An Unbiased and Robust Privacy-Preserving Fingerprinting Scheme for Relational Databases

Wenhao Wang, Shujie Cui, Hui Cui, Jiabao Qiu, Shuguang Yuan, Xiaojie Zhu, *Senior Member, IEEE*, Jing Yu, Chi Chen, and Xun Yi

***Abstract*—Sharing relational databases is essential in today's data-driven world for fostering collaboration, enhancing efficiency, and enabling real-time data access. However, privacy and copyright issues arise when sharing privacy-sensitive or valuable data. Additionally, high utility is required in shared data to enable accurate data mining and analysis. Entry-level differentially private fingerprinting schemes (DPFS) could address these concerns. In a DPFS, data can be securely shared without leaking original values while still supporting accurate analysis. Moreover, detectable fingerprints can deter unauthorized redistribution. However, existing DPFSs often lack utility—due to format changes and entry-wise bias—or robustness, as fingerprints can be removed undetected. In this paper, we propose an unbiased and robust differential privacy-based fingerprinting scheme (DPFS), which ensures that the fingerprinted copy remains an unbiased estimate of the original data. By incorporating differential privacy noise, our scheme effectively mitigates alteration, collusion, and hybrid attacks. Our DPFS satisfies ϵ-entry-level differential privacy, enabling clients to conduct unbiased analysis. To improve robustness, we design group-based fingerprint detection, which estimates the mean of injected noise per group with error tolerance. We provide a theoretical robustness analysis and propose a method for achieving optimal robustness. Experiments on four real-world databases show that our scheme consistently detects fingerprints and improves accuracy by up to 20% on machine learning tasks compared to existing DPFSs.**

***Index Terms*—Relational database, fingerprinting, differential privacy, robustness.**

I. Introduction

In THE era of big data, sectors such as finance, where companies regularly exchange large datasets for analysis and insights, face the dual challenge of protecting intellectual property and individual privacy. For example, in Fin-tech, shared relational databases often contain transactional details crucial for market analysis yet sensitive in nature. These scenarios underscore the urgent need for innovative approaches protecting data privacy and copyright, safeguarding both the proprietary interests of database owners and the privacy of the individuals represented within these datasets.

The majority of existing studies on database security (e.g., [1], [2], [3]) address data privacy and copyright protection through a two-stage approach, typically involving data sanitization followed by fingerprint embedding. But these methods frequently alter a substantial number of database entries, significantly compromising the data utility [4]. There have been a few studies (e.g., [4], [5], [6]) leveraging differentially private fingerprinting schemes (DPFS) to simultaneously address both issues by adding the fingerprint embedding based on the differential privacy noise directly to the data. However, existing DPFS schemes lack robustness [6], as the embedded fingerprint can be easily removed or compromised by altering entries within the fingerprinted database. Also, they suffer from poor utility when applied to real-valued databases [4], [5]. When fingerprinting real-valued databases, they first convert the data into intervals. This transformation discards valuable information, making it unsuitable for tasks requiring unbiased estimation—such as accurately estimating the database's mean [7].

The key security property in fingerprinting schemes [8] is called Robustness. There are two types of attacks against robustness [8]: Alteration attacks and Collusion attacks. In the alteration attacks, adversaries alter the entry of the fingerprint copy to remove the fingerprint of the database. For collusion attacks, multiple adversaries collectively eliminate fingerprints by merging multiple copies to generate a new database. A unique type of alteration attack targeting real-valued fingerprinted databases is the noise attack, which involves injecting unbiased noise into fingerprint entries [9], [10]. Due to the unbiased nature of the injected noise, the utility of the fingerprinted database is partially preserved, ensuring that its usability remains intact to a certain extent. Additionally, through our analysis, we found that an adversary may combine noise attacks and collusion attacks to remove fingerprints, which we refer to as a **hybrid attack**. Unfortunately, hybrid attacks are not considered in existing DPFSs [5], [6] (see Section II-B).

Our Design Goals and Approach: In this paper, we aim to design a DPFS for databases that effectively prevents alteration, collusion and hybrid attacks while preserving data

Received 24 March 2025; revised 16 October 2025; accepted 26 November 2025. Date of publication 10 December 2025; date of current version 26 December 2025. The associate editor coordinating the review of this article and approving it for publication was Prof. Wei Ni. *(Corresponding author: Shujie Cui.)*

Wenhao Wang, Shujie Cui, and Hui Cui are with the Faculty of Information Technology, Monash University, Melbourne, VIC 3800, Australia (e-mail: wenhao.wang@monash.edu; shujie.cui@monash.edu; hui.cui@monash.edu).

Jiabao Qiu, Shuguang Yuan, Jing Yu, and Chi Chen are with the Key Laboratory of Cyberspace Security Defense, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100045, China (e-mail: yujing@iie.ac.cn; yuanshuguang@iie.ac.cn; chenchi@iie.ac.cn).

Xiaojie Zhu is with the King Abdullah University of Science and Technology (KAUST), Thuwal 23955, Saudi Arabia (e-mail: xiaojie.zhu@kaust.edu.sa).

Xun Yi is with the School of Computing Technologies, RMIT University, Melbourne, VIC 3000, Australia (e-mail: xun.yi@rmit.edu.au).

Digital Object Identifier 10.1109/TIFS.2025.3642579

utility. One challenge here is to achieve high reliability which is measured by low misdiagnosis false hits (DFH) and low misprosecute false hits (PFH) [8]. Another challenge is to maintain the fingerprint integrity [8] under the disturbance caused by alternation and hybrid attacks.

Inspired by the *ϵ-entry-level fingerprinting* [4], which retains the primary key attribute in the sanitized database to enhance utility for relational data, we propose an ϵ-entry-level fingerprinting scheme for relational databases by injecting unbiased Laplace noise into database entries. We choose Laplace noise for its symmetry around zero, which helps achieve low PFH and DFH while providing stronger privacy guarantees than Gaussian noise [11]. However, other unbiased symmetric differentially private noise can also serve as an alternative [12], [13].

To achieve robustness in the DPFS, our main strategy is to divide the noises into three ranges and randomly divide the entries into many groups. For each entry in each group, we randomly select one of the ranges and add unbiased Laplace noise from that range to the entry. Each entry's noise range and group identifier are used as the fingerprint to detect copy right issues. The noise mean of each group is used to detect any copyright issue, where a tolerance θ is introduced for robustness guarantee.

For the utility evaluation of the DPFS, we check if it supports unbiased mean analysis which it is the most commonly used function in data analysis and forms the basis for many other statistical methods [7], [14]. Ideally, the mean of the shared values is an unbiased estimation of the original values. Since we only inject unbiased noise into the original database, our scheme inherently supports highly accurate mean analysis in an unbiased manner. Additionally, our DPFS also supports most of the other basic operations over relational databases, such as group by and join. Because of the properties supported by our DPFS, our DPFS can be applied in various real-world application to protect data privacy and copyrights. For example, our entry-level DPFS can be adopted in the vertical federated learning (VFL) [15] (where each party holds disjoint attributes of the same relational database) to allow participants sharing relational databases and performing model training without disclosing their own private information.

We summarize our contributions as follows:

1) We propose an entry-level differentially private (DP) fingerprinting scheme employing Laplace noise. The proposed scheme is robust against alteration attacks, collusion attacks, and hybrid attacks while preserving utility. Additionally, the proposed fingerprinting ensures reliability, achieving a PFH of $\frac{1}{2^{2m}}$ and a DFH of $\frac{1}{2^{2m}}$, where m is an adjustable integer parameter.
2) Following the definition of ϵ-entry-level differential privacy, We formally prove that our scheme achieves ϵ-entry-level differential privacy. We also formally analyze the false hit rate and robustness of our scheme against attacks. Based on the robustness analysis, we identify a method to determine the optimal parameter settings for maximizing robustness.
3) We conducted comprehensive experiments in the vertical federated learning scenario to evaluate both utility and robustness of our DPFS. The results show that our approach can improve the utility up to 20%. In terms of the robustness, our scheme can achieve much better success rate in detecting the fingerprint in the scenario of collusion and hybrid attacks comparing to existing works.

II. Related Work and Background

This section discusses the related works on privacy- preserving database fingerprinting (PPFS).

A. Privacy-Preserving Database Fingerprinting

Most existing PPFSs integrate database sanitization and copyright protection through a two-stage process. Bertino et al. [1] utilize the binning method to sanitize a database and then apply a watermark to the sanitized database. Kieseberg et al. [16] and Schrittwieser et al. [2] both employ k-anonymity for data sanitization before embedding fingerprints to secure ownership. Gambs et al. [3] utilize (α, β)-privacy [17] to construct and fingerprint a database, specifically inserting the fingerprint into fake tuples. These methods address copyright protection and data sanitization separately, and the privacy models used by these schemes eliminate part or all of the primary key attributes of the relational database, which impedes certain queries on the shared database.

So far, only Ji et al. [4], Hu et al. [6], and Zhang et al. [5] protect data privacy and copyright in one stage. Like ours at a high level, both use DP techniques, generate fingerprints based on the injected noises, and adopt group-based detection. However, the specific design for the three aspects is different, which makes a significant difference in robustness and utility. Ji et al. [4] pioneered the introduction of entry-level differential privacy for sharing relational data. They proposed an entry-level differential privacy fingerprinting scheme (DPFP) based on the scheme by Li et al. [8]. The least significant bits of the entries are used as the fingerprint. However, the scheme in [4] lacks noise robustness and cannot release continuous attributes. When fingerprinting continuous values, it transforms them into intervals encoded by n bits, then randomly perturbs the last K bits of the interval, preventing the fingerprint value from being an unbiased estimation of its original. This limitation hinders some statistical analyses, such as unbiased mean analysis.

Zhang et al. further extended DPFP [4] by incorporating collusion-resistant coding to enhance collusion robustness. However, it inherits the utility limitations of DPFP, making it unable to achieve high utility on real-valued databases.

Hu et al. [6] use unbiased Gaussian noise for DP fingerprinting on aggregated data without primary key attributes and use the variance of Gaussian noise to form fingerprints. It selects four Gaussian distributions (G_1, G_2, G_3, G_4) and divides the dataset D into four subsets (D_1, D_2, D_3, D_4). Fingerprint insertion involves adding noise from G_i to each entry of D_i. Fingerprint detection extracts the noise from each subset and tests whether the variance of these noises matches the corresponding preset Gaussian distribution. Since noise and collusion attacks alter the variance of the Gaussian noise, this scheme lacks noise and collusion robustness.

B. Attacks

1) Alteration Attack: The most common threats to fingerprinting and watermarking are alteration attacks [18], [19]. Bit-flipping and noise attacks are the most prevalent types of alteration attacks [9], [10], [18], [19], [20]. In a bit-flipping attack, the attacker is assumed to flip the least significant bits [18], [19]. Noise attacks inject unbiased noise into database entries [9], [10].

For fingerprinting schemes, the attacker's capabilities are assumed to be limited, because the attacker needs to preserve the utility of the fingerprinted database; otherwise the attack is meaningless. Thus, the bit-flipping attack assumes that the attacker only flips the least significant bit of the fingerprinted database, and the noise attack assumes that the attacker only injects unbiased noise into the fingerprinted database [9], [10], [19], [20], [21]. Furthermore, current entry-level fingerprinting schemes do not consider column-removal attacks [4], [5], [6], because deleting columns can sometimes prevent downstream machine-learning tasks. For example, in machine-learning tasks, suppose the recipient wants to learn the relationship between the features and the label; if the fingerprinted copy has two columns that are both features, eliminating a column would cause the learning task to fail. In this paper, we follow the assumption in previous work that the adversary does not eliminate columns, and we include a deletion attack [4] in our threat model, where the adversary randomly deletes rows in the database.

2) Collusion Attack: A collusion attack occurs when multiple recipients compare their received fingerprinted copies and use them to generate a new copy without a fingerprint [8], [22]. Collusion attacks can be conducted in many ways. For instance, a majority-collusion attack [23], [24] utilises majority voting to form a colluded database, and simple-collusion attack [25] compares the copies to identify common patterns or differences that indicate fingerprints. We do not consider majority and simple collusion attacks, as they are ineffective against our scheme (see Section III-B). Instead, we focus on mean-collusion and probability-collusion attacks, as they preserve the unbiasedness of the fingerprinted database and do not compromise data utility (see Definitions 7 and 8).

3) Hybrid Attack: A practical adversary may combine alteration attacks and collusion attacks to eliminate fingerprints. However, existing works rarely consider this type of attack, as the collusion robustness in these methods relies on the **merchant assumption** [5], [8], [25], which prevents adversaries from modifying identical content among colluded copies. To the best of our knowledge, no entry-level differentially private fingerprinting scheme has considered such an attack.

In this paper, we incorporate the **Hybrid attack** into our threat model, making it more practical than previous works [4], [5], [6]. Specifically, we define the Hybrid attack as a combination of **noise attacks** and **collusion attacks**. We consider two variants: 1) **Pre-Collusion Attack** – The attacker first applies a noise attack and then proceeds with a collusion attack. 2) **Post-Collusion Attack** – The attacker first applies a collusion attack and then proceeds with a noise attack.

TABLE I
NOTATIONS

Symbol	Description
$\mathbf{R}$	original database
$\mathbf{R}'$	a neighboring database of $\mathbf{R}$
$\mathcal{K}$	the fingerprint
$\widetilde{\mathbf{R}}$	fingerprinted database
$G(x, y)$	(x,y)-th group
$Mean(\cdot)$	Mean value
$\mathbf{R}_i$	i-th row of $\mathbf{R}$
$\mathbf{R}_i[c]$	c-th attribute of $\mathbf{R}_i$
ϵ	privacy budget
Δ	database sensitivity
n	number of noise in $G(x, y)$
n_c	number of colluders
m	number of y in $G(x, y)$
$\mathbf{R}_i.Pmykey$	the primary key attribute of $\mathbf{R}_i$

C. Collusion Codes

The notion of a collusion code was introduced by Boneh et al. [26]. They consider distributing n fingerprinted copies to recipients where any subset of c recipients may collude. The goal is to design a fingerprinting scheme that identifies the colluding c-subset from the observed pirated copy; a code with this property is called *totally c-secure*. They show, however, that no code is totally c-secure when $c \geq 2$ and $n \geq 3$. This led to the relaxed notion of a *c-secure ϵ-error* code, which succeeds with probability at least $1 - \epsilon$. In the same work they prove a theoretical lower bound on the code length, $\frac{1}{2}(c-3)\log\left(\frac{1}{c\epsilon}\right)$, and propose the Boneh–Shaw code that achieves sub-optimal length $\mathcal{O}\left(c^4 \log\frac{1}{\epsilon} \log\frac{n}{\epsilon}\right)$. Subsequent work by Tardos [27] follows this principle and gives a code attaining the length of $\mathcal{O}\left(c^2 \log\frac{n}{\epsilon}\right)$.

Despite their usefulness, code-based schemes have fundamental limitations. First, they rely on the *marking/merchant assumption* [8], which assumes colluders cannot modify positions where their copies agree; this ignores practical hybrid attacks that may alter the same entry across colluders. Second, when the tolerated number of colluders c grows or the target error ϵ becomes small, the required code length becomes very large, incurring significant storage overhead. Since our method does not rely on collusion-resistant codes, it does not inherit these limitations.

III. Privacy Model and Threat Model

In this section, we present the models considered in this paper. We provide the notations to be used in this paper in Table I.

A. Privacy Model

The ϵ-entry-level DP, a variation of ϵ-DP, is proposed to better suit database management system design. The key difference between DP and entry-level DP is that DP removes the primary key attribute in the sanitized database, while entry-level DP preserves it, achieving a better balance of utility and privacy for relational data. The following are the relevant definitions of entry-level DP.

Definition 1: (Neighboring relational databases) [28]: Two relational databases **R** and **R'** are called neighboring databases if they only differ by **one entry**, an attribute value of one individual's record.

Definition 2 (Sensitivity of a Relational Database [28]): Given a pair of neighboring relational databases $\mathbf{R}$ and $\mathbf{R}'$ that differ by one entry (e.g., the c-th attribute of the i-th row, $\mathbf{R}_i[c]$ and $\mathbf{R}_i[c]'$) the sensitivity is defined as $\Delta = sup_{\mathbf{R},\mathbf{R}'}|\mathbf{R} - \mathbf{R}'|_F = sup_{\mathbf{R}_i[c],\mathbf{R}_i[c]'}|\mathbf{R}_i[c] - \mathbf{R}_i[c]'|$, where the *sup* means the supremum.

Definition 3 (ϵ- entry-level differential privacy) [28]: A random mechanism $\mathcal{M}$ with domain $\mathcal{D}$ satisify ϵ-entry-level differential privacy if for any two neighboring relational databases $\mathbf{R}, \mathbf{R}' \in \mathcal{D}$ and for all $S \in \text{Range}(\mathcal{M})$, it holds that $Pr[\mathcal{M}(\mathbf{R}) = S] \leq e^{\epsilon} Pr[\mathcal{M}(\mathbf{R}') = S]$, where $\epsilon > 0$.

Theorem 1 (Composition Theorem) Given two mechanism $\mathcal{M}_1(\cdot)$ satisfy ϵ_1-entry-level-DP and $\mathcal{M}_2(\cdot)$ satisfy ϵ_2-entry-level-DP, their combination satisfy $\epsilon_1 + \epsilon_2$-entry-level-DP (proof sees Appendix F).

Generally speaking, the Definition 3 indicates that one can not distinguish between $\mathbf{R}_i[c]$ and $\mathbf{R}_i[c]'$ by inferring the random result $\widetilde{\mathbf{R}}_i[c] \in S$.

Definition 4 (The Laplace distribution) [14]: The Laplace distribution (centered at 0) with scale b is the distribution with probability density function (p.d.f):

$$Lap(x|b) = \frac{1}{2b} e^{-\frac{|x|}{b}}, x \in (-\infty, +\infty)$$

The variance of this distribution is $2b^2$, and the p.d.f of the distribution is an even function.

Definition 5 (The Laplace mechanism) [14]: Given any function $f: \mathbb{N}^{\chi} \to \mathbb{R}^k$, the Laplace mechanism is defined as:

$$\mathcal{M}(f(x), \epsilon) = f(x) + (Y_1, \ldots, Y_k)$$

where Y_i are i.i.d random variables drawn from $Lap\left(\frac{\Delta}{\epsilon}\right)$ where Δ is the sensitivity of $f(\cdot)$. The Laplace mechanism preserves ϵ-differential privacy [14].

Definition 6 (Unbiased distribution): A probability distribution $\mathcal{F}$, if its expectation $E[\mathcal{F}]$ equals 0, we call $\mathcal{F}$ an unbiased distribution. We call the sample result of $\mathcal{F}$ an unbiased noise.

The Definition 5 is the so-called Laplace mechanism, it can be directly transformed into an ϵ-entry-level mechanism, i.e., an ϵ-entry-level algorithm, by remaining the primary key attribute of the sanitized database. In this paper, we consider that the data owner can customize their privacy guarantee by setting the sensitivity Δ and privacy budget ϵ, and such setting can also be found in [6].

B. Threat Model

As done in previous works [4], [6], [18], we assume the adversary or recipient can only access the published fingerprinted database; other information used when generating or detecting the fingerprint is private. The recipient may maliciously distribute the received fingerprinted copy or alter the copy with either alteration or collusion attacks to bypass fingerprint detection. Specifically, we only consider alteration, collusion and hybrid attacks that do not affect mean analysis. In particular, the collusion attacks considered in this paper are the *mean-collusion attack* and the *probability-collusion attack*. The considered hybrid attack was defined in Section II-B.

In DP, when an attacker sees multiple released results, privacy degrades. Thus, hybrid and collusion attacks reduce the guarantees of entry-level DP in accordance with composition: observing multiple fingerprinted databases causes the privacy loss (budget) to add up (Theorem 1). Moreover, by DP's post-processing property, any further manipulation of the released DP outputs—including collusion or hybrid strategies—cannot increase the privacy loss beyond what composition already accounts for.

The mean-collusion attack [22] utilizes the mean of multiple unbiased estimators to form a collusion database. From the attacker's perspective, such an attack results in a more accurate and thus less private database. This is due to the weak law of large numbers, which states that the mean of multiple unbiased independent estimators converges to the expectation of the unbiased estimator, which is the original sensitive value. In a probability-collusion attack, the colluders construct a new database by randomly selecting a candidate from their copies for each entry [24]. This approach results in an unbiased colluded database, and all attackers share the same risk of being accused.

We do not consider the majority-vote collusion attack [24], where colluders select the most frequent value for each entry across their copies to construct a new database. This attack is ineffective against our scheme because we apply continuous Laplace noise to each entry, ensuring that every fingerprinted entry is unique. As a result, all values appear with equal frequency, making it impossible for the attackers to determine the most frequent ones.

Similarly, we do not consider the simple-collusion attack [6], which creates a new database by identifying identical entries across multiple fingerprinted copies. This attack is not applicable to our method because the continuous noise introduced during the fingerprinting process ensures that no two entries are identical across different fingerprinted databases, thus making the attack ineffective.

Suppose n_c database recipients use their fingerprint databases $\{\widetilde{\mathbf{R}}^1, \ldots, \widetilde{\mathbf{R}}^{n_c}\}$ to perform a collusion attack. Let $Collusion_i = \{\widetilde{\mathbf{R}}^j{}_i | j \in [1, n_c]\}$, and for each attribute c, let $Collusion_i[c] = \{\widetilde{\mathbf{R}}^j{}_i[c] | \mathbf{R}_i^j \in Collusion_i\}$. Formally, the two collusion attacks are defined as below:

Definition 7 (Mean-collusion attack): A mean collusion attack is that for each primary key in the shared databases, calculate $\widehat{\mathbf{R}}_i[c] = Mean(Collusion_i[c]) = \frac{\sum_{j=1}^{n_c} \widetilde{\mathbf{R}}^j{}_i[c]}{n_c}$.

Definition 8 (Probability-collusion attack): A probability collusion attack is that for each primary key in the shared databases, calculate $\widehat{\mathbf{R}}_i[c] = Random(Collusion_i[c])$, where $Random(\cdot)$ means uniformly chose a value from the input set.

IV. Proposed Scheme

In this section, we present the details of the proposed ϵ-entry-level DP fingerprinting scheme, DP-Lap.

A. Overview

We present the system overview in Fig. 5. The colored flags represent the fingerprints assigned to the database recipients. We consider a database owner with a relational database denoted as $\mathbf{R}$. Let $\widetilde{\mathbf{R}}$ be an instance of the fingerprinted database. Our fingerprinting system aims to achieve two goals: (1) privacy and utility guarantees. For each entry $\mathbf{R}_i[c]$, the

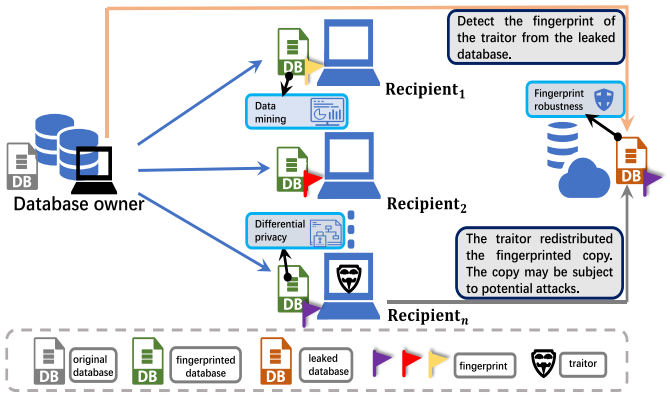

Fig. 1. In the entry-level differential privacy (DP) fingerprinting system, all shared copies meet the requirements for fingerprint robustness, entry-level differential privacy, and data utility.

database recipient cannot distinguish between $\mathbf{R}_i[c]$ and $\mathbf{R}_i[c]'$ by inferring $\widetilde{\mathbf{R}}_i[c]$ in its received copy $\widetilde{\mathbf{R}}$, while the copy still provides valuable information. (2) Robustness guarantee, where the fingerprint is still detected even in the presence of noise attacks or collusion attacks.

1) Privacy and Utility Guarantees: are achieved by injecting i.i.d. unbiased Laplace noise into each entry in the non-primary key part of $\mathbf{R}$, resulting in an entry-level differentially private database $\widetilde{\mathbf{R}}$. The unbiased Laplace noise preserves the statistical utility of the resulting database, thereby satisfying the privacy and utility guarantees.

Algorithm 1 Fingerprint Generation

Input: The original database $\mathbf{R}$, sensitivity Δ, privacy budget ϵ, θ, and group number m.
Output: $\widetilde{\mathbf{R}}$ and $\mathcal{K}$.

```
Generate 3 Laplace distributions (No_0, No_*, No_1) based on (Δ, ϵ, θ)
Generate the chosen distribution Ch based on (Δ, ϵ, θ)
Copy the primary keys from R to R̃ and K
for each R_i[c] excluding the primary key do
    x ←Ch {1, 0, *}
    y ←$ {1, ..., m}
    K_i[c] = (x, y)
    noise ←No_x I_x
    R̃_i[c] = R_i[c] + noise
end for
return The fingerprinted database R̃ and the fingerprint K.
```

2) Robustness Against Attacks: primarily lies in the fingerprint detection method. Our main strategy is to provide tolerance for the detection, and we achieve that with two techniques. First, we divide the entries into groups and detect the mean of a group of entries' noises rather than each single entry. Second, for each group's noise mean, we add a configurable tolerance $\theta \in (0, +\infty)$. Roughly, the noise is selected from $(-\infty, +\infty)$. In our approach, when injecting noise, we divide the range $(-\infty, +\infty)$ into 3 sub-ranges: $(-\infty, -\theta]$, $(-\theta, \theta)$, and $[\theta, +\infty)$, and the noise added into each entry is selected from one of the ranges. For fingerprint detection, we only consider the entries whose noises are in $(-\infty, -\theta]$ or $[\theta, +\infty)$. Specifically, for each group, the noise means of the entries whose recorded ranges are in $(-\infty, -\theta]$ and $[\theta, +\infty)$ should be greater and less than 0, respectively. Indeed, they should be greater/less than $\theta/-\theta$. To be robust against attacks, we set it to 0. By doing so, our approach can tolerate noise alteration (see analysis in Section V). For detection, each entry's noise range and group are stored secretly in a relational table $\mathcal{K}$ on the data owner side.

B. Fingerprint Generation

We first give the basic algorithm for generating the fingerprint with heavy storage overhead. Later we provide the optimised version that reduces the storage space to $\mathcal{O}(1)$ based on the pseudorandom number generator. The detail of generating fingerprint is shown in Algorithm 1.

1) Parameters: The process takes 4 parameters: sensitivity Δ, privacy budget ϵ, θ, and group number m, as the input in addition to the original database and outputs a fingerprinted database $\widetilde{\mathbf{R}}$ and the fingerprint $\mathcal{K}$. The sensitivity Δ and the privacy budget ϵ determine the security guarantee of DP.

θ is a positive real number, and it affects the robustness of our scheme. In section V, we formally analyze how θ affects the robustness against noise and collusion attacks. In particular, robustness reaches its peak when $\theta = 2 \cdot \frac{\Delta}{\epsilon}$ (refer to Section V).

The group number m must be smaller than the total number of entries in the database, and it determines the false-hit rate and inversely affects the robustness of our scheme. More groups will achieve a low false-bit rate but reduces the robustness of the system (formal analysis is given in Section V).

2) Noise Generation: Our algorithm first derives 3 probability distributions, denotes as No_0, No_1 and No_*, based on the input parameters from a well-designed Laplace distribution $Lap\left(\frac{\Delta}{\epsilon}\right)$. The noise in them are selected from $I_0 = (-\infty, -\theta]$, $I_1 = [\theta, \infty)$, and $I_* = (-\theta, \theta)$ respectively, with the probability density functions (p.d.f.) shown in eq (1).

$$No_i(x) = \begin{cases} \dfrac{Lap\left(x|\frac{\Delta}{\epsilon}\right)}{\int_{I_i} Lap\left(t|\frac{\Delta}{\epsilon}\right) \mathrm{d}t}, & x \in I_i, \\ \quad i \in \{0, 1, *\} & \\ 0, & otherwise \end{cases} \tag{1}$$

3) Noise Selection: To achieve DP privacy, the noise range used for each entry is selected based on the distribution defined by eq (6). In the equation, Λ is the integral of $Lap\left(x|\frac{\Delta}{\epsilon}\right)$ over the interval $[\theta, \infty)$, since $Lap\left(x|\frac{\Delta}{\epsilon}\right)$ is an even probability dense function (p.d.f), the integral of $Lap\left(x|\frac{\Delta}{\epsilon}\right)$ over the interval $(-\infty, \theta]$ is also Λ and the integral of $Lap\left(x|\frac{\Delta}{\epsilon}\right)$ over the interval $(-\theta, +\theta)$ is $1 - 2\Lambda$.

$$\begin{cases} Pr(Ch = 0) = \int_{I_0} Lap\left(x|\frac{\Delta}{\epsilon}\right) \mathrm{d}x = \Lambda \\ Pr(Ch = 1) = \int_{I_1} Lap\left(x|\frac{\Delta}{\epsilon}\right) \mathrm{d}x = \Lambda \\ Pr(Ch = *) = \int_{I_*} Lap\left(x|\frac{\Delta}{\epsilon}\right) \mathrm{d}x = 1 - 2\Lambda \end{cases} \tag{2}$$

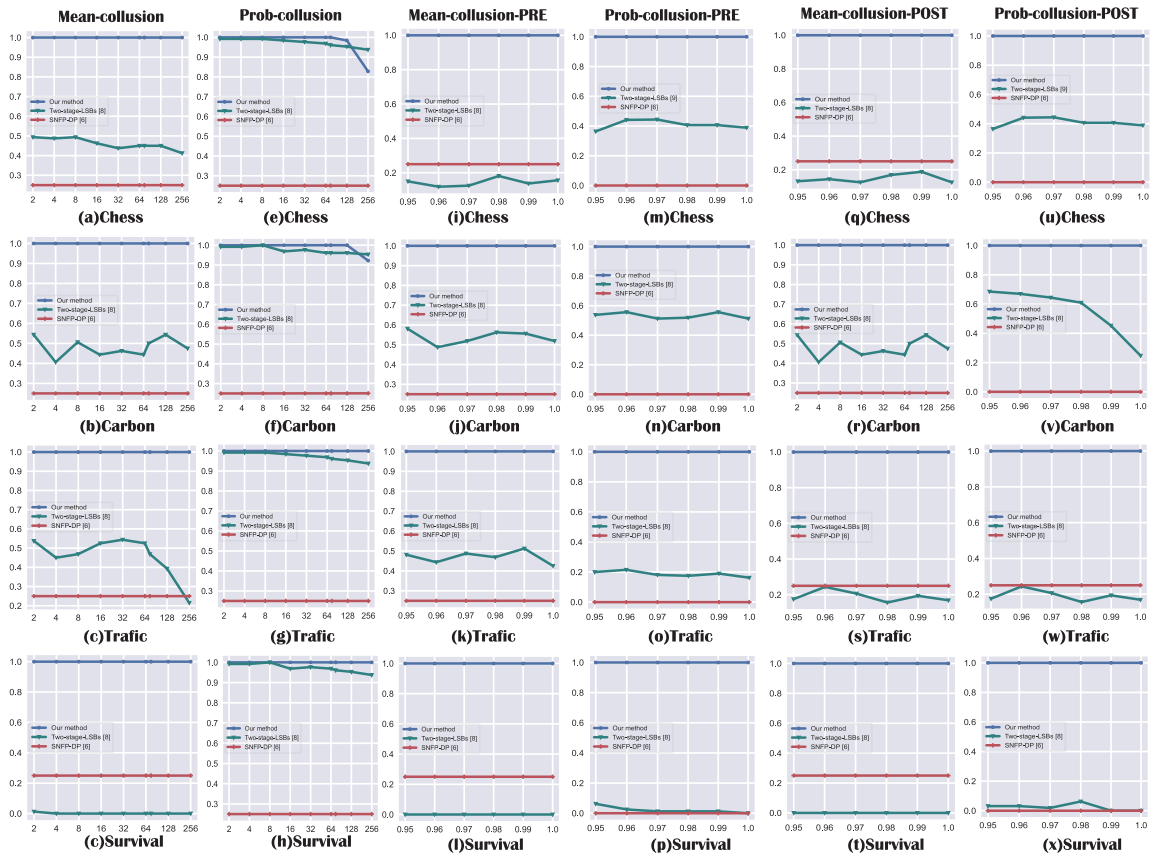

Fig. 2. Fingerprint recovery results under collusion and hybrid attacks. The x-axis in (a)–(h) represents the number of colluding attackers, while in (i)–(x), it denotes the attack rate.

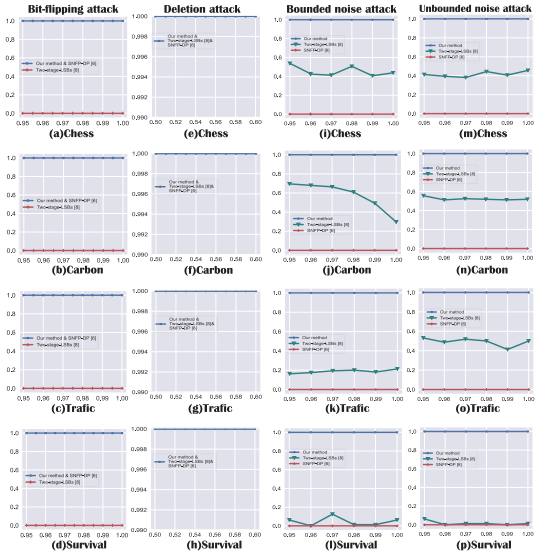

Fig. 3. Fingerprint recovery results against various alteration attacks. The x-axis in (a)-(p) represents attack rates, with $\epsilon = 1$, $\Delta = 2.5$.

The noise range x and a random number in $[1, m]$ determine each entry's group tag (x, y) and is stored in the database $\mathcal{K}$. Note that the size of $\mathcal{K}$ can be reduced to $2m$ from N by storing a list of the entry locations, (i, c), to each group (x, y), where $x \in \{0, 1\}$.

Theorem 2:: Noise generated by Algorithm 1 follows $Lap\left(\frac{\Delta}{\epsilon}\right)$.

Proof: Let NO be the noise distribution of the noise generated by Algorithm 1, we can calculate the p.d.f. of NO, shown in eq (5).

$$NO(x) = \begin{cases} No_0(x) \cdot Pr(Ch = 0) = Lap\left(x|\frac{\Delta}{\epsilon}\right), x \in \left(-\infty, \theta\right] \\ No_*(x) \cdot Pr(Ch = *) = Lap\left(x|\frac{\Delta}{\epsilon}\right), x \in (-\theta, \theta) \\ No_1(x) \cdot Pr(Ch = 1) = Lap\left(x|\frac{\Delta}{\epsilon}\right), x \in \left[\theta, +\infty\right) \end{cases} \tag{3}$$

The noise distribution NO is the well-designed Laplace distribution $Lap\left(\frac{\Delta}{\epsilon}\right)$. The proof is completed.

Theorem 3:: Algorithm 1 is ϵ-entry-level differentially private.

Proof: Since we consider neighboring databases that have only a pair of different data entries that differ by at most Δ. Then, by applying Definition 3, we can have eq (10).

$$\begin{aligned} \frac{Pr(\mathcal{M}(\mathbf{R})) = \widetilde{\mathbf{R}})}{Pr(\mathcal{M}(\mathbf{R}')) = \widetilde{\mathbf{R}})} &\overset{(1)}{=} \frac{Pr(\mathbf{R}_i[c] = \widetilde{\mathbf{R}}_i[c])}{Pr(\mathbf{R}_i[c]' = \widetilde{\mathbf{R}}_i[c])} \\ &\overset{(2)}{=} \frac{e^{-\frac{\epsilon|\mathbf{R}_i[c] - \widetilde{\mathbf{R}}_i[c]|}{\Delta}}}{e^{-\frac{\epsilon|\mathbf{R}_i[c]' - \widetilde{\mathbf{R}}_i[c]|}{\Delta}}} \overset{(3)}{\leq} e^{\frac{\epsilon|\mathbf{R}_i[c] - \mathbf{R}_i[c]'|}{\Delta}} \overset{(4)}{\leq} e^{\epsilon} \end{aligned} \tag{4}$$

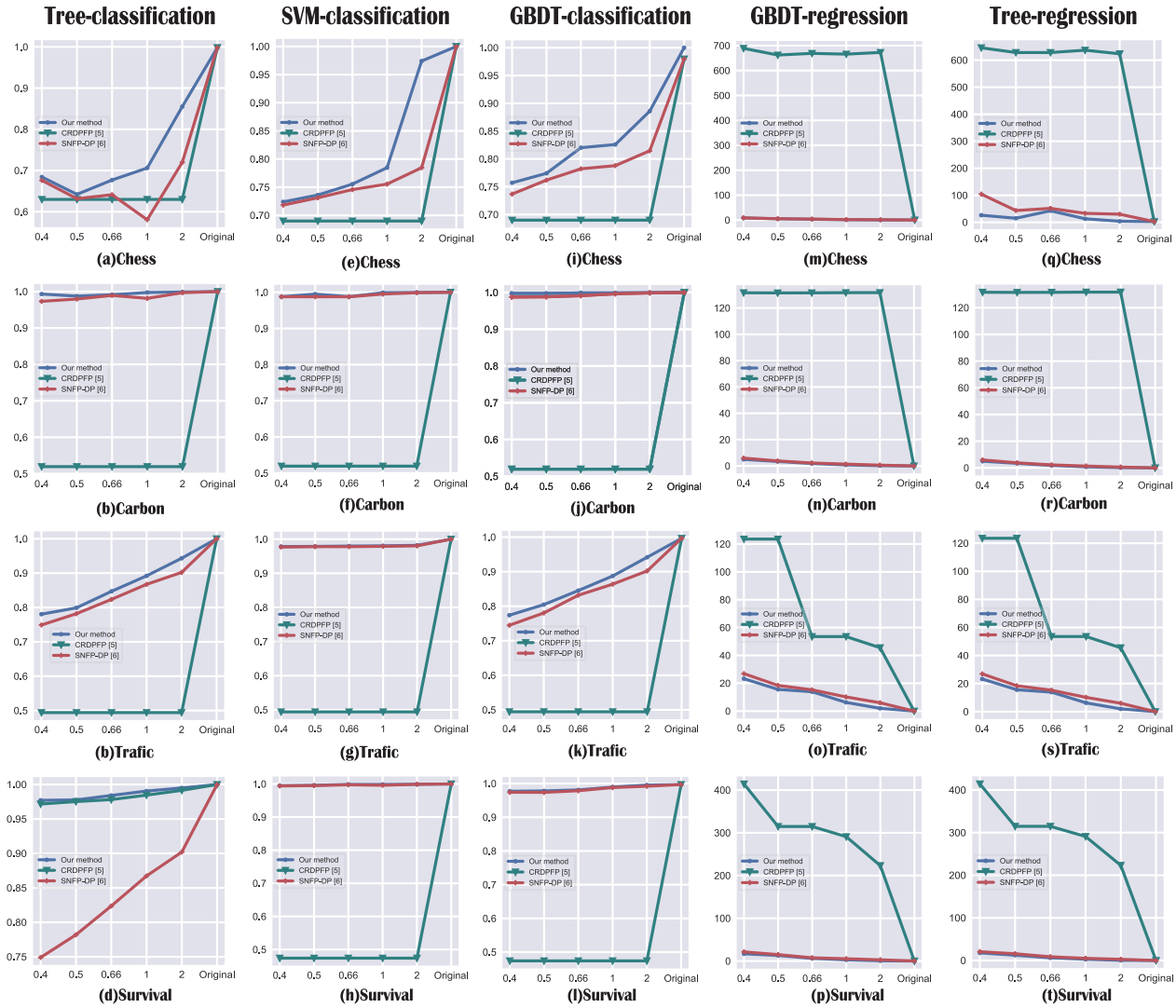

Fig. 4. Machine learning utility results on various privacy budgets with fixed sensitivity 1, the y-axis in (m)-(t) denotes the mean square error, y-axis in (a)-(l) denotes the correct ratio.

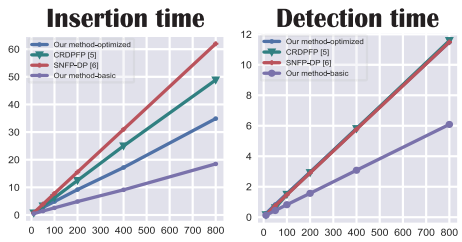

Fig. 5. In the figure, the x-axis represents the number of entries in the database, where each unit corresponds to 10,000 entries. For example, a value of 1 on the x-axis represents 10,000 entries in the database, 2 represents 20,000 entries, and so on. The y-axis represents the processing time in seconds.

In the equation, $\overset{(1)}{=}$ can be obtained by assuming (without loss of generality) that **R** and **R'** differ at cth attribute of ith row, and thus the probability ratio at other entries cancel out. $\overset{(2)}{=}$ can be obtained by Theorem 1, $\overset{(3)}{\leq}$ can be obtained by the triangle inequality: $-(|\mathbf{R}_i[c]-\widetilde{\mathbf{R}}_i[c]|-|\mathbf{R}_i[c]'-\widetilde{\mathbf{R}}_i[c]|) \leq |\mathbf{R}_i[c]-\mathbf{R}_i[c]'|$, $\overset{(4)}{\leq}$ is because $\mathbf{R}_i[c]$ and $\mathbf{R}_i[c]'$ differ by at most Δ, the proof is complete.

4) Privacy Loss: The entry-level differential privacy follows composition theorem 1, i.e., the privacy budget will add up in different published databases. Therefore, privacy loss will increase as the number of fingerprint databases increases. The owner has to limit the number of published databases based on the privacy budget.

5) Storage Overhead Optimisation: For each database share, the owner has to store a database $\mathcal{K}$ to detect the fingerprint. Thus, the storage overhead is $\mathcal{O}(n \cdot N)$, where N is the size of the published database, n is the number of published databases. Since the number of published databases is limited due to privacy concerns, $\mathcal{O}(n \cdot N)$ is usually affordable. However, the storage space will be heavy for large size databases. Considering $\mathcal{K}$ only stores the group tags (x, y), to reduce the storage overhead, we can use a pseudorandom generator *PRG* to generate x and y with a seed, rather than select them randomly. As a result, for each database share,

Algorithm 2 Optimised Fingerprint Generation

Input: The original database $\mathbf{R}$, sensitivity Δ, the privacy budget ϵ, robustness parameter θ such that $\int_{\theta}^{+\infty} Lap\left(x|\frac{\Delta}{\epsilon}\right) dx = \frac{p}{2q}$, where p, q are relatively primary numbers, group number m, and the security parameter λ

Output: $\widetilde{\mathbf{R}}$ and seed $\mathcal{S}$.

Generate three noise distributions (No_0, No_*, No_1) based on $(\Delta, \epsilon, \theta)$
Construct three sets $K_0 = \{1, 2, \ldots, p\}$, $K_* = \{p+1, q+2, \ldots, 2q-p\}$, $K_1 = \{2q-p+1, \ldots, 2q\}$
$\mathcal{S} \leftarrow \{0,1\}^{\lambda}$
for each $\mathbf{R}_i[c]$ excluding primary keys **do**
  $t \leftarrow PRG(\mathcal{S}||\mathbf{R}_i.Pmykey||c||1) \mod 2q$
  $x = l$, where $t+1 \in K_l$
  $y \leftarrow (PRG(\mathcal{S}||\mathbf{R}_i.Pmykey||c||2) \mod m) + 1$
  $noise \overset{No_x}{\leftarrow} I_x$
  $\widetilde{\mathbf{R}}_i[c] = \mathbf{R}_i[c] + noise$
end for
return $\widetilde{\mathbf{R}}$ and $\mathcal{S}$.

the data owner just needs to store a seed and re-generate (x, y) when detecting fingerprint.

The specific algorithm to optimize the storage overhead is shown in Algorithm 2. In the original algorithm, y is randomly selected from $[1, m]$, and it can be replaced with $y = (PRG(\mathcal{S}||\mathbf{R}_i.Pmykey||c||2) \mod m) + 1$ without affecting the security since the pseudorandom generator is usually considered indistinguishable from a real random generator. The challenge here is to generate x of each entry following the distribution defined in eq (6) so that the noise added to each entry can satisfy ϵ-entry level differential privacy. Our main idea is to use the pseudorandom generator (PRG) to simulate the distribution Ch as defined in eq (6). Specifically, we construct 3 sets: $K_0 = \{1, 2, \ldots, p\}$, $K_* = \{p+1, q+2, \ldots, 2q-p\}$, $K_1 = \{2q-p+1, \ldots, 2q\}$, where p and $2q$ are coprime and $\frac{p}{2q} = \int_{\theta}^{+\infty} Lap\left(x|\frac{\Delta}{\epsilon}\right) \mathrm{d}x$.[1] By computing $t \leftarrow PRG(\mathcal{S}||\mathbf{R}_i.Pmykey||c||1) \mod 2q$, we can get a random value $t \in [0, 2q-1]$. Since both K_0 and K_1 contains p elements, it is clear that $Pr(t+1 \in K_l) = \frac{p}{2q} = \Lambda = Pr(Ch = l)$ for $l \in \{0, 1\}$.

Theorem 4:: Algorithm 2 satisfy ϵ-entry level differential privacy.

Proof: We follow the proof of Theorem 3 and need only to demonstrate that the noise injected into each entry is independently and identically distributed (i.i.d.) unbiased Laplace noise. Consequently, based on the proof of Theorem 3, we can establish that Algorithm 2 is ϵ-entry level differentially private.

It is evident that the noise produced by Algorithm 2 spans over $(-\infty, +\infty)$. Next, we prove that the noise that follows the distribution $Lap\left(\frac{\Delta}{\epsilon}\right)$.

We follow the proof of Theorem 3. Recall for each entry $\mathbf{R}_i[c]$, the seed of the pseudorandom generator PRG is $\mathcal{S}||\mathbf{R}_i.PmyKey||c||1$, since the seed to generate noise is unique for each entry, thus, the random number generated by PRG can be deemed as independent. Let NO be the noise distribution of the noise generated by Algorithm 2. We can calculate the p.d.f. of NO, shown in eq (5).

$$NO(x) = \begin{cases} No_0(x) \cdot Pr(t+1 \in K_0) = Lap\left(x|\frac{\Delta}{\epsilon}\right), \\ \qquad x \in (-\infty, \theta] \\ No_*(x) \cdot Pr(t+1 \in K_*) = Lap\left(x|\frac{\Delta}{\epsilon}\right), \\ \qquad x \in (-\theta, \theta) \\ No_1(x) \cdot Pr(t+1 \in K_1) = Lap\left(x|\frac{\Delta}{\epsilon}\right), \\ \qquad x \in [\theta, +\infty) \end{cases} \tag{5}$$

Algorithm 3 Fingerprint Detection

Input: The original database $\mathbf{R}$, the fingerprinted/leaked database $\widetilde{\mathbf{R}}$, the fingerprint $\mathcal{K}$.

Output: Fail | Succeed

for each $\widetilde{\mathbf{R}}_i[c]$ excluding primary keys **do**
  $(x, y) \leftarrow \mathcal{K}_i[c]$
  $G(x, y) \leftarrow G(x, y) \cup (\widetilde{\mathbf{R}}_i[c] - \mathbf{R}_i[c])$
end for
for each group $G(x, y)$ **do**
  if $x = 1$ and $Mean(G(x, y)) \leq 0$ **then**
    return False
  end if
  if $x = 0$ and $Mean(G(x, y)) \geq 0$ **then**
    return False
  end if
end for
return Succeed

The noise distribution NO is the well-designed Laplace distribution $Lap\left(\frac{\Delta}{\epsilon}\right)$. According to the proof procedure in Theorem 3, we can conclude that Algorithm 2 satisfies ϵ-entry level differential privacy.

C. Fingerprint Detection

Fingerprint detection aims at identifying if a database $\widetilde{\mathbf{R}}$ a fingerprinted copy of the database owner's database $\mathbf{R}$, where the two databases have the same primary keys and the same number of records and attributes. Our fingerprint detection is group-based. We first extract the noise data for each entry by subtracting the two databases. Note that the two databases are aligned first based on the primary keys. Based on the group tag recorded in $\mathcal{K}$ or re-generated based on the seed, we calculate the mean of the noises with the same group tag (x, y). Finally, we check if the mean of each group with $x = 0$ is greater than 0 and the mean of each group with $x = 1$ is smaller than 0. If it is true for all those groups, $\widetilde{\mathbf{R}}$ is a fingerprinted copy of $\mathbf{R}$.

1) Correctness: Correctness is the property that if a fingerprint is successfully inserted into a database, it will also be successfully extracted from that database. Recall that in Algorithm 1, the value of x indicates the noise is selected from No_x, and all the noises in No_1 and No_0 are in ranges

[1]We set $pand2q$ to coprime to ensure that the sizes of these sets are as tight as possible.

$[\theta, +\infty)$ and $(-\infty, -\theta]$, respectively. Thus, the noises marked with $x = 1$ are all greater than θ and marked with $x = 0$ are all less than $-\theta$. So their mean must be greater than 0 and less than 0, respectively. If $\widetilde{\mathbf{R}}$ is a fingerprinted copy of $\mathbf{R}$ and never be modified, the detection must output succeed.

Note that there is a common unresolved issue in fingerprinting schemes, i.e., the fingerprint does not work when the database contains a small number of entries. In our scheme, the group number $2m$ must be smaller than the total number of entries in the database, and it will not be secure for databases with fewer than $2m$ entries. The schemes proposed in [4], [6], and [8] also encounter similar issues.

D. False Hit

1) Misprosecution False Hits: As done in previous work [6], [8], [29], we consider the scenario where a fingerprinted copy is leaked, and the detection process aims to identify the traitor among all copyholders. The *misprosecution false hit* is defined as the probability that an innocent copyholder is falsely detected as the traitor.

Our scheme has a low misprosecution false-hit rate because it relies on group-based detection and the groups for each fingerprinted copy are determined randomly. In our scheme, the false hit happens only when the noises of the leaked copy satisfy the $2m$ detection conditions of the innocent copyholder's fingerprint $\mathcal{K}_I$. Let us first see the probability for one group, say $G(x = 1, y = 1)$. Assume the first k entries of the database belong to $G(1, 1)$. To satisfy the detection condition of this group, the noise mean of the first k entries in the leaked database should be greater than 0 since $x = 1$. Indeed, their noises are selected from $(-\infty, +\infty)$ based on the Laplace distribution $Lap\left(\frac{\Delta}{\epsilon}\right)$ (Theorem 1). Considering the p.d.f. of $Lap\left(\frac{\Delta}{\epsilon}\right)$ is an even function, i.e., centered at 0, the actual noise mean of the first k entries in the leaked database is greater than 0 with a probability of $\frac{1}{2}$. There is a false hit only when all the $2m$ groups satisfy the conditions, the probability of which is $\frac{1}{2^{2m}}$.

2) Misdiagnosis False Hits: A *misdiagnosis false hit* means when a fingerprint is detected in a database that does not contain a fingerprint. In this case, the noise mean of each group is unknown; thus, by natural assumption, the probability of the mean of each group being greater or less than 0 is $\frac{1}{2}$. Detection succeeds only when all detectable groups satisfy their detection condition, so the probability of a misdiagnosis false hit is also $\frac{1}{2^{2m}}$.

In general, $2m = 128$ is considered a secure false hit rate [28], [29], which means $m > 64$.

E. Fingerprint Generation and Detection Complexity Analysis

In the fingerprint generation step (Algorithm 2), for each entry, we generate the tag (x, y) by calling a PRG twice and insert Laplace noise. Each entry's insertion takes $\mathcal{O}(2PRG)$ time, resulting in an overall insertion complexity of $\mathcal{O}(2nPRG)$, where n is the number of entries in the database.

For fingerprint detection, the process involves generating the tag (x, y) for each entry using the PRG, then extracting and assigning the noise to a group based on the tag (x, y). This operation also has a time complexity of $\mathcal{O}(2nPRG)$. After assigning the noise, we calculate the mean noise for each group. Notably, the sum of the noise for each group can be accumulated during the assignment, so calculating the mean for each group takes only $\mathcal{O}(1)$ time. Since there are $2m$ groups, the time complexity for this step is $\mathcal{O}(m)$. Thus, the total detection time complexity is $\mathcal{O}(m + 2nPRG)$.

F. Sensitivity Setup

In this work, sensitivity is *set by the data owner* rather than computed as the global range (max–min) as in traditional local DP [30], or enforced via clipping to a fixed threshold C as in DP machine learning [31]. We adopt the *metric DP* setting [32], [33], [34], where the privacy guarantee need not be uniform across all values, enabling a better utility–privacy trade-off. In our method we add Laplace noise with scale b, so any neighboring pair with sensitivity Δ receives differential-privacy protection with budget $\epsilon = \Delta/b$. This metric-DP sensitivity choice is standard in DP fingerprinting: [6] lets the data owner choose the sensitivity, and [4], [5] adopt the same setting on real datasets. Accordingly, we also adopt the metric differential privacy (DP) framework, where the data owner customizes Δ. We then require ϵ–entry-level DP *only* for neighboring databases whose differing entries change by less than Δ.

V. Robustness Analysis

We refer to the lower bound of probability of each detectable group $G(x, y)$ satisfying its detection condition under attacks as the metric of robustness. And we analyze the robustness under the fixed Δ and ϵ.

A. Robustness Against Noise Attacks

Our fingerprint detection is group-based, and the detection succeeds only if all detectable groups satisfy their detection conditions. Thus, we demonstrate the robustness of our scheme by showing that the probability of each detectable group $G(x, y)$ satisfying its detection condition under a noise attack is high. When subjected to a noise attack, the probability that each detectable group $G(x, y)$ satisfies its detection condition $D_{x,y}$ is greater than $\left(1 - \frac{Var[\mathcal{F}]}{n \cdot \theta^2}\right)$ where $Var[\mathcal{F}]$ is the variance of the noise distribution used in the noise attack, n is the number of the extracted noises in $G(x, y)$. It demonstrates when $n \cdot \theta^2$ is large, the noise robustness is achieved. For the detailed calculation see Appendix A.

The following lemma reveals n is proportional to the size of the fingerprinted database.

Lemma 1: Let n be the number of noise data in the detectable group $G(x, y)$, n is proportional to the number of entries in the fingerprinted database $\widetilde{\mathbf{R}}$.

Proof: We can calculate the probability of each entry in $\widetilde{\mathbf{R}}$ is assigned a group tag (x,y) $(x \in \{0, 1\}, y \in [1, m])$ equal to $Pr(Ch = x)\frac{1}{m} = \Lambda \cdot \frac{1}{m}$, therefore, we can calculate $n \approx N \cdot \gamma \cdot \Lambda \cdot \frac{1}{m}$, where N is the number of rows in $\widetilde{\mathbf{R}}$, γ is the number of attributes in $\widetilde{\mathbf{R}}$, Λ is $Pr(Ch = x)$, (eq(6)), thus n is proportional

to $N \cdot \gamma$, the number of entries in the fingerprinted database $\widetilde{\mathbf{R}}$. The proof is complete.

$$\begin{cases} Pr(Ch = 0) = \int_{I_0} Lap\left(x|\frac{\Delta}{\epsilon}\right) \mathrm{d}x = \Lambda \\ Pr(Ch = 1) = \int_{I_1} Lap\left(x|\frac{\Delta}{\epsilon}\right) \mathrm{d}x = \Lambda \\ Pr(Ch = *) = \int_{I_*} Lap\left(x|\frac{\Delta}{\epsilon}\right) \mathrm{d}x = 1 - 2\Lambda \end{cases} \tag{6}$$

B. Collusion Robustness Analysis

Collusion robustness refers to the ability to successfully detect the traitor's fingerprint from the colluded copy with a high probability.

Here again, we demonstrate collusion robustness by showing that when detecting the traitor's fingerprint from the colluded copy, each detectable group has a high probability of satisfying its detection condition.

For mean-collusion, we can calculate that the probability of each detectable group $G(x, y)$ satisfying its detection condition is greater than $\left(1 - \frac{2(n_c-1)\cdot(Var[L])}{n\cdot\theta^2}\right)$, where n_c is the number of copies used in collusion, L is the Laplace distribution following $Lap\left(\frac{\Delta}{\epsilon}\right)$. This shows that given fixed values of n_c, Δ and ϵ collusion robustness is achieved when $n \cdot \theta^2$ is large (for detailed calculations, see Appendix B).

For probability-collusion, we can calculate the probability of each detectable group $G(x, y)$ satisfying its detection condition is greater than $\left(1 - \frac{(n_c^2-n_c)Var[L]}{n\cdot\theta^2}\right)$. (for detailed calculations, see Appendix C).

C. Robustness Against Hybrid Attacks

Since a hybrid attack combines noise and collusion attacks, its robustness can be inferred from the robustness against these individual attacks. So we omit the independent robustness analysis against hybird attack.

D. Collusion False Hit Analysis

We calculate the probability of prosecuting the innocent for collusion, i.e., the probability of using the innocent fingerprint $\mathcal{K}_I$ to successfully detect the fingerprint from the colluded copy. The probability of a collusion false hit is $\frac{1}{2^{2m}}$. Usually, when $2m = 128$, it is considered false-hit secure (for detailed calculations, see Appendix D and Appendix E).

E. Robustness Parameter θ

A fundamental question arises: how do we determine the robustness parameter, θ, to maximize the robustness of the fingerprint? Referring to the noise-robustness and collusion-robustness analyses in Appendices A, C, and B, it follows that robustness is maximized when $n\theta^2$ is maximized, since for a detection group $G(x, y)$ the lower bound on the detection-success probability is maximized at the same point.

As per the proof provided in Lemma 1, we have $n \approx N \cdot \gamma \cdot \Lambda \cdot \frac{1}{m}$, where $\Lambda = \int_{\theta}^{+\infty} Lab\left(x|\frac{\Delta}{\epsilon}\right) = \frac{1}{2} \cdot e^{-\frac{\theta\epsilon}{\Delta}}$. With γ and m fixed, maximizing $n \cdot \theta^2$ entails finding the extreme point of $e^{-\frac{\theta\epsilon}{\Delta}} \cdot \theta^2$. This can be achieved by determining the zero point of its partial derivative, $\frac{\partial\left(e^{-\frac{\theta\epsilon}{\Delta}}\cdot\theta^2\right)}{\partial\theta} = 0$, yielding $\theta = 2 \cdot \frac{\Delta}{\epsilon}$. Thus, when $\theta = 2 \cdot \frac{\Delta}{\epsilon}$, $n \cdot \theta^2$ attains its maximum, thereby achieving peak robustness.

According to the above analysis, since $n \approx N \cdot \gamma \cdot \Lambda \cdot \frac{1}{m}$ and the robustness is proportional to $n \cdot \theta^2$, we can see that the robustness is inversely proportional to m. Therefore, we want m to be minimized while maintaining a low false hit rate, thus in our experiments, we set m to be 64.

VI. Experiment and Analysis

We evaluate the robustness, utility, and performance of our scheme using real databases. All experiments were conducted on a single machine equipped with an Intel i7-10750H CPU (2.60GHz, up to 2.59GHz) and 16.0 GB of RAM, running Windows 10 OS.

The robustness is evaluated by assessing the recover rate of the fingerprint when it is subjected to attacks. We evaluate the utility in two ways. We first measure the mean average error under different privacy parameters. Second, we train machine learning tasks with the datasets protected with our scheme and evaluate the model's accuracy. For performance evaluation, we measure the fingerprint generation and detection times. In this experiment, we did not compare DPFP [4]; instead, we compared its collusion-resistant version, CRDPFP [5]. CRDPFP is adapted from DPFP, inheriting all its properties while further enhancing collusion resistance (refer to Section II-A).

A. Dataset

We conduct experiments on four datasets: the **Carbon Dataset**[2] contains 8,064,821 rows with two continuous attributes, *Latitude* and *Longitude*. The **Chess Dataset**[3] contains 6,101,773 rows with *WhiteRatingDiff* and *BlackRatingDiff*. The **Traffic Dataset**[4] consists of 87,820,726 rows with two continuous attributes, *Light* and *Raining*. The **Survive Dataset**[5] contains 88,809,775 rows with *Age start* and *Age end*.

For the **Carbon, Chess, Traffic, and Survive** datasets, we extracted the continuous attributes and assigned an additional *id* attribute as the primary key. Together, these attributes form relational databases for our study.

B. Robustness Evaluation

As demonstrated in Section V on robustness analysis, the robustness is proportional to the number of entries in the database. This relationship motivates us to test the robustness with datasets of different sizes. Four datasets obtained from Kaggle with a difference of approximately 4 million entries are used for this test.

[2]https://www.kaggle.com/datasets/epa/carbon-monoxide
[3]https://www.kaggle.com/datasets/arevel/chess-games
[4]https://www.kaggle.com/datasets/xxjcaxx/trafficsimulator?select=streets.csv
[5]https://www.kaggle.com/datasets/louise2001/survival-analysis-synthetic-data

1) Parameters Setup: In the robustness experiment, we set the sensitivity Δ to 2.5 and the privacy budget ϵ to 1. We set θ to $2 \cdot \frac{\Delta}{\epsilon}$ (this setting maximizes the noise robustness and collusion robustness as discussed in Section IV-B) and set m to 64. The parameters of the baseline schemes are the same as ours, and the parameter δ in the Gaussian mechanism from [6] set to 10^{-5}.

2) Baselines: We evaluate the robustness with real-valued data. For a fair comparison, we do not directly compare our scheme with [4], [5], as their approach projects real-valued data into intervals, altering the data format and significantly compromising the utility of the database (refer to Section II for a detailed explanation). Instead, we compare our scheme with the method proposed in [8], which serves as the original version of [4] and [5] and is designed for real-valued data. Specifically, we evaluate robustness against the two-stage LSB fingerprinting approach proposed in [8] and the Laplace mechanism. To achieve this, we embed the fingerprint using the method from [8] into the least significant bits of the database after it has been sanitized by the Laplace mechanism.

We also evaluate the robustness of our method against the state-of-the-art DP fingerprinting scheme, SNFP-DP [6], which was originally designed for aggregated data (i.e., data without a primary key). For our experiments, we adapt this scheme to work with relational data.

3) Alteration Attack: We use the following three alteration attack categories to perform robustness experiments: bit-flipping attack, noise attack, and deletion attack. For both bit-flipping and noise attacks, we increase the proportion of altered entries from 95% to 100% and pick entries randomly. For the bit-flipping attack, we flipped all the bits of the picked entry's fractional part. We employed two kinds of noise attacks, namely, bounded noise attack by injecting noise following the uniform distribution $U(-1.5, 1.5)$ and unbounded noise attack by injecting the noise following the Laplace distribution $Lap(2)$. For the deletion attack, we randomly remove 50% to 60% rows from the fingerprinted database.

Robustness is measured by the fingerprint recovery rate, which is calculated as the number of groups satisfying the detection conditions divided by the total number of fingerprint groups. We use this criterion because the two baseline methods also use group-based detection.

The results in Fig. (3) demonstrate that the fingerprint recovery rate of our scheme remains 1 across all test cases. This indicates that, regardless of the number of entries altered by the attacker, our scheme can successfully detect the fingerprint. Whereas, the two-stage LSBs scheme has low robustness against both bit-flipping and noise attacks, and the SNFP-DP scheme has low robustness against noise attacks.

The two-stage LSBs scheme only inserts fingerprints in the least significant bits (LSBs), which leads to low robustness against bit-flipping and noise attacks. The low noise robustness of SNFP-DP is due to its use of variance as fingerprints. The SNFP-DP scheme first extracts the noise data by subtracting the fingerprinted data from the original data and then calculates the variance of the extracted noise data. The detection process checks whether the variance equals a preset value. Since a noise attack affects the variance calculation, SNFP-DP has low noise robustness.

4) Collusion Attack: For the collusion attack, we tested the robustness of our scheme against both mean-collusion and probability-collusion attacks (definition 7 8). Again, we measure the fingerprint recovery rate in different cases where the number of colluding members increases from 2 to 256. The experimental results are shown in Fig. (2).

For the mean-collusion attack (Fig. (2) (a)-(d)), our scheme maintains a recovery rate of 1, even with 256 colluders, which is much higher than that of the two baseline methods.

For the probability-collusion attack (Fig. (2) (e)–(h)), our scheme's recovery rate on the Chess and Carbon datasets is higher than that of the two-stage-LSBs when the number of colluders is below 128. However, when the number of colluders reaches 256, our scheme's fingerprint recovery rate starts to decline and falls below that of the two-stage-LSBs. Nevertheless, such a large number of colluders is impractical in real-world scenarios. In typical cases, both our scheme and the two-stage-LSBs demonstrate robustness against the probability-collusion attack.

On the Survival and Traffic datasets, our scheme achieves a 100% recovery rate, which is nearly twice that of the two-stage-LSBs.

The robustness of our scheme against mean-collusion attacks is a bit stronger than probability-collusion attacks. The reason is that the mean-collusion robustness decreases linearly as the number of colluders increases, whereas the probability-collusion robustness decreases quadratically. Recall that in Section V, the mean-collusion robustness is $\mathcal{O}\left(1 - \frac{2(n_c-1)\cdot \mathrm{Var}[L]}{n\cdot\theta^2}\right)$ and the probability-collusion robustness is $\mathcal{O}\left(1 - \frac{(n_c^2-n_c)\cdot \mathrm{Var}[L]}{n\cdot\theta^2}\right)$, where n_c is the number of colluders.

As the figures illustrated, SNFP-DP has low robustness against collusion attacks, because collusion attacks alter the Gaussian noise embedded in the fingerprinted copy, resulting in a change in the noise variance. Since SNFP-DP relies on detecting the noise variance for fingerprint detection, the scheme exhibits low robustness against collusion attacks. The two-stage LSBs method first utilizes Laplace noise to sanitize the entries of the sensitive database and then embeds the fingerprint into the LSBs of the sanitized database. Since each copy is sanitized independently, the fingerprinted value differs, and the mean-collusion attack alters the LSBs. Therefore, the two-stage-LSBs method has low robustness against mean-collusion attacks.

5) Hybrid Attack: The baseline in this experiment remains the same as that in Section VI-B2. In this experiment, we set the number of colluders to 8, which is higher than the setting of 3 used in [5]'s evaluation.

We use Laplace noise drawn from Lap(2) to perform a hybrid attack. We define *collusion-attack-PRE* as an attack in which the attacker applies noise to the fingerprint copy before the collusion attack. Conversely, *collusion-attack-POST* refers to an attack where the attacker applies noise to the colluded copy after the collusion attack.

The attack results are shown in Fig. (2) (i)–(x). The results demonstrate that our scheme consistently achieves a 100% recovery rate, whereas the two-stage-LSBs achieve less than 60%, and SNFP-DP fails to recover any fingerprints (0% recovery).

TABLE II
MEAN, MAE AND MSE

Dataset	Attribute	$\frac{\Delta}{\epsilon}$	$Mean_O$	MAE_O	MSE_O	$Mean_S$	MAE_S	MSE_S
Carbon	Longitude	0.0	37.669	**0**	**0**	37.669	0	0
		0.5	37.669	**0.5**	**0.5**	37.668	0.952	2.001
		1.0	37.668	**1.0**	**2.0**	37.668	1.305	3.5
		1.5	37.668	**1.5**	**4.5**	37.669	1.724	5.998
		2.0	37.669	**2.0**	**8.0**	37.670	2.175	9.498
		2.5	37.669	**2.5**	12.5	37.670	2.642	14.002
Carbon	Latitude	0.0	-97.784	**0**	**0**	-97.784	0	0
		0.5	-97.784	**0.5**	**0.5**	-97.783	0.951	2.001
		1.0	-97.784	**1.0**	**2.0**	-97.783	1.304	3.5
		1.5	-97.785	**1.5**	**4.5**	-97.783	1.722	6.0
		2.0	-97.785	**2.0**	**8.0**	-97.783	2.175	9.498
		2.5	-97.783	**2.5**	**12.5**	-97.784	2.645	14.002
Chess	White RatingDiff	0.0	0.580	**0**	**0**	0.580	0	0
		0.5	0.580	**0.5**	**0.5**	0.581	0.951	2.001
		1.0	0.580	**1.0**	**2.0**	0.580	1.306	3.502
		1.5	0.579	**1.5**	**4.5**	0.580	1.724	5.994
		2.0	0.581	**2.0**	**8.0**	0.583	2.176	9.490
		2.5	0.582	**2.5**	**12.5**	0.577	2.645	14.002
Chess	Black RatingDiff	0.0	-0.367	**0**	**0**	-0.367	0	**0**
		0.5	-0.367	**0.5**	**0.5**	-0.367	0.951	2.0
		1.0	-0.368	**1.0**	**2.0**	-0.368	1.305	3.5
		1.5	-0.367	**1.5**	**4.5**	-0.368	1.724	6.001
		2.0	-0.368	**2.0**	**8.0**	-0.365	2.176	9.496
		2.5	-0.369	**2.5**	**12.5**	-0.367	2.643	13.998
Traffic	Light	0.0	16.165	**0**	**0**	16.165	0	0
		0.5	16.165	**0.5**	**0.5**	16.165	0.952	2.0
		1.0	16.165	**1.0**	**2.0**	16.165	1.305	3.501
		1.5	16.165	**1.5**	**4.5**	16.164	1.724	5.999
		2.0	16.165	**2.0**	**8.0**	16.164	2.175	9.502
		2.5	16.166	**2.5**	**12.5**	16.166	2.644	13.998
Traffic	Raining	0.0	0.49	**0**	**0**	0.49	0	0
		0.5	0.49	**0.5**	**0.5**	0.49	0.951	2.0
		1.0	0.49	**1.0**	**2.0**	0.49	1.304	3.497
		1.5	0.49	**1.5**	**4.5**	0.49	1.722	6.002
		2.0	0.49	**2.0**	**8.0**	0.49	2.175	9.505
		2.5	0.49	**2.5**	**12.5**	0.49	2.645	13.995
Survive	Age start	0.0	5.23	**0**	**0**	5.23	0	0
		0.5	5.23	**0.5**	**0.5**	5.23	0.952	1.997
		1.0	5.23	**1.0**	**2.0**	5.23	1.305	3.499
		1.5	5.23	**1.5**	**4.5**	5.23	1.724	5.999
		2.0	5.23	**2.0**	**8.0**	5.23	2.175	9.502
		2.5	5.23	**2.5**	**12.5**	5.23	2.643	14.002
Survive	Age end	0.0	44.44	**0**	**0**	44.44	0	0
		0.5	44.44	**0.5**	**0.5**	44.44	0.951	2.0
		1.0	44.44	**1.0**	**2.0**	44.44	1.304	3.500
		1.5	44.44	**1.5**	4.5	44.44	1.722	5.999
		2.0	44.44	**2.0**	**8.0**	44.44	2.175	9.503
		2.5	44.44	**2.5**	**12.5**	44.44	2.645	13.995

The robustness of our scheme against hybrid attacks can be attributed to its resilience against both noise and collusion attacks. Since SNFP-DP and two-stage-LSBs exhibit low robustness against noise and collusion attacks, they also show low robustness against hybrid attacks.

C. Data Utility Evaluation

1) Parameters Setup: In the data utility experiment, we fixed the sensitivity $\Delta = 1$ and varied the privacy budget ϵ from the set $\{0.4, 0.5, \frac{2}{3}, 1, 2\}$. The robustness parameter θ was set to $2 \cdot \frac{\Delta}{\epsilon}$, consistent with the robustness experiment. Our goal was to compare data utility under the same privacy guarantee, so we substituted the approximate differentially private Gaussian mechanism in [6] with the Laplace mechanism, as in our approach. All other parameters of the baselines are the same as ours.

2) Mean, Mean Absolute Error and Mean Square Error: This work focuses on the most commonly used function mean analysis for data utility. Error measurement is applied to evaluate the utility of the fingerprinted dataset. The error is defined as Mean Absolute Error (MAE) [6] and Mean Square Error (MSE) [35]. Table II presents our experimental results, where $Mean_O$, MAE_O and MSE_O denote the mean, MAE and MSE of the fingerprinted copy processed by our method, and $Mean_S$, MAE_S, MSE_S denote the mean, MAE and MSE of the fingerprinted copy processed by SNFP-DP. In the table, we calculate the MSE, MAE and the mean of the fingerprinted database under five different $\frac{\Delta}{\epsilon}$ values. The results show that the means of the original and fingerprinted datasets are essentially unchanged. This is because the injected noise is unbiased and there is no clipping bias from sensitivity clipping (Section IV-F). Consequently, both our method and SNFP-DP produce unbiased fingerprinted copies.

3) Baselines: We compare our method to the baseline method, SNFP-DP, which uses the Laplace mechanism to achieve the same privacy guarantee as our method. The results indicate that under the same privacy budget, their method yields higher MAE and MSE than ours. This is because SNFP-DP requires adding noise from four Laplace distributions with different variances to form the fingerprint, necessitating the addition of more noise than our method due to the parallel composition theorem [6]. Although MAE and MSE increase as Δ/ϵ grows, the data owner can tune the trade-off mechanism to balance privacy and utility. Given a fixed sensitivity Δ, the smaller the privacy budget ϵ, the higher the privacy guarantee, and the lower the data utility.

The SNFP-DP scheme and our scheme inject unbiased noise into the original data, making the fingerprinted data an unbiased estimator of the original. Therefore, both schemes are capable of unbiased mean analysis. In contrast, the CRDPFP scheme converts continuous values into intervals, causing the fingerprinted values to fail as unbiased estimators of the original values. Thus, the CRDPFP scheme is invalid for unbiased mean analysis.

4) Machine Learning Models: We also evaluate data utility for machine-learning tasks. Specifically, we train decision tree classification, decision tree regression, gradient boosted decision tree (GBDT) classification, GBDT regression, and SVM classification on fingerprinted datasets and assess their accuracy. We choose SVM, decision trees, and GBDT because they are commonly used for tabular data [4], [36], [37]. Entry-level differentially private fingerprinting can be used in the context of vertical federated learning. In a vertical federated learning model, each party holds data with the same primary key but different attributes [15], [38], [39]. For example, two parties may hold databases consisting of different attributes of a single relational database: the first party holds the features, while the second party holds the labels. They aim to jointly train a machine learning model but, due to privacy concerns, do not share their data with each other. Entry-level differentially private fingerprinting is well-suited for this scenario because it allows the data owner to release the database without revealing sensitive information. Consequently, if the second party wants to train a machine learning model, the first party can use the entry-level differentially private fingerprinting scheme to send their data to the second party for model training. Therefore, we consider the machine learning utility in the context of vertical federated learning, where the data owner sends the sanitized features to the receiver, and the receiver utilizes the sanitized features and the labels they hold to train a machine learning model.

For this experiment, we separately synthesize the binary and continuous label for datasets mentioned in Section VI-A. We first synthesize binary labels for classification tasks and

continuous labels for regression tasks. We assume that the receiver holds the labels. Therefore, when embedding a fingerprint, the label part remains intact; we only fingerprint the remaining part. We extract 10,000 data records from the synthetic dataset. 80% of the data records are used for training, and the remaining 20% are used for testing. **Baslines.** We compare the utility of our scheme with CRDPFP [5] and SNFP-DP [6]. To ensure the SNFP-DP scheme provides the same privacy guarantee as CRDPFP and our scheme, we replace its originally used Gaussian mechanism [40] with Laplace mechanism.

The experimental results are summarized in Fig. 4. Panels (a–d) present the prediction accuracy of decision tree classification; (e–h) show the prediction accuracy of SVM classification; (i–l) display the prediction accuracy of GBDT classification; (m–p) present the mean squared error (MSE) of GBDT regression; and (q–t) show the MSE of tree regression. For all the tests, we increase the privacy budget ϵ from 0.4 to 2, and "Original" denotes the original dataset. Since CRDPFP can only process discrete values, it needs to convert data into discrete intervals and perturb their least significant bits, as a result, it can only reach a shallow depth for decision trees thus leading to poor performance in both decision tree regression and decision tree classification tasks. CRDPFP also performs worse for SVM, because when transferring the data, the diversity is lost, preventing a precise hyperplane from being learned. To get the same privacy guarantee, SNFP-DP adds significantly more noise than our method, resulting in worse performance than ours.

D. Performance Evaluation

We test the execution time for fingerprint generation and detection in both the basic and optimized versions of our scheme and the results are shown in Fig. 5. The parameter setup is identical to that used in the robustness experiment.

To ensure a fair comparison, we perform fingerprint insertion and detection on the same device and use the CSV-formatted datasets, consistent with the open-science ecosystem (e.g., Kaggle typically stores tabular data in CSV). To remove network-latency bias and ensure a fair comparison, we evaluate on a single device (single server) with local data, although real deployments often involve downloading data over the Internet.

We randomly select $\{100000, 500000, 1000000, 2000000, 4000000, 8000000\}$ entries from the Carbon dataset to create six databases of varying sizes, allowing us to evaluate the generation and detection time of the proposed scheme. We compare these results with those of [4] and [6]. And our competitors SNFP-DP [6] and CRDPFP [5]. The generation and detection in our optimised scheme are slower than those in our basic scheme due to the PRG, which is a trade-off for reducing the storage overhead. Note that in our experiment, the basic version stores all the fingerprints in the memory. In practice, they might need to be stored on the disk and it takes a long time to read/write data from/into the disk, which is true for all PPFS schemes that require storing the fingerprints. Whereas, our optimised version does not require disk access, which could outperform the basic version.

Our optimized scheme outperforms [4] and [6] in terms of fingerprint generation. This is because we do not need to perform complex seed permutations to generate the fingerprint string, as required in [6], nor do we need to transform real-valued data into intervals, as required in [4]. Specifically, when generating a fingerprint for the 8,000,000-entry database, the time consumed by [6] is 177% of ours, and the time consumed by [4] is 141% of ours. The detection time of our optimized scheme remains nearly identical to [4] and [6], indicating no performance loss. The experimental results consistently demonstrate that the time complexity is linear with respect to the size of the database, which aligns with our analysis presented in Section IV-E.

VII. Conclusion

In this paper, we present a novel differentially private fingerprinting scheme (DPFS) for relational databases. The proposed DPFS is well suited for protecting the privacy and copyright of shared real-valued relational databases. The novelty of our approach lies in achieving collusion robustness **without relying on collusion-resistant codes** [5], [8], [25], which serve as the foundation of collusion robustness in current entry-level schemes [5]. Instead, our scheme achieves robustness through its own design. The proposed scheme supports unbiased mean analysis and is robust against various attacks, including noise attacks, collusion attacks, and hybrid attacks. This addresses the existing gap in both utility for real-valued databases and robustness within differentially private database fingerprinting schemes. Furthermore, we provide an approach to optimize robustness against these attacks. While hybrid attacks have not been extensively studied in collusion-resistant code-based schemes [5], [8], [25], a practical adversary could still launch such an attack to eliminate fingerprints, making it a real-world threat model (refert to Section II-B). This consideration makes our threat model more practical compared to previous works [5], [8], [25]. We evaluate the utility and robustness of our scheme, and the experimental results demonstrate our scheme has stronger robustness and better utility than existing state-of-the-art differentially private database fingerprinting schemes.

Future work: In this study, we considered the **mean collusion attack** and **probability collusion attack**, as these attacks do not alter the unbiasedness of the fingerprinted copy. In future work, it would be interesting to extend our analysis to additional collusion attacks to assess their impact on fingerprinting robustness. Additionally, we employed the **Laplace mechanism** for fingerprinting, as it satisfies the ϵ-entry-level differential privacy (DP) guarantee. However, the Laplace noise in our scheme can be replaced with **symmetric Gaussian noise**, which could be explored in future research. Furthermore, investigating methods for generating fingerprints using **non-symmetric, unbiased DP noise** presents an interesting direction for future studies.

Appendix A
Calculation of Noise Robustness

$$Pr(Mean(G(1,y)) \leq 0) = Pr\left(\frac{\sum_{i=1}^{n} x_i + y_i}{n} \leq 0\right)$$

$$
\begin{aligned}
&\overset{(1)}{=} Pr\left(\frac{\sum_{i=1}^{n} y_i}{n} \leq -\frac{\sum_{i=1}^{n} x_i}{n} < -\theta < 0\right)\\
&= Pr\left(\frac{\sum_{i=1}^{n} y_i}{n} - 0 \leq -\frac{\sum_{i=1}^{n} x_i}{n} < -\theta\right)\\
&\leq Pr\left(|\frac{\sum_{i=1}^{n} y_i}{n} - 0| > \frac{\sum_{i=1}^{n} x_i}{n} > \theta\right)\\
&\overset{(2)}{=} Pr\left(|\frac{\sum_{i=1}^{n} y_i}{n} - E\left[\frac{\sum_{i=1}^{n} y_i}{n}\right]| > \frac{\sum_{i=1}^{n} x_i}{n} > \theta\right)\\
&\overset{(3)}{\leq} \frac{Var\left[\frac{\sum_{i=1}^{n} y_i}{n}\right]}{\left(\frac{\sum_{i=1}^{n} x_i}{n}\right)^2} \leq \frac{Var\left[\frac{\sum_{i=1}^{n} y_i}{n}\right]}{\theta^2} = \frac{Var[\mathcal{F}]}{n \cdot \theta^2}
\end{aligned} \tag{7}
$$

We only calculate the probability of the detectable group $G(1, y)$ satisfy its detection condition $D_{1,y} = Mean(G(1, y)) > 0$, since the detection performed on detectable group $G(1, y)$ and detectable group $G(0, y)$ are dual to ($y \in [1, m]$), the calculated result is also the probability of the detectable group $G(0, y)$ satisfy its detection condition $D_{0,y} = Mean(G(0, y)) < 0$, where $Mean(\cdot)$ is the mean function.

Let the unbiased noise distribution $\mathcal{F}$ be the distribution that the attacker used to generate noise. Let $G(1, y)$ be a detectable group. Since the noise data in $G(1, y)$ are all generated from the noise distribution No_1, therefore, each noise in $G(1, y)$ ranges from $[\theta, +\infty)$, suppose there are n noise data in $G(1, y)$, let $G(1, y) = \{x_1, \ldots, x_n\}$, where $x_i (i \in [1, n])$ denote the noise data ranges to $[\theta, +\infty)$. Let $G(1, y) = \{x_1 + y_1, \ldots, x_n + y_n\}$ denote the group being compromised by the noise attack, where $y_i (i \in [1, n])$ is the i.i.d noise follows the noise distribution $\mathcal{F}$. Detectable group $G(1, y)$ fails to satisfy its detection condition only if $Mean(G(1, y)) \leq 0$. Based on the above setting, we can calculate the probability of the detectable group $G(1, y)$ fails to satisfy its detection condition as follows.

In the above calculation, $\overset{(1)}{=}$ is because the noise distribution No_1 ranges to $[\theta, +\infty)$, $\overset{(2)}{=}$ is because $\mathcal{F}$ is unbiased, $\overset{(3)}{=}$ can be obtained by the Chebyshev's inequality.

The calculation gives a probability bound of each detectable group fails to satisfy its detection condition, for each detectable group $G(x, y)(x \in \{0, 1\}, y \in [1, m])$, the probability of the fails to satisfy its detection condition is less than $\frac{Var[\mathcal{F}]}{n \cdot \theta^2}$, which indicate for each detectable group $G(x, y)$ the probability to satisfy its detection condition is greater than $\left(1 - \frac{Var[\mathcal{F}]}{n \cdot \theta^2}\right)$. The calculation is completed.

Appendix B
Calculation of Mean-Collusion Robustness

$$
\begin{aligned}
&Pr(Mean(\widehat{G(1, y)}) \leq 0)\\
&= Pr\left(\frac{\sum_{i=1}^{n} c_i + L_i^*}{n} \leq 0\right)\\
&\overset{(1)}{=} Pr\left(\frac{\sum_{i=1}^{n} c_i}{n} \leq -\frac{\sum_{i=1}^{n} L_i^*}{n} < -\frac{\theta}{n_c} < 0\right)\\
&\overset{(2)}{=} Pr\left(\frac{\sum_{i=1}^{n} c_i}{n} - 0 \leq -\frac{\sum_{i=1}^{n} L_i^*}{n} < -\frac{\theta}{n_c} < 0\right)\\
&\leq Pr\left(|\frac{\sum_{i=1}^{n} c_i}{n} - 0| > \frac{\sum_{i=1}^{n} L_i^*}{n} > \frac{\theta}{n_c}\right)\\
&= Pr\left(|\frac{\sum_{i=1}^{n} c_i}{n} - E\left[\frac{\sum_{i=1}^{n} c_i}{n}\right]| > \frac{\sum_{i=1}^{n_1} L_i^*}{n} > \frac{\theta}{n_c}\right)\\
&\overset{(3)}{\leq} \frac{Var\left[\frac{\sum_{i=1}^{n} y_i}{n}\right]}{\left(\frac{\sum_{i=1}^{n} L_i^*}{n}\right)^2} \leq \frac{Var\left[\frac{\sum_{i=1}^{n} c_i}{n}\right]}{\left(\frac{\theta}{n_c}\right)^2} = \frac{Var[\mathcal{C}]}{n \cdot \left(\frac{\theta}{n_c}\right)^2}\\
&\overset{(4)}{=} \frac{(n_c - 1) \cdot Var[L]}{n \cdot \theta^2}
\end{aligned} \tag{8}
$$

The collusion robustness calculation is similar to the noise robustness calculation performed in Appendix A, we only calculate the probability of the detectable group $G(1, y)$ satisfy its detection condition $D_{1,y} = Mean(G(1, y)) > 0$, since the detection performed on detectable group $G(1, y)$ and detectable group $G(0, y)$ are dual ($y \in [1, m]$), the calculated result is also the probability of the detectable group $G(0, y)$ satisfy its detection condition $D_{0,y} = Mean(G(0, y)) < 0$, where $Mean(\cdot)$ is the mean function.

Let $\widehat{\mathbf{R}}$ be the copy colluded by $\{\widetilde{\mathbf{R}^1}, \ldots, \widetilde{\mathbf{R}^{n_c}}\}$, suppose the fingerprint $\mathcal{K}_r$ is used to detect the fingerprint from $\widehat{\mathbf{R}}$, where $\mathcal{K}_r$ is the fingerprint of the copy $\widetilde{\mathbf{R}^r}$, which is used to perform collusion attack ($\widetilde{\mathbf{R}^r} \in \{\widetilde{\mathbf{R}^1}, \ldots, \widetilde{\mathbf{R}^{n_c}}\}$). Suppose there are n noise data in $\widehat{G(1, y)}$, let $\widehat{G(1, y)} = \{z_1, \ldots, z_n\}$, according to the collusion procedure we can get $z_i = \frac{\sum_{j=1}^{n_c} l_{i,j}}{n}$, where $l_{i,j}$ is the noise that follows the unbiased distribution $Lap\left(\frac{\Delta}{\epsilon}\right)$, for each extracted noise $z_i \in G(1, y)$ there is an $l_{i,j}$ generated by the noise distribution No_1 (eq (1)), $l_{i,j} > \theta$, we rewrite $z_i = \frac{\sum_{j=1}^{n_c} l_{i,j}}{n_c}$ as $z_i = \frac{\sum_{j=1}^{n_c - 1} l_{i,j}}{n_c} + \frac{L_i}{n_c}$, where L_i is the noise greater than θ, let the distribution of the random variable $\frac{\sum_{j=1}^{n_c - 1} l_{i,j}}{n_c}$ be $\mathcal{C}$ since each $l_{i,j}$ follows the unbiased distribution $Lap\left(\frac{\Delta}{\epsilon}\right)$, the distribution $\mathcal{C}$ is also unbiased, and the variance of $\mathcal{C}$ is $\frac{2(n_c - 1) \cdot \left(\frac{\Delta}{\epsilon}\right)^2}{(n_c)^2}$. At last, we rewrite z_i as $z_i = c_i + L_i^*$, where $c_i \sim \mathcal{C}$, L_i^* is the noise larger than $\frac{\theta}{n_c}$. We now calculate the probability of the detectable group $\widehat{G(1, y)} = \{z_1, \ldots, z_n\}$ fails to satisfy its detection condition. $G(1, y)$ fails to satisfy its detection condition only if $Mean(G(1, y)) \leq 0$, where the $Mean(\cdot)$ is the mean function.

The calculation is shown above. Among the calculation $\overset{(1)}{=}$ is because, the noise $L_i^* (1 \geq i \geq n_1)$ are greater than $\frac{\theta}{n_c}$. $\overset{(2)}{=}$ is because $c_i (1 \geq i \geq n)$ is the i.i.d noise follows the unbiased noise distribution $\mathcal{C}$, $\overset{(3)}{=}$ is because the Chebyshev's inequality. $\overset{(4)}{=}$ is because $Var[\mathcal{C}] = \frac{(n_c - 1) \cdot Var[L]}{(n_c)^2}$.

The calculation gives a probability bound for each detectable group $G(x, y)(x \in 0, 1, y \in [1, m])$ fails to satisfy its detection condition. For each detectable group $G(x, y)(x \in \{0, 1\}, y \in [1, m])$, the probability of failing to satisfy its detection is less than $\frac{Var[\mathcal{C}]}{n \cdot \left(\frac{\theta}{n_c}\right)^2}$, by replacing $Var[\mathcal{C}]$ with $\frac{2(n_c - 1) \cdot \left(\frac{\Delta}{\epsilon}\right)^2}{(n_c)^2}$, we can have a more detailed bound $\frac{2(n_c - 1) \cdot \left(\frac{\Delta}{\epsilon}\right)^2}{n \cdot \theta^2}$, thus for each detectable group, the probability of satisfing its detection condition is greater than $(1 - \frac{2(n_c - 1) \cdot \left(\frac{\Delta}{\epsilon}\right)^2}{n \cdot \theta^2})$, the calculation is complete.

Appendix C
Calculation of Probability-Collusion Robustness

The probability-collusion robustness calculation is similar to the calculation of both the noise robustness

(Appendix A) and the mean-collusion robustness (Appendix B), we only calculate the probability of the detectable group $G(1,y)$ satisfy its detection condition $D_{1,y} = Mean(G(1,y)) > 0$, since the detection performed on detectable group $G(1,y)$ and detectable group $G(0,y)$ are dual to $(y \in [1,m])$, the calculated result is also the probability of the detectable group $G(0,y)$ satisfy its detection condition $D_{0,y} = Mean(G(0,y)) < 0$, where $Mean(\cdot)$ is the mean function.

Let $\widehat{\mathbf{R}}$ be the copy colluded by $\{\widetilde{\mathbf{R}^1},\ldots,\widetilde{\mathbf{R}^{n_c}}\}$, suppose the fingerprint $\mathcal{K}_r$ is used to detect the fingerprint from $\widehat{\mathbf{R}}$, where $\mathcal{K}_r$ is the fingerprint of the copy $\widetilde{\mathbf{R}^r}$, which is used to perform collusion attack $(\widetilde{\mathbf{R}^r} \in \{\widetilde{\mathbf{R}^1},\ldots,\widetilde{\mathbf{R}^{n_c}}\})$. Suppose there are n noise data in $G(1,y)$, let $\widehat{G(1,y)} = \{x_1,\ldots,x_{n_1},y_1,\ldots,y_{n-n_1}\}$, where $x_i(1 \le i \le n_1)$ is the real number larger than θ, $y_i(1 \le i \le n-n_1)$ is the i.i.d noise follows unbiased Laplace distribution L. We next calculate the probability of the group $\widehat{G(1,y)}$ fails to satisfy it's detection condition $De(1,y)$.

$$
\begin{aligned}
&Pr(Mean(\widehat{G(1,y)}) \le 0)\\
&= Pr\left(\frac{\sum_{i=1}^{n-n_1} x_i + \sum_{i=1}^{n_1} y_i}{n} \le 0\right)\\
&\overset{(1)}{=} Pr\left(\frac{\sum_{i=1}^{n-n_1} y_i}{n} \le -\frac{\sum_{i=1}^{n_1} x_i}{n} < -\frac{n_1\theta}{n} < 0\right)\\
&= Pr\left(\frac{\sum_{i=1}^{n-n_1} y_i}{n} - 0 \le -\frac{\sum_{i=1}^{n_1} x_i}{n} < -\frac{n_1\theta}{n}\right)\\
&\le Pr\left(|\frac{\sum_{i=1}^{n-n_1} y_i}{n} - 0| > \frac{\sum_{i=1}^{n_1} x_i}{n} > \frac{n_1\theta}{n}\right)\\
&\overset{(2)}{=} Pr\left(|\frac{\sum_{i=1}^{n-n_1} y_i}{n} - E\left[\frac{\sum_{i=1}^{n-n_1} y_i}{n}\right]| > \frac{\sum_{i=1}^{n_1} x_i}{n} > \frac{n_1\theta}{n}\right)\\
&\overset{(3)}{\le} \frac{Var\left[\frac{\sum_{i=1}^{n-n_1} y_i}{n}\right]}{\left(\frac{\sum_{i=1}^{n_1} x_i}{n}\right)^2} \le \frac{Var\left[\frac{\sum_{i=1}^{n-n_1} y_i}{n}\right]}{\left(\frac{n_1\theta}{n}\right)^2} \overset{(4)}{=} \frac{(n-n_1)Var[L]}{(n_1\cdot\theta)^2}\\
&\overset{(5)}{\approx} \frac{(n_c^2-n_c)Var[L]}{n\cdot\theta^2}
\end{aligned}
\tag{9}
$$

The calculation is shown above. Among the calculation $\overset{(1)}{=}$ is because, the noise $x_i(1 \ge i \ge n_1)$ are greater than θ. $\overset{(2)}{=}$ is because $y_i(1 \le i \le n-n_1)$ is the i.i.d noise follows the unbiased noise distribution $Lap\left(\frac{\Delta}{\epsilon}\right)$, $\overset{(3)}{\le}$ is because the Chebyshev's inequality. $\overset{(4)}{=}$ is because y_i follows L. $\overset{(5)}{\approx}$ is because according to the procedure of probability-collusion attack, about $\frac{1}{n_c}$ entries of the probability-collusion database $\widehat{\mathbf{R}}$ are come from $\widetilde{\mathbf{R}^r}$, the database with fingerprint $\mathcal{K}_r$ on it. Therefore, about $\frac{1}{n_c}$ noise in $\widehat{G(1,y)} = \{x_1,\ldots,x_{n_1},y_1,\ldots,y_{n-n_1}\}$ are come from $\widetilde{\mathbf{R}^r}$, that is $n_1 \approx \frac{n}{n_c}$.

The calculation gives a probability bound for each detectable group $G(x,y)(x \in 0,1,y \in [1,m])$ fails to satisfy its detection condition. For each detectable group $G(x,y)(x \in \{0,1\},y \in [1,m])$, the probability of failing to satisfy its detection is less than $\frac{(n_c^2-n_c)Var[L]}{n\cdot\theta^2}$, thus for each detectable group, the probability of satisfing its detection condition is greater than $\left(1-\frac{(n_c^2-n_c)Var[L]}{n\cdot\theta^2}\right)$, the calculation is completed.

Appendix D
Calculation of Mean-Collusion False Hit

The notations used in Appendix B are still used in this calculation. Suppose K_I (the fingerprint of the innocent) is used to detect fingerprints from the collusion database $\widehat{\mathbf{R}}$, each noise z_i, which is extracted from $\widehat{\mathbf{R}}$ can be represent as $\frac{\sum_{j=1}^{n_c} l_{i,j}}{n_c}$, where $l_{i,j}$ is the noise that follows the unbiased distribution $Lap\left(\frac{\Delta}{\epsilon}\right)$ and the p.d.f. of $Lap\left(\frac{\Delta}{\epsilon}\right)$ is an even function, therefore, each $l_{i,j}$ has an equal probability $\frac{1}{2}$ of being greater than or less than 0, thus the extracted noise has an equal probability $\frac{1}{2}$ of being greater than or less than 0, recall that each detectable group $G(x,y)$ contains the extracted noise, therefore the probability of $Mean(G(x,y)) > 0$ or $Mean(G(x,y)) < 0$ equals $\frac{1}{2}$, where $Mean(\cdot)$ is the mean function, i.e., for each detectable group $G(x,y)$, the probability to satisfy its detection condition $D_{x,y}$ equals $\frac{1}{2}$. Since detection succeeds only if all detectable groups $G(x,y)(x \in \{0,1\},y \in [1,m])$ satisfy its detection condition, we can have $Pr(collusion-false-hit) = \frac{1}{2^{2m}}$. The calculation is completed.

Appendix E
Calculation of Probability-Collusion False Hit

The notations used in Appendix B are still used in this calculation. Suppose K_I (the fingerprint of the innocent) is used to detect fingerprints from the collusion database $\widehat{\mathbf{R}}$, each noise z_i, which extracted from $\widehat{\mathbf{R}}$. According to the probability-collusion procedure, the noise z_i follows the unbiased distribution $Lap\left(\frac{\Delta}{\epsilon}\right)$ and the p.d.f. of $Lap\left(\frac{\Delta}{\epsilon}\right)$ is an even function, therefore, each z_i has an equal probability $\frac{1}{2}$ of being greater than or less than 0, thus the extracted noise has an equal probability $\frac{1}{2}$ of being greater than or less than 0, recall that each detectable group $G(x,y)$ contains the extracted noise, therefore the probability of $Mean(G(x,y)) > 0$ or $Mean(G(x,y)) < 0$ equals $\frac{1}{2}$, where $Mean(\cdot)$ is the mean function, i.e., for each detectable group $G(x,y)$, the probability to satisfy its detection condition $D_{x,y}$ equals $\frac{1}{2}$. Since detection succeeds only if all detectable groups $G(x,y)(x \in \{0,1\},y \in [1,m])$ satisfy its detection condition, we can have $Pr(collusion-false-hit) = \frac{1}{2^{2m}}$. The calculation is completed.

Appendix F
Proof of Composition Theorem 1

The composition theorem is central to differential-privacy–based definitions: it quantifies how privacy guarantees degrade when an adversary observes multiple DP outputs from the same dataset, i.e., it bounds the cumulative privacy loss under repeated releases. We prove the composition theorem to show how privacy loss degrades when using entry-level fingerprinting to release multiple fingerprinted copies; since the randomness in our release is independent each time, we treat each mechanism's output as independent.

Proof: We adopt the setting of Theorem 2. Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be entry-level differentially private mechanisms with privacy budgets ϵ_1 and ϵ_2, respectively. Denote their outputs by $\widetilde{\mathbf{R}}_1$ and $\widetilde{\mathbf{R}}_2$. We show that the composite mechanism $\mathcal{M} := (\mathcal{M}_1,\mathcal{M}_2)$ satisfies $(\epsilon_1+\epsilon_2)$–entry-level DP.

$$
\frac{Pr(\mathcal{M}(\mathbf{R})) = (\widetilde{\mathbf{R}}_1,\widetilde{\mathbf{R}}_2))}{Pr(\mathcal{M}(\mathbf{R}')) = (\widetilde{\mathbf{R}}_1,\widetilde{\mathbf{R}}_2))}
$$

$$\stackrel{(1)}{=} \frac{Pr(\mathcal{M}_1(\mathbf{R}) = \widetilde{\mathbf{R}}_1) \cdot Pr(\mathcal{M}_2(\mathbf{R}) = \widetilde{\mathbf{R}}_2)}{Pr(\mathcal{M}_1(\mathbf{R}') = \widetilde{\mathbf{R}}_1) \cdot Pr(\mathcal{M}_2(\mathbf{R}') = \widetilde{\mathbf{R}}_2)}$$
$$\stackrel{(2)}{=} \epsilon_1 + \epsilon_2 \tag{10}$$

In the equation, $\stackrel{(1)}{=}$ expands the probability using the joint mechanism $\mathcal{M} = (\mathcal{M}_1, \mathcal{M}_2)$, and $\stackrel{(2)}{=}$ follows from the definition of ϵ-entry-level DP. This completes the proof.

REFERENCES

[1] E. Bertino, B. C. Ooi, Y. Yang, and R. H. Deng, "Privacy and ownership preserving of outsourced medical data," in *Proc. 21st Int. Conf. Data Eng.*, Tokyo, Japan, Apr. 2005, pp. 521–532.

[2] S. Schrittwieser, P. Kieseberg, I. Echizen, S. Wohlgemuth, N. Sonehara, and E. Weippl, "An algorithm for K-anonymity-based fingerprinting," in *Proc. Int. Workshop Digit. Watermarking*, Jul. 2012, pp. 439–452.

[3] S. Gambs, J. Lolive, and J.-M. Robert, "Entwining sanitization and personalization on databases," in *Proc. Asia Conf. Comput. Commun. Secur.*, May 2018, pp. 207–219.

[4] T. Ji, E. Ayday, E. Yilmaz, M. Li, and P. Li, "Privacy-preserving database fingerprinting," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2023, pp. 10–14722.

[5] S. Zhang, Y. Zhu, and A. Zeng, "Collusion-resilient privacy-preserving database fingerprinting," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 8306–8321, 2024.

[6] Y. Hu, A. Hu, C. Li, P. Li, and C. Zhang, "Towards a privacy protection-capable noise fingerprinting for numerically aggregated data," *Comput. Secur.*, vol. 119, Aug. 2022, Art. no. 102755.

[7] Q. Ye, H. Hu, X. Meng, and H. Zheng, "PrivKV: Key-value data collection with local differential privacy," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 317–331.

[8] Y. Li, V. Swarup, and S. Jajodia, "Fingerprinting relational databases: Schemes and specialties," *IEEE Trans. Dependable Secure Comput.*, vol. 2, no. 1, pp. 34–45, Jan. 2005.

[9] F. Sebé, J. Domingo-Ferrer, and A. Solanas, "Noise-robust watermarking for numerical datasets," in *Proc. Int. Conf. Model. Decisions Artif. Intell.*Cham, Switzerland: Springer, 2005, pp. 134–143.

[10] J. Franco-Contreras and G. Coatrieux, "Robust watermarking of relational databases with ontology-guided distortion control," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 9, pp. 1939–1952, Sep. 2015.

[11] A. Blanco-Justicia, D. Sánchez, J. Domingo-Ferrer, and K. Muralidhar, "A critical review on the use (and misuse) of differential privacy in machine learning," *ACM Comput. Surv.*, vol. 55, no. 8, pp. 1–16, Aug. 2023.

[12] J. Dong, A. Roth, and W. J. Su, "Gaussian differential privacy," 2019, *arXiv:1905.02383*.

[13] Q. Geng, P. Kairouz, S. Oh, and P. Viswanath, "The staircase mechanism in differential privacy," *IEEE J. Sel. Topics Signal Process.*, vol. 9, no. 7, pp. 1176–1184, Oct. 2015.

[14] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, nos. 3–4, pp. 211–407, 2014.

[15] Y. Liu et al., "Vertical federated learning: Concepts, advances, and challenges," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 7, pp. 3615–3634, Jul. 2024.

[16] P. Kieseberg, S. Schrittwieser, M. Mulazzani, I. Echizen, and E. Weippl, "An algorithm for collusion-resistant anonymization and fingerprinting of sensitive microdata," *Electron. Markets*, vol. 24, no. 2, pp. 113–124, Jun. 2014.

[17] V. Rastogi, D. Suciu, and S. Hong, "The boundary between privacy and utility in data publishing," in *Proc. 33rd Int. Conf. Very Large Data Bases*, 2007, pp. 531–542.

[18] R. Agrawal and J. Kiernan, "Watermarking relational databases," in *Proc. 28th Int. Conf. Very Large Databases*. Amsterdam, The Netherlands: Elsevier, 2002, pp. 155–166.

[19] M.-R. Xie, C. Wu, J.-J. Shen, and M. Hwang, "A survey of data distortion watermarking relational databases," *Int. J. Netw. Secur.*, vol. 18, no. 6, pp. 1022–1033, 2016.

[20] S. Voloshynovskiy, S. Pereira, T. Pun, J. J. Eggers, and J. K. Su, "Attacks on digital watermarks: Classification, estimation based attacks, and benchmarks," *IEEE Commun. Mag.*, vol. 39, no. 8, pp. 118–126, Aug. 2001.

[21] J. Kiernan, R. Agrawal, and P. J. Haas, "Watermarking relational data: Framework, algorithms and analysis," *VLDB J. Int. J. Very Large Data Bases*, vol. 12, no. 2, pp. 157–169, Aug. 2003.

[22] D. Megías and A. Qureshi, "Collusion-resistant and privacy-preserving P2P multimedia distribution based on recombined fingerprinting," *Expert Syst. Appl.*, vol. 71, pp. 147–172, Apr. 2017.

[23] E. Yilmaz and E. Ayday, "Collusion-resilient probabilistic fingerprinting scheme for correlated data," 2020, *arXiv:2001.09555*.

[24] Y. Jiang, E. Yılmaz, and E. Ayday, "Robust fingerprint of location trajectories under differential privacy," in *Proc. Privacy Enhancing Technol. Privacy Enhancing Technol. Symp.*, 2022, pp. 1–5.

[25] D. Boneh and J. Shaw, "Collusion-secure fingerprinting for digital data," *IEEE Trans. Inf. Theory*, vol. 44, no. 5, pp. 1897–1905, May 1998.

[26] D. Boneh and J. Shaw, "Collusion-secure fingerprinting for digital data," in *Proc. Annu. Int. Cryptol. Conf.* Cham, Switzerland: Springer, 1995, pp. 452–465.

[27] G. Tardos, "Optimal probabilistic fingerprint codes," *J. ACM*, vol. 55, no. 2, pp. 1–24, May 2008.

[28] T. Ji, E. Ayday, E. Yilmaz, and P. Li, "Privacy-preserving database fingerprinting," 2021, *arXiv:2109.02768*.

[29] C.-H. Chang and L. Zhang, "A blind dynamic fingerprinting technique for sequential circuit intellectual property protection," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 1, pp. 76–89, Jan. 2014.

[30] T. Wang, J. Blocki, N. Li, and S. Jha, "Locally differentially private protocols for frequency estimation," in *Proc. 26th USENIX Secur. Symp.*, Aug. 2017, pp. 729–745.

[31] M. Abadi et al., "Deep learning with differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2016, pp. 308–318.

[32] R. Liu and C. Qiu, "PAnDA: Rethinking metric differential privacy optimization at scale with anchor-based approximation," 2025, *arXiv:2509.08720*.

[33] K. Chatzikokolakis, M. E. Andrés, N. E. Bordenabe, and C. Palamidessi, "Broadening the scope of differential privacy using metrics," in *Proc. Int. Symp. Privacy Enhancing Technol. Symp.* Cham, Switzerland: Springer, 2013, pp. 82–102.

[34] M. Alvim, K. Chatzikokolakis, C. Palamidessi, and A. Pazii, "Invited paper: Local differential privacy on metric spaces: Optimizing the trade-off with utility," in *Proc. IEEE 31st Comput. Secur. Found. Symp. (CSF)*, Jul. 2018, pp. 262–267.

[35] K. Zhang et al., "Bounded and unbiased composite differential privacy," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2024, pp. 972–990.

[36] L. Grinsztajn, E. Oyallon, and G. Varoquaux, "Why do tree-based models still outperform deep learning on typical tabular data?," in *Proc. Adv. Neural Inf. Process. Syst.*, Sep. 2022, pp. 507–520.

[37] B. Sun et al., "SuperTML: Two-dimensional word embedding for the precognition on structured tabular data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, Jul. 2019, pp. 2973–2981.

[38] Y. Wu, S. Cai, X. Xiao, G. Chen, and B. C. Ooi, "Privacy preserving vertical federated learning for tree-based models," 2020, *arXiv:2008.06170*.

[39] Y. Zheng, S. Xu, S. Wang, Y. Gao, and Z. Hua, "Privet: A privacy-preserving vertical federated learning service for gradient boosted decision tables," *IEEE Trans. Services Comput.*, vol. 16, no. 5, pp. 3604–3620, Sep. 2023.

[40] S. Meiser and E. Mohammadi, "Tight on budget?: Tight bounds for R-fold approximate differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2018, pp. 247–264.