A New Watermarking Approach for Relational Data

Vahab Pournaghshband
University of California, Berkeley
Computer Science Division
Berkeley, CA 94720-1776

vahab@berkeley.edu

ABSTRACT
The importance of digital watermarking for digital assets such as relational databases to preserve their copyrights is becoming more and more important as time goes by. In the past few years, a large number of techniques have been proposed for hiding copyright marks specifically on relational databases. In this paper, we present an effective watermarking technique for relational data that is robust against various attacks. While previous techniques have been mainly concerned with introducing errors into the actual data, our approach inserts new tuples that are not real and we call them "fake" tuples, to the relation as watermarks. We will show that our approach leads to an effective technique that is robust against different forms of malicious attacks as well as benign updates to the data.

Categories and Subject Descriptors
H.2.0 [**Database Management**]: General – *Security, integrity, and protection*

General Terms
Security

Keywords
Watermarking, Relational Database

1. INTRODUCTION

1.1 What is Digital Watermarking?
Watermarking is a form of information hiding with a goal of preserving the copyright of the digital asset such as multimedia, software, and database. There are many discussions on what the characteristics of a robust digital watermarking are. For instance, Petitcolas et al have proposed the following characteristics for a robust watermarking [5.] (1) Marks should not degrade the perceived quality of the work, (2) if multiple marks are inserted in a single relational database, then they should not interfere with each other, (3) if different copies of an object are distributed with different marks, then different users should not be able to process their copies in order to generate a new copy that identifies none of them, and (4) the mark should survive all attacks that do not degrade the work's perceived quality.

Despite the differences on characterizing robust watermarking, they all share a common idea. The key idea behind any sort of digital watermarking is to introduce imperceptible (so that the attacker can not detect them) and tolerable (to ensure that the value of data is not greatly depreciated) errors to the object.

1.2 Digital Watermarking for Relational Data
The main objective of watermarking relational databases is to deter data piracy and protect copyright of relational data. Considering the properties of the relational databases, here we concentrate on three characteristics of the robust watermarking process designed specifically for relational data. These characteristics state that a robust watermarking for relational data should ensure that the attacker is not able to

1. Destroy the watermark without destroying the data
2. Retrieve the original database unless he has access to a similar database
3. The watermark is preserved after benign updates too

In this paper, in subsequent sections, we use the Flight Scheduling Database shown in Figure 1.

Flight Number	Departure	Arrival	Day	Departure Time	Duration	Carrier Type
F102	San Jose	Paris	TU	13:20	11:30	Boeing
B36	Boston	London	MO	16:00	7:10	Airbus
K733	Miami	Brasilia	MO	11:55	8:20	Boeing
L181	Moscow	Cairo	SA	5:30	5:15	Boeing

Figure 1. Flight Scheduling Database

1.3 Previous Related Work
Here we briefly discuss two of the previous approaches related to our work for watermarking relational databases. First, the method given in Agrawal et al[1] utilizes the pseudorandom number generator algorithm to identify the marked tuples and attributes, and also the degree of error to the marked attributes. The private

ACM-SE '08, March 28–29, 2008, Auburn, AL, USA.

key, used for copyright verifiability, is the seed for the pseudorandom number generator algorithm. Figure 2 shows a watermarked version of our Flight Scheduling Database using this approach (marked values are indicated in bold).

Flight Number	Departure	Arrival	Day	Departure Time	Duration	Carrier Type
F102	San Jose	Paris	TU	13:20	**11:00**	Boeing
B36	Boston	London	MO	**16:10**	7:10	Airbus
K733	Miami	Brasilia	MO	11:55	8:20	Boeing
L181	Moscow	Cairo	SA	5:30	**5:25**	Boeing

Figure 2. Watermarked Flight Scheduling Database Using Agrawal et al

Second, is the approach proposed by Zhang et al using embedded images [4.] In other words, in their approach, they embed images into relational database as the watermarks. While these techniques are mainly concerned with introducing errors into the actual data, our approach which will be discussed in Section 2 inserts new "fake" tuples to the relation as watermarks.

2. OUR APPROACH

In our approach, unlike previous approaches we concentrate on tuples with their entirety rather than a subset of their attributes. Our approach aims to generate *fake tuples* and insert them erroneously into the database.

It is a big challenge to figure out what and how many fake tuples should be inserted into the relation. This is because marks should not by any means degrade the quality of the data. For the number of fake tuples, we expect that this number is decided by the database owner. He should use his own judgment to ensure the value of data is not significantly compromised. Regarding creation of fake tuples, although this can be done manually by the database owner which is a viable approach, our effort is to make this process as automatic as possible. Therefore, our goal has been to develop an insertion algorithm that, with little supervision, can effectively generate the fake tuples. In the following subsections, we first discuss our watermark insertion algorithm and then present the watermark detection algorithm and show how it works.

2.1 Watermark Insertion

Our watermark insertion unlike Agrawal et al [1] and Zhang et al[4] is rather probabilistic and uses probability distributions to determine the properties of the mark. Hence, there is no private key used to generate the arks into the relation. Note that the watermarking insertion presented in this section requires some information that is expected to be provided by the database owner.

The following is the pseudocode for the watermark insertion:

---

0. $InsertWatermark(\Re, N', A, \bigcup_{a_i \in A}\{(p_i, F_i)\})$
1. *for* N' *tuples*
   {
2. *for* $c_i \in C$
3. $\tau.c_i = GenerateCandidateKey(c_i, \Re)$
4. *for each* $a_i \in A$
   {
5. *if* $(Bernoulli(p_i) == 1)$
6. $\tau.a_i = UniformSampler(F_i)$
7. *else*
8. $\tau.a_i = BiasedSampler(a_i, \Re)$
   }
9. $\Re.insert(\tau)$
   }

An explanation of the above pseudocode is as follows:

0. The parameters required by the DB owner are:
   R: the relation
   N': the number of fake tuples
   A: the set of attributes that are not candidate keys
   $\{(p_i,F_i)\}$: set of ordered pairs corresponding to each element a_i in A;
   for each ordered pair corresponding to a_i in A:
   p_i: sensitivity of the attribute a_i; the sensitivity is a value between 0 and 1 inclusive where 0 means insensitive and 1 means highly sensitive. (default: 0)
   F_i: set of fake values for attribute a_i (default: {})
1. It assures generating N' tuples as requested by the DB owner
2&3. C is the set of candidate keys and the *GenerateCandidateKey* procedure is expected to return a unique value for a candidate key using some pattern recognition algorithm to ensure consistency and ultimately leading to imperceptability.
4. for each attribute that is not a candidate key:
5. *Bernoulli*(p_i) runs a Bernoulli sampling with a probability p_i of returning 1 (i.e. success) to decide whether to pick a fake value for τ_i or share a value with some other tuple(s).
6. *UniformSampler* takes a set of fake values and picks a value from that set uniformly. For instance, the distribution for $F_{departure}$ = {Razan, Keroona, Bilbao} would be 1/3 for each value.
7. *BiasedSampler* takes the relation R and an attribute a_i; it first determines the distinct values for a_i in R and construct the *BiasedSampler* by the relative frequency of each value for a_i. For instance, for the attribute "carrier type" in the flight scheduling database, the distribution for the *BiasedSampler* is constructed as P(Boeing) = 3/4 and P(Airbus)=1/4.

2.1.1 Primary Key Collision

If the user has set the database such that once there is a collision, the tuple is replaced by the new one, then it is fine since it will simply delete the fake tuple and includes the new real tuple. The subtlety arises when the user has set the database to not permit overwriting. In this case, our approach would fail since it would occasionally not let the real tuple to be inserted to the relation. The fact that the primary key ranges are very wide for most applications makes the chances of occurring such collisions very rare.

Figure 3 shows the watermarked flight scheduling database by adding fake tuples. The added fake tuple which is indicated as bold shows a flight departing from a small city in Western Iran and arriving in a small town in central Ghana.

Flight Number	Departure	Arrival	Day	Departure Time	Duration	Carrier Type
F102	San Jose	Paris	TU	13:20	11:30	Boeing
B36	Boston	London	MO	16:00	7:10	Airbus
K733	Miami	Brasilia	MO	11:55	8:20	Boeing
L181	Moscow	Cairo	SA	5:30	5:15	Boeing
F127	**Razan**	**Apam**	**MO**	**16:00**	**8:20**	**Boeing**

Figure 3. Watermarked Flight Scheduling Database

2.2 Watermark Detection

The following is the pseudocode for the watermark detection (where its parameters are: the relation (R), the set of fake tuples initially inserted into R ($\{\tau\}$), and the similarity score (σ).

```
DetectWatermark(ℜ, {τ}, σ)
{
    for each r ∈ ℜ
    {
        for each τ ∈ {τ}
        {
            if (similarity(r, τ)) ≥ σ)
                return true
        }

    return false
}
```

The above watermark detection algorithm looks for a fake tuple in the suspicious relation. However, the exact match might not be found either by the attacker changing the value of some attributes as an attempt to destroy the watermark or by the benign user who changes these attributes repeatedly over time. To identify these and suspect piracy correctly, we run a similarity measurement on these tuples seeming to be the same. For the attacker it is important not to destroy the data, so the ones with higher sensitivity are less likely to be destroyed. Therefore, incorporating sensitivity of attributes while calculating the similarity score, seems reasonable. The similarity score between two tuples could be calculated as follows:

$$S = \sum_{i=1}^{n} w_i \cdot x_i$$

Where n is the number of attributes,

$$x_i = \begin{cases} 1 & a_i\text{'s match} \\ 0 & otherwise \end{cases},$$

and

$$w_i = \frac{p_i}{\sum_{j=1}^{n} p_j}$$

(where p_i is the sensitivity of the attribute a_i; p_i =1 for the candidate keys).

If the score is higher than a static threshold, σ, then it suspects piracy.

Unlike other algorithms, our detection algorithm is not an *inverse* algorithm to the watermark generating algorithm. In fact, our algorithm checks to see whether our tuples exist or has been changed. It checks it via primary key. As soon as it finds one match (i.e. identical or similar tuples), detection is done. Note that the detection will fail for the watermarked database when all of the fake tuples are deleted by benign deletions. However, we believe that this is a very unlikely scenario.

Comparing the previous approaches to ours, our insertion algorithm is probabilistic (thus, every time the insertion algorithm is run, it produces a different output) as opposed to those approaches that produce deterministic errors. On the other hand, our detection algorithm is relatively deterministic compared to approaches like Agrawal et al's which is probabilistic and is normally not confident about the existence of a piracy.

3. ANALYSIS OF ROBUSTNESS

3.1 Basic Attacks

Basic attacks simply attempt to destroy the watermark by changing the values of the attributes. The similarity measurement discussed in section 2.2 ensures robustness against these attacks.

3.2 False Claim of Ownership

This could happen when the attacker falsely claims that he owns the watermarked relation by adding his own watermarks, either by introducing errors in some attributes or by adding his own fake tuples. In both cases, we win. We know the original data and we can confidently claim that all the attacker's marks are included in

the set which contains whatever has been altered from the original database (inserted tuples or changes in attributes). On the other hand, the attacker is not able to confidently come up with such a set.

3.3 Updatability

As discussed in Section 1.2, we know how important it is for a watermark to persist after benign updates. Agrawal et al[1] introduced the notation of incremental updatability where any changes to the marked attributes would be instantly remarked to preserve the watermark. However, there are risks and drawbacks involved in this approach that could be potentially problematic. There is a risk of update attack where the attacker makes an exhaustive updates over all tuples and their attributes and point out the marks by observing the result of remark process. Our approach, on the other hand, watermarks the relation only once, and therefore it is robust against this attack.

In addition to the update attacks, applying the idea of incremental updatability requires reimplementation of DBMS operators including update, insert, and delete operators. While this is not a risk, it could be a drawback since all database systems must incorporate the idea into their operators for this to ensure persistency of watermark over updates. Our approach, however, does not require the reimplementation of DBMS operators.

3.4 Sensitivity

In previous approaches, sensitivity of each attribute is assumed to be either Boolean (sensitive or insensitive which indicates that the attribute can be watermarked or not) [1] or is ignored completely (that is, the entire relation is assumed to be tolerable to some universal degree of error) [4]. Our approach, however, assumes a degree of sensitivity for each attribute.

3.5 Multiple Public Verifiability

As Li et al [3] claims, all existing watermarking schemes for relational databases are secret key based, requiring a secret key to be presented in proof of ownership (for instance, in Agrawal et al [1] the secret key is the seed of the pseudorandom number generator). 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. In our approach, only one or a very few fake tuples are need to convince the court, so the ownership can be publicly verified more than once until all the fake tuples are revealed.

3.6 Proof of Ownership

Assume that a database is watermarked by introducing errors to numerous attributes of a subset of tuples in the relation. Now the question is how to prove to the court that what they have intentionally marked is erroneous and the corresponding values in the original database are correct. While this seems to be a feasible task, it is in fact very burdensome. Back to our flight scheduling example, it is probably harder to convince the court that the maximum altitude of flight L181 is 20,800 feet and not 16,250 than proving there is no flight going from Razan to Apam. Proving ownership in the previous methods is even more burdensome since we have to go through this process for every single mark made in the relation.

3.7 Subset of Attributes Attack

In the previous approaches, only a subset of attributes could be candidates for watermarking due to various reasons. Candidate keys as well as non-numerical and highly sensitive attributes are possible reasons for an attribute not to be marked. However, the risk here is that the attacker can take these attributes and be confident that there is no mark and hence lead to false claim of ownership.

4. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a new approach using fake tuples and discussed the insertion and detection watermarking algorithms in details. Our goal has been to ensure that our watermarking approach satisfies the three key elements of any watermarking relational data (as discussed in section 1). While we discussed our approach's robustness analytically, we did not evaluate the approach quantitively. In fact, evaluating watermarks for relational database is a challenge and requires further consideration. However, the persistency of the watermark after both malicious and benign updates, as a subproblem, might be evaluated by acquiring access to a log of user queries on a particular database over a reasonably long period of time, and then run the log on the watermarked database and observe whether the watermark detection algorithm will confirm the watermark. While this evaluation process sounds plausible, it is application-specific and may not be generalized very well.

2. REFERENCES

[1] Agrawal, R., Haas, P., and Kiernan, J. 2003. Watermarking relational data: framework, algorithms and analysis. *The VLDB Journal* 12, 2 (Aug. 2003), 157-169. DOI= http://dx.doi.org/10.1007/s00778-003-0097-x

[2] Li, Y. and Deng, R. H. 2006. Publicly verifiable ownership protection for relational databases. In *Proceedings of the 2006 ACM Symposium on information, Computer and Communications Security* (Taipei, Taiwan, March 21 - 24, 2006). ASIACCS '06. ACM, New York, NY, 78-89. DOI= http://doi.acm.org/10.1145/1128817.1128832

[3] Zhou, X., Huang, M., and Peng, Z. 2007. An additive-attack-proof watermarking mechanism for databases' copyrights protection using image. In *Proceedings of the 2007 ACM Symposium on Applied Computing* (Seoul, Korea, March 11 - 15, 2007). SAC '07. ACM, New York, NY, 254-258. DOI= http://doi.acm.org/10.1145/1244002.1244066

[4] Zhang, Z., Jin, X., Wang, J., Li, D. 2004. Watermarking Relational Database Using Image. In *Proceedings of the Third International Conference on Machine Leaning and Cybernetics*, (Shanghai, August 26 – 29, 2004).

[5] Petitcolas, F. A., Anderson, R. J., and Kuhn, M. G. 1998. Attacks on Copyright Marking Systems. In *Proceedings of the Second international Workshop on information Hiding* (April 14 - 17, 1998). D. Aucsmith, Ed. Lecture Notes In Computer Science, vol. 1525. Springer-Verlag, London, 218-238.

[6] Sion, R., Atallah, M., and Prabhakar, S. 2003. Rights protection for relational data. In *Proceedings of the 2003 ACM SIGMOD international Conference on Management of Data* (San Diego, California, June 09 - 12, 2003). SIGMOD '03. ACM, New York, NY, 98-109. DOI= http://doi.acm.org/10.1145/872757.872772

[7] Yong, Z., Xia-mu, N., Khan, A., Qiong, L., and Qi, H. 2006. A novel method of watermarking relational databases using character string. In *Proceedings of the 24th IASTED international Conference on Artificial intelligence and Applications* (Innsbruck, Austria, February 13 - 16, 2006). V. Deved, Ed. International Association Of Science And Technology For Development. ACTA Press, Anaheim, CA, 120-124.

[8] Anderson, R. J. 2001 *Security Engineering: a Guide to Building Dependable Distributed Systems*. 1st. John Wiley & Sons, Inc.

[9] Fu, Y., Jin C., and Ma, C. 2007. A Novel Relational Database Watermarking Algorithm. Intelligence and Security Informatics. Lecture Notes in Computer Science, vol. 4430. Springer, 208-219.