Kuwait J. Sci. **43 (2)** pp. 121-138, 2016

Prefix oriented N4WA coding scheme for improved tampering detection in relational data

Waqas Haider[1], Muhammad Sharif[2], Hadia Bashir[2], Mudassar Raza[2,*], Mussarat Yasmin[2]

[1]*School of Engineering and Information Technology, ADFA@UNSW, ACT, Canberra Australia*

[2]*Department of Computer Science, COMSATS Institute of Information Technology, WahCantt, Paksitan*

[*]*Corresponding Author: Email: mudassarr@gmail.com*

Abstract

In this paper, a memory efficient N4WA coding scheme (breifed in section 3.1) is presented to detect tampering and restoring modified data values in relational data. The existing watermarking schemes for tampering detection do not provide necessary components, such as accurate localization and restoration in cooperation. These flaws degrade the quality and dependability of the tampering detection scheme. However, in the proposed scheme, firstly the compressed code W is generated from original data M, using N4WA encoding and stored at certification authority CA. Secondly, the assumed suspicious data M' is given to N4WA encoding for generating another compressed code W'. Meanwhile, the genrated W and W' are compared to detect and accurately localize tampering in M'. Finally utilizing localization information T_c^r, W and N4WA decoding assist in restoring the tampered data values. Investigational outcomes indicate that for the database relations of all data types, the minimum to maximum alterations are detected, localized and restored successfully.

Keywords: Lossless compression; relational data restoration; tampering attacks; zero watermarking.

1. Introduction

In order to facilitate tampering detection and copyright protection services of digital contents, digital watermarking is considered as a tool, which comprises variety of information hiding schemes. The watermarking schemes were introduced for multimedia contents by Chen *et al.* (2005). However, in the last decade, scientists have applied the notion and intent of watermarking in relational databases (RDBs) to develop tampering detection and copyright protection services.

The watermarking schemes for RDBs are classified as both robust and fragile (e.g. tampering detection schemes). Robust schemes are aimed at copyright protection (Agrawal *et al.*, 2003; Al-Haj & Odeh, 2008; Cao, 2008; Guo *et al.*, 2006; Guo *et al.*, 2006; Gupta & Pieprzyk, 2009; Hu *et al.*, 2005; Huang *et al.*, 2004; Li & Deng, 2006; Sion, 2004; Sion *et al.*, 2005; Zhang *et al.*, 2004; Zhou & Li, 2008; Zhou *et al.*, 2007) whereas fragile schemes are intended fortampering detection and integrity proof of relational data (Abdel-Hamid & Tahar, 2008; Bhattacharya & Cortesi, 2009a; Bhattacharya & Cortesi, 2009b; Gao *et al.*, 2013; Guo *et al.*, 2006; Guo, *et al.*, 2006; Halde & Cortesi, 2010; Halder *et al.*, 2011; Halder *et al.*, 2010; Kamel, 2009; Khan & Husain, 2013; Li *et al.*, 2004; Prasannakumari, 2009; Rashidi, 2010; Tsai *et al.*, 2007; Tsai *et al.*, 2006).

The factors related to quality and dependability of tampering detection scheme are as follows:

1. Accuracy of locating error
2. Restoration of tampered values
3. No injection of signature in the original content of the data
4. Memory consumption

In order to achieve the third factor, a number of distortion free tampering detection schemes have been presented in the literature for RDBs. Among them, no scheme has cell level localization and restoration capability, nor is it memory efficient. Therefore, the proposed scheme is designed to accomplish all the four factors presented above. The paper has the following sequence: The literature review about distortion-free tampering detection schemes is given in section 2. Section 3 demonstrates the proposed distortion free tampering detection scheme for RDBs. Experimental results are elaborated in section 4 for different tampering attacks and concluding remarks are presented in section 5.

2. Literature review

The existing distortion-free tampering detection schemes have been conferred by several scientists (Bhattacharya & Cortesi, 2009a; Bhattacharya & Cortesi, 2009b; Gao *et al.*, 2013; Halder & Cortesi, 2010; Halder *et al.*, 2011; Halder *et al.*, 2010; Kamel, 2009; Khan & Husain, 2013; Li *et al.*, 2004). In the scheme which is presented by Li, Guo *et al.*(2004), all tuples are divided into groups using a secure key. In these groups infusion and verification of watermark have been done for each group autonomously. However, tuple positions are utilized as a cover for watermark. The approach has no restoration or accurate localization features. Also, it is not applicable for RDBs having multiple data type attributes. In the scheme presented by Bhattacharya &

Cortesi (2009a), watermark is developed by grouping the tuples using a secure key, calculating hash of each group and then generating permutations of tuples. At the same time, tuples reordering is utilized as a watermarking cover, but the scheme has no restoration or accurate localization features. Likewise, in Bhattacharya & Cortesi (2009b) approach, the binary image of actual data content is extracted as watermark. Then any change to actual content of the data reflects modification in the watermark. Nevertheless, this scheme also lacks accurate localization and restoration features. Moreover, in the schemes proposed by (Halder & Cortesi, 2010; Halder *et al.*, 2011; Halder *et al.*, 2010), the notion and intent of distortion-free fragile watermarking are applied for the protection of intellectual property of the circuit design. Although Halder's schemes address persistency issues related to watermark, they do not provide restoration feature. In addition, in the scheme that is presented by (Kamel, 2009) a virtual operation is used to generate and embed watermark. Even though the scheme can serve attributes of all data types in RDBs, it has no localization or restoration capability.

In Khan & Husain (2013) recent scheme, the watermark constitutes the concatenation of local characteristics of numeric data, including length, digit count (frequency of all data values) and range. The concatenated watermark is encrypted using a secure key and stored at third place for tampering verification. Although this approach provides tampering detection as well as characterization capability for numeric data, it has no restoration feature. Similarly, in the work presented by Gao *et al.*(2013), error correction codes (ECCs) are utilized for detecting, locating and restoring tampering in relational database. In this scheme, Reed Solomon codes are utilized to generate watermark. Although generated watermark assists with tampering detection, localization and restoration, the scheme can protect selective or sensitive attributes of the whole database. Interestingly, the major limitation of this approach is regarding the memory resource utilization as Reed Solomon codes are a part of ECCs and were developed for reliability purposes, not for memory efficiency. Further more, this scheme provide tuple's group level localization and consequently lacks accuracy.

In the above discussion, four important issues have been identified in the existing fragile watermarking schemes for RDBs. First, distortion-free fragile schemes do not provide necessary components, such as localization and restoration in cooperation. These issues degrade tampering detection scheme's dependability or reliability. The proposed scheme provides tampering detection along with cell level localization and restoration of tampered data values. Second, some of the fragile schemes are limited to specific data types which ultimately affect the scalability. In contrast, the proposed scheme can serve all data types, in the RDBs, including text, numeric, alpha numeric, along with its feature to convert M in to the binary form. Third, distortion oriented

fragile schemes which have been mentioned in many papers (Guo *et al.*, 2006; Guo *et al.*, 2006; Prasannakumari, 2009; Rashidi, 2010; Tsai *et al.*, 2007; Tsai *et al.*, 2006) compromise the quality, integrity and usability of RDBs. For instance, the real world sensitive database applications like vehicle registration, citizen registration etc. cannot bear distortion. Therefore, such schemes are not applicable to certain relational database applications. Finally, (Gao *et al.*, 2013) scheme provides tampering detection, group level localization and restoration features, but it lacks accuracy in tampering detection and utilizes more memory if applied upon the whole database. In contrast, the proposed scheme is applicable to the whole database and utilizes minimum memory resources as reflected in Table 2 (Section 4.3).

3. Proposed work

The proposed tampering detection scheme exhibits the following properties:

1. It is distortion-free to maintain data quality and usability.
2. It provides cell level (attribute value level) localization capability.
3. It has restoration capability.
4. It can serve the attributes of multiple data types in RDBs.
5. There is no need for original database relations during verification.
6. The approach is memory efficient if compared with Gao *et al.*, 2013 work (see Table 2 in Section 4.3).

The overall proposed scheme for the improved tampering detection in RDBs is given in Figure 1.

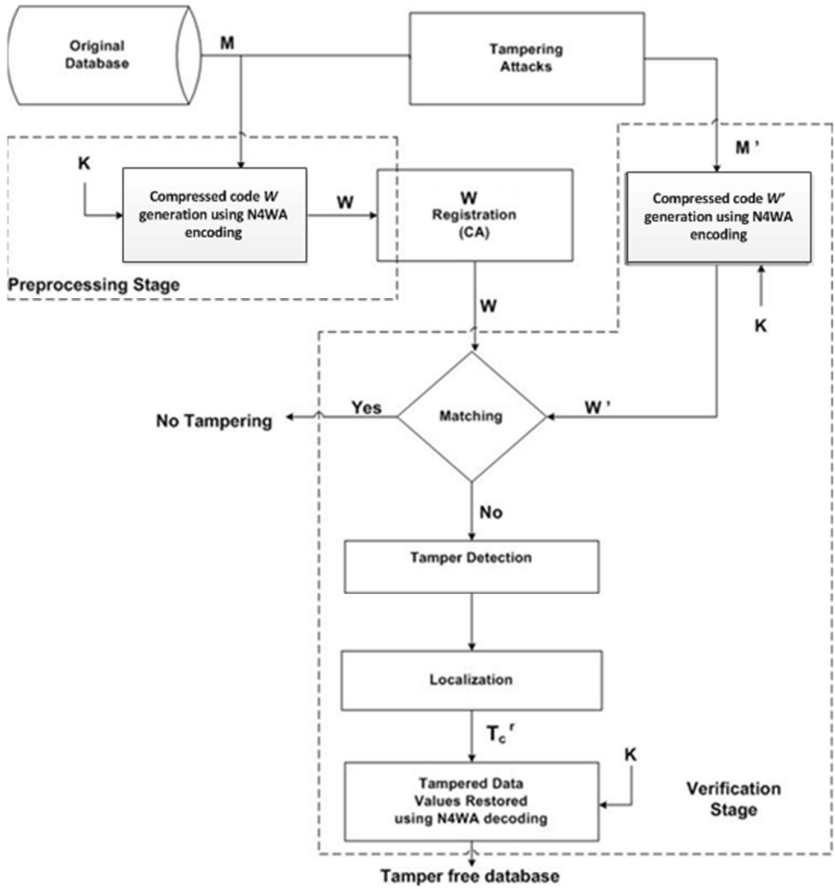

Fig. 1. Proposed tampering detection scheme for relational databases

Symbols and their description which have been used to intricate the proposed scheme are given in Table 1. In Figure 1, *M* is considered as the original relational database. Before publishing *M* and protecting it from tampering attacks, a secure key *K* based compressed code *W* is extracted (briefed in Section 3.2) at preprocessing stage by converting *M* into the binary form and applying N4WA encoding scheme (briefed in section 3.1). The generated *W* is registered at *CA* for verification and restoration purposes. For instance, when *M* is published, it can be a subject to any tampering attack to become *M'*. Attacks are supposed to comprise of illegal modification of any tuple's attribute value, insertion of any illegal tuple(s), deletion of any tuple(s) and modification of multiple attribute values. In the proposed scheme, at verification stage, the same key *K* is used to generate another compressed code *W'* by converting *M'* into binary form and applying N4WA encoding.

Table1. Notations and parameters

Symbols	Description	Symbols	Description
M	Original categorical relational database	P_S	Sorted Primary key hash of all tuples in M
M'	Suspicious relational data base	P_S'	Sorted Primary key hash of all tuples in M'
W	Extracted compressed code of M in binary	B_j^i	Binary form of each tuple in M or M'
W'	Extracted compressed code of M' in binary	$T_i.A_j$	The jth attribute of ith tuple in M or M'
CA	Certification Authority	T_c^r	Tamperinged location of M' in terms of rows and columns
λ	Number of attributes in the relation	$M_c'^r$	Modified cell value at tampered location
ω	Total number of tuples in M or M'	M_c^r	Original cell value at tampered location
P_i^H	Primary key hash of all tuples in M or M'	W_T	Tuple's compressed code
K	Secret Key		

As it can be seen in Figure 1, the two different compressed codes W and W' are extracted at two different times with different inputs. Inspite of this, the alteration detection and localization procedures utilize comparison of W and W' provided by CA to detect and localize tampering in M' (briefed in section 3.3). Once modified data location is identified in terms of its row and column, the restoration procedure consumes location information T_c^r, K and N4WA decoding scheme to recover the modified part of M'.

3.1 N4WA encoding/decoding scheme

In the N4WA encoding/decoding scheme, N4 represents the size of the input to the suggested coding scheme such as the size of the input is 4 bits. WA refrers to "waqas ali" to associate indentification to the proposed scheme. The N4WA coding scheme works on binary string of user-defined block size and here the size is of 4 bits. The key motivation behind N4WA encoding / decoding scheme is that, it does not depend on M during verification and restoration. At the same time, the scheme ensures considerable difference of sizes between M and W for minimum memory resource utilization. Moreover, encode M in such a way that in case of any error in M the restoration is reliable and possible. The N4WA coding scheme is elaborated in Figure 2.

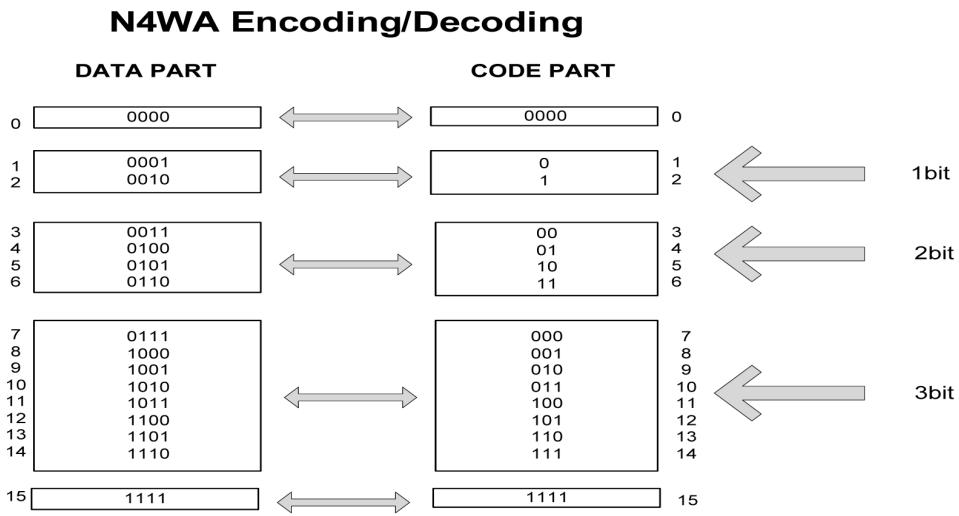

Fig. 2. 4 Bit data replacement using N4WA codingscheme

For instance, if any data unit is in the binary form and segmented in block size of 4, the possible combinations of bits are given in the data part as shown in Figure 2. The bits replacement during encoding given in the code part. Notice how the possible bit combinations of 3, 2 and 1 are interchangeable with 4 bit code part. Intelligently, the two substitutions of two different bit combinations at index 0 and index 15 have remained unchanged.

In the proposed improved tampering detection scheme, the N4WA encoding has been utilized to generate compressed codes *W* and *W'*. On the other hand, N4WA decoding has been utilized for restoration of modified data values in *M'*. The suggested coding scheme assists the proposed scheme to achieve major objectives such as tampering detection, localization and restoration. In order to further understand N4WA coding, consider Figure 3, where *M* is encoded to form *W* using N4WA encoding.

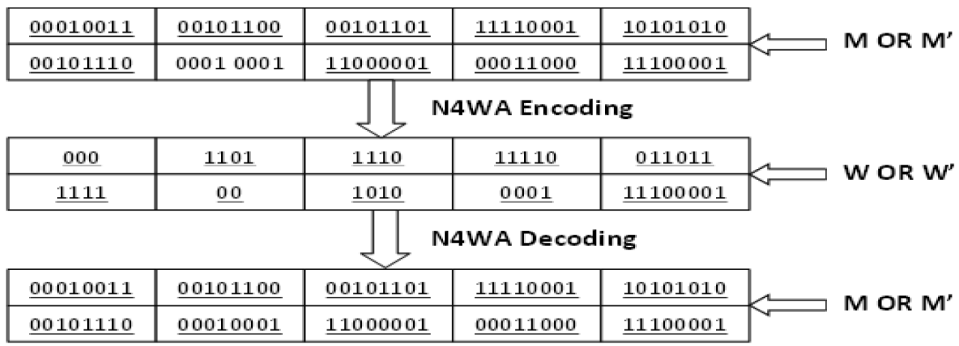

Fig. 3. Demonstration of N4WA encoding/decoding upon M or M' and W or W'

It can be observed from Figure 3 that during encoding, 4 bit data is substituted virtually with possible code bits. Similarly, during N4WA decoding, the code part is reversible with related data part. Notice that N4WA encoding/decoding scheme is a secret key K based scheme (see Figure 1), the replacement indices of data part bits are only known to the administrator. Furthermore, for building K more securely, possible combinations of code part indices are utilized. Also, it can be noticed from Figure 3 that through N4WA encoding, the size of W is small compared to Me.g., M = 64 bits and W=40 bits.

3.2. Compressed code generation

Suppose there is a data base or an entity set M having the primary key P and attributes A. Any tuple T is a subset of M ($T \subseteq M$) denoted by $T(P, A_1, A_2, A_3 \ldots A_\lambda)$. Algorithm 1 describes the compressed code W generation process using N4WA encoding.

Algorithm 1 Compressed code generation

Input $\rightarrow M$

1: *for i=1 to* ω *do*

2: $P_i^H = HASH(K, P_i)$

3: *end for*

4: $Sort(K, P_i^H) \rightarrow P_s$

5: *for i=1 to* ω

6: *for j=1 to* λ *do*

7: $Convert\ in\ binary\ (T_i.A_j) \rightarrow B^i_j$

8: $N4WA\ encoding\ (B^i_j, K) \rightarrow W$ // K specifies the indexes of code part

9: *end for*

10: *end for*

11: $W \rightarrow CA$ // here W' will also be registered at certification authority

In algorithm 1, secure primary key hash of each tuple in M are calculated in lines 1-4. Then primary key hash values of all tuples P_i^H are securely sorted and stored at P_s. However, the P_i^H hash calculations and sorting are virtual processes. Further, using nested looping, each tuple has beenconverted in binary B^i_j (see line 7) and securely encoded with N4WA encoding to form W (see line 8) in lines 5-10. In the end, the generated W has been registered at CA in line 11.

3.3. Tampering detection and localization

Algorithm 2 Detecting and Localizing Tampering

Input $\rightarrow M'$

1: *repeat steps 1 to 5 of algorithm 1 and generate* P'_s

2: *if* $Ps == P'_s$

3: *Check & Alert for (No tamperinging in primary key or keys & No insertion of tuple or tuples & No deletion of tuple or tuples)*

4: *do step 7*

5: *else*

6: *Check & Alert for (tamperinging in primary keys or insertion of tuple or deletion of tuple)*

7: *repeat steps 6 to 13 of algorithm 1 and generate W'*

8: *if W = = W'*

9: *Alert for (M' has no tamperinging)*

10: *else*

11: *for i=1 to* ω

12: *for j=1 to* λ *do*

13: *Compare* (B^i_j *of W,* B^i_j *of W') // locating tamperinged tuple and tamperinged attribute*

14: *get tamperinged location where* (B^i_j *of* $W \neq B^i_j$ *of W')* $\rightarrow T^r_c$

15: $T^r_c \rightarrow M'^r_c$

16: *end for*

17: *end for*

Algorithm 2 describes the detection and localizing tampering in *M*'. In this algorithm, in order to generate P_S' (see line 1) and *W'* from *M'* (see line 7), the steps are similar to algorithm 1. The only difference is the input. In algorithm 1, the input is *M* while in algorithm 2, it is *M*'. In lines 2-6, after secure calculation of P_S', the array of hash values as P_S and P_S' are compared to detect tamperinging in primary key values in M'. This tampering might comprise of illegal insertion/deletion of tuple(s) and modification of primary key values. In line 7, if there is no tampering in primary keys of M', then W' is generated and compared with *W*. Further, in lines 8-9, if W' is equal to *W* along with $P_S = P_S'$, then $M = M'$ which reflects no tampering in M'. Moreover, lines 11-17 present that, if $W \neq W'$ then there is tampering in tuples attribute values. Once tampering is determined, the binary form of each tuple of *W* and W' is compared

(see line 13). The purpose of comparison is to detect altered location in terms of tuple and its tampered attribute. Whilst, the tampered locations are represented as T_c^r where *r* represents altered tuples and *c* represents its attributes (see line 14). Finally, in line 15 of algorithm 2, the modified values are stored in $M_c^{'r}$ for data level localization purpose to increase accuracy of locating tampering.

3.4. Tampering recovery

Algorithm 3 describes the restoration of modified values of *M'*. For recovery, in lines 1-7 of algorithm 3, the tampered locations T_c^r are taken as input and then utilizing *W* from *CA*, tampered location values are decoded (see line 5) using N4WA decoding to get original values as M_c^r.

Algorithm 3 Tampering recovery

Input $\rightarrow T^r_c$

1: *for r = 1 to n*

2: *for c = 1 to n do*

3: $r \rightarrow i$

4: $c \rightarrow j$

5: *N4WA decoding* (B^i_j *of W, K)* $\rightarrow M^r_c$ *// recovering original attribute value using N4WA decoding*

6: *end for*

7: *end for*

3.5. Example

In order to explain how the proposed scheme works, consider an attribute value “42”, which is represented as *M*. At first, the value “42” is converted into binary which is “00101010”. After the binary conversion, the 8 bit code is partitioned into 4 bit codes as “0010” and “1010”. Now the resultant 4 bit codes are given to N4WA encoding to generate *W* which is “1011”. As a result, “0010” is replaced with “1”, and “1010” is replaced with “011”. Afterwards, *W* and its location are stored at *CA*. Suppose that *M* is subjected to a tampering attack and the value “42” becomes “13”. According to the proposed scheme, the value “13” is represented as *M'*. The compressed code *W'* for *M'* is regenerated using N4WA encoding which is “0000110”. Now the two compressed codes *W* and *W'* are compared to detect and localize tampering. Mismatching of *W* and *W'* leads to tampering. At the end, the location of tampered value is taken and using *W*, the original value is restored.

4. Results and discussion

The scheme demonstrated in this paper is capable of detecting the following malicious modifications, (e.g. tampering attacks): (1) Insertion/Deletion of one or more tuple(s), (2) Modification of one or multiple attribute values.

In order to determine the behavior of the proposed scheme against these malicious modifications, a dataset has been utilized, which is available at *http://archive.ics.uci.edu/ml/datasets.html* and is known as Forest Cover Type dataset. The dataset has been considered as M in the current study. This dataset has 581,10 tuples alongwith 44 Boolean attributes, 1 categorical attribute and 10 integers attributes. Experiments have been conducted in Turbo C++ on Intel Core 3.2GHz i3 CPU with a RAM of 2 GB. In order to compose attacks and predict the behavior of the proposed scheme while countering to tampering attacks, the scheme is tested at different error rates α which is defined as:

α =Number of Modified data units of M / Total number of data units in M

where the data unit can be a tuple or attribute values. α is self controlled for experimental purposes and the results have been achieved at different values of α. Breifing the applied probalistic framework which is utilized in evaluating the behaviour of the proposed scheme against different tampering attacks, the framework has considered the probability that the event does not arise. For instance, If P_x is the probability of any event for which the evaluation is intended, e.g. the probability that shows the behaviour of scheme against attacks, then $1 - P_x$ is the probability that the event does not occur. This conceptual model is further designed accordingly and breifed in each of the following section.

4.1. Insertion / deletion of one or more tuples

In the scheme, during preprocessing phase with the input M, the array P_S holds sorted primary key hash values of all tuples (see algorithm 1 line 5). Suppose one or more illegal tuples are inserted or deleted in M, then it becomes M'. Afterwards, during verification stage of M', all tuples primary key hash values are sorted and placed in P'_s. At this stage, behavior of the presented scheme is testable. Mismatching of P_S and P'_s identifies the insertion or deletion of tuple or tuples (see algorithm 2 lines 2-6). Analyzing data structure P'_s at verification stage in case of illegal insertion/deletion, there is an equal probability of each place in P'_s to have illegal tuple primary key hash or misplacement of genuine primary key hash value. When a tuple or n tuples are inserted or deleted illegally, the size of P'_s increases to n. The probability that the scheme will detect insertion of one or multiple tuples is given in equation 1.

$$P_{iT} = 1 - \sum_{i=1}^{n} \frac{1}{T_i^p} \tag{1}$$

where P_{iT} denotes the probability of detecting insertion or deletion of one or multiple tuples and $\sum_{i=1}^{n} \frac{1}{T_i^p}$ denotes chances of failure (not detectable) of all places in P'_s. The term T_i^p is further described in equation 2.

$$T_i^p = p_1 \cup p_{i=1\ldots n} \tag{2}$$

Considering equation 1 and equation 2, $\frac{1}{p_1}$ denotes chances of failure, when one tuple is inserted or deleted and $\frac{1}{p_{i=1\ldots n}}$ represents chances of failure when more than one tuple are inserted or deleted.

In Figure 4, the horizontal plane represents the total number of tuples in the dataset under observation and the vertical plane shows the proposed scheme chances of failure scale.

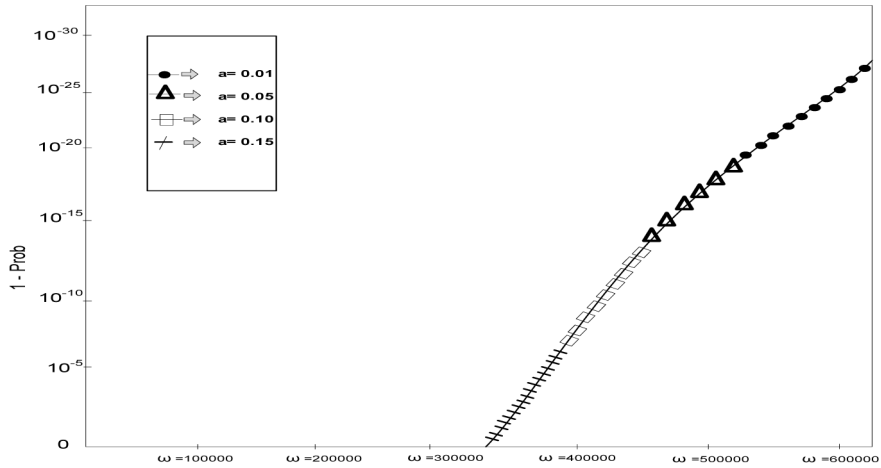

Fig. 4. Probability of failure while Insertion /Deletion of massive tuples

Figure 4 demonstrates the behavior of the proposed scheme under different volumes of insertion/deletion attacks. For example, the volume of attack is represented by α and has been calculated as the *total number of inserted/deleted* ω *in M/ total number of* ω *in M.* The dataset under observation has ω from 0 to 340000 and with any value of α (0.01, 0.05, 0.10 and 0.15), the scheme's resistance capability is 100% by using Equation 1. However, it can be observed From Figure 4 that the probability of failure in detecting massive tuple insertion or deletion with increasing error rate α is very low and is negligible. Therefore, the proposed scheme resists well in minimum to maximum tuple insertion or deletion scenarios.

4.2. Modification of single or multiple attribute values

Suppose an attribute value or multiple attribute values are modified illegally. The illegal modification of attribute or attributes affects the tuple(s) compressed code W_T which is defined in Equation 3.

$$W_T = P_i^H \cap w_1 \cap w_2 \cap ...w_\lambda \tag{3}$$

where P_i^H denotes primary key hash of i[th] tuple and $w_{1to\lambda}$ denotes cells compressed code of i[th] tuple. Any attribute modification is detectable by utilizing the mismatching of B_j^i of W and W' (see algorithm 2 line 7-14). In addition, testing the behavior of the proposed scheme at this stage, the probability that the scheme will detect modification of one or multiple attribute values has been given in equation 4.

$$P_{mT} = 1 - \frac{1}{[p_{1t} \cdot p_{a1} + \sum_{j=1}^{n} p_{aj}] + [\sum_{i=1}^{n} p_i \cdot p_{a1} + \sum_{j=1}^{n} p_{aj}]} \tag{4}$$

where P_{mT} denotes the probability of detecting illegal modification of one or multiple attribute values, $\frac{1}{p_{1t} \cdot p_{a1} + \sum_{j=1}^{n} p_{aj}}$ denotes chances of failure (not detectable) when single tuple compressed code is modified and when single or multiple attribute values are changed, $\frac{1}{\sum_{i=1}^{n} p_i \cdot p_{a1} + \sum_{j=1}^{n} p_{aj}}$ denotes chances of failure (not detectable) when multiple tuples compressed codes have been changed along with single or multiple attribute values change. Specifically in equation 4, p_{1t} shows an error in detecting single tuple alteration; p_{a1} shows an error in detecting the single attribute value modification; $\sum_{j=1}^{n} p_{aj}$ denotes an error in detecting the multiple attribute values modifications and $\sum_{i=1}^{n} p_i$ represents an error in detecting the multiple tuples alterations. In Figure 5, the volume of attack has been represented as α and is calculated as *total number of modified attribute valuesin M/ total number of attribute values in M.*

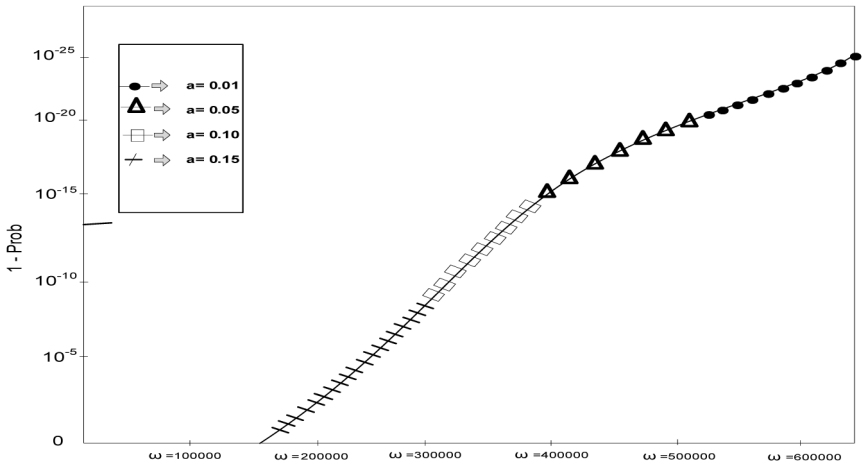

Fig. 5. Probability of failure while modification of attribute values

Figure 5 depicts that the dataset under observation has ω from 0 to 170000 and with any value of α (0.01, 0.05, 0.10 and 0.15), the resistance capacity of the scheme is 100% by using Equation 4. However, it can be observed that the probability of failure in detecting the massive attribute modification with increasing error rate α is very low and thus negligible. Therefore, the proposed scheme resists well in minimum to maximum modification of attribute values scenario.

4.3 Comparitive results

In this section the advocated scheme is evaluated in comparison with the existing schemes of tampering detection for relational data. Three parameters such as accurate localization (e.g cell level identification), restoration and memory consumption are considered in the evaluation process.

Table 2. Proposed scheme comparison with existing schemes

S. No	Schemes by Authors	Accurate Localization	Restoration	Memory Consumption
1.	(Bhattacharya & Cortesi, 2009)	No	No	N/A
2.	(Halder & Cortesi, 2010; Halder *et al.*, 2011; Halder *et al.*, 2010	Yes	No	N/A
3.	(Kamel, 2009)	No	No	N/A
4.	(Khan & Husain, 2013)	No	No	N/A
5.	(Li *et al.*, 2004)	No	No	N/A
6.	(Gao *et al.*, 2013)	No	Yes	High
7.	Proposed (N4WA)	Yes	Yes	Low

It can be observed from Table 2 that there is no restoration feature in most of the existing schemes which ultimately degrades the dependability of the protection. The scheme presented by Gao (2013) has provided restoration feature as it utilize Reed Solomon codes. Neverthless, the theoratical evaluation confirms that it consumes high amount of memory in restoration process. In spite of memory consumption utilization, the compression utility of the proposed scheme overcomes this problem as depicted in Table 2. Moreover, the suggested scheme provides accurate localization and restoration in cooperation.

4.4. Discussion

According to the blind verification property of the tampering prevention scheme, the process for integrity verification of *M* and restoration of *M'* should not depend on the original relational database or in other words, it should not depend on *M*. Therefore, in the proposed scheme, the integrity verification, given in algorithm 2, and restoration, given in algorithm 3, do not depend on *M*. In fact they depend on the compressed codes *W* or *W'*. In addition, notice that in line 15 of algorithm 2, $M_c^{'r}$ represents the modified values at tampered locations and in line 5 of algorithm 3, M_c^{r} represents the original values at tampered locations. Both representations $M_c^{'r}$ and M_c^{r} are utilized to achieve the attribute value level localization or cell level localization which ultimately increase the proposed scheme localization accuracy.

5. Conclusions

An improved tampering detection scheme for relational data has been presented in this paper which is distortion free and provides memory efficient tampering detection, localization and restoration of the tampered data values for relational databases. Aprefix oreinted N4WA encoding/decoding scheme is presented, which has the capability of preventing distortions in the relations of the original database. The proposed scheme accurately locates tampering and provides a capability to restore tampered values. Thus, the problems of integrity, usability of data, accuracy of detecting tampering, memory consumption and restoration of modified values are handled without the need to depend on the original content of the database. Also, the results have justified the applicability and capability of suggested scheme for reliable tampering detection. In the future, we intend to apply the N4WA encoding/decoding scheme to develop copyright protection scheme for relational databases.

References

Abdel-Hamid, A.T. & Sofiene, T. (2008) Fragile IP watermarking schemes. In: Adaptive Hardware and Systems, 2008. AHS'08. NASA/ESA Conference on, pp. 513-519.

Agrawal, R., Haas, P.J. & Kiernan, J. (2003) Watermarking relational data: framework, algorithms and analysis. The VLDB journal,**12**:157-169.

Al-Haj, A. & Odeh, A. (2008) Robust and blind watermarking of relational database systems. Journal of Computer Science, **4**:1024.

Bhattacharya, S. & Cortesi, A. (2009a) A distortion free watermark framework for relational databases. In: ICSOFT (2), pp. 229-234.

Bhattacharya, S. & Cortesi, A. (2009b) A generic distortion free watermarking scheme for relational databases. Information Systems Security.Springer, Pp. 252-264.

Cao, H.W.X.C.Z. (2008) A speech based algorithm for watermarking relational databases. International Symposiums on Information Processing, pp. 603-606

Chen, M., He, Y. & Lagendijk, R.L. (2005) A fragile watermark error detection scheme for wireless video communications. Multimedia, IEEE Transactions on, **7**:201-211.

Gao, L., Wang, D. & Hamadou, A. (2013) New fragile database watermarking scheme with restoration using reed-solomon codes. Journal of Computational and Theoretical Nanoscience, **10**:147-153.

Guo, F., Wang, J., Zhang, Z., Ye, X. & Li, D. (2006) An improved algorithm to watermark numeric relational data. Information Security Applications. Springer, Pp. 138-149.

Guo, H., Li, Y., Liu, A. & Jajodia, S. (2006) A fragile watermarking scheme for detecting malicious modifications of database relations. Information Sciences, **176**:1350-1378.

Gupta, G. & Pieprzyk, J. (2009) Database relation watermarking resilient against secondary watermarking attacks. Information Systems Security. Springer Pp. 222-236.

Halder, R. & Cortesi, A. (2010) Persistent watermarking of relational databases. In: Proceedings of the IEEE International Conference on Advances in Communication, Network, and Computing, (CNC 2010), October, pp. 4-5.

Halder, R., Dasgupta, P., Naskar, S. & Sarma, S.S. (2011) An Internet-based ip protection scheme for circuit designs using linear feedback shift register-based locking. Engineering Letters, **19**:84.

Halder, R., Pal, S. & Cortesi, A. (2010) Watermarking schemes for relational databases: survey, classification and comparison. Journal of Universal Computer Science, **16**:3164-3190.

Hu, T.L., Chen, K. & Dong, J.X. (2005) Garwm: Towards a generalized and adaptive watermark scheme for relational data. Advances in Web-Age Information Management. Springer, Pp. 380-391.

Huang, M., Cao, J., Peng, Z. & Fang, Y. (2004) A new watermark scheme for relational data. In: Computer and Information Technology, International Conference, pp. 946-950.

Kamel, I. (2009) A schema for protecting the integrity of databases. Computers & Security, **28**:698-709.

Khan, A. & Husain, S.A. (2013) A fragile zero watermarking scheme to detect and characterize malicious modifications in database relations. The Scientific World Journal 2013: 1-16 Article ID 796726

Li, Y. & Deng, R.H. (2006) Publicly verifiable ownership protection for relational databases. In: Proceedings of the 2006 ACM Symposium on Information, computer and communications security, pp. 78-89.

Li, Y., Guo, H. & Jajodia, S. (2004) Tampering detection and localization for categorical data using fragile watermarks. In: Proceedings of the 4th ACM workshop on Digital rights management, pp. 73-82.

Prasannakumari, V. (2009) A robust tampering proof watermarking for data integrity in relational databases. Research Journal of Information Technology, **1**:115-121.

Rashidi, H. (2010) A novel watermarking scheme for detecting and recovering distortions in database tables. arXiv preprint arXiv:1009.0827

Sion, R. (2004). Proving ownership over categorical data. In: Data Engineering, 2004. Proceedings. 20th

International Conference on, pp. 584-595.

Sion, R., Atallah, M.J. & Prabhakar, S. (2005) Rights protection for categorical data. Knowledge and Data Engineering, IEEE Transactions, **17**:912-926.

Tsai, M.H., Hsu, F.Y., Chang, J.D. & Wu, H.C. (2007) Fragile database watermarking for malicious tampering detection using support vector regression. In: Intelligent Information Hiding and Multimedia Signal Processing, 2007. IIHMSP 2007. Third International Conference, pp. 493-496.

Tsai, M.H., Tseng, H.Y. & Lai, C.Y. (2006) A database watermarking scheme for temper detection. In: Journal of Computer Information Systems, 2006

Zhang, Y., Niu, X.M. & Zhao, D. (2004) A method of protecting relational databases copyright with cloud watermark. International Journal of Information Technology, **1**:206-210.

Zhou, C.W.J.W.M. & Li, G.C.D. (2008) Atbam: An arnold transform based method on watermarking relational data. International Conference on Multimedia and Ubiquitous Engineering, pp. 263-270

Zhou, X., Huang, M. & Peng, Z. (2007) An additive-attack-proof watermarking scheme for databases' copyrights protection using image. In: Proceedings of the 2007 ACM symposium on Applied computing, pp. 254-258.

Submitted **:** 13/08/2014

Revised **:** 30/10/2014

Accepted **:** 02/11/2014

نظام تشفير N4WA باعتماد الموجهة البادئة لتحسين الكشف عن التلاعب في البيانات العلاقية

[1]وقاص حيدر، [2]محمد شريف، [2]هادية بشير، [2،*]مدثر رضا، [2]مسرت ياسمين

[1]كلية الهندسة وتكنولوجيا المعلومات، ADFA@UNSW

[2]قسم علوم الحاسوب، معهد كومساتس لتكنولوجيا المعلومات، واهكانت، باكستان

المؤلف : mudassarr@gmail.com

خلاصة

تم في هذا البحث تقديم نظام تشفير (N4WA) يستخدم الذاكرة بكفاءة (وصف النظام بإيجاز في القسم 3.1)، يستخدم النظام في كشف التلاعب واستعادة قيم البيانات المعدلة في البيانات العلاقية. نظم المعلومات المائية الحالية للكشف عن التلاعب لا تحتوي على المكونات الضرورية مثل التعاون بين التحديد الدقيق والاستعادة. لذا تحد هذه الأخطاء من جودة واعتمادية طرق الكشف عن التلاعب. ولكن في النظام المقترح، يتم أولاً توليد شفرة مضغوطة ش من البيانات الأصلية ص بواسطة نظام (N4WA) للتشفير وتخرين هذه الشفرة لدى جهة موثوق بها. ثانيا تعطى البيانات المفترضة صَ والمشكوك بها إلى نظام (N4WA) لتوليد شفرة مضغوطة أخرى شَ. ثم يتم مقارنة الشفرتين ش و شَ لتحديد أماكن التلاعب في الشفرة شَ. وأخيراً باستخدام مواقع التلاعب المحددة يقوم نظام N4WA بالتعاون مع الشفرة الأصلية ش باسترجاع القيم الأصلية للبيانات. تشير النتائج أن في علاقات قواعد البيانات على اختلاف أنواعها ، تم كشف التلاعب من أصغره إلى أكبره وتحديد مواقعه واستعادة قيمه الأصلية بنجاح.

الكلمات المفتاحية:

ضغط بدون فقدان ، استعادة البيانات العلاقية ، هجمات التلاعب ، خلو التشويه.