World Applied Sciences Journal 19 (9): 1273-1277, 2012
ISSN 1818-4952
© IDOSI Publications, 2012
DOI: 10.5829/idosi.wasj.2012.19.09.1865

Self-Constructing Fragile Watermark Algorithm for Relational Database Integrity Proof

Sajid Iqbal, Azhar Rauf, Saeed Mahfooz, Shah Khusro and Sajid H. Shah

Department of Computer Science, University of Peshawar Khyber Pakhtunkhwa, Pakistan

Abstract: Database watermarking has gained a keen interest of researchers since the coin was tossed by Agarwal. Since then many approaches have been adopted to protect the copyright of relational databases. The concept was proposed after getting inspiration from watermarking digital assets. Watermarking has been helpful in integrity check of relational databases as well and so far techniques have been proposed for integrity check and temper detection. In the paper, an algorithm is proposed called Self-Constructing Fragile Watermark (SCFW) for integrity check of relational databases. Proposed scheme consists of highly dependent watermark information. Any tempering or leakage is reflected over the watermark information. Experiments show that proposed technique is affective against temper detection. Even a little amount of tempering is reflected efficiently because of the self-replicating nature of proposed algorithm.

Key words: Fragile Watermark · Integrity · Relational Databases copyright and ownership

INTRODUCTION

Piracy and ownership conflict became a major issue when organizations started sharing their digital assets for research and business developmental purposes. To cope with this problem, the concept of digital, imperceptible, undetectable and intangible watermarking [1] was proposed to prove the ownership. Inspired from digital watermark, researchers started to work on the protection of databases from piracy and leakage. The need of database watermarking started as many organizations and industries wanted to publish their databases for research purposes as well as sharing their statistics for the proof of prosperity. The concept of database watermarking is basically a two steps process: Watermarking Insertion and Watermark Detection. Watermark insertion is hiding of information in least significant bits (LSB) of numeric attributes of some records. These numeric attributes are selected by a pre-defined watermarking function [2-5]. Later on if an ownership conflict occurs, the real owner can prove his ownership by detection of embedded watermark. The embedded watermark can be used to identify leakage of secret data or prove integrity of database.

Relational database watermarking can be broadly categorized into two groups: Robust and Fragile. Robust watermarking is meant for the purpose of ownership proof. They are supposed to be strong enough so that even if a small portion of database is copied or violated against copyrights, the watermark pattern can still be found. On the other hand, fragile watermark performs the purpose of integrity proof. They are deliberately made fragile so that a small change in a database causes the destruction of watermark information and identifies any leakage. Fragile watermarking algorithms are designed as such to minimize the time of embedding and detection.

Fragile watermarking schemes are basically meant for checking and maintaining the integrity of databases. The owner who wants to send his database to another party or provides his database for publication purposes, protects his database by embedding watermark information. Owner or receiver can check the integrity of his database later by simply running his watermark detection algorithm. If the detection algorithm regenerates the bit-string which was originally embedded by owner's algorithm the database is in its original form otherwise tempered.

Corresponding Author: Sajid Iqbal, Department of Computer Science, University of Peshawar Khyber Pakhtunkhwa, Pakistan.

This paper proposes a novel method for integrity check of database by a mechanism of embedding self constructing fragile watermark information. By self constructing we mean that the secret key is provided by the owner of database and this key is then responsible for the creation of dependent keys. This dependency of the embedded keys makes the approach efficient as any tempering or changes are reflected in the entire relation. Experiments show the effectiveness and strength of the proposed mechanism. Rest of the paper is organized as: section 2 describes related work. Section 3 is about the proposed insertion and detection watermark algorithms. Section 4 shows experimental results and analysis, section 5 is conclusion and finally section 6 gives a brief comparison of proposed work and existing technique.

Related Work: The protection of relational databases with watermark was first proposed by [1]. Since then approaches have been made to protect relational databases against both tempering and copyright violation.

The leakage identification by the use of a shadowed watermark technique is proposed in [6]. Their mechanism consists of a main key and shadowed key. Main key is responsible for the generation of main watermark information and shadow key is responsible for the generation of shadow watermark information. The database to be watermarked is sent to a trusted watermark server (TWS) which actually embeds the watermark information to the database. Main key is contributed by the owner of database and shadow key is contributed by TWS which is delivered later to the user. In case of any leakage identification, the owner checks the main key against every data holder. If the detection of main watermark key is successful, the user associated with concerned database is considered as an accused.

One obvious problem with shadow watermark is that there are chances of accusing an innocent user. The tempering can be caused by a malicious attacker. Furthermore including a third party server can cause overhead during the watermark insertion process.

[7] proposes tamper detection algorithm which is based on predictive difference of Support Vector Regression (SVR). They divide the attributes set into two parts. One of the selected attributes is called object value and remaining attributes are feature value for SVM training. The selected attribute generates Huffman code and produces minor pay load. This minor pay load is used for the detection of tampering. Authors do not embed any key rather the information of original database is used as watermark information. This technique does not cause distortion to original values.

Proposed Algorithm: The proposed scheme, self-constructing fragile watermark (SCFW) is used for the integrity checking. It mainly consists of a primary watermark bit-string W_P which is private to the owner. This primary watermark is responsible for the generation of secondary watermark information W_S and similarly secondary watermark is responsible for the generation of tertiary watermark W_T depending on the choice of owner and fraction of database selected for watermark insertion. Table 1 shows the notations used in the proposed algorithm.

Watermark Insertion Algorithm: When a database owner has to send a copy of his private data to a remote user he simply passes a key K_P which is private to owner and fraction of the database ξ to be marked to watermark embedding function F_E. The watermark embedding function on the basis of provided parameters, divides the table logically into three equal groups and starts marking the first group 'g_1' with the primary watermark information 'K_P'.

Table 1: Notations Used

K_P	Owner Defined Key
ξ	Fraction of Tuples to be marked
W_P	Primary Watermark Bit-String
F_E	Watermark Embedding Function
W_S	Secondary Watermark Key
W_T	Tertiary Watermark Key
R_W	Watermarked Relation
v	Candidate Attributes for watermark
o-bit	Bit from attribute to be marked
w-bit	Bit from watermark information

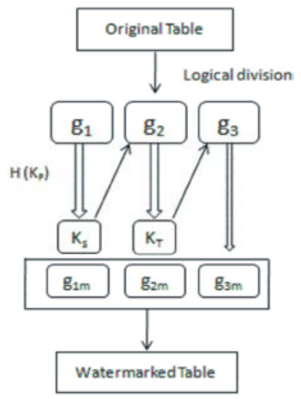

Fig. 1: Watermark insertion algorithm

During the insertion process of watermark information to the first group 'g_1', the bits which are replaced are stored in an array. These bits serve the purpose of watermark information for the second group 'g_2'. Once the insertion process of 'g_1' is completed, insertion to the 'g_2' is started from replaced bits of 'g_1'. Similarly watermark is embedded to the 'g_2' and all the replaced bits are stored in an array as they are to be embedded in 'g_3'. After the successful embedding of 'g_2', stored bits are embedded to the 'g_3' and in this way all groups are watermarked with dependency. The technique is shown in Figure 1.

The proposed Embedding algorithm is described as following:

Algorithm 1: Watermark Insertion Algorithm

```
//Insert a Watermark information String to relation R and return watermarked relation $R_W$
//Parameters $K_P$ and $\xi$ are private to the owner

• Calculate primary watermark information

$W_p = H(K_p)$

• Divide R into three groups: R-> R($g_1$, $g_2$, $g_3$)
• For each tuple t belongs to $g_i$ do
• s = H ($K_P$ concatenated t.P_Key)
• If (s mod $\xi$ equals zero) then
• Attr_index i = s mod v
• Bit_index j = s mod 2
• Mark bit = (O_bit XOR W_bit)
• Append O_bit to $W_{i+1}$
• If not within usability
• Rollback
• Else commit and go to step 3
• Exit and return $R_W$
```

The parameters K_P and ξ are private to the owner. First of all the private key K_P of owner is encrypted using a one way hash function which is SHA-1 [8] in our case. Then the relation to be marked is divided into three equal group g_1, g_2 and g_3. The hashed key of the owner is embedded to the first logical group g_1. In step 5 a tuple is checked if it is to be marked or not. The selection of tuple and attribute is done by the hash value applied to the private key of owner and primary key of under consideration tuple [5]. If a tuple qualifies for watermark then a specific attribute is selected among candidate attributes to be marked. Finally at step 7, least significant bit is selected to be marked among available LSB for marking, which in our proposed technique are two.

Bits which are marked during the embedding process of first group g_1 are stored and serve the purpose of watermark information for the marking of second group g_2 and similarly the LSBs of g_2 marked during marking process serve the purpose of watermark information for g_3. Once the marking process of a tuple is done, it is checked for the usability. If marked value is within usability range, the operation is committed otherwise roll backed.

Watermark Detection Algorithm: In order to check the integrity of database or detect our inserted watermark information, watermark detection algorithm is run against the relational database. If detection algorithm retrieves embedded watermark information successfully from all the three groups and the bit sequence retrieved from g_1 and the detected bit sequence matches the sequence which was originally embedded to them, the detection is successful otherwise not. Detection algorithm is given as:

Algorithm 2: Watermark detection Algorithm

```
//Algorithm to return a watermark string W[] from a marked relation $R_W$
// Parameters $K_P$ and $\xi$ are private to the owner

• Initialize detected watermark string $W_i$[ ]
• For each tulple t belongs to group gi do
• s = H (Ki concatenated t.P_Key)
• If (s mod $\xi$ equals zero) then
• Attr_index i = s mod v
• Bit_index j = s mod 2
• Targeted_bit XOR $K_P$
• Append bit value to $M_I$ [ ]
```

Experiments and Analysis: A number of experiments are carried out to find the detection rate of tempering. Experiments were performed on a machine with 2.0 GHz of Intel Dual core CPU, 2 GB of RAM and running windows XP as operating system. The database that we used in our experiments is TPC-E [9] which consisted of total of 14 attributes out of which 8 are numeric and 6 are non-numeric and one primary key attribute. All of the 8 numeric attributes were considered as candidate attributes for marking. We used our insertion watermark for different value of ξ and each time the results proved

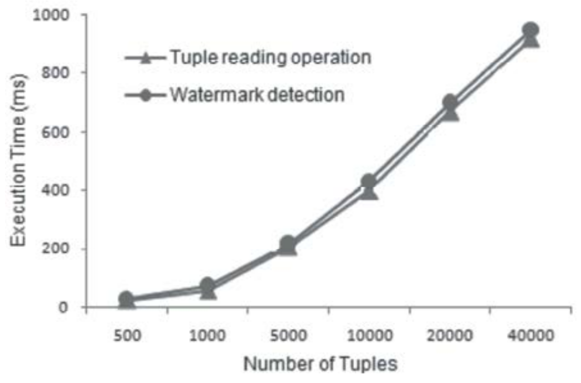

Fig. 2: Execution time comparison between ordinary tuple reading and watermark detection

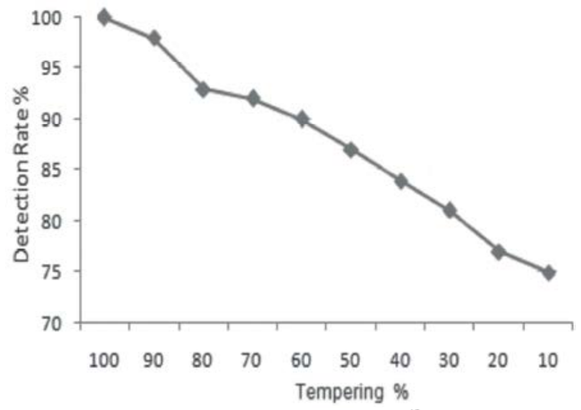

Fig. 3: Detection rate of tempering for ξ = 40%

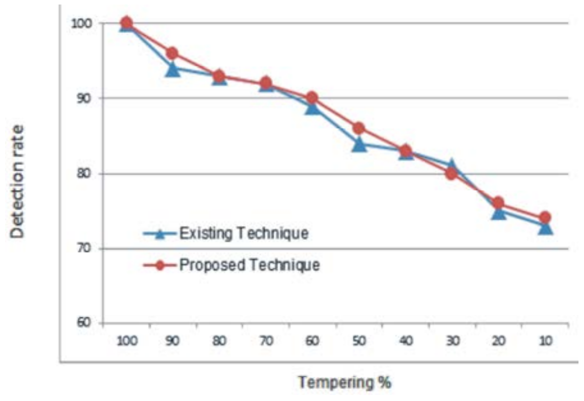

Fig. 4: Comparison of existing and proposed techniques

to be efficient. The LSBs were set to 2. As the rate of distortion produced by the first and second LSB is same. We setup two types of experiments. First we compared the time required for watermark detection against the ordinary database reading operations. Second we compared the detection rate of our embedded watermark information against different level of tempering.

Figure 2 shows the comparison between execution time of ordinary tuple selection and detection rate of embedded information. Different number of tuples was selected and queries were run against them to calculate the execution time. Each experiment was run many times and then the average value was taken. Figure 3 shows the detection rate for different percent of tempering. It is clear from experiments that the proposed technique is effective against the temper detection and only a small amount of tempering to the relation is reflected and detected efficiently.

Comparison: This section provides a comparison of existing technique [6] and our proposed technique. The comparison of both the existing and proposed technique shows that our technique has shown better results at several occasions. Readings show that upon tempering of 90%, 70%, 50%, 20% and 10% the proposed technique has shown better results. The graphical comparison is shown in Figure 4.

CONCLUSION

In this paper we propose a novel self constructing fragile database watermarking (SCFW) technique for the integrity check of relational database. In the proposed scheme, we embedded a secret key which is private to the owner. This key is then responsible for the generation of other watermark information as we are embedding three different keys which are strongly dependent on each other. This dependency makes the watermark more fragile because tempering to any one of keys causes tempering of the dependent keys. Experiments show that the proposed technique is effective against tempering of watermark information at any level as it is reflected in all groups of the relation.

REFERENCES

1. Agrawal, R. and J. Kiernan, 2002. Watermarking relational databases. In the Proceedingsof the 28th VLDB Conference, Hong Kong, China, pp: 155-166.
2. Yue, K., P. Chen, Y. He and X. Chen, 2009. A Cluster-Based Watermarking Technique for Relational Database. First International Workshop on Database Technology and Applications, Dbta, pp: 107-110.

3. Jiang, C., X. Chen and Z. Li, 2009. Watermarking Relational Databases for Ownership Protection Based on DWT. In the Proceedings of Fifth International Conference on Information Assurance and Security, pp: 305-308.
4. Dong, X., X. Li, G. Yu and L. Zheng, 2009. An algorithm resistive to invertibility attack in watermarking relational databases. IEEE Control and decision conference, China, pp: 1532-1537.
5. Hu, Z., Z. Cao and J. Sun, 2009. An Image Based Algorithm for Watermarking Relational Databases. In the Proceedings of International Conference on Measuring Technology and Mechatronics Automation, pp: 425-428.
6. Xian, H. and D. Feng, 2009. Leakage Identification for Secret Relational Data Using Shadowed Watermarks. In the Proceedings of International Conference on Communication Software and Networks, pp: 473-478.
7. Wu, H., F. Hsu and H. Chen, 2008. Tamper Detection of Relational Database Based on SVR Predictive Difference. Eighth International Conference on Intelligent Systems Design and Applications, pp: 403-408.
8. The keyed-hash message authentication code (HMAC), http://www.csrc.nist.gov/publications/fips/fips198/fips-198a.pdf.
9. TPC-E Benchmark Specification, 2008. [Online]. Available: http://www.tpc.org.