A DISTORTION FREE WATERMARK FRAMEWORK FOR RELATIONAL DATABASES

Sukriti Bhattacharya and Agostino Cortesi
Dipartimento di Informatica, Universita Ca' Foscari di Venezia, Via Torino 155, 30170 Venezia, Italy

Keywords: Database watermarking, HMAC, Abstract interpretation.

Abstract: In this paper we introduce a distortion free invisible watermarking technique for relational databases. The main idea is to build the watermark after partitioning tuples with actual attribute values. Then, we build hash functions on top of this grouping and get a watermark as a permutation of tuples in the original table. As the ordering of tuples does not affect the original database, this technique is distortion free. Our contribution can be seen as an application to relational databases of software watermarking ideas developed within the Abstract Interpretation framework.

1 INTRODUCTION

A watermark can be considered to be some kind of information that is embedded into underlying data for tamper detection, localization, ownership proof, and/or traitor tracing purposes (Agrawal et al., 2003). Watermarking techniques apply to various types of host content. Here, we concentrate on relational databases. Rights protection for such data is crucial in scenarios where data are sensitive, valuable and nevertheless they need to be outsourced. A good example is data mining application, where data are sold in pieces to parties specialized in mining it, e.g. sales patterns database, oil drilling data, financial data. Other scenarios involve for example online B2B interactions, e.g., airline reservation and scheduling portals, in which data are made available for direct, interactive use. Given the nature of most of the data, it is hard to associate rights of the originator over it (Lafaye, 2007). Watermarking can be used to solve these issues. Unlike encryption and hash description, typical watermarking techniques modify ordinal data as a modulation of the watermark information and inevitably cause permanent distortion to the original data and therefore cant not meet the integrity requirement of the data as required in some applications.

The first well-known database watermarking scheme for relational databases was proposed by Agrawal and Kiernan (Agrawal et al., 2003) for watermarking numerical values. The fundamental assumption is that the watermarked database can tolerate a small amount of errors. Since any bit change to a categorical value may render the value meaningless, Agrawal and Kiernan's scheme cannot be directly applied to watermarking categorical data. To solve this problem, Sion (Sion, 2004) proposed to watermark a categorical attribute by changing some of its values to other values of the attribute (e.g., 'red' is changed to 'green') if such change is tolerable in certain applications.

All of the work cited so far, assume that minor distortions caused to some attribute data can be tolerated to some specified precision grade. However some applications in which relational data are involved cannot tolerate any permanent distortions and data's integrity needs to be authenticated. To meet this requirement, we further strengthen this approach and propose a distortion free watermarking algorithm for relational databases based on reordering tuples. The robustness of the proposed watermarking obviously depends on the size of the individual groups so it is specifically designed for large databases. The resulting watermark is robust against various forms of malicious attacks and updates to the data in the table.

Database watermarking consists of two basic processes: watermark insertion and watermark detection (Agrawal et al., 2003), as illustrated in Figure 1. For watermark insertion, a key is used to embed watermark information into an original database so as to produce the watermarked database for publication or distribution. Given appropriate key and watermark information, a watermark detection process can be applied to any suspicious database so as to determine whether or not a legitimate watermark can be

Bhattacharya S. and Cortesi A. (2009).
A DISTORTION FREE WATERMARK FRAMEWORK FOR RELATIONAL DATABASES.
In *Proceedings of the 4th International Conference on Software and Data Technologies*, pages 229-234
DOI: 10.5220/0002256402290234

detected. A suspicious database can be any watermarked database or innocent database, or a mixture of them under various database attacks.

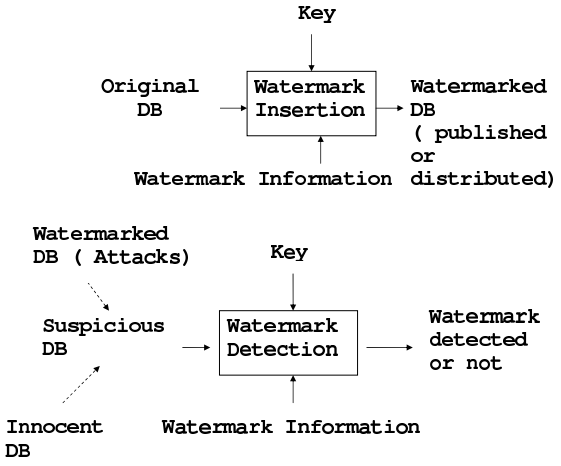

Figure 1: Basic watermarking process.

While the basic processes in database watermarking are quite similar to those in watermarking multimedia data, the approaches developed for multimedia watermarking cannot be directly applied to databases because of the difference in data properties. These differences include (Agrawal et al., 2003):

- A multimedia object consists of a large number of bits with considerable redundancy. Thus, the watermark has a large cover in which to hide. A database relation consists of tuples, each of which represents a separate object. The watermark needs to be spread over these separate objects.;
- The relative spatial/temporal positioning of various pieces of a multimedia object typically does not change. Tuples of a relation, on the other hand, constitute a set, and there is no implied ordering between them.
- Multimedia objects typically remain intact; portions of an object cannot be dropped or replaced arbitrarily without causing perceptual changes in the object. On the other hand, tuple insertions, deletions, and updates are the norm in the database setting.

Once outsourced, i.e., out of the control of the watermarker, data might be subjected to a set of attacks or transformations; these may be malicious e.g., with the explicit intent of removing the watermark - or simply the result of normal use of the data. An effective watermarking technique must be able to survive such use. The main idea of this paper is to build the watermark after partitioning tuples which depend on the actual attribute values and then to build hash functions on top of these groupings. Our contribution can be seen as an application to relational databases of software watermarking ideas developed within the abstract interpretation frame work (Cousot and Cousot, 2004) and (Cousot and Cousot, 2007).

The paper is organized as follows. In section 2 we formalize the definition of tables in relational database and hence a formal definition of watermaking process of a table in relational database is given. Section 3 illustrates how a table in relational database can be abstracted. Then a distortion free algorithm for watermark embedding and verification, on top of this abstraction, is provided in section 4 and 5, respectively. Section 6 illustrates the algorithms by a suitable example. The correctness proof of this watermarking process is given in section 7. A robustness analysis of this watermarking scheme is given in section 8. Finally we draw our conclusions in section 9.

2 PRELIMINARIES

This section contains formal definitions (Collberg and Thomborson, 2002) and (Haan and Koppelaars, 2007) of tables in relational database and database watermarking.

Definition 2.1: ***Function.***
Let Π_i be the projection function which selects the i-th coordinate of a pair. F is a function over the set A into set B $\Leftrightarrow F \in \wp(A \times B) \bigwedge (\forall p_1, p_2 \in F : p_1 \neq p_2 \Rightarrow \Pi_1(p_1) \neq \Pi_1(p_2)) \bigwedge \{\Pi_1(p) | p \in F\} = A$.

Definition 2.2: ***Set Function.***
A set function is a function in which every range element is a set. Formally, let F is a set function $\Leftrightarrow$ F is a function and $(\forall c \in dom(F) : F(c)$ is a set).
For instance we can express information about companies and their locations by means of a set function over the domain {Company, Location}, namely.
(Company;{'Natural Join', 'Central Boekhuis', 'Oracle', 'Remmen & De Brock'}) (Location, {'New York', 'Venice', 'Paris'})

Definition 2.3: ***Table.***
Given two sets H and K, a table over H and K is a set of functions T over the same set H and into the same set K. i.e. $\forall$ t $\in$ T: t is a function from H to K.
For instance consider a table containing data on employees:
The table is represented by the set of functions t_1, t_2, t_3 where dom(t_i) = emp_no, emp_name, emp_rank and

Table 1: EMPLOYEE.

emp_no	emp_name	emp_rank
100	John	Manager
101	David	programmer
103	Albert	HR

for instance t_1(emp_name) = John.

There is a correspondence between tuples and functions. For instance, t_1 corresponds to the following tuple: (emp_no, 100), (emp_name, John), (emp_rank, manager). The first coordinates of the ordered pairs in a tuple are referred to as the attributes of that tuple.

Definition 2.4: ***Watermarking.***
A watermark W for a table T over H into K, is a predicate such that W(T) is true and the probability of $W(T')$ being true with $T' \in \wp(H \times K) \backslash T$ is negligible.

3 DATABASE ABSTRACTION

Our watermarking technique can be seen as an extension to relational databases of general ideas as software watermarking as introduced by P. Cousot and R.Cousot (Cousot and Cousot, 2004).

Given a relational database (i.e. a set of tables), we assume that the semantics of a query is precisely defined, and its answer is unique. Let A be the set of attributes in T. The domain D of a categorical attribute x ∈ A in table T is the set of all possible values of x, and the (finite) value set V (⊆ D) is the set of values of x that are actually present in T. We can partition the tuples in T by grouping the values of attribute x by P = $[[v_i] : 1 \leq i \leq N$, where $\forall t \in T : t.x = v_i \Leftrightarrow t \in [v_i]$. The frequency q_i of v_i is the number of tuples in $[v_i]$. The data distribution of x (in T) is the set of pairs $\tau = \{(v_i, f_i) | 1 \leq i \leq N\}$. So the entire database can be partitioned into N fixed mutual exclusive areas based on each categorical value v_i. The above concept leads to an abstraction as depicted in Figure 2.

Given a table T over A into D and a categorical attribute x ∈ A and P = $\{[v_i] : 1 \leq i \leq N\}$, for each set $S \subseteq T$, We can define a concretization map γ_x as follows:

$$\begin{cases} \gamma_x(v_i,h) & = S \subseteq T \mid \forall t \in S : t \in [v_i] \wedge \text{size of S is h} \\ \gamma_x(\top) & = T \\ \gamma_x(\bot) & = \emptyset \end{cases}$$

The best representation of a set of tuples with attribute x is captured by the corresponding abstraction function α_x :

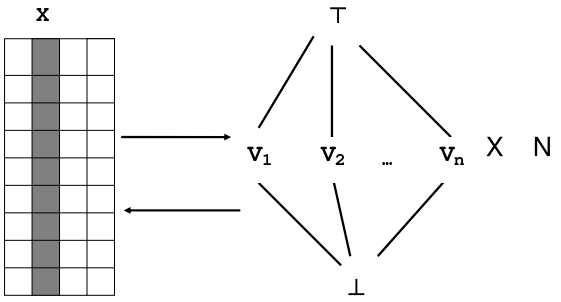

Figure 2: Table Abstraction (Galois Connection).

$$\alpha_x(S) = \begin{cases} (v_i,h) & \text{if } \forall t \in S : t.x = v_i \wedge \text{size of S is h} \\ \top & \text{if } \exists t_1, t_2 \in S : t_1.x \neq t_2.x \\ \bot & \text{if } S = \emptyset \end{cases}$$

We may prove that (α_x, γ_x) form a Galois insertion (Myrvold and Ruskey, 2001) with α_x monotone and γ_x weakly monotone, i.e. $(v,u) \leq (v,m) \Rightarrow (\cup\gamma(v,u)) \subseteq (\cup\gamma(v,u))$.

4 WATERMARK EMBEDDING

The watermark embedding process is partition based on a partition P = $\{[v_i] : 1 \leq i \leq N\}$ corresponding to a given categorical value x. This partitioning can be seen as a virtual grouping which does not change the physical position of the tuples. After grouping, all tuples in each group are sorted according to their primary key . Like grouping, the sorting operation does not change the physical position of tuples either. Each group is then considered (hence, water marked)independently.

Recall that we denote by q_k the frequency of $[v_k]$, i.e the number of tuples in $[v_k]$. A (keyed) group hash value (H_{qk}) is computed using a HMAC (HMAC, 2002) function based on the tuple hash values in a sorted order. Then, a watermark W = extractBits($H_{qk}, ln(q_k)$) of length ln(q_k) is derived from H_{qk}. The watermark W is embedded into this group by permuting the order of the tuples. The new order π can be easily calculated from W using Myrvold and Ruskey's linear permutation unranking algorithm (Myrvold and Ruskey, 2001) based on W:

Algorithm 1: Watermark Embedding in $[v_k]$.

Begin
for i = 1 to q_k **do** // *number of tuples in group* G_k
h_i = HMAC($\Re, r_i$) // *tuple hash of tuple* r_i
end for
H_1 = HMAC($\Re, h_1$)
for i = 2 to q_k **do**
H_i = HMAC($\Re$, $H_{(i-1)}$ * $p_i^{h_i}$)
//*where* p_i *is the i-th prime number. e.g.* p_2*=2,* p_5*=7 and so on*
end for
W = **ExtractBits**($H_{qk}, ln(q_k)$)
Unrank(q_k, W, π_k)
End
Procedure ExtractBits(H_{qk}, l)
l = Concatenation of first l selected bits from H_{qk}
// assuming H_{qk} *is longer than l*
return l
End
Procedure Unrank(q_k, W, π_k)
π_k = (0, ... , q_{k-1})
if ($q_k > 0$) **then**
swap($\pi[q_k$-1], π_k[W mod q_k])
Unrank($q_k - 1$, $\lfloor W/q_k \rfloor$, π_k)
end if
End

Algorithm 2: Watermark verification in $[v_k]$.

Begin
for i = 1 to q_k **do** // *number of tuples in group* G_k
h_i = HMAC($\Re, r_i$) // *tuple hash of tuple* r_i
end for
H_1 = HMAC($\Re, h_1$)
for i = 2 to q_k **do**
H_i = HMAC($\Re$, $H_{(i-1)}$ * $p_i^{h_i}$)
// *where* p_i *is the i-th prime number. e.g.* p_2*=2,* p_5*=7 and so on*
end for
W = **ExtractBits**($H_{qk}, ln(q_k)$)
W'=**rank**(q_k, π_k, π_k^{-1})
if (W = W') **then**
return (Authentication Preserved)
else
return (Authentication Violated)
End
Procedure rank(q_k, π_k, π_k^{-1})
if(q_k = 1) **then return** 0
s = $\pi_k[q_k - 1]$
swap($\pi_k[q_k - 1], \pi_k[\pi_k^{-1}[q_k - 1]]$)
swap($\pi_k^{-1}[s], \pi_k^{-1}[q_k - 1]$)
return(s+q_k * rank($q_k - 1$, π_k, π_k^{-1}))
End

5 WATERMARK VERIFICATION

A very important problem in a watermarking scheme is synchronization, that is, we must ensure that the watermark extracted is in the same order as that embedded. If synchronization is lost, even if no modifications have been made, the embedded watermark cannot be correctly verified. In watermark detection, a group of q_k tuples are selected and their ordering π_k is identified, where π_k is a permutation of (0, . . . , $q_k - 1$). A watermark W' can be derived from π_k using Myrvold and Ruskey's linear permutation ranking algorithm (Myrvold and Ruskey, 2001). Based on the tuple hash of the sorted tuples, a group hash value is computed (H_{qk}) using a HMAC fuction with same key value $\Re$ used during embedding. Then, a watermark W is extracted from the group hash. If W matches W', the tuples in this group are authentic; otherwise, the data in this group have been modified or tampered with.

6 EXAMPLE

This section illustates an example of applying this watermarking technique on the table PARTS = (prts_no,color) 2 which only has 14 tuples with primary key prts_no. TABLE 2 is the original table before partitioning operation.

Table 3 is the partitioned table based on the color attribute with hash values h_i, associated with each tuple. The table contains three partitions $[v_{red}]$,$[v_{blue}]$ and $[v_{black}]$. Frequencies of each group are q = 8, q = 4 and q = 2, respectively.

For simplicity, we only show hypothetical hash values, not the real ones, for each tuple. Group hash values H_{blue},H_{red} and H_{black} are calculated using tuple hash values h_i, $1 \leq i \leq 14$ associated with corresponding tuples and a secured key $\Re$. Let us suppose that W_{blue}= ExtractBits (H_{blue}, ln (q_{blue})) = $(10)_2$,W_{red}= ExtractBits(H_{red}, ln (q_{red})) = $(011)_2$ and W_{black}= ExtractBits(H_{black},ln (q_{black})) = $(0)_2$, respectively. Table 4 shows the watermark embedding operation on Table 2 using unrank function as illustrated in Algorithm 1 for each partition, respectively. The final watermark value associated with the initial PARTS table (Table 2) is equal to W = $W_{blue} \circ W_{red} \circ W_{black}$ =

Table 2: PARTS (before partitioning).

prts_no	color
1007P	BLUE
1017P	RED
1027P	RED
1030P	BLACK
1032P	RED
1040P	RED
1044P	BLUE
1049P	RED
1051P	BLUE
1055P	RED
1057P	BLACK
1067P	RED
1077P	BLUE
1087P	RED

Table 3: PARTS (after partitioning).

prts_no	color	h_i
1007P	BLUE	1900
1044P	BLUE	1790
1051P	BLUE	1990
1077P	BLUE	1654
1017P	RED	2000
1027P	RED	1799
1032P	RED	1929
1040P	RED	1897
1049P	RED	1700
1055P	RED	1690
1067P	RED	1770
1087P	RED	2790
1030P	BLACK	1999
1057P	BLACK	1970

$(10)_2 \circ (011)_2 \circ (0)_2 = (100110)_2$.

Observe that any change to the initial table can clash against the watermark W, which is embedded into the final table. Applying Algorithm 2 to Table 4 we may easily check that the authentication is preserved.

7 CORRECTNESS

In this section we show the correctness proof of the algorithm discussed so far.

Theorem. Let T be a table over H into K, and let $[v_i] : 1 \leq i \leq N$ be a partition of T. Let π_k be the permutation computed by Algorithm 1 and let $\pi = \pi_1 \circ \pi_2 \circ \ldots \circ \pi_N$ be the compositions of those permutations. Then the ordered set $\langle W_1, W_2, \ldots, W_N \rangle$ is a watermark for T, that can be embedded in the table itself as $\pi(T)$.

Table 4: PARTS (Watermarked).

prts_no	color
1044P	BLUE
1077P	BLUE
1007P	BLUE
1051P	BLUE
1027P	RED
1032P	RED
1087P	RED
1049P	RED
1055P	RED
1067P	RED
1017P	RED
1040P	RED
1057P	BLACK
1030P	BLACK

Proof. Let T be a table containing η tuples and let $P = [v_i] : 1 \leq i \leq N$ be a partition with frequencies $q_i | 1 \leq i \leq N$, where each $q_i \leq \eta$. The group hash value H_{qk} is computed using tuple hash values h_i, for $1 \leq i \leq q_k$ in the set $[v_k]$. All the hash values are computed using a HMAC (HMAC, 2002) function by a secret key $\Re$ known by the database owner. Using H_{qk}, the watermark W_k of length $\ln(q_k)$ is calculated and embedded in $[v_k]$ by Algorithm 1. So, the value of W_k depends only on the $\Re$ and on tuples in $[v_k]$.

Suppose the tuple $t_m \in [v_k]$ is tampered somehow. So the hash value h_m associated to t_m will correspond to a different group hash value H'_{qk} by ideal hash function properties. Since an ideal HMAC is injective, when using the secure key $\Re$ and two different messages M_1 and M_2, we would get HMAC($\Re, M_1$)$\neq$HMAC($\Re, M_1$). Therefore, corresponding to the tampered t_m we would get a different watermark W'_k. When the frequencies q_k are big enough to guarantee that the probability of guessing the permutation π_k is negligible, we get that $\langle W_1, W_2, \ldots, W_N \rangle$ is a watermark according to the Definition 2.4.

8 ROBUSTNESS

We analyze now the probability that all alterations are correctly localized in corresponding groups over a watermarked table. We consider three kind of alterations:

- Modify an attribute value;

- Insert a tuple;
- Delete a tuple.

We assume that partition $[v_k]$ consists of q_k tuples; thus, the length of the embedded watermark W_k for $[v_k]$ is $ln(q_k)$.

Suppose a single attribute value is modified in the watermarked relation. Without loss of generality, assume $r_i.A_j$ is modified (i.e., the j-th attribute of the i-th tuple) in $[v_k]$. The modification will affect the tuple hash h_i, the group hash H_{q_k} and thus the embedded watermark W_k. After the modification, each bit of W_k has equal probabilities to match the corresponding bit of W_k', which is the watermark extracted from the group. Therefore, the probability that the modification can be correctly detected (i.e., $W_k = W_k'$) is

$$Prob_{alt} = 1 - \frac{1}{2^{ln(q_k)}} \quad (1)$$

Let a single tuple be inserted into the watermarked relation. The inserted tuple will be allocated to one of the N partitions $[v_k]$ in the watermark detection. Since the added tuple will affect the group hash as well as the embedded watermark in a random way, the probability that the insertion can be detected (i.e., the watermark verification result is false for the affected group) is

$$Prob_{ins} = 1 - \frac{1}{2^{ln(q_k+1)}} \quad (2)$$

Let a single tuple be deleted from the watermarked relation. Exactly one partition $[v_k]$ will lose the deleted tuple in watermark detection. The absence of the deleted tuple in the group will affect the group hash as well as the embedded watermark in a random way. The probability that the deletion can be detected (i.e., the watermark verification result is false for the affected group) is

$$Prob_{del} = 1 - \frac{1}{2^{ln(q_k-1)}} \quad (3)$$

So in general we conclude that the probability that any alteration on the table is eventually located by the algorithm is $O(1 - \frac{1}{2^{ln(N)}})$, where N is the size (number of tuples) of the table. This ensures that the watermarking techinique satisfies the requirement of Definition 2.4.

9 CONCLUSIONS

The proposed watermarking scheme is group based and a group watermark is associated to a specific order of tuples in that group. The length of the watermark W_k for k-th group is ln(the number of tuples in k-th group).The longer a watermark, the more secure is the scheme. And the final watermark associated with the table will be the concatenated watermark strings associated with each group. The fact that this algorithm is group based, yields to the following strength points:

- We are able to detect and locate modifications as we can trace the group which is possibly affected when a tuple t_m is tampered;
- It is distortion free and invisible as it consists into a permutation of the original table tuples. Therefore no space overhead is required.

Observe that, this watermarking technique can be tuned according to different security levels, by returning the partitioning through the use of multiple attributes instead of a single one.

ACKNOWLEDGEMENTS

Work partially supported by Italian MIUR CONIF '07 project "SOFT".

REFERENCES

Agrawal, Kiernan, and Haas (2003). Watermarking relational data: framework, algorithms amd analysis. The VLDB Journal.

Collberg and Thomborson (2002). Watermarking, tamper-proofing, and obfuscation - tools for software protection. IEEE Trans. Software Engrg.

Cousot, P. and Cousot, R. (2004). An abstract interpretation-based framework for software watermarking. 31st ACM SIGPLANSIGACT symposium on Principles of programming languages.

Cousot, P. and Cousot, R. (2007). Abstract interpretation and application to static analysis . 1st IEEE and IFIP International Symposium on Theoretical Aspects of Software Engineering.

Haan and Koppelaars (2007). *Applied Mathematics for database Professionals.* Apress.

HMAC (2002). The keyed-hash message authentication code. FEDERAL INFORMATION PROCESS STANDARDS PUBLICATION.

Lafaye, J. (2007). An analysis of database watermarking security. 3rd International Symposium on Information Assurance and Security.

Myrvold and Ruskey (2001). Ranking and unranking permutations in linear time. Inf. Process. Lett.

Sion (2004). Proving ownership over categorical data. IEEE International Conference on Data Engineering.