A Robust, Distortion Minimizing Technique for Watermarking Relational Databases Using Once-for-all Usability Constraints

M. Kamran, Sabah Suhail, and Muddassar Farooq

Abstract—Ownership protection on relational databases – shared with collaborators (or intended recipients) – demands developing a watermarking scheme that must be able to meet four challenges: (1) it should be robust against different types of attacks that an intruder could launch to corrupt the embedded watermark; (2) it should be able to preserve the knowledge in the databases to make them an effective component of knowledge-aware decision support systems; (3) it should try to strike a balance between the conflicting requirements of database owners, who require soft usability constraints, and database recipients who want tight usability constraints that ensure minimum distortions in the data; and (4) last but not least, it should not require that a database owner defines usability constraints for each type of application and every recipient separately. The major contribution of this paper is a robust and efficient watermarking scheme for relational databases that is able to meet all above-mentioned four challenges. The results of our experiments prove that the proposed scheme achieves 100% decoding accuracy even if only one watermarked row is left in the database.

Index Terms—Data usability constraints, distortion free, database watermarking, data quality, right protection, ownership protection, robust watermarking.

1 INTRODUCTION

WATERMARKING, without any exception, has been used for ownership protection of a number of data formats – images, video, audio, software, XML documents, geographic information system (GIS) related data, text documents, relational databases etc. – that are used in different application domains. Recently, intelligent mining techniques are being used on data, extracted from relational databases, to detect interesting patterns (generally hidden in the data) that provide significant support to decision makers in making effective, accurate and relevant decisions; as a result, sharing of data between its owners and data mining experts (or corporations) is significantly increasing. Consequently, it has become relevant (in this context) to explore suitable watermarking techniques for ownership rights protection of relational databases that should be imperceptible, robust with blind decoding. Moreover, once the owner of data embeds the watermark, the distortions in the original data are kept within certain limits, which are defined by the usability constraints, to preserve the knowledge contained in the data.

An intended recipient (Bob) wants the data owner (Alice) to define tight usability constraints so that he gets accurate data. For maximum robustness of watermark, Alice, on the other hand, wants to have larger bandwidth on manipulations performed during embedding of a watermark which is only possible if she puts soft usability constraints. To conclude, Bob and Alice have conflicting requirements: Bob wants “minimum distortions in the watermarked data” while Alice wants to produce “watermarked data having strong ownership”. Any watermark embedding technique that strives for a compromise bandwidth allows an attacker (Mallory) to corrupt or remove the watermark by slightly surpassing the available bandwidth. The compromise bandwidth is achieved once Alice defines the usability constraints (by studying the semantics of a given application) in such a way that the embedded watermark is not only robust but also causes minimum distortions to the underlying data. The job of analyzing the semantics of each application and use it to define usability constraints is not only cumbersome but also inefficient for a data owner. Remember, the robustness of a watermark is measured by the watermark decoding accuracy that in turn depends on the bandwidth available for manipulation. These observations provide us the motivation for undertaking research reported in this paper. The major contributions of the work presented in this paper are presented in the following:

• We propose a novel watermark decoding algorithm

1. In this paper – unless otherwise specified – the terms data, dataset, and database are used interchangeably.
which ensures that its decoding accuracy is independent of the usability constraints (or available bandwidth). As a result, our approach facilitates Alice to define usability constraints only once for a particular database for every possible type of intended application. Moreover, it also ensures that the watermark introduces the least possible distortions to the original data without compromising the robustness of the inserted watermark.

- The proposed algorithm embeds every bit of a multi-bit watermark (generated from date-time) in each selected row (in a numeric attribute) with the objective of having maximum robustness even if an attacker is somehow able to successfully corrupt the watermark in some selected part of the dataset.
- We prove the robustness of our watermarking scheme by analyzing its decoding accuracy under different types of malicious attacks using a real world dataset.
- We provide solutions to resolve conflicting ownership issues in case of the additive attack in which Mallory inserts his own watermark in Alice's watermarked database.

The rest of the paper is organized as follows. Section 2 discusses the existing watermarking techniques and their shortcomings. A brief overview of the different stages of the proposed watermarking technique is described in Section 3. Section 4 provides a detailed description of the proposed technique. The robustness study of the watermarking approach is presented in Section 5. Finally, we conclude the paper with an outlook to future work.

2 RELATED WORK

In the current section, we will provide a brief overview of recently proposed watermarking techniques for relational databases. The objective is to clearly understand the limitations of prior art. Agrawal et al. [3] proposed a bit-resetting algorithm that employs the principle of setting the Least Significant Bit (LSB) of the candidate attribute of the selected subset of tuples. The parameters selection for watermarking is based on computing Message Authenticated Code (MAC), where MAC is calculated using the secret key and the tuple’s primary key. This technique assumes unconstrained LSB manipulation during watermark embedding process. Such out-of-bound modification of data might also generate undesirable results. Although LSB-based data hiding techniques are efficient, but an attacker is able to easily remove watermark by simple manipulation of data: for example shifting LSB. Other bit-resetting techniques like [1], [4], [5], [6],[7],[8], [9] also have similar robustness related shortcomings as well.

Sion et al. [10] proposed a statistical-based algorithm in which a database is partitioned into a maximum number of unique, nonintersecting subsets of tuples. The data partitioning concept is based on the use of special marker tuples, making it vulnerable to watermark synchronization errors particularly in the case of tuple insertion and deletion attacks, as the position of marker tuples is disturbed by these attacks. Such errors may be reduced if marker tuples are stored during watermark embedding phase and the same may be used for constructing the data partitions again during watermark decoding phase. But using the stored marker tuples to reconstruct the partitions violates the requirement of “blind decoding” of watermark. Furthermore, the threshold technique for bit decoding involves arbitrarily chosen thresholds – without following any optimality criteria – that are responsible for the error in the decoding process. The concept of usability bounds on data is used in this technique to control distortions introduced in the data during watermark embedding. However, an attacker can corrupt the watermark by launching large scale attacks on large number of rows. Moreover, the decoding accuracy is dependent on the usability bounds set by the data owner; as a result, the decoding accuracy is deteriorated if an attacker violates these bounds. An important shortcoming of this approach is that the data owner needs to specify usability constraints separately for every type of application that will use data. Later improvements [2], [11],[12], [13] have tried to solve the problem of synchronization errors only.

Another class of watermarking techniques is distortion-free techniques. Using these techniques, data is delivered to the intended recipients without making any distortion in the data. The techniques reported in [14] [15] are vulnerable to even minor malicious attacks, and therefore, cannot be used for enforcing ownership protection.

The constrained data content modifying techniques add some new content in the database to embed the watermark. The content is added subject to given usability constraints. If an attacker successfully attacks the watermarked content, the watermark information is lost without compromising data quality. The techniques reported in [16], [17] face such problems.

A recent survey [18] presented a review of database watermarking techniques but, to the best of our knowledge, no watermarking technique for relational databases exists which ensures that the decoding accuracy is independent of the usability constraints; as a consequence, an owner does not need to define them for each type of intended application and use. Our proposed technique not only has this feature but it is also able to meet the conflicting “robustness requirement” of the data owner and “minimum distortions requirement” of the intended recipient.

3 APPROACH OVERVIEW

Figure 1 shows the block diagram summarizing the main components of our watermarking technique. The date-time is used to generate the watermark bits. Using date-time, as the foundation of a watermark, may also serve the purpose to counter additive attacks. We will shortly discuss this point in detail.
A robust watermark algorithm is used to embed watermark bits into the dataset of Alice. The watermark embedding algorithm takes a secret key ($K_s$) and the watermark bits ($W$) as input and converts a dataset $D$ into watermarked dataset $D_W$. For an easy reference, Table 1 lists the major symbols used in this paper.

The modifications (distortions) made by watermarking are bounded by the usability constraints matrix $G$. In our technique, it is defined only once for every possible type of application that will eventually use the dataset.

The watermark encoding process can be summarized in the following steps:

**Watermark Bits Generation.** Watermark bits string "$W" is generated from UTC (Coordinated Universal Time) date-time which is the primary time standard used to synchronize the time all over the world [19]. These bits are given as input to the watermark encoding function.

**Data Partitioning.** The dataset $D$ is partitioned into $m$ non-overlapping partitions by using the secret key $K_s$ in conjunction with a cryptographic secure hash function.

**Selection of dataset for watermarking.** To minimize distortions, only few tuples are selected for watermarking in this step.

**Watermark Embedding.** The watermark bits are embedded in the selected tuples using a robust watermarking function. The bit embedding statistics $\Delta$ are used to compute the correction factor $\tau$. Our technique embeds each bit of the watermark in every selected tuple of each partition; as a result, it is robust against malicious attacks even (after an attack) if only one watermarked row is left in the data. The watermarked dataset $D_W$ is delivered to Bob where an intruder – Mallory – aims at destroying the watermark by launching different types of attacks.

Watermark decoding is the process of extracting the embedded watermark from the watermarked dataset $D_W$, using the secret parameters: the secret key $K_s$, correction factor $\tau$, and decoding threshold $\gamma$. The decoding algorithm is blind as the original dataset $D$ is not needed for successfully decoding the embedded watermark.

The watermark decoding process can be summarized in the following steps:

**Data Partitioning.** The data partitions are generated by using the same data partition algorithm as in the watermark encoding phase.

**Identification of Watermarked Dataset.** The watermarked tuples are identified using the same procedure that has been used to select them for inserting watermark bits in the encoding phase.

**Watermark Decoding.** In this stage, the correction factor $\tau$ and the decoding threshold $\gamma$ are used to decode the watermark bits. The decoding algorithm is blind and its decoding accuracy does not depend on the usability constraints. As a result, 100% decoding accuracy is achieved irrespective of the amount of data alterations made by an attacker in the watermarked data.

**Majority Voting.** The majority voting is used to correctly decode an inserted watermark bit. This step is optional and is done to provide security against a sophisticated attacker who is able to flip the watermark bits in selected tuples only.

4 Proposed Methodology

4.1 Data Partitioning

The dataset $D$ is a database relation with scheme $D = (PK, A_0, ..., A_{n-1})$, where $PK$ is the primary key attribute and $A_0, ..., A_{n-1}$ are $n$ other attributes. The

---

TABLE 1 Notations

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D$</td>
<td>Original data</td>
</tr>
<tr>
<td>$D_W$</td>
<td>Watermarked data</td>
</tr>
<tr>
<td>$\rho$</td>
<td>Percentage change made on the data values</td>
</tr>
<tr>
<td>$G$</td>
<td>Usability constraints set</td>
</tr>
<tr>
<td>$\tau$</td>
<td>Correction factor</td>
</tr>
<tr>
<td>$W$</td>
<td>Encoded watermark</td>
</tr>
<tr>
<td>$W'$</td>
<td>Decoded watermark</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>Decoding threshold</td>
</tr>
<tr>
<td>$\beta$</td>
<td>Number of tuples in the dataset</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>Amount of alterations made by an attacker</td>
</tr>
<tr>
<td>$\Delta$</td>
<td>A vector containing data manipulations statistics</td>
</tr>
<tr>
<td>$\kappa$</td>
<td>Percentage of inserted tuples</td>
</tr>
<tr>
<td>$\omega$</td>
<td>Percentage of altered tuples</td>
</tr>
<tr>
<td>$DA$</td>
<td>Percentage of decoding accuracy</td>
</tr>
</tbody>
</table>

---

5. The decoding accuracy may decrease in case of combination of different attacks.
partition algorithm divides the dataset \( D \) into \( m \) non-overlapping partitions namely \( \{S_0, \ldots, S_{m-1}\} \) such that for any two partitions \( S_i \cap S_j = \emptyset \). Moreover, the partition sets must be non-empty and collectively exhaustive to \( D \) such that \( S_0 \cup S_1 \cup \ldots \cup S_{m-1} = D \). The data partitioning algorithm partitions the dataset into logical groups by using data partitioning algorithm proposed in [2]. Partitioning is based on a secret key \( K \) and a cryptographic hash function Message Digest (MD5) [20].

**Definition 1.** [Hash function.] A hash function \( H \) maps a variable-size input \( \Upsilon \) to a fixed-size string \( h \), called the hash value \( h \), as:

\[
H : \Upsilon \rightarrow h \tag{1}
\]

For each tuple \( r \in D \), the data partitioning algorithm computes Message Authentication Code (MAC) in order to assign tuples to the partitions using a hash function \( H \) as

\[
\text{par}(r) = H(K_s||H(r.PK||K_s)) \mod m \tag{2}
\]

where \( r.PK \) is the primary key of the tuple \( r \), \( H() \) is a secure hash function and \( | \) is the concatenation operator. Algorithm 1 lists the steps of data partitioning process.

**Algorithm 1 Get_Partitions**

- **Input:** Dataset \( D \), Secret Key \( K_s \), Number of partitions \( m \)
- **Output:** Data partitions \( S_0, \ldots, S_{m-1} \)

\[
\begin{align*}
1: & \quad \text{for each Tuple reD do} \\
2: & \quad \text{par}(r) = H(K_s||H(r.PK||K_s)) \mod m \\
3: & \quad \text{insert r into } S_{\text{par}(r)} \\
4: & \quad \text{end for} \\
5: & \quad \text{return } S_0, \ldots, S_{m-1}
\end{align*}
\]

4.2 Selection of dataset for watermarking

The following two steps are applied on the dataset to select tuples for watermarking.

4.2.1 Threshold Computation

In this step, a threshold is computed for each attribute. If the value of any attribute of a tuple is above its respective computed threshold, it is selected for watermarking.

**Definition 2.** [Data selection threshold.] Given a dataset \( D \), a function \( \Phi \) is used to calculate data selection threshold for constructing \( D'_T \) from \( D \).

\[
\Phi : D \rightarrow D'_T \tag{3}
\]

The data selection threshold for an attribute is calculated using equation (4).

\[
T = c \ast \mu + \sigma \tag{4}
\]

where \( \mu \) is the mean, \( \sigma \) is the standard deviation of the values of an attribute \( A \) in \( D \), and \( c \) is the confidence factor with a value between 0 and 1. The confidence factor \( c \) is kept secret to make it very difficult for an attacker to guess the selected tuples in which the watermark is inserted.

We select only those tuples, during the encoding process, whose values are above \( T \).

**Remark 1.** Note that the manner in which \( T \) is calculated here is different from the one used in [10], [2], and [11], where \( c \) is multiplied by \( \sigma \) instead of \( \mu \). The major shortcoming of the data selection threshold formula (or reference point formula) in [10], [2], and [11] is that an attacker may roughly guess the fact that the tuples—having value above \( \mu \)—were watermarked by simply observing that adding any positive number in \( \mu \) will result in a value higher than \( \mu \), if the value of confidence parameter \( c \) is between 0 and 1. On the other hand, in our approach an attacker can not guess the values of watermarked tuples because any tuple—having value below, equal to, or above \( \mu \)—can be selected for watermarking depending on the value of the secret parameter \( c \).

In this way, we reduce the number of tuples to be watermarked; as a result, data distortions during watermark embedding are minimized. Algorithm 2 depicts different steps of this phase.

In order to ensure that the tuples, for which if any of the attribute values is above \( T \), are included in the to-be-watermarked tuples set, a union of tuples in this phase is taken. Formally speaking, a table may be represented as:

\[
\begin{pmatrix}
A_0 & A_1 & \ldots & A_{n-1} \\
R_0 & v_{00} & v_{01} & \ldots & v_{0(n-1)} \\
R_1 & v_{11} & v_{11} & \ldots & v_{1(n-1)} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
R_{\beta-1} & v_{(\beta-1)0} & v_{(\beta-1)1} & \ldots & v_{(\beta-1)(n-1)}
\end{pmatrix}
\]

If we consider the tables’ rows and columns as a matrix, the union of attributes \( (A_0 \text{ to } A_{n-1}) \) of a tuple \( (R_0) \) based on the computed threshold value \( (T_0) \) is:

\[
\begin{pmatrix}
v_{00} & v_{01} & \ldots & v_{0(n-1)} \\
v_{11} & v_{11} & \ldots & v_{1(n-1)} \\
\vdots & \vdots & \ddots & \vdots \\
v_{(\beta-1)0} & v_{(\beta-1)1} & \ldots & v_{(\beta-1)(n-1)}
\end{pmatrix}
\]

Where, \( v \) represents the data values. Finally, the tuples that are above the threshold value of their respective attributes i.e \( \forall R_{T} \) will transform the dataset \( D \) into \( D'_T \). Hence,

\[
D'_T \leftarrow \forall R_{T} \tag{5}
\]

After this step, the dataset \( D'_T \) is given as an input to the next phase.

**Algorithm 2 Get_Data_Selection_Threshold**

- **Input:** Data partitions \( S_0, \ldots, S_{m-1}, c \)
- **Output:** Dataset \( D'_T \)

\[
\begin{align*}
1: & \quad \text{for } i = 0 \text{ to } m - 1 \text{ do} \\
2: & \quad \text{for each Attribute } A \in S_i \text{ do} \\
3: & \quad \text{Compute } \mu \text{ and } \sigma \text{ on } A \\
4: & \quad \text{Calculate } T \text{ using equation (4)} \\
5: & \quad \text{end for} \\
6: & \quad \text{end for} \\
7: & \quad \text{return } D'_T \leftarrow \forall R_{T}
\end{align*}
\]

4.2.2 Hash Value Computation

In this step, a cryptographic hash function MD5 is applied on the selected dataset to select only those tuples which have an even hash value. This step achieves two
objectives: (i) it further enhances the watermark security by hiding the identity of the watermarked tuples from an intruder; and (2) it further reduces the number of to-be-watermarked tuples to limit distortions in the dataset.

The dataset $D_T$ is used to select tuples with even hash values and put them in the dataset dataset $D_T''$. The steps involved in this phase are illustrated in Algorithm 3.

**Algorithm 3 Get_Even_Hash_Value_Dataset**

<table>
<thead>
<tr>
<th>Input: Dataset $D_T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output: $D_T''$</td>
</tr>
<tr>
<td>1: for each $r \in D_T$ do</td>
</tr>
<tr>
<td>2: $\text{Even}_\text{Value}(r) = H(K_r</td>
</tr>
<tr>
<td>3: if $\text{Even}_\text{Value}(r) = 0$ then</td>
</tr>
<tr>
<td>4: insert $r$ into $D_T''$</td>
</tr>
<tr>
<td>5: else</td>
</tr>
<tr>
<td>6: don’t consider this tuple for watermarking</td>
</tr>
<tr>
<td>7: end if</td>
</tr>
<tr>
<td>8: end for</td>
</tr>
<tr>
<td>9: return $D_T''$</td>
</tr>
</tbody>
</table>

The dataset $D_T''$, consisting of $\zeta$ tuples, is the subpart of the dataset $D$ and is not physically separated from the rest of the parts of $D$.

Note: As selection of $\zeta$ tuples is also based on the value of data selection threshold, Mallory may try to corrupt the embedded watermark by changing the data values such that the data selection threshold value is disturbed and hence Alice is unable to detect the watermarked tuples in the watermark detection phase. But, since Mallory has no knowledge of the confidence factor $c$; therefore, he may be able to only arbitrarily attack some selected part of the watermarked data to corrupt the watermark with some probability $P$. This probability is made smaller by using data selection threshold $T$ and even hash values as proven in Proposition 1.

**Definition 3.** $P_{\text{Success}}(\omega)$ is the probability that an attacker is successful in changing the values of $\omega\%$ watermarked tuples such that data selection threshold is modified.

**Proposition 1.** $P_{\text{Success}}(\omega)$ approximately approaches to zero, in case of large databases.

**Proof:** If $P_{\text{Success}}(\omega)$ denotes the probability that Mallory is successful in changing the values of $\omega\%$ watermarked tuples such that data selection threshold is modified. Now, for a watermarked dataset with a total of $\beta$ tuples and $n$ numeric attributes (other than the primary key), Mallory has to target at least $\left(\frac{\omega}{2}+1\right)$ tuples to change the value of more than $50\%$ watermarked tuples to undo the effect of the majority voting, so:

$$\omega = \frac{\beta}{2}$$

(6)

If $\zeta$ represents the number of watermarked tuples, the probability for any particular tuple being watermarked is $\frac{\zeta}{n}$. So, for Mallory, the probability of successfully changing the value of $\frac{\beta}{2} + 1$ watermarked tuples such that data selection threshold is modified is:

$$P_{\text{Success}}(\omega) = \left(\frac{\zeta}{n}\right)^{\frac{\beta}{2} + 1}$$

(7)

Substituting equation (6) in equation (7), we get:

$$P_{\text{Success}}(\omega) = \left(\frac{\zeta}{\beta}\right)^{(\omega+1)}$$

(8)

Recall that all the numeric attributes play their role in data selection threshold; therefore, the above equation becomes:

$$P_{\text{Success}}(\omega) = \left(\frac{\zeta}{\beta}\right)^{(\omega+1)^n}$$

(9)

It is clear from equation (9) that smaller values of the fraction $\frac{\zeta}{\beta}$ means that the probability of successfully attacking watermarked tuples is also smaller. This value will become even smaller for larger databases; hence we can write:

$$\lim_{(\frac{\zeta}{\beta}) \to 0} \Rightarrow P_{\text{Success}}(\omega) \to 0$$

(10)

Let us take an example of a very small dataset that has 100 tuples and each tuple has 3 attributes. Suppose in this dataset, only 10 tuples are selected for watermarking. The probability of successfully changing the value of data selection threshold by attacking (50%+1) of watermarked tuples (6 tuples to be more precise) is:

$$P_{\text{Success}}(\omega) = \left(\frac{10}{100}\right)^{(\frac{10}{2}+1)^3} = \left(\frac{10}{100}\right)^{(51)^3}$$

$$= 1 \times 10^{-153}$$

This probability is very small, and will become smaller for larger datasets. Moreover, ideally speaking an attacker would not like to change the values of tuples too much because this will make the data absolutely useless for any recipient.

### 4.3 Generation of Watermark bits

In this step, date-time stamp is used to generate watermark bits.

**Definition 4.** [Watermark generating function.] A watermark generation function $\Psi$ transforms an alpha-numeric string $\lambda$ to an $l$-bits long binary bit string $\{b_0b_1b_2...b_{l-1}\}$.

$$\Psi : \lambda \rightarrow \{b_0b_1b_2...b_{l-1}\}$$

(11)

The watermark generating function $\Psi$ takes date-time stamp as an input and then generates watermark bits $\{b_0b_1b_2...b_{l-1}\}$ from this date-time stamp. The date-time stamp “might" also help to identify additive attacks in which an attacker wants to re-watermark the dataset. To construct a watermarked dataset, these watermark bits are embedded in the original dataset by using the following watermark embedding algorithm.
4.4 Watermark Embedding Algorithm

The watermarking algorithm uses multi-bit watermarking property and is scalable to any number of attributes. For a better understanding, we assume that the partition set $S_i$ in dataset $D_T$ contains a single attribute $v_i \in S_i$. The encoding function generates bits $b_0, b_1, b_2, \ldots, b_{l-1}$, where $l$ is the length of the watermark. Since our technique embeds watermark bits $-b_0, b_1, b_2, \ldots, b_{l-1}$ in each partition of $S_i$; therefore, the watermark bits can be recovered from the remaining partitions if the watermark is removed from a particular partition $S_i$.

Definition 5. [Watermark embedding function.] Watermark embedding function $\phi$ transforms the input data set $D$ to a watermarked data $D_W$ after performing some data manipulations. Formally,

$$\phi : (D, W) \rightarrow D_W$$

Definition 6. [Data manipulations vector.] Data manipulations vector simply keeps a record of transformations of $D$ into $D_W$ as:

$$\Delta \leftarrow (D_W - D)$$

A parameter $\delta_{ij}$ denotes the one instance of data modification performed on a tuple $i$ in $j^{th}$ column.

The watermark embedding function uses the parameter $\delta_{ij}$ to modify the attribute data value $v_{ij}$ of a tuple $i$ in $j^{th}$ column. The watermark embedding function for a tuple $i$ in $j^{th}$ column is:

$$\phi = \delta_{ij} + v_{ij}$$

If the bit $b$ is equal to 1, the bit encoding algorithm computes the $\delta_{ij}$, subject to the constraint set $G$, on the data value $(v_{ij})$ of an attribute as:

$$\delta_{ij} = \rho \% \text{ of } v_{ij} \text{ with } \rho > 0$$

Similarly, if the bit $b$ is equal to 0, then $\delta_{ij}$ is computed as:

$$\delta_{ij} = \rho \% \text{ of } v_{ij} \text{ with } \rho < 0$$

The value of $\rho$ remains fixed for every watermarked tuple. It is same in magnitude for watermark bit 0 and 1 but has an opposite sign. This value of $\rho$ (along with the associated sign) is used during watermark decoding phase. It is important to emphasize that $\rho$ is not contained in the original data; therefore, its use in the watermark decoding phase does not violate the property of blind watermark decoding. Its value is defined by the data owner and is also kept secret. Our technique brings an overall change in the data values of attributes using equations (15) and (16) instead of LSB only to overcome shortcomings of LSB based techniques.

To ensure that $\delta_{ij}$ changes the value of $v_{ij}$ within allowed limits, a feasible range for manipulating the dataset is defined by the constraint set $G$. This set ensures the usability of data by enforcing a tolerance level on the value of each attribute $v_{ij}$. As a rule of thumb, the usability constraints are dependent on an application: for example, a player could only be placed in the dataset of a Junior team if his age is in between a maximum and minimum value. In this application, the minimum and maximum value could be used to define the degree by which the age of a player can be modified ($\delta_{ij}$) during the watermark encoding process.

$$\delta_{min} \leq \delta_{ij} \leq \delta_{max}$$

Sometimes, the constraints are imposed on data statistics. For example, a dataset may require that the mean or the standard deviation of the watermarked data must be equal to the mean or the standard deviation of the original data:

$$\mu_D = \mu_{D_W}; \text{ and } \sigma_D = \sigma_{D_W}$$

In our technique, we define the usability constraints in terms of the mean and the standard deviation of the watermarked attribute. For this, it is ensured that the two measures remain approximately the same before and after watermarking of data. This objective is achieved by using very tight usability constraints; as a consequence, the requirement for minimum data distortions is implicitly met. The data manipulation statistics in $\delta$ are recorded for each encoding step in $\Delta$ and are used to compute the secret correction factor $\tau$.

Since the dataset $D_T$ is the part of a particular partition $S_k$; therefore, the partitions $S_k^W$ contains these watermarked records. All such $S_k^W$ are inserted into $D_W$ to get the watermarked dataset. The dataset $D_W$ is then made publicly accessible for the intended recipients. The steps of embedding phase are depicted in Algorithm 4.

4.4.1 Computing correction factor value for making decoding accuracy independent of usability constraints

Definition 7. [Data usability constraints.] Given a dataset $D$, the data usability constraints $G$ is a set with elements $\mu$ and $\sigma$, for bounding data manipulations vector $\Delta$, to transform $D$ into $D_W$.

These data usability constraints usually affect the watermark decoding accuracy. But we want to make
the decoding accuracy independent of these usability constraints.

**Definition 8.** [Usability independent watermark decoding.] Given a set of usability constraints \( G \), usability independent watermark decoding implies that the decoding accuracy is not a function of any element of \( G \).

The set of attacks may include different combinations of tuple deletion, insertion, and alteration attacks and might well surpass the available bandwidth identified by using the usability constraints.

**Lemma 1.** The watermark decoding accuracy can be independent of usability constraints \( G \), if and only if the watermark decoding accuracy remains unchanged even if an attacker is able to surpass the bounds on usability constraints.

**Proof:** (Proof by contradiction.) Assume to the contrary that the decoding accuracy is changed, if an attacker is able to surpass the usability constraints and still the decoding accuracy is said to be independent of the usability constraints. Let \( D A_1 \) be the decoding accuracy of watermarking decoding algorithm for decoding the embedded watermark from an un-attacked watermarked dataset \( D W_1 \). For testing the resilience of embedded watermark, the data owner Alice launches an attack \( A t t a c k_1 \), which surpasses some of the usability constraints set \( G \) with elements \( \mu \) and \( \sigma \), and produces an attacked dataset \( D W_2 \). Now, Alice decodes the embedded watermark from \( D W_2 \) with \( D A_2 \) accuracy.

Now there are two cases: (1) \( D A_2 = D A_1 \) and (2) \( D A_2 \neq D A_1 \).

But, since the data usability constraints are surpassed (\( \mu \) and \( \sigma \) have changed), so according to our assumption:

\[
D A_2 \neq D A_1
\]

But this is not possible for decoding accuracy to be independent of usability constraints as it is against the definition of usability independent watermark decoding given in Definition 8; so \( D A_2 \) needs to be equal to \( D A_1 \) for our assumption to be true.

**Definition 9.** [Decoding threshold.] Given a watermarked dataset \( D W \), decoding threshold \( \gamma \) is a variable that is used to decode a watermark \( W \) from \( D W \).

The decoding threshold is computed using a correction factor.

**Definition 10.** [Correction factor.] Given a watermarked dataset \( D W \), correction factor \( \tau \) is a constant to set the value of \( \gamma \) such that a watermark \( W \) is correctly decoded from \( D W \).

The relation for computing the value of \( \gamma \) is:

\[
\gamma = \text{val} - \tau
\]

where, \( v_{ij}' \) represent the value of \( j^{th} \) attribute in \( i^{th} \) row. The motivation for using \( \tau \) along with \( \text{val} \) in equation 17 is to account for the possible errors introduced by an attacker in the watermarked dataset.

We have done a number of pilot studies to conclude that the correction factor must be less than the minimum value of \( \delta \) for every watermarked tuple if the decoding accuracy were to be made independent of the usability constraints. Following theorem explains the logic behind this conclusion.

**Theorem 1.** The value of correction factor \( \tau \) must be less than the minimum absolute value of \( \delta \) in order to get an appropriate value of \( \gamma \), if the decoding accuracy were to be made independent of the usability constraints.

**Proof:** Let \( v_{ij} \) be the value of an unsign numeric attribute \( A \) in \( i^{th} \) tuple in the original dataset. After embedding a watermark bit \( b \), the value of the attribute becomes \( v_{ij} \). Now, according to Algorithm 4, if \( b = 1 \), then \( \rho > 0 \) as depicted in equation (15), and if \( b = 0 \), then \( \rho < 0 \) according to equation (16).

The value \( v_{ij}' \) is computed as:

\[
v_{ij}' = v_{ij} + \delta_{ij}
\]

Suppose that the value of decoding threshold \( \gamma \) (Definition 9) is greater than or equal to zero if the embedded bit was 1; otherwise the value of \( \gamma \) is less than zero.

Now using equations (17) and (18), and the fact that \( v_{ij} > 0 \); if \( \rho > 0 \) then \( \text{val} > 0 \); and if \( \rho < 0 \) then \( \text{val} < 0 \). So, the parameter \( \text{val} \) is always greater than zero if embedded bit is 1 (as \( \rho \) is positive according to equation (15)). Therefore, for \( \gamma \) to be greater than or equal to zero (and hence to decode the watermark bit as 1), the necessary condition is:

\[
\text{val} > \tau
\]

Similarly, if the embedded bit was 0 then the parameter \( \text{val} \) is always less than zero, (as \( \rho \) is negative according to equation (16)). Therefore, for \( \gamma \) to be less than zero (and hence to decode the watermark bit as 0), the necessary condition is:

\[
\text{val} < \tau
\]

These two conditions are only true if and only if \( \tau \) is less than the absolute value of the parameter \( \text{val} \), so:

\[
\tau < |\text{val}|
\]

According to Lemma 1, the watermark decoding accuracy can only be independent of usability constraints if the above condition is met for every possible value of the parameter \( \text{val} \) even if an attacker is able to surpass the bounds on the usability constraints. Also, if usability constraints are tight and the parameter \( \text{val} \) is calculated using equation (18); then it is similar to calculating \( \delta \) (equations (15) and (16)), except \( \text{val} \) will be calculated
from a different dataset (the watermarked dataset). As a consequence, the values of parameters $\text{val}$ and $\delta$ will be approaching each other; so $\text{val}_{ij} \approx \delta_{ij}$ for a tuple $i$ in $j^{th}$ column. Hence, we can conclude, the necessary condition for value of $\tau$ is:

$$\tau < |\delta|$$

This condition needs to be satisfied for every possible absolute value of $\delta$; therefore, the above condition becomes:

$$\tau < |\delta_{\text{min}}|$$

where, $\delta_{\text{min}}$ denotes the minimum value of $\delta$ among all its possible values and hence the theorem is proven.

Figure 2 shows the feasible region for different values of $\tau$ for different usability constraints, as obtained by the outcome of our pilot studies. In this figure y-axis shows the minimum absolute values of $\delta$ obtained with values of $\rho = \{0.1\%, 0.2\%, ..., 1\%, 2\%, ..., 10\% \}$ and x-axis shows the corresponding optimum upper bound for the values of $\tau$ bounded by its maximum value for corresponding value of $\rho$.

**Corollary 1.** For every possible value of $\tau$ (defined using Theorem 1) and $\tau_{ij} > 0$; if $\gamma$ is positive, a bit is decoded as 1; and if $\gamma$ is negative, and a bit is decoded as 0.

**Corollary 2.** $\gamma$ as defined above, ensures that the watermark decoding accuracy is independent of each element of usability constraints set $G$ with elements $\mu$ and $\sigma$.

**Proof:** Let $\mu_1$ and $\sigma_1$ be the mean and standard deviation of all the values of $j^{th}$ attribute in a dataset $D_W$. Also, consider that all values of $j^{th}$ attribute are greater than zero. If $D_W$ has $\beta$ tuples, $\mu_1$ can be calculated as $\mu_1 = \frac{\sum_{i=1}^{\beta} v'_{ij}}{\beta}$ and $\sigma_1 = \sqrt{\frac{1}{\beta} \sum_{i=1}^{\beta} (v_{ij} - \mu_1)^2}$, where $v'_{ij}$ is the value of $i^{th}$ tuple in $j^{th}$ attribute.

Suppose, an attacker – Mallory – launches attacks on this attribute such that new mean and standard deviation become $\mu_2$ and $\sigma_2$ and all values of $j^{th}$ attribute remain greater than zero. If $\delta_{ij} \Delta_j$ represents the change (positive or negative) made by Mallory in a tuple $i$ of column $j$, the new value becomes:

$$v''_{ij} = v'_{ij} + \delta_{ij}$$

And for all the tuples in $D_W$, the new dataset after attacks can be represented as:

$$D'_W = D_W + \Delta'$$

Since, the tuples’ values have been changed and value of $\mu_2$ and $\sigma_2$ depend on the values of tuples (see formulas for $\mu$ and $\sigma$); therefore, we have four cases: (1) $\mu_2 = \mu_1$ and $\sigma_2 = \sigma_1$; (2) $\mu_2 = \mu_1$ and $\sigma_2 \neq \sigma_1$; (3) $\mu_2 \neq \mu_1$ and $\sigma_2 = \sigma_1$; and (4) $\mu_2 \neq \mu_1$ and $\sigma_2 \neq \sigma_1$. For all of these four cases, only two options for $\gamma$ are available: (1) for $\rho > 0$, equations (17), (18) and (19) give $\gamma > 0$ and a bit is decoded as 1 (Corollary 1 of Theorem 1) irrespective of the amount change represented by $\mu_2$ and $\sigma_2$; and (2) similarly for $\rho < 0$, equations (17), (18) and (19) give $\gamma < 0$ and a bit is decoded as 0 (Corollary 1 of Theorem 1) irrespective of the amount change represented by $\mu_2$ and $\sigma_2$.

Therefore, $\gamma$ ensures that the watermark decoding accuracy is independent of each element of usability constraints set $G$.

**4.4.2 Once-for-all usability constraints**

Before going into the details of this subsection, it is important to define some basic concepts.

**Definition 11.** [Information Loss] If $I$ is the information obtained from the original data $D$, and $I_W$ is the information obtained from the watermarked data $D_W$, then percentage information loss $I_{\text{Loss}}$ as a result of watermarking is:

$$I_{\text{Loss}} = \left| \frac{I - I_W}{I} \right| * 100$$

We measure information loss in terms of data statistics such as mean, standard deviation, and data distribution etc. The data distortions, in turn, are represented in terms of information loss. The small values of information loss mean less data distortions and vice versa.

**Definition 12.** [Fit data.] A watermarked data $D_W$ is said to be fit for a particular application App if it does not violate the bounds on data distortions related to that App.

**Lemma 2.** If a watermarked dataset $D_{W_1}$ is fit for an application which allows the minimum possible distortions $d_u$ in the original data then $D_{W_1}$ is fit for all other possible applications of the same dataset.

**Proof:** Let $\{d_1, d_2, ..., d_k\}$ be the sorted list (in ascending order) of constraints on the upper bound of data distortions $\{d_1, d_2, ..., d_k\}$ acceptable by $\mathcal{R}$ data recipients $\{\text{Rec}_1, \text{Rec}_2, \text{Rec}_3, ..., \text{Rec}_k\}$ for applications $\{\text{App}_1, \text{App}_2, \text{App}_3, ..., \text{App}_k\}$. 

Fig. 2. Feasible region for values of $\tau$. 

| Diagram | Feasible region for values of $\tau$. |
If \( \{ I_{Loss1}, I_{Loss2}, \ldots, I_{LossK} \} \) is the information loss after distortions \( \{ d_1, d_2, \ldots, d_K \} \) in the original dataset \( D \), then by using Definition 11:

\[
I_{Loss1} < I_{Loss2} < I_{Loss2} \ldots < I_{LossK-1} < I_{LossK}
\]

Now, as the minimum information loss \( I_{Loss1} \) is possible with distortions \( d_1 \) having upper bound \( d_1^n \), and:

\[
d_1^n < d_2^n < \ldots < d_{K-1}^n < d_K^n
\]

So, the constraints on upper bounds \( \{ d_1^n, d_2^n, \ldots, d_K^n \} \) of data distortions \( \{ d_2, d_3, \ldots, d_k \} \), acceptable by data recipients \( \{ Rec_2, Rec_3, \ldots, Rec_k \} \), will always be satisfied by \( d_1^n \) and hence, the dataset \( D_{W1} \) will be fit for all applications \( \{ App_1, App_2, App_3, \ldots, App_K \} \).

**Definition 13.** [Severity of attack.] The severity of an attack, \( \varsigma \), is a vector containing percentage of attacked tuples and degree of alteration in certain statistics.

\[
\varsigma = \{ \eta, \alpha \} \tag{23}
\]

Where, \( \eta \) is the percentage of attacked tuples and \( \alpha \) is the degree of alteration. The degree of alteration for tuple insertion and tuple alteration attacks is the amount of alteration. In comparison, for tuple deletion attacks \( \varsigma \) contains \( \alpha \) denoting the percentage of usability constraints violated during launch of deletion attacks.

**Definition 14.** [Once-for-all usability constraints.] Usability constraints can be said to be “once-for-all” if a particular usability constraints matrix \( G_p \) results in minimum possible distortions, acceptable to the recipient, yet ensuring the maximum possible watermark robustness – acceptable to the data owner.

**Theorem 2.** If watermark decoding accuracy is independent of usability constraints, then usability constraints definition is “once-for-all”.

**Proof:**

Let \( \{ G_1, G_2, \ldots, G_k \} \) be the usability constraints for watermarking to deliver the watermarked data to \( K \) data recipients. Also, suppose that the set \( \{ d_1, d_2, \ldots, d_k \} \) denotes the sorted list (in ascending order) of the amount of data distortions acceptable by \( K \) data recipients \( \{ Rec_1, Rec_2, \ldots, Rec_k \} \) with the corresponding usability constraints \( \{ G_1, G_2, \ldots, G_k \} \). So, \( Rec_1 \) is the data recipient who accepts the minimum possible distortions \( d_1 \) in the original dataset \( D \) after embedding a watermark \( W \). Also, let that \( d_1^n \) be the upper bound for the distortions, \( d_1 \).

If \( Rob_{max} \) is the maximum watermark robustness achieved by watermark \( W \) with \( d_1^n \) distortions, then for \( Rob_{max} \), the decoding accuracy \( DA \) after any attack (insertion, deletion, alteration, and every possible combination of these attacks), with severity \( \varsigma \), is 100%.

But according to Theorem 1, the value of \( \tau \) is such that \( DA \) is 100% – independent of usability constraints and for that matter \( DA \) is independent of the amount of distortions \( \{ d_1, d_2, \ldots, d_k \} \). So, \( Rob_{max} \) is achieved by having any amount of distortion from the possible data distortions set \( \{ d_1, d_2, \ldots, d_k \} \).

Also other data recipients \( \{ Rec_2, Rec_3, \ldots, Rec_k \} \) would allow data distortions with an upper bound \( \{ d_2^n, d_3^n, \ldots, d_k^n \} \) respectively, which could effectively allow more robustness than allowed by data distortions \( d_1^n \). But, as Alice has already achieved \( Rob_{max} \) with \( d_1^n \); therefore, she does not need to define different data usability constraints for any other possible application or use of the same dataset with maximum allowed distortions in the set \( \{ d_2^n, d_3^n, \ldots, d_k^n \} \). Moreover, according to Lemma 2 the data with distortions \( d_1 \) will be fit for all applications which allow data distortions \( \{ d_2, d_3, \ldots, d_k \} \). So, strictly speaking Alice has defined “once-for-all” usability constraints for watermarking a dataset \( D \) for every possible application and use.

**Here, we revisit the claims made about the major contributions of this paper in Introduction and we prove that they have been met by our watermark embedding approach.**

1) A novel watermark decoding algorithm which:(1) ensures that its decoding accuracy is independent of the usability constraints (or available bandwidth); and (2) enables “once-for-all” usability constraints definition by providing the maximum robustness with the least possible distortions.

2) The robustness of watermark is achieved even if an attacker is somehow able to successfully corrupt the watermark in some selected part of the dataset.

3) The robustness of proposed watermarking scheme by analyzing its decoding accuracy under different types of malicious attacks other than additive attacks.

4) The robustness of proposed scheme against additive attacks is achieved by ensuring that Mallory can not remove the Alice’s watermark from her watermarked dataset and hence she can prove the presence of her watermark in Mallory’s watermarked dataset.

**Remark 2.** Theorems 1 and 2 prove the major claim (Claim 1) by achieving two objectives:(1) making the decoding accuracy independent of the usability constraints; and (2) defining “once-for-all” usability constraints. The other three claims are dependent (minor) claims that have been implicitly proven formally.

### 4.5 Watermark Decoding

The watermark decoding algorithm extracts the embedded watermark using the secret parameters: \( K_s, m, \tau \). The watermark bits are decoded in the reverse order – the last embedded bit is decoded first and so on. This order is preferred because it is easier to detect the manipulations done while computing the last watermark bit. The algorithm starts by generating the data partitions \( S_0, \ldots, S_{m-1} \) using the watermarked dataset \( D_W \), the secret key \( K_s \), and the number of partitions \( m \) as input
to the data partitioning algorithm. It then generates dataset $D'_T$ from $D_T$ by computing even hash values as discussed in Section 4.2. In the next step, a parameter $val$ is computed using equation (18).

To extract the embedded bit, the decoding threshold $\gamma$ (calculated using equation (17)) is used to decide whether the decoded bit is 0 or 1.

**Algorithm 5 Detect_Watermark**

**Input:** Watermarked dataset $D_W$, $K_s$, $m$, $\tau$, Watermark length $l$  
**Output:** Detected Watermark $W_D$

1. $ones=0$
2. $zeros=0$
3. $S_{W_0}, \ldots, S_{W_{m-1}} = Get$ _Partitions_ $(D_W, K_s, m)$
4. for each partition $S_{W_j}$ do
5. $D_{W_j} = Get$ _Data_Selection_ _Threshold_ $(R_{W_j}, \phi)$
6. $D_{W_j} = Get$ _Even_Hash_Value_ _Dataset_ $(D_{W_j}, K_s)$
7. for each row $r$ in $D'_T$ do
8. $\gamma = val-\tau$
9. if $\gamma \geq 0$ then
10. $ones[r]=ones[r]+1$
11. else
12. $zeros[r]=zeros[r]+1$
13. end if
14. end for
15. if $ones[r]>zeros[r]$ then
16. $b[r]=1$
17. else
18. $b[r]=0$
19. end if
20. end for
21. return $W_D$

During the decoding phase, if $\gamma$ for a tuple is greater than or equal to 0, the decoded watermark bit is 1; otherwise, it is 0. Recall that in the watermark embedding phase, if for a particular watermark bit, the change $\rho$ in data statistics was negative then that change can always be detected as negative by utilizing the knowledge of $\rho$ until and unless an attacker is able to change the value of watermarked tuple to zero or multiply the same with a negative number. After decoding all the bits from the watermarked dataset, the majority voting scheme is used to eliminate decoding errors (if any) as a result of malicious attack (or attacks). The steps of watermark decoding are shown in Algorithm 5.

It is important to emphasize that our watermark decoding algorithm is blind because it does not need the original data or the embedded watermark bits during the watermark decoding process.

## 5 Experiments and Results

In this section, we report the results of our experiments to substantiate the claimed merits of our watermarking approach. The major motivation of designing experiments is to prove the Claim 1 which guarantees that the decoding accuracy is independent of the usability constraints under different attack scenarios (see last section). We have selected a subset of 50000 tuples from a real-life dataset that shows the power consumption rates of consumers. The dataset is available through CIMEG project\(^7\. The watermark length is 29 bits (as the conversion of UTC data-time to binary string yields a bit string consisting of 29 bits). The number of partitions $m=100$ and $\rho = \pm 0.3$ are used. Moreover, the value of $\tau$ is calculated using Theorem 1. All experiments have been performed on a server that has Pentium(R) Dual-Core CPU 2.10GHz with 4GB of RAM.

### 5.1 Data Distortions

Table 2 shows the values of parameters $\mu$ and $\sigma$ for original and watermarked data once we set the above-mentioned parameters. One can easily notice that the changes in $\mu$ (mean) and $\sigma$ (standard deviation) are 0.004% and 0.003% respectively, that are very small; hence Bob’s requirement for minimum data distortions is fulfilled. We can further reduce these values by applying tighter usability constraints and it will not effect the decoding accuracy as proven in Theorem 1. Hence, a sub-part of the Claim 1 about “minimizing data distortions” has been proven by this experiment.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu$</td>
<td>0.004%</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>0.003%</td>
</tr>
</tbody>
</table>

### 5.2 Attack Analysis

Consider Alice generates a data $D_W$ by inserting a watermark $W$ in the dataset $D$ using our watermark embedding function $\phi$. An attacker – Mallory – wants to launch different types of attacks on the watermarked data in order to corrupt or delete the watermark but at the same time he wants to preserve data quality so that it remains useful for recipients as well. We suppose that he has no access to the original dataset $D$ and does not have access to the secret parameters used in the embedding of the watermark: $K_s$, $m$, $c$, $\rho$, and $\tau$. Mallory, with the above-mentioned assumptions, cannot generate the data partitions $\{S_0, \ldots, S_m-1\}$ because this requires the knowledge of $K_s$ and $m$; as a result, he cannot corrupt (or remove) certain watermark bits from some selected partitions. Moreover, it is also not possible for him to guarantee that his attack will not violate the usability constraints because he does not have access to the original dataset $D$. Mallory’s Dilemma (as an attacker) is: how can he successfully corrupt the watermark without violating the usability constraints? We have studied the robustness of our watermarking scheme against tuple deletion, insertion and alteration attacks. Moreover, we have also analyzed sophisticated attacks – multifaceted and additive – in our attack model as well. We also test the distinguished characteristic of our technique, usability constraints independent watermark decoding accuracy, by launching the aforementioned attacks with a severity that violates the usability constraints by surpassing the available bandwidth.

---

5.2.1 Deletion Attack

In this attack, Mallory deletes selected tuples from a watermarked dataset with an aim to remove the watermark. He can either randomly delete tuples or selects them in a sophisticated manner on the basis of statistical distribution of attribute values. Suppose he drops $\eta$ tuples from Alice’s watermarked dataset. We have used different values of $\eta$ and then analyzed its effect on the decoding accuracy and also compared our results with that of a threshold based database watermarking technique (see Figure 3). In Figure 3, WRDOBT refers to a threshold based database watermarking technique [2], that also used different optimization schemes for watermark embedding. It is evident from Figure 3 that the decoding accuracy of our technique remains 100% even when more than 90% of the tuples are deleted, whereas the decoding accuracy of [2] is deceased when more than 80% tuples are deleted. Moreover, our technique also overcomes the synchronization errors issue of [10] because it does not use any marker tuple or position of tuples for partitioning or watermark embedding.

The same robustness is observed once an attacker launches high frequency deletion attacks (see Figure 4) such that usability constraints are violated, that is, $\mu$ and $\sigma$ are changed after these attacks. Our investigation reveal that the defense against such attacks is made possible because: (1) we have embedded each bit of the watermark in every watermarked tuple; and (2) the decoding accuracy is independent of usability constraints as the value of $\tau$ was set according to Theorem 1. So, usability constraints are “once-for-all” according to Lemma 2 and Theorem 2.

5.2.2 Insertion Attack

We tested our technique against two type of insertion attacks: fixed insertion and constraint reliant insertion. In the first attack, Mallory inserts new $\eta$ tuples by replicating values of existing $\beta$ tuples. Our technique is resilient against this attack as shown in Figure 5. The reason for this robustness is that: (1) blind insertion simply does not affect the watermarked tuples; and (2) marker-free data partitioning and watermark embedding also ensure that synchronization errors are prevented.

In the second attack, he generates the tuple values based on the mean $\mu$ and the standard deviation $\sigma$ of watermarked dataset. He deviates tuples values by $\pm \alpha\%$ from the original values in the watermarked dataset. The results in Figure 5 show that for all possible values of $\alpha$ the decoding accuracy remains 100%. In Figure 5 the lines showing decoding accuracy for different values of $\alpha$ have been superimposed on each other.

We have also performed experiments for insertion attacks with high severity which violate usability constraints with an aim to destroy the embedded watermark. The results reported in Figure 6 show that the proposed scheme is able to correctly detect the watermark with 100% accuracy without taking into account the amount of usability constraints violated while the decoding accuracy of WRDOBT [2] decreases significantly when usability constraints are violated. Our technique is able to handle every range of data usability constraints violations, but, in Figure 6 the results for data usability constraints violated uptil 25% are reported for brevity.

The reason for this desirable behavior is ensured through Theorem 1 by ensuring that the decoding accuracy of our decoding algorithm does not depend on the usability constraints. Hence, the usability constraints are “once-for-all” according to Lemma 2 and Theorem 2.
5.2.3 Alteration Attack

Mallory alters the attribute values with an aim to flip the watermark bits. He can again do it in two ways: fixed alteration attack and constraint reliant alteration attack. In the fixed alteration attack, he alters $\eta$ selected tuples from the total of $\beta$ tuples. He may choose a fixed value $\pm \alpha$ and alters all $\eta$ tuples with this amount. Figure 7 shows the results for fixed alteration attack on $D_W$ and it shows that the proposed technique is robust against this attack as well.

In constraint reliant attack, Mallory alters tuple values in the range $\pm \alpha \%$ such that mean $\mu$ and standard deviation $\sigma$ of watermarked dataset is preserved. Figure 7 show that our decoding algorithm achieves 100% watermark decoding accuracy and hence shows strong resilience against constraint resilience attack is proven.

We have also performed experiments for high range alteration attacks in which an attacker may perform large scale alteration attacks with a high severity, using larger values of $\pm \alpha$, with an aim to destroy the embedded watermark by violating some of the usability constraints.

Figure 8 shows that the proposed scheme is able to correctly detect the watermark with 100% accuracy without taking into account the amount of alterations made by an attacker. However, in Figure 8 the results for violations only up to 25% in the usability constraints are reported for brevity. Again it is clear that our technique outperforms WRDOBT [2] and we believe that the reason for this desirable behavior stems in Theorem 1 that ensures that the decoding accuracy of our decoding algorithm is independent of the usability constraints; and as a consequence, the usability constraints are “once-for-all” according to Lemma 2 and Theorem 2.

5.2.4 Multifaceted Attack

A sophisticated attacker can generate any permutation of insertion, deletion or alteration attacks – choosing in between fixed and constraints reliant attacks – to launch a multifaceted attack. Our pilot studies reveal that combining insertion with alteration attack is more lethal. It is obvious from Table 3 that our technique is resilient against multifaceted attack as well. The reason behind this desirable behavior is that the majority voting corrects the decoding errors introduced due to attacks. This behavior, as expected, is in compliance with Theorem 1. However, if Mallory inserts more than 50% (of original tuples) new tuples and deletes more than 50% of original tuples, the decoding accuracy may decrease and we recommend the use majority voting for that purpose to reduce the decoding errors to some extent.

Let us consider an interesting scenario: if an attacker alters the Alice’s data to some signed or zero-valued data then the decoding accuracy might be decreased. Consider the case when an attacker inserts the signed data (by changing the sign of an attribute value), the watermark decoding accuracy will definitely be degraded. The reason is that, for a particular watermark bit, the expected decoder value ($\gamma$) should be positive but it turned out to be negative. In another case, its expected value of $\gamma$ should be negative but it turned out to be positive. In such cases, if an attacker is able to alter sign of more than 50% (to evade the effect of majority voting) of the watermarked tuples, the watermark decoding
accuracy might suffer. However, Alice knows that her watermarked attribute is unsigned so she may easily take the absolute values of the signed tuples; as a result, she can detect the watermark form the absolute values of the tuples. So strictly speaking, an attacker cannot change the sign of the attribute values in some selected tuples, if the same attribute does not contain any signed value prior to the attack. Next bet for an attacker is to corrupt the embedded watermark by changing the values of some tuples to zero. But Alice can chose to watermark tuples with non-zero values only; as a result, she will definitely leave the rows with zero values during the process of watermark detection.

Our technique is best suited for datasets that contain unsigned numeric attributes. In the real world, almost all kinds of numeric datasets contain attribute(s) with unsigned numeric data: e.g. medical datasets, weather datasets, sales datasets, game datasets etc. However, if a dataset contains signed attribute (or an attribute having valid zero values) then that attribute should not be selected for watermarking using the proposed approach. If such an attribute is selected for watermarking, the majority voting scheme helps to remove the decoding errors if an attacker is able to change the sign bit of less than 50% of marked tuples.

<table>
<thead>
<tr>
<th>κ</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
<th>30</th>
<th>35</th>
<th>40</th>
<th>45</th>
<th>50</th>
<th>ω</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>0.5</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>1.0</td>
</tr>
<tr>
<td>15</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>1.5</td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>2.0</td>
</tr>
<tr>
<td>25</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>2.5</td>
</tr>
<tr>
<td>30</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>3.0</td>
</tr>
<tr>
<td>35</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>3.5</td>
</tr>
<tr>
<td>40</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>4.0</td>
</tr>
<tr>
<td>45</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>4.5</td>
</tr>
<tr>
<td>50</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>5.0</td>
</tr>
</tbody>
</table>

Remark 3. We believe such accurate decoding was achieved because we set the value of τ according to Theorem 1. Hence, another sub-part of the Claim 1 about “making the watermark decoding accuracy independent of usability constraints” and the second and third claims (minor claims) regarding robustness of the proposed scheme against various malicious attacks have been proven empirically.

5.2.5 Collusion attack
The collusion attacks are possible if the different sets of the same data are watermarked with a different watermark. Remember that our technique provides a data owner an efficient way of defining “once-for-all usability constraints” such that a particular data can be used in every possible application (Lemma 2 and Theorem 2); therefore, an owner does not need to use different watermarks for different data recipients who may or may not use the data for different applications. The proposed technique makes the watermark robustness independent of usability constraints; therefore, it is possible for a data owner to deliver the same set of watermarked data to multiple recipients without compromising watermark robustness and data usability. Consequently, collusion attacks are implicitly handled by the proposed technique.

5.2.6 Additive attack: Scenarios and Solutions
In additive attack, Mallory may attempt to establish a plausible but spurious claim of ownership by trying to splice (or insert) his watermark with that of Alice. The conflict in ownership can be resolved by integrating a trusted third party which facilitates distribution of key among the involved parties. When Alice shares her data, she affixes 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 [3].

The other option might be that the owner of the database can request a secret key – from the trusted party – which is usually employed as a secret parameter during the encoding and decoding phases. The key is obviously delivered on a particular date-time. Such time constraints can also help in resolving ownership conflicts: the owner can claim the insertion of watermark before an attacker did so by taking date-time, issued by the trusted party, as a reference.

One of the postulation 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 (the owner) can demonstrate the presence of watermarks in Mallory’s dataset D’ since it belongs to her; whereas Mallory cannot illustrate the existence of his marks in Alice’s original dataset D.

Remark 4. Hence, the last claim (minor claim) for providing different solutions to counter additive attacks has been proven in Section 5.2.6. The major requirement of Alice to have “100% robustness” against every kind of malicious attack is ensured by our decoding scheme irrespective of the severity of the any particular attack. Similarly, the major requirement of Bob to have “minimum distortions” in the watermarked dataset has been met also; therefore, now Alice does not need to define different usability constraints when she wants to share her data for any possible application or use as proven in Lemma 2 and Theorem 2; as a result, a sub-part of the Claim 1 regarding “once-for-all” usability constraints has been proven.

6 Conclusion
In this paper, we have proposed a technique that is highly resilient against insertion, deletion, alteration and multifaceted attack yet it results in minimum distortions
in the original dataset. Regardless of the severity of malicious attack on the watermarked data, the watermark bits are successfully decoded with 100% accuracy because the decoding accuracy of the proposed approach is independent of the usability constraints. Moreover, our security mechanism also helps to resolve ownership conflicts over watermarked dataset in case of additive attacks. All these features facilitate Alice to define “once-for-all” usability constraints for her dataset for its every possible application or use. Furthermore, our technique provides her “maximum possible robustness” and delivers data to Bob with “minimum data distortions”. The results of our experiments on a real world dataset substantiate our claims. Recall that the proposed technique is restricted to numeric unsigned data only. A logical extension of this research is to make it scale to signed data and non-numeric relational datasets as well. We are also looking to find more elegant ways to solve the problem of additive attacks. Our solutions to these challenging problems will be the subject of the forthcoming publications.

ACKNOWLEDGMENTS

First author would like to thank Higher Education Commission (HEC) of Pakistan for funding a Ph.D. fellowship under its indigenous scheme with the grant number 063-111271-eg3-028. The first and third authors of this paper are supported, in part, by the National ICT R&D Fund, Ministry of Information Technology, Government of Pakistan through a project “Remote Patient Monitoring System”.

REFERENCES


M. Kamran got his BS and MS degree in Computer Science in 2005 and 2008 respectively. Currently he is pursuing his Ph.D. studies at NUANCES, Islamabad. He is also working as a research associate at nexGIN RC–NUCES. His research interests include the use of machine learning and evolutionary computations techniques for data security in databases and health informatics, data mining, and decision support systems.

Sabah Suhail did BS in Software Engineering in 2008 from Fatima Jinnah Women University, Rawalpindi, Pakistan and MS in Information Security from National University of Science and Technology (NUST), Islamabad, Pakistan in 2012. Currently, she is working as a research student at nexGIN RC–NUCES. Her research interests include data security and decision support systems.

Muddassar Farooq completed his D.Sc. in Informatics from Technical University of Dortmund, Germany, in 2006. Currently he is working as professor and head, department of electrical engineering, NUANCES, Islamabad, Pakistan. He is also the director of nexGIN RC–NUCES. His research interests include nature inspired applied systems, usable security, natural computing and nature inspired computer and network security systems. He has authored several papers and book chapters in these areas.