

**Robust and Invisible Watermarking  
Algorithm for Relational Databases  
Copyright Protection Using Arnold  
Transform**

**Nehad Hameed Hussein**

**Computer Engineering Department, Baghdad College  
of Economic Sciences  
Baghdad, Iraq**

**Abstract**—Digital watermarking for relational databases appeared as a candidate solution to provide copyright protection, tamper detection, and maintaining integrity of

relational data. In this paper, an efficient database watermarking algorithm is proposed that based on embedding the binary image and secret text message in non-numeric mutli-word attributes of all database tuples. The binary image is scrambled first using Arnold transform to guarantee security of watermark even when the database is hacked. After that the tuples are partitioned logically using MD5 hash function using secret parameters.

The algorithm takes advantage of the fact, that modern text encoding systems like Unicode offer multiple different characters describing whitespaces. Normal space and thin space characters are selected for representing and embedding watermark bits 0 and 1, respectively. These will not affect the size of database and ensure that the non-whitespace characters aren't modified. The scheme is non-intrusive, resilient, blind, reversible and suitable for databases of any size with reasonable performance on embedding and extraction. Moreover, it supports incremental watermarking to manage with the dynamic nature database systems are subject to. Experimental results demonstrated the efficiency and robustness of the algorithm against common database attacks and is free distortion and imperceptible, as proved by the analysis.

### الخلاصة

اصبح استخدام العلامات المائية ضرورة ملحة في الوقت الحالي لغرض ضمان حماية ملكية البيانات من السرقة وادعاء الملكية الخاطئة. في هذا البحث تم تسليط الضوء على استخدام العلامات المائية في اثبات اصالة قواعد البيانات العلائقية من خلال اخفاء صورة رقمية من النوع الثنائي مع رسالة نصية قصيرة ضمن قاعدة البيانات المعنية. لغرض ضمان امنية العلامة المائية يتم اولا عمل تغيير في مواقع عناصر الصورة وتغيير تسلسلها من خلال خوارزمية Arnold وبعد ذلك اضافة الرسالة اليها ثم تضمينها في قاعدة البيانات. تم الاعتماد على استخدام البيانات المخزنة في قاعدة البيانات من النوع النصي المتعدد حيث يتم اولا تغيير تسلسل الصفوف الموجودة في قاعدة البيانات بطريقة عشوائية بالاعتماد على دالة MD5 وتم اخفاء الصورة والرسالة في الفراغات الموجودة بين الكلمات وتوزيعها بصورة عشوائية بالاعتماد على دالة توزيع متباينة. يتم اخفاء البيانات في كل الصفوف في وقت واحد لضمان حماية قاعدة البيانات من مختلف انواع الهجمات التي تحدث على قواعد البيانات كاضافة صفوف جديدة او مسح صفوف موجودة او تغيير في صفوف معينة. الخوارزمية المطبقة تضمن ان تكون العلامة المائية مخفية وغير قابلة للاكتشاف لاعتمادها تخزين العلامة المائية بطريقة غير مرئية في الفراغات بين الكلمات.

### Keywords

**-Relational database, watermark, Copyright protection, MD5, Arnold transform, Multi-words nonnumeric attributes.**

## I. INTRODUCTION

The unlimited growth of the Internet opens a wide range of web-based services, such as database as a service, digital repositories, e-commerce, online decision support system etc. These applications make the digital data, such as images, video, audio, database content etc. easily accessible and usable by ordinary people around the world. As a result, such digital products are facing serious challenges like piracy, redistribution, ownership claiming, forgery, theft etc. Digital watermarking is an effective solution to meet such challenges.

Digital watermarking is a technique which can protect the data like images, video, audio, database, etc. ownership by embedding watermark into the data of database [1]. It allows the user to add a layer of protection to the digital content by identifying copyright ownership and delivering a tracking capability. The watermark can be any kind of information that is embedded into database. In the relational database applications the digital watermarks are useful in many practices, including: ownership Assertion, fingerprinting, and fraud and tamper Detection [2]. The watermark should not significantly affect the quality of original data and should not be able to destroy easily [3].

But compared with multimedia data, digital watermarking of the relational database is more difficult due to many features of relational database data: (1) The relational database data is composed of some independent tuple. (2) The value of each field of each tuple is certain, and the redundancy is small. (3) The order of the tuples of the relational database is in disorder. Frequent Updating: tuples may be inserted, deleted, or updated during normal database operations [1]. Due to these differences between relational and multimedia data, there exist no images or audio watermarking methods which are suitable for watermarking of relational databases [2].

In this study, we proposed a relational database robust invisible watermarking algorithm based on using the binary image and secret text message as watermark and embedding it in multi-words non-numeric attributes. First, the binary image is scrambled k-times using Arnold transform then concatenated with secret text message for generating the binary watermark. After that partitioning the database tuples into uniform distributed partitions using MD5 hash function and embedding one bit from watermark into all tuples within each partition. The embedding process is achieved through representing the watermark bits using Unicode whitespaces in the multi-words nonnumeric attribute. The approach is free-distortions by just substituting whitespace characters. This will minimize visible recognizability and ensure that the non-whitespace characters aren't modified. The algorithm is robust against common attacks such as subset addition, alteration and deletion attacks.

The rest of the paper is organized as follows. Section II gives an overview of the related work. Section III describes the detail of the used methodology and the proposed algorithms for watermark encoding and decoding. Section IV shows the experimental results and analysis. Section V concludes the method and gives suggestions for future work.

## II. RELATED WORKS

Originally, most of the work on watermarking was expended in the watermarking of text, digital images, video, and audio like [5-7, 20]. In the recent years watermarking of database systems started to receive attention because of the increasing use of it in many real-life applications. In 2002 Agrawal and Kiernan [8] proposed a watermarking algorithm that embeds the watermark bits in the least significant bits (LSB) of selected attributes of a selected subset of tuples. They based on numerical attribute and used HASH function for database partitioning. [1, 2, 4, 9, 10] are examples of other works based on this bit level watermarking method.

In 2006 Zhang et al. [11] suggested databases watermarking technique based on content characteristic. The watermark insertion phase extracts some bits, called local characteristic, from the characteristic attribute of tuple and embeds those bits into the watermark attribute of the same tuple. In 2007 Xiao et al. introduced a novel robust watermarking algorithm based on the second-LSB of numeric attribute for embedding the images watermark bits [12].

In 2008 Al-Haj and Odeh [9] proposed a watermarking scheme which is based on hiding binary image in spaces of non-numeric multi-words attributes of subsets of tuples. The database is divided into non-intersecting subsets of tuples. The short strings of the watermark image are embedded into each tuple subset. In 2008 Yang W. et al [13] proposed the use the owner's speech to generate unique watermark. The speech is prepared and watermark is generated. The bit-level marking is performed during watermark embedding phase in the LSB of the selected numeric attribute.

In 2009 the approach in [14] was aimed to generate fake tuples and insert them erroneously into the database. Rather than inserting fake tuples, the author in [15] proposed another watermarking technique by inserting a virtual attribute in the relation which will serve as watermark containing parity checksum of all other attributes and an aggregate value obtained from any one of the numeric attribute of all tuples.

In 2010, Hanyurwimfura et al [16] proposed a relational database watermarking method for non-numeric multi-words attributes. The watermark is embedded by horizontally shifting the location of a word within selected attribute of selected tuples. The location where the watermark to be inserted is determined by the Levenshtein distance between two successive words.

In 2011, Li M. et al [17] proposed an asymmetric watermarking scheme for integrity verification of database. The watermarked database can be generated only by the owner with the private key and can be verified by the public with the public key. Zhang L. et al [2] suggested a new method for protecting both textual and numerical data of relational databases. This is done by embedding special mark and watermark bits into textual attributes and numerical attributes respectively.

Wang et al [10] proposed in 2012 the watermark embedding occurs by Arnold transforming and scrambling technology and modifying the parity of the low decimal number of numeric attribute, connected with some attribute in the physical storage space in the relational databases. In 2013, [4] introduced watermark algorithm by based on the analysis of the value of the multi-words attributes by computing the statistics of characters and finds the corresponding ASCII values of them and embeds it. In 2014,

El-Haggar et al [18] based on the selected multi-words attribute to generate the watermark and embeds it by changing the state of bits by XORing it.

Our contribution focuses on embedding the watermark in the multi-words nonnumeric attributes values only without affecting the size and visibility of the attribute value. The suggested algorithm uses Unicode whitespace characters for representing the watermark bits and substitutes them with the whitespaces between words in the multi-word attributes values.

In order to improve the invisibility and randomness of the watermark, the algorithm scrambles the watermark image to eliminate the correlation of each pixel and make the distribution of pixels unsystematic. Even if the database is damaged in the course of regular use, the extracted damaged watermark bits are distributed the whole image after performing the inverse Arnold transformation, which is not obvious to human visual system. At the same time, the extracted watermark is scrambled image, attackers do not know how to recover the original image at all. Moreover, a tuple partitioning scheme is based on the watermark length as additional private parameter. The watermark embeds one bit per partition, wherein every tuple holds this bit.

### III.DATABASE WATERMARKING ALGORITHM

#### A. Notations

Suppose the relational database  $D = (PK, A_0, A_1, \dots, A_{n-1})$ , in which the  $PK$  is the primary key attribute,  $A_0, \dots, A_{n-1}$  are  $n$  other attributes. The database has  $k$  tuples where, the  $\Omega_i$  is  $i^{th}$  tuple in the relational database ( $1 \leq i \leq k$ ). We denote by  $\Omega.PK$ ,  $\Omega.A_i$  the value of the  $PK$  and the candidate attribute  $A_i$  respectively in tuple. We assume there is at least one multi-words non-numeric attribute in the database called  $A^m$  which used for watermark embedding. For easy reference, table 1 lists the symbols used in the suggested algorithm.

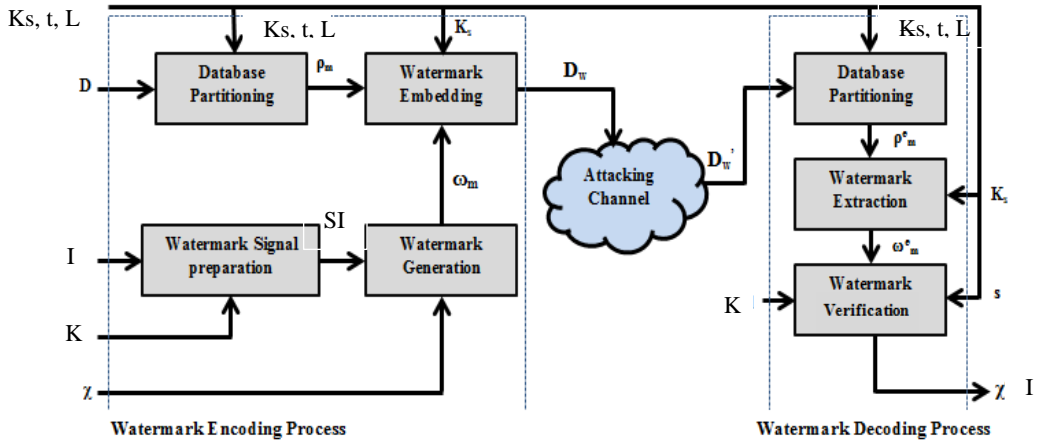
TABLE 1: PAPER NOTATIONS

Symbol	Description
D	Unwatermarked relational database
$D_w$	Watermarked database
$\omega_L$	Binary Watermark
$\Omega_i$	$i^{th}$ tuple in D
L	Watermark length
PK	Primary Key
$K_s$	Secret Key
$A^m$	Multi-words nonnumeric attribute used for watermark embedding
$A_i$	$i^{th}$ attribute in D
$\rho_L$	Database Partitions
t	Number of bits in secret text message
K	Number of scrambling times
$\chi_T$	Secret message
$I_{M \times N}$	Binary watermark image
$SI_{M \times N}$	Scrambled Watermark Image
$\beta$	Position within $A^m$ to hide a bit
$\mu$	Whitespaces count in $A^m$ value
$\omega_m^e$	Extracted Binary Watermark
$\chi_t^e$	Extracted secret message

### B. Approach Overview

Figure 1 shows the block diagram summarizing the main components of our watermarking technique. The concatenation of the binary image and secret text message is used to generate the watermark bits  $\omega$ . A robust watermark algorithm is used to embed invisible free-distortion watermark bits into the relational database.

The using of secret text message with binary image will provide two-ways for database authenticity verification through making the watermark more robust against potential attacks where the attacker should remove both secret text message and binary image to breaking the watermark.



**Fig. 1 Stages of watermark encoding and decoding processes.**

Mainly, the watermark algorithm consists of two processes are: watermark encoding, and watermark decoding processes. Watermarking encoding process used for embedding the watermark in  $D$  using the secret parameters. However, it can be summarized in the following steps:

**Watermark Generation:** the watermark used in the algorithm is binary image  $I_{M \times N}$  and secret text message  $\chi$ , the binary image will concatenate with secret text message to generate the final watermark. The generated watermark will have  $L$ -bits length.

**Database Partitioning:** The database  $D$  is partitioned into  $m$  partitions by using the secret key  $K_s$  and binary image size in conjunction with a MD5 cryptographic secure hash function. The partitions namely  $\{\rho_0, \rho_1, \dots, \rho_{m-1}\}$ . Each partition  $\rho_i$  contains on the average  $|D|/m$  tuples approximately.

**Watermark Embedding:** The watermark bits  $\omega_m$  are embedded in the database using a robust embedding function. The suggested technique embeds the watermark bit  $\omega_i$  within every tuple in the corresponding partition  $\rho_i$ , which means the first watermark bit  $\omega_0$  is embedded in every tuple in first partition  $\rho_0$  and so on. The embedding process achieves through whitespaces characters substitution in the multi-words nonnumeric attribute in the database.

Watermark decoding is the process of extracting the embedded watermark from the watermarked database  $D_w$ , using the secret parameters: the secret key  $K_s$ , the binary image size, and watermark length  $L$ . The decoding algorithm is blind as the original database  $D$  is not needed for successfully decoding the embedded watermark. The decoding process can be shortened in the following steps:

**Database Partitioning:** The watermarked database  $D_w$  is partitioned by using the same database partition algorithm as in the watermark encoding phase.

**Watermark Extraction:** in this stage, the majority voting [3, 4, 10, 14, 19, 21] will be used for detecting the watermark bits. The inserted watermark bits  $\omega_i$  are extracted from all the tuples in each partition and compared. In the majority voting, the errors are mostly eliminated.

**Watermark Verification:** after extracting the watermark bits, this process will start that includes the splitting of the watermark  $\omega_m^e$  into scrambled image and the secret message from and verifying them after descrambling the image.

### C. Watermark Generation

The watermark used in this paper  $\omega_m$  is the concatenation of the binary image  $I_{M \times N}$  and secret text message  $\chi_t$ . The generated watermark has  $(M * N) + t$  bits, where  $(M * N)$  is the number of bits in binary image and  $t$  is the number of bits in secret message.

The length of watermark  $\omega$  will be used in the database partitioning process for partitioning the database to  $m$  partitions.

[Definition #1] *Watermark Generation function.* A watermark generation function  $\Psi$  is used for generating the  $L$ -bits long binary watermark vector  $\omega_L \{\omega_0 \omega_1 \omega_2 \dots \omega_{L-1}\}$  through concatenating the scrambled image  $SI_{M \times N}$  and the secret text message  $\chi_t$  for

$$\Psi: (SI, \chi) \rightarrow \{\omega_0 \omega_1 \dots \omega_{L-1}\} \quad (1)$$

The watermark generation function  $\Psi$  performs  $k$ -times Arnold transform on the binary image and concatenates the scrambled image with the secret text message  $\chi_t$ .

The secret message  $\chi_t$  will be decomposed into two equal size strings and padded in the start and the end of the  $\lambda_s$  for generating the final watermark. This is shown clearly in the expression below and the complete algorithm in figure 2.

$$\{\chi_0 \chi_1 \chi_2 \dots \chi_{(t/2-1)}\} \parallel \{SI_{(0,0)} SI_{(0,1)} SI_{(0,2)} \dots SI_{(m-1,n-1)}\} \parallel \{\chi_{(t/2)} \chi_{(t/2+1)} \chi_{(t/2+2)} \dots \chi_{(t-1)}\} \rightarrow \{\omega_0 \omega_1 \omega_2 \dots \omega_{L-1}\}$$



**ALGORITHM 1: WATERMARK GENERATING****Inputs:** Binary Image  $I_{M \times N}$ , secret text message  $\chi_b$  and  $k$ -times Arnold scrambling.**Output:** watermark length  $L$ , watermark bits  $\{\omega_0 \omega_1 \dots \omega_{L-1}\}$ **1:**  $\{\omega_0 \omega_1 \omega_2 \dots \omega_{L-1}\} \rightarrow \{ \}$ ,**2:**  $L = 0$ ;**3:**  $x = \text{Size}(I_{M \times N})$ **4:**  $t = \text{Length}(\chi)$ **5:**  $x + t \rightarrow L$ **6:**  $\text{SI} = \text{Arnold}(\mathbf{I}, \mathbf{K})$ ;**7:**  $\{\chi_0 \chi_1 \dots \chi_{(t/2-1)}\} \parallel \{\text{SI}_{(0,0)} \text{SI}_{(0,1)} \dots \text{SI}_{(m-1,n-1)}\} \parallel \{\chi_{t/2} \chi_{(t/2+1)} \dots \chi_{(t-1)}\} \rightarrow \{\omega_0 \omega_1 \dots \omega_{L-1}\}$ **8: Return:**  $L, \{\omega_0 \omega_1 \dots \omega_{L-1}\}$ **Fig. 2 Watermark generation algorithm****D. Database Partitioning**

The database  $D$  is partitioned into  $L$  partitions by using the secret key  $K_s$  in conjunction with a cryptographic hash function. The generated partitions namely  $\{\rho_0, \rho_1, \dots, \rho_{L-1}\}$  such that for any two partitions  $\rho_i \cap \rho_j = \emptyset$ . The partition sets must be non-empty and their union leads to  $D$  such that  $\rho_0 \cup \rho_1 \cup \dots \cup \rho_{L-1} = D$ . Moreover, each partition  $\rho_i$  contains on the average  $|D|/L$  tuples approximately.

[Definition #2] *partitioning methods:* Given  $D$ , a database of  $n$  tuples and the number of partitions  $L$ , a partitioning algorithm partitions the tuples into  $L$  logical non-overlapping partitions ( $L \leq n$ ).

Cryptographic hash function Message Digest (MD5) is widely used in the database partitioning [1, 4, 17, 19, 21, 22]. It takes an input (or 'message') and returns a fixed-size string, which is called the hash value or MAC (Message Authentication Code). MD5 is a widely used with a 128-bit hash value. It has the additional characteristics that i) given  $Y$ , it is easy to compute  $h$ , and ii) given  $h$ , it is hard to compute  $Y$  such that  $H(Y)=h$ , and iii) given  $Y$ , it is hard to find another message  $Y'$  such that  $H(Y)=H(Y')$ .

The partitioning algorithm used here partitions database  $D$  into  $L$  logical groups by using algorithm proposed in [21] with some refinements. Partitioning is based on a secret key  $K_s$  and MD5.

The number of partitions will depend on the length of watermark  $L$  to partitioning the database into equal number of the length (number of bits) of watermark. Figure 3 shows the database partitioning algorithm.

In order to increase the security, the secret message length  $t$  will be concatenated with the  $K_s$  as input to the hash function. An attacker cannot predict the tuple belongs to any  $i^{\text{th}}$  partition without the knowledge of the secret key  $K_s$ , the watermark length  $L$ , and the secret message length  $t$  which are kept secret to the database owner only.

[Definition #3]. Hash function: A hash function  $H$  maps a variable-size input  $Y$  to a fixed-size string  $h$ , called the hash value, as:

$$H: Y \rightarrow h \quad (2)$$

For each tuple  $\Omega \in D$ , the data partitioning algorithm computes MAC in order to assign tuples to the partitions using a hash function  $H$  as

$$Partition(\Omega) = H((K_s \parallel t) \parallel H(\Omega.PK \parallel K_s)) \text{ MOD } L \quad (3)$$

Where  $\Omega.PK$  is the primary key of the tuple  $\Omega$ ,  $H()$  is a hash function,  $t$  is the number of bits of the secret message,  $L$  is watermark length, and  $\parallel$  is the concatenation operator.

So to get MD5 hash value, first compute the hash value for the concatenation of  $\Omega.PK$  with  $K_s$  then compute the hash value of the concatenation of  $K_s$ ,  $t$  and the computed hash value of  $\Omega.PK$  and  $K_s$ . The remainder of the hash value with  $L$  is the final MAC.

$$(4) \quad \forall \Omega_j \in D \left\{ \begin{array}{l} \Omega_j \in \rho_i \text{ if: } H((K_s \parallel t) \parallel H(\Omega_j.PK \parallel K_s)) \text{ MOD } L = i \\ \Omega_j \notin \rho_i \text{ otherwise} \end{array} \right.$$

Where  $1 \leq j \leq n$  and  $0 \leq i \leq L-1$

#### ALGORITHM 2: DATABASE PARTITIONING

**Inputs:** database  $D$ , secret key  $K_s$ , secret message length  $t$ , and the watermark length  $L$

**Output:** Data partitions  $\rho_0 \rho_1 \dots, \rho_{L-1}$

- 1:  $\{\rho_0 \rho_1 \dots \rho_{L-1}\} \rightarrow \{\}$
- 2: For each  $\Omega \in D$  do
- 3: Part( $\Omega$ ) =  $H((K_s \parallel t) \parallel H(\Omega.PK \parallel K_s)) \text{ MOD } L$
- 4: Insert  $\Omega$  into  $\rho_{part(\Omega)}$
- 5: End For
- 6: Return  $\{\rho_0 \rho_1 \dots, \rho_{L-1}\}$

Fig. 3 Database partitioning algorithm

#### E. Watermark Embedding:

In watermark embedding stage the watermark  $\omega_L$  is inserted into a database  $D$  in form of 0,1 sequences. The watermark embedding process is a *character substitution process*. The essential of our embedding algorithm is embedding the watermark bits through inserting the bits as whitespaces within multi-words non-numeric attributes values. We take advantage of the fact, that modern text encoding systems like Unicode has multiple different characters for describing whitespaces, whose difference is in the character width only.

We specifically choose two particular Unicode whitespace characters, which are mapped to bits 0 and 1 of watermark. The two selected characters are normal space (u+0200) and thin space (u+2009) for representing watermark bits 0 and 1, respectively. The thin space is selected because its space width (1.5 em) is very near to the width of the normal space (1.4 em).

This makes the difference is not visibly recognizable in width between the adjacent words by humans eyes. This embedding process makes the watermark is invisible and undetectable by Human Visual System (HVS) through substituting the normal whitespaces with thin spaces.

The suggested technique embeds the  $i^{th}$  bit from the watermark  $\omega_L$  within every tuple in the  $i^{th}$  partition  $\rho_i$ , where the first watermark bit  $\omega_0$  is embedded in every tuple  $\Omega$  in first partition  $\rho_0$  and the second watermark bit  $\omega_1$  is embedded in every tuple in second partition  $\rho_1$  and so on. This will help in detecting and preventing the potential attacks like subset tuples insertion, deletion, or update.

$$\forall \Omega_j \in \rho_i : \omega_i \rightarrow \rho_i \cdot \Gamma_j \cdot A^m \cdot \mu_j \quad \begin{matrix} 1 \leq j \leq n \\ 0 \leq i \leq L \end{matrix} \quad (5)$$

Where  $\Omega_j$  is the  $j^{th}$  tuple within  $i^{th}$  partition  $\rho_i$  and  $\mu$  is the total number of whitespaces in the  $i^{th}$  tuple.  $n$  is the number of tuples within each partition and  $L$  is the number of partitions.

[Definition #4] *Watermark embedding function: watermark embedding function  $\phi$  transforms the input database  $D$  to a watermarked database  $D_W$  after performing some data computations and substitutions. Formally,*

$$\phi: (D, \omega_m) \rightarrow D_W \quad (6)$$

[Definition #5]. *Whitespace position function: whitespace position function  $\Upsilon$  returns the position of a whitespace character within the value of the multi-words non-numeric attribute which will subsequently be substituted.*

$$\Upsilon: (PK_j, i, L, \mu_j) \rightarrow \beta \quad (7)$$

For embedding  $\omega_i$ , we use  $i, n, PK_j, \mu_j$  as inputs parameters to the whitespace position function  $\Upsilon$ , which returns the exact location to hide the current  $\omega_i$ . In our algorithm,  $\beta$  is the position of a whitespace character which will subsequently be substituted.

Function (7) is used for computing the position of the whitespace used for watermark substitution in the multi-words non-numeric attribute.  $PK_j$  the primary key of the  $j^{th}$  tuple,  $i$  is the tuple's MAC  $0 \leq i \leq L-1$ ,  $n$  is the number of tuples in partition MAC, and  $\mu_j$  is the total number of whitespaces in  $A_j^m$  attribute.

$$\Upsilon(PK_j, i, n, \mu_j) = ((PK_j \bmod n) \text{ XOR } (n \bmod (i+1))) \bmod \mu_j \quad (8)$$

The dependency of function  $\gamma$  on the number of tuples in each partition will make the detection of tuples insertion and deletion attacks more efficient, where inserting new tuples or deleting ones will cause to changing the number of tuples in each partition and hence changing the computed value of  $\beta$  for each tuple in the watermark decoding process.

Equation (8) selects a whitespace from the given attribute for substituting it with the Unicode whitespace that representing the watermark bit. Since  $\gamma$  is influenced by the number of the current partition and the number of tuples in that partition, it is also influenced by the secret key as shown in equation (7) and therefore unpredictable. Moreover, since the result of  $\gamma$  is affected by the PK of the current tuple, it ensures that different tuples which are within the same partition and having similar values for  $A_j^m$  will still result in different whitespace locations. Figure 4 shows the watermark embedding algorithm. The watermark embedding process includes the following steps:

**Step 1: Initialize variables  $n, \beta, \mu$ :**

**Step 2: Iterate through watermark  $\omega_i \in \omega_L$ :**

- A. Iterate through partitions  $\rho_i \in \rho_L$ :
  - i) Iterate through the tuples in the  $i^{th}$  partition where  $\Omega \in \rho_i$ :
    - a. Calculate the total number of tuples  $n$  in the  $i^{th}$  partition  $\rho_i$
- B. Iterate through each tuple in  $\rho_i$  where  $\Omega_j \in \rho_i$ :
  - a. Calculate the total number of whitespaces  $\mu_j$  within the value of attribute  $A_j^m$ .
  - b. Calculate whitespace location  $\beta$  used for embedding watermark bit  $\omega_i$  according to Equation (8).
  - c. Substitute the whitespace at the previously calculated location with  $\omega_i$  representing Unicode whitespace.
    - i. If  $\omega_i = 0$  then: substitute the whitespace in  $\beta_j$  location within  $A_j^m$  by normal space (u + 0020)
    - ii. If  $\omega_i = 1$  then: substitute the whitespace in  $\beta_j$  location within  $A_j^m$  by thin space (u + 2009)

**ALGORITHM 3: WATERMARK EMBEDDING***Inputs: (Database Partitions  $\rho_L$ , Watermark  $\omega_L$ , Secret key  $K_s$ , and embedding Attribute  $A_j^m$ )**Output: Watermarked Database  $D_w$* **1:**  $n=0$ ;  $\mu=0$ ;  $\beta=0$ ;**2:** For each  $\omega_i \in \omega_L$  do    **3:** For each  $\rho_i \in \rho_L$  do        **4:** For each  $\Omega \in \rho_i$  do            **5:**  $n= n + 1$ ; // Return the number of tuples in the  $\rho_i$             **6:** } // End third For**7:** For each  $\Omega_j \in \rho_i$  do**8:**  $\mu_j= \text{count}(A_j^m)$  //Calculate the number of whitespaces in  $A_j^m$ **9:**  $\beta_j = ((PK_j \bmod n) \text{ XOR } (n \bmod (i+1))) \bmod \mu_j$  // Return the location of watermark embedding position**10:** If  $\omega_i=0$  then**11:**  $[u + 0020] \rightarrow A_j^m.\beta_j$  //Substitute the whitespace in  $\beta_j$  within  $A_j^m$  by normal space ( $u+0020$ )**12:** End If**13:** Else If  $\omega_i=1$  then**14:**  $[u + 2009] \rightarrow A_j^m.\beta_j$  //Substitute the whitespace in  $\beta_j$  within  $A_j^m$  by thin space ( $u+2009$ )**15:** End If**16:** } // End fourth For**17:** } // End second For**18:** } // End first For**19:** Return: Watermarked database  $D_w$ **Fig 4 Watermark embedding algorithm.**

Choosing two characters for representing the whitespaces in the text offers an additional advantage, that is, potential copyright breach can even be detected when the watermarked database has not been duplicated as complete data, but parts of its textual contents have been copied and pasted. Also the number and position of attacked tuple(s) can be determined precisely through following the location of embedded whitespaces changing in the watermark encoding process.

**F. Watermark Decoding**

The watermark decoding algorithm extracts the embedded watermark  $\omega^e \{\omega_0^e \dots \omega_{L-1}^e\}$  using the secret parameters:  $K_s$ ,  $L$ , and  $t$ . The algorithm starts by generating the data partitions  $\rho_0, \dots, \rho_{L-1}$  using the watermarked database  $D_w$ ,  $K_s$ ,  $L$ , and  $t$ .

The next stage after database partitioning is watermark extraction stage. During watermark encoding process each watermark bit  $\omega_i$  is embedded in every tuples within  $i^{th}$  partition  $\rho_i$ , so that the watermark bits will be extracted from all tuples within same partition and computed the number of ones and zeros. If at least one bit differs from the other extracted bits, this means the database *might* be exposed to some attack. This is solved by majority voting technique.

Finally, the last stage is watermark verification stage that will be used for regenerating the secret message and binary image from the extracted bits.

### G. Watermark Extraction

The watermark extraction process is accomplished through majority voting. The majority voting mechanism is used for voting to the value with more occurrences in the set of values. As the watermark bits are embedded  $L$  times in the database, each watermark bit is extracted  $n$  times ( $n$  the number of tuples in each partition), where for an embedded watermark bit  $\omega_i^e$ , it is extracted from partition  $\rho_j$ , where  $j \bmod L = i$ . The extracted bits are decoded using the majority voting mechanism.

[Definition #6] *Majority voting: Let values set  $V_n = \{V_0 V_1 \dots V_{n-1}\}$ , and the values of  $V$  are  $X$  or  $Y$  only. The majority voting function will compute the times of occurrences of  $X$  and  $Y$  in  $V$  and return the value with more occurrences. It returns  $X$  if  $X_{numbers} > Y_{numbers}$ , and  $Y$  if  $Y_{numbers} > X_{numbers}$ .*

For each marked bit, count the numbers of its value to be zeroes or ones respectively, and then a majority voting mechanism is used to decide the final value of this bit. It votes the value (0 or 1) with largest count to be assigned to the extracted watermark bit  $\omega_i^e$ . The majority voting technique is illustrated by the example in Fig. 5, where the watermark length is 5-bits and the number of tuples in  $i^{th}$  partition is 6.

Tuples/watermark	$\omega_0$	$\omega_1$	$\omega_2$	$\omega_3$	$\omega_4$
$\Omega_0$	0	1	1	0	1
$\Omega_1$	1	0	0	1	1
$\Omega_2$	1	0	0	1	1
$\Omega_3$	1	1	0	0	1
$\Omega_4$	0	0	0	0	1
$\Omega_5$	1	1	1	0	0
<b>Watermark</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>

Fig 5 Majority voting mechanism example.

[Definition #7] *Watermark extraction function: it extracts the watermark  $\omega^e: \{\omega_0^e \omega_1^e \dots \omega_{L-1}^e\}$  from the watermarked database  $D_w$  using majority voting function.*

$$\mathcal{F}: D_w \rightarrow \omega^e \quad (9)$$

The partitioning process used in watermark encoding process has to be done prior to watermark extraction by using the function given in Equation (3). After that, the number of tuples in the current partition is iteratively calculated and extract the watermarked bit embedded within every tuple of the current partition (which should be the same for every tuple within current partition) using the function given in Equation (8) and count the occurrences of zeroes and ones.

After the extraction for all tuples of the current partition has been done, the variable whose counter is higher (zeroes or ones) is chosen to be the extracted bit of the watermark for the current partition, and the corresponding bit value is concatenated to a string variable  $\omega^e$ . Figure 6 depicts the algorithm of watermark extraction process. However, the steps of the watermark extraction process are:

**Step 1: Initialize variables  $n, \beta, \mu, ones, zeros == 0$ :**

**Step 2: Initialize watermark  $\omega_L^e = \{ \}$ :**

**Step 3: Partition the watermarked database  $D_w$  according to database partitioning Eq. (3)**

**Step 4: Iterate through  $i^{th}$  partition  $\rho_i \in \rho_L$ :**

- i. Iterate through the tuples in the  $i^{th}$  partition  $\rho_i$  where  $\Omega_1 \in \rho_i$ :
  - a. Calculate the number of tuples  $L$  in the  $i^{th}$  partition  $\rho_i$
- ii. Iterate through each tuple in  $\rho_i$  where  $\Omega_j \in \rho_i$ :
  - a. Calculate the total number of whitespaces  $\mu_j$  within attribute  $A_j^m$ :
  - b. Calculate whitespace location within  $A_j^m$  used for watermarking according to Eq. (8).
- iii. If  $A_j^m$  at location  $\Upsilon(PK_j, i, n, \mu_j) = /u + 0020$  then: zeros = zeros + 1;
- iv. Else if  $A_j^m$  at location  $\Upsilon(PK_j, i, L, \mu_j) = /u + 2009$  then: ones = ones + 1;
- v. Use majority voting mechanism for selecting  $\omega_i^e$  value:
  - a. If zeros > ones: the bit extracted from the  $i^{th}$  partition is 0 hence:  $\omega_i^e = 0$ ,
  - b. Otherwise  $\omega_i^e = 1$ .
  - c. Insert the extracted bit  $\omega_i^e$  in the  $\omega_L^e$

Before watermark verification process, the watermarked relation should be restored to its original form through reversing process. Reversing process is very simple, since all thin spaces have to be substituted with regular spaces. On the one hand, reversal can be a highly commended feature if the removal of all watermarking-based distortions represents a requirement.

Modifications of the marked data which alter bits in the watermark can be localized precisely. Since the  $\omega_i$  is embedded many times within  $\rho_i$ , the extraction process results in a series of multiple extracted watermarks. Since one bit of the watermark is embedded in all tuples of an entire partition, we can easily localize the  $i^{th}$  partition that has the irregular modification.

Moreover, once the partition in which modifications has obviously been localized, the localization process can be further refined by sequentially analyzing each tuple of the  $i^{th}$  partition and determine the tuple(s) that has the modification.

Depending on the watermark length  $L$  in the partitioning algorithm makes the localization procedure normally quite accurate since the number of partitions is very high, resulting in less tuples per partition and therefore limiting the number of tuples, modifications may have been introduced to.

**H. Watermark Verification:**

After extracting the watermark  $\omega^e_L$ , this process will start by the extracting of the secret text message  $\chi^e_t$  and binary image from the extracted watermark. Binary image was scrambled k-times so that it should be descrambled using Arnold transform used in the watermark generating process.

[Definition #8] *Watermark regenerating function.* A watermark regenerating function  $\Theta$  extracts the binary image and the secret text message  $\chi^e$  from the extracted watermark  $\omega^e$ .

$$\Theta: \omega^e \rightarrow (SI, \chi^e) \quad (10)$$

However, the steps of the watermark verification include:

1. Extract secret text message from  $\omega^e_L$ : in the watermark generation process, the secret message  $\chi_t$  is decomposed into two equal size strings and padded in the start and end of the  $\omega$  so that the extracted message  $\chi^e$  is formulated as:

$$\{\omega^e_0 \ \omega^e_1 \dots \omega^e_{(t/2-1)}\} \parallel \{\omega^e_{(L-t/2)} \ \omega^e_{L-t/2+1} \dots \omega^e_{(L-1)}\} \rightarrow \{\chi^e_0 \ \chi^e_1 \dots \chi^e_{t-1}\}$$

2. The remaining of  $\omega^e$  is the scrambled image bits that formulated in the below expression.

$$\{\omega^e_{t/2} \ \omega^e_{t/2+1} \dots \omega^e_{(L-t/2-1)}\} \rightarrow \{SI_{(0,0)} \ SI_{(0,1)} \dots \ SI_{(m-1, n-1)}\}$$

3. Descrambling the extracted image: the SI should be descrambled k-times using Arnold transform. The result is the embedded image.



**ALGORITHM\_4: WATERMARK\_EXTRACTING**

Inputs: (Watermarked Database  $D_w$ , watermark length  $L$ , secret message length  $t$ , Secret key  $K_s$ , Attribute used in the watermark hiding  $A_j^m$ )

Output: Extracted Watermark bits  $\{\omega_0^e \omega_1^e \dots \omega_{L-1}^e\}$

```

1:  $\{\omega_0^e \omega_1^e \dots \omega_{L-1}^e\} \rightarrow \{ \}$ ;
2:  $\{\rho_0 \rho_1 \dots \rho_{L-1}\} \rightarrow \{ \}$ ;
3:  $n = 0; \mu = 0; \beta = 0;$ 
4: zeros  $[0 \dots L-1] = 0;$ 
5: ones  $[0 \dots L-1] = 0;$ 
6: DATABASE_PARTITIONING ( $D_w, K_s, t, L$ )
   i. For each  $\Omega \in D_w$  do
   ii. Part( $\Omega$ ) =  $H((K_s || t) || H(\Omega.PK || K_s)) \bmod L$ 
   iii. Insert  $\Omega$  into  $\rho_{part(\Omega)}$ 
   iv. End For
   v. Return:  $\{\rho_0, \rho_1, \dots, \rho_{L-1}\}$ 
7: For each  $\rho_i \in \rho_L$  do
8: For each  $\Omega \in \rho_i$  do
   9:  $n = n + 1;$  //return the number of tuples in the  $\rho_i$ 
   10: End For
11: For each  $\Omega_j \in \rho_i$  do
12:  $\mu_j = \text{count}(A_j^m)$  // calculate the total number of whitespaces within  $A_j^m$ 
13:  $\beta_j = ((PK_j \bmod n) \text{ XOR } (n \bmod (i+1))) \bmod \mu_j$  // return the location of watermark substitution
14: If  $A_j^m[\beta_j] == /u + 0020$  then:
15: zeros  $[i] = \text{zeros}[i] + 1;$ 
   16: End If
17: Else if  $A_j^m[\beta_j] == /u + 2009$  then:
18: ones  $[i] = \text{ones}[i] + 1;$ 
   19: End If
   20: End For
   21: End For
22: Majority Voting Mechanism:
   i. For  $i=0$  to  $L-1$  do
   ii. If zeros  $[i] > \text{ones}[i]$ 
   iii. Bit 0  $\rightarrow \omega_i^e$  :
   iv. End If
   v. Else if zeros  $[i] < \text{ones}[i]$ 
   vi. Bit 1  $\rightarrow \omega_i^e$  :
   vii. End If
   viii. End For
23: Return: Extracted watermark  $\omega^e \{\omega_0^e \omega_1^e \dots \omega_{L-1}^e\}$ 

```

**Fig 6 Watermark extraction algorithm.**

The degree of error correction can be predetermined in order to decide the acceptable database attacking level. The algorithm of watermark verification is shown in figure 7.

**ALGORITHM\_5: WATERMARK\_VERIFYING**

Inputs: *Extracted watermark  $\omega^e_L$ , and Arnold scrambling  $k$ -times*

Output: *Binary Image, secret text message  $\chi^e$ .*

- 1:  $\{\chi^e_0 \dots \chi^e_{t-1}\} \rightarrow \{\}$
- 2:  $\{I_0 \dots I_{MXN}\} \rightarrow \{\}$
- 3: *Extract text message  $\chi^e$  from extracted watermark  $\omega^e$* 
  - i. For  $i=0$  to  $t/2-1$  do {
  - ii.  $\{\chi^e_i\} = \{\omega^e_i\}$
  - iii.  $i = i + 1;$  }
  - iv. For  $j = m-t/2$  to  $m-1$  do {
  - v.  $\{\chi^e_j\} = \{\omega^e_j\}$
  - vi.  $j = j + 1;$  }
  - vii. *Return:*  $\{\chi^e_0 \dots \chi^e_{t-1}\}$
- 4: *Extract scrambled image from the remaining bits of  $\omega^e$* 
  - i. For  $i=t/2$  to  $m-t/2-1$  do {
  - ii.  $\{\lambda^e_i\} = \{\omega^e_i\}$
  - iii.  $i = i + 1;$  }
  - iv. *Return:*  $\{SI_0 SI_1 \dots SI_{MXN}\}$
- 5: *Descrambling the image using Arnold Transform for  $k$ -times*
- 6: *Return:*  $I_{MXN}, \chi^e$

**Fig 7 Watermark verification algorithm.**

**IV. EXPERIMENT RESULTS AND PERFORMANCE ANALYSIS**

**A. Experiment Results:**

The experiments were performed on a computer running Microsoft Windows 8.1 Pro, with Intel Core i5 2.50 GHz Processor and 4 GB memory. In experiment, the algorithm was implemented on Microsoft SQL Server 2008 and Matlab R2011. We applied our algorithm on the database of the students' information with 20000 tuples, with 10 attributes, Student ID is primary key and there are three attributes are multi-words nonnumeric attribute. Table 2 shows the values of the parameters used for algorithm testing.


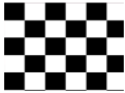

**TABLE 2: PARAMETERS VALUES USED IN ALGORITHM TESTING.**

Parameter	Value
secret text message bits $\chi$	/This is watermarked database/
Secret message length $t$	(30 char * 8 bit= 240 bit)
secret key $K_s$	034\$7*8nhmHx
Arnold Transform $k$ -times	16
Percentage of marked tuples	100% (constant)

### B. The Size of the Image Watermark

In this experiment the parameters appeared in Table 2 remain constant in the experience testing. We ran experiments with the following binary images sizes (in bits) 16 X 16, 32 X 32, and 64 X 64. In addition to it's usefully in the testing the algorithm with different watermark images, the selection of binary images with different sizes will benefit in testing the algorithm with different number of partitions where the number of partitions always depends on the length of the watermark. The secret message length (30 characters) was supposed constant with all images watermarks. Table 3 shows the selected test binary images with the final watermark length after adding the secret message to the binary images.

**TABLE 3: TESTING BINARY IMAGES WITH FINAL WATERMARK LENGTH.**

Testing Binary Image Watermark	Binary Image Size (bits)	Watermark Size (binary image + secret message)
 Image #1	16 X 16 = 256 bits	256 + 240 = 296 bits
 Image #2	32 X 32 = 1024 bits	1024 + 240 = 1264 bits
 Image #3	64 X 64 = 4096 bits	4096 + 240 = 4336 bits

### C. Security Analysis

In this subsection, the security of the algorithm is analysed. The security of watermarking means that watermarking information could not been found easily by the attackers. Only the person who holds the keys can find and extract the locations of watermarking information when the watermark algorithm is known. In the proposed algorithm, besides the application of secrete key  $K_s$ , the Arnold scrambling number  $K$ , the length of secret message  $t$  and the watermark length  $L$  are also secret and used in the watermark partitioning and locating procedure of watermark embedding.

In the watermark generation, the Arnold scrambling number  $K$  is used to scrambling the image to  $k$ -times to eliminate the correlation of each pixel and make the distribution of pixels disorderly and unsystematic, which will improve the robustness and security. Even if the database is damaged in the course of regular use, the extracted damaged watermark bits are distributed the whole image after performing the inverse Arnold transformation, which is not obvious to HVS. At the same time, the extracted watermark is a scrambled image, attackers do not know how to recover the original image at all unless known the scrambling number.

In addition to the using of binary image as watermark the using of secret text message will provide additional security through distributing its bits within the binary image will cause to increase image pixels distribution more disorderly and allowing to verifying two watermarks instead of one.

The secret key, primary key and watermark length  $L$  are concatenated and hashed to partition the tuples to logical partitions. As a result, the attacker should guess the watermark length  $L$  and secrete key  $K_s$  to know the tuples belonged to each partition. In this stage, the security is very important so that we focused on it by using more than one secret parameter to partition the tuples.

Moreover, the attacker could not exactly locate the positions of the whitespace used for watermark bits embedding unless know the number of tuples  $n$  in each partition that is used to compute the location of whitespace that will be used for watermark bit embedding. The number of tuples in each partition is variable so that knowing the number of tuples in specific partition cannot lead to knowing the number of tuples in other partition. This makes any attack that affect the number of tuples in the relation is very ineffective like insert and delete attack.

#### ***D. Attack Analysis***

In this section, we discuss the attacker model and the possible malicious attacks that can be performed. Assume that Alice is the owner of the data set  $D$  and has marked  $D$  by using a watermark  $\omega_L$  to generate a watermarked data set  $D_w$ . The attacker Mallory can perform several types of attacks in the hope of corrupting or even deleting the embedded watermark.

We assume that Mallory has no access to the original data set  $D$  and does not know any of the secret information used in the embedding of the watermark, including the secret key  $K_s$ , the watermark length, the secret text message length  $t$ , and the  $K$  number of Arnold scrambling. Given these assumptions, Mallory cannot generate the data partitions  $\rho_0, \rho_1, \rho_L$  because this requires the knowledge of the secret key  $K_s$ , the secret text message length  $t$ , and the watermark length  $L$ , thus Mallory cannot intentionally attack certain watermark bits. Moreover, any data manipulations executed by Mallory cannot be checked against the usability constraints because the original data set  $D$  is unknown. Under these assumptions, Mallory is faced with the problem of trying to destroy the watermark and at the same time of not destroying the data. However we have tested the robustness of the proposed algorithm by simulating several types of attacks on the relation as listed in the following subsections.

##### ***1. Subset Selection attack:***

In this type of attack, Mallory randomly selects and uses a subset of the original database. He may use the selected tuples for creating new database or add them to his existed database. Mallory hopes by doing so that the selected subset will not contain the watermark [23].

However, since the proposed algorithm embeds the watermark in the whole database by embedding the watermark bits in all tuples, this attack is of little or no threat. We select different ratio of the original data to simulate such attacks. We can see it has no effect at all to our watermark by selecting 50% of the watermarked relation. Even when 20% of the data is selected, outline drawing of the watermark image can be successfully recovered. A small part of the marked relation is enough for a successful detection since the watermark bits are distributed in all tuples.

The graph shown in Fig. 8-a indicates that the watermark will remain even if the attacker selects a subset as small as 5% of the original database. With testing different watermark length 496 bits (496 partition), watermark length with 1264 bits, and watermark length with 4336 bits, the detection rate remains very high even about 30 - 40% tuples selected, while the curve is rapidly increasing as long as the number of selected tuples was being increased. That's no matter how the small subset he selects, the watermark will remain in the selected subset providing the required copyright protection.

## **2. Subset Alteration Attack**

In this attack, Mallory alters the data value of  $x$  tuples. Here, Mallory is faced with the challenge that altering the data may disturb the watermark; however, Mallory does not have access to the original data  $D$  and thus may easily violate the usability constraints and render the data useless. The alteration attack perturbs the data in hope of introducing errors in the embedded watermark bits [23].

Depending on the watermark used in the testing, the detection rate varies greatly. With watermark length 496 bits (496 partition), the detection rate is as high as 92% when 90% of the tuples have been altered. While the detection rate of watermark length with 4336 bits starts to drop between 90% and 100% in an early stage (at about 25% altered tuples), its curve is rather flat and the detection rate is still at approximately 80% with 95% tuples altered. In contrast, using 1264 watermark length, the detection rate remains very high until about 60 - 70% tuples altered, while the curve is rapidly decreasing to 70 to 75% with 95% tuples altered. As shown in Fig. 8-b, as general the watermark will remain in the ratio 95% even if 70% of the tuples of the database are modified. This is because that the watermarks are repeatedly embedded for many times in all tuples of the database, making this type of attack ineffective.

## **3. Subset Deletion Attack:**

In this type of attack, Mallory may delete a subset of the tuples of the watermarked database and hope that the watermark will be removed. Mallory deletes  $x$  tuples from the marked data set [23]. If the tuples are randomly deleted, then, on average, each partition loses  $x/m$  tuples.

The suggested watermarking technique relies on all tuples to embedding the watermark. This indicates that the watermark will be removed only and only if, all the database tuples were deleted. That is, removing more than 95% of the database will not result in removing the watermark where the remaining tuples 5% will certainly contain some watermark bits. So, due to this fact, this type of attack also ineffective.

The results for the subset deletion attack are shown in Fig. 8-c. Deleting 90% of the initial database results in a watermark detection rate of approximately 99.2% with 496 watermark used, and approximately 91.2% with 1264 watermark used. With a growing number of partitions, the detection rate continuously decreases to as low as 72% with 4336 watermark. This leads to the fact that is: the watermark detection rate will decrease as long as the length of watermark is increased.

#### **4. Subset Insertion Attack**

In this type of attack, Mallory may insert some similar tuples without watermarks to the watermarked relation including normal update and hope that the original watermark will be removed. This is one of the most difficult attacks to defeat. Mallory decides to insert  $n$  tuples to the data set  $DW$  hoping to perturb the embedded watermark [23]. However, the watermark embedding is not based on a single tuple and is based on a cumulative hiding function that operates on all the tuples in the partition. Thus, the effect of adding tuples is a minor perturbation to the value of the hiding function and thus to the embedded watermark bit.

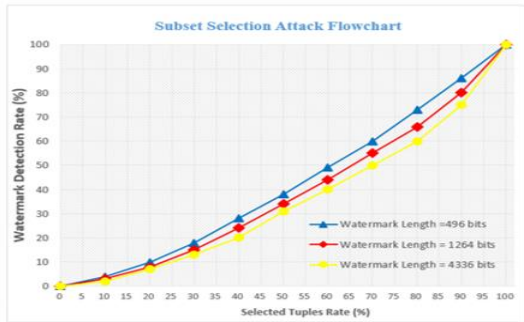
In this experiment, we added new  $n$  tuples to the original relation without passing through a rewatermarking process. We inserted new tuples until nearly doubles the size of the original database. When the relation increases to about 150% of its original size, still resulting in a watermark detection rate of 99% with 496 partitions. The detection rate at a relation size about 190% of the original tuples, decreases to 85.5% with 496 partitions, 79.8% with 1264 partitions and 76.3% with 4336 partitions as shown in figure 8-d.

However, Inserting new tuples to destroy watermark will not succeed as database partitioning and whitespace position equations depend on the all tuples in the database and the marked tuples are distributed according to the hash function based on the tuples primary keys and secret parameters, thus two tuples cannot have the same primary key or MAC. This form of attack has little impact on the watermark embedded through our algorithm.

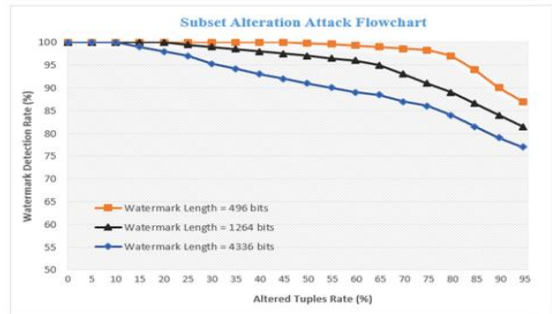
#### **5. Subset Reverse Order Attack**

In the subset reverse order attack Mallory attempts to removing watermark by exchanging or reordering the positions of tuples or attributes. It tries to destroy watermark information through changing the sequence of tuples, and aims at watermark information with semantic meaning [23].

The algorithm proposed here partitions tuples by the modulo on the hash value of the primary key value and other secret parameters. As a result, no matter how the sequence among tuples is changed, a given tuple will be assigned into a same partition, if the primary key value and secret parameters are unchanged. Also, the changes in the attributes order are not affected the watermark embedding and extracting because the algorithm is not depended on the order of attributes. Hence, this type of attack has no influence on the process of extracting watermark. Figure 8-e shows the watermark detection rate always will be 100% with any number of changing in the tuples or attributes order.

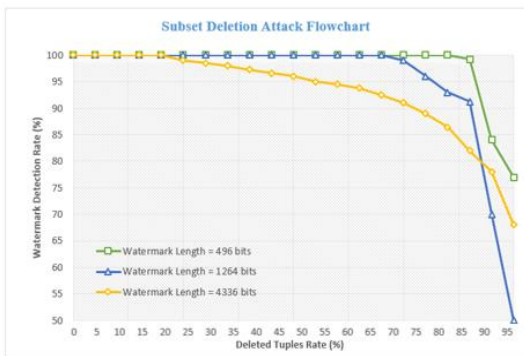


8-A: Subset selection attack.

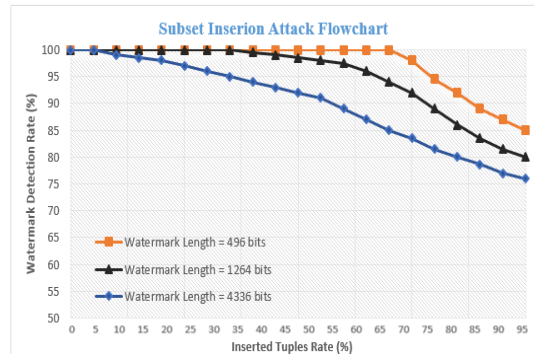


8-B: Subset modification

attack.

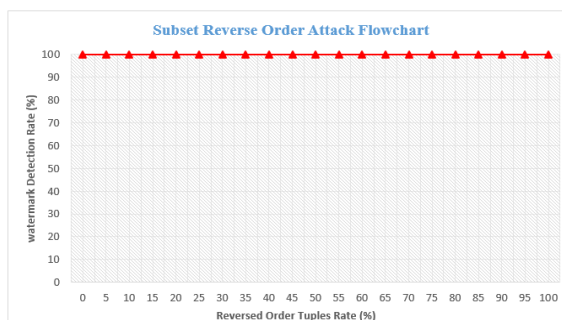


8-C: Subset deletion attack.



8-D: Subset Insertion

attack.



8-E: Subset reverse order attack.

Fig 8: Many Attack Types Simulation

## **6. Additive attack:**

In additive attack, Mallory may attempt to establish a plausible but spurious claim of ownership by trying to insert his watermark with that of Alice [23]. The conflict in ownership can be resolved by using two methods:

Integrating a trusted third party which facilitates distribution of key among the involved parties. When Alice shares her data, she attaches the key issued by the trusted third party to the dataset. Using this secure append-only key, governed by the trusted third party, can resolve the data ownership dispute by verifying that Alice's watermark is present in the dataset and also Alice's key is appended before Mallory key.

The other option might be that Alice can request a secret key from the trusted party. The key is obviously delivered on a date-time. Such time constraints can also help in agreeing ownership encounters: Alice can claim the insertion of watermark before Mallory did so by taking date-time, issued by the trusted party, as a reference. This time constraints can be added within secret message embedded in the relation allowing the time of embedding to be used as indication of the Alice ownership of the relation.

One of the statement for thwarting ownership argument is in which both the parties are able to successfully extract their watermarks from each other's original datasets. But this is not possible because Alice can demonstrate the presence of watermarks in Mallory's dataset since it belongs to her; whereas Mallory cannot illustrate the existence of his marks in Alice's original dataset.

## **7. Bit-Flipping Attack**

In the bit-flipping attack where Mallory randomly selects some bits and changes their values. Assume that Mallory changes each least significant bit. Changing MSB is not preferred because it will lead to lose the database availability. Mallory can change more than one bit from the LSB of the specific attribute value but he also wants to keep the availability of the database [23]. The suggested algorithm is resisted to such type of attack and such attacks have zero effect on the watermarked database because of the watermark bits always are embedded by substituting the whitespaces and don't embedded using LSB substituting methods.

There are many other types of attacks that operate in the LSB bit level like bit-setting, bit-clipping, and bit-complementing attacks. All these types of attacks cannot remove the watermark and only causing to affect the database.

## **E. Performance and efficiency analysis**

The algorithm is analysed and tested in the terms of the performance and efficiency through applying the following tests that prove the algorithm as applicable and efficient watermarking algorithm. These tests include:



### 1. *Blindness test:*

The suggested watermarking algorithm is blind, meaning that the original relation database is not required for watermark verification. Only secret parameters are required for watermark extraction and verification through the watermark image and secret message display.

### 2. *Reversibility test:*

The watermark embedding process is not affecting the bits levels of attributes values thus the watermark can be extracted through whitespaces substitution only without corrupting or losing original information.

### 3. *Effect to the data set:*

The algorithm effect on the data is tested using three methods: by the watermark invisibility, by the database size after watermarking it, and finally by the mean and variance statistics computations.

### 4. *Watermark invisibility:*

The suggested algorithm produces invisible watermark that means the watermark bits are hidden within the relation and cannot be seen by HVS. This was achieved by embedding the watermark bits through the whitespaces substitution only in the multi-words non-numeric attributes using Unicode whitespaces.

### 5. *Size/space consumption:*

The watermarked relation consumes the same amount of disk-space as the original relation on byte-level. This because of, the suggested embedding process neither add fake tuples and/or attributes, nor add (or remove) bytes from the attribute values.

### 6. *Data usability:*

The usability of the watermarked database can be identified statistically by checking the difference in mean and variance of the database before and after being watermarked. Ten watermarked attributes are selected to calculate the mean changes and variance changes as shown in table 4. As shown in the table 4 the mean value and variance change are very small and notice the change of mean and variance is imperceptible.

**TABLE 4: USABILITY DEMONSTRATION USING MEAN AND VARIANCE DIFFERENCES.**

Attribute Name	Mean	Variance	Mean Changes	Variance Changes
A1	115.30676210	67834.327	0.02357190	0.19321804
A2	126.66720931	99043.024	0.00789442	0.00036631
A3	95.122780542	9116.882	0.10044682	0.09424770
A4	122.78921093	769001.225	0.23400533	0.00721441
A5	125.67290032	8631.443	0.02022056	0.00398234
A6	113.04321189	55621.320	0.00983211	0.10262100
A7	110.88945532	89120.627	0.00849202	0.00079220
A8	104.11904629	12893.055	0.00729390	0.10322707
A9	123.18730321	67256.298	0.10365913	0.10044258
A10	120.76239109	12734.188	0.03682901	0.00216807

### *F. Regarding databases without a primary key*

The proposed algorithm assumed that the database to be watermarked has a primary key. The primary key is used for database partitioning and determining the location of whitespace to be substituted. For databases without a primary key, we can easily add in a virtual primary key that can be constructed from some MSB of each tuple's attributes or create fake attribute to be used as primary key.

### G. Incremental watermark update

If the need to update the watermarked database arises, it will definitely affect the encoded watermark. To ensure the robustness of the watermark, only the modified tuples need to be watermarked in order to keep the distribution of tuples per watermark bit as uniform as possible. This operation will not affect the rest of tuples in the database. Given a number of newly added tuples  $\Omega^+$  to the tuples of the original watermarked relation that consists of  $n$  tuples, we now have a larger database with tuples  $\Omega^* = \Omega_n \cup \Omega^+$ . As  $|\Omega_n|$  increases due to the insertion of new tuples  $\Omega^+ = \{\Omega_{1+}, \Omega_{2+}, \dots, \Omega_{3+}\}$ .

When watermarking newly added tuples  $\Omega^+$ , the corresponding partitions for the new tuple  $\Omega_{i+}$  where  $\Omega_{i+} \in \Omega^+$  can easily be calculated, since partitioning does not depend on other elements in relation instead of the primary keys  $PK$ . Finally, the watermark embedding proceed can be followed, except when iterating the partitions, just the non-watermarked tuples are subject to embedding watermark bits. After watermarking the new added tuples, the number of collectively embedded watermarks does not change since the number of partitions remains constant, but single watermark bits are embedded more often because the number of tuples is increased.

### H. For error-intolerant databases

The algorithm assumed that the database relation to be watermarked has at least one multi-words nonnumeric attribute and can tolerate small changes introduced by watermark encoding algorithm. The proposed algorithm can easily be modified to be applicable if a relation does not such attribute or if the multi-words nonnumeric attributes cannot tolerate any modifications. For example, instead of hiding the watermark bits in the whitespaces of the multi-words nonnumeric attributes, watermark bits can be stored in the float part of the numeric attributes, in the seconds part of the time attribute, or even create fake extra attribute to store the watermark.

## V. CONCLUSION

In this paper, we proposed a scheme for embedding and verifying integrity information for relational databases by substituting the Unicode white spaces in the multi-words non-numeric attributes. In addition to the binary image we used secret message as watermark data. Multiple watermark can assure multiple levels of copy right protection and database integrity. The proposed scheme does not require any additional storage to store the watermark data, nor does it introduce any distortion to the original data.

Experimental results show that the scheme can facilitate checking and maintaining the integrity of published relational databases on the Internet in an automatic and efficient manner. Even though all experimentally simulated attacks lead to a notable decrease in the watermark detection rate, the perceptual verification level hardly drops below the 90%, even after attacks which heavily distort the original relation database.

---

---

**REFERENCES**

- [1] Yanmin W. and Yuxi G., *The Digital Watermarking Algorithm of the Relational Database Based on the Effective Bits of Numerical Field*, IEEE Computer Society, World Automation Congress (WAC), ISSN: 2154-4824, 2012.
- [2] Halder R., Pal S. and Cortesi A., *Watermarking Techniques for Relational Databases: Survey, Classification and Comparison*, Journal of Universal Computer Science, USA, vol. 16, no. 21, ISSN: 3164-3190, 2010
- [3] Zhang L., Gao W., Jiang N., and Zhang Y., *Relational Databases Watermarking for textual and numerical data*, IEEE Computer Society, International Conference on Mechatronic Science, Electric Engineering and Computer, Jilin, China, ISSN: 978-1-61284-722-1, 2011.
- [4] Pramod A., Vikramsinh M., Vijay D., and Vishal V., *Speech Based Watermarking for Non-numeric Relational Database*, International Journal of Innovative Research in Engineering & Science, ISSN 2319-5665, issue 2, volume 4, April 2013.
- [5] M. Dhande, A. Kotyankar, and N. Mannadiar, “*Watermarking for Security in Database*” International Journal on Recent and Innovation Trends in Computing and Communication. Vol: 4 Issue: 4, 2016.
- [6] S. Katzenbeisser and F. A. Petitcolas, *Information Hiding Techniques for Steganography and Digital Watermarking*, Artech House, 2000.
- [7] G. Langelaar and I. Setyawan, *Watermarking Digital Image and Video Data*, IEEE Signal Processing, vol. 17, pp. 20–43, September 2000.
- [8] R. Agrawal and J. Kiernan, *Watermarking relational databases*, Proceedings of the 28th Very Large Data Bases VLDB Conference, Hong Kong, China, vol. 28, pp. 155–166, 2002
- [9] Al-Haj, A. and Odeh, A., *Robust and blind watermarking of relational database systems*, Journal of Computer Science, 4:1024–1029. 2008
- [10] Wang Y., Zhu G., and Zhang S., *Research On The Watermarking Algorithm Based On Numerical Attribute In the Relational Database*, IEEE Computer Society, International Conference on Computer Science and Electronics Engineering, 978-0-7695-4647, 2012.
- [11] Zhang, Y., Niu, X., Zhao, D., Li, J., and Liu, S., *Relational databases watermark technique based on content characteristic*, IEEE Computer Society, In Proceedings of the 1st International Conference on Innovative Computing, Information and Control, Beijing, China. 2006.
- [12] Xiao, X., Sun, X., and Chen, M., *Second-LSB-dependent robust watermarking for relational database*, IEEE Computer Society, In Proceedings of the 3rd International Symposium on Information Assurance and Security (IAS '07), pages 292–300, Manchester, United Kingdom, 2007.
- [13] Wang, H., Cui, X., and Cao, Z., *A speech based algorithm for watermarking relational databases*, IEEE Computer Society, In Proceedings of the 2008 International Symposiums on Information Processing (ISIP '08), Moscow, Russia, 2008.
- [14] Pournaghshband, V., *A new watermarking approach for relational data*. ACM Press, In Proceedings of the 46th Annual Southeast Regional Conference (ACM-SE '08), Pages 127–131, Auburn, Alabama, 2008.

- 
- [15] Prasannakumari, V., *A robust tamper proof watermarking for data integrity in relational databases*, Research Journal of Information Technology, 1:115–121, 2009.
- [16] Hanyurwimfura D. Liu Y. and Liu Z., *Text format based relational database watermarking for non-numeric data*, IEEE Computer Society, International Conference on Computer Design and Applications, 978-1-4244-7164-5, 2010.
- [17] Li M., and Zhao W., *An Asymmetric Watermarking Scheme for Relational Database*, IEEE Computer Society, In Proceedings of 3rd International Conference on Communication Software and Networks (ICCSN), 978-1-61284-486-2, 2011.
- [18] El-Haggag N., El-khouly M., and Abu El Alla S., *Blind Watermarking Technique for Relational Database*, COMPUSOFT, An international journal of advanced computer technology, Volume-II, Issue-V, 2014.
- [19] R. Gadiya and Prashant A. Kale., *Watermarking Technology for Relational Database*, International Journal of Modern Trends in Engineering and Research, VOL. 20, NO. 1, 2016.
- [20] Saman Iftikhar, M. Kamran, and Zahid Anwar, “*RRW—A Robust and Reversible Watermarking Technique for Relational Data*” in IEEE Transon Knowledge and Data Engineering, VOL 27, NO. 4, April 2015.
- [21] M. Xie, C. Wu, J. Shen, and M. Hwang; *A Survey of Data Distortion Watermarking Relational Databases*, International Journal of Modern Trends in Engineering and Research, VOL.18, NO. 2, 2016.
- [22] S. Alfagi, A. Manaf, B. Hamida, S. Khan and Ali A. Elrowayati, *Survey on Relational Database Watermarking Techniques*, ARPN Journal of Engineering and Applied Sciences, VOL. 11, NO. 1, JANUARY 2016.
- [23] Shehab M., Bertino E., and Ghafoor A., *Watermarking Relational Databases Using Optimization-Based Techniques*, IEEE Transactions on Knowledge and Data Engineering, VOL. 20, NO. 1, 2008.