Differentially Private High-Dimensional Data Publication via Sampling-Based Inference

Rui Chen[*]
Samsung Research America, USA
rui.chen1@samsung.com

Qian Xiao
National University of Singapore, Singapore
xiaoqian@nus.edu.sg

Yu Zhang
Hong Kong Baptist University, Hong Kong
yuzhang@comp.hkbu.edu.hk

Jianliang Xu
Hong Kong Baptist University, Hong Kong
xujl@comp.hkbu.edu.hk

ABSTRACT

Releasing high-dimensional data enables a wide spectrum of data mining tasks. Yet, individual privacy has been a major obstacle to data sharing. In this paper, we consider the problem of releasing high-dimensional data with differential privacy guarantees. We propose a novel solution to preserve the joint distribution of a high-dimensional dataset. We first develop a robust sampling-based framework to systematically explore the dependencies among all attributes and subsequently build a dependency graph. This framework is coupled with a generic threshold mechanism to significantly improve accuracy. We then identify a set of marginal tables from the dependency graph to approximate the joint distribution based on the solid inference foundation of the junction tree algorithm while minimizing the resultant error. We prove that selecting the optimal marginals with the goal of minimizing error is NP-hard and, thus, design an approximation algorithm using an integer programming relaxation and the constrained concave-convex procedure. Extensive experiments on real datasets demonstrate that our solution substantially outperforms the state-of-the-art competitors.

Categories and Subject Descriptors

K.4.1 [**COMPUTERS AND SOCIETY**]: Privacy

Keywords

Differential privacy, high-dimensional data, joint distribution, dependency graph, junction tree algorithm

1. INTRODUCTION

With the rapid development of computing technologies, high-dimensional data has become prevalent in many application domains. Releasing such data has been a prerequisite for many data mining tasks. However, individual privacy has been a major public concern in data sharing. In this paper, we consider the problem of releasing high-dimensional data under *differential privacy* [6], a privacy notion widely advocated due to its strong privacy guarantee. There have been a series of techniques proposed for differentially private low-dimensional data publication [1,2,6,9,12,17,18,24,25,27,30]. Unfortunately, when applied to high-dimensional data, all these techniques suffer from the *curse of dimensionality*, that is, they cannot achieve either reasonable scalability or desirable utility [21,29]. Special treatment is needed to overcome the challenges incurred by high dimensionality.

In addressing the challenges, one of the most promising ideas is to decompose high-dimensional data into a set of low-dimensional marginal tables $\mathcal{C}$, along with an inference mechanism that infers the joint data distribution from $\mathcal{C}$. This has been the rationale of the very recent study [29] in which *PrivBayes* is proposed to learn a set of low-dimensional conditional probabilities via a Bayesian network and approximate the joint distribution by the chain rule for Bayesian networks. While making substantial progress, it has two major limitations due to the constraint of differential privacy. First, one has to minimize probes over the underlying dataset; otherwise the result of a probe becomes largely inaccurate. This implies that one often has to compromise more sophisticated algorithms that require excessive access to the data. For this reason, *PrivBayes* employs a simple greedy algorithm to learn a Bayesian network, whose performance is sensitive to the randomly selected initial attribute, and limits the size of each attribute's parent set to be identical. Second, with the increasing number of attributes, the privacy budget used for determining each attribute's parent set decreases quickly, making the learned conditional probabilities unreliable. As a result, *PrivBayes* is still not able to adequately capture the characteristics of the underlying data in order to maximize data utility.

In practice, the independence properties exist in many high-dimensional datasets, which is the rationale behind probabilistic graphical models [15]. It is beneficial to systematically explore such (conditional) independences among attributes. However, conducting such an exploration under differential privacy requires non-trivial efforts. The key technical challenge is how to reliably learn all attributes' pairwise correlations using a limited privacy budget. For a dataset with d attributes, existing techniques require to divide the

[*]Part of this work was done while the author was with Hong Kong Baptist University

KDD'15, August 10-13, 2015, Sydney, NSW, Australia.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-3664-2/15/08 ...$15.00.
DOI: http://dx.doi.org/10.1145/2783258.2783379.

given privacy budget into $\binom{d}{2}$ portions, each being used for a pair of attributes. There is no doubt that none of these techniques can learn the correlations with reasonable accuracy. We address this challenge by proposing a sampling-based testing framework and a generic threshold mechanism. This design allows us to learn the pairwise correlations without splitting the privacy budget in proportion to $\binom{d}{2}$ and further enjoy the privacy budget amplification due to sampling.

With the learned correlations, how to develop an inference model to estimate the joint distribution with minimum error remains a second challenge. We decompose this task into two steps. First, we build a solid statistical inference foundation by employing the well-established junction tree algorithm. Unfortunately, the junction tree algorithm does not take into consideration the differential privacy constraint. Directly injecting noise into the marginals of the cliques returned by the junction tree algorithm usually leads to excessive noise. We then formulate an optimization problem for finding the optimal marginals with the minimum error and solve it by the constrained concave-convex procedure.

Contributions. Our key contribution is a novel sampling-based solution for publishing high-dimensional data under differential privacy, which features a solid statistical inference foundation. More specifically, we make the following contributions:

First, we design a sampling-based testing framework to systematically explore pairwise dependencies while satisfying differential privacy. This framework is made possible by a generic threshold mechanism, which is an extended version of the sparse vector technique [10] and the threshold query technique [16]. This threshold mechanism allows to use *multiple* thresholds and may substantially improve the accuracy of many algorithms.

Second, we propose to apply the junction tree algorithm to establish an inference mechanism for inferring the joint data distribution. How to generate differentially private marginals to feed the inference mechanism with minimum error is vital to the final accuracy. We prove that this optimization problem is NP-hard and propose an approximation algorithm using an integer programming relaxation and the constrained concave-convex procedure.

Third, we extend the mutual consistency technique in [21] to the general case where the noisy marginals are of different sizes and attributes may not be binary, and propose a simple yet effective thresholding strategy to mitigate the systematic bias due to rounding negative noisy counts to 0. We also show how to efficiently generate synthetic datasets from the noisy marginals using the junction tree representation.

We conduct extensive experiments over five standard real datasets for two different analysis tasks. We show that our solution not only significantly outperforms *PrivBayes* [29], the state-of-the-art technique for releasing high-dimensional data, but also achieves comparable, and sometimes better, accuracy to that of *PriView* [21], the state-of-the-art technique tailored to generating k-way marginals over high-dimensional data. Moreover, our solution works for both binary and non-binary data while *PriView* only works for binary data[1].

[1]To handle non-binary data, *PriView* requires adapting existing covering design algorithms to the case where different views may have different sizes [21]. To our best knowledge, there does not exist such an algorithm.

2. RELATED WORK

Most of the existing works on differentially private data publishing focus on low-dimensional data (e.g., marginal tables or histograms). Xiao et al. [24] propose to apply the wavelet transformation to an input histogram and add noise to wavelet coefficients. Hay et al. [12] exploit the consistency constraints that should hold over the noisy output to improve accuracy. These two methods are special cases of a more general matrix mechanism [17], which calculates the answers to a set of queries from another set of properly selected queries called query strategy. Yaroslavtsev et al. [26] further explore the idea of using a query strategy by adding non-uniform noise. Yuan et al. [27] introduce the low-rank mechanism for answering batch linear counting queries based on low-rank matrix approximation. Hardt et al. [9] present an algorithm based on multiplicative weights and the exponential mechanism. In another line of research, Xu et al. [25] propose to group a histogram's adjacent bins with close counts to trade for smaller Laplace noise. A similar idea is proposed by Acs et al. [1]. They design a hierarchical bisection algorithm to identify a good grouping scheme. Zhang et al. [30] indicate that global clustering achieves better utility than local grouping. Li et al. [18] propose to answer range queries by taking into consideration both the underlying data and the query set.

In general, the above methods cannot overcome the inherent challenges due to the curse of dimensionality. In spite of its wide applications, differentially private high-dimensional data publication has been relatively rarely studied. Barak et al. [2] show how to construct a synthetic database to preserve all low-dimensional marginals by adding noise to the Fourier domain. The problem in [2] is equivalent to publishing OLAP cubes, which is studied by Ding et al. [5]. They first compute a subset of cuboids and then generate the remaining cuboids from this subset. The main limitation of these two approaches is their exponential complexity in the dimensionality of the domain. Mohammed et al. [20] introduce probabilistic generalization to overcome the curse of dimensionality. However, with the increasing dimensionality, the benefit of generalization diminishes rapidly. Cormode et al. [4] solely consider the scalability aspect of the problem. They design a statistical process to compute a private summary without materializing the entire contingency table.

Very recently, Qardaji et al. [21] study how to generate accurate k-way marginals for a *binary* dataset. They propose *PriView* that uses covering design to select a set of low-dimensional marginals called *views* and then generates k-way marginals based on maximum entropy optimization. The work closest to ours is *PrivBayes* [29], which iteratively learns the parent sets of the attributes in a Bayesian network by applying the exponential mechanism with a surrogate function for mutual information. Compared with *PrivBayes*, our solution features a systematic exploration of attribute correlations and a series of new generic techniques, which together achieve substantially better performance. The connection between probabilistic inference and differential privacy is also studied in [23].

3. PROBLEM FORMULATION

3.1 Problem Statement

In this paper, we consider the following problem: *Given a dataset D with d attributes (either numerical or categori-*

cal) $\mathcal{A} = \{A_1, A_2, \cdots, A_d\}$*, we want to generate a synthetic dataset that accurately preserves the joint distribution of the tuples in D while satisfying differential privacy.*

We denote the value domain of an attribute A_i by Ω_i and its size (i.e., the number of distinct values in Ω_i) by $|\Omega_i|$[2]. Therefore, the entire output domain is defined by $\Omega_1 \times \Omega_2 \times \cdots \times \Omega_d$, whose size is $|\Omega_1| \times |\Omega_2| \times \cdots \times |\Omega_d|$. We focus on the case where $|\Omega_1| \times |\Omega_2| \times \cdots \times |\Omega_d|$ is too large to be handled by the existing low-dimensional data publishing techniques. We assume that the size of the dataset D (i.e., the number of tuples), denoted by $|D|$, is known. This is a common assumption under our definition of neighboring databases [6]. Note that the dataset D does not have to be a relational table. It can also be, for example, a set-valued dataset in which all attributes are binary.

Differential Privacy. Differential privacy is built on the notion of indistinguishability of two *neighboring* databases. We consider two databases D and D' to be neighbors if D can be obtained from D' by changing the value of exactly one tuple. Intuitively, differential privacy guarantees that any computational result from D and D' will be statistically indistinguishable. A formal definition is given below.

Definition 1. (ϵ-Differential Privacy [6]) A randomized algorithm $\mathcal{M}$ satisfies ϵ-differential privacy, if for any two neighboring databases D and D', and for any $O \subseteq Range(\mathcal{M})$,

$$P[\mathcal{M}(D) \in O] \leq \exp(\epsilon) \cdot P[\mathcal{M}(D') \in O],$$

where the probability is taken over $\mathcal{M}$'s randomness.

Differential privacy can be achieved by the *Laplace mechanism* [6], which injects properly calibrated Laplace noise into a function's output to mask the impact of any single tuple. The maximal impact of a tuple to the output of a function f is called its *sensitivity.* For any two neighboring databases D and D', the sensitivity of $f : D \to \mathbb{R}^d$ is defined as $\Delta f = \max_{D,D'} \|f(D) - f(D')\|_1$.

THEOREM 1. [6] *For any function* $f : D \to \mathbb{R}^d$*, the mechanism* $\mathcal{M}$*,*

$$\mathcal{M}(D) = f(D) + \left\langle \mathtt{Lap}_1\left(\frac{\Delta f}{\epsilon}\right), \ldots, \mathtt{Lap}_d\left(\frac{\Delta f}{\epsilon}\right) \right\rangle$$

gives ϵ*-differential privacy, where* $\mathtt{Lap}_i\left(\frac{\Delta f}{\epsilon}\right)$ *are i.i.d Laplace variables with scale parameter* $\frac{\Delta f}{\epsilon}$*.*

3.2 Junction Tree Algorithm

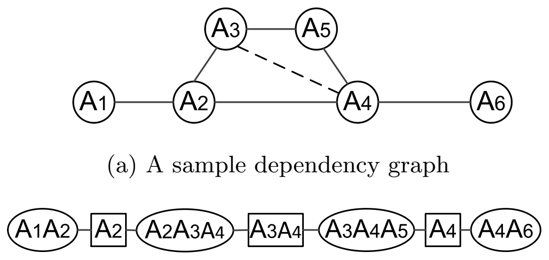
The key to overcoming the curse of dimensionality in our problem is to factorize the joint probability distribution into modular components based on conditional independences that exist in many real-world datasets. Probabilistic graphical models are an elegant tool for identifying such a modular structure. *Markov networks* are the most widely used graphical model based on *undirected* graphs. In our problem, we record the independences in terms of a *dependency graph*, which is essentially a Markov network in that the nodes represent the attributes in a dataset, and the edges correspond to the dependencies between the attributes [15].

The *junction tree algorithm* is a standard method to parameterize a Markov network so that the joint distribution and marginal distributions can be readily calculated. The general idea is to represent the joint distribution $P(\mathcal{A})$ of all attributes $\mathcal{A} = \{A_1, A_2, \cdots, A_d\}$ in a dataset as a function of the *marginals* of a set of maximal *cliques* and their *separators.* Let C_i be a clique in the junction tree T, and $S_{ij} = C_i \cap C_j$ be the separator between cliques C_i and C_j. The joint distribution can be calculated as follows:

$$P(\mathcal{A}) = \frac{\prod_{C_i \in T} P(C_i)}{\prod_{S_{ij} \in T} P(S_{ij})}, \tag{1}$$

where $P(C_i)$ and $P(S_{ij})$ are the marginal distributions of the cliques and the separators, respectively. This establishes the theoretical foundation of exactly inferring the joint distribution from a junction tree.

A junction tree can be constructed in two steps. The first step is triangulation. An undirected graph is said to be *triangulated* if and only if there is an edge between any two non-successive vertices in every cycle. A triangulated graph is guaranteed to have a junction tree. Then we can construct the junction tree in one of two basic ways: (1) based on variable elimination and (2) based on direct graph manipulation. In either way, the resultant junction tree satisfies the *running intersection property*: for each pair of vertices u and v, all vertices on the path between u and v contain the intersection $u \cap v$.



(a) A sample dependency graph

(b) A junction tree

Figure 1: A dependency graph and its junction tree

EXAMPLE 1. *Figure 1(a) gives the dependency graph G of a sample dataset with attributes* $\mathcal{A} = \{A_1, A_2, \cdots, A_6\}$*. Since the cycle* $A_2 - A_3 - A_5 - A_4 - A_2$ *has non-successive vertices that are not connected, we add the edge* (A_3, A_4) *to make G triangulated, as illustrated by the dashed line. Note that the triangulation procedure may not be unique. For example, adding the edge* (A_2, A_5) *also makes G triangulated. A junction tree of the triangulated graph is given in Figure 1(b), where cliques are represented by oval nodes and separators by rectangle nodes. Similarly, the junction tree of a triangulated graph is not unique.*

4. OUR SOLUTION

Broadly, our solution first systematically explores pairwise dependencies to construct the dependency graph and then differentially privately infers the joint distribution by applying the junction tree algorithm. More specifically, our solution is composed of the following four steps.

1. **Build the dependency graph.** The first step is to learn the *pairwise* correlations of all attributes under a sampling-based testing framework, from which the dependency graph is generated.
2. **Form attribute clusters.** We apply the junction tree algorithm to the dependency graph in order to

[2]Continuous attributes can be discretized to fit into our solution.

generate the set of cliques to form the inference foundation and further identify a collection of attribute clusters to derive all the cliques' noisy marginals with the minimum error.

3. **Generate noisy marginals.** For each attribute cluster, we generate a differentially private marginal table and enforce consistency constraints over all such marginals.

4. **Produce a synthetic dataset.** We make use of the noisy marginal tables and the inference model to efficiently generate a synthetic dataset.

Since only the first and third steps require access to the input dataset, we divide the total privacy budget ϵ into two portions with ϵ_1 being used for the first step and ϵ_2 for the third step. In the following, we will show that the first and third steps are ϵ_1- and ϵ_2-differentially private, respectively. Hence, by the composition property [6], our solution satisfies ϵ-differential privacy as a whole, where $\epsilon = \epsilon_1 + \epsilon_2$.

4.1 Constructing the Dependency Graph

To build the dependency graph, we propose a sampling-based dependency testing framework. The theoretical foundation of our framework is derived from *log-linear models* [3] that have been extensively studied in the statistical literature. The saturated log-linear model states that the joint distribution can be modeled as a summation of *effects* whose dimensionality ranges from 0 up to d (i.e., the correlations of pairs of attributes, triples, quadruples, and so forth). Yet, recent studies [13] have shown the diminishing return of maintaining higher order correlations, suggesting that pairwise correlations are most important to approximate the joint distribution. This result justifies why our framework focuses on learning pairwise correlations.

In the literature, the correlation between two attributes can be measured by several metrics, such as chi-squared test χ^2, mean-square contingency, Cramer's V ϕ_c and mutual information I, among others. In this paper, we choose mutual information due not only to its small sensitivity but also to its capability of capturing both linear and non-linear correlations. Given two attributes A_k and A_l, the *mutual information* $I(A_k, A_l)$ is defined as

$$I(A_k, A_l) = \sum_{i=1}^{|\Omega_k|} \sum_{j=1}^{|\Omega_l|} p_{ij} \log \frac{p_{ij}}{p_{i\cdot} p_{\cdot j}},$$

where p_{ij} is the fraction of the tuples whose $A_k = i$ (i.e., the ith value in Ω_k) and $A_l = j$, $p_{i\cdot} = \sum_j p_{ij}$ and $p_{\cdot j} = \sum_i p_{ij}$. As suggested by [14], we can consider that A_k and A_l are independent if $I(A_k, A_l) \le \theta_{kl}$ for some small $\theta_{kl} > 0$. Yet, how to set a reasonable θ_{kl} is unknown. In this paper, we perform a more formal analysis on the choice of θ_{kl} by establishing the connection between mutual information and Cramer's V ϕ_c. Mutual information can be well approximated as $I(A_k, A_l) \approx \frac{\min(|\Omega_k|-1, |\Omega_l|-1)\phi_c^2}{2}$. Since ϕ_c has a well-interpreted level of dependency (e.g., $\phi_c = 0.2$ stands for weak dependency), we design $\theta_{kl} = \sigma \min(|\Omega_k| - 1, |\Omega_l| - 1)$, where σ is a parameter controlled by the ϕ_c value representing the desired level of dependency (e.g., $\phi_c = 0.2$).

We present our differentially private sampling-based framework in Algorithm 1. Previous work [13] has indicated that the sample size required for learning correlations for a specified degree of accuracy can be independent of the dataset size. This observation is vital to allow us to enjoy the sampling property of differential privacy [19].

```
Algorithm 1 Generate Dependency Graph
Input: Dataset $D$ with attributes $\mathcal{A} = \{A_1, A_2, \cdots, A_d\}$
Input: Privacy parameter $\epsilon_1$
Output: Dependency graph $G$
 1: Initialize $G = (V, E)$ with $V = \{A_1, A_2, \cdots, A_d\}$ and $E = \emptyset$;
 2: Calculate sampling rate $\beta$;
 3: Generate $D_s$ by sampling tuples in $D$ with rate $\beta$;
 4: $\epsilon_a = \ln(e^{\epsilon_1} - 1 + \beta) - \ln \beta$;
 5: $\eta = \mathtt{Lap}\left(\frac{2\Delta I}{\epsilon_a}\right)$;
 6: for each attribute pair $(A_k, A_l)$ do
 7:     $\widetilde{I}(A_k, A_l) = I(A_k, A_l) + \mathtt{Lap}\left(\frac{2\Delta I}{\epsilon_a}\right)$;
 8:     $\widetilde{\theta}_{kl} = \theta_{kl} + \eta$;
 9:     if $\widetilde{I}(A_k, A_l) \ge \widetilde{\theta}_{kl}$ then
10:         Add edge $(A_k, A_l)$ in $G$;
11: return $G$;
```

Theorem 2. [19] *Let $\mathcal{M}$ be an ϵ-differentially private algorithm and $\mathcal{M}_\beta$ be another algorithm that first independently samples each tuple in its input dataset with probability β and then applies $\mathcal{M}$ to the sampled dataset. $\mathcal{M}_\beta$ satisfies $\ln(1 + \beta(e^\epsilon - 1))$-differential privacy.*

In Line 2, we calculate the sampling rate β by first determining a good sample size n_s. Our intuition is to select the sample size that minimizes the ratio of noise magnitude to the range of mutual information I on the sampled dataset, which is equivalent to identifying the best signal-to-noise ratio [29]. To quantify this size, we need to know the amplified privacy parameter ϵ_a due to sampling and the sensitivity of I. From Theorem 2, we have $\epsilon_a = \ln\left(e^{\epsilon_1} - 1 + \frac{n_s}{|D|}\right) - \ln \frac{n_s}{|D|}$. Next we give the sensitivity of I.

Theorem 3. [29]

$$\Delta I = \begin{cases} \frac{1}{n}\log n + \frac{n-1}{n}\log\frac{n}{n-1}, & \textit{if any attr. is binary} \\ \frac{2}{n}\log\frac{n+1}{2} + \frac{n-1}{n}\log\frac{n+1}{n-1}, & \textit{otherwise} \end{cases}$$

where n is the sampled dataset size.

Note that ΔI is insensitive to attributes' domain sizes. It only cares whether an attribute is binary. In Algorithm 1, if *all* attributes of the input dataset are binary, we use $\Delta I = \frac{1}{n}\log n + \frac{n-1}{n}\log\frac{n}{n-1}$; otherwise we use $\Delta I = \frac{2}{n}\log\frac{n+1}{2} + \frac{n-1}{n}\log\frac{n+1}{n-1}$. The reason of this choice will be made clear soon. Thus, the best sample size can be calculated as

$$n_s = \begin{cases} \underset{n}{\operatorname{argmin}} \dfrac{\frac{1}{n}\log n + \frac{n-1}{n}\log\frac{n}{n-1}}{\ln\left(e^{\epsilon_1} - 1 + \frac{n}{|D|}\right) - \ln\frac{n}{|D|}}, & \text{if all attr. are binary} \\ \underset{n}{\operatorname{argmin}} \dfrac{\frac{2}{n}\log\frac{n+1}{2} + \frac{n-1}{n}\log\frac{n+1}{n-1}}{\ln\left(e^{\epsilon_1} - 1 + \frac{n}{|D|}\right) - \ln\frac{n}{|D|}}, & \text{otherwise} \end{cases}$$

where $1 < n \le |D|$ stands for all possible sample sizes.

In Line 3, we sample the tuples in D by including each tuple with probability $\beta = \frac{n_s}{|D|}$. Then we learn the pairwise correlations between all attributes over the sampled dataset D_s. A simple attempt is to split the privacy parameter ϵ_a

into $\binom{d}{2}$ portions, each being used for a pair of attributes. However, when d is relatively large, this simple scheme can barely obtain reliable results. In addressing this problem, we generalize the threshold query technique [16], which is derived from the sparse vector technique [10]. The key difference is that *in our problem, for different pairs of attributes A_k and A_l, $I(A_k, A_l)$ needs to be compared with different thresholds $\theta_{kl} = \sigma \min(|\Omega_k| - 1, |\Omega_l| - 1)$, while in [16] all queries are compared with the same threshold.* We call this generalized mechanism the *threshold mechanism*. The general intuition is that if what one wants to learn from a sequence of queries is a sequence of *yes-or-no* answers with respect to some threshold values (i.e., whether a query answer can pass the threshold), instead of the noisy values of the queries, there is no need to split the privacy parameter in proportion to the number of queries. Intuitively, this is possible if the single tuple difference changes all yes-or-no answers in the same way with respect to the thresholds. The threshold mechanism is instantiated in Lines 5-10. We will give its formal privacy guarantee later.

In Line 5, we draw a random Laplace variable η with scale $\frac{2\Delta I}{\epsilon_a}$ to add randomness to all thresholds, where ΔI is the *maximum* sensitivity of all $I(A_k, A_l)$. That is, if there is *at least* an attribute A_i in D with $|\Omega_i| > 2$, we should use $\Delta I = \frac{2}{|D_s|} \log \frac{|D_s|+1}{2} + \frac{|D_s|-1}{|D_s|} \log \frac{|D_s|+1}{|D_s|-1}$, where $|D_s|$ is the actual size of D_s[3]. This is a must to establish the privacy guarantee of the threshold mechanism. In Lines 6-10, for each pair of attributes A_k and A_l, we calculate the noisy version of $I(A_k, A_l)$, denoted by $\widetilde{I}(A_k, A_l)$, by adding $\mathtt{Lap}(\frac{2\Delta I}{\epsilon_a})$. Here the noise scale does not depend on the number of queries, which is the key benefit of employing the threshold mechanism. In Line 9, $\widetilde{I}(A_k, A_l)$ is compared with the noisy threshold $\widetilde{\theta}_{kl} = \theta_{kl} + \eta$. If $\widetilde{I}(A_k, A_l) \geq \widetilde{\theta}_{kl}$, it indicates that A_k and A_l are correlated, and we record this correlation by adding an edge between them in the dependency graph.

Now we formally prove the privacy guarantee of Algorithm 1, including that of the threshold mechanism.

THEOREM 4. *Algorithm 1 is ϵ_1-differentially private.*

PROOF. Let $\mathcal{M}$ be the version of Algorithm 1 without sampling (i.e., without Lines 2-3) and $\mathcal{M}_s$ be Algorithm 1. We first prove that $\mathcal{M}$ is ϵ_a-differentially private and then make use of Theorem 2 to show that $\mathcal{M}_s$ is ϵ_1-differentially private. Essentially, $\mathcal{M}$ outputs a vector $\mathbf{v} = [v_1, \cdots, v_w]$, where v_i takes a binary value to indicate whether the corresponding pair of attributes is correlated, and $w = \binom{d}{2}$. Let $P(\mathbf{v} = \mathbf{a})$ and $P'(\mathbf{v} = \mathbf{a})$ be the probabilities that $\mathbf{a} \in \{0,1\}^w$ is produced by two neighboring databases D and D', respectively. By definition, we want to prove that for all D and D' and for all output vectors $\mathbf{a}$ of $\mathcal{M}$, $\frac{P(\mathbf{v}=\mathbf{a})}{P'(\mathbf{v}=\mathbf{a})} \leq \exp(\epsilon_\alpha)$. Let $v^{<i}$ denote the answers to the first $(i-1)$ elements in $\mathbf{v}$ and $a_i \in \{0,1\}$ be an answer.

$$\begin{aligned}\frac{P(\mathbf{v}=\mathbf{a})}{P'(\mathbf{v}=\mathbf{a})} &= \frac{\prod_{i=1}^{w} P(v_i = a_i | v^{<i})}{\prod_{i=1}^{w} P'(v_i = a_i | v^{<i})} \\ &= \prod_{i:a_i=1} \frac{P(v_i=1|v^{<i})}{P'(v_i=1|v^{<i})} \cdot \prod_{i:a_i=0} \frac{P(v_i=0|v^{<i})}{P'(v_i=0|v^{<i})}\end{aligned} \tag{2}$$

[3]Note that $|D_s|$ may not equal n_s due to the Bernoulli sampling process (Line 3) required by Theorem 2.

Once $v^{<i}$ is fixed, $v_i = 1$ iff $I_i + \mathtt{Lap}(\frac{2\Delta I}{\epsilon_a}) \geq \theta_i + \eta$, where I_i and θ_i are the mutual information and threshold being used for generating v_i. Let $H_i(\eta)$ be the probability that $v_i = 1$ on D when the threshold is $\theta_i + \eta$ and $H_i'(\eta)$ be that on D'. Let $\lambda = \frac{2\Delta I}{\epsilon_a}$. We have

$$H_i(\eta) \;=\; P(v_i = 1|v^{<i}) = P(\mathtt{Lap}(\lambda) + I_i - \theta_i \geq \eta \mid v^{<i})$$

Let $f(y|\mu,\lambda) = \frac{1}{2\lambda}\exp\left(-\frac{|y-\mu|}{\lambda}\right)$. We can rewrite $H_i(\eta)$ as $H_i(\eta) = \int_\eta^\infty f(y|I_i - \theta_i, \lambda)\mathrm{d}y$. After making the substitution $u = y - \Delta I$, we get

$$H_i(\eta) = \int_{\eta-\Delta I}^{\infty} f(u|I_i - \theta_i - \Delta I, \lambda)\mathrm{d}u$$

By definition of sensitivity, $H_i(\eta) \leq \int_{\eta-\Delta I}^\infty f(u|I_i' - \theta_i, \lambda)\mathrm{d}u = H_i'(\eta - \Delta I)$. Recall that η is a random Laplace variable with scale $\frac{2\Delta I}{\epsilon_a}$. It holds that $p(\eta = x) \leq \exp(\frac{\epsilon_a}{2})p(\eta = x - \Delta I)$, where $p(\cdot)$ is the probability density function. We have

$$\begin{aligned}&\prod_{i:a_i=1} P(v_i = 1|v^{<i}) = \int_{-\infty}^{\infty} p(\eta = x) \prod_{i:a_i=1} H_i(x)\mathrm{d}x \\ &\leq \exp(\frac{\epsilon_a}{2}) \int_{-\infty}^{\infty} p(\eta = x - \Delta I) \prod_{i:a_i=1} H_i(x)\mathrm{d}x \\ &\leq \exp(\frac{\epsilon_a}{2}) \int_{-\infty}^{\infty} p(\eta = x - \Delta I) \prod_{i:a_i=1} H_i'(x - \Delta I)\mathrm{d}x \\ &= \exp(\frac{\epsilon_a}{2}) \int_{-\infty}^{\infty} p(\eta = x) \prod_{i:a_i=1} H_i'(x)\mathrm{d}x \\ &= \exp(\frac{\epsilon_a}{2}) \prod_{i:a_i=1} P'(v_i = 1|v^{<i})\end{aligned} \tag{3}$$

Similarly, we can prove that

$$\prod_{i:a_i=0} P(v_i = 0|v^{<i}) \leq \exp(\frac{\epsilon_a}{2}) \prod_{i:a_i=0} P'(v_i = 0|v^{<i}) \tag{4}$$

Using Formulae 2, 3 and 4, we derive

$$\frac{P(\mathbf{v}=\mathbf{a})}{P'(\mathbf{v}=\mathbf{a})} \leq \exp(\frac{\epsilon_a}{2}) \cdot \exp(\frac{\epsilon_a}{2}) = \exp(\epsilon_a) \tag{5}$$

That is, $\mathcal{M}$ is ϵ_a-differentially private. Due to Theorem 2, $\mathcal{M}_s$ achieves ϵ_1-differential privacy because

$$\ln(1 + \beta(e^{\epsilon_a} - 1)) = \ln\left(1 + \beta(\frac{e^{\epsilon_1} - 1}{\beta} + 1 - 1)\right) = \epsilon_1.$$

This concludes the proof. □

We would like to stress that in general, to make the threshold mechanism work, all queries do *not* need to have the same sensitivity as long as we add noise to the queries and thresholds based on the *maximum* sensitivity of all queries. This explains why we have to use the larger ΔI if there is an attribute that is not binary. The randomness added to all threshold values must come from the same Laplace variable; otherwise we cannot establish the above proof.

For ease of discussion, in the following sections, we assume that the dependency graph is connected. Otherwise, we simply process each connected component separately.

4.2 Forming Attribute Clusters

With the learned dependency graph, we can feed it into the junction tree algorithm introduced in Section 3.2 to obtain the set of cliques and separators. A simple idea is to directly generate noisy marginals based on these cliques so that we can infer the joint distribution. There is no need to generate marginals for the separators as they can always be derived from the cliques. However, the junction tree algorithm selects the cliques irrespective of the differential privacy constraint. In fact, the number of marginals is directly related to the accuracy of the estimated joint distribution. Our insight is that we can properly merge the cliques into larger and fewer clusters from which we can derive the joint distribution with less noise.

Consider merging cliques $C_1, \cdots, C_k$ into a cluster CL, which contains the *set* of attributes in $C_1, \cdots, C_k$. Let the set of attributes be $\{A_1, \cdots, A_l\}$, ϵ_2 be the privacy budget for generating the noisy marginals and the total number of marginals be m. Clearly, we can generate the cliques' marginals from CL's marginal with the total noise variance $k \times 2(\frac{2m}{\epsilon_2})^2 \times |\Omega_1| \times \cdots \times |\Omega_l|$, where $2(\frac{2m}{\epsilon_2})^2$ is the noise variance added to each entry. We illustrate the benefit of merging cliques by the following example.

Example 2. *Continue with Example 1. Assume that the domain sizes of A_1, A_2, A_3, A_4, A_5, A_6 are 2, 2, 3, 4, 3 and 2, respectively. If we directly add Laplace noise to the marginals of the cliques A_1A_2, $A_2A_3A_4$, $A_3A_4A_5$ and A_4A_6, the total variance of the marginals is $\frac{8960}{\epsilon_2^2}$. Alternatively, we can merge A_1A_2 and $A_2A_3A_4$ into $A_1A_2A_3A_4$, merge $A_3A_4A_5$ and A_4A_6 into $A_3A_4A_5A_6$, add Laplace noise to $A_1A_2A_3A_4$ and $A_3A_4A_5A_6$, and derive the cliques' marginals from $A_1A_2A_3A_4$ and $A_3A_4A_5A_6$. The total variance of the cliques' marginals is $\frac{7680}{\epsilon_2^2}$. Finally, it is also possible to merge A_1A_2 and A_4A_6 into $A_1A_2A_4A_6$ and merge $A_2A_3A_4$ and $A_3A_4A_5$ into $A_2A_3A_4A_5$, and achieve a total variance $\frac{6656}{\epsilon_2^2}$.*

There are two important observations from Example 2: (1) If we properly merge the cliques and then derive their marginals from the merged clusters, we are able to reduce the magnitude of noise; (2) the merging procedure does not have to depend on the junction tree structure (e.g., merging A_1A_2 and A_4A_6, which are not adjacent in the junction tree, leads to an even smaller variance). We formally define the *OptimalMerging* problem as follows.

Definition 2. (OptimalMerging) Given a set of attributes $\mathcal{A} = \{A_1, \cdots, A_d\}$ and the set of cliques $\mathcal{C} = \{C_1, \cdots, C_k\}$ derived from $\mathcal{A}$, merge the cliques into the set of clusters $\mathcal{CL} = \{CL_1, \cdots, CL_m\}$ such that: (1) The total noise variance of the cliques' marginals $\sum_{i=1}^{m} \frac{8m^2}{\epsilon_2^2}|CL_i| \prod_{A_j \in CL_i} |\Omega_j|$ is minimum w.r.t. all possible m values, (2) each clique is in exactly one cluster, and (3) the clusters contain all cliques.

While it has been clear that finding the optimal merging scheme may substantially improve the accuracy of the estimated joint distribution, the OptimalMerging problem is, unfortunately, NP-hard to solve.

Theorem 5. *The OptimalMerging problem is NP-hard.*

Proof. We establish the NP-hardness of our problem by the reduction from 3-PARTITION [7]: *Given a multiset S of $n = 3m$ positive integers $\{I_1, I_2, \cdots, I_n\}$ such that $\forall i \in \{1, 2, \cdots, n\}$, $\frac{B}{4} < I_i < \frac{B}{2}$ and $\sum_i I_i = mB$, can S be partitioned into m disjoint subsets S_1, S_2, $\cdots$, S_m such that $\forall j \sum_{I_i \in S_j} I_i = B$ and $\cup_j S_j = S$?*

Algorithm 2 Form Attribute Clusters

```
Input: Dependency graph $G$
Input: Attribute set $\mathcal{A} = \{A_1, A_2, \cdots, A_d\}$
Output: Junction cliques $\mathcal{C}$
Output: Separators $\mathcal{S}$
Output: Attribute clusters $\mathcal{CL}$
1: $(\mathcal{C}, \mathcal{S}) \leftarrow$ JunctionTreeAlgorithm($G$);
2: for $i = |\mathcal{C}|$ to 1 do
3:     $\mathcal{CL}_i \leftarrow$ IdentifyOptimalMerging($\mathcal{C}$, $i$);
4:     Calculate the total noise variance $Var(\mathcal{CL}_i)$ of the
       cliques' marginals derived from $\mathcal{CL}_i$;
5: $\mathcal{CL} = \text{argmin}_{\mathcal{CL}_i}(Var(\mathcal{CL}_i))$;
6: return $\mathcal{C}$, $\mathcal{S}$, and $\mathcal{CL}$;
```

Since $\frac{B}{4} < I_i < \frac{B}{2}$, each S_i must contain exactly three integers from S. We now show a polynomial-time reduction from 3-PARTITION to OptimalMerging. Given an instance of 3-PARTITION with n positive integers $\{I_1, I_2, \cdots, I_n\}$ and a bound B, we construct the corresponding instance of OptimalMerging as follows. We transform the input integers to 3-PARTITION to $\{m^{10I_1}, m^{10I_2}, \cdots, m^{10I_n}\}$ as the input to OptimalMerging, where $\forall i \in \{1, 2, \cdots, n\}$, $m^{\frac{10B}{4}} < m^{10I_i} < m^{\frac{10B}{2}}$. We restrict OptimalMerging to just consider merging n non-overlapping cliques into m clusters. Given a fixed m, since $\frac{8m^2}{\epsilon_2^2}$ is a constant, OptimalMerging aims to minimize $\sum_{i=1}^{m} |S_i| \prod_{m^{10I_j} \in S_i} m^{10I_j}$, which we call the objective function. Now ask whether one can obtain a value of the objective function at most $3m^{10B+2}$. It is easy to see that if there is a 3-PARTITION, putting all integers in a subset into a cluster gives at most $m \cdot 3m \cdot m^{10B}$ for the objective function. Conversely, suppose that there is a merging scheme such that the objective function value is at most $m \cdot 3m \cdot m^{10B}$. We can map the clusters to the subsets in 3-PARTITION. Note that no subset can add to more than B, since otherwise the product in that cluster would be at least m^{10B+10}, which is already more than the objective function value $m \cdot 3m \cdot m^{10B}$. Since each I_i is in the range of $(\frac{B}{4}, \frac{B}{2})$ and $\sum_i I_i = mB$, it follows that all subsets in 3-PARTITION have size exactly 3. This establishes the proof. □

Considering the hardness of OptimalMerging, we design an approximation algorithm using an integer programming relaxation and the constrained concave-convex procedure. Algorithm 2 gives the pseudocode of our solution. For each possible number m of clusters, we identify the corresponding optimal merging scheme by the `IdentifyOptimalMerging` procedure, which will be discussed next in detail.

We first introduce the notations for modeling the OptimalMerging problem for a given number m of clusters. For simplicity of notation, we slightly abuse some notations we used before. Suppose there are a total of d attributes and the junction tree contains n cliques. The size of the ith attribute is denoted by c_i. We define the occurrence matrix $\mathbf{O} = [o_{i,j}]_{d \times n}$ where $o_{i,j}$ equals 1 when the ith attribute is contained in the jth clique and 0 otherwise. Let p_i be the product of the attributes' domain sizes in the ith clique C_i. We define $z_{i,k} \in \{0, 1\}$ as the indicator of the ith clique in the kth cluster, i.e., $z_{i,k} = 1$ implies that the ith clique C_i has been merged into the kth cluster. The objective function to find an optimal merging scheme for a given number

m of clusters is formulated as

$$\begin{aligned}
\min_{\mathbf{Z},r} \quad & \ln\left(\sum_{k=1}^{m}\frac{\prod_{i=1}^{n}p_i^{z_{i,k}}\sum_{i=1}^{n}z_{i,k}}{\prod_{i=1}^{d}c_i^{\sum_{j=1}^{n}z_{j,k}o_{i,j}-1}}\right)-\lambda r\\
\text{s.t.} \quad & z_{i,k}\in\{0,1\},\ \forall i,k\\
& \sum_{k=1}^{m}z_{i,k}=1,\ \forall i\\
& \|\mathbf{z}_i-\mathbf{z}_j\|_2^2\geq r,\ \forall i\neq j\\
& \sum_{i=1}^{n}z_{i,k}\geq 1,\ \forall k \qquad (6)
\end{aligned}$$

where $\mathbf{Z}$ is an $n\times m$ matrix with $z_{i,k}$ as its (i,k)th element, $\mathbf{z}_i$ denotes the ith column of $\mathbf{Z}$, λ is a positive constant to balance the trade-off between the two terms, and $\|\cdot\|_2$ denotes the ℓ_2 norm of a vector. Note that the first term in the objective function is to minimize the logarithm of the total noise variance derived from the m marginals (see Definition 2), the first two constraints guarantee that each clique is merged into exactly one cluster, and the last constraint ensures that each cluster contains at least one clique. Moreover, the third constraint enforces the assignments of any two clusters to be different, and we expect that the difference between the assignments captured by r is as large as possible, leading to the term '$-\lambda r$' in the objective function. Since the factor $\frac{8m^2}{\epsilon_2^2}$ is a constant for each term in the summation of the logarithmic function, it is omitted from the objective function.

We introduce m new variables $\{t_k\}_{k=1}^{m}$ which upper-bound each term in the summation of the first term in the objective function of Problem (6) via the exponential function, i.e., $\frac{\prod_{i=1}^{n}p_i^{z_{i,k}}\sum_{i=1}^{n}z_{i,k}}{\prod_{i=1}^{d}c_i^{\sum_{j=1}^{n}z_{j,k}o_{i,j}-1}}\leq\exp\{t_k\}$, which implies that

$$\ln\left(\sum_{i=1}^{n}z_{i,k}\right)+\sum_{i=1}^{n}z_{i,k}\ln p_i\leq t_k+\sum_{i=1}^{d}\left(\sum_{j=1}^{n}z_{j,k}o_{i,j}-1\right)\ln c_i.$$

Then we can reformulate Problem (6) as

$$\begin{aligned}
\min_{\mathbf{Z},\mathbf{t},r} \quad & \ln\left(\sum_{k=1}^{m}\exp\{t_k\}\right)-\lambda r\\
\text{s.t.} \quad & z_{i,k}\in\{0,1\},\ \forall i,k\\
& \sum_{k=1}^{m}z_{i,k}=1,\ \forall i\\
& r-\|\mathbf{z}_i-\mathbf{z}_j\|_2^2\leq 0,\ \forall i\neq j\\
& \sum_{i=1}^{n}z_{i,k}\geq 1,\ \forall k\\
& \sum_{i=1}^{n}z_{i,k}\ln p_i-t_k-\sum_{i=1}^{d}\left(\sum_{j=1}^{n}z_{j,k}o_{i,j}-1\right)\ln c_i\\
& +\ln\left(\sum_{i=1}^{n}z_{i,k}\right)\leq 0,\ \forall k \qquad (7)
\end{aligned}$$

where $\mathbf{t}=[t_1,\ldots,t_m]^T$. Problem (7) is an NP-hard integer programming problem. Here we relax the first constraint of Problem (7) to $z_{i,k}\geq 0,\ \forall i,k$, that is, $z_{i,k}$ is relaxed to lie in $[0,1]$, instead of belonging to the binary set $\{0,1\}$. After the relaxation, $z_{i,k}$ can be viewed as the probability of the ith clique belonging to the kth cluster. We call this relaxed Problem (7).

Relaxed Problem (7) is a non-convex problem since the third and fifth constraints are non-convex. Here we use the constrained concave-convex procedure (CCCP) [28] to solve it. The CCCP method requires that the non-convex constraints can be formulated as the difference of two convex functions. To guarantee that relaxed Problem (7) can be solved by the CCCP method, we reformulate the third and fifth non-convex constraints as

$$\underbrace{r}_{\text{Convex}}-\underbrace{\|\mathbf{z}_i-\mathbf{z}_j\|_2^2}_{\text{Convex}}\leq 0$$

$$\underbrace{\sum_{i=1}^{n}z_{i,k}\ln p_i-t_k-\sum_{i=1}^{d}\left(\sum_{j=1}^{n}z_{j,k}o_{i,j}-1\right)\ln c_i}_{\text{Convex}}$$

$$-\underbrace{\left(-\ln\left(\sum_{i=1}^{n}z_{i,k}\right)\right)}_{\text{Convex}}\leq 0$$

Through the reformulation, we can see that the requirement of the CCCP method is satisfied, making it capable of solving relaxed Problem (7). The CCCP method is an iterative method. In each iteration, the CCCP method first replaces the concave parts in the objective function and constraints with their first-order Taylor expansions based on the solution in the previous iteration and then solves the resulting convex subproblem. Specifically, for relaxed Problem (7), the lth iteration of the CCCP method solves the following problem as

$$\begin{aligned}
\min_{\mathbf{Z},\mathbf{t},r} \quad & \ln\left(\sum_{k=1}^{m}\exp\{t_k\}\right)-\lambda r\\
\text{s.t.} \quad & z_{i,k}\geq 0,\ \forall i,k\\
& \sum_{k=1}^{m}z_{i,k}=1,\ \forall i\\
& r-2(\mathbf{z}_i-\mathbf{z}_j)^T\mathbf{p}_{i,j}^{(l-1)}+q_{i,j}^{(l-1)}\leq 0,\ \forall i\neq j\\
& \sum_{i=1}^{n}z_{i,k}\geq 1,\ \forall k\\
& \sum_{i=1}^{n}z_{i,k}\ln p_i-t_k-\sum_{i=1}^{d}\left(\sum_{j=1}^{n}z_{j,k}o_{i,j}-1\right)\ln c_i\\
& +\ln\left(\sum_{i=1}^{n}z_{i,k}^{(l-1)}\right)+\frac{\sum_{i=1}^{n}(z_{i,k}-z_{i,k}^{(l-1)})}{\sum_{i=1}^{n}z_{i,k}^{(l-1)}}\leq 0,\ \forall k \qquad (8)
\end{aligned}$$

where $z_{i,k}^{(l-1)}$ is the solution in the $(l-1)$th iteration of the CCCP method, $\mathbf{p}_{i,j}^{(l-1)}=\mathbf{z}_i^{(l-1)}-\mathbf{z}_j^{(l-1)}$, and $q_{i,j}^{(l-1)}=\|\mathbf{p}_{i,j}^{(l-1)}\|_2^2$. Note that the last two terms in the third and fifth constraints of Problem (8) compose the first-order Taylor expansion of the corresponding convex parts at $\mathbf{Z}^{(l-1)}$, the solution in the previous iteration of the CCCP method. Problem (8) is obviously convex since both the objective function and constraints are convex, and we can use some solver such as the CVX [8] to solve it. Moreover, the CCCP method can guarantee to converge to a local optimum of relaxed Problem (7) by solving Problem (8) in each iteration.

Recall that the above solution gives the probabilities of a clique belonging to different clusters. We thus assign a clique to the cluster with the largest probability. We calculate the total noise variance for this merging scheme. We iterate the `IdentifyOptimalMerging` procedure for all possible numbers of clusters from 1 to $|\mathcal{C}|$, and return the scheme with the minimal total noise variance.

4.3 Generating Noisy Marginals

Given the clusters identified by Algorithm 2, we use the Laplace mechanism to generate their corresponding noisy marginal tables. Let the number of clusters be m. For each cluster's marginal table, we add Laplace noise $\texttt{Lap}(\frac{2m}{\epsilon_2})$ to each entry's count. Therefore, the noisy marginals satisfy ϵ_2-differential privacy. In our problem, *mutual consistency* among different noisy marginals is critical because for any separator $S_{ij} = C_i \cap C_j$ we need to guarantee that the noisy marginal of S_{ij} constructed from C_i is identical to that constructed from C_j so as to obtain consistent inference. We extend the post-processing technique in [21] to the general case where the noisy marginals are of different sizes and attributes may not be binary.

Consider a set of clusters $CL_1, \cdots, CL_l$. Let $A = CL_1 \cap CL_2 \cap \cdots \cap CL_l \neq \emptyset$. We use T_{CL_i} to denote CL_i's marginal, $T_{CL_i}[A]$ to denote A's marginal constructed from CL_i and $T_{CL_i}[A] \equiv T_{CL_j}[A]$ to denote that the two marginals are identical. We want to ensure $T_{CL_1}[A] \equiv \cdots \equiv T_{CL_l}[A]$. We achieve this goal in two steps. The first step is to derive the best approximation of A's marginal table. Let a be a possible value in A's domain and $T_A(a)$ be the count of a in A's marginal. Since each $T_{CL_i}(a)$ is an independent observation of $T_A(a)$, we use *inverse-variance weighting* [11] to give the approximation of $T_A(a)$ that minimizes the variance of the weighted average as follows:

$$T_A(a) = \frac{\sum_{i=1}^{l} T_{CL_i}(a)/\sigma_i^2}{\sum_i 1/\sigma_i^2},$$

where $\sigma_i^2 = \prod_{A_j \in (CL_i \setminus A)} |\Omega_j|$ is proportional to the variance of $T_{CL_i}[A](a)$. The second step is to update all T_{CL_i}'s to be consistent with T_A. The general idea is to distribute the difference between T_A and $T_{CL_i}[A]$ to all entries in T_{CL_i} with $A = a$, that is

$$T_{CL_i}(e) \leftarrow T_{CL_i}(e) + \frac{T_A(a) - T_{CL_i}(a)}{\prod_{A_j \in (CL_i \setminus A)} |\Omega_j|},$$

where e is the entries with their $A = a$.

To make all marginals consistent, we need to perform a sequence of mutual consistency steps. The order of these steps is critical, otherwise previously consistent results may be invalidated by subsequent steps. As suggested in [21], this problem can be avoided by enforcing a partial order under the subset relation on all non-empty intersections of some subset of the clusters and following a topological order over these intersections. For space reasons, we refer the interested reader to [21] for more details.

In addition to mutual consistency, we propose a simple yet effective thresholding strategy to mitigate the systematic bias due to rounding negative noisy counts to 0. We select a positive integer threshold such that the noisy counts above the threshold sum up to a number N that is closest to $|D|$. We then normalize all noisy counts above the threshold by multiplying $|D|/N$ and set all noisy counts below the threshold to 0. The rationale behind this strategy is that we want to filter out the bias introduced by the positive Laplace noise added to low true counts.

Table 1: Dataset statistics

Dataset	Type	Data Size	Attr. Number	Domain Size
AOL	Binary	619,418	45	2^{45}
Retail	Binary	88,162	50	2^{50}
BR2000	Non-binary	38,000	14	$\approx 2^{32}$
Adult	Non-binary	45,222	15	$\approx 2^{52}$
TPC-E	Non-binary	40,000	24	$\approx 2^{77}$

4.4 Producing Synthetic Datasets

With the junction tree and the noisy marginals, we can calculate the joint distribution by Equation 1. However, directly sampling a synthetic dataset from the joint distribution is computationally prohibitive. To this end, we provide an efficient way to generate a synthetic dataset by a series of local computations. We start by randomly choosing an initial clique from the junction tree and sampling its attributes from its marginal distribution, and then continuously sample other attributes in the cliques adjacent to the cliques whose attributes have been fully sampled from their *conditional distribution*. A clique is adjacent to another clique if they share a common separator. We terminate this process when all attributes have been sampled. It is easy to verify that all these probability distributions are available from the noisy marginals of the cliques and that the joint distribution of the synthetic dataset will be identical to that calculated from Equation 1.

5. EXPERIMENTAL EVALUATION

In this section, we demonstrate the performance of our solution (referred to as *JTree*) by comparing with two state-of-the-art techniques, namely *PrivBayes* [29] and *PriView* [21]. Moreover, for SVM classification, we also compare with *PrivateSVM* [22], a method specialized for SVM classification.

We make use of five standard real datasets (both binary and non-binary) in our experiments. For binary datasets, we deliberately choose the ones with larger domain sizes to test the performance of *JTree*. We use AOL, the real dataset with the largest domain size used in [21], and another benchmark frequent itemset mining dataset Retail[4]. AOL is a search log dataset that includes users' search keywords and is preprocessed to contain 45 binary attributes [21]. Retail is a retail market basket dataset, where each record consists of the distinct items purchased in a shopping visit. We preprocess Retail to include 50 binary attributes (for the reason of reproducibility, we choose the top 50 most frequent items as the binary attributes). For non-binary datasets, we use the same datasets used in [29]. BR2000 contains the demographics information collected from Brazil in 2000. Adult contains census data from the 1994 US census. TPC-E contains information of "Trade", "Security", "Security status" and "Trade type" tables in the TPC-E benchmark. We summarize the statistics of the datasets in Table 1.

5.1 Evaluation Methodology

We consider the same analysis tasks in [29], namely *k-way marginals* and *SVM classification*. Since *PriView* on-

[4]Retail is available at http://fimi.ua.ac.be/data/.

ly works for binary datasets and cannot generate synthetic datasets for SVM classification, for binary datasets we only report the results on k-way marginals. We follow the same evaluation scheme used in *PriView*: We use privacy budget $\epsilon \in \{0.1, 1.0\}$ and generate 200 random k-way marginals for each $k \in \{4, 6, 8\}$. We then plot the average L_2 *error*, which is normalized by the data size.

For non-binary datasets, when k is relatively large (e.g., 4, 6 or 8), a k-way marginal is normally very sparse, and therefore the evaluation scheme used by *PriView* may be significantly biased. As such, we choose to follow the same methodology used in *PrivBayes*. We generate *all* 2-way and 3-way marginals and report the average *total variation distance* between the original datasets and the noisy datasets. In addition, we test the classification results with SVM classifiers. Due to space limitation, we only report the results on Adult, which is the most widely used benchmark dataset for classification analysis. We train SVM classifiers on Adult to predict whether an individual (1) is a male, (2) holds a post-secondary degree, (3) has salary $> 50K$ per year, and (4) has never married. We evaluate each classification task with privacy budget $\epsilon \in \{0.2, 0.5, 0.8, 1.0\}$. Each task uses 80% of the tuples in Adult as the training set and the remaining 20% for prediction. As in *PrivBayes*, we also employ the *misclassification rate* as the performance metric.

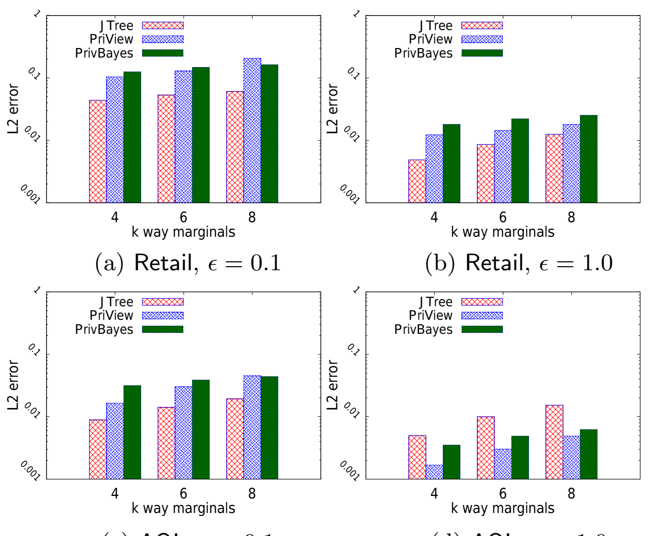
For *PriView*, we report the results based on its full version with the ripple non-negativity technique. For *PrivBayes*, as suggested in [29], we allocate half of the privacy budget on the construction of the Bayesian network and the other half on generating conditional distributions. We set $\theta = 4$ (for θ-usefulness) to select the appropriate value of the degree of a Bayesian network, as recommended in [29]. For *JTree*, when $\epsilon = 0.1$, we allocate $\epsilon_1 = 0.05$ on junction tree construction and $\epsilon_2 = 0.05$ on marginal generation. For all other ϵ values, we allocate $\epsilon_1 = 0.1$ on junction tree construction and the rest on marginal generation. All experimental results we report below are the average of ten runs.

5.2 Results on Binary Datasets

In the first set of experiments, we compare the performance of the three solutions on the binary datasets under different privacy budgets, as presented in Figure 2. It can be seen that the accuracy of *JTree* is substantially better than that of *PrivBayes* in most cases. Note that the Y-axis is in log-scale. Compared with *PriView*, *JTree* also achieves comparable accuracy. The superiority of *JTree* is more observable when ϵ is small (e.g., $\epsilon = 0.1$). For the only case where *JTree* is less accurate than *PriView*, the L_2 error of *JTree* is already very small. Furthermore, we deem that this represents an acceptable trade-off as *JTree* is a generic framework that publishes synthetic datasets for different analysis tasks, while *PriView* is the state-of-the-art technique specifically tailored for k-way marginals. In addition, *JTree* can work seamlessly on non-binary datasets, which is important for many real-world applications.

5.3 Results on Non-Binary Datasets

k-Way Marginals. In the second set of experiments, we study the average total variation distance of *PrivBayes* and *JTree* for varying ϵ values on non-binary datasets and present the results in Figure 3. Recall that *PriView* cannot process non-binary datasets, and therefore it is not reported here. As can be observed, *JTree* substantially outperforms *PrivBayes* in almost all cases. In some cases, the total variation distance of *JTree* is just half of that of *PrivBayes*. The only case that *PrivBayes* performs better than *JTree* is $\epsilon = 0.2$ on TPC-E, but even in this case, the accuracy of *JTree* is still close to that of *PrivBayes*.



(a) Retail, $\epsilon = 0.1$ (b) Retail, $\epsilon = 1.0$

(c) AOL, $\epsilon = 0.1$ (d) AOL, $\epsilon = 1.0$

Figure 2: L_2 error of k-way marginals on binary datasets

SVM Classification. In the last set of experiments, we compare *JTree*, *PrivBayes* and *PrivateSVM* for SVM classification. Due to space limitation, we only report the results on Adult, the benchmark dataset for classification. Figure 4 shows the misclassification rate of each method under different ϵ values. The misclassification rate of the original dataset (denoted as *Non Private*) stands for the best performance we can achieve. We can observe that *JTree* consistently outperforms *PrivBayes* on Adult. Compared with *PrivateSVM* that is specialized for SVM classification, *JTree* also achieves comparable performance. We interpret the slight performance degradation as the reasonable price for providing a generic data publishing solution.

6. CONCLUSION

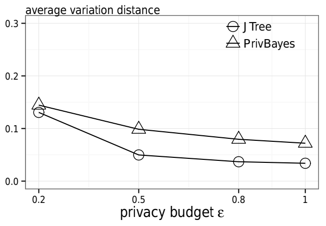
Publishing high-dimensional data with differential privacy guarantees is one of the most fundamental and challenging problems. In this paper, we have proposed a novel sampling-based inference framework to preserve the joint distribution of high-dimensional data under differential privacy. This framework features a sophisticated systematic exploration of pairwise attribute dependencies and establishes the solid inference foundation based on the junction tree algorithm, along with a series of new techniques. Extensive experiments on real benchmark datasets demonstrate that our solution substantially outperforms the state-of-the-art competitors.

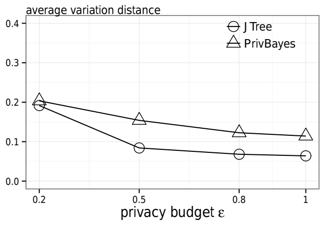
7. ACKNOWLEDGMENTS

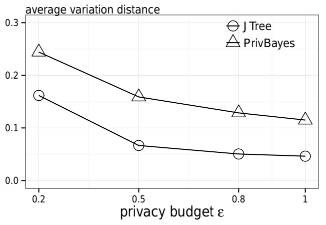
We would like to thank all reviewers for their valuable suggestions. This work was supported by RGC/GRF Grants HKBU 211512 and 12200114 and NSFC 61305071.

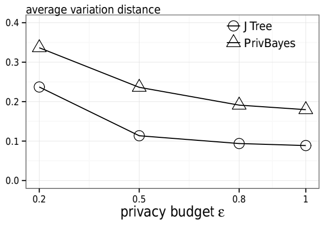
8. REFERENCES

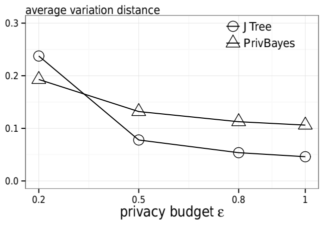
[1] G. Acs, C. Castelluccia, and R. Chen. Differentially private histogram publishing through lossy compression. In *ICDM*, 2012.


(a) BR2000, 2-way


(b) BR2000, 3-way


(c) Adult, 2-way


(d) Adult, 3-way


(e) TPC-E, 2-way

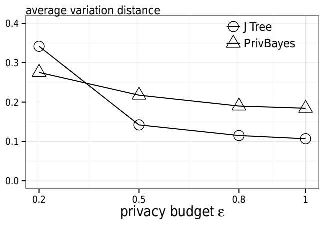
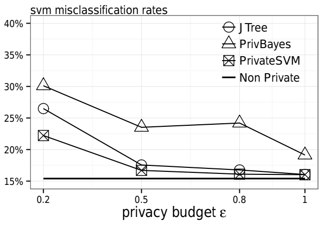
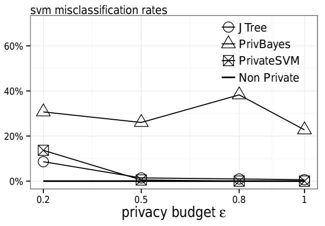
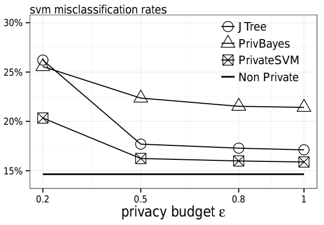

(f) TPC-E, 3-way

Figure 3: Total variation distance of k-way marginals on non-binary datasets


(a) Adult, Y=gender


(b) Adult, Y=education


(c) Adult, Y=salary

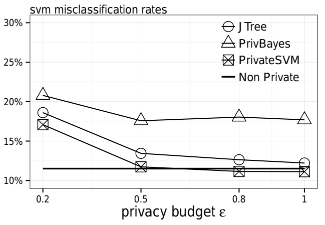

(d) Adult, Y=marital

Figure 4: SVM misclassification rates on non-binary datasets

[2] B. Barak, C. Dwork, S. Kale, F. McSherry, and K. Talwar. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *PODS*, 2007.

[3] Y. M. Bishop, S. E. Fienberg, and P. W. Paul. *Discrete multivariate analysis.* MIT Press, 1975.

[4] G. Cormode, C. M. Procopiuc, D. Srivastava, and T. T. L. Tran. Differential private summaries for sparse data. In *ICDT*, 2012.

[5] B. Ding, M. Winslett, J. Han, and Z. Li. Differentially private data cubes: optimizing noise sources and consistency. In *SIGMOD*, 2011.

[6] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, 2006.

[7] M. R. Garey and D. S. Johnson. *Computers and Intractability: a guide to the theory of NP-Completeness.* W. H. Freeman, 1979.

[8] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`, 2014.

[9] M. Hardt, K. Ligett, and F. McSherry. A simple and practical algorithm for differentially private data release. In *NIPS*, 2012.

[10] M. Hardt and G. N. Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *FOCS*, 2010.

[11] J. Hartung, G. Knapp, and B. K. Sinha. *Statistical meta-analysis with applications.* John Wiley & Sons, 2008.

[12] M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially private histograms through consistency. *PVLDB*, 3(1):1021–1032, 2010.

[13] I. F. Ilyas, W. Markl, P. Haas, P. Brown, and A. Aboulnaga. CORDS: Automatic discovery of correlations and soft functional dependencies. In *SIGMOD*, 2004.

[14] G. D. Kleiter. The posterior probability of Bayes nets with strong dependences. *Soft Computing*, 3:162–173, 1999.

[15] D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques.* MIT Press, 2009.

[16] J. Lee and C. Clifton. Top-k frequent itemsets via differentially private FP-trees. In *SIGKDD*, 2014.

[17] C. Li, M. Hay, G. Miklau, and A. McGregor. Optimizing linear counting queries under differential privacy. In *PODS*, 2010.

[18] C. Li, M. Hay, G. Miklau, and Y. Wang. A data- and workload-aware query answering algorithm for range queries under differential privacy. *PVLDB*, 7(5):341–352, 2014.

[19] N. Li, W. H. Qardaji, and D. Su. On sampling, anonymization, and differential privacy or, k-anonymization meets differentail privacy. In *ASIACCS*, 2012.

[20] N. Mohammed, R. Chen, B. C. M. Fung, and P. S. Yu. Differentially private data release for data mining. In *SIGKDD*, 2011.

[21] W. Qardaji, W. Yang, and N. Li. Priview: practical differentially private release of marginal contingency tables. In *SIGMOD*, 2014.

[22] B. Rubinstein, P. L. Bartlett, L. Huang, and N. Taft. Learning in a large function space: privacy-preserving mechanisms for SVM learning. *Journal of Privacy and Confidentiality*, 4(1):65–100, 2012.

[23] O. Williams and F. McSherry. Probabilistic inference and differential privacy. In *NIPS*, 2010.

[24] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transform. In *ICDE*, 2010.

[25] J. Xu, Z. Zhang, X. Xiao, and G. Yu. Differentially private histogram publicaiton. In *ICDE*, 2012.

[26] G. Yaroslavtsev, G. Cormode, C. M. Procopiuc, and D. Srivastava. Accurate and efficient private release of datacubes and contingency tables. In *ICDE*, 2013.

[27] G. Yuan, Z. Zhang, M. Winslett, X. Xiao, Y. Yang, and Z. Hao. Low-rank mechanism: optimizing batch queries under differential privacy. *PVLDB*, 5(11):1352–1363, 2012.

[28] A. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 15(4):915–936, 2003.

[29] J. Zhang, G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao. Privbayes: Private data release via bayesian networks. In *SIGMOD*, 2014.

[30] X. Zhang, R. Chen, J. Xu, X. Meng, and Y. Xie. Towards accurate histogram publication under differential privacy. In *SDM*, 2014.