of queries Q that can be issued by the applications associated with DB. Under the continuous processing of queries in Q over DB, the initial state d_1 goes through a set of valid states $d_2, d_3, \ldots, d_n$. Now we mention two invariants *Stable Cells* and *Semantics−based Properties w.r.t. Q* which remain unchanged over all the n states $d_1, d_2, \ldots, d_n$:

1) **Stable Cells:** Stable cells *w.r.t.* a given set of queries are the cells of the database that are not affected by the queries at all when DB will be in any of the n states d_i, $i \in [1..n]$. Let $CELL_{d_i}$, $i \in [1..n]$ be the set of cells in the i^{th} state d_i of DB. We denote the set of stable cells in state d_i *w.r.t.* the set of queries Q by $STB^Q_{d_i} \subseteq CELL_{d_i}$. For each tuple $t \in d_i$ we denote the stable part of t by $STB^Q_t \subseteq STB^Q_{d_i}$. Thus, $STB^Q_{d_i} = \bigcup_{t_j \in d_i} STB^Q_{t_j}$. The set $STB^Q_{d_i}$ or subset of it (as tuples may be deleted from d_i) remains invariant over all the n valid states d_i, $\forall i \in [1..n]$ under the processing of queries in Q.

2) **Semantics-based Properties of the data:** Given a database state d_i, $i \in [1..n]$ of DB, we can identify some semantics-based properties of the data in d_i *w.r.t.* a given set of queries. The properties include $intra-cell$ (IC), $intra-tuple$ (IT) and $Intra-attribute\ among-tuples$ (IA) property. We denote the set of semantics-based properties in state d_i *w.r.t.* the set of queries Q by $PR^Q_{d_i}$. For each tuple $t \in d_i$ we denote the set of IC, IT properties as $PR^Q_t \in PR^Q_{d_i}$. Note that IA properties can not be determined at tuple level. Thus, $PR^Q_{d_i} = \{\bigcup_{t_j \in d_i}(IC^Q_{t_j} \cup IT^Q_{t_j})\} \cup IA^Q_{d_i}$ where $IA^Q_{d_i}$ is the $Intra-attribute\ among-tuples$ (IA) property in the state d_i *w.r.t.* Q. Observe that in this case also the set $PR^Q_{d_i}$ or subset of it (as tuples may be deleted from d_i) remains invariant over all the n valid states d_i, $\forall i \in [1..n]$ under the processing of queries in Q.

Now we describe how to obtain the two invariants discussed above from any database state.

Given a database DB bounded with a set of queries Q. Let ATT^{update} be the set of attributes of DB that may be affected by the queries in Q that perform update operation. That is, attributes in ATT^{update} are targeted to modify by the UPDATE statements in Q. Complement of ATT^{update} *i.e.* $\neg ATT^{update}$ are the set of attributes whose values will never be updated or modified by the queries in Q. Thus we can identify the set of cells $STB^Q_{d_i}$, $i \in [1..n]$ in state d_i *w.r.t.* Q corresponding to the attributes in $\neg ATT^{update}$ which remains invariant over all the n valid states.

Given a database in state d_i and a set of queries Q, we can identify the set of semantics-based invariant properties $PR^Q_{d_i}$ in d_i *w.r.t.* Q as follows:

- **Intra-cell (IC) Property:** In this case individual cells of the relational databases represent some specific properties of interests. For instance, let the value val of a cell be $a \leq val \leq b$ over all the valid states, where a, b represent integer values. The $Intra-cell$ (IC) property can be represented by $[a, b]$ from the domain of intervals. Consider a publisher database that contains all the article information published by the publisher with *price* as numeric data type attribute and the associated applications can only decrease the values of the *price* attribute. Suppose the publisher can offer at most 50% discount to all the articles. Thus the article with price $100 can be at least $50. The $intra-cell$ property for this cell can be represented by the interval [50, 100]. Observe that, although the actual values may change under the processing by the queries, their properties represented by the elements from the domain of intervals remain unchanged.
- **Intra-tuple (IT) Property:** An intra-tuple property is a property which is extracted based on the inter-relation between two or more attribute values in the same tuple. For instance, let a and b be the values of two numeric attributes in the same tuple. Suppose there exists a relation $a + b \leq 10$ between these two attribute values. This can be abstracted by the relational abstract domain *viz* the domain of octagons [18] which represents the $Intra-tuple$ (IT) property. Consider an employee database with two attributes *Basic_Sal* and *Gross_Sal*. There is a relation between these two attribute values: *Gross_Sal* includes *Basic_Sal* plus HR, different allowances, incentives etc whose values are calculated as the percentage of the *Basic_Sal*. Observe that although the update operation may modify the values of the inter-related attributes but their relational property should remain intact.
- **Intra-attribute among-tuples (IA) property:** The $Intra-attribute\ among-tuples$ (IA) property is very difficult one to obtain from the set of independent tuples in a relation. This property can also be obtained in a similar way as that of IC or IT property. An example of such property is that in an employee database $\#male\ employee = \#female\ employee \pm 1$, where # denotes the cardinality of a set.

Observe that since *Stable Cells* and $Semantics-based\ Properties$ discussed above remain invariant under the processing of the data in a database, we can exploit these invariant information to perform persistent watermarking of the databases.

IV. Generic framework for persistent watermarking techniques

In previous section, we discussed how to extract the invariants from the database though its state may go through a set of n valid states under the processing by the queries in Q. We now exploit these invariants to perform persistent watermarking of the database.

Given a database DB in initial state d_1 bounded with a set of queries Q. We perform following steps to obtain a persistent watermarking:

Step 1: Identify the set of stable cells $STB^Q_{d_1} = \bigcup_{t_j \in d_1} STB^Q_{t_j} \subseteq CELL_{d_1}$ in the state d_1 which are not affected by the queries in Q.

Step 2: Extract the set of semantics-based properties $PR^Q_{d_1} = \{\bigcup_{t_j \in d_1}(IC^Q_{t_j} \cup IT^Q_{t_j})\} \cup IA^Q_{d_1}$ that consists of $intra-cell$ (IC), $intra-tuple$ (IT) and $intra-attribute\ among-tuples$ (IA) properties from the database information in state d_1 *w.r.t.* Q.

Step 3: Encode the stable part $STB^Q_{d_1}$ and the set of properties $PR^Q_{d_1}$ by suitable functions f^s_{encode} and g^p_{encode} respectively. For example, we can use *minimal perfect hash* function as g^p_{encode} that maps all these properties (sorted in intrinsic order, *viz*, lexicographic order) into a set of integers that preserve monotone property.

This invariant information can be exploited in different existing watermarking techniques:

1) **Watermarking based on numerical data type attribute [7], [8]:** AHK algorithms or image based techniques identify some fit tuples first, and then embed the watermark information at bit-level over numeric data type attribute. They can be used for embedding the watermark information into the stable portion only in order to provide the persistency of the watermark.
2) **Content characteristics as watermark information [9], [10]:** In these techniques, the extracted information from the database content is treated as watermark and is embedded into the data of the same database. But since the content keeps changing under various operations in Q, the modification of the content may lead to the extracted information different in verification phase from that in embedding phase. Thus, in this case also we can use these techniques in order to extract only the invariant properties and embed it into the stable part of the database content.
3) **Watermarking based on tuples or attributes insertion [12], [13]:** The values of the fake tuples or virtual attributes depend on the database content, and thus the schemes suffer from the same problem as before. So in these cases, we can also exploit the encoded values obtained from the invariant properties and stable cells to generate fake tuples or virtual attributes.

We can also use the invariant information to perform distortion-free watermarking:

1) **Extracting Hash Value as watermark information [14]:** In this technique, tuple level and then group level hash value is computed. From these hash values a watermark of specific length is extracted and based on this watermark information database tuples are permuted. So hash value based on the extracted invariant information results into a persistent watermark.
2) **Combining owner's mark and database feature as watermark information [15]:** Since database feature is content dependent, we can also exploit the invariant properties in this technique to generate invariant database feature.
3) **Converting database relation into binary form used as watermark information [16]:** The binary form of the database which is used as watermark, depends on the most signification bits (MSBs) of the database information. Modification of the database information may also lead to a change in MSB yielding to a different binary form. Thus, although this technique is able to detect any tamper to the data, but unable to perform the ownership proof in such situation. In the next section we strictly improve this technique by exploiting the invariant information extracted from the database that serves as a way to obtain persistent watermark by generating a binary form of the database.

A. A new persistent watermarking technique

The watermarking technique we propose combines the features of both the database and its associated owner's signature. It consists of two phases: Partitioning and Watermarking. Notice that the algorithm in [16] generates only the binary form of the database by extracting the MSBs of the database content: we strictly improve the technique in terms of the persistency of the watermark.

Let the database schema be $DB(PK, A_1, A_2, A_3, \ldots, A_\gamma)$ where PK is the primary key. We divide the attribute set $\{A_1, A_2, A_3, \ldots, A_\gamma\}$ into two parts: the set of attributes belonging to the stable part $A_{stable} = \{A^s_1, A^s_2, \ldots, A^s_q\}$ and the set of attributes belonging to the unstable part $A_{var} = \{A^v_1, A^v_2, \ldots, A^s_p\}$ where $p+q = \gamma$. Let K be the private key which is large enough to thwart Brute-force attack and is known to the database owner only.

1) Partitioning: We use the partitioning algorithm of [13] with slight modifications. We use the private key K and the number of partitions m as hidden parameters. We consider the value of the primary key PK and the values of the attributes $A^s_1, A^s_2, \ldots, A^s_q$ belonging to the stable part of DB as the partitioning attributes. Suppose $|d|$ represents the number of tuples in database state d which is to be partitioned into m non-overlapping partitions, namely $\{S_1, S_2, \ldots, S_m\}$ such that each partition contains on average $\frac{|d|}{m}$ tuples. Observe that $S_i \cap S_j = \emptyset$ where $i \neq j$. The following two steps determine which partition a tuple belongs to:

Step 1: $\nu = (hash(K \circ PK \circ A^s_1 \circ A^s_2 \circ \cdots \circ A^s_q) \ mod \ m) + 1$.
Step 2: Assign the tuple into the partition S_ν.

2) Watermarking: The watermarking algorithm W-Create is depicted in Figure 1. The algorithm takes m non-overlapping partitions $\{S_1, S_2, \ldots, S_m\}$ of the database in state d bounded with a set of queries Q, the owner's signature ξ converted into a binary form

of length m (where m is the no. of partitions), and the private key K. It generates the watermark W whose schema is $W(PK, b_1, \ldots, b_\gamma, p_1, p_2, p_3)$.

Algorithm 1: W-Create

Input: m non-overlapping partitions $\{S_1, S_2, \ldots, S_m\}$ of database $DB(PK, A_1, \ldots, A_\gamma)$ in state d bounded with set of queries Q, Binary form of owner's signature ξ of length m, Private Key K
Output: An watermark table $W(PK, b_1, \ldots, b_\gamma, p_1, p_2, p_3)$

Determine *intra − attribute among − tuples* property IA_d^Q;
FOR each partition $S_v \subseteq d$, $v \in [1....m]$ **DO**
  FOR each tuple $r \in S_v$ **DO**
    Construct tuple t in W with primary key $t.PK = r.PK$;
    Determine IC_r^Q, IT_r^Q;
    $t.p_1 = g^p_{encode}(IC_r^Q)$;
    $t.p_2 = g^p_{encode}(IT_r^Q)$;
    $t.p_3 = g^p_{encode}(IA_d^Q)$;
    $V_{stable} = r.A_1^s \circ r.A_2^s \circ \cdots \circ r.A_q^s$;
    FOR (i=1; i≤ γ; i=i+1) **DO**
      $val = G_i(K \circ r.PK \circ V_{stable})$;
      $j = val\%(no.\ of\ attributes\ in\ r)+1$;
      IF j^{th} attribute in r belongs to stable part A_{stable} **THEN**
        $t.b_i$= (MSB of j^{th} attribute in r) XOR ($\xi[v-1]$);
      ELSE $t.b_i$=0;
      END IF;
      delete the j^{th} attribute from r;
    END FOR;
  END FOR;
END FOR;

Figure 1. Watermark Creation Algorithm

For each tuple r in the partition S_v, the algorithm generates a tuple t in W whose primary key is equal to the primary key of r just to identify the tuples in W uniquely. Then the algorithm adds three values for the attributes p_1, p_2 and p_3 in t that correspond to the encoded values of *intra − cell*, *intra − tuple* properties for each tuple r and encoded value of *intra − attribute among − tuples* property for the whole database state d. If any of these properties is missing, we simply put 0 into the corresponding cell. G_i represents the i^{th} value obtained from the pseudorandom sequence generator (for instance, Linear Feedback Shift Register [19]) seeded with primary key, V_{stable} which is computed from the stable part of tuple r, and the private key K. For all i from 1 to γ, the i^{th} value val chooses an attribute randomly in r. If this attribute belongs to the stable part A_{stable}, the algorithm puts the MSB of that attribute value XOR-ed with $(v-1)$-th watermark bit (where v is the partition index and assuming array indexing for signature ξ starts from 0) into the corresponding i^{th} attribute in t, and puts 0 if the attribute belongs to the variable part A_{var}. Observe that this operation combines the database feature with owner's signature.

It is worthwhile to mention that since the generation of the watermark W is based only on the invariant properties and the stable part of the database, the watermark detection algorithm is deterministic in nature rather than probabilistic as in [16]. The input of the detection algorithm are m non-overlapping partitions of the database in state d' bounded with Q, Binary form of owner's signature ξ of length m, Private Key K, and the output is the watermark W'. Finally we use the boolean function $match(W, W')$ to compare W' with the original watermark W which is obtained in the watermark creation phase. Note that the function $match(W, W')$ compare tuple by tuple taking into account the the primary key of W and W'. As tuples may be deleted from or added to the initial state d and yield to a different state d', only those tuples whose primary keys are common in both W and W' are compared. If $match(W, W') = True$, then the claim of the ownership is true, and false otherwise. Due to the space limitation, we skip the detail discussions of the watermark detection algorithm.

V. Illustration with an example

Consider an employee database that consists of a table *emp* with *eID* as the primary key as shown in Figure 2. Suppose the queries associated with the database are only able to increase the basic and gross salary of the employees by at most 30%. Thus, A_{stable} =

eID	Name	Basic Sal (euro)	Gross Sal (euro)	Age	DNo
E001	Bob	1000	1900	48	2
E002	Alice	900	1685	29	1
E003	Matteo	1200	2270	58	2
E004	Tom	600	1190	30	2
E005	Marry	1350	2542.5	55	1

Figure 2. Table *emp*

$\{Name, Age, Dno\}$ and $A_{var} = \{Basic\ Sal, Gross\ Sal\}$. Suppose, the private key K = "*UNIVERSE*" and $m = 3$, that is, the tuples of *emp* will be grouped into 3 distinct partitions S_1, S_2 and S_3 based on the value computed using:

$$hash(eID \circ \text{"}UNIVERSE\text{"} \circ Name \circ Age \circ Dno)\ mod\ 3) + 1$$

Let the 1^{st} bit value in the owner's signature S *i.e.* $S[0]$ be 1. Consider the first partition S_1 and a tuple r, say, $\langle E002, Alice, 900, 1685, 29, 1\rangle$ in that partition. For this tuple r we create a tuple t in watermark table W with $t.PK = E002$.

For the employee represented by r the basic salary can be at most 1170 euro. Thus, we can represent

the basic salary of r by the $intra-cell$ (IC) property $[900, 1170]$ from the domain of intervals. The relation between the two attributes *Basic Sal* and *Gross Sal* can be represented, for instance, by the following equation: assuming *Gross Sal* includes *Basic Sal*, 65% of the *Basic Sal* as *PF, HRA* etc and minimum of 200 euro as incentive,

$$Gross\ Sal \geq \frac{(165 \times Basic\ Sal)}{100} + 200$$

Thus, the $intra-tuple$ (IT) property can be obtained by abstracting the above relation by the elements from the domain of polyhedra [20] *i.e.* by the linear equation just mentioned. The $intra-attribute\ among-tuples$ property may be: "The number of employees in every department is more than 2". This can also be represented by $[3, +\infty]$ in the domain of intervals. Suppose after encoding these three properties, we obtain the encoded values k_1, k_2, k_3. Therefore, the values of the attributes p_1, p_2, p_3 in t will be k_1, k_2, k_3 respectively.

Suppose the random selection of the attributes in r based on the random value generated by the pseudorandom sequence generator yield to the selection order as follows: $\langle 1685, 1, 29, 900, Alice \rangle$. Since 1685 and 900 belongs to A_{var}, the corresponding bit-values in t will be 0 and for other attribute values (*i.e.* '1', '29' and '*Alice*') the corresponding bit-values in t will be the MSB of '1', '29' and '*Alice*' XORed with $S[0] = 1$ *i.e.* $0 \otimes 1=1$, $0 \otimes 1=1$ and $0 \otimes 1=1$ respectively. Thus the tuple t in W would be $\langle E002, 0, 1, 1, 0, 1, k_1, k_2, k_3 \rangle$.

After performing the similar operations for all the tuples in all partitions, the watermark W is generated.

VI. Time and Space Complexity

The time to generate watermark W depends on the following operations:

1) To identify stable and unstable cells.
2) Encoding $intra-attribute\ among-tuples$ property obtained from the database state *w.r.t.* a given set of queries.
3) Partitioning of tuples into m distinct partitions.
4) For each tuple:
   a) Random selection of the attributes based on the pseudorandom values.
   b) Checking of the attributes belonging to the stable part or not, and marking MSBs by signature bits.
   c) Encoding $intra-cell$ and $intra-tuple$ properties *w.r.t.* a given set of queries.

In our scheme we consider the database as having a fixed structure under various DML operations. The only thing that can vary is the number of tuples over different valid states under the processing of queries over it. Thus time complexity of the watermark generation depends on the number of tuples η in state d linearly *i.e.* time complexity of the watermark generation algorithm $W-Create$ is $O(\eta)$ where η is the number of tuples in the database state.

Given the database schema as $DB(PK, A_1, A_2, A_3, \ldots, A_\gamma)$, the number of attributes in W is $\gamma + 4$, since it includes 3 more attributes p_1, p_2 and p_3 corresponding to the semantics-based properties. Let the values of the primary key PK and the values of the encoded properties p_1, p_2 and p_3 can be represented by total σ bits. Thus the total number of bits in W is $(\sigma + \gamma) \times \eta$, and the space complexity can be represented by $O(\eta)$ as well.

VII. Discussions

Let us briefly discuss how the proposal relates with the Li and Deng technique [16].

- The watermark detection algorithm of [16] is parameterized with a threshold value. The lower value of the threshold would increase the probability to success the watermark verification. Probabilistic schemes may yield to false negative and false positive. We strictly improve the technique by exploiting the invariants of the database state and thus, the verification phase is deterministic in nature.
- In [16], Li and Deng suggest how to choose the most significant bits (MSBs) from the attribute values of different types. However, instead of choosing only the fixed MSB, we can also randomize it by introducing a parameter λ, where λ represents the number of most significant bits available to choose. We choose the j^{th} bit from the λ available MSBs of the attribute A using the following equation: $j = hash(PK \circ A)\ mod\ \lambda$.
- In watermark W, the cells that correspond to the unstable part of the database contain only 0s. As we statically know the set of attributes belonging to the unstable part of the database, we can simply omit that unstable binary part containing only 0s from W to improve the efficiency in terms of time and space complexity.
- As suggested in [16], we can also use the *watermark generation* parameter $x \leq \gamma$, that controls the number of attributes in W. In such case, the schema of W will be $W(PK, b_1, \ldots, b_x, p_1, p_2, p_3)$. Thus the number of bits in watermark would be $(\sigma + x) \times \eta$ where η is the number of tuple and σ is the total number of bits required to represent the primary key and properties p_1, p_2, p_3.
- Unlike [16], the proposed scheme is not publicly verifiable, as the embedding and verification phases are based on partitioning of the tuples using a private key.

VIII. Conclusions

In the context of the database watermarking, we deal with a burning issue called persistency of the watermark under the processing by a given set of queries bounded with the database. We discuss how to extract two invariants: stable cells and semantics-based properties of the data at different level. We can exploit these invariants into the existing watermarking techniques effectively.

Acknowledgement

Work partially supported by Italian MIUR COFIN'07 project "SOFT" and by RAS project TESLA - Tecniche di enforcement per la sicurezza dei linguaggi e delle applicazioni.

References

[1] B. Desai and W. Currie, "Application service providers: a model in evolution," in *ICEC '03: Proceedings of the 5th international conference on Electronic commerce*. Pittsburgh, Pennsylvania: ACM, 2003, pp. 174–180.

[2] H. Hacigumus, B. Iyer, and S. Mehrotra, "Providing database as a service," in *ICDE '02: Proceedings of the 18th International Conference on Data Engineering*. IEEE Computer Society, 2002, pp. 29–38.

[3] Q. Zhu, Y. Yang, J. Le, and Y. Luo, "Watermark based copyright protection of outsourced database," in *10th International Database Engineering and Applications Symposium (IDEAS'06)*. Delhi, India: IEEE CS, December 2006, pp. 301–308.

[4] P. Jurczyk and L. Xiong, "Privacy-preserving data publishing for horizontally partitioned databases," in *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*. Napa Valley, California, USA: ACM, 2008, pp. 1321–1322.

[5] J. Hu and P. W. P. J. Grefen, "Component based system framework for dynamic b2b interaction," in *COMPSAC '02: Proceedings of the 26th International Computer Software and Applications Conference on Prolonging Software Life: Development and Redevelopment*. IEEE Computer Society, 2002, pp. 557–562.

[6] S. Khanna and F. Zane, "Watermarking maps: hiding information in structured data," in *SODA '00: Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2000, pp. 596–605.

[7] Agrawal, Rakesh and Haas, Peter J. and Kiernan, Jerry, "Watermarking relational data: framework, algorithms and analysis," *The VLDB Journal*, vol. 12, no. 2, pp. 157–169, 2003.

[8] X. Zhou, M. Huang, and Z. Peng, "An additive-attack-proof watermarking mechanism for databases' copyrights protection using image," in *SAC '07: Proceedings of the 2007 ACM symposium on Applied computing*. Seoul, Korea: ACM Press, 2007, pp. 254–258.

[9] Y. Zhang, X. Niu, D. Zhao, J. Li, and S. Liu, "Relational databases watermark technique based on content characteristic," in *First International Conference on Innovative Computing, Information and Control (ICICIC 2006)*. Beijing, China: IEEE Computer Society, 30 August - 1 September 2006.

[10] H. Guo, Y. Li, A. Liua, and S. Jajodia, "A fragile watermarking scheme for detecting malicious modifications of database relations," *Information Sciences*, vol. 176, pp. 1350–1378, May 2006.

[11] R. Sion, M. Atallah, and S. Prabhakar, "Rights protection for categorical data," *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, vol. 17, pp. 912–926, July 2005.

[12] V. Pournaghshband, "A new watermarking approach for relational data," in *ACM-SE 46: Proceedings of the 46th Annual Southeast Regional Conference on XX*. Auburn, Alabama: ACM Press, 2008, pp. 127–131.

[13] V. Prasannakumari, "A robust tamperproof watermarking for data integrity in relational databases," *Research Journal of Information Technology*, vol. 1, pp. 115–121, July 2009.

[14] Y. Li, H. Guo, and S. Jajodia, "Tamper detection and localization for categorical data using fragile watermarks," in *DRM '04: Proceedings of the 4th ACM workshop on Digital rights management*. Washington DC, USA: ACM Press, 2004, pp. 73–82.

[15] M.-H. Tsai, H.-Y. Tseng, and C.-Y. Lai, "A database watermarking technique for temper detection," in *Proceedings of the 2006 Joint Conference on Information Sciences, JCIS 2006*. Kaohsiung, Taiwan, ROC: Atlantis Press, October 8-11 2006.

[16] Y. Li and R. H. Deng, "Publicly verifiable ownership protection for relational databases," in *ASIACCS '06: Proceedings of the 2006 ACM Symposium on Information, computer and communications security*. Taipei, Taiwan: ACM, 2006, pp. 78–89.

[17] S. Bhattacharya and A. Cortesi, "A generic distortion free watermarking technique for relational databases," in *Proceedings of the Fifth International Conference on Information Systems Security (ICISS 2009)*. Kolkata, India: LNCS Springer Verlag, December 2009, pp. 252–264.

[18] A. Miné, "The octagon abstract domain," *Higher Order Symbol. Comput.*, vol. 19, no. 1, pp. 31–100, 2006.

[19] R. Halder, P. Dasgupta, S. Naskar, and S. S. Sarma, "An internet-based ip protection scheme for circuit designs using linear feedback shift register (lfsr)-based locking," in *Proceedings of the 22nd ACM/IEEE Annual Symposium on Integrated Circuits and System Design(SBCCI'09)*. Natal, Brazil: ACM Press, 31st Aug–3rd Sep 2009.

[20] L. Chen, A. Miné, and P. Cousot, "A sound floating-point polyhedra abstract domain," in *APLAS '08: Proceedings of the 6th Asian Symposium on Programming Languages and Systems*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 3–18.