Adaptive and Robust Watermark for Generative Tabular Data

Dung Daniel Ngo[1, †] **Archan Ray**[2, †] **Akshay Seshadri**[2, †] **Daniel Scott**[1, *] **Saheed Obitayo**[1] **Niraj Kumar**[2]
Vamsi K. Potluru[1] **Marco Pistoia**[2] **Manuela Veloso**[1]

[1]AI Research, JPMorganChase, New York, NY 10017, USA
[2]Global Technology Applied Research, JPMorganChase, New York, NY 10001, USA

Abstract

In recent years, watermarking generative tabular data has become a prominent framework to protect against the misuse of synthetic data. However, while most prior work in watermarking methods for tabular data demonstrate a wide variety of desirable properties (e.g., high fidelity, detectability, robustness), the findings often emphasize empirical guarantees against common oblivious and adversarial attacks. In this paper, we study a flexible and robust watermarking algorithm for generative tabular data. Specifically, we demonstrate theoretical guarantees on the performance of the algorithm on metrics like fidelity, detectability, robustness, and hardness of decoding. The proof techniques introduced in this work may be of independent interest and may find applicability in other areas of machine learning. Finally, we validate our theoretical findings on synthetic and real-world tabular datasets.

1 INTRODUCTION

In the age of increasingly sophisticated generative models [OpenAI, 2022, Team et al., 2024, Zhang et al., 2022, Jiang et al., 2023, Grattafiori et al., 2024], the use of synthetic data has become ubiquitous in areas where often real data is scarce (e.g., two examples) or under intense regulatory guidelines (e.g., health care [Gonzales et al., 2023] and finance [Assefa et al., 2020, Potluru et al., 2023]). This is partly because generating synthetic data is cost-efficient and, importantly, eliminates the need for human involvement [Monarch, 2021, Fui-Hoon Nah et al., 2023]. However, there is a growing concern that carelessly adopting synthetic data with the same frequency as human-generated data may lead to misinformation and privacy breaches [Carlini et al., 2021, Stadler et al., 2022]. Furthermore, quality-control in synthetic data is important as repeatedly training models on low-quality AI-generated data exhibit gradual degradation in performance [Shumailov et al., 2024] (also called 'model collapse'). For instance, in medical setting, many hospitals with fewer patients do not have enough internal data to train their rare disease prediction model. While using synthetic data to augment their internal model is a promising direction, repeatedly generating and relying on synthetic patient data as the primary source of data bring forth both ethical and performance concerns. Hence, they often rely on data sharing initiatives, where a shared platform have aggregated real patient data from other large hospitals as well as industry firms and academic institutes. Here, one of the data sharing platform's goal is to ensure that the downstream small hospital can verify that this data source comes from real patient data and not synthetic data. From this data taken from the shared platform, the smaller hospitals can then train their prediction models without worrying about the authenticity of their data source.

To this end, watermarking generative data has gained prominence [Kirchenbauer et al., 2023, Kuditipudi et al., 2024, An et al., 2024, He et al., 2024, Zheng et al., 2024]. Here, a hidden pattern is embedded in the data that may be indiscernible to an oblivious human, yet can be detected through an efficient procedure. Several measures are used to gauge the quality of such algorithms including fidelity, detectability, and robustness [Katzenbeisser and Petitcolas, 2000, Atallah et al., 2001]. Watermarking sequential data is well-studied, but algorithms for tabular data have only recently gained attention. This is partly because tabular datasets often follow relational algebra, and are therefore harder to watermark. While prior works [He et al., 2024, Zheng et al., 2024] have proposed watermarking for tabular data, a comprehensive theoretical framework has remained elusive.

[†]Equal Contribution
[*]Work done while at AI Research, JPMorganChase

Proceedings of the 42nd Conference on Uncertainty in Artificial Intelligence (UAI 2026), PMLR 337:4796–4831.

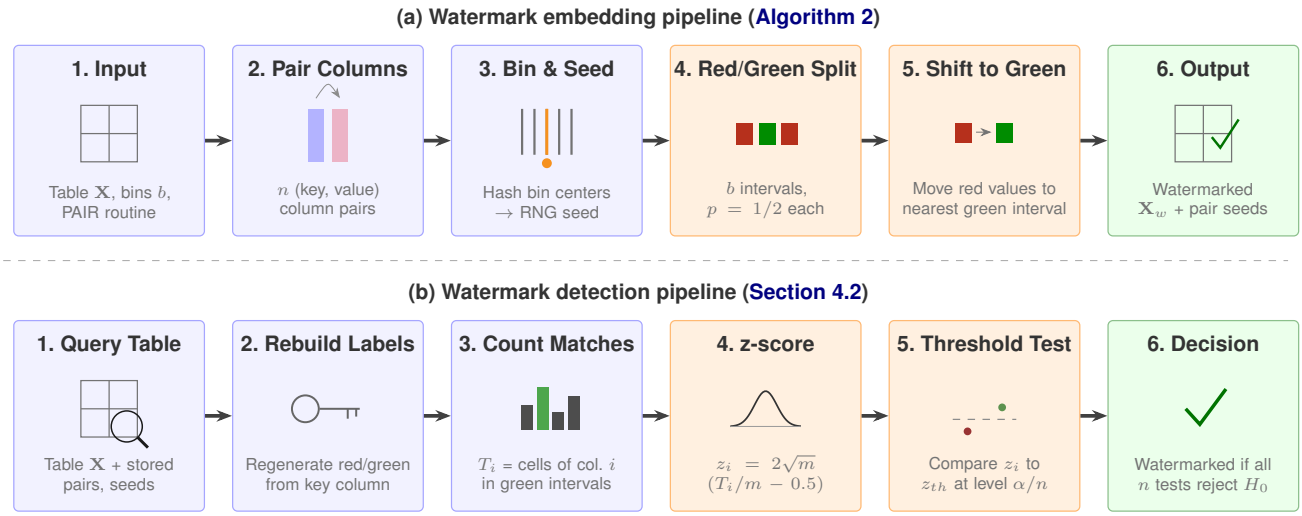

Figure 1: **Watermark embedding and detection pipelines**.

1.1 OUR CONTRIBUTIONS

Theoretical framework. We present a comprehensive theoretical study of a simple watermarking algorithm for continuous tabular data with several desirable properties. (1) **Fidelity:** We demonstrate that data that is watermarked according to our algorithm is similar to the input data. Moreover, we can control how close they are. (2) **Detection:** We show that there is a simple algorithm to detect our watermark. (3) **Robustness:** We show that our watermarking algorithm is robust to arbitrary additive noise, and modifications due to downstream tasks such as truncation and feature selection. (4) **Decoding:** We present the first study of algorithms capable of decoding the query function used to determine whether data has been watermarked.

Empirical analysis. Our second contribution is a systematic empirical study of several watermarking algorithms, including our proposed variant. Our results support theoretical findings and show that our method matches He et al. [2024] in fidelity and utility, but is more robust to additive noise, while theirs is stronger against table-level attacks. Compared to Zheng et al. [2024], our approach offers higher fidelity and cell-level robustness, but lower downstream utility. Against Fang et al. [2025], our method is more robust to table-level attacks, though with reduced utility.

2 RELATED WORK

Watermarking relational databases has been extensively studied in the literature [Agrawal and Kiernan, 2002, Sion et al., 2003, Shehab et al., 2007, Lin et al., 2021, Li et al., 2023, Kamran et al., 2013, Hwang et al., 2020, Hu et al., 2019], with methods classified as either non-reversible (embedding bits into data or statistics) [Agrawal and Kiernan, 2002, Hamadou et al., 2011a] or reversible (allowing recovery of original data) [Hu et al., 2019, Hwang et al., 2020]. These works mainly focus on minimizing changes to the mean and variance of the data, while ignoring how the watermarked dataset performs for downstream machine learning tasks [Kamran and Farooq, 2018b].

Recent advances in watermarking generative tabular data has been inspired from the watermarking techniques for large language models [Kamaruddin et al., 2018, Aaronson, 2023, Kuditipudi et al., 2024, Kirchenbauer et al., 2023], using either post-process or generation-time watermarks [Fang et al., 2025]. Generation-time methods, such as those in Zhu et al. [2025], show strong robustness but rely on access to the sampling process, which is often impractical.

Inspired by Kirchenbauer et al. [2023]'s idea for generating sequences from specific partitions in the probability space, [He et al., 2024, Zheng et al., 2024] embeds the watermark by forcing the values in generated columns to specific regions of the real number line. Particularly, WGTD [He et al., 2024] proposed a binning scheme with neighboring 'red' and 'green' intervals to ensure high data fidelity and robustness against additive noise attack. TabularMark [Zheng et al., 2024] focused on embedding the watermark only in the prediction target feature for both regression and classification task. Importantly, these algorithms are simple, allowing us to study them theoretically as our proposed algorithm derives from this class of watermarking algorithms. Additional details on related work is discussed in Appendix A.

3 NOTATION AND ALGORITHMS

In this section, we first state the notation used throughout. We also state the main algorithm that we study in our work and provide a brief overview of the algorithm.

Key A	Value B	Value A	Key B
K_1	...	V_1	...
K_2	...	V_2	...
K_3	...	V_3	...

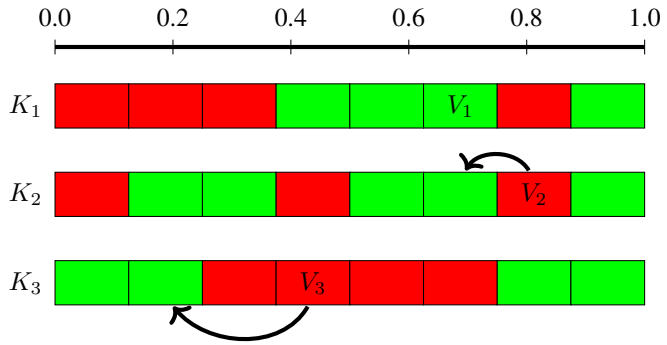

Figure 2: Illustrative example of Algorithm 2 on a tabular dataset with 3 rows and 4 columns. This structure corresponds to 2 pairs of $(key, value)$ columns. In the first row, the element V_1 is already in a 'green' interval. Meanwhile, the other elements, V_2 and V_3, have to be moved from 'red' interval to a nearby 'green' interval.

3.1 NOTATION

For $n \in \mathbb{N}^+$, we write $[n]$ to denote $\{1, \cdots, n\}$. For a matrix $\mathbf{X} \in \mathbb{R}^{m \times 2n}$, we denote ℓ_∞-norm of $\mathbf{X}$ as $\|\mathbf{X}\|_\infty = \max_{i \in [m], j \in [2n]} |\mathbf{X}_{i,j}|$. For an interval $g = [a, b]$, we denote the center of g as $\text{center}(g) = {}^{(a+b)}/_2$. For any $x \in \mathbb{R}$, $\lfloor x \rfloor$ is the largest integer that is less than or equal to x, and $x^\circ = x - \lfloor x \rfloor$ denotes its fractional part. For a tabular dataset $\mathbf{X} \in [0, 1]^{m \times 2n}$, $\mathbf{X}_i$ is the i-th column of $\mathbf{X}$. We denote $\mathbf{X}_w$ as the watermarked table. Define $\text{append}(x, Y)$ as the function that appends x to an ordered list Y.

3.2 A SIMPLE ALGORITHM FOR WATERMARKING.

Algorithm 1 PAIR

Input: Probability vector $p \in \mathbb{R}^{2n}$, seed s.

Initialize RNG with seed s.
Initialize Pairs $= []$, $S \leftarrow [2n]$.
Create a random permutation σ of $[2n]$ using p.
for $i \in [n]$ **do**
    Pairs $= \text{append}([(\sigma_{2i-1}, \sigma_{2i})], \text{Pairs})$.

Output: Pairs.

In this work, we study Algorithm 2 for watermarking tabular data. At a high level, Algorithm 2 partitions the feature space into pairs of $(key, value)$ columns using a subroutine PAIR (see Figure 2). There are several ways to implement PAIR, we give one such in Algorithm 1. We finely divide the range of elements in each key column into bins of size ${}^1/_b$ to form b consecutive intervals. The center of the intervals for each key column is used to compute a hash, which becomes the seed for a random number generator (RNG). The corresponding $value$ column is then partitioned into intervals of size ${}^1/_b$, and the RNG is used to label these intervals as red or green with equal probability. The watermark is embedded into a $value$ column of $\mathbf{X}$ by looking

Algorithm 2 Pairwise Tabular Watermarking

Input: Tabular dataset $\mathbf{X} \in \mathbb{R}^{m \times 2n}$, number of bins $b \in \mathbb{N}^+$, pairing subroutine PAIR (e.g., Algorithm 1), hash function HASH.

Construct n $(key, value)$ pairs using PAIR.
Divide the key columns into bins of width ${}^1/_b$ to form intervals denoted $\{I_j\}_{j \in [b]} \mid I_j = [{}^{j-1}/_b, {}^j/_b]$.
for each key column, **k do**
    Initialize $c = []$. For each $i \in [m]$, $c \leftarrow \text{append}(\text{center}(I_j), c)$ where $\mathbf{k}_i^\circ \in I_j$.
    $seed \leftarrow \text{HASH}(\text{c})$. Set RNG with $seed$.
    $c \sim \text{Unif}(\{\text{red}, \text{green}\})^b$, $G \leftarrow \{i : c_i = \text{green}\}$.
    for each $x \in value$ column **do**
        $G^* \leftarrow \arg\min_{g \in G} |x^\circ - \text{center}(I_g)|$.
        if $x^\circ \notin \cup_{g \in G} I_g$ **then**
            $x \leftarrow \lfloor x \rfloor + x_w$ with $x_w \sim \text{Unif}\,(\cup_{g \in G^*} I_g)$.

Output: Watermarked dataset $\mathbf{X}_w$, list of $(key, value)$ columns.

at the fractional part of the input data in the column, and choosing a green interval that is closest to this fractional part. This process is repeated until all the $value$ columns are watermarked. Note, in practice, it is not necessary to return entire the list of $(key, value)$ pairs. It suffices to store the parameters of PAIR (e.g., seed) that is used to generate the specific instance of $(key, value)$ pairs that are created while running Algorithm 2. In what follows, we analyze several key properties of Algorithm 2. We start by stating a key assumption in our analysis.

Assumption 3.1. *We assume that (i) the red and green intervals are labeled independently with probability ${}^1/_2$, and (ii) different data points/samples (rows) are independent. We note that the independence across samples is a standard assumption made for simplifying the statistical analysis in machine learning literature, while the assumption that red/green labels are chosen independently is a feature of the*

algorithm. Importantly, we do not make any assumption on the independence of columns.

Notes on the analysis of Algorithm 2. While a real-world tabular dataset may contain many categorical features, embedding the watermark in these features may cause significant changes in the meaning for the entire row. Given a dataset $\mathbf{X}$ with mixed data types, one may construct a subset of the dataset consisting of only continuous data. This subset can then be watermarked and reincorporated within the original tabular data. Given that the watermarking due to Algorithm 2 is only on the fractional part of the data, throughout our theoretical analyses, we assume that the $\mathbf{X} \in [0,1]^{m\times 2n}$. Finally, to avoid trivial situations, we assume that at least one of $\min(m,n) > 1$.

4 PROPERTIES OF PAIRWISE TABULAR WATERMARKING ALGORITHM

In this section, we show that the watermarking algorithm (Algorithm 2) satisfies several desirable properties. Specifically, in Section 4.1, we show that the watermarked data is close to the original data. In Section 4.2, we show that there is a principled method for detecting (with high probability) whether a given tabular data is watermarked according to Algorithm 2. In Section 4.3, we show that our watermarking algorithm is robust to many types of noise, as well as common downstream tasks such as feature selection and truncation. Finally, in Section 4.4, we provide explicit sample/query complexity upper and lower bounds for the task of decoding the watermarking scheme.

4.1 FIDELITY

A desirable property of a watermarking algorithm is that the watermark does not change the data significantly, or in other words, maintains a 'high fidelity' with the original data. We show that Algorithm 2 satisfies such a property in the sense that the watermarked data is close to the true data (with respect to entry-wise ℓ_∞-distance) with high probability over the choice of red/green labeling of the intervals and choice of the fractional part of the data for watermarking.

Theorem 4.1 (Fidelity). *Let $\mathbf{X} \in [0,1]^{m\times 2n}$ be a tabular dataset, and $\mathbf{X}_w$ is its watermarked version from Algorithm 2. With probability at least $1-\delta$, for $\delta \in (0,1)$, the entry-wise ℓ_∞-distance between $\mathbf{X}$ and $\mathbf{X}_w$ is bounded above by:*

$$\|\mathbf{X} - \mathbf{X}_w\|_\infty \leq \min\left\{\frac{1}{2b}\left(\log_2\left(\frac{mn}{\delta}\right)+1\right), 1\right\} \quad (1)$$

Observe that as the number of bins increase, the watermarked data is guaranteed to be close to the input data with high probability. Intuitively, this is because (1) a large distance between an element x of the input tabular data and the corresponding element x_w in the watermarked data implies that there are many red-labeled intervals between x and x_w, and (2) it is very unlikely that a large number of contiguous intervals are assigned red (since red and green are chosen with equal probability). Furthermore, Theorem 4.1 yields a corollary that upper bounds the Wasserstein distance between the empirical distributions of $\mathbf{X}$ and $\mathbf{X}_w$. Together, these results show that $\mathbf{X}_w$ is, in expectation, close to $\mathbf{X}$, so downstream tasks on $\mathbf{X}_w$ incur only $O(1/b)$ additional error with high probability.

Corollary 4.2 (Wasserstein distance). *Let $F_{\mathbf{X}} = \sum_{j=1}^{m} \frac{1}{m}\delta_{\mathbf{X}[j,:]}$ be the empirical distribution built on $\mathbf{X} \in [0,1]^{m\times 2n}$, and $F_{\mathbf{X}_w} = \sum_{j=1}^{m} \frac{1}{m}\delta_{\mathbf{X}_w[j,:]}$ be the empirical distribution built on $\mathbf{X}_w$. With probability at least $(1-\delta) \in (0,1)$, the k-Wasserstein distance is upper bounded by*

$$\mathcal{W}_k(F_{\mathbf{X}}, F_{\mathbf{X}_w}) \leq \frac{\sqrt{n/2}}{b}\left(\log_2\left(\frac{mn}{\delta}\right)+1\right). \quad (2)$$

4.2 DETECTION

Suppose that the data provider watermarked some input tabular data and allowed others to use it. At a later time, an external party claims that the data being used by them was supplied by the data provider. To check the veracity of such claims, we devise a procedure that can 'detect' whether a given tabular data has been watermarked according to Algorithm 2.

Recall that given access to the $(key, value)$ columns, the data provider can use the key columns to reconstruct the red/green intervals for each $value$ column. If any data falls in red bins, it is likely not watermarked. However, due to noise or other computational factors, some elements in $\mathbf{X}_w$ may fall outside green bins. To address this, we formulate detection as a hypothesis test and characterize the null distribution, enabling robust detection against minor data modifications.

Hypothesis test

H_0 : Dataset X is not watermarked

$H_{0,i}$: The i-th $value$ column is not watermarked

H_1 : Dataset X is watermarked

That is, when the null hypothesis holds, all of the individual null hypotheses for the i-th value column must hold simultaneously. Thus, the data provider who wants to detect the watermark for a dataset $\mathbf{X}$ would need to perform the hypothesis test for each $value$ column individually. If the goal is to reject the null hypothesis H_0 when the p-value is

less than a predetermined significant threshold α (typically 0.05 to represent 5% risk of incorrectly rejecting the null hypothesis), then the data provider would check if the p-value for each individual null hypothesis $H_{0,i}$ is lower than α/n (after accounting for the family-wise error rate using Bonferroni correction for testing n hypotheses [Bonferroni, 1936]).

An important step towards performing such a hypothesis test is to understand the null distribution, i.e., a baseline on how the input data is distributed in the red/green intervals without performing any watermarking (lines 7-11 in Algorithm 2). The following lemma gives us such a baseline.

Lemma 4.3. *Let F be the probability distribution of the data x which is to be watermarked (a single element of a value column of the input table) with support in $[0, 1]$. Define $\mathcal{G}$ as the union of intervals which are labeled green. Then, we have $\Pr[x \in \mathcal{G}] = 1/2$ where the probability is over $x \sim F$ and over the choice of the labels of the intervals (red/green).*

Notably, Lemma 4.3 states that irrespective of the distribution from which the elements of the input data are sampled from, the probability that they fall in a green bin is $1/2$ (over random choice of red and green bins as well as the data distribution). Therefore, if T_i denotes the number of elements in the i-th $value$ column that falls into a green interval, then, under the individual null hypothesis $H_{0,i}$, we know that $T_i \sim B(m, 1/2)$. Using the Central Limit Theorem, we have $2\sqrt{m}\left(\frac{T_i}{m} - \frac{1}{2}\right) \to \mathcal{N}(0, 1)$ as $m \to \infty$. Hence, we use the statistic for a one-proportion z-test for testing each individual null hypothesis, which is

$$z_i = 2\sqrt{m}\left(\frac{T_i}{m} - \frac{1}{2}\right). \tag{3}$$

For a given pair of $(key, value)$ columns, one can calculate the corresponding z-score by counting the number of elements in $value$ column that are in green intervals. Given a significance level α, z_{th}, the threshold for the z-score, can be computed and used to reject each null hypothesis (at significance level α/n) using the quantile function of the standard normal distribution.

4.3 ROBUSTNESS

In practice, data that has been watermarked can get perturbed due to several reasons. These could be due to addition of noise to the data for data processing, or because the user decides to select only a few features from the large tabular data that is relevant to their application. We call changes of this form 'oblivious attacks' since they can affect the statistical watermark embedded in the data, but are not done with the intention of changing the watermark. On the other hand, a malicious user or an 'adversary' can try to modify the data while still maintaining the watermark. Changes of this type are called 'adversarial attacks'. In this section, we focus on robustness against oblivious attacks.

The main robustness result that we prove in this study is a bound on the number of entries of the data that one needs to corrupt in order to break the watermark. Interestingly, we have both a distribution dependent as well as a distribution independent bound for the case of additive noise, which we present below.

Theorem (G.1, Informal)**.** *Fix a column in the watermarked dataset. Let n_a be the number of cells in this column that one can inject (additive) noise into. Suppose that the noise injected into these cells is drawn i.i.d. from some distribution. Then, there is a number $\gamma \in [0, 1]$ depending on the noise distribution such that $n_a \geq \frac{m - z_{\text{th}}\sqrt{m}}{2-\gamma}$, for the expected z-score to be less than z_{th} (i.e., to remove the watermark).*

A direct consequence is that one needs to perturb at least $(m - z_{\text{th}}\sqrt{m})/2$ cells (Remark G.2) to ensure that the watermark is 'broken', no matter what the noise distribution is. Intuitively, the entity injecting noise does not know what intervals are labeled red/green. Therefore, when the noise is injected 'blindly', the perturbed elements can not only fall into a red interval but also into a green interval. Thus, to remove the watermark, one needs to add noise into sufficiently many cells so that enough cells fall into a red interval.

We can use Theorem G.1 to get explicit bounds for specific distributions such as uniform or Gaussian distribution (see Appendix G for details). Similarly, we also prove that our watermarking scheme is robust under other types of tasks that may be performed in data science applications such as feature selection or truncation of elements (to save memory). Proofs of robustness for these tasks is given in Appendix G.

Truncation. Often, due to compute resources, $\mathbf{X}_w$ can be truncated. We define truncation of $x \in \mathbf{X}_w$ as

$$x_{\text{tr}} = \text{truncate}(x, p) = \frac{\lfloor 10^p \cdot x \rfloor}{10^p}. \tag{4}$$

In Theorem 4.4, we show that the probability that the watermarked value under truncation falls in a red bin increases with b (and inversely to $1/b$). This result presents an interesting trade-off between smaller bin width for higher fidelity (see Theorem 4.1) and bigger bin width for better robustness, which has not been studied in prior work on watermarking tabular data.

Theorem 4.4. *Let p be the precision of a given watermarked element $x \in I_j = [j-1/b, j/b]$ truncated using Equation (4). The probability that the truncated element x_{tr} falls out of its original green interval is*

$$\Pr[x_{\text{tr}} \notin I_j] = \frac{(b-1)^{10^p} + b^{10^p - 1}(c \cdot b - j + 1)}{b^{10^p}}$$

where $c \in \{0, 0.01, \cdots, 0.99\}$ are the left grid points of I_j intervals.

Feature selection. Feature selection is column sampling procedure which is often of relevance in data science [Cherepanova et al., 2023, Covert et al., 2023, Xu et al., 2023]. Since the watermarks are done on pairs of columns of the original table, to detect watermark after feature selection, some of the $key, value$ pairs would be required to be retained. As such, we study the problem of robustness to feature selection. Specifically, given the importance of each feature columns are known a priori, Lemma G.6 demonstrates that if PAIR uses feature importance to construct pairs, then the probability that a pair is retained post feature selection is greater than the probability of preserving pairs constructed uniformly at random.

Lemma (G.6, Informal)**.** *The probability of sampling a* $(key, value)$ *pair from the top* k *important columns according to* p*, is greater when sampling without replacement according to* p *than when sampling without replacement uniformly.*

Intuitively Lemma G.6 holds since the problem of pairing boils down to how the permutation of $[2n]$ is constructed (using sampling without replacement). For additional results on feature selection, see Appendix G.2.

4.4 DECODING WATERMARK

For any watermarking scheme, it is important that it is secure against spoofing. Specifically, we do not want an adversary to decode the watermarking scheme so that they can make changes to it without the data provider's knowledge. In this regard, it is important to understand how many queries are required by an adversary to learn a watermarking scheme. In turn, this would enable us to bound the total number of queries we allow a single user to make to the watermarked tabular data before we run the risk of giving out the exact parameters of our algorithms. Although such a result has not been obtained in prior work on watermarking generative tabular data, we believe it would showcase the power of any watermarking scheme that is similar to Algorithm 2 (such as the one by He et al. [2024]).

The decoding setting that we consider in this study is where a user (possibly an adversary) is able to ask or 'query' the data provider whether the tabular data used by them is watermarked. An adversary can break a large table into multiple chunks of smaller tables that is then queried. Based on the answers (whether or not the table is watermarked), the adversary can then learn the 'query function' (which depends on specific parameters used by the data provider) which says whether or not the data is watermarked. Once this query function is learned by the adversary, they can use it as an oracle/labeling mechanism to see if some spoofing technique is able to generate data that the data provider thinks is watermarked. Without access to such a query function, the adversary would potentially need to query the data provider a very large number of times because every query to the data provider potentially takes time/cost which makes running spoofing (e.g., brute force) algorithms infeasible.

We demonstrate that the number of queries to the tabular data an adversary would require to learn the watermarking algorithm presented in our work (Algorithm 2) is $\widetilde{O}(mnb)$, where b is the number of bins used to embed the watermark in Algorithm 2, and the adversary uses a query table of size $m \times n$. We informally state this result below, and defer the formal statement and its proof to Appendix H.

Theorem (H.2, Informal)**.** *A query function for the watermarking scheme in Algorithm 2 can be learned up to error* $\epsilon \in (0,1)$ *with probability at least* $1-\delta \in (0,1)$ *by making* $O\left(\frac{mnb\log(mnb)\log(1/\epsilon)+\log(1/\delta)}{\epsilon}\right)$ *queries to the detector using tables of size* $m \times n$*, given the number of bins* b*.*

In practice, the data provider using Algorithm 2 to watermark their data can artificially limit the total number of queries by an order of magnitudes less than the query complexity of Theorem H.2 to avoid divulging the watermarking scheme. We also observe that the query complexity of learning Algorithm 2 is exactly same as the query complexity of learning the algorithm of He et al. [2024]. This is an artifact of how we derive our bounds (via VC dimension). Beyond table like queries, we also study the case when queries are of the form of rows. For the formal theorem statement and an analysis for the row-query case, see Appendix H.3.

5 EXPERIMENTS

In this section, we empirically evaluate the our watermarking algorithm on synthetic and real datasets to verify our theoretical claims. Formally, we answer the following questions: (**Q1**) can our watermarking method ensure high data fidelity and downstream utility? (Section 4.1); (**Q2**) is our watermarking method detectable for datasets of various size and distribution? (Section 4.2); (**Q3**) is our watermarking method robust against oblivious and adversarial attacks? (Section 4.3).

Datasets. For synthetic data experiments, we generate datasets of various size using the standard Gaussian distribution. For experiments on real datasets, we evaluate on Boston Housing and California Housing Price for regression task, and Adult and Wilt for classification task. For each dataset, we generate corresponding synthetic datasets using Gaussian Copula [Masarotto and Varin, 2012]. We use Random Forest as the downstream machine learning algorithm for all datasets. For detection, we use a p-value threshold of 0.05. For additional detail, see Appendix B.

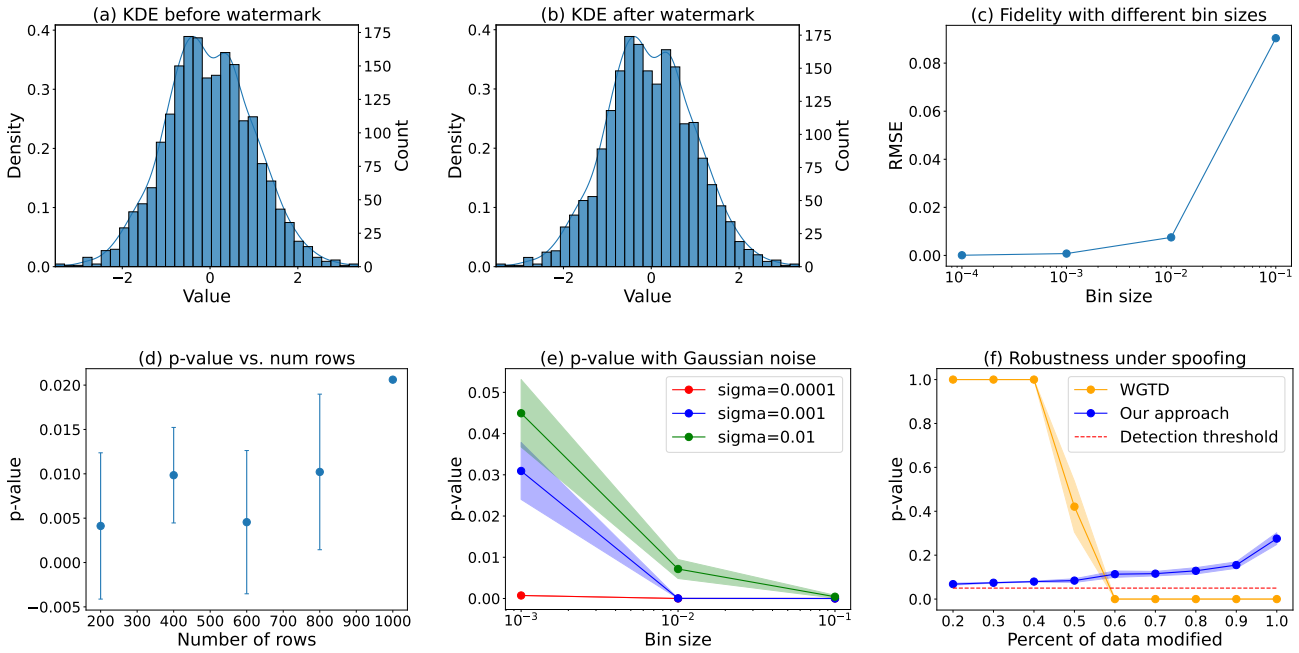

Figure 3: (a, b) KDE plot of the Gaussian data before and after watermarking respectively. (c) Smaller bin sizes result in higher fidelity (lower RMSE). (d) p-value remain below 0.05 across datasets. (e) Smaller bin sizes are more susceptible to zero-mean Gaussian noise with standard deviation sigma. (f) At 60% of data processed by Algorithm 3 (see Appendix C), WGTD can be spoofed while Algorithm 2 remains robust.

	Original	**Watermarked**	**MLE Gap**
Boston	0.330	0.330 ± 0.000	0.000
California	84512.544	84586.003 ± 74.140	74.459
Adult	0.741	0.741 ± 0.000	0.000
Wilt	0.496	0.496 ± 0.000	0.000

Table 1: We report the downstream utility as RMSE for regression task or ROC-AUC score for classification task. The MLE gap is the difference between the downstream performance of watermarked data and real data. For comparison to other baselines see Table 9. While MLE gap seems large for California, the relative gap is of the order of $O(10^{-3})$.

5.1 FIDELITY

To verify (Q1), we first examine the effect of Algorithm 2 on the data distribution of synthetic datasets using KDE plots (Fig. 3a and 3b). We observe minimal difference between the data distribution before and after watermarking.

To compare downstream task utility, we focus on machine learning efficiency gap (MLE Gap) [Xu et al., 2019], which measures the differences in performance of models trained on synthetic data and original data. To compare across different choices of the bin size b in Algorithm 2, we compute root mean squared error (RMSE) for classification error. Particularly, we vary the bin size b between 10^{-4} and 10^{-1}. We observe that the bin size is directly proportional to RMSE, and so inversely proportional to utility (Figure 3c). This is expected as smaller bin size implies that the nearest green interval used for watermarking in Line 10 of Algorithm 2 is closer to the original data sample.

For real-world datasets, we report the RMSE for regression and ROC-AUC score for classification in Table 1 over 10 trials. We observe that the downstream utility of the watermarked dataset does not decrease significantly compared to the original dataset. Thus, we conclude that our Algorithm 2 maintain high data fidelity and downstream utility.

5.2 DETECTABILITY

To investigate (Q2), we measure the detectability of our watermarking scheme by measuring the z-score on the watermarked dataset. We generate synthetic datasets, each containing 50 columns and vary the number of rows from 200 to 1000 to examine the effect of number of samples on the p-value. We report the average p-value over 5 trials. We observe that irrespective of the dataset sizes, the p-value stays below the 0.05 threshold (Figure 3d), allowing us to always detect the watermark.

In Table 2, we report the p-value from the detection process described in Section 4.2 for the watermarked real-world datasets. We observe that the p-value for all watermarked

datasets are below the 0.05 threshold (a parameter for the detection algorithm). This result suggests that the watermarking algorithm in Algorithm 2 remains easily detectable across different types of synthetic and real-world datasets.

5.3 ROBUSTNESS

To examine (Q3), we examine the detectability of our watermarking scheme under some oblivious and adversarial attacks. For oblivious attacks, we consider table-level attacks (e.g., row shuffling, row deletion, feature selection) and row-level attacks (e.g., additive noise, value alteration, truncation). For adversarial attack, we consider a spoofing attack, where the adversary attempts to create a synthetic dataset that can fool the detector in Section 4.2 without knowledge of the parameters used in Algorithm 2.

Oblivious attacks. For the synthetic dataset, we first consider the additive noise attack on a dataset of size 2000×4. We simulate this attack by adding zero-mean Gaussian noise with the standard deviation varying from 10^{-3} to 10^{-1} and measure its effects on the p-value. We plot this over 5 trials in Figure 3e. We find that with a similar level of added noise, a smaller bin size results in a higher p-value. However, this does not adversely affect detectability.

To validate robustness against feature selection, we create two set of datasets of size 75×75 using Algorithm 1 with p as (1) uniform and (2) proportional to feature importance. For each dataset, we train a Random Forest classifier and use the calculated feature importance to drop the least important subsets of columns varying from 20% to 80%. We measure the percentage of column pairs preserved over 5 trials and report the statistics in Figure 4. We observe that more $(key, value)$ pairs under the feature importance is preserved compared to uniform pairings. For real-world datasets, we employ eight common operations (attacks) on tabular data, divided into two categories: (1) table-level attacks: row shuffling, row deletion, column deletion, feature selection according to feature importance; and (2) cell-level attacks: Gaussian noise addition, uniform noise addition, value alteration, and truncation. We report the mean and standard deviation of the p-values from the hypothesis tests in Table 2, computed over 10 trials. Across all datasets and attacks, the watermark remains detectable (i.e., p-value remains ≤ 0.05). We also plot the number of column pairs preserved in each dataset after some columns are dropped due to feature selection under the two pairing routines discussed in Figure 4. With increasing number of columns dropped during feature selection, feature importance pairing always retains more column pairs compared to uniformly random pairing.

Adversarial attack. To compare the robustness of our algorithm and the WGTD watermark [He et al., 2024] against spoofing attack, we use Algorithm 3 (see Appendix C for details). This spoofing algorithm is novel, and is designed to subtly alter a dataset by modifying the decimal points of its elements so that the dataset appears to be watermarked. For each element, Algorithm 3 identifies the closest fractional part from the watermarked dataset to the current element. This fractional part is then copied over to replace the fractional part of the current element. This ensures that the watermark is embedded in a way that is minimally invasive, preserving the fidelity, detectability, overall structure, and integrity of the original data while introducing a subtle, detectable pattern. The experimental results are in Figure 3f.

We observe that one can successfully spoof the WGTD after replacing about 60% of the synthetic dataset. That is, the spoofed dataset generated from Algorithm 3 will be classified as 'watermarked' by He et al. [2024]'s detector as the p-value drops below 0.05. On the other hand, our watermark approach (Algorithm 2) cannot be spoofed by using Algorithm 3 at all. Even after all data points in the dataset have been modified, the p-value from our detection hypothesis test still remains above 0.05. This shows that Algorithm 2 is more robust to simple spoofing attacks simply by leveraging the structure of the tabular dataset.

6 DISCUSSION AND FUTURE WORK

In this work, we propose a simple watermarking scheme for tabular numerical datasets. Our watermarking method partitions the feature space into pairs of $(key, value)$ columns using knowledge of the feature importance in the downstream task. We use the center of the bins from each key column to generate randomized red and green intervals and watermark the $value$ columns by promoting its value to fall in green intervals. Compared to prior work in watermarking generative tabular data, we provide a more rigorous theoretical analysis of different attacks to the watermark.

Several avenues for future work remain open. This includes adapting our watermarking algorithm for categorical data, or embedding watermarking in the probability spaces pre-generation (similar to methods for LLMs [Venugopal et al., 2011]). This is partly because it is non-trivial to understand how perturbing the space from which samples are drawn by the generative process affects categorical data generation. Cryptographic ideas [Kuditipudi et al., 2024, Christ et al., 2023] can be exploited to embed watermarks pre-generation, but understanding the induced distortions remains an open problem. Moreover, the theoretical analysis becomes much more complicated as the generation process is often non-linear. Finally, our analysis for decoding the watermark uses a specific embedding that allows us to apply VC dimensional bounds for learning the query function using neural networks with piecewise linear polynomial functions. It remains open if one can derive better query complexity bounds without resorting only to neural networks.

Dataset	Watermarked	Row Shuffled	Row Del.	Col. Del.	Importance	Gaussian	Uniform	Alteration	Truncation
Boston	0.007 ± 0.000	0.012 ± 0.003	0.011 ± 0.008	0.011 ± 0.007	0.007 ± 0.000	0.018 ± 0.008	0.018 ± 0.008	0.018 ± 0.009	0.002 ± 0.000
California	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.014 ± 0.008	0.018 ± 0.005	0.013 ± 0.007	0.000 ± 0.000
Adult	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.014 ± 0.008	0.010 ± 0.009	0.018 ± 0.005	0.000 ± 0.000
Wilt	0.000 ± 0.000	0.004 ± 0.008	0.010 ± 0.009	0.087 ± 0.029	0.000 ± 0.000	0.014 ± 0.008	0.017 ± 0.007	0.038 ± 0.040	0.021 ± 0.000

Table 2: p-values following several attacks on watermarked datasets.

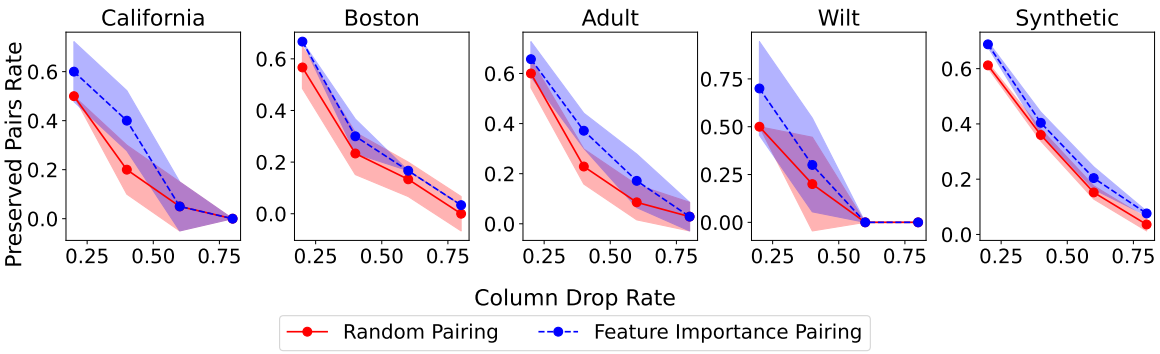

Figure 4: **Robustness under PAIR.** Each experiment is repeated 5 times, and the mean and standard deviation are reported. Across all datasets, feature importance pairing retains more column pairs than random pairing.

Acknowledgements

The authors would like to thank Niccolò Dalmasso and Tyler Chen for valuable discussions during the early stages of this project.

Disclaimer: This paper was prepared for informational purposes by the Artificial Intelligence Research group and the Global Technology Applied Research center of JPMorgan Chase & Co. and its affiliates ("JP Morgan") and is not a product of the Research Department of JP Morgan. JP Morgan makes no representation and warranty whatsoever and disclaims all liability, for the completeness, accuracy or reliability of the information contained herein. This document is not intended as investment research or investment advice, or a recommendation, offer or solicitation for the purchase or sale of any security, financial instrument, financial product or service, or to be used in any way for evaluating the merits of participating in any transaction, and shall not constitute a solicitation under any jurisdiction or to any person, if such solicitation under such jurisdiction or to such person would be unlawful.

References

S. Aaronson. 'Reform' AI Alignment with Scott Aaronson. AXRP - the AI X-risk Research Podcast, 2023. URL https://axrp.net/episode/2023/04/11/episode-20-reform-ai-alignment-scott-aaronson.html.

R. Agrawal and J. Kiernan. Watermarking relational databases. In *VLDB'02: Proceedings of the 28th International Conference on Very Large Databases*, pages 155–166. Elsevier, 2002.

B. An, M. Ding, T. Rabbani, A. Agrawal, Y. Xu, C. Deng, S. Zhu, A. Mohamed, Y. Wen, T. Goldstein, et al. Benchmarking the robustness of image watermarks. *arXiv preprint arXiv:2401.08573*, 2024.

S. A. Assefa, D. Dervovic, M. Mahfouz, R. E. Tillman, P. Reddy, and M. Veloso. Generating synthetic data in finance: opportunities, challenges and pitfalls. In *Proceedings of the First ACM International Conference on AI in Finance*, pages 1–8, 2020.

M. J. Atallah, V. Raskin, M. Crogan, C. Hempelmann, F. Kerschbaum, D. Mohamed, and S. Naik. Natural language watermarking: Design, analysis, and a proof-of-concept implementation. In *Information Hiding: 4th International Workshop, IH 2001 Pittsburgh, PA, USA, April 25–27, 2001 Proceedings 4*, pages 185–200. Springer, 2001.

P. L. Bartlett, N. Harvey, C. Liaw, and A. Mehrabian. Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks. *Journal of Machine Learning Research*, 20(63):1–17, 2019.

R. Bhatia and C. Davis. A better bound on the variance. *The american mathematical monthly*, 107(4):353–357, 2000.

C. Bonferroni. Teoria statistica delle classi e calcolo delle probabilita. *Pubblicazioni del R istituto superiore di scienze economiche e commericiali di firenze*, 8:3–62, 1936.

V. Braverman, R. Ostrovsky, and G. Vorsanger. Weighted sampling without replacement from data streams. *Information Processing Letters*, 115(12):923–926, 2015.

N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson, et al. Extracting training data from large language models. In *30th USENIX security symposium (USENIX Security 21)*, pages 2633–2650, 2021.

V. Cherepanova, R. Levin, G. Somepalli, J. Geiping, C. B. Bruss, A. G. Wilson, T. Goldstein, and M. Goldblum. A performance-driven benchmark for feature selection in tabular deep learning. *Advances in Neural Information Processing Systems*, 36:41956–41979, 2023.

M. Christ, S. Gunn, and O. Zamir. Undetectable watermarks for language models. *arXiv preprint arXiv:2306.09194*, 2023.

V. Chvátal. Mastermind. *Combinatorica*, 3:325–329, 1983. doi: 10.1007/BF02579188. URL https://doi.org/10.1007/BF02579188.

I. C. Covert, W. Qiu, M. Lu, N. Y. Kim, N. J. White, and S.-I. Lee. Learning to maximize mutual information for dynamic feature selection. In *International Conference on Machine Learning*, pages 6424–6447. PMLR, 2023.

P. S. Efraimidis and P. G. Spirakis. Weighted random sampling with a reservoir. *Information processing letters*, 97 (5):181–185, 2006.

L. Fang, A. Liu, H. P. Zou, H. Zhang, and P. S. Yu. Rintaw: A robust invisible watermark for tabular generative models. In *The 1st Workshop on GenAI Watermarking*, 2025.

R. A. Fisher and F. Yates. *Statistical tables for biological, agricultural and medical research*. Hafner Publishing Company, 1953.

F. Fui-Hoon Nah, R. Zheng, J. Cai, K. Siau, and L. Chen. Generative ai and chatgpt: Applications, challenges, and ai-human collaboration, 2023.

A. Gonzales, G. Guruswamy, and S. R. Smith. Synthetic data in health care: A narrative review. *PLOS Digital Health*, 2(1):e0000082, 2023.

A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, et al. The llama 3 herd of models, 2024.

A. Hamadou, X. Sun, S. A. Shah, and L. Gao. A weight-based semi-fragile watermarking scheme for integrity verification of relational data. *International Journal of Digital Content Technology and Its Applications*, 5:148–157, 2011a. URL https://api.semanticscholar.org/CorpusID:58278938.

A. Hamadou, X. Sun, S. A. Shah, and L. Gao. A weight-based semi-fragile watermarking scheme for integrity verification of relational data. *International Journal of Digital Content Technology and Its Applications*, 5:148–157, 2011b. URL https://api.semanticscholar.org/CorpusID:58278938.

H. He, P. Yu, J. Ren, Y. N. Wu, and G. Cheng. Watermarking generative tabular data, 2024. URL https://arxiv.org/abs/2405.14018.

D. Hu, D. Zhao, and S. Zheng. A new robust approach for reversible database watermarking with distortion control. *IEEE Transactions on Knowledge and Data Engineering*, 31(6):1024–1037, 2019. doi: 10.1109/TKDE.2018.2851517.

M.-S. Hwang, M.-R. Xie, and C.-C. Wu. A reversible hiding technique using lsb matching for relational databases. *Informatica*, 31(3):481–497, Jan. 2020. ISSN 0868-4952. doi: 10.15388/20-INFOR411. URL https://doi.org/10.15388/20-INFOR411.

S. R. Jammalamadaka and A. Sengupta. *Topics in circular statistics*, volume 5. world scientific, 2001.

A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.

N. Jovanović, R. Staab, and M. Vechev. Watermark stealing in large language models, 2024. URL https://arxiv.org/abs/2402.19361.

N. S. Kamaruddin, A. Kamsin, L. Y. Por, and H. Rahman. A review of text watermarking: theory, methods, and applications. *IEEE Access*, 2018.

M. Kamran and M. Farooq. A comprehensive survey of watermarking relational databases research. *arXiv preprint arXiv:1801.08271*, 2018a.

M. Kamran and M. Farooq. A comprehensive survey of watermarking relational databases research, 2018b. URL https://arxiv.org/abs/1801.08271.

M. Kamran, S. Suhail, and M. Farooq. A robust, distortion minimizing technique for watermarking relational databases using once-for-all usability constraints. *IEEE Transactions on Knowledge and Data Engineering*, 25 (12):2694–2707, 2013. doi: 10.1109/TKDE.2012.227.

S. Katzenbeisser and F. Petitcolas. Digital watermarking. *Artech House, London*, 2:2, 2000.

J. Kirchenbauer, J. Geiping, Y. Wen, J. Katz, I. Miers, and T. Goldstein. A watermark for large language models. *arXiv preprint arXiv:2301.10226*, 2023.

D. E. Knuth. The computer as master mind. *Journal of Recreational Mathematics*, 9(1):1–6, 1976.

R. Kuditipudi, J. Thickstun, T. Hashimoto, and P. Liang. Robust distortion-free watermarks for language models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=FpaCL1MO2C.

G. Kurz, I. Gilitschenski, and U. D. Hanebeck. Efficient evaluation of the probability density function of a wrapped normal distribution. In *2014 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, pages 1–5. IEEE, 2014.

W. Li, N. Li, J. Yan, Z. Zhang, P. Yu, and G. Long. Secure and high-quality watermarking algorithms for relational database based on semantic. *IEEE Transactions on Knowledge and Data Engineering*, 35(7):7440–7456, 2023. doi: 10.1109/TKDE.2022.3194191.

C.-C. Lin, T.-S. Nguyen, and C.-C. Chang. Lrw-crdb: Lossless robust watermarking scheme for categorical relational databases. *Symmetry*, 13(11), 2021. ISSN 2073-8994. doi: 10.3390/sym13112191. URL https://www.mdpi.com/2073-8994/13/11/2191.

K. V. Mardia and P. E. Jupp. *Directional statistics*. John Wiley & Sons, 2009.

G. Masarotto and C. Varin. Gaussian copula marginal regression. *Electronic Journal of Statistics*, 6:1517–1549, 2012. URL https://api.semanticscholar.org/CorpusID:53962593.

R. M. Monarch. *Human-in-the-Loop Machine Learning: Active learning and annotation for human-centered AI*. Simon and Schuster, 2021.

OpenAI. Chatgpt: Optimizing language models for dialogue, 2022. URL https://openai.com/blog/chatgpt.

N. Patki, R. Wedge, and K. Veeramachaneni. The synthetic data vault. In *2016 IEEE international conference on data science and advanced analytics (DSAA)*, pages 399–410. IEEE, 2016.

V. K. Potluru, D. Borrajo, A. Coletta, N. Dalmasso, Y. El-Laham, E. Fons, M. Ghassemi, S. Gopalakrishnan, V. Gosai, E. Kreačić, et al. Synthetic data applications in finance. *arXiv preprint arXiv:2401.00081*, 2023.

S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

M. Shehab, E. Bertino, and A. Ghafoor. Watermarking relational databases using optimization-based techniques. *IEEE transactions on Knowledge and Data Engineering*, 20(1):116–129, 2007.

I. Shumailov, Z. Shumaylov, Y. Zhao, N. Papernot, R. Anderson, and Y. Gal. Ai models collapse when trained on recursively generated data. *Nature*, 631(8022):755–759, 2024.

R. Sion, M. Atallah, and S. Prabhakar. Rights protection for relational data. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, SIGMOD '03, page 98–109, New York, NY, USA, 2003. Association for Computing Machinery. ISBN 158113634X. doi: 10.1145/872757.872772. URL https://doi.org/10.1145/872757.872772.

T. Stadler, B. Oprisanu, and C. Troncoso. Synthetic data–anonymisation groundhog day. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 1451–1468, 2022.

G. Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love, P. Tafti, L. Hussenot, P. G. Sessa, A. Chowdhery, A. Roberts, A. Barua, A. Botev, A. Castro-Ros, A. Slone, A. Héliou, A. Tacchetti, A. Bulanova, A. Paterson, B. Tsai, B. Shahriari, C. L. Lan, C. A. Choquette-Choo, C. Crepy, D. Cer, D. Ippolito, D. Reid, E. Buchatskaya, E. Ni, E. Noland, G. Yan, G. Tucker, G.-C. Muraru, G. Rozhdestvenskiy, H. Michalewski, I. Tenney, I. Grishchenko, J. Austin, J. Keeling, J. Labanowski, J.-B. Lespiau, J. Stanway, J. Brennan, J. Chen, J. Ferret, J. Chiu, J. Mao-Jones, K. Lee, K. Yu, K. Millican, L. L. Sjoesund, L. Lee, L. Dixon, M. Reid, M. Mikuła, M. Wirth, M. Sharman, N. Chinaev, N. Thain, O. Bachem, O. Chang, O. Wahltinez, P. Bailey, P. Michel, P. Yotov, R. Chaabouni, R. Comanescu, R. Jana, R. Anil, R. McIlroy, R. Liu, R. Mullins, S. L. Smith, S. Borgeaud, S. Girgin, S. Douglas, S. Pandya, S. Shakeri, S. De, T. Klimenko, T. Hennigan, V. Feinberg, W. Stokowiec, Y. hui Chen, Z. Ahmed, Z. Gong, T. Warkentin, L. Peran, M. Giang, C. Farabet, O. Vinyals, J. Dean, K. Kavukcuoglu, D. Hassabis, Z. Ghahramani, D. Eck, J. Barral, F. Pereira, E. Collins, A. Joulin, N. Fiedel, E. Senter, A. Andreev, and K. Kenealy. Gemma: Open models based on gemini research and technology, 2024. URL https://arxiv.org/abs/2403.08295.

A. Venugopal, J. Uszkoreit, D. Talbot, F. J. Och, and J. Ganitkevitch. Watermarking the outputs of structured prediction with an application in statistical machine translation. In *Proceedings of Empirical Methods for Natural Language Processing*, 2011.

T. Vieira. Algorithms for sampling without replacement. https://timvieira.github.io/blog/post/2019/09/16/algorithms-for-sampling-without-replacement/, 2019. Accessed: 2025-10-01.

X. Xiao, X. Sun, and M. Chen. Second-lsb-dependent robust watermarking for relational database. In *Third Interna-*

tional Symposium on Information Assurance and Security, pages 292–300, 2007. doi: 10.1109/IAS.2007.25.

L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni. Modeling tabular data using conditional gan. *Advances in neural information processing systems*, 32, 2019.

L. Xu, R. Wang, F. Nie, and X. Li. Efficient top-k feature selection using coordinate descent method. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 10594–10601, 2023.

S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Y. Zheng, H. Xia, J. Pang, J. Liu, K. Ren, L. Chu, Y. Cao, and L. Xiong. Tabularmark: Watermarking tabular datasets for machine learning, 2024. URL https://arxiv.org/abs/2406.14841.

C. Zhu, J. Tang, J. M. Galjaard, P.-Y. Chen, R. Birke, C. Bos, and L. Y. Chen. Tabwak: A watermark for tabular diffusion models. In *ICLR*, 2025.

Adaptive and Robust Watermark for Generative Tabular Data (Supplementary Material)

Dung Daniel Ngo[1, †] **Archan Ray**[2, †] **Akshay Seshadri**[2, †] **Daniel Scott**[1, *] **Saheed Obitayo**[1] **Niraj Kumar**[2]
Vamsi K. Potluru[1] **Marco Pistoia**[2] **Manuela Veloso**[1]

[1]AI Research, JPMorganChase, New York, NY 10017, USA
[2]Global Technology Applied Research, JPMorganChase, New York, NY 10001, USA

A ADDITIONAL RELATED WORK

	WGTD [He et al., 2024]	TabularMark [Zheng et al., 2024]	**Ours**
Data Type	Numerical	Numerical, Categorical	Numerical
Fidelity	$1/b$	$\min\left\{1/2b\left(\log\left(m/\delta\right)-1\right),1\right\}$[1]	$\min\left\{1/2b\left(\log\left(mn/\delta\right)-1\right),1\right\}$
Detection Test	χ^2-test	One Proportion z-test	One Proportion z-test with Bonferroni correction
Robustness	Additive noise	Additive Noise, Insertion, Deletion, Shuffle, Row Subset, Invertibility	Additive Noise, Truncation, Feature Selection
Spoofing	No	Yes	Yes
Learning	$\tilde{O}(mnb)$	$\tilde{O}(mnb)$	$\tilde{O}(mnb)$

Table 3: A comparison of the setting and results from [He et al., 2024, Zheng et al., 2024] and our technique for post-process watermarking generative tabular data. The fidelity measures the ℓ_∞-distance between the original and watermarked datasets. For spoofing attack, we list whether the watermark is robust against a simple spoofing procedure (Section 5.3). The statistical learning upper bounds represent the query complexity required by an adversary to learn the watermark scheme.

There exists a long line of research on watermarking relational databases [Agrawal and Kiernan, 2002, Sion et al., 2003, Shehab et al., 2007, Lin et al., 2021, Li et al., 2023, Kamran et al., 2013, Hwang et al., 2020, Hu et al., 2019]. Agrawal and Kiernan [2002] is the first work to tackle this problem setting, where the watermark was embedded in the least significant bit of some cells, i.e., setting them to be either 0 or 1 based on a hash value computed using primary and private keys. Subsequently, Xiao et al. [2007], Hamadou et al. [2011b] proposed an improved watermarking scheme by embedding multiple bits. Another approach is to embed the watermark into the data statistics. Notably, Sion et al. [2003] partitioned the rows into different subsets and embedded the watermark by modifying the subset-related statistics. This approach is then improved upon by Shehab et al. [2007] to protect against insertion and deletion attacks by optimizing the partitioning algorithm and using hash values based on primary and private keys. To minimize data distortion, the authors modeled the watermark as an optimization problem solved using genetic algorithms and pattern search methods. However, this approach is strictly limited by the requirement for data distribution and the reliance on primary keys for partitioning algorithms. For a comprehensive survey on prior work in watermarking databases, see Kamran and Farooq [2018a].

[†]Equal Contribution
[*]Work done while at AI Research, JPMorganChase
[1]Zheng et al. [2024] did not provide theoretical analysis of the fidelity. This bound comes from our analysis with a slightly different union bound event.

Both the prior works [He et al., 2024, Zheng et al., 2024] and our proposed approach maintain high data fidelity, measured by comparing the utility of using the watermarked dataset compared to the original unwatermarked dataset on downstream machine learning tasks. While all three approaches use a similar hypothesis testing procedure to detect the watermark, He et al. [2024]'s watermark is the easiest to detect. The additional cost of watermark detection in Zheng et al. [2024] and our schemes are due to the selection of the watermarked features. At detection time, the third-party need to identify either the target watermarked feature (for Zheng et al. [2024]'s watermark) or all $(key, value)$ feature pairs (for our approach). Compared to He et al. [2024], our approach is more robust to a simple spoofing attack (Section 5.3), where an adversary attempts to create harmful content with a target watermark embedded [Jovanović et al., 2024]. On the other hand, while Zheng et al. [2024] provided a comprehensive study of potential attacks against their watermark, their approach is less robust to the feature selection attack compared to ours. This difference stems from how the watermarked feature is chosen in their watermark. Since Zheng et al. [2024] mainly watermarks only the target feature (with possible extensions to multiple features), if an oblivious data scientist decides to drop this watermarked feature then the watermark is effectively erased. Meanwhile, we specifically design the $(key, value)$ PAIR routine to preserve the watermark even when a large number of columns is dropped. For security attacks on the watermark, we provide the first upper bound on query complexity required by an adversary to learn the watermark scheme. Under our analysis in Section 4.4, this upper bound is the same for all three tabular watermarking approaches.

Novelty. In our watermarking approach, we draw from the vast literature of red-green list watermarking technique for LLMs. However, our application of this idea to tabular data involves judiciously leveraging the tabular structure of the dataset in our PAIR routine (Algorithm 1), which has non-trivial implications for downstream tasks like feature selection (see Figure 4 and Appendix G.2). Furthermore, our technical contributions are a comprehensive theoretical study of watermarking continuous tabular data that was previously missing from prior work. From a theoretical standpoint, we improve upon the previous results on several fronts. For detection, Lemma 4.3 in our study holds for every distribution in the non-asymptotic regime, whereas the analogous result in WGTD only holds asymptotically. For robustness, TabularMark only prove results for additive noise attacks for particular distributions or do not account for the perturbed elements to fall back into a green bin (instead of a red bin). We, on the other hand, provide results for arbitrary distributions for additive noise attacks, and additionally analyze truncation and feature selection attacks, which are novel contributions. In addition, we study adversarial attacks to the watermark such as the spoofing attack in Figure 3f. Finally, we investigate in detail the decoding of our watermarking algorithm as well as the WGTD algorithm, which includes theoretical (upper) bounds on the query complexity. Our proof technique involves constructing a neural network to decode the watermarks (and obtaining bounds on its VC dimension), which to our knowledge is novel in the watermarking literature.

Compared to WGTD and TabularMark, our watermark embedding algorithm requires additional time to run the PAIR subroutine (Algorithm 1). Generating a permutation of the columns takes $O(n)$ time, and creating the pairs take $O(n)$ time in total. The additional memory requirement here is to keep tracks of the pairs, which is stored in a list of size n. For watermark detection, we need to check the number of elements in green list intervals for each (key-value) pair, which lead to a multiplicative factor of $O(n)$ in run time more than WGTD and TabularMark. We still need to keep track of the list of (key, value) columns used in watermark embedding at detection time. In simulations, the entire end-to-end process (watermarking embedding, detection, measuring the effect of all oblivious and adversarial attacks) takes less than a minute per run.

B ADDITIONAL EXPERIMENTAL DETAILS

For reproducibility, we set a specific random seed to ensure that the data deformation and model training effects are repeatable on a similar hardware and software stack. To eliminate the randomness of the results, the experimental outcomes are averaged over multiple runs from each watermarked training set.

For synthetic data generation using the generative method mentioned in the paper, we follow the default hyperparameters found in the SDV library [Patki et al., 2016]. We run the experiments on a machine type of g4dn.4xlarge consisting of 16 CPU, 64GB RAM, and 1 GPU. For fidelity and utility evaluation, default parameters of Random Forest classifier and regressor with a seed of 42 was used. The synthetic data generation, training and evaluation process typically finishes within 4 hours. Python 3.8 version was used to run the experiments.

B.1 DATASETS.

Boston Housing Prices. This dataset contains information collected by the U.S. Census Service concerning housing in the area of Boston, Massachusetts. It contains 506 entries each of which represent aggregated data about 14 features for homes. Each entry contains the following features such as number of rooms, per capita crime rate by town, pupil teach ratio by town, index accessibility to highways, and value of home of a representative home in the housing record. The attributes are a mixture of continuous and categorical data types. The dataset has a continuous target label indicating the Median value of owner-occupied homes in $1000's. Due to this, the dataset is used for the regression task to predict the median value of owner-occupied homes in Boston suburbs.

California Housing Prices. The dataset was collected from the 1990 U.S. Census and includes various socio-economic and geographical features that are believed to influence housing prices in a given California district[2]. It contains 20,640 records and 10 attributes, each of which represents data about homes in the district. Similar to the previous dataset, the attributes are a mixture of continuous and categorical data types. The dataset is used for regression task, with the target variable being the median house value.

Wilt. This dataset is a public dataset that is part of the UCI Machine Learning Repository[3]. It contains data from a remote sensing study that involved detecting diseased trees in satellite imagery. The data set consists of image segments generated by segmenting the pan-sharpened image. The segments contain spectral information from the Quickbird multispectral image bands and texture information from the panchromatic (Pan) image band. This dataset includes 4,889 records and 6 attributes. The attributes are a mixture of numerical and categorical data types. The data set has a binary target label, which indicates whether a tree is wilted or healthy. Therefore, the dataset has a classification task, which is to classify the tree samples as either diseased or healthy.

Adult. This is a public dataset that is part of the UCI Machine Learning Repository[4]. It contains census data from 1994 with demographic information for a subsection of the US population. This dataset includes 48842 records and 14 attributes. The attributes are a mixture of numerical and categorical data types. The label for each record is a binary variable indicating whether an individual's annual income exceeds $50,000. Thus, the learning task is that of binary classification.

B.2 ADDITIONAL FIDELITY EXPERIMENT

Dataset	California	Boston	Adult	Wilt	Gaussian	Uniform
Distance	7.456e-05 ±1.588e-07	9.235e-05 ±1.593e-05	6.900e-05 ±2.356e-07	2.511e-04 ±2.790e-06	9.784e-05 ±9.729e-07	4.768e-05 ±8.984e-07

Table 4: The Wasserstein-1 distance between the original dataset and the watermarked dataset. All experiments are repeated 5 times and the mean and standard deviation are reported.

First, we want to clarify that the fidelity results for California Housing dataset seems abnormal due to the distribution of the target variable 'median_house_value'. This label ranges from $\$14999 - \$500000$, and follows a right-skewed distribution with a notable spike at the far right end of the distribution. This peak is a common artifact in housing datasets where values are capped or clipped at a maximum threshold. Comparatively, the variance and MLE Gap in the fidelity experiment is of order of 10^{-4} w.r.t. the range. Hence, the result for this dataset is a direct result of intrinsic dataset characteristic.

We further examine the effect of our watermarking algorithm 2 on fidelity for both real-world and synthetic datasets. We generate two types of synthetic dataset: Gaussian dataset of size 10000×10 with samples drawn i.i.d. from the standard Gaussian distribution, and Uniform dataset of the same size with samples drawn i.i.d. from the $\mathsf{Unif}[0, 1]$ distribution. All datasets are watermarked using Algorithm 2 with 1000 bins.

In Table 4, we first flatten both the original dataset and the watermarked dataset, then calculate the entry-wise ℓ_1 distance between these flattened datasets. The experiments are repeated 5 times, and we report the mean and standard deviation of the results. Across all datasets, we observe small differences between the original and watermarked datasets, showing that our watermarking approach maintain high fidelity. Additionally, we evaluate the fidelity of the watermarking procedure

[2]Data available on the StatLib repository at https://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html
[3]Data available on the UCI platform at https://archive.ics.uci.edu/dataset/285/wilt
[4]Data available on the UCI platform at https://archive.ics.uci.edu/dataset/2/adult

under different tabular generators: TVAE and CTGAN [Xu et al., 2019] (Table 5) and downstream learners: XGBoost and Linear/Logistic Regression (Table 6). We observe that the performance of our watermark in these experiments stays consistent with our prior results in Section 5.

	Boston	MLE Gap	California	MLE Gap	Adult	MLE Gap	Wilt	MLE Gap
TVAE	0.086	0.000	91960.528	0.000	0.793	0.000	0.895	0.000
CTGAN	0.438	0.000	81883.920	0.000	0.899	0.000	0.617	0.002

Table 5: We report the downstream utility as either RMSE for regression task or ROC-AUC score for classification task. The MLE gap is the difference between the downstream performance of watermarked data and real data.

	Boston				California				Adult				Wilt			
	XGBoost	MLE Gap	Linear	MLE Gap	XGBoost	MLE Gap	Linear	MLE Gap	XGBoost	MLE Gap	Linear	MLE Gap	XGBoost	MLE Gap	Linear	MLE Gap
GaussianCopula	0.359	0.000	0.349	0.000	85382.301	158.94	80135.876	0.000	0.469	0.000	0.436	0.000	0.032	0.000	0.031	0.000
TVAE	0.098	0.000	0.098	0.000	81801.875	0.000	86054.062	0.000	0.310	0.000	0.351	0.000	0.096	0.000	0.091	0.000
CTGAN	0.466	0.000	0.434	0.000	85961.290	0.000	86822.823	0.000	0.361	0.000	0.393	0.000	0.325	0.000	0.308	0.000

Table 6: We report the downstream utility as either RMSE for regression task or ROC-AUC score for classification task. The MLE gap is the difference between the downstream performance of watermarked data and real data.

B.3 ADDITIONAL ROBUSTNESS EXPERIMENT

In this experiment, we investigate the p-value on the watermarked dataset after it has been modified by some adversary. For the synthetic dataset, we generate a dataset of size 10000×21 for a multi-class classification problem using the SCIKIT-LEARN package. Specifically, the samples in the features columns are drawn i.i.d. from a standard Gaussian distribution, and among them, there are 4 informative features, 4 redundant features (i.e., random linear combinations of the informative features), 4 duplicated features (i.e., drawn randomly from the informative and redundant features), and the rest are filled with random noise. The label column contains 5 different classes. Half of the columns are randomly chosen as the key columns, and the other half are $value$ columns. We choose to perform experiment on this synthetic dataset to have more control over the features parameters in the datasets.

In Table 7, we report the p-value of the watermarked dataset after some columns have been dropped randomly. The proportion of column drops varies between 0.2 and 0.8. We repeat this experiment 5 times and report both the mean and average values. We observe that the p-value for all configurations are below the 0.05 threshold, and we can conclude that the presence of the watermark can always be detected after some columns have been dropped.

Column Drop Proportion	0.2	0.3	0.4	0.5	0.6	0.7	0.8
p-value	0.017 ± 0.007	0.014 ± 0.008	0.013 ± 0.009	0.012 ± 0.008	0.010 ± 0.005	0.042 ± 0.018	0.030 ± 0.018

Table 7: The calculated p-value after some columns have been dropped from the watermarked dataset. All experiments are repeated 5 times and the mean and standard deviation are reported.

In the next experiment, we examine the p-value after the watermarked dataset has been 'attacked' by an adversary adding some zero-mean noise. Specifically, we consider 4 types of noise distributions: (i) Uniform: noise drawn i.i.d. from $\mathsf{Unif}[0, 1]$, (ii) Gaussian: noise drawn i.i.d. from $\mathcal{N}(0, 1)$, (iii) Laplace: noise drawn i.i.d. from a zero-mean Laplace distribution with scale 1, and (iv) Exponential: noise drawn i.i.d from a zero-mean Exponential distribution with rate $\lambda = 1$. We repeat this experiment 5 times and report the mean and standard deviation of the calculated p-value. We observe that across all attacks, the p-value is below the 0.05 threshold and we can successfully detect the watermark even after the watermarked dataset has been tampered with using additive noise.

B.4 COMPARISON TO PRIOR WORK

We investigate the performance of our watermarking technique on four different real-world datasets and compare it with prior watermarking algorithms as benchmark.

Noise Type	Uniform	Gaussian	Laplace	Exponential
p-value	0.013 ± 0.007	0.015 ± 0.007	0.015 ± 0.007	0.008 ± 0.006

Table 8: The calculated p-value after additive noise attack with different noise distributions. All experiments are repeated 5 times and the mean and standard deviation are reported.

Experimental Setup. We evaluate our watermark and the baseline approaches on four real-world datasets: **Boston Housing** and **California Housing Price** for regression task, as well as **Adult** and **Wilt** for classification task. For each dataset, we generate corresponding synthetic datasets using Gaussian Copula [Masarotto and Varin, 2012]. We use Random Forest as the downstream machine learning algorithm for all datasets.

Baseline approaches. We compare our performance to three other tabular watermarking techniques: WGTD [He et al., 2024], TabularMark [Zheng et al., 2024], and RINTAW [Fang et al., 2025]. As the source codes are unavailable, we implement these watermarking techniques using the authors' specifications. That is, for WGTD implementation, we set the number of 'green list' intervals to be $m = 500$, for a total of 1000 intervals between 0 and 1. For TabularMark implementation, we always embed the watermark into 10% of the label column, with number of unit domains $k = 500$, perturbation range $p = 25$. For classification task, we select a half of the possible categories to be 'green' instead of using the aforementioned unit domains and perturbation range. Finally, for RINTAW implementation, we use the default masking ratio of 0.

Experimental results. First, we study the fidelity of our watermarking approach on these datasets. Particularly, we are interested in measuring both downstream performance metric (RMSE for regression task and ROC-AUC score for classification task) and the Machine Learning Efficiency (MLE), i.e., the gap between the performance of watermarked data and the real test data. Each experiment is repeated 10 times and the mean and standard deviation for the target performance metric are reported. The result of this experiment is summarized in Table 9. Overall, RINTAW achieves the best downstream performance among all watermarking methods. However, our approach achieves the smallest MLE Gap in three datasets (Boston Housing, Adult, and Wilt), and the second smallest MLE gap in the remaining dataset (California Housing). That is, for both regression and classification, our watermark approach leads to minimal distortion compared to the real, unwatermarked dataset.

	Boston		California		Adult		Wilt	
	RMSE	MLE Gap	RMSE	MLE Gap	ROC-AUC	MLE Gap	ROC-AUC	MLE Gap
Ours	0.330 ± 0.000	**0.000**	84586.003 ± 74.140	74.459	0.741 ± 0.000	**0.000**	0.496 ± 0.000	**0.000**
WGTD	0.334 ± 0.005	0.004	84832.096 ± 220.436	319.557	0.742 ± 0.004	0.004	0.496 ± 0.000	**0.000**
TabularMark	0.987 ± 0.048	0.656	$\mathbf{84521.708 \pm 59.071}$	**9.164**	0.742 ± 0.004	0.001	0.534 ± 0.157	0.037
RINTAW	$\mathbf{0.328 \pm 0.002}$	0.001	85047.119 ± 601.940	534.575	$\mathbf{0.738 \pm 0.004}$	0.003	$\mathbf{0.493 \pm 0.002}$	0.003

Table 9: We report the RMSE for regression task and ROC-AUC score for classification task. Each experiment is repeated 10 times and the mean and standard deviation are reported. Furthermore, we also report the MLE gap, which is the difference between the downstream performance of watermarked data and real test data. The best results in each column is highlighted in bold.

Second, we investigate the detectability and robustness of our watermark under different threat models. Particularly, we employ eight common operations (attacks) on tabular data, divided into two categories: (1) table level attacks: row shuffling, row deletion, column deletion, feature selection according to feature importance; and (2) cell level attacks: Gaussian noise addition, uniform noise addition, value alteration, and truncation. For row deletion, we randomly remove 20% of rows. For column deletion, we randomly remove 2 feature columns. For feature selection according to feature importance, we keep the top 40% of columns with the highest feature importance. For Gaussian noise addition, we inject i.i.d. Gaussian noise with mean zero and variance one into numeric columns. For uniform noise addition, we inject noise drawn i.i.d. from $\mathsf{Unif}[-0.1, 0.1]$ into numeric columns. For value alteration, we modify the numeric columns by multiplying them with a random factor drawn uniformly at random from $[0.9, 1.1]$, i.e., increase or decrease each numeric cell by at most 10% of their original value. For truncation, we truncate all numeric float values to only 2 decimal places. For each attack, we report the mean p-value from the detection procedure, where the dataset is detected as watermarked if p-value < 0.05. The result of this experiment is summarized in Table 10. Among all considered watermarking techniques, our method is the most consistent at detecting the embedded watermark under all threat models. On the other hand, WTGD is generally robust to

table level attacks (row deletion, column deletion, feature importance column deletion), but is more susceptible to additive noise attacks. TabularMark is the least robust method, as it often fails to detect watermark under most cell level attacks (additive noise, value alteration, and truncation). Finally, RINTAW is very robust to row deletion and row shuffling, but may fail to detect the watermark under column deletion and cell level attacks. Overall, our watermarking technique achieves both minimal distortion (small MLE gap in Table 9) and high fidelity/robustness (consistently low p-value in Table 10).

Dataset	Method	Watermarked	Row Shuffled	Row Del.	Col. Del.	Importance	Gaussian	Uniform	Alteration	Truncation
Boston	**Ours**	0.007 ± 0.000	0.012 ± 0.003	0.011 ± 0.008	0.011 ± 0.007	0.007 ± 0.000	0.018 ± 0.008	0.018 ± 0.008	0.018 ± 0.009	0.002 ± 0.000
	WTGD	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	1.000 ± 0.490	0.462 ± 0.011	0.418 ± 0.015	0.000 ± 0.000	0.000 ± 0.000
	TabularMark	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	1.000 ± 0.400	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
	RINTAW	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.529 ± 0.212	0.594 ± 0.137	0.581 ± 0.381	0.386 ± 0.315	0.000 ± 0.000
California	**Ours**	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.014 ± 0.008	0.018 ± 0.005	0.013 ± 0.007	0.000 ± 0.000
	WTGD	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.482 ± 0.006	0.273 ± 0.161	0.055 ± 0.100	0.000 ± 0.000
	TabularMark	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.000 ± 0.000
	RINTAW	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.022 ± 0.027	0.520 ± 0.325	0.820 ± 0.105	0.442 ± 0.046	0.000 ± 0.000
Adult	**Ours**	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.014 ± 0.008	0.010 ± 0.009	0.018 ± 0.005	0.000 ± 0.000
	WTGD	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.446 ± 0.012	0.376 ± 0.186	0.000 ± 0.000	0.000 ± 0.000
	TabularMark	0.039 ± 0.039	0.358 ± 0.277	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.398 ± 0.390
	RINTAW	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.358 ± 0.150	0.561 ± 0.313	0.530 ± 0.246	0.667 ± 0.149	0.000 ± 0.000
Wilt	**Ours**	0.000 ± 0.000	0.004 ± 0.008	0.010 ± 0.009	0.087 ± 0.029	0.000 ± 0.000	0.014 ± 0.008	0.017 ± 0.007	0.038 ± 0.040	0.021 ± 0.000s
	WTGD	0.034 ± 0.016	0.338 ± 0.167	0.345 ± 0.168	0.381 ± 0.147	1.000 ± 0.000	0.483 ± 0.014	0.481 ± 0.010	0.486 ± 0.006	0.090 ± 0.124
	TabularMark	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
	RINTAW	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.626 ± 0.130	0.584 ± 0.340	0.363 ± 0.342	0.440 ± 0.280	0.700 ± 0.109

Table 10: We report the p-value of each detection algorithm for different threat models and datasets. Each experiment is repeated 10 times and the mean and standard deviation are reported. Except for Wilt dataset and column deletion attack, our watermark can always be reliably detected for the remaining datasets and threat models, i.e., $p < 0.05$. On the other hand, WTGD often fails to detect the watermark under Gaussian noise addition and uniform noise addition. Furthermore, TabularMark fails to detect the watermark in most cell level attacks (additive noise, value alteration, and truncation). Finally, RINTAW may fail to detect the watermark under feature importance column drop and cell level attacks.

B.5 IMPLEMENTATION DETAILS

In this subsection, we provide the pseudo-code of our core implementations: PAIR subroutine (Algorithm 1), Pairwise Tabular Watermarking (Algorithm 2), and the detection process (Section 4.2).

Helper function: Create Green List

```
Algorithm CREATE_GREEN_LIST(value, bins, green_list_percent):
    decimal := fractional_part(value)
    b := digitize(decimal, bins)                # bin index for decimal part
    h := hash(b)
    rng_h := RNG(seed=h)                        # deterministic per seed value

    choices := [1, 2, ..., len(bins)-1]
    k := floor(len(bins) * green_list_percent)
    green := rng_h.choice(choices, size=k, replace=False)

    return green
```

Algorithm 1: PAIR subroutine

```
Algorithm PAIR_COLUMNS(X, labels=None, min_decimals=2, pair_percent=1):
    # 1) Build sampling distribution over columns
    if labels is provided:
        importance := RandomForestFeatureImportance(X, labels)
    else:
        importance := uniform_over_columns(X)
    importance := smooth_zeros_and_normalize(importance)

    # 2) Sample a column order and keep valid float columns
    order := sample_without_replacement(columns(X), p=importance)
```

```
    valid := valid_float_columns(X, min_decimals)
    cols := filter(order, valid)

    # 3) Pair neighboring columns in sampled order
    seed_cols := cols[0::2]
    mark_cols := cols[1::2]
    if len(seed_cols) > len(mark_cols):
        seed_cols := seed_cols[:-1]

    n_pairs := min(ceil(len(seed_cols) * pair_percent), len(seed_cols))
    pairs := []
    for i in 0..n_pairs-1:
        pairs.append((seed_cols[i], mark_cols[i]))

    return pairs
```

Algorithm 2: Watermark

```
Algorithm WATERMARK(X, pairs, bins, green_list_percent, corresponding_decimals):
    function WM_ONE(seed_val, mark_val):
        green := CREATE_GREEN_LIST(seed_val, bins, green_list_percent)

        d, nd := decimal_part_and_precision(mark_val)
        nd := max(nd, corresponding_decimals)
        b := digitize(d, bins)

        if b in green:
            return mark_val

        g := nearest(green, b)
        d_new := uniform(bins[g-1], bins[g])    # sample in nearest green bin
        d_new := truncate_to_decimals(d_new, nd)

        return round(round(mark_val - d, nd) + d_new, nd)

    W := copy(X)
    pair_map := {}

    for (s, t) in pairs:
        for r in rows(X):
            W[r, t] := WM_ONE(X[r, s], X[r, t])
        pair_map[s] := t

    return W, {"pairs": pair_map}
```

Algorithm: Detection

```
Algorithm DETECT(X, bins, green_list_percent, alpha=0.05, labels=None):
    n_check := num(X.columns)/2
    best := {p=+inf, z=None, seed_col=None, mark_col=None}

    if labels is provided:
        X := reorder_columns_by_feature_importance(X, labels)
    else:
        X := random_column_permutation(X)

    for each seed_col in columns(X):
        for each mark_col in columns(X), mark_col != seed_col:
            violations := 0
```

```
            for (s, m) in first_n_row_pairs(X, n_check, seed_col, mark_col):
                green := CREATE_GREEN_LIST(s, bins, green_list_percent)
                b := digitize(fractional_part(m), bins)
                if b not in green:
                    violations := violations + 1

            z := z_statistic(successes=n_check - violations, total=n_check)
            p := z_to_p(z)

            if (p <= alpha) and (z > 0):
                return True, {p, z, violations, seed_col, mark_col}

            if p < best.p:
                best := {p, z, violations, seed_col, mark_col}

    return False, best
```

C SPOOFING ALGORITHM

In this section, we provide the detailed spoofing Algorithm 3 described in Section 5. Given access to a watermarked dataset, either by the WGTD watermark [He et al., 2024] or by Algorithm 2 and without knowing the exact underlying parameters used for watermarking, an adversary attempt to create a synthetic dataset that can reliably be detected as 'watermarked' by the corresponding detector. The goal of this 'spoofing' attack is to then generate a dataset with toxic or harmful content that are detrimental to the original watermark provider's reputation. Watermark spoofing attacks have been studied extensively in both traditional relational database watermark, and more recently, in large language model watermarks. Our proposed algorithm 3 presents a first attempt at studying the effects of spoofing attacks on generative tabular data that was missing from prior work in this line of research.

Algorithm 3 Fractional Replacement

Input: Synthetic dataset $\mathbf{S} \in \mathbb{R}^{m \times n}$, Watermarked dataset $\mathbf{W} \in \mathbb{R}^{m \times n}$.
Output: Modified dataset $\mathbf{X}'$.

Compute the fractional parts of all elements in $\mathbf{S}$ as $\mathbf{S\_fractions}$.
Compute the fractional parts of all elements in $\mathbf{W}$ as $\mathbf{W\_fractions}$.
for each element x_{ij} in $\mathbf{S}$ **do**
    Extract the integer part: $\lfloor x_{ij} \rfloor$.
    Compute the fractional part: frac_part $= |x_{ij} - \lfloor x_{ij} \rfloor|$.
    Find the closest fractional part: closest_frac $= \arg\min_{f \in \mathbf{S\_fractions} \cup \mathbf{W\_fractions}} |f - \text{frac\_part}|$.
    if $x_{ij} < 0$ **then**
        Replace x_{ij} with $\lfloor x_{ij} \rfloor -$ closest_frac.
    else
        Replace x_{ij} with $\lfloor x_{ij} \rfloor +$ closest_frac.

D ADDITIONAL DETAILS ON THE ALGORITHMS IN SECTION 3

Note on Algorithm 1. There are several algorithms that creates a weighted permutation of $[2n]$ in Line 3 of Algorithm 1 [Fisher and Yates, 1953, Efraimidis and Spirakis, 2006, Braverman et al., 2015, Vieira, 2019].

Runtime of Algorithm 2. The majority of the computational bottleneck in Algorithm 2 is from the two `for` loops that run over $keys$ and $values$ respectively. The `for` loop corresponding to each $value$ column requires to go through each of the values and move them to the nearest green interval or keep them unchanged. Finding the nearest green interval requires $O(\log b)$ time, and so this loop requires $O(m \log b)$ time. In fact, this is the dominating cost within each of the `for` loop corresponding to each key column. Thus, the total time of running Algorithm 2 is $O(mn \log b)$.

Parameters of Algorithm 2. In Algorithm 2, the seeds are generated using a hash function, where the input is a list of the center of each cell in the key column (line 4). In Algorithm 1, the seed in line 1 can be chosen at will. We save the exact (key-pair) columns generated by Algorithm 1, and use it for detection with Bonferroni correction. If the detector does not have access to the original pairing of keys and values, we perform detection as follows: For each value column, check each individual key column to find a match by using the stored seed and calculate the number of elements in green list intervals defined by this seed. This matching process between key and value columns take at most $O(n^2)$, where n is the number of pairs. We additionally note that when we have a sufficiently large number of bins and data points (rows), it is unlikely that two different columns generate the same seeds, as it is unlikely that the fractional parts for a given feature across all the data points lie in the exact same bins. Therefore, the matching process is likely to identify the key columns correctly.

In practice, one can look at the target dataset to be watermarked and decide the appropriate bin sizes needed. For example, if the dataset has already been truncated to only 2 decimal digits, choosing the number of bins to be 100 (respectively, bin size of 0.01) is a good starting point. The data owner can also decide the bin size by choosing their tolerance for downstream utility and robustness. That is, the fidelity constraint represents an upper bound on the bin size, while the robustness constraints represent different lower bounds on the bin size. In our robustness experiments, we use $b = 1000$ bins for all evaluations.

E PROOF OF FIDELITY

In this section, we provide the analysis for the fidelity property of our watermarking algorithm. At first glance, the theorem statements in Theorem 4.1 and Corollary 4.2 are similar to that of He et al. [2024]. However, there is a crucial main difference between our Algorithm 2 and their Algorithm 1 that leads to different analysis. Specifically, in Algorithm 2, we do not force the red and green intervals to always be next to each other like in Algorithm 1 of He et al. [2024]. Hence, when we promote a cell in red interval to the nearest green interval (Line 9-10 in Algorithm 2), this 'destination' green interval can be far away from the original red interval. Meanwhile, for He et al. [2024], the distance between any red and green intervals is always 1. Hence, their analysis for the ℓ_∞ distance between the original and watermarked datasets is incompatible with our setting. In the following proofs of Theorem 4.1 and Corollary 4.2, we provide high-probability upper bounds on the distances, while the guarantees in He et al. [2024] always hold (with probability 1).

Proofs of Theorem 4.1

Proof. Let x be an element from the one of the value columns in the table $\mathbf{X}$, such that $x \in I_j$, where I_j is the interval $\left[\frac{j-1}{b}, \frac{j}{b}\right]$ and the label of I_j is red. Then, let x_w be the corresponding watermarked value in $\mathbf{X}_w$. If $|x - x_w| > \frac{k}{b}$, then x_w is either in one of the intervals $I_1, \ldots, I_{\frac{j-k}{b}}$ or in one of the intervals $I_{\frac{j+k}{b}}, \ldots, I_b$. By Algorithm 2, we know that there are no green intervals between $I_{\frac{j-k}{b}}$ and $I_{\frac{j+k}{b}}$. Since we label each interval as red or green independently with probability $1/2$, the probability that there are no green intervals between $I_{\frac{j-k}{b}}$ and $I_{\frac{j+k}{b}}$ is $(\frac{1}{2})^{2k-1}$, where we use the fact that there are $2k-1$ intervals between $I_{\frac{j-k}{b}}$ and $I_{\frac{j+k}{b}}$ (excluding these). Then, we have

$$\Pr\left(|x - x_w| > \frac{k}{b}\right) \leq \frac{1}{2^{2k-1}}.$$

Now using the union bound over all x in the value columns of $\mathbf{X}$, we have

$$\Pr\left(\|\mathbf{X} - \mathbf{X}_w\|_\infty > \frac{k}{b}\right) \leq \frac{mn}{2^{2k-1}}.$$

Setting $\frac{mn}{2^{2k-1}} \leq \delta$ and solving for k gives us

$$\Pr\left(\|\mathbf{X} - \mathbf{X}_w\|_\infty > \frac{1}{2b}\left(\log_2\left(\frac{mn}{\delta}\right) + 1\right)\right) \leq \delta.$$

Since $\|X - X_w\|_\infty \leq 1$ always, we have with probability at least $1 - \delta$ for $\delta \in (0, 1)$,

$$\|\mathbf{X} - \mathbf{X}_w\|_\infty \leq \min\left\{\frac{1}{2b}\left(\log_2\left(\frac{mn}{\delta}\right) + 1\right), 1\right\}.$$

□

Remark E.1 (Downstream processing)**.** *Since the input data* $\mathbf{X}$ *and the watermarked data* $\mathbf{X}_w$ *are close according to Theorem 4.1, we can conclude that any downstream processing performed on these data sets will give similar results, as long as this processing can be implemented by a "nice" function.*

For example, if the downstream processing is described by a Lipschitz function $f : \mathbb{R}^{m\times 2n} \to \mathbb{R}^{a\times b}$ *with Lipschitz constant* L*, then* $\|f(\mathbf{X}) - f(\mathbf{X}_w)\|_r \le L\|\mathbf{X} - \mathbf{X}_w\|_d$ *for some appropriate norms* $\|\cdot\|_d$ *and* $\|\cdot\|_r$ *on the domain and the range of* f*, respectively. Then, using equivalence of norms in finite dimensions, there is some constant* C *(possibly dimension dependent) such that* $\|\cdot\|_d \le C\|\cdot\|_\infty$*. Then, by Theorem 4.1, we have*

$$\|f(\mathbf{X}) - f(\mathbf{X}_w)\|_r \le LC \min\left\{\frac{1}{2b}\left(\log_2\left(\frac{mn}{\delta}\right) + 1\right), 1\right\}. \tag{5}$$

Hence, for sufficiently large b*, the output of such a downstream task acting on the input data and the watermarked data are going to be very similar.*

Proof of Corollary 4.2

Proof. With probability $1 - \delta$, using Theorem 4.1 and definition of Wasserstein distance between empirical distributions, we have

$$\begin{aligned}
&\mathcal{W}_k(F_{\mathbf{X}}, F_{\mathbf{X}_w}) \\
&\le \left(\sum_{j=1}^{m} \frac{1}{m}\|\mathbf{X}[j,:] - \mathbf{X}_w[j,:]\|_2^k\right)^{1/k} \\
&\le \left(\sum_{j=1}^{m} \frac{1}{m}\left(\sqrt{2n}\|\mathbf{X}[j,:] - \mathbf{X}_w[j,:]\|_\infty\right)^k\right)^{1/k} \\
&\le \max_{j\in[m]}\left(\sqrt{2n}\|\mathbf{X}[j,:] - \mathbf{X}_w[j,:]\|_\infty\right) \\
&= \sqrt{2n}\|\mathbf{X} - \mathbf{X}_w\|_\infty \\
&\le \frac{\sqrt{n/2}}{b}\left(\log_2\left(\frac{mn}{\delta}\right) + 1\right).
\end{aligned}$$

□

F PROOFS OF DETECTABILITY

In this section, we provide the analysis for the detection procedure described in Section 4.2. Similar to He et al. [2024] and Zheng et al. [2024], we employ a hypothesis testing framework to detect our water in order to be robust to minor editing attacks. However, compared to He et al. [2024], we *do not need* the assumptions that (i) the number of bins $b \to \infty$ and (ii) the data comes from a distribution with continuous probability density function for our analysis to hold. This implies in particular that our results here are non-asymptotic, distribution independent, and exact. Consequently, the proof structure for our Lemma 4.3 is different from that of He et al. [2024, Lemma 1]. On the other hand, Zheng et al. [2024] does not provide theoretical analysis for the null distribution before the dataset is watermarked.

Proof of Lemma 4.3

Proof. Let $I_j = \left[\frac{j-1}{b}, \frac{j}{b}\right]$ for all $j \in [b]$ denote an interval on $[0, 1]$. Define the indicator random variable $\mathbf{1}_g : [b] \to \{0, 1\}$ as $\mathbf{1}_g(j) = 1$ if I_j is labeled green. Then given $\mathbf{1}_g$, we know which intervals are green and which intervals are red. Also, define $\mathcal{G} = \cup_{j\in[b],\mathbf{1}_g(j)=1} I_j$ as the union of the intervals which are labeled green. So by definition $\mathcal{G}$ is a set valued random

variable. Therefore conditioned on $\mathbf{1}_g$ we have

$$\Pr\left[x \in \mathcal{G} \mid \mathbf{1}_g\right] = \sum_{j \in [b], \mathbf{1}_g(j)=1} \Pr[x \in I_j] \tag{6}$$

$$= \sum_{j=1}^{b} \Pr[x \in I_j] \cdot \mathbf{1}_g(j) \tag{7}$$

where the second equality uses the fact that $\mathbf{1}_g(j) \in \{0, 1\}$ for all $j \in [b]$. Then, using the fact that the expected value of conditional probability of an event gives the probability of that event we have

$$\Pr(x \in \mathcal{G}) = \mathbb{E}\left[\Pr\left[x \in \mathcal{G} \mid \mathbf{1}_g\right]\right] \tag{8}$$

$$= \sum_{j=1}^{b} \Pr[x \in I_j] \cdot \mathbb{E}[\mathbf{1}_g(j)] \tag{9}$$

$$= \frac{1}{2} \sum_{j=1}^{b} \Pr[x \in I_j] \tag{10}$$

$$= \frac{1}{2}, \tag{11}$$

where in the second equality we use that the expected value of $\mathbf{1}_g(j)$ is the probability that I_j is labeled green (this is equal to $1/2$ by assumption), and in the last equality we use the fact that x is supported in $[0, 1]$.

□

G PROOFS OF ROBUSTNESS

In this section, we provide the analysis for the results presented in Section 4.3. Compared to prior work of He et al. [2024] and Zheng et al. [2024], we show theoretically that our watermarking approach is robust to both additive noise (addressed in prior work), truncation, and feature selection (novel contributions). In our analysis for robustness against additive noise, we first provide a meta-theorem (Theorem G.1) that show our watermarking approach is robust to *any* additive noise attack where the noise is sampled i.i.d. from an arbitrary distribution. The specific results for uniform noise (Corollary G.3) and Gaussian noise (Corollary G.4) are presented as corollaries. Due to the structural differences in how we embed the watermark in Algorithm 3 compared to prior work, their robustness analysis (or lack thereof) does not apply to our setting.

Proof of Theorem 4.4

Proof. If x_{tr} falls out of the original green interval I_j, then we know that either $x_{\text{tr}} > {}^{j}/_{b}$ or $x_{\text{tr}} < {}^{j-1}/_{b}$. Since $b \leq 10^p$, we know that the green interval I_j lies in the union of at most two consecutive grids. We consider two cases depending on whether I_j contains a grid point or not.

When I_j does not contain a grid point. Since the truncation function defined in Eq. (4) always truncate an element x to the nearest left grid point, which lies outside of I_j, we have

$$\Pr\left[x_{\text{tr}} \notin I_j | c \notin I_j, \forall c \in \text{grid}\right] = 1.$$

When I_j contains a grid point. Let c denote the grid point in I_j. Then, when the interval I_j contains a grid point, we have:

$$\begin{aligned}\Pr[x_{\text{tr}} \notin I_j | c \in I_j] &= \Pr\left[x \in \left[\frac{j-1}{b}, c\right)\right] \\ &= \frac{c - \frac{j-1}{b}}{\frac{1}{b}} \\ &= c \cdot b - j + 1\end{aligned}$$

Then, summing over all possible events, we have the probability that the truncation attack successfully moves a watermarked element out of its original green interval is:

$$\begin{aligned}
&\Pr[x_{\text{tr}} \notin I_j] \\
&= \Pr[x_{\text{tr}} \notin I_j | c \notin I_j, \forall c \in \text{grid}] \cdot \Pr[c \notin I_j, \forall i \in \text{grid}] \\
&+ \Pr[x_{\text{tr}} \notin I_j | c \in I_j] \cdot \Pr[c \in I_j] \\
&= \Pr[c \notin I_j, \forall c \in \text{grid}] + (c \cdot b - j + 1) \Pr[c \in I_j] \\
&= \left(\frac{b-1}{b}\right)^{10^p} + \frac{c \cdot b - j + 1}{b} \\
&= \frac{(b-1)^{10^p} + b^{10^p - 1}(c \cdot b - j + 1)}{b^{10^p}}
\end{aligned}$$

Hence, the probability that x_{tr} is in a red interval is $\rho_{\text{trunc}} = \frac{(b-1)^{10^p} + b^{10^p-1}(c \cdot b - j + 1)}{2b^{10^p}}$.

□

G.1 ROBUSTNESS TO ADDITIVE NOISE

Theorem G.1 (Robustness under noise)**.** *Fix a column in the watermarked dataset. Let n_a be the number of cells in this column that an adversary can inject noise into. Let S be a subset of $[b]$ such that $|S| = n_a$. Also let the noise injected into each cell is drawn i.i.d. from some distribution, i.e., $\epsilon \sim \mathcal{D}$.*

Define parameter γ as

$$\gamma \coloneqq \frac{1}{n_a} \sum_{i \in S} \sum_{j \in [b]} \Pr[(x_i + \epsilon_i)^\circ \in I_j], \tag{12}$$

where $\gamma \leq 1$ and ϵ_i is the noise added to x_i. Then, we must have

$$n_a \geq \frac{m - z_{\text{th}}\sqrt{m}}{2 - \gamma}, \tag{13}$$

for the expected z-score to be less than z_{th} (i.e., to remove the watermark).

Proof. Define $x^\circ = x - \lfloor x \rfloor$ as the fractional part of any $x \in \mathbb{R}$.

Let $I_j = \left[\frac{j-1}{b}, \frac{j}{b}\right]$ for all $j \in [b]$ denote an interval on $[0, 1]$. Define the indicator random variable $\mathbf{1}_g : [b] \to \{0, 1\}$ as $\mathbf{1}_g(j) = 1$ if I_j is labeled green. Then given $\mathbf{1}_g$, we know which intervals are green and which intervals are red. Also define $\mathcal{G} = \cup_{j \in [b], \mathbf{1}_g(j) = 1} I_j$ as the union of the intervals which are labeled green. So by definition $\mathcal{G}$ is a set valued random variable.

Furthermore, we define S to be a subset of row indices, $[m]$, in which we inject noise, such that $|S| = n_a$. For each $i \in S$, let ϵ_i be i.i.d. samples drawn from $\mathcal{D}$, and for each $i \notin S$, $\epsilon_i = 0$.

We want to find a bound on n_a such that in expectation the number of cells in a column that are watermarked is below the required z-score threshold. Given z-score threshold equal to z_{th}, the minimum number of cells that need to be in the green intervals is $T_0 = \frac{z_{\text{th}}\sqrt{m} + m}{2}$. The expected number of cells in the column where we add noise that is still in a green interval is

$$\mathbb{E}\left[\sum_{i \in S} \mathbf{1}[(x_i + \epsilon_i)^\circ \in \mathcal{G}] \mid \mathbf{1}_g\right] = \sum_{i \in S} \Pr\left[(x_i + \epsilon_i)^\circ \in \mathcal{G} \mid \mathbf{1}_g\right]. \tag{14}$$

For all $i \in S$, conditioned on $\mathbf{1}_g$ we have

$$\Pr\left[(x_i + \epsilon_i)^\circ \in \mathcal{G} \mid \mathbf{1}_g\right] = \sum_{j \in [b], \mathbf{1}_g(j) = 1} \Pr[(x_i + \epsilon_i)^\circ \in I_j] \tag{15}$$

$$= \sum_{j=1}^{b} \Pr[(x_i + \epsilon_i)^\circ \in I_j] \cdot \mathbf{1}_g(j), \tag{16}$$

where the second equality uses the fact that $\mathbf{1}_g(j) \in \{0,1\}$ for all $j \in [b]$.

From Eq. (14) and Eq. (16) and taking expectation over the labeling of the intervals we have

$$\mathbb{E}\left[\sum_{i\in S}\mathbf{1}[(x_i+\epsilon_i)^\circ \in \mathcal{G}]\right] = \mathbb{E}\left[\mathbb{E}\left[\sum_{i\in S}\mathbf{1}[(x_i+\epsilon_i)^\circ \in \mathcal{G}] \mid \mathbf{1}_g\right]\right] \tag{17}$$

$$= \sum_{i\in S}\sum_{j\in[b]} \Pr[(x_i+\epsilon_i)^\circ \in I_j]\cdot \mathbb{E}\left[\mathbf{1}_g(j)\right] \tag{18}$$

$$= \frac{n_a\gamma}{2}, \tag{19}$$

where the last line uses the definition of γ in Eq. (12).

Now we prove that $0 \le \gamma \le 1$. Since γ is the sum of probabilities, it is non-negative. Since,

$$\gamma = \frac{1}{n_a}\sum_{i\in S}\sum_{j\in[b]} \Pr[(x_i+\epsilon_i)^\circ \in I_j], \tag{20}$$

and for every $i \in S$, there is exactly one $j \in [b]$ such that $(x_i+\epsilon_i)^\circ \in I_j$. So, we must have $\gamma \le 1$.

The expected number of cells which are in a green interval within the column is then:

$$m - n_a + \frac{n_a\gamma}{2}. \tag{21}$$

So, to break the watermark Eq. (21) should be less than T_0, which gives

$$m - n_a + \frac{n_a\gamma}{2} \le \frac{z_{\text{th}}\sqrt{m}+m}{2} \tag{22}$$

$$n_a \ge \frac{m - z_{\text{th}}\sqrt{m}}{2-\gamma}. \tag{23}$$

□

Remark G.2 (Distribution independent robustness)**.** *We observe that since $0 \le \gamma \le 1$, the lower bound on n_a can also be expressed independent of γ as*

$$n_a \ge \frac{1}{2}\left(m - z_{\text{th}}\sqrt{m}\right). \tag{24}$$

Thus, even when the distribution of the noise is not known, one can have an estimate on the maximum number of cells in a column of the watermarked table that needs to be corrupted to remove the watermark.

Corollary G.3 (Robustness under uniform noise)**.** *Fix a column in the watermarked dataset. Let n_a be the number of cells in this column that an adversary can inject noise into. Let S be a subset of $[b]$ such that $|S| = n_a$. Also let the noise injected into each cell is drawn i.i.d from an uniform distribution, i.e., $\epsilon \sim \mathsf{Unif}[-\sigma,\sigma]$ for $0 < \sigma \le 1$.*

Define parameter γ as

$$\gamma := \frac{1}{2n_a\sigma}\sum_{i\in S}\sum_{j=1}^{b}\Bigg(\max\left(0,\min\left(\frac{j}{b},x_i+\sigma\right)-\max\left(\frac{j-1}{b},x_i-\sigma\right)\right)$$
$$+\max\left(0,\min\left(\frac{j}{b}-1,x_i+\sigma\right)-\max\left(\frac{j-1}{b}-1,x_i-\sigma\right)\right)$$
$$+\max\left(0,1+\min\left(\frac{j}{b},x_i+\sigma\right)-\max\left(1+\frac{j-1}{b},x_i-\sigma\right)\right)\Bigg), \tag{25}$$

where $\gamma \le 1$. Then, we must have

$$n_a \ge \frac{m - z_{\text{th}}\sqrt{m}}{2-\gamma}, \tag{26}$$

for the expected z-score to be less than z_{th} (i.e., to remove the watermark).

Proof. We just need to show Eq. (25) holds, since the remaining claims of the statement follow directly from Theorem G.1. There are the following three possible cases for $(x_i + \epsilon_i)^\circ$ using $\epsilon_i \sim \mathsf{Unif}[-\sigma, \sigma]$

$$(x_i + \epsilon_i)^\circ = \begin{cases} 1 + x_i + \epsilon_i & x_i + \epsilon_i \in [-1, 0) \\ x_i + \epsilon_i & x_i + \epsilon_i \in [0, 1] \\ x_i + \epsilon_i - 1 & x_i + \epsilon_i \in (1, 2]. \end{cases} \tag{27}$$

Then using the union bound we have,

$$\Pr[(x_i + \epsilon_i)^\circ \in I_j] = \Pr\left[\frac{j-1}{b} - 1 \le x_i + \epsilon_i \le \frac{j}{b} - 1\right] + \Pr\left[\frac{j-1}{b} \le x_i + \epsilon_i \le \frac{j}{b}\right] + \Pr\left[1 + \frac{j-1}{b} \le x_i + \epsilon_i \le 1 + \frac{j}{b}\right]. \tag{28}$$

Equating Eq. (28). Since $e_i \sim \mathsf{Unif}[-\sigma, \sigma]$, then $x_i + e_i \sim \mathsf{Unif}[x_i - \sigma, x_i + \sigma]$. Then we have

$$\Pr\left[\frac{j-1}{b} \le x_i + \epsilon_i \le \frac{j}{b}\right] = \frac{\max\left(0, \min\left(\frac{j}{b}, x_i + \sigma\right) - \max\left(\frac{j-1}{b}, x_i - \sigma\right)\right)}{2\sigma}, \tag{29}$$

where $\max\left(0, \min\left(\frac{j}{b}, x_i + \sigma\right) - \max\left(\frac{j-1}{b}, x_i - \sigma\right)\right)$ is the overlap between the intervals $[x_i - \sigma, x_i + \sigma]$ and $\left[\frac{j-1}{b}, \frac{j}{b}\right]$.

Similarly, we have

$$\Pr\left[\frac{j-1}{b} - 1 \le x_i + \epsilon_i \le \frac{j}{b} - 1\right] = \frac{\max\left(0, \min\left(\frac{j}{b} - 1, x_i + \sigma\right) - \max\left(\frac{j-1}{b} - 1, x_i - \sigma\right)\right)}{2\sigma} \tag{30}$$

$$\Pr\left[1 + \frac{j-1}{b} \le x_i + \epsilon_i \le 1 + \frac{j}{b}\right] = \frac{\max\left(0, 1 + \min\left(\frac{j}{b}, x_i + \sigma\right) - \max\left(1 + \frac{j-1}{b}, x_i - \sigma\right)\right)}{2\sigma} \tag{31}$$

Combining everything. Plugging Eq. (30), Eq. (31), and Eq. (29) in Eq. (28) and subsequently in Eq. (12) gives us the claim. As stated before, the rest of the proof follows from Theorem G.1. □

Corollary G.4 (Robustness under Gaussian noise)**.** *Fix a column in the watermarked dataset. Let n_a be the number of cells in this column that an adversary can inject noise into. Let S be a subset of $[b]$ such that $|S| = n_a$. Also let the noise injected into each cell is drawn i.i.d from an Gaussian distribution, i.e., $\epsilon \sim \mathcal{N}(0, \sigma^2)$ for $0 < \sigma \le 1$.*

Define parameter γ as

$$\gamma := \frac{1}{n_a} \sum_{i \in S} \sum_{j=1}^{b} \frac{1}{\sqrt{2\pi}\sigma} \int_{\frac{j-1}{b}}^{\frac{j}{b}} \sum_{k=-\infty}^{\infty} \exp\left(-\frac{(\theta/2\pi - x_i + 2\pi k)^2}{2\sigma^2}\right) d\theta, \tag{32}$$

where $\gamma \le 1$ and ϵ_i is the noise added to x_i. Then, we must have

$$n_a \ge \frac{m - z_{\text{th}}\sqrt{m}}{2 - \gamma}, \tag{33}$$

for the expected z-score to be less than z_{th} (i.e., to remove the watermark).

Proof. Similar to Corollary G.3 we only need to show Eq. (32) holds, since the rest of the proof follows from Theorem G.1. Observe that when $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, $(x_i + \epsilon_i)^\circ$ can be expressed as:

$$(x_i + \epsilon_i)^\circ = x_i + \epsilon_i - \lfloor x_i + \epsilon_i \rfloor. \tag{34}$$

Observe that the random variable $(x_i + \epsilon_i)^\circ$ is always the fractional part of $x_i + \epsilon_i$, and $x_i + \epsilon_i \sim \mathcal{N}(x_i, \sigma^2)$. The distribution of $(x_i + \epsilon_i)^\circ$ is exactly the wrapped normal distribution [Jammalamadaka and Sengupta, 2001, §2.2.6] with the density function

$$f(\theta; x_i, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \sum_{k=-\infty}^{\infty} \exp\left(-\frac{(\theta/2\pi - x_i + 2\pi k)^2}{2\sigma^2}\right), \tag{35}$$

and is obtained by wrapping the normal distribution around the unit circle, and rescaling the support of the random variable to $[0, 1]$. Then we have

$$\Pr\left[\frac{j-1}{b} \leq (x_i + \epsilon_i)^\circ \leq \frac{j}{b}\right] = \frac{1}{\sqrt{2\pi}\sigma} \int_{\frac{j-1}{b}}^{\frac{j}{b}} \sum_{k=-\infty}^{\infty} \exp\left(-\frac{(\theta/2\pi - x_i + 2\pi k)^2}{2\sigma^2}\right) d\theta. \tag{36}$$

Since $\Pr\left[\frac{j-1}{b} \leq (x_i + \epsilon_i)^\circ \leq \frac{j}{b}\right] = \Pr\left[(x_i + \epsilon_i)^\circ \in I_j\right]$, using the definition of γ we have our claim. □

Remark G.5. *While Eq. (32) looks ominous, it has been extensively studied in the literature. Importantly there are analytical expressions (that avoids the infinite sum, e.g., von Mises distribution closely matches the density function of wrapped normal distribution [Mardia and Jupp, 2009]) that can approximate Eq. (35) well. Furthermore, one can also empirically approximate Eq. (32) by first approximating the infinite sum using small values of k [Kurz et al., 2014] and then evaluating the integral.*

G.2 FEATURE SELECTION

Algorithm 2 takes a black-box pairing subroutine PAIR as an input to determine the set of $(key, value)$ columns. We consider following two feature pairing schemes:

- **P.1** *Uniform*: features are paired uniformly at random,
- **P.2** *Feature importance*: features are paired according to the feature importance ordering, where features with similar importance are paired.

For **P.2**, without loss of generality, we assume that the columns of the original dataset are ordered in descending order of feature importance. Note that this reordering of features does not affect the uniform pairing scheme (**P.1**) and only serves to simplify notation for **P.2** in our analysis. Now consider the scenario that a data scientist pre-processes the watermarked dataset $\mathbf{X}_w$ and keep the columns with largest importance scores. If the features are selected using weights scaling according to **P.2**, we want to understand how many $(key, value)$ pairs that are watermarked due to Algorithm 2 are preserved.

Before proceeding, we require some notations which we state first. Let $T = (t_1, \ldots, t_{2n})$ be an ordered list of columns of a tabular dataset. Let $p = (p_1, \ldots, p_{2n})$ be non-uniform sampling probability of each column. Without loss of generality, assume $p_1 \geq \cdots \geq p_{2n} > 0$. Let $u = (u_1, \ldots, u_{2n})$ such that for all $i \in [2n]$, $u_i = \frac{1}{2n}$. Let $k \in \mathbb{N}^+$, such that $k \leq [2n]$, and let $T_k = (t_1, \ldots, t_k)$.

Let σ_r be a permutation of $[2n]$ obtained using any probability vector $r \in \mathbb{R}^{2n}$. For $m = 1, \ldots, n$ define random variables and their sum as

$$X_m \coloneqq \mathbb{1}\{\sigma_{2m-1}, \sigma_{2m} \in T_k\}, \quad \text{and} \quad X \coloneqq \sum_{i=1}^{n} X_m. \tag{37}$$

X_m checks whether consecutive elements in position $2m-1$ and $2m$ after permutation are in T_k, and so X counts the number of times a $(key, value)$ pair is formed from the columns in T_k. We begin by stating the following theorem on the conditional probability of constructing a pair from top-k items given $s_k < |T_k| - 1$ samples from the top-k items have already been sampled.

Lemma G.6. *Let $p \in \mathbb{R}^{2n}$ be a probability vector sorted in descending order and u is uniform probability distribution on $[2n]$. Let $T_k = [k]$ and $\sum_{i\in[k]} p_i > \frac{k}{2n}$. Let C_k be the samples which have been drawn that lie in T_k (so $|C_k| \leq 2m \leq k-2$), then if $p_i \geq 1/(2n)$ for all $i \in [k]$, for any $2 < m < n$,*

$$\Pr_p[X_{m+1} = 1 \mid C_k] \geq \Pr_u[X_{m+1} = 1 \mid C_k]. \tag{38}$$

Proof. Let S_m be the set of items that have not yet been sampled. By construction, $C_k \subset [n] \setminus S_m$. Then we have

$$\Pr_p[X_{m+1} = 1 \mid C_k] = \sum_{i \in T_k \setminus C_k} \frac{p_i}{\sum_{j\in S_m} p_j} \cdot \frac{(\sum_{j\in T_k\setminus C_k} p_j - p_i)}{\sum_{j\in S_m} p_j - p_i}. \tag{39}$$

Let $Q_m = \sum_{j \in T_k \setminus C_k} p_j$, $R_m = \sum_{j \in S_m} p_j$, and $T_m = R_m - Q_m$. We have $Q_m - p_i \geq (|T_k \setminus C_k| - 1)p_k$ and $T_m \leq (|S_m| - |T_k \setminus C_k|)p_k$. Using this we have

$$\frac{\sum_{j \in T_k \setminus C_k} p_j - p_i}{\sum_{j \in S_m} p_j - p_i} = \frac{Q_m - p_i}{Q_m + T_m - p_i} \tag{40}$$

$$\geq \frac{(|T_k \setminus C_k| - 1)p_k}{(|T_k \setminus C_k| - 1)p_k + T_m} \tag{41}$$

$$\geq \frac{(|T_k \setminus C_k| - 1)p_k}{(|T_k \setminus C_k| - 1)p_k + (|S_m| - |T_k \setminus C_k|)p_k} \tag{42}$$

$$= \frac{|T_k \setminus C_k| - 1}{|S_m| - 1}, \tag{43}$$

where in the first inequality we use: if $a \geq t$ then $a/(a+b) \geq t/(t+b)$ for a fixed $b \geq 0$. Then plugging this in Eq. (39) we have

$$\Pr_p[X_{m+1} = 1 \mid C_k] \geq \frac{|T_k \setminus C_k| - 1}{|S_m| - 1} \cdot \sum_{i \in T_k \setminus C_k} \frac{p_i}{\sum_{j \in S_m} p_j} \tag{44}$$

$$= \frac{|T_k \setminus C_k| - 1}{|S_m| - 1} \cdot \frac{Q_m}{Q_m + T_m} \tag{45}$$

$$\geq \frac{|T_k \setminus C_k| - 1}{|S_m| - 1} \cdot \frac{|T_k \setminus C_k| p_k}{|T_k \setminus C_k| p_k + T_m} \tag{46}$$

$$\geq \frac{|T_k \setminus C_k| - 1}{|S_m| - 1} \cdot \frac{|T_k \setminus C_k| p_k}{|T_k \setminus C_k| p_k + (S_m - |T_k \setminus C_k|)p_k} \tag{47}$$

$$= \frac{|T_k \setminus C_k| \cdot (|T_k \setminus C_k| - 1)}{|S_m| \cdot (|S_m| - 1)}, \tag{48}$$

where in the third line we again use: if $a \geq t$ then $a/(a+b) \geq t/(t+b)$ for a fixed $b \geq 0$.

For uniform probabilities, for all $i \in [2n]$ we have $u_i = \frac{1}{2n}$, and so we have

$$\Pr_u[X_{m+1} = 1 \mid C_k] = \sum_{i \in T_k \setminus C_k} \frac{u_i}{\sum_{j \in S_m} u_j} \cdot \frac{(\sum_{j \in T_k \setminus C_k} u_j - u_i)}{\sum_{j \in S_m} u_j - u_i} = \sum_{i \in T_k \setminus C_k} \frac{1}{|S_m|} \cdot \frac{|T_k \setminus C_k| - 1}{|S_m| - 1} \tag{49}$$

$$= \frac{|T_k \setminus C_k| \cdot (|T_k \setminus C_k| - 1)}{|S_m| \cdot (|S_m| - 1)}. \tag{50}$$

Then combining Eq. (48) and Eq. (50) we have our claim. □

While Lemma G.6 shows that the conditional probability that a pair from top k is formed, at any given point given some of the items appeared prior to that round were from the top-k, when compared to uniform sampling, computing an expectation bound is harder due to sampling without replacement (SWOR).

An equivalent problem to consider for the expectation bound is the following:

Problem G.7. *Let there be $2n$ distinct items ordered $T = \{t_1, \ldots, t_{2n}\}$ with probabilities $p_1 \geq \cdots \geq p_{2n} > 0$ and $\sum_{i=1}^{2n} p_i = 1$. Let $T_k = \{1, \ldots, k\}$ be the top-k indices. Define $P_k = \sum_{i=1}^{k} p_i$, and $\|p_{T_k}\|_2^2 = \sum_{i=1}^{k} p_i^2$. We create a random permutation σ of $[2n]$ by either:*

- *(a)* ***Uniform SWOR****: all permutations equally likely without replacement, or*
- *(b)* ***Proportional SWOR****: sequential sampling without replacement. At each step one samples index i with probability proportional to p_i and remaining elements in T.*

For $m = 1, \ldots, n$ define random variables and the sum of the random variables as

$$X_m := \mathbb{1}\{\sigma_{2m-1}, \sigma_{2m} \in T_k\}, \quad \textit{and} \quad X := \sum_{i=1}^{n} X_m. \tag{51}$$

Compare $\mathbb{E}_p[X]$ (Item (b)) with $\mathbb{E}_u[X]$ (Item (a)).

We have the following results on the expectations

Theorem G.8 (Uniform sampling baseline)**.** *Under uniform sampling* $\mathbb{E}_u[X] = \frac{k(k-1)}{2(2n-1)}$.

Proof. There are $(2n)!$ permutations of $2n$ distinct items. Fixing $j \in [n]$, the number of permutations of $[2n]$ such that the items in positions $2j-1$ and $2j$ are in T_k is $2\binom{k}{2}(2n-2)!$, where the $(2n-2)!$ counts the number of permutations of the remaining $2n-2$ items. Using the fact uniform SWOR for $2n$ items is equivalent to uniformly choosing a permutation of $2n$ items the probability that sampling a permutation σ such that $\sigma_{2j-1}, \sigma_{2j} \in T_k$ is

$$\Pr[\sigma_{2j-1}, \sigma_{2j} \in T_k] = \frac{2\binom{k}{2}(2n-2)!}{(2n)!} = \frac{k(k-1)}{2n(2n-1)}. \tag{52}$$

From Eq. (51) we have that

$$\mathbb{E}_u[X] = \sum_{j=1}^{n} \Pr_u[\sigma_{2j-1}, \sigma_{2j} \in T_k] = \frac{k(k-1)}{2(2n-1)}. \tag{53}$$

□

Theorem G.9. $\mathbb{E}_p[X] \geq \mathbb{E}_u[X]$ *if either of these conditions hold:*

1. $P_k^2 - \|p_{T_k}\|_2^2 \geq \frac{k(k-1)}{2(2n-1)}$,
2. $P_k^2 \geq \frac{k(k-1)}{2(2n-1)} \cdot \frac{1}{1-\frac{\eta^2}{k}}$, *where* $\max_{i\in[k]} p_i \leq \eta\mu$,
3. $\frac{P_k^2}{k}\left(k - 1 - \frac{(\rho-1)^2}{4\rho}\right) \geq \frac{k(k-1)}{2(2n-1)}$, *where* $\rho = \frac{\max_{i\in[k]} p_i}{\min_{i\in[k]} p_i}$,
4. $P_k^2 \geq \frac{k(k-1)}{2(2n-1)} \cdot \frac{1}{1-\frac{e^\delta}{k}}$, *where* $\delta \geq \log k - H_2(w)$,
5. $P_k^2 \geq \frac{k(k-1)}{2(2n-1)} \cdot \frac{1}{1-\frac{1}{N_{\text{eff}}}}$, *where* $N_{\text{eff}} = \frac{1}{\sum_{i\in[k]} w_i^2}$.

Proof. We split the proof for each of the items above in the following paragraphs:

Proof for Item 1. For some rate vector $r = (r_1, \ldots, r_{2n})$ (one can set $r = p$ in case of Item (b) and $r = u$ in case of Item (a)), define

$$R \coloneqq \sum_{j=1}^{2n} r_j, \quad \text{and} \quad Q_k \coloneqq \sum_{j\in T_k} r_j, \tag{54}$$

Define X with respect to r analogous to Eq. (51). The probability that the first two items both lie in T_k is:

$$\Pr_r(X_1 = 1) = \sum_{i\in T_k} \frac{r_i(Q_k - r_i)}{R(R - r_i)}. \tag{55}$$

Let $r^{(m)}$ be the remaining rate vector before forming adjacent pairs in index $2m-1$ and $2m$, and let $T^{(m)}$ and $T_k^{(m)}$ be the remaining items and the remaining top-k items at round m. To simplify notation, define

$$S(r^{(m)}; T_k^{(m)}) \coloneqq \sum_{i\in T_k^{(m)}} \frac{r_i^{(m)}(Q_k^{(m)} - r_i^{(m)})}{R^{(m)}(R^{(m)} - r_i^{(m)})}. \tag{56}$$

Then we have

$$\mathbb{E}_r[X] = \sum_{m=1}^{n} S(r^{(m)}; T_k^{(m)}). \tag{57}$$

Observe that for $r = p$ and $R = 1$ we have $T_k^{(1)} = T_k$. Using $1 - p_i \leq 1$ we have

$$\Pr_p[X_1 = 1] = S(p; T_k) = \sum_{i \in T_k} \frac{p_i(P_k - p_i)}{1 - p_i} \geq \sum_{i \in T_k} p_i(P_k - p_i) = P_k^2 - \|p_{T_k}\|_2^2. \tag{58}$$

Using linearity of expectation and the fact that X_i's are indicator random variables, we have

$$\mathbb{E}_p[X] \geq S(p; T_k), \tag{59}$$

then if

$$P_k^2 - \|p_{T_k}\|_2^2 \geq \frac{k(k-1)}{2(2n-1)}, \tag{60}$$

we have $\mathbb{E}_p[X] \geq \mathbb{E}_u[X]$.

For the remainder of the proof define $\mu \coloneqq P_k/k$, and normalize p_{T_k} such that for all $i \in [k]$, $w_i = p_i/P_k$. So, we have $\sum_{i \in [k]} w_i = 1$, and $\|p_{T_k}\|_2^2 = P_k^2 \sum_{i \in [k]} w_i^2$.

Proof for Item 2. $\max_{i \in [k]} p_i \leq \eta\mu$, is equivalent to the statement $\max_{i \in [k]} w_i \leq \frac{\eta}{k}$. Then we have $w_i^2 \leq \frac{\eta^2}{k^2}$ for all $i \in [k]$ and so $\sum_{i \in [k]} w_i^2 \leq \frac{\eta^2}{k}$, which gets us

$$P_k^2 - \|p_{T_k}\|_2^2 \geq P_k^2(1 - \frac{\eta^2}{k}). \tag{61}$$

Thus, if $P_k^2 \geq \frac{k(k-1)}{2(2n-1)} \cdot \frac{1}{1 - \frac{\eta^2}{k}}$, then $\mathbb{E}_p[X] \geq \mathbb{E}_u[X]$.

Proof for Item 3. We have $\rho \geq 1$ by assumption. Let $\sigma = \frac{1}{k} \sum_{i \in T_k} (p_i - \mu)^2$ be the variance. Then

$$\sum_{i \in T_k} p_i^2 = k(\mu^2 + \sigma^2) = \frac{P_k^2}{k} + k\sigma^2. \tag{62}$$

So we have

$$S(p; T_k) = (1 - \frac{1}{k})P_k^2 - k\sigma^2. \tag{63}$$

Let $a = \min_{i \in T_k} p_i$ and $b = \max_{i \in T_k} p_i$. Using Bhatia-Davis inequality [Bhatia and Davis, 2000, Theorem 1] we have $\sigma^2 \leq (b - \mu)(\mu - a)$. We know $\rho = b/a$ and $a \leq \mu \leq b$. Then substituting $b = \rho a$ and maximizing σ^2 over a for a fixed μ we have

$$a^* = \frac{\mu(\rho + 1)}{2\rho}, \qquad \mu - a^* = \frac{\mu(\rho - 1)}{2\rho}, \quad \text{and} \quad b^* - \mu = \frac{\mu(\rho - 1)}{2}, \tag{64}$$

and so

$$\sigma^2 \leq \frac{\mu^2(\rho - 1)^2}{4\rho}. \tag{65}$$

Then we have

$$S(p; T_k) = (1 - \frac{1}{k})P_k^2 - k\sigma^2 \geq (1 - \frac{1}{k})P_k^2 - \frac{k\mu^2(\rho - 1)^2}{4\rho} = \frac{P_k^2}{k}\left(k - 1 - \frac{(\rho - 1)^2}{4\rho}\right). \tag{66}$$

Then if $\frac{P_k^2}{k}\left(k - 1 - \frac{(\rho-1)^2}{4\rho}\right) \geq \frac{k(k-1)}{2(2n-1)}$ we have $\mathbb{E}_p[X] \geq \mathbb{E}_u[X]$.

Proof for Item 4. Define order 2 Rényi entropy of the normalized top-k weights w as

$$H_2(w) \coloneqq -\log \sum_{i\in[k]} w_i^2. \tag{67}$$

From the definition of δ we have

$$\delta \geq \log k + \log \sum_{i\in[k]} w_i^2 = \log\left(k \sum_{i\in[k]} w_i^2\right) \tag{68}$$

$$\sum_{i\in[k]} w_i^2 \leq \frac{e^\delta}{k}. \tag{69}$$

Then we have

$$P_k^2 - \|p_{T_k}\|_2^2 \geq P_k^2\left(1 - \frac{e^\delta}{k}\right), \tag{70}$$

which then implies that if $P_k^2 \geq \frac{k(k-1)}{2(2n-1)} \cdot \frac{1}{1-e^\delta/k}$ we have $\mathbb{E}_p[X] \geq \mathbb{E}_u[X]$.

Proof of Item 5. Note N_{eff} is the effective support of p_{T_k}. Observe $\frac{1}{\sum_{i\in[k]} w_i^2} = \frac{P_k^2}{\|p_{T_k}\|_2^2}$. This then implies that $P_k^2 - \|p_{T_k}\|_2^2 = P_k^2\left(1 - \frac{1}{N_{\text{eff}}}\right)$. So if $P_k^2 \geq \frac{k(k-1)}{2(2n-1)} \cdot \frac{1}{1-\frac{1}{N_{\text{eff}}}}$ we have $\mathbb{E}_p[X] \geq \mathbb{E}_u[X]$.

Finally observe that $N_{\text{eff}} \geq \frac{k}{\eta^2}$ for assumption in Item 2, $N_{\text{eff}} = O\left(\frac{k}{\rho}\right)$ for assumption in Item 3, and $N_{\text{eff}} \geq ke^{-\delta}$ for assumption in Item 4. □

Remark G.10. *Theorem G.9 identifies regions in the function $p \mapsto \mathbb{E}_p[X]$ for any probability vector p where $\mathbb{E}_p[X] \geq \mathbb{E}_u[X]$. In fact there are other regions in the function where $\mathbb{E}_p[X] \geq \mathbb{E}_u[X]$. For example, one can show that any p can be constructed from u by transferring probability mass from the bottom index to the top index sequentially. Naturally, one way to show $\mathbb{E}_p[X] \geq \mathbb{E}_u[X]$ would be to show that the corresponding potential function $f(p) = \sum_{i=1}^k \frac{p_i(S-p_i)}{1-p_i}$, where $S = \sum_{i\in[k]} p_i$, is monotonically increasing or that it is Schur convex. Unfortunately, this function f is concave when considering transfers within the top-k indices which breaks down the argument. We therefore leave it as an open problem to show that $\mathbb{E}_p[X] \geq \mathbb{E}_u[X]$ for any p.*

H DECODING ALGORITHMS

Consider the case that the continuous space in $(0, 1]$ is uniformly discretized using bins of size $1/b$ i.e., construct bins in $(0, 1]$ as – $\left\{(0, \frac{1}{b}], (\frac{1}{b}, \frac{2}{b}], \ldots, (\frac{b-1}{b}, 1]\right\}$. Then each of these bins are labeled $\{0, 1\}$ with some probability $1 - p_i$ and p_i respectively. We save this data-structure. During query time, each entry of the input table/row is checked to see if the values fall into the bins labeled 1. One can then compute z-scores based on the count of watermarked data identified in the input table. The output of the query function is 0 or 1 depending on whether the table is not watermarked or watermarked, respectively.

For a given input to the query function and the output watermark label, we formalize this problem as a learning problem as follows.

Assumptions. We assume that the number of bins b is known to the detection algorithm. We make such an assumption because we use a statistical learning algorithm for detection, and if b is not known apriori, the VC dimension of the relevant hypothesis class becomes infinity, giving us vacuous sample complexity bounds. The algorithm generating the watermark fixes a probability $p \in (0, 1)$, which is hidden from the detection algorithm. The watermarking algorithm draws b independent and identically distributed samples from the Bernoulli distribution with parameter p. Let $y_1, \ldots, y_b \in \{0, 1\}$ denote the observed values. For $k \in [b]$, we interpret y_k to be the label of the interval (or bin) $\left(\frac{k-1}{b}, \frac{k}{b}\right]$.

Query function. Let $\Pi\colon (0,1] \to [b]$ denote the canonical projection onto the bins, defined as $\Pi(x) = k$ if $x \in \left(\frac{k-1}{b}, \frac{k}{b}\right]$ for $k \in [b]$. Clearly, Π determines the index of the bin into which the input falls. Given a tabular data $\mathbf{X} \in (0,1]^{m\times n}$ as input, we denote $\Pi^{m\times n}\colon (0,1]^{m\times n} \to [b]^{m\times n}$ to be the function that implements Π entry wise on $\mathbf{X}$. Let $F_z\colon [b]^{m\times n} \to \mathbb{R}$ be a z-score function (chosen based on the problem), which first maps the index of the bin in each entry of the tabular data to the corresponding label of that bin, and then processes these labels in an appropriate fashion.

We define the *query function* $Q : (0,1]^{m\times n} \to \{0,1\}$ given by $Q(\boldsymbol{x}) = \operatorname{sgn}(F_z(\Pi^{m\times n}(\boldsymbol{x})))$, where $\operatorname{sgn}\colon \mathbb{R} \to \{0,1\}$ is the sign (or indicator) function defined as

$$\operatorname{sgn}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{71}$$

The query function tells us whether (1) or not (0) the input data is watermarked. Since the labels y_k are not known to the detection algorithm, the query function is also not known.

Goal. We want to approximate the query function up to a small prediction error with high probability. Suppose that $\mathbf{X}$ is the data random variable, taking values in $(0,1]^{m\times n}$. We are given M independent and identically distributed training samples $(\mathbf{X}_1, Q(\mathbf{X}_1)), \ldots, (\mathbf{X}_M, Q(\mathbf{X}_M))$. We wish to use these training samples to learn the function Q, such that given $\epsilon \in (0,1)$ and $\delta \in (0,1)$, we have

$$\Pr_{\mathbf{X}_1,\ldots,\mathbf{X}_M}\left(\Pr_{\mathbf{X}'}\left(\mathcal{A}((\mathbf{X}_i)_{i=1}^M, \mathbf{X}') \neq Q(\mathbf{X}')\right) \leq \epsilon\right) \geq 1-\delta, \tag{72}$$

where $\mathcal{A}\colon ((0,1]^{m\times n})^M \times (0,1]^{m\times n} \to \{0,1\}$ is an algorithm that takes the training data $\mathbf{X}_1, \ldots, \mathbf{X}_M \in (0,1]^{m\times n}$ as input and outputs the query function $\mathcal{A}((\mathbf{X}_i)_{i=1}^M, \cdot)$. Given training data $\mathbf{X}_1, \ldots, \mathbf{X}_M \in (0,1]^{m\times n}$, the quantity $\Pr_{\mathbf{X}'}\left(\mathcal{A}((\mathbf{X}_i)_{i=1}^M, \mathbf{X}') \neq Q(\mathbf{X}')\right)$ is the expected value of the 0-1 loss function (with respect to $\mathbf{X}'$). Note that for Equation (72) to hold, the number of samples M depends on ϵ, δ, and possibly other parameters in the problem. M can be interpreted as the number of queries required to learn the label assignments up to a small error with high probability.

Since the number of bins b is known to the decoding algorithm, it suffices to consider query functions of the form $h = h' \circ \Pi^{m\times n}$, where $h'\colon [b]^{m\times n} \to \{0,1\}$. Furthermore, we suppose that the data is binned before implementing the learning algorithm. Thus, we can write $\mathcal{A}(\cdot,\cdot) = \mathcal{A}'((\Pi^{m\times n})^M(\cdot), \Pi^{m\times n}(\cdot))$, where $\mathcal{A}'\colon ([b]^{m\times n})^M \times [b]^{m\times n} \to \{0,1\}$. Denote $\overline{\mathbf{X}} = \Pi^{m\times n} \circ \mathbf{X}$ to be the random variable taking values in $[b]^{m\times n}$, obtained by binning the entries of $\mathbf{X}$. Then, Equation (72) can be rewritten as

$$\Pr_{\overline{\mathbf{X}}_1,\ldots,\overline{\mathbf{X}}_M}\left(\Pr_{\overline{\mathbf{X}}'}(\mathcal{A}'((\overline{\mathbf{X}}_i)_{i=1}^M), \overline{\mathbf{X}}') \neq Q'(\overline{\mathbf{X}}')) \leq \epsilon\right) \geq 1-\delta, \tag{73}$$

For this reason, it suffices to assume that the input data corresponds to the labels of the bins.

H.1 VC DIMENSION BOUNDS FOR TABULAR DATA USING QUERY FUNCTION OF He et al. [2024]

In He et al. [2024], the z-score function is of the form $F_z(\mathbf{X}) = \beta_0 + \sum_{j=1}^n \beta_j T_j(\mathbf{X}) + \sum_{j=1}^n \alpha_j T_j(\mathbf{X})^2$, where for all i, j, we have $\beta_i, \alpha_j \in \mathbb{R}$ and $T_j(\mathbf{X})$ is the Hamming weight of the label string corresponding to the jth column of $\mathbf{X}$. As noted earlier in this section, it suffices to focus our attention to the case where the input is in $[b]^{m\times n}$, obtained by binning the original data. Our goal is to embed all possible query functions into a neural network and use known bounds on the VC dimension for learning the true query function.

Theorem H.1 (Query complexity bound for querying He et al. [2024] with tabular data)**.** *Given the number of bins b, number of rows m and columns n of the query table, there is a neural network that can use*

$$O\left(\frac{mnb\log(mnb)\log(1/\epsilon) + \log(1/\delta)}{\epsilon}\right) \tag{74}$$

training data (labeled data consisting of query table and whether or not the table is watermarked according to He et al. [2024]), to learn the function which determines whether an input table is watermarked, with error at most $\epsilon > 0$ with respect to 0-1 *loss (see Equation (73)), and with probability greater than or equal to $1-\delta$ over the training samples.*

Proof. We begin by embedding the labels $k \in [b]$ into vectors, so as to vectorize the inputs $[b]^{m\times n}$. To that end, associate the label $k \in [b]$ to the standard unit vector $\mathbf{e}_k \in \mathbb{R}^b$, where $\mathbf{e}_k$ is the vector with 1 at the kth entry and zero elsewhere. Let

$\mathbf{X} \in [b]^{m \times n}$ be the input data, and let it be mapped to the vector $\oplus_{j=1}^{n} \oplus_{i=1}^{m} \mathbf{e}_{\mathbf{X}_{ij}}$. Here, the columns of $\mathbf{X}$ are vertically stacked on top of each other after vectorizing.

Let $\mathbf{w} \in \{0,1\}^b$ denote a candidate vector of labels. Observe that $\langle \mathbf{w}, \mathbf{e}_k | \mathbf{w}, \mathbf{e}_k \rangle = \mathbf{w}_k$ for all $k \in [b]$. Furthermore, for $j \in [n]$, we have $\left\langle \mathbf{w}^{\oplus m}, \oplus_{i=1}^{m} \mathbf{e}_{\mathbf{X}_{ij}} \middle| \mathbf{w}^{\oplus m}, \oplus_{i=1}^{m} \mathbf{e}_{\mathbf{X}_{ij}} \right\rangle = \sum_{i=1}^{m} \mathbf{w}_{\mathbf{X}_{ij}}$. If $\mathbf{w}$ is the correct labeling vector for the jth column, then $\sum_{i=1}^{m} \mathbf{w}_{\mathbf{X}_{ij}} = T_j(\mathbf{X})$. By definition, we have $T_j(\mathbf{X}) \in [0, m]$ for all $j \in [n]$ and all $\mathbf{X} \in [b]^{m \times n}$.

Next, we show how to obtain $T_j(\mathbf{X})^2$ from $T_j(\mathbf{X})$. First, add another layer where we map $T_j(\mathbf{X})$ to $(m+1)T_j(\mathbf{X})$, by choosing the weight to be $m+1$. Next, choose the following piecewise polynomial activation function:

$$\psi(z) = \begin{cases} 0 & \text{if } z \leq 0 \\ z & \text{if } z \in (0, m] \\ z^2 & \text{if } z > m. \end{cases} \tag{75}$$

Then, since $T_j(\mathbf{X}) \in [0, m]$, we obtain $\psi(T_j(\mathbf{X})) = T_j(\mathbf{X})$. On the other hand, since $(m+1)T_j(\mathbf{X}) \in \{0\} \cup [m+1, m(m+1)]$, we obtain $\psi((m+1)T_j(\mathbf{X})) = (m+1)^2 T_j(\mathbf{X})^2$. The constant factor of $(m+1)^2$ can be absorbed into the weights in the next layer.

This motivates us to propose the following neural network architecture for the problem of learning the query function (see Figure 5). The input nodes contain vectorized indices, such that columns of $\mathbf{X}$ are stacked on top of each other. Therefore,

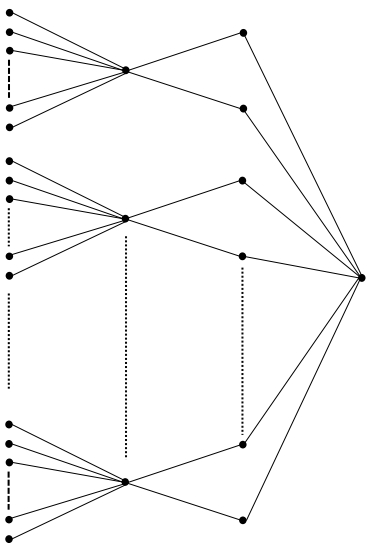

Figure 5: Neural network architecture to embed the problem of learning the query function. The activation function is applied at every node, except the nodes in the input and the output layers.

there are a total of mnb input nodes. The first hidden layer computes $T_1(\mathbf{X}), \ldots, T_n(\mathbf{X})$. Since the $T_j(\mathbf{X}) \in [0, m]$ for all $j \in [n]$, the activation function acting on the first hidden layer does nothing. The second layer effectively fans out $T_j(\mathbf{X})$ for $j \in [n]$. The top node maps $T_j(\mathbf{X})$ to itself, whereas the bottom node maps $T_j(\mathbf{X})$ to $(m+1)T_j(\mathbf{X})$. Thus, applying the activation function at the second hidden layer gives us $T_j(\mathbf{X})$ and $(m+1)^2 T_j(\mathbf{X})^2$ for $j \in [n]$. The output layer is obtained by applying weights and biases to the second hidden layer. Thus, the output of the neural network is of the form $F_z(\mathbf{X}) = \beta_0 + \sum_{j=1}^{n} \beta_j T_j(\mathbf{X}) + \sum_{j=1}^{n} \alpha_j T_j(\mathbf{X})^2$. Since we need to compute the VC dimension, we apply the sign (or indicator) function at the output layer. Therefore, the hypothesis class defined by this neural network is the set of functions computed by it over all the weights and biases at each layer, with a sign function (see (71)) applied at the end. By construction, all the query functions are a part of this hypothesis class. Thus, an upper bound on the VC dimension of the hypothesis class defined by the neural network also gives an upper bound on the VC dimension of the hypothesis class defined by all the query functions.

There are a total of $O(mnb)$ weight and bias parameters before the first layer, $O(n)$ parameters before the second layer, and finally, $O(n)$ parameters before the output layer. Thus, we have a total of $\text{wt} = O(mnb)$ parameters. We have a total of

$L = 3$ layers. Let wt_i denote the number of weight and bias parameters from the input layer till the ith layer. Then, the effective depth of the neural network is $\overline{L} = \sum_{i=1}^{L} \text{wt}_i/\text{wt}$ is of order 1. Then, by [Bartlett et al., 2019, Theorem 6], the VC dimension of the neural network hypothesis class is bounded above by $O(mnb\log(mnb))$. Therefore, by [Shalev-Shwartz and Ben-David, 2014, Theorem 6.8], we can infer that

$$M = O\left(\frac{mnb\log(mnb)\log(1/\epsilon) + \log(1/\delta)}{\epsilon}\right) \tag{76}$$

training samples are sufficient to learn the query function to within an error of ϵ (with respect to the $0-1$ loss) with probability greater than or equal to $1-\delta$ over the training samples.

Now, denote $\mathcal{A}'(\text{training data}, \cdot)$ to be the query function that is learned from the training data. Then, if the distribution of the data is uniform, then $\Pr_{\overline{\mathbf{X}}'}(\mathcal{A}'(\text{training data}, \overline{\mathbf{X}}') \neq Q'(\overline{\mathbf{X}}'))$ is equal to the fraction of indices where the learned query function differs from the true query function. As a result, setting $\epsilon = 0.5/(mnb)$, we obtain the situation where we learn the query function exactly. Thus, we need $O((mnb)^2\log(mnb)\log(b) + mnb\log(1/\delta))$ samples to learn the query function exactly with probability at least $1-\delta$ over the training samples. □

H.2 VC DIMENSION BOUNDS FOR TABULAR DATA USING QUERY FUNCTION OF ALGORITHM 2

Unlike He et al. [2024], in our work, we only watermark n columns of the input table (see Algorithm 2). Thus, during query time, one would need to identify first the columns of the query table that are watermarked (referred to as value columns). As stated in Section 4.2, the query function returns 1 if the z-score of each of the value columns are greater than an input threshold $z_{\text{th}} \geq 0$. Thus, we prove the complexity of decoding Algorithm 2 for $\alpha \leq 0.5$ (which corresponds to $z_{\text{th}} \geq 0$).

Theorem H.2 (Query complexity bound for decoding Algorithm 2)**.** *Given the number of bins b, number of rows m and columns $2n$ of the query table, and assuming that the significance level α for z-test in Section 4.2 is at most 0.5, there is a neural network that can use*

$$O\left(\frac{mnb\log(mnb)\log(1/\epsilon) + \log(1/\delta)}{\epsilon}\right) \tag{77}$$

training data (labeled data consisting of query table and whether or not the table is watermarked according to Algorithm 2), to learn the function which determines whether an input table is watermarked, with error at most $\epsilon > 0$ with respect to 0-1 loss (see Equation (73)), and with probability greater than or equal to $1-\delta$ over the training samples.

Proof. Following the proof of Theorem H.1, we embed data $\mathbf{X} \in [b]^{m\times 2n}$ before feeding into a neural network. First, we embed the labels $k \in [b]$ into vectors, so as to vectorize the inputs $[b]^{m\times 2n}$. To that end, associate the label $k \in [b]$ to the standard unit vector $\mathbf{e}_k \in \mathbb{R}^b$, where $\mathbf{e}_k$ is the vector with 1 at the kth entry and zero elsewhere. Let $\mathbf{X} \in [b]^{m\times 2n}$ be the input data, and let it be mapped to the vector $\oplus_{j=1}^{2n}\oplus_{i=1}^{m}\mathbf{e}_{\mathbf{X}_{ij}}$. Here, each column of each row of $\mathbf{X}$ are vertically stacked on top of each other after vectorizing.

Now, our goal is to show that there is a neural network architecture with appropriate weights, biases, and activation function such that the output of the neural network is the true query function. The true query function outputs 1 *if and only if* the z-score of all the *value* columns is above the threshold z_{th}. As per Section 4.2, we denote the z-score of the j-th column as $z_j = 2(T_j(\mathbf{X})/\sqrt{m}) - \sqrt{m}$, for all $j \in [2n]$. For obtaining the true query function, we need to know which columns correspond to the value columns as well as which bins are watermarked in a given value column. Since we seek to embed the true query function into a neural network, we assume that we know the indices $\mathcal{V}$ corresponding to the value column as well as the true labeling vectors. Algorithm 2, by design, ensures that $\mathcal{V} \subseteq [2n]$ and $|\mathcal{V}| = n$.

The first hidden layer contains a total of $2n$ nodes. Let $\mathbf{w} \in \{0,1\}^b$ denote a candidate vector of labels (1 is interpreted as watermarked, while 0 is interpreted as not watermarked). Observe that $\langle \mathbf{w}, \mathbf{e}_k | \mathbf{w}, \mathbf{e}_k\rangle = \mathbf{w}_k$ for all $k \in [b]$. Furthermore, for all $j \in [n]$, we have $\left\langle \mathbf{w}^{\oplus m}, \oplus_{i=1}^{m}\mathbf{e}_{\mathbf{X}_{ij}} \middle| \mathbf{w}^{\oplus m}, \oplus_{i=1}^{m}\mathbf{e}_{\mathbf{X}_{ij}} \right\rangle = \sum_{i=1}^{m}\mathbf{w}_{\mathbf{X}_{ij}}$. If $j \in \mathcal{V}$ and $\mathbf{w}$ is the correct labeling vector for the jth column, then $\sum_{i=1}^{m}\mathbf{w}_{\mathbf{X}_{ij}} = T_j(\mathbf{X})$. Since $T_j(\mathbf{X}) \in [0,m]$, we add a bias of $\sqrt{m}+1$ so that $T_j(\mathbf{X})+\sqrt{m}+1 > \sqrt{m}$. If, on the other hand, if $j \notin \mathcal{V}$, i.e., we have a key column, then we set the weight $\mathbf{w}$ equal to the zero vector and add a bias of $m+\sqrt{m}+2 > \sqrt{m}$. (We choose such a bias to internally distinguish a key column from a value column, since $T_j(\mathbf{X}) + \sqrt{m} + 1 < m + \sqrt{m} + 2$.) Subsequently, we apply the following piecewise-linear activation function:

$$\psi(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ 1 + \frac{1}{2n} & \text{if } 0 \leq x \leq \sqrt{m} \\ x & \text{if } x > \sqrt{m}. \end{cases} \tag{78}$$

Then, the output of the first hidden layer (after applying the weights, biases, and activation function) is equal to $T_j(\mathbf{X}) + \sqrt{m} + 1$ if $j \in \mathcal{V}$, while it is equal to $m + \sqrt{m} + 2$ if $j \notin \mathcal{V}$.

For the second layer, our goal is to obtain the z-scores for the value columns and compare it to the threshold. To achieve this, we add a weight of $2/\sqrt{m}$ and a bias of $-\sqrt{m} - 2 - 2/\sqrt{m} - z_{\text{th}}$. Applying such a weight an bias to the value column $j \in \mathcal{V}$ gives $(2/\sqrt{m})(T_j(\mathbf{X}) + \sqrt{m} + 1) - \sqrt{m} - 2 - 2/\sqrt{m} - z_{\text{th}} = z_j - z_{\text{th}}$, where z_j is the z-score of the jth column. Now, if $z_j - z_{\text{th}} \leq 0$ (or $z_j \leq z_{\text{th}}$), then the activation function outputs 0. On the other hand, if $z_j - z_{\text{th}} > 0$, then since $z_j \leq \sqrt{m}$ and $z_{\text{th}} \geq 0$ (as $\alpha \leq 0.5$ by assumption), we have $z_j - z_{\text{th}} \leq \sqrt{m}$, so that the activation function outputs $1 + 1/(2n)$. Note that for the *key* columns, we can set the weights as 1 and biases as 0. Then applying the activation function to these nodes, we have the output $m + \sqrt{m} + 2$ as before.

Finally, in the third layer (which is also the final/output layer), for each column corresponding to the values, we apply a weight of 1 and a bias of $-n$. For the key columns, we apply weight and bias equal to 0 (which is equivalent to ignoring the key columns at the final layer). Therefore, the output at the final layer is $\sum_{j \in \mathcal{V}} (1 + 1/(2n)) \mathbf{1}[z_j - z_{\text{th}} > 0] - n$, where $\mathbf{1}[A]$ denotes the indicator function of the event A (i.e., $\mathbf{1}[A] = 1$ if A is true and 0 otherwise). It can be verified that the output is positive if and only if $z_j > z_{\text{th}}$ for all $j \in \mathcal{V}$. Then, we apply the sign function given in (71) to this output. Thus, such a network can learn the true query function required. A schematic of the proposed neural network architecture is given in Figure 6.

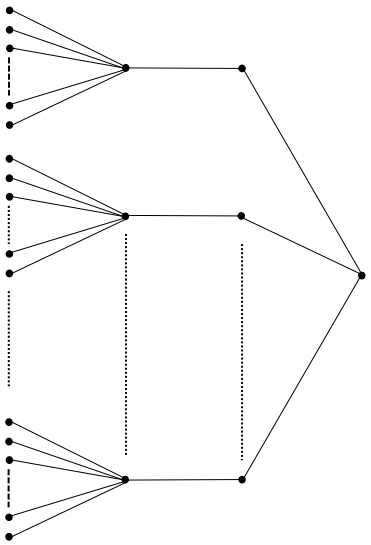

Figure 6: Neural network architecture to embed the problem of learning the query function for dataset watermarked according to Algorithm 2. The activation function is applied at every node, except the nodes in the input and the output layers.

The total number of parameters required to represent the weights and biases before the first layer is $O(mnb)$, before the second layer is $O(n)$, and before the third layer is $O(n)$. Thus we have a total of $O(mnb)$ parameters. We also have 3 layers in the architecture. The effective depth of our neural network is $O(1)$. Then, using [Bartlett et al., 2019, Theorem 6], the VC dimension of the neural network hypothesis class is bounded above by $O(mnb \log(mnb))$. Using [Shalev-Shwartz and Ben-David, 2014, Theorem 6.8],

$$M = O\left(\frac{mnb \log(mnb) \log(1/\epsilon) + \log(1/\delta)}{\epsilon}\right) \tag{79}$$

training samples are sufficient to learn the query function up to error ϵ with respect to the $0-1$-loss with probability greater than or equal to $1 - \delta$ over the training samples.

Now, denote $\mathcal{A}'(\text{training data}, \cdot)$ to be the query function that is learned from the training data. Then, if the distribution of the data is uniform, then $\Pr_{\overline{\mathbf{X}}'}(\mathcal{A}'(\text{training data}, \overline{\mathbf{X}}') \neq Q'(\overline{\mathbf{X}}'))$ is equal to the fraction of indices where the learned query function differs from the true query function. As a result, setting $\epsilon = 0.5/(mnb)$, we obtain the situation where we learn the query function exactly. Thus, we need $O((mnb)^2 \log(mnb) \log(b) + mnb \log(1/\delta))$ samples to learn the query function exactly with probability at least $1 - \delta$ over the training samples. □

Corollary H.3 (Query complexity of row queries to both He et al. [2024] and Algorithm 2)**.** *For queries of the form* $\mathbf{X} \in \mathbb{R}^{m \times 1}$ *and query function that computes* $z = 2T(\mathbf{X})/\sqrt{m} - \sqrt{m}$ *or any linear function of* $T(\mathbf{X})$*, there is a neural network that can learn this query function up to error* ϵ *in* 0-1 *loss using* $O\left(\frac{mb\log(1/\epsilon)+\log(1/\delta)}{\epsilon}\right)$ *samples with probability* $1-\delta$ *over the training samples.*

Proof. Using $n = 1$ in Theorem H.2 gives us the result. For He et al. [2024], observe that when a single row is input, $T_j(\mathbf{X})$ for all $j \in n$ is in $\{0, 1\}$. As such, the statistical test uses z-score tests on the standard normal distribution, and not the χ^2 test. The hypothesis class corresponding to the neural architecture as defined in Theorem H.2 is thus PAC learnable for this problem, and so the query complexity bound follows by setting $n = 1$. □

H.3 A LOWER BOUND FOR ROW QUERIES TO He et al. [2024] GIVEN THE z-SCORES

Consider the case that the continuous space in $(0, 1]$ is uniformly discretized using bins of size $1/b$, i.e., construct bins in $(0, 1]$ as – $\left\{(0, \frac{1}{b}], (\frac{1}{b}, \frac{2}{b}], \ldots, (\frac{b-1}{b}, 1]\right\}$. Then each of these bins are labeled $\{0, 1\}$ with some probability $1 - p$ and p respectively. We save this data-structure, and allow anyone to query it with $S \in (0, 1]^m$, responding with the z-score (Equation 3) and whether the data is watermarked or not. In this section, we want to bound the minimum number of queries one can make to the model and estimate the watermarking scheme with high accuracy. Specifically we ask:

When the number of bins b is known, what is the query complexity to correctly identify all red and green intervals used for watermarking?

Observe that for a row, the z-score is computed as: $2\sqrt{m}\left(\frac{T}{m} - \frac{1}{2}\right)$, where T is the number of values in the row falling in the bins with label 1.

When b is known to the adversary, the problem can be reduced to the 2-color Mastermind game. In 2-color mastermind the codemaker creates a secret code using a sequence of colored pegs, and the codebreaker guesses the sequence. At each query by the codebreaker, the codemaker provides the number of pegs they correctly guessed. As such, given z is the output of each query, one can easily compute the number of values in each bin $\{0, 1\}$. Thus, given any b, one can reduce this problem to the 2-color Mastermind game. This specific form of Mastermind has been extensively studied in the literature [Chvátal, 1983, Knuth, 1976]. In particular, with 2 possible colors: red and green, the maximum number of row queries needed to recover the exact watermarking scheme is $\Theta\left(\frac{b\log(2)}{\log(b)}\right)$. Hence, the 'red-green' watermarking scheme by He et al. [2024] can be learned by an adversary making $\Theta\left(\frac{b\log(2)}{\log(b)}\right)$ queries to the detector. Note that this lower bound does not directly apply to the upper bounds derived using VC theory in Appendix H.1 and Appendix H.2, since here we assume access to the z-score for each query.