HashMark: Watermarking Tabular/Synthetic Data For Machine Learning Via Cryptographic Hash Functions

Anonymous Author(s)
Affiliation
Address
email

Abstract

Watermarking is a critical tool for protecting datasets against malicious or unauthorized use, yet existing methods often face limitations in data type support, fidelity preservation, and detection efficiency. In this work, we introduce HashMark, a novel and versatile watermarking scheme for tabular datasets, including synthetic data, without imposing restrictions on data types. At its core, HashMark employs a cryptographic hash function to map *any* data into binary values, enabling efficient and robust watermark embedding. Our design generalizes and simplifies some prior approaches, such as the recent works Ngo et al. (arXiv 2024) and TabularMark (ACM CCS 2024), while addressing their key shortcomings. Unlike Ngo et al., HashMark supports categorical and mixed-type data with a unified framework. Compared to TabularMark, it enables efficient watermark detection without requiring access to the original dataset. Further, unlike TabularMark, we present experiments for categorical data. Finally, we run experiments comparing the accuracy of synthetically generated data and watermarked, synthetic data on three classifiers over several datasets using three approaches for generating synthetic data. These experiments clearly demonstrate a negligible impact on utility for intended machine learning tasks when HashMark is used.

1 Introduction

As financial institutions increasingly rely on data-driven systems for risk assessment, fraud detection, regulatory compliance, and AI-driven decision-making, ensuring data integrity, provenance, and ownership is paramount. Data watermarking—the practice of embedding imperceptible markers or identifiers into datasets—offers a powerful mechanism for protecting sensitive financial data, establishing ownership, and verifying authenticity across complex data pipelines. In contexts where financial data is shared with third parties, sold to analytics providers, or used to train machine learning models, watermarking provides a means to trace data lineage, deter unauthorized use, and ensure accountability. Moreover, with the growing adoption of generative models and synthetic data in finance—for tasks such as scenario simulation, stress testing, and customer behavior modeling—watermarking plays a critical role in guaranteeing the traceability and responsible use of AI-generated financial datasets.

Previous research on watermarking typically focuses on image, audio, or text data [3, 35, 39, 42, 44], with less attention given to tabular data, one of the most common and essential data formats in machine learning. Tabular data presents unique challenges for watermarking: (1) Precise values lack perceptual redundancy, making even minor changes impactful; (2) Mixed data types (categorical, numerical) require tailored strategies; (3) Resilience is needed against insertions, deletions, and foreign key modifications. Existing attempts to provide watermarking for tabular data often focus solely on relational data [2, 17, 18, 22, 25, 26, 33, 34]. Existing methods have proposed watermarking

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

Table 1: Comparison of HashMark with prior works (transposed). Detection Cost refers to the information needed to detect the watermark efficiently. "# Modification" refers to the number of cells that need to be modified to embed the watermark.

	Ngo et al.	Zheng et al.	HashMark_1	HashMark_2
# Modification	All	All	$\mathbf{\Theta(1)}$	All
Fidelity	High	High	V.High	High
Deletions	Allowed	Allowed	Limited	**Allowed**
Permutations	Allowed	Allowed	Limited	**Allowed**
Data Types	Numerical	Any*	**Any**	**Any**
Detection Cost	High	V.High	**V.Low**	Low

techniques that either alter specific data points or embed identifiers at a statistical level. Only recently have watermarking approaches specifically designed for tabular data been proposed. These include the works of [16], [43], and [29]. However, these approaches often face challenges related to computational complexity, scalability, and storage requirements.

Our Motivation. Watermarking tabular data is crucial in maintaining data provenance within large organizations, where information flows across multiple departments (especially in large financial organizations) and systems in non-adversarial settings. In such environments, employees typically do not attempt to remove watermarks, which enables effective tracking of data lineage, ensures integrity, and facilitates compliance with internal policies and regulatory requirements. By embedding identifiable markers in datasets, organizations can monitor data movement, quickly trace discrepancies, and uphold accountability throughout the data lifecycle, which is essential for informed decision-making and trust in data-driven processes. The growth in synthetic data also adds another dimension to the problem, as enterprises must effectively identify and distinguish synthetic data from original data. Note that synthetic data is an effective tool for producing a dataset that protects the privacy of confidential data while still allowing for downstream utility, similar to the original data.

While no watermarking scheme is entirely immune to removal (most recently [40] showed that under even mild assumptions, strong LLM watermarking is impossible)—just as encryption can be broken, DRM bypassed, or licenses violated—the value lies in raising the cost of misuse and enabling accountability in practical, non-adversarial scenarios. Tabular data remains a fundamental medium for information sharing, particularly within enterprises, necessitating continual advancements in watermarking techniques. Our research tackles essential shortcomings in previous studies. By advancing robustness and applicability, we contribute to a framework that strengthens data governance and mitigates unauthorized use.

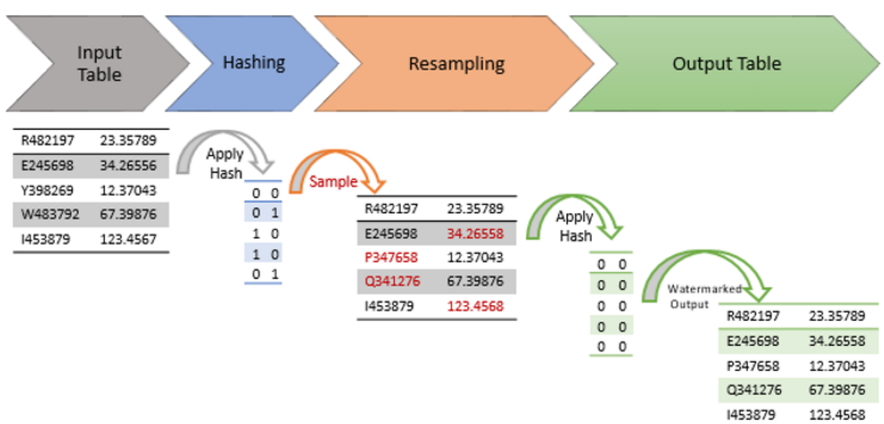


Figure 1: HashMark_2: On the left is the source input table, to be watermarked, containing cells of two columns - one text and the other numerical. After applying the hash function to each cell, the hashed values are shown next. In the middle, we show how values are adjusted to be able to hash to 0. For text data, we replace it with a new value, and for numerical data, we add in the smallest decimal place. On the right is the watermark embedded table where all cells hash to 0.

1.1 Our Contributions

We introduce HashMark, a suite of simple yet powerful watermarking protocols for tabular datasets. Our approach embeds bits into select table cells using a *cryptographic, seeded hash function*, ensuring that the output looks uniformly random without the knowledge of the seed. A hash function is versatile in its agnosticism on the input data type, working over numeric and alphanumeric inputs.

We present two variants, $\mathsf{HashMark}_1$ and $\mathsf{HashMark}_2$, each offering unique properties. In both schemes, we map cell contents to a target bit (0 or 1) via the seeded hash function. If the cell content does not map to the target bit, we carefully modify the cell values while preserving the dataset's fidelity. For numerical values, we make minimal perturbations (e.g., incrementing by 10^{-c}). For alphanumeric values, we apply rejection sampling from the original distribution.

- Thus, HashMark offers *high fidelity* as the changes in the dataset to embed the watermark are minimal in the case of numerical values (due to small perturbation) and none in the case of alphanumeric values (due to rejection sampling from the *same* distribution).
- Meanwhile, the *detection cost is low* in HashMark as it only requires the knowledge of the seed of the hash function, an artifact of our simpler design. Meanwhile, [29] requires remembering how the columns in the dataset are paired. [43] requires the knowledge of the entire source dataset to detect the watermark.
- HashMark can *support any data*, as explained above. Meanwhile, [29] naively cannot support categorical data. While [43] claims to support any data, their exposition does not clarify how their approach translates to textual data[1] (marked as Any*) in Table 1.

$\mathsf{HashMark}_2$. Figure 1 pictorially represents $\mathsf{HashMark}_2$. $\mathsf{HashMark}_2$ embeds the same target bit (say 0) at *all* positions in the dataset. It uses the hash function for the binary mapping and then applies the above-outlined "adjustment" procedure to ensure that every cell maps to 0 under the seeded hash function. This is akin to prior approaches of [29] and [43][2]. Our detection algorithm relies on a statistical test.

$\mathsf{HashMark}_1$. For static datasets (e.g., unique IDs, timestamps, categorical labels), $\mathsf{HashMark}_1$ modifies only a constant $\ell \ll N$ cells, ensuring high fidelity. It uses two PRGs: G_1 derives ℓ pseudrandom bits, while G_2 selects ℓ cell locations. Each of the ℓ cell locations is adjusted until it hashes to the desired bit produced earlier by G_1. Detection verifies these bits using the same PRGs. Its advantages include: minimal distortion (only ℓ cells altered), and security relies on the pseudorandomness of G_1 and G_2. Note that minor permutations or deletions of rows compromise detection since they disrupt cell positioning. Partial robustness to these changes is possible if watermarking is restricted to fixed columns.

Additionally,

- We theoretically analyze fidelity and model the watermark removal process.
- Extensive experiments validate our approach. For $\mathsf{HashMark}_1$, we show high embedding efficiency while maintaining classification accuracy across three classifiers. For $\mathsf{HashMark}_2$, we evaluate both Gaussian and synthetic datasets, analyzing fidelity through z-score, mean-squared error, and robustness to noise. Results confirm that the watermarked synthetic data has a negligible impact on classification accuracy. We employ four datasets to train synthesizers, produce synthetic data, and watermark this synthetic data before running two classifiers. Additionally, we are the first to study watermarking for alphanumeric columns concretely. While prior work TabularMark [43] claimed to offer support for alphanumeric columns, the details were underspecified.

2 HashMark: Element Wise Tabular Watermarking

[1] [43] focuses on categorical data (e.g., education level, marital status), their watermarking distorts integer-based distributions by adding floating-point perturbations, harming utility. Restricting to integer-based perturbations could lead to some gaps in the range of the column. We argue that such columns should not be watermarked. Further, they do not support unrestricted alphanumeric data (e.g., ASINs) or test such cases.

[2] Indeed, one can conceivably correlate our binary hashing approach with the red-green paradigm adopted by these works. However, our construction vastly simplifies their approaches.

Algorithm 1 Embedding Algorithm

Input: Sampling Algorithm for Dataset $\mathcal{D}$ $\mathsf{Generate}$
Secret Seed seed
Number of Rows: ℓ
Associated Distribution: ρ
Column $column$ of dataset $\mathbf{X}$
$seed \xleftarrow{\$} \mathcal{S}$ *//S is the seed space of the hash function.*

for $i = 1$ **to** ℓ **do**
  while $\mathcal{H}(seed, \mathcal{D}[i]) \neq 0$ **do**
    $new\_value \leftarrow \mathsf{Generate}(\rho, \mathcal{D}[i])$ *//Addnl. parameters could include t for threshold constrained sampling.*
    $\mathcal{D}[i] \leftarrow new\_value$
  end while
end for

At its core, any watermarking approach needs to ensure that the utility of the data is preserved even after embedding the watermark. Furthermore, the detectability of the watermark is preserved even after modification by both adversarial and honest actions. We have two constructions $\mathsf{HashMark}_1, \mathsf{HashMark}_2$ with various properties and an implicit trade-off.

However, before examining the constructions, it is instructive to consider the commonalities. Both the constructions will rely on applying a seeded hash function $\mathcal{H}$ that can take any inputs and produce an output bit. Such a binary hash function enables us to map any cell (numerical, textual, categorical, etc.) to either 0 or 1, depending on the function's description. They will also rely on modifying a cell's contents through invoking the function Generate (until it satisfies some $\mathcal{H}$-based property). The question remains of how to instantiate this function. Algorithm 1 provides a template for how to approach this embedding of the watermark. Due to space constraints, we defer an expanded discussion to the appendix (Section D.2), but summarize below:

- **Caveat of Rejection Sampling:** Columns with small fixed ranges (e.g., marital status, education, salary tiers) should not be watermarked, since rejection sampling can skew distributions and harm utility. Treat all values in such columns as valid (always mapping to the desired bit).
- **Numerical Values:** Perturb values slightly by adding 10^{-c}, where c is a scheme parameter, until the resulting value hashes to the desired bit. Fidelity bound (See Theorem 1) $\mathbb{E}[||\mathbf{X} - \mathbf{X}_w||_\infty] \leq (\ln N + 2) \cdot 10^{-c}$. Supports truncation up to b decimal places.
- **Alphanumeric/Textual Data:** Use rejection sampling to resample values until they hash to the desired bit. Fidelity guarantee via Jensen-Shannon Divergence (See Theorem 2): $JSD(\rho||\rho') \approx 0.215$.
- **Preserving Correlations:** Sample rows from the learned distribution ρ (e.g., synthetic data generator) to preserve correlations. Reject and resample rows that don't satisfy watermarking constraints. Satisfying all columns can be costly (2^n time). Instead, use a threshold t: accept rows where at least t out of n cells meet the constraint; adjust detectability accordingly.

3 Conclusion

We present HashMark, a hash-based framework for watermarking financial datasets, strengthening data integrity, auditability, and provenance in AI-driven financial systems. HashMark supports both numerical and categorical attributes common in finance (e.g., transaction records, customer profiles, risk metrics), improves upon prior methods [16, 29, 43], and enables secure, efficient, and compliant data sharing. Beyond protecting sensitive financial information, HashMark is particularly suited for watermarking synthetic financial data used in stress testing, fraud detection, and regulatory reporting, thereby facilitating accountability and regulatory compliance.

References

[1] Scott Aaronson. ‘reform’ ai alignment with scott aaronson. AXRP - The AI X-risk Research Podcast, 2023.

[2] Rakesh Agrawal and Jerry Kiernan. Watermarking relational databases. In *Proceedings of the 28th International Conference on Very Large Data Bases*, VLDB ’02, page 155–166. VLDB Endowment, 2002.

[3] Sajjad Bagheri Baba Ahmadi, Gongxuan Zhang, Mahdi Rabbani, Lynda Boukela, and Hamed Jelodar. An intelligent and blind dual color image watermarking for authentication and copyright protection. *Applied Intelligence*, 51(3):1701–1732, March 2021.

[4] E. Alpaydin and Fevzi. Alimoglu. Pen-Based Recognition of Handwritten Digits. UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5MG6K.

[5] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, September 2004.

[6] David Byrd, Vaikkunth Mugunthan, Antigoni Polychroniadou, and Tucker Balch. Collusion resistant federated learning with oblivious distributed differential privacy. In *Proceedings of the Third ACM International Conference on AI in Finance*, ICAIF ’22, page 114–122, New York, NY, USA, 2022. Association for Computing Machinery.

[7] Miranda Christ and Sam Gunn. Pseudorandom error-correcting codes. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024*, pages 325–347, Cham, 2024. Springer Nature Switzerland.

[8] Miranda Christ, Sam Gunn, and Or Zamir. Undetectable watermarks for language models. In *ICLR*, 2024.

[9] Wikipedia Contributions. Z-test, 2025. Accessed: 2025-01-30.

[10] Dhruvil Dave. Github commit messages dataset. Kaggle Dataset, 2023.

[11] Jaiden Fairoze, Sanjam Garg, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, and Mingyuan Wang. Publicly-detectable watermarking for language models. *IACR Communications in Cryptology*, 1(4), 2025.

[12] J.L. Fleiss, B. Levin, and M.C. Paik. *Statistical Methods for Rates and Proportions*. Wiley Series in Probability and Statistics. Wiley, 2013.

[13] Eva Giboulot and Teddy Furon. Watermax: breaking the LLM watermark detectability-robustness-quality trade-off. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[14] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O’Reilly Media, 2nd edition, 2019.

[15] 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, 2011.

[16] Hengzhi He, Peiyu Yu, Junpeng Ren, Ying Nian Wu, and Guang Cheng. Watermarking generative tabular data, 2024.

[17] Donghui Hu, Dan Zhao, and Shuli Zheng. A new robust approach for reversible database watermarking with distortion control. *IEEE Transactions on Knowledge and Data Engineering*, 31(6):1024–1037, 2018.

[18] Min-Shiang Hwang, Ming-Ru Xie, and Chia-Chun Wu. A reversible hiding technique using lsb matching for relational databases. *Informatica*, 31(3):481–497, 2020.

[19] Bargav Jayaraman, Lingxiao Wang, David Evans, and Quanquan Gu. Distributed learning without distress: privacy-preserving empirical risk minimization. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 6346–6357, Red Hook, NY, USA, 2018. Curran Associates Inc.

[20] Brian Johnson. Wilt. UCI Machine Learning Repository, 2013. DOI: https://doi.org/10.24432/C5KS4M.

[21] Nurul Shamimi Kamaruddin, Amirrudin Kamsin, Lip Yee Por, and Hameedur Rahman. A review of text watermarking: Theory, methods, and applications. *IEEE Access*, 6:8011–8028, 2018.

[22] Muhammad Kamran, Sabah Suhail, and Muddassar 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.

[23] R. Kelley Pace and Ronald Barry. Sparse spatial autoregressions. *Statistics and Probability Letters*, 33(3):291–297, 1997.

[24] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 17061–17084. PMLR, 23–29 Jul 2023.

[25] Wenling Li, Ning Li, Jianen Yan, Zhaoxin Zhang, Ping Yu, and Gang Long. Secure and high-quality watermarking algorithms for relational database based on semantic. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[26] Chia-Chen Lin, Thai-Son Nguyen, and Chin-Chen Chang. Lrw-crdb: Lossless robust watermarking scheme for categorical relational databases. *Symmetry*, 13(11):2191, 2021.

[27] J. Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991.

[28] Guido Masarotto and Cristiano Varin. Gaussian copula marginal regression. *Electronic Journal of Statistics*, 6(none):1517 – 1549, 2012.

[29] Dung Daniel Ngo, Daniel Scott, Saheed Obitayo, Vamsi K. Potluru, and Manuela Veloso. Adaptive and robust watermark for generative tabular data. *Statistical Frontiers in LLMs and Foundation Models at NeurIPS'24*, abs/2409.14700, 2024.

[30] Neha Patki, Roy Wedge, and Kalyan Veeramachaneni. The synthetic data vault. In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 399–410, 2016.

[31] PromptCloud. Amazon product dataset 2020. Kaggle Dataset, 2020.

[32] Cemal Okan Sakar, Suleyman Olcay Polat, Mete Katircioglu, and Yomi Kastro. Real-time prediction of online shoppers' purchasing intention using multilayer perceptron and lstm recurrent neural networks. *Neural Computing and Applications*, 31:6893 – 6908, 2018.

[33] Mohamed Shehab, Elisa Bertino, and Arif Ghafoor. Watermarking relational databases using optimization-based techniques. *IEEE Transactions on Knowledge and Data Engineering*, 20(1):116–129, 2008.

[34] 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)*, pages 98–109, New York, NY, USA, 2003. Association for Computing Machinery.

[35] Mingtian Tan, Tianhao Wang, and Somesh Jha. A somewhat robust image watermark against diffusion-based editing models, 2023.

[36] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, and Paul van Mulbregt. SciPy 1.0: Fundamental algorithms for scientific computing in python, 2020.

[37] X. Xiao, X. Sun, and M. Chen. Second-lsb-dependent robust watermarking for relational database. In *Third International Symposium on Information Assurance and Security (IAS)*, pages 292–300, 2007.

[38] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. *Modeling tabular data using conditional GAN*. Curran Associates Inc., Red Hook, NY, USA, 2019.

[39] M. Yamni, H. Karmouni, M. Sayyouri, and H. Qjidaa. Efficient watermarking algorithm for digital audio/speech signal. *Digital Signal Processing*, 120:103251, 2022.

[40] Hanlin Zhang, Benjamin L. Edelman, Danilo Francati, Daniele Venturi, Giuseppe Ateniese, and Boaz Barak. Watermarks in the sand: impossibility of strong watermarking for language models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[41] Hanlin Zhang, Benjamin L. Edelman, Danilo Francati, Daniele Venturi, Giuseppe Ateniese, and Boaz Barak. Watermarks in the sand: impossibility of strong watermarking for language models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[42] Xiaorui Zhang, Xun Sun, Xingming Sun, Wei Sun, and Sunil Kumar Jha. Robust reversible audio watermarking scheme for telemedicine and privacy protection. *Computers, Materials & Continua*, 71(2):3035–3050, 2022.

[43] Yihao Zheng, Haocheng Xia, Junyuan Pang, Jinfei Liu, Kui Ren, Lingyang Chu, Yang Cao, and Li Xiong. Tabularmark: Watermarking tabular datasets for machine learning. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, CCS '24, page 3570–3584, New York, NY, USA, 2024. Association for Computing Machinery.

[44] Xin Zhong, Pei-Chi Huang, Spyridon Mastorakis, and Frank Y. Shih. An automated and robust image watermarking scheme based on deep neural networks. *IEEE Transactions on Multimedia*, 23:1951–1961, 2021.

A Related Work

Watermarking Tabular Data. Watermarking tabular data has been extensively studied. [2] pioneered a scheme embedding watermarks in the least significant bit of specific cells using hash values based on primary and private keys. Subsequent works by [37] and [15] improved this by embedding multiple bits. Another approach embeds watermarks in statistical properties. [34] introduced a method that partitions dataset rows and modifies subset statistics, later refined by Shehab et al. [33] to resist insertion and deletion attacks using optimized partitioning and hash-based embedding. Their approach, however, relies on assumptions about data distribution and primary keys.

Inspired by watermarking techniques in large language models [1, 21, 24], [16], [29], and [43] proposed watermarking schemes for generative tabular data using red-green interval partitioning.

[16] introduced a data binning approach, ensuring values lie near green intervals and using statistical hypothesis testing for detection. However, assuming continuous distributions makes it vulnerable to feature selection and truncation attacks. [29] paired columns into key-value sets, deriving a seed from the key column to generate bins for the value column. Entries falling in red bins were resampled from green bins. While novel, this method suffers from two key weaknesses: (i) detection requires prior knowledge of the column pairing or an exhaustive search across all pairs, and (ii) relying on key column-derived seeds introduces low entropy, weakening the pseudorandomness of bin assignments and potentially compromising security. It is important to note that even with knowledge of column pairing, any deletion of rows will trigger an error when calculating the key column-derived seed, which is not explored or discussed in the paper. [43] took a similar approach, embedding watermarks as additive noise within predefined bins. They assumed noise follows a bounded range $[-p, p]$, partitioned into red and green bins, with watermarking achieved by sampling noise only from green bins. Despite robustness claims and categorical feature support, their method has several limitations. First, detection requires access to the original dataset, making watermark verification infeasible in practical scenarios where datasets are modified or shuffled. Second, row-matching under permutation increases detection complexity. Finally, their claimed support for categorical data is unclear and lacks empirical validation - (a) Their protocol description focuses only on categorical data, i.e., those with a fixed range (e.g., education level, employee designation, marital status, etc.). They suggest encoding it first as integers and then applying their embedding techniques. However, this method is flawed because these differences often result in floating-point values, distorting the expected integer-based distribution. Restricting differences to integers could also leave gaps in the data (by omitting particular values from the range), harming its utility. Instead, we argue against watermarking such columns altogether, and (b) it does not address unrestricted categorical data (e.g., alphanumeric ASINs) or provide experiments for such cases. The above is summarized in Table 1.

Watermarking for LLMs. Many watermarking schemes for LLMs take advantage of the sampling algorithm that generates each token of an LLM output. [8] observed that these LLM output tokens correlate with the randomness used in the token sampling algorithm. This correlation is efficiently communicable for many LLM outputs by replacing this randomness with cryptographic pseudorandomness. Subsequent works [11, 7] have built upon this idea by incorporating error correction and public identifiability into these watermarks. However, robustness remains a persistent issue for this line of work, and a recent impossibility result [41] demonstrated that an adversary that can efficiently perturb or resample the output can always remove a watermark. Another line of work, which has been the source of inspiration for more recent watermarking schemes for tabular data, include [1, 21, 24]. [24] introduced the red-green list paradigm, forming the basis of several works [16, 43, 29]. More recently, [13] improved on the works employing the red-green list paradigm.

B Preliminaries

Notations. For $n \in \mathbb{N}^+$, we denote by $[n]$ the set $\{1, \ldots, n\}$. For a set X, we denote by $x \xleftarrow{\$} X$ that a value x is sampled uniformly at random from X.

Seeded Hash Function. A function $\mathcal{H} : \mathcal{S} \times \mathcal{X} \to \mathcal{Y}$ is a hash function, modeled as a random oracle, if the computation of $\mathcal{H}(S, X)$ for a random $S \xleftarrow{\$} \mathcal{S}$ and any $X \in \mathcal{X}$ is indistinguishable from $Y \xleftarrow{\$} \mathcal{Y}$. In our application, we will suppress the presence of the seed distribution $\mathcal{S}$ and we will set $\mathcal{Y} := \{0, 1\}$.

C Problem Formulation

Our dataset is a matrix $\mathbf{X}$ of dimension $m \times n$. It is important to stress that $\mathbf{X}$ contains both numerical values, alphabetical and alphanumeric. We assume each column i contains m i.i.d points from a distribution ρ_i. For simplicity, we will define a function $\mathsf{Generate}$ that takes as input a probability distribution ρ_i and a sample p_i from the distribution ρ_i to produce a new sample p_i'. When ρ_i is undefined, we can still extract a new sample using just p_i. The goal is to generate a watermarked $\mathbf{X}_w$ with the following properties:

Fidelity : The watermarked dataset $\mathbf{X}_w$ is "close" to the original data set $\mathbf{X}$. In our approach for numerical data, we show that $\mathbf{X}_w$ and $\mathbf{X}$ are close in the L_∞ distance. See Theorem 1.

Detectability : Efficient testing can reliably identify the watermarking. In our first variant, we will rely on cryptographic properties to ensure detection, while in the second variant, we will rely on statistical testing.

Robustness : The watermarked dataset $\mathbf{X}_w$ is resistant to various perturbations observed in common usage. Some of these include removing or permutations of rows and columns and modifying cell content.

Utility : The watermarked dataset $\mathbf{X}_w$ is still useful for intended downstream tasks such as machine learning tasks. Through empirical testing, we will show that there is a negligible difference in accuracy.

D Dataset Details

Wilt. Wilt [20] is the public dataset from the UCI Machine Learning Repository from a remote sensing study on detecting diseased trees in satellite imagery. It comprises 4,839 image segments with spectral and texture features from Quickbird multispectral and panchromatic bands. The dataset includes six numerical and categorical attributes and a binary classification task: identifying trees as wilted or healthy. We generate synthetic datasets. There are 4839 records with 6 features (including the target) and 2 classes. This dataset is licensed under a Creative Commons Attribution 4.0 International (CC BY 4.0) license.

California Housing Prices. The California Housing Prices dataset [23, 14], sourced from the 1990 U.S. Census, contains 20,640 records with 10 socio-economic and geographical attributes influencing housing prices. It has a multi-target label indicating proximity to the ocean, making it a multi-class classification problem. It has 5 classes. This dataset is licensed under Apache License Version 2.0.

HOG. The HOG feature dataset [4] is generated with the histogram of oriented gradients (HOG) features extracted from the digits dataset, combined with their categories. There are 16 features, 10992 records, and 10 classes. This dataset is licensed under a Creative Commons Attribution 4.0 International (CC BY 4.0) license.

Shoppers Dataset. The shoppers dataset [32] aimed to capture the shoppers purchasing intent. There are 12,330 records with 18 attributes with two classes. The dataset is licensed under a Creative Commons Attribution 4.0 International (CC BY 4.0) license.

Amazon ASINs. We used the Amazon Product Details Dataset [31]. For our experiments, we parsed the dataset only to extract the unique identifiers for Amazon products, generating 30,000 actual ASINs. This dataset is licensed under CC0.

Gitcommit Hashes. We used the Gitcommit Messages dataset [10]. It contains 4.3 million records, from which we only extracted the hashes for the gitcommit messages. The dataset is licensed under the Open Data Commons Attribution License (ODC-By) v1.0.

Algorithm 2 HashMark$_1$ Embedding Algorithm

Input: Original Dataset $\mathbf{X}$ of dimension $m \times n$
Probability Distributions $\rho_1, \ldots, \rho_n$.
PRG $G_1 : \mathcal{X}_1 \to \{0,1\}^\ell$
PRG $G_2 : \mathcal{X}_2 \to [m]^\ell \times [n]^\ell$
$X_1 \xleftarrow{\$} \mathcal{X}_1, X_2 \xleftarrow{\$} \mathcal{X}_2$
$\{bit_i\}_{i=1}^{\ell} \xleftarrow{\$} G_1(X_1)$
$\{(row_i, col_i)\}_{i=1}^{\ell} \leftarrow G_2(X_2)$
$seed \xleftarrow{\$} \mathcal{S}$ *//$\mathcal{S}$ is the seed space of $\mathcal{H}$*

for $i = 1$ **to** ℓ **do**
  while $\mathcal{H}(seed, \mathbf{X}[row_i, col_i]) \neq bit_i$ **do**
    $new\_value \xleftarrow{\$}$ Generate$(\rho_i, \mathbf{X}[row_i, col_i])$
    $\mathbf{X}[row_i, col_i] \leftarrow new\_value$
  end while
end for

D.1 HashMark$_1$: Embedding Pseudorandom Bits

We begin by describing our first approach to watermarking. This approach ensures high fidelity and detectability but suffers from issues when it comes to robustness. The embedding algorithm is formally defined in Algorithm 2. We start with an original dataset $\mathbf{X}$ of dimension $m \times n$. The idea is to sample ℓ pseudorandom bits. Let us call it $bit_1, \ldots, bit_\ell$. Additionally, we also sample ℓ cells defined by (row_i, col_i) in $\mathbf{X}$. By modifying the cell content suitably, we ensure that $\mathcal{H}(\mathbf{X}[row_i, col_i]) = bit_i$.

Detecting HashMark$_1$

To detect, the algorithm needs:

- Knowledge of X_1 to retrieve the original binary string of $bit_1, \ldots, bit_\ell$.
- Knowledge of X_2 to first identify the target cells (row_i, col_i), and then using $\mathcal{H}$ to retrieve $bit'_1, \ldots, bit'_\ell$.
- The watermark detection is successful iff $(bit_1, \ldots, bit_\ell) = (bit'_1, \ldots, bit'_\ell)$

However, this scheme is low-robust because the detection algorithm critically relies on extracting the cell where the watermark was embedded. This would be meaningless if the first row (or the first column) were removed. The benefit of this approach is that only ℓ of the spots are touched, which is a tunable parameter. This ensures very high fidelity and utility. The detectability is also reducible to the hardness of the underlying cryptographic primitives (and does not rely on a statistical measure).

D.2 Defining Generate

The crux of our construction is instantiating the function Generate that helps modify the content of the dataset to satisfy the hashing requirement. In this section, we focus on defining this function along with some optimizations. However, before we proceed, we must discuss a caveat to our approach. This is a limitation of rejection-sampling-based approaches. Let C be a column with a fixed range. Some examples of such columns include marital status, education level, designation at a company, and base salary tiers at a company, among others. If one were to apply a hash function, mapping elements in the range of 0 or 1, some elements in the range might be hashed to an undesired bit. The ensuing watermarked dataset will be constrained to remove these elements from the range, resulting in a skewed distribution, which will prevent utility. Therefore, it is essential not to embed the watermark in these columns, as this could skew the resulting distribution. In other words, we consider every element in the range to be "valid," i.e., hashing to the desired bit.

In the ensuing discussion, we focus solely on generating values for the remaining attributes/columns. We will focus on embedding the watermark and later define fidelity, i.e., how close the watermarked distribution is to the un-watermarked one. The proofs of the following are deferred to Section G in the appendix.

Numerical Values. Suppose a column C consists of numerical data, specifically floating-point values. In that case, the generate function can take the old value and add 10^{-c} for some constant c that is a scheme parameter. This ensures that the perturbation does not adversely impact the fidelity. Formally, we have the following theoretical guarantee, as measured by the expected difference in L_∞ between the unwatermarked and watermarked distributions.

Theorem 1. *Let* $\mathbf{X}$ *be the original dataset and* $\mathbf{X}_w$ *be the watermarked dataset of size* N *where* $x'_i \in \mathbf{X}_w$ *is generated as follows:*

$$x'_i = x_i + k_i \cdot 10^{-c},$$

where $k_i = \min\{k \geq 0 \mid \mathcal{H}(x_i + k \cdot 10^{-c}) = 0$*,* $\mathcal{H}$ *is a seeded hash function as defined before, and* $c \geq 0$ *is some integer. Then,*

$$\mathbb{E}[||\mathbf{X} - \mathbf{X}_w||_\infty] \leq (\ln N + 2) \cdot 10^{-c}$$

Our approach can be easily extended to support truncation up to b decimals place if only the value until the first b decimal places are included in the input to $\mathcal{H}$.

Alphanumeric/Textual Data. In the case of textual data, the generate function can reject and re-sample from the underlying distribution for the feature ρ_i. Then, one can measure the fidelity of the watermarked dataset by measuring the Jensen-Shannon Divergence [27] between the watermarked and the un-watermarked dataset. Formally, we get the following theoretical guarantee:

Theorem 2. *Let* ρ *be the distribution of an alphanumeric column where we embed the watermark. Let* ρ' *be the modified distribution consisting only of those values that hash to 0. Then, the Jensen-Shannon Divergence is:*

$$JSD(\rho||\rho') = \frac{3}{4}\log(\frac{4}{3}) \approx 0.215$$

Preserving Correlations. Datasets often contain correlations between various features or attributes. Any watermarking approach should ensure that these correlations are preserved. Rejection sampling column-wise can often lead to a loss of such correlations. We now detail how to preserve correlations.

- Let ρ be a probability distribution that defines the underlying dataset. This can contain both categorical (aka alphanumeric values) and numerical values. For example, a synthetic data generation algorithm (such as the ones employed in our experiments) is trained on a source (i.e., the original dataset), which yields a distribution ρ from which one can sample as many rows as needed. These synthetic data algorithms have been experimentally shown to be close to the original dataset for various machine learning tasks, serving as a heuristic proof of correlation preservation.
- Let $R \xleftarrow{\$} \rho$ be a row sampled from this distribution. Further, let this row R be such that there exist cells that do not map to the desired bit.
- We can now reject R and resample from ρ until the sampled row satisfies the required constraint. However, such rejection and resampling until *every* cell maps to the desired bit can be computationally expensive. For n columns, this can take 2^n time. Instead, one can choose a threshold t such that if t of the n cells in a row R map to the desired bit, it is marked as accepted. The detectability threshold can be suitably set to account for this modification.

D.3 $\mathsf{HashMark}_2$: Global Embedding

Unlike $\mathsf{HashMark}_1$, $\mathsf{HashMark}_2$ is more resilient to various perturbations and cell modification. The embedding approach is visually represented in Figure 1 and described in Algorithm 1. The crux of the strategy is to embed a global bit (say 0) in *every* cell of the dataset $\mathbf{X}$ using a binary hash function $\mathcal{H}$—consequently, a watermarked table to have more values that hash to 0 than an unwatermarked table. Detection is performed by using the secret description of the hash function to hash the data and count the number of cells that map to zero. Additional methods can allow the user to check only a subset of locations, making a slight skew more pronounced. This approach has the versatility of embedding a watermark in an existing dataset or generating a watermarked dataset at the source. The latter is a setting suitable for synthetic data.

Detecting $\mathsf{HashMark}_2$**.** To detect $\mathsf{HashMark}_2$, we use a one-proportion z-test [12], which is a statistical test used to determine whether the single sample rate, for example, the success rate in the number of entries that map to 0, is significantly different from a hypothesized population rate. We define the null hypothesis as:

$$H_0 : \text{Dataset } \mathbf{X} \text{ is not watermarked}$$

However, we note that if the null hypothesis holds, then so does a hypothesis $H_{0,i}$: The i-th column is not watermarked also holds. This reduces the problem of rejecting H_0 to simply rejecting $H_{0,i}$ for each column i.

Let T_i represent the number of elements in the i-th *value* column that hash to 0. Under the i-th null hypothesis, $H_{0,i}$ should follow the Bernoulli Distribution B with probability $1/2$ as an ideal hash function $\mathcal{H}$ will output 0 or 1 with probability 1/2. Let m be the total number of rows, i.e., $T_i \sim B(m, 1/2)$ for a sufficiently large number of rows m. By the Central Limit Theorem (CLT), for large m, we obtain that:

$$2\sqrt{m}\left(\frac{T_i}{m} - \frac{1}{2}\right) \sim \mathcal{N}(0, 1)$$

where $\mathcal{N}(0, 1)$ is the normal distribution. Thus, the test statistic for a one-proportion z-test is:

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

For each column, the detection algorithm computes a z-score by counting values that hash to 0. To account for multiple hypothesis testing (e.g., 5 columns at $\alpha = 0.05$), per-column thresholds α_i are adjusted (e.g., $\alpha_i = 0.01$). If a column's z-score exceeds its threshold, the null hypothesis is rejected, indicating a watermark. Otherwise, no conclusion is made.

To prevent spoofing (where forgers combine valid watermarked datasets), we use a secret seed in the hash function (Algorithm 1). Each dataset's watermark uses a unique $seed$, making concatenated forgeries detectable as inconsistent.

Robustness to Deletion, Permutation. It is clear that the permutation of rows does not impact the count T_i. $H_{0,i}$ is evaluated for every column i. This implies that the permutation of the column from position i to some j will still have its corresponding null hypothesis $H_{0,j}$ and evaluated. Now, observe that the detection algorithm performs multiple hypothesis tests conducted simultaneously. Therefore, removing columns implies that one has to compute α_i as a function of α and the number of remaining columns. This guarantees robustness to column deletion. Removal of rows implies a smaller m. This results in an increase in the error in the CLT approximation. However, in practice, a rule-of-thumb for applying Z-test has been for $m > 50$ [9]. However, if $m < 50$, one could apply the Z-test on H_0 and not individual $H_{0,i}$.

Finally, as remarked before, one can also modify the application of $\mathcal{H}$ to ensure support for truncation.

D.4 Analysis on Removal of HashMark

Before we look at the mathematical analysis, we discuss the modes of attacks to remove the watermark. The property of the ideal hash function $\mathcal{H}$ implies that the perturbation of a cell content initially mapping to 0 can flip to 1, with a probability 0.5. Further, a secret seed (of the seeded hash function) implies that an adversary, without knowledge of this seed, cannot determine the actual mapping of the bit.

This section will study the effort required for the perturbation to remove the watermark. Specifically, an adversary can only modify r cells by adding noise. We will analyze the expected number of r. Note that an adversary, adding noise to every cell in a column, can remove the watermark. This is true for every scheme [16, 29, 43]. Experimentally, we show the results comparing with [29] in Section E.

In the analysis below, we assume there are M values in total. Of this, N is the number of values with the property they hash to a desired bit. In $\mathsf{HashMark}_1$, we have $N = \ell$ while $M = mn$. In $\mathsf{HashMark}_2$, we have $N = M = m$ as described above. The proof of the following are deferred to Section G in the appendix.

Proposition 1. *Given values $val_1, \ldots, val_M$. Then, the minimum number of values needed to ensure that the Z-score remains α is given by:*

$$\alpha \cdot \frac{\sqrt{M}}{2} + \frac{M}{2}$$

Proof. Of the m values, we need to compute T_i that ensures that the score is α. We use Equation 1 as:

$$\alpha = \frac{2(T_i - 0.5M)}{\sqrt{M}}$$

Then, $T_i = 0.5M + \alpha\sqrt{M}/2$. In other words, we need at least $0.5M + \alpha\sqrt{M}/2$ values to ensure a Z-score of α. Call this value T_α. □

Theorem 3. *Let r be the number of cells an adversary can modify. This modification is done by sampling noises $\epsilon_1, \ldots, \epsilon_r \xleftarrow{\$} \mathcal{D}$. Then, we have:*

$$\mathbb{E}[r] := 2 \cdot (N - T_\alpha) \cdot \frac{M}{N}$$

for any error distribution $\mathcal{D}$.

Proof of Theorem 3. First, observe that for any value val_i such that $\mathcal{H}(val_i) = 0$:

$$\Pr[\mathcal{H}(val_i + \epsilon_i) = 1] = \frac{1}{2}$$

for any $\epsilon_i \xleftarrow{\$} \mathcal{D}$. We already know that one needs at least $T_\alpha = 0.5M + \alpha\sqrt{M}/2$ cells to be unmodified to get a score of α (from Proposition 1). To achieve the watermark removal, we need to add noise to the remaining $N - T_\alpha$ cells. Observe that this follows a hypergeometric distribution - in a sample of size M, N successes exist (i.e., mapping to 0). Then, the expected number of tries to pick at least $(N - T_\alpha)$ successfully is given by: $\approx (N - T_\alpha) \cdot M/N$. Therefore, we get:

$$\mathbb{E}[r] := 2 \cdot (N - T_\alpha) \cdot \frac{M}{N}$$

□

Note that in $\mathsf{HashMark}_1$ where $N < M$, the number of tries needed for the adversary is inversely proportional to N, making $\mathsf{HashMark}_1$ more robust to noise addition attacks. Meanwhile, in $\mathsf{HashMark}_2$, since $M = N$, the number of tries needed is much smaller. Consequently, one can envision $\mathsf{HashMark}_2$ where only a specific subset of cells (chosen at random) is embedded with the bit. While this makes it more resilient to modification attacks, the problem of efficiently identifying this subset of cells becomes paramount.

Other Attacks. We look at some additional attack vectors.

- Data Augmentation Attacks: Adding data reduces the z-score. However, since the secret information is unknown to an attacker, one can expect that half of the new content will map to 0 on average. For example, if one had m rows in a column that all map to 0, adding another m rows will reduce the z-score by a factor of $\sqrt{2}$, on expectation.
- Feature Selection: Observe that the choice of z-score threshold depends on the number of columns in the dataset. This is discussed in 5.1.1. Therefore, reducing the number of columns will consequently require a higher threshold.

HashMark **and Applications.** Watermarking tabular data provides verifiable guarantees for data integrity in organizational settings where datasets are routinely shared. When a watermark embedded using $\mathsf{HashMark}_2$ is detected in a dataset D, two key properties hold: (1) Theorem 5.3 ensures an expected upper bound on the number of modified cells, limiting undetected alterations; and (2) if an attacker injects $\gamma \cdot m$ additional rows into an m-row dataset, the detection signal degrades predictably, with the z-score scaling by $\sqrt{(1+\gamma)}$. These mechanisms establish a measurable trust boundary, enabling provenance tracking while tolerating benign modifications. By formalizing such robustness-utility tradeoffs, our work advances watermarking techniques for practical data governance.

E Experimental Results

In this section, we focus on experimentation for embedding watermarks in numerical data, specifically floating-point values. Our experiments were performed on an Apple MacBook M1 Pro with 16GB of memory running Sonoma 14.3. We used Python 3.11. We instantiated the hash function using SHA-256 from the hashlib module. We select a random seed for evaluating the hash function. We implemented $\mathsf{Generate}$ by adding 10^{-c} to the value until it hashes to 0. Our choice of c is specified for each context separately. Due to space constraints, we will focus on $\mathsf{HashMark}_2$ in this section and defer the experiments pertaining to $\mathsf{HashMark}_1$ to the appendix.

We defer the experiments pertaining to $\mathsf{HashMark}_1$ to the appendix in Section E.2

E.1 Evaluation of $\mathsf{HashMark}_2$

In this section, we evaluate the performance of $\mathsf{HashMark}_2$ along the following dimensions:

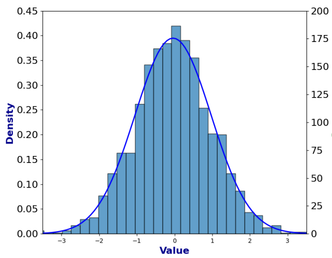
- **Performance (vs the work of [29]) on Gaussian Datasets:** Following [29], we test $\mathsf{HashMark}_2$ on Gaussian data (1 column, 2000 rows). With $c = 10$, $\mathsf{HashMark}_2$ matches their robustness and fidelity while being significantly simpler, which proves that complex watermarking isn't necessary.

  Fidelity: The KDE plots (Figs. 2a-2b) show nearly identical distributions before and after watermarking. Figure 2d, which shows how the choice of 10^{-c} in $\mathsf{Generate}$ impacts the mean-squared error (MSE), confirms that smaller c values (larger perturbations) increase MSE, as expected.

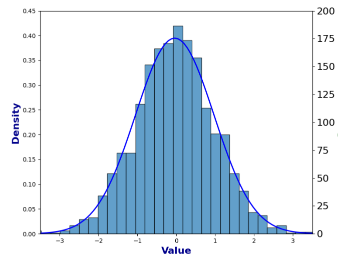
  Robustness: Figure 2c demonstrates that z-scores grow with more rows, improving detection confidence. When adding Gaussian noise (Fig. 2e), smaller c values yield lower z-scores, showing greater noise sensitivity. Crucially, our z-scores consistently surpass Ngo et al.'s under identical conditions (Fig. 6). Extended results (Figs. 8a, 8b) reinforce these findings and are deferred to the appendix.

  For completeness in Figure 7, we reproduce the plot from Ngo et al. for the abovementioned experiments. We also present additional plots for $\mathsf{HashMark}_2$ in Figure 8. Figure 8a extends Figure 2d for a wider choice of c while Figure 8b extends Figure 2c for a larger number of rows. These additional plots are in line with the conclusions drawn above.
- **Utility for Real-Life Datasets:** Following prior works such as [16] and [29], we evaluate the utility of our proposed approach $\mathsf{HashMark}_2$ by testing it on four real-world datasets. These datasets are first used to train neural network-based and statistical-based generative methods. The trained generative method is then used to generate synthetic datasets. Specifically, we utilize CTGAN [38], Gaussian Copula [28], and TVAE [38] to represent GAN-based, copula-based, and VAE-based generators, respectively, for generating tabular data. We utilize the Synthetic Data Vault [30] as our library and employ the default parameters. The dataset was randomly partitioned with 25% test cases. While we defer a discussion on the dataset to Section D, we summarize the findings of our experiment below in Table 2. Our experiments indicate that the watermarking has a negligible impact on the accuracy of the synthetic dataset, even for a multi-class classification problem.
- **Fidelity for Alphanumeric Synthetic Data:** We evaluate $\mathsf{HashMark}_2$'s performance on alphanumeric attributes by measuring the Jensen-Shannon divergence (JSD) between watermarked synthetic data (where all values hash to 0) and real datasets. Using SciPy's JSD implementation [36] with 30 trials, we find:
  - **ASINs** (10-character alphanumeric): 0.1090 ± 0.0016 JSD (vs. Amazon Product Dataset [31])
  - **Git commit hashes** (40-character hex): 0.002176 ± 0.0003 JSD (vs. GitHub Commit Messages [10])

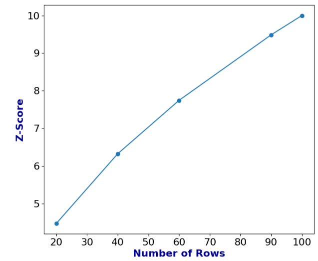
  Lower JSD values indicate better preservation of the original distribution, demonstrating $\mathsf{HashMark}_2$'s effectiveness for alphanumeric data.
- $\mathsf{HashMark}_2$ **with simpler classifiers and dataset:** Prior experiments were on datasets with multiple attributes and complex machine learning models. We wanted to study $\mathsf{HashMark}_2$'s impact on the accuracy of simpler machine learning models with fewer columns. Specifically, we ran experiments using one attribute and two classes on these simple classifiers - linear



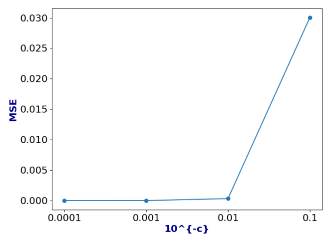
(a)



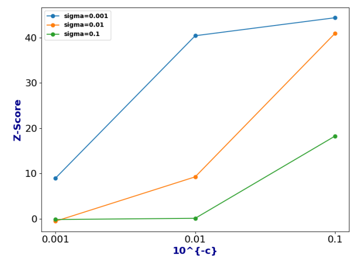
(b)



(c)



(d)



(e)

Figure 2: Plot of various experiments on Gaussian dataset. Figures 2a and 2b show the distribution of the data, before and after watermarking. Value refers to the actual value in the dataset. Figure 2c shows the variation of the z-score with the number of rows sampled. Figure 2d plots the variation of the mean-squared error (MSE) for different choices of c. Figure 2e plots the change in z-score when compared with the choice of c for various Gaussian noises.

Table 2: Accuracy comparison of different classifiers and synthesizers across four datasets on synthetic and watermarked synthetic data. Standard deviations are included for each record. W/M = Watermarked synthetic dataset, while Non-W/M refers to an unwatermarked but synthetic dataset.

Dataset	Classifier	Synthesizer	Non-W/M (%)	W/M (%)
Wilt	XGB	CTGAN	83.63 ± 4.63	83.31 ± 5.01
		Copula	94.38 ± 0.53	94.40 ± 0.52
		TVAE	94.87 ± 0.37	94.89 ± 0.39
	RF	CTGAN	84.45 ± 5.74	84.30 ± 5.70
		Copula	94.39 ± 0.52	94.40 ± 0.52
		TVAE	94.34 ± 0.37	94.34 ± 0.38
Housing	XGB	CTGAN	49.26 ± 2.38	49.11 ± 2.68
		Copula	55.15 ± 5.12	55.66 ± 4.77
		TVAE	61.55 ± 2.39	61.13 ± 2.46
	RF	CTGAN	48.31 ± 1.90	48.14 ± 2.00
		Copula	52.97 ± 5.83	53.04 ± 5.93
		TVAE	62.30 ± 1.92	62.40 ± 1.77
HOG	XGB	CTGAN	77.65 ± 2.07	77.62 ± 2.08
		TVAE	89.77 ± 1.59	89.34 ± 1.76
	RF	CTGAN	74.40 ± 4.41	74.39 ± 4.48
		TVAE	91.20 ± 2.16	91.28 ± 2.16
Shoppers	XGB	CTGAN	86.43 ± 0.79	85.28 ± 1.95
		Copula	86.01 ± 1.38	86.56 ± 1.41
		TVAE	87.94 ± 0.61	87.85 ± 0.54
	RF	CTGAN	87.77 ± 0.82	86.00 ± 2.74
		Copula	86.05 ± 1.40	85.78 ± 1.38
		TVAE	88.71 ± 1.00	88.10 ± 1.23

regression, logistic regression, and decision tree. We present our findings in Table 3. To summarize, we demonstrate that the perturbation parameter (i.e., adding 10^{-c}) controls the deviation from the value. However, even with a smaller value of c, there is a negligible difference in the model performance.

- Constrained Sampling, Threshold, and Z-Score: We also investigate the utility of constrained sampling, i.e., one in which we sample a row from the distribution ρ and we check if at

Table 3: Model Performance Under Watermarking Perturbation (10^{-c}). W/M = Watermarked dataset. For Logistic/Decision Tree, we report accuracy; for Linear Regression, we report R^2 values.

	Logistic Reg.		Linear Reg.		Decision Tree	
	Orig.	W/M	Orig. (R^2)	W/M (R^2)	Orig.	W/M
$c = 2$	99.98%	99.64%	1.000000	0.999899	100%	100%
$c = 4$	99.98%	99.98%	1.000000	1.000000	100%	99.995%
$c = 6$	99.98%	99.98%	1.000000	1.000000	100%	99.961%

least t fraction of the n columns in a row hash to 0. If not, we reject that row and resample another. This process is repeated until an appropriately sized dataset is generated, ensuring that correlations are preserved. We summarize our findings across Tables 6 and **??** for the four datasets. While increasing t does increase the running time of watermarked dataset generation, we find no significant difference in accuracy; however, we do notice an increase in z-score, as expected.

E.2 Evaluation of $\mathsf{HashMark}_1$

We begin by benchmarking the performance of $\mathsf{HashMark}_1$ along the following axes:

- Varying ℓ, we wish to study the running time of the watermarking process. We break down the running time of watermarking as (a) the cost of identifying locations to embed the watermark and (b) the time taken to run Generate to embed the desired bits.
- The utility of the watermarked dataset vs. the original dataset for downstream machine learning tasks.
- The role of ℓ in accuracy, i.e., how does the accuracy change when more bits are embedded?

Performance of Embedding Process. In Figure 3, we plot the time, in seconds, against the number of bits being embedded. We split the cost as follows: to generate locations for embedding (dubbed pair generation time) and then modify the cell content until it hashes to the desired bit. Recall that the pair generation time requires using a seed to produce ℓ cell positions, which only contain floating point values. We then use the same seed to generate ℓ bits additionally. As one can observe, the embedding time is much smaller than the pair generation time, and it takes less than 10 milliseconds to embed as many as 1000 bits.

Dataset. We study the above for a specific dataset - the adult census income dataset from [6, 19] to predict if an individual earns over $50,000 per year. The preprocessed dataset has 105 features and 45,222 records with a 25% positive class (i.e., 25% of the records have class 1 while the rest are in class 0) We randomly split into training and testing datasets. We observed that the dataset consisted of integers or floating point values with at least eight decimal places. This leads us to choose $c = 6$ and embed only in the floating point values.

Downstream Utility. We embed $\ell = 384$ bits [3] They are:

- Logistic Regression Classifier with maximum iterations as 1000
- Random Forest Classifier with 100 estimators
- MLP Classifier with hidden layer sizes 100, 50; maximum iterations=1000, and learning rate – 0.0001

We plotted the difference in accuracy when run on the original versus the watermarked dataset in Figures 4 and 5 for each of the 1000 runs. Meanwhile, in Table 4, we present the average accuracy of the 1000 runs. Identical behavior was observed in the Logistic Regression classifier with less than 0.005% difference observed in the accuracy of the other two classifiers. This shows that $\mathsf{HashMark}_1$'s embedding has a negligible impact on the accuracy of the classifier. For completeness, we also plot

[3] Choice of ℓ is set to be 384 because it is the number of bits in a standard hash-based watermarking scheme albeit for messaging applications (i.e., signatures) known as BLS Signature [5]. Note that this corresponds to less than 1% of the number of cells in the dataset.

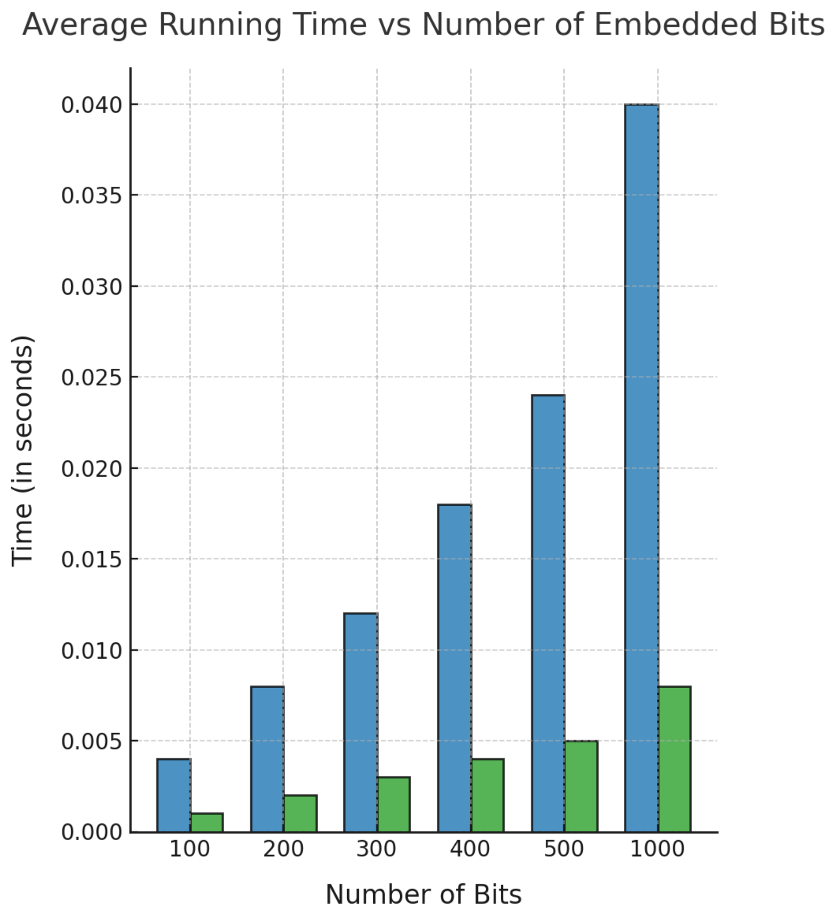


Figure 3: Embedding Time as a function of ℓ for $\mathsf{HashMark}_1$. Here, the blue column refers to the cost of generating valid cells to embed in the dataset, while the green column is the cost of modifying the content to make it hash to the desired bit.

Table 4: Classification accuracy (%) with and without watermarking. In addition to this, we add the standard deviation of each record.

Model	Logistic Regression	Random Forest	MLP Classifier
Original	84.021 ± 0.3	85.186 ± 0.27	83.504 ± 0.44
Watermarked	84.021 ± 0.3	85.188 ± 0.28	83.508 ± 0.446

the difference in accuracy between the original and watermarked dataset in Figures 4 and 5, in each of the 1000 runs. As can be observed, the most significant difference in accuracy is less than 0.005%.

Finally, in Figure 5b, we plot the impact of increasing ℓ on the accuracy of the logistic regression classifier. As expected, larger ℓ does cause an impact in accuracy, though the degradation is minimal.

F Additional Experiments

We also present additional experiments studying the variation of MSE with respect to the choice of c for further values of c. Similarly, we also show how the Z-score varies for larger sampled rows. This is done in Figure 8.

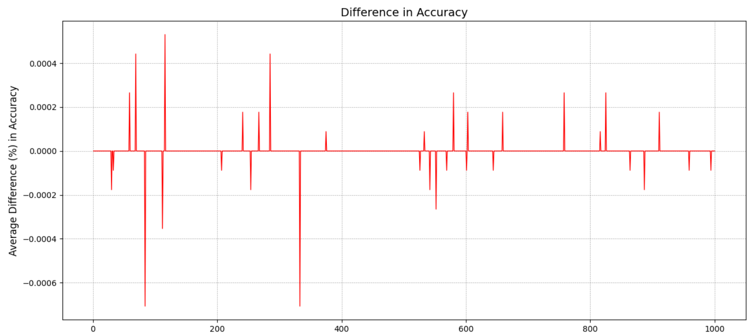
In Figure 7, we reproduce Figure 2 from Ngo et al. [29]. This shows that the performance of $\mathsf{HashMark}_2$, as seen in Figure 2, matches (or surpasses) similar experiments from Ngo et al. This is

Table 5: Effect of constraint threshold t on synthetic data quality across two datasets. We report the average z-scores, sampling time (in seconds), and classification accuracy (in %) using different classifiers and synthesizers. This is with respect to $\mathsf{HashMark}_2$. Accuracy is shown for both the non-W/M and W/M settings.

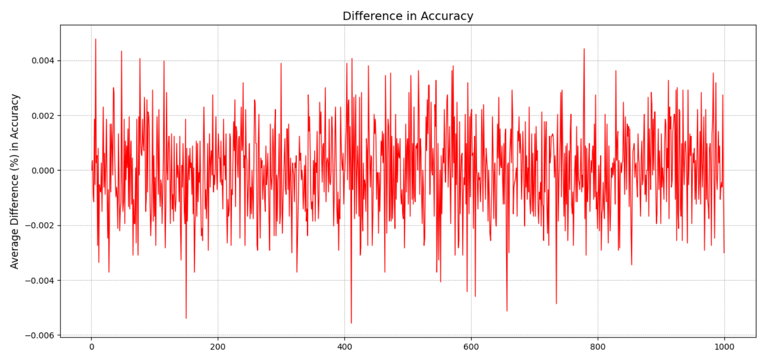
Dataset	t	z-score	Sampling Time (s)	Classifier	Synthesizer	Non-W/M (%)	W/M (%)
Wilt (5 cols, 3629 samples)	1/4	1.74 ± 0.22	64.08 ± 6.68	XGB	TVAE	95.24 ± 0.57	95.07 ± 0.84
					GC	94.33 ± 0.31	94.53 ± 0.24
					CTGAN	83.65 ± 3.24	81.79 ± 6.76
				RF	TVAE	94.78 ± 0.30	94.86 ± 0.42
					GC	94.43 ± 0.31	94.45 ± 0.30
					CTGAN	84.31 ± 1.93	84.76 ± 1.54
	1/3	1.92 ± 0.24	65.45 ± 4.90	XGB	TVAE	94.86 ± 0.44	95.19 ± 0.43
					GC	94.33 ± 0.31	94.53 ± 0.24
					CTGAN	84.07 ± 4.94	85.62 ± 6.19
				RF	TVAE	94.84 ± 0.45	94.76 ± 0.66
					GC	94.43 ± 0.31	94.45 ± 0.30
					CTGAN	86.08 ± 6.52	86.60 ± 6.82
	1/2	9.43 ± 0.44	65.05 ± 1.26	XGB	TVAE	95.02 ± 0.44	95.31 ± 0.48
					GC	94.33 ± 0.31	94.43 ± 0.33
					CTGAN	82.55 ± 7.06	84.23 ± 6.56
				RF	TVAE	94.73 ± 0.43	94.68 ± 0.45
					GC	94.43 ± 0.31	94.43 ± 0.30
					CTGAN	80.46 ± 7.11	80.50 ± 6.03
	2/3	22.73 ± 0.30	108.95 ± 4.59	XGB	TVAE	95.22 ± 0.32	95.17 ± 0.65
					GC	94.33 ± 0.31	94.26 ± 0.46
					CTGAN	78.83 ± 9.36	78.86 ± 9.80
				RF	TVAE	94.83 ± 0.66	94.83 ± 0.25
					GC	94.43 ± 0.31	94.41 ± 0.34
					CTGAN	82.12 ± 5.57	84.21 ± 5.18
	3/4	23.20 ± 0.16	116.91 ± 9.98	XGB	TVAE	95.21 ± 0.32	95.32 ± 0.53
					GC	94.33 ± 0.31	94.26 ± 0.46
					CTGAN	78.38 ± 6.28	77.47 ± 6.41
				RF	TVAE	94.93 ± 0.41	94.84 ± 0.29
					GC	94.43 ± 0.31	94.41 ± 0.34
					CTGAN	79.87 ± 6.53	80.74 ± 5.64
Housing (9 cols, 15480 samples)	1/4	2.84 ± 0.72	449.17 ± 40.27	XGB	TVAE	63.35 ± 0.76	63.43 ± 0.79
					GC	52.82 ± 3.99	52.27 ± 3.24
					CTGAN	47.07 ± 2.58	46.59 ± 2.15
				RF	TVAE	62.79 ± 0.59	62.93 ± 0.43
					GC	53.60 ± 2.02	53.71 ± 3.27
					CTGAN	45.72 ± 2.02	46.50 ± 2.31
	1/3	2.63 ± 0.64	415.68 ± 7.37	XGB	TVAE	62.75 ± 1.54	62.86 ± 1.42
					GC	52.82 ± 3.99	52.27 ± 3.24
					CTGAN	46.48 ± 1.73	46.63 ± 2.89
				RF	TVAE	61.91 ± 2.63	61.93 ± 2.29
					GC	53.60 ± 2.02	53.71 ± 3.27
					CTGAN	48.99 ± 1.39	48.69 ± 1.20
	1/2	18.27 ± 0.20	552.12 ± 12.20	XGB	TVAE	60.95 ± 3.12	61.02 ± 3.05
					GC	52.82 ± 3.99	52.76 ± 2.63
					CTGAN	47.70 ± 1.95	48.75 ± 3.12
				RF	TVAE	63.38 ± 0.16	63.30 ± 0.45
					GC	53.60 ± 2.02	53.45 ± 2.90
					CTGAN	49.81 ± 2.78	47.59 ± 3.02
	2/3	34.43 ± 0.29	848.09 ± 17.84	XGB	TVAE	61.75 ± 2.03	61.88 ± 1.73
					GC	53.60 ± 4.82	52.91 ± 2.85
					CTGAN	47.77 ± 2.52	46.29 ± 3.79
				RF	TVAE	62.24 ± 1.30	62.24 ± 1.55
					GC	54.06 ± 2.88	53.74 ± 3.21
					CTGAN	48.81 ± 2.08	48.13 ± 2.04
	3/4	53.74 ± 0.29	1632.13 ± 79.29	XGB	TVAE	62.13 ± 1.85	62.83 ± 1.97
					GC	52.82 ± 3.99	53.91 ± 3.73
					CTGAN	46.84 ± 3.37	48.00 ± 2.20
				RF	TVAE	60.86 ± 2.19	60.87 ± 2.20
					GC	53.60 ± 2.02	53.75 ± 2.66
					CTGAN	49.89 ± 2.68	48.56 ± 1.73

Table 6: Effect of constraint threshold t on synthetic data quality across two datasets. We report the average z-scores, sampling time (in seconds), and classification accuracy (in %) using different classifiers and synthesizers. This is with respect to $\mathsf{HashMark}_2$. Accuracy is shown for both the non-W/M and W/M settings.

Dataset	t	z-score	Sampling Time (s)	Classifier	Synthesizer	Non-W/M (%)	W/M (%)
HOG (18 cols, 8244 samples)	1/4	-5.77 ± 0.78	373.24 ± 105.90	XGB	TVAE	88.52 ± 4.74	87.53 ± 4.89
HOG (18 cols, 8244 samples)	1/4	-5.77 ± 0.78	373.24 ± 105.90	XGB	CTGAN	73.74 ± 3.15	72.92 ± 3.75
HOG (18 cols, 8244 samples)	1/4	-5.77 ± 0.78	373.24 ± 105.90	RF	TVAE	92.48 ± 1.33	93.03 ± 1.22
HOG (18 cols, 8244 samples)	1/4	-5.77 ± 0.78	373.24 ± 105.90	RF	CTGAN	74.56 ± 3.28	74.24 ± 2.90
HOG (18 cols, 8244 samples)	1/3	-4.89 ± 1.15	511.61 ± 8.76	XGB	TVAE	88.84 ± 1.90	90.64 ± 1.19
HOG (18 cols, 8244 samples)	1/3	-4.89 ± 1.15	511.61 ± 8.76	XGB	CTGAN	70.44 ± 6.26	70.87 ± 5.59
HOG (18 cols, 8244 samples)	1/3	-4.89 ± 1.15	511.61 ± 8.76	RF	TVAE	91.36 ± 0.94	91.92 ± 1.00
HOG (18 cols, 8244 samples)	1/3	-4.89 ± 1.15	511.61 ± 8.76	RF	CTGAN	73.29 ± 3.81	73.25 ± 4.12
HOG (18 cols, 8244 samples)	1/2	7.46 ± 0.18	797.56 ± 12.58	XGB	TVAE	91.43 ± 1.27	91.49 ± 0.85
HOG (18 cols, 8244 samples)	1/2	7.46 ± 0.18	797.56 ± 12.58	XGB	CTGAN	75.44 ± 3.19	75.82 ± 3.40
HOG (18 cols, 8244 samples)	1/2	7.46 ± 0.18	797.56 ± 12.58	RF	TVAE	92.43 ± 1.01	91.86 ± 0.89
HOG (18 cols, 8244 samples)	1/2	7.46 ± 0.18	797.56 ± 12.58	RF	CTGAN	74.32 ± 3.27	74.00 ± 3.09
HOG (18 cols, 8244 samples)	$2/3$	31.40 ± 0.21	9868.32 ± 8790.14	XGB	TVAE	88.80 ± 2.53	87.78 ± 2.71
HOG (18 cols, 8244 samples)	$2/3$	31.40 ± 0.21	9868.32 ± 8790.14	XGB	CTGAN	72.71 ± 2.93	73.34 ± 3.31
HOG (18 cols, 8244 samples)	$2/3$	31.40 ± 0.21	9868.32 ± 8790.14	RF	TVAE	91.78 ± 1.63	91.47 ± 2.50
HOG (18 cols, 8244 samples)	$2/3$	31.40 ± 0.21	9868.32 ± 8790.14	RF	CTGAN	72.31 ± 3.75	72.71 ± 4.08
HOG (18 cols, 8244 samples)	$3/4$	40.77 ± 0.16	35088.75 ± 30542.58	XGB	TVAE	90.83 ± 1.00	90.74 ± 1.09
HOG (18 cols, 8244 samples)	$3/4$	40.77 ± 0.16	35088.75 ± 30542.58	XGB	CTGAN	75.26 ± 4.04	74.95 ± 4.00
HOG (18 cols, 8244 samples)	$3/4$	40.77 ± 0.16	35088.75 ± 30542.58	RF	TVAE	88.92 ± 3.03	88.08 ± 3.70
HOG (18 cols, 8244 samples)	$3/4$	40.77 ± 0.16	35088.75 ± 30542.58	RF	CTGAN	70.71 ± 5.23	70.20 ± 5.15
Shopper (12 cols, 9247 samples)	1/4	-2.11 ± 1.38	438.51 ± 5.14	XGB	TVAE	87.78 ± 0.78	87.78 ± 0.76
Shopper (12 cols, 9247 samples)	1/4	-2.11 ± 1.38	438.51 ± 5.14	XGB	GC	85.51 ± 0.63	85.80 ± 0.76
Shopper (12 cols, 9247 samples)	1/4	-2.11 ± 1.38	438.51 ± 5.14	XGB	CTGAN	87.35 ± 0.35	87.06 ± 0.95
Shopper (12 cols, 9247 samples)	1/4	-2.11 ± 1.38	438.51 ± 5.14	RF	TVAE	88.74 ± 0.32	88.74 ± 0.45
Shopper (12 cols, 9247 samples)	1/4	-2.11 ± 1.38	438.51 ± 5.14	RF	GC	85.62 ± 0.43	85.99 ± 0.98
Shopper (12 cols, 9247 samples)	1/4	-2.11 ± 1.38	438.51 ± 5.14	RF	CTGAN	87.95 ± 0.49	87.91 ± 0.28
Shopper (12 cols, 9247 samples)	1/3	-3.37 ± 1.26	639.55 ± 64.59	XGB	TVAE	88.13 ± 0.63	87.86 ± 0.86
Shopper (12 cols, 9247 samples)	1/3	-3.37 ± 1.26	639.55 ± 64.59	XGB	GC	85.51 ± 0.63	85.28 ± 1.17
Shopper (12 cols, 9247 samples)	1/3	-3.37 ± 1.26	639.55 ± 64.59	XGB	CTGAN	84.76 ± 1.05	84.94 ± 1.31
Shopper (12 cols, 9247 samples)	1/3	-3.37 ± 1.26	639.55 ± 64.59	RF	TVAE	88.18 ± 0.53	88.06 ± 0.81
Shopper (12 cols, 9247 samples)	1/3	-3.37 ± 1.26	639.55 ± 64.59	RF	GC	85.62 ± 0.43	85.70 ± 0.68
Shopper (12 cols, 9247 samples)	1/3	-3.37 ± 1.26	639.55 ± 64.59	RF	CTGAN	88.01 ± 0.62	87.80 ± 0.69
Shopper (12 cols, 9247 samples)	1/2	9.22 ± 1.27	939.33 ± 107.06	XGB	TVAE	87.27 ± 1.33	87.54 ± 0.94
Shopper (12 cols, 9247 samples)	1/2	9.22 ± 1.27	939.33 ± 107.06	XGB	GC	85.51 ± 0.63	86.10 ± 0.94
Shopper (12 cols, 9247 samples)	1/2	9.22 ± 1.27	939.33 ± 107.06	XGB	CTGAN	85.27 ± 1.54	85.59 ± 1.64
Shopper (12 cols, 9247 samples)	1/2	9.22 ± 1.27	939.33 ± 107.06	RF	TVAE	88.61 ± 0.56	88.28 ± 0.54
Shopper (12 cols, 9247 samples)	1/2	9.22 ± 1.27	939.33 ± 107.06	RF	GC	85.62 ± 0.43	85.65 ± 0.71
Shopper (12 cols, 9247 samples)	1/2	9.22 ± 1.27	939.33 ± 107.06	RF	CTGAN	87.80 ± 0.25	87.57 ± 0.69
Shopper (12 cols, 9247 samples)	2/3	34.14 ± 0.28	3690.59 ± 252.79	XGB	TVAE	87.46 ± 0.69	88.01 ± 0.20
Shopper (12 cols, 9247 samples)	2/3	34.14 ± 0.28	3690.59 ± 252.79	XGB	GC	85.51 ± 0.63	85.74 ± 0.61
Shopper (12 cols, 9247 samples)	2/3	34.14 ± 0.28	3690.59 ± 252.79	XGB	CTGAN	85.89 ± 0.57	85.59 ± 1.70
Shopper (12 cols, 9247 samples)	2/3	34.14 ± 0.28	3690.59 ± 252.79	RF	TVAE	88.48 ± 0.32	88.30 ± 0.64
Shopper (12 cols, 9247 samples)	2/3	34.14 ± 0.28	3690.59 ± 252.79	RF	GC	85.62 ± 0.43	86.49 ± 0.56
Shopper (12 cols, 9247 samples)	2/3	34.14 ± 0.28	3690.59 ± 252.79	RF	CTGAN	87.89 ± 0.88	87.82 ± 0.59
Shopper (12 cols, 9247 samples)	3/4	43.50 ± 0.42	9276.41 ± 1742.76	XGB	TVAE	88.10 ± 0.92	88.39 ± 0.78
Shopper (12 cols, 9247 samples)	3/4	43.50 ± 0.42	9276.41 ± 1742.76	XGB	GC	85.51 ± 0.63	86.06 ± 1.24
Shopper (12 cols, 9247 samples)	3/4	43.50 ± 0.42	9276.41 ± 1742.76	XGB	CTGAN	86.75 ± 0.81	86.44 ± 0.43
Shopper (12 cols, 9247 samples)	3/4	43.50 ± 0.42	9276.41 ± 1742.76	RF	TVAE	88.52 ± 0.55	88.17 ± 0.88
Shopper (12 cols, 9247 samples)	3/4	43.50 ± 0.42	9276.41 ± 1742.76	RF	GC	85.62 ± 0.43	86.77 ± 0.74
Shopper (12 cols, 9247 samples)	3/4	43.50 ± 0.42	9276.41 ± 1742.76	RF	CTGAN	87.80 ± 0.58	87.63 ± 0.74



(a) Plot of the difference in accuracy between the original and the watermarked dataset in each of the 1000 iterations for the Logistic Regression Classifier



(b) Plot of the difference in accuracy between the original and the watermarked dataset in each of the 1000 iterations for the Random Forest Classifier

Figure 4: Experiments pertaining to $\mathsf{HashMark}_1$ for the Adult Census Dataset (Part 1).

especially important considering that $\mathsf{HashMark}_2$ is conceptually simpler while offering support for categorical data and being more secure. Recall that $\mathsf{HashMark}_2$ uses a truly random value as seed, while Ngo et al. opt for a heuristic approach to obtain seed via pairing algorithm, which are often poor sources of entropy.

G Deferred Proofs

Proof of Theorem 1. For each element x_i in $\mathbf{X}$, let x_i' be the corresponding element in $\mathbf{X}_w$. As defined above:

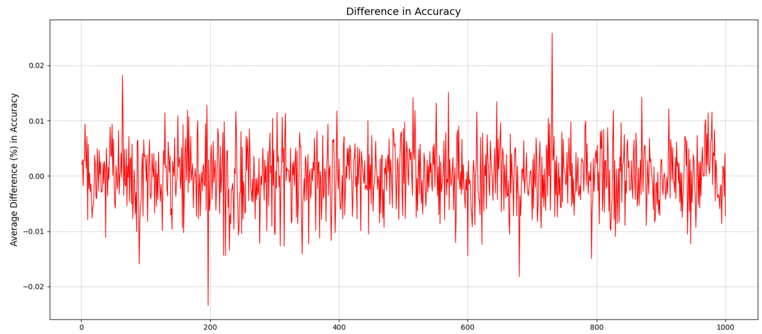
$$x_i' = x_i + k_i \cdot 10^{-c},$$

where $k_i = \min\{k \geq 0 \mid \mathcal{H}(x_i + k \cdot 10^{-c}) = 0$. In other words, $|x_i - x_i'| = k_i \cdot 10^{-c}$.

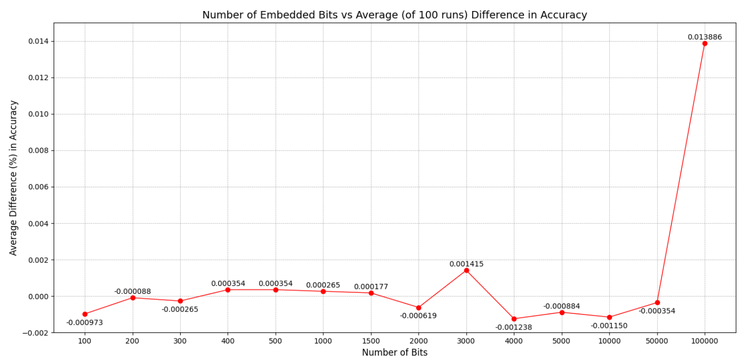
Recall that $\mathcal{H}$ maps to 0 and 1 with equal probability. Therefore, for a given $x_i' = x_i + k_i \cdot 10^{-c}$, the hash function should have mapped to 1 for every choice from 0 to $k_i - 1$ and succeed in time k_i. In other words, $\Pr[K_i = k] = \left(\frac{1}{2}\right)^{k+1}$, i.e., it follows a geometric distribution.

Now, $||\mathbf{X} - \mathbf{X}_w||_\infty = \max_i |x_i - x_i'| = \max_i k_i \cdot 10^{-c}$. We can use the well-known approximation for the maximum of n i.i.d geometric variables to get $\mathbb{E}[\max_i k_i] = 0.5 + H_N / \ln 2$ where H_N is the N-th harmonic number. Further $\ln N \leq H_N \leq 1 + \ln N$ or $H_N \leq \ln N + 1$. This gives us that:

$$\begin{aligned}\mathbb{E}[||\mathbf{X} - \mathbf{X}_w||_\infty] &\leq \left(0.5 + \frac{\ln(N) + 1}{\ln 2}\right) \cdot 10^{-c} \\ &\leq (\ln N + 2) \cdot 10^{-c}\end{aligned}$$

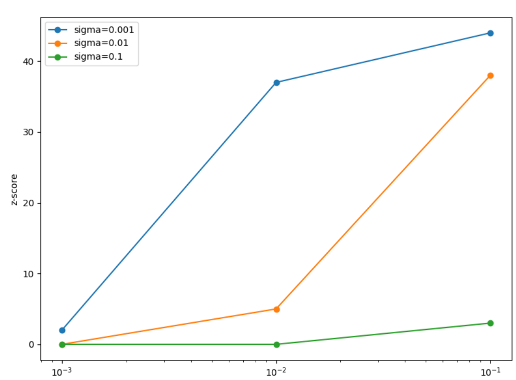


(a) Plot of the difference in accuracy between the original and the watermarked dataset in each of the 1000 iterations for the MLP Classifier

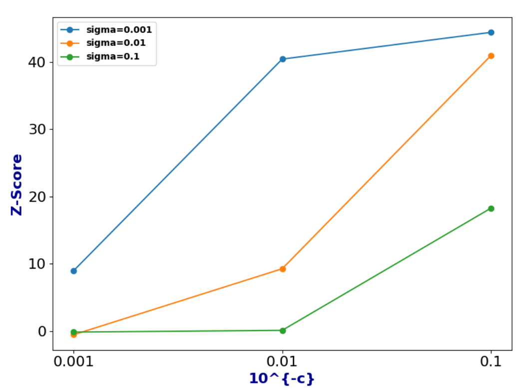


(b) Average Difference in the accuracy of the logistic regression classifier as the number of bits embedded (ℓ) increases.

Figure 5: Experiments pertaining to $\mathsf{HashMark}_1$ for the Adult Census Dataset (Part 2).

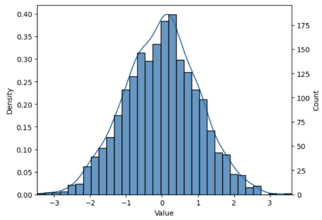


(a)

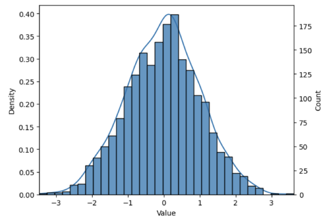


(b)

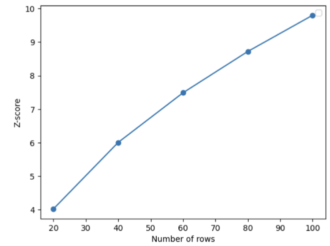
Figure 6: This figure shows the evaluation of the robustness of Gaussian noise by studying the z-score across various choices of standard deviation. To the left, we show the results from [29], and to the right, we show the results from our own experiment. Observe similar behavior across both works.



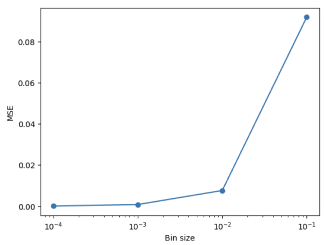
(a) KDE before watermark



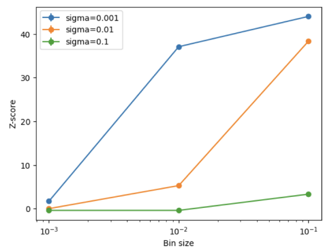
(b) KDE after watermark



(c) z-score with number of samples

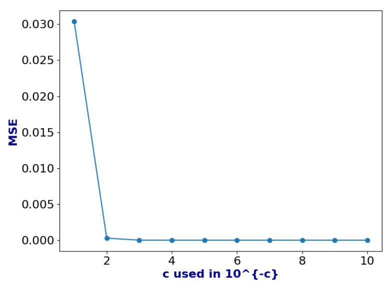


(d) Fidelity with different bin sizes

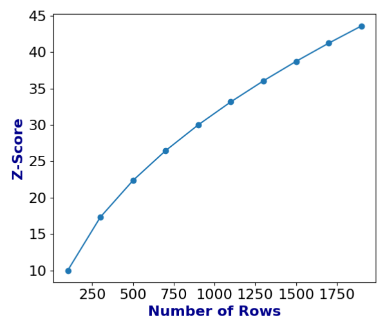


(e) z-score with Gaussian noise

Figure 7: This is a reproduction of Figure 2 from Ngo et al. [29].



(a)



(b)

Figure 8: Plot of additional experiments on Gaussian dataset. Figure 8a plots MSE for more values of c. Figure 8b shows how the z-score changes when more rows are involved in the computation.

Proof of Theorem 2. The Jensen-Shannon Divergence (JSD) measures the similarity between two probability distributions. It is defined as:

$$JSD(P||Q) = \frac{1}{2}D(P||M) + \frac{1}{2}D(Q||M) \tag{2}$$

where $M = \frac{1}{2}(P+Q)$ is the midpoint distribution, and $D(P||Q)$ is the Kullback-Leibler Divergence, defined as: $D(P||Q) = \sum_x P(x)\log(\frac{P(x)}{Q(x)})$.

Let us find: $JSD(\rho||\rho')$. Partition the set of all values X into X_0 and X_1 where X_b consists of those values in X that hashes to bit b. Note that ρ' is only defined on X_0 giving:

$$\rho'(x) = \begin{cases} \frac{\rho(x)}{Z} & x \in X_0 \\ 0 & \text{otherwise} \end{cases}$$

Here, Z is a normalization term needed to ensure that the sum of probabilities in ρ' is 1. Since the hash function is ideal, i.e., maps to 0 and 1 with equal probability, Z is approximately 0.5 or $\rho'(x) = 2 \cdot \rho(x)$ for $x \in X_0$.

Now, let's find the midpoint distribution $M(x) = \frac{1}{2}(\rho(x) + \rho'(x))$. We get:

$$M(x) = \begin{cases} \frac{3}{2}\rho(x) & x \in X_0 \\ \frac{1}{2}\rho(x) & \text{otherwise} \end{cases}$$

Now, we can compute the Kullback-Leibler divergences:

$$\begin{aligned} D(\rho||M) &= \sum_{x \in X} \rho(x) \log(\frac{\rho(x)}{M(x)}) \\ &= \sum_{x \in X_0} \rho(x) \log(\frac{\rho(x)}{\frac{3}{2}\rho(x)}) + \sum_{x \in X_1} \rho(x) \log(\frac{\rho(x)}{\frac{1}{2}\rho(x)}) \end{aligned}$$

Simplifying, we get $D(\rho||M) = 0.5(\log(2) + \log(2/3)) = 0.5 \log(4/3)$. Similarly, we get: $D(\rho'||M) = \log(4/3)$. Plugging this in Equation 2, we get:

$$JSD(\rho||\rho') = \frac{3}{4} \log(\frac{4}{3}) \approx 0.215$$

□

Proof of Proposition 1. Of the m values, we need to compute T_i that ensures that the score is α. We use Equation 1 as:

$$\alpha = \frac{2(T_i - 0.5M)}{\sqrt{M}}$$

Then, $T_i = 0.5M + \alpha\sqrt{M}/2$. In other words, we need at least $0.5M + \alpha\sqrt{M}/2$ values to ensure a Z-score of α. Call this value T_α. □

Proof of Theorem 3. First, observe that for any value val_i such that $\mathcal{H}(val_i) = 0$:

$$\Pr[\mathcal{H}(val_i + \epsilon_i) = 1] = \frac{1}{2}$$

for any $\epsilon_i \xleftarrow{\$} \mathcal{D}$. We already know that one needs at least $T_\alpha = 0.5M + \alpha\sqrt{M}/2$ cells to be unmodified to get a score of α (from Proposition 1). To achieve the watermark removal, we need to add noise to the remaining $N - T_\alpha$ cells. Observe that this follows a hypergeometric distribution - in a sample of size M, N successes exist (i.e., mapping to 0). Then, the expected number of tries to pick at least $(N - T_\alpha)$ successfully is given by: $\approx (N - T_\alpha) \cdot M/N$. Therefore, we get:

$$\mathbb{E}[r] := 2 \cdot (N - T_\alpha) \cdot \frac{M}{N}$$

□

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification:The abstract and introduction encapsulate the paper's contributions: the development of an agent framework featuring fine-grained security tiers, alongside the introduction of a novel benchmark dataset for systematic evaluation of agent behavior in scenarios necessitating privacy protection

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations in Section **??**.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The contribution of this work does involve theoretical results and proofs.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We describe the details of the evaluation experiments we run in Section E and the supplementary material.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code is provided with a readme file containing the instruction of running the experiments. We also provide a validated Croissant file for our dataset JSON file.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We provide the code and also the description of experiments in Section E and the supplementary material.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We include the standard deviation in a numerical format in the experimental sections.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The experimental evaluation details the setting clearly.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: There are no ethical implications of our work nor privacy related concerns. Our queries were received by LLM prompts. The databases were created using Python's faker library and random choice of values. They are completely synthetically generated.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Privacy is a fundamental human right and a paramount concern in the era of AI. Our work advances the protection of sensitive information, keeping in mind regulations.

    Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The results of the paper do not pose such risks as there is no real-world data involved.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: A dedicated section on dataset details is included.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve any crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve any crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.