Privacy-Preserving Database Fingerprinting

Tianxi Ji
Texas Tech University
tiji@ttu.edu

Erman Ayday
Case Western Reserve University
exa208@case.edu

Emre Yilmaz
University of Houston-Downtown
yilmaze@uhd.edu

Ming Li
University of Texas at Arlington
ming.li@uta.edu

Pan Li
Case Western Reserve University
lipan@case.edu

***Abstract*—When sharing relational databases with other parties, in addition to providing high quality (utility) database to the recipients, a database owner also aims to have (i) privacy guarantees for the data entries and (ii) liability guarantees (via fingerprinting) in case of unauthorized redistribution. However, (i) and (ii) are orthogonal objectives, because when sharing a database with multiple recipients, privacy via data sanitization requires adding noise once (and sharing the same noisy version with all recipients), whereas liability via unique fingerprint insertion requires adding different noises to each shared copy to distinguish all recipients. Although achieving (i) and (ii) together is possible in a naïve way (e.g., either differentially-private database perturbation or synthesis followed by fingerprinting), this approach results in significant degradation in the utility of shared databases. In this paper, we achieve privacy and liability guarantees simultaneously by proposing a novel entry-level differentially-private (DP) fingerprinting mechanism for relational databases without causing large utility degradation.**

The proposed mechanism fulfills the privacy and liability requirements by leveraging the randomization nature of fingerprinting and transforming it into provable privacy guarantees. Specifically, we devise a bit-level random response scheme to achieve differential privacy guarantee for arbitrary data entries when sharing the entire database, and then, based on this, we develop an ϵ-entry-level DP fingerprinting mechanism. We theoretically analyze the connections between privacy, fingerprint robustness, and database utility by deriving closed form expressions. We also propose a sparse vector technique-based solution to control the cumulative privacy loss when fingerprinted copies of a database are shared with multiple recipients.

We experimentally show that our mechanism achieves strong fingerprint robustness (e.g., the fingerprint cannot be compromised even if the malicious database recipient modifies/distorts more than half of the entries in its received fingerprinted copy), and higher database utility compared to various baseline methods (e.g., application-dependent database utility of the shared database achieved by the proposed mechanism is higher than that of the considered baselines).

I. Introduction

Massive data collection and availability of relational databases (collection of data records with the same attributes [13]) are very common in the current big data era. This results in an increasing demand to share such databases with (or among) different database recipients/service providers (SPs), such as companies, research institutions, or hospitals, for the purpose of "do-it-yourself" calculations, like personal advertisements, social recommendations, and customized healthcare.

Most databases include personal data, and thus they usually contain sensitive and proprietary information, e.g., medical records collected as part of an agreement which restricts redistribution. This poses three major challenges in database sharing with different SPs: (1) **privacy**, the database owner is obligated to protect the privacy of data entries in the shared database to comply with the privacy policy and ensure confidentiality, (2) **liability**, the database owner needs to prevent illegal redistribution of the shared databases, and eventually prosecute the malicious SPs who leak its data, and (3) **utility**, the shared database needs to maintain high utility to support accurate data mining and analysis.

Many works have attempted to address the challenges on privacy and liability in isolation. To address the privacy challenge, various data sanitization metrics are proposed, e.g., k-anonymity [47], l-diversity [41], t-closeness [31], and differential privacy (DP) [16]. Among them, DP has been developed as a *de facto* standard for responding to statistical queries from databases with provable privacy guarantees. It can also be used to share personal data streams or an entire database (i.e., identity query) in a privacy-preserving manner [11], [24]. Differentially-private mechanisms hide the presence or absence of a data record in the database by perturbing the query results with noise calibrated to the query sensitivity.

To protect copyright and deter illegal redistribution, different database watermarking and fingerprinting mechanisms are devised to prove database ownership (i.e., identifying the database owner from shared databases) [1], [46] and database possession (i.e., differentiating between the SPs who received copies of the database) [35], [22], [26], [23], [50]. In practice, when sharing a database with a specific SP, the database owner embeds a unique fingerprint (a binary string customized for the SP) in the database. The embedded fingerprint is hard to be located and removed even if a malicious SP attacks the fingerprinted database (to identify and distort the fingerprint).

Only a few works have attempted to combine database sanitization and fingerprinting in database sharing. In particular, [45], [4], [29] propose inserting fingerprints into databases sanitized using k-anonymity, and [17] proposed embedding fingerprints into databases sanitized by the (α, β)-privacy model [44]. However, these works solve the aforementioned challenges in a two-stage (sequential) manner, where data sanitization is conducted before fingerprinting. As a result, they end up changing a large amount of entries in the database and they significantly compromise the utility of the shared

Network and Distributed System Security (NDSS) Symposium 2023
27 February - 3 March 2023, San Diego, CA, USA
ISBN 1-891562-83-5
https://dx.doi.org/10.14722/ndss.2023.24693
www.ndss-symposium.org

database (corroborated in Section VII). The only work that attempts to integrate privacy protection and fingerprinting is proposed in [20]. However, [20] injects continuous-valued Gaussian noise to the data, considers various combinations of variances as fingerprints, and relies on learning algorithms to fit the Gaussian noises. Thus, [20] is vulnerable if a malicious SP compromises a large portion of fingerprinted data entries (shown in Section VII). Besides, these works do not address the critical problem of controlling cumulative privacy loss if the same database is repeatedly shared with multiple SPs.

In this paper, we bring together data sanitization and fingerprinting in a unified mechanism, consider a stronger privacy model compared to previous works, and develop entry-level DP fingerprinting for relational database sharing. In what follows, we summarize the main contributions and insights of our work, and discuss its limitation caused by a unique requirement of DBMS (Database Management System) design.

Main Contributions. Database fingerprinting is a randomized scheme (that essentially performs bitwise randomization, i.e., randomly changes **insignificant** bits of randomly selected data entries [1]), and thus is naturally endowed with certain level of privacy. Yet, this hidden property (privacy protection) is ignored in the literature. **We harness the intrinsic randomness introduced by fingerprinting and transform it into a provable privacy guarantee.** In particular,

- We propose a bit-level random response scheme, which fingerprints (marks) insignificant bits of data entries using pseudorandomly generated binary mark bits, to achieve ϵ-entry-level DP for the entire database. Then, based on this scheme, we devise the ϵ-entry-level DP fingerprinting mechanism.
- We establish a comprehensive and solid theoretical foundation to quantify the properties of the proposed ϵ-entry-level DP fingerprinting mechanism from 3 dimensions: (i) the privacy guarantee of it under attribute inference attack, (ii) the fingerprint robustness of the mechanism when it is subject to various attacks targeting on the inserted marks, and (iii) the relationship among privacy, utility, and fingerprint robustness.
- We devise a sparse vector technique (SVT)-based solution to control the cumulative privacy loss when different fingerprinted versions of a database are shared with multiple SPs.
- We evaluate the proposed mechanism using two real-life databases. Experiment results show that our mechanism (i) provides higher fingerprint robustness than a state-of-the-art database fingerprinting mechanism [35], and (ii) achieves higher database utility than the two-step methods (i.e., either local DP-based perturbation, data synthesis under central DP, or k-anonymity followed [35]) and the one-step approach (i.e., Gaussian noise-based fingerprinting [20]) by considering specific applications.

Insights. This paper is the first to show the feasibility of considering privacy and liability in a unified mechanism to simultaneously protect data privacy and prevent unauthorized redistribution. Our mechanism can help a database owners (i) generate privacy-preserving fingerprinted databases based on their requirements on utility, privacy, and fingerprint robustness and (ii) assess the privacy leakage under multiple sharings and set the privacy budget accordingly in each sharing.

Limitations. In this work, we consider the sharing of entire relational databases, where each data record can be uniquely identified by an immutable pseudo-identifier (i.e., the primary key) in order to support common database operations, e.g., union and intersection, which all depend on the value of primary keys. This is a **unique and hard requirement** of DBMS, and thus in this work, we do not consider membership inference attacks as they become irrelevant under these settings. We further discuss this in detail in Section III.

Roadmap. In Section II we review related works followed by the privacy, system, objectives, and threat models in Section III. In Section IV, we present the entry-level DP fingerprinting mechanism. Then, we theoretically investigate the relationships between database utility, fingerprint robustness, and privacy guarantees in Section V. We develop the sparse vector technique (SVT)-based mechanism to share multiple fingerprinted databases under entry-level DP in Section VI. We evaluate the proposed scheme via extensive experiments in Section VII. We provide further discussions and point out open problems with potential solutions in Section VIII. Finally, Section IX concludes the paper.

II. Related Work

Database watermarking/fingerprinting. The seminal work of database watermarking (that embeds the same bit-string in selected insignificant bits to all shared database copies to claim ownership) is proposed in [1]. Based on [1], several database fingerprinting techniques were proposed [46], [35], [38]. In particular, [35] is considered as the state-of-the-art that also best suits for fingerprinting the entire database. This is because [35] enables the insertion and extraction of arbitrary bit-strings in relational databases and it also provides an extensive robustness analysis. In Section VII, we develop three baselines based on [35] (i.e., either local DP perturbation, DP database synthesis, or k-anonymity followed by database fingerprinting via [35]) and compare them with our proposed mechanism.

Data sanitization followed by watermarking/fingerprinting. Some works attempted to protect data privacy and ensure liability in isolation when sharing databases [17], [4], [29], [45]. To be more specific, Bertino et al. [4] adopted the binning method [36] to generalize the database first, then watermark the binned data to protect copyright. Kieseberg et al. [29] and Schrittwieser et al. [45] proposed fingerprinting a database generalized by k-anonymity. Gambs et al. [17] sanitized the database using the (α, β)-privacy model [44], which selects a true data record in the domain of the database with probability α and includes a fake data record outside the domain of the database with probability β, and then they embed personalized fingerprint in the database. These studies usually change a large amount of data entries, which degrades database utility. In particular, it has been observed that k-anonymity may create data records that leak information due to the lack of diversity in some sensitive attributes, and it does not protect against attacks based on background knowledge [18].

All these schemes embed watermark or fingerprint into already sanitized databases, instead of considering sanitization

and marking (fingerprinting) together as a unified process. Such sequential processing of a database will result in significant degradation in utility. This is because both sanitization and fingerprinting are achieved via noise addition (first adding noise to protect privacy, then adding noise to achieve liability guarantee) which over-distorts the database. Our work is different from previous ones, since we unify data sanitization and fingerprinting (to have higher data utility). We achieve provable privacy guarantees during fingerprint insertion by adopting a customized privacy model for DBMS and harnessing the randomness of fingerprinting.

Database sanitization together with fingerprinting. The closest work to ours is a concurrent paper [20], which inserts Gaussian noises with various pre-determined variances to different blocks of a database to protect data privacy. The various combination of noise variances for data blocks plays the role of tractable fingerprints. However, the mechanism in [20] is vulnerable even to the subset attack (Section V-C), because in the fingerprints detection phase, [20] needs to use learning algorithms to fit the inserted Gaussian noise and re-calculate the corresponding variances. Hence, the fingerprint robustness of [20] is very sensitive to the adopted learning algorithms and the size of the database. Besides, [20] adds Gaussian noise to data entries, which will significantly reduce the data utility. In contrast, our mechanism changes each insignificant bit with a certain probability, so some data entries will be intact. In Section VII, we will show also compare with [20].

III. Privacy, System, Objectives, and Threat

Here, we discuss the considered privacy model, database fingerprinting system, objectives of the database owner and malicious SPs, and various threats. The frequently used notations in this paper is listed in Table I. In what follows, we first review the definition of a relational database and its unique features, which are important for our specific definition of the privacy model.

notations	descriptions
$\mathbf{R}$	original database
$\mathbf{R}'$	a neighboring database of $\mathbf{R}$
$\mathcal{M}(\mathbf{R})$	fingerprinted database
$\overline{\mathbf{R}}$	leaked (pirated) database
$\mathbf{r}_i$	ith row of $\mathbf{R}$
$\mathbf{r}_i[t,k]$	the kth insignificant bit of the tth attribute of $\mathbf{r}_i$
p	the probability of changing an insignificant bit in an entry in the database
B	mark bit to fingerprint a bit position $B \sim \text{Bernoulli}(p)$
ϵ,ϵ_2,ϵ_3	privacy budgets
Δ	database sensitivity

TABLE I. Frequently used notations in the paper.

Definition 1 (Relational database [13]): A relational database denoted as $\mathbf{R}$ is a collection of T-tuples. Each tuple represents a data record containing T ordered attributes. Each data record is also associated with a primary key, **which is used to uniquely identify that record**. We denote the ith data record in $\mathbf{R}$ as $\mathbf{r}_i$ and its primary key as $\mathbf{r}_i.PmyKey$.

Unique Features of a Relational Database. In order to support database operations, such as union, intersection, and update, the primary keys should **not** be changed if a database is fingerprinted or pirated [35], [1], [34].[1] Due to the uniqueness and immutability of the primary keys in relational databases, the presence of a specific data record is not a private information in general. In other words, it is no secret whether an individual's data record (a specific T-tuple) is present in a database or not. Hence, the common definition of neighboring databases (which differ by one row) in the differential privacy literature does not apply in the case of sharing relational databases. Thus, we consider an alternative definition of neighboring relational databases and their sensitivity as follows.

Definition 2 (Neighboring relational databases): Two relational databases $\mathbf{R}$ and $\mathbf{R}'$ are called neighboring, if they only differ by **one entry**, i.e., an attribute of a single individual.

Definition 3 (Sensitivity of a relational database): Given a pair of neighboring relational databases $\mathbf{R}$ and $\mathbf{R}'$ that differ by one entry (e.g., the tth attribute of the ith row, $\mathbf{r}_i[t]$ and $\mathbf{r}_i[t]'$), the sensitivity is defined as $\Delta = \sup_{\mathbf{R},\mathbf{R}'} ||\mathbf{R}-\mathbf{R}'||_F = \sup_{\mathbf{r}_i[t],\mathbf{r}_i[t]'} |\mathbf{r}_i[t] - \mathbf{r}_i[t]'|$, where sup and F represent the supreme value and the matrix Frobenius norm, respectively.

A. Privacy Model

Next, we give our privacy model customized specifically for databases with immutable primary keys.

Definition 4 (ϵ-entry-level DP): A randomized mechanism $\mathcal{M}$ with domain $\mathcal{D}$ satisfies ϵ-entry-level differential privacy if for any two neighboring relational databases $\mathbf{R}, \mathbf{R}' \in \mathcal{D}$, and for all $\mathcal{S} \in \text{Range}(\mathcal{M})$, it holds that $\Pr[\mathcal{M}(\mathbf{R}) = S] \leq e^{\epsilon} \Pr[\mathcal{M}(\mathbf{R}') = S]$, where $\epsilon > 0$.

Remark 1: Definition 4 is adapted from the conventional notation of ϵ-DP [16], which obfuscates the presence or absence of an entire row in $\mathbf{R}$. Since the database recipient can easily identify if an individual is present in $\mathbf{R}$ by directly checking its primary key, the conventional ϵ-DP is **not appropriate** in the setting we consider. In contrast, our privacy model, which aims at obscuring the specific value of an arbitrary entry in $\mathbf{R}$, better suits the requirement of DBMS design. As discussed in Section I, destroying pseudo-identifiers to prevent linkability or membership inference attacks becomes an ill-posed problem for our considered case of DBMS. Thus, in this paper, we focus on the attribute inference attacks instead of membership inference attacks. As a matter of fact, in addition to common database applications (e.g., SQL, merging, splitting, union, and intersection), there are quite a few applications requiring consideration of attribute inference attacks over membership inference attacks, such as clustering-based applications, where the goal is to assign individuals to different clusters (determine their membership). An example is the construction of a recommendation system based on the

[1] In DBMS design, the primary keys are required to be immutable, as updating a primary key can lead to the update of potentially many other tables or rows in the system. The reason is that in DBMS, a primary key also serves as a foreign key (a column that creates a relationship between two tables in DBMS). For instance, consider the database in Section VII in which each data record represents a student. Here, the primary key of the data record can be chosen as the student's unique identification number, which can then be used to refer to another table keeping the their real name, email, etc.

attributes of participants, in which the goal is to recommend movies or products to each dataset participant while preserving the privacy of the attributes of the participants [27]. Another example is the community detection in social networks under the setting of edge-DP (where hiding the presence or absence of a specific node is an ill-posed problem) [24], [25].

Similar to the conventional DP, we define (ϵ, δ)-entry-level DP as $\Pr[\mathcal{M}(\mathbf{R}) = S] \leq e^{\epsilon} \Pr[\mathcal{M}(\mathbf{R}') = S] + \delta$, $\delta \in [0, 1]$.

B. System, Objectives, and Potential Threats

We present the system model in Figure 1. We consider a database owner with a relational database denoted as $\mathbf{R}$, who wants to share it with at most C SPs (e.g., to receive specific services). To prevent unauthorized redistribution of the database by a malicious SP (e.g., the jth SP in Figure 1), the database owner includes unique fingerprints in all shared copies of the database. The fingerprint essentially changes different entries in $\mathbf{R}$ at different positions (indicated by the yellow dots). The fingerprint bit-string customized for the jth SP (SP_j) is denoted as f_{SP_j}, and the database received by SP_j is represented as $\widetilde{\mathbf{R}}_j$. Both f_{SP_j} and $\widetilde{\mathbf{R}}_j$ are obtained using the proposed mechanism discussed in Section IV. $\widetilde{\mathbf{R}}$ represents an instance of the privacy-preserving fingerprinted database.

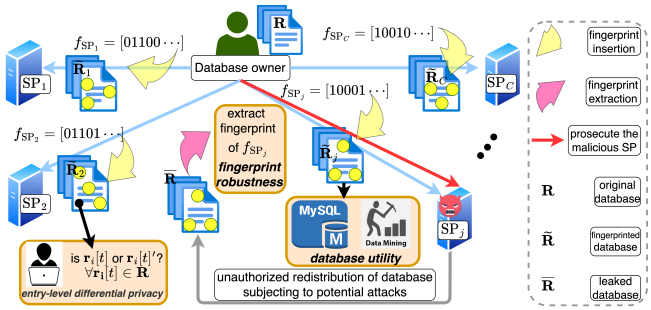


Fig. 1. System model. All shared copies of the database meet entry-level differential privacy, fingerprint robustness, and database utility requirements.

Objectives of database owner. In general, a database recipient (SP) can be any of the following: (1) an honest party who will use the received database to do SQL queries or data mining, (2) an attacker who will hijack the database to make illegal profits by making pirate copies of it, or (3) a curious party who will try to infer the original data entries. Since an SP can potentially play any of these three roles, the objectives of a database owner are to make sure that the shared database have

- (i) high utility in order to support accurate database queries and data mining tasks,
- (ii) liability guarantees to discourage illegal redistribution, i.e., successfully extract a malicious SP's fingerprint (even if a malicious SP distorts the fingerprint to mitigate detection) if the database is redistributed without authorization,
- (iii) entry-level privacy guarantees against attributes inference attacks, i.e., a data analyst cannot distinguish between $\mathbf{r}_i[t]$ and $\mathbf{r}_i[t]'$ by inferring its received copy.

Although (ii) and (iii) are different demands, they can be achieved at the same time, but at the cost of (i) (formally discussed in Section V). In this paper, we assume that the database owner is benign (i.e., it will not modify its own database in order to frame any SP).

Objectives of malicious SPs. From the perspective of malicious SPs, their objectives are to

- (a) redistribute received databases (make pirated copies) without being accused by means of distorting the inserted fingerprint and/or infer the original sensitive data entries,
- (b) preserve database utility to gain illegal profit.

Since the malicious SPs will introduce extra utility loss while distorting the fingerprint, (a) and (b) are also conflicting. Additionally, we assume that all malicious SPs are rational (i.e., they will not over-distort the content of a fingerprinted database, otherwise they cannot make illegal profit out of a pirated copy with poor utility).

Threats. Since we consider developing a mechanism to simultaneously achieve data privacy and liability guarantees, we also need to address the corresponding threats from these two aspects. In particular, the malicious SP can

- Infer the original values of data entries (in shared databases) by using its prior knowledge or other revealed data entries (we consider an adversary who knows all data entries except for one, and uses advanced learning methods to infer the original value of the unknown data entry).
- Conduct various attacks to distort the embedded fingerprint bit-strings, e.g., random bit flipping attack, subset attack, and correlation attack. In Section V, we discuss these attacks in detail and derive closed-form fingerprint robustness expression for each of them.

IV. Privacy-Preserving Fingerprinting

In this section, we first present the design principles of the proposed mechanism and also discuss some plausible but not viable alternatives. Next, we develop a general condition for a bit-level random response scheme to achieve entry-level DP database release/sharing. Then, we devise a concrete mechanism built upon such a scheme to achieve provable privacy guarantees for fingerprint insertion.

Principles of Mechanism Design. The core idea of database fingerprinting is to introduce small errors by changing randomly selected insignificant bits of encoded data entries using a certain probability [1], [35]. The collections of selected bits vary for different SPs and their fingerprinted values are determined by the unique fingerprint bit-strings of the SPs. Thus, database fingerprinting is a randomized mechanism, which essentially performs bitwise-randomization, i.e., changes the data values by introducing noise at the bit-level of data entries instead of directly perturbing the data (i.e., introducing noise at the entry-level). As a result, we also establish our entry-level DP fingerprinting scheme by conducting bitwise-randomization. To achieve a provable privacy guarantee, we calibrate the flipping probability (p) and the number of insignificant bits (K) based on the sensitivity of the data entries. Note that to achieve the desired privacy guarantee, we only

need to calibrate the binary noise (fingerprint) to "obfuscate" a certain number of insignificant bits that lead to the maximum difference between any pair of entries, instead of letting the binary noise "overwhelm" all the bits.

Other Plausible but not Viable Solutions. The entrywise-randomization adopted by the conventional DP output perturbation mechanisms, e.g., [11], [24], are infeasible as a building block of a fingerprinting mechanism, because they change all data entries by adding noises drawn from some probability distributions. Although local DP via randomized response only changes each data entry with a particular probability [3], connecting such probability with the randomly generated fingerprint bit-string is not straightforward. This is because randomly changing each bit of each data entry (by fingerprinting) may not lead to the identical random effect required by local DP. Hence, it is also not suitable for designing a database fingerprinting mechanism.

Another possible solution is to synthesize differentially-private relational databases while keeping the primary keys intact, and then inserting fingerprints into the results. This approach is also not viable, because data synthesis techniques usually generate artificial databases by sampling from noisy marginal and joint distributions of attributes, which require the clustering of similar attributes [9]. These methods heavily depend on the accurate clustering of highly correlated variables. Besides, data synthesis also requires additional computation to analyze other similar and public data to identify correlations and important marginals. To show the advantage of our mechanism, in Section VII, we compare it with DPSyn (a novel data synthesis technique) [33] followed by fingerprinting in [35].

Since the other solutions fail due to the aforementioned reasons, we consider achieving entry-level DP for the released database by using bit-level random response. In particular, when sharing a database with a specific SP, the values of selected insignificant bits of selected data entries are determined by XORing them with random binary variables, which vary for different data sharing instances (with various SPs). Such modification of bit positions in the database using different binary values can also be considered as inserting different fingerprints, which can be used to accuse a malicious SP if there is a data leakage. Moreover, to achieve high utility for the shared database, we simultaneously achieve entry-level DP and fingerprinting, instead of achieving them in a two-step appraoch (fingerprinting a differentially-private database) The two-step approach is suboptimal compared with our mechanism, because we directly harness the randomness (noises) introduced in fingerprint insertion and transform it into a provable privacy guarantee (entry-level DP). Thus, the privacy guarantee can be interpreted as "achieved free" during the fingerprint insertion. Whereas, the two-step solutions need to assign separate randomness (noises) budgets to achieve privacy and liability guarantee in a sequential manner.

A. Privacy-preserving Sharing via Bit-level Randomization

Traditional DP guarantees that the computed statistics from a database (e.g., mean or histogram) are independent of the absence or presence of an individual. However, in this work we consider the release (sharing) of the entire fingerprinted database, and the existence of a particular individual can be easily determined by checking its primary key in the released copy (discussed in Section III). Therefore, we focus on the privacy of database entries (attributes of individuals).

Definition 5 (Bit-level random response): A bit-level random response scheme (pseudorandomly) selects some bits of some data entries in a database and changes the bit values of such entries by conducting an XOR operation on them with independently generated random binary mark bits, denoted as B, where $B \sim \text{Bernoulli}(p)$.

Database fingerprinting schemes only mark the insignificant bits of the data entries to introduce tolerable error in the database. In this paper, we assume that the kth to the last bit of an entry is its kth insignificant bit. If the kth insignificant bit of attribute t of data record $\mathbf{r}_i$ (represented as $\mathbf{r}_i[t,k]$) is selected, then the bit-level random response scheme changes its value as $\mathbf{r}_i[t,k] \oplus B$, where $\oplus$ is the XOR operator, and B is a Bernoulli random variable with parameter p.

We develop the following condition for such a scheme to achieve ϵ-entry-level DP on the entire database.

Theorem 1: Given a relational database $\mathbf{R}$ with sensitivity Δ (Definition 3), a bit-level random response scheme, which only changes the last K bits of data entries, satisfies ϵ-entry-level DP if $K = \lfloor \log_2 \Delta \rfloor + 1$ and $p \geq \frac{1}{e^{\epsilon/K}+1}$.

Proof: Since we consider neighboring databases that have only a pair of different data entries which differ by at most Δ, it requires $K = \lfloor \log_2 \Delta \rfloor + 1$ bits to encode the difference. Then, by applying Definition 4, we have

$$
\begin{aligned}
&\frac{\Pr\left(\mathcal{M}(\mathbf{R}) = \widetilde{\mathbf{R}}\right)}{\Pr\left(\mathcal{M}(\mathbf{R}') = \widetilde{\mathbf{R}}\right)} \\
&\overset{(a)}{=} \prod_{k=1}^{K} \frac{\Pr\left(\mathbf{r}_i[t,k] \oplus B_{i,t,k} = \widetilde{\mathbf{r}}_i[t,k]\right)}{\Pr\left(\mathbf{r}'_i[t,k] \oplus B'_{i,t,k} = \widetilde{\mathbf{r}}_i[t,k]\right)} \\
&= \prod_{k=1}^{K} \frac{\Pr\left(B_{i,t,k} = \mathbf{r}_i[t,k] \oplus \widetilde{\mathbf{r}}_i[t,k]\right)}{\Pr\left(B'_{i,t,k} = \mathbf{r}'_i[t,k] \oplus \widetilde{\mathbf{r}}_i[t,k]\right)} \\
&\overset{(b)}{=} \prod_{k=1}^{K} \frac{p^{\left(\mathbf{r}_i[t,k] \oplus \widetilde{\mathbf{r}}_i[t,k]\right)} (1-p)^{\left(1-\mathbf{r}_i[t,k] \oplus \widetilde{\mathbf{r}}_i[t,k]\right)}}{p^{\left(\mathbf{r}'_i[t,k] \oplus \widetilde{\mathbf{r}}_i[t,k]\right)} (1-p)^{\left(1-\mathbf{r}'_i[t,k] \oplus \widetilde{\mathbf{r}}_i[t,k]\right)}} \\
&\overset{(c)}{=} \prod_{k=1}^{K} \left(\frac{1-p}{p}\right)^{\left((\mathbf{r}_i[t,k] - \mathbf{r}'_i[t,k])(2\widetilde{\mathbf{r}}_i[t,k]-1)\right)} \\
&\leq \prod_{k=1}^{K} \left(\frac{1-p}{p}\right)^{\left(\left|\mathbf{r}_i[t,k] - \mathbf{r}'_i[t,k]\right|(2\widetilde{\mathbf{r}}_i[t,k]-1)\right)} \\
&\leq \prod_{k=1}^{K} \frac{1-p}{p},
\end{aligned}
$$

where (a) can be obtained by assuming (without loss of generality) that $\mathbf{R}$ and $\mathbf{R}'$ differ at the tth attribute of the ith row, and thus the probability ratio at other entries cancel out. $\mathbf{r}_i[t,k]$ (or $\mathbf{r}'_i[t,k]$) represents the kth least significant bit of the tth attribute of $\mathbf{r}_i$ (or $\mathbf{r}'_i$), $B_{i,t,k}$ (or $B'_{i,t,k}$) is the random mark bit fingerprinted on $\mathbf{r}_i[t,k]$ (or $\mathbf{r}'_i[t,k]$), and $\widetilde{\mathbf{r}}_i[t,k]$ is

the identical result of the bit-level random response at this bit position. (b) is because each of the last K bits of entry $\mathbf{r}_i[t]$ (or $\mathbf{r}_i[t]'$) are changed independently with probability p, and (c) can be obtained by applying $u \oplus v = (1-u)v + u(1-v)$ for any binary variable u and v. Then, by making $\prod_{k=1}^{K} \frac{1-p}{p} \leq \epsilon$, we complete the proof. ■

In Figure 2, we present a toy example of using bit-wise randomization to achieve entry-level DP on a database $\mathbf{R}$. $\mathbf{R}'$ is the neighboring relational database of $\mathbf{R}$ (Definition 2). $\mathbf{R}$ and $\mathbf{R}'$ only differs in the 2nd attribute of the ith data record (highlighted cells in the upper panel of Figure 2). In this example, we assume $\Delta = 3$. Then, according to Theorem 1, by just flipping the last $K = \lfloor \log_2 3 \rfloor + 1 = 2$ insignificant bits of entries in $\mathbf{R}$ with probability p, a certain level of entry-level DP can be achieved. The lower panel of Figure 2 shows the binary representation of $\mathbf{R}$ and $\mathbf{R}'$, where the last 2 bits subject to bit-wise randomization are underlined. In the next section, we will show that the desired randomness (probability p) can be obtained as a result of fingerprint insertion.

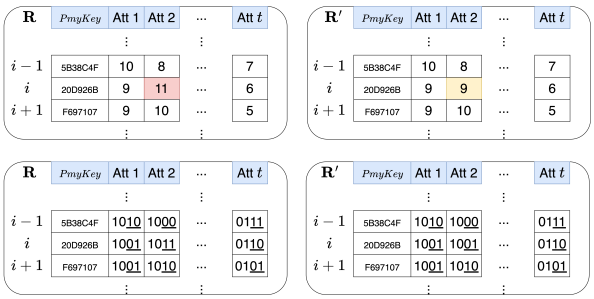


Fig. 2. An example of bit-wise randomizing last $K = 2$ insignificant bits of each entry to achieve entry-level DP on database $\mathbf{R}$ with sensitivity $\Delta = 3$.

B. ϵ-Entry-level Differentially-Private Fingerprinting

Due to the randomness involved in the bit-level random response scheme, for any given p and $\mathbf{R}$, the output databases will vary for each different run. However, in order to detect the guilty SP who leaks the database, it is required that the fingerprinted database shared with a specific SP must be unique and it can be reproduced by the database owner even if the mark bits, i.e., B's, are generated randomly. In this section, we discuss how to develop an instantiation of an ϵ-entry-level differentially-private fingerprinting mechanism based on the bit-level random response scheme, i.e., a mechanism that satisfies Theorem 1, and at the same time, is reproducible when sharing a fingerprinted copy with any specific SP using a given Bernoulli distribution parameter p (i.e., the probability of a bit being changed due to fingerprinting).

First, we collect all fingerprintable bits in $\mathbf{R}$, i.e., all insignificant bits (the last K bits) of all entries, in a set $\mathcal{P}$: $\mathcal{P} = \left\{ \mathbf{r}_i[t,k] \middle| i \in [1,N], t \in [1,T], k \in [1, \min\{K, K_t\}] \right\}$, where N is the number of data records in $\mathbf{R}$, and K_t represents the number of bits to encode the tth attribute in $\mathbf{R}$. When the database owner wants to share a fingerprinted copy of $\mathbf{R}$ with an SP with a publicly known external ID denoted as ID_{external}, it first generates an internal ID for this SP denoted as ID_{internal}. We will elaborate the generation of ID_{internal} in Section VI-A. Then, the database owner generates the unique fingerprint for this SP via $\mathbf{f} = HMAC(\mathcal{Y}|ID_{\text{internal}})$, which is a message authentication code (MAC) involving a cryptographic hash function and a secret cryptographic key ($\mathcal{Y}$ is the secret key of the database owner and $|$ represents the concatenation operator). We use L to denote the length of the generated fingerprint.[2]

The database owner also has a cryptographic pseudorandom sequence generator $\mathcal{U}$, which selects the data entries and their insignificant bits, and determines the mask bit x and fingerprint bit f (which is an element of the fingerprint bit-string $\mathbf{f}$) to obtain the Bernoulli random variable (i.e., $B = x \oplus f$). To be more specific, for each $\mathbf{r}_i[t,k]$ in $\mathcal{P}$, the database owner sets the initial seed as $s = \{\mathcal{Y}|\mathbf{r}_i.PmyKey|t|k\}$. If $\mathcal{U}_1(s) \mod \lfloor \frac{1}{2p} \rfloor = 0$ ($p = \frac{1}{e^{\epsilon/K}+1}$), then $\mathbf{r}_i[t,k]$ is fingerprinted. Next, the database owner decides the value of mask bit x by checking if $\mathcal{U}_2(s)$ is even or odd, and sets the fingerprint index $l = \mathcal{U}_3(s) \mod L$. By doing so, it obtains the mark bit as $B = x \oplus \mathbf{f}(l)$, and finally it changes the bit value of $\mathbf{r}_i[t,k]$ with $\mathbf{r}_i[t,k] \oplus B$. We summarize the steps to generate a fingerprinted database in Algorithm 1.

Algorithm 1: Generate $\mathcal{M}(\mathbf{R})$ for SP ID_{external}.

Input : Database $\mathbf{R}$, privacy budget ϵ, number of changeable bits K, Bernoulli distribution parameter $p = 1/(e^{\epsilon/K}+1)$, pseudorandom number sequence generator $\mathcal{U}$, database owner's secret key $\mathcal{Y}$

Output: fingerprinted database $\mathcal{M}(\mathbf{R})$ with ϵ-entry-level DP

```
Construct the fingerprintable set P.
Generate the internal ID, i.e., ID_internal for this SP
  (will be elaborated in Section VI-A).
Generate the fingerprint string, i.e.,
  f = HMAC(Y|ID_internal).
forall r_i[t,k] ∈ P do
    Set pseudorandom seed s = {Y|r_i.PmyKey|t|k},
    if U_1(s) mod ⌊1/(2p)⌋ = 0 then
        Set mask bit x = 0, if U_2(s) is even;
          otherwise x = 1.
        Set fingerprint index l = U_3(s) mod L.
        Let fingerprint bit f = f(l).
        Obtain mark bit B = x ⊕ f.
        Set r_i[t,k] = r_i[t,k] ⊕ B. {insert fingerprint}
```

Theorem 2: Algorithm 1 is ϵ-entry-level DP.

Proof: Since the value of $\mathcal{U}_j(s)$ (the jth random value generated by $\mathcal{U}$) is uniformly distributed for a seed s [8], we have $\Pr\left(\mathcal{U}_1(s) \mod \lfloor \frac{1}{2p} \rfloor = 0\right) = 1/\lfloor \frac{1}{2p} \rfloor > 2p$. Similarly, $\Pr(x = 0) = \frac{1}{2}$, thus, for any given fingerprint bit f, we also have $\Pr\left(B = 1, \mathcal{U}_1(s) \mod \lfloor \frac{1}{2p} \rfloor = 0\right) \geq \frac{1}{2} 2p = p$, which suggests that each $\mathbf{r}_i[t,k]$ will be changed (i.e., XORed by 1) with probability higher than p, and this satisfies the condition in Theorem 1. ■

Remark 2: The proposed database fingerprinting scheme is different from the existing ones discussed in Section II, as all

[2]We use MD5 to generate a 128-bits fingerprint string, since if the database owner shares C copies of its database and $L \geq \ln C$, the fingerprinting mechanism can thwart exhaustive search and various types of attacks [35].

existing schemes fingerprint each selected bit by replacing it with a new value obtained from the XOR of pseudorandomly generated mask bit x and fingerprint bit f. Hence, the new value is independent of the original bit value in the relational database. This is why the privacy guarantees of existing fingerprinting schemes cannot be explicitly analyzed. On the contrary, we fingerprint each selected bit by XORing it with a Bernoulli random variable B to make the fingerprinted entries dependent on the original bit value in the relational database. This enables us to derive a tight upper bound on the ratio of the probabilities of a pair of neighboring databases returning identical fingerprinted outcomes, which is the key step to further connect this bound to a provable privacy guarantee.

Note that $\mathcal{U}$ produces a sequence of random numbers using an initial seed, and it is computationally prohibitive to compute the next random number in the sequence without knowing the seed. Thus, from an SP's point of view, the results of $\mathcal{M}(\mathbf{R})$'s are random. However, $\mathcal{M}(\mathbf{R})$ can be reproduced by the database owner who has access to its own private key as well as the external and the determined internal IDs of SPs.

C. Extracting the Fingerprint

When the database owner observes a leaked (or pirated) database denoted as $\overline{\mathbf{R}}$, it will try to identify the traitor (malicious SP) by extracting the fingerprint from $\overline{\mathbf{R}}$ and comparing it with the fingerprints of SPs who have received a copy of its database. We present the fingerprint extraction procedure from a leaked fingerprinted database in Algorithm 2.

Algorithm 2: Fingerprint extraction procedure

Input : The original database $\mathbf{R}$, the leaked database $\overline{\mathbf{R}}$, the Bernoulli distribution parameter p, database owner's secret key $\mathcal{Y}$, pseudorandom number sequence generator $\mathcal{U}$, and a fingerprint template.

Output: The extracted fingerprint from the leaked database.

1 Initialize $\mathbf{c}_0(l) = \mathbf{c}_1(l) = 0, \forall l \in [1, L]$.
2 Construct the fingerprintable set $\overline{\mathcal{P}}$.
3 **forall** $\overline{\mathbf{r}_i} \in \overline{\mathcal{P}}$ **do**
4 Set pseudorandom seed $s = \{\mathcal{Y}|\mathbf{r}_i.PmyKey|t|k\}$,
5 **if** $\mathcal{U}_1(s) \bmod \lfloor \frac{1}{2p} \rfloor = 0$ **then**
6 Set mask bit $x = 0$, if $\mathcal{U}_2(s)$ is even; otherwise $x = 1$.
7 Set fingerprint index $l = \mathcal{U}_3(s) \bmod L$.
8 Recover mark bit $B = \overline{\mathbf{r}_i}[t, k] \oplus \mathbf{r}_i[t, k]$.
9 Recover fingerprint bit $f_l = x \oplus B$.
10 $\mathbf{c}_1(l) + +$, if $f_l = 1$; otherwise $\mathbf{c}_0(l) + +$.
11 **forall** $l \in [1, L]$ **do**
12 $\mathbf{f}(l) = 1$, if $\mathbf{c}_1(l) > \mathbf{c}_0(l)$; otherwise, $\mathbf{f}(l) = 0$.
13 Return the extracted fingerprint bit string $\mathbf{f}$.

Specifically, the database owner first initiates a fingerprint template $(f_1, f_2, \cdots, f_L) = (?, ?, \cdots, ?)$. Here, "?" means that the fingerprint bit at that position remains to be determined. Then, the database owner locates the positions of the fingerprinted bits as in Algorithm 1, and fills in each "?" using majority voting. To be more precise, it first constructs the fingerprintable sets $\overline{\mathcal{P}}$ from $\overline{\mathbf{R}}$, i.e., $\overline{\mathcal{P}} = \big\{\overline{\mathbf{r}_i}[t, k] \big| i \in [1, \overline{N}], t \in [1, T], k \in [1, \min\{K, K_t\}]\big\}$, where $\overline{\mathbf{r}_i}[t, k]$ is the kth insignificant bit of attribute t of the ith data record in $\overline{\mathbf{R}}$, and $\overline{N}$ is the number of records in $\overline{\mathbf{R}}$. Note that $\overline{N}$ may not be equal to N, because a malicious SP may conduct the subset attack (as will be discussed in Section V-C2) to remove some data records from the received database before leaking it. Second, the database owner selects the same bit positions, mask bit x, and fingerprint index l using the pseudorandom seed $s = \{\mathcal{Y}|\mathbf{r}_i.PmyKey|t|k\}$. Third, it recovers the mark bit as $B = \overline{\mathbf{r}_i}[t, k] \oplus \mathbf{r}_i[t, k]$ and fingerprint bit at index l as $f_l = x \oplus B$. Since the value of f_l may be changed due to the attacks launched by a malicious SP, the database owner maintains and updates two counting arrays $\mathbf{c}_0$ and $\mathbf{c}_1$, where $\mathbf{c}_0(l)$ and $\mathbf{c}_1(l)$ record the number of times f_l is recovered as 0 and 1, respectively. Finally, the database owner sets $\mathbf{f}(l) = 1$, if $\mathbf{c}_1(l) > \mathbf{c}_0(l)$, otherwise $\mathbf{f}(l) = 0$. The database owner compares the constructed fingerprint bit-string with the fingerprint customized for each SP who has received the database, and one of these SPs will be considered as guilty if there is a large overlap between its fingerprint and the constructed one. It has been shown that the database owner can correctly identify the malicious SP as long as the overlapping between fingerprints is above 50% [26].

V. Associating Privacy, Fingerprint Robustness, and Database Utility

Previously, we have presented a mechanism that achieves provable privacy guarantees when fingerprinting a database. Here, we investigate its impact on the database utility and fingerprint robustness, and also establish the connection between p (the probability of changing one insignificant bit of a data entry),[3] entry-level DP guarantee (ϵ), fingerprint robustness, and utility of shared databases. We visualize the relationships between these in Figure 3, where the arrow means "leads to". We have the high-level conclusion that privacy and fingerprint robustness are not conflicting objectives that can be achieved at the same time, however, at the cost of database utility.

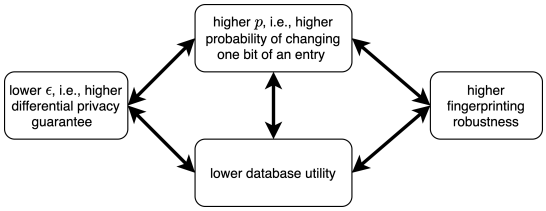


Fig. 3. Relationship among p (probability of changing one insignificant bit of an entry), privacy guarantee (ϵ), fingerprint robustness, and database utility.

A. Privacy against Attribute Inference Attacks

After receiving the fingerprinted database $\mathcal{M}(\mathbf{R})$, a malicious SP can leverage sophisticated learning methods to infer the original value of each data entry. In this section, we show

[3] p is the probability of the mark bit B taking value 1 (the probability of a specific bit is fingerprinted and changed, i.e., XORed by 1). The probability of the mark bit taking the value 0 is also p by design (i.e., the probability of a specific bit is fingerprinted but not changed, i.e., XORed by 0). Thus, the probability of a specific bit position being fingerprinted is $2p$ (see Line 6 in Algorithm 1). We would like to remind that $2p < 1$, as the probability of a specific bit position is not selected to be marked is $1 - 2p$.

that under our proposed privacy model (i.e., entry-level DP, Definition 4), the malicious SP's inference capability can never exceed a certain threshold.

In particular, we consider a malicious SP who has access to $\mathbf{R}_{/\boldsymbol{r}_i[t]}$ (i.e., the original values of all data entries except the tth attribute of the ith data record, $\boldsymbol{r}_i[t]$), and its inference capability is defined as $\text{InfCap} = \Pr(\boldsymbol{r}_i[t] = \zeta_1 | \mathcal{M}(\mathbf{R}), \mathbf{R}_{/\boldsymbol{r}_i[t]})$, which is the posterior probability of the unknown entry $\boldsymbol{r}_i[t]$ taking a specific value ζ_1. InfCap covers a wide-range of inference attacks using learning-based techniques, as most of the learning frameworks give outputs in terms of posterior probabilities, e.g., Bayesian inference and deep learning. We can have the following proposition about the inference capability of a malicious SP.

Proposition 1: No matter what learning-based inference attack the malicious SP conducts, its inference capability can never be higher than $\frac{\psi e^\epsilon}{\psi e^\epsilon + 1}$, i.e., $\text{InfCap} \leq \frac{\psi e^\epsilon}{\psi e^\epsilon + 1}$, where $\psi = \frac{\Pr(\boldsymbol{r}_i[t] = \zeta_1 | \, \mathbf{R}_{/\boldsymbol{r}_i[t]})}{\Pr(\boldsymbol{r}_i[t] = \zeta_2 | \, \mathbf{R}_{/\boldsymbol{r}_i[t]})}$ is the ratio of the malicious SP's prior knowledge of the unknown entry $\boldsymbol{r}_i[t]$ taking different values (i.e., ζ_1 and ζ_2) given all other entries are known.

Proposition 1 can be proved by using techniques presented in [37], [24], and we omit it due to space limitation. Given ψ, $\frac{\psi e^\epsilon}{\psi e^\epsilon + 1}$ decreases as ϵ decreases, it means that the higher the entry-level DP guarantee (smaller ϵ) is, the lower the inference capability of malicious SPs becomes. The above considered adversary who knows the entire database except one data entry is a standard threat model in the DP literature. In some of the real-world attacks, an adversary can also utilize some publicly known auxiliary information (e.g., correlations among data entries [37], [51], [11], [49] or social connections [30], [24]) to improve its inference capability. We will also work along this direction in future work.

The goal of attribute inference attack using data correlations is to compromise data privacy. In Section V-C3, we discuss its counterpart that compromises the fingerprint robustness; malicious SPs can also leverage the discrepancy between data correlations before and after fingerprint insertion to distort the embedded fingerprint bits.

B. Database Utility

Fingerprinting naturally changes the content of the database, and thus degrades the utility. Here, we evaluate the utility of a fingerprinted database from both data accuracy and data correlation perspectives: we quantify the impact of Algorithm 1 on the accuracy of each fingerprinted data entry and the joint probability distribution of any pair of attributes. The theoretical analyses are summarized in Proposition 2 and 3. These considered utility metrics are application independent, and in general, the higher the accuracy of data entries and pairwise joint distributions are, the better the task-specific application utilities get (e.g., classification accuracy and mean square error). We empirically validate this statement by considering task-specific application utilities in Section VII.

Proposition 2: Let $\mathbf{r}_i[t]$ and $\widetilde{\mathbf{r}_i[t]}$ be the original and the fingerprinted values of the tth attribute of the ith row. Then, the expected error caused by fingerprinting, i.e., $\mathbb{E}_{B \sim \text{Bernoulli}(p)} \left[\left| \mathbf{r}_i[t] - \widetilde{\mathbf{r}_i[t]} \right| \right]$, falls in $[0, \Delta p]$, where Δ is the sensitivity of a pair of neighboring relational databases. and p is the probability of a mark bit B taking value 1.

The proof is in Appendix A. Clearly, the higher the value of p, the larger the expected absolute difference between a fingerprinted data entry and the original value. It suggests that the database owner can set the value of p based on its requirement of data entry accuracy when generating a fingerprinted database, which achieves a certain level of entry-level DP, and vice versa. This leads us to the next corollary.

Corollary 1: Define **fingerprint density** as $||\mathcal{M}(\mathbf{R}) - \mathbf{R}\,||_{1,1}$, where $||\cdot||_{1,1}$ is the matrix $(1,1)$-norm which sums over the absolute value of each entry in the matrix. Then, we have $\mathbb{E}_{B \sim \text{Bernoulli}(p)} \left[||\mathcal{M}(\mathbf{R}) - \mathbf{R}\,||_{1,1} \right] \in [0, \Delta p N T]$.

In Section VI-B, we will exploit fingerprint density to develop a support vector technique (SVT)-based solution to share fingerprinted databases with multiple SPs.

Proposition 3: Let $\Pr(\mathbf{R}[t] = \pi, \mathbf{R}[z] = \omega)$ and $\Pr(\widetilde{\mathbf{R}[t]} = \pi, \widetilde{\mathbf{R}[z]} = \omega)$ be the joint probability of the tth attribute taking value π and the zth attribute taking value ω before and after fingerprint insertion, respectively. Then, $\Pr(\widetilde{\mathbf{R}[t]} = \pi, \widetilde{\mathbf{R}[z]} = \omega)$ falls in the range of $\Big[\Pr\big(\mathbf{R}[t] = \pi, \mathbf{R}[z] = \omega\big)(1-p)^{2K} + \Pr_{\min}(\mathbf{R}[t], \mathbf{R}[z])\big(1 - (1-p)^K\big)^2$, $\Pr\big(\mathbf{R}[t] = \pi, \mathbf{R}[z] = \omega\big)(1-p)^{2K} + \Pr_{\max}(\mathbf{R}[t], \mathbf{R}[z])\big(1 - (1-p)^K\big)^2 \Big]$. $\Pr_{\min}(\mathbf{R}[t], \mathbf{R}[z])$ (or $\Pr_{\max}(\mathbf{R}[t], \mathbf{R}[z])$) is the minimum (or maximum) joint probability of attributes t and z in $\mathbf{R}$.

The proof is in Appendix B. By marginalizing over $\mathbf{R}[t]$ and $\widetilde{\mathbf{R}[z]}$, we can have the following corollary.

Corollary 2: Let $\Pr(\mathbf{R}[t] = \pi)$ and $\Pr(\widetilde{\mathbf{R}[t]} = \pi)$ be the marginal probability of the tth attribute taking value π before and after fingerprint insertion, respectively. Then, $\Pr(\widetilde{\mathbf{R}[t]} = \pi)$ belongs to $\Big[\Pr\big(\mathbf{R}[t] = \pi\big)(1-p)^{2K} + \Pr_{\min}(\mathbf{R}[t])\big(1 - (1-p)^K\big)^2$, $\Pr\big(\mathbf{R}[t] = \pi\big)(1-p)^{2K} + \Pr_{\max}(\mathbf{R}[t])\big(1 - (1-p)^K\big)^2 \Big]$, where $\Pr_{\min}(\mathbf{R}[t])$ (or $\Pr_{\max}(\mathbf{R}[t])$) is the minimum (or maximum) marginal probability of the tth attribute in $\mathbf{R}$.

Thus, when p is small, both joint distributions and marginal distributions will be close to that of the original databases, i.e., the fingerprinted database will have higher statistical utility.

C. Fingerprint Robustness against Attacks

Although, Li et al. [35] analyzed fingerprint robustness by studying the false negative rate (i.e., the probability that the database owner fails to extract the exact fingerprint from a pirated database), they do not establish the direct connection between the robustness and the tuning parameter (the fingerprinting ratio, which can be interpreted as a counterpart of p in our work) in their mechanism.

In this paper, we investigate the robustness of the proposed fingerprinting mechanism against three attacks, i.e., the random bit flipping attack [1], [35], [14], subset attack [35], [10], [50], [14], and correlation attack [50], [26]. In the following, we quantitatively analyze the relationship between p (the probability of changing one insignificant bit of a data entry) and fingerprint robustness against these four attacks. The

relationship between ϵ and fingerprint robustness can easily be obtained by applying Theorem 1.

1) Robustness Against Random Flipping Attack: In random flipping attack, a malicious SP flips each of the K last bits of data entries in $\widetilde{\mathbf{R}}$ with probability γ_{rnd} with the goal of distorting the data in the fingerprinted positions. In [26], the authors have empirically shown that the malicious SP ends up being uniquely accusable as long as the extracted fingerprint from the leaked database has more than 50% matches with the malicious SP's fingerprint.

Yet, as the database owner shares more fingerprinted copies of database with different SPs, to uniquely hold the correct malicious SP responsible, it requires more bit matches between the extracted fingerprint and the malicious SP's fingerprint. Thus, the number of bit matches (denoted as D, $D \leq L$) should be set based on the number of fingerprinted sharings of the database. Appendix C discusses how to determine D.

Given the determined D, we evaluate the robustness of the proposed fingerprinting mechanism against random bit flipping attack in terms of the probability (denoted as $P_{\text{rbst\_rnd}}$) that the database owner successfully extracts any D fingerprint bits of the malicious SP. Let the lth bit of the fingerprint string be embedded w_l times in $\widetilde{\mathbf{R}}$ (which happens with probability $(1/L)^{w_l}$). Thus, to extract this fingerprint bit correctly from a copy of $\widetilde{\mathbf{R}}$ that is compromised by the random bit flipping attack, the database owner needs to make sure that at most $\lfloor w_l/2 \rfloor$ bits in $\widetilde{\mathbf{R}}$ that are marked by the lth bit of the fingerprint string are flipped by the malicious SP, which happens with probability $p_l = \sum_{q=0}^{\lfloor w_l/2 \rfloor} \binom{w_l}{q} \gamma_{\text{rnd}}^q (1-\gamma_{\text{rnd}})^{w_l - q}$.

Let m be the number of fingerprinted bit positions in the database ($m \leq NKT$) received by the malicious SP, and define set $\mathcal{W}$ as $\mathcal{W} = \{w_1, w_2, \cdots, w_L > 0 | \sum_{l=1}^{L} w_l = m\}$. Let also $\mathcal{L}_D$ be the collection of any D bits of the malicious SP's fingerprint ($|\mathcal{L}_D| = D$). Then, we can obtain the closed form expression of $P_{\text{rbst\_rnd}}$ in terms of p as $P_{\text{rbst\_rnd}} = \sum_{m=1}^{NKT} \Big(\sum_{w_l \in \mathcal{W}, l \in [1,L]} \sum_{\mathcal{L}_D} \prod_{l \in \mathcal{L}_D} p_l (\frac{1}{L})^{w_l} \Big) \binom{NKT}{m} (2p)^m (1-2p)^{NKT-m}$, which is monotonically increasing with p ($p < 0.5$) (detailed analysis is deferred to Appendix C). Thus, a higher p leads to more robustness against the random bit flipping attack.

2) Robustness Against the Subset Attack: In subset attack, the malicious SP generates a pirated database by selecting each data record in $\widetilde{\mathbf{R}}$ for inclusion (in the pirated database) with probability γ_{sub}. This attack is shown to be much weaker than the random bit flipping attack [35], [26], [50]. According to [35] (page 40), the subset attack cannot succeed (i.e., distorting even one fingerprint bit) unless the malicious SP excludes all the rows fingerprinted by at least one fingerprint bit. Thus, we measure the robustness of the proposed fingerprinting mechanism against subset attack using the probability (denoted as $P_{\text{rbst\_sub}}$) that the malicious SP fails to exclude all fingerprinted rows involving a particular fingerprint bit (note that our analysis can be easily generalized for excluding a fraction of fingerprinted rows). Since the probability that a specific row is fingerprinted by a specific fingerprint bit is $1-(1-p/L)^{KT}$, we have the closed form expression of $P_{\text{rbst\_sub}}$ in terms of p (the probability of changing an insignificant bit of an entry) as $P_{\text{rbst\_sub}} = 1 - \sum_{n=1}^{N} \binom{N}{n} (\gamma_{\text{sub}})^n \Big[1-(1-p/L)^{KT}\Big]^n (1-p/L)^{KT(N-n)} = 1 + \Big[1-(1-p/L)^{KT}\Big]^N - \Big[1-(1-p/L)^{KT} + \gamma_{\text{sub}}(1-p/L)^{KT}\Big]^N$. Clearly, the larger p leads to less difference between $1-(1-p/L)^{KT}$ and $1-(1-p/L)^{KT} + \gamma_{\text{sub}}(1-p/L)^{KT}$, which suggests that $P_{\text{rbst\_sub}}$ also monotonically increases with p.

Note that in the subset attack, the primary keys are shared intact, but the content is shared partially. It is different with the scenario where the malicious SP just leaks specific attributes without the primary keys. Our proposed mechanism can only have privacy guarantee but no liability guarantee for the latter scenario. However, without the primary keys, the leaked information is not a valid relational database anymore and it cannot support operations like database union and intersection, thus, it is considered to have no database utility and it will not help the malicious SP make illegal profit.

3) Robustness Against Correlation Attack: In [26], the authors identify a correlation attack against database fingerprinting mechanisms, which takes advantage of the intrinsic correlation between data entries in the database to infer and compromise the potentially fingerprinted bit positions. In particular, the malicious SP changes the insignificant bits of entries in $\widetilde{\mathbf{R}}$ if the data entries satisfy $\big| \Pr\big(\widetilde{\mathbf{R}[t]} = \pi, \widetilde{\mathbf{R}[z]} = \omega\big) - \Pr\big(\mathbf{R}[t] = \pi, \mathbf{R}[z] = \omega\big)\big| \geq \tau, \forall z \in [1,T], \forall \omega$, where τ is a predetermined parameter for this attack.

Similar to [26], we adopt the confidence gain of the malicious SP (denoted as G) to analyze the robustness of the proposed fingerprinting mechanism against the correlation attack. The confidence gain measures the knowledge of a potentially fingerprinted data entry under correlation attack over random guess. To be more specific, G is defined as the ratio between the probability that a specific entry (whose original tth attribute takes value π) will be selected to be compromised in the correlation attack and the probability that such entry will be selected to be compromised in the random bit flipping attack. Mathematically, this can be shown as $G = \frac{1 - \prod_{z \in [1,t], z \neq t} \prod_{\omega} \Pr\Big(\big|\Pr\big(\widetilde{\mathbf{R}[t]} = \pi, \widetilde{\mathbf{R}[z]} = \omega\big) - \Pr\big(\mathbf{R}[t] = \pi, \mathbf{R}[z] = \omega\big)\big| \leq \tau\Big)}{(1-(1-p)^K)\Pr(\mathbf{R}[t] = \pi)}$.

In Appendix D, we show that G decreases as p increases when our proposed entry-level DP fingerprinting mechanism is used. This implies that the robustness of our proposed fingerprinting mechanism also increases with p.

VI. Sharing Multiple Databases

A major challenge in practical use of DP is that data privacy degrades if the same statistics are repeatedly calculated and released using the same differentially-private mechanism. The same is true for sharing a database with multiple SPs. If different fingerprinted copies of the same database are shared multiple times, the average of them may converge to the original database (known as average attack), which implies that privacy guarantee of Algorithm 1 degrades linearly with number of sharings.

If the privacy budget is depleted, then the database curator will just stop answering to the same queries (e.g., page 42 and 56 of [16]). Thus, in practice, the database owner will also release its database only to a limited number of SPs, and for each released copy, it will have certain data privacy and fingerprint robustness requirements. According to Figure 3, these requirements can both be fulfilled if the utility of the shared database is compromised to a certain

extent (but not significantly as will be corroborated in Section VII). Based on Section V, we know that the database utility can be characterized by the fingerprint density (defined in Corollary 1), because the higher the fingerprint density is, the lower the database utility becomes, and the database owner can control the utility of repeatedly shared databases using fingerprint density. Hence, we let the database owner only share a fingerprinted database, $\mathcal{M}(\mathbf{R})$, if its fingerprint density, $||\mathcal{M}(\mathbf{R}) - \mathbf{R}||_{1,1}$, is beyond a predetermined publicly known numerical threshold, Γ, in order to meet the requirement of high data privacy guarantee and fingerprint robustness.

As discussed in Section IV-B, the fingerprinted database $\mathcal{M}(\mathbf{R})$ customized for a particular SP depends on an internal ID assigned by the database owner to the corresponding SP. Since the internal ID of the SP is an input for inserting the fingerprint (Algorithm 1), whether $||\mathcal{M}(\mathbf{R}) - \mathbf{R}||_{1,1}$ is higher than Γ also depends on the assigned internal ID. As a consequence, when an SP queries the database, the database owner needs to keep generating a new internal ID for it until the resulting fingerprint density is above Γ. Moreover, this process (i.e., internal ID generation and fingerprint density comparison with the threshold) also needs to be performed in a privacy-preserving manner. The reason is that according to Section V (Proposition 2 and Corollary 1), fingerprint density provides additional knowledge about the fingerprint robustness and privacy guarantee. If a malicious SP accurately knows for sure that its received database has fingerprint density higher than Γ, i.e., the database owner cannot plausibly deny, the malicious SP can estimate the percentage of changed entries due to fingerprint, and further distort the fingerprint.

The above discussion inspires us to resort to the sparse vector technique (SVT) [16], [40], that only releases a noisy query result when it is beyond a noisy version of Γ, to design a mechanism for sharing multiple entry-level differentially-private fingerprinted databases and at the same time controlling the cumulative privacy loss. The unique benefit of SVT is that it can answer multiple queries while paying the cost of privacy only for the ones satisfying a certain condition, e.g., when the result is beyond a given threshold. In Section VI-A, we present an intermediate step which considers only one SP, determines its internal ID, and conducts the comparison between the resulting fingerprint density and threshold under entry-level DP guarantee. In Section VI-B, we compose this intermediate step for C times to determine the internal IDs for C SPs and share different fingerprinted databases.

A. Intermediate Step: Determining Internal ID for One SP

As elaborated earlier, the database owner needs to assign an internal ID to an SP in order to achieve $||\mathcal{M}(\mathbf{R}) - \mathbf{R}||_{1,1} > \Gamma$ for the purpose of simultaneously meeting data privacy and fingerprint robustness requirements. To achieve differential privacy for this intermediate step, we perturb both $||\mathcal{M}(\mathbf{R}) - \mathbf{R}||_{1,1}$ and Γ, and consider the noisy comparison $||\mathcal{M}(\mathbf{R}) - \mathbf{R}||_{1,1} + \mu > \Gamma + \rho$, where μ and ρ are Laplace noises. Establishing the noisy comparison is a standard approach in SVT (see [16] page 57, and [40] page 639).

Next, we formally present the intermediate step. When the database owner receives a query from a new SP (suppose that this SP is the cth SP and $c \in [1, C]$), it generates an instance of internal ID for the cth SP via $ID^c_{\text{internal}} = Hash(\mathcal{K}|c|i)$, where $i \in \{1, 2, \cdots\}$ denotes the sequence number of this trial to generate ID^c_{internal}. Then, the database owner generates the fingerprinted database via Algorithm 1 with the internal ID set as ID^c_{internal} in Line 2. Similarly, we denote the fingerprinted database generated for the cth SP at the ith trial as $\mathcal{M}^c_i(\mathbf{R})$. Next, the database owner conducts the noisy comparison $||\mathcal{M}^c_i(\mathbf{R}) - \mathbf{R}||_{1,1} + \mu_i > \Gamma + \rho_i$, where $\mu_i \sim \text{Lap}(\Delta/\epsilon_2)$ and $\rho_i \sim \text{Lap}(\Delta/\epsilon_3)$. Here, ϵ_2 and ϵ_3 are the privacy budgets used to control the accuracy of the noisy comparison. If $||\mathcal{M}^c_i(\mathbf{R}) - \mathbf{R}||_{1,1} + \mu_i > \Gamma + \rho_i$ holds, then the database owner returns a symbol $\top$ and immediately terminates the intermediate step. This means that ID^c_{internal} generated at the ith trial for the cth SP can lead to a fingerprinted database satisfying the data privacy and fingerprint robustness requirements. Otherwise, the database owner returns a symbol $\perp$, increases i by 1, and continues the process. We summarize this intermediate step in Algorithm 3. This entire process achieves entry-level DP as proven in the following theorem. Note that the identified ID^c_{internal} is not released to the SP. As we will show in Section VII-C, an instance of ID^c_{internal} satisfying $||\mathcal{M}^c_i(\mathbf{R}) - \mathbf{R}||_{1,1} + \mu_i > \Gamma + \rho_i$ can usually be generated in 1 or 2 trials depending on the ratio of ϵ_2 and ϵ_3.

Algorithm 3: Determine the Internal ID for One SP

Input : Original database $\mathbf{R}$, fingerprinting scheme $\mathcal{M}$, sequence number of a new SP, i.e., c, threshold Γ, and privacy budget ϵ, ϵ_2, and ϵ_3.
Output: $\{\perp, \perp, \cdots, \perp, \top\}$.

1 **forall** $i \in \{1, 2, 3, \cdots\}$ **do**
2 Get an instance of internal ID for the cth SP, $ID^c_{\text{internal}} = Hash(\mathcal{K}|c|i)$.
3 Get $\mathcal{M}^c_i(\mathbf{R})$ by calling Algorithm 1 with ID^c_{internal} and privacy budget ϵ.
4 Sample $\mu_i \sim \text{Lap}(\frac{\Delta}{\epsilon_2})$ and $\rho_i \sim \text{Lap}(\frac{\Delta}{\epsilon_3})$.
5 **if** $||\mathcal{M}^c_i(\mathbf{R}) - \mathbf{R}||_{1,1} + \mu_i \geq \Gamma + \rho_i$ **then**
6 Output $a_i = \top$. {i**th trial meets the requirement**}
7 Terminate the algorithm.
8 **else**
9 Output $a_i = \perp$. {**trial does not meet the requirement**}

Theorem 3: Algorithm 3 achieves $(\epsilon_2 + \epsilon_3)$-entry-level DP.

Proof: Suppose that Algorithm 3 terminates with l outputs (it takes l tries to determine ID^c_{internal}, leading to a "TRUE" condition for the noisy comparison). We represent the output sequence as $\mathbf{a}$, i.e.,

$$\mathbf{a} = [a_1, a_2, \cdots, a_l] = \{\perp\}^{l-1} \cup \{\top\}.$$

By defining

$$f_i(\mathbf{R}, z) = \Pr(||\mathcal{M}_i(\mathbf{R}) - \mathbf{R}||_{1,1} + \mu_i < \Gamma + z_i),$$
$$g_i(\mathbf{R}, z) = \Pr(||\mathcal{M}_i(\mathbf{R}) - \mathbf{R}||_{1,1} + \mu_i \geq \Gamma + z_i),$$

where z_i is an instance of ρ_i generated at Line 5 in Algorithm 3. Then, we can have

$$
\begin{aligned}
&\frac{\Pr\Big(\text{DetermineTheInternalIDforOneSP}(\mathbf{R})=\mathbf{a}\Big)}{\Pr\Big(\text{DetermineTheInternalIDforOneSP}(\mathbf{R}')=\mathbf{a}\Big)}\\
=&\frac{\int_{-\infty}^{\infty}\Pr(\rho_i=z_i)\prod_{i=1}^{l-1}f_i(\mathbf{R},z_i)g_l(\mathbf{R},z_i)dz_i}{\int_{-\infty}^{\infty}\Pr(\rho_i=z_i)\prod_{i=1}^{l-1}f_i(\mathbf{R}',z_i)g_l(\mathbf{R}',z_i)dz_i}\\
\overset{(*)}{=}&\frac{\int_{-\infty}^{\infty}\Pr(\rho_i=z_i-\Delta)\prod_{i=1}^{l-1}f_i(\mathbf{R},z_i-\Delta)g_l(\mathbf{R},z_i-\Delta)dz_i}{\int_{-\infty}^{\infty}\Pr(\rho_i=z_i)\prod_{i=1}^{l-1}f_i(\mathbf{R}',z_i)g_l(\mathbf{R}',z_i)dz_i}\\
=&\clubsuit,
\end{aligned}
$$

where $(*)$ is obtained by changing all the integration variables, i.e., z_i's, to $(z_i-\Delta)$'s, $\forall i\in\{1,2,3,\cdots\}$. Next, we investigate the three parts of the integrand in the numerator of $\clubsuit$ separately.

First, we have $\Pr(\rho_i=z_i-\Delta)\le e^{\epsilon_3}\Pr(\rho_i=z_i)$, as ρ_i is attributed to a Laplace distribution whose parameter is calibrated using Δ.

Second, suppose $\mathbf{R}$ and $\mathbf{R}'$ differs at r_{ij} and r'_{ij}. Then,

$$
\begin{aligned}
&||\mathcal{M}_i(\mathbf{R}')-\mathbf{R}'||_{1,1}-||\mathcal{M}_i(\mathbf{R})-\mathbf{R}||_{1,1}\\
=&|\widetilde{r'_{ij}}-r'_{ij}|-|\widetilde{r_{ij}}-r_{ij}|\le\Delta,
\end{aligned}
$$

where $\widetilde{r_{ij}}$ (or $\widetilde{r'_{ij}}$) is the fingerprinted version of r_{ij} (or r'_{ij}). The equality follows from that for any specific SP (which uniquely determines a pseudorandom seed), Algorithm 1 will select exactly the same bit positions in both $\mathbf{R}$ and $\mathbf{R}'$ to insert the fingerprint, and all selected bits except for the different entry between $\mathbf{R}$ and $\mathbf{R}'$ will also be replaced with the exact same bit values. The inequality is because both $|\widetilde{r_{ij}}-r_{ij}|$ and $|\widetilde{r'_{ij}}-r'_{ij}|$ are upper bounded by Δ. As a result, for the second part in $\clubsuit$, we obtain

$$
\begin{aligned}
&f_i(\mathbf{R},z_i-\Delta)\\
=&\Pr\big(||\mathcal{M}_i(\mathbf{R})-\mathbf{R}||_{1,1}+\mu_i<\Gamma+z_i-\Delta\big)\\
\le&\Pr\big(||\mathcal{M}_i(\mathbf{R}')-\mathbf{R}'||_F+\mu_i-\Delta<\Gamma+z_i-\Delta\big)\\
=&f_i(\mathbf{R}',z_i),
\end{aligned}
$$

where the inequality holds since we replace $||\mathcal{M}_i(\mathbf{R})-\mathbf{R}||_{1,1}$ by a smaller value, i.e., $||\mathcal{M}_i(\mathbf{R}')-\mathbf{R}'||_F-\Delta$, which decreases the probability.

Third, since μ_i is a Laplace noise, which is also calibrated using Δ, we have

$$
\begin{aligned}
&g_l(\mathbf{R},z_i-\Delta)\\
=&\Pr\Big(||\mathcal{M}_i(\mathbf{R})-\mathbf{R}||_{1,1}+\mu_i\ge\Gamma+z_i-\Delta\Big)\\
\le&e^{\epsilon_2}\Pr\Big(||\mathcal{M}_i(\mathbf{R}')-\mathbf{R}'||_{1,1}+\mu_i-\Delta\ge\Gamma+z_i-\Delta\Big)\\
=&e^{\epsilon_2}g_l(\mathbf{R}',z_i).
\end{aligned}
$$

Hence,

$$
\begin{aligned}
\clubsuit\le&\frac{\int_{-\infty}^{\infty}e^{\epsilon_3}\Pr(\rho_i=z_i)\prod_{i=1}^{l-1}f_i(\mathbf{R}',z_i)e^{\epsilon_2}g_l(\mathbf{R}',z_i)dz_i}{\int_{-\infty}^{\infty}\Pr(\rho_i=z_i)\prod_{i=1}^{l-1}f_i(\mathbf{R}',z_i)g_l(\mathbf{R}',z_i)dz_i}\\
=&e^{\epsilon_2+\epsilon_3}.
\end{aligned}
$$

which completes the proof. ■

Note that although we allocate the privacy budget ϵ when generating the fingerprinted database for the SP at Line 3 in Algorithm 3, it does not contribute to the total privacy loss. This is because here, ID^c_{internal} is used for fingerprint insertion, but the numerical fingerprinted database has not been shared yet.

B. Composition of Intermediate Steps: Releasing Multiple Fingerprinted Databases

We have presented an intermediate step, in which, to guarantee that an SP receives a copy of fingerprinted database, the database owner keeps generating an instance of internal ID for it until the noisy comparison result is "TRUE". Now, we show how to compose the intermediate steps for C times to determine the internal IDs for C SPs, and at the same time, share the corresponding fingerprinted databases (generated using their final internal IDs) with them. The workflow is summarized in Algorithm 4. Its differences with Algorithm 3 are highlighted in the boxes.

Algorithm 4: Share Fingerprinted Databases with C SPs

Input : Original database $\mathbf{R}$, fingerprinting scheme $\mathcal{M}$, sequence number of SPs, i.e., $\{1,2,\cdots,C\}$, threshold Γ, and privacy budget ϵ, ϵ_2, ϵ_3 and δ'.

Output: $\{a_1,a_2,a_3,\cdots,a_C\}$.

```
1  Set count = 0.
2  forall [cth SP c ∈ {1,2···,C}] do
3      forall i ∈ {1,2,3,···} do
4          Generate an instance of internal ID for the cth
           SP via ID^c_internal = Hash(K|c|i).
5          Generate M^c_i(R) by calling Algorithm 1 with
           ID^c_internal and privacy budget ε.
6          Sample μ_i ~ Lap(Δ/ε_2) and ρ_i ~ Lap(Δ/ε_3).
7          if ||M^c_i(R) − R||_{1,1} + μ_i ≥ Γ + ρ_i then
8              [Output a_i = M^c_i(R)].
9          else
10             Output a_i = ⊥.
```

If the database owner wants to share its database with more than C different SPs, it can reduce the value of the fingerprint density threshold Γ, which, however, compromises the privacy and fingerprint robustness of shared databases, because reducing Γ increases utility of shared databases. We show the privacy guarantee of Algorithm 4 in Theorem 4.

Theorem 4: Algorithm 4 achieves (ϵ_0,δ_0)-entry-level DP with $\epsilon_0=\sqrt{2C\ln(1/\delta')}(\epsilon+\epsilon_2+\epsilon_3)+C\big(\epsilon(e^{\epsilon}-1)+(\epsilon_2+\epsilon_3)(e^{\epsilon_2+\epsilon_3}-1)\big)$, and $\delta_0=2\delta'$.

Proof: Algorithm 4 is the composition of C rounds of Algorithm 3 together with C rounds of Algorithm 1. According to the advanced composition theorem [16], C rounds of Algorithm 3 and C rounds of Algorithm 1 are $(\sqrt{2C\ln(\frac{1}{\delta'})}(\epsilon_2+\epsilon_3)+C(\epsilon_2+\epsilon_3)(e^{\epsilon_2+\epsilon_3}-1),\delta')$-entry-level DP and $(\sqrt{2C\ln(\frac{1}{\delta'})}\epsilon+C\epsilon(e^{\epsilon}-1),\delta')$-entry-level DP, respectively. Then, by simple composition of those two, we complete the proof. ■

Privacy budget allocation. In practice, given the cumulative privacy budget ϵ_0 and δ_0, we need to decide the values of ϵ, ϵ_2, and ϵ_3. Since ϵ is used to obtain the fingerprinted database, its value should be determined based on the specific database of interest and the requirements about database utility and fingerprint robustness (discussed in Section V).

Furthermore, we note that $(\epsilon_2 + \epsilon_3)$ is used to obtain the internal IDs of SPs. Once ϵ is decided, the database owner can solve for $(\epsilon_2+\epsilon_3)$ numerically, i.e., $\left(\sqrt{2C\ln(1/\delta')}-C\right)(\epsilon_2+\epsilon_3)+C(\epsilon_2+\epsilon_3)e^{\epsilon_2+\epsilon_3} = \epsilon_0 - \left(\sqrt{2C\ln(1/\delta')}-C\right)\epsilon - C\epsilon e^{\epsilon}$. Suppose the numerical solution is $(\epsilon_2 + \epsilon_3) = \epsilon^*$. Then, we need to allocate ϵ^* to ϵ_2 and ϵ_3. Inspired by the analysis in [40], we observe that ϵ_2 and ϵ_3 control the accuracy of noisy comparison, i.e., $||\mathcal{M}_i(\mathbf{R}) - \mathbf{R}||_{1,1} + \mu_i \geq \Gamma + \rho_i$ (or equivalently, $||\mathcal{M}_i(\mathbf{R}) - \mathbf{R}||_{1,1} - \Gamma \geq \rho_i - \mu_i$) in both Algorithms 3 and 4. To boost the accuracy of the noisy comparison, we minimize the variance of the difference between ρ_i and μ_i. Since they are both Laplace random variables, the variance of their difference is $2(\Delta/\epsilon_2)^2 + 2(\Delta/\epsilon_3)^2$. Clearly, given ϵ^*, the variance is minimized when $\epsilon_2 = \epsilon_3 = \epsilon^*/2$.

Note that in classical SVT, the database owner does not respond to all the queries, i.e., it merely reports "$\perp$" if the considered noisy comparison is "FALSE" (page 55 [16]), yet, this is not user-friendly in database sharing, especially, when the database owner still has remaining privacy budget. In our proposed SVT-based solution, we make sure all SPs get their fingerprinted databases as long as they are among the top C SPs sending the query request. This is achieved by letting the database owner keep generating new internal IDs for the SPs until the noisy comparison turns out to be "TRUE", and this approach does not violate the design principle of SVT.

VII. Experiments

We evaluate the developed entry-level differentially-private relational database fingerprinting mechanism under both single and multiple database sharing scenarios.

A. Experiment Setup

Databases. We consider two publicly available databases from UCI machine learning repository [2]. First is a medium size nursery school application database, which contains data of 12,960 applicants. Each applicant has 8 categorical attributes, e.g., "form of the family" (complete, completed, incomplete, or foster). Each data record is associated with one of the five labels, i.e., "not_recom", "recommend", "very_recom", "priority", and "spec_prior". Second is a large size Census database recording 14 discrete or categorical attributes (e.g., age, workclass, and marital-status) of 32,561 individuals, in which each individual is labeled as either '$> 50K$' or '$\leq 50K$', which represents the income. Since both databases contain categorical attributes, we need to encode them as integers before fingerprinting.

Database encoding. Similar to [26], to fingerprint discrete attributes (e.g., "age" in Census database), the database owner will first sorts the values in an ascending order and then divides them into non-overlapping ranges, which are then encoded as ascending integers starting from 0. For categorical attributes, e.g., "marital-status" in the census database, the instances are first mapped to a high dimensional space via the word embedding. Words (instances) having similar meanings appear roughly in the same area of the space, and the values of their integer codes will also be close. In the considered nursery school application database, the maximum integer representation of a data entry is 4 (we do not fingerprint the labels, which will be used in a classification task to evaluate the utility of the fingerprinted database). Note that we drop the attribute of "fnlwgt" in the Census database, because it represents the number of people the census believes a specific row represents. After dropping the "fnlwgt" attribute, each row of the Census database can be interpreted as a specific individual. Besides, we encode "capital-gain", "capital-loss", and "native-country" attributes as binary, because the columns of "capital-gain" and "capital-loss" are very sparse, and nearly all "native-country" values are "United-States". After encoding, the maximum integer representation of a data entry in the Census database is 15 (we also do not fingerprint the binary labels, i.e., '$> 50K$' or '$\leq 50K$', in order to conduct task specific utility evaluations).

Sensitivity control on nursery school application database. Since the integer representations of data entries vary from 0 to 4 in the nursery school application database, the sensitivity is $\Delta = 4$. Thus, the proposed mechanism needs to fingerprint $K = \log_2 4 + 1 = 3$ (see Theorem 1) least significant bits of each data entry. This, however, may significantly compromise the utility of the fingerprinted database. To control the sensitivity (and hence improve the utility), we make the following observation. We calculate the fraction of pairwise absolute differences taking a specific value (between the attributes) and show the results in Table II. Clearly, in each class, a large portion of the absolute differences are 0 and 1, and only a small fraction of them have difference larger than 1. Thus, in the experiments, we consider sensitivity $\Delta = 1$ with the assumption that the different entries in a pair of neighboring nursery databases can change by at most by 1, otherwise, it introduces a rare event (e.g., outliers that occurs with very low probability) in the database. Our approach to control the sensitivity is similar to the restricted sensitivity [5] (that calculates sensitivity on a restricted subset of the database, instead of all possible data records) and smooth sensitivity [28] (which smooths the data records after partitioning them into non-overlapping groups). Note that it has been widely recognized that rare events or outliers consume extra privacy budget, and this is a common problem in differentially-private database queries [12], [39], [15], [32]. Controlling local and global sensitivity in differential privacy is a separate topic, and it is beyond the scope of this paper.

abs. diff.	0	1	2	3	4
not_recom	40%	46.79%	7.08%	5.12%	1%
recommend	93.75%	6.25%	0	0	0
very_recom	35.10%	49.71%	10.66%	4.39%	0.13%
priority	36.04%	44.65%	11.31%	5.07%	2.93%
spec_prior	50.50%	37.19%	9.01%	3.29%	0

TABLE II. Fraction of pairwise absolute differences between instances of attributes.

Note that for the Census database, we do not control its sensitivity, because the absolute differences between attributes of individuals are more evenly distributed. Since the maximum integer representation of a data entry in the Census database

is 15, the sensitivity is $\Delta = 15$, and hence, the proposed mechanism needs to fingerprint $K = \lfloor \log_2 15 \rfloor + 1 = 4$ least significant bits of each data entry.

Post-processing. After fingerprinting a database ($\mathbf{R}$), some entries may have integer representations that are outside the domain of the original database. For example, in the Nursery school database, the maximum integer is 4, i.e., "100", after fingerprinting, it may become 5, i.e., "101", which is not in the original database domain. Thus, we also need to post-process the resulting database ($\mathcal{M}(\mathbf{R})$) to eliminate entries that are not in the original domain. Otherwise, the database recipient can understand that these entries are changed due to fingerprinting. Due to the post-processing immunity property of differential privacy, there is no privacy degradation in this step. Even though the post-processing may alter some fingerprinted entries, it has negligible impact on the fingerprint robustness, because it only changes a small fraction of fingerprinted entries, and in the fingerprint extraction phase, we determine the value of each bit in the fingerprint by counting how many times it has been extracted as 1 or 0 followed by majority voting, i.e., each bit of the fingerprint is recovered by the majority voting on the positions marked by this fingerprint bit (i.e., Line 12 in Algorithm 2. Generally, post-processing steps are able to make a fingerprinted database meet the domain requirements so as to achieve better utility in downstream applications. For example, post-processing steps can let a fingerprinted database preserve the column- and row-wise data correlations and the covariance matrix of the database [26], which are frequently utilized to establish predictive models, e.g., regression and probability fitting.

Baseline Methods. We compare our mechanism (that simultaneously achieves privacy and liability guarantees) with six baselines summarized in Table III. In particular, baselines (i), (ii), and (iii) are naïve two-step approaches. Baseline (iv), discussed in Section II, is a one-step solution that brings together data sanitization and fingerprinting. Baselines (v) and (vi) achieve data perturbation and fingerprinting only, respectively. In baselines (i) and (v), data perturbation is achieved via local DP. In baseline (ii), data synthesis is obtained using DPSyn [9], [33], which generates differentially-private version of given databases by clustering similar attributes, and then perturbing the cell counts of the joint histograms for each cluster. The fingerprinting scheme used in all baselines (i), (ii), (iii), and (vi) is the database fingerprinting scheme developed in [35].

baseline (i)	data perturbation followed by fingerprinting	two-step
baseline (ii)	data synthesis followed by fingerprinting	two-step
baseline (iii)	k-anonymity-based fingerprinting	two-step
baseline (iv)	privacy-protection fingerprinting via Gaussian noise [20]	one-step
baseline (v)	data perturbation only via local differential privacy	no liability
baseline (vi)	fingerprinting only via mechanism developed in [35]	no privacy

TABLE III. COMPARING BASELINES CONSIDERED IN THE EXPERIMENTS. NOTE THAT BASELINES (I)-(IV) PROVIDE BOTH PRIVACY AND LIABILITY GUARANTEES DURING DATABASE SHARING. BASELINE (V) ONLY PROVIDES PRIVACY GUARANTEE (I.E., NO FINGERPRINT ROBUSTNESS), AND BASELINE (VI) ONLY PROVIDES FINGERPRINT ROBUSTNESS (I.E., NO PRIVACY GUARANTEE).

Experiment Outline. To show the performance of our proposed mechanism, we conduct extensive experiments focusing on fingerprint robustness and database utility. This is because we cannot directly compare the privacy guarantees with the baselines, as the privacy definitions vary for different baselines; our mechanism uses entry-level DP, whereas baseline (i) and (v), (iii), and (ii) and (iv), respectively, adopt local DP, k-anonymity, and conventional centralized DP. To enhance readability, we lists the considered experiments as follows.

In Section VII-B1, we compare the fingerprint robustness with all baselines (except for (v)) under the following scenarios: baseline (i) changes the same amount of data with our mechanism via LDP perturbation using the same ϵ with us and fingerprinting via [35]; baseline (vi) directly changes the same amount of data with our mechanism; baselines (ii) and (iv) use the same ϵ values with us; and baseline (iii) adopts 2-anonymity. We do not compare with baseline (v) because it does not provide fingerprint robustness. We cannot require baselines (ii), (iii), and (iv) change the same amount of data entries with us because (ii) adopts DPSyn to synthesize a completely new database using the probabilistic generative model, (iii) generalizes a significant amount of data entries to achieve 2-anonymity, and (v) can change all data using continuous Gaussian noise.

In Section VII-B2, we compare database utilities achieved by all methods when: baselines (i), (ii), (iv), and (v) adopts the same ϵ values with us; baseline (iii) adopts 2-anonymity; and baseline (vi) changes the same amount of data entries with our mechanism. Note that same ϵ does not lead to the same privacy guarantee due to different privacy definitions (as discussed before).

In section VII-C, we study the cumulative privacy loss when our mechanism is repeatedly applied to share databases with different SPs. We do not compare with other baselines, because our mechanism is the only one that applies Advanced Composition Theorem via SVT and the others just apply simple (linear) composition, so our mechanism is guaranteed to reduce cumulative privacy loss by an order of $\mathcal{O}(\sqrt{C})$ if the database is shared C times [16].

B. Evaluations for One-time Sharing

We first consider that the database owner only releases the database with one SP. Thus, only Algorithm 1 is invoked.

1) Fingerprint Robustness: Among common attacks against database fingerprinting mechanisms (i.e., random flipping attack, subset attack, and superset attack [35]), random flipping attack is shown to be the most powerful one [35], [50]. This is because the flipped data entries might create a fingerprint pattern that misleads the database owner during the fingerprint extraction phase [35], [51]. Thus, we investigate the fingerprint robustness of the proposed mechanism and the baselines against this attack. In Section VIII, we discuss how to make the proposed mechanism robust against more sophisticated correlation attacks [26]. In particular, in favor of the malicious SP, we let the malicious SP randomly flip 50% of the bit positions in its received copy of the database. Then, we measure the fingerprint robustness using the number of bit matches between the malicious SP's fingerprint and the one extracted from $\overline{\mathbf{R}}$ (the compromised database).

As per the experiment outline discussed in Section VII-A, for the Nursery database we select ϵ from $\{1, 2, \cdots, 7\}$. In Figure 4, we scatter the values of accuracy and number

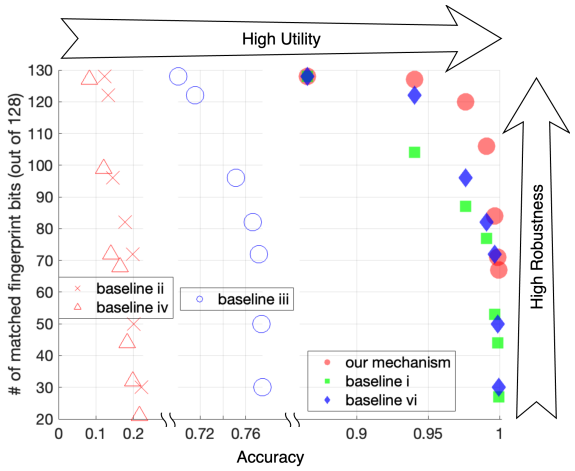


Fig. 4. Robustness comparison of fingerprinted Nursery databases.

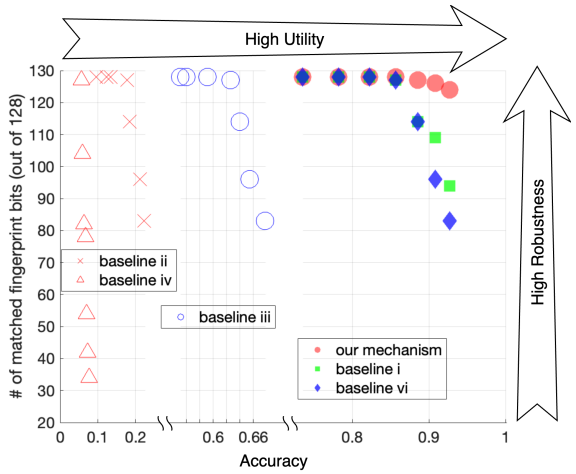


Fig. 5. Robustness comparison of fingerprinted Census databases.

of matched fingerprint bits obtained by all methods. Note that higher accuracy suggests higher utility of the obtained databases, and higher number of matched bits suggest higher robustness against 50% random flipping. Clearly, the databases obtained by our mechanism (represented by red dots) achieve the highest robustness given the same database accuracy and the highest accuracy given the same robustness. In particular, we outperform baseline (i), because the inserted noises (marks) generated by our mechanism serve the purposes of privacy protection and fingerprinting simultaneously, whereas (i) inserts noises twice to protect privacy and perform fingerprinting separately. We achieve higher robustness than baseline (vi), as (vi) only fingerprints one attribute for each selected row, and our mechanism can fingerprint multiple attributes. Baseline (iii) leads to lower database accuracy, because to achieve 2-anonymity, it needs to change all entries in some columns due to attributes generalization. Baselines (ii) and (iv) result in the lowest database accuracy, because they synthesize a new database and add continuous Gaussian noise to the original database, respectively. It is noteworthy that although baseline (iv) is a one-step solution like ours, it has the lowest robustness, because it needs to use learning algorithms to fit the inserted Gaussian noise, re-calculate the corresponding variance, and then recover the inserted fingerprints. Thus, when a large portion of data entries have been compromised, the obtained variance is highly inaccurate, so does the recovered fingerprints. For example, when $\epsilon \geq 5$, baseline (iv) can only achieve less than 50 fingerprint bit matches (out of 128), which suggests that the malicious SP can avoid being accused. This has been empirically validate in [26]: as long as a malicious SP can compromise more than half of the fingerprint bits (e.g., achieving less than 64 matches out of 128), the database owner will accuse another innocent SP with large probability. In contrast, when $\epsilon \geq 5$, our mechanism can still achieve more than 64 bit matches, which suggests that the malicious SP will end up being uniquely identifiable.

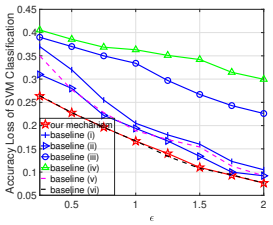
For the Census database, we select ϵ from $\{6, 7, \cdots, 12\}$ to achieve high accuracy for all the methods. The comparison of fingerprint robustness is shown in Figure 5. Similar to the Nursery database, our mechanism also outperforms all baselines in terms of database accuracy and fingerprint robustness.

2) Utility of the Shared Database: To show the utility guarantees of the proposed entry-level DP fingerprint mechanism, we conduct the comparison by considering specific applications, where we use fingerprinted databases (ours and the baselines) to do linear SVM classification and principal component analysis (PCA). Please refer to [21] for the experiments considering task-independent comparison, e.g., change of variance of attributes and the accuracy of SQL queries.

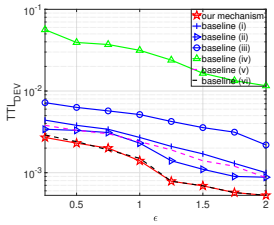
To perform classification, we adopt a multi-class support vector machine (SVM) classifier and use 65% of data records for training and the rest for testing. We evaluate the utility of various fingerprinted databases by comparing the **fingerprinted testing accuracy** (i.e., SVM classifier trained on fingerprinted training data and then tested on the original testing data) with the **original testing accuracy** (i.e., SVM classifier trained on the original training data and then tested on the original testing data). Thus, the smaller the difference between fingerprinted testing accuracy and original testing accuracy (i.e., accuracy loss), the higher the utility.

The utility for PCA is defined using the total deviation, $\mathrm{TTL}_{\mathrm{DEV}} = \sum_{i=1}^{T} |\lambda_i - \widetilde{\mathbf{v}}_i^T \mathbf{C} \widetilde{\mathbf{v}}_i|$, here T is the number of attributes (T is 8 and 13 for Nursery and Census databases, respectively), $\mathbf{C}$ is the empirical covariance matrix of the original (non-fingerprinted) database, λ_i values are the eigenvalues of $\mathbf{C}$, and $\widetilde{\mathbf{v}}_i$ vectors are the eigenvectors of the empirical covariance matrix of fingerprinted database. $\mathrm{TTL}_{\mathrm{DEV}}$ measures the deviation of the variance (of the fingerprinted database) from λ_i in the direction of the ith component of $\mathbf{C}$. The smaller $\mathrm{TTL}_{\mathrm{DEV}}$ is, the higher the utility.

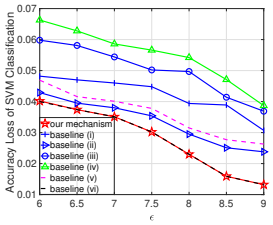
In Figure 6 (a) and (b) by varying ϵ from 0.25 to 2, we compare the utilities for SVM and PCA achieved by our mechanism and all baselines on the Nursery database. Clearly, our mechanism (red lines with pentagrams) achieves higher database utilities in both applications. Particularly, for baseline (i), a large portion of data entries are substituted with other values with high probability when ϵ is small, which leads to inaccurate task-specific applications, and fingerprinting after LDP perturbation further compromises the utility. Baseline (ii) can outperform baseline (i), because DPSyn generates the synthetic database by sampling from the noisy marginals of



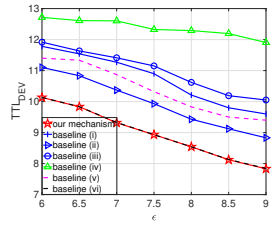
(a) SVM on Nursery Database.



(b) PCA on Nursery Database.



(a) SVM on Census Database.



(b) PCA on Census Database.

Fig. 6. Database utility in task-specific applications.

the clustered attributes and also involves techniques proposed in [43] to constrain the noisy marginals to be consistent with one another [9]. However, baseline (ii) still achieves lower utility compared with our mechanism, because the synthetic database is further distorted by fingerprint insertion. Baseline (iii) achieves the lowest utility in all two-step approaches, because it needs to modify a lager portion of data entries to achieve 2-anonymity. Thus, two-step approaches are highly suboptimal. Although baseline (iv) is a one-step approach, it has the worst utility among all the mechanisms, because it introduces Gaussian noises that completely overwhelm the original data. Baseline (v) leads to higher utility than baseline (i), because it does not involve fingerprinting. Although baseline (vi) has the similar performance with us, it cannot provide any privacy guarantees.

The same experiment results using the Census database when ϵ varies from 6 to 9 are shown in Figure 6 (c) and (d). As discussed in Section VII-B1, we consider high ϵ values to achieve high utility for all mechanisms. Clearly, our proposed mechanism still outperforms all baselines in terms of both accuracy loss of SVM and total deviation of PCA. Note that the accuracy loss on Census data is much less than the Nursery data, because Census data is highly unbalanced. The total deviation is much higher for the Census data, because it has more columns and large value of sensitivity, which lead to a large Frobenius norm of the empirical covariance matrix.[4]

C. Evaluations for Multiple Sharing

Next, we consider the scenario where at most 100 SPs query the entire database over time in a sequential order (i.e., $C = 100$). As discussed in Section VI, the database owner performs the noisy comparison ($||\mathcal{M}_i^c(\mathbf{R}) - \mathbf{R}||_{1,1} + \mu_i \geq \Gamma + \rho_i$, where Γ is the fingerprint density, μ_i and ρ_i are Laplace noises) to determine the proper internal IDs for the SPs to generate the fingerprinted databases. In the experiment, we set $\Gamma = (1/2 + 1/\sqrt{12})\Delta pNK$. The reason is that according to Corollary 1, the expected value of $||\mathcal{M}_i^c(\mathbf{R}) - \mathbf{R}||_{1,1}$ falls in $[0, \Delta pNT]$. Since we do not have any assumption on the database, and the pseudorandom number generator $\mathcal{U}$ generates each random number with equal probability, we approximately model $||\mathcal{M}_i^c(\mathbf{R}) - \mathbf{R}||_{1,1}$ as a uniformly distributed random variable in the range of $[0, \Delta pNT]$. Then, its mean and standard deviation are $\Delta pNT/2$ and $\Delta pNK/\sqrt{12}$, respectively.

Moreover, we consider the cumulative privacy loss as $\epsilon_0 = 40$ and $\delta_0 = 2 * 10^{-3}$. If $\epsilon_0 < 40$ and the database owner still wants to generate fingerprinted databases with the identical privacy and fingerprint robustness guarantees as when $\epsilon_0 = 40$, it will end up sharing its database with fewer number of SPs. To achieve a decent database utility, we set the privacy budget to generate the entry-level differentially-private fingerprinted database as $\epsilon = 0.5$. Then, by solving the privacy budget allocation problem (Section VI-B) numerically, we have $\epsilon_2 + \epsilon_3 = 0.002$ approximately.

We first investigate the impact of privacy allocation between ϵ_2 and ϵ_3 on the total number of trials to determine the proper internal IDs for all 100 SPs. We take the Nursery database as an example, vary the ratio between ϵ_2 and ϵ_3 from $9:1$ to $1:1$, and show the results in Table IV. We observe that as the difference between ϵ_2 and ϵ_3 decreases (their ratio decreases), Algorithm 4 terminates with fewer internal ID generation trials. Especially, when $\epsilon_2 : \epsilon_3 = 9:1$, 81 (181 instead of 100) additional trials are made, whereas, when $\epsilon_2 : \epsilon_3 = 1:1$, only 56 (156 instead of 100) additional trials are made. Since the cumulative privacy loss is identical for the different total number of trials reported in Table IV, this suggests that when $\epsilon_2 = \epsilon_3$, the internal ID generation efficiency is higher from the perspective of the data recipients (i.e., the probability that the database owner can generate a proper internal ID for an SP in 1 or 2 trials increases). This finding validates our suggestion of equally dividing the privacy budget between ϵ_2 and ϵ_3 to reduce the number of trials to generate proper internal IDs. Besides, we also would like to highlight that by adopting SVT, we significantly reduce the cumulative privacy loss, because otherwise, the privacy loss will be $(\epsilon + \epsilon_2 + \epsilon_3) \times (\text{Total No. of trials})$. For instance, when Total No. of trials $= 181$, the privacy would be $\epsilon = (0.5 + 0.02) \times 181 = 90.862$ without SVT (compared to a cumulative privacy loss of 40 with SVT).

$\epsilon_2 : \epsilon_3$	9 : 1	7 : 1	5 : 1	3 : 1	1 : 1
No. of trials	181	177	173	165	156

TABLE IV. IMPACT OF THE RATIO BETWEEN ϵ_2 AND ϵ_3 ON THE TOTAL NUMBER OF INTERNAL ID GENERATION TRIALS FOR 100 SPS.

VIII. DISCUSSION

Our work is a first step in uniting provable privacy guarantee and database fingerprinting. We believe that it will draw attention to other challenges and urgent research problems, which we plan to investigate in the future.

Mitigation of correlation attacks. Ji et al. [26] have developed a mitigation technique to alleviate the correlation attacks against database fingerprinting. Their technique modifies a fingerprinted database (via optimal transport technique) to make sure that it has similar column- and row-wise joint distributions with the original database. Since their technique only changes

[4] We do not fine-tune the training parameters in SVM and PCA algorithms, because it is out of the scope of this paper.

the non-fingerprinted data entries and it can be applied as a post-processing step after any fingerprinting mechanism, it can also be utilized following our mechanism to defend against the correlation attacks. In case of such an integration, our privacy guarantee will still hold because of the immunity property of differential privacy for post-processing [16].

Defending against collusion attacks. Another widely studied threat is the collusion attack where multiple malicious SPs ally together to generate a pirated database from their unique fingerprinted copies with the hope that none of them will be traced back. Several works have proposed collusion-resistant fingerprinting mechanisms in the literature [7], [6], [48], [42]. To develop a entry-level DP and collusion-resistant fingerprinting mechanism, one solution is to replace the fingerprint generation step (i.e., Line 3 of Algorithm 1) with the Boneh-Shaw (BS) codes [6] and decide p (the probability of changing one insignificant bit of an entry) based on ϵ and the number of 1's in the BS codeword. We will explore this extension in future work.

Two-step solutions versus our mechanism. Making the two-step solutions outperform our mechanism is still an open problem. Since currently, our mechanism treats each attribute equally sensitive. One potential approach is to take advantage of the semantic meaning of the attributes and then inject varying amounts of noise and insert different density of fingerprints to various portions of the database based on their sensitive level (e.g., some attributes like salary or health conditions may be more sensitive or private than others). However, this two-step approach will require domain experts on the database and can involve extra data analysis before it is subject to privacy protection and fingerprinting.

IX. Conclusions

In this paper, we have proposed a novel mechanism that unites provable privacy and database fingerprinting for sharing relational databases. We first devised a bit-level random response scheme to achieve ϵ-entry-level DP guarantee for the entire database, and then developed a concrete entry-level DP database fingerprinting mechanism on top of it. We have also provided the closed form expressions to characterize the connections between database utility, privacy protection, and fingerprint robustness. Finally, we developed a SVT-based solution to share entry-level DP fingerprinted databases with multiple recipients, and at the same time, control the cumulative privacy loss. Experimental results on two real relational databases show that we can achieve higher fingerprint robustness than a state-of-the-art database fingerprinting mechanism and achieve higher database utility than other baselines.

Acknowledgment

For the research reported in this publication, Erman Ayday was partly supported by the National Library of Medicine of the National Institutes of Health under Award Number R01LM013429 and by the National Science Foundation under grant numbers 2141622, 2050410, 2200255, and OAC-2112606. Pan Li was partially supported by the National Science Foundation under Grants EEC-2133630 and CNS-2125460.

References

[1] Rakesh Agrawal, Peter J Haas, and Jerry Kiernan. Watermarking relational data: framework, algorithms and analysis. *The VLDB journal*, 12(2):157–169, 2003.

[2] Arthur Asuncion and David Newman. Uci machine learning repository, 2007.

[3] Raef Bassily, Kobbi Nissim, Uri Stemmer, and Abhradeep Thakurta. Practical locally private heavy hitters. *Journal of Machine Learning Research*, 21(16):1–42, 2020.

[4] Elisa Bertino, Beng Chin Ooi, Yanjiang Yang, and Robert H Deng. Privacy and ownership preserving of outsourced medical data. In *21st International Conference on Data Engineering (ICDE'05)*, pages 521–532. IEEE, 2005.

[5] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. Differentially private data analysis of social networks via restricted sensitivity. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 87–96, 2013.

[6] Dan Boneh and James Shaw. Collusion-secure fingerprinting for digital data. In *Annual International Cryptology Conference*, pages 452–465. Springer, 1995.

[7] Dan Boneh and James Shaw. Collusion-secure fingerprinting for digital data. *IEEE Transactions on Information Theory*, 44(5):1897–1905, 1998.

[8] Dan Boneh and Victor Shoup. A graduate course in applied cryptography. *Recuperado de https://crypto. stanford. edu/˜ dabo/cryptobook/BonehShoup_0_4. pdf*, 2017.

[9] Claire Mckay Bowen and Joshua Snoke. Comparative study of differentially private synthetic data algorithms from the nist pscr differential privacy synthetic data challenge. *Journal of Privacy and Confidentiality*, 11:1, 2021.

[10] Thach V Bui, Binh Q Nguyen, Thuc D Nguyen, Noboru Sonehara, and Isao Echizen. Robust fingerprinting codes for database. In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 167–176. Springer, 2013.

[11] Thee Chanyaswad, Alex Dytso, H Vincent Poor, and Prateek Mittal. Mvg mechanism: Differential privacy under matrix-valued query. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 230–246, 2018.

[12] Kamalika Chaudhuri and Nina Mishra. When random sampling preserves privacy. In *Annual International Cryptology Conference*, pages 198–213. Springer, 2006.

[13] Edgar F Codd. A relational model of data for large shared data banks. In *Software pioneers*, pages 263–294. Springer, 2002.

[14] Scott Craver, Nasir Memon, B-L Yeo, and Minerva M Yeung. Resolving rightful ownerships with invisible watermarking techniques: Limitations, attacks, and implications. *IEEE Journal on selected Areas in communications*, 16(4):573–586, 1998.

[15] Cynthia Dwork and Jing Lei. Differential privacy and robust statistics. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 371–380, 2009.

[16] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

[17] Sébastien Gambs, Julien Lolive, and Jean-Marc Robert. Entwining sanitization and personalization on databases. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 207–219, 2018.

[18] Srivatsava Ranjit Ganta, Shiva Prasad Kasiviswanathan, and Adam Smith. Composition attacks and auxiliary information in data privacy. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 265–273, 2008.

[19] Ronald L Graham, Donald E Knuth, Oren Patashnik, and Stanley Liu. Concrete mathematics: a foundation for computer science. *Computers in Physics*, 3(5):106–107, 1989.

[20] Yun Hu, Aiqun Hu, Chunguo Li, Peng Li, and Chunyu Zhang. Towards a privacy protection-capable noise fingerprinting for numerically aggregated data. *Computers & Security*, page 102755, 2022.

[21] Tianxi Ji, Erman Ayday, Emre Yilmaz, and Pan Li. Differentially-private

fingerprinting of relational databases. *arXiv preprint arXiv:2109.02768*, 2021.

[22] Tianxi Ji, Erman Ayday, Emre Yilmaz, and Pan Li. Robust fingerprinting of genomic databases. In *30th International Conference on Intelligent Systems for Molecular Biology*, ISMB'21, Oxford, England, 2021. Oxford University Press.

[23] Tianxi Ji, Erman Ayday, Emre Yilmaz, and Pan Li. Towards robust fingerprinting of relational databases by mitigating correlation attacks. *IEEE Transactions on Dependable and Secure Computing*, pages 1–15, 2022.

[24] Tianxi Ji, Pan Li, Emre Yilmaz, Erman Ayday, Yanfang Ye, and Jinyuan Sun. Differentially private binary-and matrix-valued data query: an xor mechanism. *Proceedings of the VLDB Endowment*, 14(5):849–862, 2021.

[25] Tianxi Ji, Changqing Luo, Yifan Guo, Qianlong Wang, Lixing Yu, and Pan Li. Community detection in online social networks: a differentially private and parsimonious approach. *IEEE transactions on computational social systems*, 7(1):151–163, 2020.

[26] Tianxi Ji, Emre Yilmaz, Erman Ayday, and Pan Li. The curse of correlations for robust fingerprinting of relational databases. In *24th International Symposium on Research in Attacks, Intrusions and Defenses*, RAID '21, page 412–427, New York, NY, USA, 2021. Association for Computing Machinery.

[27] Jinyuan Jia and Neil Zhenqiang Gong. {AttriGuard}: A practical defense against attribute inference attacks via adversarial machine learning. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 513–529, 2018.

[28] Georgios Kellaris and Stavros Papadopoulos. Practical differential privacy via grouping and smoothing. *Proceedings of the VLDB Endowment*, 6(5):301–312, 2013.

[29] Peter Kieseberg, Sebastian Schrittwieser, Martin Mulazzani, Isao Echizen, and Edgar Weippl. An algorithm for collusion-resistant anonymization and fingerprinting of sensitive microdata. *Electronic Markets*, 24(2):113–124, 2014.

[30] Daniel Kifer and Ashwin Machanavajjhala. No free lunch in data privacy. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 193–204, 2011.

[31] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 106–115. IEEE, 2007.

[32] Ninghui Li, Wahbeh Qardaji, and Dong Su. On sampling, anonymization, and differential privacy or, k-anonymization meets differential privacy. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, pages 32–33, 2012.

[33] Ninghui Li, Zhikun Zhang, and Tianhao Wang. Dpsyn: Experiences in the nist differential privacy data synthesis challenges. *Journal of Privacy and Confidentiality*, 11:2, 2021.

[34] Yingjiu Li, Vipin Swarup, and Sushil Jajodia. Constructing a virtual primary key for fingerprinting relational data. In *Proceedings of the 3rd ACM workshop on Digital rights management*, pages 133–141, 2003.

[35] Yingjiu Li, Vipin Swarup, and Sushil Jajodia. Fingerprinting relational databases: Schemes and specialties. *IEEE Transactions on Dependable and Secure Computing*, 2(1):34–45, 2005.

[36] Zhen Lin, Michael Hewett, and Russ B Altman. Using binning to maintain confidentiality of medical data. In *Proceedings of the AMIA Symposium*, page 454. American Medical Informatics Association, 2002.

[37] Changchang Liu, Supriyo Chakraborty, and Prateek Mittal. Dependence makes you vulnberable: Differential privacy under dependent tuples. In *NDSS*, volume 16, pages 21–24, 2016.

[38] Siyuan Liu, Shuhong Wang, Robert H Deng, and Weizhong Shao. A block oriented fingerprinting scheme in relational database. In *International conference on information security and cryptology*, pages 455–466. Springer, 2004.

[39] Edward Lui and Rafael Pass. Outlier privacy. In *Theory of Cryptography Conference*, pages 277–305. Springer, 2015.

[40] Min Lyu, Dong Su, and Ninghui Li. Understanding the sparse vector technique for differential privacy. *Proceedings of the VLDB Endowment*, 10(6), 2017.

[41] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3–es, 2007.

[42] Birgit Pfitzmann and Michael Waidner. Asymmetric fingerprinting for larger collusions. In *Proceedings of the 4th ACM conference on Computer and communications security*, pages 151–160, 1997.

[43] Wahbeh Qardaji, Weining Yang, and Ninghui Li. Priview: practical differentially private release of marginal contingency tables. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1435–1446, 2014.

[44] Vibhor Rastogi, Dan Suciu, and Sungho Hong. The boundary between privacy and utility in data publishing. In *Proceedings of the 33rd international conference on Very large data bases*, pages 531–542. Citeseer, 2007.

[45] Sebastian Schrittwieser, Peter Kieseberg, Isao Echizen, Sven Wohlgemuth, Noboru Sonehara, and Edgar Weippl. An algorithm for k-anonymity-based fingerprinting. In *International Workshop on Digital Watermarking*, pages 439–452. Springer, 2011.

[46] Radu Sion, Mikhail Atallah, and Sunil Prabhakar. Rights protection for relational data. *IEEE transactions on knowledge and data engineering*, 16(12):1509–1525, 2004.

[47] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[48] Yacov Yacobi. Improved boneh-shaw content fingerprinting. In *Cryptographers' Track at the RSA Conference*, pages 378–391. Springer, 2001.

[49] Bin Yang, Issei Sato, and Hiroshi Nakagawa. Bayesian differential privacy on correlated data. In *Proceedings of the 2015 ACM SIGMOD international conference on Management of Data*, pages 747–762, 2015.

[50] Emre Yilmaz and Erman Ayday. Collusion-resilient probabilistic fingerprinting scheme for correlated data. *arXiv preprint arXiv:2001.09555*, 2020.

[51] Emre Yilmaz, Tianxi Ji, Erman Ayday, and Pan Li. Genomic data sharing under dependent local differential privacy. In *Proceedings of the Twelveth ACM Conference on Data and Application Security and Privacy*, pages 77–88, 2022.

APPENDIX A
PROOF OF PROPOSITION 2

Proof: The fingerprinting mechanism only changes the last K bits of selected data entries, thus, we have

$$
\begin{aligned}
&\mathbb{E}\Big[\Big|\widetilde{\mathbf{r}_i[t]}-\mathbf{r}_i[t]\Big|\Big]\\
=&\mathbb{E}\Big[\Big|\sum_{k=1}^{K}\widetilde{\mathbf{r}_i[t,k]}2^{\widetilde{\mathbf{r}_i[t,k]}-1}-\mathbf{r}_i[t,k]2^{\mathbf{r}_i[t,k]-1}\Big|\Big]\\
=&\mathbb{E}\Big[\Big|\sum_{k=1}^{K}(\mathbf{r}_i[t,k]\oplus B)2^{\mathbf{r}_i[t,k]\oplus B-1}-\mathbf{r}_i[t,k]2^{\mathbf{r}_i[t,k]-1}\Big|\Big]\\
=&\mathbb{E}\Big[\Big|\textstyle\sum_{k=1}^{K}(\mathbf{r}_i[t,k]+B-2\mathbf{r}_i[t,k]B)2^{\mathbf{r}_i[t,k]+B-2\mathbf{r}_i[t,k]B-1}-\mathbf{r}_i[t,k]2^{\mathbf{r}_i[t,k]-1}\Big|\Big]\\
=&\Big|\sum_{k=1}^{K}\Big((1-\mathbf{r}_i[t,k])2^{-\mathbf{r}_i[t,k]}-\mathbf{r}_i[t,k]2^{\mathbf{r}_i[t,k]-1}\Big)\Big|p\\
=&\Big|\sum_{k=1}^{K}\Big(\overline{\mathbf{r}_i[t,k]}2^{\overline{\mathbf{r}_i[t,k]}-1}-\mathbf{r}_i[t,k]2^{\mathbf{r}_i[t,k]-1}\Big)\Big|p.
\end{aligned}
$$

Since $\sum_{k=1}^{K}\overline{\mathbf{r}_i[t,k]}2^{\overline{\mathbf{r}_i[t,k]}-1}$ is the decimal representation of the complement of the last K bits of $\mathbf{r}_i$, and according to Definition 3, $\sum_{k=1}^{K}\left(\overline{\mathbf{r}_i[t,k]}2^{\overline{\mathbf{r}_i[t,k]}-1}-\mathbf{r}_i[t,k]2^{\mathbf{r}_i[t,k]-1}\right)$ falls

in the range of $[-\Delta, \Delta]$, so its absolute value falls in $[0, \Delta]$, which completes the proof. ■

APPENDIX B
PROOF OF PROPOSITION 3

Proof: We let binary vector $\tau_t \in \{0,1\}^K$ (or $\tau_z \in \{0,1\}^K$) represent the K mark bits embedded to entries of the tth (or zth) attribute. $||\cdot||_0$ denotes the l_0 norm, which counts the number of nonzero entries in a vector. $\pi_{[2]}$ and $\omega_{[2]}$ are the binary representations of π and ω, respectively. Then, we have

$$\begin{aligned}
&\Pr(\widetilde{\mathbf{R}[t]} = \pi, \widetilde{\mathbf{R}[z]} = \omega) \\
=& \sum_{\tau_t \in \{0,1\}^K} \sum_{\tau_z \in \{0,1\}^K} \Pr\Big(\mathbf{R}[t] = \pi_{[2]} \oplus \tau_t, \mathbf{R}[z] = \omega_{[2]} \oplus \tau_z\Big) \\
&\times p^{||\tau_t||_0}(1-p)^{K-||\tau_t||_0} p^{||\tau_z||_0}(1-p)^{K-||\tau_z||_0} \\
=& \Pr\Big(\mathbf{R}[t] = \pi_{[2]}, \mathbf{R}[z] = \omega_{[2]}\Big)(1-p)^{2K} \\
&+ \sum_{\tau_t \in \{0,1\}^K/\mathbf{0}} \sum_{\tau_z \in \{0,1\}^K/\mathbf{0}} \Pr\Big(\mathbf{R}[t] = \pi_{[2]} \oplus \tau_t, \mathbf{R}[z] = \omega_{[2]} \oplus \tau_z\Big) \\
&\times p^{||\tau_t||_0}(1-p)^{K-||\tau_t||_0} p^{||\tau_z||_0}(1-p)^{K-||\tau_z||_0}.
\end{aligned}$$

By denoting the second summand in the above equation as $\diamondsuit$, we have

$$\begin{aligned}
\diamondsuit &\le \max_{\Pr}(\mathbf{R}[t], \mathbf{R}[z]) \times \Bigg(\sum_{\tau_t \in \{0,1\}^K/\mathbf{0}} \sum_{\tau_z \in \{0,1\}^K/\mathbf{0}} \\
&\quad p^{||\tau_t||_0}(1-p)^{K-||\tau_t||_0} p^{||\tau_z||_0}(1-p)^{K-||\tau_z||_0} \Bigg) \\
&= \max_{\Pr}(\mathbf{R}[t], \mathbf{R}[z]) \times \\
&\quad \times \Bigg(\sum_{\tau_t \in \{0,1\}^K/\mathbf{0}} p^{||\tau_t||_0}(1-p)^{K-||\tau_t||_0} \Bigg) \\
&\quad \times \Bigg(\sum_{\tau_z \in \{0,1\}^K/\mathbf{0}} p^{||\tau_z||_0}(1-p)^{K-||\tau_z||_0} \Bigg) \\
&= \max_{\Pr}(\mathbf{R}[t], \mathbf{R}[z]) \Big(1 - (1-p)^K\Big)^2.
\end{aligned}$$

Similarly, we also have $\diamondsuit \ge \Pr_{\min}(\mathbf{R}[t], \mathbf{R}[z]) \Big(1 - (1-p)^K\Big)^2$, thus, the proof is completed. ■

APPENDIX C
ANALYSIS OF $P_{\text{rbst\_rnd}}$ IN RANDOM BIT FLIPPING ATTACK

Note that a rational database owner will not change more than 50% of the bit positions in a database, because it will significantly compromise the database utility, and a malicious SP can flip the bits back and then launch an attack.

First, we show how to determine D (number of bit matches with the malicious SP's fingerprint) given C (number of times a database can be shared), and L (length of fingerprinting string). If the database owner shares its database with C different SPs. To make the extracted fingerprint have the most bit matches with the malicious SP, it requires that the probability of having more than D bit matches is higher than $1/C$, i.e., $\binom{L}{D}(\frac{1}{2})^D(\frac{1}{2})^{L-D} \ge \frac{1}{C}$, which can be solved analytically.

One can easily check that the closed form expression of $P_{\text{rbst\_rnd}}$ in terms of p is

$$P_{\text{rbst\_rnd}} = \sum_{m=1}^{NKT} \Big(\sum_{w_l \in \mathcal{W}, l \in [1,L]} \sum_{\mathcal{L}_D} \prod_{l \in \mathcal{L}_D} p_l (\tfrac{1}{L})^{w_l} \Big) \binom{NKT}{m} (2p)^m (1-2p)^{NKT-m},$$

which can be obtained by marginalizing all instances of malicious SP's fingerprint, all collections of D fingerprint bits of it, and the number of fingerprinted bits m. To show that higher p leads to more robustness against the random bit flipping attack, it is equivalent to show that $P_{\text{rbst\_rnd}}$ is monotonically increasing with p ($0 < p < 0.5$). To this end, we define function $f(m) = \sum_{w_l \in \mathcal{W}, l \in [1,L]} \sum_{\mathcal{L}_D} \prod_{l \in \mathcal{L}_D} p_l (\frac{1}{L})^{w_l}$ (m is embedded as a parameter of set $\mathcal{W}$, i.e., $\mathcal{W} = \{w_1, w_2, \cdots, w_L > 0 | \sum_{l=1}^{L} w_l = m\}$. Then, $P_{\text{rbst\_rnd}}$ represents the expected value of $f(m), m \sim \text{Binomial}(NKT, 2p)$. As a result, it is sufficient to show that $f(m)$ is monotonically increasing with m. First, we observe that p_l (see Section V-C1, $p_l = \sum_{q=0}^{\lfloor \frac{w_l}{2} \rfloor} \binom{w_l}{q} \gamma_{\text{rnd}}^q (1-\gamma_{\text{rnd}})^{w_l - q}$) is the cumulative distribution function of a binomial distribution function, which is monotonically increasing with w_l, thus the multiplication of all p_l's, i.e., $\prod_{l \in \mathcal{L}_D} p_l$ is increasing with $m = \sum_l w_l$. Second, it is easy to check that the carnality of $\mathcal{W}$ is $m! S(w, L)$, where $S(w, L)$ represents Stirling number of the second kind (i.e., the number of ways to partition a set of w objects into L non-empty subsets) [19]. Since $m!$ grows faster then L^{w_l} as m increases (in real-life applications, we have $m \gg L \gg \ln N$), we can conclude that $f(m)$ is monotonically increasing with m (the number of fingerprinted bit positions). When $0 < p < 0.5$, $P_{\text{rbst\_rnd}}$ can be characterized as the summation of monotonically increasing functions with respect to m and p, which suggests that the higher the value of p, the more robust of the proposed fingerprinting mechanism is against the random bit flipping attack.

APPENDIX D
ANALYSIS OF G IN CORRELATION ATTACK

As per proposition 3, $\Big| \Pr\Big(\widetilde{\mathbf{R}[t]} = \pi, \widetilde{\mathbf{R}[z]} = \omega\Big) - \Pr\Big(\mathbf{R}[t] = \pi, \mathbf{R}[z] = \omega\Big) \Big|$ falls in the range of $[0, \max\{|A|, |B|\}]$, where A and B, respectively, are

$$A = \Pr\Big(\mathbf{R}[t] = \pi, \mathbf{R}[z] = \omega\Big)\Big((1-p)^{2K} - 1\Big) + \Pr_{\min}(\mathbf{R}[t], \mathbf{R}[z])\Big(1 - (1-p)^K\Big)^2,$$
$$B = \Pr\Big(\mathbf{R}[t] = \pi, \mathbf{R}[z] = \omega\Big)\Big((1-p)^{2K} - 1\Big) + \Pr_{\max}(\mathbf{R}[t], \mathbf{R}[z])\Big(1 - (1-p)^K\Big)^2.$$

Without any assumption on the database, we consider each of the point τ (threshold) in $[0, \max\{|A|, |B|\}]$ has equal probability density [26]. Thus, $G = \frac{1 - \prod_{z \in [1,T], z \ne t} \prod_{\omega} \frac{\tau}{\max\{|A|, |B|\}}}{(1-(1-p)^K)\Pr(\mathbf{R}[t] = \pi)}$. Let $\lambda = 1 - (1-p)^K \in [0, 1 - (\frac{1}{2})^K]$, then, we have

$$A = \Pr\Big(\mathbf{R}[t] = \pi, \mathbf{R}[z] = \omega\Big)\Big((1-p)^{2K} + 1\Big)(-\lambda) + \Pr_{\min}(\mathbf{R}[t], \mathbf{R}[z])\lambda^2,$$
$$B = \Pr\Big(\mathbf{R}[t] = \pi, \mathbf{R}[z] = \omega\Big)\Big((1-p)^{2K} + 1\Big)(-\lambda) + \Pr_{\max}(\mathbf{R}[t], \mathbf{R}[z])\lambda^2.$$

Thus, $G = \frac{1 - (\frac{\tau}{\mathcal{O}(\lambda)})^{\sum_{z \in [1,T], z \ne t} k_z}}{\mathcal{O}(\lambda)}$, and k_z is the number of possible instances of attribute z. Since both the numerator and denominator increases with p, but the denominator grows with a much higher rate, G decreases as p increases.