Primary Key Free Watermarking for Numerical Tabular Datasets in Machine Learning

Xin Che[1][0000−0001−8784−5345], Mohammad Akbari[2][0000−0002−5663−3101], Shaoxin Li[1][0009−0009−0695−8803], David Yue[3][0000−0002−8271−348X], Yong Zhang[2][0000−0002−0238−0719], and Lingyang Chu*[1][0000−0002−8937−1750]

[1] McMaster University, Hamilton, Ontario, Canada.
{chex5,li2018,chul9}@mcmaster.ca
[2] Huawei Technologies Canada Co., Ltd. Burnaby, BC, Canada.
{mohammad.akbari,yong.zhang3}@huawei.com
[3] University of Toronto, Toronto, Ontario, Canada.
david.yue@mail.utoronto.ca

Abstract. High-quality tabular datasets are often traded by their owners as valuable digital assets due to their scarcity and usefulness in training machine learning models. A pivotal concern when trading the datasets is their ownership, which is seriously threatened by piracy due to the simplicity of reselling illegal copies. This produces an urgent demand for an effective watermarking method to demonstrate the ownership of the dataset. Existing database watermarking methods rely on either a primary key or a virtual primary key to watermark a tabular dataset. These methods cannot work well in the context of machine learning, because a primary key can be easily modified without affecting the machine learning utility of a tabular dataset, and a virtual primary key is often not robust against watermark-removing attacks. How to watermark a tabular dataset without using a primary key or virtual primary key is a challenging task that has not been systematically studied before. In this paper, we tackle this task by a novel primary key free method that embeds a sinusoidal signal as the watermark into a discrete-time signal constructed from the tabular dataset. We conduct an in-depth theoretical analysis on the exceptional robustness of our watermark against five challenging attacks, and further validate the robustness through comprehensive experiments on two real-world datasets.

Keywords: primary key free · tabular dataset watermarking · robust.

1 Introduction

Artificial intelligence powered by machine learning has brought significant benefits to the modern society. In many successful applications, large machine learning models are fueled by huge amount of tabular datasets, such as marketing data [8], healthcare data [4], environment & climate data [13], and sensor

* Corresponding author

data [16]. Due to the enormous efforts and cash invested in data collection and management, these datasets are often regarded as high value digital assets of their owners. However, when these datasets are traded in the market, their safety is usually threatened by piracy due to the simplicity of creating and reselling illegal copies. This produces an urgent demand for an effective watermarking method that embeds a detectable watermark into a dataset to demonstrate the ownership of the dataset and its copies.

As discussed later in Section 2, existing methods are mostly not robust to watermark removing attacks because they embed their watermarks based on the primary key (PK) [2,3,15,37] or virtual primary key (VPK) [3,9,10,25]. The use of PK and VPK significantly weaken the robustness of existing methods against attacks, because both PK and VPK can be easily modified by an attacker to remove the watermark without significantly decreasing the machine learning utility of the dataset. As far as we know, how to watermark a tabular dataset without using a primary key (or a virtual primary key) is a novel and challenging task that has not been well studied in the literature.

In this paper, we systematically tackle this task by formulating and solving a novel problem named primary key free watermarking for numerical tabular datasets. We make the following contributions. First, we propose the novel task of primary key free watermarking for tabular datasets. The goal is to embed and detect watermarks on tabular datasets without using primary key (or a virtual primary key) while achieving good robustness against watermark-removing attacks. Second, we successfully tackle the problem with a carefully designed watermarking method. The key idea is to first map the data instances in a tabular dataset to a discrete-time signal, and then embed a watermark by adding a sinusoidal signal to the discrete-time signal. The watermark can be accurately detected by checking the existence of the sinusoidal signal. Last, we conducted extensive experiments on two real-world datasets to compare the performance of our method with five state-of-the-art baseline methods. The experimental results demonstrate the superior robustness of our watermark against six challenging watermark-removing attacks.

2 Related Work

Many existing works [3, 10, 12, 15, 19, 25, 33–35] have been proposed to embed and detect watermarks in tabular datasets. Our work is related to the following methods.

The **primary key methods** [2,3,17,35] rely on the primary key of a tabular dataset to embed and detect watermarks. Most primary key methods [2, 3, 12, 35, 37] use a primary key to uniquely identify watermarked data instances in order to accurately detect watermark. Some other works [5, 15, 20, 26, 32–35, 35] use primary key to organize data instances in groups to embed and detect watermarks. These methods work well when the primary key of a watermarked tabular dataset stays unchanged. However, they cannot accurately detect the watermark if an attacker modifies the primary key. Since modifying the primary

key of a tabular dataset often does not reduce its machine learning utility in training good machine learning models, the resale value of the tabular dataset in the application areas of machine learning is not affected. Therefore, the primary key methods cannot effectively protect tabular datasets against piracy, because an attacker can easily modify the primary key of a watermarked tabular dataset to create an illegal copy, which successfully escapes watermark detection and also preserves the resale value.

The **virtual primary key methods** [3,9,10,25] compute a virtual primary key (VPK) from data instances and use it as a substitution of primary key to embed and detect watermarks. For example, Agrawal et al. [3] use the most significant bits of an attribute to compute VPK. Li et al. [25] select multiple attributes to compute VPK. By using a VPK, existing primary key methods can be extended to watermark a tabular dataset. These methods are robust to primary key modification, because they do not use primary key and the VPK is secretly computed from data instances. However, a watermark embedded by using VPK is often not robust against watermark-removing attacks [3,22,34,35], because modifying the data instances changes the values of VPK [10,11,34]. As a result, by slightly modifying the tabular dataset, an attacker can remove a VPK-based watermark without causing much damage to the machine learning utility of the dataset.

To the best of our knowledge, our work is the first in the literature to watermark a numerical tabular dataset without using a primary key (or a virtual primary key) while achieving outstanding robustness against watermark-removing attacks. This makes our work particularly effective in protecting numerical tabular datasets against piracy in the application areas of machine learning.

3 Task Definition

In this section, we first introduce a typical application example of the proposed numerical tabular dataset watermarking task in Figure 1. Then, we give the formal definition of our task.

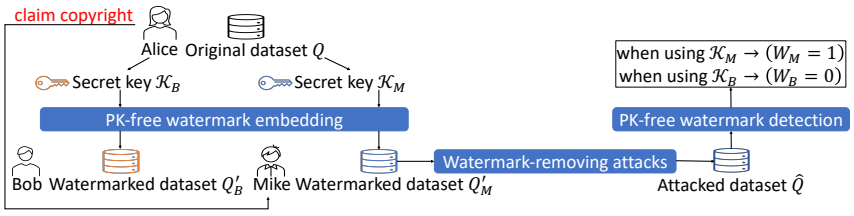


Fig. 1. A typical application scenario.

Example 1 (A typical application). As shown in Figure 1, Alice owns a valuable numerical tabular dataset Q and she wants to sell it to Bob and Mike. Before

selling Q, Alice uses two different secret keys $\mathcal{K}_B$ and $\mathcal{K}_M$ to embed two different watermarks in Q. The watermarked dataset Q'_B produced by $\mathcal{K}_B$ is sold to Bob. The watermarked dataset Q'_M produced by $\mathcal{K}_M$ is sold to Mike. Mike uses watermark-removing attacks to modify Q'_M into $\hat{Q}$ and put $\hat{Q}$ on the market for sale. Alice uses each of $\mathcal{K}_B$ and $\mathcal{K}_M$ to detect watermark from $\hat{Q}$, which produces $W_B = 0$ and $W_M = 1$, respectively. Since $W_M = 1$ means $\hat{Q}$ is watermarked by $\mathcal{K}_M$, Alice demonstrates her ownership on $\hat{Q}$ and she also knows $\hat{Q}$ comes from the copy she sold to Mike.

Next, we first define some related concepts and then introduce the formal definition of our task, which includes two parts such as primary key (PK)-free watermark embedding and PK-free watermark detection.

Definition 1 (Numerical tabular dataset). *A numerical tabular dataset is represented by a matrix, denoted by $Q \in \mathbb{R}^{n \times d}$, where each row stores one data instance and each column corresponds to one* attribute *of the data instances. A tabular dataset is also called a "dataset" in short.*

Definition 2 (Machine learning utility). *Given a numerical tabular dataset Q, the machine learning utility (MLU) of Q, denoted by $\texttt{MLU}(Q)$, indicates the effectiveness of Q in training good machine learning models. It is measured by the performance of a machine learning model trained on Q [19, 24].*

Definition 3 (Secret key). *Denote by Q an original dataset that is not embedded with a watermark and by Q' the watermarked dataset that is embedded with a watermark. A secret key, denoted by $\mathcal{K}$, is a cryptographic key that is used to embed and detect watermark from Q'.*

The secret key $\mathcal{K}$ is completely different from a primary key (PK) or a virtual primary key (VPK). In typical watermarking systems [3], $\mathcal{K}$ often consists of a set of variables storing the information to generate and identify a watermark. While PK and VPK are unique identifiers of data instances. Different watermarking systems use different types of secret keys. Leaking $\mathcal{K}$ exposes the watermark embedded in Q', which makes it vulnerable to watermark-removing attacks. Thus, $\mathcal{K}$ is often kept secret by the owner of the dataset Q.

Definition 4 (PK-free watermark embedding). *Given Q, $\mathcal{K}$ and a positive threshold γ, the process of watermark embedding produces Q' by modifying the data instances in Q. This process should satisfy: (1) no primary key is used; (2) Q' carries a watermark that can be verified by $\mathcal{K}$; and (3) $|\texttt{MLU}(Q) - \texttt{MLU}(Q')| \leq \gamma$.*

In the above conditions, (1) requires the embedded watermark to be independent from primary key, which improves the robustness of watermark against primary key modification. (2) means Q' is watermarked by $\mathcal{K}$. (3) establishes an MLU constraint, which limits the damage on $\texttt{MLU}(Q)$ caused by the modification on Q when embedding the watermark. This preserves the resale value of Q' because $\texttt{MLU}(Q')$ is close to $\texttt{MLU}(Q)$.

Definition 5 (PK-free watermark detection). *Given $\mathcal{K}$ and a suspicious dataset $\hat{Q}$ that may or may not be watermarked by $\mathcal{K}$, the process of watermark detection verifies whether $\hat{Q}$ is watermarked by $\mathcal{K}$. This process returns a binary variable $W \in \{0, 1\}$, where $W = 1$ means $\hat{Q}$ is watermarked by $\mathcal{K}$ and $W = 0$ means $\hat{Q}$ is not watermarked by $\mathcal{K}$. This process should satisfy: (1) no primary key is used; (2) the original dataset Q is not used; and (3) the value of W cannot be flipped without significantly modifying $\hat{Q}$.*

In the above conditions, (1) requires the watermark detection to be primary key free, which mitigates the influence of primary key modification on watermark detection. (2) is a classic requirement of blind watermark detection [15]; it reduces the risk of unauthorized access to the original dataset Q, because the watermark can be detected without revealing Q [3,15,34]. (3) means an attacker has to significantly modify $\hat{Q}$ in order to escape watermark detection. Since a larger modification on $\hat{Q}$ causes more damage to $\texttt{MLU}(\hat{Q})$, it reduces more resale value of $\hat{Q}$, which lowers the interest of the attacker in attacking $\hat{Q}$.

4 PK-free Watermark Embedding

The **key idea** of PK-free watermark embedding is to first map the data instances to a discrete-time signal in a two-dimensional space, and then add a sinusoidal signal with a specific frequency to the discrete-time signal by modifying the data instances. This embeds the sinusoidal signal as a watermark into the dataset.

4.1 Mapping Dataset to Discrete-time Signal

To map the data instances in a tabular dataset to a discrete-time signal in a two-dimensional space, we design a pair of mapping functions, denoted by $\phi_x(\cdot)$ and $\phi_y(\cdot)$, where $\phi_x(\cdot)$ maps a data instance to a real-valued x-coordinate and $\phi_y(\cdot)$ maps the same data instance to a real-valued y-coordinate. This maps each data instance to a pair of x and y coordinates, which represents a point in a two-dimensional space. Then, the points of all the data instances are summarized in groups to form the discrete-time signal.

Denote by $Q_{i,:}$ the i-th data instance in a tabular dataset $Q \in \mathcal{R}^{n \times d}$, and by $\mathbf{e}_x, \mathbf{e}_y \in \mathbb{R}^d$ a random pair of *orthogonal vectors* with L2-norm equal to one. The mapping function $\phi_y(\cdot)$ that maps $Q_{i,:}$ to a y-coordinate y_i is defined as

$$y_i = \phi_y(Q_{i,:}) = Q_{i,:}\mathbf{e}_y^\top, \tag{1}$$

which is simply the projection of $Q_{i,:}$ on $\mathbf{e}_y$. The mapping function $\phi_x(\cdot)$ that maps $Q_{i,:}$ to an x-coordinate x_i is defined as

$$x_i = \phi_x(Q_{i,:}) = \frac{\lfloor \frac{Q_{i,:}\mathbf{e}_x^\top}{b} \rfloor}{\tau}, \tag{2}$$

where $b \in \mathbb{R}^+$ and $\tau \in \mathbb{R}^+$ are positive real-valued hyperparameters, and $\lfloor \cdot \rfloor$ is the flooring operator that rounds a real number down to the closest integer. The

Algorithm 1 Watermark embedding

Input: An original dataset Q and a secret key $\mathcal{K}$.
Output: A watermarked dataset Q'.
1: Initialize Q' as a zero matrix in the same size as Q.
2: **for** each data instance $Q_{i,:}$ in Q **do**
3: Compute: $x_i = \phi_x(Q_{i,:})$.
4: Update: $Q'_{i,:} = Q_{i,:} + \lambda \sin(2\pi\theta x_i) * \mathbf{e}_y$. (See Equation (4))
5: **end for**
6: Return Q'.

numerator $\lfloor \frac{Q_{i,:}\mathbf{e}_x^\top}{b} \rfloor$ in Equation (2) conducts a binning operation that projects $Q_{i,:}$ into a bin and returns the index of the bin. The hyperparameter b is the *bin width* of each bin. By dividing the index of bin over τ, Equation (2) maps the index of bin into an x-coordinate, where $\frac{1}{\tau}$ is the *step size* between the x-coordinates of neighbouring bins.

The tuple (x_i, y_i) mapped from $Q_{i,:}$ represents a *point* in a two-dimensional space. By mapping each $Q_{i,:}$ in Q to a point, we obtain a set of points, denoted by $Z = \big\{(x_i, y_i) \mid x_i = \phi_x(Q_{i,:}), y_i = \phi_y(Q_{i,:}), i \in \{1, \ldots, n\}\big\}$. The points in Z cannot form a discrete-time signal because some points may have the same x-coordinates and different y-coordinates due to the binning operation in Equation (2). To convert the points in Z into a discrete-time signal, we first group each subset of points with the same x-coordinates into a bin, denoted by

$$B_h = \{(x_i, y_i) \mid x_i = h, (x_i, y_i) \in Z\}, \tag{3}$$

where h is the value of the x-coordinates of the points in B_h. Then, we summarize the points in B_h into a mean point, denoted by $(\overline{x}, \overline{y})$, where $\overline{x} = h$ is the mean of the x-coordinates of the points in B_h, and $\overline{y}$ is the mean of the y-coordinates of the points in B_h. By doing the above summarization for each possible value of h, we convert the points in Z into a set of mean points with distinct x-coordinates. This set of mean points forms the discrete-time signal, denoted by T. This maps Q to the discrete-time signal T, which is written as $T = \varphi(Q)$.

4.2 Adding Sinusoidal Signal

In this section, we introduce how to add a sinusoidal signal with a specific frequency to the discrete-time signal T by slightly modifying the data instances in Q. The sinusoidal signal is denote by $y = \lambda \sin(2\pi\theta x)$, where λ is the amplitude of the signal, and θ is the frequency of the signal. To add the sinusoidal signal into T, we first map $Q_{i,:}$ to $x_i = \phi_x(Q_{i,:})$, and then update $Q_{i,:}$ by

$$Q'_{i,:} = Q_{i,:} + \lambda \sin(2\pi\theta x_i) * \mathbf{e}_y. \tag{4}$$

By applying Equation (4) on every $Q_{i,:}$ in Q, we modify Q into a watermarked dataset Q', where the sinusoidal signal is embedded as a watermark.

We summarize the method to generate Q' in Algorithm 1, where the secret key, denoted by $\mathcal{K} = \{\mathbf{e}_x, \mathbf{e}_y, \theta, b, \tau\}$, contains the necessary variables to verify the watermark (i.e., the sinusoidal signal). The time complexity of Algorithm 1 is $O(nd)$, where n and d are the numbers of rows and columns in Q, respectively.

Theorem 1. *If Q' is obtained by Algorithm 1, then $T' = \varphi(Q')$ contains a component of the sinusoidal signal with frequency θ.*

Proof. We prove this theorem by showing that, for every mean point $(\overline{x}', \overline{y}') \in T'$, the analytical form of $\overline{y}'$ contains a sinusoidal term $\lambda \sin(2\pi\theta\overline{x}')$. Without loss of generality, we assume $\overline{x}' = h$ and derive the analytical form of $\overline{y}'$ as follows.

Since $\overline{x}' = h$, we know $(\overline{x}', \overline{y}')$ is summarized from the bin

$$B'_h = \big\{(x'_i, y'_i) \mid x'_i = h, (x'_i, y'_i) \in Z'\big\}, \tag{5}$$

where $Z' = \big\{(x'_i, y'_i) \mid x'_i = \phi_x(Q'_{i,:}), y'_i = \phi_y(Q'_{i,:}), i \in \{1, \ldots, n\}\big\}$.

For each point $(x'_i, y'_i) \in B'_h$, we can derive from Equation (4) and $\mathbf{e}_x\mathbf{e}_y^\top = 0$ that $Q'_{i,:}\mathbf{e}_x^\top = Q_{i,:}\mathbf{e}_x^\top$. Then, we can derive from Equation (2) that

$$x'_i = \phi_x(Q'_{i,:}) = \frac{\lfloor \frac{Q'_{i,:}\mathbf{e}_x^\top}{b} \rfloor}{\tau} = \frac{\lfloor \frac{Q_{i,:}\mathbf{e}_x^\top}{b} \rfloor}{\tau} = \phi_x(Q_{i,:}) = x_i. \tag{6}$$

Since $(x'_i, y'_i) \in B'_h$, we know $x'_i = h$ by the definition of B'_h. Thus,

$$x'_i = x_i = h. \tag{7}$$

Since $\mathbf{e}_y\mathbf{e}_y^\top = 1$, we can derive from Equations (1) and (4) that

$$y'_i = Q'_{i,:}\mathbf{e}_y^\top = Q_{i,:}\mathbf{e}_y^\top + \lambda \sin(2\pi\theta x_i) = y_i + \lambda \sin(2\pi\theta x_i). \tag{8}$$

By plugging Equation (7) into the above equation, we have

$$y'_i = y_i + \lambda \sin(2\pi\theta h), \tag{9}$$

which holds for every point $(x'_i, y'_i) \in B'_h$.

Since $\overline{y}'$ is the mean of the y-coordinates of all the points $(x'_i, y'_i) \in B'_h$, we can derive the analytical form of $\overline{y}'$ as

$$\overline{y}' = \frac{1}{|B'_h|} \sum_{(x'_i, y'_i) \in B'_h} y'_i = \Big(\frac{1}{|B'_h|} \sum_{(x'_i, y'_i) \in B'_h} y_i\Big) + \lambda \sin(2\pi\theta h). \tag{10}$$

Since $\overline{x}' = h$, the analytical form of $\overline{y}'$ is

$$\overline{y}' = \Big(\frac{1}{|B'_h|} \sum_{(x'_i, y'_i) \in B'_h} y_i\Big) + \lambda \sin(2\pi\theta \overline{x}'), \tag{11}$$

which contains the sinusoidal term $\lambda \sin(2\pi\theta\overline{x}')$ with frequency θ.

Algorithm 2 Watermark detection

Input: A suspicious dataset $\hat{Q}$, a confidence level $p \in [0, 1]$ and $\mathcal{K}$.
Output: The detection result $W \in \{0, 1\}$.
1: Obtain the discrete-time signal $\hat{T} = \varphi(\hat{Q})$.
2: Obtain the spectrum power $\hat{P}(\theta)$ from $\hat{T}$ by LSP [36].
3: Use LSP to estimate the threshold η_p from p.
4: **If** $\hat{P}(\theta) \geq \eta_p$, **then** return $W = 1$. (*Watermark is detected*)
5: **If** $\hat{P}(\theta) < \eta_p$, **then** return $W = 0$. (*Watermark is not detected*)

According to Theorem 1, T' contains the sinusoidal signal with frequency θ, which means Q' is successfully embedded with the sinusoidal signal by Algorithm 1.

Theorem 2. *Denote by η the maximum absolute value of all the entries in $\mathbf{e}_y$ and by $\triangle_{i,j} = |Q_{i,j} - Q'_{i,j}|$ the absolute modification made on the j-th attribute of $Q_{i,:}$ when embedding the watermark. If Q' is obtained by Algorithm 1, then $\triangle_{i,j} \leq \lambda\eta$.*

Proof. Since each $Q'_{i,:}$ is obtained by modifying $Q_{i,:}$ using Equation (4) and $-1 \leq \sin(2\pi\theta x_i) \leq 1$, we have $\triangle_{i,j} \leq \lambda\eta$.

5 PK-free Watermark Detection

In this section, we introduce how to detect a watermark from a suspicious dataset $\hat{Q}$ that may or may not be watermarked by a secret key $\mathcal{K}$.

Denote by $\hat{T}$ the discrete-time signal of $\hat{Q}$. We obtain $\hat{T}$ by mapping the data instances in $\hat{Q}$ in the same way as how we map Q to T, that is, $\hat{T} = \varphi(\hat{Q})$. This process requires to know the variables $\mathbf{e}_x$, $\mathbf{e}_y$, b and τ in $\mathcal{K}$. Then, we check whether $\hat{T}$ contains the sinusoidal signal with the frequency $\theta \in \mathcal{K}$ by checking the spectrum power of $\hat{T}$ at the frequency θ, denoted by $\hat{P}(\theta)$.

We use Lomb-Scargle Periodogram (LSP) [27, 36] to compute the spectrum power $\hat{P}(\theta)$ of $\hat{T}$. LSP provides a probabilistic method to determine whether a sinusoidal signal is a true signal in $\hat{T}$ [36]. Denote by $p \in [0, 1]$ the probability of a sinusoidal signal being a true signal in $\hat{T}$, LSP estimates a threshold η_p based on p. If $\hat{P}(\theta) \geq \eta_p$, then the probability of $\hat{T}$ containing the sinusoidal signal with frequency θ is at least p. This allows us to use p as a confidence level when detecting watermark from $\hat{T}$. For example, we can set $p = 0.99$ and compare $\hat{P}(\theta)$ with the threshold $\eta_{0.99}$. If $\hat{P}(\theta) \geq \eta_{0.99}$, then $\hat{T}$ is watermarked by $\mathcal{K}$ at the confidence level of 0.99. Otherwise, $\hat{T}$ is not watermarked by $\mathcal{K}$ at the confidence level of 0.99. Algorithm 2 summarizes how to detect watermark. The time complexity is $O\big(n(d+1)\big)$, where n and d are the number of rows and columns of $\hat{Q}$, respectively.

6 Threat Model and Attacks

In this section, we provide a comprehensive discussion of the four requirements of the threat model and delve into six typical watermark-removing attacks that an attacker might employ.

Threat model. Following the literature [14, 21, 29, 30, 34], we consider a typical threat model consisting of four requirements: **(1)** the attacker can access the watermarked dataset Q'; **(2)** the attacker cannot access the original dataset Q; **(3)** the attacker cannot access the secret key $\mathcal{K}$; and **(4)** the attacker cannot change the feature space of the original attributes in Q'. Here, the requirement (4) is practical because the semantic meaning (e.g., meta-data) carried by the original attributes of Q' is a valuable part of Q'. Moreover, if the feature space of an attacked dataset, denoted by $\tilde{Q}$, is different from the feature space of Q', then a machine learning model trained on $\tilde{Q}$ cannot generalize to new data instances represented by the original attributes of Q'.

Watermark-removing attacks. We consider the following typical attacks in the literature. **(1)** Uniform alteration (UA) [15, 30, 34] adds uniform noise sampled from $U[-\rho_{ua}, \rho_{ua}]$ to the attributes of all the data instances in Q'. A larger ρ_{ua} implies a stronger attack. **(2)** Row deletion (RD) [15, 21, 30, 34] deletes uniformly sampled data instances of Q'. Denote by ρ_{rd} the proportion of the deleted data instances in Q', a larger ρ_{rd} implies a stronger attack. **(3)** Row insertion (RI) [15,21,30,34] inserts noise data instances to Q'. For each noise data instance, the j-th entry is sampled from a uniform distribution $U[\mu_j - \sigma_j, \mu_j + \sigma_j]$, where μ_j and σ_j are the mean and standard deviation of j-th attribute of Q'. Denote by ρ_{ri} the proportion of inserted noise data instances, a larger ρ_{ri} implies a stronger attack. **(4)** Column deletion (CD) deletes uniformly sampled columns in Q'. Denote by ρ_{cd} the proportion of the deleted columns, a larger value of ρ_{cd} implies a stronger attack. **(5)** PCA attack (PCA) modifies the data instances in Q' by using principal component analysis (PCA) [1] to perform dimensionality reduction. We map the data instances back to the original feature space after discarding k dimensions in the feature space spanned by eigenvectors. Denote by $\rho_{pca} = \frac{k}{d}$ the proportion of discarded dimensions, a larger ρ_{pca} implies a stronger attack. **(6)** Re-watermarking (RE) [18] attacks the original watermark in Q' by embedding a new watermark into Q'. We use the proposed watermarking method to embed the new watermark. Denote by ρ_{re} the amplitude of the new sinusoidal signal $y = \rho_{re} * \sin(2\pi\theta x)$ embedded into Q', a larger ρ_{re} implies a stronger attack.

7 Experiments

In this section, we conduct comprehensive experiments on two real-world datasets to study the performance of our method and five baseline methods. We focus on answering two questions: **(1)** How robust are the watermarks of each watermarking method against the attacks? **(2)** How is the machine learning utility of

Table 1. Information of datasets.

Dataset	#Instances	#Attributes	#Classes
Forest cover type (FCT) dataset [29]	581,012	54	7
Gas sensor array drift (GSAD) dataset [7]	13,910	128	6

a watermarked dataset affected by the attacks? All the experiments were conducted on a desktop with an Intel(R) Core(TM) i9-10900K CPU @ 3.70GHz and 64 gigabytes of RAM.

Datasets. We use the two real-world datasets FCT[4] and GSAD[5] in Table 1. Since the original FCT dataset is too big for the baseline methods OBT [34] and IP [19] to finish the experiments in practical time, we uniformly sample 50% of instances from the original FCT dataset to do our experiments. Following the setting of [20], we use the top-4 (top-20) attributes with the largest information gain to embed watermark in FCT (GSAD). Denote by m the number of attributes used to embed watermark. Embedding watermark in this way increases the cost of conducting column deletion attack, because deleting a column with a larger information gain causes more damage to the machine learning utility of the dataset.

Machine learning utility (MLU). We evaluate the MLU of a dataset by the testing accuracy of a machine learning model. Each dataset is uniformly split into a training set and a testing set with a ratio of 4:1. The training set is used to embed/detect watermarks and train the machine learning model. The testing set is used to evaluate the machine learning model's testing accuracy, which is regarded as the MLU of the training dataset. We evaluate MLU by two machine learning models, such as multi-class logistic regression (LR) model [23] and multi-class support vector machine (SVM) [38].

Baseline methods. We use five baseline methods, such as NR [33], OBT [34], IP [19], GAHSW [15], and SCPW [30]. These methods need a primary key or a virtual primary key to work properly. Since an attacker can easily modify the primary key to remove a watermark without damaging the dataset's machine learning utility, we develop two versions of implementations for each baseline method by using two state-of-the-art virtual primary key generation methods. One version uses M-Scheme [25] to implement the baselines as NR-M, OBT-M, IP-M, GAHSW-M and SCPW-M. The other version uses HQR [10] to implement the baselines as NR-H, OBT-H, IP-H, GAHSW-H and SCPW-H.

Secret keys. For each of the compared methods, we use 10 independent secret keys, denoted by $\mathcal{K}_{(1)}, \ldots, \mathcal{K}_{(10)}$, to embed watermarks. The baseline methods use a sequence of bits as a secret key and we use 16 bits as the default length of each secret key. For each baseline method, we use 10 secret keys with maximum pairwise hamming distance. For our method, the sampled values of θ, b and τ in the 10 secret keys are listed in Table 2. The vectors $\mathbf{e}_x$ and $\mathbf{e}_y$

[4] `https://archive.ics.uci.edu/dataset/31/covertype`
[5] `https://archive.ics.uci.edu/dataset/224/gas+sensor+array+drift+dataset`

Table 2. The θ, b and τ of the 10 secret keys used for FCT and GSAD.

Dataset	FCT										GSAD									
Secret keys	$\mathcal{K}_{(1)}$	$\mathcal{K}_{(2)}$	$\mathcal{K}_{(3)}$	$\mathcal{K}_{(4)}$	$\mathcal{K}_{(5)}$	$\mathcal{K}_{(6)}$	$\mathcal{K}_{(7)}$	$\mathcal{K}_{(8)}$	$\mathcal{K}_{(9)}$	$\mathcal{K}_{(10)}$	$\mathcal{K}_{(1)}$	$\mathcal{K}_{(2)}$	$\mathcal{K}_{(3)}$	$\mathcal{K}_{(4)}$	$\mathcal{K}_{(5)}$	$\mathcal{K}_{(6)}$	$\mathcal{K}_{(7)}$	$\mathcal{K}_{(8)}$	$\mathcal{K}_{(9)}$	$\mathcal{K}_{(10)}$
θ	30	27	30	32	29	34	37	40	38	39	30	32	34	33	35	36	38	29	35	37
b ($\times 10^{-2}$)	0.8	0.5	0.5	2.0	3.0	0.8	2.0	2.0	1.0	1.0	0.2	0.1	0.5	1.0	0.2	2.0	1.0	2.0	0.4	0.8
τ ($\times 10^4$)	1.0	1.0	0.3	0.5	1.0	0.5	0.9	0.8	0.6	0.8	0.5	0.4	1.0	0.6	0.8	1.0	0.3	0.3	0.5	0.3

for each secret key are randomly sampled as a pair of orthogonal vectors with L2-norm equal to one.

Watermark strength. The watermark strength in a dataset refers to the intensity of the embedded watermark signal [28]. A stronger watermark enhances robustness against removal attacks but also increases dataset modification, reducing MLU [31]. Thus, increasing watermark strength trades MLU for robustness. To fairly compare the robustness of all methods, we allow each to trade up to 0.01 MLU for robustness, setting $\gamma = 0.01$ for the MLU constraint in condition (3) of Definition 4. Based on this constraint, we set the default value of λ for our method to 20 for GSAD and 0.8 for FCT unless otherwise specified.

7.1 How to Evaluate Performance?

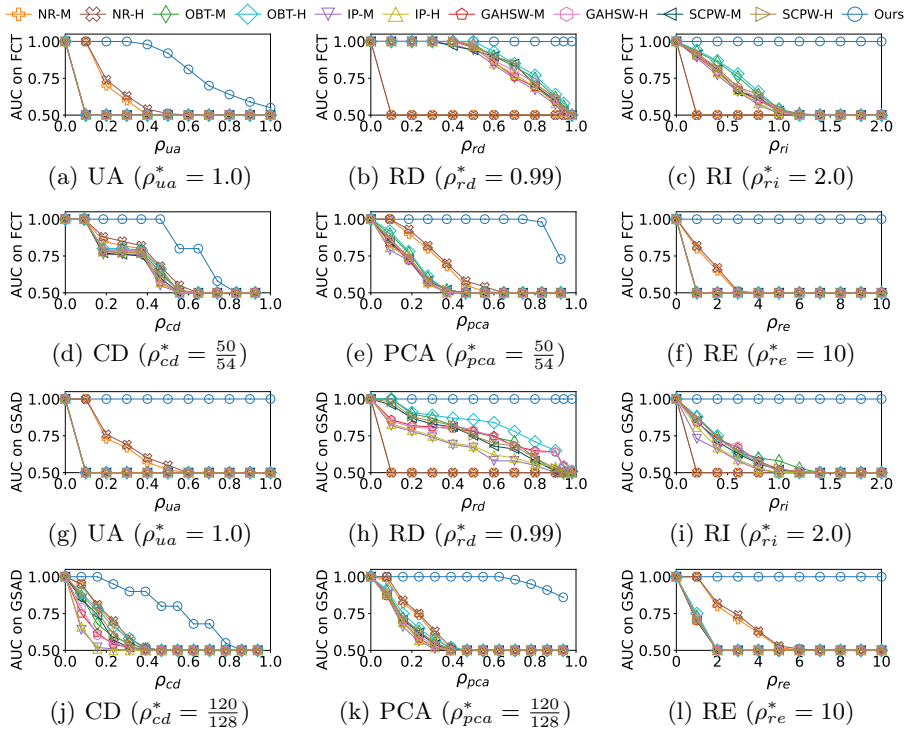
We evaluate the performance of a watermarking method by performing the following steps.

Step1: embedding watermarks. Denote by $Q_{(0)}$ the training set of an original dataset that is not watermarked. We use each of the secret keys $\mathcal{K}_{(1)}, \ldots, \mathcal{K}_{(10)}$ to embed a watermark in $Q_{(0)}$. This produces 10 watermarked datasets, denoted by $Q'_{(1)}, \ldots, Q'_{(10)}$, where $\mathcal{K}_{(i)}$ is the ground truth secret key of $Q'_{(i)}$. The ground truth secret key of $Q_{(0)}$ is denoted by $\mathcal{K}_0$, which means no watermark is embedded in $Q_{(0)}$. In this way, we construct a collection of datasets denoted by $\mathcal{C} = \{Q_{(0)}, Q'_{(1)}, \ldots, Q'_{(10)}\}$.

Step 2: conducting attacks. We attack the datasets in $\mathcal{C}$ before detecting the watermarks. For each attack, we produce a collection of attacked datasets, denoted by $\tilde{\mathcal{C}} = \{\tilde{Q}_{(0)}, \tilde{Q}_{(1)}, \ldots, \tilde{Q}_{(10)}\}$.

Step 3: detecting watermarks. We use each of $\mathcal{K}_{(1)}, \ldots, \mathcal{K}_{(10)}$ to detect watermark from the datasets in $\tilde{\mathcal{C}}$. Denote by $\texttt{Detect}(\tilde{Q}_{(j)}, \mathcal{K}_{(i)}) \to W$ the process of using $\mathcal{K}_{(i)}$ to detect watermark from $\tilde{Q}_{(j)}$. If $\texttt{Detect}(\tilde{Q}_{(j)}, \mathcal{K}_{(i)}) = 1$, then we have a positive detection. If $\texttt{Detect}(\tilde{Q}_{(j)}, \mathcal{K}_{(i)}) = 0$, then we have a negative detection. A positive detection is a true positive if $\mathcal{K}_{(i)}$ is the ground truth secret key of $\tilde{Q}_{(j)}$; otherwise, it is a false positive. A negative detection is a true negative if $\mathcal{K}_{(i)}$ is not the ground truth secret key of $\tilde{Q}_{(j)}$; otherwise, it is a false negative. Last, we compute the true positive rate and false positive rate.

Step 4: evaluating performance. We evaluate the performance of a watermarking method by the area under curve (AUC) of the receiver operating characteristic (ROC) curve [6]. A larger AUC means a better performance. For each compared method, the ROC curve is obtained by changing the value of the



(a) UA ($\rho^*_{ua} = 1.0$) (b) RD ($\rho^*_{rd} = 0.99$) (c) RI ($\rho^*_{ri} = 2.0$)

(d) CD ($\rho^*_{cd} = \frac{50}{54}$) (e) PCA ($\rho^*_{pca} = \frac{50}{54}$) (f) RE ($\rho^*_{re} = 10$)

(g) UA ($\rho^*_{ua} = 1.0$) (h) RD ($\rho^*_{rd} = 0.99$) (i) RI ($\rho^*_{ri} = 2.0$)

(j) CD ($\rho^*_{cd} = \frac{120}{128}$) (k) PCA ($\rho^*_{pca} = \frac{120}{128}$) (l) RE ($\rho^*_{re} = 10$)

Fig. 2. The AUC on FCT shown in (a)-(f) and GSAD shown in (g)-(l). The $\rho^*_{ua}, \rho^*_{rd}, \rho^*_{ri}, \rho^*_{cd}$, ρ^*_{pca} and ρ^*_{re} are the maximum values of each parameter.

threshold that decides whether a watermark exists or not. Different methods use different thresholds, for our method, the threshold is η_p in Algorithm 2.

7.2 How Robust Are the Watermarks?

In this section, we analyze the robustness of watermark against the attacks listed in Section 6. Figure 2 shows the AUC of each watermarking method on FCT and GSAD. The y-axis shows the AUC and the x-axis shows the strength of each attack controlled by $\rho_{ua}, \rho_{rd}, \rho_{ri}, \rho_{cd}$, ρ_{pca} and ρ_{re} in Section 6.

We can see in Figure 2 that the AUC of most methods drops when the attack strength increases. A slower dropping speed of AUC means a better performance, because it implies the watermarking method is more robust to withstand a stronger attack. Since the AUC on FCT and GSAD show similar trends, we focus on explaining the results on FCT.

The AUC of baseline methods. As shown in Figure 2, the AUC of the baseline methods are inferior to our method because they are affected by attacks due to the following reasons. First, since these methods use virtual primary keys to identify the watermarked groups of data instances, their watermarks are not

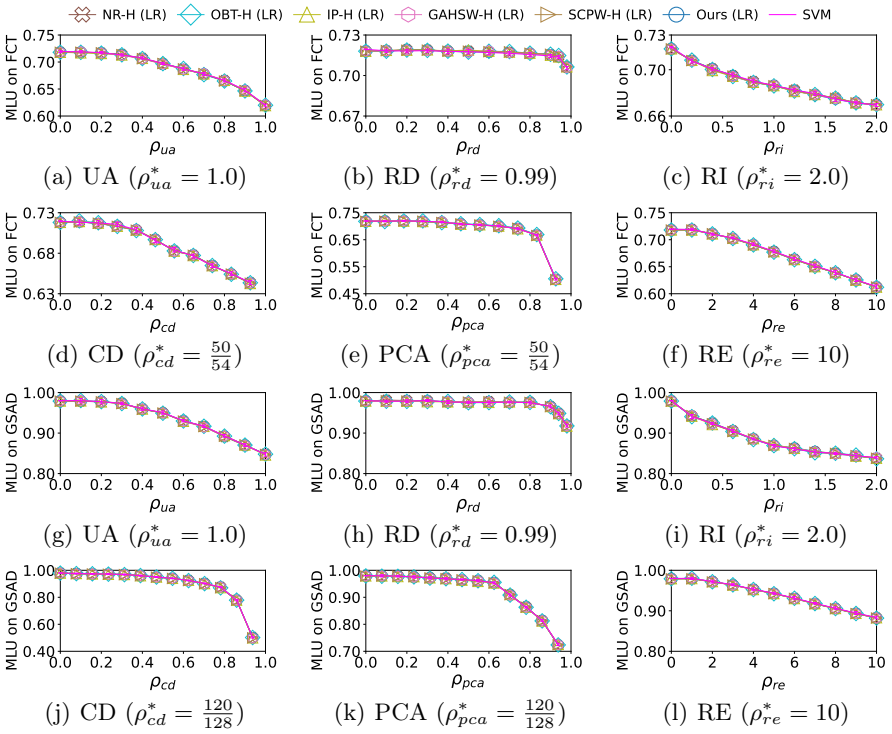
robust to uniform alteration, column deletion, PCA attack and re-watermark. This is because such attacks change the values of virtual primary key, which corrupts the identification of the watermarked groups. Second, the watermarks of OBT, IP, GAHSW and SCPW are affected by row deletion (RD) and row insertion (RI), because RD removes watermarked data instances from watermarked groups and RI adds noise data instances into watermarked groups. Both the effects weakens the watermark signal. NR is affected by row deletion and row insertion because it relies on the accurate alignment between the data instances before and after attack. The alignment between data instances is disrupted by RD and RI because they change the number of data instances.

The AUC of our method. As shown in Figure 2, our method achieves the best AUC, which demonstrates the outstanding robustness of our watermark. In the following, we discuss the performance of our method against each attack. **(1)** Uniform alteration (UA). In Figures 2(a) and 2(g), our method is robust to UA because: **i)** it is primary key free, enhancing noise robustness; **ii)** the x-axis binning operation stabilizes the discrete time signal against noise; and **iii)** the Lomb-Scargle Periodogram (LSP) [27, 36] is noise-resistant. UA has a bigger impact upon our method on FCT than on GSAD because GSAD has more attributes than FCT, thus embedding a watermark causes less damage to the MLU of GSAD than FCT. This allows our method to embed a stronger watermark on GSAD without violating the MLU constraint in Definition 4. **(2)** Row deletion (RD) and row deletion (RD). In Figures 2(b), 2(c), 2(h) and 2(i), our method achieves outstanding AUC against RD and RI. Because RD removes watermarked points from B'_h and RI adds noise points in B'_h, but neither changes the remaining watermarked points in B'_h. Thus, the watermark signal is largely retained. **(3)** Column deletion (CD). In Figures 2(d) and 2(j), CD impacts our method's AUC because deleting one of the m watermarked columns (i.e., attributes) in Q' removes $\frac{1}{m}$ of the watermark signal. However, since our watermark is embedded in multiple columns that are unknown to the attacker, we still achieve high AUC even when half the columns of Q' are randomly deleted. **(4)** PCA attack. In Figures 2(e) and 2(k), the impact of PCA attack is smaller than column deletion. This is because the dimensions discarded by PCA attack often have small information gain but we embed our watermark in the columns with large information gain. **(5)** Re-watermarking (RE). In Figures 2(f) and 2(l), the AUC of our watermark is robust against RE. This is because embedding a new watermark adds random noise to Q', which has a similar effect to uniform alteration attack. Therefore, our watermark is robust against RE.

7.3 How Is MLU Affected by the Attacks?

Figure 3 shows the MLU of each watermarked dataset under different strengths of attacks. The x-axis is defined in the same way as Figure 2 and the y-axis shows the MLU of the attacked watermarked datasets.

Since drawing all the MLU curves is too crowded, we simplify the view as follows. First, for MLU measured by LR, we only draw the MLU of our method



(a) UA ($\rho^*_{ua} = 1.0$) (b) RD ($\rho^*_{rd} = 0.99$) (c) RI ($\rho^*_{ri} = 2.0$)

(d) CD ($\rho^*_{cd} = \frac{50}{54}$) (e) PCA ($\rho^*_{pca} = \frac{50}{54}$) (f) RE ($\rho^*_{re} = 10$)

(g) UA ($\rho^*_{ua} = 1.0$) (h) RD ($\rho^*_{rd} = 0.99$) (i) RI ($\rho^*_{ri} = 2.0$)

(j) CD ($\rho^*_{cd} = \frac{120}{128}$) (k) PCA ($\rho^*_{pca} = \frac{120}{128}$) (l) RE ($\rho^*_{re} = 10$)

Fig. 3. The MLU on FCT shown in (a)-(f) and GSAD shown in (g)-(l) The $\rho^*_{ua}, \rho^*_{rd}, \rho^*_{ri}, \rho^*_{cd}$, ρ^*_{pca} and ρ^*_{re} are the maximum values used for each parameter.

and each baseline method using HQR [10] as VPK, marked with a "(LR)" suffix. We omit the baseline methods using M-Scheme [25] as VPK because their absolute MLU difference is at most 0.004. Second, for MLU measured by SVM, we draw one curve marked "SVM" in Figure 3. This curve with an error bar represents the mean and standard deviation (std) of the MLU of all methods, including baseline methods using both VPK versions and our method. The std is at most 0.003 in all cases.

Why are the MLU curves close? This is due to the small constraint of $\gamma = 0.01$. Since Q' from different watermarking methods is computed from the same dataset Q, the MLU of different Q' are almost identical. Thus, when these Q' undergo the same attack, they exhibit similar MLU curves.

What did we learn from the MLU curves? The MLU of the attacked watermarked dataset decreases with stronger attacks. This means an attacker cannot indefinitely increase the strength of attack to remove a watermark, because a stronger attack will reduce more MLU, which causes more damage to the resale value of the dataset. Since all watermarking methods have similar MLUs, the method with the slowest AUC drop as attack strength increases offers the best protection. Our method provides the best protection, because its AUC drops the slowest in Figure 2.

8 Conclusion

In this paper, we propose a novel primary key free watermarking method for tabular datasets. Different from many existing watermarking methods, our method does not use a primary key to embed and detect watermarks. This makes it particularly suitable for watermarking tabular datasets used for machine learning, because such datasets often do not come with a primary key and an existing primary key can be easily modified without degrading the machine learning utility of the dataset. As demonstrated by extensive experiments, our method achieves outstanding robustness against many watermark-removing attacks, which provides strong protection on watermarked datasets.

References

1. Abdi, H., Williams, L.J.: Principal component analysis. Computational Statistics **2**, 433–459 (2010)
2. Agrawal, R., Haas, P.J., Kiernan, J.: Watermarking relational data: Framework, algorithms and analysis. The VLDB Journal **12**, 157–169 (2003)
3. Agrawal, R., Kiernan, J.: Watermarking relational databases. In: VLDB. pp. 155–166 (2002)
4. Anand, A., Singh, A.K.: Watermarking techniques for medical data authentication: A survey. Multimedia Tools and Applications **80**, 165–197 (2021)
5. Bhattacharya, S., Cortesi, A.: A distortion free watermark framework for relational databases. In: ICSOFT. pp. 229–234 (2009)
6. Bradley, A.P.: The use of the area under the roc curve in the evaluation of machine learning algorithms. Pattern Recognition **30**, 1145–1159 (1997)
7. Das, P., Manna, A., Ghoshal, S.: Gas sensor drift compensation by ensemble of classifiers using extreme learning machine. In: ICSGE. pp. 197–201 (2020)
8. Even, A., Shankaranarayanan, G., Berger, P.D.: Economics-driven data management: An application to the design of tabular data sets. IEEE TKDE **19**, 818–831 (2007)
9. Gort, M.L.P., Díaz, E.A., Uribe, C.F.: A highly-reliable virtual primary key scheme for relational database watermarking techniques. In: CSCI. pp. 55–60 (2017)
10. Gort, M.L.P., Feregrino-Uribe, C., Cortesi, A., Fernández-Peña, F.: Hqr-scheme: A high quality and resilient virtual primary key generation approach for watermarking relational data. Expert Systems with Applications **138**, 795–825 (2019)
11. Gort, M.L.P., Feregrino-Uribe, C., Cortesi, A., Fernández-Pena, F.: A double fragmentation approach for improving virtual primary key-based watermark synchronization. IEEE Access **8**, 504–516 (2020)
12. Gupta, G., Pieprzyk, J.: Database relation watermarking resilient against secondary watermarking attacks. In: ICISS. pp. 222–236 (2009)
13. Hashim, H.: Hybrid warehouse model and solutions for climate data analysis. Journal of Computer and Communications **8**, 75–98 (2020)
14. Hu, D., Wang, Q., Yan, S., Liu, X., Li, M., Zheng, S.: Reversible database watermarking based on order-preserving encryption for data sharing. ACM Transactions on Database Systems **48**, 1–25 (2023)
15. Hu, D., Zhao, D., Zheng, S.: A new robust approach for reversible database watermarking with distortion control. IEEE TKDE **31**, 1024–1037 (2018)

16. Hülsmann, J., Traub, J., Markl, V.: Demand-based sensor data gathering with multi-query optimization. In: VLDB. vol. 13, pp. 2801–2804 (2020)
17. Iftikhar, S., Kamran, M., Anwar, Z.: Rrw—a robust and reversible watermarking technique for relational data. IEEE TKDE **27**, 1132–1145 (2014)
18. İşler, D., Cabana, E., Garcia-Recuero, A., Koutrika, G., Laoutaris, N.: Freqywm: Frequency watermarking for the new data economy. arXiv preprint arXiv:2312.16547 (2023)
19. Kamran, M., Farooq, M.: An information-preserving watermarking scheme for right protection of emr systems. IEEE TKDE **24**, 1950–1962 (2011)
20. Kamran, M., Farooq, M.: A formal usability constraints model for watermarking of outsourced datasets. IEEE TIFS **8**, 1061–1072 (2013)
21. Kamran, M., Farooq, M.: A comprehensive survey of watermarking relational databases research. arXiv preprint arXiv:1801.08271 (2018)
22. Kumar, S., Singh, B.K., Yadav, M.: A recent survey on multimedia and database watermarking. Multimedia Tools and Applications **79**, 149–197 (2020)
23. Kwak, C., Clayton-Matthews, A.: Multinomial logistic regression. Nursing research **51**, 404–410 (2002)
24. Li, Q., Wang, X., Pei, Q., Lam, K.Y., Zhang, N., Dong, M., Leung, V.C.: Database watermarking algorithm based on decision tree shift correction. IEEE Internet of Things Journal **9**, 24373–24387 (2022)
25. Li, Y., Swarup, V., Jajodia, S.: Constructing a virtual primary key for fingerprinting relational data. In: Proceedings of the ACM Workshop on Digital Rights Management. pp. 133–141 (2003)
26. Liu, Q., Xian, H., Zhang, J., Liu, K.: A random reversible watermarking scheme for relational data. In: SecureComm. pp. 413–430 (2022)
27. Lomb, N.R.: Least-squares frequency analysis of unequally spaced data. Astrophysics and Space Science **39**, 447–462 (1976)
28. Podilchuk, C.I., Delp, E.J.: Digital watermarking: Algorithms and applications. IEEE Signal Processing Magazine **18**, 33–46 (2001)
29. Rani, S., Halder, R.: Comparative analysis of relational database watermarking techniques: An empirical study. IEEE Access **10**, 970–989 (2022)
30. Ren, Z., Fang, H., Zhang, J., Ma, Z., Lin, R., Zhang, W., Yu, N.: A robust database watermarking scheme that preserves statistical characteristics. IEEE TKDE (2023)
31. Šarčević, T., Mayer, R., Rauber, A.: Adaptive attacks and targeted fingerprinting of relational data. In: ICBD. pp. 5792–5801 (2022)
32. Sebé, F., Domingo-Ferrer, J., Castella-Roca, J.: Watermarking numerical data in the presence of noise. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems **14**, 495–508 (2006)
33. Sebé, F., Domingo-Ferrer, J., Solanas, A.: Noise-robust watermarking for numerical datasets. Modeling Decisions for Artificial Intelligence pp. 134–143 (2005)
34. Shehab, M., Bertino, E., Ghafoor, A.: Watermarking relational databases using optimization-based techniques. IEEE TKDE **20**, 116–129 (2007)
35. Sion, R., Atallah, M., Prabhakar, S.: Rights protection for relational data. In: SIGMOD. pp. 98–109 (2003)
36. VanderPlas, J.T.: Understanding the lomb–scargle periodogram. The Astrophysical Journal Supplement Series **236**, 1–16 (2018)
37. Wang, C., Li, Y.: A copyright authentication method balancing watermark robustness and data distortion. In: CSCWD. pp. 1178–1183 (2023)
38. Wang, Z., Xue, X.: Multi-class support vector machine. Support Vector Machines Applications pp. 23–48 (2014)