Distortion-free Watermarking Scheme for Compressed Data in Columnar Database

Waheeb Yaqub[1], Ibrahim Kamel[2] and Zeyar Aung[3]
[1]*Center for Cyber Security, New York University, Abu Dhabi, U.A.E.*
[2]*Department of Electrical and Computer Engineering, University of Sharjah, Sharjah, U.A.E.*
[3]*Department of Computer Science, Khalifa University of Science and Technology, Masdar Institute, Abu Dhabi, U.A.E.*

Keywords: Data Integrity, Digital Watermarking, Columnar Databases, Information Hiding, Database Security.

Abstract: Digital watermarking is an effective technique for protecting various databases against data copyright infringement, piracy, and data tampering in the columnar database. Increasing deployments of various database systems and their versatile applications have raised the need for better watermarking schemes that are tailored to the target database systems' specific architecture. Most of the existing digital watermarking schemes do not take into consideration the side effects that watermarking might have on the database's important characteristics such as data compression and overall performance. In this research, we propose a distortion-free fragile watermarking scheme for columnar database architecture without interfering its underlying data compression scheme and its overall performance. The proposed scheme is flexible and can be adapted to various distributions of data. We tested our proposed scheme on both synthetic and real-world data, and proved its effectiveness.

1 INTRODUCTION

Databases have served the business intelligence industry well over the last few decades. However, with the rise of Big Data Analytics and the vast quantities of data that reside in organizations, fast analysis of data is becoming more difficult. Columnar databases (in contrast to row-wise relational databases) boost performance of data analytic transactions by reducing the amount of data that needs to be read from disk. They succeeded to achieve that by storing data column-wise (all values of an attribute are clustered together). This allowed retrieving only those attributes required by the query and leaving the rest of the tuple on the disk. The similar values in columnar data allowed achieving better compression rate for data stored in columnar databases. One of which is decreasing the bottleneck between RAM and Hard drive. For instance, in traditional relational database answering a range query would require loading the set of records that falls in the range query and then executing projection query to discard the unwanted attributes. In columnar DB, in contrast, answering range queries would require loading only the columns that contain the attributes that are associated with query. Therefore, the query performance is often increased, especially when the database is very large. Another important advantage of storing data in columns, it lends itself naturally to compression. Columnar DB achieves much higher compression rate and allows many queries to be answered on compressed data.

The database research community came to realize the importance of protecting the integrity of the database content, particularly those that are kept on the cloud or published on the web. Databases usually contain sensitive and critical information such as salaries, expenses, loans, inventory value, etc. Thus, unauthorized changes to databases might result in significant losses to organizations and individuals. The use of secure hash like MD5 or SHA and Digital Signatures for protecting the integrity of the whole databases would detect unauthorized alteration but cannot localize the attack. Using Digital Signature or SHA at a finer level, e.g., attribute level would be very costly and requires significant space overhead. Digital watermarking is one of the techniques that can be used to protect data integrity. Digital watermarking was an attractive technology because of its lightweight and efficiency. Originally digital watermarking research focused on multimedia objects such as image, audios and video (Lee and Jung, 2001; Pot-

Yaqub, W., Kamel, I. and Aung, Z.
Distortion-free Watermarking Scheme for Compressed Data in Columnar Database.
DOI: 10.5220/0006857203430353
In *Proceedings of the 15th International Joint Conference on e-Business and Telecommunications (ICETE 2018) - Volume 2: SECRYPT*, pages 177-187
ISBN: 978-989-758-319-3

dar et al., 2005; Bajpai and Kaur, 2016; Asikuzzaman and Pickering, 2017). In the recent years, researchers started focusing on methods to watermark relational databases (Agrawal and Kiernan, 2002; Khanna and Zane, 2000; Kamel, 2009; Li et al., 2004; Li and Deng, 2006; Guo et al., 2006; Kamel et al., 2013; Camara et al., 2014; Kamel et al., 2016; Rani et al., 2017). Watermarking schemes exploit redundancy in the data objects to hide the secret message (digital watermark). Alphanumeric data stored in databases usually contain less redundancy than image and video data, and thus more difficult to watermark. Moreover, the dynamic nature of databases makes the process of designing a watermarking scheme even more challenging. A digital watermark is a secret pattern of bits inserted in the data objects to verify the integrity of the content or to identify ownership. Digital watermarking can be classified into two categories: Robust and Fragile. Robust watermarking is used for copyright protection and designed to withstand attacks like compression, cropping, scaling, and any other geometrical transformations. Fragile watermarks, on the other hand, get corrupted with slight change to the content, while allowing authorized operations. In other words, fragile watermarking is used to allow legitimate operations to the data and detects any unauthorized changes. One of the main requirements of a successful watermark to be inconspicuous. In its simplest form, watermark bits can be stored in the least significant bits of an image, for example.

Recently, there have been several studies for protecting data integrity in relational databases. The detection rate depends on the type and the value of the modification. Most of the existing techniques for protecting relational database integrity store the watermark by distorting bits of the protected attributes (Agrawal and Kiernan, 2002; Pérez Gort et al., 2017). The most serious limitation of the existing integrity protection techniques is that they introduce distortions to the watermarked attributes. Therefore, these techniques affect the data values and change them. While these types of watermarking can be used in applications that can tolerate distortion, e.g., meteorological readings; they are not suitable for many other applications that deal with data like, employees salaries, account balance, inventory values, medical data, etc.

This paper proposes a new watermarking scheme for detecting unauthorized modification in columnar databases. The proposed fragile watermarking scheme works on both uncompressed and compressed columnar databases without the need to decompress the data. The proposed algorithm can detect, with high probability, unauthorized changes to altered attributes. To the best of authors knowledge, there is no published study on watermarking columnar databases for protecting data integrity. The basic idea behind the proposed scheme is to organize columns of data into groups of fixed size. Initially, groups are rearranged according to a predetermined order. The watermarking algorithm sorts the group in an order that corresponds to value of secret digital watermark. Unauthorized users, who do not know the secret watermark, can still change any of the attribute values in the group. However, this blind changes will, with high probability, disturb the secret order. On the other hand, authorized users who know the secret watermark, need to re-adjust the group after making any change. This paper made the following contributions:

- Propose a watermarking scheme to protect the data integrity of columnar databases.
- The proposed scheme introduce zero distortion to the protected attributes.
- Watermark insertion and detections algorithms work directly on the compressed data without the need to decompress.
- Integrity detection algorithm can identify, with high probability, the attacked attribute value.
- Provides comprehensive simulation experiments on performance of the proposed scheme using synthetic and real data.

2 RELATED WORK

One of the first works in distortion-free database watermarking for tamper detection was by Li et al. (Li et al., 2004). The scheme relied on creating a permutation of tuples in the database. The authors proposed virtual grouping of the tuples in the table with the help of a cryptographic hash function. Each tuple's hash value is compared with the group's hash value to decide the membership of the tuple to a virtual group. Each group is watermarked separately. The watermarking is based on changing the order of a tuple with its neighbouring tuple depending on the tuples' hash value. For instance, if the watermark bit value is zero and the first tuple's hash value is greater than the second tuple's hash value, then the pairs are switched in location. This scheme can only localise the attack up to the group size, and the attacked tuple's exact location cannot be determined. Furthermore, the update process is costly due to the involvement of cryptographic hash functions. Moreover, every group has a watermark to be stored separately. The relationship between consecutive pairs are compared, and hence

the attack on indirect neighbour values may not be detected.

Khan and Husain (Khan and Husain, 2013) proposed a fragile scheme based on characteristics of data such as the frequency of distribution of digit counts, length and range of numerical data values. They relied on third party such as certification authority to store the watermark value in order to achieve zero watermarking property i.e. distortion free. The scheme proposed by Khan and Husain is similar to distortion-free based on hash value of the database that is stored in third party's vicinity. Such techniques will not be able to localize the attack, thereby making the whole relation as malicious whenever attacks are detected.

Bhattacharya and Cortesi (Bhattacharya and Cortesi, 2009) also proposed a watermarking scheme which detects malicious changes. The main difference between Li et al. and Bhattacharya and Cortesi is that the former virtually groups the tuples based on the hash values of the primary keys and the secret key, while the latter carries out grouping based on categorical attribute values.

The public watermarking scheme proposed in Li and Deng (Li and Deng, 2006) embeds the watermark without limiting the data type of attributes to numerical or categorical. The scheme first creates a relation W out of the original relation T. The relation W is of size $n \times \eta$, where η is less than or equal to the number of attributes. The attribute values in W are the Most Significant Bits (MSB) of those in T. The relation W is made public such that anyone can verify the authenticity of MSBs of the relation T. The drawbacks are that: (1) If an attacker changes the LSB of T, it will not be detected. (2) Publishing W can be costly in terms of space in cases of large databases with millions of tuples.

All the fragile watermarking methods of relational database for tamper detection mentioned above (Li et al., 2004; Khan and Husain, 2013; Bhattacharya and Cortesi, 2009; Li and Deng, 2006) either perform detection only or detection as well as localisation of an attack up to a virtual group. (Kamel et al., 2016) proposed a scheme that can localise the attack to the granularity of one or two candidate tuples.

Unfortunately, all of the above-mentioned techniques do not cater for the underlying architecture and data compression schemes in DBMSs. While most of the DBMSs rely on their own storage and query processing strategies for performance optimization, the watermarking techniques proposed in last few years (Li et al., 2004; Li and Deng, 2006; Zhang et al., 2006; Wang et al., 2008; Bhattacharya and Cortesi, 2009; Hu et al., 2009; Rao and Prasad, 2012; Kamel et al., 2013; Franco-Contreras et al., 2014; Kamel et al., 2016; Rani et al., 2017) do not take into consideration the performance degradation from watermark embedding. Although Rani et al. (Rani et al., 2017) considered only the distributed database architecture at which their watermarking scheme to be deployed. Whereas, none of the previously mentioned authors considered the underlying compression scheme used by database.

3 PROPOSED WATERMARKING SCHEME-COLUMNARWM

This section introduces the proposed scheme for watermarking. The aim of this watermarking scheme is to detect and identify tampering of the data stored in the columnar database. The proposed watermarking scheme has the following desirable properties:

- Distortion free: the proposed the scheme will not introduce any distortion or modification to the values of the underlying data.
- Compression independent: the proposed scheme can be applied directly to compressed data in columnar DB.
- Allowing incremental updates: watermarked data can be updated by simply updating small set of data.
- Blind: verification of the watermark existence does not require knowledge of the original database.
- Modular: each column in columnar database can watermarked separately and not relying on the content of the other columns.
- Detect and localize attacks: the proposed scheme will be able to detect attacks with high tamper detection rate and localize the victim data element in most cases.
- Incur minimal performance degradation

Attributes in a columnar database is stored in a separate file (Abadi et al., 2008; Abadi et al., 2009; Abadi et al., 2012). The proposed scheme benefits from the redundant order of data items of a column by hiding the watermark in the relative order of the data items. Each file (attribute) is processed separately; meaning the algorithm watermarks each column separately. Each column (attribute) is organized into groups of g data elements each. Each group is watermarked separately. Notice here that a group contains a set of values belong to one attribute. The group is watermarked by reordering its data elements in such

a way that the group's new order corresponds to some unique watermark value. The re-arrangement of data items in each group is done relative to a secret order called "reference order", can be any order, e.g., ascending order of data item values.

The proposed scheme ("ColumnarWM") reorders each database column with respect to reference order I_o, according to watermark value W. The reference order I_o and watermark value W are kept secret, therefore any unauthorized change will distort the watermark. ColumnarWM neither introduces any distortion nor affects the usability of the data, it simply stores the watermark value W in the relative order of the data items in a column of the database. Moreover, the scheme adds the watermark to each column separately. Therefore, guaranteeing modular access of each column, which is the main feature of a columnar database over row database. ColumnarWM inserts the watermark without affecting the compression ratio of the compressing technique used in a columnar database. The desirable properties of the proposed scheme such as distortion-free, modular access and invariable compression assure that the scheme will not interfere with columnar database main operations.

The first step in watermarking phase is to divide the entire column of a database into a number of groups, each group has g data elements of a column. This step is called grouping. Then each group is watermarked and verified independently. By organizing column data elements into groups and watermarking them independently, the proposed scheme ensures that process of watermarking is incremental. New data elements form a new group that need to be watermarked separately and independent of the rest of the column. The data elements in each group of the column are sorted with respect to selected reference order I_o. Subsequently, each group is watermarked according to watermark value W using the proposed watermark embedding algorithm WATERMARKEMBED. The new re-ordered data elements of the group are the watermarked order I_w.

Since the order of the data elements in the group represents the watermark value W. There is a need for a one to one mapping between watermark value W and order of the data elements. We use a bijection mapping proposed in (Kamel et al., 2016; Kamel and Kamel, 2011) which is simply a one to one mapping that uniquely maps the order of the list of entries to a numerical number.

Hence, given an ordered group of data elements the watermark value W can be recovered. Authorized users can verify the integrity of attribute by simply knowing the reference order I_o and watermark value W. An integrity verification algorithm can extract the watermark from a group of data elements; if the watermark is correct, it can be concluded that data at rest is integral otherwise attacked.

Recall that the proposed scheme ColumnarWM hides the watermark in the relative order of data elements of the group. Hence theoretically, there are $g!$ unique watermark values that can be inserted as the order of data elements. On one side we have a group that contains g data elements $c_1, ..., c_g$. On the other side we have all possible values of watermark W in integer decimal number. Given that, there might not be any clear pattern relating the integer numbers to the permutations of data elements. We used the method proposed in (Kamel et al., 2016; Kamel and Kamel, 2011) for mapping unique W integer values of size r digits. The method relies on converting the integer valued W to factorial number base system. Main reason for converting integer number W to factoradic base number is that there's a close connection between factoradic numbers and permutations of data values. Therefore, before embedding the watermark, we first convert watermark value W from decimal format to factoradic format W_f. Then the group of data elements are ordered according to reference order I_o. Finally the watermark value W_f is embedded using algorithm WATERMARKEMBED. The authenticity of database is assured by performing integrity verification using the reference order I_o after applying de-watermarking algorithm. The de-watermarking algorithm which is inverse of watermark embedding algorithm. The de-watermarking algorithm tries to reconstruct the data elements i.e. column's reference order I_o from the watermarked order I_w. If the resulting order of each group confirms with reference order I_o, then the database is authentic.

Some notations that will be used throughout this paper are summarized in Table 1. To describe the proposed scheme for protecting the integrity of columnar database, we will present the following in the next section 3.1.

- An algorithm for inserting watermark in column
- Integrity algorithm for that will point out if unauthorized modification carried out on protected column
- Victim identification algorithm for limited cases (sub operation in integrity check algorithm).

The following two sections present the description of the two algorithms: WATERMARKEMBED and INTEGRITYVERIFICATION. The attributes to be watermarked are to be decided by the database owner. To process queries, a columnar database reads attribute A_i from disk pages of secondary storage.

Algorithm 1 (WATERMARKEMBED) is used for

grouping, enforcing reference order, and inserting the watermark in groups of the columnar database.

Table 1: Frequently-used notations in this study.

Symbol	Description
G	group of the relation R
g	The number of data elements in a group
W	Secret watermark value in decimal
I_o	Reference order by which data elements' integrity is verified
$P_d, \hat{P}_d$	Probability and estimated probability of detecting an attack on the watermarked database
$P_{l1}, \hat{P}_{l1}$	Probability and estimated probability of localisation up to 1 victim (i.e., exactly pinpointing the victim)
$P_{l2}, \hat{P}_{l2}$	Probability and estimated probability of localisation up to 2 victims (i.e., either one of them is the victim)

3.1 Watermark Embedding

The watermarking procedure WATERMARKEMBED includes sub-operations: (a) the column data elements are organized to groups of size g, (b) each group content is ordered according to reference order I_o, and (c) the group content is watermarked using the value W which is only known to database owner. For simplicity, let us consider that the attributes are stored in pure column-store model, i.e. each attribute is stored in separate file. Although our proposed scheme can be easily extended to multi-column columnar database. Multi-column columnar database is a simple extension of pure column columnar database, which stores more than one attribute in one file.

The core operation of the ColumnarWM scheme's embedding algorithm is left-circular shift on subsets of data elements of column from the reference order. First the embedding algorithm sorts the group data elements according to reference order. This step is crucial so that when watermark extraction(dewatermarking) is carried out the reference order can be used as a validation check for degradation of watermark. Then the watermark value W_f is used to shuffle the data elements of the group to reflect the watermark. To further clarify the watermark embedding process, consider the group content be $\{c_2, c_1, c_5, c_3, c_4\}$ of size $g = 5$. In general the whole column content is divided into groups of size g [line2]. Let us consider the group content after enforcing reference order be $\{c_1, c_2, c_3, c_4, c_5\}$ [line5]. Let us further assume that watermark value in factoradic number system W_f is 421 [line6]. Both watermark value and reference order are secret and only known to the database owner. Since the watermark value is of three digits, therefore the LeftCircularShift method will be executed three times in a loop [line7-8]. In the first iteration of loop all data elements of $\{c_1, c_2, c_3, c_4, c_5\}$ are shifted left by four positions (by first most significant digit of W_f) resulting into $\{c_5, c_1, c_2, c_3, c_4\}$. The data element c_5 is in its final position and will not be considered in circular-shift of remaining two iterations of the loop. Hence, only four elements will considered $\{c_1, c_2, c_3, c_4\}$. Since the next digit of W_f is 2, only $\{c_1, c_2, c_3, c_4\}$ elements are shifted left by two positions, resulting in $\{c_3, c_4, c_1, c_2\}$(the whole group looks like $\{c_5, c_3, c_4, c_1, c_2\}$). Similarly, the third iteration will freeze c_3 and remaining data elements are shifted left by 1. Resulting in output of $\{c_1, c_2, c_4\}$. Final group's output after completing the for loop iterations will be $\{c_5, c_3, c_1, c_2, c_4\}$. In general, each group can have different watermark value W that can be generated using single secret provided by database owner; however the discussion assumes only assumes watermark value is same for all groups $(G_1, G_2, G_3, ..., G_m)$.

3.1.1 Factorial Number System

Unlike traditional numbering systems, the factorial number system has mixed base. The base of the i^{th} digit is different from the base of j^{th} digit, $\forall\, i \neq j$ (on the contrary, the base of all digits in the binary system is always 2). The value of i^{th} digit is strictly less than or equal to its base value. The weight value of i^{th} digit equals $i!$. A factorial number with n digits is represented as:

$$\sum_{b=1}^{n} [a_b * b!] \tag{1}$$

where b indicates the digit index; a_k is a constant, which can take the values from 0 to b only. For example, the constant a in the least significant digit can take the values 0 or 1, the 2^{nd} digit can take the values 0, 1, or 2; while the constant a in the 3^{rd} digit can take the values 0, 1, 2, or 3. This is different from traditional numbering systems where each digit can take values from 0 to $(base - 1)$. The integer $(349)_{10}$ in the decimal system can be represented in the factorial system as follow:

$$(349)_{10} = (1*1!) + (0*2!) + (2*3!) + (4*4!) + (2*5!) = (24201)_{factorial}$$

Factorial numbering system is a number system where the base of the b^{th} place is $b!$, and the allowed coefficients are between 0 and b.

Algorithm 1: WATERMARKEMBED algorithm.

1: **procedure** WATERMARKEMBED(W, I_o)
2: Divide a column into m groups ($G_1, G_2, G_3, ..., G_m$)
3: C is a set of g data elements that belongs to the same group in the original physical order
4: **for** $i \leftarrow 0$ **to** $m; i \leftarrow i+1$ **do**
5: $Sort(C_i, I_o)$
6: $W_f \leftarrow factoradic(W)$
7: **for** $i \leftarrow 0$ **to** $|W_f|-1$ **do**
8: $LeftCircularShift(W_{d-i-1}, C[i:d])$
9: return(C)

3.2 Integrity Verification

This section shows how integrity of the data residing in secondary storage is checked prior to its use by DBMS. The data integrity can be conducted occasionally or on every read of column by DBMS. The idea is that given a group of data elements of column and the watermark value W the Integrity verification algorithm can recover the reference order I_o. The algorithm INTEGRITYVERIFICATION includes sub-operations: (a) finding group, (b) extracting watermark by (reverse process of WATERMARKEMBED) on group, and (c) ATTACKDETECT procedure detects and localises of attack on the column. To detect and localise an attack, we need to know the I_o and W. First, all the data elements associated with a single group is retrieved [line2]. Then REVERSEWATERMARKEMBED algorithm is performed on the group using secret W [line4]. It is the reverse procedure of Algorithm WATERMARKEMBED. For example, if Algorithm WATERMARKEMBED was used to insert the watermark by left circular shifting of the values by factoradic elements of $W = \{W[0], W[1], W[2], \ldots, W[g-1]\}$, then the reverse algorithm is simply right circular shifting of the factoradic elements of $W = \{W[g-1], W[g-2], W[g-3], \ldots, W[1], W[0]\}$.

Once the watermark is extracted from the group, the ATTACKDETECT procedure is used to detect attack and localise the victim. The ATTACKDETECT is a simple procedure that detect and localise an attack by checking each data element of group. The group content is considered not attacked if all data elements in the group follow the reference order I_o. If any of the data elements don't follow the reference order its location index is extracted to pinpoint the victim of an attack. Detailed example on how localisation can be carried out on a group is presented in Scenario 1 and 2. Hence, an attack will be detected if the de-watermarked group C_r does not follow the reference order I_o that was originally used in WATERMARKEMBED.

Algorithm 2: INTEGRITYVERIFICATION algorithm.

1: **procedure** INTEGRITYVERIFICATION(W, I_o)
2: Divide a column into m groups ($G_1, G_2, G_3, ..., G_m$)
3: C is a set of g data elements that belongs to the same group in the original physical order
4: REVERSEWATERMARKEMBED(C_r, W)
5: **if** ATTACKDETECT(C_r, I_o) == FALSE **then**
6: return(C_r)
7: **else**
8: group C_r has been attacked at location $index, index-1$

The following simple example will better clarify the detection and localisation algorithms. The following are the two possible scenarios that can occur in detecting and localising the attack. After grouping let the $G = \{700, 900, 300, 550, 500, 620, 1000\}$. Assume that the reference order is ascending order. Therefore, the group content after enforcing reference order will be $G = \{300, 500, 550, 620, 700, 900, 1000\}$. The group content should follow the reference order after the REVERSEWATERMARKEMBED algorithm performed on the group.

Scenario 1 (Attribute Attacked, Detected nd Identified): The attacker increased the 2nd element (index starting from 0) 550 by approximately 29.1 percent and the value became 710. Consequently, the group content after REVERSEWATERMARKEMBED becomes $G = \{300, 500, 710, 620, 700, 900, 1000\}$. By inspecting the content, we can see that the 2nd element 710 and the 3rd element 620 is out of the reference order I_o. Therefore, the attack has been detected. To localise the attack, the group content is checked whether the 2nd element has been increased or the 3rd element has been decreased. Therefore, we compare the two possible victim candidates with their respective immediate neighbours and distant neighbours to identify the victim. The immediate neighbours of the 2nd element (710) are the 1st element (500) and the 3rd element (620) and the distant neighbours are the 0th element (300) and the 4th element (700). Similarly, the immediate neighbours of the 3rd element are 710 and 700, while the distant neighbours are 500 and 900. By inspecting the distant neighbours of the 3rd element (500 and 900), it is observed that they follow the reference order, i.e., $500 \leq 620 \leq 900$. While inspecting the 2nd element's distant neighbours, it is clear that they do not follow the reference order, i.e., $300 \leq 710 \nleq 700$. Thus, the 2nd element 710 is the victim. (See Figure 1.)

Scenario 2 (Attribute Attacked, Detected and Identified Up To Two Victims): Once again, the 2nd element (index starting from 0) is attacked, but this time decreased by approximately 11 percent. The

value has been decreased from 550 to 490. The group content after REVERSEWATERMARKEMBED is $G = \{300, 500, 490, 620, 700, 900, 1000\}$, it is clear that the elements in group do not follow the reference order I_o, and hence there has been an attack on the group. The relationship between the 1st and the 2nd elements are not according to reference order. Therefore, these two elements are victim candidates. By inspecting the distant (next to immediate) neighbours of the candidate victims, we can see that the 1st element follows the reference order, i.e., $500 \leq 620$, and so does the 2nd element, i.e., $300 \leq 490 \leq 700$. Thus, in this scenario, we can only localise the attack up to two possible victims instead of pinpointing the exact one.

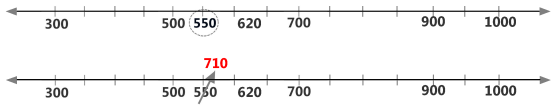

Figure 1: Scenario 1 explained on the number line. In the first line, the group content follows the reference order. In the second line, the 2nd element has been attacked, i.e., its value has been increased from 550 to 710. The attack can be detected because the change in value of 550 leads it to fall outside its immediate neighbour's range of 500 to 620. Furthermore, the victim is identified if the change makes the value fall outside the distant neighbour range, i.e., outside 300 to 700.

In INTEGRITYVERIFICATION first, the group is made associating the first g number of data elements to G_1, subsequently, G_2 is the second batch of data elements in column and so on. Once the group is formed, in the third step, the physical order group content C_r is de-watermarked using REVERSEWATERMARKEMBED. After REVERSEWATERMARKEMBED, the group C_r is checked for attack detection and localisation using ATTACKDETECT. If there is no attack detected, the group content is forwarded for further query processing. Otherwise dropped based on the policies set in DBMS.

3.3 Watermarking Compressed Data

One of the most important properties of the proposed watermarking scheme is it can be applied on compressed database. Also attack detection and victim identification can be performed on the compressed version without need to decompress the database. The proposed algorithms WATERMARKEMBED and INTEGRITYVERIFICATION can be used to watermark compressed data as well as uncompressed data. With a simple pre-processing step, the algorithms can be extended to watermark the compressed data due to the fact that our proposed scheme depends only on the ordering of data elements of column.

The watermark embedding scheme presented in Algorithm 1 can easily be extended to MonetDB patched dictionary compression. Each dictionary word can be considered as an actual value in a page. Therefore, a 1- to 4-byte dictionary word representation of column values can be sorted and modified in terms of location without affecting the compression scheme itself. Our experiments in Section 4 show that the simulation results achieved on the uncompressed data using the original WATERMARKEMBED algorithm (Algorithm 1) are similar to those on MonetDB's compressed data using a slightly modified algorithm with a preprocessing step.

4 EXPERIMENTAL SETUP

To formalise the experimental setup, let P_d be the probability of successful detection of an attack on a group, P_{l1} be the probability of successful localisation of the exact victim, and P_{l2} be the probability of successful localisation of the two possible victims in case the exact victim cannot be pinpointed. Let $\hat{P}_d$, $\hat{P}_{l1}$, and $\hat{P}_{l2}$ be the estimated probabilities.

Let a successful event be the detection and localisation of an attack, while an unsuccessful event be an undetected (and hence not localised) attack. For each group size g, there will be two Bernoulli distributions with P_d and P_{l1}, while $(1-P_d)$ and $(1-P_{l1})$ will be the probabilities of unsuccessful detection and unsuccessful localisation.

All experiments in this study generally involve three steps. Initially, the data are watermarked according to watermarking scheme proposed in Section 3.1. Secondly, the watermarked data are attacked using one of the attack models presented in Section 4.1. Finally, the attacked data are de-watermarked, and the attack is attempted to be detected and localised. Each experiment is repeated 1000 times to estimate the probabilities $\hat{P}_d$ and $\hat{P}_{l1}$. The experiments are carried out with varying parameters of the watermarking scheme. Those parameters are the group size, attack percentage, and standard deviation of the synthetic data. Furthermore, multiple attack models are performed on the synthetic and the real data to estimate the probabilities of detection and localisation. For instance, when the relationship between the group size g vs. the estimated probabilities is examined, for each group size g the experiment is repeated 1000 times (with a different victim randomly selected each time) and the resultant averaged probabilities are calculated.

4.1 Attack Models

For our proposed fragile watermarking scheme, the following attack models are taken into consideration:

Modification Attack: Mallory (attacker) increases or decreases a specific percentage of value of a protected numerical attribute. Only a single victim data element is randomly picked from a set of data elements, and the protected numerical attribute value is attacked (changed).

Superset Attack: Mallory inserts new data elements into the database such that they might affect the database watermark. This attack model is simulated by inserting new data elements at randomly selected locations. The probabilities of detection and localisation are observed for the following numbers of insertions: 1, 2, 4, 8, 16, 25, and 50.

Deletion Attack: Mallory deletes a subset of data elements from the existing group. The data elements to be deleted are selected randomly. The probabilities of detection and localisation are observed for the following numbers of deleted data elements: 1, 2, 4, 8, 16, 25, and 50.

4.2 Results on Synthetic Data

To compare how watermarking of compressed data affects the probabilities of detection and localisation, we carry out experiments on both uncompressed and compressed synthetic data. The synthetic data are tested in order to find the relationship between the parameters (group size, distribution of data, and standard deviation) vs. the probabilities of detection and localisation.

In the following experiments, the relationship between the probabilities of detection $\hat{P}_d$ and exact localisation $\hat{P}_{l1}$ vs. one of the parameters is analyzed while the rest of the parameters are kept constant to some value. For each experiment, the parameters used are kept at their default values according to Table 2 unless mentioned otherwise in the experiment itself.

4.2.1 Uncompressed Data

We first examine the proposed scheme's performance on the raw uncompressed data. First, we test various group sizes g ranging from 1 to 1000. The other parameters are kept at their default values according to Table 2. Only a single element is attacked in the column. From Figure 2(a) it can be observed that the estimated probabilities of detection $\hat{P}_d$ and exact localisation $\hat{P}_{l1}$ of the attack increase as the number of data elements in the group increases. The main reason behind this increasing trend in the probabilities is because in a larger group the attacked victim will disturb the reference order easily since there are more possible values from the same distribution. Small group sizes are tested only to examine the relationship between $\hat{P}_d$ and $\hat{P}_{l1}$ vs. g. It can be noticed from Figure 2(a) that $\hat{P}_d$ and $\hat{P}_{l1}$ can reach 1.0 if g is chosen to be more than 200. The attack on the watermarked attribute values is detected if the reference order I_o is violated. Therefore, it is obvious that as the attack percentage increases, so do $\hat{P}_d$ and $\hat{P}_{l1}$ as depicted in Figure 2(b). In this experiment, the attack percentage is changed as 0.5, 1, 3, 5, 10, 15, 20, 30, 50, and 100, while the other parameters are kept constant at their default values.

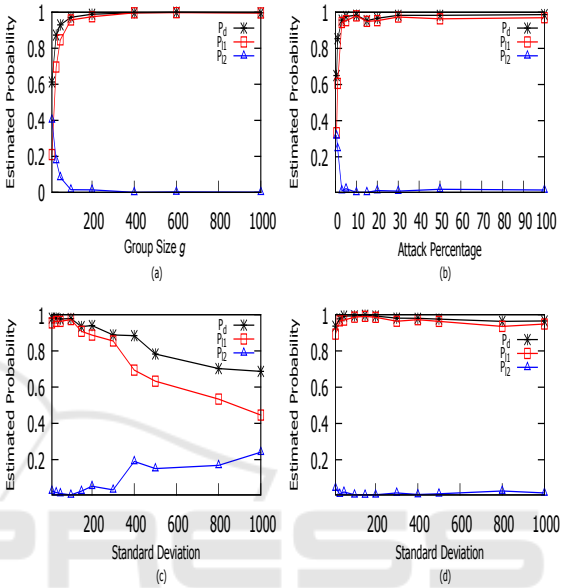

Figure 2: Experimental results for synthetic uncompressed data. Estimated probabilities of detection $\hat{P}_d$ and localisation up to 1 victim $\hat{P}_{l1}$ and up to 2 victims $\hat{P}_{l2}$ for (a) different group sizes, (b) different attack percentages, (c) different standard deviations, and (d) increasing group sizes simultaneously with increasing standard deviations.

Moreover, detection and localisation of attacks rely on the differences between the victim and its immediate neighbour values. Thus, as the standard deviation of the data values stored in the database increases, the consecutive elements' differences also increase, and this negatively affects $\hat{P}_d$ and $\hat{P}_{l1}$. This effect can be resolved if the reference order is based on some cryptographic hash. Such a trend is depicted in Figure 2(c). In reality, different attributes in databases are of varying standard deviations. Therefore, it is unfeasible to make a fixed assumption on the data's standard deviation. The experiment in Figure 2(d) is carried out to show that if we simultaneously increase

Table 2: Default parameter values for synthetic data.

	Data Distribution	Mean	Standard Deviation	Attack Percentage	Group Size
Uncompressed	Normal	1000	10% of mean	5%	100
MonetDB Compressed	Normal	32768	10% of mean	5%	100

the group size as the standard deviation increases, the desired $\hat{P}_d$ and $\hat{P}_{l1}$ can be achieved.

4.2.2 MonetDB Compression

In this section, we will examine the proposed scheme's performance on the compressed data using MonetDB/X100 compression scheme. MonetDB stores the data as a patched dictionary where the most frequent integers or values are represented by 1 to 4 bytes. The vectors that MonetDB/X100 pass around to the CPU are of sizes 256 elements to $16K$ elements per vector, which provide the optimal performance (Boncz et al., 2005). Therefore, to ensure a good performance, the group size g can be kept to the similar size of the vector.

The original synthetic data are not in compressed form. As a preprocessing step, the synthetic data is virtually converted into a 2-byte representation similar to MonetDB's patched dictionary compression scheme. The rest of the experiment is performed in the manner similar to the one for the uncompressed data.

It can be observed in Figure 3(a) that estimated probabilities of detection $\hat{P}_d$ and localisation $\hat{P}_{l1}$ increase with the increase of group size, as expected. Furthermore, these probabilities decrease with the increase of standard deviation as depicted in Figure 3(b). We have observed that the behaviour of our proposed scheme on the MonetDB compressed data is similar that on the uncompressed data. Therefore, it is confirmed that by introducing some minor changes, the proposed Algorithm 1 is easily adaptable to MonetDB's patched dictionary compression with a minimal effect on its operations.

Figures 3(a) and 3(b) are for single element attack where the attacker randomly picks a single attribute value and either increases or decreases the original value by an attack percentage.

Similarly, Mallory can insert new attribute values (data elements) to the database. Such an attack can always be detected because it changes the group's size and can be localised if the inserted values disturb the reference order I_o. However, as the number of inserts increases, the estimated probabilities decrease as shown in Figure 3(c). This is because of inserting new values alongside with their new neighbours, thus maintaining the reference order. It should be noted

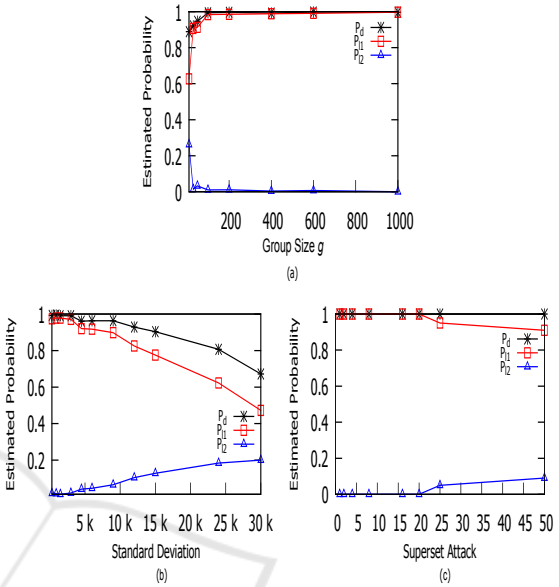

Figure 3: Experimental results for synthetic compressed data with MonetDB/X100 patched dictionary compression. Estimated probabilities of detection $\hat{P}_d$ and localisation up to 1 victim $\hat{P}_{l1}$ and up to 2 victims $\hat{P}_{l2}$ for (a) different group sizes, (b) different standard deviations, and (c) different sizes of insertion attack (superset attack).

that if the group size is chosen to be large enough, all these attacks can be handled and the desired detection and localisation can be achieved. The group size set for experiment in Figures 3(c) is 100 and this is considered minimal when compared to a single page of MonetDB that can store up to 1K attribute values.

The deletion attack model experiment is not shown here because any deleted data elements can be detected with estimated probability of 1.0 but cannot be localised. Such detection is due to the fixed group size, which can clearly indicate that there is/are missing data element(s) based on the group size. Since our watermarking scheme relies on immediate neighbour values, when a data element (value) is deleted, the neighbouring attribute values still follow the reference order. Consequently, the deleted data elements cannot be localised.

4.3 Results on Real-world Data

We use the publicly available "Forest Cover Type" (UCI Knowledge Discovery in Databases Archive,

1999). This dataset was used and tested for watermarking in (Agrawal and Kiernan, 2002) and many other previous works. The dataset contains 581012 tuples. From 61 attributes in the dataset, we focus on two numerical attributes: "Elevation" and "H_Dist_To_Rd". Both attributes exhibit high standard deviations, which made them attractive choices for testing detection and localisation performances of the proposed scheme.

In the experiments, the relationship between the probabilities of detection $\hat{P}_d$ and exact localisation $\hat{P}_{l1}$ vs. one of the parameters is analyzed. For each experiment the parameters used are kept constant according to group size g to 100, attack percentage to five and number of subsets attacked to one unless mentioned otherwise in the experiment itself.

4.3.1 MonetDB Compression

For the MonetDB patched dictionary compressed data, the relationship between the estimated probabilities of detection and localisation vs. the group size g is first analyzed. It can be observed from Figure 4(a) that $\hat{P}_d$ and $\hat{P}_{l1}$ increase with the group size. Furthermore, we tested the attribute H_Dist_To_Rd. It exhibits a very high standard deviation, which is $\sqrt{2431272} = 1559.25$. From Figure 4(b), it can be seen that the estimated probabilities $\hat{P}_d$ and $\hat{P}_{l1}$ do not reach 1.0 until the group size reaches 2000. This phenomenon is due to the high standard deviation of the attribute. However, as we increase the group size, the probability of unsuccessful detection decreases exponentially as depicted in Figure 4(c).

Most of the previously introduced watermarking schemes did not have any considerations on how the underlying data will be affected by compression. In this study, through the experiments on both the synthetic and the real-world data, we have asserted that the proposed watermarking scheme is compatible with the underlying architecture of compressed columnar database such as MonetDB.

5 CONCLUSIONS

In this study, we first examined the architecture of columnar database MonetDB and its associated compression schemes. Then, we introduced a fragile distortion-free watermarking scheme for columnar databases that takes the compression schemes as well as the other watermarking requirements into account. All algorithms in the proposed watermarking scheme are designed to achieve high accuracy (detection rate), high localisability (in pinpointing the tampered data).

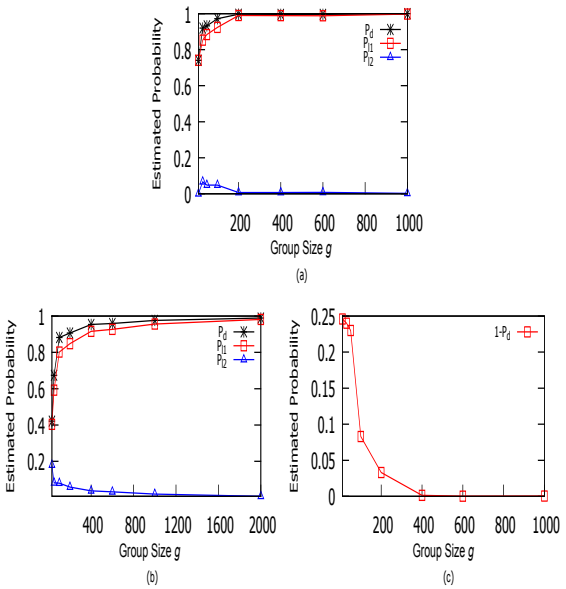

Figure 4: Experimental results for real-world data with MonetDB/X100 patched dictionary compression. Estimated probabilities of detection $\hat{P}_d$ and localisation up to 1 victim $\hat{P}_{l1}$ and up to 2 victims $\hat{P}_{l2}$ for (a) different group sizes in Elevation attribute, (b) different group sizes in H_Dist_To_Rd attribute. Sub-figure (c) shows the estimated probability of unsuccessful detection.

Experimental results on both the synthetic and the real data demonstrated that we have achieved those objectives.

REFERENCES

Abadi, D., Boncz, P., Harizopoulos, S., Idreos, S., and Madden, S. (2012). The design and implementation of modern column-oriented database systems. *Found. Trends Databases*, 5:197–280.

Abadi, D. J., Boncz, P. A., and Harizopoulos, S. (2009). Column-oriented database systems. *PVLDB*, 2:1664–1665.

Abadi, D. J., Madden, S. R., and Hachem, N. (2008). Column-stores vs. row-stores: How different are they really? In *SIGMOD*, pages 967–980.

Agrawal, R. and Kiernan, J. (2002). Watermarking relational databases. In *VLDB*, pages 155–166.

Asikuzzaman, M. and Pickering, M. R. (2017). An overview of digital video watermarking. *IEEE Transactions on Circuits and Systems for Video Technology*, PP(99):1–1.

Bajpai, J. and Kaur, A. (2016). A literature survey - various audio watermarking techniques and their challenges. In *6th International Conference - Cloud System and Big Data Engineering*, pages 451–457.

Bhattacharya, S. and Cortesi, A. (2009). A distortion free

watermark framework for relational databases. In *CISOFT-EA*, pages 229–234.

Boncz, P. A., Zukowski, M., and Nes, N. (2005). MonetDB/X100: Hyper-pipelining query execution. In *CIDR*, pages 225–237.

Camara, L., Li, J., Li, R., and Xie, W. (2014). Distortion-free watermarking approach for relational database integrity checking. *Mathematical problems in engineering*.

Franco-Contreras, J., Coatrieux, G., Cuppens, F., Cuppens-Boulahia, N., and Roux, C. (2014). Robust lossless watermarking of relational databases based on circular histogram modulation. *IEEE Transactions on Information Forensics and Security*, 9(3):397–410.

Guo, H., Li, Y., Liu, A., and Jajodia, S. (2006). A fragile watermarking scheme for detecting malicious modifications of database relations. *Information Sciences*, 176(10):1350 – 1378.

Hu, Z., Cao, Z., and Sun, J. (2009). An image based algorithm for watermarking relational databases. In *2009 International Conference on Measuring Technology and Mechatronics Automation*, pages 425–428.

Kamel, I. (2009). A schema for protecting the integrity of databases. *Computers & Security*, 28(7):698 – 709.

Kamel, I., AlaaEddin, M., Yaqub, W., and Kamel, K. (2016). Distortion-free fragile watermark for relational databases. *International Journal of Big Data Intelligence*, 3(3):190–201.

Kamel, I. and Kamel, K. (2011). Toward protecting the integrity of relational databases. In *Internet Security World Congress*, pages 258–261. IEEE.

Kamel, I., Yaqub, W., and Kamel, K. (2013). An empirical study on the robustness of a fragile watermark for relational databases. In *9th International Conference on Innovations in Information Technology (IIT)*, pages 227–232.

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

Khanna, S. and Zane, F. (2000). Watermarking maps: Hiding information in structured data. In *11th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '00, pages 596–605. Society for Industrial and Applied Mathematics.

Lee, S.-J. and Jung, S.-H. (2001). A survey of watermarking techniques applied to multimedia. In *IEEE International Symposium on Industrial Electronics Proceedings (ISIE)*, pages 272–277.

Li, Y. and Deng, R. H. (2006). Publicly verifiable ownership protection for relational databases. In *ACM Symposium on Information, Computer and Communications Security*, ASIACCS '06, pages 78–89. ACM.

Li, Y., Guo, H., and Jajodia, S. (2004). Tamper detection and localization for categorical data using fragile watermarks. In *4th ACM Workshop on Digital Rights Management*, DRM '04, pages 73–82. ACM.

Pérez Gort, M. L., Feregrino Uribe, C., and Nummenmaa, J. (2017). A minimum distortion: High capacity watermarking technique for relational data. In *5th ACM Workshop on Information Hiding and Multimedia Security*, pages 111–121. ACM.

Potdar, V. M., Han, S., and Chang, E. (2005). A survey of digital image watermarking techniques. In *3rd IEEE International Conference on Industrial Informatics INDIN*, pages 709–716.

Rani, S., Koshley, D. K., and Halder, R. (2017). Partitioning-insensitive watermarking approach for distributed relational databases. In *Transactions on Large-Scale Data-and Knowledge-Centered Systems XXXVI*, pages 172–192. Springer.

Rao, B. V. S. and Prasad, M. V. N. K. (2012). Subset selection approach for watermarking relational databases. In *Data Engineering and Management*, pages 181–188. Springer Berlin Heidelberg.

UCI Knowledge Discovery in Databases Archive (1999). Forest CoverType.

Wang, C., Wang, J., Zhou, M., Chen, G., and Li, D. (2008). Atbam: An arnold transform based method on watermarking relational data. In *2008 International Conference on Multimedia and Ubiquitous Engineering (MUE)*, pages 263–270.

Zhang, Y., Niu, X., Zhao, D., Li, J., and Liu, S. (2006). Relational databases watermark technique based on content characteristic. In *First International Conference on Innovative Computing, Information and Control - Volume I (ICICIC'06)*, pages 677–680.