SEMI-FRAGILE WATERMARKING SCHEME FOR RELATIONAL DATABASE TAMPER DETECTION

***Saeed Arif Shah*[1], *Imran Ali Khan*[2*], *Syed Zaki Hassan Kazmi*[3], *Fariza Hanum Binti Md Nasaruddin*[4]**

[1]Saudi Electronic University Riyadh, Kingdom of Saudi Arabia

[2]COMSATS University Islamabad, Abbottabad, Pakistan

[3]University of AJK, Pakistan

[4]Faculty of Computer Science and Information Technology, University of Malaya, 50603, Kuala Lumpur, Malaysia

Email: s.shah@seu.edu.sa[1], imran@cuiatd.edu.pk[2*] (corresponding author), zaki.mzd@gmail.com[3], fariza@um.edu.my[4]

DOI: https://doi.org/10.22452/mjcs.vol34no1.1

ABSTRACT

Most of data over the Internet today is hosted on outsourced third-party servers which are not trusted. Sometimes data is to be distributed to other organizations or individuals for pre-agreed use. In both of these scenarios data is susceptible to malicious tampering so there is a need for some mechanism to verify database integrity. Moreover, the authentication process should be able to differentiate between valid updates and malicious modifications. In this paper, we present a novel semi-fragile watermarking scheme for relational database integrity verification. Besides detection and localization of database tampering, the proposed scheme allows modifications to the data that need periodic updates, without requiring re-watermarking. Watermark embedding is distortion free, as it is done by adjusting the text case of selected data values resulting in retention of semantic meaning of data. Additionally, group-based embedding ensures the localization of tampering up to group level. We implemented a proof of concept application of our watermarking technique. Theoretical analysis and experiments show that even a single value modification can be detected with very high probability besides detection of attacks like tuple insertion and tuple deletion.

Keywords: ***Semi-fragile watermark, Relational Database Security, Integrity***

1.0 INTRODUCTION

Use of Internet servers to host data for online availability has increased tremendously during past years. Due to increased hardware and management cost, data owners prefer to outsource data storage and management requirements to third party servers [1]Third party servers host multiple owners' data on comparatively very low cost. The aforesaid approach solves the affordability problem but creates issue of database integrity [1][2][3]. Third party servers may not meet the security requirements and hence can compromise on client's data integrity. Similarly, when data is to be distributed to individuals or organization for some defined purpose the data owner want to verify the distributed data integrity against potential tampering. Both these scenarios raise the need for integrity verification and tamper detection of digital data.

Digital watermarking can be used for tamper detection in digital contents. Fragile watermark particularly is used for this purposed, due to its property that embedded watermark changes if there is any tampering done to the host data. On the contrary robust watermark survives different types of attack like modification, deletion, sorting etc. so is used for copyright protection and owner verification. Robust watermark based schemes have been proposed for copyright protection and owner verification of relational database [4]–[10]. Most of these methods work only for numeric data. For tamper detection and integrity verification of relational data, there are few fragile watermarking schemes proposed so far [1], [11], [12]. To the best of our knowledge, Gou's scheme [1] is most relevant published work in this domain; therefore, it will be our focus for comparison.

Previously proposed fragile watermarking schemes have certain limitations. The most significant limitation of fragile watermarking schemes is their complete fragileness i.e., a change of even a single bit is detected as tampering. Updating the watermarked data, even by the owner is regarded as tampering [1], [11], [12], [17], [18]. In such situation re-watermarking is required after every update made by authorized user. This re-watermarking is not a feasible solution for the data that is already stored elsewhere or distributed to other parties. In real world applications, there may be a portion of data that may need periodic updates or sometimes owner may want to update some small fraction of the data; such modifications are legal so should not be detected as tampering. Rest of the data which does not require frequent updates or the data which needs protection against any sort of modification there must be some a mechanism for tamper detection and localization while allowing necessary updates.

Considering all of the aforementioned issues, we propose a semi fragile watermarking scheme for authentication of relational database which allow updates to the part of data that needs frequent updates while detecting all sorts of tampering in rest of the data. Our method does not require re-watermarking after allowed/legitimate modification. The scheme works by calculating a semi-fragile watermark from the contents to be authenticated and hiding it in the same data without introducing any semantic distortion in the cover data.

Rest of the paper is organized as follows. Section 2 discusses the related works. In Section 3 proposed scheme is described, where preliminaries followed by embedding and detection algorithms are presented. Section 4 presents some distinguished features of our scheme. Security analysis and experimental results are presented in Section 5 and 6, respectively. Section 7 concludes the paper.

2.0 RELATED WORK

Work on relational database can be classified into two main categories namely, robust watermarking and fragile watermarking. Robust watermarking is for copyrights protection while fragile watermarking is for database integrity verification.

In [5] Agrawal *et.al.* presented their pioneer database watermarking scheme for owner verification. Based on the assumption that there is some numeric data which can tolerate small amount of error, authors proposed an algorithm to embed watermark in LSB's of securely selected attribute values. Authors established a good framework for relational database watermarking, but simple LSB manipulation for hiding watermark can lead to significant distortion in the database, leaving it less useful. The watermark cannot survive simple attacks, such as bit shifting and rounding.

Sion *et al.*[9], proposed a robust watermarking approach of hiding watermark in numerical data. Initially, tuples are divided into partitions by using marker tuples. Later a single bit is embedded into each partition by manipulating its statistics. This scheme comparatively introduces low distortion in the data and is robust against various attacks, such as subset attack and rounding attack. However, it is affected by tuple deletion and insertion attacks that cause synchronization error.

David Gross *et al.*, also proposed a watermarking scheme for query preserving relational data[13]. Scheme claimed to be query preserving for local queries. To embed watermark, first local queries are identified and then the selected data values are modified while preserving the queries results. The scheme only provides copyrights protection solution.

In[8] Shehab *et al.*, proposed an optimization based watermarking scheme for numerical data. First the relational data watermarking problem is expressed as constrained optimization problem which is then solved using Genetic algorithms. After partitioning data into subsets, the distribution of each partitions is modified to embed a single bit. Embedding is carried out by solving the optimization problem for either maximization or minimization. The technique provides an effective solution for numerical data as it introduces minimum distortion to the data and is more robust against different attacks like sorting, insertion deletion.

Mustafa Bilgehan Imam oglu et. al.[14] proposed a reversible database watermarking approach with firefly optimization algorithm. Watermark embedding is done into specially selected tuples using Difference expansion watermarking DEW[15]. For optimized distortion minimization caused by watermark insertion and to enhance the capacity they employed firefly algorithm. Firefly algorithm determines suitability of attributes for watermark embedding within the selected tuples. Watermark extraction algorithm extracts the embedded watermark data from the database and compares it with the authentic one. The scheme is reversible as the algorithm additionally reverses the watermarked database to original after watermark extraction and verification. The scheme, however, is applicable to numeric data only.

Few other robust watermarking schemes have been proposed so far like [6], [10] but those are applicable to numeric data only.

S. Iftikhar et. al. in [16] proposed a robust and reversible watermarking scheme for relational data. For watermarking, feasible (candidate) feature from the database are selected using mutual information *MI*. Features selected to watermark are analyzed and ranked by defining secret parameters in data preprocessing phase. By employing GA based optimization scheme an optimum watermark string is created. Later on, the watermark is embedded in the selected feature(s) using parameter β which is the optimized value from the GA and ηra change matrix. In decoding phase preprocessing step is performed again; decoding is done using *MI*, β and ηr to recover the watermark. The scheme is semi blind as it needs some information for recovery.

There are only a few schemes found in literature on relational database tamper detection using fragile watermarking[11][12][17][18]. Since the Guo's [1] work is most relevant work to our proposed scheme, therefore, we discuss it in detail. In Guo's solution, initially tuples are partitioned into different groups, and then two watermarks are constructed for each group; one watermark for every tuple and other for every attribute using one-way hash of most significant bit (MSB) of respective tuple/ attribute. These watermarks are then embedded in that tuple/attribute's least significant bits (LSB). Two LSBs are used for watermark embedding, one for tuple and the other for attribute watermark thus creating a grid. This approach provides attribute level tamper detection and localization but has certain shortcomings. Firstly, it only works for numeric data. Although the authors proposed an alternate embedding method for categorical data that suggests the use of tuple order for watermark embedding[12] but that suffers from tuple sorting operation, which is detected as tampering, though it is a legal operation. Secondly, to embed watermark two least significant bits (LSB's) are used that introduce significant distortion in the data that is not resilient enough. For database, semantic value of data is more important than multimedia data and sometimes change of a single bit may leave it useless. Examples are Name, phone number, account number, age, price etc.

Another fragile watermarking scheme is proposed by Wu *et. al.* in[11]. By employing Support Vector Regression (SVR), content characteristics are obtained; which are encoded using Huffman encoding and then saved as payload information separately. The payload information is later used for tamper detection. Storage of payload information separately makes this scheme non-blind. Blindness, however, is an important characteristic of a watermarking based solution. Moreover, the scheme is applicable to numeric data only.

To the best of our knowledge there is no semi-fragile watermarking solution for tamper detection in categorical relational data presented so far, although one can see a good amount of work for multimedia data [19]–[23] but that is not applicable to relational data. Our proposed scheme addresses above mentioned issues with following contributions:

- The scheme allows the owner to verify the integrity of database by detecting malicious modifications made to the database, while allowing necessary updates.

- Watermark is embedded using the case of selected text that does not change the meaning of data. Therefore, we can claim that our proposed scheme is distortion free query preserving watermarking scheme.

The proposed scheme is applicable to non-numeric data specifically alphanumeric data. Watermark embedding and extraction process inside a single group is shown in figures 1 and 2 respectively.

3.0 PROPOSED SCHEME

The proposed solution has three parts. The first part defines the framework for the decisions regarding which portion of the document is to be treated as fragile against tampering and in what part of the document modification may be allowed. In the second part watermark calculation and embedding is done. The last part is about authentication and tamper detection in the suspected watermarked document.

3.1 Preliminaries

The main idea of our solution is to construct a semi-fragile watermark in such a way that could allow legal database updates while detecting malicious ones. In Previously proposed schemes [1],[5] watermark was calculated using hash values of all of the tuples in any particular group. This tuple hash value is calculated using Hashed Message Authentication Code (HMAC) that is a one-way hash function. The inputs of HMAC are secret key *K* and all of the attribute values in selected tuple. Given the same input message and the key, it would return the same value every time [24], [25] Any change to the input will randomize the output of HMAC[1].

We can obtain a semi-fragile watermark if we could define some criteria for attributes selection and their inclusion in tuple hash calculation. For this purpose, first we define three attributes categories, namely Sensitive Attributes (*S*), Non-Sensitive (*NS*) attributes, and Semi Sensitive (*SS*) attributes.

"*S*" category contains attributes that do not need to be updated once they are recorded and any modification in these attributes must be detected during verification, such as personal data (Name, Date_of_Birth, SSN, or in case of some product: ProductID, Manufacturing date etc.). Non-Sensitive are those attributes that may require updates, more or less frequently, for instance in case of inventory system; Items_in_stock etc. Third category namely Semi Sensitive (*SS*) attributes are those that allow constrained updates. These are the attributes whose values can be changed within certain boundaries e.g. in case of address of a person, a change of address may be allowed without changing state and country or in case of phone number we can allow to update phone number without changing country code.

Let $R = \{Pki, A1, A2 \dots An\}$ be a relation with primary key Pk_i and other attributes A_j (j=1…n). We can define subsets of *R* as *S* (Including Pk), *NS* and *SS* such that:

$$S \cup NS \cup SS = R \quad and \tag{1}$$

$$S \cap NS \cap SS = \varphi \tag{2}$$

As an illustration, consider the following relation named "**Student**":

Student *{ID_Number, Name, Birth_Date, Nationality, Phone, Address, Courses_Passed}*
Here we can define three categories as follows.
Since ID, Name, Birth date and nationality mostly does not change once recorded and require no updates so we can categorize them as sensitive:

S= {ID_Number, Name, Birth_Date, Nationality}

Number of course passed need to be updated on term-to-term basis so it may be categorized as NS. Phone number and Address may be placed in Semi-sensitive category, because a person may change his phone number, but country code would be the same, similarly a person may change his temporary address, but his state/country would be unchanged.

NS= {Courses_Passed}
SS= {Phone, Address}
Next step is to define some constraints denoted by CT for semi sensitive attributes.
Let CT_1 and CT_2 be constraints for attributes Phone and Address, respectively then CT1 and CT2 can be stated as follows:
CT1: allow changes in phone except country code
Example: Phone=0086**********

Here "*" denotes the updateable part.
CT2: allow to change address except state and country
Example: Address=********, ********, ********, California, USA.
Once this categorization and constraint definition is done a semi-fragile watermark can be calculated using the tuple hash. The tuple hash is calculated using SHA based HMAC, which is seeded with secret key *K*, sensitive attributes set S, and generalized values of SS (obtained after applying constraints). Non-sensitive attributes are not included in tuple hash calculation.
Hence, the tuple hash will be:

$$T_H = HMAC\left(K||S||SS\ (Generalized)\right) \tag{3}$$

The data block for above HAMC consists of set of sensitive attributes(S) and generalized semi-sensitive attribute (SS) defined earlier in this section.

3.2 Watermark Embedding

The watermarking process consists of three phases, namely preprocessing, watermark calculation and watermark embedding. A description of watermark embedding, and detection is given in Algorithms 1 and 5, respectively. Note that the algorithms are presented in a simplified way for better understanding.
Phase one consists of preprocessing steps, where all tuples are partitioned into "*g*" number of groups, each with almost equal number of tuples using the same partitioning algorithm employed by some of the earlier schemes[1].

Partitioning is done simply by taking mod of primary key hash of each tuple and the "*g*"; remainder of this operation defines the corresponding group of the tuple.

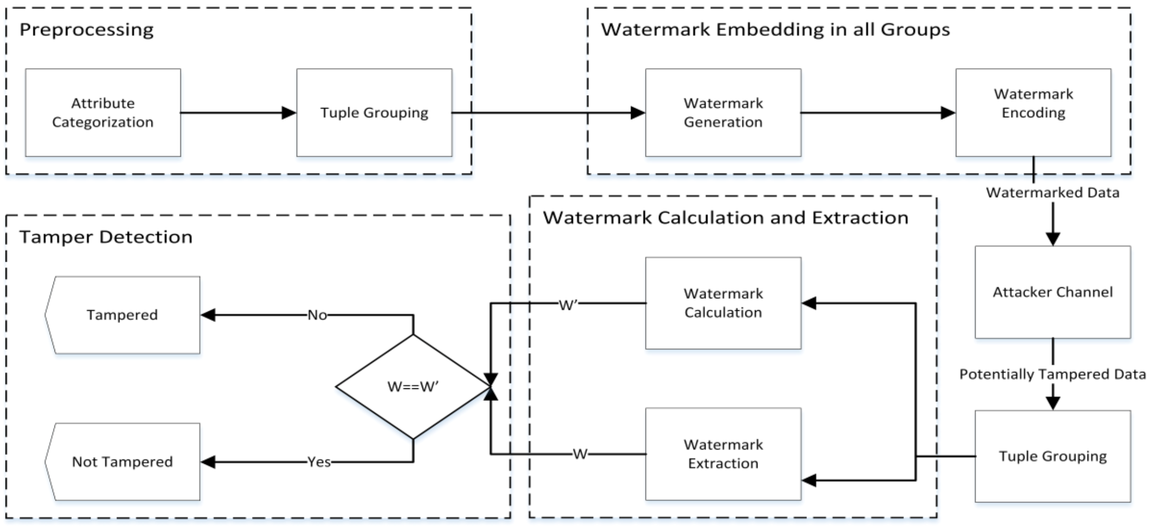

Fig. 1: Watermark Embedding and Tamper Detection Process

Table 1: Notations and parameters

Notation	Description
r	Row or record of a relation
K	Secret key known only to owner
1/m	Fraction of tuples to be watermarked
V	Number of watermarkable attributes
G	Number of groups
T_{Hi}	Tuple Hash Value of ith tuple
ℓ	Size of watermark
T	Number of tuples in a group
W	Embedded Watermark
Wm	Extracted watermark from marked data

Table 2: Terms and Abbreviation

Terms	Description
HMAC	Hashed Message Authentication Code
LSB	Least Significant Bit
SHA	Secure Hash Algorithm
MSB	Most Significant Bit

In the second phase watermark is calculated in three steps namely C1, C2, C3 for every group separately. To maintain synchronization tuples are securely sorted using *pkHashvalues* within each group in step C1 and then tuple hash is calculated using HMAC in C2. Using tuple hash values, group hash is calculated in C3 that gives us the candidate watermark. The size/length of watermark depends on number of attributes selected for watermark embedding. It is important to note that the SHA1 returns 160 bits HMAC output. If the number of bits required to be embedded is less than 160 than algorithm extracts required number of MSB; otherwise cyclically appends bits from HMAC (starting from MSB) till the desired watermark length is achieved[1]. For embedding watermark bit, we used case of selected attribute [26]. This method does not alter the contents of database while watermark embedding so it is a distortion free embedding.

Note that before tuple hash calculation in C2 text case is transformed to either upper or lower, that is a virtual operation not actually committed against database. The same transformation is done while calculating tuple hash during detection. This ensures that any case alteration would not be detected as tampering

3.3 Database Tamper Detection

The watermarked data may be exposed to attacks intending to modify database contents. In order to verify the

[1]Alternately, if one decides to keep group size large enough then SHA-256 or SHA-512 can be used.

integrity of watermarked database, we need to authenticate it for potential tampering.

```
Algorithm 1 Watermark Embedding
Input: R, K, m
  • Partitions of attributes into subset: S, SS, NS
  • Defined generalization for Semi Sensitive attributes.
Preprocessing
Grouping: Securely divide all tuples into groups using primary key hash (pkHash)
  i. $pkHash_i$ = HMAC $(K, Pk_i)$ //$Pk_i$ is the primary key of ith tuple)
  ii. Place tuples with same value of "pkHash mod g" in one group "$G_k$"
(Repeat following two phases for each group "$G_k$")
Watermark Calculation
C1. Sort tuples in group $G_k$ in ascending order according to their pkHash values
C2. Calculate tuple hash for every tuple in the group (after case normalization)
        $T_{Hi}=HMAC(S, SS$ (Generalized))
C3. Calculate Group Hash
    $Hg = HMAC(K, T_{H1}, T_{H2}, \cdots, T_{Ht})$
C4. Constructs candidate watermark by converting "$Hg$" into bit sequence
Watermark Embedding
(Repeat E1 through E4 for every tuple in the group)
E1. Select tuple $r_i$ from $G_k$ if its pkHash mod m=0
E2. Select the attribute $r_i.A_j$ within tuple $r_i$ where j= $pkHash$ mod $v$
E3. Extract subsequent watermark bit as $W[i]$ from the same group's candidate watermark and embed it in selected attributes using rules defined in table 3.
E4. Increment $\ell$ for each embedded bit
```

Table 3

Single bit embedding rules
Rule-I: To embed watermark bit "1"
Change text case of $r_i.A_j$ to title case //If not already so
Rule-II: To embed watermark bit "0"
If $r_i.A_j$ is a single word then change $r_i.A_j$ case to lowercase
If $r_i.A_j$ is multiword sentence then change case to sentence case

For authentication, we must know the secret key "*K*" and number of groups "g". For tamper detection, each group is authenticated independently. Using parameters *K* and *g,* the watermark *W,* which is supposed to be embedded is constructed first. Then *W';* the watermark actually present in the database is extracted. Algorithm 2 outlines the steps involved in detection process.

In step D1 and D2 the watermark *W* is constructed using the same procedure adopted during embedding. The W′ is extracted in step D3 by examining the text case of securely selected data values using the same set of rules we followed while embedding. For instance, if the examined data has title case then extracted watermark bit would be "1".

```
Algorithm 2 Watermark Detection
Input: R(marked), K, m, $\ell$
D1. Grouping
Using the same key K and number of groups "g", securely divide all tuples into groups as was done in Preprocessing step P2.
(Repeat Steps D2 to D4 for every group)
D2. Construction of Watermark W
Watermark of length $\ell$ constructed using the same process used in embedding steps C1 through C4
D3. Extract Watermark W′
D3.1 Securely select tuples and attribute //same as in embedding steps E1 & E2.
  D3.2 Extract $W'$ by examining text case of selected data values
 D4. Watermark verification
If $W == W'$ Then data is not tampered
Otherwise the particular group has been tampered.
```

In step D4 comparison is made between the constructed watermark W (the originally embedded), with the extracted watermark W'. If they both match to each other than we can claim that the data is not tampered with. If someone has maliciously tampered database contents than that will change W and a mismatch will confirm the tampering as well as its location that which group has been tampered.

4.0 DISCUSSION

In this section, some distinguished features of our scheme are discussed, particularly those related to semi-fragileness, updateability, query preservation and watermarking of non-numeric data. At the end of this section, a comparison with Guo's scheme is presented.

4.1 Semi-Fragile Watermark

We calculate watermark from the group hash that is the output of HMAC. In case of some malicious attack, any change to input leads to change in the watermark. Since carefully identified non-sensitive and semi sensitive attributes are excluded from watermark calculation so any change to these attributes will not affect watermark this gives us semi fragile watermark.

4.2 Data Updates

Sometimes the database may need to be updated after watermarking. Since our watermark is semi-fragile therefore, it allows modification of the attributes which need frequent update and identified earlier as semi sensitive and non-sensitive. Any other update that is made to sensitive data is detected as tampering. If modifications are done against data that is already defined as semi-sensitive and non-sensitive, there is no need of re-watermarking, as these updates never harm the watermark. However, if some value/values from sensitive attributes or sensitive part of semi-sensitive attribute are modified then we have to re-watermark the data. Since every group is watermarked independently so, only the group containing updated data would need re-watermarking. This way re-watermarking can be avoided for the whole database and it is one of the advantages of grouping.

4.3 Query Preserving embedding

Watermark embedding is done by adjusting the case of selected data according to predefined rules that does not change the meaning of data so query result will not be affected even after embedding.

4.4 Solution for Non-Numeric Data

Our scheme is applicable to non-numeric (specifically alphanumeric) data as compared to Gou's [1] scheme that works for numeric data only. For numeric data, we propose to embed watermark in secretly selected LSBs as proposed by Agrawal [5]. In that case, we only need to modify our algorithms slightly. For instance, in algorithm 1 we need to modify C2 and E3. In C2, just ignore the LSBs while calculating tuple hash, whereas in E3 embed watermark in those ignored LSBs using Agrawal's [5] method instead of using rules in table 3.

4.5 Comparison with Guo's scheme

A comparison with Guo's work shows that our scheme has following significant features:

1. Proposed scheme use semi-fragile watermark while Guo's scheme uses fragile watermark.
2. In Guo's method use of two LSB's for embedding watermark considerably changes the meaning of data, whereas, our embedding algorithm does not change data value. This feature also ensures the query preserving support of our scheme that is not provided by Guo's method.
3. The Guo's technique only works for numeric data whereas our algorithm is applicable to database which has at least one non-numeric attribute.
4. The proposed scheme however does not work for the relational data which consists of numeric attributes only.

5.0 SECURITY ANALYSIS

5.1 Threat Model

The objective of our scheme is to detect malicious modification made to database, while allowing some legal updates. The goal of an attacker would be to make desired changes to watermarked data without being detected, specifically without disturbing the watermark. In this section, we show that in our scheme, error rate in detection of tampering like attribute data modification, tuple insertion and tuple deletion is very low.

5.2 Secure Embedding and Verification

An important requirement for fragile watermarking system is the prevention of un-authorized verification and embedding. Since in our scheme, whole process of grouping and tuple/attribute selection is controlled by a key-based HMAC therefore only authorized persons can embed or verify the watermark.

5.3 Single Value Modification Detection

Tampering made to any sensitive attributes data value will result in randomization of tuple hash that will change the group hash. Since the embedded watermark "W" is extracted from group hash, therefore, the watermark will also be randomized. For illustration assume that $r_i.A_j$ (ith attribute of jth tuple) is modified, this will randomize tuple hash T_{Hi} and the group Hash Hg. After modification each bit of W has equal probability to be the same or not. In other words, each bit of embedded watermark W has equal probability to match the correspond bit of extracted watermark W'.

We can define the probability of error as probability that W matches W'even after modification of watermarked data i.e. P_{ERROR} is the probability that $W=W'$ after tampering. Therefore

$$P_{ERROR} = \frac{1}{2^{\ell}} \qquad (4)$$

where "ℓ" is watermark length

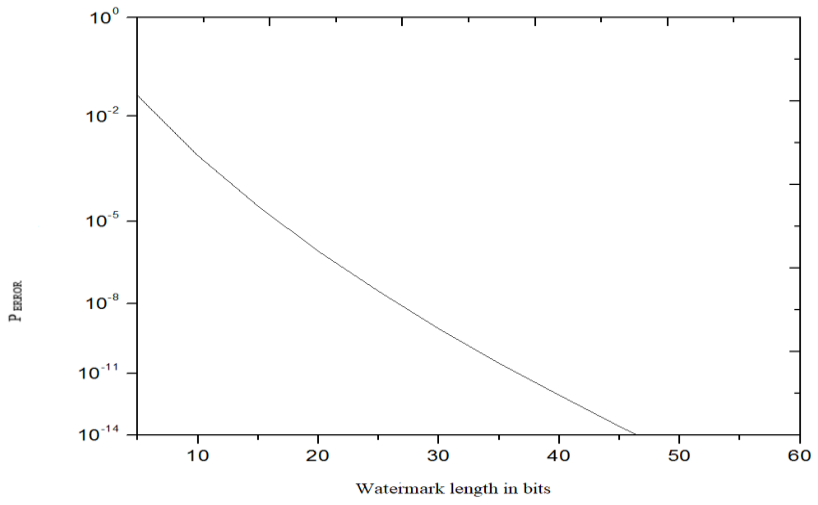

Fig.2: Value Modification Detection Error

Fig.2 shows the probability of error in the detection of single value modification for varying watermark length. We can see the probability of error in this case is very low and it decreases exponentially with increasing watermark size.

5.4 Tuple Insertion Detection

Since the inserted tuple will affect group in which it falls and consequently the watermark of that group will be randomized. There are two possibilities for newly inserted tuple; it may be added to marked tuples, thus increasing the watermark length, or it may fall in unmarked tuples category. With $'\ell'$ tuples, marked among a total of t tuples, the probability that a tuple is marked is

$$P_{\text{marked}} = \frac{\ell}{t} \qquad (5)$$

Now the probability of error in detection of new tuple insertion can be formulated as follows

$$P_{\ ERROR} = \frac{\ell+1}{t+1}\left(\frac{1}{2^{\ell+1}}\right) + \frac{t-\ell+1}{t+1}\left(\frac{1}{2^{\ell}}\right) (6)$$

The first term in equation (6) represents the probability of error in tamper detection if the inserted tuple falls into the marked tuples and the second term gives the probability that inserted tuple fall in unmarked tuples category.

In Fig.3 we plotted error rate given in equation 6 in detection of tuple insertion for two different group sizes (t=100 and t=50). Along x-axis, we varied watermark size (in bits). We see that in this case error rate also decreases exponentially. We also note that small group size is slightly better than large group size for correct verification of malicious tuple insertion. However, from this we cannot deduce that large group size increases the error rate, from equation 6 it is clear that major factor involved here is the watermark size (length). Therefore, by increasing ℓ we can achieve same results.

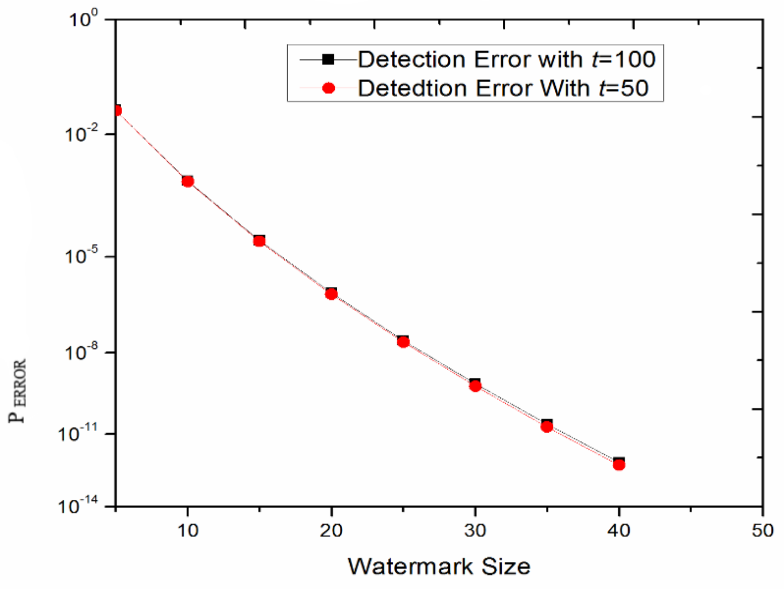

Fig. 3: Tuple Insertion Detection Error

5.5 Tuple Deletion Detection

A deleted tuple also randomizes the embedded watermark because of randomized group hash resulted due to this deletion. The deleted tuple may be a marked tuple (containing watermark bit) or not. If the deleted tuple is marked one, then the size of watermark will decrease otherwise number of unmarked tuple decreases by one. In this scenario the error rate in detection of tuple deletion is given in equation 7.

$$P_{ERROR} = \frac{\ell-1}{t-1}\left(\frac{1}{2^{\ell-1}}\right) + \frac{t-\ell-1}{t-1}\left(\frac{1}{2^{\ell}}\right) \qquad (7)$$

Fig.4 shows that chances of error in detection of deleted tuple are very low; this is due to the fact that change of even a single bit in watermarked data results in randomization of tuple hash, group hash and subsequently the watermark calculated from group hash. We observe from the analysis that watermark size is an important factor for reduction in probability of error. The watermark length should be in a reasonable ratio of the group size. Based on above investigation and our experiments we recommend it should not be less than 25% of total number of tuples in a group.

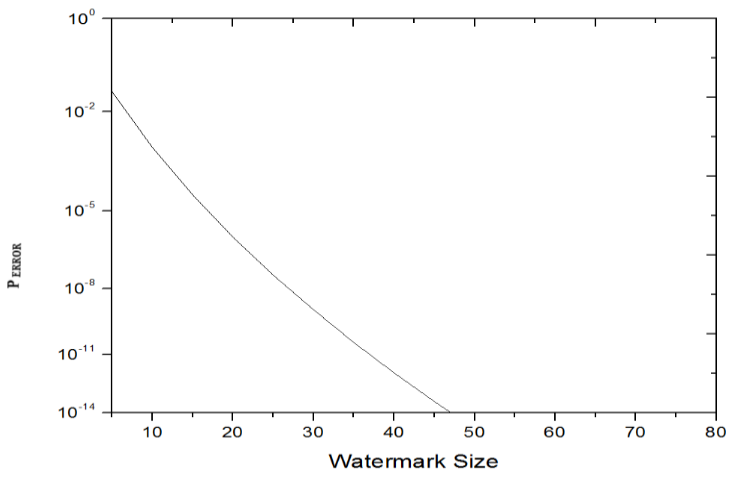

Fig.4: Error in Deleted Tuple Detection

Experimental Results

5.6 Relation Between '*g*' and '*t*'

The experiments discussed in this section analyze the detection rate against varying watermark and group sizes, therefore we first explain the relationship between number of groups 'g' and group size 't'. The hash function we used evenly distributes all tuples into 'g' number of groups, thus the group size t depends on g. For larger values of g there will be more number of groups and hence the group size 't' will be smaller. For instance, if there are 100 tuples in a relation, then for g=10, *t* would be around 10 and similarly for g=20 *t* would be around 5. Therefore, to vary group size we actually vary the number of groups.

5.7 Experimental Setup

For proof of concept and experiments, we built software using c# with SQL-server 2005 database platform. Experiments were conducted on Intel core-2 duo CPU 1.7 GHz with 2 GB ram. We used real life data from "US Medicare Plan 2008" database having more than one million records. For testing purpose, only a subset of the database is used. The experiments were performed for three kinds of data tampering attacks namely multiple value modification, multiple tuples insertion and multiple tuples deletion. For each kind of experiment, the group size varies from 25 to 150 with an increment of 25. We kept the watermark length equal to 40% of group size, which means watermark also varies with group size. Each experiment was repeated multiple times and results were averaged and recorded as detection rate. It is important to note that the detection rate is complementary to$Perr$. Detection rate is the percentage of correctly detected modification whereas $Perr$ is the probability of error in correct detection of tampering. We can see that these results are consistent with our previous analysis.

5.8 Multiple Value Modification

For this attack we performed two different experiments. In first experiment, we randomly modified 10 to 90 percent of total tuples 'n' for varying group length *t* and recorded their detection rate. Only sensitive data items were modified in this case. From the results shown in table 3 we can see that detection rate is very high and only a few modifications are missed. With larger groups and watermarks 100% detection rate can be achieved. In second experiment, keeping all other parameter the same, we randomly modified all types of data i.e. sensitive, semi sensitive and non-sensitive. In this case the detection rate is lower than that of first experiment as shown in table 4(b). The reason is obvious, as in this case some modified data values might belong to non-sensitive or semi sensitive category. This shows the semi fragileness of the scheme. Our experiments show that if only non-sensitive data is modified than detection rate is almost zero.

Table 4(a): Detection Rate (%) for Multiple Values Modification (Sensitive data only)

Modified Tuples (%)	Group Size *t*					
	25	50	75	100	125	150
10	99	100	100	100	100	100
30	100	100	100	100	100	100
50	100	100	100	100	100	100
70	100	100	100	100	100	100
90	100	100	100	100	100	100

Table 4(b): Detection Rate (%) for Multiple Values Modification (All data)

Modified Tuples (%)	Group Size *t*					
	25	50	75	100	125	150
10	60	79	86	91	99	100
30	72	80	94	99	100	100
50	88	88	96	100	100	100
70	100	100	100	100	100	100
90	100	100	100	100	100	100

5.9 Multiple Tuple Deletion

Table 4 shows the result of detection rates for multiple tuples deletion. Here we tested our algorithm with varying *t* and deletion rate that is the ratio between total tuples and the randomly deleted tuples. We see that when the number of deleted tuples is small, all the deletions are correctly detected but as we increase deletion rate, detection rate decreases significantly. We observed that for *t* equal to 25 and deletion rate equal to 90%, detection rate is only 5. This is obvious, as with 90% deleted tuple we are only left with a few tuples and there is a high possibility that most of the group were completely deleted. However, by increasing group size and watermark we can achieve 100% detection rate even with high deletion rate.

Table 5: Detection Rate (%) for Multiple Tuple Deletion

Deleted Tuples	Group Size *t*					
(%)	25	50	75	100	125	150
10	100	100	100	100	100	100
30	90	100	100	100	100	100
50	60	98	100	100	100	100
70	25	75	100	100	100	100
90	5	20	50	80	95	100

6.0 CONCLUSIONS AND FUTURE WORK

In this paper, we presented a novel semi-fragile watermarking scheme for integrity verification of relational database. A semi-fragile watermark calculated from database content has the ability to detect malicious database tampering besides allowing legal modifications. Moreover, the watermark embedding is distortion free and embedded watermark does not change the semantic meaning of data which makes it a query preserving watermarking method. Beside detection, we can also locate tampering position up to group level. Theoretical analysis shows that detection probability of even minor tempering, such as value modification, tuple deletion and insertion is very high, and it is very difficult for an attacker to modify database without altering embedded watermark. Our scheme supersedes existing techniques in terms of semi fragileness and query preservation for categorical data. In the future, we would like to develop a semi-fragile watermarking solution for streaming data, try to optimize the attribute categorization procedure proposed in our scheme.

REFERENCES

[1] H. Guo, Y. Li, A. Liu, and S. Jajodia, “A fragile watermarking scheme for detecting malicious modifications of database relations,” *Inf. Sci. (Ny).*, vol. 176, no. 10, pp. 1350–1378, 2006.

[2] M. K. Walia, M. N. Halgamuge, N. D. Hettikankanamage, and C. Bellamy, “Cloud computing security issues of sensitive data,” in *Handbook of Research on the IoT, Cloud Computing, and Wireless Network Optimization*, Hershey, PA, USA: IGI Global, 2019, pp. 60–84.

[3] E. Bertino, “Data security and privacy in the IoT,” in *Advances in Database Technology - EDBT*, 2016, vol. 2016-March, pp. 1–3.

[4] C. Constantin, D. Gross-Amblard, and M. Guerrouani, “Watermill: an optimized fingerprinting system for highly constrained data,” *Proc. 7th Work. Multimed. Secur. - MM&Sec ’05*, p. 143, 2005.

[5] R. Agrawal, P. J. Haas, and J. Kiernan, “Watermarking relational data: framework, algorithms and analysis,” *VLDB J.*, vol. 12, pp. 157–169, 2003.

[6] M. Kamran, S. Suhail, and M. Farooq, “A robust, distortion minimizing technique for watermarking relational databases using once-for-all usability constraints,” *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 12, pp. 2694–2707, 2013.

[7] Y. Li, H. Guo, and S. Wang, “A multiple-bits watermark for relational data,” *J. Database Manag.*, vol. 19, no. 3, pp. 1–21, 2008.

[8] M. Shehab, E. Bertino, and A. Ghafoor, “Watermarking relational databases using optimization based

techniques," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 1, pp. 116–129, 2008.

[9] R. Sion, M. Atallah, and S. Prabhakar, "Rights protection for relational data," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 12, pp. 1509–1525, 2004.

[10] V. Khanduja, O. P. Verma, and S. Chakraverty, "Watermarking relational databases using bacterial foraging algorithm," *Multimed. Tools Appl.*, vol. 74, no. 3, pp. 813–839, 2013.

[11] H. C. Wu, F. Y. Hsu, and H. Y. Chen, "Tamper detection of relational database based on SVR predictive difference," in *Proceedings - 8th International Conference on Intelligent Systems Design and Applications, ISDA 2008*, vol. 3, pp. 403–408.

[12] Y. Li, H. Guo, and S. Jajodia, "Tamper detection and localization for categorical data using fragile watermarks," *DRM 2004 Proc. Fourth ACM Work. Digit. Rights Manag.*, pp. 73–82, 2004.

[13] D. Gross-amblard, L. C. Cnam, S. Martin, and P. Cedex, "Query-preserving Wa termarking of R elational D atabases and XML D ocuments," pp. 191–201, 2003.

[14] M. B. Imamoglu, M. Ulutas, and G. Ulutas, "A new reversible database watermarking approach with firefly optimization algorithm," *Math. Probl. Eng.*, vol., 2017.

[15] A. M. Alattar, "Reversible watermark using the difference expansion of a generalized integer transform," *IEEE Trans. Image Process.*, vol. 13, no. 8, pp. 1147–1156, Aug. 2004.

[16] S. Iftikhar, M. Kamran, and Z. Anwar, "RRW - A Robust and Reversible Watermarking Technique for Relational Data," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 4, pp. 1132–1145, 2015.

[17] B. A. Hamida, R. F. Olanrewajub, and K. Lumpur, "A zero-distortion fragile watermarking scheme to detect and localize malicious modifications in textual database relations," in Applied Information Technology, vol. 84, no. 3, pp. 404–413, 2016.

[18] A. Khan and S. A. Husain, "A fragile zero watermarking scheme to detect and characterize malicious modifications in database relations," in *The Scientific World Journal*, 2013, vol. 2013.

[19] Q. Wang, G. Cheng, and H. Liu, "A semi-fragile watermarking algorithm against geometric transformation for vector geographic data," in *ACM International Conference Proceeding Series*, 2017, pp. 11–15.

[20] S. Saha, D. Bhattacharyya, and S. K. Bandyopadhyay, "Security on fragile and semi-fragile watermarks authentication," *Int. J. Comput. Appl.*, vol. 3, no. 4, pp. 23–27, 2010.

[21] M. A. Nematollahi, M. A. Akhaee, S. A. R. Al-Haddad, and H. Gamboa-Rosales, "Semi-fragile digital speech watermarking for online speaker recognition," *EURASIP J. Audio, Speech, Music Process.*, vol. 2015, no. 1, p. 31, Oct. 2015.

[22] X. Yu, C. Wang, and X. Zhou, "Review on semi-fragile watermarking algorithms for content authentication of digital images," *Futur. Internet*, vol. 9, no. 4, pp. 1–17, 2017.

[23] C. Ling, O. Ur-Rehman, and W. Zhang, "Semi-fragile watermarking scheme for H.264/AVC video content authentication based on manifold feature," *KSII Trans. Internet Inf. Syst.*, vol. 8, no. 12, pp. 4568–4587, 2014.

[24] Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno. Cryptography Engineering: Design Principles and Practical Applications. 2010, Wiley Publishing.

[25] Stallings, William. Cryptography and Network Secuirty, Fifth Edition. Vol. 139. N.p., 2011. Network. Web.

[26] S. A. Shah, S. Xingming, H. Ali, and M. Abdul, "Query preserving relational database watermarking," *Inform.*, vol. 35, no. 3, pp. 391–396, 2011.